SBC 80/10 AND SBC 80/10A SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL

Manual Order Number: 9800230F

PREFACE

This manual provides general information, installation, programming information, principles of operation, and service information for the Intel SBC 80/10 and SBC 80/10A Single Board Computers. Unless specified otherwise, references to the SBC 80/10 are valid for both systems. The areas where differences occur are identified as "SBC 80/10 only" or "SBC 80/10A only." Additional systems information and component part details are available in the following documents:

- Intel Microcomputer Systems Data Book, Part No. 98-414
- Intel 8080 Microcomputer Systems User's Manual, Part No. 98-153
- Intel Multibus Interfacing Application Note, AP-28
- Intel 8255 Programmable Peripheral Interface Application Note, AP-15
- Intel 8251 Universal Synchronous/Asynchronous Receiver/Transmitter Application Note, AP-16

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (Continued)

CHAPTER 1

INTRODUCTION

The SBC 80/10 and SBC 80/10A are members of Intel's complete line of OEM computer systems which take full advantage of Intel's LSI technology to provide economical, self-contained computer based solutions for OEM applications. The SBC 80/10 and SBC 80/10A are complete computer systems, each on a single 6.75-by-12 inch printed circuit card. The CPU, system clock, read/write memory, non-volatile read-only-memory, I/O ports and drivers, serial communications interface, bus control logic and drivers all reside on the board.

Throughout this manual, reference to the SBC 80/10 are valid for both the SBC 80/10 and SBC 80/10A. The areas where differences occur are identified as "SBC 80/10 only" and SBC 80/10A only."

Intel's powerful 8-bit n-channel MOS 8080A CPU, fabricated on a single LSI chip, is the central processor for the SBC 80/10 and SBC 80/10A. The 8080A contains six 8-bit general purpose registers and an accumulator. The six general purpose registers may be addressed individually or in pairs providing both single and double precision operators. The 8080A has a 16-bit program counter which allows direct addressing of up to 64K of memory. An external stack, located within any portion of memory, may be used as a last in/first out stack to store and retrieve the contents of the program counter, flags, accumulator and all of the six general purpose registers. A sixteen bit stack pointer controls the addressing of this external stack. This stack provides almost unlimited subroutine nesting. Sixteen-line address and eight-line bi-directional data busses are used to facilitate easy interface to memory and I/O.

The powerful 8080A instruction set allows the user to write efficient programs in a minimum amount of time. The accumulator group instructions include arithmetic and logical operators with direct, register direct, and immediate addressing modes. Move, load, and store instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using all addressing modes. The ability to branch to different portions of a program is provided with jump, jump conditional, and computed jumps. The ability to conditionally and unconditionally call to and return from subroutines is provided. The RESTART (or single byte call instruction) is used for interrupt operation. Double precision operators such as stack manipulation and double add instructions extend both the arithmetic and interrupt handling capability of the 8080A. The ability to increment and decrement memory, the six general registers, and the accumulator is provided as well as extended increment and decrement instructions to operate on the register pairs and stack pointer.

The difference between the SBC 80/10 and SBC 80/10A is in the type and quantity of memory available on each board.

The SBC 80/10 contains 1K 8-bit words of read/write memory using Intel's 8111 Low Power Static RAMs. Sockets for up to 4K 8-bit words of non-volatile read-only memory may be added in 1K byte increments using Intel's 8708 Erasable and Electrically Reprogrammable ROMs (EPROMs) or Intel's 8308 Metal Masked ROMs.

The SBC 80/10A contains 1K 8-bit words of read/write memory using Intel's 8102 Low Power Static RAMs. Sockets for up to 4K or 8K words of non-volatile read-only memory are provided on the SBC 80/10A. Up to 4K of read-only memory may be added in 1K byte increments using Intel's 8708 Erasable and Electrically Reprogrammable ROMs (EPROMs), Intel's 2758 Erasable and Electrically Reprogrammable ROMs (EPROMs), or Intel's 8308 Metal Masked ROMs. Optionally up to 8K words of read-only memory may be added in 2K byte increments using Intel's 2716 Erasable and Electrically Reprogrammable ROMs (EPROMs) or Intel's 2316E Metal Masked ROMs.

The SBC 80/10 contains 48 programmable parallel I/O lines implemented using two Intel 8255 Programmable Peripheral Interface devices. The software is used to configure the I/O lines in combinations of uni-directional input/output, and bidirectional ports. Therefore, the I/O interface may be customized to meet specified peripheral requirements. In order to take full advantage of the large number of possible I/O configurations, sockets are provided for interchangable I/O line drivers and terminators. Hence, the flexibility of the I/O interface is further enhanced by the capability of selecting the appropriate optional line drivers and terminators for each application.

A programmable serial communications interface using Intel's 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) is contained on the board. The USART can be programmed by the systems software to provide virtually any serial data transmission technique presently in use (including IBM Bi-Sync). The mode of operation (i.e., synchronous or asynchronous), data format, control character format, parity, and asynchronous serial transmission rate (within limitations given later) are all under program control). The 8251 provides full duplex, double buffered transmission and receive capability. Parity, overrun, and framing error detection are all incorporated in the USART. The inclusion of jumper selectable teletype, or RS232C compatible interfaces on the board in conjunction with the USART provide a direct interface to a teletype, CRT, RS232C compatible devices, and asynchronous and synchronous modems.

A single-level interrupt may originate from any one of six sources including the USART, Programmable I/O interface, and two user designated interrupt request lines. When an interrupt request is recognized, a RESTART 7 instruction is generated. The processor responds by suspending program execution and executing a user defined interrupt service routine originating at location 38_{16}.

Memory and I/O expansion may be achieved using standard Intel boards. Memory may be expanded to 65,536 bytes by adding user specified combinations of SBC 016 16K byte RAM board, SBC 406 6K byte and SBC 416 16K byte PROM boards. Input/output capacity may be expanded in increments of 4 input ports and 4 output ports using SBC 508 Input/Output boards. Expandable backplanes and cardcages are available to support multi-board systems.

The development cycle of SBC 80/10 based OEM products may be significantly reduced using the Intellec Microcomputer Development System. The resident assembler, text editor, and system monitor greatly simplify the design, development, and debug of SBC 80/10 based system software. A unique In-Circuit Emulator (ICE-80) option provides the capability of executing and debugging OEM system software directly on the SBC 80/10.

Intel's high-level language, PL/M, can be used to significantly decrease the time required to develop OEM system software.

CHAPTER 2

FUNCTIONAL DESCRIPTION

For descriptive purposes, the circuitry on the SBC-80/10 can be divided into six functional blocks:

1) CPU Set
2) System Bus Interface
3) Random Access Memory (RAM)
4) Read Only Memory (ROM/PROM) Logic
5) Serial I/O Interface
6) Parallel I/O Interface

as shown in Figure 2-1.

The CPU Set consists of the 8080A Control Processor, the 8224 Clock Generator and the 8238 System Controller. The CPU Set is the heart of the SBC-80/10. It performs all system processing functions and provides a stable timing reference for all other circuitry in the system. The CPU Set generates all of the address and control signals necessary to access memory and I/O ports both on the SBC-80/10 and external to the SBC-80/10. The CPU Set is capable of fetching and executing any of the 8080's seventy-eight instructions. The CPU Set responds to interrupt requests originating both on and off the SBC-80/10, to HOLD requests from modules wishing to acquire control of the system bus, and to WAIT requests from memory or I/O devices having an access time which is slower than the 8080's cycle time.

The System Bus Interface includes an assortment of circuitry which gates interrupt requests, HOLD requests, READY (no wait inputs and the system reset input to the appropriate pins of the CPU Set. Other circuits drive the various external system control signals. The

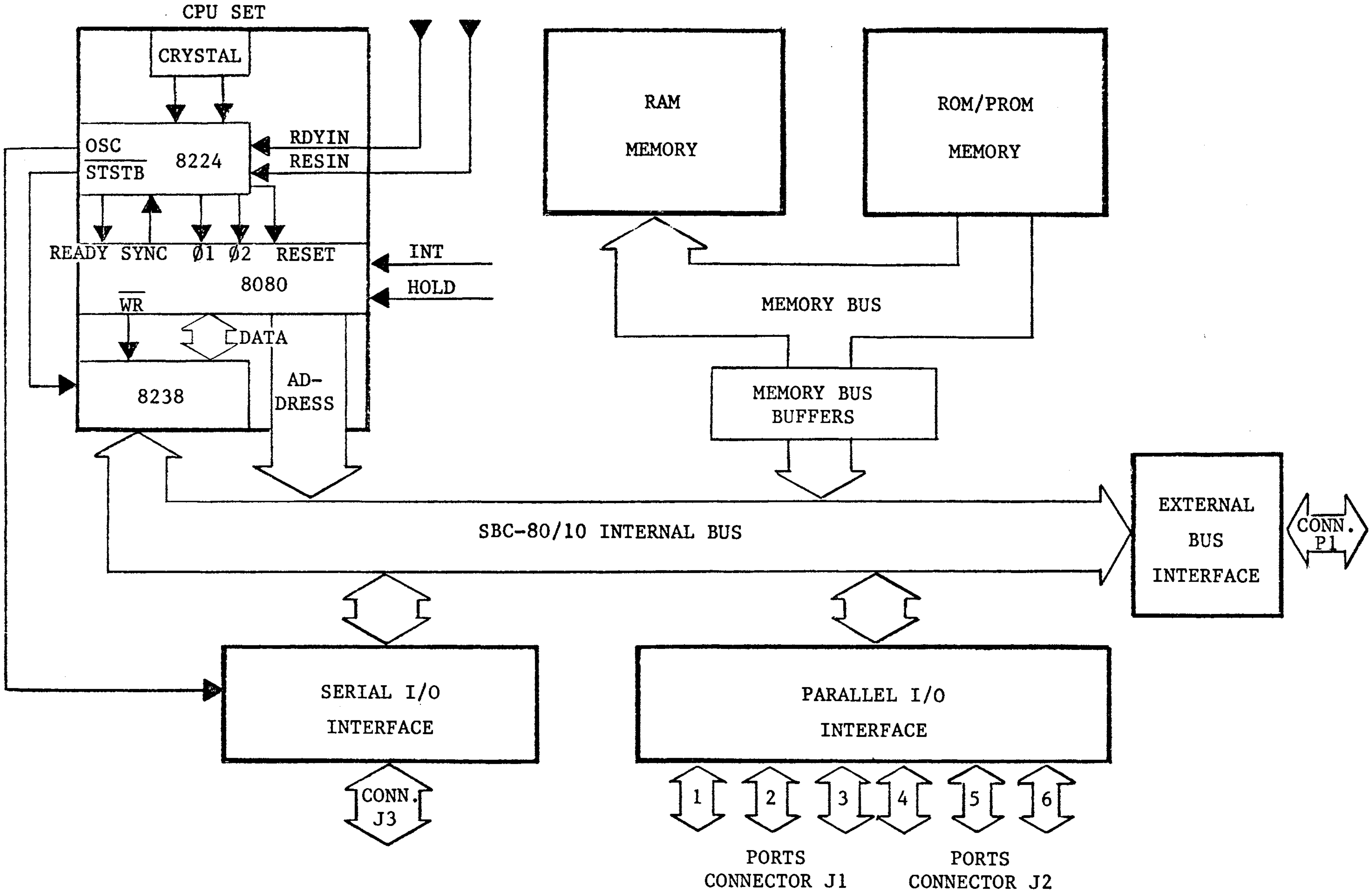


FIGURE 2-1. FUNCTIONAL BLOCK DIAGRAM

System Bus Interface also includes two 8216 bi-directional bus drivers which drive the memory data bus on the SBC-80/10. Six 8226 devices drive the external system data and address busses.

The Random Access Memory (RAM) section provides the SBC 80/10 and SBC 80/10A user with 1024 X 8-bits of on board read/write storage. Eight Intel 8111 Low Power Static RAMs (256 x 4-bit each) are mounted on the SBC 80/10. The SBC 80/10A has eight Intel 8102 Low Power Static RAM chips (1024 x 1-bit each). Both boards contain the necessary acknowledge and memory address decoding logic.

The Read Only Memory (ROM/EPROM) section provides the user with the necessary provisions for installing up to 4096 x 8-bits of ROM or EPROM on the SBC 80/10 and up to 8192 x 8-bits of ROM or EPROM on the SBC 80/10A. The 80/10 and 80/10A have four 24-pin sockets that can accept either Intel 8708 Erasable and Electrically Reprogrammable ROM (EPROM) chips, or Intel 8308 Metal Masked ROM chips. Optionally, the SBC 80/10A accepts Intel 2716 Erasable and Electrically Reprogrammable ROM (EPROM) chips, Intel 2758 Erasable and Electrically Reprogrammable ROM (EPROM) chips, or Intel 2316E Metal Masked ROM chips. The total ROM/EPROM memory capacity using 8208, 8308 or 2758 chips is 4K x 8-bits and 8K x 8-bits using 2716 or 2316 E chips. Both the 80/10 and 80/10A boards include the necessary acknowledge and memory address decoding circuitry.

The Serial I/O Interface, using Intel's 8251 USART device, provides a bi-directional serial data communications channel that can be programmed to operate with most of the current serial data transmission protocols. Synchronous or asynchronous mode, buad rate, character length, number of stop bits and the choice of even, odd or no parity are all program selectable. The user also has the option of configurating the Serial I/O Interface as an EIA RS232 interface or as a Teletype-compatible current loop interface.

The Parallel I/O Interface, using two Intel 8255 Programmable Peripheral interface devices, provides 48 signal lines for the transfer and control of data to or from peripheral devices. Eight lines already have a bidirectional driver and termination network permanently installed. This bidirectional network allows these eight lines to be inputs, outputs, or bidirectional (selected via jumpers). The remaining 40 lines, however, are uncommitted. Sockets are provided for the installation of drivers or termination networks as required to meet the specific needs of the user system.

CHAPTER 3

THEORY OF OPERATION

In the preceding chapter we introduced each of the SBC-80/10 functional blocks and defined what each block was capable of doing. In this chapter we shall go one step further and describe how each block performs its particular function(s). The text will constantly refer to the SBC-80/10 schematics, provided in Appendix A.

Note: Both active-high (positive true) and active-low (negative true) signals appear on the SBC-80/10 schematics. To eliminate any confusion when reading this chapter, the following convention will be adhered to: whenever a signal is active-low, its mnemonic is followed by a slash; for example, MRDC/ means that the level on that line will be low when the memory read command is true (active). If the signal is subsequently inverted, thus making it active-high, the slash is omitted; for example, MRDC means that the level on that line will be high when the memory read command is true.

3.1 THE CPU SET

The CPU Set consists of three Intel® integrated circuit devices:

* 8080A Central Processor Unit
* 8224 Clock Generator
* 8238 System Controller

and an 18.432 MHz crystal that establishes the frequency of oscillation for the 8224 device via a 10pF capacitor, as shown in Figure 3-0. Together, the elements in the CPU Set perform all central processing functions. The following paragraphs describe how the elements within the CPU Set interact with all other logic on the SBC-80/10. The interaction between the IC's within the CPU Set, however, is not described. Instead, the reader is referred to the Intel® "8080 Microcomputer Systems User's Manual" for a detailed description of the 8080, 8224 and 8238 devices.

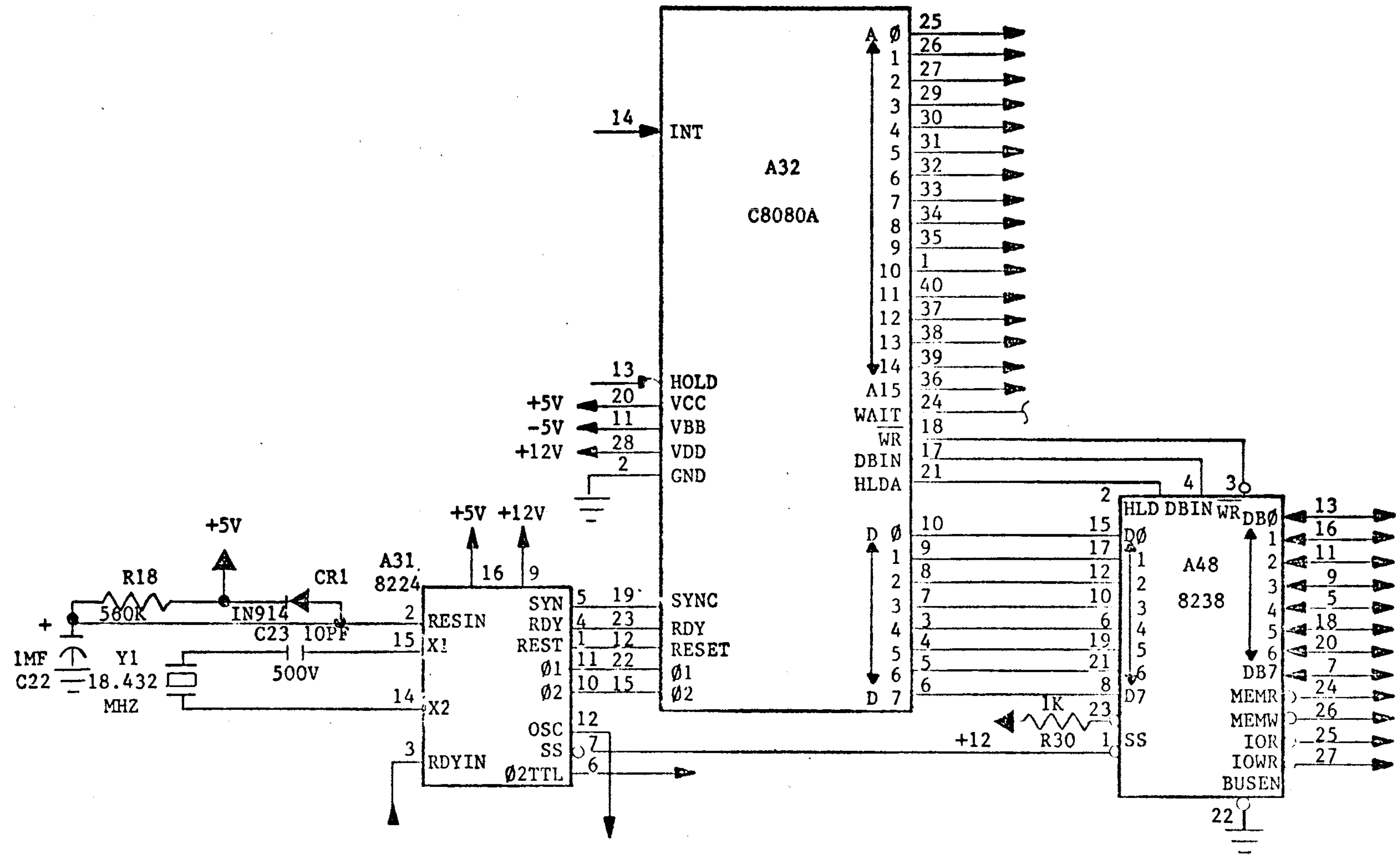


FIGURE 3-0. THE CPU SET

The CPU Set is shown on sheet 1 of the SBC-80/10 schematic (Appendix A).

3.1.1 INSTRUCTION TIMING

The activities of the CPU Set are cyclical. The CPU fetches an instruction, performs the operations required, fetches the next instruction, and so on. This orderly sequence of events requires precise timing. The 8224 Clock Generator, provides the primary timing reference for the CPU Set. The crystal in conjunction with a 10pF capacitor tunes an oscillator within the 8224 to precisely 18.432 MHz. The 8224 "divides" the oscillations by nine to produce two-phase timing inputs (Ø1 and Ø2) for the 8080. The Ø1 and Ø2 signals define a cycle of approximately 488 ns. duration. A TTL level phase 2 (Ø2TTL) signal is also derived and made available to external logic. In addition, the output of the oscillator is buffered and brought out on OSC so that other system timing signals can be derived from this stable, crystal controlled source (e.g., the serial I/O baud rate is derived from OSC). All processing activities of the CPU Set are referred to the period of the Ø1 and Ø2 clock signals.

Within the 8080 CPU Set, an <u>instruction cycle</u> is defined as the time required to fetch and execute an instruction. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's operating registers. During the execution part, the instruction is decoded and translated into specific processing activities.

Every instruction cycle consists of one, two, three, four or five machine cycles. A <u>machine cycle</u> is required each time the CPU accesses memory or an I/O port. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices.

Each machine cycle consists of three, four or five states. A state is the smallest unit or processing activity and is defined as the interval between two successive positive-going transitions of the Ø1 clock pulse.

There are three exceptions to the defined duration of a state. They are the WAIT state, the hold (HLDA) state and the halt (HLTA) state. Because the WAIT, the HLDA, and the HLTA states depend upon external events, they are by their nature of indeterminate length. Even these exceptional states, however, must be synchronized with the pulses of the driving clock. Thus the duration of all states, including these, are integral multiples of the clock pulse.

To summarize, then, each clock period marks a state; three to five states summarize a machine cycle; and one to five machine cycles comprise an instruction cycle. A full instruction cycle requires anywhere from four to seventeen states for its completion, depending on the kind of instruction involved.

There is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times that the processor must reference a memory address or an I/O address, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it transmits one address per machine cycle. Thus, if the fetching and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, then the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that, there are no fast rules. It depends on the kind of instruction. The input (INP) and the output (OUT), instructions each require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle to complete the transfer.

Every machine cycle within an instruction cycle consists of three to five active states (referred to as T1, T2, T3, T4, and T5). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. Figure 3-1 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referred to transitions of the Ø1 and Ø2 clock pulses.

At the beginning of each machine cycle (in state T1), the 8080 activates its SYNC output and issues status information on its data bus. The 8224 accepts SYNC and generates an active-low status strobe (STSTB/) as soon as the status data is stable on the data bus. The status information indicates the type of machine cycle in progress.

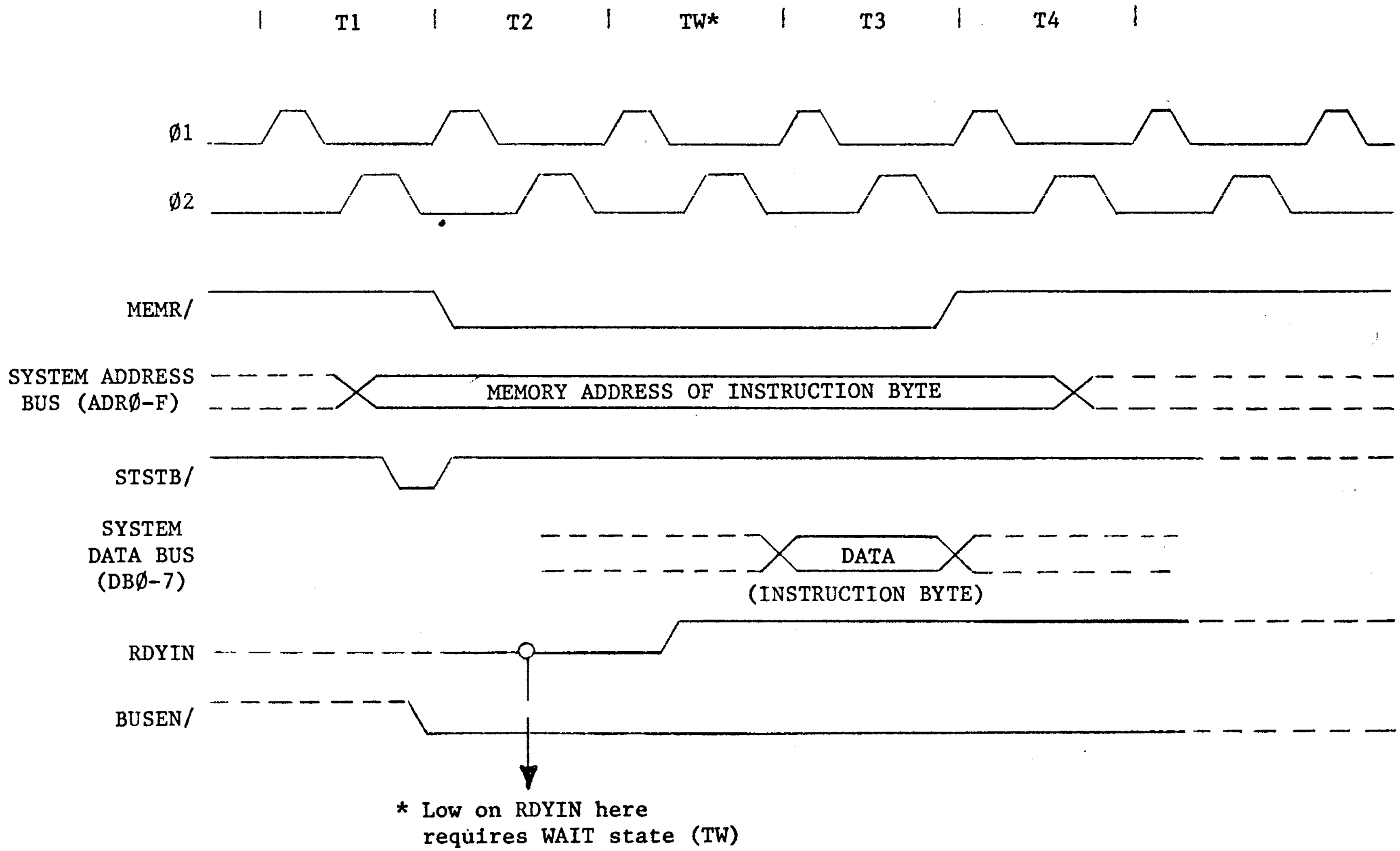


FIGURE 3-1. TYPICAL FETCH MACHINE CYCLE

The 8238 System Controller accepts the status bits from the 8080 and STSTB/ from the 8224, and uses them to generate the appropriate control signals (MEMR/, MEMW/, IOR/ and IOWR/) for the current machine cycle.

The rising edge of Ø2 during T1 loads the processor's address lines (A0 - A15). These lines become stable within a brief delay of the Ø2 clocking pulse, and they remain stable until the first Ø2 pulse after state T3. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This it does by pulling the 8224's RDYIN line low. As long as the RDYIN line remains low, the CPU Set will idle, giving the memory time to respond to the addressed data request. The 8224 synchronizes RDYIN with internal processor timing and applies the result to the 8080's READY input. The processor responds to a wait request by entering an alternative state (TW) at the end of T2, rather than proceeding directly to the T3 state. A wait period may be of indefinite duration. The 8080 remains in the waiting condition until its READY line again goes high. The cycle may then proceed, beginning with the rising edge of the next Ø1 clock. A WAIT interval will therefore consist of an integral number of TW states and will always be a multiple of the clock period.

The events that take place during the T3 state are determined by the kind of machine cycle in progress. In a FETCH machine cycle, the CPU Set interprets the data on its data bus as an instruction. During a MEMORY READ, signals on the same bus are interpreted as a data word.

The CPU Set itself outputs data on this bus during a MEMORY WRITE machine cycle. And during I/O operations, the CPU Set may either transmit or receive data, depending on whether an INPUT or an OUTPUT operation is involved. Consider the following two examples.

Figure 3-2 illustrates the timing that is characteristic of an input instruction cycle. During the first machine cycle (M1), the first byte of the two-byte IN instruction is fetched from memory. The 8080 places the 16-bit memory address on the system bus near the end of state T1. The 8238 activates the memory read control signal (MEMR/) during states T2 and T3 (and any intervening wait states, if required). During the next machine cycle (M2), the second byte of the instruction is fetched. During the third machine cycle (M3), the IN instruction is executed. The 8080 duplicates the 8-bit I/O address on address lines ADR0-7 and ADR8-F. The 8238 activates the I/O read control signal (IOR/) during states T2 and T3 of this cycle. In all cases the system bus enable input (BUSEN/) to the 8238 allows for normal operation of the data bus buffers and the read/write control signals. If BUSEN/ goes high the data bus output buffers and control signal buffers are forced into a high-impedance state.

Figure 3-3 illustrates an instruction cycle during which the CPU Set outputs data. During the first two machine cycles (M1 and M2), the CPU Set fetches the two-byte OUT instruction. During the third machine cycle (M3), the OUT instruction is executed. The 8080 duplicates the 8-bit I/O address on lines ADR0-7 and ADR8-F. The 8238 activates an advanced I/O write control signal (IOWR/) at the beginning of state T2 of this cycle. The nature and implications of the 8238 timing

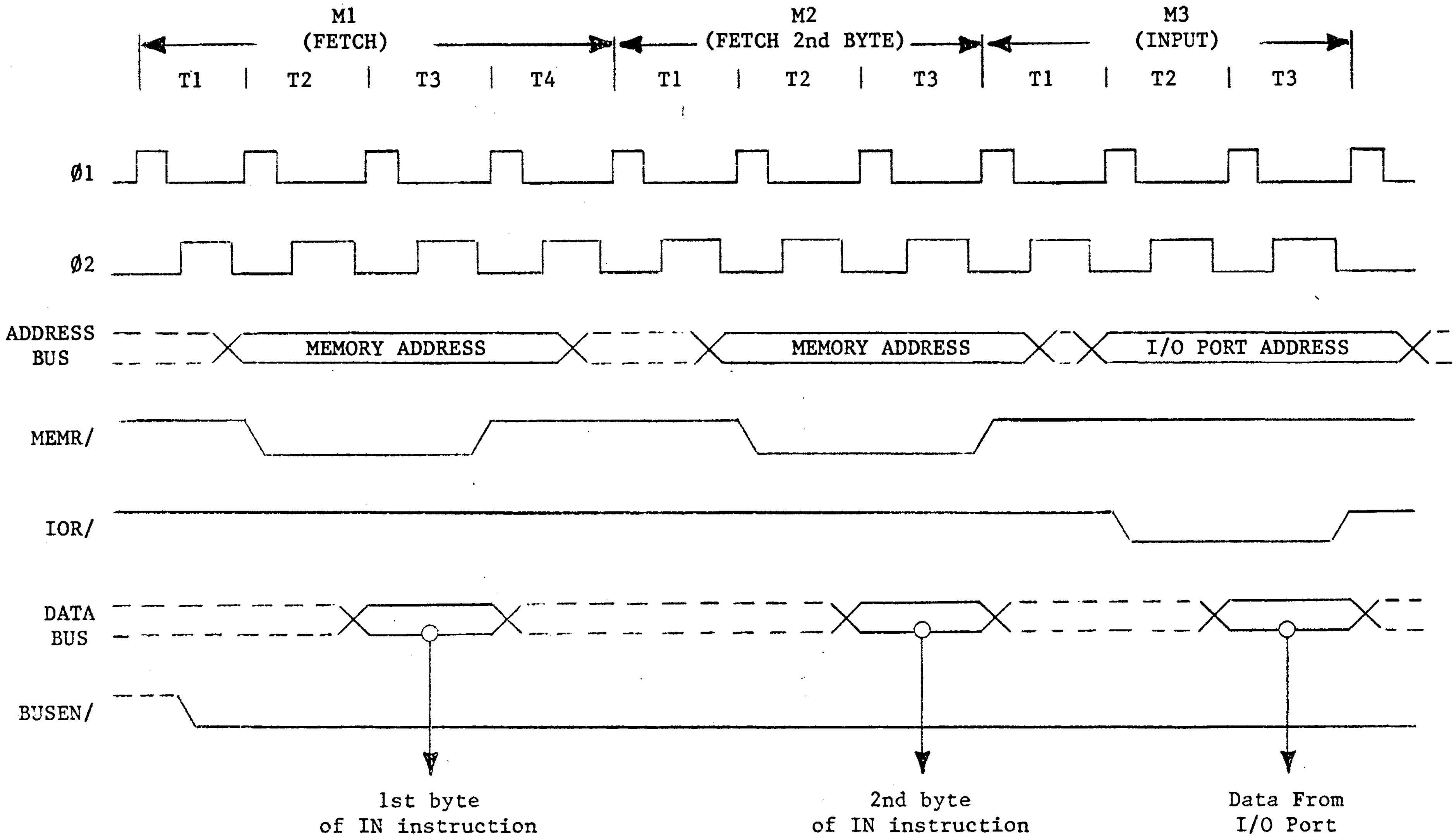


FIGURE 3-2. INPUT INSTRUCTION CYCLE

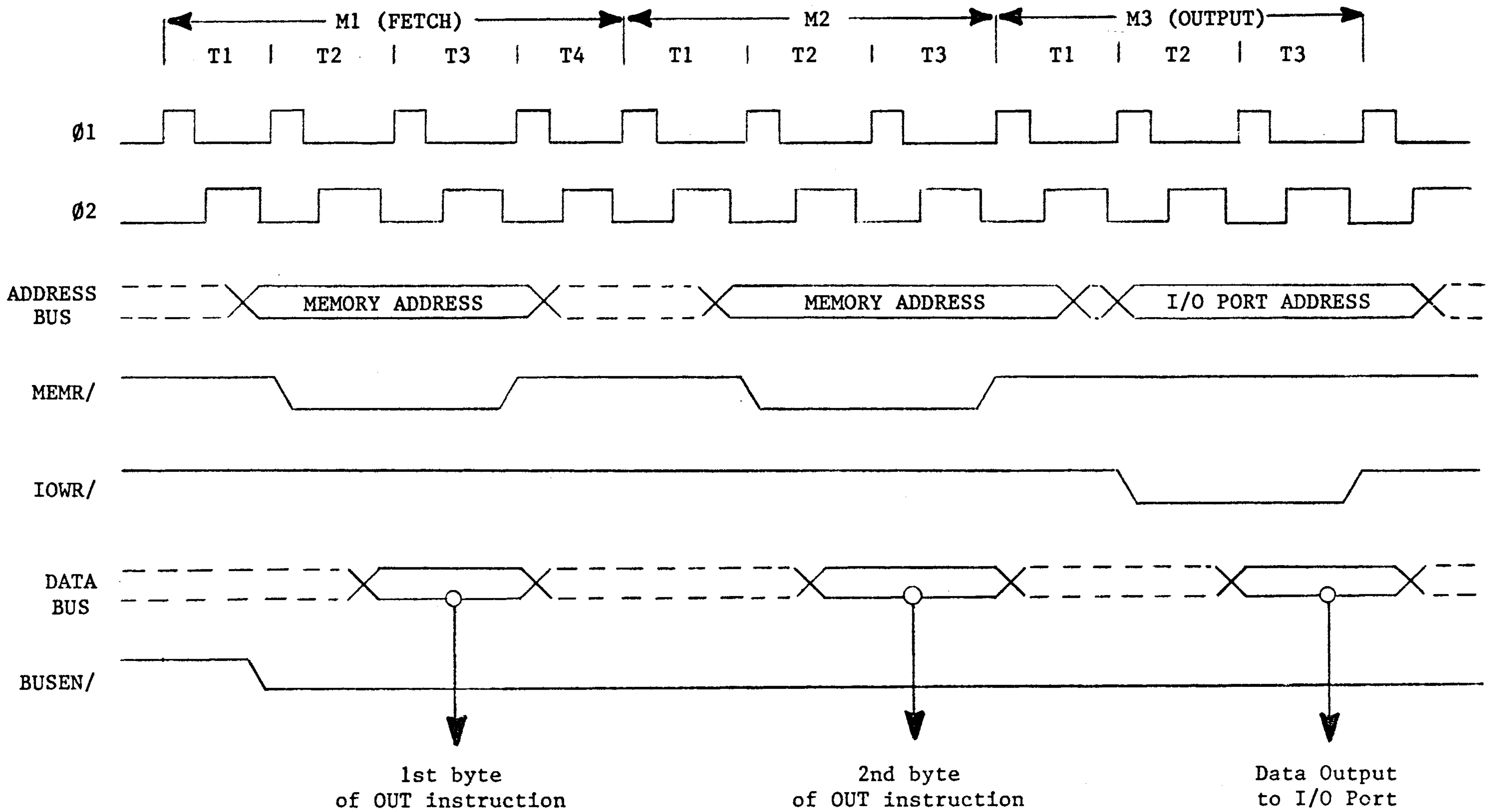


FIGURE 3-3. OUTPUT INSTRUCTION CYCLE

will be explained later (p. 3-17). The 8238 outputs the data onto the system bus at the end of state T2. Data on the bus remains stable throughout the remainder of the machine cycle. BUSEN/ must be low to prevent the output and control buffers from being forced into the high-impedance state.

Observe that a RDYIN signal is necessary for completion of an output machine cycle. Unless such an indication is present, the processor enters the TW state, following the T2 state. Data on the output lines remains stable in the interim, and the processing cycle will not proceed until the RDYIN line again goes high.

The 8080 generates a WR/ output for qualification of the advanced I/O write (IOWR/) and memory write (MEMW/) control signals from the 8238, during those machine cycles in which the CPU Set outputs data. The negative-going leading edge of WR/ is referred to the rising edge of the first Ø1 clock pulse following T2. WR/ remains low until re-triggered by the leading edge of Ø2, during the state following T3. Note that any TW states intervening between T2 and T3 of the output machine cycle will necessarily extend WR/.

All processor machine cycles consist of at least three states: T1, T2, and T3 as just described. If the CPU Set has to wait for a RDYIN response, then the machine cycle may also contain one or more TW states. During the three basic states, data is transferred to or from the CPU Set.

After the T3 state, however, it becomes difficult to generalize. T4 and T5 states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding through the T4 and T5 states every time. Thus the 8080 may exit a machine cycle following the T3, the T4, or the T5 state and proceed directly to the T1 state of the next machine cycle.

3.1.2 INTERRUPT SEQUENCES

The 8080 has the built-in capacity to handle external interrupt requests. Peripheral logic can initiate an interrupt simply by driving the processor's interrupt (INT) line high. The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. An interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the Ø2 clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The contents of the program counter are latched onto the address lines during T1, but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be. In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be saved in the stack.

This in turn permits an orderly return to the interrupted program after the interrupt request has been processed.

Because the 8238's INTA/ output (pin 23) is tied to +12 volts, the 8238 blocks incoming data and automatically inserts a Restart (RST 7) instruction onto the 8080 data bus during state T3, when the interrupt is acknowledged by the 8080. RST is a special one-byte call instruction that facilitates the processing of interrupts (the ordinary program call instruction is three bytes long). The RST 7 instruction causes the 8080 to branch program control to the instruction being stored in memory location 38_{16}.

3.1.3 HOLD SEQUENCES

By activating the 8080's HOLD input, an external device can cause the CPU Set to suspend its normal operations and relinquish control of the address and data busses. The CPU Set responds to a request of this kind by floating its address and data outputs, so that these exhibit a high impedance to other devices sharing the busses. At the same time, the processor acknowledges the HOLD by placing a high on on its HLDA output pin. During an acknowledged HOLD, the address and data busses are under control of the peripheral which originated the request, enabling it to conduct off board memory transfers without processor intervention.

3.1.4 HALT SEQUENCES

When a halt instruction (HLT) is executed, the 8080 enters the halt state after state T2 of the next machine cycle. There are only three ways in which the 8080 can exit the halt state:

- A high on the 8224 reset input (RESIN/) will always reset the 8080 to state T1; reset also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next Ø1 clock pulse.
- An interrupt (i.e., INT goes high while INTE is enabled) will cause the 8080 to exit the halt state and enter state T1 on the rising edge of the next Ø1 clock pulse. NOTE: The interrupt enable (INTE) flag must be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a reset signal.

3.1.5 START-UP SEQUENCE

When power is applied initially to the 8080, the processor begins operating immediately. The contents of its program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, the CPU Set power-up sequence begins with a reset. An external RC network is connected to the 8224's RESIN/ input. The slow transition of the power supply rise is sensed by an internal Schmitt Trigger which converts the slow transition into a clean, fast edge on the RESIN/ line when the input level reaches a predetermined value.

An active RESIN/ input to the 8224 produces a synchronized RESET signal which restores the processor's internal program counter to zero. Program execution thus begins with memory location zero, following a reset. Systems which require the processor to wait for an explicit start-up signal will store a halt instruction (HLT) after enabling interrupts in this location. A manual or an automatic INTERRUPT will be used for starting. In other systems, the processor may begin executing its stored program immediately. Note, however, that the reset has no effect on status flags, or on any of the processor's working registers (accumulator, indices, or stack pointer). The contents of these registers remain indeterminate, until initialized explicitly by the program.

In addition to generating a RESET signal, the RESIN/ input causes the 8224's status strobe (STSTB/) output to remain true (low). This allows both the 8080 and 8238 to be reset by a power-up sequence or an externally generated RESIN/ condition.

3.2 SYSTEM BUS INTERFACE LOGIC

The System Bus Interface logic consists of three general groups of circuitry:

1) assorted gates that accept the various bus control signals, the interrupt request lines, the ready indications and then applies these signals to the CPU Set,

2) the system bus drivers, and

3) the Failsafe circuitry which generates an acknowledgment during interrupt sequences and during those cycles in which an acknowledgment is not returned because a non-existent device was inadvertently addressed.

Each group is described in the following paragraphs.

3.2.1 SYSTEM CONTROL SIGNAL LOGIC

Interrupt Requests:

Four interrupt request lines are ORed together at A17-6 (ref. Appendix A) and applied to the 8080's INT input. Two of the interrupt request lines are from external sources: EXT INTR 1/ which enters the SBC-80/10 at connector J1 pin 49 and EXT INTR 2/ which enters the SBC-80/10 at P1-42. The other two interrupt requests originate on the SBC-80/10: INT 55/ is an interrupt request from ports 1 or 2 in the Parallel I/O Interface (see Section 3.6.2); and INT 51/ is an interrupt request from the 8251 USART in the Serial I/O Interface (see Section 3.5.4).

Hold Requests:

If the SBC-80/10 is operating in a system with other modules sharing a common external bus, another module can acquire control of the external bus by activating the 8080's HOLD/ input (connector pin P1-15). HOLD/ is inverted and applied to the 8080's HOLD pin. As described in Section 3.1.3, the 8080 will subsequently activate its hold acknowledge (HLDA) output. HLDA is, in turn, latched by a 74LS74 flip flop (at A29). The Q output from the D-type latch (DHLDA) disables the 8097 circuits (A47) that drive the external read/write control outputs: MRDC/, MWTC/, IORC/ and IOWC/. DHLDA also disables the external system address and data bus drivers by asserting a high at their active-low chip select (CS/) input pins. As a result of DHLDA, all of the above-mentioned drivers enter the high-impedance state. The $\overline{Q}$ output from the DHLDA output informs other modules of this condition via the BUSY/ output (connector pin P1-17). BUSY/ is driven by transistor Q5.

System Reset:

Connector pin P1-14 on the SBC-80/10 can be used to accept an externally generated SYSTEM RESET signal and to transfer a SBC-80/10 generated RESET signal to other modules in the system. If jumper pair 54-55 is connected, a RESET from the 8224 will be gated through the Q4 transistor to connector pin P1-14, thus resetting other modules in the system during power-up sequences. An externally generated SYSTEM RESET is accepted at P1-14, buffered, applied to the 8080's RESET input and made available to other logic on the SBC-80/10.

I/O Ready Generation

During each serial or parallel I/O cycle, a "ready" indication (IORDYIN/) is returned to the CPU Set. The three chip select lines for the 8251 and the two 8255 devices are ORed together (at A17-8 on sheet 3 of the schematic). The resultant output is then NANDed (at A44-11) with the I/O read (IOR) or the advanced I/O write (ADV IOW) signal to produce IORDYIN/. Recall from Section 3.1 that the 8238 System Controller (in the CPU Set) generates the I/O write control output at the beginning of all I/O write cycles. The IOW/ signal occurs earlier than the 8080's WR/ output. The 8238's IOW/ signal, alone, is labeled ADV IOW/. IOW/ is also synchronized with the 8080's WR/ output to produce the system write command IOWC/. ADV IOW/ allows the ready indication to be returned early enough to avoid an unnecessary wait state (see Figure 3-4). The IOWC/ signal causes an I/O device to actually write the data, later in the I/O cycle.

Ready Inputs:

Recall from Section 3.1.1 that the CPU Set must see a ready indication before proceeding to internal state T3 during all machine cycles. The 74S20 section at A57 on sheet 1 of the schematic OR's the following ready indications:

1) INT ACK/ or TIME OUT ACK/ from the Failsafe logic (see Section 3.2.3),

2) IORDYIN/ from the Serial and Parallel I/O Interfaces,

3) PROM RDYIN/ from the ROM/PROM logic (see Section 3.4), and

4) RAM RDYIN/ from the RAM section (see Section 3.3).

The resultant output indicates an on-board memory or I/O access and is used to disable the external data bus drivers at A53 and A54. This output from A57-8 is also ORed (at A30-3) with the externally generated AACK/ (connector pin P1-25) and XACK/ (connector pin P1-23) inputs. The output from A30-3 is then applied to the CPU Set's RDYIN input (pin 3 on the 8224). When the SBC-80/10 CPU Set accesses an external module, the AACK/ or XACK/ input informs the CPU Set that the external device is ready. AACK/ is an advanced acknowledge that allows certain OEM modules to be accessed faster.

Figure 3-4 illustrates basic timing for the ready indications.

Bus Clock Generation:

The OSC output from the CPU Set (18.432 MHz frequency) is applied to the clock input of a 74LS74 D-type flip flop (at A29-11 on sheet 1 of the schematic). The $\overline{Q}$ output from this latch is tied to its own D input. Consequently, the Q output exhibits half the frequency of the OSC input. This 9.216 MHz output is buffered and made available to external modules on the common clock (CCLK/) line (via connector pin P1-31) and the bus clock (BCLK/) line (via connector pin P1-13).

3.2.2 SYSTEM BUS DRIVERS

The SBC-80/10 internal memory data bus (DM0-DM7) is driven by

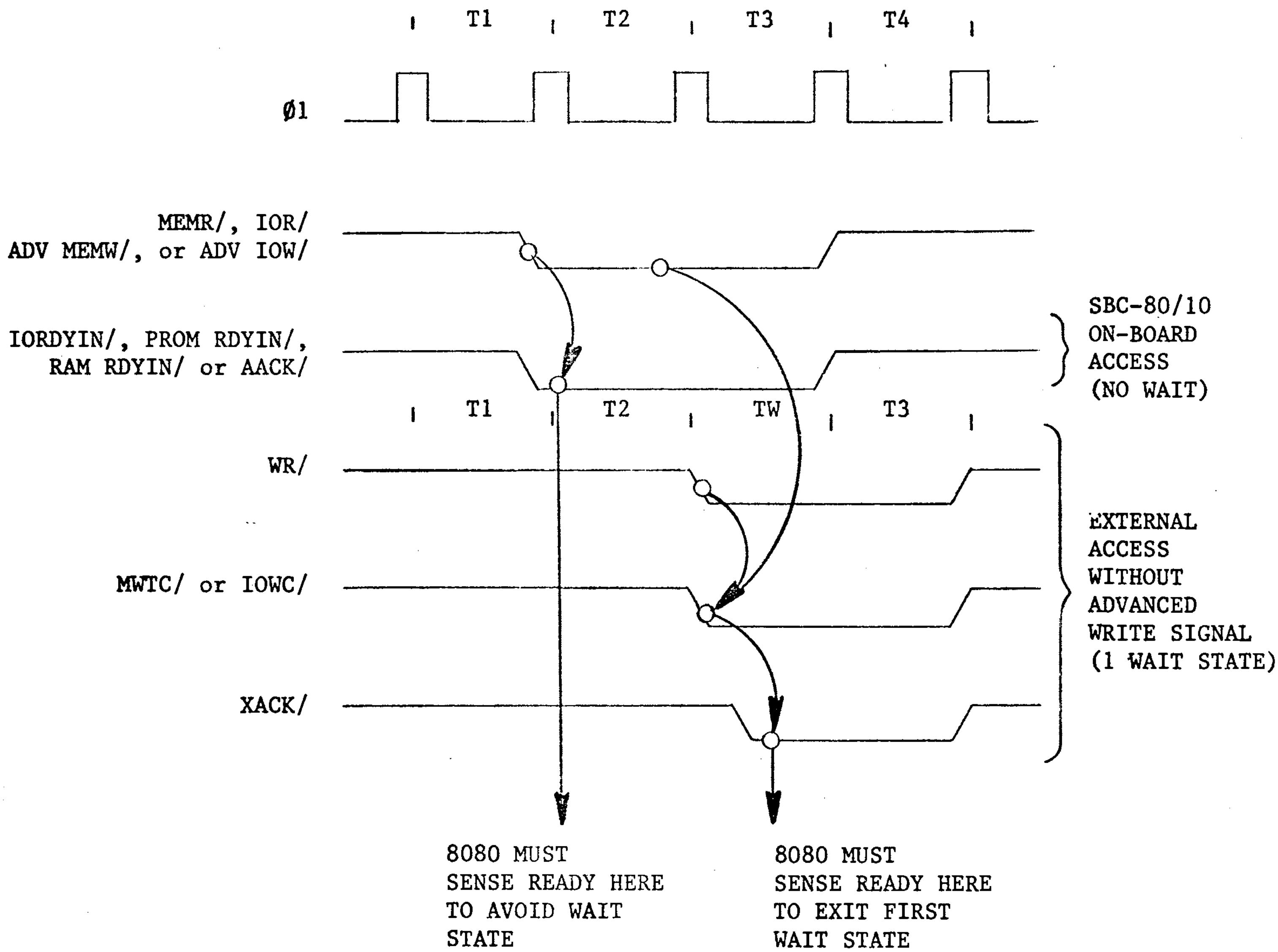


FIGURE 3-4. READY TIMING

two 8216 bidirectional bus drivers, shown at A55 and A56 on sheet 3 of the schematic. All data being transferred to/from the RAM memory (see Section 3.3) or ROM/PROM memory (see Section 3.4) is routed through these two devices. The chip select (CS/) input is provided by the MEM CMD/ signal which is the result of ORing RAM RDYIN/ and PROM RDYIN/. The direction enable (DIEN) input to the 8216's is provided by the memory read (MEMR) signal.

When the SBC-80/10 communicates with an external module, the data is driven by two 8226 bidirectional data bus drivers at A53 and A54 on sheet 1 of the schematic. The direction input to the 8226's is provided by the OR of memory read (MEMR) and I/O read (IOR). The 8226 devices will be disabled during 8080 HOLD sequences. The eight data bus lines to the 8226 bus drivers enter/leave the SBC-80/10 via the P1 edge connector.

The external 16-bit system address bus is driven by four 8226 bi-directional bus drivers. However, because the direction enable pin (EN/) on these 8226 devices is tied to ground, they can only be used to transmit addresses to external modules; they will not receive addresses from external modules. Consequently, the SBC-80/10 can access other modules, but other modules cannot access the memory or I/O controllers on the SBC-80/10. Like the data bus drivers, these 8226 devices are disabled during 8080 HOLD sequences.

3.2.3 FAILSAFE TIMER

When the 8080 acknowledges an interrupt request, the 8238 System Controller "forces" an RST 7 instruction onto the 8080's data bus (see Section 3.1.2). In order to read this RST 7 instruction, however, the 8080 must sense a ready indication. The 8080 acknowledges an interrupt by setting status bit 0 (DO) during the status output portion of each machine cycle (i.e., when STATUS STROBE is true). When this occurs, the 9602 one-shot (shown at A28 on sheet 5 of the schematic) is reset causing a low signal on its output (INTR ACK/). This output is then gated through to the RDYIN pin on the 8224 as described in Sections 3.2.1.

The Failsafe timer also performs another function. If the CPU Set tries to access a memory or I/O device but that device, for some reason, does not return a ready indication, then the 8080 remains in a wait state until ready is received. The Failsafe timer is designed to prevent hanging the system up in this way. The 9602 one-shot is triggered by STATUS STROBE at the beginning of each machine cycle. If the one-shot is not re-triggered (i.e., if another cycle does not begin) within 9 ms., then the 9602 times out and its output (also labeled TIME OUT ACK/) is gated through to the RDYIN pin on the 8224, thus allowing the 8080 to exit the wait state. This can be very helpful during system debugging.

3.3 RANDOM ACCESS MEMORY (RAM)

The Random Access Memory (RAM) provides the user with 1024 (1K) x 8-bits of read/write storage that requires no clocks or refresh to operate. The SBC 80/10 and SBC 80/10A utilize two different configurations, therefore each configuration is discussed separately in paragraphs 3.3.1 and 3.3.2.

3.3.1 SBC 80/10 RAM

The RAM logic consists of eight Intel 8111 256 x 4-bit Low Power Static RAM chips, an Intel 3205 three-to-eight decoder for chip selection and assorted gates as shown on sheet 2 of the SBC-80/10 schematic (Figure A-1).

The 8111 RAM devices used on the SBC 80/10 have a maximum access time of 500 nsec. Each chip has eight address inputs (A0-A7) that select one of the 256 four-bit segments, active-low write (W/) and chip enable (CE/) inputs and an output disable (OD) input. Each chip also has four common data input/output pins (I/O1-I/O4). A high on the OD input disables output and allows the I/O pins to be used for input. During memory read accesses, the data is read out nondestructively and has the same polarity as the input data.

The least significant system address lines (ADR0-ADR7) are applied to the eight address input pins on each 8111 RAM. The most significant eight system address lines (ADR8-ADRF) feed a 3205 decoder. Each of the four most significant decoder outputs are applied to the chip enable (CE/) inputs on two RAM chips. One RAM in each pair reads or writes data bits 0 to 3 (DM0-DM3) while the other RAM reads or writes data bits 4 to 7 (DM4-DM7) for each RAM access. One of the decoder outputs will be activated (low) whenever the value on the system address bus is within the range 3C00-3FFF (hexadecimal).

During memory write cycles, the advanced memory write signal (ADV MEMW/) is applied to the write input (W/) on each RAM. A high on the active-low memory read line (MEMR/) allows the selected RAM's I/O pins to be used to accept the data which is to be written into the addressed location. During memory read cycles, the level on ADV MEMW/ is high but is low on MEMR/ thus allowing the addressed data to be ready out and onto the data bus.

During all RAM access cycles, the active decoder output is NANDed with ADV MEMW or MEMR (at A44-3) to produce a ready indication for the CPU Set (RAM RDYIN/). The 8238 System Controller (see Section 3.1) generates ADV MEMW or MEMR early enough in the memory cycle to allow RAM RDYIN/ to appear at the CPU Set in time to prevent the occurrence of any wait states. Figure 3-5 illustrates RAM access timing.

Whenever SBC 80/10 RAM is accessed, the data is transferred to/from the RAM chips on the memory data bus (DM0-DM7). Lines DM0-DM7 are interfaced to the system data bus through two Intel 8216 bidirectional bus drivers (shown at A55 and A56 on sheet 3 of the schematic) as described in Section 3.2.

3.3.2 SBC 80/10A RAM

The SBC 80/10 RAM logic consists of eight Intel 8102 1024 x 1-bit Low Power Static RAM chips, an Intel 3205 three-to-eight decoder, and assorted gates as shown on sheet 2 of the SBC 80/10A schematic (Figure A-2).

The 8102 RAM devices used on the SBC 80/10A have a maximum access time of 450 nsec. Each RAM chip has ten address inputs (ADR0-ADR9) that select one of the 1024 bits, an active low write (ADV MEMW/) and chip enable. A high on the ADV MEMW/ input allows a memory read access.

The ten least significant address lines (ADR0-ADR9) are applied to the ten address input pins on each 8102 RAM. The six most significant address lines (ADRA-ADRF) feed a 3205 decoder. The output of the 3205 decoder is applied to each Chip Enable (CE/) input to the eight 8102 RAM's. When the value on the system address bus is within the range 3C00-3FFF the decoder output will be activated (low).

During all RAM access cycles, the active decoder output produces a ready indication for the CPU set (RAM RDYIN/). The 8238 System Controller (see Section 3.1) generates ADV MEMW/ or MEMR/ early enough in the memory cycle to allow RAM RDYIN/ to appear at the CPU set in time to prevent the occurrence of any wait states. Figures 3-5 illustrates RAM access timing.

Whenever SBC 80/10A RAM is accessed, the data is transferred to/from the RAM chips on the memory data bus (DM0-DM7). Lines DM0-DM7 are interfaced to the system data bus through two Intel 8216 bidirectional bus drivers (shown at A55 and A56 on sheet 3 of the schematic) as described in section 3.2.

3.4 READ ONLY MEMORY (ROM/EPROM)

The SBC 80/10 and 80/10A have provisions for installing 4096 (4K) x 8-bit words of read only memory in sockets already on the PC board. Four Intel 8708 1K x 8-bit Erasable and Electrially Re-programmable Read Only Memory (EPROM) chips or four 8308 1K x 8-bit Metal Masked Read Only Memory (ROM) chips can be installed in the four 24-pin sockets shown on sheet 3 of the schematics (APPENDIX A). Optionally the SBC 80/10A has provisions for installing 4096 (4K) x 8-bits of read only memory in the sockets using four Intel 2758 (1K x 8-bits) Erasable and Electrically Reprogrammable Read Only Memory (EPROM) chips or installing 8192 (8K) x 8-bit words of read only memory using either Intel 2716 2K x 8-bit Erasable and Electrically Reprogrammable Read Only Memory (EPROM) chips or Intel 2316E 2K x 8-bit Metal Masked Read Only Memory (ROM) chips.

In addition to the four 24-pin sockets, the ROM/PROM logic includes an Intel 3205 Decoder for address decoding and several assorted gates used in generating the ready indication.

When addressing up to 4K of ROM, address lines ADR0-ADR9 are applied to the address pins A0-A9 at each of the four sockets. The remaining address lines, ADRA-ADRF are decoded by the 3205 device at A42. Each of the four least significant decoder outputs are applied to the chip select (CS/) pin at one of four sockets. One chip select line will be activated whenever the value on the system address bus is between 0000 and 0FFF (hexadecimal). In addition, when the four most significant address lines are low (i.e., the address is less than 0FFF) during a memory read cycle, the output from the 74LS00 section at A39-3 is NANDed with MEMR to produce a ready indication (PROM RDYIN/) for the CPU Set. PROM RDYIN/ is thus generated in time to allow all ROM/PROM reads to occur without any wait states. PROM RDYIN/ has the same timing as RAM DRYIN/, as shown in Figure 3-5.

When using the optional 2716 or 2316E chips with the 80/10A, address lines ADR0-ADRA are applied to the address pins at each of the four sockets. The remaining address lines, ADRB-ADRF are decoded by the 3205 three-to-eight decoder. Each of the four least significant decoder outputs are applied to the Chip Select (CS/) pin at one of four sockets. One chip select line will be enabled when the value on the system address bus is between 0000 and 1FFF (hexadecimal).

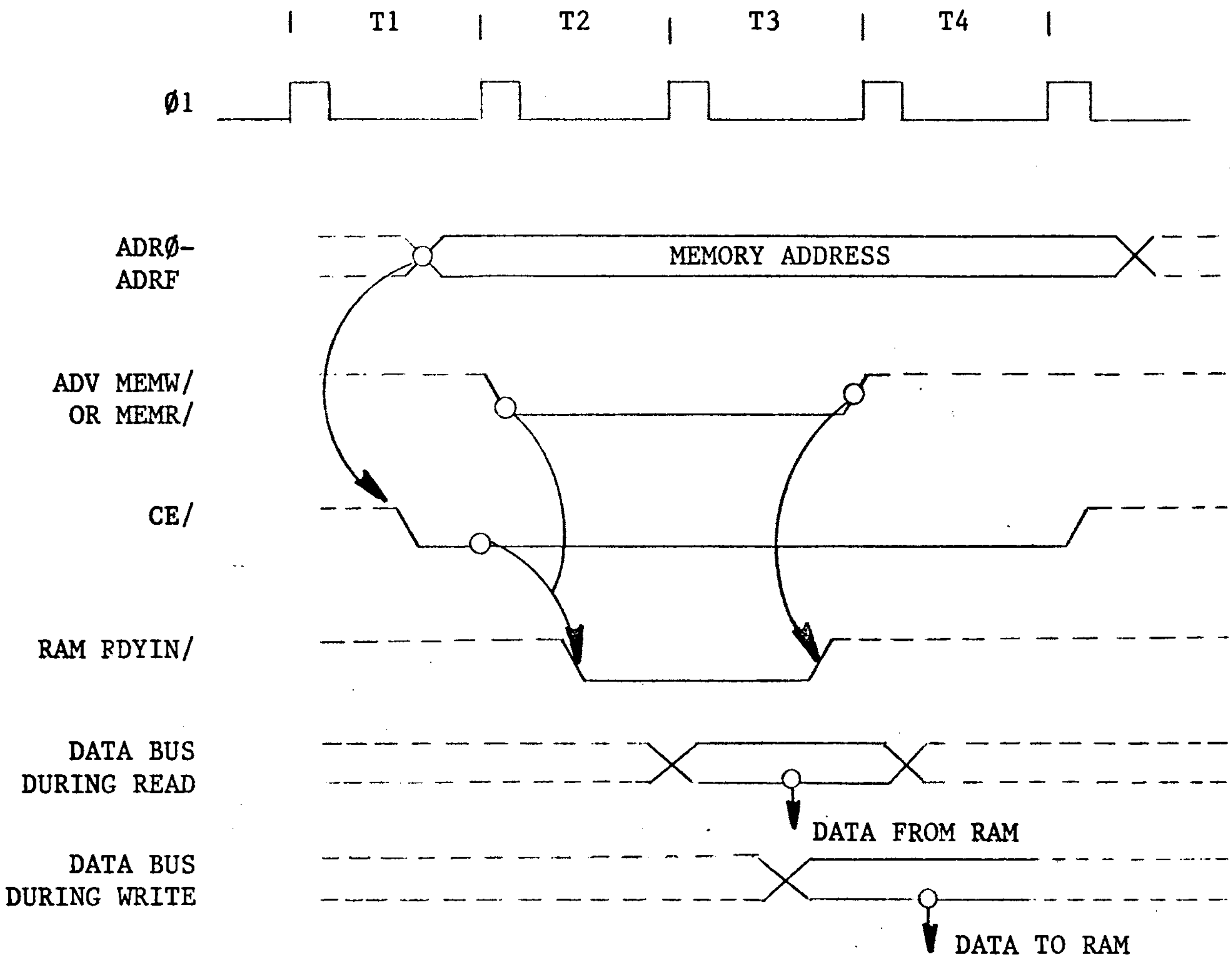


FIGURE 3-5. RAM ACCESS TIMING

In addition when the three most significant address lines are low (i.e., the address is less than 1FFF) during a memory read cycle, the output from the 74LS00 at A39-3 is NANDed with MEMR/ to produce a ready indication PROM RDYIN/ for the CPU set. PROM RDYIN/ is generated in time to allow all ROM/PROM reads to occur without any wait states. PROM RDYIN/ has the same timing as RAM RDYIN/, as shown in figure 3-5.

Whenever one of the ROM/PROM devices are read, the data from the chips output pins (01-08(is placed on the memory data bus (DM0-DM7) which is interfaced to the system bus via two Intel 8216 bidirectional bus drivers (at A55 and A56), as described in Section 3.2.

3.5 SERIAL I/O INTERFACE

The Serial I/O Interface logic provides the SBC 80/10 with a serial data communications channel that can be programmed to operate with most of the current serial data transmission protocols, synchronous or asynchronous. Baud rate, character length, number of stop bits and even/odd parity are program selectable. In addition, the serial I/O Interface can be configured (through jumper connections) as an EIA RS232C interface or as a Teletype-compatible current loop interface.

The Serial I/O Interface logic consists primarily of an Intel 8251 USART device and a counting network for baud rate selection, as shown on sheet 4 of the SBC 80/10 schematic (Appendix A). Before describing the specific operation of the Serial I/O logic however, we will summarize the general operational characteristics of the 8251

USART, because it essentially defines the character of the Serial I/O Interface.

3.5.1 INTEL 8251 OPERATIONAL SUMMARY

The 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter designed specifically for the 8080 Micro-computer System. Like other I/O devices in the 8080 Micro-computer System its functional configuration is programmed by the systems software for maximum flexiibility. The 8251 can support virtually any serial data technique currently in use (including IBM "Bi-Sync").

Modem Control

The 8251 has a set of control inputs and outputs that can be used to simplify the interface to almost any Modem. The modem control signals are general purpose in nature and can be used for functions other than Modem control, if necessary.

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

The $\overline{DSR}$ input signal is general purpose in nature. Its condition can be tested by the CPU using a Status Read operation. The $\overline{DSR}$ input is normally used to test Modem conditions such as Data Set Ready.

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

The $\overline{DTR}$ output signal is general purpose in nature. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{DTR}$ output signal is normally used for Modem control such as Data Terminal Ready or Rate Select.

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

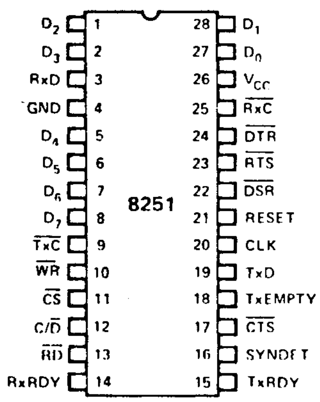
The $\overline{RTS}$ output signal is general purpose in nature. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{RTS}$ output signal is normally used for Modem control such as Request to Send.

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

A "low" on this input enables the 8251 to transmit data (serial) if the TxEN bit in the Command byte is set to a "one". This is very important to remember!

USART

PIN CONFIGURATION



Pin Name	Pin Function
D_7 D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

Pin Name	Pin Function
$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET	Sync Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

FIGURE 3-6. 8251 PIN ASSIGNMENTS

TXRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. It can be used as an interrupt to the system or for polled operation when the CPU can check TXRDY using a status read operation. TXRDY is active only when $\overline{CTS}$ is enabled. TXRDY is automatically reset when a character is loaded from the CPU.

TXE (Transmitter Empty)

When the 8251 has no characters to transmit, the TxE output will go "high". It resets automatically upon receiving a character from the CPU. TXE can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplexed operational mode.

In SYNChronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be transmitted automatically as "fillers".

$\overline{\text{TXC}}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the frequency of $\overline{\text{TXC}}$ is equal to the actual Baud Rate (1X). In Asynchronous transmission mode, the frequency of $\overline{\text{TXC}}$ is a multiple of the actual Baud Rate. A portion of the mode instruction selects the value of the multiplier; it can be 1X, 16X or 64X the Baud Rate.

For Example:

If Baud Rate equals 4800 Baud,

$\overline{\text{TXC}}$ equals 4800 Hz (1X)

$\overline{\text{TXC}}$ equals 76.8 kHz (16X)

$\overline{\text{TXC}}$ equals 307.2 kHz (64X).

The falling edge of $\overline{\text{TXC}}$ shifts the serial data out of the 8251.

RXRDY (Receiver Ready)

This output indicates that the 8251 contains a character that is ready to be input to the CPU. RXRDY can be connected to the interrupt structure of the CPU or for polled operation the CPU can check the condition of RXRDY using a status read operation. RXRDY is automatically reset when the character is read by the CPU.

$\overline{RXC}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the frequency of $\overline{RXC}$ is equal to the actual Baud Rate (1X). In Asynchronous Mode, the frequency of $\overline{RXC}$ is a multiple of the actual Baud Rate. A portion of the mode instruction selects the value of the multiplier; it can be 1X, 16X or 64X the Baud Rate.

For Example:

If Baud Rate equals 300 Baud,
$\overline{RXC}$ equals 300 Hx (1X)
$\overline{RXC}$ equals 4800 Hz (16X)
$\overline{RXC}$ equals 19.2 kHz (64X).
If Baud Rate equals 2400 Baud,
$\overline{RXC}$ equals 2400 Hz (1X)
$\overline{RXC}$ equals 38.4 kHz (16X)
$\overline{RXC}$ equals 153.6 kHz (64X).

Data is sampled into the 8251 on the rising edge of $\overline{RXC}$.

Note: In most communications systems, the 8251 will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the Same. Both $\overline{TXC}$ and $\overline{RXC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

*SYNDET (SYNC Detect)

This pin is used in SYNCHronous Mode only. It is used as either input or output, programmable through the Control Word. It is reset to "low" upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251 has located the SYNC character in the Receive mode. If the 8251 is programmed to use double Sync characters, then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input, (external SYNC detect mode), a positive going signal will cause the 8251 to start assembling data characters on the falling edge of the next $\overline{RXC}$. Once in SYNC, the "high" input signal can be removed. The duration of the high signal should be at least equal to the period of $\overline{RXC}$.

Programming the 8251

Prior to starting data transmission or reception, the 8251 must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251 and must immediately follow a Reset operation (internal or external).

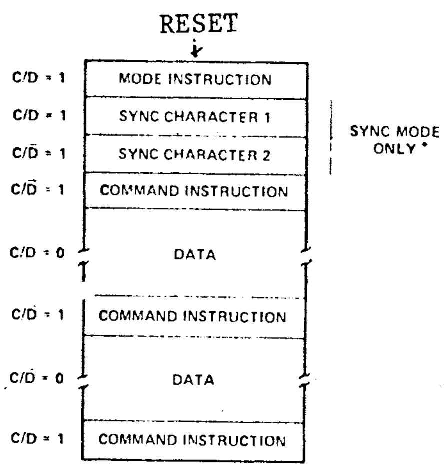
The control words are split into two formats:

1. Mode Instruction,
2. Command Instruction.

Both the Mode and Command instructions must conform to a specified sequence for proper device operation. The Mode Instruction must be inserted immediately following a Reset operation, prior to using the 8251 for data communication.

*This function is not used or made available to the user on the SBC 80/10.

All control words written into the 8251 after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251 at any time in the data block during the operation of the 8251. To return to the Mode Instruction format a bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251 back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters (see Figure 3-7).



*The second SYNC character is skipped if MODE instruction has programmed the 8251 to single character Internal SYNC Mode. Both SYNC characters are skipped if MODE instruction has programmed the 8251 to ASYNC mode.

FIGURE 3-7. TYPICAL 8251 DATA BLOCK

Mode Instruction:

This format defines the general operational characteristics of the 8251. It must follow a Reset operation (internal or external). Once the Mode instruction has been written into the 8251 by the CPU, SYNC characters or Command instructions may be inserted.

The 8251 can be used for either synchronous or asynchronous data communications. The two least significant bits of the Mode Instruction control word specify synchronous or asynchronous operation. The format for the remaining bits in the control word depends on the mode chosen by bits 0 and 1. Figure 3-8 shows the control word format for the asynchronous mode, while Figure 3-9 illustrates the control word format for the synchronous mode.

Command Instruction:

Once the functional definition of the 8251 has been programmed by the Mode Instruction and the Sync Characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode instruction has been written into the 8251 and Sync characters inserted, if necessary, then all further "control writes" ($C/\overline{D} = 1$) will load the Command Instruction. A Reset operation (internal or external) will return the 8251 to the Mode Instruction Format.

Figure 3-10 illustrate the format of a Command Instruction control word.

Status Read Definition

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251 has facilities that allow the programmer to "read" the status of the device at any time during the functional operation.

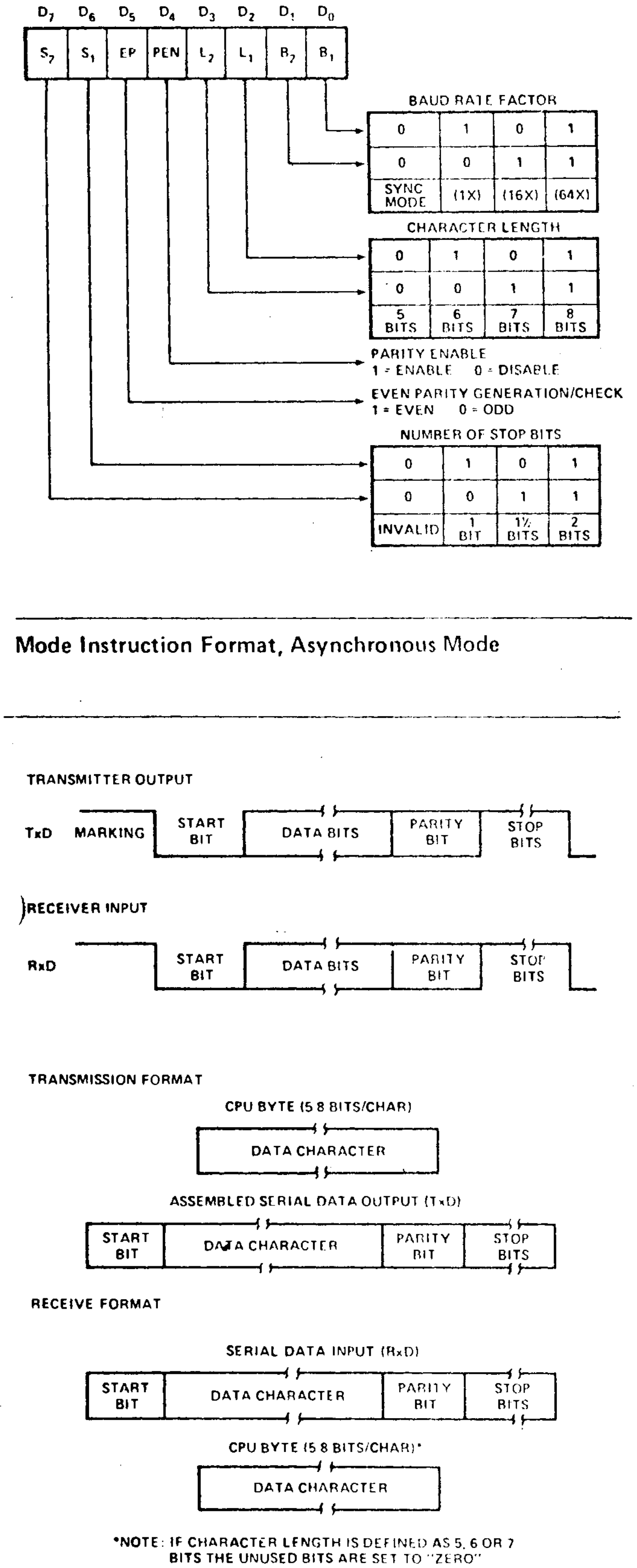


FIGURE 3-8. ASYNCHRONOUS MODE.

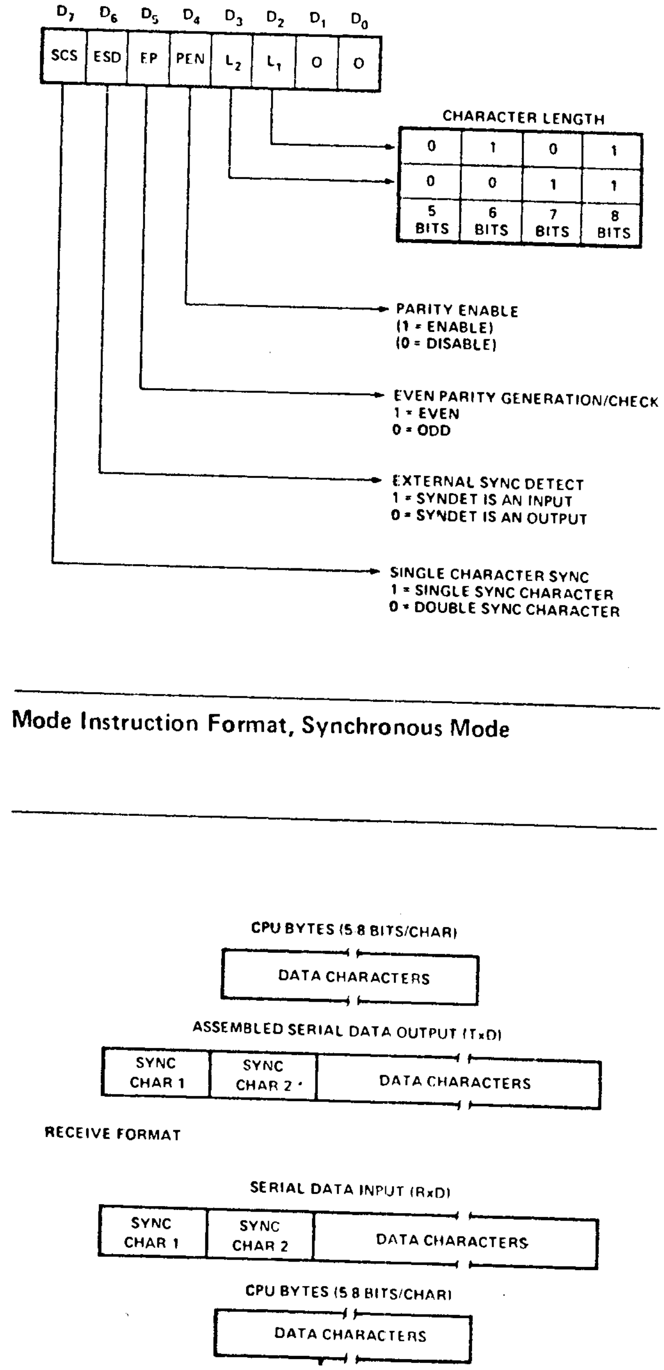


FIGURE 3-9. SYNCHRONOUS MODE.

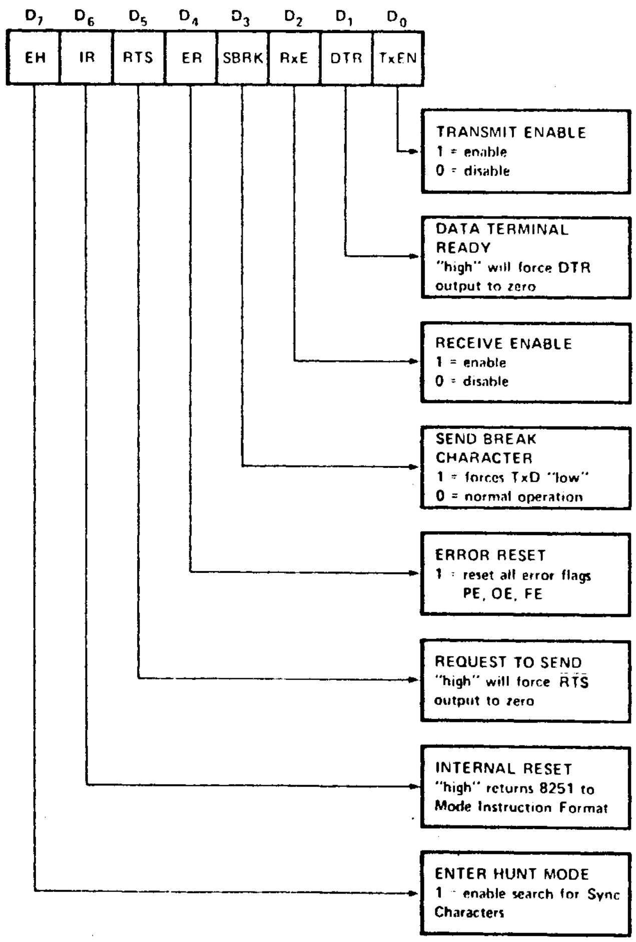


FIGURE 3-10. COMMAND INSTRUCTION FORMAT

A normal "read" command is issued by the CPU with the $C/\overline{D}$ input at one to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251 can be used in a completely Polled environment or in an interrupt driven environment (refer to Figure 3-11).

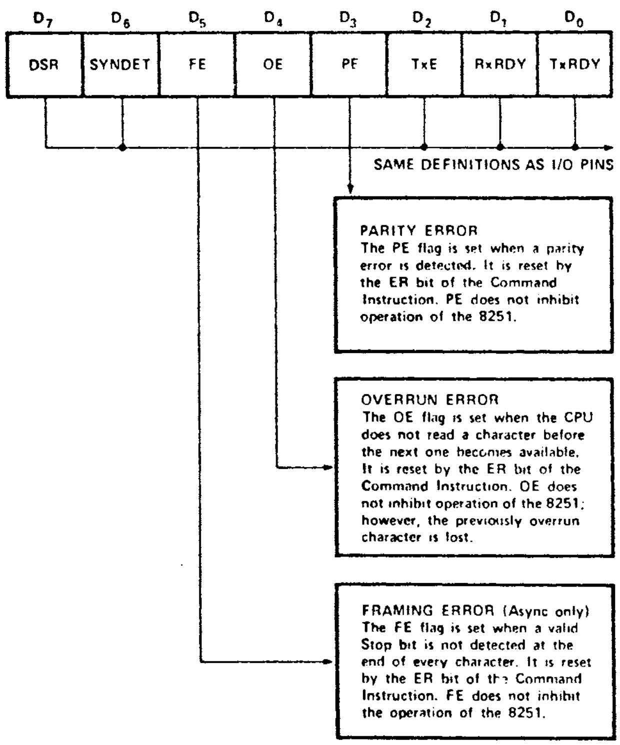


FIGURE 3-11. STATUS READ FORMAT

8251 DATA TRANSFERS

Once programmed, the 8251 is ready to perform its communication functions. The TXRDY output is raised "high" to signal the CPU that the 8251 is ready to receive a character. This output (TXRDY) is reset automatically when the CPU writes a character into the 8251. On the other hand, the 8251 receives serial data from the MODEM or

I/O device; upon receiving an entire character the RXRDY output is raised "high" to signal the CPU that the 8251 has a complete character ready for the CPU to fetch. RXRDY is reset automatically upon the CPU read operation.

The 8251 cannot begin transmission until the TXEN (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send (CTS) input. The TXD output will be held in the marking state upon Reset.

Asynchronous Mode (Transmission):

Whenever a data character is sent by the CPU the 8251 automatically adds a Start bit (low level) and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TXD output. The serial data is shifted out on the falling edge of $\overline{TXC}$ at a rate equal to 1, 1/16 or 1/64 that of the $\overline{TXC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TXD if commanded to do so.

When no data characters have been loaded into the 8251 the TXD output remains "high" (marking) unless a BREAK (continuously low) has been programmed.

Asynchronous Mode (Receive):

The RXD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center. If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RXD pin with the rising edge of $\overline{RXC}$. If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. This character is then loaded into the parallel I/O buffer of the 8251. The RXRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN flag is raised (thus the previous character is lost). All of the error flags can be reset by a command instruction. The occurrence of any of these errors will not stop the operation of the 8251.

Synchronous Mode (Transmission):

The TXD output is continuously high until the CPU sends its first character to the 8251 which usually is a SYNC character. When the $\overline{CTS}$ line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of $\overline{TXC}$. Data is shifted out at the same rate as the $\overline{TXC}$.

Once transmission has started, the data stream at TXD output must continue at the $\overline{TXC}$ rate. If the CPU does not provide the 8251 with a character before the 8251 becomes empty, the SYNC characters (or character if in single SYNC word mode) will be automatically inserted in the TXD data stream. In this case, the TXEMPTY pin will momentarily go high to signal that the 8251 is empty and SYNC characters are being sent out. The TXEMPTY pin is internally reset by the next character being written into the 8251.

Synchronous Mode (Receive):

In this mode, character synchronization can be internally or externally achieved. If the internal SYNC mode has been programmed, the receiver starts in a HUNT mode. Data on the RXD pin is then sampled in on the rising edge of RXC. The content of the RX buffer is continuously compared with the first SYNC character until a match occurs. If the 8251 has been programmed for two SYNC characters, the subsequent received character is also compared. When both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ.

Parity error and overrun error are both checked in the same way as in the Asynchronous receive mode.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost.

3.5.2 Serial I/O Configurations

The 8251 USART presents a parallel, eight-bit interface to the CPU Set via the system data bus (DB0-DB7) and presents an EIA RS232C* or TTY current loop* interface to an external device (via edge connector J3). The 8251's interface with the CPU Set is enabled by a low

*Electrical interfaces provided on SBC 80/10.

level on its chip select (CS/) pin. CS/ is low when the I/O address on the system address bus is between EC and EF (hexadecimal). Address bits 2 through 7 are decoded (at A14) to produce the CS/ input. The

TABLE 3-0. SERIAL COMMUNICATION (8251) ADDRESS ASSIGNMENTS

I/O ADDRESS (BASE 16)	COMMAND	FUNCTION
ED OR EF	OUTPUT	CONTROL WORD
EC OR EE	OUTPUT	DATA
ED OR EF	INPUT	STATUS
EC OR EE	INPUT	DATA

least significant address bit, ADRO, is applied to the 8251's C/$\overline{D}$ input (pin 12) thus indicating a control (if set) or data (if reset) byte on the data bus.

An output instruction (IOW/ is true) to port ED or EF (CS/ is low and ADRO is high) causes the 8251 to accept a control byte through its data bus pins. The control byte can be either a mode instruction or a command instruction, depending on the sequence in which it is sent. The various bits in the mode control word specify the baud rate multiplexer, character length, parity and the number of stop bits as described in Section 3.5.1. Note that the actual baud rate selected is dependent on the configuration of the baud rate jumper network (refer to Section 3.5.3). The various bits in the command control word instruct the USART to enable/disable the receiver and transmitter, to reset errors, to reset internal control and return to the mode control cycle, and to set/clear the Data Terminal Ready output.

An output instruction to port EC or EE (CS/ and ADRO are low) causes the 8251 USART to accept a data byte through its data bus pins. Bit 0 is the least significant bit and bit 7 is the most significant bit. The 8251 will subsequently transmit the data byte (if the transmitter is enabled), in serial fashion, to the external device as described in Section 3.5.1.

An input instruction (IOR/ is true) to port ED or EF (CS/ is low and ADRO is high) causes the 8251 USART to place a status byte onto the system bus. The status bits are the result of status and error checking functions performed within the USART (see Section 3.5.1).

An input instruction (IOR/ is true) to port EC or EE (CS/ and ADRO are low) causes the USART to output a data byte (previously received from the external device) from its data bus pins. Bit 0 is the least significant bit and bit 7 is the most significant bit.

Timing for the USART's internal function is provided by the ∅2TTL signal (see Section 3.1.1). The USART is reset by the occurrence of a high level on the RESET line.

The 8251 USART transmits and receives serial data, synchronously or asynchronously, as described in Section 3.5.1. By jumper-connecting the 8251 pins to different external lines, the Serial I/O logic can present either a Teletype-compatible current loop interface or an EIA RS232C interface to an external device. If the TTY-compatible current loop interface is used, the connections listed in Table 4-1 are required (see Section 4.1).

If the EIA RS232C interface is used, the connections listed in Table 4-2 are required (see Section 4.1).

3.5.3 BAUD RATE CLOCK GENERATION

The baud rate clock network consists of a 93S16 'divide-by-15' counter, two 74161 'divide-by-16' counters and wire-wrap jumpers for baud rate clock selection. The 93S16 counter is driven by the oscillator output (OSC) from the CPU Set. The QD output from this counter, in turn, drives the two 74161 counters. The outputs from these counters, each providing a different clock frequency, are tied to jumper pins that can be connected to the BAUD RATE CLK line. The available frequencies are listed in Table 4-3 (located in Section 4.2). Recall that the effective baud rate of the 8251 USART is also dependent on the state of the 8251's internal frequency divider and the mode of operation (refer to Section 3.5.1). The 8251 is capable of dividing the baud rate clock by 1, 16 or 64.

3.5.4 SERIAL I/O INTERRUPTS

The Serial I/O logic can be configured with different forms of an interrupt request mechanism. By connecting jumper pair 16-17 and disconnecting 15-16, the user can allow the 8251's Receiver Ready (RXRDY) output (pin 14) to generate an interrupt request (INT51/) to the CPU Set. RXRDY goes high whenever the receiver enable bit of the command word has been set and the 8251 contains a character that is ready to be input to the CPU Set. The user can also choose to have the 8251's Transmitter Ready (TXRDY) or the Transmitter Empty (TXE) output activate the INT51/ interrupt request. If jumper pair 19-21 is connected, a high on TXRDY (pin 15) will activate INT51/. If jumper pair 18-19 is connected instead, an active TXE (pin 18) output will generate INT51/. TXE goes high when the 8251 has no characters to transmit. TXRDY is high when the 8251 is ready to accept a character from the CPU Set. Both TXE and TXRDY are enabled by setting the transmit enable bit of the command word. Notice on the schematic that, if jumper pairs 19-20 and 15-16 are connected, Serial I/O interrupts are inhibited.

Upon receiving an interrupt, the program can determine the actual condition which is responsible for the interrupt (RXRDY, TXRDY or TXE) by reading the status of the 8251 device as described in Section 3.5.1. The interrupt request will be removed when the data is transferred to/from the 8251, as required. Note that the TXE or TXRDY output will be high, and consequently maintain an interrupt request, during all idle periods, since the 8251's transmit buffer will remain empty. To disable the transmitter, and the resultant interrupt request, the program can issue a command instruction to the 8251 with the TXEN bit (bit 0) equal to zero (refer to Section 3.5.1). The transmitter should not be disabled until TXE is high.

3.6 PARALLEL I/O INTERFACE

The Parallel I/O Interface logic on the SBC-80/10 provides forty- eight (48) signal lines for the transfer and control of data to or from peripheral devices. Eight lines have a bidirectional driver and termination network permanently installed. The remaining forty lines are uncommitted. Sockets are provided for the installation of active driver networks or passive termination networks. The optional drivers and terminators are installed in groups of four by insertion into the 14-pin sockets.

All forty-eight signal lines emanate from the I/O ports on two Intel 8255 Programmable Peripheral Interface devices, as shown on sheet 5 of the SBC-80/10 schematic (Appendix A). The two 8255 devices allow for a wide variety of I/O configurations. Before describing the possible configurations, however, we will summarize the general operational characteristics of the 8255 device.

3.6.1 INTEL 8255 OPERATIONAL SUMMARY

The 8255 contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.

<u>Port A</u>: One 8-bit data output latch/buffer and one 8-bit data input latch.

<u>Port B</u>. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

<u>Port C</u>: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into

two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with Ports A and B.

The 8080 CPU dictates the operating characteristics of the ports by outputting two different types of control words to the 8255:

(1) mode definition control word (bit 7 = 1)

2) port C bit set/reset control word (bit 7 = 0)

Bit 7 of each control word specifies its format, as shown in Figures 3-13 and 3-14, respectively.

Mode Selection

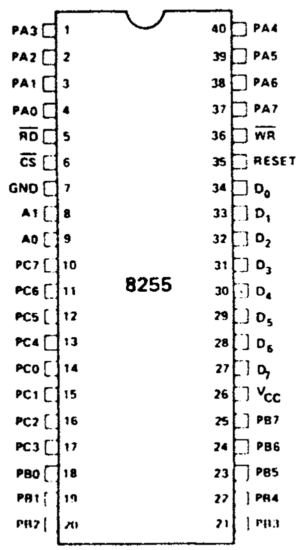
There are three basic modes of operation that can be selected by the system software:

Mode 0 - Basic Input/Output
Mode 1 - Strobed Input/Output
Mode 2 - Bi-Directional Bus

When the RESET input goes "high" all ports will be set to the Input mode 0 (i.e., all 24 lines will be in the high impedance state). After the RESET is removed the 8255 can remain in the Input mode with no additional initialization required. During the execution of the system program, the other modes may be selected using a single OUTput instruction. This allows a single 8255 to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed except for $\overline{OBF}$ in modes 1 and 2.

PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

FIGURE 3-12. 8255 PIN ASSIGNMENTS.

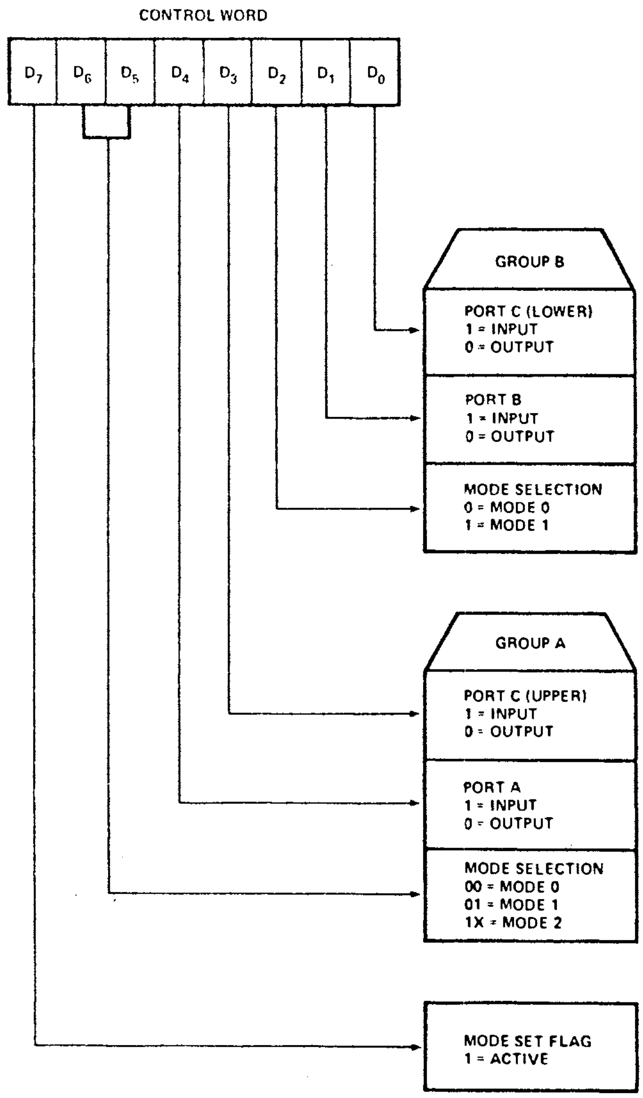


FIGURE 3-13. MODE DEFINITION CONTROL WORD FORMAT.

Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction (see Figure 3-14). This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255 is programmed to operate in Mode 1 or Mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from Port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the Bit set/reset function of Port C.

This function allows the Programmer to disallow or allow specific I/O devices to interrupt the CPU without effecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) - INTE is SET - Interrupt enable
(BIT-RESET) - INTE is RESET - Interrupt disable

Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

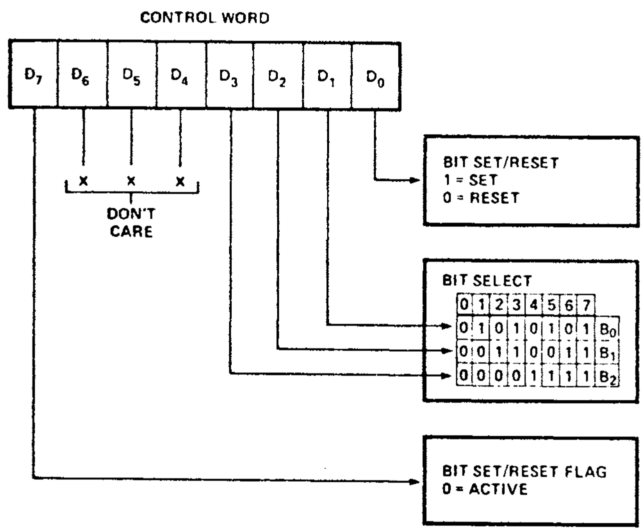


FIGURE 3-14. BIT SET/RESET CONTROL WORD FORMAT.

Operating Modes

Mode 0 (Basic Input/Output):

This functional configuration provides simple Input and Output operations for each of the three ports. No "hand-shaking" is required, data is simply written to or read from a specified port. Mode 0 timing is illustrated in Figure 3-15.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.

• 16 different Input/Output configurations are possible in this Mode. Figure 3-16 shows two possible configurations.

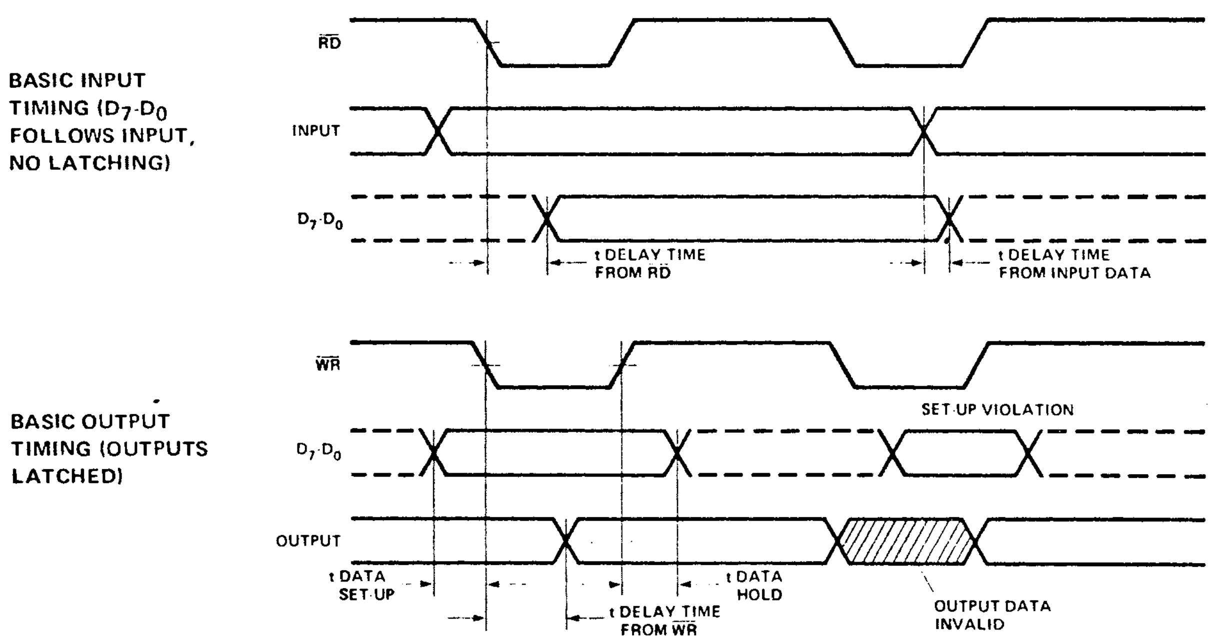


FIGURE 3-15. 8255 MODE 0 TIMING

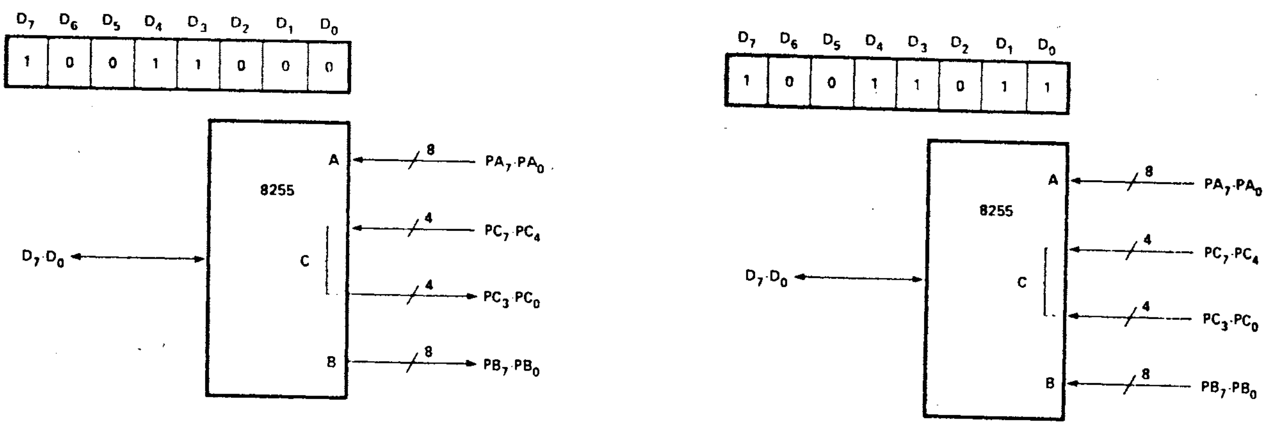


FIGURE 3-16. EXAMPLES OF MODE 0 CONFIGURATION.

Mode 1 (Strobed Input/Output):

This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In Mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Defintions:

- Two transfer ports (A and B).
- Each transfer port contains one 8-bit data port and 4 bits from one half of the control/data port (Port C).
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.

Input Control Signal Defintion for Mode 1

$\overline{\text{STB}}$ (Strobe Input)

A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by the falling edge of the STB input and is reset by the rising edge of the $\overline{\text{RD}}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the rising edge of $\overline{\text{STB}}$ if IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{RD}}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC4.

INTE B

Controlled by bit set/reset of PC2.

Figure 3-17 illustrates the Mode 1 input configuration, while Figure 3-18 shows the basic timing for Mode 1 input.

Output Control Signal Definition for Mode 1

$\overline{\text{OBF}}$ (Output Buffer Full F/F)

The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the $\overline{\text{WR}}$ input and reset by the falling edge of the $\overline{\text{ACK}}$ input signal.

$\overline{\text{ACK}}$ (Acknowledge Input)

A "low" on this input informs the 8255 that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set by the rising edge of $\overline{\text{ACK}}$ if $\overline{\text{OBF}}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{WR}}$.

INTE A

Controlled by bit set/reset of PC6.

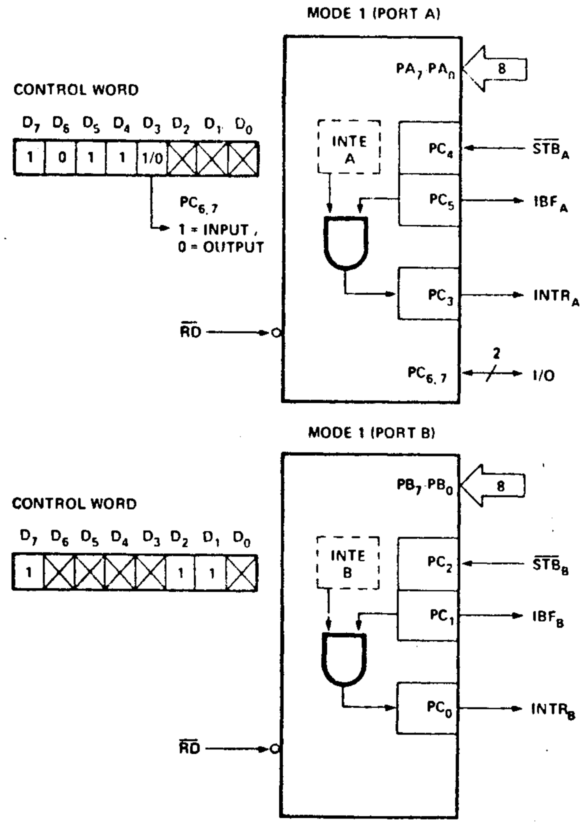


FIGURE 3-17. MODE 1 INPUT CONFIGURATION

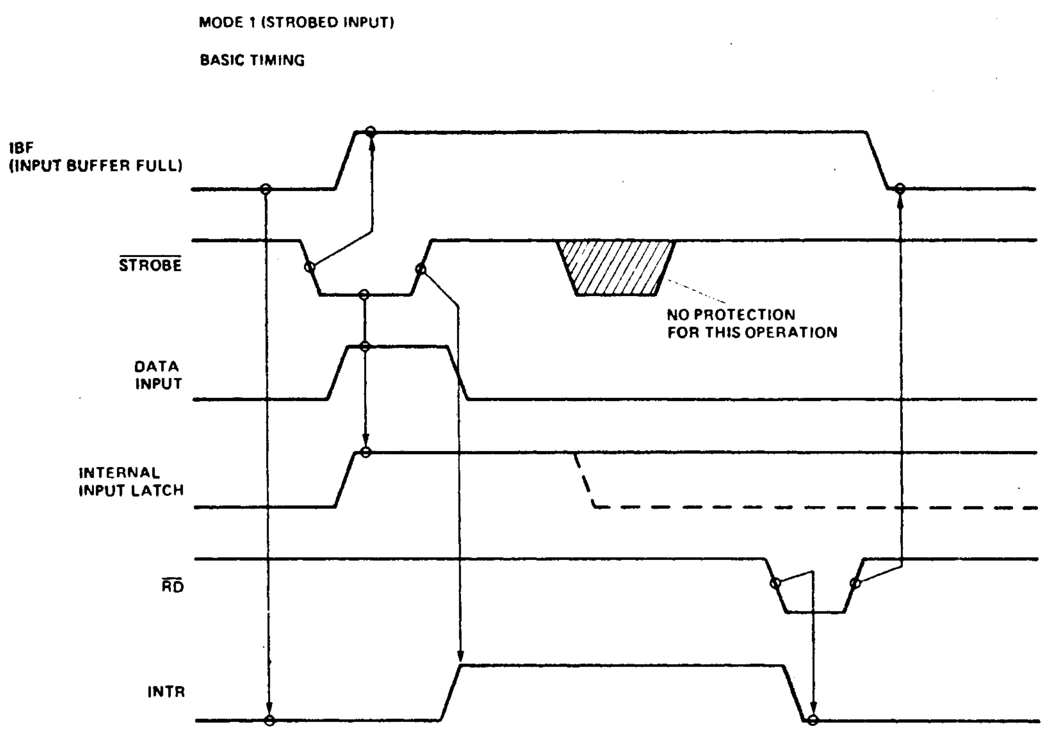


FIGURE 3-18. 8255 MODE 1 INPUT TIMING

INTE B

Controlled by bit set/reset of PC2.

Figure 3-19 illustrates the Mode 1 output configuration , while Figure 3-20 shows basic Mode 1 output timing.

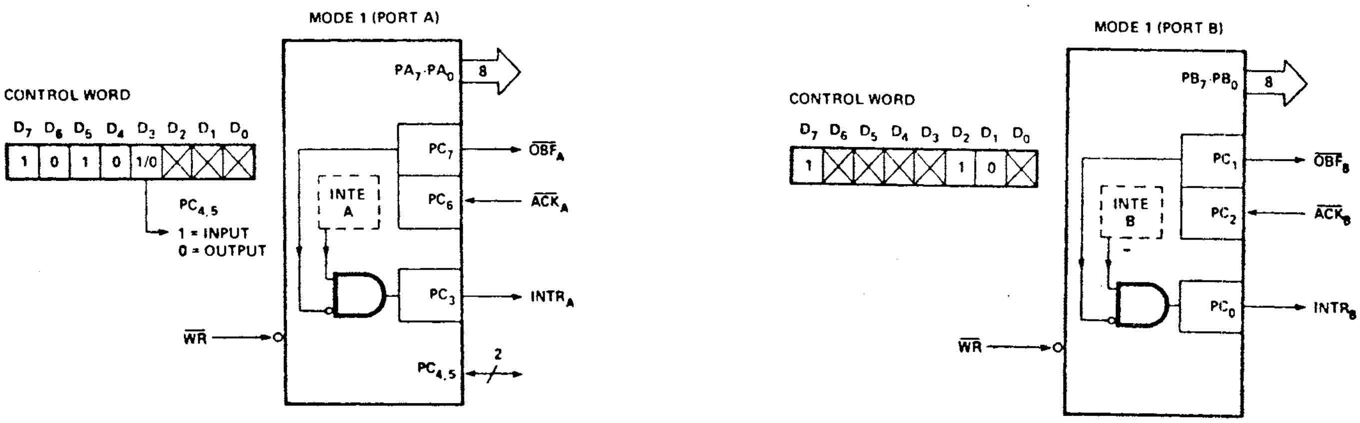


FIGURE 3-19. MODE 1 OUTPUT CONFIGURATION.

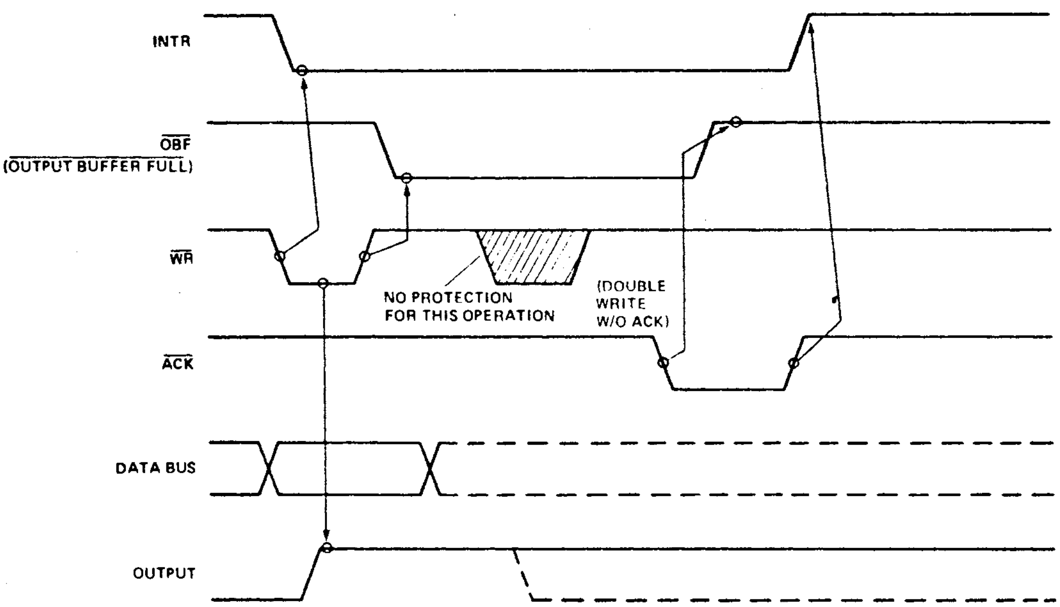


FIGURE 3-20. MODE 1 BASIC OUTPUT TIMING

Mode 2 (Strobed Bi-Directional Bus I/O):

This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bi-directional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to Mode 1. Interrupt generation and enable/disable functions are also available.

Mode 2 Basic Functional Definitions:

- Used in Port A only.
- One 8-bit, bi-directional data Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional data port (Port A).

Bi-Directional Bus I/O Control Signal Definition

INTR (Interrupt Request)

A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operation Control Signals

$\overline{\text{OBF}}$ (Output Buffer Full)

The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to Port A.

$\overline{\text{ACK}}$ (Acknowledge)

A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high-impedance state.

INTR A and B (The INTE flip-flop associated with $\overline{OBF}$)

Controlled by bit set/reset of PC6 (INTE1)

Input Operation Control Signals

$\overline{STB}$ (Strobed Input)

A "low" on this input indicates that data has been loaded into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE flip-flop associated with IBF)

Controlled by bit set/reset PC4 (INTE 2)

$$INTR_A = PC6 \cdot OBF_A + PC4 \cdot IBF_A$$

Figure 3-21 illustrates the port configuration for Mode 2, Figure 3-22 shows Mode 2 timing, and Table 3-1 summarizes 8255 Mode definition.

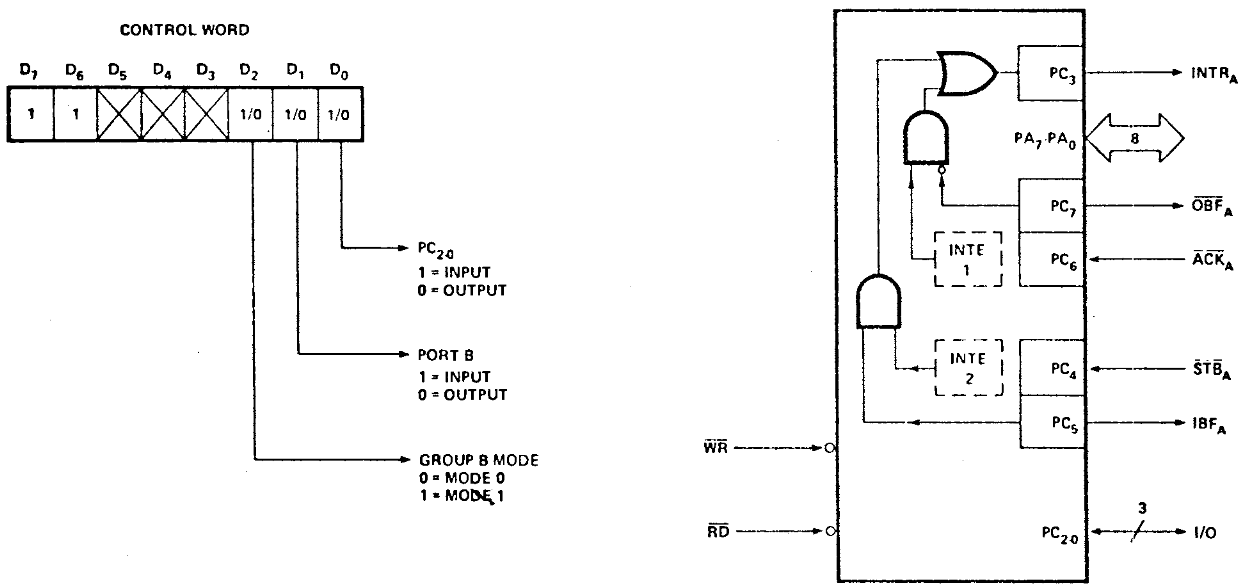


FIGURE 3-21. MODE 2 PORT CONFIGURATION

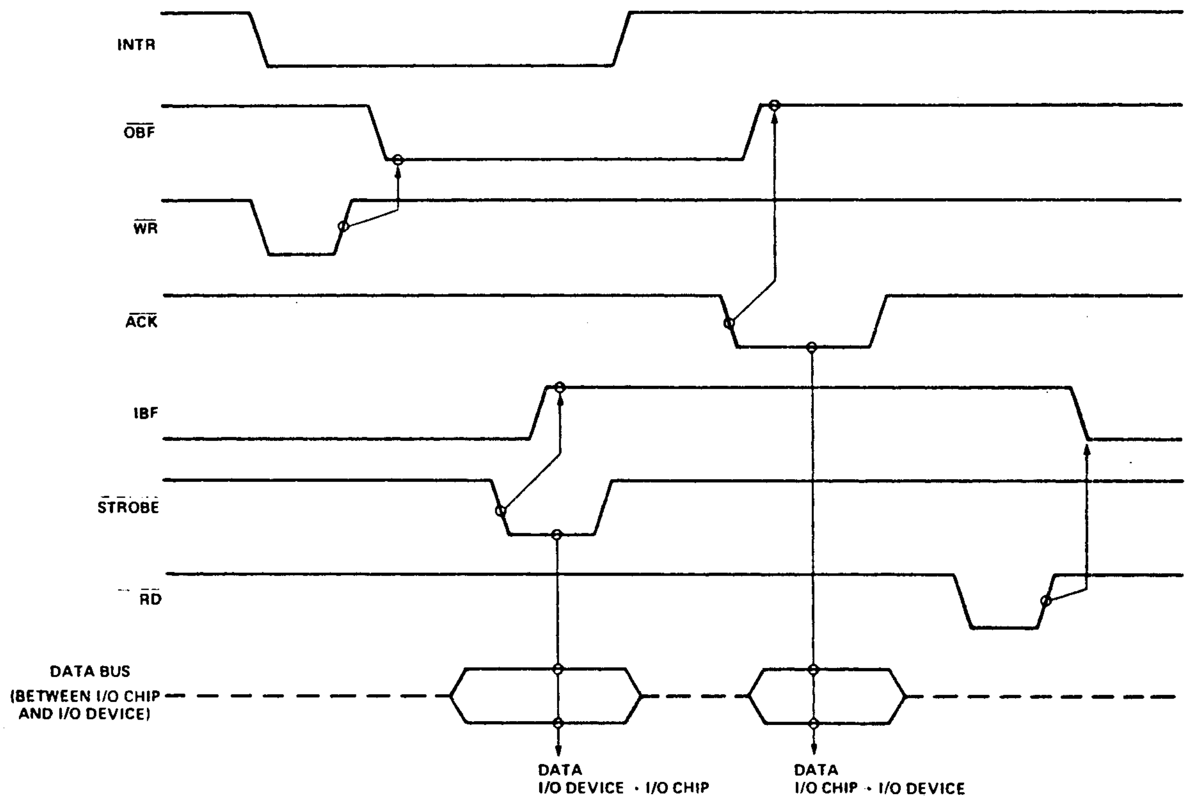


FIGURE 3-22. MODE 2 TIMING

MODE DEFINITION SUMMARY TABLE

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA_0	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_1	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_2	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_3	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_4	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_5	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_6	IN	OUT	IN	OUT	$\longleftrightarrow$
PA_7	IN	OUT	IN	OUT	$\longleftrightarrow$
PB_0	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_1	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_2	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_3	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_4	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_5	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_6	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PB_7	IN	OUT	IN	OUT	— (MODE 0 OR MODE 1 ONLY)
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC_5	IN	OUT	IBF_A	I/O	IBF_A
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$

TABLE 3-1. 8255 MODE DEFINITION SUMMARY

3.6.2 PARALLEL I/O CONFIGRATIONS

Referring to sheet 5 of the schematic, we see that there are two 8255 devices, one located at A19, the other at A20. For convenience the following device designations will be used: The device at A19 is called the "group 1" device, while the device at A20 is referred to as the "group 2" device. Each device has three eight-bit ports. The "group 1" ports are designated Ports 1, 2 and 3 while the "group 2" ports are designated Ports 4, 5 and 6.

The group 1 and group 2 devices both communicate with the CPU Set using the same signal lines: the 8-bit data bus, DB0-DB7, and seven control/address lines; ADR0, ADR1, RESET, IOR/, IOW/, CS1/, and CS2/. The data lines bring control bytes or data bytes to an 8255 or deliver data from an 8255 to the CPU Set. The chip select control signals (CS1, and CS2/) select the group 1 and group 2 devices, respectively, when the proper I/O address appears on the system address bus. CS1/ and CS2/ are the result of decoding address bits 2 through 7 (ADR2-ADR7), as shown on sheet 4 of the schematic (at A14). The two least significant address bits select the control register (when programming an 8255) or one of the three I/O ports (when reading or writing data). IOR/ (8255 → CPU Set) and IOW/ (CPU Set → 8255) indicate the direction of data flow, as summarized in Table 3-2. Specific I/O addresses for the six ports and two 8255 control registers on the SBC-80/10 are listed in Table 3-3.

A high on the RESET line clears all internal 8255 registers including the control register; all ports (A, B and C) are set for input.

Though both 8255's maintain the same interface (at different

TABLE 3-2. 8255 BASIC OPERATION

A1	A0	IOR/	IOW/	CS/	Input Operation (Read)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (Write)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
x	x	x	x	1	Data Bus → High-Impedance
1	1	0	1	0	Illegal

TABLE 3-3. PARALLEL I/O PORT ADDRESSES

Port	8255 Device Location	*Eight-Bit Address (Hexadecimal)
1	8255 #1 Port (A)	E4
2	8255 #1 Port (B)	E5
3	8255 #1 Port (C)	E6
-	8255 #1 Control	E7 For I/O write only.
4	8255 #2 Port (A)	E8
5	8255 #2 Port (B)	E9
6	8255 #2 Port (C)	EA
-	8255 #2 Control	EB For I/O write only.

*Note: If address = 111001xx, CS1/ is activated.
If address = 111010xx, CS2/ is activated.

I/O addresses) with the CPU Set, the interface between the group 1 device and edge connector J1 is significantly different than the interface between the group 2 device and its associated edge connector (J2). This gives the user a great deal of flexibility when configuring the system's external parallel I/O devices. Because of those flexible "external" interfaces, however, not all ports are capable of operating in each 8255 mode, though all ports can be programmed as either input or output. The group 1 ports can fully utilize the 8255's multi-mode and external interrupt capabilities as described in Section 3.6.1. The group 2 ports, however, are limited to a single mode of operation. The allowable port configurations for both groups are summarized below:

<u>Port 1 (Group 1 Port A)</u>

Mode 0 Input
Mode 0 Output (Latched)
Mode 1 Input (Strobed)
Mode 1 Output (Latched)
Mode 2 Bidirectional

<u>Port 2 (Group 1 Port B)</u>

Mode 0 Input
Mode 0 Output (Latched)
Mode 1 Input (Strobed)
Mode 1 Output (Latched)

<u>Port 3 (Group 1 Port C)</u>

Mode 0 8 Bit Input
Mode 0 8 Bit Output (Latched)

Note: Control mode dependent upon Port A and B mode.

<u>Ports 4 and 5 (Group 2 Port A, B)</u>

Mode 0 Input
Mode 0 Output (Latched)

Port 6 (Group 2 Port C)

Mode 0 8 Bit Input
Mode 0 8 Bit Output
Mode 0 4 Bit Input/4 Bit Output (Unlatched/latched)
Mode 0 4 Bit Output/4 Bit Input (Latched/unlatched)

Group 1

Port 1 is the most versatile of the six ports. It can be programmed to function in any one of the three 8255 operating modes. This first port is the only port that already includes a permanent bidirectional driver/termination network (two 8226 bus driver devices at A1 and A2).

Before Port 1 is programmed for input or output in any one of three operating modes (as described in Section 3.6.1), certain jumper connections must be made to allow the port to function properly in the chosen mode. The 40-41-42-43 jumper pad specifies the direction of data flow for the two 8226 bidirectional bus drivers. If input in mode 0 or mode 1 is to be programmed for Port 1, jumper pair 41-42 should be connected. If output in mode 0 or mode 1 is to be used, jumper pair 40-41 should be connected. If Port 1 is to be programmed for bidirectional mode 2, then jumper pair 41-43 should be connected. This connection allows the output acknowledge, ACK/, that is input on bit 6 of Port 3 to dynamically dictate direction for the two 8226 devices.

Another jumper pad (48-49-50-51) enables interrupts for Port 1 when it is in mode 1 or mode 2. Jumper pair 49-50 should be connected to allow the INTR output (see Section 3.6.1) from bit 3 of Port 3 to activate an interrupt request (INT55/) from the 74LS02 gate at A45. In mode 0, during which there is no provision for interrupts, jumper pairs 48-49 and 50-51 must be connected to allow use of bit 3 of port 3 and to inhibit Port 1 interrupts.

Because the 8226 bus drivers are inverting devices, all data input to or output from Port 1 is considered to be negative true with respect to the levels at the J1 edge connector.

Port 2 can be programmed for input or output in either mode 0 or mode 1 (see Section 3.6.1). If Port 2 is to be used for input (in either mode), terminator networks must be installed in the sockets at A5 and A6. Because these networks must be passive, data that is input to Port 2 will be positive true. If Port 2 is to be used for output (in either mode), driver networks must be installed in the sockets at A5 and A6. Assuming that the drivers are inverting devices, then the data being output will be negative true at the J1 edge connector.

When Port 2 is programmed for mode 1, interrupts can be enabled by connecting jumper pair 45-46. This connection allows the INTR output from bit 0 of Port 3 to activate the interrupt request (INT55/) to the CPU set. When Port 2 is in mode 0, jumper pairs 44-45 and 46-47 must be connected to allow use of bit 0 of Port 3 and to inhibit Port 2 interrupts.

As was described in Section 3.6.1, the use of Port 3 is dependent on the modes programmed for Ports 1 and 2. If Port 1 is in mode 1 or mode 2, bits 3, 4, 5, 6 and 7 of Port 3 can have dedicated control functions.

Port 3 bit 3	→	INTR	(interrupt request) - input or output	
Port 3 bit 4	←	STB/	(input strobe)	mode 1 input or mode 2
Port 3 bit 5	→	IBF	(input buffer full flag)	
Port 3 bit 6	←	ACK/	(output acknowledge)	mode 1 output or mode 2
Port 3 bit 7	→	OBF/	(output buffer full flag)	

If Port 2 is in mode 1, bits 0, 1 and 2 of Port 3 have dedicated control functions:

Port 3 bit 0	→	INTR	(interrupt request)	- input or output
Port 3 bit 1	→	IBF	(input buffer full)	input only
Port 3 bit 2	←	STB/	(input strobe	input only
Port 3 bit 1	→	OBF/	(output buffer full)	output only
Port 3 bit 2	←	ACK/	(output acknowledge)	output only

While certain Port 3 bits are available if Port 1 is in mode 1 or if Port 2 is in mode 0, the use of Port 3 as an eight-bit data path is restricted to those configurations that have both Port 1 and Port 2 programmed for mode 0. In this case all 8 bits of Port 3 can be programmed for mode 0 input (termination networks must be installed in the sockets at A3 and A4) or output (driver networks must be installed at A3 and A4). Note: If Port 1 and 2 are not both in mode 0, then a driver network must be installed in the sockets at A3 and a termination network must be installed at A4, so that the Port 3 control lines can function properly.

Group 2

The three ports on the group 2 device can be programmed for input or output, but only in mode 0. If Port 4 is programmed for input, termination networks must be installed in the sockets at A7 and A8. The data being input will be in positive true form. If Port 4 is programmed for output, driver networks must be installed at A7 and A8. Assuming that inverting drivers are used, then the data will be considered negative true at the J2 edge connector.

If Port 5 is programmed for input, termination networks must be installed in the sockets at A21 and A11. If Port 5 is programmed for output, driver networks must be installed at A21 and A11.

All eight bits of Port 6 can be programmed for input or output, or four bits can be programmed for input while the other four bits are programmed for output (see Section 3.6.1). Driver termination networks must be installed in the sockets at A9 and A10 as listed in Table 3-4.

TABLE 3-4. Port 6 I/O CONFIGURATIONS

	Sockets at A9	Sockets at A10
8-bit Input	Terminators*	Terminators*
8-bit Output	Drivers**	Drivers**
Upper 4-bits Input/ Lower 4-bits Output	Terminators*	Drivers**
Lower 4-bits Input/ Upper 4-bits Output	Drivers**	Terminators*

*Positive-true data.
**Negative-true data if inverting drivers.

In Section 4.2, all of the user options for configuring parallel I/O on the SBC-80/10 are summarized for convenient reference.

CHAPTER 4

USER SELECTABLE OPTIONS

The SBC-80/10 provides the user with a powerful, but flexible I/O capability for both parallel and serial transfers. The serial I/O Interface, using Intel's 8251 USART, provides a serial data communications channel that can be programmed to operate with most of the current serial data transmission protocols. Synchronous or asynchronous mode, baud rate, character length, number of stop bits and even/odd parity are all program selectable. In addition, the user has the option, through jumper connections, of configuring the Serial I/O Interface as an EIA RS232C interface or as a Teletype-compatible current loop interface.

The Parallel I/O Interface, using two Intel 8255 Programmable Peripheral Interface devices, provides 48 signal lines for the transfer and control of data to or from peripheral devices. Eight lines already have a bidirectional driver and termination network permanently installed. The remaining 40 lines, however, are uncommitted. Sockets are provided for the installation of active driver networks or passive termination networks as required to meet the specific needs of the user system.

In this chapter, we will reiterate each of the options available to the user, and summarize, for easy reference, the specific information required to implement the user's tailored I/O configuration. Section 4.1 deals with the Serial I/O Interface, while Section 4.2 covers Parallel I/O options. Section 4.3 will describe general options not covered in the other two sections.

4.1 SERIAL I/O INTERFACE OPTIONS

There are three general areas of Serial I/O options:

1) choice of interface type, RS232C or current loop,

2) baud rate and program-selectable mode options,

3) choice of an interrupt request mechanism.

The first two are covered in the following paragraphs; the third, choice of interrupt mechanism, is quite simple and is fully explained in Section 3.5.4.

4.1.1 INTERFACE TYPE

The user has the choice of configuring the Serial I/O logic to present either an EIA RS232C or a 20 mA current loop interface to an external device. If a Teletype-compatible current loop interface is used, the 8251 I/O pins should be connected to the external Teletype lines as listed in Table 4-1. The reader control logic is controlled by the output DSR/ from the 8251. If an EIA RS232C interface is used, the 8251 can assume the role of a "data set" (see Table 4-2a) or a partial "data processing terminal" (see Table 4-2b)[1]. Pin definitions for the 8251 USART are listed in Section 3.5.1.

4.1.2 BAUD RATE AND PROGRAM-SELECTABLE SERIAL I/O OPTIONS

Before beginning Serial I/O operations, the 8251 must be program-initialized to support the desired mode of operation. The CPU initializes the 8251 by outputting a set of control bytes to the USART device. These control words specify:

* synchronous or asynchronous operation,
* baud rate factor,
* character length,
* number of stop bits,
* even/odd parity.
* parity/no parity

[1]In this role, cable modifications must be made to conform with RS232 standards.

TABLE 4-1. 20 mA CURRENT LOOP SERIAL I/O INTERFACE

8251 PIN MNEMONIC	PIN NO.		CONNECTOR PIN NO.	JUMPER[3] CONNECTIONS
TXD	19	TTY Tx	J3-25	1-2
DTR/	24	TTY RD CONTROL	J3-6	23-24
(1) RTS/	23	(CTS/)	-	27-29, 30-31
(1) CTS/	17	(RTS/)	-	27-29
(2) TXC	9	(Baud Rate Clk)	-	33-34 (8-4, 56-57)
(2) RXC	25	(Baud Rate Clk)	-	35-36 (8-4, 56-57)
TXD	3	TTY Rx	J3-22	38-39
-	-	TTY Rx RET	J3-23	-
-	-	TTY Tx RET	J3-24	-
-	-	TTY RD CTL RET	J3-16	-

Notes: (1) The 8251's RTS/ output is connected to the CTS/ input through jumper pair 27-28. The command instruction word for the 8251 must enable RTS/.

(2) TXC and RXC are connected to the Baud Rate Clk line via jumpers 33-34 and 35-36. The Baud Rate Clk should be configured for 110 baud by connecting jumpers 8-4 and 56-57 (see Table 4-3), and the 8251 should be programmed for a baud rate factor of 64 (see Section 4.2).

(3) The SBC 80/10 comes with these jumper connections made.

TABLE 4-2a. RS232C INTERFACE, "DATA SET" ROLE

8251 PIN MNEMONIC	PIN NO.	LINE FUNCTION	CONNECTOR PIN NO.	JUMPER CONNECTIONS	JUMPER REMOVAL
RXD	3	TRANSMITTED DATA	J3-3	37-38	39-38
TXD	19	RECEIVED DATA	J3-5	2-3	1-2
(1) CTS/	17	REQ TO SEND	J3-7	27-28	27-29
RTS/	23	CLEAR TO SEND	J3-9	29-30	-
DTR/	24	DATA SET READY	J3-11	22-23	23-22
(2) DSR/	22	DATA TERMINAL RDY	J3-14	25-26	-
-	-	PROTECTIVE GROUND	J3-1	-	-
-	-	SIGNAL GROUND	J3-13	-	-

TABLE 4-2b. RS232C INTERFACE, "DATA PROCESSING TERMINAL" ROLE[1]

8251 PIN MNEMONIC	PIN NO.	LINE FUNCTION	CONNECTOR PIN NO.	JUMPER CONNECTIONS	JUMPER REMOVAL
TXD	19	TRANSMITTED DATA	J3-5	2-3	1-2
RXD	3	RECEIVED DATA	J3-3	37-38	36-39
RTS/	23	REQ TO SEND	J3-9	29-30	27-29
(1) CTS/	17	CLEAR TO SEND	J3-7	27-28	-
DTR/	24	DATA TERMINAL RDY	J3-11	22-23	23-24
(3) TXC	9	TRANSMIT CLOCK	J3-14	32-33	26-25
(2) DSR/	22	DATA SET RDY	J3-14	25-26	-
(3) RXC	25	RECEIVE CLOCK	J3-22	36-39	35-36
-	-	PROTECTIVE GROUND	J3-1	-	-
-	-	SIGNAL GROUND	J3-13	-	-

Notes: (1) The CTS/ input pin on the 8251 must be "low" to enable the 8251 to transmit.

(2) When connector pin J3-14 is jumpered (25-26) to the DSR/ input, J3-14 cannot be used to supply an external transmit clock.

(3) In the asynchronous mode, TXC and RXC can be connected to externally supplied clocks via jumpers 32-33 and 36-39, or they can be connected to the internal Baud Rate Clk via jumpers 33-34 and 35-36, regardless of the mode.

[1] In this role, cable modifications must be made to conform with RS232 standards.

As explained in Section 3.5.1, there are two types of control words: (1) Mode instruction and (2) Command instruction. The Mode instruction initializes the 8251 USART. Because the USART supports either synchronous or asynchronous operation, the Mode instruction has one format for synchronous operation and another for asynchronous. The two least significant bits of the Mode instruction byte specify the format. If D0 and D1 both equal 0, synchronous operation is indicated; otherwise, it is asynchronous. The Mode instruction format for asynchronous operation is illustrated in Figure 3-8. The Mode instruction for synchronous operation is shown in Figure 3-9.

Notice in Figure 3-8 that the baud rate factor is specified by the two least significant bits of the instruction byte (labeled B1 and B2). During asynchronous communications, the Baud Rate Clock frequency supplied to the 8251's TXC and RXC input pins is divided by the baud rate factor to produce the effective baud rate (i.e., the frequency at which data bits are serially transmitted by the 8251 USART). Consequently, the Baud Rate Clock, as well as the program-selected baud rate factor, must be considered in implementing the desired effective baud rate. The Baud Rate Clock frequency is selected through various jumper connections as shown on sheet 4 of the SBC-80/10 schematic (Appendix A). The selection of an effective baud rate is summarized in Table 4-3.

Notice from the schematic that TXC and RXC inputs can be supplied by externally supplied clocks (via connector pins J3-14 and J3-22, respectively), instead of using the Baud Rate Clock, if jumpers 32-33 and 36-39 are connected and jumpers 33-34 and 35-36 are disconnected.

TABLE 4-3. BAUD RATE SELECTION

JUMPER CONNECTION	EFFECTIVE BAUD RATE (Hz)		
	SYNCHRONOUS MODE	ASYNCHRONOUS MODE	
		BAUD RATE FACTOR=16 (2)	BAUD RATE FACTOR=64 (2)
10-4	-		4800
11-4	-	9600 (3)	2400
12-4	-	4800	1200
5-4	38,400	2400	600
6-4	19,200	1200	300
7-4	9600	600	150
(1) 8-4	4800	300	75
(1) 8-4, 56-57	6980	-	110 (TTY)

Note: (1) If jumper pair 56-57 is not connected, the frequency at jumper pole 8 is 4.8 KHZ. If jumper 56-57 is connected, however, the frequency at jumper pole 8 is 6.98 KHZ which, with a programmed baud rate factor of 64, provides an effective baud rate of approximately 110 baud for Teletype use.

(2) Baud rate factor is software selectable.

(3) Caution: Baud Rate Factor = 16

4.2 PARALLEL I/O OPTIONS

The Parallel I/O Interface consists of six 8-bit I/O ports implemented with two Intel 8255 Programmable Peripheral Interface devices. The primary user considerations in determining how to use each of the six I/O ports are:

1) Choice of operating mode (as defined in Section 3.6.1),
2) direction of data flow (input, output or bidirectional),
3) choice of driver/termination networks for port's data path.

In the following paragraphs, we will define the capabilities of each port and summarize, in tables, that information which is necessary to use the port in each of its potential configurations. Each table will list the port I/O address, the control register address and the format for the control word which is output to the 8255 by the CPU Set and which specifies the particular configuration to be used. Each table will also summarize all of the relevant information concerning the choice and use of driver/termination networks, the data polarity, the connecting of jumpers and what they enable, and any restrictions on the use of the other two ports in each group. Examples of suitable driver/ termination networks are listed in Section 5.1.

4.2.1 PORT 1 (GROUP 1 PORT A)

Port 1 is the only port that already includes a permanent bi-directional driver/termination network (two 8226 Bidirectional Bus Drivers). Port 1 is also the only port which can be programmed to function in any one of the three 8255 operating modes, which were defined in Section 3.6.1. Before Port 1 is programmed for input or output in any one of the three modes, certain jumper connections must be made to allow the port to function properly in the chosen mode. Other jumper connections must be made to enable interrupts when Port 1 is in mode 1 or mode 2. In all, there are five potential configurations for Port 1. All of the necessary information for implementing each configuration has been summarized in the following tables:

PORT 1 CONFIGURATIONS		TABLE
Mode	Direction	
1. Mode 0	Input	Table 4-4
2. Mode 0	Output (Latched)	Table 4-5
3. Mode 1	Input (Strobed)	Table 4-6
4. Mode 1	Output (Latched)	Table 4-7
5. Mode 2	Bidirectional	Table 4-8

TABLE 4-4. PORT 1, MODE 0 INPUT CONFIGURATION

PORT 1 ADDRESS: E4, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	1	x	x	x	x

DRIVER/TERMINATION NETWORKS: Two Intel® 8226 Bidirectional Bus Drivers permanently installed at A1 and A2.

DATA POLARITY: Negative-true.

JUMPER CONNECTIONS: 41-42 to enable input at 8226's. Remove 40-41.

PORT 2 RESTRICTIONS: None; port 2 can be programmed for mode 0 or mode 1, input or output (see Section 4.2.2)

PORT 3 RESTRICTIONS: None; port 3 can be programmed for mode 0, 8-bit input or output, unless port 2 is in mode 1. (see Section 4.2.3).

TABLE 4-5. PORT 1, MODE 0 LATCHED OUTPUT CONFIGURATION

PORT 1 ADDRESS: E4, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	0	x	x	x	x

DRIVER/TERMINATION NETWORKS: Two Intel® 8226 Bidirectional Bus Drivers permanently installed at A1 and A2.

DATA POLARITY: Negative-true.

JUMPER CONNECTIONS: 40-41 to enable output at 8226's.

PORT 2 RESTRICTIONS: None; port 2 can be programmed for mode 0 or mode 1, input or output (see Section 4.2.2)

PORT 3 RESTRICTIONS: None; port 3 can be programmed for mode 0, input or output, unless port 2 is in mode 1 (see Section 4.2.3).

TABLE 4-6. PORT 1, MODE 1 STROBED INPUT CONFIGURATION

PORT 1 ADDRESS: E4, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
0/1	0/1	1	1	0/1	x	x	x

DRIVER/TERMINATION NETWORKS: Two Intel®8226 Bidirectional Bus Drivers permanently installed at A1 and A2. A driver network must be installed at A3 and a termination network must be installed at A4.

DATA POLARITY: Negative-true. The polarity of Port 3 control outputs is dependent on the type of driver installed at A3.

JUMPER CONNECTIONS: 41-42 to enable input at 8226's; connect 49-50 to enable interrupt request via INT55/. Remove 40-41, 48-49, 50-51.

PORT 2 RESTRICTIONS: None; port 2 can be programmed for mode 0 or mode 1, input or output (see Section 4.2.2).

PORT 3 RESTRICTIONS: Port 3 bits perform the following dedicated functions:

*Bits 0, 1 and 2 - dedicated to control of port 2 if port 2 is in mode 1 (see Tables 4-9 to 4-12).
*Bit 3 - INTR (interrupt request) output for port 1.
*Bit 4 - STB/ (strobe) input for port 1.
*Bit 5 - IBF (input buffer full) output for port 1.
*Bit 6 & *Bit 7 - Only one bit can be used. If input use bit 6; do not use bit 7. Bit 3 of Control Word=1. If output use bit 7 and remove jumper between 13-14; do not use bit 6. Bit 3 of Control Word=0.

TABLE 4-7. PORT 1, MODE 1 LATCHED OUTPUT CONFIGURATION

PORT 1 ADDRESS: E4, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	1	0	x	x	x	x

DRIVER/TERMINATION NETWORKS: Two Intel®8226 Bidirectional Bus Drivers permanently installed at A1 and A2. A driver network must be installed at A3 and a termination network must be installed at A4.

DATA POLARITY: Negative-true. The polarity of Port C control outputs is dependent on the type of driver installed at A3.

JUMPER CONNECTIONS: 40-41 to enable output at 8226's; connect 49-50 to enable interrupt request via INT55/. Remove 48-49, 50-51.

PORT 2 RESTRICTIONS: None; port 2 can be programmed for mode 0 or mode 1, input or output (see Section 4.2.2).

PORT 3 RESTRICTIONS: Port 3 bits perform the following dedicated functions:

*Bits 0, 1 and 2 - dedicated to the control of port 2 if port 2 is in mode 1 (see Tables 4-11 and 4-12).
*Bit 3 - INTR (interrupt request) output for port 1.
*Bit 4 - can be used for input if bit 3 of control word = 1
*Bit 5 - cannot be used if PC4 is used; can be used for output if control word bit 3 = 0 (PC4 cannot be used then).
*Bit 6 - ACK/ (acknowledge) input for port 1.
*Bit 7 - OBF/ (output buffer full) output for port 1.

TABLE 4-8. PORT 1, MODE 2 BIDIRECTIONAL CONFIGURATION

PORT 1 ADDRESS: E4, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	1	x	x	x	x	x	x

DRIVER/TERMINATION NETWORKS: Two Intel® 8226 Bidirectional Bus Drivers permanently installed at A1 and A2. A driver network must be installed at A3 and a termination network must be installed at A4.

DATA POLARITY: Negative-true. The polarity of Port C control outputs is dependent on the type of driver installed at A3.

JUMPER CONNECTIONS: 41-43 to allow ACK/ input on PC6 to dynamically change data direction at 8226's (input when ACK/ = 1 and output when ACK/ = 0); connect 49-50 to enable interrupt request via INT55/. Remove 40-41, 48-49, 50-51.

PORT 2 RESTRICTIONS: None.

PORT 3 RESTRICTIONS: Port 3 bits perform the following dedicated functions:

*Bits 0 and 1 - can be used for output if bit 3 of control word = 0
*Bit 2 - cannot be used if PC0 and PC1 are used; can be used for input if control word bit 3 = 1 (PC0 and PC1 cannot be used then).
*Bit 3 - INTR (interrupt request) output for port 1.
*Bit 4 - STB/ (strobe input for port 1.
*Bit 5 - IBF (input buffer full) output for port 1.
*Bit 6 - ACK/ (acknowledge) input for port 1.
*Bit 7 - OBF/ (output buffer full) output for port 1.

4.2.2 PORT 2 (GROUP 1 Port B)

Port 2 can be programmed for input or output in either mode 0 or mode 1. If Port 2 is to be used for input, in either mode, terminator networks must be installed in the sockets at A5 and A6. If Port 2 is to be used for output, in either mode, driver networks must be installed in the sockets at A5 and A6. When Port 2 is programmed for mode 1, interrupts can be enabled by connecting jumper pair 45-46. The four potential configurations for Port 2 are summarized in the following tables:

PORT 2 CONFIGURATIONS		TABLE
Mode	Direction	
1. Mode 0	Input	Table 4-9
2. Mode 0	Output (Latched)	Table 4-10
3. Mode 1	Input (Strobed)	Table 4-11
4. Mode 1	Output (Latched)	Table 4-12

TABLE 4-9. PORT 2, MODE 0 INPUT CONFIGURATION

PORT 2 ADDRESS: E5, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	x	x	x	x	0	1	x

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A5 and A6.

DATA POLARITY: Positive-true.

JUMPER CONNECTION: None.

PORT 1 RESTRICTIONS: None (see Section 4.2.1).

PORT 3 RESTRICTIONS: None, port 3 can be programmed for mode 0, input or output, unless port 1 is in mode 1 or mode 2 (see Section 4.2.3).

TABLE 4-10. PORT 2, MODE 0 LATCHED OUTPUT CONFIGURATION

PORT 2 ADDRESS: E5, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	x	x	x	x	0	0	x

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A5 and A6.

DATA POLARITY: Negative-true, assuming that inverting drivers are at A5 and A6.

JUMPER CONNECTIONS: None.

PORT 1 RESTRICTIONS: None (see Section 4.2.1).

PORT 3 RESTRICTIONS: None, port 3 can be programmed for mode 0 or mode 1, 8-bit input or output, unless port 1 is in mode 1 or mode 2 (see Section 4.2.3).

TABLE 4-11. PORT 2, MODE 1 STROBED INPUT CONFIGURATION

PORT 2 ADDRESS: E5, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	x	x	x	1	1	x

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A5 and A6. A driver network must be installed at A3 and a termination network must be installed at A4.

DATA POLARITY: Positive-true. The polarity of Port C control outputs is dependent on the type of driver installed at A3.

JUMPER CONNECTIONS: 45-46 to enable interrupt request via INT55/.
Remove 44-45, 46-47.

PORT 1 RESTRICTIONS: None.

PORT 3 RESTRICTIONS: Port 3 bits perform the following dedicated functions:

*Bit 0 - INTR (interrupt request) output for port 2.
*Bit 1 - IBF (input buffer full) output for port 2.
*Bit 2 - STB/ (strobe) input for port 2.
*Bit 3 to Bit 7 - dedicated to control of port 1 if port 1 is in mode 1 (see Tables 4-4 to 4-7).

TABLE 4-12. PORT 2, MODE 1 LATCHED OUTPUT CONFIGURATION

PORT 2 ADDRESS: E5, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	x	x	x	1	0	x

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A5 and A6. A driver network must be installed at A3 and a termination network must be installed at A4.

DATA POLARITY: Negative-true, assuming that inverting drivers are at A5 and A6. The polarity of Port C control outputs is dependent on the type of driver installed at A3.

JUMPER CONNECTIONS: 45-46 to enable interrupt request via INT55/
Remove 44-45, 46-47.

PORT 1 RESTRICTIONS: None.

PORT 3 RESTRICTIONS: Port 3 bits perform the following dedicated functions:

*Bit 0 - INTR (interrupt request) output for port 2.
*Bit 1 - OBF/ (output buffer full) output for port 2.
*Bit 2 - ACK/ (acknowledge) input for port 2.
*Bit 3 - P3-7 - dedicated to control of port 1 if port 1 is in mode 1 (see Tables 4-4 to 4-7).

4.2.3 PORT 3 (GROUP 1 PORT C)

The use of Port 3 is dependent on the modes programmed for Ports 1 and 2 (refer to Tables 4-4 to 4-12). While certain Port 3 bits are available if Port 1 is in mode 1 or if Port 2 is in mode 0, the use of Port 3 as an 8-bit data path is restricted to those configurations that have both Port 1 and Port 2 programmed for mode 0. In this case, all eight bits of Port 3 can be programmed for mode 0 input (see Table 4-13) or output (see Table 4-14). A 4-bit input/4-bit output configuration is never possible for group 1 Port C.

Note: If Ports 1 and 2 are not both in mode 0, then a driver network must be installed in the sockets at A3 and a termination network must be installed at A4, so that the Port 3 control lines can function properly.

4.2.4 PORTS 4 AND 5 (GROUP 2 PORTS A AND B)

Ports 4 and 5 can be programmed for input or output but only in mode 0. The two potential configurations for each port are summarized in the following tables:

CONFIGURATIONS			TABLE
PORT	MODE	DIRECTION	
1. Port 4	Mode 0	Input	Table 4-15
2. Port 4	Mode 0	Output (Latched)	Table 4-16
1. Port 5	Mode 0	Input	Table 4-17
2. Port 5	Mode 0	Output (Latched)	Table 4-18

TABLE 4-13. PORT 3, MODE 0, 8-BIT INPUT CONFIGURATION

PORT 3 ADDRESS: E6, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	1	0	x	1

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A3 and A4.

DATA POLARITY: Positive-true.

JUMPER CONNECTIONS: 46-47 and 44-45 to disable port 2 interrupts and enable P3-0; connect 48-49 and 50-51 to disable port 1 interrupts and enable P3-3.

PORT 1 AND 2 RESTRICTIONS: Both ports 1 and 2 must be in mode 0.

TABLE 4-14. PORT 3, MODE 0, 8-BIT LATCHED OUTPUT CONFIGURATION

PORT 3 ADDRESS: E6, CONTROL REGISTER ADDRESS: E7

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	0	0	x	0

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A3 and A4.

DATA POLARITY: Negative-true, assuming that inverting drivers are installed at A3 and A4.

JUMPER CONNECTIONS: 46-47, and 44-45 to disable port 2 interrupts and enable P3-0; connect 48-49 and 50-51 to disable port 1 interrupts and enable P3-3.

PORT 1 AND 2 RESTRICTIONS: Both ports 1 and 2 must be in mode 0.

TABLE 4-15. PORT 4, MODE 0, INPUT CONFIGURATION

PORT 4 ADDRESS: E8, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	1	x	0	x	x

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A7 and A8.

DATA POLARITY: Positive-true.

JUMPER CONNECTIONS: None.

PORT 5 AND 6 RESTRICTIONS: None; ports 5 and 6 can be programmed for mode 0, input or output (also see Section 4.2.5).

TABLE 4-16. PORT 4, MODE 0 LATCHED OUTPUT CONFIGURATION

PORT 4 ADDRESS: E8, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	0	x	0	x	x

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A7 and A8.

DATA POLARITY: Negative-true, assuming that inverting drivers are installed at A7 and A8.

JUMPER CONNECTIONS: None.

PORT 5 AND 6 RESTRICTIONS: None; ports 5 and 6 can be programmed for mode 0, input or output (also see Section 4.2.5).

TABLE 4-17. PORT 5, MODE 0 INPUT CONFIGURATION

PORT 5 ADDRESS: E9, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	x	0	1	x

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A11 and A21.

DATA POLARITY: Positive-true.

JUMPER CONNECTIONS: None.

PORT 4 AND 6 RESTRICTIONS: None; ports 4 and 6 can be programmed for mode 0, input or output (also see Section 4.2.5).

TABLE 4-18. PORT 5, MODE 0 LATCHED OUTPUT CONFIGURATION

PORT 5 ADDRESS: E9, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	x	0	0	x

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A11 and A21.

DATA POLARITY: Negative-true, assuming that inverting drivers are installed at A11 and A21.

JUMPER CONNECTIONS: None.

PORT 4 AND 6 RESTRICTIONS: None; ports 4 and 6 can be programmed for mode 0, input or output (also Section 4.2.5).

4.2.5 PORT 6 (GROUP 2 PORT C)

All eight bits of Port 6 can be programmed for mode 0 input or output, or four bits can be programmed for mode 0 input while the other four bits are programmed for mode 0 output. The four potential configurations for Port 6 are summarized in the following tables:

PORT 6 CONFIGURATIONS	TABLE
1. MODE 0 8-BIT INPUT	Table 4-19
2. MODE 0 8-BIT OUTPUT (LATCHED)	Table 4-20
3. MODE 0 UPPER 4-BIT INPUT/LOWER 4-BIT OUTPUT	Table 4-21
4. MODE 0 UPPER 4-BIT OUTPUT/LOWER 4-BIT INPUT	Table 4-22

TABLE 4-19. PORT 6, MODE 0, 8-BIT INPUT CONFIGURATION

PORT 6 ADDRESS: EA, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	1	0	x	1

DRIVER/TERMINATION NETWORKS: Termination networks must be installed at A9 and A10.

DATA POLARITY: Positive-true.

JUMPER CONNECTIONS: None.

PORT 4 AND 5 RESTRICTIONS: None (see Section 4.2.4).

TABLE 4-20. PORT 6, MODE 0, 8-BIT LATCHED OUTPUT CONFIGURATION

PORT 6 ADDRESS: EA, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	0	0	x	0

DRIVER/TERMINATION NETWORKS: Driver networks must be installed at A9 and A10.

DATA POLARITY: Negative-true, assuming that inverting drivers are installed at A9 and A10.

JUMPER CONNECTIONS: None.

PORT 4 AND 5 RESTRICTIONS: None (see Section 4.2.4).

TABLE 4-21. PORT 6, MODE 0 UPPER 4-BIT INPUT/LOWER 4-BIT LATCHED OUTPUT CONFIGURATION

PORT 6 ADDRESS: EA, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	1	0	x	0

DRIVER/TERMINATION NETWORKS: A termination network must be installed at A9 and a driver network must be installed at A10.

DATA POLARITY: The upper 4-bits will be in positive-true form; however, the lower four bits will be in negative-true form if an inverting driver is installed at A10.

JUMPER CONNECTIONS: None.

PORT 4 AND 5 RESTRICTIONS: None (see Section 4.2.4).

TABLE 4-22. PORT 6, MODE 0 UPPER 4-BIT LATCHED OUTPUT/LOWER 4-BIT INPUT CONFIGURATION

PORT 6 ADDRESS: EA, CONTROL REGISTER ADDRESS: EB

CONTROL WORD FORMAT:

7	6	5	4	3	2	1	0
1	0	0	x	0	0	x	1

DRIVER/TERMINATION NETWORKS: A driver network must be installed at A9 and a termination network must be installed at A10.

DATA POLARITY: The lower 4-bits will be in positive-true form; however, the upper 4-bits will be in negative-true form if an inverting driver is installed at A9.

JUMPER CONNECTIONS: None.

PORT 4 AND 5 RESTRICTIONS: None (see Section 4.2.4).

TABLE 4-23. PARALLEL I/O ADDRESS AND SOCKET ASSIGNMENTS

PORT	I/O ADDRESS	SOCKET NUMBERS
1	E4	BI-DIRECTIONAL DRIVER/ TERMINATOR AT A1, A2
2	E5	A5, A6
3	E6	A3, A4*
4	E8	A7, A8
5	E9	A11, A21
6	EA	A9, A10**

*Note requirements specified in Tables 4-4 through 4-14.

**Note requirements specified in Tables 4-15 through 4-22.

4.3 GENERAL OPTIONS

There are several other options that may be useful. Details are provided in the following paragraphs.

4.3.1 SYSTEM RESET OUTPUT

The user can enable a SYSTEM RESET output from the SBC 80/10 by connecting jumper pair 54-55. This allows the reset signal which is generated on the SBC 80/10 during power-up sequences (see Section 3.1.5) to be made available to other modules in the system via connector P1-14. Notice on the schematic that a SYSTEM RESET input is accepted by the SBC 80/10 and P1-14 and applied to the 8080 regardless of jumper connections.

4.3.2 DISABLE BUS CLOCK SIGNALS

The bus clock BCLK/ (connector pin P1-13) or the constant clock CCLK/ (P-31) outputs can be disabled (if more drive, or a different frequency is needed) by disconnecting jumper pair 61-63 or 62-64, respectively. When connected, both BCLK/ and CCLK/ provide a 9.216 MHz timing reference to other modules.

4.3.3 ACKNOWLEDGE INPUTS

The SBC bus has defined two types of acknowledges; transfer acknowledge (XACK/) and advance acknowledge (AACK/). XACK/ is the <u>required</u> response of a memory or I/O port which indicates that the specified read/write operation has been completed. That is, data has been placed on (READ command) or accepted from (WRITE command) the system data bus lines. XACK/ is asynchronous with BCLK/. AACK/ is an advance acknow-

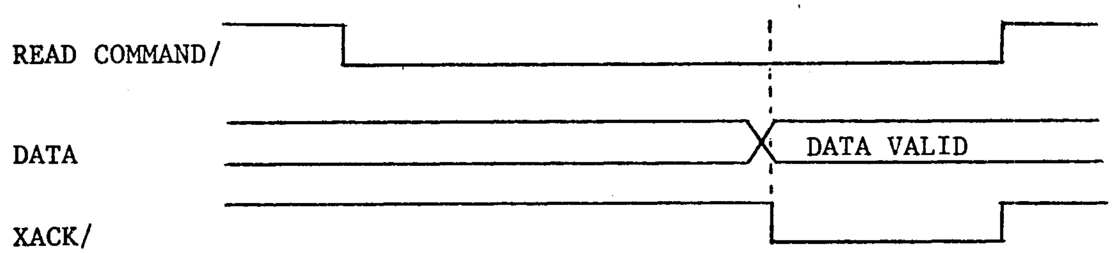


FIGURE 4-1. READ COMMAND WITH XACK/

ledge in response to a memory or I/O port command. This optional acknowledge is used only with 8080 CPU-based systems where maximum system performance is needed. Figure 4-2 shows timing of the SBC 80/10 "READING" memory using the AACK/ signal.

AACK/ is a response to a READ command indicating that data will be valid on the bus by the time the 8080 needs it. Thus, if the access time of the slave device is less than t_{ACC} (command to data needed by the 8080), the slave module has t_{8KD} (command to 8080 sample point of bus acknowledge) to indicate that the data on the bus will be valid when the 8080 needs it. If the access time of a module is less than t_{8KD} then only XACK/ need be used and the 8080 will run at maximum speed.

If the access time of a module is greater than t_{8KD}, but less than t_{ACC}, AACK/ must be used, if the 8080 is to run at maximum speed. If AACK/ is not used and instead XACK/ is used, the 8080 will execute one more wait state than is necessary. AACK/ is asynchronous with BCLK/.

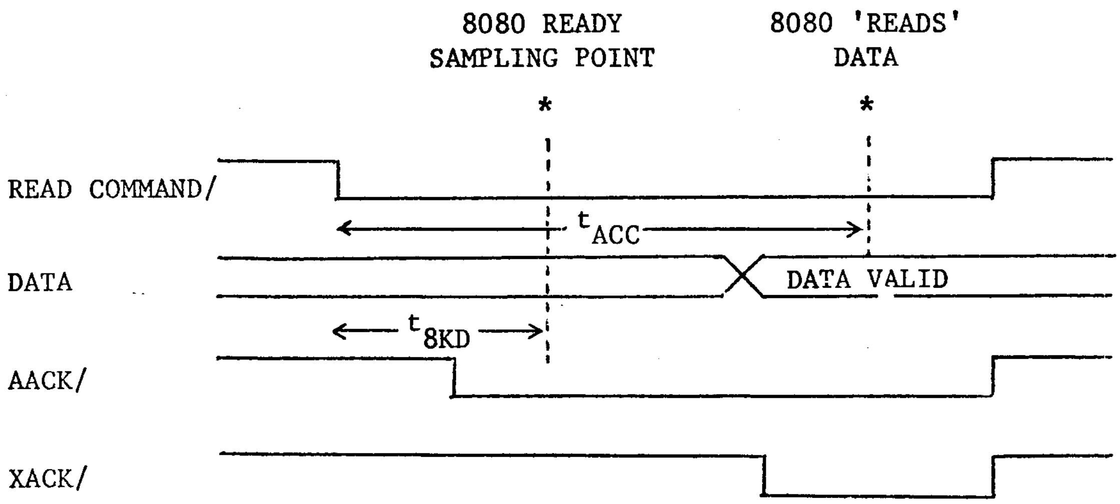


FIGURE 4-2. READ COMMAND WITH AACK/

AACK/ is also an advance response to a WRITE command indicating that the slave module will have accepted the data from the system bus by the time the 8080 has completed the WRITE. Figure 4-3 shows timing of the SBC 80/10 "WRITING" memory using the AACK/ signal.

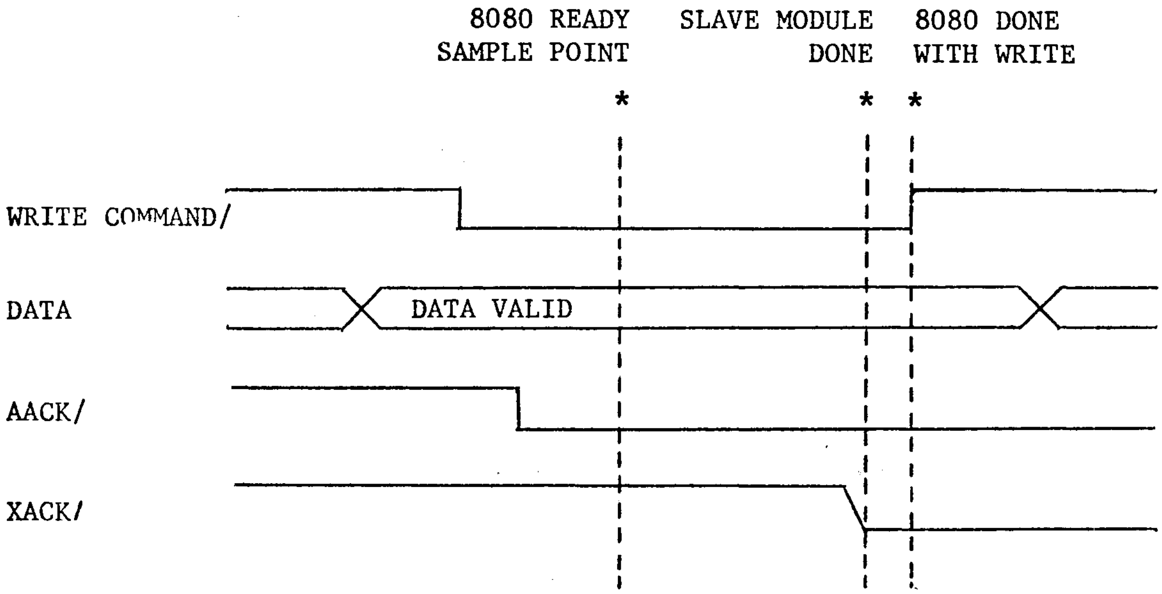


FIGURE 4-3. WRITE COMMAND WITH AACK/

When modules that generate proper AACK/ are used with the SBC 80/10, jumper pair 52-53 should be connected to allow AACK/ to be accepted (at P1-25) and gated to the RDYIN pin on the 8224 clock generator. If this option is used, caution should be taken to insure that all the modules on the bus meet the SBC 80/10 timing requirements.

4.3.4 INTERRUPT SOURCES

There are six sources of interrupts on the SBC 80/10 board, two from the serial I/O section (see 3.5.4), two from the parallel I/O section (see 3.6), and two from external sources. One of the external sources is INTR1/. INTR1/ is connected to the SBC bus (P1-42) and is the only interrupt line that other SBC modules can use to interrupt the SBC 80/10. The other external interrupt is EXT INTRØ/. This interrupt is connected to the parallel I/O connector J1 (pin 49) and can be used by an external device to interrupt the SBC 80/10. Both external interrupts are negative true logic, a TTL low ($V_{IN} < 0.4$ volts) will cause the 8080 to interrupt and execute a RST 7 instruction. The processor can then read in the status registers of the possible interrupting devices to determine which device generated the interrupt. Then the processor can jump to the correct interrupt service routine, service that device, enable interrupts, and return.

4.4 DEFAULT OPTIONS

Table 4-24 lists the default options jumpered on the SBC 80/10. These options permit the SBC 80/10 to communicate to a TTY; they also provide power-up reset, bus clock, and the communication clock to the system bus. If the SBC 80/10 is driving the bus clock (BCLK/) and/or the communications clock (CCLK/), the system bus must be limited to 7 inches. This limitation is due to the SBC 80/10's limited drive capability on these clock lines. The system bus can be extended beyond 7 inches if the user provides BCLK/ and CCLK/.

TABLE 4-24. DEFAULT OPTION

DEFAULT JUMPERS	REFERENCE	DESCRIPTION
1 - 2	4.1.1	Connect 8251 T_xD to 20 mA Current Loop Driver
23 - 24		Connect 8251 DTR/ to TTY Reader Control Circuit
39 - 38		Connect 8251 R_xD to 20 mA Current Loop Receiver
4 - 8	4.1.2	Generates 6.98K Baud Rate Clock
57 - 56		Generates 6.98K Baud Rate Clock
34 - 33		Connect 8251 T_x Clock to Baud Rate Clock
35 - 36		Connect 8251 R_x Clock to Baud Rate Clock
27 - 29		Connect 8251 RTS/ to 8251 CTS/
19 - 20	3.5.4	Disable T_xRDY Interrupt from 8251
16 - 15	3.5.4	Disable R_xRDY Interrupt from 8251
26 - 25	4.1.2	Connect DTR/ Receiver to 8251 DSR/ Input
30 - 31	4.1.2	Connect Set Clear to Send Driver to +12V
40 - 41	4.2.1	Enable Port 1 Bi-directional Drivers to Output
54 - 55	4.3.1	Connect Power-Up Reset to System Bus
62 - 64	4.3.2	Connect 9.216 MHz Clock to Communication Clock Line
61 - 63	4.3.2	Connect 9.216 MHz Clock to Bus Clock Line
*65 - 66	4.5	Configures SBC 80/10A for 4K ROM/PROM
*68 - 69	4.5	
*73 - 74	4.5	
*76 - 78	4.5	

*Used on SBC 80/10A only.

4.5 JUMPER CONFIGURATION FOR ROM/EPROM INSTALLATION

The SBC 80/10A has jumpers which allow installation of up to 4K or up to 8K bytes of read only memory. Up to 4K bytes can be installed using Intel's 8708 Erasable and Electrically Reprogrammable ROMs (EPROM), Intel's 8308 Metal Masked ROMs, or Intel's 2758 Erasable and Electrically Reprogrammable ROMs (EPROM). Up to 8K bytes can be installed using Intel's 2716 Erasable and Electrically Reprogrammable ROMs (EPROM) or Intel's 2316E Metal Masked ROMs. Table 4-25 lists the jumper configurations for 4K and 8K bytes of read only memory. Table 4-26 lists the addresses for each PROM socket in 4K and 8K configurations.

TABLE 4-25. PROM JUMPER CONFIGURATION

	JUMPER			
4K	65 - 66	68 - 69	73 - 74	76 - 78
8K	66 - 67	69 - 70	74 - 75	77 - 78
*4K	66 - 67	69 - 71	73 - 74	76 - 78

*Using Intel's 2758 Erasable and Electrically Reprogrammable ROMs (EPROM)

TABLE 4-26. PROM ADDRESSES

	CHIP ADDRESS			
	A23	A24	A25	A26
4K	0 - 3FF	400 - 7FF	800 - BFF	C00 - FFF
8K	0 - 7FF	1000 - 17FF	800 - FFF	1800 - 1FFF

CHAPTER 5

SYSTEM INTERFACING

The SBC-80/10, with its memory and I/O ports, is a complete computer on a single printed circuit board. However, the SBC-80/10 can also serve as a primary master module within an expanded system, communicating with numerous memory and I/O modules. In this chapter we identify each of the SBC-80/10's external connections and define all signals on the external system bus.

5.1 ELECTRICAL CONNECTIONS

The SBC-80/10 comes on a 12.00 X 6.75 inch printed circuit board, 0.50 inch thick and weighing 12 oz. The DC power requirements are listed in Table 7-1.

The SBC-80/10 has five edge connectors, as shown in Figure 5-1. Edge connectors at the top of the module are designed for compatibility with both flat cable and round cable hardware. All parallel I/O functions are paired with an independent signal ground pin. This allows flat cable implementation to utilize an alternate signal/ground scheme for reduction of cross talk. Round cables may easily be implemented as twisted pair with an individual ground pin for every return wire. The serial connection hardware has similar flexibility but ground return lines are not as extensive. The connector is wired for RS232C compatibility, thus, only one signal ground is provided.

CAUTION

All pin numbers listed in the following tables refer to numbers printed on the board, not to mating connector pin positions. When specifying pin numbers for cable harnesses, use caution since pin numbering is not necessarily the same as the connector pin numbering scheme.

The Parallel I/O Interface communicates with external I/O devices via two 50-pin double-sided PC edge connectors (J1 and J2), 0.1 inch centers. External devices can be attached to J1 or J2 using any of the following mating connectors:

J1 and J2 Mating Connectors

Connector Type	Vendor	Part No.
Flat Cable	3M AMP	3415-0001 2-86792-3
Soldered	AMP VIKING TI	2-583715-3 3VH25/1JV-5 H312125
Wire-wrap	TI VIKING CDC ITT	H 311125 3VH25/1JND-5 VPB01B25D00A1 EC4A050A1A
Crimp	AMP	1-583717-1

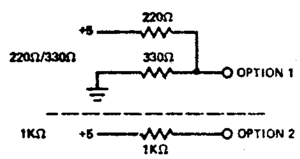
Tables 5-1 and 5-2 provide pin lists for the J1 and J2 connectors, respectively. The following TTL line drivers and Intel terminators are all compatible with the I/O driver sockets in the Parallel I/O Interface:

Driver	Characteristic	Sink Current (ma)
7438	I,OC	48
7437	I	48
7432	NI	16
7426	I,OC	16
7409	NI,OC	16
7408	NI	16
7403	I,OC	16
7400	I	16

Note: I = inverting; N.I. = non-inverting
OC = open collector

I/O Terminators:

Terminators: 220Ω/330Ω divider or 1 kΩ pull up



SBC-901

SBC-902

See Appendix C for schematics

TABLE 5-1. PIN ASSIGNMENTS FOR CONNECTOR J1
(Parallel I/O Interface - Group 1)

PIN	SIGNAL	PIN	SIGNAL
1	PORT 2 - BIT 3	2	GND
3	PORT 2 - BIT 2	4	↑
5	PORT 2 - BIT 1	6	
7	PORT 2 - BIT 0	8	
9	PORT 2 - BIT 4	10	
11	PORT 2 - BIT 5	12	
13	PORT 2 - BIT 6	14	
15	PORT 2 - BIT 7	16	
17	PORT 3 - BIT 3	18	
19	PORT 3 - BIT 2	20	
21	PORT 3 - BIT 4	22	
23	PORT 3 - BIT 6	24	
25	PORT 3 - BIT 0	26	
27	PORT 3 - BIT 5	28	
29	PORT 3 - BIT 1	30	
31	PORT 3 - BIT 7	32	
33	PORT 1 - BIT 7	34	
35	PORT 1 - BIT 6	36	
37	PORT 1 - BIT 5	38	
39	PORT 1 - BIT 4	40	
41	PORT 1 - BIT 1	42	
43	PORT 1 - BIT 0	44	
45	PORT 1 - BIT 2	46	
47	PORT 1 - BIT 3	48	↓
49	EXT INTR 1/	50	GND

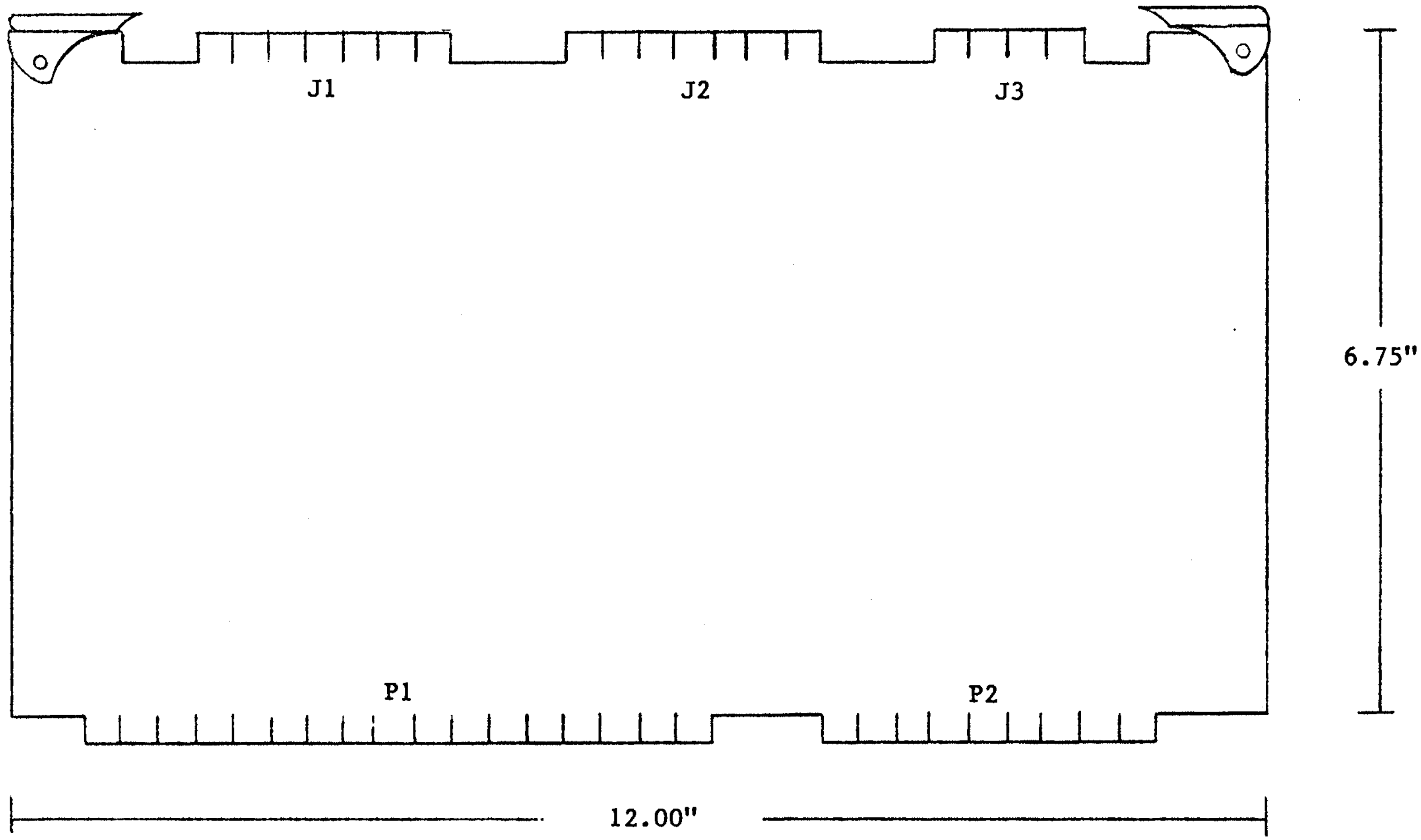


FIGURE 5-1. EDGE CONNECTORS

TABLE 5-2. PIN ASSIGNMENTS FOR CONNECTOR J2
(Parallel I/O Interface - Group 2)

PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND
3	PORT 5 - BIT 3	4	↑
5	PORT 5 - BIT 0	6	
7	PORT 5 - BIT 1	8	
9	PORT 5 - BIT 2	10	
11	PORT 5 - BIT 4	12	
13	PORT 5 - BIT 5	14	
15	PORT 5 - BIT 6	16	
17	PORT 5 - BIT 7	18	
19	PORT 6 - BIT 3	20	
21	PORT 6 - BIT 2	22	
23	PORT 6 - BIT 1	24	
25	PORT 6 - BIT 0	26	
27	PORT 6 - BIT 4	28	
29	PORT 6 - BIT 5	30	
31	PORT 6 - BIT 6	32	
33	PORT 6 - BIT 7	34	
35	PORT 4 - BIT 7	36	
37	PORT 4 - BIT 6	38	
39	PORT 4 - BIT 5	40	
41	PORT 4 - BIT 4	42	
43	PORT 4 - BIT 0	44	
45	PORT 4 - BIT 1	46	
47	PORT 4 - BIT 2	48	↓
49	PORT 4 - BIT 3	50	GND

The Serial I/O Interface communicates with an external I/O device via a 26-pin double-sided PC edge connector (J3), 0.1 inch centers. An external device can be connected to J3 using a 3M 3462-0001 flat cable connector or one of the following soldered connectors: TI H312113 or AMP 1-583715-1. Table 5-3 provides a pin list for connector J3.

The SBC-80/10 communicates with other system modules via an 86-pin double-sided edge connector (P1), 0.156 inch centers. This edge connector will accept any of the following mating connectors: CDC VPBO1E43A000A1, Micro Plastics MP-0156-43-BW-4 or ARCO AE 443WP1. Section 5.2 defines each of the external system bus signals and includes a pin list for P1 (Table 5-5).

TABLE 5-3. PIN ASSIGNMENTS FOR CONNECTOR J3
(Serial I/O Interface)

PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	CHASSIS GND	2	
3	TRANSMITTED DATA	4	
5	RECEIVED DATA	6	TTY RD CONTROL
7	REQ TO SEND	8	
9	CLEAR TO SEND	10	
11	DATA SET READY	12	
13	GND	14	Tx CLK/DATA TERMINAL RDY
15	DATA CARRIER RETURN	16	TTY RD CONTROL RETURN
17		18	
19		20	
21		22	RECEIVE CLK/TTY Rx DATA RETURN
23	TTY Rx DATA		
25	TTY Tx DATA	24	TTY Tx DATA RETURN
		26	GND

The 60-pin double-sided edge connector labeled P2 in Figure 5-1 allows access to various test points on the SBC-80/10 (see Table 5-4). The following wire-wrap connectors will attach to P2:

CDC VPB01B30A00A2,
TI H311130 and
AMP PE5-14559

TABLE 5-4. PIN ASSIGNMENTS FOR CONNECTOR P2
(Auxilliary Connector)

SIGNAL NAME	PIN ASSIGNMENT	COMMENT
OSC	P2 - 28	TEST POINT
RAM 3C00 ENABLE/	P2 - 30	
RAM 3D00 ENABLE/	P2 - 32	
RAM 3E00 ENABLE/	P2 - 34	
RAM 3F00 ENABLE/	P2 - 36	
OSC INH/	P2 - 40	
DATA BUS INH/	P3 - 42	
BAUD RATE CLK TTY	P2 - 44	
COUNT 1 ENABLE 1	P2 - 46	
BAUD RATE CLK	P2 - 50	
COUNT 2 ENABLE/	P2 - 52	
TIME OUT ENABLE/	P2 - 54	
B & C CLK SET/	P2 - 55	
STATUS STROBE	P2 - 56	
RDY IN INH/	P2 - 57	
BAUD RATE CLEAR/	P2 - 58	
OSC/2	P2 - 60	TEST POINT

5.2 EXTERNAL SBC 80/10 SYSTEM BUS SUMMARY

A significant measure of the SBC-80/10's power and flexibility can be attributed to its external system bus. In expanded systems, the external bus structure allows for master-slave relationships between the various system modules. The bus includes its own clock (BCLK/) which is derived independently from the processor clock. Actual transfers via the bus, however, proceed asynchronously with respect to the bus clock. Thus, the transfer speed is dependent on the transmitting and receiving devices only. Once a module has gained control of the bus by activating the BPRN input to the SBC-80/10, single or multiple read/write transfers can proceed at a maximum rate of 5 million data bytes per second. The 16 system address lines allow the SBC-80/10 to support up to 65,536 bytes of storage. The signal lines on the external system bus are defined as follows:

BCLK/	Bus clock; used to synchronize bus control circuits on all master modules. BCLK/ has a period of ~110 nanoseconds (9.216 MHz frequency), 30% - 70% duty cycle. BCLK/ may be slowed, stopped or single stepped, if desired (see Section 4.4).
INIT/	Initialization signal; resets the entire system to a known internal state.
BPRN	Bus priority input signal; indicates to the SBC-80/10 that a higher priority master module is requesting use of the system bus. BPRN suspends the processing activity and drivers of the SBC-80/10.
BUSY/	Bus busy signal; indicates that the bus is currently in use. BUSY/ prevents all other master modules from gaining control of the bus. BUSY/ is driven by the HLDA/ output from the SBC-80/10 in response to a BPRN input. It indicates that the bus is available.
MRDC/	Memory read command; indicates that the address of a memory location has been placed on the system address lines and specifies that the contents of the addressed location are to be read and placed on the system data bus.

MWTC/	Memory write command; indicates that the address of a memory location has been placed on the system address lines and that a data word has been placed on the system data bus. MWTC/ specifies that the data word is to be written into the addressed memory location.
IORC/	I/O read command; indicates that the address of an input port has been placed on the system address bus and that the data at that input port is to be read and placed on the system data bus.
IOWC/	I/O write command; indicates that the address of an output port has been placed on the system address bus and that the contents of the system data bus are to be output to the addressed port.
XACK/	Transfer acknowledge signal; the required response of an external memory location or I/O port which indicates that the specified read/write operation has been completed. That is, data has been placed on, or accepted from, the system data bus lines.
AACK/	Advance acknowledge signal; used with 8080 CPU-based systems. AACK/ is an advance acknowledge, in response to a memory read command, that allows the memory to complete the access without requiring the CPU to wait.

TABLE 5-5. PIN ASSIGNMENTS FOR CONNECTOR P1
(External System Bus)

	(COMPONENT SIDE)			(CIRCUIT SIDE)		
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Signal GND
	3	VCC	+ 5VDC	4	VCC	+ 5VDC
	5	VCC	+ 5VDC	6	VCC	+ 5VDC
	7	VDD	+12VDC	8	VDD	+12VDC
	9	VBB	- 5VDC	10	VBB	- 5VDC
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15	BPRN	Bus Pri. In	16	[1]	
	17	BUSY/	Bus Busy	18	[1]	
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknow	24	[1]	
SPARES	25	AACK/	Special	26		
	27			28		
	29			30		
	31	CCLK/	Constant Clock	32		
	33	[1]		34		
INTERRUPTS	35	[1]		36	[1]	
	37	[1]		38	[1]	
	39	[1]		40	[1]	
	41	[1]		42	INTR1/	Interrupt request
ADDRESS	43	ADRE/	Address Bus	44	ADRF/	Address Bus
	45	ADRC/		46	ADRD/	
	47	ADRA/		48	ADRB/	
	49	ADR8/		50	ADR9/	
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
DATA	59	[1]	Data Bus	60	[1]	Data Bus
	61	[1]		62	[1]	
	63	[1]		64	[1]	
	65	[1]		66	[1]	
	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77	VBB[1]	-10VDC	78	VBB[1]	-10VDC
	79	VAA	-12VDC	80	VAA	-12VDC
	81	VCC	+ 5VDC	82	VCC	+ 5VDC
	83	VCC	+ 5VDC	84	VCC	+ 5VDC
	85	GND	Signal GND	86	GND	Signal GND

[1] Used by Intellec® MDS Bus.

CCLK/	Constant clock; provides a clock signal of constant frequency (9.216 MHz) for use by option memory and I/O expansion boards. CCLK/ coincides with BCLK/ and has a period of ~110 nanoseconds. 30% - 70% duty cycle (see Section 4.4).
INTR1/	Externally generated interrupt request.
ADRO/-ADRF/	16 Address lines: used to transmit the address of the memory location or I/O port to be accessed. ADRF/ is the most significant bit.
DATO/-DAT7/	Bi-directional data lines; used to transmit/receive information to/from a memory location or I/O port. DAT7/ is the most significant bit.

5.3 RS232C CABLING

When the Serial I/O Interface is configured as an RS232C interface, the J3 edge connector can be cabled such that a RS232C pin-compatible connector is presented to the user's terminal or modem. A 26-pin mating connector, 3M 3462-0001, should be attached to the J3 edge connector on the SBC-80/10 and to a 25-wire flat cable, 3M 3349/25. The flat cable is, in turn, attached to the RS232C pin-compatible connector, 3M 3483-1000. Table 5-6 equates the J3 edge connector pins with the associated RS232-compatible pins on the 3M 3483-1000 connector.

Note: Using this 3M cable assemble, the RS232C connector pin-outs are MDS compatible. That is, if the SBC 80/10 is set up to drive a TTY, an MDS modified TTY can be used directly with the SBC 80/10.

TABLE 5-6. J3/RS232C CONNECTOR PIN CORRESPONDENCE

SIGNAL NAME	J3 CONNECTOR PIN NO.	RS232C CONNECTOR PIN NO.
CHASSIS GND	1	1
	2	14
TRANSMITTED DATA	3	2
	4	15
RECEIVED DATA	5	3
TTY RD CONTROL	6	16
REQ TO SEND	7	4
	8	17
CLEAR TO SEND	9	5
	10	18
DATA SET READY	11	6
	12	19
GND	13	7
Tx CLK/DATA TERMINAL RDY	14	20
DATA CARRIER RETURN	15	8
TTY RD CONTROL RETURN	16	21
	17	9
	18	22
	19	10
	20	23
	21	11
RECEIVE CLK/TTY Rx DATA RETURN	22	24
TTY Rx DATA	23	12
TTY Tx DATA RETURN	24	25
TTY Tx DATA	25	13

5.4 TELETYPE MODIFICATIONS

The ASR-33 Teletypewriter must be modified for use with the SBC 80/10 Boards. Appendix B is a procedure for modifying the ASR-33 Teletypewriter.

CHAPTER 6

INTERFACING TO MULTIBUS MASTERS

The SBC 80/10's system bus structure permits interfacing to one other Multibus-Compatible Master module. This interface is accomplished using the serial priority scheme as shown in figure 6-1 using the Intel SBC 604 Cardcage/Backplane. The SBC 80/10 does not provide the Bus Priority Request Out (BPRO/) signal and therefore, the SBC 80/10 can only be used with one other Multibus master. For these configurations, the SBC 80/10 must always have lower priority than the other Multibus master and a wire must be added from the master's BREQ/ pin (pin 18) to the SBC 80/10 BPRN pin (pin 15). In the configuration shown in figure 6-1 the SBC 80/10 acquires control of the Multibus whenever BREQ/ generated by the Diskette Controller is in the high state. This occurs whenever the Diskette Controller is not using the Multibus. Similarly BREQ/ is driven to the low state when the Diskette Controller acquires control of the Multibus disabling the SBC 80/10 from accessing the Multibus.

For a detailed description of Multibus interfacing refer to the Intel Multibus Interfacing Application Note (AP-28).

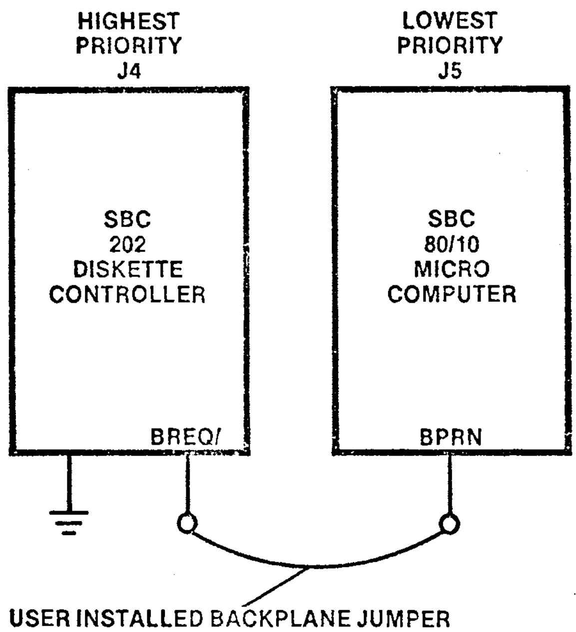


Figure 6-1. Serial Priority Configuration with another Multibus Master

CHAPTER 7

SPECIFICATIONS

7.1 DC POWER REQUIREMENTS

DC Power Requirements are given in Table 7-1.

7.2 AC CHARACTERISTICS

Detailed timing diagrams for memory, I/O and Bus exchange operations are provided in Figures 7-1 through 7-3. Tables 7-3 and 7-4 provide design limits for SBC-80/10 outputs and requirements for its inputs. These values are theoretical limits based on a "worst-on-worst" case analysis using vendor information and approximations where necessary. Approximations include establishing non-zero propagation delay minimums and extended delays if capacitive loading exceeds vendor ratings. In all such cases, approximations are conservative (e.g., 2 ns minimum for standard TTL, 4 ns minimum for tri-state turn-offs or turn-ons). Rise and fall times are assumed to be zero unless a three-state high impedance state or open collector circuit is involved.

7.3 DC CHARACTERISTICS

DC Characteristics are given in Table 7-4.

7.4 ENVIRONMENT

Temperature extremes can cause instability, or result in permanent damage to the circuits on the module. Ambient temperature must, therefore, be maintained within the limits of $0^{\circ}C$ to $55^{\circ}C$. Exercise caution in locating the module, giving particular attention to radiant and conducive sources of heat. Remember that the module itself, when installed, will contribute some heat to the environment. Maintain an adequate clearance to permit the convective dissipation of heat from the elements on the card.

Relative humidity should not exceed 90%, non-condensing.

7.5 BOARD OUTLINE

See Figure 7-4.

7.6 COMPATIBLE CONNECTORS

Table 7-5 lists compatible connectors which mate to the SBC-80/10 and SBC 80/10A PC edge connectors.

TABLE 7-1. DC POWER REQUIREMENTS

—	Without EPROM[1]	With 8708 EPROM[2]	With 2758 or 2716 EPROM[3]
V_{CC} +5V ± 5%	I_{CC} = 2.9A	4.0A	4.36A
V_{DD} +12V ± 5%	I_{DD} = 150mA	400mA	150.mA
V_{BB} -5V ± 5%	I_{BB} = 2mA[4]	200mA	2mA[4]
V_{AA} -12V ± 5%	I_{AA} = 175mA	175mA	175mA

1. Does not include power required for optional ROM/EPROM, I/O drivers or I/O terminators.
2. With four Intel 8708 EPROMs and 220Ω/330Ω terminators installed for 48 input lines; all terminator inputs low.
3. With four Intel 2758 or 2716 EPROMs and 220Ω/330Ω terminators installed for 48 input ports; all terminator inputs low.
4. Required for RS232C drivers.

TABLE 7-2. AC CHARACTERISTICS WITH BUS EXCHANGE

PARAMETER	OVERALL		WITH BUS EXCHANGE READ		WITH BUS EXCHANGE MEMORY WRITE		DESCRIPTION	REMARKS
	MIN (ns)	MAX (ns)	MIN (ns)	MAX (ns)	MIN (ns)	MAX (ns)		
t_{AS}	82		82		658		Address Setup Time to Command	
t_{AH}	61		0		61		Address Hold Time	
t_{DS}	140		—		140		Data Setup Time to Command	
t_{DH}	61		0		61		Data Hold Time	
$t_{ACKØ}$			68	191			First ACK Sampling Point of Current Cycle	Generates 0 Wait States
t_{ACK1}			551	684	-60	132	Second ACK Sampling Point of Current Cycle	Generates 1 Wait State
t_{ACK2}			1034	1174	423	625	Third ACK Sampling Point of Current Cycle	Generates 2 Wait States
t_{CY}	483	493					ACK & BPRN Sample Cycle Time	
t_{WC}			596	796	1412	1516	Command Width	Read, 0 Wait States Write, 2 Wait States
t_{ACC}				344			Read Access Time	▷1
t_{8KD}				68		-60	Advanced ACK Response Time for Minimum Delay	▷1
t_{8KO}	0	100	0	100	0	100	Advanced ACK Turn Off Delay	▷1
t_{XKD}	0		0				XACK Delay From Valid Data or Write	
t_{XKO}	0	100	0	100	0	100	XACK Turn Off Delay	
t_{DBS}		3500					Bus Sample to Exchange Initiation	▷1 Assume HOLD/ becomes active prior to DAD instruction
t_{BS}	0	493					BPRN Sampling Point Delay	
t_{DBY}	358	700					Bus Busy Turn On Delay	

▷1 Memory and I/O access occurs with no wait states.

TABLE 7-3. AC CHARACTERISTICS WITH CONTINUOUS BUS CONTROL

PARAMETER	OVERALL		CONTINUOUS BUS CONTROL READ		CONTINUOUS BUS CONTROL MEMORY WRITE		DESCRIPTION	REMARKS
	MIN (ns)	MAX (ns)	MIN (ns)	MAX (ns)	MIN (ns)	MAX (ns)		
t_{AS}	82		82		658		Address Setup Time to Command	
t_{AH}	79		0		79		Address Hold Time	
t_{DS}	140		–		140		Data Setup Time to Command	
t_{DH}	79		0		79		Data Hold Time	
$t_{ACKØ}$			68	191			First ACK Sampling Point of Current Cycle	Generates 0 Wait States
t_{ACK1}			551	684	-60	132	Second ACK Sampling Point of Current Cycle	Generates 1 Wait State
t_{ACK2}			1034	1177	423	625	Third ACK Sampling Point of Current Cycle	Generates 2 Wait States
t_{CY}	483	493					ACK & BPRN Sample Cycle Time	
t_{SEP}	259		613		259	[2]	Command Separation	
t_{WC}			596	796	1412	1516	Command Width	Read, 0 Wait States Write, 2 Wait States
t_{ACC}	344			344			Read Access Time	[1]
t_{8KD}				68		-60	Advanced ACK Response Time for Minimum Delay	[1]
t_{8KO}	0	100	0	100	0	100	Advanced ACK Turn Off Delay	
t_{XKD}	0		0				XACK Delay From Valid Data or Write	
t_{XKO}	0	100	0	100	0	100	XACK Turn Off Delay	
t_{BCY}	107	110					Bus Clock Cycle Time	80/10 Generator
t_{BW}	25	85					Bus Clock Low or High Periods	80/10 Generator
t_{INT}	3000						Initialization Width	After all voltages have stabilized

[1] MAX assumes no acknowledge delays.

[2] Write Command to next Read Command separation.

TABLE 7-4. DC CHARACTERISTICS

SIGNALS	SYMBOL	PARAMETER DESCRIPTION	TEST CONDITIONS	MIN	MAX	UNITS
ADRØ/-ADRF/	V_{OL}	Output Low Voltage	I_{OL} = 50 mA		0.6	V
ADDRESS	V_{OH}	Output High Voltage	I_{OH} = -10 mA	2.4		V
	V_{IL}	Input Low Voltage			0.95	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.45		-0.25	mA
	I_{IH}	Input Current at High V	V_{IN} = 5.25V		10	µA
	*C_L	Capacitive Load			18	pF
MROC/,MWTC/	V_{OL}	Output Low Voltage	I_{OL} = 32 mA		0.4	V
IORC/,IOWC/	V_{OH}	Output High Voltage	I_{OH} = -5.2 mA	2.4		V
	I_{LH}	Output Leakage High	V_O = 2.4		40	µA
	I_{LL}	Output Leakage Low	V_O = 0.4		-40	µA
	*C_L	Capacitive Load			15	pF
DATØ/-DAT7/	V_{OL}	Output Low Voltage	I_{OL} = 50 mA		0.6	V
	V_{OH}	Output High Voltage	I_{OH} = -10 mA	2.4		V
	V_{IL}	Input Low Voltage			0.95	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.45		-0.25	mA
	I_{LH}	Output Leakage High	V_O = 5.25		100	µA
	I_{LL}	Output Leakage Low	V_O = 0.45		100	µA
	*C_L	Capacitive Load			18	pF
INT1/	V_{IL}	Input Low Voltage			0.8	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.4V		-2.2	mA
	I_{IH}	Input Current at High V	V_{IN} = 5.5V		1	mA
	*C_L	Capacitive Load			18	pF
BPRN,XACK	V_{IL}	Input Low Voltage			0.8	V
AACK	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.5		-2.6	mA
	I_{IH}	Input Current at High V	V_{IN} = 2.7V		0.30	mA
	*C_L	Capacitive Load			18	pF
BUSY/	V_{OL}	Output Low Voltage	I_{OL} = 25 mA		0.4	V
OPEN COLLECTOR	*C_L	Capacitive Load			20	pF
INT	V_{OL}	Output Low Voltage	I_{OL} = 32 mA		0.6	V
(SYSTEM RESET)	V_{OH}	Output High Voltage	OPEN COLLECTOR			
	V_{IL}	Input Low Voltage			0.7	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IH}	Input Current at High V	V_{IN} = 5.5		0.2	mA
	I_{IL}	Input Current at Low V	V_{IN} = 0.3		-0.9	mA
	*C_L	Capacitive Load			38	pF
BCLK/ + CCLK/	V_{OL}	Output Low Voltage	I_{OL} = 20 mA		0.5	V
	V_{OH}	Output High Voltage	I_{OH} = -1 mA	2.7		V
	*C_L	Capacitive Load			18	pF

*Capacitive values are approximations only.

TABLE 7-4. DC CHARACTERISTICS (Continued)

SIGNALS	SYMBOL	PARAMETER DESCRIPTION	TEST CONDITIONS	MIN	MAX	UNITS
EXT INTRØ/	V_{IL}	Input Low Voltage			0.8	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.4V	6.8		mA
	I_{IH}	Input Current at High V	V_{IN} = 5.5V		2	mA
	*C_L	Capacitive Load			18	pF
PORT E4 BIDIRECTIONAL DRIVERS	V_{OL}	Output Low Voltage	I_{OL} = 20 mA		.45	V
	V_{OH}	Output High Voltage	I_{OH} = -12 mA	2.4		V
	V_{IL}	Input Low Voltage			.95	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.45		-5.25	mA
	I_{LH}	Output Leakage High	V_O = 5.25		.30	mA
	*C_L	Capacitive Load			18	pF
8255 DRIVER/ RECEIVER	V_{OL}	Output Low Voltage	I_{OL} = 1.7 mA		.45	V
	V_{OH}	Output High Voltage	I_{OH} = -50 µA	2.4		V
	V_{IL}	Input Low Voltage			.8	V
	V_{IH}	Input High Voltage		2.0		V
	I_{IL}	Input Current at Low V	V_{IN} = 0.45		10	µA
	I_{IH}	Input Current at High V	V_{IN} = 5.0		10	µA
	*C_L	Capacitive Load			18	pF

*Capacitive values are approximations only.

TABLE 7-5. COMPATIBLE CONNECTOR HARDWARE

FUNCTION	# OF PINS	CENTERS (inches)	CONNECTOR TYPE	VENDOR	VENDOR PART #	INTEL PART #
PARALLEL I/O	25/50	0.1	FLAT CRIMP ↓	3M 3M AMP ANSLEY SAE	3415-0000 WITH EARS 3415-0001 W/O EARS 88083-1 609-5015 SD6750 SERIES	SBC-955 (CABLE ASSY.)
SERIAL I/O	13/26	0.1	FLAT CRIMP ↓	3M AMP ANSLEY SAE	3462-0001 CRIMP 88106-1 609-2615 SD6726 SERIES	SBC-956 (CABLE ASSY.)
PARALLEL I/O	25/50	0.1	SOLDERED ↓	AMP VIKING TI	2-583485-6 3VH25/1JV5 H312125	N/A
SERIAL I/O	13/26	0.1	SOLDERED ↓	TI AMP	H312113 1-583485-5	N/A
AUXILIARY [1]	30/60	0.1	SOLDERED ↓	VIKING TI	3VH30/1JN5 H312130	N/A
BUS [1]	43/86	0.156	SOLDERED ↓	CDC MICRO PLASTICS ARCO VIKING	VPB01E43D00A1 [3] MP-0156-43-BW-4 AE443WP1 LESS EARS 2VH43/1^V5	N/A
PARALLEL I/O [1]	25/50	0.1	WIREWRAP ↓	TI VIKING CDC ITT CANNON	H311125 3VH25/1JND5 VPB01B25D00A1 [3] EC4A050A1A	N/A
SERIAL I/O [1]	13/26	0.1	WIREWRAP	TI	H311113	N/A
AUXILIARY [2] [1]	30/60	0.1	WIREWRAP ↓	CDC TI	VPB01B30A00A2 [3] H311130	MDS-980
BUS [2] [1]	43/86	0.156	WIREWRAP ↓	CDC CDC VIKING	VFB01E43D00A1 or [3] VPB01E43A00A1 2VH43/1AND5	MDS-985
SBC 201 SBC 501 SBC 508 SBC 905, etc.	50/100	0.1	SOLDER TAIL	VIKING	3VH50/1JN5	MDS-990
			SOLDER PAK (RAYCHEM)	CDC	VPB04B50E00A1E [3]	N/A

[1] Connector heights are not guaranteed to conform to OEM packaging equipment. Intel OEM and Intellec® Development System motherboards offer complete mechanical compatibility.

[2] Wirewrap pin lengths are not guaranteed to conform to OEM packaging equipment. Intel connectors and OEM and Intellec® Development System motherboards offer complete mechanical compatibility.

[3] CDC VPB01 ..., VPB02 ..., VPB04 ..., etc. are identical connectors with different electroplating thicknesses or metal surfaces.

NOTE: See next page for vendor addresses, telephone numbers and TWX numbers.

VENDORS ADDRESSES

The following information is for our customers' convenience only. Intel does not represent these vendors, guarantee availability nor continued quality of their products.

CDC CONNECTOR DIVISION
31829 W. LaTienda Drive
Westlake Village, CA 91361

213-889-3535
TWX 910-494-1224

VIKING INDUSTRIES, INC.
21001 Nordhoff Street
Chatsworth, CA 91311

213-341-4330
TWX 910-494-2094

Connector Systems
TEXAS INSTRUMENTS, INC.
34 Forest Street
Attleboro, MA 02703

617-222-2800

AMP Incorporated
P.O. Box 3608
Harrisburg, PA 17105

717-564-0100
TWX 510-657-4110

T & B/ANSLEY
Subsidiary of Thomas & Betts Corporation
3208 Humbolt Street
Los Angeles, CA 90031

213-223-2331
TWX 910-321-3938

STANFORD APPLIED ENGINEERING, INC. (SAE)
340 Martin Avenue
Santa Clara, CA 95050

408-243-9200
TWX 910-338-0132

3M Connectors
Electronic Products Division, Bldg. 223-4E
3M COMPANY
3M Center
St. Paul, MN 55101

612-733-1110

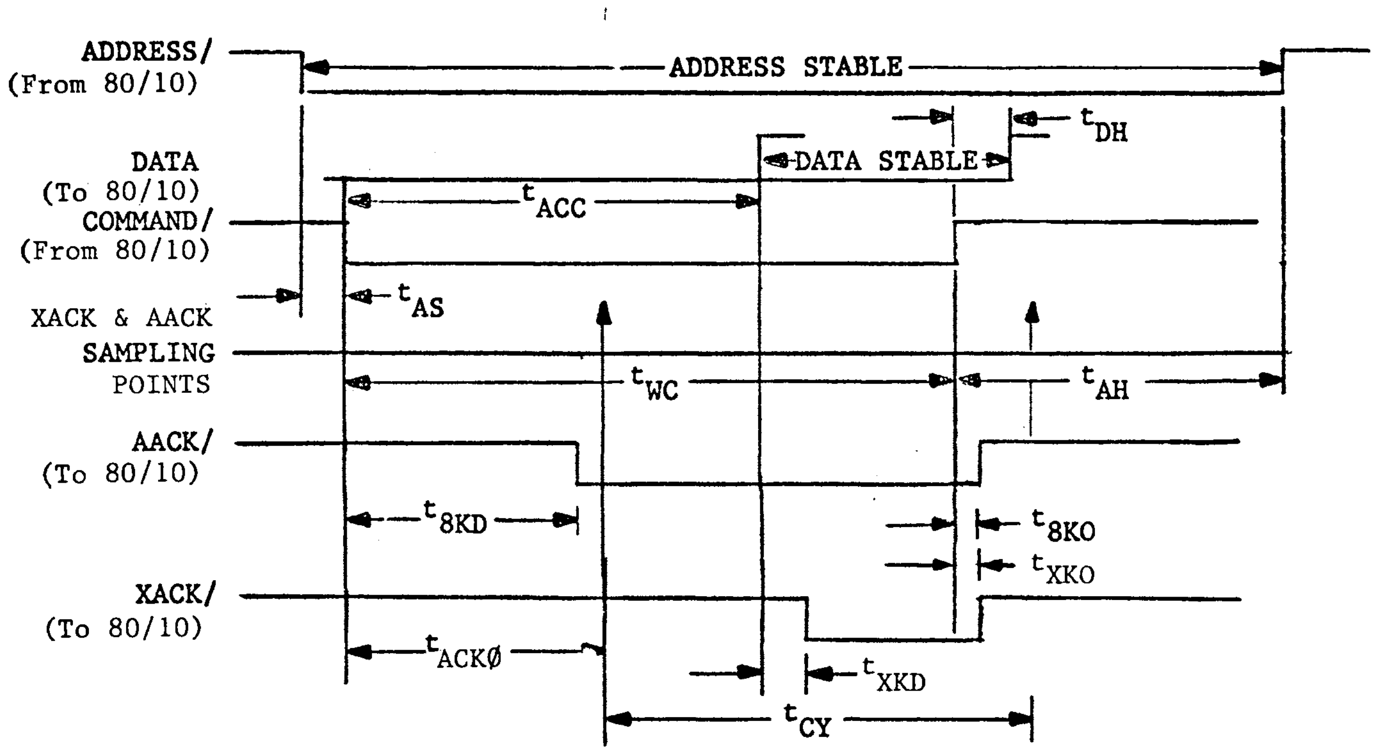


*FIGURE 7-1. MEMORY AND I/O READ TIMING (CONTINUOUS BUS CONTROL)

*NOT DRAWN TO SCALE.

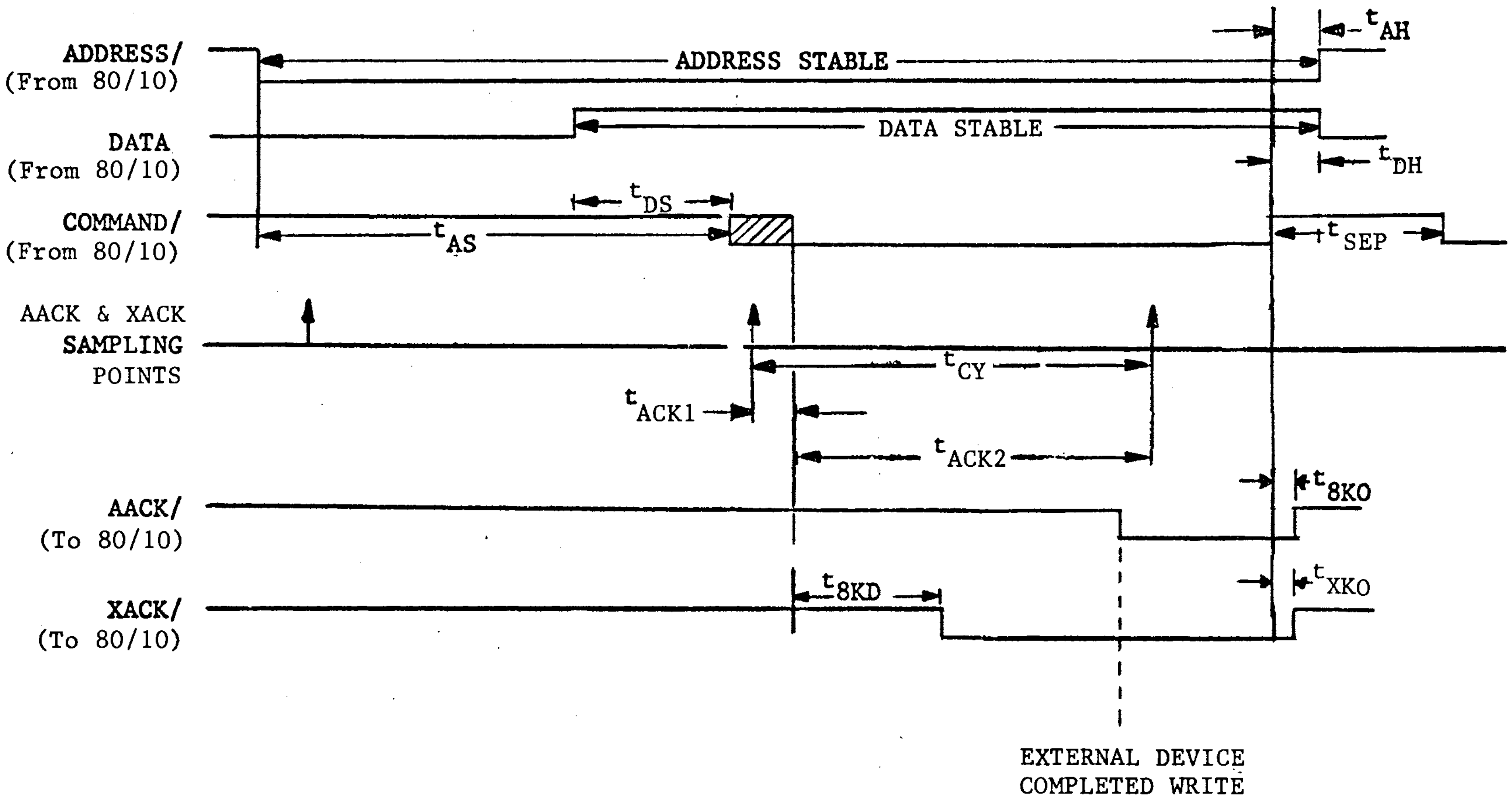


*FIGURE 7-2. MEMORY AND I/O WRITE TIMING (CONTINUOUS BUS CONTROL)

*NOT DRAWN TO SCALE.

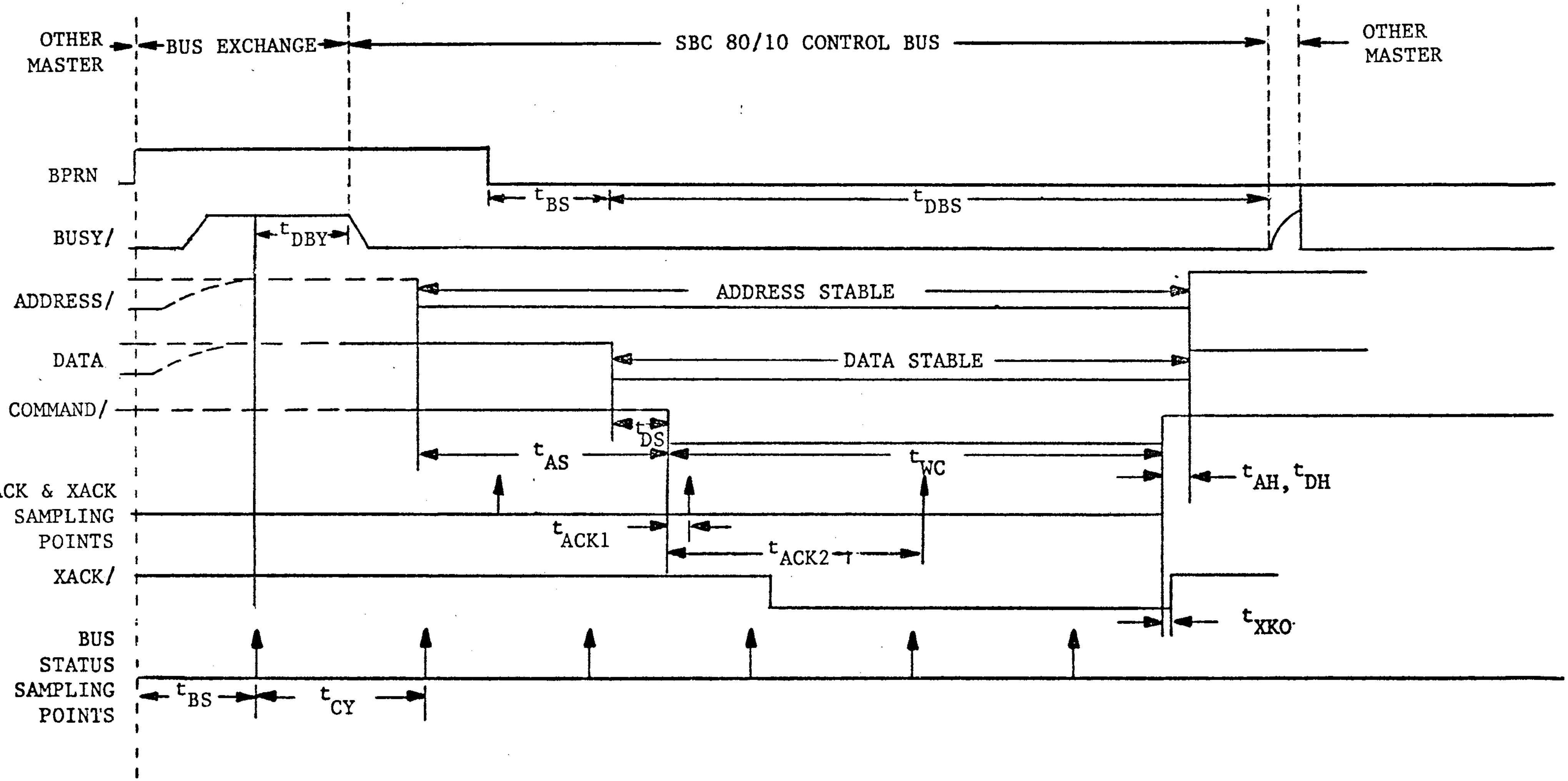


*FIGURE 7-3. BUS EXCHANGE (WRITE)

*NOT DRAWN TO SCALE.

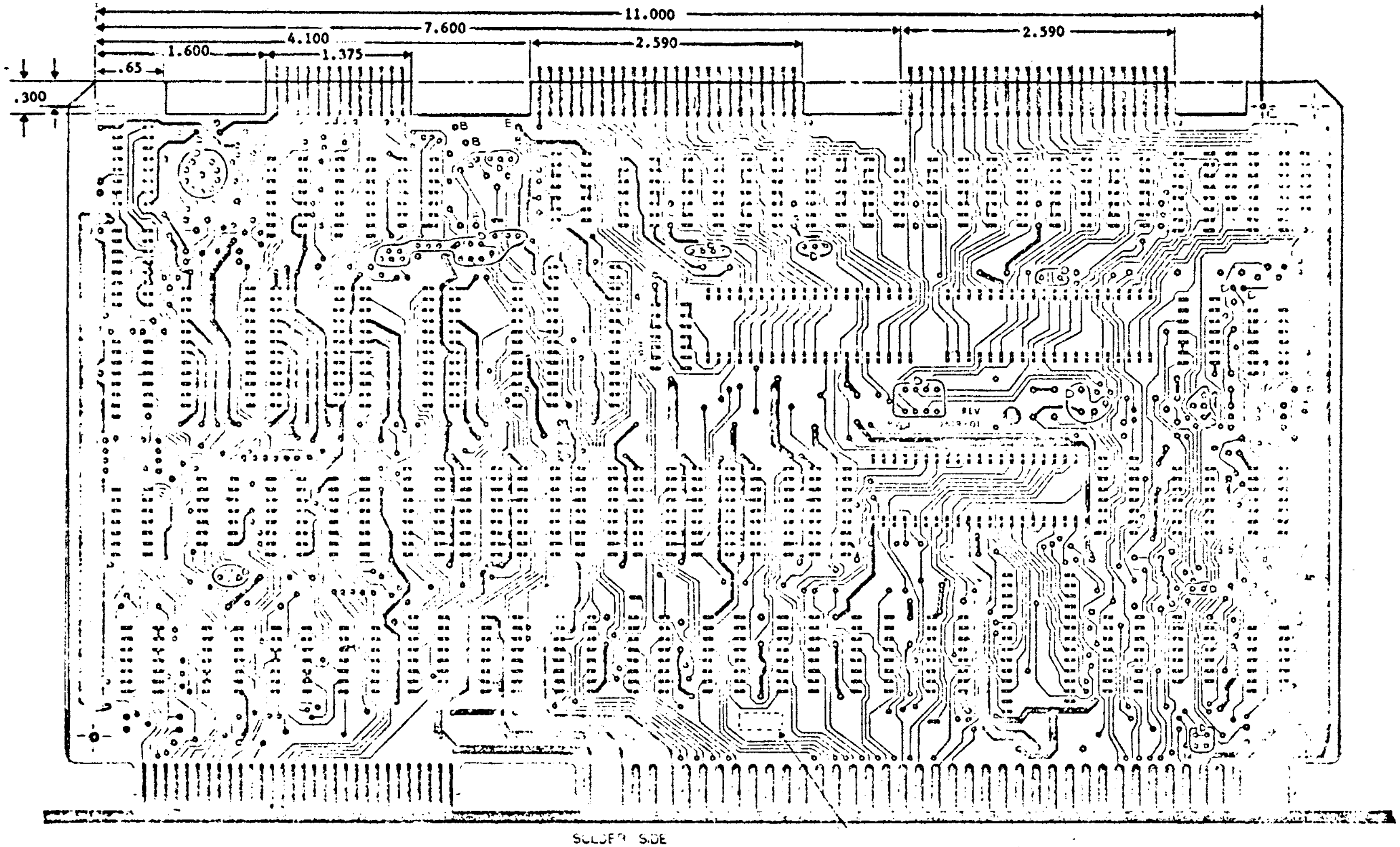


Figure 7-4. SBC 80/10 and SBC 80/10A Dimension Drawing - Page 1 of 2

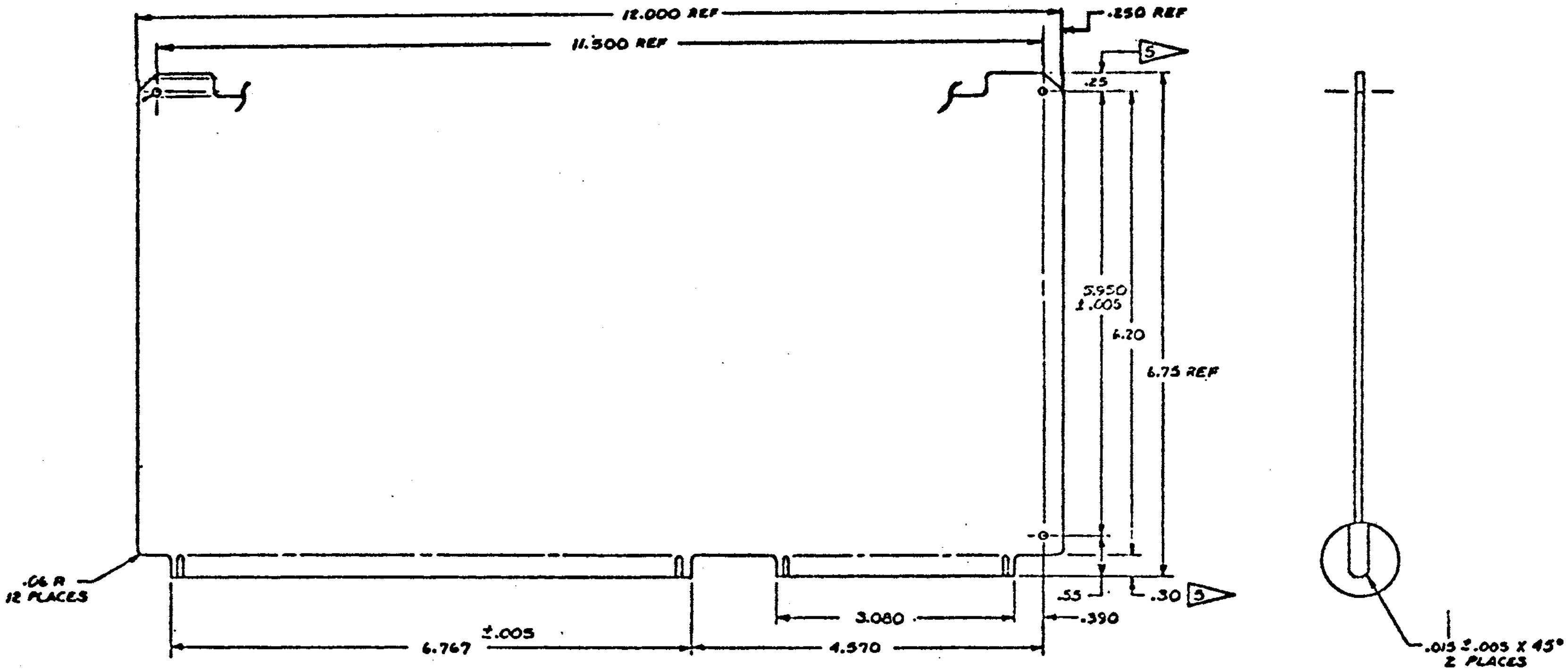


Figure 7-4. SBC 80/10 and SBC 80/10A Dimension Drawing - Page 2 of 2

NOTES:

1. MATERIAL : .062 THK, 1 OZ COPPER CLAD, NATURAL EPOXY GLASS, TYPE G10 (2OZ AFTER PLATING THRU)
2. BOARD EDGES ARE LOCATED FROM INDEX HOLES. INDEX HOLES ARE ON .050 GRID INTERSECTION AND ARE USED FOR ARTWORK REGISTRATION AND MAY BE USED AS TOOLING HOLES, PLATING OPTIONAL.
3. HOLES ARE PLATED THRU WITH COPPER WALL THICKNESS OF .0007 MINIMUM.
4. HOLE SIZES SPECIFIED ARE AFTER PLATING; ±.003 TOLERANCE
5. CONTACT FINGERS ARE OVERPLATED WITH A MINIMUM OF 50 MILLIONTHS GOLD OVER NICKEL TO DIMENSION SHOWN.
6. APPLY SOLDER MASK OVER SOLDER PLATE USING MATERIAL; MECUMASK GREEN
7.
8. DRILL FROM CIRCUIT SIDE.
9. TRACE WIDTHS MUST BE WITHIN .004 OF ARTWORK NEGATIVES.
10. APPLY SILKSCREEN ON COMPONENT SIDE, AFTER SOLDER MASK IS APPLIED, USING WHITE EPOXY INK.

APPENDIX A

SCHEMATICS

Schematic drawings for the SBC-80/80 and SBC 80/10A are provided in this appendix. Information and diagrams in this section are subject to change without notice. References should be made to schematics shipped with this module.

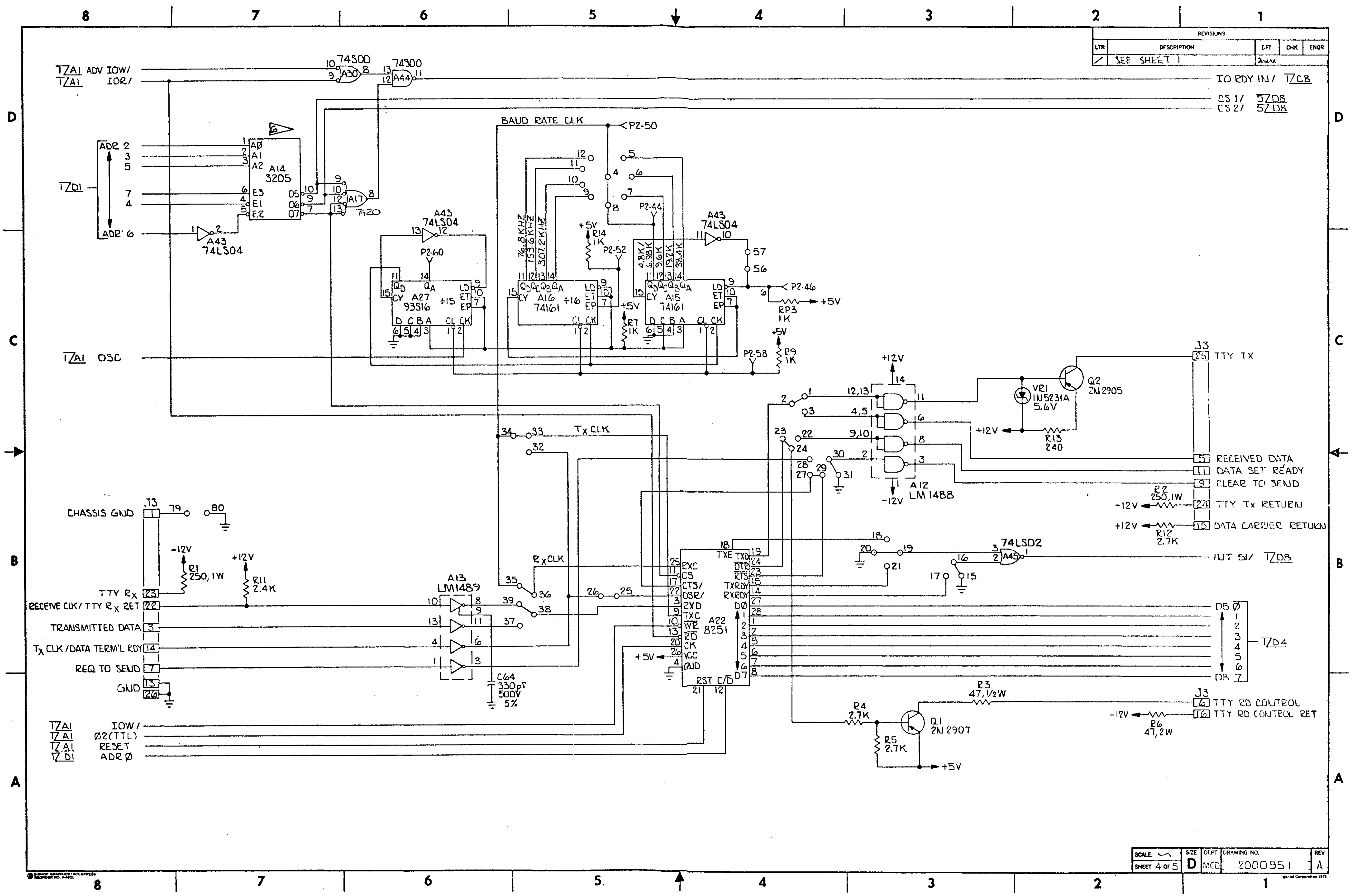


FIGURE A-2. SBC 80/10A SCHEMATIC (SHEET 4 OF 5)

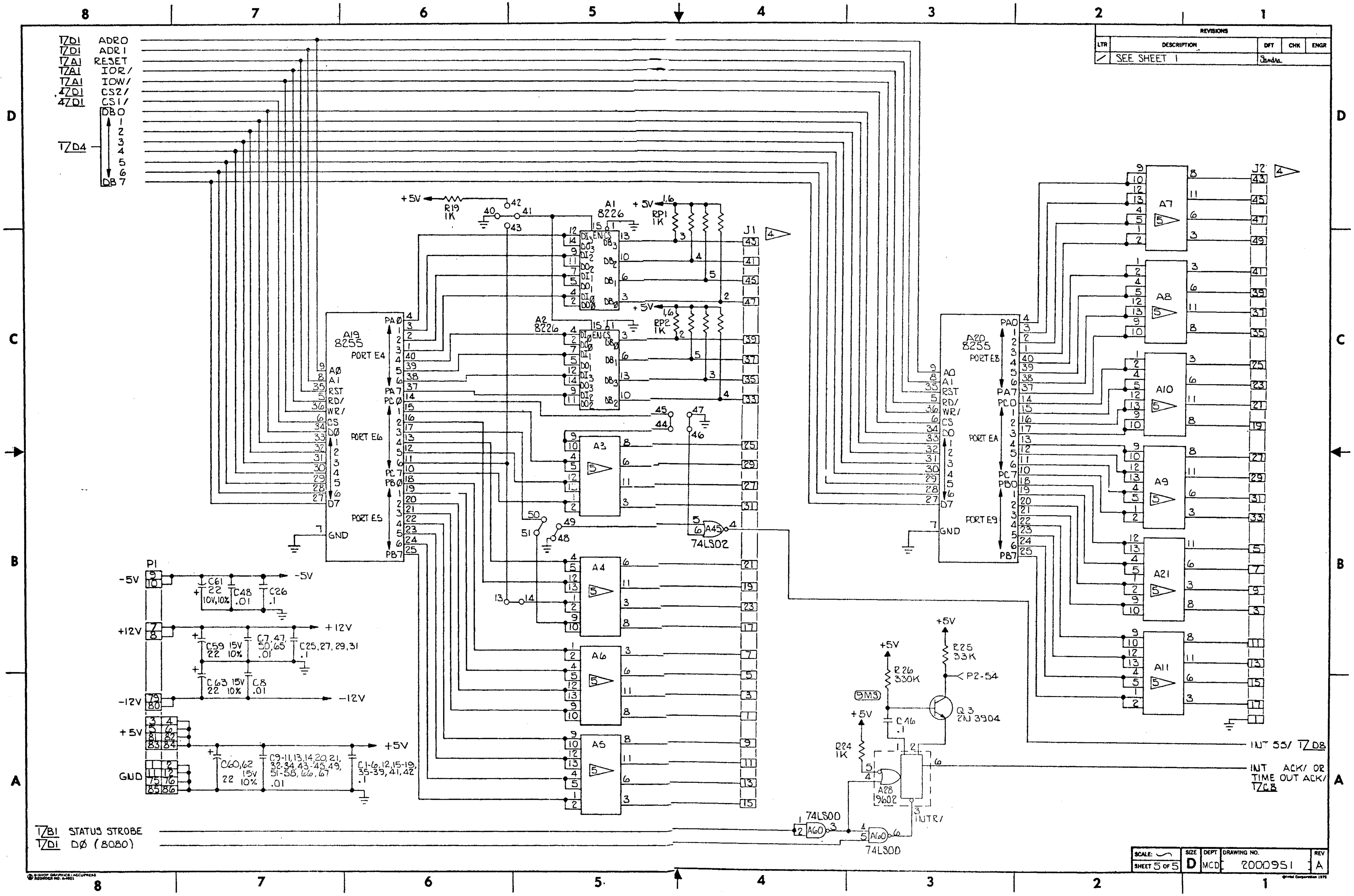


FIGURE A-2. SBC 80/10A SCHEMATIC (SHEET 5 OF 5)

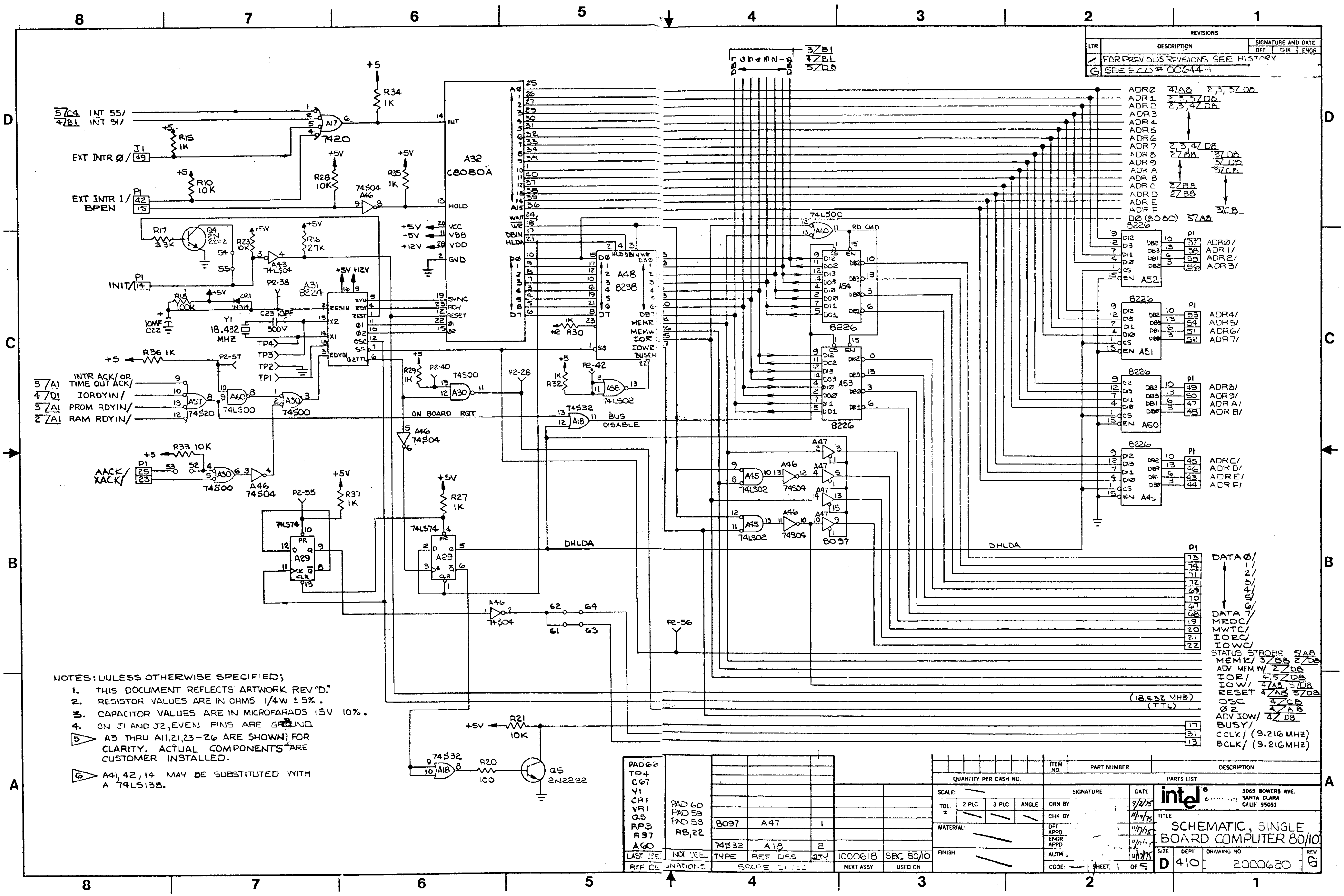


FIGURE A-1. SBC 80/10 SCHEMATIC (SHEET 1 OF 5)

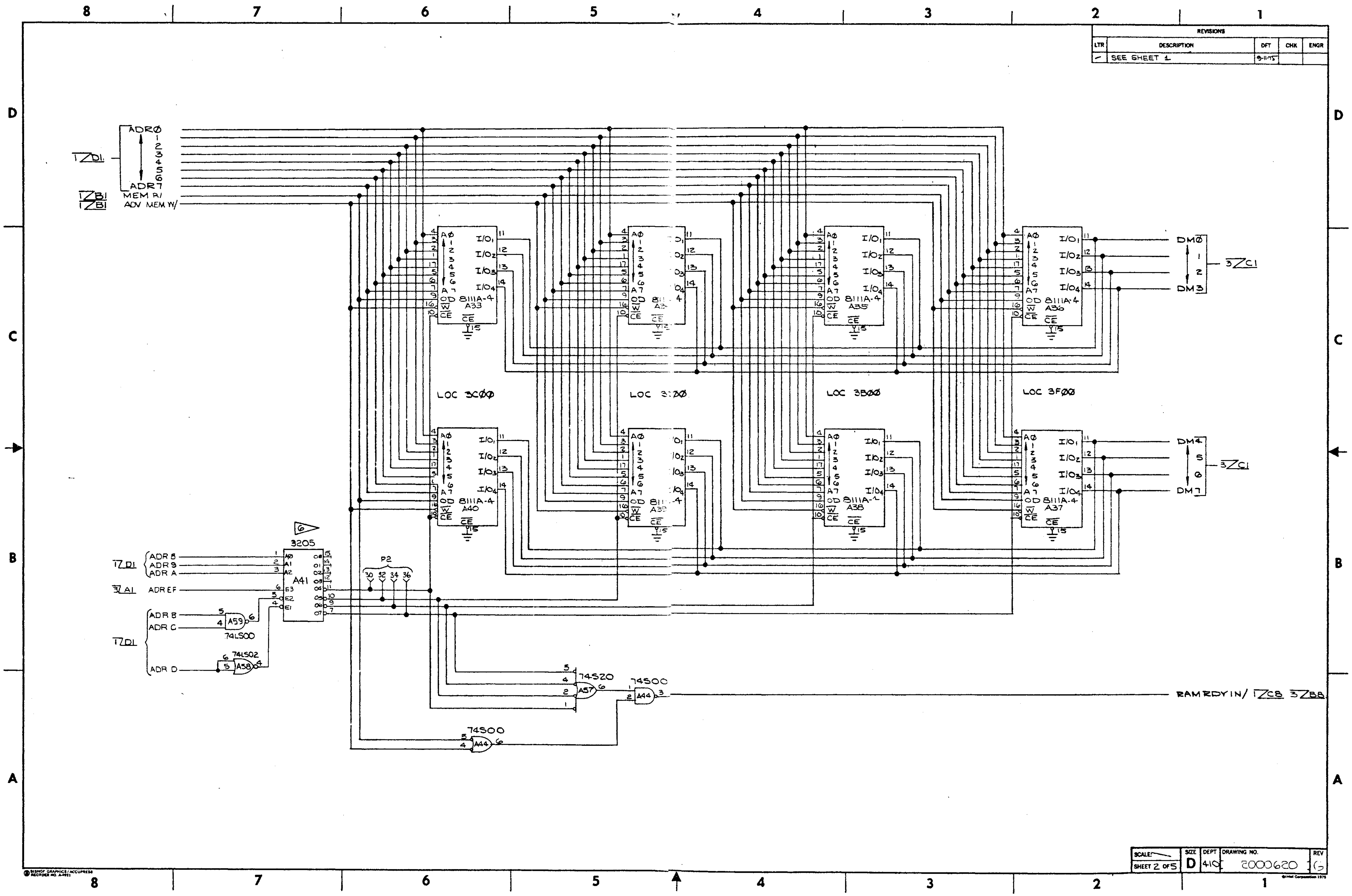


FIGURE A-1. SBC 80/10 SCHEMATIC (SHEET 2 OF 5)

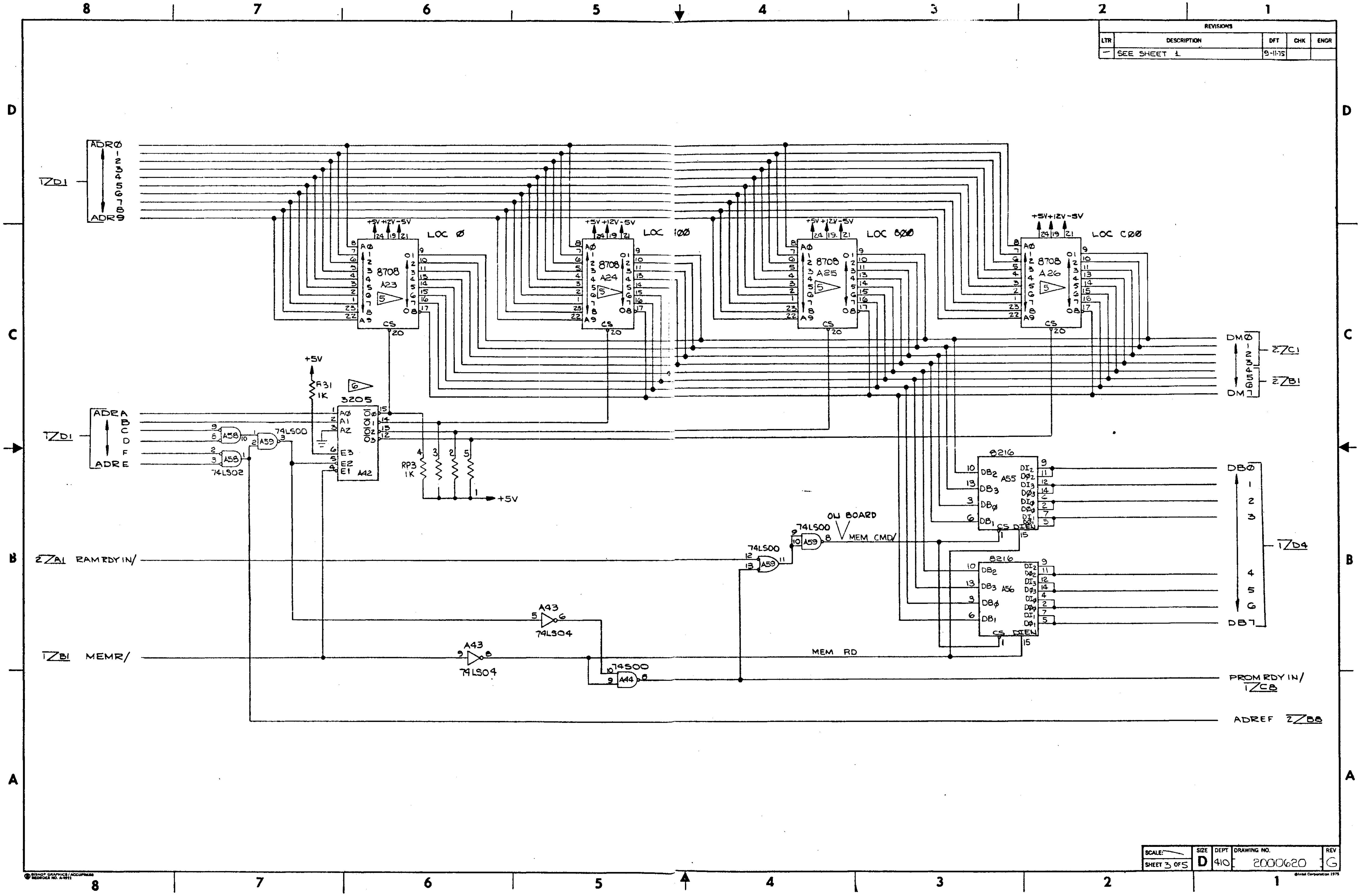


FIGURE A-1. SBC 80/10 SCHEMATIC (SHEET 3 OF 5)

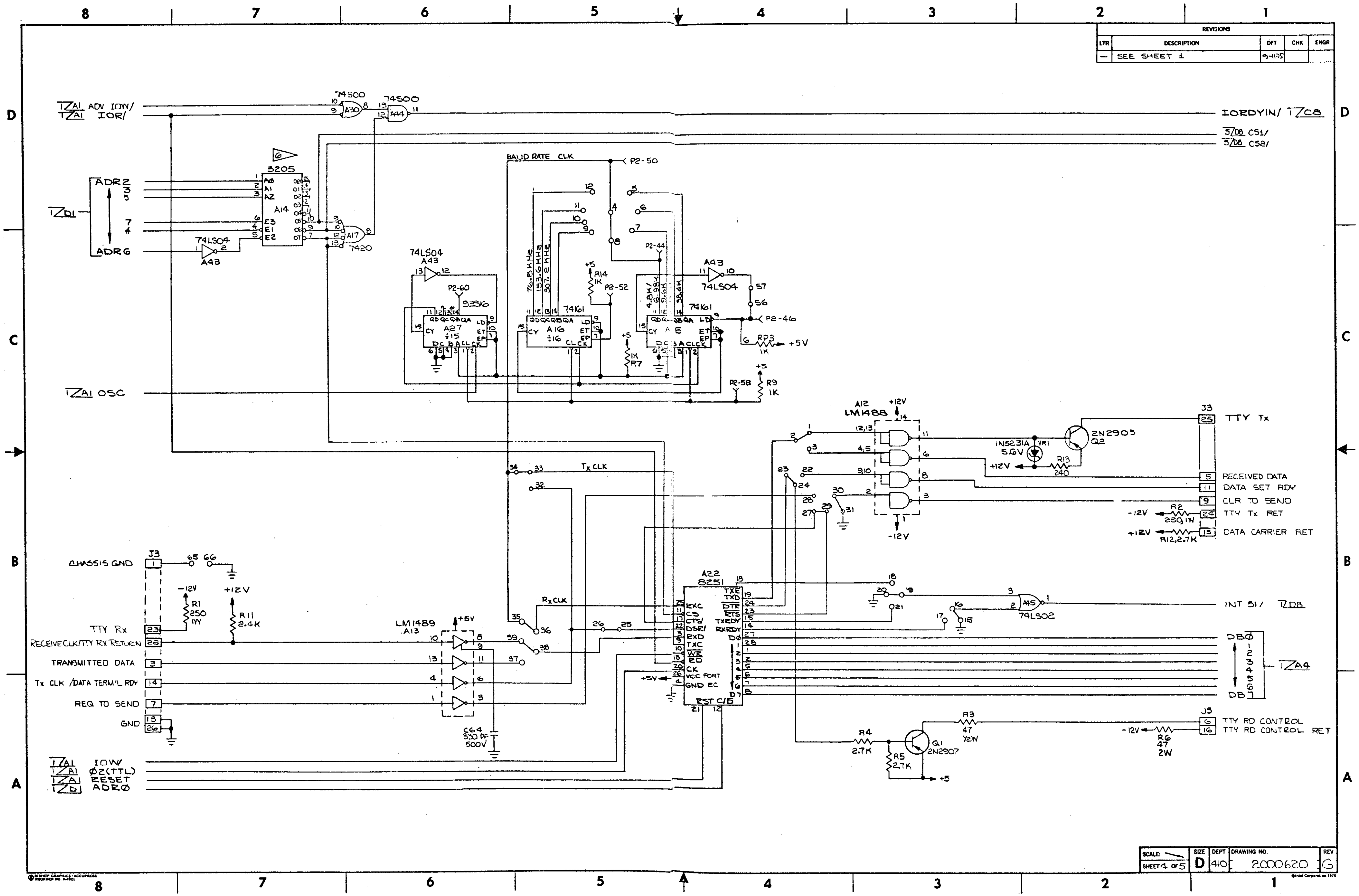


FIGURE A-1. SBC 80/10 SCHEMATIC (SHEET 4 OF 5)

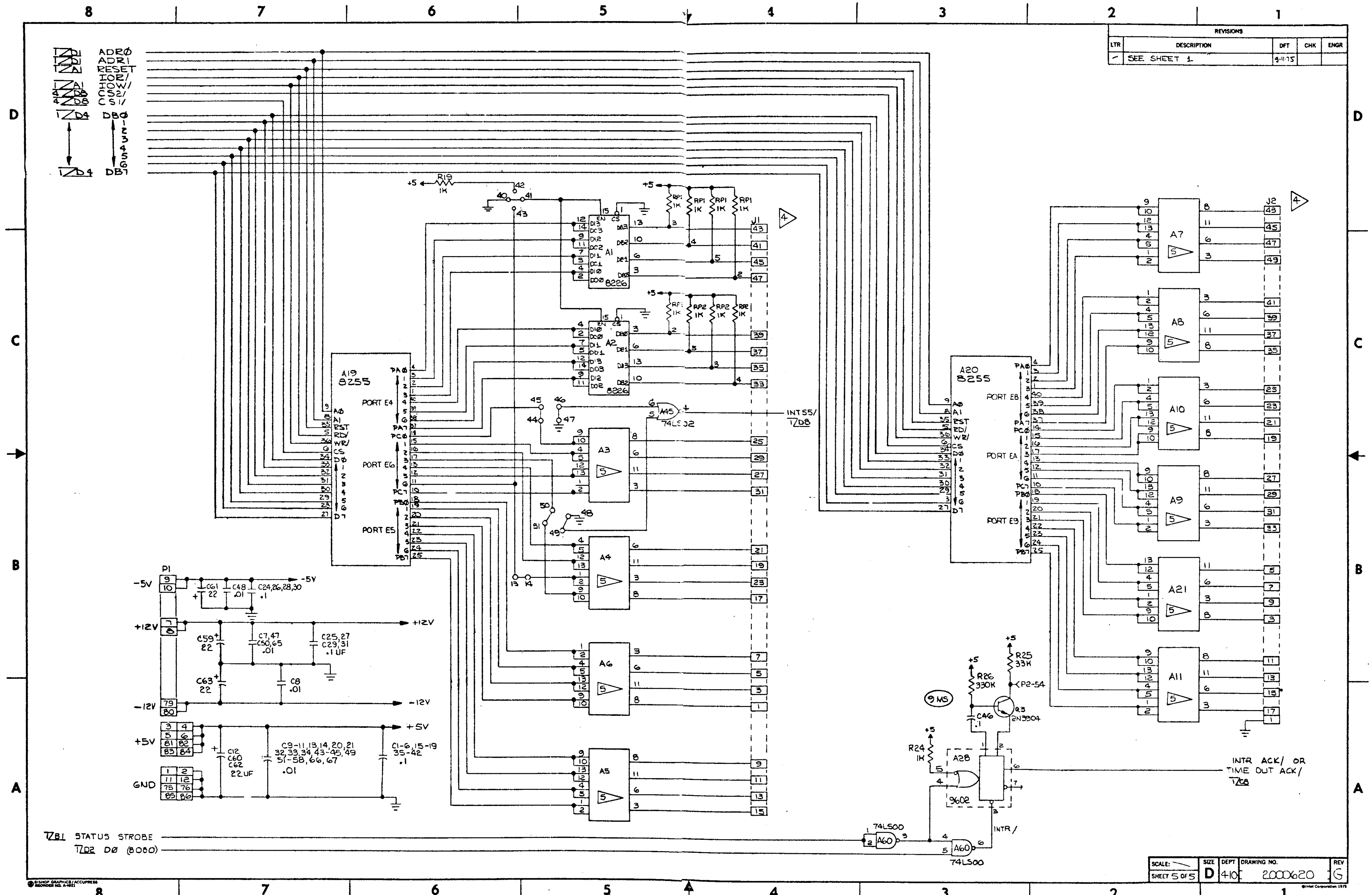


FIGURE A-1. SBC 80/10 SCHEMATIC (SHEET 5 OF 5)

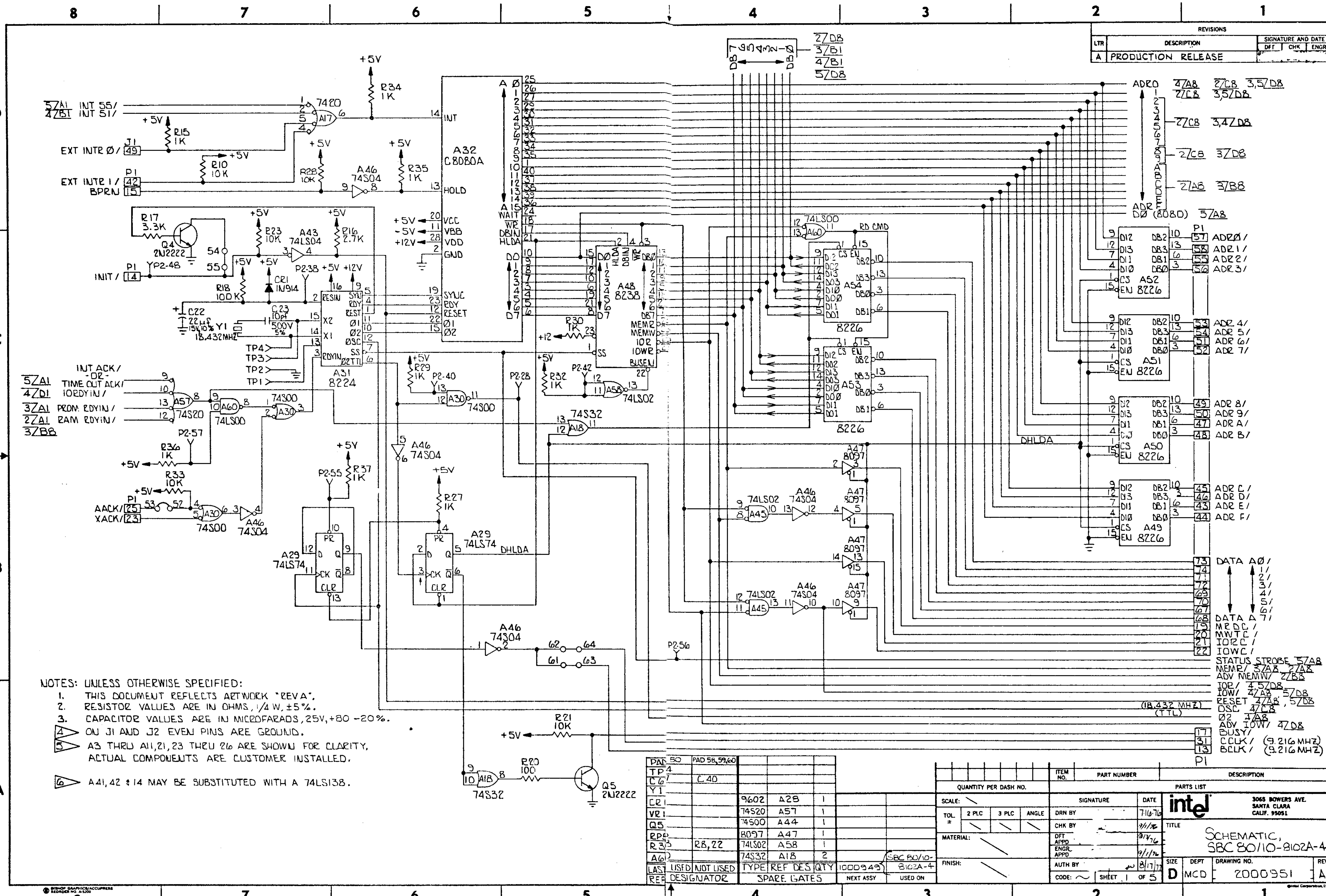


FIGURE A-2. SBC 80/10A SCHEMATIC (SHEET 1 OF 5)

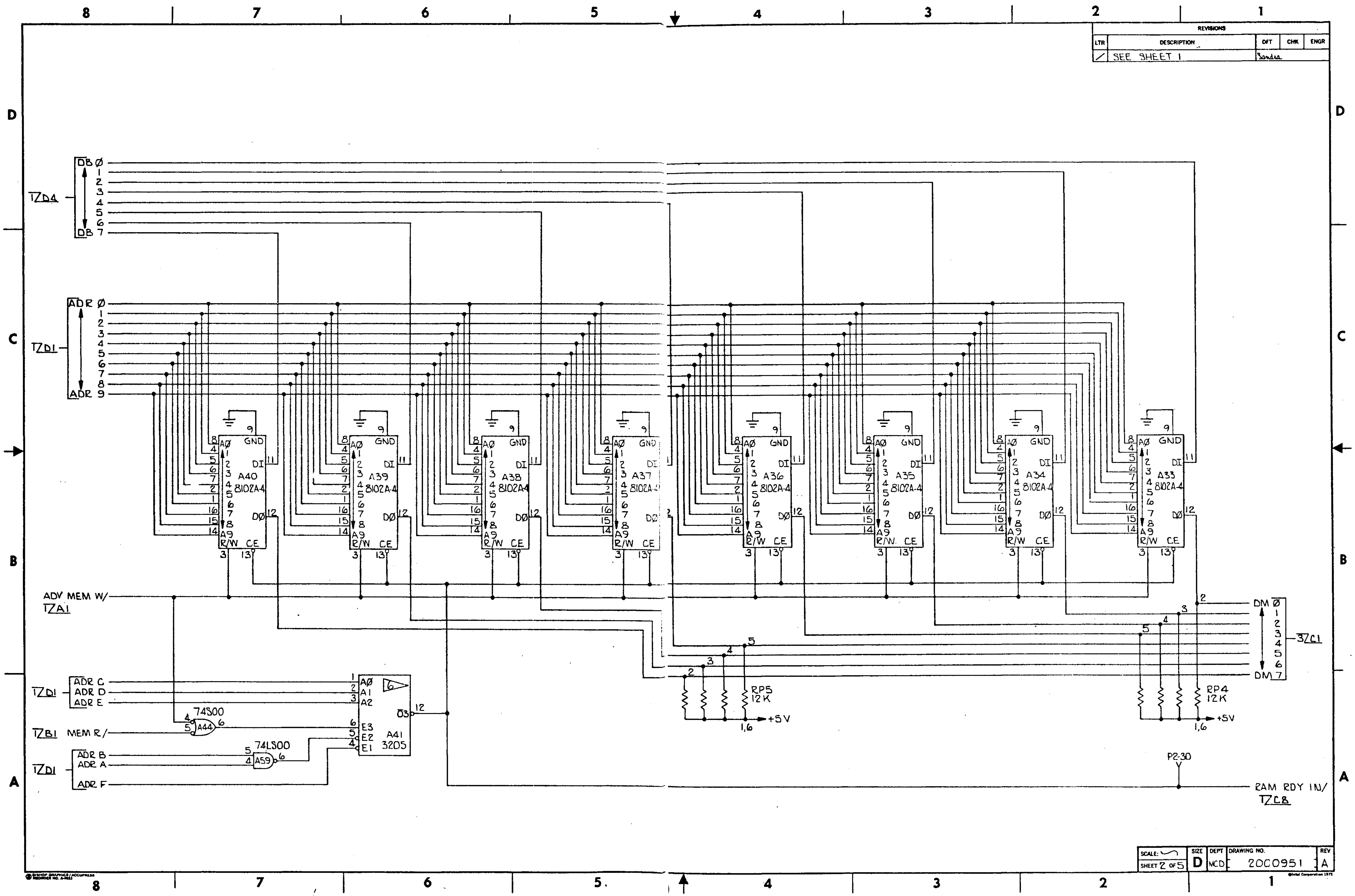
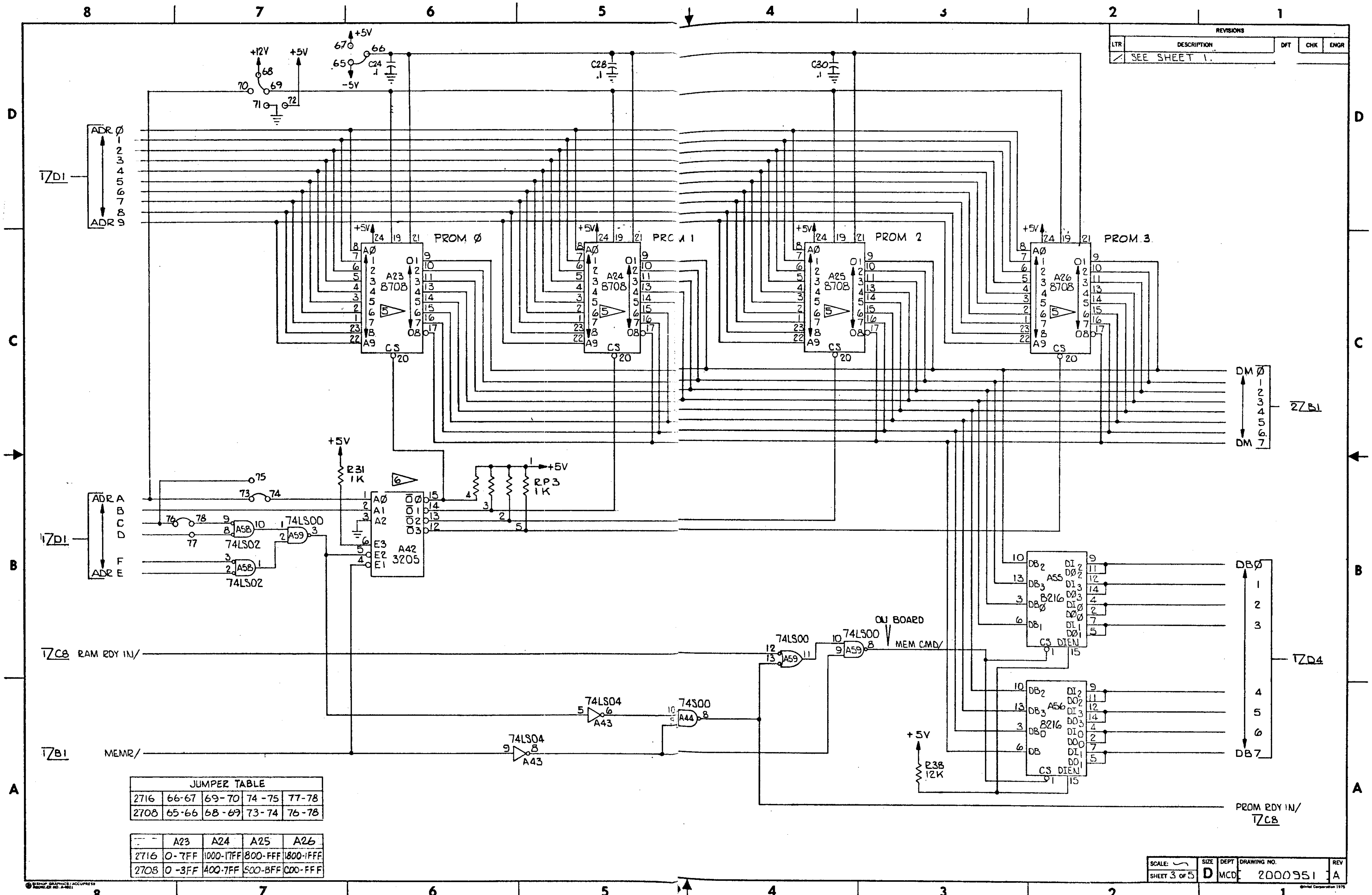


FIGURE A-2. SBC 80/10A SCHEMATIC (SHEET 2 OF 5)



JUMPER TABLE				
2716	66-67	69-70	74-75	77-78
2708	65-66	68-69	73-74	76-78

	A23	A24	A25	A26
2716	0-7FF	1000-17FF	800-FFF	1800-1FFF
2708	0-3FF	400-7FF	800-BFF	C00-FFF

FIGURE A-2. SBC 80/10A SCHEMATIC (SHEET 3 OF 5)

APPENDIX B
TELETYPEWRITER MODIFICATIONS

B-1. INTRODUCTION

This appendix provides information required to modify a Model ASR-33 Teletypewriter for use with certain Intel SBC 80 computer systems.

B-2. INTERNAL MODIFICATIONS

WARNING

Hazardous voltages are exposed when the top cover of the teletypewriter is removed. To prevent accidental shock, disconnect the teleprinter power cord before proceeding beyond this point.

Remove the top cover and modify the teletypewriter as follows:

a. Remove blue lead from 750-ohm tap on current source register; reconnect this lead to 1450-ohm tap. (Refer to figures B-1 and B-2.)

b. On terminal block, change two wires as follows to create an internal full-duplex loop (refer to figures B-1 and B-3):

   1. Remove brown/yellow lead from terminal 3; reconnect this lead to terminal 5.
   2. Remove white/blue lead from terminal 4; reconnect this lead to terminal 5.

c. On terminal block, remove violet lead from terminal 8; reconnect this lead to terminal 9. This changes the receiver current level from 60 mA to 20 mA.

A relay circuit card must be fabricated and connected to the paper tape reader drive circuit. The relay circuit card to be fabricated requires a relay, a diode, a thyractor, a small 'vector' board for mounting the components, and suitable hardware for mounting the assembled relay card.

A circuit diagram of the relay circuit card is included in figure B-4; this diagram also includes the part numbers of the relay, diode, and thyractor. (Note that a 470-ohm resistor and a 0.1 μF capacitor may be substituted for the thyractor.) After the relay circuit card has been assembled, mount it in position as shown in figure B-5. Secure the card to the base plate using two self-tapping screws. Connect the relay circuit to the distributor trip magnet and mode switch as follows:

a. Refer to figure B-4 and connect a wire (Wire 'A') from relay circuit card to terminal L2 on mode switch. (See figure B-6.)

b. Disconnect brown wire shown in figure B-7 from plastic connector. Connect this brown wire to terminal L2 on mode switch. (Brown wire will have to be extended.)

c. Refer to figure B-4 and connect a wire (Wire 'B') from relay circuit board to terminal L1 on mode switch.

B-3. EXTERNAL CONNECTIONS

Connect a two-wire receive loop, a two-wire send loop, and a two-wire tape reader control loop to the external device as shown in figure B-4. The external connector pin numbers shown in figure B-4 are for interface with an RS232C device.

B-4. SBC 530 TTY ADAPTER

The SBC 530, which converts RS232C signal levels to an optically isolated 20 mA current loop interface, provides signal translation for transmitted data, received data, and a paper tape reader relay. The SBC 530 interfaces an Intel SBC 80 computer system to a teletypewriter as shown in figure B-8.

The SBC 530 requires +12V at 98 mA and −12V at 98 mA. An auxiliary supply must be used if the SBC 80 system does not supply this power. A schematic diagram of the SBC 530 is supplied with the unit. The following auxiliary power connector (or equivalent) must be procured by the user:

Connector, Molex 09-50-7071
Pins, Molex 08-50-0106
Polarizing Key, Molex 15-04-0219

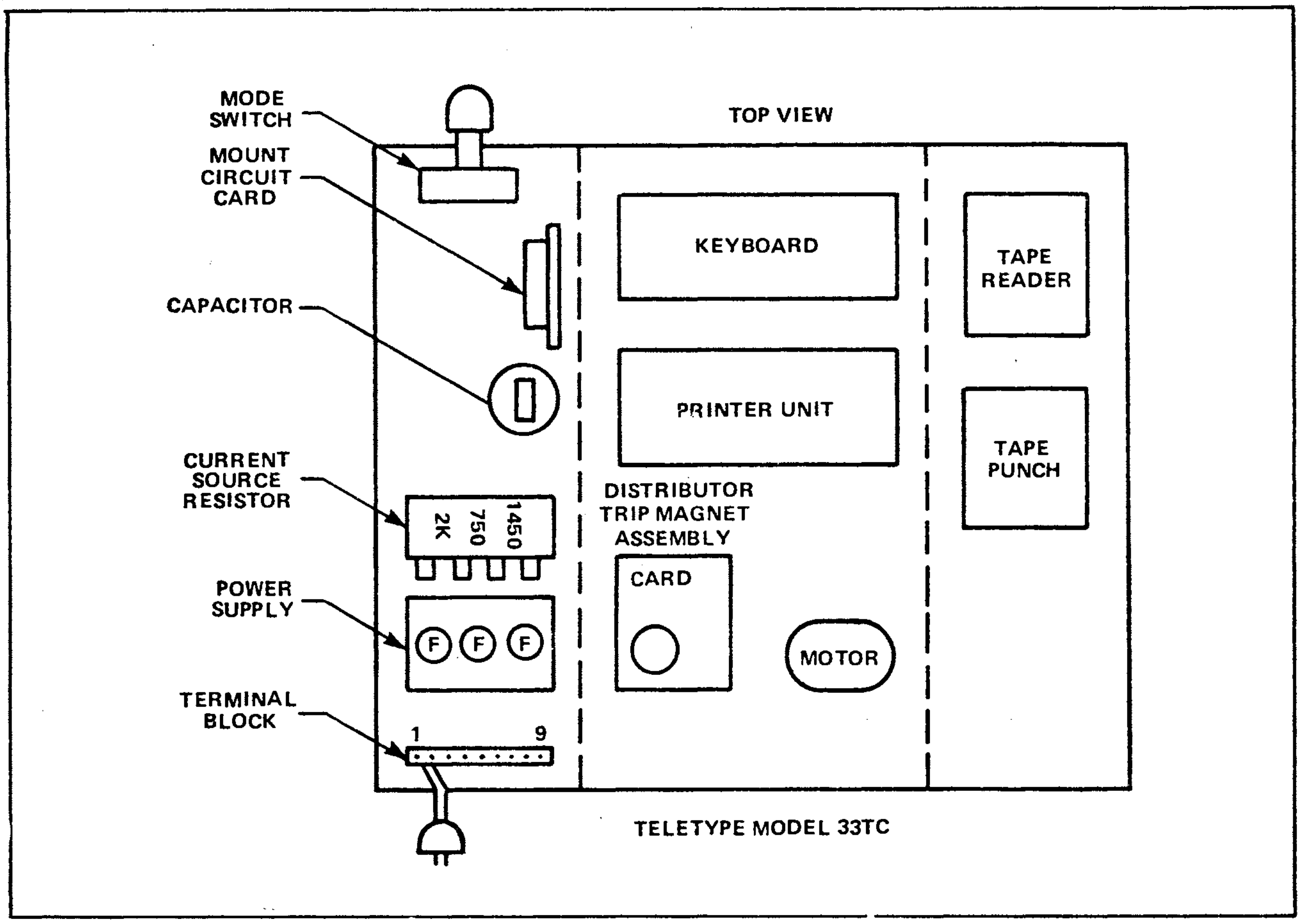


Figure B-1. Teletype Component Layout

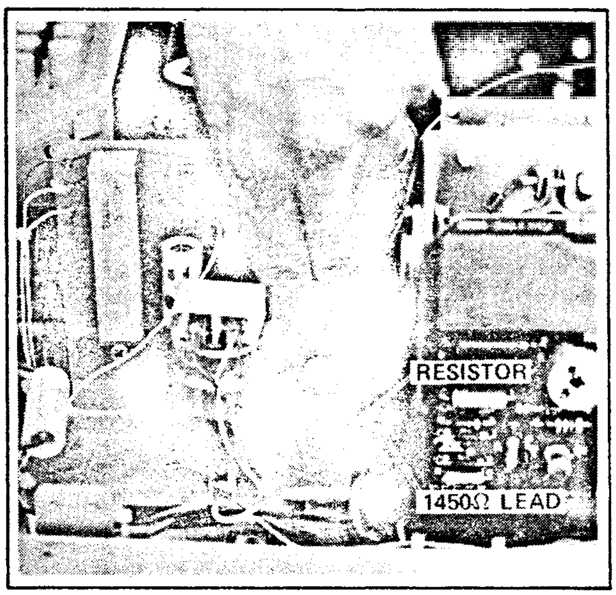


Figure B-2. Current Source Resistor

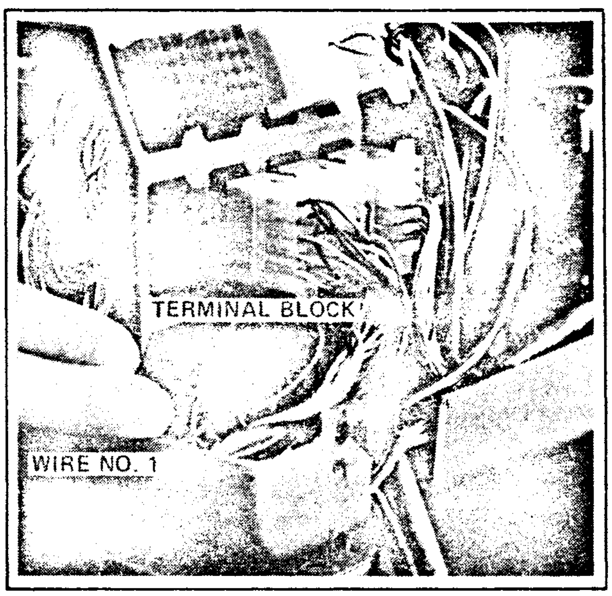


Figure B-3. Terminal Block

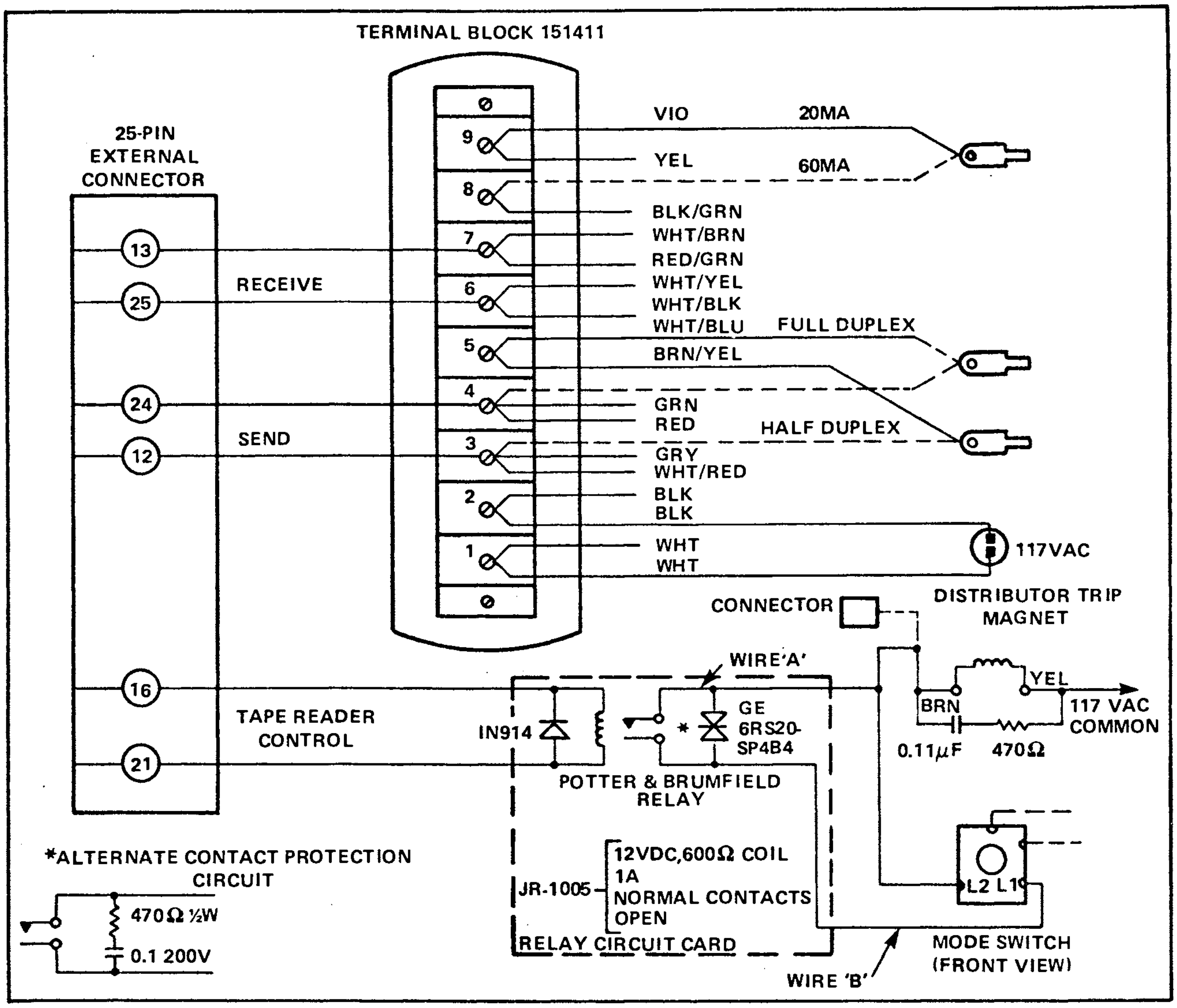


Figure B-4. Teletypewriter Modifications

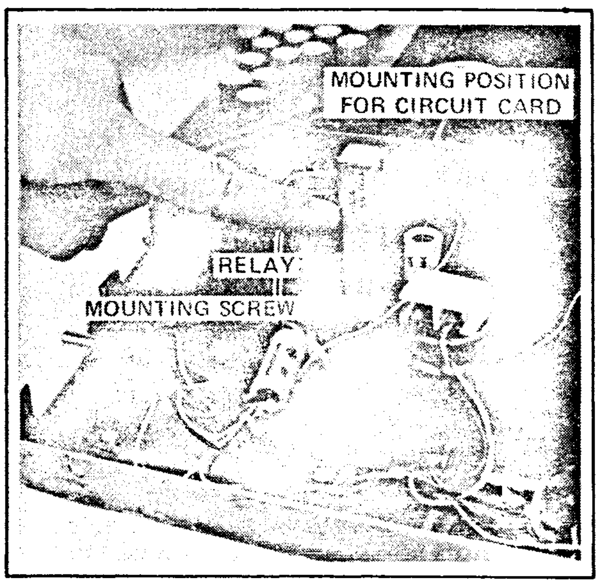


Figure B-5. Relay Circuit

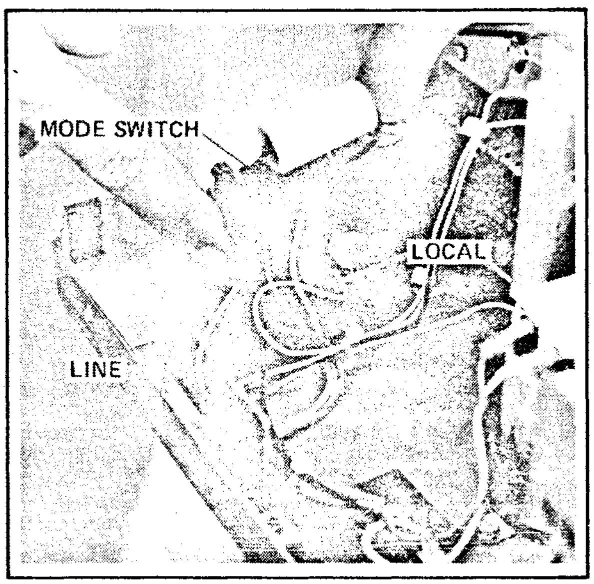


Figure B-6. Mode Switch

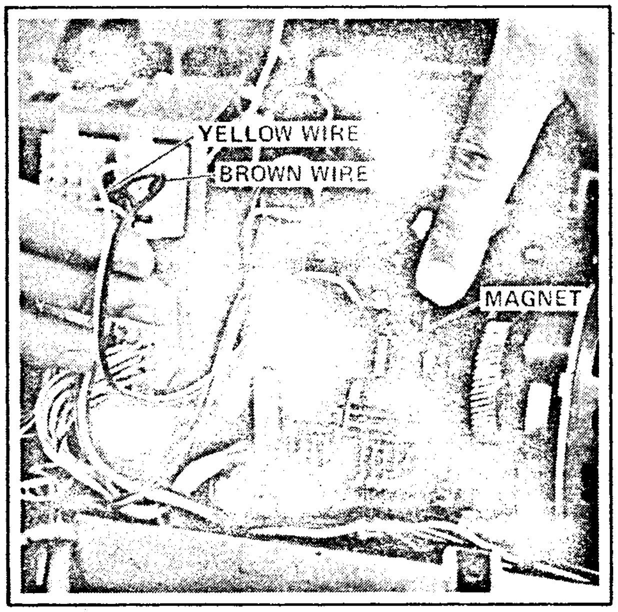


Figure B-7. Distributor Trip Magnet

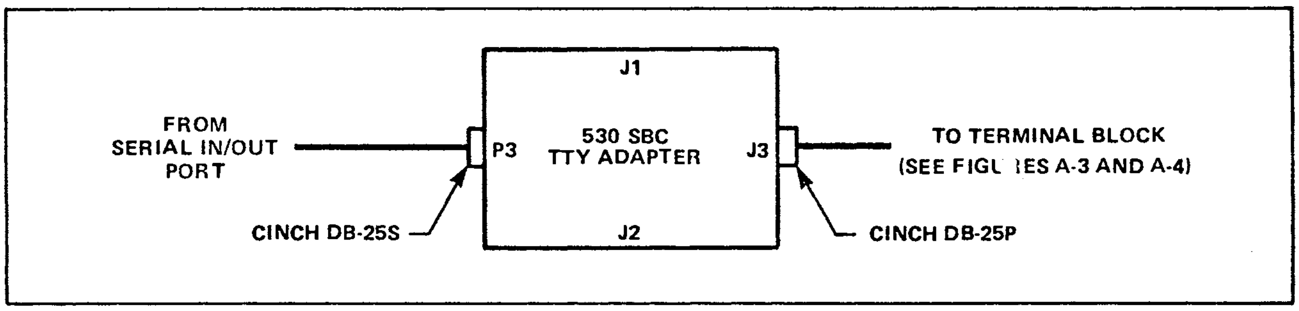


Figure B-8. TTY Adapter Cabling

APPENDIX C

8080 INSTRUCTION SET SUMMARY

A computer, no matter how sophisticated, can only do what it is "told" to do. One "tells" the computer what to do via a series of coded instructions referred to as a Program. The realm of the programmer is referred to as Software, in contrast to the Hardware that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's Instruction Set.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (e.g., OR the contents of two registers) and register operate instructions (e.g., increment a register) are included in the instruction set. A computer's instruction set will also have instructions that move data between registers, between a register and memory, and between register and an I/O device. Most instruction sets also provide Conditional Instructions. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (i.e., a series of 1's and 0's), that is called Machine Code. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is Assembly Language. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the Source Program) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the Object Code). Each assembly language instructions is converted into one machine code instruction (1 or more bytes) by an Assembler program. Assembly languages are usually machine dependent (i.e., they are usually able to run on only one type of computer).

THE 8080 INSTRUCTION SET

The 8080 instruction set includes five different types of instructions:

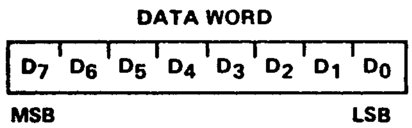
- *Data Transfer Group* – move data between registers or between memory and registers.
- *Arithmetic Group* – add, subtract, increment or decrement data in registers or in memory.
- *Logical Group* – AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory.
- *Branch Group* – conditional and unconditional jump instructions, subroutine call instructions and return instructions.
- *Stack, I/O and Machine Control Group* – includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

Instruction and Data Formats

Memory for the 8080 is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

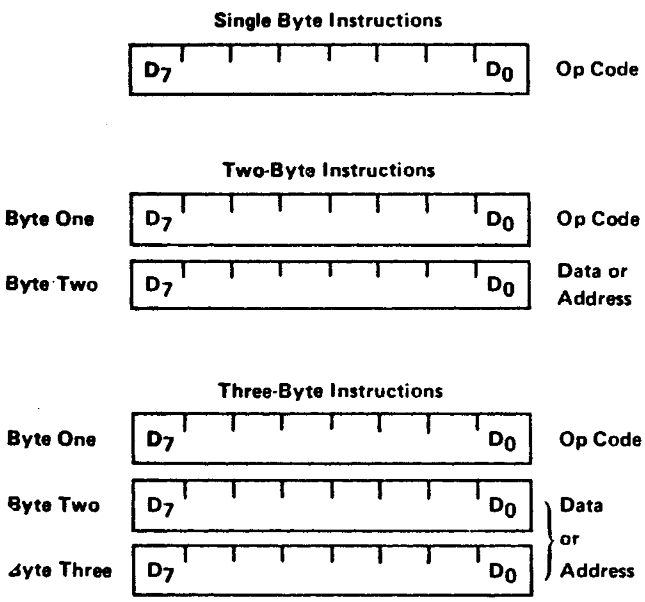
The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8080 is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8080, BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8-bit number) is referred to as the Most Significant Bit (MSB).

The 8080 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



Addressing Modes

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- *Direct* – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- *Register* – The instruction specifies the register-pair in which the data is located.
- *Register Indirect* – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- *Immediate* – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- *Direct* – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- *Register Indirect* – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences).

RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

Condition Flags

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0 (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction of a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r 1,r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L. (DD= destination, SSS=source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp — One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;

D represents the D,E pair with D as the high-order register and E as the low-order register;

H represents the H,L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP — The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh — The first (high-order) register of a designated pair.

rl — The second (low-order) register of a designated register pair.

PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits, respectively).
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits, respectively).
r_m	Bit m of the register r (bits are number 7 through 0 from left to right).
Z,S,P,CY,AC	The condition flags: Zero, Sign, Parity, Carry, and Auxiliary Carry, respectively.
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"A
∧	Logical AND
⊻	Exclusive OR
∨	Inclusive OR
+	Addition
–	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e.g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7, respectively.

Description Format

The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The MAC 80 assembler format, consisting of the instruction mnemonic and operand fields, is printed in BOLDFACE on the left side of the first line.

2. The name of the instruction is enclosed in parenthesis on the right side of the first line.

3. The next line(s) contain a symbolic description of the operation of the instruction.

4. This is followed by a narative description of the operand of the instruction.

5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page A-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

Data Transfer Group

This group of instructions transfer data to and from registers and memory. Condition flags are not affected by any instruction in this group.

MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S

Cycles: 1
States: 5
Addressing: register
Flags: none

MOV r,M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MOV M, r (Move to memory)

((H)(L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S

Cycles:	2
States:	7
Addressing:	reg. indirect
Flags:	none

MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles:	2
States:	7
Addressing:	immediate
Flags:	none

MVI M, data (Move to memory immediate)

((H)(L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles:	3
States:	10
Addressing:	immed./reg. indirect
Flags:	none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles:	3
States:	10
Addressing:	immediate
Flags:	none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles:	4
States:	13
Addressing:	direct
Flags:	none

STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles:	4
States:	13
Addressing:	direct
Flags:	none

MOV M, r (Move to memory)

((H)(L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: none

MVI M, data (Move to memory immediate)

((H)(L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles: 3
States: 10
Addressing: immediate
Flags: none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles: 4
States: 13
Addressing: direct
Flags: none

STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles: 4
States: 13
Addressing: direct
Flags: none

0	0	R	P	0	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

Unless otherwise indicated, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic, and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADD M (Add Memory)

(A) ← (A) + ((H)(L))

The content of the memory location whose address is contained in the H and L register is added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ADI data (Add Immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the constant of the accumulator. The result is placed in the accumulator.

1	1	0	0	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ADC r (Add Register with Carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADC M (Add Memory with Carry)

(A) ← (A) + ((H)(L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

1	0	0	0	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ACI data (Add Immediate with Carry)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

1	1	0	0	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SUB r (Subtract Register)

(A) ← (A) – (r)

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SUB M (Subtract Memory)

(A) ← (A) – ((H)(L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SUI data (Subtract Immediate)

(A) ← (A) – (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	1	0	1	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SBB r (Subtract Register with Borrow)

(A) ← (A) – (r) – (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SBB M (Subtract Memory with Borrow)

(A) ← (A) – ((H)(L)) – (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SBI data (Subtract Immediate with Borrow)

(A) ← (A) – (byte 2) – (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	1	0	1	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INR r (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags except CY are affected.

0	0	D	D	D	1	0	0

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

INR M (Increment Memory)

((H)(L)) ← ((H)(L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags except CY are affected.

0	0	1	1	0	1	0	0

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement Register)

(r) ← (r) – 1

The content of register r is decremented by one. Note: All condition flags except CY are affected.

0	0	D	D	D	1	0	1

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

DCX rp (Decrement register pair)

(rh)(rl) ← (rh)(rl) – 1

The content of the register pair rp is decremented by one. Note: No condition flags are affected.

0	0	R	P	1	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

DAD rp (Add register pair to H and L)

(H)(L) ← (H)(L) + (rh)(rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: Only the CY flag is affected. It is set if there is a carry out of the double precision add; otherwise it is reset.

0	0	R	P	1	0	0	1

Cycles: 3
States: 10
Addressing: register
Flags: CY

DAA (Decimal Adjust Accumulator)

The 8-bit number in the accumulator is adjusted to form two 4-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.

2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

0	0	1	0	0	1	1	1

Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

Logical Group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (AND Register)

$(A) \leftarrow (A) \wedge (r)$

The content of register r is logically ANDed with the content of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

1	0	1	0	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (AND memory)

$(A) \leftarrow (A) \wedge ((H)(L))$

The contents of the memory location whose address is contained in the H and L registers is logically ANDed with the content of the accumulator. The CY flag is cleared.

1	0	1	0	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANI data (AND immediate)

$(A) \leftarrow (A) \wedge (\text{byte } 2)$

The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

1	1	1	0	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR Register)

$(A) \leftarrow (A) \veebar (r)$

The content of register r is exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	0	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

XRA M (Exclusive OR Memory)

$(A) \leftarrow (A) \veebar ((H)(L))$

The content of the memory location whose address is contained in the H and L registers is exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	0	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XRI data (Exclusive OR immediate)
(A) ← (A) V (byte 2)
The content of the second byte of the instruction is exclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	1	1	0	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ORA r (OR Register)
(A) ← (A) V (r)
The content of register r is inclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	1	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR Memory)
(A) ← (A) V ((H)(L))
The content of the memory location whose address is contained in the H and L registers is inclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	0	1	1	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate)
(A) ← (A) V (byte 2)
The content of the second byte of the instruction is inclusive-ORed with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

1	1	1	1	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

CMP r (Compare Register)
(A) – (r)
The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).

1	0	1	1	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

CMP M (Compare memory)
(A) – ((H)(L))
The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 of (A) = ((H)(L)). The CY flag is set to 1 if (A) < ((H)(L)).

1	0	1	1	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S.P,CY,AC

CPI data (Compare immediate)

$(A) - (\text{byte } 2)$

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).

1	1	1	1	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n); (A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low-order bits and the CY flag are both set to the value shifted out of the high-order bit position. Only the CY flag is affected.

0	0	0	0	0	1	1	1

Cycles: 1
States: 4
Flags: CY

RRC (Rotate right)

$(A_n) \leftarrow (A_{n-1}); (A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high-order bit and the CY flag are both set to the value shifted out of the low-order bit position. Only the CY flag is affected.

0	0	0	0	1	1	1	1

Cycles: 1
States: 4
Flags: CY

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n); (CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low-order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high-order bit. Only the CY flag is affected.

0	0	0	1	0	1	1	1

Cycles: 1
States: 4
Flags: CY

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1}); (CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high-order bit is set to the CY flag and the CY flag is set to the value shifted out of the low-order bit. Only the CY flag is affected.

0	0	0	1	1	1	1	1

Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (A)$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). No flags are affected.

0	0	1	0	1	1	1	1

Cycles: 1
States: 4
Flags: none

CMC (Complement carry)

$(CY) \leftarrow (CY)$

The CY flag is complemented. No other flags are affected.

0	0	1	1	1	1	1	1

Cycles: 1
States: 4
Flags: CY

STC (Set carry)

(CY) ← 1

The CY flag is set to 1. No other flags are affected.

0	0	1	1	0	1	1	1

Cycles: 1
States: 4
Flags: CY

Branch Group

This group of instructions alter normal sequential program flow.

Condition flags are not affected by an instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ – not zero (Z=0)	000
Z – zero (Z = 1)	001
NC – no carry (C = 0)	010
C – carry (CY = 1)	011
PO – parity odd (P = 0)	100
PE – parity even (P = 1)	101
P – plus (S = 0)	110
M – minus (S = 1)	111

JMP addr (Jump)

(PC) → (byte 3)(byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	0	0	1	1
low-order addr							
high-order addr							

Cycles: 3
States: 10
Addressing: immediate
Flags: none

Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3)(byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

1	1	C	C	C	0	1	0
low-order addr							
high-order addr							

Cycles: 3
States: 10
Addressing: immediate
Flags: none

CALL addr (Call)

((SP) – 1) ← (PCH)

((SP) – 2) ← (PCL)

(SP) ← (SP) – 2

(PC) ← (byte 3)(byte 2)

The high-order 8 bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order 8 bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5
States: 17
Addressing: immed./reg. indirect
Flags: none

Ccondition addr (Condition call)

If (CCC),

((SP) – 1) ← (PCH)

((SP) – 2) ← (PCL)

(SP) ← (SP) – 2

(PC) ← (byte 3)(byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	1	0	0
low-order addr							
high-order addr							

Cycles: 3/5
States: 11/17
Addressing: immed./reg. indirect
Flags: none

RET (Return)

(PCL) ← ((SP));

(PCH) ← ((SP) + 1);

(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order 8 bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order 8 bits of register PC. The content of register SP is incremented by 2.

1	1	0	0	1	0	0	1

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

Rcondition (Conditional return)

If (CCC),

(PCL) ← ((SP))

(PCH) ← ((SP) + 1)

(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	0	0	0

Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

RST n (Restart)

((SP) – 1) ← (PCH)

((SP) – 2) ← (PCL)

(SP) ← (SP) – 2

(PC) ← 8 * (NNN)

The high-order 8 bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order 8 bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter After Restart

PCHL (Jump H and L indirect – move H and L to PC)

(PCH) ← (H)

(PCL) ← (L)

The content of register H is moved to the high-order 8 bits of register PC. The content of register L is moved to the low-order 8 bits of register PC.

1	1	1	0	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

FLAG WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

PUSH rp (Push)

((SP) – 1) ← (rh)
((SP) – 2) ← (rl)
(SP) ← (SP) – 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Note: Register pair rp=SP may not be specified.

1	1	R	P	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

POP rp (Pop)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. Note: Register pair rp=SP may not be specified.

1	1	R	P	0	0	0	1

Cycles: 3
States: 10
Addressing: reg. indirect.
Flags: none

PUSH PSW (Push processor status word)

((SP) – 1) ← (A)
$((SP) - 2)_0 \leftarrow (CY), ((SP) - 2)_1 \leftarrow 1$
$((SP) - 2)_2 \leftarrow (P), ((SP) - 2)_3 \leftarrow 0$
$((SP) - 2)_4 \leftarrow (AC), ((SP) - 2)_5 \leftarrow 0$
$((SP) - 2)_6 \leftarrow (Z), ((SP) - 2)_7 \leftarrow (S)$
(SP) ← (SP) – 2

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

1	1	1	1	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

1	1	1	1	0	0	0	1

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XTHL (Exchange stack top with H and L)

(L) ↔ ((SP))

(H) ↔ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

1	1	1	0	0	0	1	1

Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

SPHL (Move HL to SP)

(SP) ← (H)(L)

The contents of registers H and L (16 bits) are moved to register SP.

1	1	1	1	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

IN port (Input)

(A) ← (data)

The data placed on the 8-bit bidirectional data bus by the specified port is moved to register A.

1	1	0	1	1	0	1	1
port							

Cycles: 3
States: 10
Addressing: direct
Flags: none

OUT port (Output)

(data) ← (A)

The content of register A is placed on the 8-bit bidirectional data bus for transmission to the specified port.

1	1	0	1	0	0	1	1
port							

Cycles: 3
States: 10
Addressing: direct
Flags: none

EI (Enable interrupt)

The interrupt system is enabled following the execution of the next instruction.

1	1	1	1	1	0	1	1

Cycles: 1
States: 4
Flags: none

DI (Disable interrupts)

The interrupt system is disabled immediately following the execution of the DI instruction.

1	1	1	1	0	0	1	1

Cycles: 1
States: 4
Flags: none

HLT (Halt)

The processor is stopped. The registers and flags are unaffected.

0	1	1	1	0	1	1	0

Cycles: 1
States: 7
Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.

0	0	0	0	0	0	0	0

Cycles: 1
States: 4
Flags: none

Summary of Processor Instructions

MNEMONIC	DESCRIPTION	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	CLOCK[2] CYCLES
MOV $_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVL M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

MNEMONIC	DESCRIPTION	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	CLOCK[2] CYCLES
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS - 000 B - 001 C - 010 D - 011 E - 100 H - 101 L - 110 Memory - 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

RESISTOR NETWORK SPECIFICATIONS:

RESISTANCE VALUES:
±2% (MAX)

OPERATING TEMPERATURE:
0°C TO +70°C

TEMPERATURE COEFFICIENT:
±200 PPM/°C OVER TEMPERATURE RANGE OF 0°C TO +70°C

OPERATING VOLTAGE:
6.0 VDC (MAX)

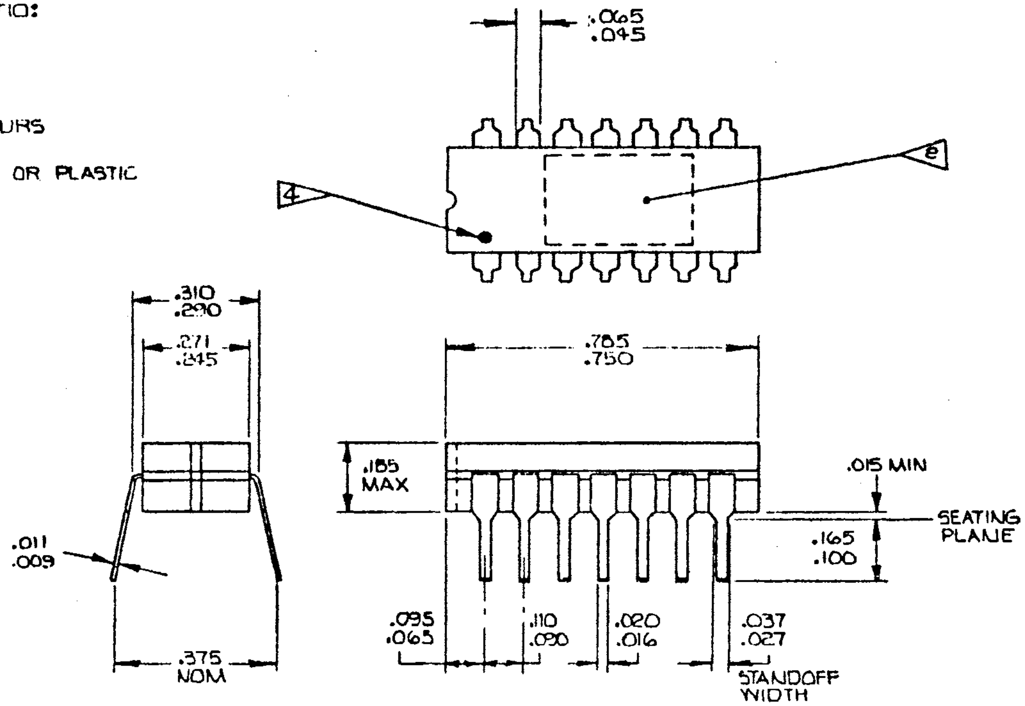
POWER RATING:
AT 70°C, 0.7 WATT PER PACK

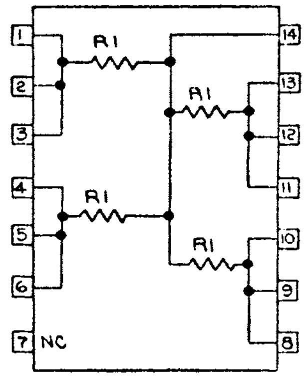
TRACKING RESISTANCE RATIO:
±1.0% (MAX)

STABILITY:
±1% YEAR (MAX)

LOAD LIFE:
±1% (ΔR) OVER 1000 HOURS

PACKAGE:
DUAL IN LINE - CERAMIC OR PLASTIC





NOTES:
UNLESS OTHERWISE SPECIFIED,

1. PART NO IS 4500645-01.

2. INK STAMP PRODUCT CODE, RESISTOR VALUE, PART NO AND DASH NUMBER WITH CONTRASTING COLOR AND MIN .12 HIGH CHARACTERS. NO OTHER MARKINGS ARE PERMITTED EXCEPT FOR MANUF BATCH NO.

E.G.) 5BC-902
R 1K
4500645-01

3. FOR PROCUREMENT SEE LV4500645

4. IDENTIFY PIN ONE CLEARLY ON TOP OF PACKAGE.

DESCRIPTION			
PARTS LIST			
intel®	3065 BOWERS AVE. SANTA CLARA CALIF. 95051		
TITLE TERMINATING PACK PULL UP			
SIZE C	DEPT 410	DRAWING NO. 4500645	REV B

APPENDIX D

SBC-901, SBC-902 SCHEMATICS

Schematic drawings for the SBC-901 and SBC-902 are provided in this appendix. Information and diagrams in this section are subject to change without notice. References should be made to schematics shipped with this module.

RESISTOR NETWORK SPECIFICATIONS:

RESISTANCE VALUES:
±2% (MAX)

OPERATING TEMPERATURE:
0°C TO +70°C

TEMPERATURE COEFFICIENT:
±200 PPM/°C OVER TEMPERATURE RANGE OF 0°C TO +70°C

OPERATING VOLTAGE:
6.0 VDC (MAX)

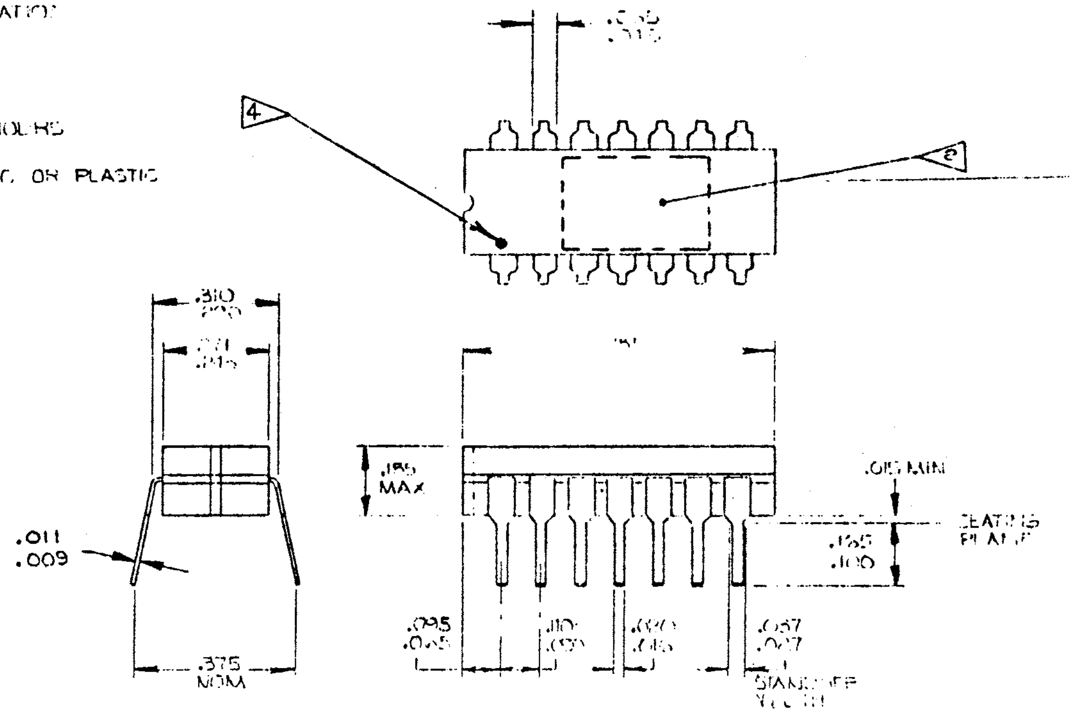
POWER RATING:
AT 70°C, 0.7 WATT PER PACK

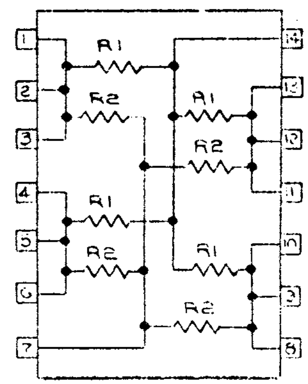
TRACKING RESISTANCE RATIO:
±1.0% (MAX)

STABILITY:
±1% YEAR (MAX)

LOAD LIFE:
±1% (ΔR) OVER 1000 HOURS

PACKAGE:
DUAL IN LINE - CERAMIC OR PLASTIC





NOTES:
UNLESS OTHERWISE SPECIFIED,

1. PART NO IS 4500644-01.

2. INK STAMP PRODUCT CODE, R1/R2 R VALUE, PART NO, AND DASH NUMBER WITH CONTRASTING COLOR INK USING MIN .05 HIGH CHARACTERS. NO OTHER MARKINGS PERMITTED EXCEPT MANUF BATCH NO.

   E.G.) IBC-301
   R220/330
   4500644-01

3. FOR PROCUREMENT SEE LV 4500644-01.

4. IDENTIFY PIN ONE CLEARLY ON TOP OF PACKAGE.

DESCRIPTION

PARTS LIST

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE TERMINATING PACK PULL UP / PULL DOWN

SIZE	DEPT	DRAWING NO.	REV
C	410	4500644	B

APPENDIX E

SBC 80P MONITOR PROGRAM LISTING

```
     TITLE '80/10 MONITOR, VERSION 1.1, 1 NOVEMBER 1976'
;
;
;***********************************************************************
;***********************************************************************
;
;                    80/10 MONITOR
;                    M80/10
;                    VERSION 1.1
;                    1 NOVEMBER 1976
;
;***********************************************************************
;***********************************************************************
;
;    (C) 1976 INTEL CORPORATION. ALL RIGHTS RESERVED. NO PART OF THIS
;    PROGRAM OR PUBLICATION MAY BE REPRODUCED, TRANSMITTED, TRANSCRIBED,
;    STORED IN A RETRIEVAL SYSTEM, OR TRANSLATED INTO ANY LANGUAGE OR
;    COMPUTER LANGUAGE, IN ANY FORM OR BY ANY MEANS, ELECTRONIC,
;    MECHANICAL, MAGNETIC, OPTICAL, CHEMICAL, MANUAL OR OTHERWISE,
;    WITHOUT THE PRIOR WRITTEN PERMISSION OF INTEL CORPORATION,
;    3065 BOWERS AVENUE, SANTA CLARA, CALIFORNIA 95051.
;
;***********************************************************************
;***********************************************************************
;
;
; ABSTRACT
; ========
;
; THIS PROGRAM RUNS ON THE SBC 80/10 BOARD AND IS DESIGNED TO PROVIDE
; THE USER WITH A MINIMAL MONITOR.  BY USING THIS PROGRAM,
; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
; A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
; ALREADY IN MEMORY.  THE MONITOR ALSO PROVIDES THE USER WITH
; ROUTINES FOR PERFORMING CONSOLE I/O AND PAPER TAPE I/O.
;
;
; PROGRAM ORGANIZATION
; ======= ============
;
; THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. FIRST THE BASIC
; MONITOR FUNCTIONS TOGETHER WITH THE CONSOLE I/O ARE LOCATED IN THE
; FIRST 1K OF ROM FOLLOWED BY THE PAPER TAPE FUNCTIONS AND I/O IN THE
; SECOND 1K OF ROM. WITHIN THE FIRST ROM IS CONTAINED THE COMMAND
; RECOGNIZER, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.
; NEXT THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS.  FINALLY,
; THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK.  WITHIN
; EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL
; ORDER, BY ENTRY POINT OF THE ROUTINE. THE SECOND ROM IS ORGANIZED
; IN THE SAME MANNER AS THE FIRST WITH THE ROUTINES WHICH IMPLIMENT
; THE COMMANDS FOLLOWED BY THE UTILITY ROUTINES WHICH ACTUALLY DO THE
```

```
; MORE DETAILED OPERATIONS.
;
; THE PROGRAM HAS BEEN PARTITIONED IN SUCH A MANNER THAT THE SECOND
; ROM NEED NOT BE PLUGGED INTO THE BOARD IF ONLY THE BASIC MONITOR
; FUNCTIONS ARE REQUIRED. HOWEVER IF THE PAPER TAPE FUCTIONS ARE DESIRED
; BOTH ROMS ARE REQUIRED.
;
; THIS PROGRAM EXPECTS TO RUN IN THE FIRST 2K OF ADDRESS SPACE.
; IF, FOR SOME REASON, THE PROGRAM IS RE-ORG'ED, CARE SHOULD
; BE TAKEN TO MAKE SURE THAT THE TRANSFER INSTRUCTIONS FOR RST 1
; AND RST 7 ARE ADJUSTED APPROPRIATELY.
;
; THE PROGRAM ALSO EXPECTS THAT RAM LOCATIONS 3C00H TO 3C3FH,
; INCLUSIVE, ARE RESERVED FOR THE PROGRAM'S OWN USE.  THESE
; LOCATIONS MAY BE ALTERED, HOWEVER, BY CHANGING THE EQU'ED
; SYMBOL "DATA" AS DESIRED.
;
; LIST OF FUNCTIONS
; ==== == =========
;
;       ********
;       1 ST ROM
;       ********
;
;       GETCM
;       -----
;
;       DCMD
;       GCMD
;       ICMD
;       MCMD
;       RCMD
;       SCMD
;       WCMD
;       XCMD
;       -----
;
;       ADRD
;       ADROUT
;       BREAK
;       CI
;       CNVBN
;       CO
;       CROUT
;       ECHO
;       ERROR
;       FRET
;       GETCH
;       GETHX
;       GETNM
;       HILO
```

```
                  ;      INUST
                  ;      NMOUT
                  ;      PRVAL
                  ;      REGDS
                  ;      RGADR
                  ;      RSTTF
                  ;      SRET
                  ;      STHF0
                  ;      STHLF
                  ;      VALDG
                  ;      VALDL
                  ;      -----
                  ;
                  ;      ********
                  ;      2 ND ROM
                  ;      ********
                  ;
                  ;      RCMD
                  ;      WCMD
                  ;      ----
                  ;
                  ;      BYTE
                  ;      DELAY
                  ;      LEAD
                  ;      PADR
                  ;      PBYTE
                  ;      PEOF
                  ;      PEOL
                  ;      PO
                  ;      RI
                  ;      RICH
                  ;      -----
                  ;
0000                     ORG     0H
                  ;
                  ;*****************************************************************
                  ;
                  ;
                  ;                         MONITOR EQUATES
                  ;
                  ;
                  ;*****************************************************************
                  ;
                  ;
001B              BRCHR EQU     1BH     ; CODE FOR BREAK CHARACTER (ESCAPE)
3C3D              BRLOC EQU     3C3DH   ; LOCATION OF USER BRANCH INSTRUCTION IN RAM
03FA              BRTAB EQU     3FAH    ; LOCATION OF START OF BRANCH TABLE IN ROM
0025              CMD   EQU     025H    ; COMMAND INSTRUCTION FOR USART INITIALIZATION
00ED              CNCTL EQU     0EDH    ; CONSOLE (USART) CONTROL PORT
00EC              CNIN  EQU     0ECH    ; CONSOLE INPUT PORT
00EC              CNOUT EQU     0ECH    ; CONSOLE OUTPUT PORT
```

```
00ED            CONST EQU    0EDH    ; CONSOLE STATUS INPUT PORT
000D            CR    EQU    0DH     ; CODE FOR CARRIAGE RETURN
3C00            DATA  EQU    15*1024 ; START OF MONITOR RAM USAGE
001B            ESC   EQU    1BH     ; CODE FOR ESCAPE CHARACTER
000F            HCHAR EQU    0FH     ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
00FF            INVRT EQU    0FFH    ; MASK TO INVERT HALF BYTE FLAG
000A            LF    EQU    0AH     ; CODE FOR LINE FEED
                ;LSGNON      EQU     ---     ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
00CF            MODE  EQU    0CFH    ; MODE SET FOR USART INITIALIZATION
                ;MSTAK       EQU     ---     ; START OF MONITOR STACK - DEFINED LATER
                ;NCMDS       EQU     ---     ; NUMBER OF VALID COMMANDS
000F            NEWLN EQU    0FH     ; MASK FOR CHECKING MEMORY ADDR DISPLAY
007F            PRTY0 EQU    07FH    ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
3C2E            REGS  EQU    DATA+64-18      ; START OF REGISTER SAVE AREA
0002            RBR   EQU    2       ; MASK TO TEST RECEIVER STATUS
0038            RSTU  EQU    38H     ; TRANSFER LOCATION FOR RST 7 INSTRUCTION
                ;RTABS       EQU     ---     ; SIZE OF ENTRY IN RTAB TABLE
001B            TERM  EQU    1BH     ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
0001            TRDY  EQU    1       ; MASK TO TEST TRANSMITTER STATUS
00FF            UPPER EQU    0FFH    ; DENOTES UPPER HALF OF BYTE IN ICMD
0004            TXBE  EQU    04H     ; USART TRANSMITTER BUFFER EMPTY
0027            TTYADV       EQU     27H     ; TTY READER ADVANCE COMMAND
0083            ONEMS EQU    131     ; 1 MILLISECOND CONSTANT
                ;
                ;
                ;*************************************************************
                ;
                ;
                ;                    MONITOR MACROS
                ;
                ;
                ;*************************************************************
                ;
                ;
      1         TRUE  MACRO  WHERE   ; BRANCH IF FUNCTION RETURNS TRUE (SUCCESS)
      1               JC     WHERE
                      ENDM
                ;
      1         FALSE MACRO  WHERE   ; BRANCH IF FUNCTION RETURNS FALSE (FAILURE)
      1               JNC    WHERE
                      ENDM
                ;
                ;
                ;*************************************************************
                ;
                ;
                ;                    USART INITIALIZATION CODE
                ;
                ;
                ;*************************************************************
                ;
```

```
                        ;
                        ;   THE USART IS ASSUMED TO COME UP IN THE RESET POSITION (THIS
                        ;   FUNCTION IS TAKEN CARE OF BY THE HARDWARE).  THE USART WILL
                        ;   BE INITIALIZED IN THE SAME WAY FOR EITHER A TTY OR CRT
                        ;   INTERFACE.  THE FOLLOWING PARAMETERS ARE USED:
                        ;
                        ;      MODE INSTRUCTION
                        ;      ==== ===========
                        ;
                        ;      2 STOP BITS
                        ;      PARITY DISABLED
                        ;      8 BIT CHARACTERS
                        ;      BAUD RATE FACTOR OF 64
                        ;
                        ;      COMMAND INSTRUCTION
                        ;      ======= ===========
                        ;
                        ;      NO HUNT MODE
                        ;      NOT(RTS) FORCED TO 0
                        ;      RECEIVE ENABLED
                        ;      TRANSMIT ENABLED
                        ;
0000   3ECF                   MVI     A,MODE
0002   D3ED                   OUT     CNCTL    ; OUTPUT MODE SET TO USART
0004   C3B202                 JMP     INUST    ; BRANCH TO COMPLETE USART INITIALIZATION
0007   00                     NOP              ; FILLER
                        ;
                        ;****************************************************************
                        ;
                        ;
                        ;                      RESTART ENTRY POINT
                        ;
                        ;
                        ;****************************************************************
                        ;
                        ;
0008                    GO:
0008   22343C                 SHLD    LSAVE    ; SAVE HL REGISTERS
000B   E1                     POP     H        ; GET TOP OF STACK ENTRY
000C   22363C                 SHLD    PSAVE    ; ASSUME THIS IS LAST P COUNTER
000F   F5                     PUSH    PSW      ; SAVE A,F/F'S
0010   210200                 LXI     H,2      ; SET HL TO 2 SO THAT STACK POINTER SAVED CORRECTLY
0013   39                     DAD     SP       ; GET STACK POINTER VALUE
0014   22383C                 SHLD    SSAVE    ; SAVE USER'S STACK POINTER
0017   F1                     POP     PSW      ; RESTORE A,F/F'S
0018   31343C                 LXI     SP,ASAVE+1       ; NEW VALUE FOR STACK POINTER
001B   C3B101                 JMP     ADROUT
                        ;
                        ;************************************************************
                        ;
                        ;
```

```
                        ;                       PRINT SIGNON MESSAGE
                        ;
                        ;
                        ;***********************************************************
                        ;
                        ;
001E                    SOMSG:
001E    219503                LXI     H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
0021    0611                  MVI     B,LSGNON        ; COUNTER FOR CHARACTERS IN MESSAGE
0023                    MSGL:
0023    4E                    MOV     C,M     ; FETCH NEXT CHAR TO C REG
0024    CDE801                CALL    CO      ; SEND IT TO THE CONSOLE
0027    23                    INX     H       ; POINT TO NEXT CHARACTER
0028    05                    DCR     B       ; DECREMENT BYTE COUNTER
0029    C22300                JNZ     MSGL    ; RETURN FOR NEXT CHARACTER
                        ;
                        ;
                        ;****************************************************************
                        ;
                        ;
                        ;                  COMMAND RECOGNIZING ROUTINE
                        ;
                        ;
                        ;****************************************************************
                        ;
                        ; FUNCTION: GETCM
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: GETCH,ECHO,ERROR
                        ; DESTROYS: A,B,C,H,L,F/F'S
                        ; DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
                        ;              AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
                        ;              CHARACTER TABLE.  IF SUCCESSFUL, THE ROUTINE
                        ;              CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
                        ;              A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
                        ;              IS TRANSFERRED TO THIS ROUTINE.  IF THE CHARACTER
                        ;              DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
                        ;              THE ERROR HANDLER.
                        ;
002C                    GETCM:
002C    312E3C                LXI     SP,MSTAK        ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
                                              ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
002F    0E2E                  MVI     C,'.'   ; PROMPT CHARACTER TO C
0031    CDF901                CALL    ECHO    ; SEND PROMPT CHARACTER TO USER TERMINAL
0034    C33C00                JMP     GTC03   ; WANT TO LEAVE ROOM FOR RST BRANCH
                        ;
0038                          ORG     RSTU    ; ORG TO RST TRANSFER LOCATION
0038    C33D3C  NOP           JMP     USRBR   ; JUMP TO USER BRANCH LOCATION
003B    00                    NOP             ; FILLER
                        ;
003C                    GTC03:
```

```
ØØ3C   CD2ØØ2          CALL    GETCH    ; GET COMMAND CHARACTER TO A
ØØ3F   CDF9Ø1          CALL    ECHO     ; ECHO CHARACTER TO USER
ØØ42   79              MOV     A,C      ; PUT COMMAND CHARACTER INTO ACCUMULATOR
ØØ43   Ø1Ø8ØØ          LXI     B,NCMDS  ; C CONTAINS LOOP AND INDEX COUNT
ØØ46   21B8Ø3          LXI     H,CTAB   ; HL POINTS INTO COMMAND TABLE
ØØ49           GTCØ5:
ØØ49   BE              CMP     M        ; COMPARE TABLE ENTRY AND CHARACTER
ØØ4A   CA55ØØ          JZ      GTC1Ø    ; BRANCH IF EQUAL - COMMAND RECOGNIZED
ØØ4D   23              INX     H        ; ELSE, INCREMENT TABLE POINTER
ØØ4E   ØD              DCR     C        ; DECREMENT LOOP COUNT
ØØ4F   C249ØØ          JNZ     GTCØ5    ; BRANCH IF NOT AT TABLE END
ØØ52   C312Ø2          JMP     ERROR    ; ELSE, COMMAND CHARACTER IS ILLEGAL
ØØ55           GTC1Ø:
ØØ55   21A6Ø3          LXI     H,CADR   ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
                                        ; /OF COMMAND ROUTINE ADDRESSES
ØØ58   Ø9              DAD     B        ; ADD WHAT IS LEFT OF LOOP COUNT
ØØ59   Ø9              DAD     B        ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
ØØ5A   7E              MOV     A,M      ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
ØØ5B   23              INX     H        ; POINT TO NEXT BYTE IN TABLE
ØØ5C   66              MOV     H,M      ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
ØØ5D   6F              MOV     L,A      ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
ØØ5E   E9              PCHL             ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
               ;
               ;
               ;*********************************************************************
               ;
               ;
               ;               COMMAND IMPLEMENTING ROUTINES
               ;
               ;
               ;*********************************************************************
               ;
               ;
               ; FUNCTION: DCMD
               ; INPUTS: NONE
               ; OUTPUTS: NONE
               ; CALLS: ECHO,NMOUT,HILO,GETCM,CROUT,GETNM
               ; DESTROYS: A,B,C,D,E,H,L,F/F'S
               ; DESCRIPTION: DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND
               ;
ØØ5F           DCMD:
ØØ5F   ØEØ2            MVI     C,2      ; GET TWO NUMBERS FROM INPUT STREAM
ØØ61   CD5BØ2          CALL    GETNM
ØØ64   D1              POP     D        ; ENDING ADDRESS TO DE
ØØ65   E1              POP     H        ; STARTING ADDRESS TO HL
ØØ66           DCMØ5:
ØØ66   CDF3Ø1          CALL    CROUT    ; ECHO CARRIAGE RETURN/LINE FEED
ØØ69   CDA8Ø1          CALL    ADRD     ; DISPLAY ADDRESS
ØØ6C           DCM1Ø:
ØØ6C   ØE2Ø            MVI     C,' '
ØØ6E   CDF9Ø1          CALL    ECHO     ; USE BLANK AS SEPARATOR
```

```
0071    7E                  MOV     A,M     ; GET CONTENTS OF NEXT MEMORY LOCATION
0072    CDC202              CALL    NMOUT   ; DISPLAY CONTENTS
0075    CDC201              CALL    BREAK   ; SEE IF USER WANTS OUT
      1            +        TRUE    EXIT    ; IF SO, BRANCH TO EXIT
0078  1 DA1702     +        JC      EXIT
007B    CDA002              CALL    HILO    ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
                                            ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
      1            +        TRUE    EXIT    ; EXIT IF NO MORE TO DISPLAY
007E  1 DA1702     +        JC      EXIT
0081    23                  INX     H       ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
0082    7D                  MOV     A,L     ; GET LOW ORDER BITS OF NEW ADDRESS
0083    E60F                ANI     NEWLN   ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
                                            ; /START OF NEW LINE
0085    C26C00              JNZ     DCM10   ; NO - NOT AT END OF LINE
0088    C36600              JMP     DCM05   ; YES - START NEW LINE WITH ADDRESS
                    ;
                    ;
                    ;*****************************************************************
                    ;
                    ;
                    ; FUNCTION: GCMD
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: ERROR,GETHX,RSTTF
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
                    ;
008B                GCMD:
008B    CD2702              CALL    GETHX   ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
      1            +        FALSE   GCM05   ; BRANCH IF NO NUMBER PRESENT
008E  1 D2A000     +        JNC     GCM05
0091    7A                  MOV     A,D     ; ELSE, GET TERMINATOR
0092    FE0D                CPI     CR      ; SEE IF CARRIAGE RETURN
0094    C21202              JNZ     ERROR   ; ERROR IF NOT PROPERLY TERMINATED
0097    21363C              LXI     H,PSAVE ; WANT NUMBER TO REPLACE SAVE PGM COUNTER
009A    71                  MOV     M,C
009B    23                  INX     H
009C    70                  MOV     M,B
009D    C3A600              JMP     GCM10
00A0                GCM05:
00A0    7A                  MOV     A,D     ; IF NO STARTING ADDRESS, MAKE SURE THAT
00A1    FE0D                CPI     CR      ; /CARRIAGE RETURN TERMINATED COMMAND
00A3    C21202              JNZ     ERROR   ; ERROR IF NOT
00A6                GCM10:
00A6    C32703              JMP     RSTTF   ; RESTORE REGISTERS AND BEGIN EXECUTION

                    ;
                    ;
                    ;*****************************************************************
                    ;
                    ;
```

```
                    ; FUNCTION: ICMD
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: ERROR,ECHO,GETCH,VALDL,VALDG,CNVBN,STHLF,GETNM,CROUT
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: ICMD IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND.
                    ;
ØØA9                ICMD:
ØØA9   ØEØ1                 MVI     C,1
ØØAB   CD5BØ2               CALL    GETNM   ; GET SINGLE NUMBER FROM INPUT STREAM
ØØAE   3EFF                 MVI     A,UPPER
ØØBØ   323A3C               STA     TEMP    ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
ØØB3   D1                   POP     D       ; ADDRESS OF START TO DE
ØØB4                ICMØ5:
ØØB4   CD2CØ2               CALL    GETCH   ; GET A CHARACTER FROM INPUT STREAM
ØØB7   CDF9Ø1               CALL    ECHO    ; ECHO IT
ØØBA   79                   MOV     A,C     ; PUT CHARACTER BACK INTO A
ØØBB   FE1B                 CPI     TERM    ; SEE IF CHARACTER IS A TERMINATING CHARACTER
ØØBD   CAE9ØØ               JZ      ICM25   ; IF SO, ALL DONE ENTERING CHARACTERS
ØØCØ   CD82Ø3               CALL    VALDL   ; ELSE, SEE IF VALID DELIMITER
     1            +         TRUE    ICMØ5   ; IF SO SIMPLY IGNORE THIS CHARACTER
ØØC3 1 DAB4ØØ     +         JC      ICMØ5
ØØC6   CD67Ø3               CALL    VALDG   ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
     1            +         FALSE   ICM2Ø   ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
ØØC9 1 D2E3ØØ     +         JNC     ICM2Ø
ØØCC   CDDFØ1               CALL    CNVBN   ; CONVERT DIGIT TO BINARY
ØØCF   4F                   MOV     C,A     ; MOVE RESULT TO C
ØØDØ   CD48Ø3               CALL    STHLF   ; STORE IN APPROPRIATE HALF WORD
ØØD3   3A3A3C               LDA     TEMP    ; GET HALF BYTE FLAG
ØØD6   B7                   ORA     A       ; SET F/F'S
ØØD7   C2DBØØ               JNZ     ICM1Ø   ; BRANCH IF FLAG SET FOR UPPER
ØØDA   13                   INX     D       ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
ØØDB                ICM1Ø:
ØØDB   EEFF                 XRI     INVRT   ; TOGGLE STATE OF FLAG
ØØDD   323A3C               STA     TEMP    ; PUT NEW VALUE OF FLAG BACK
ØØEØ   C3B4ØØ               JMP     ICMØ5   ; PROCESS NEXT DIGIT
ØØE3                ICM2Ø:
ØØE3   CD3DØ3               CALL    STHFØ   ; ILLEGAL CHARACTER
ØØE6   C312Ø2               JMP     ERROR   ; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
ØØE9                ICM25:
ØØE9   CD3DØ3               CALL    STHFØ   ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
ØØEC   C317Ø2               JMP     EXIT
                    ;
                    ;
                    ;*****************************************************************
                    ;
                    ;
                    ; FUNCTION: MCMD
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: GETCM,HILO,GETNM
```

```
                        ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                        ; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
                        ;
00EF                    MCMD:
00EF     0E03                 MVI     C,3
00F1     CD5B02               CALL    GETNM    ; GET 3 NUMBERS FROM INPUT STREAM
00F4     C1                   POP     B        ; DESTINATION ADDRESS TO BC
00F5     E1                   POP     H        ; ENDING ADDRESS TO HL
00F6     D1                   POP     D        ; STARTING ADDRESS TO DE
00F7                    MCM05:
00F7     E5                   PUSH    H        ; SAVE ENDING ADDRESS
00F8     62                   MOV     H,D
00F9     6B                   MOV     L,E      ; SOURCE ADDRESS TO HL
00FA     7E                   MOV     A,M      ; GET SOURCE BYTE
00FB     60                   MOV     H,B
00FC     69                   MOV     L,C      ; DESTINATION ADDRESS TO HL
00FD     77                   MOV     M,A      ; MOVE BYTE TO DESTINATION
00FE     03                   INX     B        ; INCREMENT DESTINATION ADDRESS
00FF     78                   MOV     A,B
0100     B1                   ORA     C        ; TEST FOR DESTINATION ADDRESS OVERFLOW
0101     CA2C00               JZ      GETCM    ; IF SO, CAN TERMINATE COMMAND
0104     13                   INX     D        ; INCREMENT SOURCE ADDRESS
0105     E1                   POP     H        ; ELSE, GET BACK ENDING ADDRESS
0106     CDA002               CALL    HILO     ; SEE IF ENDING ADDR>=SOURCE ADDR
      1                 +     FALSE   GETCM    ; IF NOT, COMMAND IS DONE
0109  1  D22C00         +     JNC     GETCM
010C     C3F700               JMP     MCM05    ; MOVE ANOTHER BYTE
                        ;
                        ;
                        ;*****************************************************************
                        ;
                        ;
                        ; FUNCTION: SCMD
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: GETHX,GETCM,NMOUT,ECHO
                        ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                        ; DESCRIPTION: SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.
                        ;
010F                    SCMD:
010F     CD2702               CALL    GETHX    ; GET A NUMBER, IF PRESENT, FROM INPUT
0112     C5                   PUSH    B
0113     E1                   POP     H        ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
0114                    SCM05:
0114     7A                   MOV     A,D      ; GET TERMINATOR
0115     FE20                 CPI     ' '      ; SEE IF SPACE
0117     CA1F01               JZ      SCM10    ; YES - CONTINUE PROCESSING
011A     FE2C                 CPI     ','      ; ELSE, SEE IF COMMA
011C     C22C00               JNZ     GETCM    ; NO - TERMINATE COMMAND
011F                    SCM10:
011F     7E                   MOV     A,M      ; GET CONTENTS OF SPECIFIED LOCATION TO A
```

```
Ø12Ø   CDC2Ø2            CALL   NMOUT   ; DISPLAY CONTENTS ON CONSOLE
Ø123   ØE2D              MVI    C,'-'
Ø125   CDF9Ø1            CALL   ECHO    ; USE DASH FOR SEPARATOR
Ø128   CD27Ø2            CALL   GETHX   ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
     1            +      FALSE  SCM15   ; IF NO VALUE PRESENT, BRANCH
Ø12B 1 D22FØ1     +      JNC    SCM15
Ø12E   71                MOV    M,C     ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
Ø12F              SCM15:
Ø12F   23                INX    H       ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
Ø13Ø   C314Ø1            JMP    SCMØ5
                  ;
                  ;
                  ;*************************************************************
                  ;
                  ;
                  ; FUNCTION: XCMD
                  ; INPUTS: NONE
                  ; OUTPUTS: NONE
                  ; CALLS: GETCH,ECHO,REGDS,GETCM,ERROR,RGADR,NMOUT,CROUT,GETHX
                  ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  ; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
                  ;              COMMAND.
                  ;
Ø133              XCMD:
Ø133   CD2ØØ2            CALL   GETCH   ; GET REGISTER IDENTIFIER
Ø136   CDF9Ø1            CALL   ECHO    ; ECHO IT
Ø139   79                MOV    A,C
Ø13A   FEØD              CPI    CR
Ø13C   C245Ø1            JNZ    XCMØ5   ; BRANCH IF NOT CARRIACE RETURN
Ø13F   CDDEØ2            CALL   REGDS   ; ELSE, DISPLAY REGISTER CONTENTS
Ø142   C32CØØ            JMP    GETCM   ; THEN TERMINATE COMMAND
Ø145              XCMØ5:
Ø145   4F                MOV    C,A     ; GET REGISTER IDENTIFIER TO C
Ø146   CD1FØ3            CALL   RGADR   ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
Ø149   C5                PUSH   B
Ø14A   E1                POP    H       ; PUT POINTER TO REGISTER ENTRY INTO HL
Ø14B   ØE2Ø              MVI    C,' '
Ø14D   CDF9Ø1            CALL   ECHO    ; ECHO SPACE TO USER
Ø15Ø   79                MOV    A,C
Ø151   323A3C            STA    TEMP    ; PUT SPACE INTO TEMP AS DELIMITER
Ø154              XCM1Ø:
Ø154   3A3A3C            LDA    TEMP    ; GET TERMINATOR
Ø157   FE2Ø              CPI    ' '     ; SEE IF A BLANK
Ø159   CA61Ø1            JZ     XCM15   ; YES - GO CHECK POINTER INTO TABLE
Ø15C   FE2C              CPI    ','     ; NO - SEE IF COMMA
Ø15E   C22CØØ            JNZ    GETCM   ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
Ø161              XCM15:
Ø161   7E                MOV    A,M
Ø162   B7                ORA    A       ; SET F/F'S
Ø163   CA17Ø2            JZ     EXIT    ; BRANCH IF AT END OF TABLE
Ø166   E5                PUSH   H       ; PUT POINTER ON STACK
```

```
0167   5E             MOV    E,M
0168   163C           MVI    D,DATA SHR 8    ; FETCH ADDRESS OF SAVE LOCATION FROM
016A   23             INX    H               ; /TABLE
016B   46             MOV    B,M     ; FETCH LENGTH FLAG FROM TABLE
016C   D5             PUSH   D       ; SAVE ADDRESS OF SAVE LOCATION
016D   D5             PUSH   D
016E   E1             POP    H       ; MOVE ADDRESS TO HL
016F   C5             PUSH   B       ; SAVE LENGTH FLAG
0170   7E             MOV    A,M     ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
0171   CDC202         CALL   NMOUT   ; DISPLAY IT
0174   F1             POP    PSW     ; GET BACK LENGTH FLAG
0175   F5             PUSH   PSW     ; SAVE IT AGAIN
0176   B7             ORA    A       ; SET F/F'S
0177   CA7F01         JZ     XCM20   ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
017A   2B             DCX    H       ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
017B   7E             MOV    A,M
017C   CDC202         CALL   NMOUT   ; DISPLAY THEM
017F             XCM20:
017F   0E2D           MVI    C,'-'
0181   CDF901         CALL   ECHO    ; USE DASH AS SEPARATOR
0184   CD2702         CALL   GETHX   ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
          1      +    FALSE  XCM30   ; NO - GO CHECK FOR NEXT REGISTER
0187 1 D29F01    +    JNC    XCM30
018A   7A             MOV    A,D
018B   323A3C         STA    TEMP    ; ELSE, SAVE THE TERMINATOR FOR NOW
018E   F1             POP    PSW     ; GET BACK LENGTH FLAG
018F   E1             POP    H       ; PUT ADDRESS OF SAVE LOCATION INTO HL
0190   B7             ORA    A       ; SET F/F'S
0191   CA9601         JZ     XCM25   ; IF 8 BIT REGISTER, BRANCH
0194   70             MOV    M,B     ; SAVE UPPER 8 BITS
0195   2B             DCX    H       ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
0196             XCM25:
0196   71             MOV    M,C     ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
0197             XCM27:
0197   110300         LXI    D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
019A   E1             POP    H       ; POINTER INTO REGISTER TABLE RTAB
019B   19             DAD    D       ; ADD ENTRY SIZE TO POINTER
019C   C35401         JMP    XCM10   ; DO NEXT REGISTER
019F             XCM30:
019F   7A             MOV    A,D     ; GET TERMINATOR
01A0   323A3C         STA    TEMP    ; SAVE IN MEMORY
01A3   D1             POP    D       ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
01A4   D1             POP    D       ; /OF SAVE LOCATION
01A5   C39701         JMP    XCM27   ; GO INCREMENT REGISTER TABLE POINTER
                 ;
                 ;
                 ;*************************************************************
                 ;
                 ;
                 ;               UTILITY ROUTINES
                 ;
```

```
                        ;
                        ;*****************************************************************
                        ;
                        ;
                        ;*****************************************************************
                        ;
                        ;
                        ; FUNCTION ADRD
                        ; INPUTS: HL - ADDRESS TO BE DISPLAYED
                        ; OUTPUTS: NONE
                        ; CALLS: NMOUT
                        ; DESTROYS: A
                        ; DESCRIPTION: ADRD OUTPUTS TO THE CONSOLE THE ADDRESS
                        ;              CONTAINED IN THE H,L REGISTERS.
                        ;
01A8                    ADRD:
01A8    7C                    MOV     A,H       ; DISPLAY FIRST HALF OF ADDRESS
01A9    CDC202                CALL    NMOUT
01AC    7D                    MOV     A,L       ; DISPLAY SECOND HALF OF ADDRESS
01AD    CDC202                CALL    NMOUT
01B0    C9                    RET               ; RETURN TO CALLING ROUTINE
                        ;
                        ;
                        ;*****************************************************************
                        ;
                        ;
                        ; FUNCTION ADROUT
                        ; INPUTS: USER REGISTERS ON THE STACK
                        ; OUTPUTS: NOTHING
                        ; CALLS: ECHO,ADRD
                        ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                        ; DESCRIPTION: ADROUT SAVES THE USER REGISTERS AND OUTPUTS TO THE
                        ;              CONSOLE THE USER P COUNTER AFTER A RST 1 INSTRUCTION.
                        ;
01B1                    ADROUT:
01B1    F5                    PUSH    PSW       ; SAVE A AND FLAGS
01B2    C5                    PUSH    B         ; SAVE B AND C
01B3    D5                    PUSH    D         ; SAVE D AND E
01B4    0E23                  MVI     C,'#'
01B6    CDF901                CALL    ECHO      ; OUTPUT '#'
01B9    2A363C                LHLD    PSAVE     ; LOAD USER P COUNTER
01BC    CDA801                CALL    ADRD      ; DISPLAY ADDRESS
01BF    C31702                JMP     EXIT      ; GET NEW COMMAND
                        ;
                        ;
                        ; FUNCTION: BREAK
                        ; INPUTS: NONE
                        ; OUTPUTS: CARRY - 1 IF ESCAPE CHARACTER INPUT
                        ;                - 0 IF ANY OTHER CHARACTER OR NO CHARACTER PENDING
                        ; CALLS: NOTHING
                        ; DESTROYS: A,F/F'S
```

```
               ; DESCRIPTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM
               ;              THE USER.  IF NO CHARACTER IS PENDING, OR IF THE
               ;              PENDING CHARACTER IS NOT THE ESCAPE, THEN A FAILURE
               ;              RETURN (CARRY=0) IS TAKEN.  IN THIS CASE, THE
               ;              PENDING CHARACTER (IF ANY) IS LOST.  IF THE PENDING
               ;              CHARACTER IS AN ESCAPE CHARACTER, BREAK TAKES A SUCCESS
               ;              RETURN (CARRY=1).
               ;
01C2           BREAK:
01C2  DBED          IN      CONST    ; GET CONSOLE STATUS
01C4  E602          ANI     RBR      ; SEE IF CHARACTER PENDING
01C6  CA1D02        JZ      FRET     ; NO - TAKE FAILURE RETURN
01C9  DBEC          IN      CNIN     ; YES - PICK UP CHARACTER
01CB  E67F          ANI     PRTY0    ; STRIP OFF PARITY BIT
01CD  FE1B          CPI     BRCHR    ; SEE IF BREAK CHARACTER
01CF  CA3B03        JZ      SRET     ; YES - SUCCESS RETURN
01D2  C31D02        JMP     FRET     ; NO - FAILURE RETURN - CHARACTER LOST
               ;
               ;
               ;*****************************************************************
               ;
               ;
               ; FUNCTION: CI
               ; INPUTS: NONE
               ; OUTPUTS: A - CHARACTER FROM CONSOLE
               ; CALLS: NOTHING
               ; DESTROYS: A,F/F'S
               ; DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
               ;              CONSOLE AND THEN RETURNS THE CHARACTER, VIA THE A
               ;              REGISTER, TO THE CALLING ROUTINE.  THIS ROUTINE
               ;              IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
               ;
01D5           CI:
01D5  DBED          IN      CONST    ; GET STATUS OF CONSOLE
01D7  E602          ANI     RBR      ; CHECK FOR RECEIVER BUFFER READY
01D9  CAD501        JZ      CI       ; NOT YET - WAIT
01DC  DBEC          IN      CNIN     ; READY SO GET CHARACTER
01DE  C9            RET
               ;
               ;
               ;*****************************************************************
               ;
               ;
               ; FUNCTION: CNVBN
               ; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
               ; OUTPUTS: A - 0 TO F HEX
               ; CALLS: NOTHING
               ; DESTROYS: A,F/F'S
               ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
               ;              CHARACTER INTO ITS CORRESPONDING BINARY VALUE.  CNVBN
               ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT.
```

```
                        ;
01DF                    CNVBN:
01DF    79                      MOV     A,C
01E0    D630                    SUI     '0'     ; SUBTRACT CODE FOR '0' FROM ARGUMENT
01E2    FE0A                    CPI     10      ; WANT TO TEST FOR RESULT OF 0 TO 9
01E4    F8                      RM              ; IF SO, THEN ALL DONE
01E5    D607                    SUI     7       ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
01E7    C9                      RET             ; SO RETURN AFTER SUBTRACTING BIAS OF 7
                        ;
                        ;
                        ;*********************************************************************
                        ;
                        ;
                        ; FUNCTION: CO
                        ; INPUTS: C - CHARACTER TO OUTPUT TO CONSOLE
                        ; OUTPUTS: C - CHARACTER OUTPUT TO CONSOLE
                        ; CALLS: NOTHING
                        ; DESTROYS: A,F/F'S
                        ; DESCRIPTION: CO WAITS UNTIL THE CONSOLE IS READY TO ACCEPT A CHARACTER
                        ;              AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.
                        ;
01E8                    CO:
01E8    DBED                    IN      CONST   ; GET STATUS OF CONSOLE
01EA    E601                    ANI     TRDY    ; SEE IF TRANSMITTER READY
01EC    CAE801                  JZ      CO      ; NO - WAIT
01EF    79                      MOV     A,C     ; ELSE, MOVE CHARACTER TO A REGISTER FOR OUTPUT
01F0    D3EC                    OUT     CNOUT   ; SEND TO CONSOLE
01F2    C9                      RET
                        ;
                        ;
                        ;*********************************************************************
                        ;
                        ;
                        ; FUNCTION CROUT
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: ECHO
                        ; DESTROYS: A,B,C,F/F'S
                        ; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
                        ;              FEED) TO THE CONSOLE.
                        ;
01F3                    CROUT:
01F3    0E0D                    MVI     C,CR
01F5    CDF901                  CALL    ECHO    ; OUTPUT CARRIAGE RETURN TO USER TERMINAL
01F8    C9                      RET
                        ;
                        ;
                        ;*********************************************************************
                        ;
                        ;
                        ; FUNCTION: ECHO
```

```
                        ; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
                        ; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
                        ; CALLS: CO
                        ; DESTROYS: A,B,F/F'S
                        ; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
                        ;              THE MONITOR, SENDS THAT CHARACTER TO THE USER
                        ;              TERMINAL.  A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
                        ;              RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS $.
                        ;
01F9                    ECHO:
01F9    41                    MOV     B,C       ; SAVE ARGUMENT
01FA    3E1B                  MVI     A,ESC
01FC    B8                    CMP     B         ; SEE IF ECHOING AN ESCAPE CHARACTER
01FD    C20202                JNZ     ECH05     ; NO - BRANCH
0200    0E24                  MVI     C,'$'     ; YES - ECHO AS $
0202                    ECH05:
0202    CDE801                CALL    CO        ; DO OUTPUT THROUGH MONITOR
0205    3E0D                  MVI     A,CR
0207    B8                    CMP     B         ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
0208    C21002                JNZ     ECH10     ; NO - NO NEED TO TAKE SPECIAL ACTION
020B    0E0A                  MVI     C,LF      ; YES - WANT TO ECHO LINE FEED, TOO
020D    CDE801                CALL    CO
0210                    ECH10:
0210    48                    MOV     C,B       ; RESTORE ARGUMENT
0211    C9                    RET
                        ;
                        ;
                        ;*********************************************************************
                        ;
                        ;
                        ; FUNCTION: ERROR
                        ; INPUTS: NONE
                        ; OUTPUTS: NONE
                        ; CALLS: ECHO,CROUT,GETCM
                        ; DESTROYS: A,B,C,F/F'S
                        ; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY AN ASTERISK)
                        ;              ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
                        ;              AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
                        ;
0212                    ERROR:
0212    0E23                  MVI     C,'#'
0214    CDF901                CALL    ECHO      ; SEND # TO CONSOLE
0217                    EXIT:
0217    CDF301                CALL    CROUT     ; SKIP TO BEGINNING OF NEXT LINE
021A    C32C00                JMP     GETCM     ; TRY AGAIN FOR ANOTHER COMMAND
                        ;
                        ;
                        ;*********************************************************************
                        ;
                        ;
                        ; FUNCTION: FRET
```

```
                ; INPUTS: NONE
                ; OUTPUTS: CARRY - ALWAYS 0
                ; CALLS: NOTHING
                ; DESTROYS: CARRY
                ; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
                ;              INDICATE FAILURE ON RETURN.  FRET SETS THE CARRY
                ;              FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
                ;              CALLER OF THE ROUTINE INVOKING FRET.
                ;
021D            FRET:
021D   37               STC             ; FIRST SET CARRY TRUE
021E   3F               CMC             ; THEN COMPLEMENT IT TO MAKE IT FALSE
021F   C9               RET             ; RETURN APPROPRIATELY
                ;
                ;
                ;***********************************************************************
                ;
                ;
                ; FUNCTION: GETCH
                ; INPUTS: NONE
                ; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
                ; CALLS: CI
                ; DESTROYS: A,C,F/F'S
                ; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
                ;              TO THE CALLING PROGRAM.
                ;
0220            GETCH:
0220   CDD501           CALL    CI      ; GET CHARACTER FROM TERMINAL
0223   E67F             ANI     PRTY0   ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
0225   4F               MOV     C,A     ; PUT VALUE IN C REGISTER FOR RETURN
0226   C9               RET
                ;
                ;
                ;***********************************************************************
                ;
                ;
                ; FUNCTION: GETHX
                ; INPUTS: NONE
                ; OUTPUTS: BC - 16 BIT INTEGER
                ;          D - CHARACTER WHICH TERMINATED THE INTEGER
                ;          CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
                ;                - 0 IF FIRST CHARACTER IS DELIMITER
                ; CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
                ; DESTROYS: A,B,C,D,E,F/F'S
                ; DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
                ;              STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
                ;              INTEGER.  IF MORE THAN 4 HEX DIGITS ARE ENTERED,
                ;              ONLY THE LAST 4 ARE USED.  THE NUMBER TERMINATES WHEN
                ;              A VALID DELIMITER IS ENCOUNTERED.  THE DELIMITER IS
                ;              ALSO RETURNED AS AN OUTPUT OF THE FUNCTION.  ILLEGAL
                ;              CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
```

```
                    ;               ERROR INDICATION.  IF THE FIRST (VALID) CHARACTER
                    ;               ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
                    ;               GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
                    ;               OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
                    ;               OF BC ARE UNDEFINED.
                    ;
0227                GETHX:
0227    E5                  PUSH    H           ; SAVE HL
0228    210000              LXI     H,0         ; INITIALIZE RESULT
022B    1E00                MVI     E,0         ; INITIALIZE DIGIT FLAG TO FALSE
022D                GHX05:
022D    CD2302              CALL    GETCH       ; GET A CHARACTER
0230    CDF901              CALL    ECHO        ; ECHO THE CHARACTER
0233    CD8203              CALL    VALDL       ; SEE IF DELIMITER
     1            +         FALSE   GHX10       ; NO - BRANCH
0236 1  D24502    +         JNC     GHX10
0239    51                  MOV     D,C         ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
023A    E5                  PUSH    H
023B    C1                  POP     B           ; MOVE RESULT TO BC
023C    E1                  POP     H           ; RESTORE HL
023D    7B                  MOV     A,E         ; GET FLAG
023E    B7                  ORA     A           ; SET F/F'S
023F    C23B03              JNZ     SRET        ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
0242    CA1D02              JZ      FRET        ; ELSE, DELIMITER WAS FIRST CHARACTER
0245                GHX10:
0245    CD6703              CALL    VALDG       ; IF NOT DELIMITER, SEE IF DIGIT
     1            +         FALSE   ERROR       ; ERROR IF NOT A VALID DIGIT, EITHER
0248 1  D21202    +         JNC     ERROR
024B    CDDF01              CALL    CNVBN       ; CONVERT DIGIT TO ITS BINARY VALUE
024E    1EFF                MVI     E,0FFH      ; SET DIGIT FLAG NON-0
0250    29                  DAD     H           ; *2
0251    29                  DAD     H           ; *4
0252    29                  DAD     H           ; *8
0253    29                  DAD     H           ; *16
0254    0600                MVI     B,0         ; CLEAR UPPER 8 BITS OF BC PAIR
0256    4F                  MOV     C,A         ; BINARY VALUE OF CHARACTER INTO C
0257    09                  DAD     B           ; ADD THIS VALUE TO PARTIAL RESULT
0258    C32D02              JMP     GHX05       ; GET NEXT CHARACTER
                    ;
                    ;
                    ;*****************************************************************
                    ;
                    ;
                    ; FUNCTION: GETNM
                    ; INPUTS: C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
                    ; OUTPUTS: TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP
                    ;                          OF STACK)
                    ; CALLS: GETHX,HILO,ERROR
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
                    ;               AND 3, INCLUSIVE,  IN THE INPUT
```

```
                   ;           STREAM AND RETURNS THEIR VALUES ON THE STACK.  IF 2
                   ;           OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
                   ;           LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
                   ;           SECOND NUMBERS WILL BE SET EQUAL.  THE LAST NUMBER
                   ;           REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
                   ;           OR AN ERROR INDICATION WILL RESULT.
                   ;
025B               GETNM:
025B    2E03             MVI     L,3       ; PUT MAXIMUM ARGUMENT COUNT INTO L
025D    79               MOV     A,C       ; GET THE ACTUAL ARGUMENT COUNT
025E    E603             ANI     3         ; FORCE TO MAXIMUM OF 3
0260    C8               RZ                ; IF 0, DON'T BOTHER TO DO ANYTHIING
0261    67               MOV     H,A       ; ELSE, PUT ACTUAL COUNT INTO H
0262               GNM05:
0262    CD2702           CALL    GETHX     ; GET A NUMBER FROM INPUT STREAM
      1            +     FALSE   ERROR     ; ERROR IF NOT THERE - TOO FEW NUMBERS
0265  1 D21202     +     JNC     ERROR
0268    C5               PUSH    B         ; ELSE, SAVE NUMBER ON STACK
0269    2D               DCR     L         ; DECREMENT MAXIMUM ARGUMENT COUNT
026A    25               DCR     H         ; DECREMENT ACTUAL ARGUMENT COUNT
026B    CA7702           JZ      GNM10     ; BRANCH IF NO MORE NUMBERS WANTED
026E    7A               MOV     A,D       ; ELSE, GET NUMBER TERMINATOR TO A
026F    FE0D             CPI     CR        ; SEE IF CARRIAGE RETURN
0271    CA1202           JZ      ERROR     ; ERROR IF SO - TOO FEW NUMBERS
0274    C36202           JMP     GNM05     ; ELSE, PROCESS NEXT NUMBER
0277               GNM10:
0277    7A               MOV     A,D       ; WHEN COUNT 0, CHECK LAST TERMINATOR
0278    FE0D             CPI     CR
027A    C21202           JNZ     ERROR     ; ERROR IF NOT CARRIAGE RETURN
027D    01FFFF           LXI     B,0FFFFH          ; HL GETS LARGEST NUMBER
0280    7D               MOV     A,L       ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
0281    B7               ORA     A         ; CHECK FOR 0
0282    CA8A02           JZ      GNM20     ; IF YES, 3 NUMBERS WERE INPUT
0285               GNM15:
0285    C5               PUSH    B         ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
0286    2D               DCR     L
0287    C28502           JNZ     GNM15
028A               GNM20:
028A    C1               POP     B         ; GET THE 3 ARGUMENTS OUT
028B    D1               POP     D
028C    E1               POP     H
028D    CDA002           CALL    HILO      ; SEE IF FIRST >= SECOND
      1            +     FALSE   GNM25     ; NO - BRANCH
0290  1 D29502     +     JNC     GNM25
0293    54               MOV     D,H
0294    5D               MOV     E,L       ; YES - MAKE SECOND EQUAL TO THE FIRST
0295               GNM25:
0295    E3               XTHL              ; PUT FIRST ON STACK - GET RETURN ADDR
0296    D5               PUSH    D         ; PUT SECOND ON STACK
0297    C5               PUSH    B         ; PUT THIRD ON STACK
0298    E5               PUSH    H         ; PUT RETURN ADDRESS ON STACK
```

```
0299                 GNM30:
0299    3D                  DCR     A        ; DECREMENT RESIDUAL COUNT
029A    F8                  RM               ; IF NEGATIVE, PROPER RESULTS ON STACK
029B    E1                  POP     H        ; ELSE, GET RETURN ADDR
029C    E3                  XTHL             ; REPLACE TOP RESULT WITH RETURN ADDR
029D    C39902              JMP     GNM30    ; TRY AGAIN
                     ;
                     ;
                     ;*****************************************************************
                     ;
                     ;
                     ; FUNCTION: HILO
                     ; INPUTS: DE - 16 BIT INTEGER
                     ;         HL - 16 BIT INTEGER
                     ; OUTPUTS: CARRY - 0 IF HL<DE
                     ;                - 1 IF HL>=DE
                     ; CALLS: NOTHING
                     ; DESTROYS: A,F/F'S
                     ; DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE.  THE
                     ;              INTEGERS ARE TREATED AS UNSIGNED NUMBERS.  THE CARRY
                     ;              BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.
                     ;
02A0                 HILO:
02A0    C5                  PUSH    B        ; SAVE BC
02A1    47                  MOV     B,A      ; SAVE A REGISTER
02A2    23                  INX     H        ; INCREMENT HL BY 1
02A3    7C                  MOV     A,H      ; WANT TO TEST FOR 0 RESULT AFTER
02A4    B5                  ORA     L        ; /INCREMENTING
02A5    2B                  DCX     H        ; RESTORE HL
02A6    37                  STC              ; SET CARRY
02A7    CAAF02              JZ      HIL05    ; IF SO, CARRY IS SET PROPERLY
02AA    7D                  MOV     A,L      ; IF NOT, MOVE L TO A
02AB    93                  SUB     E        ; SUBTRACT E
02AC    7C                  MOV     A,H      ; MOVE H TO A
02AD    9A                  SBB     D        ; SUBTRACT D WITH BORROW
02AE    3F                  CMC              ; COMPLIMENT CARRY FOR CORRECT CARRY BIT VALUE
02AF                 HIL05:
02AF    78                  MOV     A,B      ; RESTORE A
02B0    C1                  POP     B        ; RESTORE BC
02B1    C9                  RET              ; EXIT
                     ;
                     ;
                     ;*****************************************************************
                     ;
                     ;
                     ; FUNCTION INUST
                     ; INPUTS: NONE
                     ; OUTPUTS: NOTHING
                     ; CALLS: NOTHING
                     ; DESTROYS: A,H,L,SP
                     ; DESCRIPTION: INUST OUTPUTS TO THE USART THE COMMAND WORD
```

```
                        ;               AND INITIALIZES THE STACK POINTER.
                        ;
 Ø2B2                   INUST:
 Ø2B2     3E25                MVI     A,CMD
 Ø2B4     D3ED                OUT     CNCTL   ; OUTPUT COMMAND WORD TO USART
 Ø2B6     21Ø23C              LXI     H,MSTAK-44       ; LOAD POINTER TO STACK
 Ø2B9     22383C              SHLD    SSAVE   ; INITIALIZE USER STACK POINTER
 Ø2BC     312E3C              LXI     SP,MSTAK         ; INITIALIZE MONITOR STACK
 Ø2BF     C31EØØ              JMP     SOMSG   ; GO TO PRINT SIGNON MESSAGE
                        ;
                        ;
                        ;**********************************************************************
                        ;
                        ;
                        ; FUNCTION: NMOUT
                        ; INPUTS: A - 8 BIT INTEGER
                        ; OUTPUTS: NONE
                        ; CALLS: ECHO,PRVAL
                        ; DESTROYS: A,B,C,F/F'S
                        ; DESCRIPTION: NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
                        ;              A REGISTER INTO 2 ASCII CHARACTERS.  THE ASCII CHARACTERS
                        ;              ARE THE ONES REPRESENTING THE 8 BITS.  THESE TWO
                        ;              CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
                        ;              POSITION OF THE CONSOLE.
                        ;
 Ø2C2                   NMOUT:
 Ø2C2     F5                  PUSH    PSW     ; SAVE ARGUMENT
 Ø2C3     ØF                  RRC
 Ø2C4     ØF                  RRC
 Ø2C5     ØF                  RRC
 Ø2C6     ØF                  RRC             ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
 Ø2C7     CDD5Ø2              CALL    PRVAL   ;CONVERT LOWER 4 BITS TO ASCII
 Ø2CA     CDF9Ø1              CALL    ECHO    ; SEND TO TERMINAL
 Ø2CD     F1                  POP     PSW     ; GET BACK ARGUMENT
 Ø2CE     CDD5Ø2              CALL    PRVAL
 Ø2D1     CDF9Ø1              CALL    ECHO
 Ø2D4     C9                  RET
                        ;
                        ;
                        ;**********************************************************************
                        ;
                        ;
                        ; FUNCTION;  PRVAL
                        ; INPUTS: A - INTEGER, RANGE Ø TO F
                        ; OUTPUTS: A - ASCII CHARACTER
                        ; CALLS: NOTHING
                        ; DESTROYS: NOTHING
                        ; DESCRIPTION: PRVAL CONVERTS A NUMBER IN THE RANGE Ø TO F HEX TO
                        ;              THE CORRESPONDING ASCII CHARACTER, Ø-9,A-F.  PRVAL
                        ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.
                        ;
```

```
02D5                PRVAL:
Ø2D5    E6ØF              ANI     HCHAR    ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
Ø2D7    C69Ø              ADI     9ØH      ; SET UP A SO THAT A-F CAUSE A CARRY
Ø2D9    27                DAA              ; ADJUST CONTENTS OF A REGISTER
Ø2DA    CE4Ø              ACI     4ØH      ; ADD IN CARRY AND ADJUST UPPER 4 BITS
Ø2DC    27                DAA              ; ADJUST CONTENTS OF A REGISTER AGAIN
Ø2DD    4F                MOV     C,A      ; MOVE ASCII CHARACTER TO C
Ø2DE    C9                RET              ; ALL DONE
                    ;
                    ;*********************************************************************
                    ;
                    ;
                    ; FUNCTION: REGDS
                    ; INPUTS: NONE
                    ; OUTPUTS: NONE
                    ; CALLS: ECHO,NMOUT,ERROR,CROUT
                    ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    ; DESCRIPTION: REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE
                    ;              LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE.  THE
                    ;              DISPLAY IS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS
                    ;              THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS,
                    ;              AND LENGTH (8 OR 16 BITS).
                    ;
Ø2DF                REGDS:
Ø2DF    21CØØ3            LXI     H,RTAB   ; LOAD HL WITH ADDRESS OF START OF TABLE
Ø2E2                REGØ5:
Ø2E2    4E                MOV     C,M      ; GET PRINT SYMBOL OF REGISTER
Ø2E3    79                MOV     A,C
Ø2E4    B7                ORA     A        ; TEST FOR Ø - END OF TABLE
Ø2E5    C2ECØ2            JNZ     REG1Ø    ; IF NOT END, BRANCH
Ø2E8    CDF3Ø1            CALL    CROUT    ; ELSE, CARRIAGE RETURN/LINE FEED TO END
Ø2EB    C9                RET              ; /DISPLAY
Ø2EC                REG1Ø:
Ø2EC    CDF9Ø1            CALL    ECHO     ; ECHO CHARACTER
Ø2EF    ØE3D              MVI     C,'='
Ø2F1    CDF9Ø1            CALL    ECHO     ; OUTPUT EQUALS SIGN, I.E. A=
Ø2F4    23                INX     H        ; POINT TO START OF SAVE LOCATION ADDRESS
Ø2F5    5E                MOV     E,M      ; GET LSP OF SAVE LOCATION ADDRESS TO E
Ø2F6    163C              MVI     D,DATA SHR 8     ; PUT MSP OF SAVE LOC ADDRESS INTO D
Ø2F8    23                INX     H        ; POINT TO LENGTH FLAG
Ø2F9    1A                LDAX    D        ; GET CONTENTS OF SAVE ADDRESS
Ø2FA    CDC2Ø2            CALL    NMOUT    ; DISPLAY ON CONSOLE
Ø2FD    7E                MOV     A,M      ; GET LENGTH FLAG
Ø2FE    B7                ORA     A        ; SET SIGN F/F
Ø2FF    CAØ7Ø3            JZ      REG15    ; IF Ø, REGISTER IS 8 BITS
Ø3Ø2    1B                DCX     D        ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
Ø3Ø3    1A                LDAX    D        ; GET LOWER 8 BITS
Ø3Ø4    CDC2Ø2            CALL    NMOUT    ; DISPLAY THEM
Ø3Ø7                REG15:
Ø3Ø7    ØE2Ø              MVI     C,' '
```

```
0309   CDF901          CALL    ECHO     ; OUTPUT BLANK CHARACTER
030C   23              INX     H        ; POINT TO START OF NEXT TABLE ENTRY
030D   C3E202          JMP     REG05    ; DO NEXT REGISTER
                ;
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION: RGADR
                ; INPUTS: C - CHARACTER DENOTING REGISTER
                ; OUTPUTS: BC - ADDRESS OF ENTRY IN RTAB CORRESPONDING TO REGISTER
                ; CALLS: ERROR
                ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                ; DESCRIPTION: RGADR TAKES A SINGLE CHARACTER AS INPUT.  THIS CHARACTER
                ;              DENOTES A REGISTER.  RGADR SEARCHES THE TABLE RTAB
                ;              FOR A MATCH ON THE INPUT ARGUMENT.  IF ONE OCCURS,
                ;              RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE
                ;              SAVE LOCATION CORRESPONDING TO THE REGISTER.  THIS
                ;              ADDRESS POINTS INTO RTAB.  IF NO MATCH OCCURS, THEN
                ;              THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
                ;              PASSED TO THE ERROR ROUTINE.
                ;
0310            RGADR:
0310   21C003          LXI     H,RTAB   ; HL GETS ADDRESS OF TABLE START
0313   110300          LXI     D,RTABS  ; DE GET SIZE OF A TABLE ENTRY
0316            RGA05:
0316   7E              MOV     A,M      ; GET REGISTER IDENTIFIER
0317   B7              ORA     A        ; CHECK FOR TABLE END (IDENTIFIER IS 0)
0318   CA1202          JZ      ERROR    ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
031B   B9              CMP     C        ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
031C   CA2303          JZ      RGA10    ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
031F   19              DAD     D        ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
0320   C31603          JMP     RGA05    ; TRY AGAIN
0323            RGA10:
0323   23              INX     H        ; IF A MATCH, INCREMENT TABLE POINTER TO
0324   44              MOV     B,H      ; /SAVE LOCATION ADDRESS
0325   4D              MOV     C,L      ; RETURN THIS VALUE
0326   C9              RET
                ;
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION: RSTTF
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: NOTHING
                ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                ; DESCRIPTION: RSTTF RESTORES ALL CPU REGISTER, FLIP/FLOPS, STACK
                ;              POINTER AND PROGRAM COUNTER FROM THEIR RESPECTIVE
                ;              SAVE LOCATIONS IN MEMORY.  THE ROUTINE THEN TRANSFERS
```

```
                ;               CONTROL TO THE LOCATION SPECIFIED BY THE PROGRAM
                ;               COUNTER (I.E. THE RESTORED VALUE).  THE ROUTINE
                ;               EXITS WITH THE INTERRUPTS ENABLED.
                ;
0327            RSTTF:
0327    F3              DI              ; DISABLE INTERRUPTS WHILE RESTORING THINGS
0328    312E3C          LXI     SP,MSTAK        ; SET MONITOR STACK POINTER TO START
                                        ; /OF STACK
032B    D1              POP     D       ; START ALSO END OF REGISTER SAVE AREA
032C    C1              POP     B
032D    F1              POP     PSW
032E    2A383C          LHLD    SSAVE   ; RESTORE USER STACK POINTER
0331    F9              SPHL
0332    2A363C          LHLD    PSAVE
0335    E5              PUSH    H       ; PUT USER RETURN ADDRESS ON USER STACK
0336    2A343C          LHLD    LSAVE   ; RESTORE HL REGISTERS
0339    FB              EI              ; ENABLE INTERRUPTS NOW
033A    C9              RET             ; JUMP TO RESTORED PC LOCATION
                ;
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION: SRET
                ; INPUTS: NONE
                ; OUTPUTS: CARRY = 1
                ; CALLS: NOTHING
                ; DESTROYS: CARRY
                ; DESCRIPTION: SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.
                ;               SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE
                ;               CALLER OF THE ROUTINE INVOKING SRET.
                ;
033B            SRET:
033B    37              STC             ; SET CARRY TRUE
033C    C9              RET             ; RETURN APPROPRIATELY
                ;
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION: STHF0
                ; INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                ; OUTPUTS: NONE
                ; CALLS: NOTHING
                ; DESTROYS: A,B,C,H,L,F/F'S
                ; DESCRIPTION: STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
                ;               IT IS SET TO LOWER.  IF SO, STHF0 STORES A 0 TO
                ;               PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE;
                ;               OTHERWISE, THE ROUTINE TAKES NO ACTION.
                ;
033D            STHF0:
```

```
033D  3A3A3C        LDA   TEMP    ; GET HALF BYTE FLAG
0340  B7            ORA   A       ; SET F/F'S
0341  C0            RNZ           ; IF SET TO UPPER, DON'T DO ANYTHING
0342  0E00          MVI   C,0     ; ELSE, WANT TO STORE THE VALUE 0
0344  CD4803        CALL  STHLF   ; DO IT
0347  C9            RET
                ;
                ;
                ;*************************************************************
                ;
                ;
                ; FUNCTION: STHLF
                ; INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE
                ;         DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                ; OUTPUTS: NONE
                ; CALLS: NOTHING
                ; DESTROYS: A,B,C,H,L,F/F'S
                ; DESCRIPTION: STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN
                ;              HALF OF THE BYTE ADDRESSED BY REGISTERS DE.  THE
                ;              HALF BYTE USED (EITHER UPPER OR LOWER) IS DENOTED
                ;              BY THE VALUE OF THE FLAG IN TEMP.  STHLF ASSUMES
                ;              THAT THIS FLAG HAS BEEN PREVIOUSLY SET
                ;              (NOMINALLY BY ICMD).
                ;
0348            STHLF:
0348  D5            PUSH  D
0349  E1            POP   H       ; MOVE ADDRESS OF BYTE INTO HL
034A  79            MOV   A,C     ; GET VALUE
034B  E60F          ANI   0FH     ; FORCE TO 4 BIT LENGTH
034D  4F            MOV   C,A     ; PUT VALUE BACK
034E  3A3A3C        LDA   TEMP    ; GET HALF BYTE FLAG
0351  B7            ORA   A       ; CHECK FOR LOWER HALF
0352  C25B03        JNZ   STH05   ; BRANCH IF NOT
0355  7E            MOV   A,M     ; ELSE, GET BYTE
0356  E6F0          ANI   0F0H    ; CLEAR LOWER 4 BITS
0358  B1            ORA   C       ; OR IN VALUE
0359  77            MOV   M,A     ; PUT BYTE BACK
035A  C9            RET
035B            STH05:
035B  7E            MOV   A,M     ; IF UPPER HALF, GET BYTE
035C  E60F          ANI   0FH     ; CLEAR UPPER 4 BITS
035E  47            MOV   B,A     ; SAVE BYTE IN B
035F  79            MOV   A,C     ; GET VALUE
0360  0F            RRC
0361  0F            RRC
0362  0F            RRC
0363  0F            RRC           ; ALIGN TO UPPER 4 BITS
0364  B0            ORA   B       ; OR IN ORIGINAL LOWER 4 BITS
0365  77            MOV   M,A     ; PUT NEW CONFIGURATION BACK
0366  C9            RET
```

```
                          ;
                          ;***********************************************************************
                          ;
                          ;
                          ; FUNCTION: VALDG
                          ; INPUTS: C - ASCII CHARACTER
                          ; OUTPUTS: CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
                          ;                - 0 OTHERWISE
                          ; CALLS: NOTHING
                          ; DESTROYS: A,F/F'S
                          ; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
                          ;              AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
                          ;              (0-9,A-F), AND FAILURE OTHERWISE.
                          ;
0367                      VALDG:
0367    79                      MOV     A,C
0368    FE30                    CPI     '0'     ; TEST CHARACTER AGAINST '0'
036A    FA1D02                  JM      FRET    ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
036D    FE39                    CPI     '9'     ; ELSE, SEE IF IN RANGE '0'-'9'
036F    FA3B03                  JM      SRET    ; CODE BETWEEN '0' AND '9'
0372    CA3B03                  JZ      SRET    ; CODE EQUAL '9'
0375    FE41                    CPI     'A'     ; NOT A DIGIT - TRY FOR A LETTER
0377    FA1D02                  JM      FRET    ; NO - CODE BETWEEN '9' AND 'A'
037A    FE47                    CPI     'G'
037C    F21D02                  JP      FRET    ; NO - CODE GREATER THAN 'F'
037F    C33B03                  JMP     SRET    ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
                          ;
                          ;
                          ;***********************************************************************
                          ;
                          ;
                          ; FUNCTION: VALDL
                          ; INPUTS: C - CHARACTER
                          ; OUTPUTS: CARRY - 1 IF INPUT ARGUMENT VALID DELIMTER
                          ;                - 0 OTHERWISE
                          ; CALLS: NOTHING
                          ; DESTROYS: A,F/F'S
                          ; DESCRIPTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
                          ;              DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN) AND
                          ;              FAILURE OTHERWISE.
                          ;
0382                      VALDL:
0382    79                      MOV     A,C
0383    FE2C                    CPI     ','     ; CHECK FOR COMMA
0385    CA3B03                  JZ      SRET
0388    FE0D                    CPI     CR      ; CHECK FOR CARRIAGE RETURN
038A    CA3B03                  JZ      SRET
038D    FE20                    CPI     ' '     ; CHECK FOR SPACE
038F    CA3B03                  JZ      SRET
0392    C31D02                  JMP     FRET    ; ERROR IF NONE OF THE ABOVE
```

```
                         ;
                         ;*****************************************************************
                         ;
                         ;
                         ;                              MONITOR TABLES
                         ;
                         ;
                         ;*****************************************************************
                         ;
                         ;
 Ø395                    SGNON:                         ; SIGNON MESSAGE
 Ø395     ØDØA383Ø             DB      CR,LF,'8Ø/1Ø MONITOR',CR,LF
 Ø399     2F31302Ø
 Ø39D     4D4F4E49
 Ø3A1     544F52ØD
 Ø3A5     ØA
 ØØ11                    LSGNON        EQU     $-SGNON ; LENGTH OF SIGNON MESSAGE
                         ;
 Ø3A6                    CADR:                  ; TABLE OF ADDRESSES OF COMMAND ROUTINES
 Ø3A6     ØØØØ                 DW      Ø        ; DUMMY
 Ø3A8     33Ø1                 DW      XCMD
 Ø3AA     CFØ1                 DW      SCMD
 Ø3AC     EFØØ                 DW      MCMD
 Ø3AE     A9ØØ                 DW      ICMD
 Ø3BØ     8BØØ                 DW      GCMD
 Ø3B2     5FØØ                 DW      DCMD
 Ø3B4     Ø6Ø4                 DW      RCMD
 Ø3B6     41Ø4                 DW      WCMD
                         ;
 Ø3B8                    CTAB:                  ; TABLE OF VALID COMMAND CHARACTERS
 Ø3B8     57                   DB      'W'
 Ø3B9     52                   DB      'R'
 Ø3BA     44                   DB      'D'
 Ø3BB     47                   DB      'G'
 Ø3BC     49                   DB      'I'
 Ø3BD     4D                   DB      'M'
 Ø3BE     53                   DB      'S'
 Ø3BF     58                   DB      'X'
 ØØØ8                    NCMDS EQU     $-CTAB   ; NUMBER OF VALID COMMANDS
                         ;
                         ;
 Ø3CØ                    RTAB:                  ; TABLE OF REGISTER INFORMATION
 Ø3CØ     41                   DB      'A'      ; REGISTER IDENTIFIER
 Ø3C1     33                   DB      ASAVE AND ØFFH  ; ADDRESS OF REGISTER SAVE LOCATION
 Ø3C2     ØØ                   DB      Ø        ; LENGTH FLAG - Ø=8 BITS, 1=16 BITS
 ØØØ3                    RTABS EQU     $-RTAB   ; SIZE OF AN ENTRY IN THIS TABLE
 Ø3C3     42                   DB      'B'
 Ø3C4     31                   DB      BSAVE AND ØFFH
 Ø3C5     ØØ                   DB      Ø
 Ø3C6     43                   DB      'C'
 Ø3C7     3Ø                   DB      CSAVE AND ØFFH
```

```
03C8   00                 DB      0
03C9   44                 DB      'D'
03CA   2F                 DB      DSAVE AND 0FFH
03CB   00                 DB      0
03CC   45                 DB      'E'
03CD   2E                 DB      ESAVE AND 0FFH
03CE   00                 DB      0
03CF   46                 DB      'F'
03D0   32                 DB      FSAVE AND 0FFH
03D1   00                 DB      0
03D2   48                 DB      'H'
03D3   35                 DB      HSAVE AND 0FFH
03D4   00                 DB      0
03D5   4C                 DB      'L'
03D6   34                 DB      LSAVE AND 0FFH
03D7   00                 DB      0
03D8   4D                 DB      'M'
03D9   35                 DB      HSAVE AND 0FFH
03DA   01                 DB      1
03DB   50                 DB      'P'
03DC   37                 DB      PSAVE+1 AND 0FFH
03DD   01                 DB      1
03DE   53                 DB      'S'
03DF   39                 DB      SSAVE+1 AND 0FFH
03E0   01                 DB      1
03E1   00                 DB      0        ; END OF TABLE MARKERS
03E2   00                 DB      0
                  ;
                  ;
                  ;*********************************************************************
                  ;
                  ;
03E3              CPYRT:
03E3   28432920           DB      '(C) 1976 INTEL CORP'
03E7   31393736
03EB   20494E54
03EF   454C2043
03F3   4F5250
                  ;
                  ;
                  ;*********************************************************************
                  ;
                  ;
                  ;
03FA                      ORG     BRTAB
                  ;
03FA   C3E801             JMP     CO       ; BRANCH TABLE FOR USER ACCESSIBLE ROUTINES
03FD   C3D501             JMP     CI
0400   C31C05             JMP     RI
0403   C30F05             JMP     PO
                  ;
```

```
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION RCMD
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: GETCH,ECHO,CO,RICH,BYTE
                ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                ; DESCRIPTION: RCMD IMPLEMENTS THE READ HEXADECIMAL TAPE (R)
                ;              COMMAND.
                ;
0406            RCMD:
0406  CD2002          CALL    GETCH   ; GET CARRIAGE RETURN CHARACTER
0409  CDF901          CALL    ECHO    ; ECHO IT
040C  79              MOV     A,C     ; MOVE IT TO A REGISTER
040D  FE0D            CPI     CR      ; SEE IF CARRIAGE RETURN
040F  C21202          JNZ     ERROR   ; ERROR IF NOT PROPERLY TERMINATED
0412            RCM05:
0412  CD1305          CALL    RICH    ; READ CHARACTER FROM TAPE
0415  FE3A            CPI     ':'     ; SEE IF RECORD MARK
0417  C21204          JNZ     RCM05   ; TRY AGAIN IF NOT MARK
041A  AF              XRA     A       ; ZERO A REGISTER
041B  57              MOV     D,A     ; INITIALIZE D FOR HOLDING THE CHECKSUM
041C  CD9604          CALL    BYTE    ; READ TWO CHARACTERS FROM TAPE
041F  CA2C00          JZ      GETCM   ; IF ZERO RECORD LENGTH, ALL DONE
0422  5F              MOV     E,A     ; OTHERWISE, PUT THE RECORD LENGTH IN
0423  CD9604          CALL    BYTE    ; GET MSB OF LOAD ADDRESS
0426  67              MOV     H,A     ; MOVE TO H
0427  CD9604          CALL    BYTE    ; GET LSB OF LOAD ADDRESS
042A  6F              MOV     L,A     ; MOVE TO L
042B  CD9604          CALL    BYTE    ; GET RECORD TYPE
042E  4B              MOV     C,E     ; MOVE RECORD LENGTH TO C
042F            RCM10:
042F  CD9604          CALL    BYTE    ; READ DATA FROM TAPE
0432  77              MOV     M,A     ; PUT DATA INTO MEMORY
0433  23              INX     H       ; INCREMENT HL FOR NEXT LOCATION
0434  1D              DCR     E       ; DECREMENT RECORD LENGTH
0435  C22F04          JNZ     RCM10   ; LOOP UNTIL DONE
0438  CD9604          CALL    BYTE    ; READ CHECKSUM FROM TAPE
043B  C21202          JNZ     ERROR   ; CHECKSUM ERROR IF NOT ZERO
043E  C31204          JMP     RCM05   ; GET ANOTHER RECORD
                ;
                ;
                ;*****************************************************************
                ;
                ;
                ; FUNCTION WCMD
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: GETNM,LEAD,PO,PBYTE,PADR,PEOL,PEOP
```

```
                        ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                        ; DESCRIPTION: WCMD IMPLEMENTS THE WRITE HEXADECIMAL TAPE (W)
                        ;              COMMAND.
                        ;
0441                    WCMD:
0441   0E02                    MVI    C,2
0443   CD5B02                  CALL   GETNM    ; GET 2 NUMBERS FROM INPUT STREAM
0446   CDBA04                  CALL   LEAD     ; PUNCH 60 NULL CHARACTERS FOR TAPE LEADER
0449   D1                      POP    D        ; ENDING ADDRESS TO DE
044A   E1                      POP    H        ; STARTING ADDRESS TO HL
044B                    WCM05:
044B   7D                      MOV    A,L      ; MOVE L TO A
044C   C610                    ADI    16       ; INCREMENT THE LSB OF STARTING ADDRESS BY 16
044E   4F                      MOV    C,A      ; MOVE RESULT TO C
044F   7C                      MOV    A,H      ; MOVE H TO A
0450   CE00                    ACI    0        ; ADD CARRY IN FROM PREVIOUS OPERATION
0452   47                      MOV    B,A      ; SAVE RESULT IN B
0453   7B                      MOV    A,E      ; NOW MOVE LSB OF ENDING ADDRESS TO A
0454   91                      SUB    C        ; SUBTRACT LSB OF STARTING ADDRESS
0455   4F                      MOV    C,A      ; SAVE IN C
0456   7A                      MOV    A,D      ; NOW GET MSB OF ENDING ADDRESS IN A
0457   98                      SBB    B        ; SUBTRACT MSB OF STARTING ADDRESS
0458   DA6004                  JC     WCM10    ; BRANCH IF THE RECORD LENGTH IS NOT 16
045B   3E10                    MVI    A,16     ; OTHERWISE SET A TO RECORD LENGTH OF 16
045D   C36304                  JMP    WCM15    ; NOW BRANCH TO PUNCH THE RECORD
0460                    WCM10:
0460   79                      MOV    A,C      ; THIS IS THE LAST RECORD
0461   C611                    ADI    17       ; SO SET A TO REMAINING DATA LENGTH
0463                    WCM15:
0463   B7                      ORA    A        ; CHECK FOR RECORD LENGTH OF ZERO
0464   CA9004                  JZ     WCM25    ; IF IT IS, ALL DONE
0467   D5                      PUSH   D        ; OTHERWISE, SAVE ENDING ADDRESS
0468   5F                      MOV    E,A      ; PUT RECORD LENGTH IN E
0469   1600                    MVI    D,0      ; INITIALIZE D FOR HOLDING CHECKSUM
046B   0E3A                    MVI    C,':'
046D   CD0F05                  CALL   PO       ; PUNCH RECORD MARK CHARACTER
0470   7B                      MOV    A,E      ; PUT RECORD LENGTH IN A
0471   CDCF04                  CALL   PBYTE    ; PUNCH RECORD LENGTH
0474   CDC604                  CALL   PADR     ; PUNCH STARTING ADDRESS
0477   AF                      XRA    A        ; ZERO A
0478   CDCF04                  CALL   PBYTE    ; PUNCH RECORD TYPE
047B                    WCM20:
047B   7E                      MOV    A,M      ; GET DATA TO BE PUNCHED FROM MEMORY
047C   CDCF04                  CALL   PBYTE    ; PUNCH IT
047F   23                      INX    H        ; INCREMENT MEMORY ADDRESS
0480   1D                      DCR    E        ; DECREMENT LENGTH COUNT
0481   C27B04                  JNZ    WCM20    ; LOOP UNTIL ALL DATA PUNCHED
0484   AF                      XRA    A
0485   92                      SUB    D        ; PUNCH CHECKSUM
0486   CDCF04                  CALL   PBYTE
0489   D1                      POP    D        ; RESTORE ENDING ADDRESS
```

```
048A   CD0405        CALL    PECL     ; PUNCH CARRIAGE RETURN AND LINE FEED
048D   C34B04        JMP     WCM05
0490           WCM25:
0490   CDE604        CALL    PEOF     ; PUNCH END OF FILE RECORD
0493   C31702        JMP     EXIT     ; ALL DONE
               ;
               ;
               ;******************************************************************
               ;
               ;
               ; FUNCTION BYTE
               ; INPUTS: D - CURRENT VALUE OF CHECKSUM
               ; OUTPUTS: A - HEXADECIMAL CHARACTER
               ;          D - UPDATED VALUE OF CHECKSUM
               ; CALLS: RICH,CNVBN
               ; DESTROYS: A,B,C,D,F/F'S
               ; DESCRIPTION: BYTE READS 2 ASCII CHARACTERS FROM THE TELETYPEWRITER
               ;              AND CONVERTS THE CHARACTERS TO ONE HEXADECIMAL CHARACTER.
               ;              THE A REGISTER CONTAINS THE FINAL CHARACTER AND THE
               ;              D REGISTER CONTAINS THE UPDATED VALUE OF
               ;              THE CHECKSUM.
               ;
0496           BYTE:
0496   C5            PUSH    B        ; SAVE BC
0497   CD1305        CALL    RICH     ; READ ASCII CHARACTER FROM TAPE
049A   4F            MOV     C,A
049B   CDDF01        CALL    CNVBN    ; CONVERT CHARACTER TO HEXADECIMAL
049E   07            RLC              ; POSITION VALUE INTO UPPER 4 BITS
049F   07            RLC
04A0   07            RLC
04A1   07            RLC
04A2   47            MOV     B,A      ; SAVE RESULTS IN B
04A3   CD1305        CALL    RICH     ; GET ANOTHER CHARACTER FROM TAPE
04A6   4F            MOV     C,A
04A7   CDDF01        CALL    CNVBN    ; CONVERT IT
04AA   B0            ORA     B        ; OR IN THE UPPER 4 BITS
04AB   4F            MOV     C,A      ; SAVE
04AC   82            ADD     D        ; INCREMENT CHECKSUM
04AD   57            MOV     D,A
04AE   79            MOV     A,C      ; RESTORE HEX DATA TO A REGISTER
04AF   C1            POP     B        ; RESTORE BC
04B0   C9            RET
               ;
               ;
               ;******************************************************************
               ;
               ;
               ; FUNCTION DELAY
               ; INPUTS: NONE
               ; OUTPUTS: NONE
               ; CALLS: NOTHING
```

```
                      ; DESTROYS: F/F'S
                      ; DESCRIPTION: DELAY PROVIDES A PROGRAMMED DELAY OF 1 MILLISECOND
                      ;              FOR TAPE READER OPERATION.
                      ;
04B1                  DELAY:
04B1    C5                  PUSH    B        ; SAVE BC REGISTERS
04B2    0683                MVI     B,ONEMS  ; LOAD 1 MILLISECOND CONSTANT
04B4                  DEL1:
04B4    05                  DCR     B        ; DECREMENT INNER COUNTER
04B5    C2B404              JNZ     DEL1     ; JUMP IF NOT DONE
04B8    C1                  POP     B        ; RESTORE BC REGISTERS
04B9    C9                  RET              ; RETURN TO CALLING ROUTINE
                      ;
                      ;
                      ;*******************************************************************
                      ;
                      ;
                      ; FUNCTION LEAD
                      ; INPUTS: NONE
                      ; OUTPUTS: NONE
                      ; CALLS: PO
                      ; DESTROYS: B,C,F/F'S
                      ; DESCRIPTIOM: LEAD OUTPUTS 60 NULL CHARACTERS TO PAPER TAPE TO FORM A
                      ;              LEADER.
                      ;
04BA                  LEAD:
04BA    063C                MVI     B,60     ; LOAD B WITH A COUNT OF 60
04BC                  LE05:
04BC    0E00                MVI     C,0
04BE    CD0F05              CALL    PO       ; PUNCH NULL CHARACTER
04C1    05                  DCR     B        ; DECREMENT COUNT
04C2    C2BC04              JNZ     LE05     ; DO IT AGAIN IF NOT DONE
04C5    C9                  RET
                      ;
                      ;
                      ;
                      ;*******************************************************************
                      ;
                      ;
                      ; FUNCTION PADR
                      ; INPUTS: HL - ADDRESS TO BE PUNCHED
                      ; OUTPUTS: NONE
                      ; CALLS: PBYTE
                      ; DESTROYS: A
                      ; DESCRIPTION: PADR PUNCHES ON THE TELETYPEWRITER THE ADDRESS
                      ;              CONTAINED IN THE H,L REGISTERS.
                      ;
04C6                  PADR:
04C6    7C                  MOV     A,H      ; PUNCH FIRST HALF OF ADDRESS
04C7    CDCF04              CALL    PBYTE
04CA    7D                  MOV     A,L      ; PUNCH SECOND HALF OF ADDRESS
```

```
04CB   CDCF04          CALL    PBYTE
04CE   C9              RET               ; RETURN TO CALLING ROUTINE
                 ;
                 ;
                 ;*****************************************************************
                 ;
                 ;
                 ; FUNCTION PBYTE
                 ; INPUTS: A - CHARACTER TO BE PUNCHED
                 ;         D - CURRENT VALUE OF CHECKSUM
                 ; OUTPUTS: D - UPDATED VALUE OF CHECKSUM
                 ; CALLS: PRVAL,PO
                 ; DESTROYS: A,F/F'S
                 ; DESCRIPTION: PBYTE CONVERTS THE HEXADECIMAL VALUE IN THE A REGISTER
                 ;              INTO TWO ASCII CHARACTERS AND PUNCHES THESE CHARACTERS
                 ;              ON PAPER TAPE. THE CHECKSUM CONTAINED IN D IS UPDATED.
                 ;
04CF             PBYTE:
04CF   F5              PUSH    PSW       ; SAVE A,F/F'S
04D0   0F              RRC               ; POSITION UPPER 4 BITS INTO LOWER 4 BITS
04D1   0F              RRC
04D2   0F              RRC
04D3   0F              RRC
04D4   CDD502          CALL    PRVAL     ; CONVERT UPPER 4 BITS JUST ROTATED TO ASCII
04D7   CD0F05          CALL    PO        ; PUNCH CHARACTER
04DA   F1              POP     PSW       ; RESTORE A,F/F'S
04DB   F5              PUSH    PSW       ; SAVE A AGAIN
04DC   CDD502          CALL    PRVAL     ; CONVERT LOWER 4 BITS TO ASCII CHARACTER
04DF   CD0F05          CALL    PO        ; PUNCH CHARACTER
04E2   F1              POP     PSW       ; RESTORE A
04E3   82              ADD     D         ; ADD VALUE TO CHECKSUM
04E4   57              MOV     D,A       ; UPDATE D REGISTER WITH NEW CHECKSUM
04E5   C9              RET               ; RETURN TO CALLING ROUTINE
                 ;
                 ;
                 ;*****************************************************************
                 ;
                 ;
                 ; FUNCTION PEOF
                 ; INPUTS: NONE
                 ; OUTPUTS: NONE
                 ; CALLS: PO,PBYTE,PADR,LEAD
                 ; DESTROYS: A,C,D,H,L,F/F'S
                 ; DESCRIPTON: PEOF PUNCHES THE END OF FILE RECORD CONSISTING OF A RECORD
                 ;             MARK, A LOAD ADDRESS OF 0, THE RECORD TYPE, AND THE
                 ;             RECORD CHECKSUM.
                 ;
04E6             PEOF:
04E6   0E3A            MVI     C,':'
04E8   CD0F05          CALL    PO        ; PUNCH RECORD MARK
04EB   AF              XRA     A         ; ZERO CHECKSUM
```

```
04EC    57              MOV     D,A     ; SAVE IN D REGISTER
04ED    CDCF04          CALL    PBYTE   ; PUNCH RECORD LENGTH
04F0    210000          LXI     H,0     ; LOAD HL WITH ZERO ADDRESS
04F3    CDC604          CALL    PADR    ; PUNCH IT
04F6    3E01            MVI     A,1     ; LOAD A WITH RECORD TYPE
04F8    CDCF04          CALL    PBYTE   ; PUNCH IT
04FB    AF              XRA     A       ; ZERO A
04FC    92              SUB     D       ; COMPUTE CHECKSUM
04FD    CDCF04          CALL    PBYTE   ; PUNCH IT
0500    CDBA04          CALL    LEAD    ; PUNCH TRAILER
0503    C9              RET
                ;
                ;
                ;****************************************************************
                ;
                ;
                ; FUNCTION PEOL
                ; INPUTS: NONE
                ; OUTPUTS: NONE
                ; CALLS: PO
                ; DESTROYS: C
                ; DESCRIPTION: PEOL PUNCHES A CARRIAGE RETURN AND LINE FEED ONTO
                ;              PAPER TAPE.
                ;
0504            PEOL:
0504    0E0D            MVI     C,CR
0506    CD0F05          CALL    PO      ; PUNCH CARRIAGE RETURN CHARACTER
0509    0E0A            MVI     C,LF
050B    CD0F05          CALL    PO      ; PUNCH LINE FEED CHARACTER
050E    C9              RET
                ;
                ;
                ;****************************************************************
                ;
                ;
                ; FUNCTION PO
                ; INPUTS: C - CHARACTER TO BE PUNCHED
                ; OUTPUTS: NONE
                ; CALLS: CO
                ; DESTROYS: NOTHING
                ; DESCRIPTION: PO PUNCHES THE CHARACTER SUPPLIED IN THE C REGISTER TO
                ;              THE USER TELETYPEWRITER.
                ;
050F            PO:
050F    CDE801          CALL    CO      ; CALL CONSOLE OUT TO PERFORM CHARACTER OUTPUT
0512    C9              RET
                ;
                ;
                ;****************************************************************
                ;
                ;
```

```
                 ; FUNCTION RICH
                 ; INPUTS: NONE
                 ; OUTPUTS: A - ZERO, CARRY - 1 IF END OF FILE
                 ;          A - CHARACTER, CARRY  - 0 IF VALID CHARACTER
                 ; CALLS: RI
                 ; DESTROYS: A,F/F'S
                 ; DESCRIPTION: RICH TESTS FOR AN END OF FILE CONDITION.
                 ;
0513             RICH:
0513   CD1C05          CALL    RI       ; READ A CHARACTER FROM TAPE
0516   DA1202          JC      ERROR    ; JUMP IF READER TIMEOUT ERROR
0519   E67F            ANI     PRTY0    ; REMOVE PARITY BIT
051B   C9              RET              ; RETURN TO CALLING ROUTINE
                 ;
                 ;
                 ;*****************************************************************
                 ;
                 ;
                 ; FUNCTION RI
                 ; INPUTS: NONE
                 ; OUTPUTS: A - ZERO, CARRY - 1 IF END OF FILE
                 ;          A - CHARACTER, CARRY  - 0 IF VALID CHARACTER
                 ; CALLS: DELAY
                 ; DESTROYS: A,F/F'S
                 ; DESCRIPTION: RI READS A CHARACTER FROM THE TTY TAPE READER.
                 ;
051C             RI:
051C   C5              PUSH    B        ; SAVE BC
051D             RI05:
051D   DBED            IN      CNCTL    ; READ IN USART STATUS
051F   E604            ANI     TXBE     ; CHECK FOR TRANSMITTER BUFFER EMPTY
0521   CA1D05          JZ      RI05     ; TRY AGAIN IF NOT EMPTY
0524   3E27            MVI     A,TTYADV        ; ADVANCE THE TAPE
0526   D3ED            OUT     CNCTL    ; OUTPUT THE ADVANCE COMMAND
0528   0628            MVI     B,40     ; INITIALIZE TIMER FOR 40 MS.
052A             RI07:
052A   CDB104          CALL    DELAY    ; DELAY FOR 1 MILLISECONDS
052D   05              DCR     B        ; DECREMENT TIMER
052E   C22A05          JNZ     RI07     ; JUMP IF TIMER NOT EXPIRED
0531   3E25            MVI     A,CMD    ; STOP THE READER ADVANCE
0533   D3ED            OUT     CNCTL    ; OUTPUT STOP COMMAND
0535   06FA            MVI     B,250    ; INITIALIZE TIMER FOR 250 MS.
0537             RI10:
0537   DBED            IN      CONST    ; INPUT READER STATUS
0539   E602            ANI     RBR      ; CHECK FOR RECEIVER BUFFER READY
053B   C24905          JNZ     RI15     ; YES - DATA IS READY
053E   CDB104          CALL    DELAY    ; DELAY 1 MS
0541   05              DCR     B        ; DECREMENT TIMER
0542   C23705          JNZ     RI10     ; JUMP IF TIMER NOT EXPIRED
0545   AF              XRA     A        ; ZERO A
0546   37              STC              ; SET CARRY INDICATING EOF
```

```
0547    C1              POP     B           ; RESTORE BC
0548    C9              RET                 ; RETURN TO CALLING ROUTINE
0549            RI15:
0549    DBEC            IN      CNIN        ; INPUT DATA CHARACTER
054B    B7              ORA     A           ; CLEAR CARRY
054C    C1              POP     B           ; RESTORE BC
054D    C9              RET                 ; RETURN TO CALLING ROUTINE
                ;
                ;
                ;*********************************************************************
                ;
                ;
054E            COPYRT:
054E    28432920        DB      '(C) 1976 INTEL CORP'
0552    31393736
0556    20494E54
055A    454C2043
055E    4F5250
                ;
                ;
                ;*********************************************************************
                ;
                ;
3C00                    ORG     DATA
3C2E                    ORG     REGS        ; ORG TO REGISTER SAVE - STACK GOES IN HERE
                ;
3C2E            MSTAK           EQU     $           ; START OF MONITOR STACK
3C2E    00      ESAVE:          DB      0           ; E REGISTER SAVE LOCATION
3C2F    00      DSAVE:          DB      0           ; D REGISTER SAVE LOCATION
3C30    00      CSAVE:          DB      0           ; C REGISTER SAVE LOCATION
3C31    00      BSAVE:          DB      0           ; B REGISTER SAVE LOCATION
3C32    00      FSAVE:          DB      0           ; FLAGS SAVE LOCATION
3C33    00      ASAVE:          DB      0           ; A REGISTER SAVE LOCATION
3C34    00      LSAVE:          DB      0           ; L REGISTER SAVE LOCATION
3C35    00      HSAVE:          DB      0           ; H REGISTER SAVE LOCATION
3C36    0000    PSAVE:          DW      0           ; PGM COUNTER SAVE LOCATION
3C38    0000    SSAVE:          DW      0           ; USER STACK POINTER SAVE LOCATION
3C3A    00      TEMP:           DB      0           ; TEMPORARY MONITOR CELL
                ;
3C3D                    ORG     BRLOC               ; ORG TO USER BRANCH LOCATION
                ;
3C3D            USRBR:          DS      3           ; BRANCH GOES IN HERE
                ;
                ;
                        END
NO PROGRAM ERRORS
```

8080 MACRO ASSEMBLER, VER 2.4   ERRORS = 0 PAGE 37
    80/10 MONITOR, VERSION 1.1, 1 NOVEMBER 1976

SYMBOL TABLE

* Ø1

A	ØØØ7	ADRD	Ø1A8	ADROU	Ø1B1	ASAVE	3C33
B	ØØØØ	BRCHR	ØØ1B	BREAK	Ø1C2	BRLOC	3C3D
BRTAB	Ø3FA	BSAVE	3C31	BYTE	Ø496	C	ØØØ1
CADR	Ø3A6	CI	Ø1D5	CMD	ØØ25	CNCTL	ØØED
CNIN	ØØEC	CNOUT	ØØEC	CNVBN	Ø1DF	CO	Ø1E8
CONST	ØØED	COPYR	Ø54E *	CPYRT	Ø3E3 *	CR	ØØØD
CROUT	Ø1F3	CSAVE	3C3Ø	CTAB	Ø3B8	D	ØØØ2
DATA	3CØØ	DCMØ5	ØØ66	DCM1Ø	ØØ6C	DCMD	ØØ5F
DEL1	Ø4B4	DELAY	Ø4B1	DSAVE	3C2F	E	ØØØ3
ECHØ5	Ø2Ø2	ECH1Ø	Ø21Ø	ECHO	Ø1F9	ERROR	Ø212
ESAVE	3C2E	ESC	ØØ1B	EXIT	Ø217	FALSE	ØF9C
FRET	Ø21D	FSAVE	3C32	GCMØ5	ØØAØ	GCM1Ø	ØØA6
GCMD	ØØ83	GETCH	Ø22Ø	GETCM	ØØ2C	GETHX	Ø227
GETNM	Ø25B	GHXØ5	Ø22D	GHX1Ø	Ø245	GNMØ5	Ø262
GNM1Ø	Ø277	GNM15	Ø265	GNM2Ø	Ø28A	GNM25	Ø295
GNM3Ø	Ø299	GO	ØØØ8 *	GTCØ3	ØØ3C	GTCØ5	ØØ49
GTC1Ø	ØØ55	H	ØØØ4	HCHAR	ØØØF	HILØ5	Ø2AF
HILO	Ø2AØ	HSAVE	3C35	ICMØ5	ØØB4	ICM1Ø	ØØDB
ICM2Ø	ØØE3	ICM25	ØØE9	ICMD	ØØA9	INUST	Ø2B2
INVRT	ØØFF	L	ØØØ5	LEDØ5	Ø4BC	LEAD	Ø4DA
LF	ØØØA	LSAVE	3C34	LSCNO	ØØ11	M	ØØØ6
MCMØ5	ØØF7	MCMD	ØØEF	MODE	ØØCF	MSGL	ØØ23
MSTAK	3C2E	NCMDS	ØØ78	NEWLN	ØØØF	NMOUT	Ø2C2
ONEMS	Ø383	PADR	Ø4C6	PBYTE	Ø4CF	PEOF	Ø4E6
PEOL	Ø5Ø4	PO	Ø5ØF	PRTYØ	ØØ7F	PRVAL	Ø2D5
PSAVE	3C36	PSW	ØØØ6	RBR	ØØØ2	RCMØ5	Ø412
RCM1Ø	Ø42F	RCMD	Ø4Ø6	REGØ5	Ø2E2	REG1Ø	Ø2DC
REG15	Ø3Ø7	REGDS	Ø2DF	REGS	3C2E	RGAØ5	Ø316
RGA1Ø	Ø323	RGADR	Ø31Ø	RI	Ø51C	RIØ5	Ø51D
RIØ7	Ø52A	RI1Ø	Ø537	RI15	Ø549	RICH	Ø513
RSTTF	Ø327	RSTU	ØØ38	RTAB	Ø3CØ	RTABS	ØØØ3
SCMØ5	Ø114	SCM1Ø	Ø11F	SCM15	Ø12F	SCMD	Ø1ØF
SGNON	Ø395	SGMSG	ØØ1E	SP	ØØØ6	SRET	Ø33B
SSAVE	3C38	STHØ5	Ø35B	STHFØ	Ø33D	STHLF	Ø348
TEMP	3C3A	TERM	ØØ1B	TRDY	ØØØ1	TRUE	ØF9F
TTYAD	ØØ27	TXBE	ØØØ4	UPPER	ØØFF	USRBR	3C3D
VALDG	Ø367	VALDL	Ø382	WCMØ5	Ø44B	WCM1Ø	Ø46Ø
WCM15	Ø463	WCM2Ø	Ø47B	WCM25	Ø49Ø	WCMD	Ø441
XCMØ5	Ø145	XCM1Ø	Ø154	XCM15	Ø161	XCM2Ø	Ø17F
XCM25	Ø196	XCM27	Ø197	XCM3Ø	Ø19F	XCMD	Ø133

* Ø2

* Ø3

* Ø4

* Ø5

intel®

SBC 80/10 And SBC 80/10A
Single Board Computer
Hardware Reference Manual

9800230F

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. ____________

NAME ______________________________ DATE ____________
TITLE ______________________________
COMPANY NAME/DEPARTMENT ______________________________
ADDRESS ______________________________
CITY ____________ STATE ____________ ZIP CODE ____________

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

First Class
Permit No. 1040
Santa Clara, CA

BUSINESS REPLY MAIL

No Postage Stamp Necessary if Mailed in U.S.A.

Postage will be paid by:

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Attention: MCD Technical Publications