Twenty Dollars

Cromemco™

SCC

SINGLE CARD COMPUTER

INSTRUCTION MANUAL

CROMEMCO, Inc.
280 Bernardo Avenue
Mountain View, CA 94043

Part No.023-0050

January 1980

SCC I/O PORT SUMMARY

NUMBER	INPUT PORTS	OUTPUT PORTS
		BAUD RATE[1]
00H	SERIAL STATUS: TSE, RDA, IPG, TSE, RDA, SRV, ORE, FME	STOP BITS, 9600, 4800, 2400, 1200, 300, 150, 110
01H	RECEIVER DATA: DATA BYTE FROM TERMINAL	TRANSMITTER DATA: DATA BYTE TO TERMINAL
02H		COMMAND REGISTER: NOT USED, NOT USED, TBS, HBD, INE, RS7, SRK, RES
03H	INTERRUPT ADDRESS: 1, 1, L_2, L_1, L_0, 1, 1, 1	INTERRUPT MASK: TIMER 5 OR PI7, TIMER 4, TBE, RDA, TIMER 3, $\overline{INT}$, TIMER 2, TIMER 1
04H	PARALLEL IN 04H: D7[2], D6, D5, D4, D3, D2, D1, D0	PARALLEL OUT 04H: D7, D6, D5, D4, D3, D2, D1, D0
05H		TIMER 1: DELAY COUNT 1
06H		TIMER 2: DELAY COUNT 2
07H		TIMER 3: DELAY COUNT 3
08H		TIMER 4: DELAY COUNT 4
09H		TIMER 5: DELAY COUNT 5
0AH	PARALLEL IN 0AH: D7, D6, D5, D4, D3, D2, D1, D0	PARALLEL OUT 0AH: D7[3], D6, D5, D4, D3, D2, D1, D0
0BH	PAPALLEL IN 0BH: D7, D6, D5, D4, D3, D2, D1, D0	PARALLEL OUT 0BH: D7, D6, D5, D4, D3, D2, D1, D0

NOTE 1: SETTING PORT OUT 02H BIT HBD TO LOGIC 1 OCTUPLES THESE BAUD RATES.

2: PORT IN 04H BIT D7 MAY BE DEFINED AS AN AUXILIARY INTERRUPT INPUT, REPLACING TIMER 5 AS THE LEVEL 7 INTERRURPT SOURCE, BY SETTING OUT 02H BIT RS7 TO LOGIC 1.

3: PORT OUT 0AH BIT D7 CONTROLS SCC MEMORY ENABLE/DISABLE IF THE MEMORY DISABLE FUNCTION IS SELECTED. D7=LOGIC 0 ENABLES SCC MEMORY

Cromemco™

SCC

SINGLE CARD COMPUTER

INSTRUCTION MANUAL

CROMEMCO, Inc.
280 Bernardo Avenue
Mountain View, CA 94043

Part No.023-0050

January 1980

This manual was produced on a Cromemco System Three computer using the SCREEN Editor. The edited text was formatted using the Cromemco Word Processing System Formatter. Final camera-ready copy was printed on a Cromemco 3355A printer.

SCC INSTRUCTION MANUAL

TABLE OF CONTENTS

Section 1

INTRODUCTION

The Cromemco Single Card Computer (SCC) is a Z80A based, S-100 bus microcomputer subsystem. The SCC is self-contained, excepting external power, making it an ideal choice for small to medium-scale dedicated applications. The SCC provides all the features one could want in a powerful, yet reasonably priced, single card microcomputer (see Figure 1):

Processor: The powerful Z80A clocked at 4.000 MHz - with the 8080A instruction set as a subset of its own 158 instruction set.

Memory: 1 Kbyte of 4045 static RAM and space for 8 Kbytes of 2716 EPROM firmware.

Firmware: Cromemco's Z80 Monitor and 3K Control BASIC available on two ROMs (model MCB-216).

Parallel I/O: Three independent 8-bit parallel output ports, and three independent 8-bit parallel input ports, all with handshake.

Serial I/O: One RS-232/20 mA current loop serial I/O port with software selectable baud rate.

Interval Timers: Five independent interval timers with range 0 - 16.32 mSec and resolution 64 uSec generating prioritized interrupts on Count 00.

Interrupts: Supports all three Z80 interrupts modes; one SCC input port contains the highest priority interrupt requesting service; one SCC output port selectively masks-off unwanted interrupt sources.

Expandability: Compatible with all Cromemco S-100 products. Thus, the standard SCC functions may be enhanced by adding Cromemco cards for A/D and D/A conversion, RAM/ROM memory expansion, color graphics, line printer interfacing, floppy disk interfacing, and more.

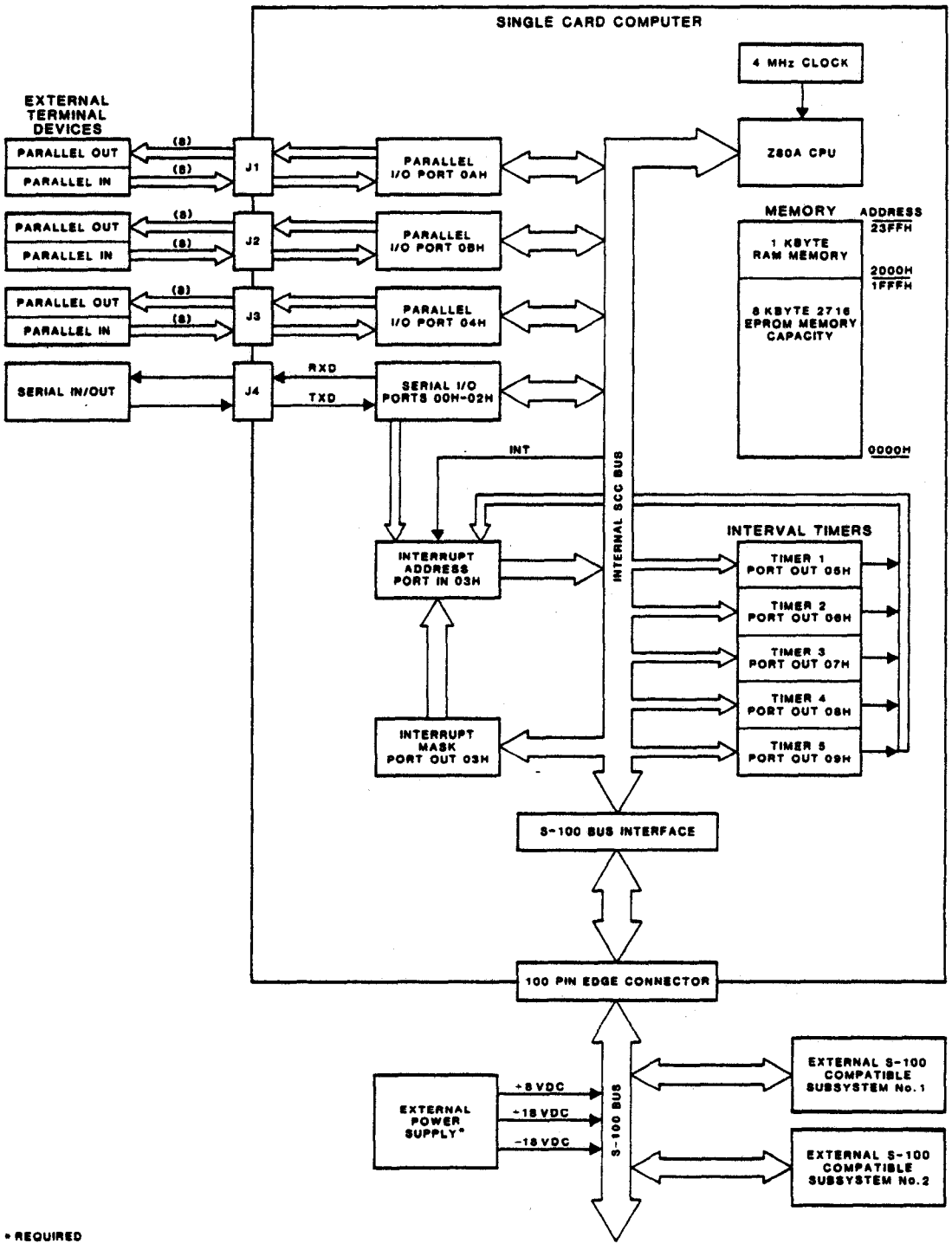


Figure 1: SCC FUNCTIONAL BLOCK DIAGRAM

SCC TECHNICAL SPECIFICATIONS

Processor Type:	Z80-A
Clock Frequency:	4.000 MHz
Instruction Set:	158 instructions including the 78 member 8080A CPU instruction set
ROM Capacity:	8 KBytes spanning address range 0000H - 1FFFH
ROM Type:	Intel 2716 EPROM, TI 2516 EPROM, or equivalent (user supplied)
RAM Capacity:	1 KBytes spanning address range 2000H - 23FFH
RAM Type:	4045 Static (included)
UART Type:	TMS 5501 or equivalent
Serial I/O Port:	RS-232 or 20 mA current loop; software selectable from 110 to 76,800 Baud, one or two stop bits
Parallel Ports:	24 lines of input data organized as three parallel 8-bit ports; 24 lines of output data organized as three parallel 8-bit ports
Interval Timers:	5 independent timers with range 0 - 16.32 mSec and 64 uSec resolution
Interrupts:	All three Z80 interrupt modes supported (IM 0, IM 1 & IM 2); prioritized IM 0 RST vectors may be automatically supplied by eight SCC interrupt sources
Bus Compatibility:	Cromemco Standard-100 (S-100)
Power Requirements:	+8 VDC @ 1.75 Amps (max) +18 VDC @ 100 mA (max) -18 VDC @ 50 mA (max)
Operating Environment:	0 - 55 degrees Celsius

ABOUT THIS MANUAL:

The remainder of this manual is divided into four major sections: Section 2, Operating Instructions; Section 3, System Considerations; Section 4, Theory of Operation; and Section 5, A Simple SCC Example System.

Section 2 provides detailed descriptions of all SCC subsystems (memory, serial/parallel I/O ports, interval timers, etc.). An adjunct to this section is Appendix A, SCC I/O Port Summary. The reader is encouraged to scan Appendix A before studying Section 2 in detail to get an overview of SCC I/O port functions. Also note the quick-reference SCC I/O Port Summary which appears on the manual overleaf.

Section 3 discusses the SCC in a system context. The various system initialization tasks and options discussed throughout Section 2 are gathered together in this section, together with a discussion of SCC development systems.

Section 4 provides an analysis of the major SCC logic functions and signal paths, and finally Section 5 presents a simple, yet complete and illustrative example system to codify many of the ideas presented earlier in the manual. The experienced user would profit from scanning Section 5 before reading the other manual sections.

Section 2

Operating Instructions

The SCC is a highly versatile control subsystem which must be defined and viewed in its final system context. Consequently, the SCC operating instructions cannot be effectively reduced to a simple step-by-step formulation. Instead, pertinent operating instructions are topically grouped by function in the subsections which follow. The SCC user is urged to read this material in its entirety with the aim of selecting among the SCC options and modes which best meet his own final system requirements.

2.1 SCC CENTRAL PROCESSING UNIT

The heart of the SCC is its Z80A central processing unit (CPU). The Z80A is an 8-bit processor which can directly address 64 KBytes of memory (locations 0000 thru FFFF hexadecimal), 256 input ports and 256 output ports (ports 00 thru FF hexadecimal). The Z80's 158-member instruction set includes the 8080's 78-member instruction set as a subset. Moreover, the Z80 supplies extended capabilities such as bit test, block search/move, relative jumps, a complete alternate register set, two index registers, three interrupt modes, a non-maskable interrupt input pin, and more, which the 8080 does not. Please refer to the Z80 Technical Manual for full details.

The Z80A is clocked at a fixed 4.000 MHz on the SCC yielding a 250 nSec CPU cycle time. Thus Z80A instruction execution time

ranges from 1.00 uSec (for 4-cycle instructions) to 5.75 uSec (for 23-cycle instructions) with no wait states. Instruction execution time may be extended an indefinite integral number of idling CPU cycles, or 'wait states', by asserting S-100 bus line RDY low to synchronize the CPU to slow memory or I/O.

Z80A support hardware on the SCC generates an ensemble of S-100 bus status and processor control signals making the SCC fully compatible with all Cromemco S-100 products. These include additional RAM memory (4KZ RAM Card, 16KZ RAM Card, 64KZ RAM Card), EPROM memory and programmer cards (Bytesaver II, 32K Bytesaver), parallel I/O cards (8PIO, Isolated 4PIO), an A/D and D/A card (D+7A), a serial interface card (TU-ART), a floppy disk controller card (4FDC), color graphics interface cards, and a line printer interface card (PRI).

2.2 SCC POWER, POWER-UP, RESET

The SCC card requires three unregulated power supply voltages: +8 VDC @ 1.75 Amps (S-100 pins 1 and 51); +18 VDC @ 100 mAmps (pin 2); and -18 VDC @ 50 mAmps (pin 52). Voltage regulators on the SCC itself regulate these voltages to +5 VDC, +12 VDC and -5 VDC respectively.

Applying power (+8 VDC) causes SCC circuitry to automatically generate a momentary low level Power-On Clear pulse on S-100 bus line $\overline{\text{POC}}$, which in turn resets the Z80A, clears SCC parallel ports OUT 0AH and OUT 0BH, unconditionally enables the SCC on-board

memory, and resets any other devices which respond to an active low level on S-100 bus line $\overline{\text{RESET}}$. The Z80A responds to an active low level on its $\overline{\text{RESET}}$ pin by (1) Disabling Interrupts, (2) Resetting its I (IM 2 Interrupt Address) Register to 00H, (3) Resetting its R (Refresh Address) Register to 00H, (4) Setting Interrupt Mode 0 (IM 0), and finally, (5) Performing a Jump to memory location 0000H. The SCC provides 8 KBytes of contiguous 2716 EPROM memory space starting at 0000H, thus the SCC user would typically locate a system initialization routine in 2716 EPROM starting at location 0000H, or alternately use the Cromemco MCB-216 ROM chip set which has provisions for either automatically running user defined 3K Control BASIC firmware, or entering the interactive 3K Control Basic command mode.

In addition to a POC, there are two other sources which may generate an SCC system reset. The first is from S-100 bus line $\overline{\text{RESET}}$ which again resets the Z80A, clears SCC parallel ports OUT 0AH and OUT 0BH, unconditionally enables SCC on-board memory, and effects any other S-100 bus cards which are connected to the $\overline{\text{RESET}}$ line. The second source is from SCC connector J3 (one of the four 25-pin connectors provided along the top edge of the SCC card), pin 13. This line is also labeled $\overline{\text{RESET}}$, and an active low level on this line causes the same SCC response as described above, but it does not force S-100 bus line $\overline{\text{RESET}}$ active low. This second method provides a means for an external device to reset the SCC subsystem, and not effect other S-100 cards in the process. Figure 2 illustrates these three reset sources.

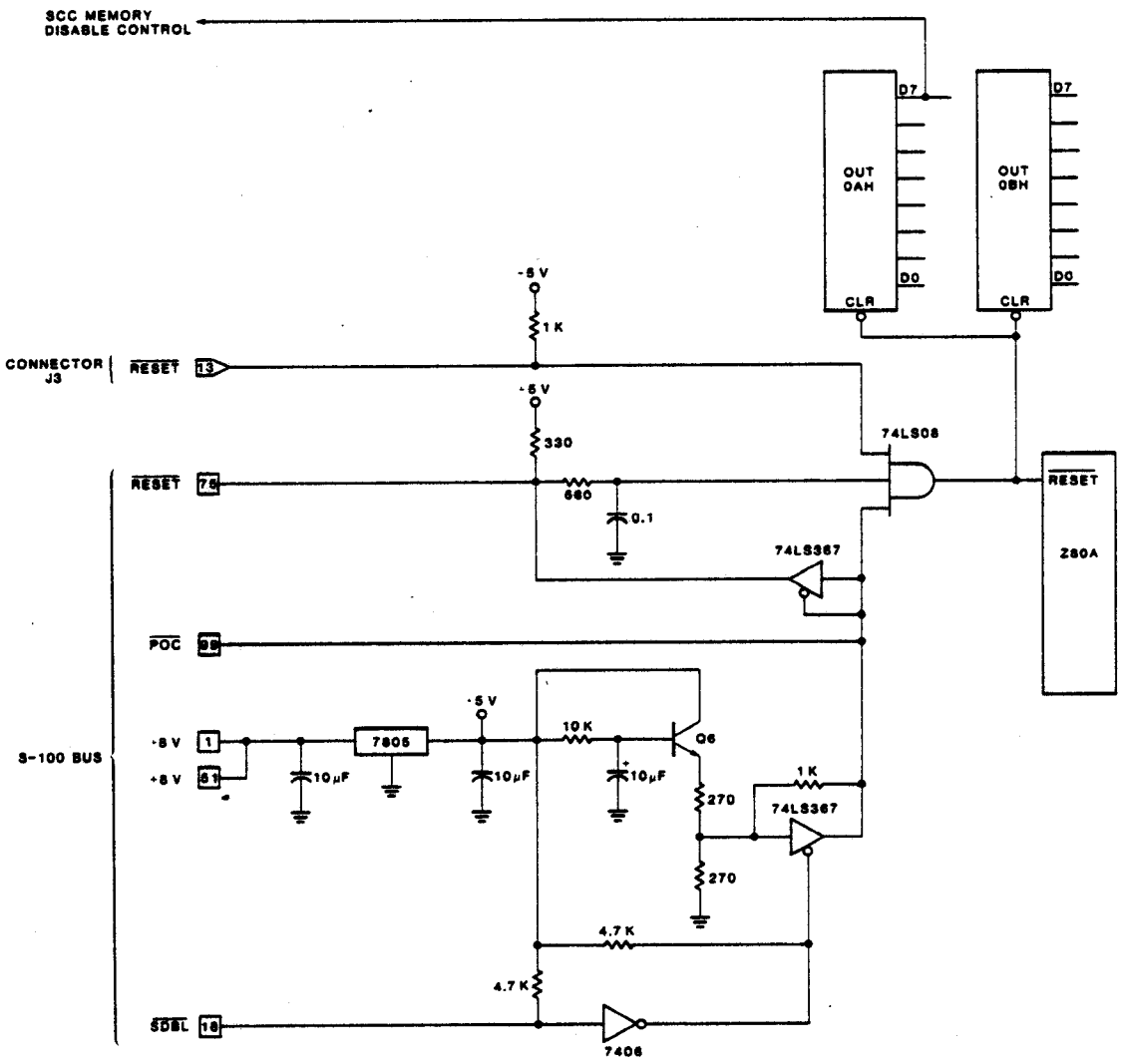


Figure 2: SCC RESET SOURCES

2.3 SCC MEMORY ORGANIZATION

The SCC provides space for 8 KBytes of 2716 EPROM memory in sockets ROM 0, ROM 1, ROM 2 and ROM 3, spanning addresses 0000H thru 1FFFH in 800H increments (2 KBytes), and 1 KByte of 4045 static RAM memory spanning 2000H thru 23FFH. The SCC memory map is shown in Figure 3.

The Intel 2716 (or TI 2516) is a 2 K by 8-bit Erasable Programmable Read Only Memory (EPROM) with a 450 nSec (max) memory access time. (Note that Texas Instrument's 2716 EPROM is not equivalent to the Intel 2716, and thus may not be used in the SCC.) The device is typically shipped from the manufacturer in the 'erased' condition (all bits set to logic 1), and is programmed by selectively resetting individual bits to the logic 0 state using a 2716 programming device such as Cromemco's 32K Bytesaver card. Once programmed, the 2716 retains the programmed bit pattern until re-programmed (in which case logic 1 bits may be re-programmed to logic 0 bits, but not vice versa), or until the entire device is erased by exposure to ultra-violet light which un-selectively returns all array bits to the logic 1 state (see manufacturer's specifications for detailed EPROM erasure procedures).

All, none, or any combination of SCC ROM sockets may be populated by user-supplied 2716 EPROMs. Performing a memory read on an empty SCC ROM socket almost always reads data 0FFH (all bits logic 1 from floating data lines); performing a memory write to an empty SCC ROM socket has no effect other than consuming time, but writing to a full socket should be avoided as there will be a data bus conflict between the Z80A write data and the 2716 read data.

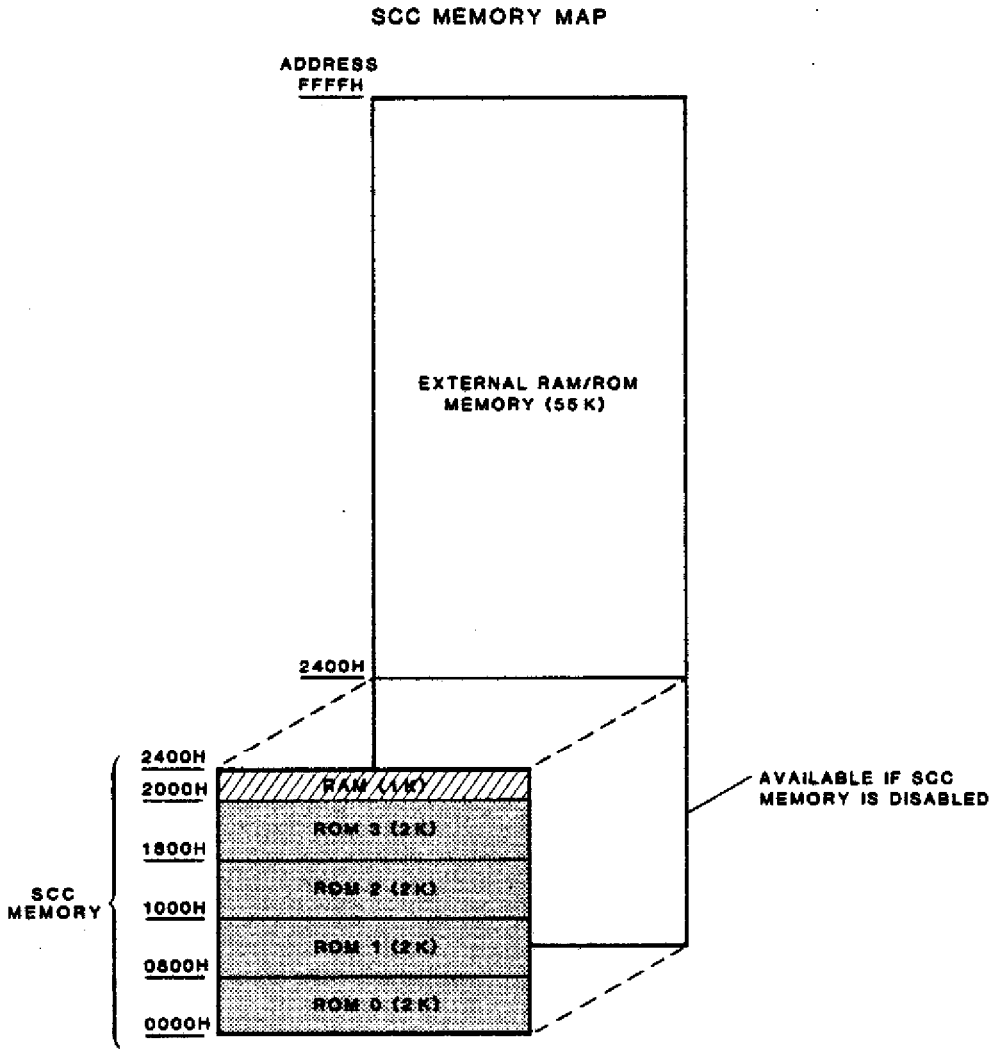


Figure 3: SCC MEMORY MAP

Since the 2716 has a memory access time of 450 nSec while the CPU cycle time is 250 nSec, the SCC automatically inserts one wait state during all memory request machine cycles to the lowest 8 KBytes of memory (0000H - 1FFFH) where 2716 EPROM memory resides. Note that these wait states will not be inserted if the SCC memory

is disabled and external user-supplied memory occupies the 0000H - 1FFFH area (see Section 2.4).

The number of required wait states versus memory access time, assuming the SCC Z80A processor with its 4.000 MHz symmetric clock, are tabulated below:

TABLE 1: WAIT STATES

WAIT STATES REQUIRED	MEMORY ACCESS TIME (NANOSECONDS)
NONE	340 (MAX)
ONE EACH M1 CYCLE ONLY	455 (MAX)
ONE PER MEMORY CYCLE	590 (MAX)

The SCC provides a jumper option which is labeled 'M1 ONLY' on the board legend (see Figure 4), and 'INSERT JUMPER FOR M1 WAITS ONLY' on the SCC Schematic Diagram. Installing an insulated jumper wire connecting the open-circuited M1 ONLY soldering pads will cause one wait state to be inserted only during M1 fetch cycles to the lowest 8 KBytes of memory. Note that even though the 2716 satisfies the 455 nSec (max) memory access requirement for M1 WAITS ONLY, it only marginally does so. Thus the SCC is shipped from Cromemco with the option de-selected. The option is included primarily to anticipate faster versions of the 2716.

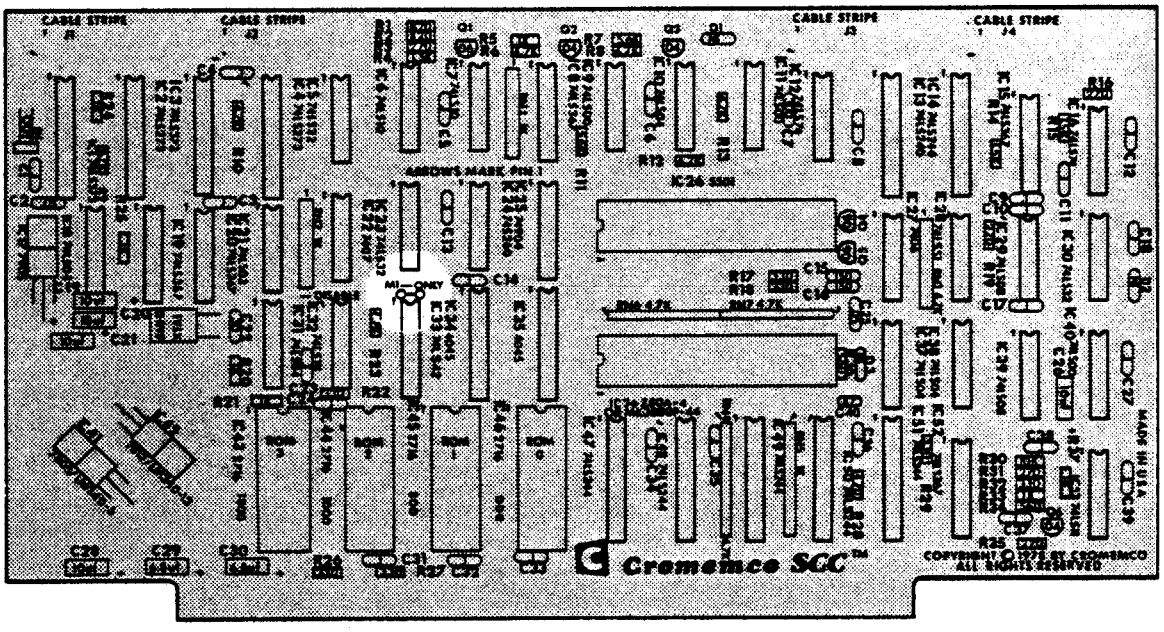


Figure 4: M1 WAITS JUMPER OPTION

The SCC read/write memory physically consists of two 4045 1 K by 4-bit static RAMs occupying sockets IC34 and IC35. Since the 4045 memory access time is 250 nSec (max), no wait states are required for these devices. IC35 contains the high-order nybble of each RAM byte, and IC34 contains all low-order nybbles. When power is first applied to the SCC, the RAM contents should be assumed random.

2.4 MEMORY EXPANSION/SCC MEMORY DISABLE

The amount of memory available to the SCC may be expanded by merely plugging additional RAM or ROM memory cards into the SCC controlled S-100 bus. The additional S-100 bus compatible memory

may overlap, be disjoint with, or be contiguous with the SCC on-board memory. It is the external memory's responsibility to insert wait states by controlling S-100 bus line RDY, if necessary, for 4 MHz operation.

Installing external memory which overlaps the SCC 0000H - 23FFH memory area gives rise to the situation illustrated in Figure 5. For memory reads, the SCC reads data only from its on-board memory in the address range 0000H - 23FFH, and from the external S-100 bus in the address range 2400H - FFFFH; any external memory residing in the 0000H - 23FFH area is simply ignored. For memory writes, the SCC writes data to its on-board memory and to the external S-100 bus in parallel, although writing to the SCC ROM area 0000H - 1FFFH does not alter the 2716 EPROM contents. Stated another way, the SCC 'listens' to only its own memory when the memory address is in the range 0000H - 23FFH, and on-board RAM memory is 'transparent' to all memory write operations (empty on-board ROM sockets are also transparent to memory writes).

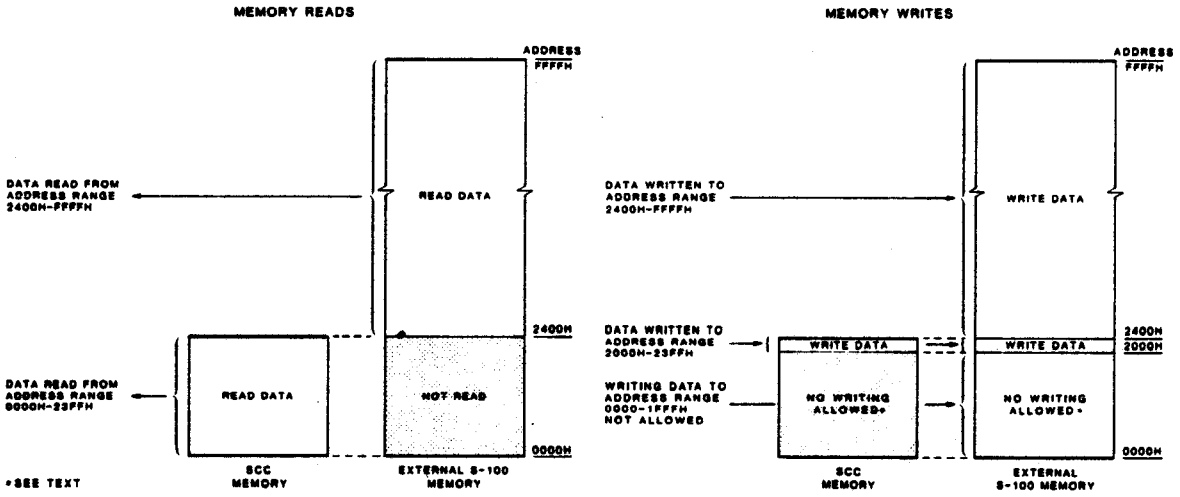


Figure 5: MEMORY OVERLAPPING THE SCC

Additional memory may be added up to the Z80A's 64 KByte direct addressing limit, and by incorporating Cromemco's memory products with Bank Select, memory may be expanded further to 512 KBytes organized as eight 'banks' of 64 KBytes each. Cromemco memory boards with Bank Select (RAM Boards: 4KZ, 16KZ, 64KZ; EPROM Boards: Bytesaver II, 16KPR, 32K Bytesaver) are mapped into any combination of Memory Banks (Bank 0 thru Bank 7) by setting an eight position slide switch on the card. Banks are enabled and disabled under software control by outputting a bank select control byte to port OUT 40H, an integral output port on each Cromemco product with Bank Select. If a logic 1 bit (D0 thru D7)

in the bank select control byte matches any Memory Bank the board is mapped into, the memory board becomes active and is thus switched 'into' the memory map. Conversely, if there is no match between any bank activated by the control byte and the banks the board is mapped into, the board goes inactive, and thus is switched 'out of' the memory map. Finally note in connection with the Bank Select feature that SCC on-board memory is not Bank Select controlled. An example will illustrate the foregoing ideas.

EXAMPLE 1

Suppose a 16KZ RAM board, with the switch settings shown in Figure 6A, is co-resident with an SCC in an S-100 bus. The switch settings map the 16KZ into the address range 0000H - 3FFFH (the lowest 16 KBytes of memory) in Memory Banks 1 and 2.

(a) When power is first applied, SCC memory unconditionally enables. The 16KZ responds to a POC by enabling only if it is mapped into Bank 0, and disabling otherwise. Thus, the 16KZ is switched out of the memory map in response to a POC. The system memory map immediately following a POC is shown in Figure 6b.

(b) Now suppose the following two instructions, residing in SCC ROM 0, are executed:

```
ADDR    OBJECT      MNEMONIC                COMMENT

0000    3E82        LD    A,10000010B       ;ENABLE BANK 1 & BANK 7
0002    D340        OUT   40H,A             ;OUTPUT CONTROL BYTE
```

The 16KZ is mapped into one of the banks activated by the bank select control byte (Bank 1), thus it is switched into the memory map (Figure 6C).

(c) Finally assume the following two instructions, also residing in SCC ROM 0, are executed:

```
ADDR    OBJECT      MNEMONIC                COMMENT

0004    3E01        LD    A,00000001B       ;ENABLE ONLY BANK 0
0006    D340        OUT   40H,A             ;OUTPUT CONTROL BYTE
```

Since the 16KZ is not mapped into Bank 0, it is switched out

of the memory map in response to the control byte, and the memory map reverts back to that shown in Figure 6b.

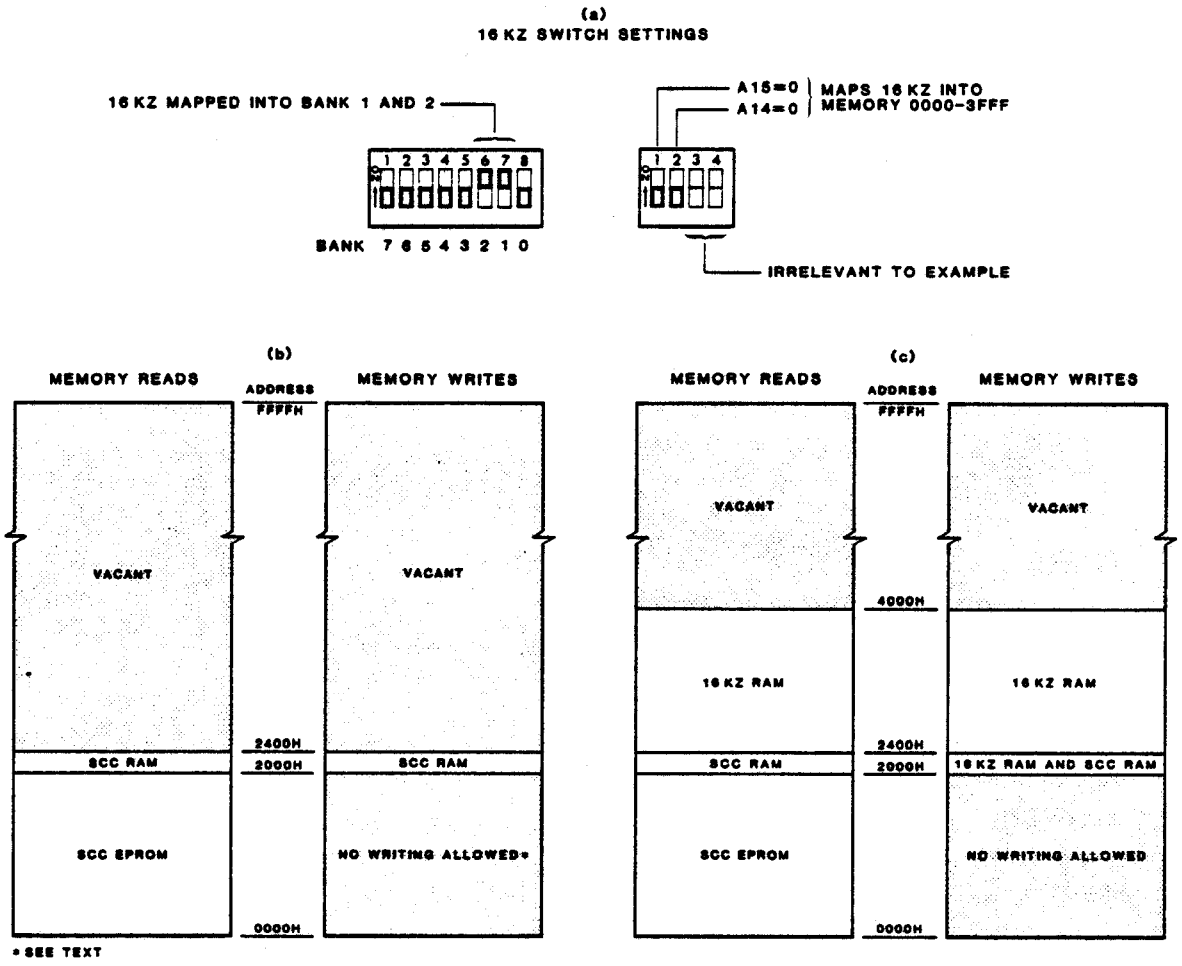


Figure 6: EXAMPLE 1 SWITCH SETTING & MEMORY MAPS

Although the SCC on-board memory does not respond to Bank Select, a functionally equivalent software controlled memory enable/disable option is provided. The Memory Disable option is exercised by cutting the existing foil trace between the two solder pads labeled 'DISABLE' on the SCC board legend (see Figure 7). With the trace cut, high order bit D7 of SCC parallel port

OUT 0AH controls the SCC memory enable/disable function. Setting bit D7 to logic 1 disables SCC memory; resetting bit D7 to logic 0 enables SCC memory. Either a POC or a RESET unconditionally enables the on-board SCC memory, since either a POC or an active low level on S-100 bus line $\overline{\text{RESET}}$ resets all parallel port OUT 0AH bits to logic 0. If the Memory Disable trace is left in its factory shipped closed condition, or if it is later re-installed, bit D7 of port OUT 0AH has no effect on SCC memory enable/disable.

EXAMPLE 2

Assume the same 16KZ RAM and SCC configuration as in Example 1, and also assume the SCC Memory Disable option is selected by cutting the foil trace.

(a) After power is first applied, the 16KZ automatically disables since it is not mapped into Bank 0, and the SCC on-board memory unconditionally enables since all bits of port OUT 0AH are reset to logic 0. The resulting memory map is shown in Figure 7A. Note that an S-100 RESET after power is applied produces the same map.

(b) Now suppose the following four instructions, residing in SCC ROM 0, are executed:

```
ADDR  OBJECT     MNEMONIC                    COMMENT

0000  3E82       LD    A,10000010B           ;ENABLE BANK 7 & BANK 2
0002  D340       OUT   40H,A                 ;OUTPUT CONTROL BYTE
0004  3E80       LD    A,10000000B           ;SET BIT D7 TO LOGIC 1
0006  D30A       OUT   0AH,A                 ;DISABLE SCC MEMORY
```

The first two instructions activate the 16KZ since it is assumed switch-mapped into Bank 2, and the next two instructions disable all SCC memory by setting bit D7 of port OUT 0AH to logic 1 (Figure 7B). Note that after the four instructions above are executed, the Z80A will fetch the next opcode from 16 KZ RAM address 0008H (the next sequential address). Thus provisions must be made to maintain program continuity when bank switching in this fashion.

(c) Finally suppose the following two instructions, residing in SCC ROM 0, are executed:

```
ADDR   OBJECT     MNEMONIC                        COMMENT

0008   3E00       LD     A,00000000B              ;RESET BIT D7 TO LOGIC 0
000A   D30A       OUT    0AH,A                    ;ENABLE SCC MEMORY
```

Since no Bank Select control word is output, the 16KZ remains in the map. The two instructions above also enable SCC memory (Figure 7c).

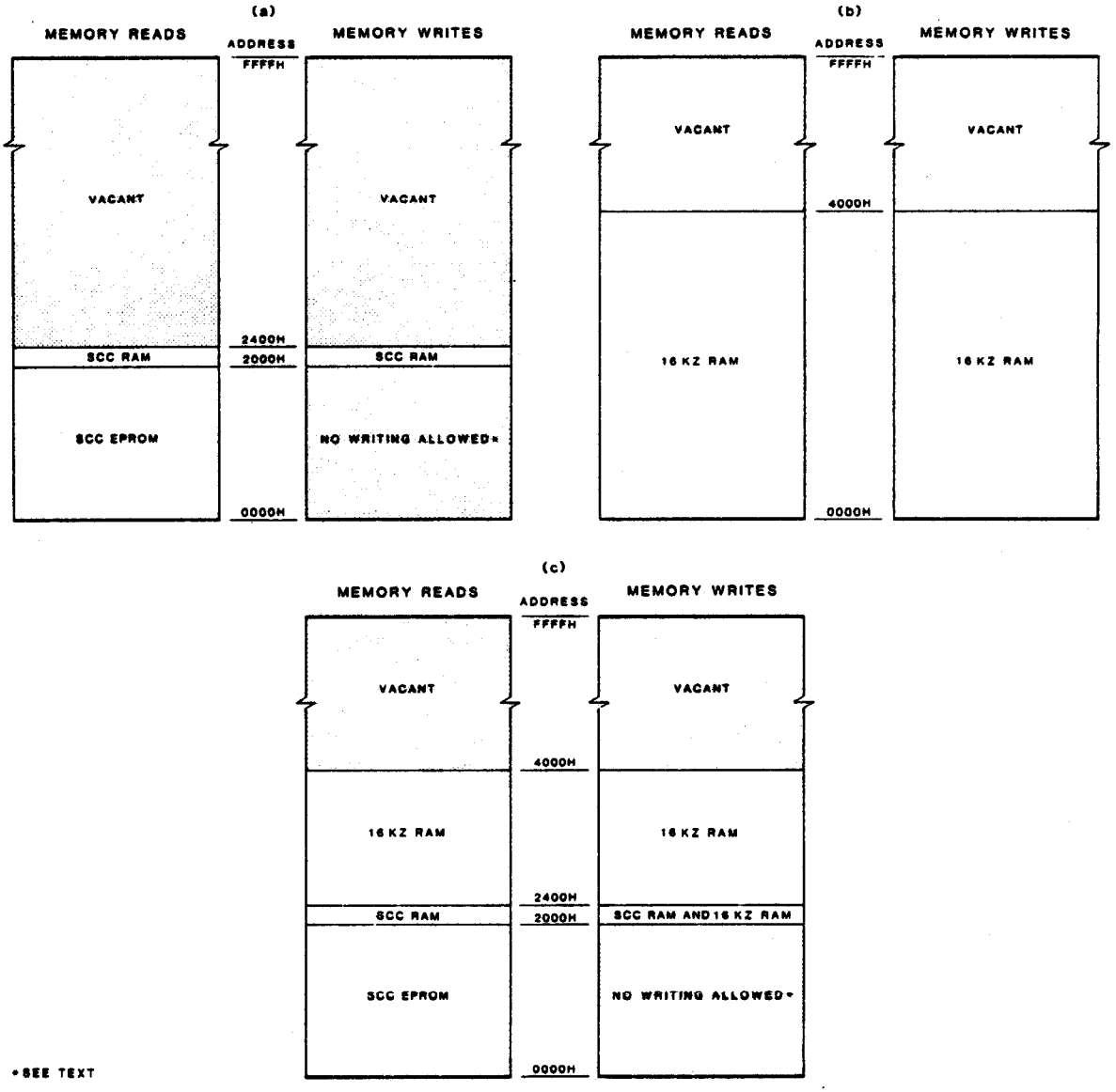


Figure 7: EXAMPLE 2 MEMORY MAPS

2.5 SCC PIN-OUTS & INTERFACE CABLING

The SCC may communicate with external terminal devices either over the S-100 bus, or thru four SCC connectors J1, J2, J3, and J4 located along the top edge of the board (see Figure 8). The S-100 bus is, of course, a general purpose interface, while connectors J1 - J4 provide the data paths for one serial and three parallel SCC I/O ports. A discussion of the SCC controlled S-100 bus is presented in Section 4.2 of this manual.

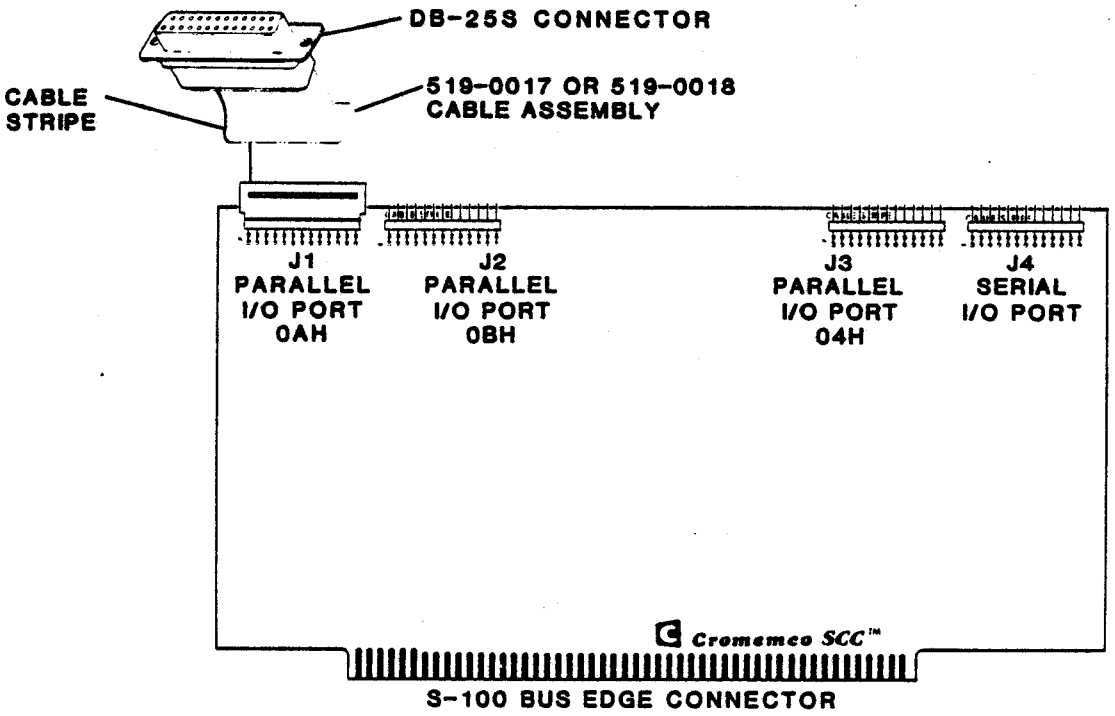


Figure 8: SCC CONNECTORS J1, J2, J3 & J4

Cromemco provides two interface cables which are plug compatible with SCC connectors J1 thru J4; Part No. 519-0017 (62 cm, $15) and Part No. 519-0018 (110 cm, $15). Each of these cable assemblies consists of a 26-pin female connector which mates

with one SCC terminal strip connector J1 - J4, a 25-conductor flat ribbon cable of length 62 or 110 cm, and a 25-pin female DB-25S EIA terminating connector. The terminal devices should be equipped with a mating 25-pin male DB-25P EIA connector when using either of these Cromemco supplied cables.

All J1 - J4 pin numbers referenced in this manual and in the SCC Schematic Diagram correspond to the DB-25s EIA pin numbers. Figure 9 illustrates the EIA numbering convention, and also shows the SCC terminal strip numbering convention.

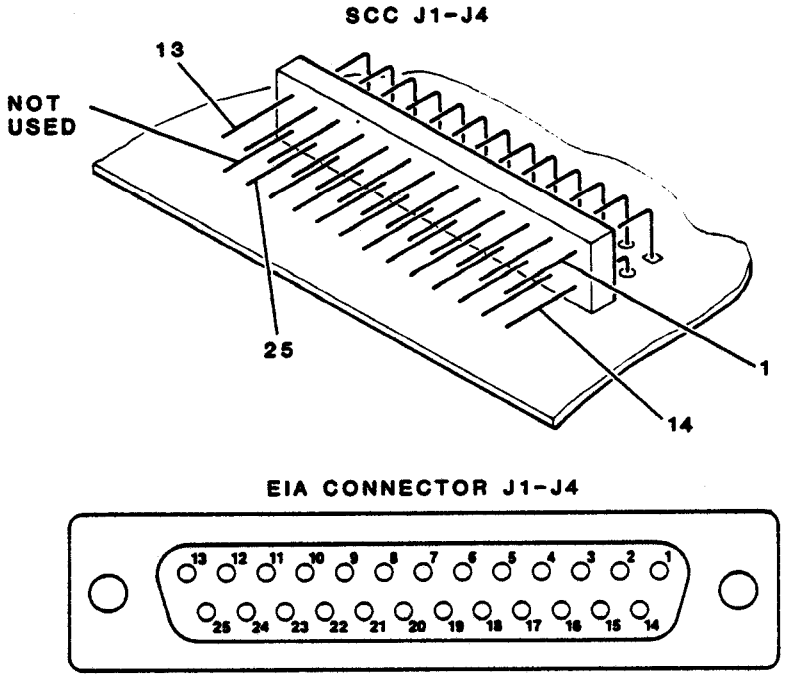


Figure 9: SCC PIN NUMBERING

Cromemco supplied cables must be installed by aligning the board legend arrow head near each connector with the ribbon cable stripe (the colored edge of the ribbon cable). With the cable stripe properly aligned, the pin-outs of SCC connectors J1 thru J4

are shown in Table 2.

Table 2

SCC CONNECTOR J1 THRU J4 PIN-OUTS

Pin	Connector J1	Connector J2	Connector J3	Connector J4
1	NOT USED	NOT USED	NOT USED	NOT USED
2	OUT 0AH, D0	OUT 0BH, D0	OUT 04H, D0	TxD
3	OUT 0AH, D1	OUT 0BH, D1	OUT 04H, D1	RxD
4	OUT 0AH, D2	OUT 0BH, D2	OUT 04H, D2	RTS
5	OUT 0AH, D3	OUT 0BH, D3	OUT 04H, D3	CTS
6	OUT 0AH, D4	OUT 0BH, D4	OUT 04H, D4	DSR
7	OUT 0AH, D5	OUT 0BH, D5	OUT 04H, D5	GROUND
8	OUT 0AH, D6	OUT 0BH, D6	OUT 04H, D6	NOT USED
9	OUT 0AH, D7	OUT 0BH, D7	OUT 04H, D7	NOT USED
10	DATA VALID 0AH	DATA VALID 0BH	NOT USED	NOT USED
11	WAIT	WAIT	NOT USED	NOT USED
12	NOT USED	NOT USED	OUTPUT ENABLE	NOT USED
13	NOT USED	NOT USED	RESET	NOT USED
14	IN 0AH, D0	IN 0BH, D0	IN 04H, D0	NOT USED
15	IN 0AH, D1	IN 0BH, D1	IN 04H, D1	NOT USED
16	IN 0AH, D2	IN 0BH, D2	IN 04H, D2	NOT USED
17	IN 0AH, D3	IN 0BH, D3	IN 04H, D3	+TTY KEYBD
18	IN 0AH, D4	IN 0BH, D4	IN 04H, D4	NOT USED
19	IN 0AH, D5	IN 0BH, D5	IN 04H, D5	NOT USED
20	IN 0AH, D6	IN 0BH, D6	IN 04H, D6	DTR
21	IN 0AH, D7	IN 0BH, D7	IN 04H, D7	DCD
22	+5 VDC	+5 VDC	NOT USED	NOT USED
23	GATE DATA 0AH	GATE DATA 0BH	INT	+TTY PRNTR
24	READ STB 0AH	READ STB 0BH	NOT USED	-TTY KEYBD
25	GROUND	GROUND	GROUND	-TTY PRNTR

Connector J1 is dedicated to SCC parallel I/O port 0AH, J2 to parallel I/O port 0BH and J3 to parallel I/O port 04H. The SCC parallel output ports are discussed in Section 2.6. Connector J4 is dedicated to the SCC serial I/O port, and it is discussed in Section 2.7.

2.6 SCC PARALLEL I/O PORTS

The SCC provides three parallel 8-bit output ports, and three parallel 8-bit input ports with fixed port addresses. Each of these six ports has its own dedicated uni-directional data lines (see SCC Schematic Diagram and Section 2.5). Two of these ports, IN 04H and OUT 04H, are an integral part of the 5501 Multi-Function I/O Controller (IC26), and the other four are implemented in either 74LS373 transparent octal latches (IN 0AH and IN 0BH), or 74LS273 octal latches with clear (OUT 0AH and OUT 0BH).

PARALLEL INPUT PORT 04H: A terminal device inputs parallel data to Z80A Register A thru port IN 04H by actively driving data lines D0 (LSB) thru D7 (MSB) on connector J3. These lines feed 74LS244 parts which represent 0.5 unit loads (logic 1) and 0.125 unit loads (logic 0), where a unit load is defined as a source current of 40 uAmps @ +2.4 VDC, and a sink current of 1.6 mAmps @ +0.4 VDC. The input data byte is not latched, but rather the Z80A must sample it in real time. The terminal device alerts the Z80A that data is available by either an interrupt on S-100 line $\overline{\text{INT}}$, or by a high-to-low transition on SCC line $\overline{\text{INT}}$. This transition causes a bit to be set in the 5501's internal interrupt register. The 5501 may be programmed to prioritize and pass along the interrupt to the Z80A, or the Z80A may poll the 5501 for the interrupt source (see Section 2.9). The Z80A finally reads IN 04H data by executing an IN A,04H instruction, which loads the data byte into Register A, the Z80A accumulator. Note that bit D7 may be programmatically defined as auxiliary interrupt input PI7 (see

Section 2.9).

PARALLEL OUTPUT PORT 04H: Parallel data is transferred from the Z80A to a terminal device via OUT 04H by first loading the data byte into Register A, then executing an OUT 04H,A instruction. Executing the output instruction causes the data byte to be routed, inverted and latched in the 5501's internal output register, which in turn presents the data to inverting 74LS240 tri-state drivers. The driver outputs feed SCC connector J3, and the drivers are brought out of the tri-state (floating) condition to actively drive D0 thru D7 in non-inverted form by asserting J3 line OUTPUT ENABLE active high. If OUTPUT ENABLE is left open, a pull-up resistor holds the line high thus permanently enabling active outputs. The 74LS240 drivers have a fan-out of 75 unit loads (logic 1) and 7.5 unit loads (logic 0). Note that an S-100 RESET does not clear port OUT 04H, nor does a POC. There are two other lines provided on connector J3; GROUND and $\overline{\text{RESET}}$. Asserting $\overline{\text{RESET}}$ active low resets the Z80A only; it does not connect to the S-100 bus line of the same name.

PARALLEL INPUT PORTS 0AH & 0BH: A terminal device inputs parallel data to the Z80A Register A thru ports IN 0AH and IN 0BH by actively driving data lines D0 thru D7 on either connector J1 (IN 0AH) or J2 (IN 0BH). Each set of lines feeds a 74LS373 transparent octal latch which represents 0.5 unit loads (logic 1) and 0.25 unit loads (logic 0). The input data byte may either be latched in the 74LS373 by a low-high-low transition on line GATE

DATA, or the Z80A may sample the data in real time if GATE DATA is held high or left floating (then the latch outputs follow its inputs). If data is to be latched, a 0 nSec set-up time, and a 10 nSec data hold time is required relative to the high-to-low transition on GATE DATA. The Z80A is alerted that new parallel input data is available again either thru S-100 bus line $\overline{\text{INT}}$, or thru J3 line $\overline{\text{INT}}$. The Z80A reads the parallel input data by executing an IN A,0AH or an IN A,0BH, which loads the data byte into Z80A Register A. Status lines READ STB pulse low while the Z80A is reading a 74LS373 latch, and its low-to-high transition is used by the terminal device to signal that the latch is ready for new data (see Figure 10).

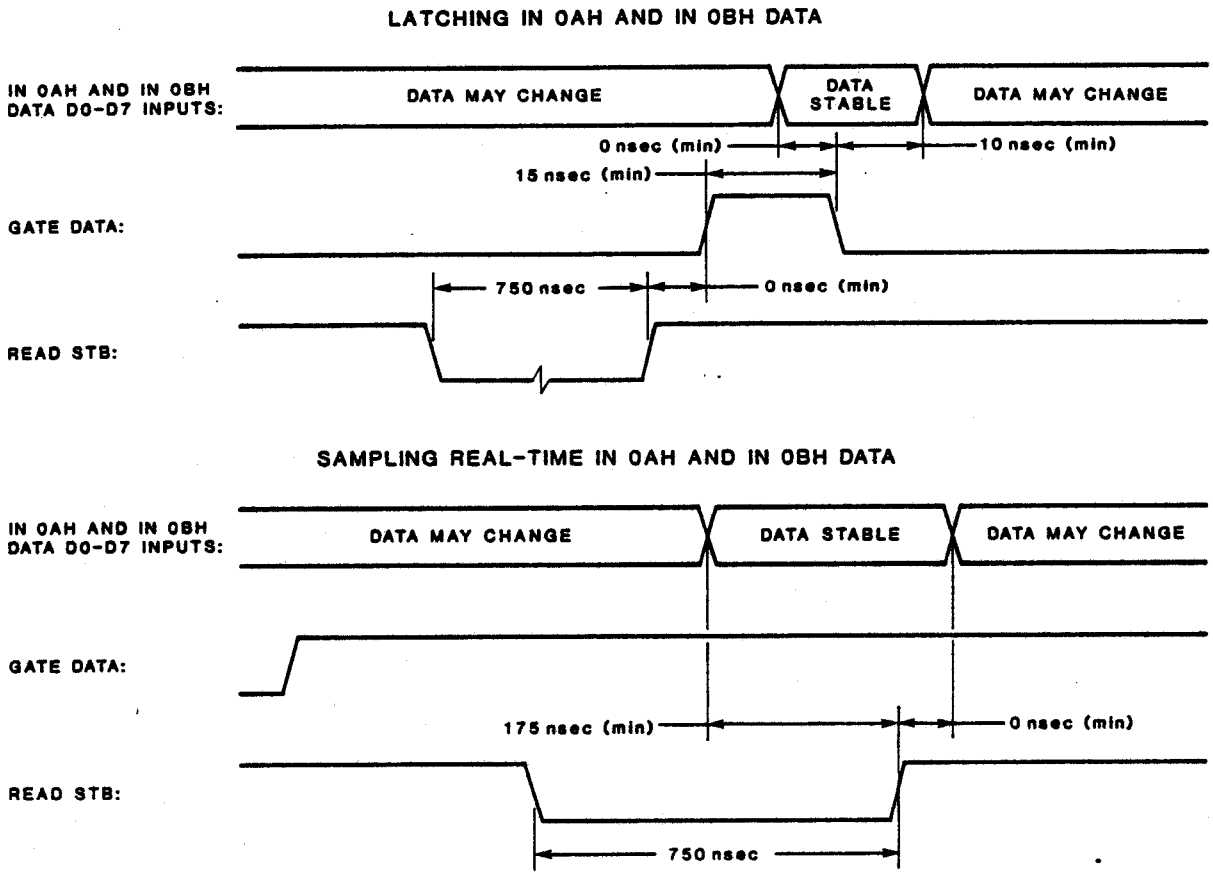


Figure 10: IN 0AH AND IN 0BH TIMING

PARALLEL OUTPUT PORTS 0AH & 0BH: Parallel data is transferred to a terminal device thru port OUT 0AH by first loading the data byte into Z80A Register A, then executing an OUT 0AH,A instruction. Likewise, port OUT 0BH is used by loading a data byte into Register A, then executing an OUT 0BH,A instruction. Executing the output instruction routes the data byte to a 74LS273 octal latch, then causes the port's associated DATA VALID line to pulse low (see Figure 11). The low-to-high transition on DATA VALID signals the terminal device that valid parallel data is available at the 74LS273 outputs D0 thru D7. Each data line has a fan-out

of 20 unit loads (logic 1) and 10 unit loads (logic 0). An S-100 RESET or a POC clears both OUT 0AH and OUT 0BH. Note that bit D7 of OUT 0AH controls SCC memory enable/disable if the MEMORY DISABLE option has been selected (see Section 2.4); otherwise D7 functions as an ordinary MSB data bit.

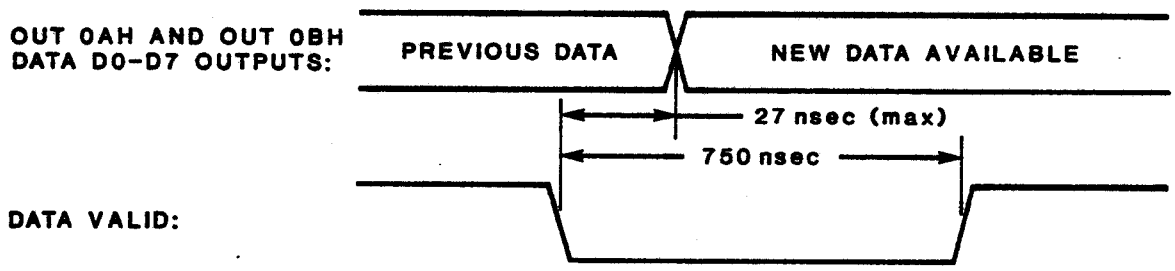


Figure 11: OUT 0AH AND OUT 0BH TIMING

Connectors J1 and J2 also supply +5 VDC (pin 22), GROUND (pin 25), and a $\overline{\text{WAIT}}$ line (pin 11) to external terminal devices. The $\overline{\text{WAIT}}$ line may be used to insert wait states to synchronize the Z80A to slow I/O devices. Example circuitry for this purpose is shown in Figure 12. To synchronize a slow output device connected to either OUT 0AH or OUT 0BH, the trailing edge of DATA VALID is used to force line $\overline{\text{WAIT}}$ active low. The Z80A then cycles in the wait state until user-defined line DATA ACCEPTED goes active low after the data has been read from the SCC output port, thus releasing the Z80A to continue program execution.

To synchronize a slow input device connected to either IN 0AH or IN 0BH, the leading edge of READ STB is used to force line $\overline{\text{WAIT}}$

active low. The Z80A then cycles in the wait state until user-defined line DATA AVAILABLE goes active low signaling that the input data has either stabilized (if sampling in real time), or has been latched by using the GATE DATA line. Line $\overline{\text{WAIT}}$ then goes high which allows the Z80A to complete the input cycle, and continue normal program execution.

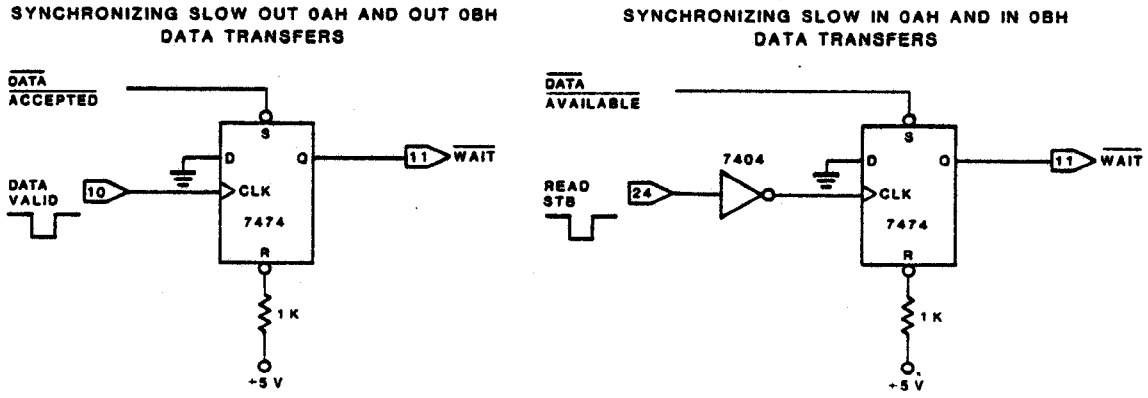


Figure 12: SYNCHRONIZING SLOW OUT 0AH & OUT 0BH I/O

2.7 SCC SERIAL I/O

The primary serial data paths between the SCC and a serial terminal device are shown in Figure 13. Z80A Register A is the source register for data bytes transmitted to the terminal, and is also the destination register for data bytes received from the terminal. The 5501 Multifunction I/O Controller functions to convert parallel CPU data to serial terminal data, and vice versa, and additionally to supply serial status information and serial data control functions. TTL-level signals on the terminal side of the 5501 are converted to and from either RS-232 or 20 mA current

loop terminal data by SCC interface circuitry. Serial data to and from the terminal interfaces the SCC at connector J4. TRANSMITTER DATA from the SCC travels to an RS-232 terminal over line RxD, J4 pin 3, while RECEIVER DATA to the SCC travels from the terminal over line TxD, J4 pin 2 (RS-232 interface signals are labeled from the terminal's viewpoint).

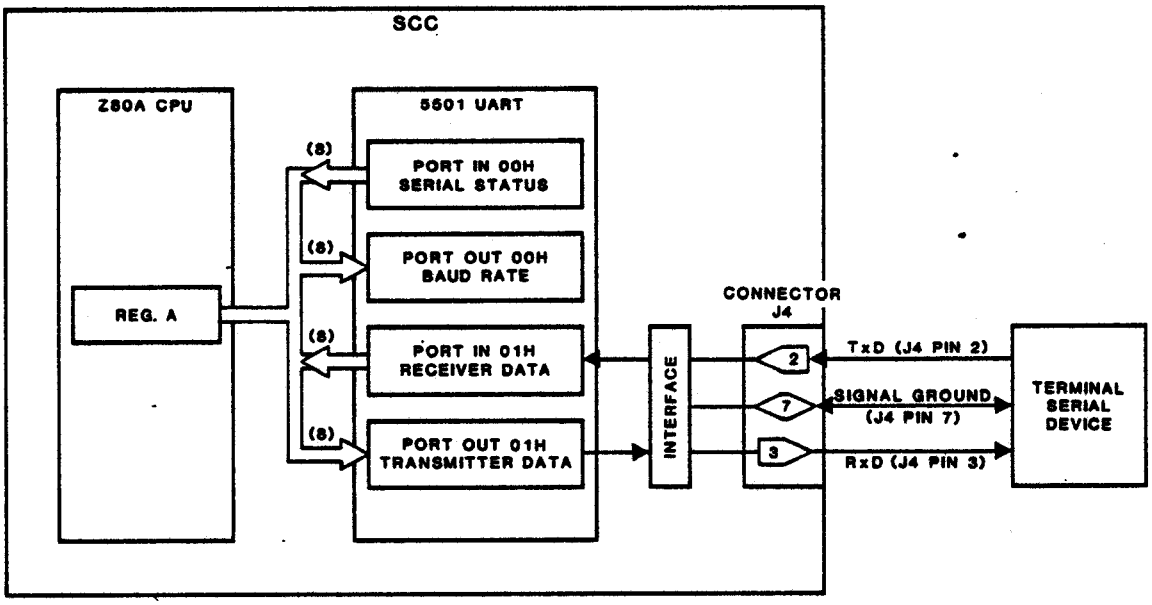


Figure 13: SCC PRIMARY SERIAL DATA PATHS

Notice that even though a single 'serial terminal device' is used in the discussions and examples which follow, it is not the intention to exclude uni-directional serial devices such as printers, keyboards, and the like. The SCC serial input and

output channels may be managed independently with any mix of RS-232 and 20 mA current loop interfaces. The only practical restriction is a common serial transmit and receive baud rate.

The Z80A may determine status conditions Read Data Available (RDA) and Transmitter Buffer Empty (TBE) by periodically sampling SERIAL STATUS port IN 00H, or by programming the 5501 so that RDA and TBE conditions issue an interrupt request and their own RST instruction vectors to the Z80A (an interrupt-driven configuration), or by programming the 5501 so that RDA and TBE conditions issue a non-vectored interrupt request, followed by Z80A polling of INTERRUPT ADDRESS port IN 03H to determine the interrupt source (a polled-interrupt configuration). SCC interrupt behavior is topically covered in Sections 2.9 and 2.10, including serial I/O examples. The reader is referred to these sections for a thorough discussion of interrupt-related topics which are only touched upon in this section.

There are seven SCC I/O ports which may play a role in serial data exchanges; SERIAL STATUS (IN 00H), BAUD RATE (OUT 00H), RECEIVER DATA (IN 01H), TRANSMITTER DATA (OUT 01H), COMMAND REGISTER (OUT 02H), INTERRUPT ADDRESS (IN 03H) and INTERRUPT MASK (OUT 03H).

SERIAL STATUS, (IN 00H):

The Z80A interrogates this port by executing an IN A,(00H) instruction to determine the 5501 serial status. Individual bits of the status word are interpreted as follows:

BIT	LABEL	FUNCTION
D7	TBE	Transmitter Buffer Empty. The Transmitter Buffer is a temporary storage register which buffers the data byte sent to TRANSMITTER DATA (port OUT 01H). A logic 1 TBE indicates that the Transmitter Buffer is ready to accept a new data byte. TBE goes to logic 1 immediately after the buffer contents are transferred to the serial transmitter, thus the new data byte may be loaded into the buffer while transmission of the previous byte is still in progress. A TBE condition generates an interrupt request which may be either enabled or disabled by bit D5 of INTERRUPT MASK port OUT 03H (see Section 2.9). TBE is cleared to logic 0 when the Transmitter Buffer is loaded, and it is set to logic 1 after a 5501 RESET issued thru COMMAND REGISTER bit RES. TBE status is also available at bit D4 (see below).
D6	RDA	Read Data Available. A logic 1 RDA indicates that a data byte is available and may be read from RECEIVER DATA (port IN 01H). If a new data byte has arrived in its entirety before the previous contents of RECEIVER DATA have been read, the new byte will overwrite the old, and the OVERRUN error flag will be set to logic 1. An RDA condition generates an interrupt request which may be either enabled or disabled by bit D4 of INTERRUPT MASK port OUT 03H (see Section 2.9). The RDA flag is reset to logic 0 by either inputting RECEIVER DATA by executing an IN A,(01H) instruction, or by a 5501 RESET. RDA status is also available at bit D3 (see below).
D5	IPG	Interrupt Pending. A logic 1 IPG indicates that one or more of the eight possible prioritized interrupt request sources is un-masked (enabled) and is currently requesting service. Bit IPG provides a means of sensing a service request from 5501 sources

when Z80A interrupts are disabled. In this mode (I/O port-driven service), the Z80A would read SERIAL STATUS port IN 00H under program control, test bit IPG, and if set to logic 1 (indicating an un-masked source is requesting service), INTERRUPT ADDRESS port IN 03H would then be read to identify the request source. EXAMPLE 5 below discusses a serial I/O programming example which uses IPG in this fashion.

D4 TBE Transmitter Buffer Empty (identical to bit D7 described above).

D3 RDA Read Data Available (identical to bit D6 described above).

D2 SRV Serial Receive. A logic 1 SRV indicates a high level (marking) on the serial data input line, and a logic 0 SRV indicates a low level (spacing) on the serial data input line. SRV goes to logic 1 when no data is being received, and is provided for break detection and test purposes.

D1 ORE Overrun Error. A logic 1 ORE indicates that a new RECEIVER DATA byte has overwritten a previous byte without the previous byte being read. ORE is reset to logic 0 after the SERIAL STATUS port is read, or by a 5501 RESET.

D0 FME Frame Error. A logic 1 FME indicates an error in one or both stop bits which "framed" the last RECEIVER DATA byte. FME remains at logic 1 until a valid character is received.

BAUD RATE, (OUT 00H):

STOP BITS	9600	4800	2400	1200	300	150	110

The Z80A sets the serial data transmit and receive baud rate, plus the number of stop bits, by outputting a control byte to this output port. The port is typically loaded as a system initialization task. A logic 0 output to bit D7 (STOP BITS) selects two stop bits, while a logic 1 to bit D7 selects one stop bit. A logic 1 in any bit position D6 thru D0 selects a baud

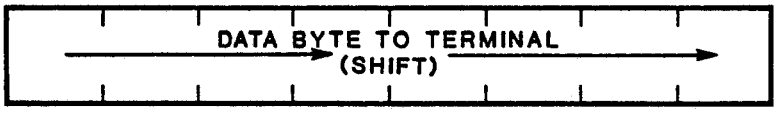
rate, with the correspondence tabulated below. If more than one of these bits is set to logic 1, the highest baud rate selected will result. If no bits D6 thru D0 are set to logic 1, the serial transmitter will be disabled. If COMMAND REGISTER bit D4 (High Baud, or HBD) is set to logic 1, the baud rate will undergo an eightfold increase. The bit assignments follow:

BIT	FUNCTION	
D7	STOP BITS	
D6	9,600 BAUD	76,800 BAUD
D5	4,800 BAUD	38,400 BAUD
D4	2,400 BAUD	19,200 BAUD
D3	1,200 BAUD	9,600 BAUD
D2	300 BAUD	2,400 BAUD
D1	150 BAUD	1,200 BAUD
D0	110 BAUD	880 BAUD
	(HBD = LOGIC 0)	(HBD = LOGIC 1)

RECEIVER DATA, (IN 01H):

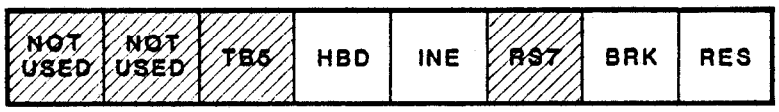
The Z80A reads an assembled data byte from the serial terminal device by reading this port. The Z80A is alerted that data is available either by testing SERIAL STATUS bit IPG and then polling INTERRUPT ADDRESS port IN 03H for the encoded RDA address, or by directly sampling SERIAL STATUS bit RDA. Note that status bit RDA is always valid, even if RDA has been masked-off as an interrupt source thru INTERRUPT MASK port OUT 03H.

TRANSMITTER DATA, (OUT 01H):



The Z80A writes to this output port to send data to the serial terminal device. The Z80A is alerted that the Transmitter Buffer is empty either by testing SERIAL STATUS bit IPG and then polling INTERRUPT ADDRESS port IN 03H for the encoded TBE address, or by directly sampling SERIAL STATUS bit TBE. Note that status TBE is always valid, even if TBE has been masked-off as an interrupt source thru INTERRUPT MASK port OUT 03H.

COMMAND REGISTER, (OUT 02H):



The Z80A configures the 5501 by loading this output port. Three COMMAND REGISTER bits may have an effect on SCC serial I/O operations; HBD, INE, BRK and RES.

BIT	LABEL	FUNCTION
D4	HBD	High Baud. Setting HBD to logic 1 octuples the 5501 internal clock frequency, causing the serial transmit and receive baud rate to octuple.
D3	INE	INTA Enable. The 5501 monitors the data bus at SYNC of each machine cycle to determine if the Z80A is entering an Interrupt Acknowledge (INTA) cycle. Resetting INE to logic 0 causes the 5501 to ignore all INTA cycles. Setting INE to logic 1 causes the TMS 5501 to accept an INTA, and gate one-of-eight RST instructions onto the data bus to service the interrupt. See Sections 2.9 and 2.10 for discussions on interrupt servicing RDA and TBE requests.

D1 BRK Break. Setting BRK to logic 1 latches the serial transmitter output XMT in the low (spacing) state. Resetting BRK to logic 0 allows the serial transmitter to operate normally. RES overrides BRK.

D0 RES 5501 Reset. Setting RES to logic 1 results in the following:

(1) SERIAL STATUS bits RDA, SBD, FBD and ORE are reset to logic 0; the serial receiver goes into the 'search for new character' mode; the current contents of RECEIVER DATA, port IN 01H, is not affected and contains the last character received.

(2) SERIAL STATUS bit TBE is set to logic 1; the serial transmitter output XMT goes high (marking). RES overrides BRK if both are set to logic 1 in the same command byte.

(3) RES is automatically reset to logic 0 after the steps listed above (and some other reset operations described in Section 2.9) are completed. The 5501 is then ready for 'normal' serial I/O operations.

INTERRUPT ADDRESS, (IN 03H):

1	1	L_2	L_1	L_0	1	1	1

This port is provided for identifying 5501 service request sources by polling. If interrupts are disabled, the Z80A may still sense an interrupt pending condition by testing SERIAL STATUS bit IPG, and then read INTERRUPT ADDRESS to determine the interrupt source. Interrupt sources TBE and RDA, along with six others, may be determined by reading and properly interpreting the contents of this port. Reading INTERRUPT ADDRESS contents 0E7H implies an RDA condition; reading an 0EFH implies a TBE condition. An RDA request has priority over a TBE request if both are simultaneously active. For full details on all eight interrupt sources and their priorities, see Section 2.9.

INTERRUPT MASK, (OUT 03H):

Individual bits output to this port selectively enable or inhibit the eight 5501 sources from issuing an interrupt request to the Z80A, and from being prioritized and encoded in the INTERRUPT ADDRESS register. Setting INTERRUPT MASK bit D5 to logic 1 enables interrupt source TBE as described above; resetting bit D5 to logic 0 inhibits it. Likewise, bit D4 controls interrupt source RDA. Note again that SERIAL STATUS bits RDA and TBE are always valid, regardless of the control bits D4 and D5 output to the INTERRUPT MASK port.

Three serial I/O example programs will now be presented. The program segments which follow are assumed resident in 2716 socket ROM 0 (0000H - 07FFH). The first example deals with initializing an SCC system connected to a serial terminal.

EXAMPLE 3

This example assumes an SCC system utilizing SCC on-board memory only, and an RS-232 CRT terminal attached to SCC serial connector J4. An S-100 bus RESET or a POC automatically starts program execution at address 0000H, so the routine begins there by moving the stack out to the end of SCC read/write memory, then by forcing a jump to the 5501 initialization routine starting at address 100H. The main initialization code is located at 100H so as to leave low memory free for RST 08H thru RST 38H service routines. Notice that the RST 00H service area is occupied by the first two instructions below. The 5501 is initialized by loading three registers: BAUD RATE (OUT 00H); COMMAND REGISTER (OUT 02H); and INTERRUPT MASK (OUT 03H). This routine configures the SCC to transfer serial data exclusively thru reading and loading I/O ports, not by responding to interrupt requests. Note that an S-

100 bus RESET or POC automatically disables interrupts, thus the Z80A will simply ignore any active low request on its INT input pin. The SCC system is configured as follows:

(1) The serial data baud rate is set to 9600, one stop bit.
(2) The 5501 internal clock is set to low baud (HBD = logic 0).
(3) The 5501 is programmed to ignore INTA cycles. This is for emphasis only, since interrupts are disabled and the Z80A will not enter an INTA cycle.
(4) Bit D7 of port OUT 0AH is used as an ordinary data bit, not as auxiliary interrupt source PI7.
(5) The 5501 is reset.
(6) All 5501 interrupt sources are un-masked, so that they may be polled thru INTERRUPT ADDRESS port IN 03H (see EXAMPLE 5 below).

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                    0001 ;
                    0002 ;   >>>  INITIALIZE SCC 5501 ROUTINE  <<<
                    0003 ;
                    0004 ;
                    0005 ;
     (0000)         0006 BAUD:   EQU  00H              ;BAUD RATE PORT
     (0002)         0007 COMREG: EQU  02H              ;COMMAND REG. PORT
     (0003)         0008 INTMSK: EQU  03H              ;INTERRUPT MASK PORT
     (23FF)         0009 ENDRAM: EQU  23FFH            ;SCC END RAM ADDR.
                    0010                               ;
                    0011         ORG  0                ;SCC POC OR RESET
                    0012                               ;
0000 31FF23         0013         LD   SP,ENDRAM        ;STACK TO ENDRAM
0003 C30001         0014         JP   IN5501           ;JUMP TO INIT.
                    0015                               ;
                    0016                               ;
                    0017 ;  INITIALIZE 5501
                    0018                               ;
                    0019         ORG  100H             ;ROUTINE START
                    0020                               ;
0100 F3             0021 IN5501: DI                    ;(EMPHASIS ONLY)
0101 3E01           0022         LD   A,00000001B      ;COMMAND BYTE
0103 D302           0023         OUT  COMREG,A         ;TO COMMAND REG.
0105 3EC0           0024         LD   A,11000000B      ;BAUD/STOP BIT BYTE
0107 D300           0025         OUT  BAUD,A           ;TO BAUD RATE REG.
0109 3EFF           0026         LD   A,11111111B      ;INTERRUPT MASK
010B D303           0027         OUT  INTMSK,A         ;UN-MASK ALL SOURCES
010D C30002         0028         JP   COMLIN           ;CONTINUED NEXT EXAMPLE
0110 (0100)         0029         END  IN5501
```

The next programming example illustrates serial I/O managed

by SERIAL STATUS bits RDA and TBE.

EXAMPLE 4

Continuing the example above, assume that after system initialization, the SCC system awaits a 'command line', which the user enters from the CRT keyboard, to determine further actions. The program segment below then receives a serial data byte from the terminal, echoes it back, stores it sequentially in a read data buffer with previously received bytes starting at RAM location RDBUFF, and then awaits another data byte. Character Carriage Return <CR> is defined to be the command line delimiter, and it is echoed to the terminal as CR-LF (Carriage Return followed by a Line Feed). After a CR character is stored in the read buffer, program execution continues by jumping to an undefined 'command line interpretation' program segment, with an assumed starting address of COMITP. Notice that RDA and TBE conditions are determined by sampling the SERIAL STATUS port, and that an overrun error, a framing error, or a command line in excess of 72 characters resets the RAM read buffer and causes the message 'RE-ENTER LINE' to be sent to the terminal.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                        0001 ;
                        0002 ;  >>>  COMMAND LINE INPUT ROUTINE <<<
                        0003 ;
                        0004 ;
      (0300)            0005 COMITP: EQU  300H              ;ARBITRARY ROM 0 ADDR
      (000D)            0006 CR:     EQU  0DH               ;ASCII 'CR'
      (0003)            0007 ERRMSK: EQU  00000011B         ;ORE & FME MASK
      (000A)            0008 LF:     EQU  0AH               ;ASCII 'LF'
      (0048)            0009 MAXLEN: EQU  72                ;MAX LINE LENGTH
      (007F)            0010 MSBMSK: EQU  01111111B         ;D7 (MSB) MASK
      (0006)            0011 RDA:    EQU  6                 ;RDA BIT D6
      (0001)            0012 RDATA:  EQU  01H               ;RECEIVER DATA PORT
      (2000)            0013 RDBUFF: EQU  2000H             ;START READ BUFF
      (0000)            0014 STATUS: EQU  00H               ;SERIAL STATUS PORT
      (0007)            0015 TBE:    EQU  7                 ;TBE BIT D7
      (0001)            0016 TDATA:  EQU  01H               ;XMITTER DATA PORT
                        0017                                ;
                        0018         ORG  200H              ;COMLIN START ADDR
                        0019                                ;IN SCC ROM 0
0200  210020            0020 COMLIN: LD   HL,RDBUFF         ;POINT TO READ BUFF
0203  0E48              0021         LD   C,MAXLEN          ;MAX LINE LENGTH
0205  DB00              0022 TSTRDA: IN   A,STATUS          ;READ SERIAL STATUS
0207  CB77              0023         BIT  RDA,A             ;RDA?
0209  28FA              0024         JR   Z,TSTRDA          ;RE-CHECK RDA IF NO
020B  E603              0025         AND  A,ERRMSK          ;RDA.  ORE OR FME?
020D  201C              0026         JR   NZ,ERROR          ;OUT MESSAGE IF SO
020F  0D                0027         DEC  C                 ;DECR BYTE COUNT
```

```
0210  2819          0028          JR   Z,ERROR            ;OUT MESSAGE IF ZERO
0212  DB01          0029          IN   A,RDATA            ;ELSE READ BYTE
0214  E67F          0030          AND  MSBMSK             ;MASK-OFF MSB
0216  77            0031          LD   (HL),A             ;CHAR TO BUFFER
0217  47            0032          LD   B,A                ;AND TO REG. B
0218  CD4B02        0033          CALL OUTCHR             ;ECHO CHARACTER
021B  3E0D          0034          LD   A,CR               ;WAS IT A 'CR'?
021D  B8            0035          CP   B                  ;
021E  2803          0036          JR   Z,COMEND           ;COM LINE END IF SO
0220  23            0037          INC  HL                 ;MOVE BUFFER POINTER
0221  18E2          0038          JR   TSTRDA             ;BACK FOR NEXT CHAR
                    0039                                  ;
0223  060A          0040 COMEND:  LD   B,LF               ;ASCII 'LF' TO REG. B
0225  CD4B02        0041          CALL OUTCHR             ;'LF' TO TERMINAL
0228  C30003        0042          JP   COMITP             ;COMLIN ROUTINE EXIT
                    0043                                  ;
022B  213A02        0044 ERROR:   LD   HL,MESBEG          ;POINT MESSAGE BEGIN
022E  0E11          0045          LD   C,MESEND-MESBEG    ;MESSAGE BYTE COUNT
0230  46            0046 NXTCHR:  LD   B,(HL)             ;CHAR TO REG. B
0231  CD4B02        0047          CALL OUTCHR             ;CHAR TO TERMINAL
0234  0D            0048          DEC  C                  ;DECR BYTE COUNT
0235  28C9          0049          JR   Z,COMLIN           ;RESTART COMLIN IF 0
0237  23            0050          INC  HL                 ;ELSE MOVE POINTER
0238  18F6          0051          JR   NXTCHR             ;OUT NEXT MESS. CHR
                    0052                                  ;
023A  0D0A          0053 MESBEG:  DB   CR,LF              ;BEGIN MESSAGE
023C  52452D45      0054          DB   'RE-ENTER LINE'    ;:
      4E544552
      204C494E
      45
0249  0D0A          0055          DB   CR,LF              ;END MESSAGE
      (024B)        0056 MESEND:  EQU  $                  ;
                    0057                                  ;
                    0058 ;
                    0059 ;  >>>  SUBROUTINE OUTCHR  <<<
                    0060 ;
                    0061 ;  CALL WITH ASCII CHARACTER IN REG. B.
                    0062 ;  RETURNS WITH REG. A & FLAGS CHANGED.
                    0063 ;
024B  DB00          0064 OUTCHR:  IN   A,STATUS           ;READ SERIAL STATUS
024D  CB7F          0065          BIT  TBE,A              ;TBE?
024F  28FA          0066          JR   Z,OUTCHR           ;LOOP BACK IF NOT
0251  78            0067          LD   A,B                ;TBE. GET CHARACTER
0252  D301          0068          OUT  TDATA,A            ;CHAR TO TERMINAL
0254  C9            0069          RET                     ;RETURN
                    0070                                  ;
0255  (0200)        0071          END     COMLIN
```

The next example illustrates how INTERRUPT ADDRESS port IN

03H may be used for serial data exchange in an I/O port-driven (Z80A interrupts disabled) environment. Note that when interrupts are disabled (by a RESET, a POC or executing a DI instruction), and when one or more 5501 interrupt sources are un-masked (enabled), then an interrupt request from one of these sources will force Z80A pin $\overline{INT}$ low, but the Z80A will ignore the request since interrupts are disabled. The Z80A discovers that an interrupt is pending by sampling SERIAL STATUS bit IPG, and may then read the highest priority enabled interrupt source requesting service by reading INTERRUPT ADDRESS port IN 03H.

EXAMPLE 5

The following program segment assumes the SCC system has been initialized as per EXAMPLE 3 prior to execution. The routine initiates a 10 mSec timing interval using TIMER 4, and at the end of the interval outputs a period character to the terminal. This process is repeated indefinitely until any terminal key is pressed, which causes a branch to address EXIT (to an undefined program segment). Note that all interrupt sources other than RDA and TIMER 4 are masked-off (disabled). If an RDA condition occurs, IPG is set to logic 1, and the contents of INTERRUPT ADDRESS port IN 03H becomes 0E7H. When TIMER 4 reaches DELAY COUNT zero (one count per 64 uSec), IPG is set to logic 1, and the INTERRUPT ADDRESS becomes 0F7H. The INTERRUPT ADDRESS contents will be 0E7H if RDA and TIMER 4 are both active, since RDA has higher priority than TIMER 4.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                        0001 ;
                        0002 ;     >>> I/O PORT POLLING <<<
                        0003 ;
                        0004 ;
                        0005 ;
      (0500)            0006 EXIT:    EQU   500H              ;ARBITRARY ROM 0 ADDR.
      (0003)            0007 INTADR:  EQU   03H               ;INTERRUPT ADDR. PORT
      (0003)            0008 INTMSK:  EQU   03H               ;INTERRUPT MASK PORT
      (0005)            0009 IPG:     EQU   05H               ;INT. PENDING BIT D5
      (0000)            0010 STATUS:  EQU   00H               ;SERIAL STATUS PORT
      (0001)            0011 TDATA:   EQU   01H               ;XMITTER DATA PORT
      (009C)            0012 TENMS:   EQU   9CH               ;10 MSEC TIMER COUNT
      (0008)            0013 TIMER4:  EQU   08H               ;TIMER 4 PORT
                        0014                                  ;
0000'                   0015          ORG   400H              ;ROUTINE START
                        0016                                  ;
0400  3E50              0017 IOPOLL:  LD    A,01010000B       ;INTERRUPT MASK ALL
0402  D303              0018          OUT   INTMSK,A          ;BUT RDA & TIMER 4
0404  3E9C              0019 CYCLE:   LD    A,TENMS           ;GET TIMER COUNT
0406  D308              0020          OUT   TIMER4,A          ;START TIMER 4
0408  DB00              0021 CHKIPG:  IN    A,STATUS          ;GET SERIAL STATUS
040A  CB6F              0022          BIT   IPG,A             ;IPG?
040C  28FA              0023          JR    Z,CHKIPG          ;RE-CHECK IF NO
040E  DB03              0024          IN    A,INTADR          ;YES.  READ INT. ADDR.
0410  FEE7              0025          CP    0E7H              ;RDA?
0412  CA0005            0026          JP    Z,EXIT            ;YES, EXIT.
0415  3E2E              0027          LD    A,'.'             ;NO, MUST BE TIMER 4
0417  D301              0028          OUT   TDATA,A           ;PERIOD TO TERMINAL
0419  18E9              0029          JR    CYCLE             ;BACK, RE-LOAD TIMER
041B  (0400)            0030          END   IOPOLL
```

The serial I/O programs discussed thus far have been written in Z80 assembly language. If programming in Cromemco's 3K Control Basic, then all serial I/O may be left to the management of the interpreter. Alternately, a 3K Control Basic program can manage 'non-standard' serial I/O thru its IN (I/O port read) function and its OUT (I/O port write) and CALL (assembly language subroutine) instructions, provided their use is consistent with the serial I/O specifications and descriptions given above.

A schematic of the RS-232 / 20 mA current loop serial

interface appears in Figure 14. A 20 mA current loop interface involves four lines: TTY KEYBD (+); TTY KEYBD (-); TTY PRNTR (+); and TTY PRNTR (-). An RS-232 interface would typically involve at least signal lines TxD, RxD and GND, and may optionally define Modem Control lines RTS (Request to Send), CTS (Clear to Send), DSR (Data Set Ready), DTR (Data Terminal Ready) and DCD (Data Carrier Detect).

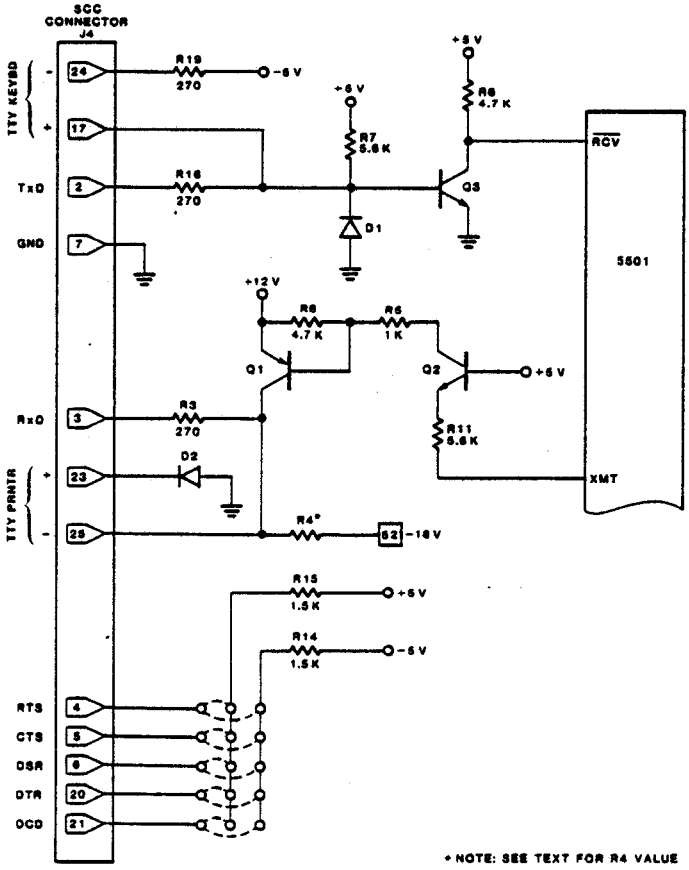


Figure 14: SCC SERIAL INTERFACE

An ASR-33 TTY interface is illustrated in Figure 15. If the SCC 20 mA current loop interface is used, SCC resistor R4 must be

replaced with an 850 ohm, 1 watt resistor . If the current loop interface is not used, leave the factory installed 3.3 Kohm, 1/4 watt resistor R4 in place to reduce power consumption and heat. Caution: 120 VAC is present on ASR-33 terminal strip 'BL'.

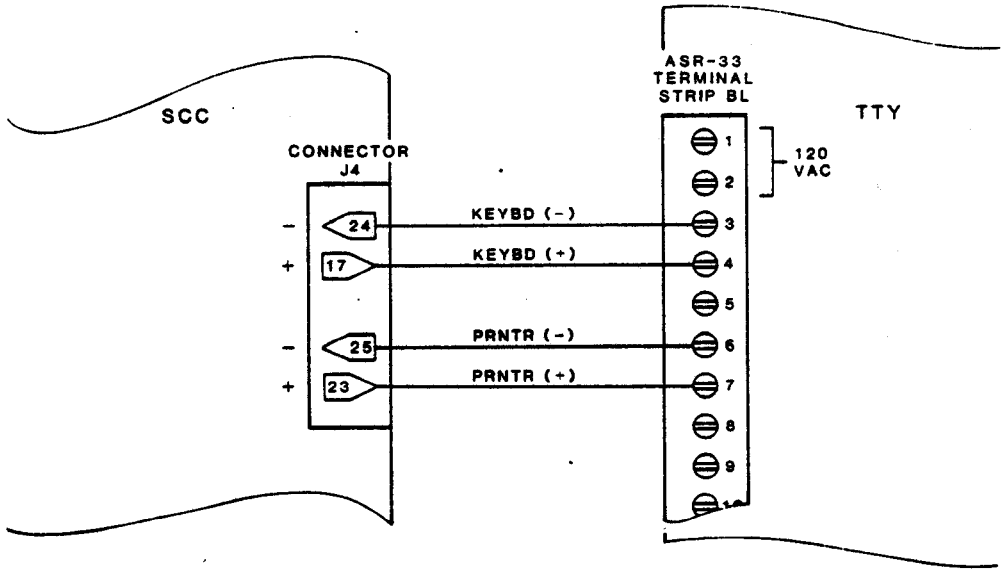


Figure 15: ASR-33 TO SCC INTERFACE

When using the SCC RS-232 interface, Modem Control lines RTS, CTS, DSR, DTR and DCD may either be left floating (factory shipped condition), or they may be selectively defined high (marking) or low (spacing) by installing insulated jumper wires between selected soldering pads as illustrated in Figure 17. The RS-232 high (spacing) level output by the SCC is +5 VDC, and the low (marking) level is -5 VDC; the SCC will accept any input levels which meet RS-232 specifications. Note that Cromemco supplied interface cables are terminated in an EIA DB-25S female connector, and the pin-outs of these cables meet RS-232 pin definitions except for pins 17 (+TTY KEYBD), 23 (+TTY PRNTR), 24 (-TTY KEYBD)

and 25 (-TTY PRNTR). Exercise caution when connecting this cable to devices which may make different use of these lines.

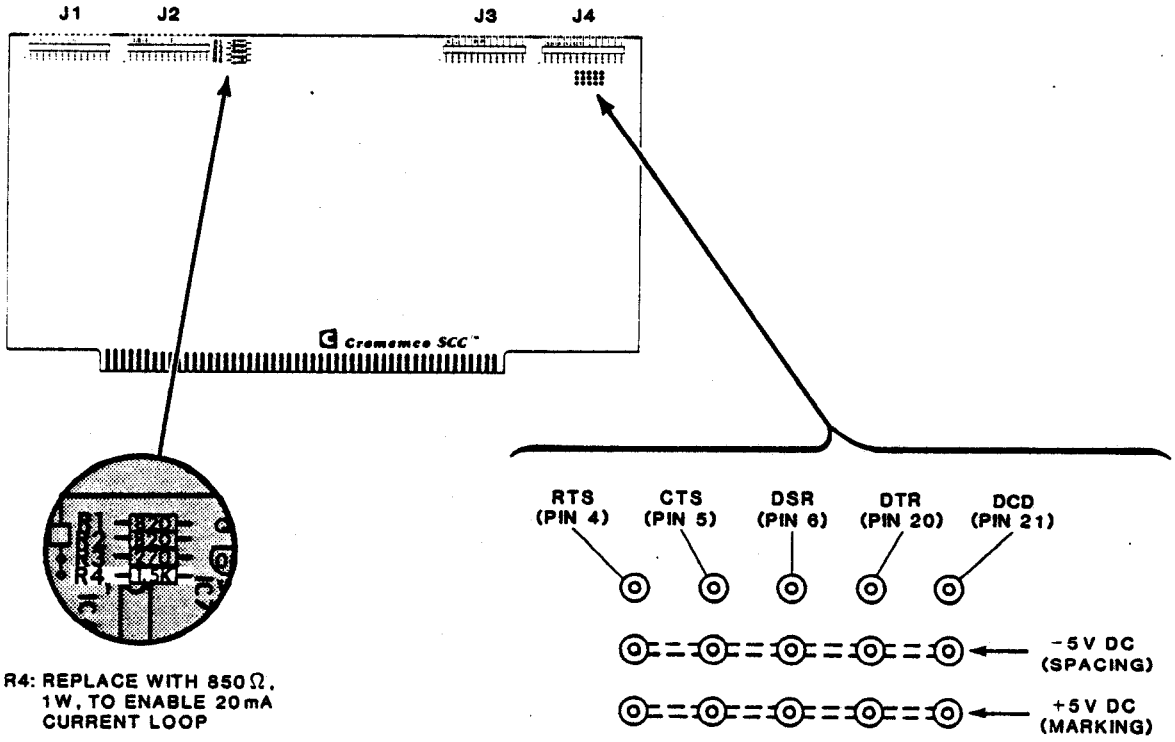


Figure 16: SCC MODEM CONTROL JUMPER PADS

2.8 SCC INTERVAL TIMERS

The SCC provides five independent interval timers, referred to explicitly as TIMER 1 thru TIMER 5 in this manual (the five timers are an integral part of the 5501 Multifunction I/O Controller). The SCC timers are selectively enabled for use, or disabled, by bits of INTERRUPT MASK port OUT 03H (see below). This is usually done as part of the system initialization routine. Setting an INTERRUPT MASK to logic 1 enables its associated timer, resetting a bit disables it.

INTERRUPT MASK BITS (OUT 03H)

(LOGIC 1 = ENABLE; LOGIC 0 = DISABLE)

BIT	FUNCTION
D7	TIMER 5 OR PI7
D6	TIMER 4
D5	TBE
D4	RDA
D3	TIMER 3
D2	INT (J4 PIN 23)
D1	TIMER 2
D0	TIMER 1

Another initialization task is choosing between TIMER 5 and auxiliary interrupt input PI7 (MSB bit D7 of port IN 04H) as a possible interrupt source. PI7 is selected by setting COMMAND REGISTER bit RS7 to logic 1, while resetting RS7 to logic 0 selects TIMER 5.

Each timer is assigned an SCC output port (see below), which may be loaded with a one-byte DELAY COUNT (00H - 0FFH, or 0 thru 255 decimal). After initial loading, the DELAY COUNT is decremented once every 64 uSec if COMMAND REGISTER bit HBD = logic 0, or once every 8 uSec if HBD = logic 1. The maximum delay from initial loading to timeout (DELAY COUNT zero) is then 16.32 mSec, and the accuracy from loading to timeout is plus 0 uSec and minus 64 uSec (or minus 8 uSec if HBD = logic 1).

SCC TIMER PORT ASSIGNMENTS

TIMER 1	OUT 05H
TIMER 2	OUT 06H
TIMER 3	OUT 07H
TIMER 4	OUT 08H
TIMER 5	OUT 09H

The Z80A is alerted that an enabled (un-masked) timer has reached DELAY COUNT zero in one of three ways:

(1) If interrupts are disabled , the Z80A finds SERIAL STATUS bit IPG set to logic 1. INTERRUPT ADDRESS port IN 03H is then read to identify the interrupt request source. Reading the INTERRUPT ADDRESS contents causes IPG to be reset to logic 0 if no other interrupt requests are pending. Notice that this method involves only reading SCC input ports SERIAL STATUS and INTERRUPT ADDRESS; the Z80A ignores its own $\overline{INT}$ input pin, and does not enter an INTA cycle.

(2) If interrupts are enabled (by executing an EI instruction) and COMMAND REGISTER bit INE (INTA Enable) is set to logic 1, and the timer in question is enabled, then a timeout interrupts the Z80A, and automatically gates an RST instruction onto the data bus in response to the Z80A-issued INTA cycle (an interrupt-driven configuration). Each timer is assigned its own unique RST instruction.

(3) If interrupts are enabled, and COMMAND REGISTER bit INE is reset to logic 0, and the timer in question is enabled, then a timeout interrupts the Z80A, and the interrupt service routine would then read INTERRUPT ADDRESS port IN 03H to identify the interrupt request source (a polled-interrupt configuration).

The above description applies to an enabled timer. A timer disabled by a logic 0 INTERRUPT MASK bit may be loaded with a DELAY COUNT, it will count down to timeout and generate an interrupt request, but the interrupt request will not pass thru the INTERRUPT MASK, will not set bit IPG to logic 1, and will not place an encoded address in INTERRUPT ADDRESS. A detailed discussion of timer service modes (2) and (3) above appears in Section 2.9, and programming examples appear in Section 2.10. The

remainder of this section assumes the timer service mode (1) above.

Again, if an enabled timer reaches timeout, bit IPG is set to logic 1. Loading a DELAY COUNT of 00H causes an immediate interrupt request, and loading a new DELAY COUNT before timeout re-initializes the timer without generating an interrupt request. After reading bit IPG set to logic 1, the Z80A reads the INTERRUPT ADDRESS port to identify the interrupt request source, and interprets its contents as follows:

INTERRUPT ADDRESS (IN 03H)

CONTENTS	INTERRUPT SOURCE	PRIORITY
0C7H	TIMER 1	HIGHEST
0CFH	TIMER 2	
0D7H	INT (J4 PIN 23)	
0DFH	TIMER 3	
0E7H	RDA	
0EFH	TBE	
0F7H	TIMER 4	
0FFH	TIMER 5 OR PI7 OR NO INT REQ	LOWEST

The contents of INTERRUPT ADDRESS always points to the highest priority enabled interrupt source currently requesting service. In particular, if a higher priority interrupt request occurs after a lower priority request, the contents of INTERRUPT ADDRESS will change to reflect the higher priority request, and the lower priority request will be 'pushed down' one position in the queue. The interrupt sources are prioritized as shown above, with TIMER 1 assigned the highest priority, and TIMER 5/PI7 the lowest.

The following programming example illustrates SCC timer initialization, simultaneous control of two independently maintained timers, generating timing intervals in excess of 16.32 mSec, and use of interrupt source PI7.

EXAMPLE 6

Suppose an SCC system is to monitor a parallel input device connected to port IN 0BH by reading it approximately once every 3 mSec. Further assume it is of interest to know how often the sampled data byte exceeds 0F0H in value. The program segment below uses TIMER 1 to generate the 3 mSec intervals, and maintains a tally of 'overcount' samples in Z80A Register C. For simplicity, it is assumed that overflowing Register C is not possible. The program also maintains TIMER 2 for the purpose of blinking an LED indicator ON and OFF once every two seconds to evidence ongoing program execution. The blinking indicator is assumed connected to bit D0 of port OUT 0AH (see Figure 17). Further, the program defines PI7 as an interrupt source by setting COMMAND REGISTER bit RS7 to logic 1. A low-to-high transition on PI7 (D7 of port IN 04H) causes the program to output the contents of Register C to an 8-bit LED display register connected to port OUT 0BH, to clear Register C, and then to resume port IN 0BH sampling as described above.

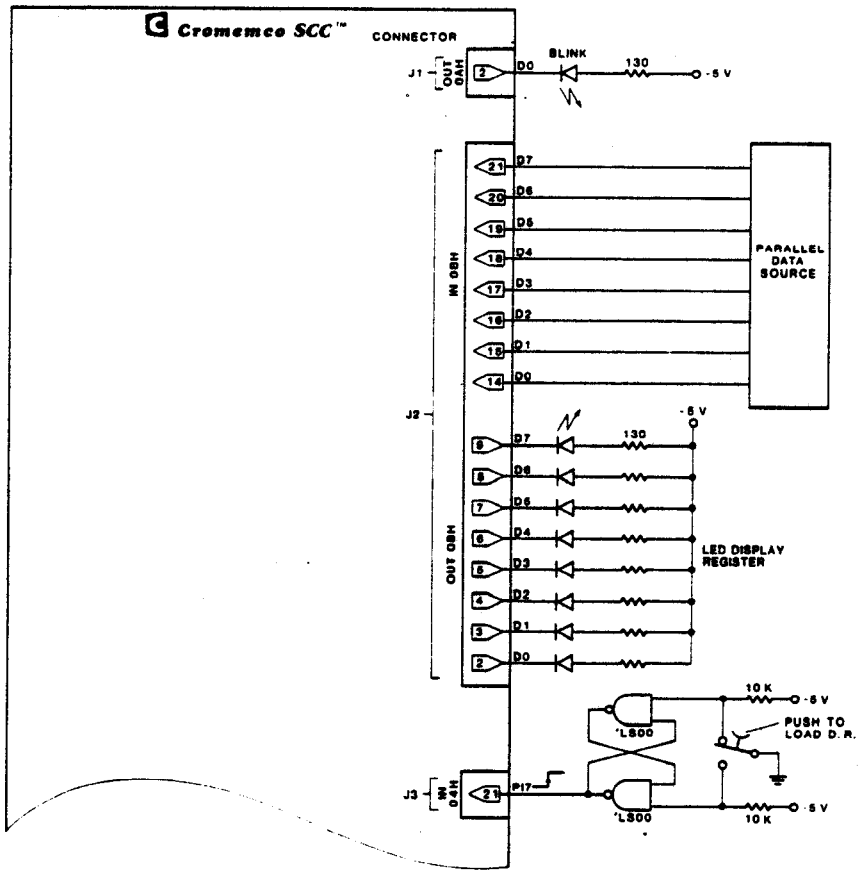


Figure 17: EXAMPLE 6 SCC CONFIGURATION

CROMEMCO CDOS Z80 ASSEMBLER version 02.15

```
                0001 ;
                0002 ;  >>>  SCC TIMER EXAMPLE ROUTINE  <<<
                0003 ;
                0004 ;
(000A)          0005 BLINK:  EQU  0AH              ;BLINK OUTPUT PORT
(0002)          0006 COMREG: EQU  02H              ;COMMAND REG. PORT
(23FF)          0007 ENDRAM: EQU  23FFH            ;END SCC RAM ADDR
(0064)          0008 HUNDRD: EQU  100D             ;PROGRAM CONSTANT
(0003)          0009 INTADR: EQU  03H              ;INTERRUPT ADDR PORT
(0003)          0010 INTMSK: EQU  03H              ;INTERRUPT MASK PORT
(0005)          0011 IPGBIT: EQU  5                ;IPG BIT D5
(000B)          0012 LITES:  EQU  0BH              ;LITES OUTPUT PORT
(00F0)          0013 MAXCNT: EQU  0F0H             ;MAX DATA COUNT
(000B)          0014 PDATA:  EQU  0BH              ;PAR'L DATA IN PORT
(0000)          0015 STATUS: EQU  00H              ;SERIAL STATUS PORT
(00C7)          0016 TM1ADR: EQU  0C7H             ;TIMER 1 INT ADDR
(002F)          0017 TM1CNT: EQU  2FH              ;TIMER 1 COUNT (3MS)
(0005)          0018 TM1PRT: EQU  05H              ;TIMER 1 PORT
```

```
        (00CF)      0019 TM2ADR: EQU  0CFH          ;TIMER 2 INT ADDR
        (009C)      0020 TM2CNT: EQU  9CH           ;TIMER 2 COUNT (10MS)
        (0006)      0021 TM2PRT: EQU  06H           ;TIMER 2 PORT
                    0022                            ;
                    0023         ORG  500H          ;ROUTINE START ROM 0
                    0024                            ;
0500    F3          0025 SCCTIM: DI                 ;DISABLE INTERRUPTS
0501    31FF23      0026         LD   SP,ENDRAM     ;MOVE STACK
0504    3E05        0027         LD   A,00000101B   ;TB5=0; HBD=0; INE=0;
0506    D302        0028         OUT  COMREG,A      ;RS7=1; BRK=0; RES=1
0508    3E83        0029         LD   A,10000011B   ;MASK ALL BUT PI7,
050A    D303        0030         OUT  INTMSK,A      ;TIMER 2 & TIMER 1
050C    0E00        0031         LD   C,0           ;ZERO OVERCOUNT
050E    1664        0032         LD   D,HUNDRD      ;1 SEC COUNT REG
0510    1E00        0033         LD   E,0           ;FIRST BLINK BYTE
0512    3E2F        0034         LD   A,TM1CNT      ;GET 3 MSEC COUNT
0514    D305        0035         OUT  TM1PRT,A      ;START TIMER 1
0516    3E9C        0036         LD   A,TM2CNT      ;GET 10 MSEC COUNT
0518    D306        0037         OUT  TM2PRT,A      ;START TIMER 2
                    0038                            ;
051A    DB00        0039 CHKIPG: IN   A,STATUS      ;READ STATUS PORT
051C    CB6F        0040         BIT  IPGBIT,A      ;IPG?
051E    28FA        0041         JR   Z,CHKIPG      ;RE-CHECK IF NO
0520    DB03        0042         IN   A,INTADR      ;YES. READ INT ADDR
0522    FEC7        0043         CP   TM1ADR        ;TIMER 1?
0524    280B        0044         JR   Z,SRVTM1      ;SERVICE IF YES
0526    FECF        0045         CP   TM2ADR        ;NO.  TIMER 2?
0528    2814        0046         JR   Z,SRVTM2      ;SERVICE IF YES
052A    79          0047         LD   A,C           ;NO.  MUST BE PI7
052B    D30B        0048         OUT  LITES,A       ;OVERCOUNT TO LITES
052D    0E00        0049         LD   C,0           ;RESET OVERCOUNT
052F    18E9        0050         JR   CHKIPG        ;BACK TO CHKIPG
                    0051                            ;
0531    DB0B        0052 SRVTM1: IN   A,PDATA       ;IN PARALLEL DATA
0533    FEF1        0053         CP   MAXCNT+1      ;DATA > MAXCNT ?
0535    3801        0054         JR   C,NOTOVR      ;NO, SKIP INC C
0537    0C          0055         INC  C             ;YES, BUMP COUNT
0538    3E2F        0056 NOTOVR: LD   A,TM1CNT      ;GET 3MS COUNT
053A    D305        0057         OUT  TM1PRT,A      ;START TIMER 1
053C    18DC        0058         JR   CHKIPG        ;BACK TO CHKIPG
                    0059                            ;
053E    15          0060 SRVTM2: DEC  D             ;100 TH. TIME?
053F    2007        0061         JR   NZ,NOTYET     ;TO NOTYET IF NO
0541    1664        0062         LD   D,HUNDRD      ;YES. 1 SEC UP
0543    7B          0063         LD   A,E           ;GET BLINK BYTE
0544    2F          0064         CPL                ;COMPLEMENT
0545    5F          0065         LD   E,A           ;STORE IN REG. E
0546    D30A        0066         OUT  BLINK,A       ;BLINK LIGHT
0548    3E9C        0067 NOTYET: LD   A,TM2CNT      ;GET 10MS COUNT
054A    D306        0068         OUT  TM2PRT,A      ;START TIMER 2
054C    18CC        0069         JR   CHKIPG        ;BACK TO CHKIPG
                    0070                            ;
054E    (0500)      0071         END  SCCTIM
```

Refer to the program text and note that the program is resident in SCC ROM 0. The DI instruction is included for emphasis only (interrupts are disabled upon a POC or RESET), and although the stack move is not needed in this program, it is included to document a typical initialization task. Also note that data output to the LED display register is in 'inverted' form, i.e., a logic 0 bit lights an LED.

The reader should observe one final note which pertains to generating "precise" timing intervals. In addition to the inherent timer inaccuracy (plus 0 uSec, minus 64 or 8 uSec), software execution overhead required to both load the timer and to respond to a timeout effectively extends the timing interval. If the overhead time can be determined, the programmed timing interval may be shortened by the same amount. In many cases, however, a precise determination may not be possible. In particular, if timer control is imbedded in complex conditonal code, or if a timer is competing with other higher priority interrupt requests, then it may only be possible to approximate the software overhead, and thus also only possible to approximate the actual time interval between programmed events.

Note that the plus 0 uSec, minus 64 or 8 uSec uncertainty is the period between initial timer loading, and the first internal 5501 clock "tick". If a timer is repeatedly reloaded after timeout but before the next "tick", then the timer will not lose "sync", and the resulting periodic intervals will exhibit the same accuracy as the crystal reference frequency.

2.9 SCC INTERRUPTS

An overview of all SCC interrupt sources and possible SCC interrupt response modes is illustrated in Figure 18 (it will be helpful to refer back to this figure often while reading this section). The ten possible SCC interrupt sources appear to the left in the figure. One interrupt source directly feeds Z80A pin $\overline{\text{NMI}}$ (Non-Maskable Interrupt), and the other nine are funneled into the Z80A's maskable INT pin. Of the ten, two are from general utility S-100 bus lines ($\overline{\text{NMI}}$ and $\overline{\text{INT}}$), and the remaining eight are managed by the 5501. Of these eight, two are again general utility ($\overline{\text{INT}}$ and PI7), and the remainder are dedicated to either serial I/O status (RDA and TBE) or interval timers (TIMER 1, TIMER 2, TIMER 3, TIMER 4 and TIMER 5).

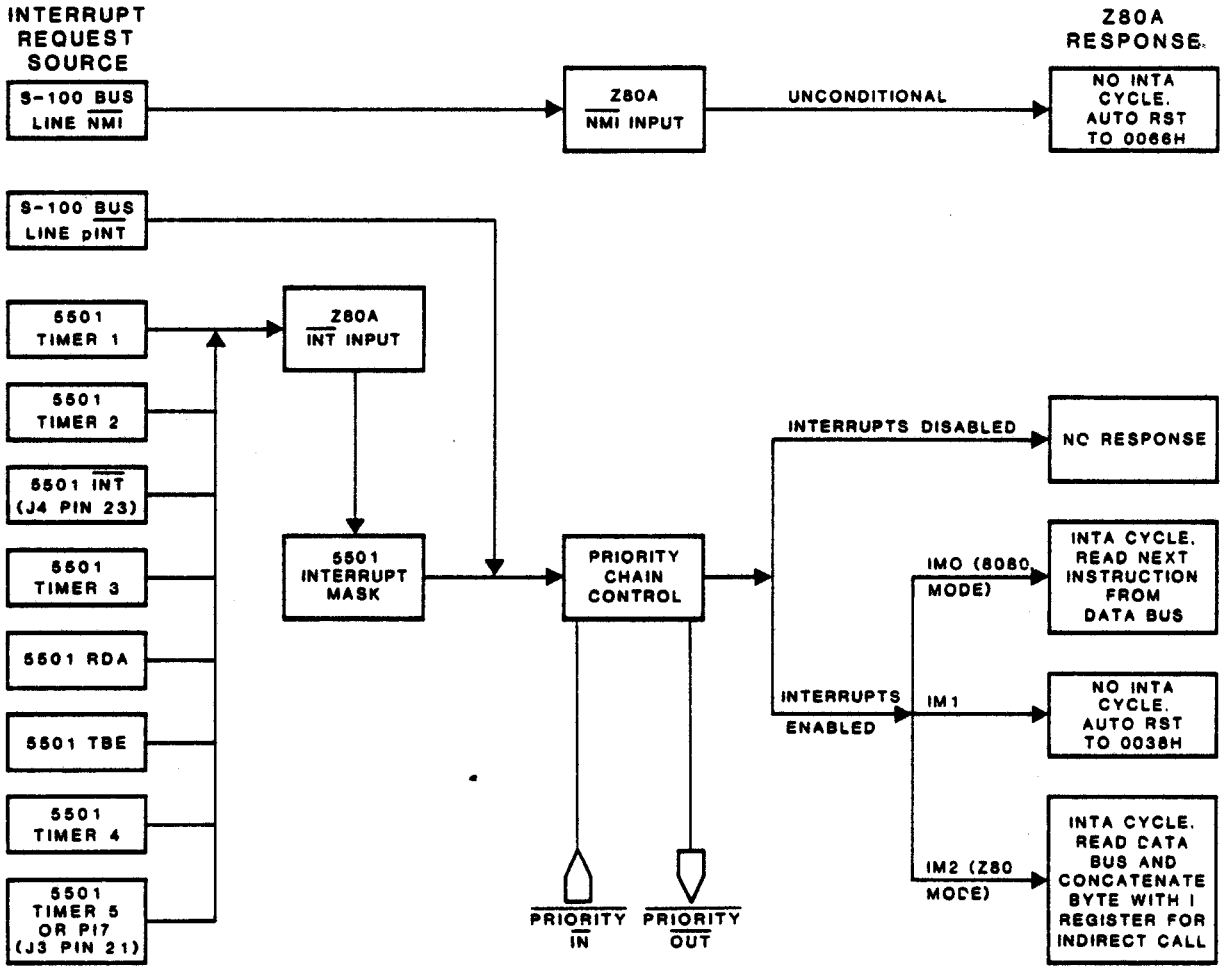


Figure 18: SCC INTERRUPT STRUCTURE

The interrupting source conditions which must be met before an interrupt request is issued are tabulated below:

Table 3

INTERRUPT REQUEST CONDITIONS

INTERRUPT SOURCE	REQUEST CONDITION
S-100 BUS NMI	TTL HIGH-TO-LOW EDGE
S-100 BUS INT	TTL LOW LEVEL
TIMER 1	DECREMENT TIMER 1 TO COUNT ZERO
TIMER 2	DECREMENT TIMER 2 TO COUNT ZERO
$\overline{\text{INT}}$ (J4 PIN 23)	TTL HIGH-TO-LOW EDGE
TIMER 3	DECREMENT TIMER 3 TO COUNT ZERO
RDA	SERIAL RECEIVER DATA AVAILABLE
TBE	SERIAL TRANSMITTER BUFFER EMPTY
TIMER 4	DECREMENT TIMER 4 TO COUNT ZERO
TIMER 5 OR PI7 (J3 PIN 21)	DECREMENT TIMER 5 TO COUNT ZERO TTL LOW-TO-HIGH EDGE

Each condition listed above does generate an 'interrupt request', but of these, only an NMI request must be unconditionally accepted (serviced) by the Z80A. In particular, an S-100 INT request is accepted only if the Z80A has enabled interrupts by executing an EI instruction. This condition also applies to the other eight sources, but in addition, they must also be enabled by logic 1 bits output to INTERRUPT MASK port OUT 03H. And even then, servicing of these eight 5501 managed interrupt sources may be postponed until interrupts from another higher priority board are serviced, as determined by the SCC 'daisy chain' interrupt priority logic. These topics will be developed later in this section.

Of the ten SCC interrupt sources, only an S-100 bus NMI may not be disabled by a software controlled mask. This source is typically dedicated to a crucial system condition, such as

detecting an impending power failure. A description of SCC response to an NMI follows:

NMI INTERRUPT RESPONSE: A Non-Maskable Interrupt Request is made to the Z80A by a falling edge on S-100 bus line $\overline{\text{NMI}}$, which sets an NMI latch internal to the Z80A (see Figure 19). The Z80A samples its NMI latch output with the rising edge of the last clock cycle at the end of every instruction executed. If an NMI is detected, the Z80A unconditionally accepts the NMI by finishing the current instruction, resetting its NMI latch and disabling maskable interrupts, pushing the Program Counter (PC) contents onto the stack, and jumping to memory address 0066H, the fixed starting address of the NMI service routine. Notice that this is an SCC ROM 0 location, if SCC on-board memory is enabled. The Z80A does not issue Interrupt Acknowledge status (sINTA) to the S-100 bus in response to an NMI. A return from the NMI service routine to the interrupted program may be performed by executing a RETN instruction, which pops the PC contents from the stack and restores the maskable interrupts enabled/disabled status which existed before the NMI, or by executing a RET instruction which pops the PC from the stack but does not restore the previous interrupt status.

It should be noted that an S-100 bus NMI is serviced even when the Z80A is in the HALTed state, that only S-100 bus line $\overline{\text{HOLD}}$ overrides NMI, and that forcing S-100 RDY low prevents the current instruction from ending, and thus holds off NMI servicing.

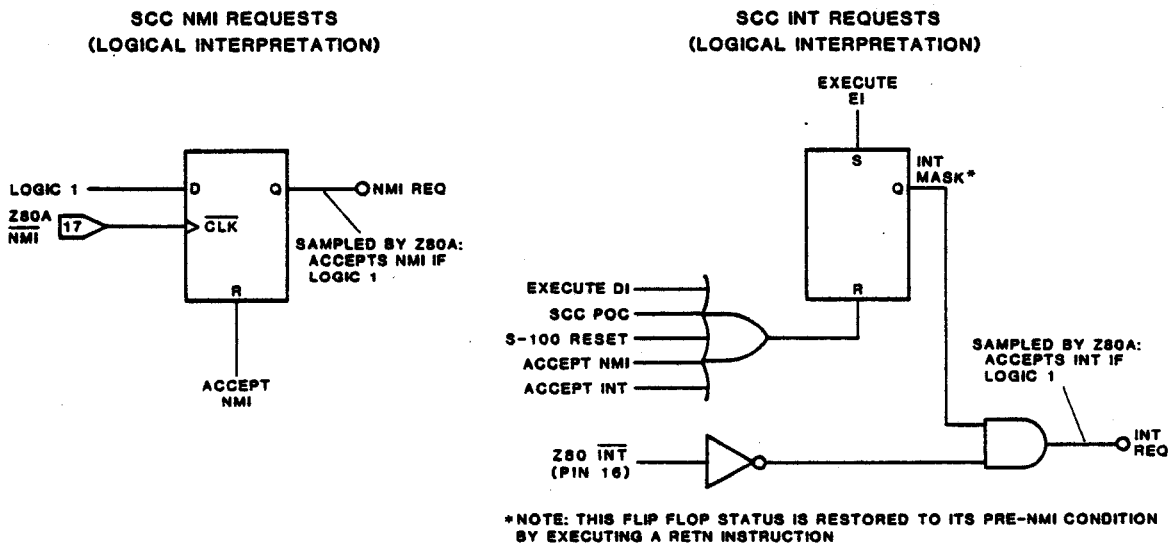


Figure 19: Z80A INTERRUPT REQUEST SAMPLING

The other nine interrupt sources request servicing thru the Z80A's maskable $\overline{INT}$ pin. The 5501 manages eight of these interrupt sources, and its own INT output pin goes high to request service for any one of them. This signal line is logically combined with S-100 bus line $\overline{INT}$ so that an interrupt request from either one (5501 pin INT, or S-100 bus line $\overline{INT}$) forces Z80A pin $\overline{INT}$ active low (see SCC Schematic Diagram). The Z80A may, or may not, accept the interrupt request depending upon whether interrupts are enabled or disabled. Figure 19 above illustrates the logical relationship between Z80A pin $\overline{INT}$ and the various masking conditions. The Z80A samples internal line INT REQ with the rising edge of the last clock cycle at the end of every instruction executed (at the same time NMI REQ is sampled; an NMI REQ has higher priority than INT REQ, however). INT REQ is a logical AND of Z80A input $\overline{INT}$ and a one bit INT MASK. If INT MASK

is logic 0 (interrupts disabled), the Z80A always samples a logic 0 INT REQ, regardless of Z80A pin $\overline{INT}$ status, and thus merely proceeds to execute the next program instruction. The one bit INT MASK is latched in the logic 0 state by any one of the following five SCC events: (1) the Z80A executes a DI instruction, or (2) the Z80A accepts an NMI request, or (3) the Z80A accepts a maskable INT request, or (4) an SCC Power-On Clear occurs, or (5) S-100 bus line $\overline{RESET}$ goes active low. Subsequently, INT MASK may be latched in the logic 1 state (interrupts enabled) only by executing an EI instruction. Notice the implication that interrupts are disabled upon entering an interrupt service routine, and that the service routine itself would have to execute an EI instruction to re-enable interrupts.

Enabling interrupts causes the Z80A to sample a logic 1 INT REQ when Z80A pin $\overline{INT}$ is active low. In this case the Z80A accepts INT, and responds in one of three 'interrupt response modes' (IM 0, IM 1 or IM 2), as previously defined by the programmer. A description of each mode follows:

INTERRUPT RESPONSE MODE 0: This mode is selected by default after a RESET or POC. It is also termed '8080 Mode' since it exactly mimics 8080A interrupt response, and is programmatically selected by executing an IM 0 instruction. In this mode, the Z80A finishes the current instruction, disables further interrupts (INT MASK is reset to logic 0), sends out S-100 bus sINTA status (Interrupt Acknowledge), then fetches next instruction opcode from the data bus. Meanwhile, the interrupting device monitors the S-100 bus for a sINTA, and typically uses its logic 1 level to gate one-of-eight RST instructions onto the bus as an interrupt service routine vector, and also to remove its interrupt request. RST 00H thru RST 38H are one byte CALL instructions to eight pre-defined memory addresses. RST 00H is a one byte CALL to 0000H, RST 08H to 0008H, and so on. Executing an RST instruction pushes the current PC contents onto the stack, then jumps to one of the eight

addresses defined above. After servicing is complete, a RET (subroutine return) instruction is typically executed which reloads the PC with its original contents, and program execution resumes from the point of interruption. Note that the 5501 may be programmed to automatically supply prioritized RST instructions in response to sINTA by setting COMMAND REGISTER bit INE to logic 1, thus IM 0 would typically (although not necessarily) be selected as the SCC interrupt mode.

INTERRUPT RESPONSE MODE 1: This mode is selected by executing an IM 1 instruction. In this mode, the Z80A responds by finishing the current instruction, disabling interrupts, and then by automatically executing an RST 38H instruction (a one byte CALL to address 0038H). The Z80A does not send out sINTA status, thus one typical service routine task would be removing the interrupt request by a hardware link to the interrupt source. Again, the last service routine instruction is typically a RET, which resumes foreground program execution from the point of interruption.

INTERRUPT RESPONSE MODE 2: This mode is selected by executing an IM 2 instruction. To use this mode, a table of interrupt service routine starting addresses (two bytes for each routine) must be maintained somewhere in memory. The Z80A responds to an interrupt request by finishing the current instruction, disabling interrupts, sending out sINTA status, then reading a 'pointer byte' from the data bus. The current contents of Z80A Register I are concatenated with this 'pointer byte' to form a table address which the Z80A then uses for an indirect CALL to the interrupt service routine. As with IM 0, the interrupt source monitors S-100 bus line sINTA, and uses its logic 1 level to gate the 'pointer byte' onto the data bus, and to remove the interrupt request. Again, the last service routine instruction is typically a RET. Note that Register I is cleared after a RESET or a POC; that it may be loaded by an LD I,A instruction; and that it is used as the upper eight bits of the table address. The 'pointer byte' is used to form the lower eight bits of the table address, and its LSB (D0) is assumed logic 0 regardless of the value supplied. The resulting table addresses must all then begin at even memory locations, and are stored in low-high Intel style in memory.

It should be noted that, like an NMI request, an INT request is serviced even when the Z80A is in the HALTed state, that S-100 bus line $\overline{\text{HOLD}}$ overrides INT, and that forcing S-100 line RDY low prevents the current instruction from ending, and thus holds off INT servicing.

Of these three possible interrupt response modes, IM 0 would typically be selected in an interrupt-driven SCC configuration (with COMMAND REGISTER bit INE set to logic 1), and IM 1 in a polled-interrupt SCC configuration (with COMMAND REGISTER bit INE reset to logic 0). NMI, IM 0 and IM 1 example service routines appear later in this section to fix the ideas outlined above. The reader is referred to the Z80A Technical Manual, included in the SCC documentation, for an expanded general discussion of Z80A interrupt response, and also to the S-100 bus NMI and INT waveforms shown in Section 4.3.

Before discussing the 5501 interrupt control features, it will be helpful to provide a context by posing some questions which must be answered before configuring the 5501. These questions pertain primarily to servicing the nine non-NMI interrupt request sources.

(1) Are program interrupts really necessary? If one or more system events require immediate attention, and the system executive program must devote its attention to tasks other than identifying and servicing these events, the answer is probably yes. If not, the executive program may detect a 'pseudo-interrupt' by testing SERIAL STATUS I/O port bit IPG, and then read the SCC INTERRUPT ADDRESS port to find the interrupt source.

(2) Should program interrupts be polled, or interrupt-driven? An interrupt-driven response is faster, but it lacks the flexibility of a polled response. In addition, a polled response might be

chosen to free low memory address area 0000H - 0037H, and the associated RST instructions, for non-interrupt related functions. In most cases, however, an interrupt-driven mode would be chosen for its speed and low programming complexity.

(3) Which interrupt sources should be enabled, and which disabled (by data sent to INTERRUPT MASK port OUT 03H)? This decision will be largely determined by the 5501's interrupt priority structure versus actual system priorities. There are three general categories of SCC interrupt sources: general utility, serial I/O status, and interval timers. By disabling the sources one by one, from the highest priority down, any one of these three categories may be assigned the highest priority, and yet allow for at least one enabled interrupt source in each of the other two categories. Serial status sources RDA and/or TBE might also be disabled, even though the SCC serial I/O port is used, since RDA and TBE conditions may always be determined by reading SERIAL STATUS port IN 00H.

With these considerations in mind, the discussion will now turn to the 5501 and its interrupt sources. The interrupt related workings of the 5501 are idealized in Figure 20. Note that this is not a partial schematic of the 5501, but rather a combination block diagram/logic diagram interpretation of its internal interrupt structure.

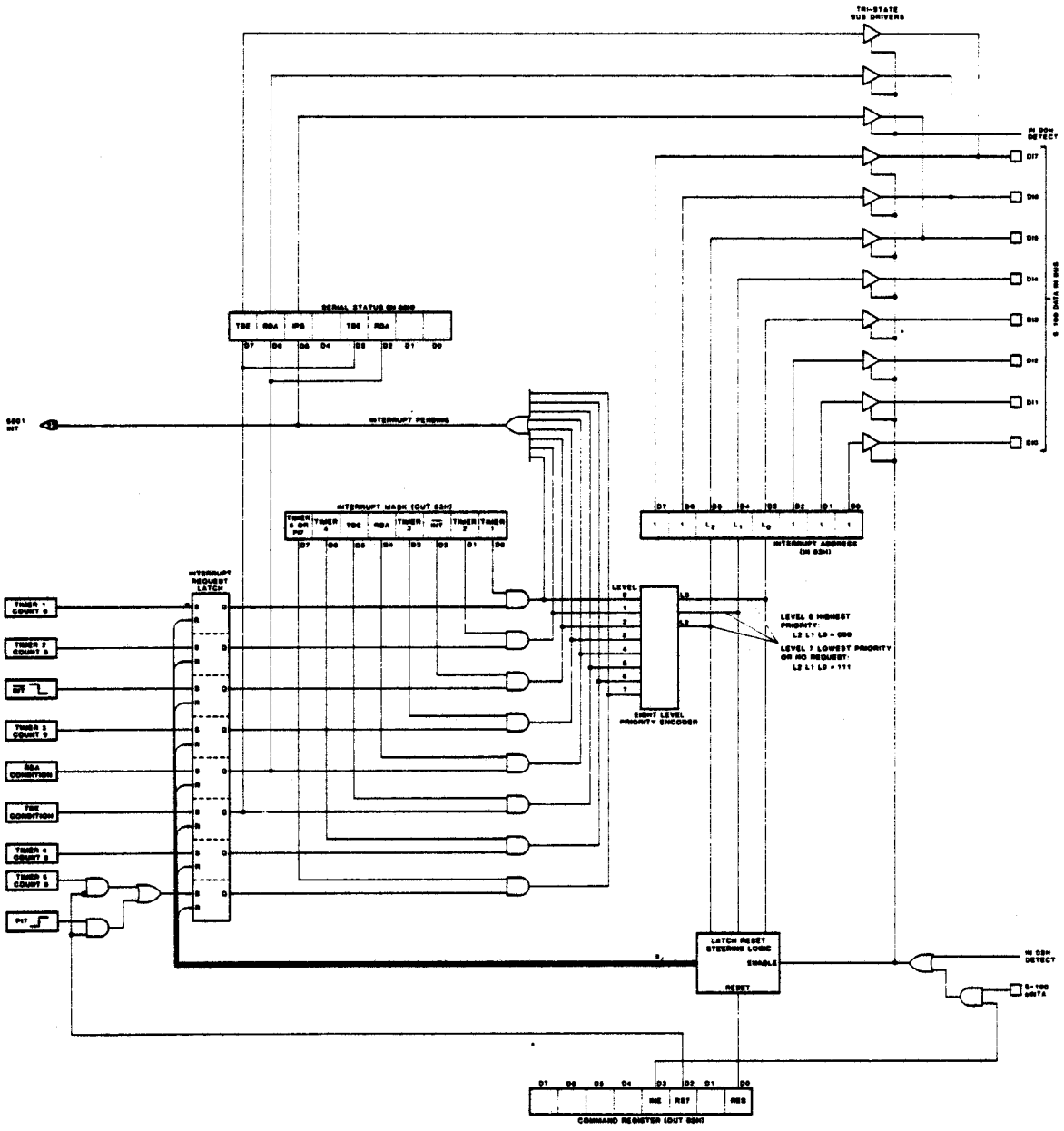


Figure 20: 5501 INTERRUPT STRUCTURE

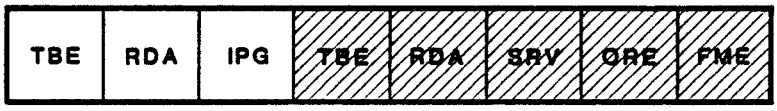
Refer to the figure, and first notice that interrupt requests from the eight sources to the left selectively set bits in an eight bit INTERRUPT REQUEST LATCH. Bit RS7 from COMMAND REGISTER port OUT 02H selects between TIMER 5 and PI7 as the eighth interrupt source. Bits of the INTERRUPT REQUEST LATCH are reset either in parallel by bit RES from the COMMAND REGISTER (a 5501 RESET), or selectively every time the INTERRUPT ADDRESS register is read, i.e., reading the TBE address from the INTERRUPT ADDRESS register resets the TBE interrupt request latch, etc. Two outputs of the INTERRUPT REQUEST LATCH may be sampled directly as RDA and TBE from SERIAL STATUS port IN 00H, and all outputs are logically ANDed with bits from INTERRUPT MASK port OUT 03H before feeding an eight level PRIORITY ENCODER. A logic 1 AND gate output implies both that its corresponding interrupt source is requesting service, and that the source is enabled (its mask bit is set to logic 1). All AND gate outputs are logically ORed to form signal IPG (Interrupt Pending). IPG goes to logic 1 when any enabled interrupt source requests service. IPG is available both as a SERIAL STATUS bit, and also as 5501 output pin INT. The PRIORITY ENCODER has eight inputs, each of which is assigned a different priority level. TIMER 1 is assigned LEVEL 0 (the highest priority), and TIMER 5/PI7 is assigned LEVEL 7, the lowest priority. Output lines L2(MSB), L1 and L0(LSB) form a three bit octal number. This number always represents the highest priority LEVEL which is currently active (logic 1) at the PRIORITY ENCODER inputs. If no inputs are active, the PRIORITY ENCODER will output LEVEL 7 (all logic 1's). The PRIORITY ENCODER outputs are used

both to form the contents of INTERRUPT ADDRESS port IN 03H, and also to steer a reset back to the correct interrupt request latch after the INTERRUPT ADDRESS register is read. The LEVEL supplied by the PRIORITY ENCODER is combined with logic 1 bits in the INTERRUPT ADDRESS register so as to form a valid Z80A restart (RST) instruction opcode; priority LEVEL 0 (TIMER 1) forms RST 00H opcode 0C7H, ... , priority LEVEL 7 (TIMER 5/PI7) forms RST 38H opcode 0FFH. The INTERRUPT ADDRESS register contents may be used in two ways: it may be read as input port IN 03H data in a non-interrupt or polled interrupt (IM 1) environment, or it may be gated onto the data bus as an RST instruction in response to an S-100 bus sINTA in an interrupt-driven (IM 0) environment. In both cases, the INTERRUPT ADDRESS contents are gated onto the S-100 Data In (DI) bus by enabling eight tri-state bus drivers, but the driver enabling conditions vary from one case to the other. In the first instance, an IN 03H DETECT condition forces the OR gate output shown in the figure to enable the drivers (an IN 03H DETECT condition exists when S-100 bus address lines form the repetitious port address 0303H AND status sINP is logic 1 AND processor control pDBIN is logic 1). In the second instance, COMMAND REGISTER bit INE (INTA Enable), used as a mask, is logically ANDed with S-100 bus status sINTA (recall that the Z80A sends out sINTA in modes IM 0 and IM 2 prompting the interrupt source to place a service routine vector on the data bus). If INE is set to logic 1, status sINTA enables the tri-state drivers causing the INTERRUPT ADDRESS contents (which form an RST opcode) to be placed on the DI bus. If, on the other hand, INE is reset to logic 0,

status sINTA causes no reponse whatsoever from the 5501.

The Z80A configures the SCC interrupt system thru COMMAND REGISTER port OUT 01H and INTERRUPT MASK port OUT 03H, and may sample interrupt system status in I/O port-driven or polled-interrupt configurations thru SERIAL STATUS port IN 00H and INTERRUPT ADDRESS port IN 03H. A description of each of these ports follows, as they pertain to SCC interrupt system operation. Refer to Appendix A for a comprehensive description of all SCC I/O ports.

SERIAL STATUS (IN 00H):



SERIAL STATUS bits RDA and TBE are mentioned here in connection with servicing interrupts only to emphasize that these bits always represent valid RDA and TBE status, even if the RDA and TBE interrupt requests are masked-off thru INTERRUPT MASK port OUT 03H.

SERIAL STATUS bit IPG (Interrupt Pending) is set to logic 1 to indicate that at least one of the unmasked 5501 interrupt sources is currently requesting service. IPG is typically used in a polled interrupt system to resolve between requests originating from the 5501, and from S-100 bus line $\overline{\text{INT}}$. Bit IPG is reset to logic 0 when all un-masked interrupt requests have been cleared (either by a 5501 reset, or by servicing).

COMMAND REGISTER (OUT 02H):

COMMAND REGISTER bit INE (INTA Enable) defines the 5501 reponse to an INTA (Interrupt Acknowledge) cycle. In an IM 0 interrupt-driven SCC system, INE would be set to logic 1. This programs the 5501 to automatically place the INTERRUPT ADDRESS register contents on the data bus in response to S-100 bus sINTA status. The INTERRUPT ADDRESS register always contains one of eight prioritized RST instructions recognized by the Z80A (see below). In an IM 1 polled interrupt SCC system, INE would be reset to logic 0. In this mode, S-100 bus sINTA status evokes no response from the 5501.

COMMAND REGISTER bit RS7 (RST 7 Select) chooses between TIMER 5 and PI7 (D7 of port IN 04H) as the lowest priority 5501 interrupt source. Resetting RS7 to logic 0 selects PI7; setting RS7 to logic 1 selects TIMER 5.

Setting COMMAND REGISTER bit RES (5501 Reset) to logic 1 initializes the 5501 as follows: the serial receiver goes into the search mode, the serial transmitter output XMT goes high (marking); all interval timers are cleared; TBE status generates an interrupt request, and all other interrupt requests are cleared. Resetting bit RES to logic 0 evokes no response from the 5501. Bit RES is automatically reset to logic 0 after 5501 reset tasks are completed.

INTERRUPT ADDRESS (IN 03H):

1	1	L_2	L_1	L_0	1	1	1

The INTERRUPT ADDRESS register plays a dual role; it may be read as SCC input port IN 03H in interrupts disabled or polled-interrupt configurations, or it may be programmed to automatically place its contents onto the data bus in response to S-100 bus sINTA status (see COMMAND REGISTER bit INE, above) in interrupt-driven configurations. The INTERRUPT ADDRESS register always contains one-of-eight RST instructions recognized by the Z80A. Each 5501 interrupt source is assigned an RST instruction and a priority (below). The INTERRUPT ADDRESS contents then points to the highest priority unmasked 5501 interrupt source currently requesting service. After the INTERRUPT ADDRESS register is either read as an input port, or after its contents are gated onto the data bus in response to an INTA cycle, the corresponding interrupt request is cleared. The INTERRUPT ADDRESS contents are then replaced with an RST opcode pointing to the next highest priority pending interrupt request, etc. Data 0FFH will be read from the INTERRUPT ADDRESS register if no unmasked 5501 interrupt request is pending.

INTERRUPT ADDRESS	INTERRUPT SOURCE	PRIORITY LEVEL
0C7H (RST 00H)	TIMER 1	0 (HIGHEST)
0CFH (RST 08H)	TIMER 2	1
0D7H (RST 10H)	INT (J3 PIN 23)	2
0DFH (RST 18H)	TIMER 3	3
0E7H (RST 20H)	RDA	4
0EFH (RST 28H)	TBE	5
0F7H (RST 30H)	TIMER 4	6
0FFH (RST 38H)	TIMER 5 OR PI7	7 (LOWEST)
0FFH (RST 38H)	NO REQUEST	7 (LOWEST)

INTERRUPT MASK (OUT 03H):

TIMER 5 OR PI7	TIMER 4	TBE	RDA	TIMER 3	$\overline{\text{INT}}$	TIMER 2	TIMER 1

Each bit position in this output port selectively masks (disables) or unmasks (enables) interrupt requests emanating from the eight 5501 interrupt sources. A logic 0 bit masks an interrupt request, a logic 1 bit unmasks it. Note that interrupt requests are latched even when the corresponding INTERRUPT MASK bits are reset to logic 0. In such cases, an interrupt request will be retained until its corresponding mask bit is programmatically changed to logic 1 (it will then become enabled, and will be prioritized with other enabled requests), or until COMMAND REGISTER bit RES is set to logic 1 which clears all interrupt requests, except TBE. An un-masked interrupt request is prioritized with all other enabled requests, and if it is the highest priority, its RST opcode is placed in the INTERRUPT ADDRESS register, SERIAL STATUS bit IPG is set to logic 1, and a service request to the Z80A is made by forcing its $\overline{\text{INT}}$ pin active low. If the Z80A has enabled interrupts by executing an EI

instruction, it services the request in either IM 0, IM 1 or IM 2, as previously defined by the programmer.

BIT	MASK REQUEST	UNMASK REQUEST	INTERRUPT SOURCE
D7	0	1	TIMER 5 OR PI7
D6	0	1	TIMER 4
D5	0	1	TBE
D4	0	1	RDA
D3	0	1	TIMER 3
D2	0	1	INT (J3 PIN 23)
D1	0	1	TIMER 2
D0	0	1	TIMER 1

Other Cromemco products may generate interrupt service requests on S-100 bus line $\overline{\text{INT}}$, including the PRI printer interface card, the TU-ART serial interface card, the 4FDC floppy disk interface, the 16FDC and the WDI Winchester Disk Interface card. When the SCC is co-resident with any of these cards, it is necessary to coordinate their interrupt responses to prevent bus conflicts during INTA cycles. This is accomplished by prioritizing the cards themselves using the INTERRUPT PRIORITY IN/OUT connector provided on each card, including the SCC (see Figure 21).

Priorities are assigned by connecting the $\overline{\text{PRIORITY OUT}}$ line from the highest priority card to the $\overline{\text{PRIORITY IN}}$ line of the second highest priority card, then connecting the $\overline{\text{PRIORITY OUT}}$ line from this second card to the $\overline{\text{PRIORITY IN}}$ line of the third highest priority card, and so on, until all the cards are connected. The $\overline{\text{PRIORITY IN}}$ line of the highest priority card is left unconnected. This interrupt priority daisy chain structure

inhibits any lower priority card from responding to an INTA cycle when a higher priority board is requesting service, but allowing it to respond (by gating an RST opcode onto the data bus) when no higher priority board is requesting service. Figure 21 below illustrates how an SCC and three TU-ART cards might be daisy-chained in an interrupt-driven serial I/O management subsystem.

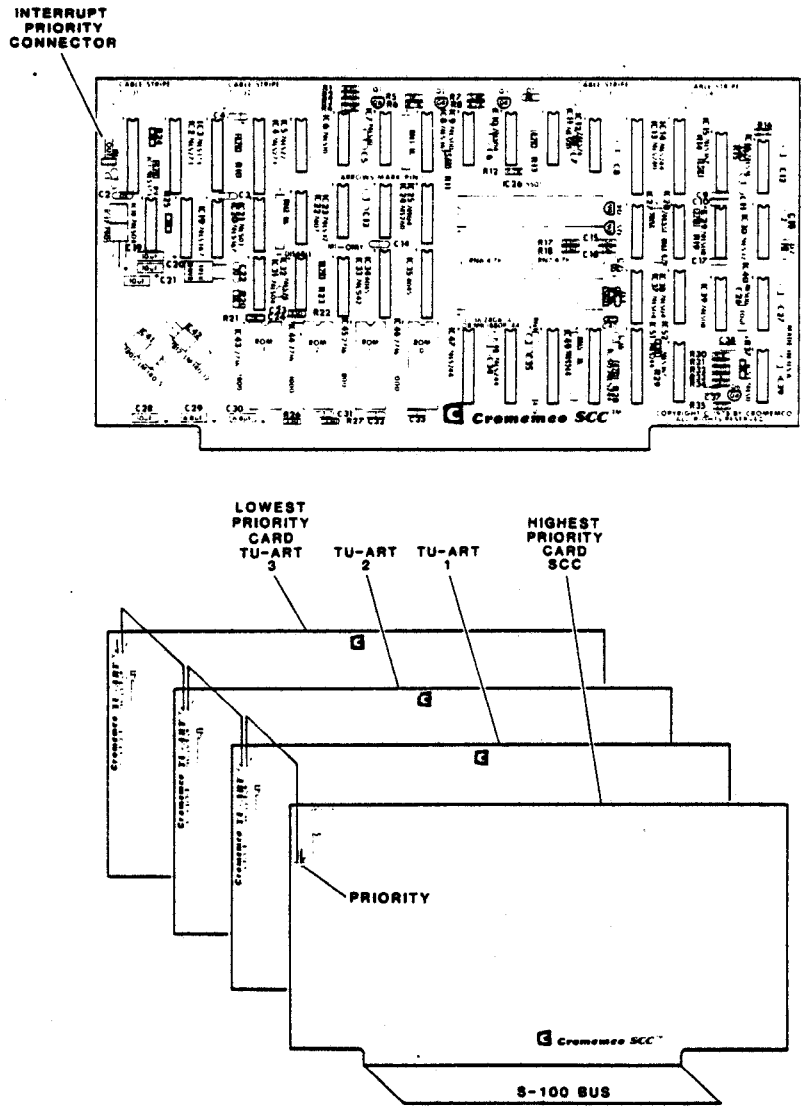


Figure 21: SCC INTERRUPT PRIORITY DAISY CHAIN

2.10 INTERRUPT PROGRAMMING EXAMPLES

Three interrupt programming examples will now be presented and discussed. The first illustrates an NMI power-failure service example, the second an IM 0 interrupt driven system, and the third assumes the same system, but with polled interrupts.

Several points must be born in mind while designing a power-failure service capability into an SCC system:

(1) Both the Z80A and the 5501 contain registers which will not retain data when power fails.

(2) Most Z80A register contents may be read, placed in non-volatile memory, and subsequently restored.

(3) The 5501 registers may not be read and subsequently restored in this fashion. In particular, timers in mid-count may not be read, and its parallel output port OUT 04H may not be sampled. Likewise, any unfinished serial I/O operations may not be read for later restoration.

(4) The other two SCC PARALLEL DATA ports, OUT 0AH and OUT 0BH, are also volatile and write-only.

In all but the simplest SCC systems, a complete 'recovery' from a power failure would then be impossible, if 'recovery' is taken to mean restoring the SCC system to the identical 'state' which existed before the failure, and then continuing program execution from the point of interruption. With this disclaimer in

mind, the next example assumes such a simple system as an NMI presentation vehicle.

EXAMPLE 7

Consider a particular SCC system whose 'state' is assumed to be completely characterized by the original initialized state of the 5501, the program data and stack in RAM, the contents of Z80A registers AF, BC, DE, HL, IX and IY, and the Program Counter, PC. Further assume that the SCC system includes 4 Kbytes of battery backed-up RAM memory spanning the address range B000H -BFFFH, and that the stack and all crucial program data is maintained in this memory area. The 120 VAC line is assumed monitored by a device which generates one TTL-level falling edge to S-100 bus line $\overline{\text{NMI}}$ upon detecting a line voltage droop. The device must be manually reset to generate another edge (to avoid multiple NMI requests). See Figure 22. The NMI service routine below then stores the volatile Z80A register contents in fail-safe RAM, and HALTs. The memory map both before and after NMI servicing is shown in Figure 23. Notice that the NMI service routine is pre-defined to start at address 0066H, and that program execution time is 98 clock cycles (24.5 uSec with a 4 MHz clock and no wait states).

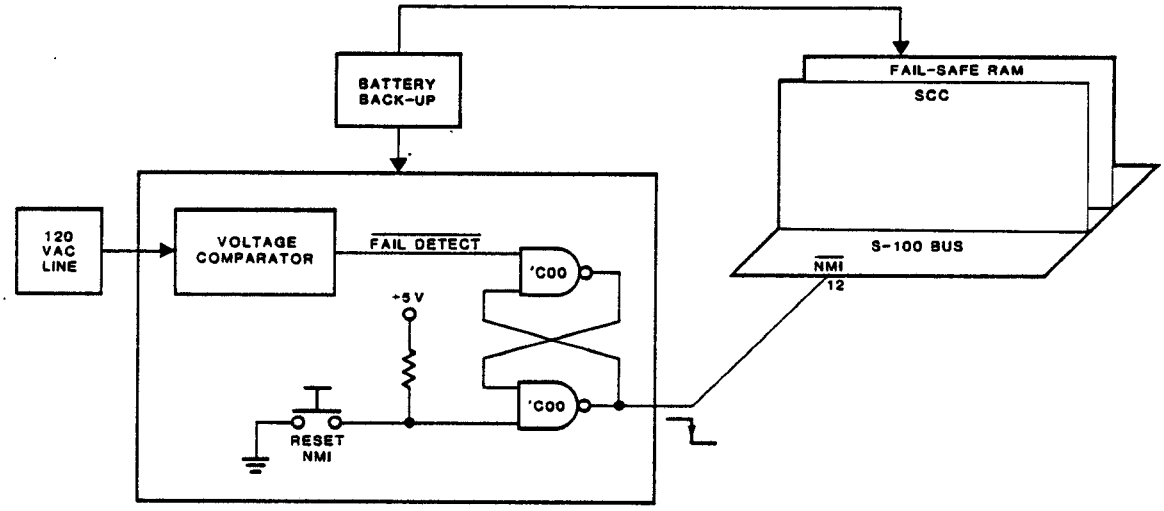


Figure 22: NMI HARDWARE

CROMEMCO CDOS Z80 ASSEMBLER version 02.15

```
                    0001 ;
                    0002 ;  >>> POWER FAILURE NMI ROUTINE <<<
                    0003 ;
                    0004 ;
      (0BFF)        0005 ENDRAM: EQU  0BFFH              ;END FAIL-SAFE RAM
                    0006                                 ;
                    0007         ORG  0066H              ;MNI SERVICE ADDR
                    0008                                 ;
0066  F5            0009 NMI:    PUSH AF                 ;REGS. AF TO STK
0067  C5            0010         PUSH BC                 ;REGS. BC TO STK
0068  D5            0011         PUSH DE                 ;REGS. DE TO STK
0069  E5            0012         PUSH HL                 ;REGS. HL TO STK
006A  DDE5          0013         PUSH IX                 ;REG. IX TO STK
006C  FDE5          0014         PUSH IY                 ;REG. IY TO STK
006E  ED73FE0B      0015         LD   (ENDRAM-1),SP      ;STORE SP
0072  76            0016         HALT                    ;HALT.  RECOVER BY
0073  (0066)        0017         END  NMI                ;RUNNING 'NMIRCV'
```

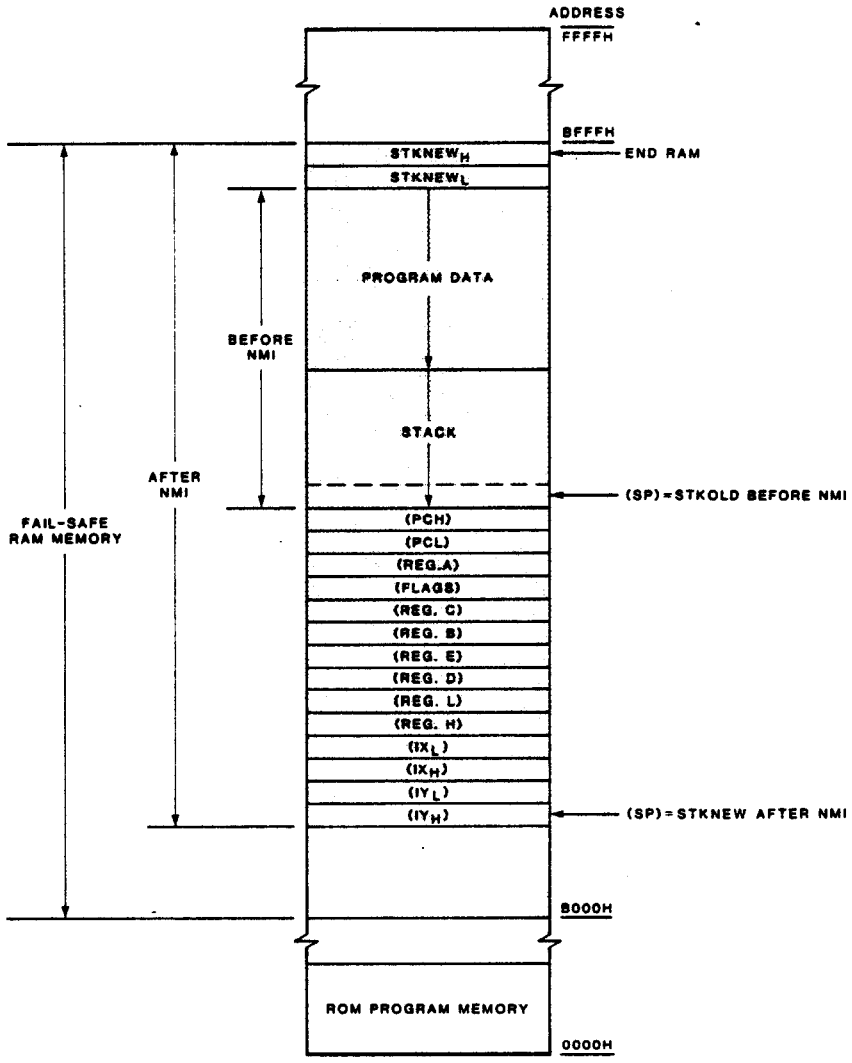


Figure 23: NMI EXAMPLE MEMORY MAP

To recover, re-applying power to the SCC would generate a POC and effect a jump to address 0000H. Typically, a system initialization routine would start there which would configure the 5501 to its pre-NMI condition. Anticipating a possible NMI, this routine would additionally not change any fail-safe RAM data, and would provide a 'jump to NMI recovery' option. Choosing this option would then cause a jump to the routine shown below (arbitrarily assumed starting at 1000H). This recovery routine then restores all vital Z80A registers, and the RET instruction pops the pre-NMI Program Counter contents from the stack, placed there by the automatic RST 0066H instruction response to an NMI, thus continuing program execution from the point of interruption.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                    0001 ;
                    0002 ; >>> NMI RECOVERY ROUTINE <<<
                    0003 ;
                    0004 ;
      (0BFF)        0005 ENDRAM: EQU  0BFFH              ;END FAIL-SAFE RAM
                    0006                                 ;
                    0007         ORG  1000H              ;ARBITRARY START ADDR
                    0008                                 ;
1000  ED7BFF0B      0009 NMIRCV: LD   SP,(ENDRAM)        ;(SP)=STKNEW
1004  FDE1          0010         POP  IY                 ;RESTORE REG. IY
1006  DDE1          0011         POP  IX                 ;RESTORE REG. IX
1008  E1            0012         POP  HL                 ;RESTORE REGS. HL
1009  D1            0013         POP  DE                 ;RESTORE REGS. DE
100A  C1            0014         POP  BC                 ;RESTORE REGS. BC
100B  F1            0015         POP  AF                 ;RESTORE REG. A & FLAGS
100C  C9            0016         RET                     ;CONTINUE PROGRAM FROM
100D  (1000)        0017         END  NMIRCV             ;POINT OF NMI
```

The next example shows complete system software for an IM 0 interrupt-driven system with three interrupt sources.

EXAMPLE 8

The SCC system illustrated in Figure 24 is contrived to illustrate an IM 0 interrupt-driven system with a background program and three foreground interrupt sources; two from the 5501 sources and one from a device driving S-100 bus line $\overline{INT}$. The system hardware consists of an SCC, a powered S-100 bus, a 1.000 Hz clock and associated interrupt request latch, and a 9600 baud CRT terminal (only signal lines are shown).

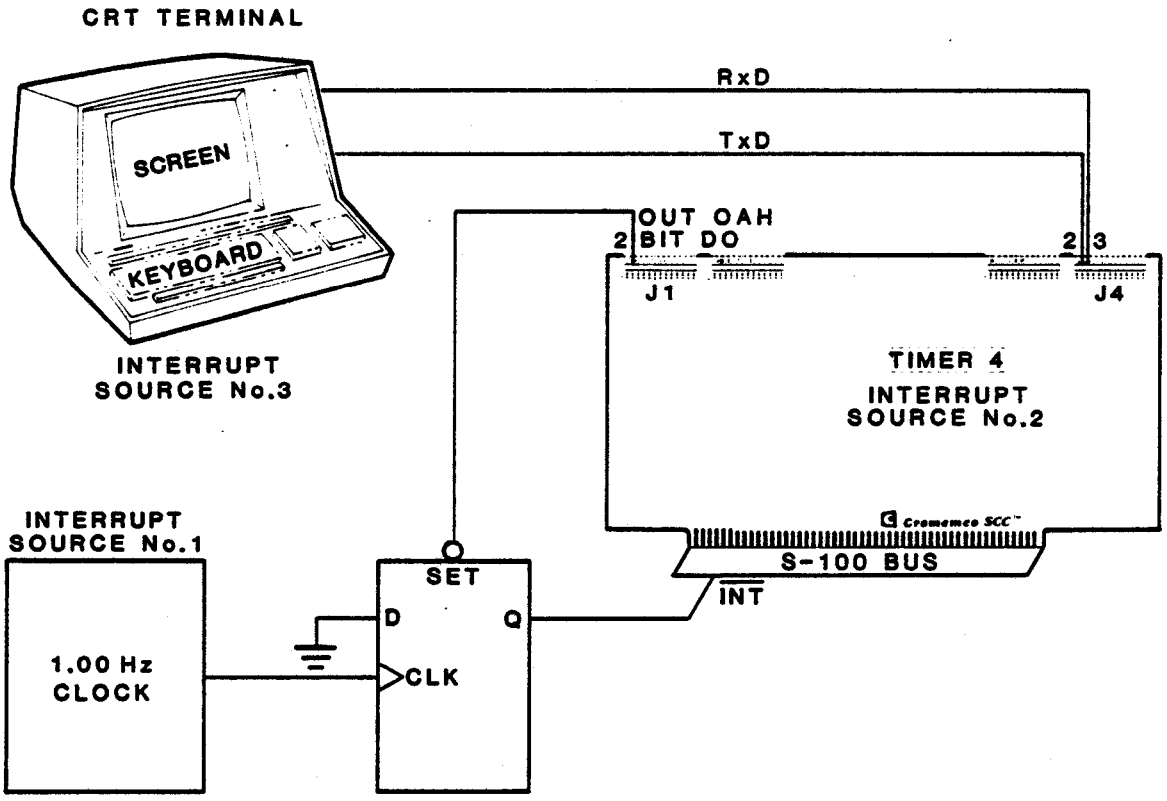


Figure 24: IM 0 SYSTEM HARDWARE

The background program, which starts at ROM 0 address 100H, has two main functions: it outputs a period character to the CRT terminal as soon as SERIAL STATUS bit TBE is sampled true (thus at near 9600 baud), and it samples a real-time seconds count at RAM address 'SECNDS', to intersperse one ASCII BEL (or 'beep') character among the periods every minute.

Interrupt source #1 is a real time seconds clock interfaced thru a request latch to S-100 bus line $\overline{\text{INT}}$. An interrupt request is made on the rising edge of the clock output once per second. Since line $\overline{\text{INT}}$ is not one of the 5501 sources, INTERRUPT ADDRESS 0FFH (RST 38H) will be gated onto the data bus in response to an INTA cycle. The RST 38H interrupt service routine first resets the interrupt request latch thru bit D0 of port OUT 0AH, then increments the real-time seconds count at RAM address 'SECNDS'. Note that the RST 38H service routine, with a pre-defined starting address of 0038H, effects a jump to the actual service routine located elsewhere in memory. While this is not actually necessary in this case, it is typically done when several 'consecutive' RST instructions are used as their corresponding call addresses are spaced only eight bytes apart, leaving little room for service code.

Interrupt source #2 is 5501 TIMER 4. This interrupt source is used to generate 10 mSec intervals. The RST 30H TIMER 4

service routine reloads TIMER 4 to start a new 10 mSec interval, and additionally sends the ASCII character located at RAM address 'CHAR' to the CRT terminal. The contents of 'CHAR' is initialized to an ASCII question mark, and this output character is interspersed with both the periods and the 'beeps' output to the CRT terminal. Since the CRT baud rate is assumed 9600, then the ratio of 'periods' to 'CHARs' to 'beeps' is approximately 576,000 to 6,000 to 1.

Interrupt source #3 is the CRT keyboard which issues an RDA interrupt request. The RST 4 service routine reads the RECEIVER DATA port to input the keyboard character, then uses it to replace the current contents of RAM address 'CHAR'. Thus, a question mark (the initial contents of 'CHAR') first appears every 10 mSec interspersed with the periods, and subsequently, any other character may be made to appear every 10 mSec by a new CRT keystroke.

Program execution begins at ROM 0 address 0000H after either a POC or an S-100 bus RESET. The system functions common to all other SCC system configurations are initialized there, and the first portion of program BAKGND initializes system functions unique to this example only.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                        0001 ;
                        0002 ;   >>> INTERRUPT-DRIVEN IM 0 EXAMPLE <<<
                        0003 ;
                        0004 ;
        (0000)          0005 BAUD:   EQU  00H                ;BAUD RATE PORT
        (0007)          0006 BEEP:   EQU  07H                ;ASCII BELL CHAR
        (2000)          0007 CHAR:   EQU  2000H              ;ASCII CHAR. ADDR
        (0002)          0008 COMREG: EQU  02H                ;COMMAND REG. PORT
        (23FF)          0009 ENDRAM: EQU  23FFH              ;END SCC RAM
        (0003)          0010 INTMSK: EQU  03H                ;INTERRUPT MASK PORT
        (000A)          0011 LATCH:  EQU  0AH                ;INTREQ LATCH PORT
        (0001)          0012 RDATA:  EQU  01H                ;SERIAL RECEIVER PORT
        (2001)          0013 SECNDS: EQU  CHAR+1             ;SECONDS RAM ADDR
        (0000)          0014 STATUS: EQU  00H                ;SERIAL STATUS PORT
        (0007)          0015 TBE:    EQU  07H                ;TBE BIT D7
        (0001)          0016 TDATA:  EQU  01H                ;SERIAL XMITTER PORT
        (009C)          0017 TENMS:  EQU  9CH                ;10 MSEC TIMER COUNT
        (0008)          0018 TIMER4: EQU  08H                ;TIMER 4 PORT
                        0019                                 ;
                        0020 ;
                        0021 ;   INITIALIZE SYSTEM.  A POC OR A RESET
                        0022 ;   FORCES A JUMP TO ADDRESS 0000H BELOW
                        0023 ;
                        0024 POC:    ORG  0                  ;POC OR RESET = JP 0
                        0025                                 ;
0000  31FF23            0026         LD   SP,ENDRAM          ;INITIALIZE STACK
0003  3EC0              0027         LD   A,11000000B        ;9600 BAUD,
```

```
0005  D300        0028          OUT   BAUD,A              ;1 STOP BIT
0007  3E09        0029          LD    A,00001001B         ;INTA ENABLE,
0009  D302        0030          OUT   COMREG,A            ;RESET TMS 5501
000B  3E50        0031          LD    A,01010000B         ;UN-MASK ONLY
000D  D303        0032          OUT   INTMSK,A            ;RDA & TIMER 4
000F  ED46        0033          IM    0                   ;INTERRUPT MODE 0
0011  FB          0034          EI                        ;ENABLE INTERRUPTS
0012  C30001      0035          JP    BAKGND              ;INITIALIZATION DONE,
                  0036                                    ;START BACKGROUND
                  0037 ;
                  0038 ;   INTERRUPT SERVICE.  AN INTERRUPT FORCES
                  0039 ;   EITHER RST 20H, RST 30H OR RST 38H.  EACH
                  0040 ;   RST BELOW JUMPS ELSWHERE FOR MORE CODE ROOM
                  0041 ;
                  0042                                    ;
                  0043 RST20:   ORG   20H                 ;RDA SERVICE
0020  C32E01      0044          JP    SRVRDA              ;JUMP TO ROUTINE
                  0045                                    ;
                  0046 RST30:   ORG   30H                 ;TIMER 4 SERVICE
0030  C33701      0047          JP    SRVTM4              ;JUMP TO ROUTINE
                  0048                                    ;
                  0049 RST38:   ORG   38H                 ;SECONDS SERVICE
0038  C34801      0050          JP    SRVSEC              ;JUMP TO ROUTINE
                  0051                                    ;
                  0052 ;
                  0053 ;   BACKGROUND PROGRAM.  IT REPEATEDLY OUTPUTS
                  0054 ;   A 'PERIOD' CHARACTER TO TERMINAL AS SOON AS
                  0055 ;   TBE TESTS TRUE, INTERSPERSED WITH A 'BEEP'
                  0056 ;   EVERY 60 SECONDS.  THIS PROGRAM IS INTERRUPTED
                  0057 ;   WHENEVER ANY OF THREE SOURCES REQUEST SERVICE.
                  0058 ;
                  0059                                    ;
                  0060          ORG   100H                ;ARBTRY. ROM 0 ADDR
                  0061                                    ;
0100  3E3F        0062 BAKGND:  LD    A,'?'               ;INIT. (CHAR) TO
0102  320020      0063          LD    (CHAR),A            ;A QUESTION MARK
0105  2A0120      0064          LD    HL,(SECNDS)         ;HL POINTS TO SECNDS
0108  AF          0065          XOR   A                   ;INIT. SECONDS
0109  77          0066          LD    (HL),A              ;COUNT TO ZERO
010A  3E01        0067          LD    A,1                 ;ENABLE SECONDS
010C  D30A        0068          OUT   LATCH,A             ;INT REQ LATCH
010E  3E9C        0069          LD    A,TENMS             ;START
0110  D308        0070          OUT   TIMER4,A            ;TIMER 4
                  0071                                    ;
0112  062E        0072 LOOP:    LD    B,'.'               ;PERIOD TO
0114  CD2401      0073          CALL  OUTCHR              ;TERMINAL
0117  7E          0074          LD    A,(HL)              ;GET SECONDS COUNT
0118  D63C        0075          SUB   60                  ;MINUTE UP?
011A  20F6        0076          JR    NZ,LOOP             ;LOOP BACK IF NO
011C  77          0077          LD    (HL),A              ;YES.  RESET SECONDS
011D  0607        0078          LD    B,BEEP              ;BEEP TERMINAL IF YES
011F  CD2401      0079          CALL  OUTCHR              ;
0122  18EE        0080          JR    LOOP                ;REPEAT PATTERN
```

```
                        0081                                    ;
                        0082 ;
                        0083 ;   SUBROUTINE OUTCHR.  OUTPUTS CONTENTS OF
                        0084 ;   REG. B TO CRT TERMINAL.
                        0085 ;
                        0086                                    ;
0124  DB00              0087 OUTCHR: IN   A,STATUS              ;GET SERIAL STATUS
0126  CB7F              0088         BIT  TBE,A                 ;TBE?
0128  20FA              0089         JR   NZ,OUTCHR             ;RE-CHECK IF NO
012A  78                0090         LD   A,B                   ;TBE.  REG. B DATA
012B  D301              0091         OUT  TDATA,A               ;BYTE TO TERMINAL
012D  C9                0092         RET                        ;RETURN
                        0093 ;
                        0094 ;   IM 0 INTERRUPT SERVICE ROUTINES.  RESTARTS
                        0095 ;   FORCE A JUMP TO ONE OF THE THREE SERVICE
                        0096 ;   ROUTINES BELOW.
                        0097 ;
                        0098 ;
                        0099 ;   RDA SERVICE.  INTERRUPT FROM CRT KEYBOARD
                        0100 ;   PLACES TYPED ASCII CHARACTER IN RAM ADDRESS
                        0101 ;   'CHAR'.
                        0102 ;                                  ;
012E  F5                0103 SRVRDA: PUSH AF                    ;REGS. AF TO STK
012F  DB01              0104         IN   A,RDATA               ;READ SERIAL DATA
0131  320020            0105         LD   (CHAR),A              ;STORE AT CHAR IN RAM
0134  F1                0106         POP  AF                    ;RESTORE REGS. AF
0135  FB                0107         EI                         ;ENABLE INTERRUPTS &
0136  C9                0108         RET                        ;CONTINUE BACKGROUND
                        0109                                    ;
                        0110 ;
                        0111 ;   TIMER 4 SERVICE.  INTERRUPT FROM TIMER 4
                        0112 ;   INDICATES 10 MSEC INTERVAL UP; ASCII CHAR
                        0113 ;   IN RAM ADDRESS 'CHAR' IS THEN OUTPUT TO
                        0114 ;   CRT TERMINAL.
                        0115 ;
                        0116                                    ;
0137  F5                0117 SRVTM4: PUSH AF                    ;REGS. AF TO STK
0138  C5                0118         PUSH BC                    ;REGS. BC TO STK
0139  3E9C              0119         LD   A,TENMS               ;10 MSEC COUNT
013B  D308              0120         OUT  TIMER4,A              ;RE-START TIMER 4
013D  3A0020            0121         LD   A,(CHAR)              ;GET (CHAR)
0140  47                0122         LD   B,A                   ;AND
0141  CD2401            0123         CALL OUTCHR                ;SEND TO TERMINAL
0144  C1                0124         POP  BC                    ;RESTORE REGS. BC
0145  F1                0125         POP  AF                    ;RESTORE REGS. AF
0146  FB                0126         EI                         ;ENABLE INTERRUPTS
0147  C9                0127         RET                        ;CONTINUE BACKGROUND
                        0128                                    ;
                        0129 ;
                        0130 ;   REAL-TIME SECONDS SERVICE.  THE CONTENTS
                        0131 ;   OF RAM ADDRESS 'SECNDS' ARE INCREMENTED
                        0132 ;   IN RESPONSE TO A REAL-TIME SECONDS INTERRUPT.
                        0133 ;
```

```
                    0134                ;
0148  F5            0135 SRVSEC: PUSH AF          ;REGS. AF TO STK
0149  E5            0136         PUSH HL          ;REGS. HL TO STK
014A  AF            0137         XOR  A           ;REMOVE S-100
014B  D30A          0138         OUT  LATCH,A     ;INT BY PULSING
014D  3E01          0139         LD   A,1         ;BIT D0 LOW,
014F  D30A          0140         OUT  LATCH,A     ;THEN BACK HIGH
0151  2A0120        0141         LD   HL,(SECNDS) ;POINT TO SECONDS
0154  34            0142         INC  (HL)        ;BUMP COUNT
0155  E1            0143         POP  HL          ;RESTORE REGS. HL
0156  F1            0144         POP  AF          ;RESTORE REGS. AF
0157  FB            0145         EI               ;ENABLE INTERRUPTS
0158  C9            0146         RET              ;CONTINUE BACKGROUND
                    0147                ;
      (0000)        0148         END
```

The last example program of this section illustrates an IM 1 polled interrupt system, assuming the same system hardware and overall system behavior as in the previous example. The difference lies in the 5501 initialization (COMMAND REGISTER bit INE is logic 0 instead of logic 1), in the Z80A interrupt mode (IM 1 is executed rather than IM 0), and in the service routine structure. An interrupt request from any of the three sources drives Z80A pin $\overline{\text{INT}}$ active low as before, but instead of reading an RST opcode from the 5501, the Z80A automatically performs an RST 38H, regardless of the interrupt source.

EXAMPLE 9

This example converts the interrupt-driven system described in the previous example to an 'equivalent' polled interrupt system. All system hardware is assumed the same, and for brevity, only the changed program segments appear below. In summary, memory locations 0008H, 000FH are replaced with new operands; the RST 20H thru RST 30H service areas 0020H - 0037H are now not used; and the RST 38H area starting at 0038H completely changes. It now polls the 5501 INTERRUPT ADDRESS port to determine the interrupt source, and directs program flow by jumping to the appropriate service routines.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                    0001 ;
                    0002 ;    >>> POLLED INTERRUPT IM 1 EXAMPLE <<<
                    0003 ;
                    0004 ;
      (0003)        0005 INTADR: EQU  03H                ;INTERRUPT ADDR PORT

                         (THE OTHER EQUATES ARE THE SAME)

                    0021 ;
                    0022 ;    INITIALIZE SYSTEM.  A POC OR A RESET
                    0023 ;    FORCES A JUMP TO ADDRESS 0000H BELOW
                    0024 ;
0000'               0025 POC:     ORG  0                 ;POC OR RESET = JP 0
                    0026                                 ;
0000  31FF23        0027          LD   SP,ENDRAM         ;INITIALIZE STACK
0003  3EC0          0028          LD   A,11000000B       ;9600 BAUD,
0005  D300          0029          OUT  BAUD,A            ;1 STOP BIT
0007  3E01          0030          LD   A,00000001B       ;INTA DISABLE,
0009  D302          0031          OUT  COMREG,A          ;RESET TMS 5501
000B  3E50          0032          LD   A,01010000B       ;UN-MASK ONLY
000D  D303          0033          OUT  INTMSK,A          ;RDA & TIMER 4
000F  ED56          0034          IM   1                 ;INTERRUPT MODE 1
0011  FB            0035          EI                     ;ENABLE INTERRUPTS
0012  C30001        0036          JP   BAKGND            ;INITIALIZATION DONE,
                    0037                                 ;START BACKGROUND
                    0038 ;
                    0039 ;    INTERRUPT SERVICE.  AN IM 1 INTERRUPT
                    0040 ;    FORCES A RST 38H ONLY.  THE POLLING AND
                    0041 ;    SERVICE VECTORING ROUTINE FOLLOWS.
                    0042 ;
                    0043                                 ;
0015                0044 RST38:   ORG  38H               ;IM 1 SERVICE ADDR
                    0045                                 ;
0038  DB03          0046          IN   A,INTADR          ;GET INTERRUPT ADDR
003A  FEE7          0047          CP   0E7H              ;RDA?
003C  CA2E01        0048          JP   Z,SRVRDA          ;JP SRVRDA IF YES
003F  FEF7          0049          CP   0F7H              ;NO.  TIMER 4?
0041  CA3701        0050          JP   Z,SRVTM4          ;JP SRVTM4 IF YES
0044  C34801        0051          JP   SRVSEC            ;NO. MUST BE SEC
                    0052                                 ;

                (THE REST OF THE PROGRAM IS UNCHANGED)

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

Section 3

SCC System Considerations

This section focuses on what are generally termed 'system considerations'. It is assumed that the user has read Section 2, Operating Instructions, and has an understanding of the SCC's memory organization, its I/O port assignments and their use, and its interrupt structure. This section first enumerates the steps and options involved in SCC system initialization, then continues with an introduction to Cromemco's 3K Control Basic and SCC Monitor programs, and concludes with a discussion of Control Basic and CDOS SCC development systems.

3.1 SCC SYSTEM INITIALIZATION

As discussed in Section 2.2, either applying power to the SCC card, or resetting the SCC by an active low level on S-100 bus line $\overline{\text{RESET}}$, triggers the following events:

(1) The Z80A is reset, meaning; interrupts are disabled, Register I is cleared to 00H, Register R is cleared to 00H, the interrupt mode is set to IM 0, and the Program Counter is cleared to 0000H. From this initialized state, the Z80A begins executing whatever program code starts at memory address 0000H.

(2) SCC parallel output ports OUT 0AH and OUT 0BH are cleared to data 00H.

(3) SCC on-board memory spanning 0000H thru 23FFH is unconditionally enabled.

Since a jump to address 0000H is performed, and since SCC socket ROM 0 spanning addresses 0000H thru 07FFH is

unconditionally enabled, then some code must be placed in ROM 0 to start-up the SCC system. This code, starting at address 0000H, would typically consist of a system initialization routine, or a jump to a system initialization routine located elsewhere in memory. Notice that while the Z80A is automatically reset after a POC, the 5501 is not. After a POC all 5501 register contents are undefined. This means that the device may 'come up' with its interval timers in mid-count, with an undefined baud rate, with random interrupts pending, etc. Thus the device must be initialized, under software control, thru bytes sent to its COMMAND REGISTER, BAUD RATE register, and INTERRUPT MASK to prevent spurious interrupts and other unpredictable behavior. The tasks and options to be considered during system initialization then include at least the following:

(1) Move the stack to RAM by loading Register SP.

(2) Select 5501 COMMAND REGISTER options:

(a) Select between high baud and low baud with bit HBD. Note that HBD effects both the serial baud rate and the interval timer count rates.

(b) Select between automatic INTA response (IM 0), or no 5501 response to an INTA cycle, with bit INE.

(c) Select between PI7 and TIMER 5 as the lowest priority interrupt source with bit RS7.

(d) Reset the 5501 by forcing bit RES to logic 1. This forces the serial receiver into the search mode; status bits RDA, SBD, ORE and FME are reset; the serial transmitter output goes high (marking); the interrupt request register is cleared except for request TBE, which is set; and the interval timers are reset to timeout without generating an interrupt request.

(3) Select the serial baud rate by loading BAUD RATE port OUT 00H.

(4) Selectively mask or unmask each 5501 interrupt source by

loading INTERRUPT MASK port 03H.

(5) Initialize the contents of 5501 parallel port OUT 04H. This port is not cleared automatically upon a POC or a RESET like parallel ports OUT 0AH and OUT 0BH.

(6) If an interrupt mode other than IM 0 is to be used, then execute either an IM 1 or an IM 2 instruction. If IM 2 is used, then initialize Register I.

(7) If interrupts are to be enabled, execute an EI instruction.

(8) If SCC memory is to be disabled, then cut PC board trace labeled 'DISABLE', and the code which sets bit D7 of port OUT 0AH must be located in, and executed from off-board memory to maintain program continuity.

(9) If Cromemco products with Bank Select are used in the system, configure the memory banks by loading bank select ports OUT 40H.

If Cromemco's 3K Control Basic firmware is resident in SCC sockets ROM 0 and ROM 1, then a POC or a RESET causes the SCC system to be automatically initialized as follows:

(1) An 00H is loaded into INTERRUPT MASK port 03H which masks-off all eight 5501 interrupt sources.

(2) If the Control Basic AUTORUN statement is used which automatically begins stored program execution, then byte 09H is loaded into COMMAND REGISTER port OUT 02H (INTA Enable and 5501 RESET), and 0C0H is loaded into BAUD RATE port OUT 00H (9600 baud, one stop bit).

(3) If the AUTORUN statement is not used, then a cold start entry is made to the Control Basic interpreter after several 'Return' keystrokes on the user's terminal. The Return characters are used to sense the terminal speed, and if the terminal baud rate is 19,200 baud, then Control Basic will load a 90H into the BAUD RATE register and 11H into the COMMAND REGISTER (High Baud and 5501 RESET). If the terminal baud rate is not 19,200 baud, then bit HBD is reset and the BAUD RATE register is made to match the terminal speed and stop bit number.

After these initialization tasks are completed, it should be noted that the SCC system may be re-configured by Control Basic's 'OUT' command, which may be used either as a direct command or as

a statement in a running program.

3.2 3K CONTROL BASIC AND SCC MONITOR

Cromemco supplies a Basic interpreter program and an SCC Monitor program in a two ROM chip set, Model MCB-216. The Basic interpreter is a slightly modified and relocated version of Cromemco's earlier 3K Control Basic, model CB-308, while SCC Monitor is a slightly modified and relocated version of Cromemco's Z80 Monitor program, model ZM-108. The two ROM chips occupy SCC sockets ROM 0 and ROM 1, spanning the address range 0000H - 1FFFH. The SCC memory map with model MCB-216 installed is shown in Figure 25. The interactive 3K Control Basic language allows the user to enter program statements from a terminal into RAM, run the program text and receive error messages at the terminal, modify, delete or add program statements, and re-run the updated program text. Between program runs the user may interrogate input/output ports, and initialize or print out Basic variable values by entering any Basic statement as a direct command, or the SCC Monitor program may be entered by issuing the 'QUIT' command. At this point, any SCC Monitor command may be issued (e.g., display memory, display Z80 registers, move memory, etc.), followed by a warm start re-entry to Control Basic.

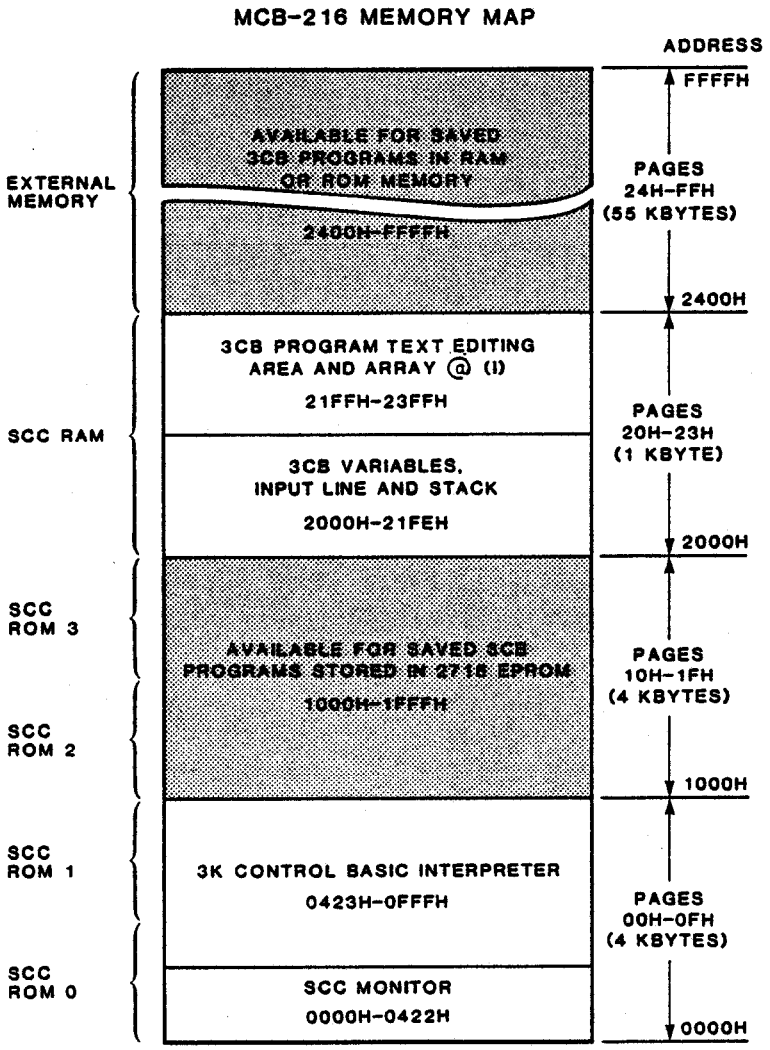


Figure 25: SCC MEMORY MAP WITH MCB-216

After verifying proper program behavior, the program text may be stored permanently in 2716 EPROM memory for later execution if the system contains a Cromemco 32K Bytesaver card. Storage is accomplished by placing an erased 2716 device in one of the 32K Bytesaver sockets, positioning its PROGRAM POWER switch to the ON position, then issuing a Control Basic 'SAVE' command (see 32K Bytesaver Instruction Manual, part number 023-0002, for full

programming details). Subsequently, this 2716 firmware may be transferred to SCC sockets ROM 2 or ROM 3 for execution, or it may be executed while resident on the 32K Bytesaver Card. Control Basic provides convenient linkage to assembly language subroutines from a running Control Basic program with the 'CALL' statement, to memory with the 'GET' function and the 'PUT' statement, and to input/output ports with the 'IN' function and the 'OUT' statement. A POC or RESET can be made to initiate automatic Basic program start-up by beginning the program text on page 10H with an 'AUTORUN' statement.

The following paragraphs are intended to provide only an introduction to Cromemco's 3K Control Basic language. For full details, refer to Cromemco's 3K Control Basic Instruction Manual, part number 023-0023.

Numbers and Constants: All Control Basic numbers are integers in the range of -32,767 to +32,767. Numbers are stored internally in two's complement form as 16-bit quantities in Intel's low-high style. For programming and input/output, constants may be expressed in either decimal, hexidecimal, or in encoded ASCII form.

Variables: There are 52 variables denoted by letters A thru Z and A0 thru Z0, and also one singly subscripted array '@(I)', whose elements are then referred to as @(0), @(1), @(2), ... , and so on. The dimension of this array is automatically set to make use of all the memory space that is allocated to, but left unused by,

the text of the current program. The same array space used by array @(I) to store two-byte Control Basic values may also be referenced as array '&(I)' to store and retrieve single byte values, and may also be referenced as array '$(I)' to store and retrieve string (alphanumeric) data.

Functions: Control Basic has ten built-in functions, described below. In the following, X and Y denote either a constant, a variable, or an expression which has a value.

ABS(X)	evaluates to the absolute value of X.
RND(X)	evaluates to a pseudo-random number between 1 and X inclusive.
SIZE	evaluates to the number of bytes allocated to, but left unused by the current program in the text area.
SGN(X)	evaluates to +1 if X is non-negative and -1 if X is negative.
AND(X,Y)	evaluates to the 16-bit Boolean AND of X and Y.
OR(X,Y)	evaluates to the 16-bit Boolean OR of X and Y.
XOR(X,Y)	evaluates to the 16-bit Boolean exclusive-OR of X and Y.
GET(X)	evaluates to the 8-bit unsigned binary value of the contents of memory address X.
IN(X)	evaluates to the 8-bit unsigned binary value read from input port X.
LOC	evaluates to the absolute memory address of @(0).

Arithmetic and Compare Operators: Control Basic allows the following operators in variable and constant expressions:

/	divide.
*	multiply or logical AND.
-	subtract, or two's complement.
+	add, or logical OR.
>	greater than (compare).
<	less than (compare).
=	equal to (compare).
#	not equal to (compare).

>= greater than or equal to (compare).
<= less than or equal to (compare).

In a logical context, Control Basic treats value 0 as FALSE and any non-zero value as TRUE; thus the addition operator (+) may be used as a logical OR operator, and the multiplication operator (*) may be used as a logical AND operator. Control Basic evaluates every comparison between two values as either 0 (FALSE) or 1 (TRUE).

Direct Commands and Statements: Listed below is Control Basic's repertoire of statements and direct commands. Statements are preceded by a line number; they become a part of the stored program text; and their execution is deferred until a RUN or an equivalent direct command is issued. Direct commands on the other hand are not preceded by a line number; they do not become a part of the stored program text; and they are executed immediately.

FORM	STMNT	COMND	REMARKS
AUTORUN	X	X	Automatic program start-up.
CALL	X	X	Assembly language subroutine call.
EPROM		X	Program 2708 EPROMs command.
FOR-TO-STEP	X	X	Loop control.
GOTO	X	X	Unconditional branch.
GOSUB	X	X	Control Basic subroutine call.
IF-THEN	X	X	Conditional statement execution.
INPUT	X	X	Inputs terminal data to program.
LIST		X	Lists program text to terminal.
LOAD		X	Loads text area for program editing.
LOCK		X	Delimits program text area in memory.
NEW		X	Clears program text area.
NEXT	X	X	Delimits FOR-TO-STEP loop.
NULL		X	Sets number of NULLs after CR-LF.
OUT	X	X	Loads an output port.
PRINT	X	X	Outputs program data to terminal.
PUT	X	X	Loads a RAM memory location.
QUIT		X	Exits Basic and enters SCC Monitor.
RDOS		X	Exits Basic and enters RDOS.

RETURN	X	X	Control Basic subroutine return.
REMARK	X	X	Program annotation.
RUN	X	X	Begins stored program execution.
SAVE		X	Moves program text to save area.
STOP	X	X	Stops stored program execution.
WIDTH		X	Sets printed page width.

Control Basic Program Compaction: Most Control Basic statements and functions may be represented in abbreviated form in the stored program text. Multiple statements separated by semi-colons on one program line are also allowed. These two features greatly reduce the amount of memory required to store Control Basic program text. This is especially important if the program text is to be committed to EPROM memory for future use. For example;

```
10 FOR J = 1 TO 1000 STEP 2
20 PRINT J,
30 NEXT J
```

could be compacted to the following equivalent form;

```
10 F.J=1TO1000S.2;P.J,;NE.J
```

and in the process, would only require roughly half the memory space.

An SCC Monitor program is also included with 3K Control Basic on the MCB-216 chip set. Control is passed from Control Basic to the SCC Monitor program by issuing the Basic 'QUIT' direct command. Control is passed from SCC Monitor back to Control Basic in one of three ways; SCC Monitor command 'G 423' is issued for a Basic initial or cold start, a 'G 447' is issued for a new Basic program warm-start, and either a 'B' or a 'G 452' for a Basic

stored program warm-start (see below for command descriptions).

The following paragraphs again only summarize the SCC Monitor program features. For full details, refer to Cromemco's Z80 Monitor Instruction Manual, part number 023-0021.

Monitor Command Prompt and Formats: The Monitor program prompts the user for a new command with the colon character, ':'. The user then responds with one or two letters which abbreviate a legal Monitor command, followed by none, one, two or three arguments. When the arguments specify memory addresses, the arguments are assumed hexadecimal values. When a range of memory addresses is specified, either absolute starting and ending addresses, or an absolute starting address followed by a 'swath' specification may be supplied. For example, both Monitor commands below cause the contents of memory locations 300H thru 4FFH (200H bytes) to be formatted and displayed on the user's terminal;

```
:DM 300 4FF <CR>

:DM 300 S200 <CR>
```

where <CR> stands for a RETURN keystroke.

Monitor Command Summary: A summary of the SCC Monitor commands appears below along with an example use.

CMND	EXAMPLE USE	FUNCTION
B	:B <CR>	Return control to Control Basic by performing a stored program warm start.
DM	:DM 100 S20 <CR>	Display Memory. Format and display the contents of memory addresses 100H thru 11FH inclusive on the user's terminal.
DR	:DR <CR>	Display Registers. Format and display the current contents of the Z80 registers F, A, B, C, D, E, H, L, I, N, SP, IX, IY, A', B', C', D', E', H' and L'.
E	:E A <CR>	Examine input port. The contents of system input port 0AH are read and displayed on the user's terminal.
G	:G 500/50F <CR>	Goto Address. If no argument is supplied, program execution resumes at the current location in register SP. If a single argument is supplied, a jump to that address is performed. If two arguments are supplied, a jump to the first argument is performed, and a break-point is specified with the second (jump to 500H with a breakpoint at 50FH in the example). The registers are stored at an encountered breakpoint, and they may be examined with the 'DR' command.
I	:I <CR>	Initialize baud rate. After <CR>, the terminal baud rate may be changed, and Monitor senses the new baud rate after several 'Return' keystrokes.
M	:M 100 S10 200 <CR>	Move data. Moves the data at addresses 100H thru 10FH to the destination area 200H thru 20FH.
O	:O 3D 40 <CR>	Output data. Output data byte 3DH to system output port number 40H.
P	:P 0 S400 E000 <CR>	Program 2708 EPROMs. Programs a 2708 EPROM chip with a starting address of E000H, and resident in a Cromemco Bytesaver II card, with memory data 0000H thru 03FFH.
R	:R 2000 S100 <CR>	Read paper tape. Reads binary or ASCII

		paper tape data and stores the data at 2000H thru 20FFH.
SM	:SM 2300 <CR>	Substitute memory. User supplies memory data at terminal. Data fills memory starting at address 2300H. Data entries are separated by a space character and are terminated with a <CR>.
SR	:SR A' <CR>	Substitute register. Substitute a new value for Z80 register A'. Monitor prompts for the hex value with a period character.
V	:V 0 S400 E000 <CR>	Verify memory. Verify that memory data blocks 0000H thru 03FFH and E000H thru E3FFH are identical. Any discrepancies are formatted and displayed on the user's terminal.
W	:W 2300 S100 <CR>	Write to terminal. Writes binary or ASCII data in memory area 2300H thru 23FFH to the terminal. This command is used for punching paper tapes.

3.3 SCC CONTROL BASIC DEVELOPMENT SYSTEMS

Source programs for an SCC system may be written in Z80 assembly language, 3K Control Basic, or in a higher level language such as Cromemco's FORTRAN IV. Developing system programs in 3K Control Basic, with linkages to only small hand-assembled Z80 subroutines, requires the minimum amount of system development hardware. A 'minimum system', which would include a powered S-100 bus, an SCC card, a Cromemco 32K Bytesaver card, an RS-232/ 20 mA terminal, and one or more erased 2716 EPROMs, is shown in Figure 26.

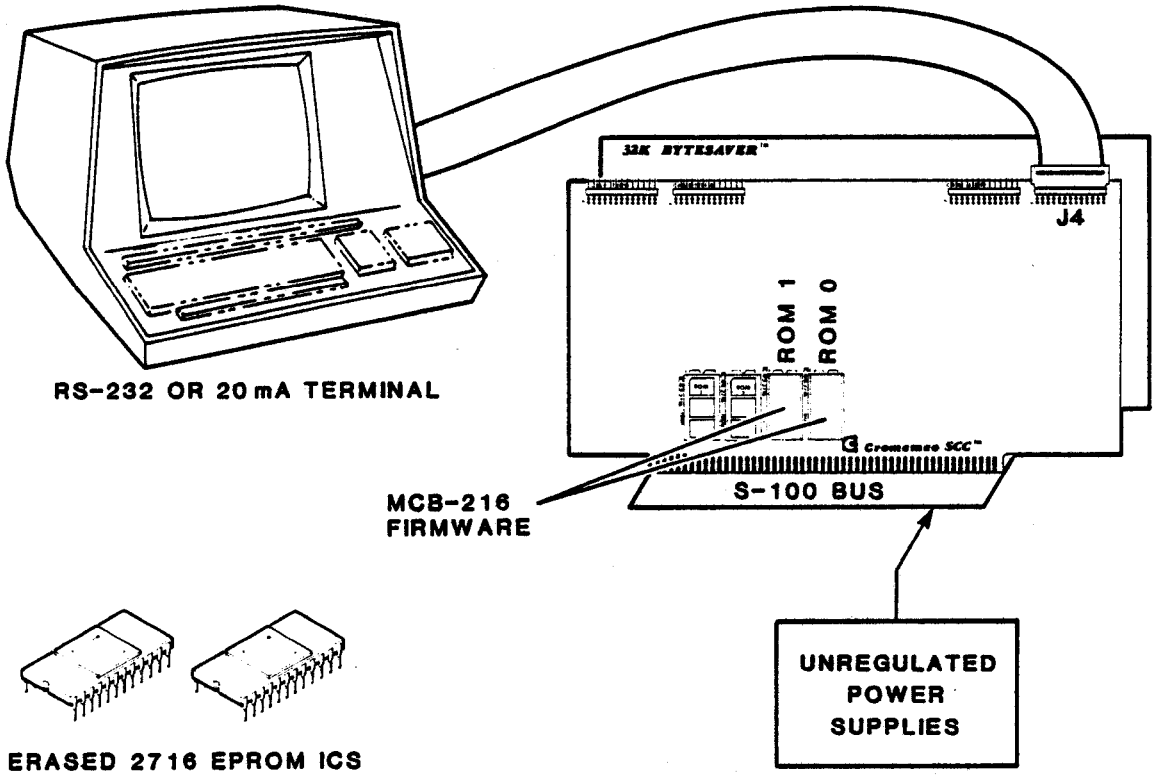


Figure 26: MINIMUM SCC DEVELOPMENT SYSTEM

In this system, program text would be entered from the terminal keyboard into the Basic program text area, spanning SCC RAM addresses 2201H thru 23FFH. The text area capacity is 511 (decimal) bytes, or approximately 2 'pages', where 1 page = 256 bytes. Each Basic program requires six bytes of overhead memory space. In addition, three bytes of overhead are required for each program line--two for the line number and one for a line delimiting CR character. Any other program character, including a space character, occupies one byte per character (spaces appearing after a program line number but before the line statement are not stored, however). Put another way, the text capacity is approximately 50 Basic program lines of seven characters per line,

or 7 Basic program lines of 70 characters per line. Very efficient use of the text area may be made if the user also takes full advantage of the Basic statement abbreviations and multiple statements per line features. It should also be noted that array @(I) is competing with program characters for text area space. Array @(I) begins with element @(0) at the end of the program text, and continues with @(1), @(2), and so on, up to the end of the text area. Each array element requires two bytes of storage.

After interactively verifying text area program behavior, the entire program text may be moved to any whole page boundary in memory by using the Basic 'SAVE ppp' command, where 'ppp' is a page number specification between 0 and 255. This command is typically used to move the program text to RAM memory, but it may also be used to program a 2716 EPROM resident in a 32K Bytesaver card. The following example will illustrate this procedure.

EXAMPLE 10

Assume an SCC system contains a 32K Bytesaver card which is switch-mapped into memory C000H thru FFFFH. Assume further that an erased 2716 EPROM occupies a 32K Bytesaver socket spanning addresses C000H thru C7FFH (or pages 192 thru 199 decimal).

The following Basic program is then entered into the text area from the terminal keyboard:

```
10 REM: THIS BASIC PROGRAM IS LOCATED ON PAGE 10H = 16D,
20 REM: AND IT IS LINKED TO THE BASIC PROGRAM ON PAGE 12H = 18D.
30 REM: DEFINE VARIABLES & ARRAY ELEMENTS.
40 A=0; B=1; C=2; @(0)=1; @(1)=2
50 REM: OUTPUT VALUES TO TERMINAL.
60 PRINT A,B,C,@(0),@(1)
70 REM: RUN PROGRAM STARTING AT PAGE 12H = 18D.
80 RUN 18
```

This program occupies 312D = 138H bytes, or 1.22 pages in the text area. To store this program in 2716 EPROM, the 32K Bytesaver

PROGRAM POWER switch would be positioned ON, and the Basic direct command 'SAVE 192' would be issued. Basic replies 'SAVED ON PAGE %C0 TO %C1' after 2716 programming is complete, where the prefix '%' is taken to mean the number following is in hexadecimal. The stored Basic program text then occupies the address range C000H thru C138H, or two of the 2716's eight page memory capacity. After programming, the PROGRAM POWER SWITCH would be positioned OFF.

Now assume the text area is cleared by issuing a Basic 'NEW' direct command, and the following program entered in the text area:

```
10 REM: THIS BASIC PROGRAM IS LOCATED ON PAGE 12H = 18D,
20 REM: AND IT IS LINKED TO THE BASIC PROGRAM ON PAGE 10H =16D.
30 REM: NOTE RE-USE OF LINE NUMBERS.
40 REM: ARRAY VALUES AND VARIABLE VALUES ARE GLOBAL QUANTITIES;
50 REM: THEY ARE DEFINED BY THE PROGRAM ON PAGE 16D.
60 REM: OUTPUT VALUES TO TERMINAL.
70 PRINT A+B-C, @(0)+@(1)
80 REM: GO BACK AND RE-RUN PROGRAM ON PAGE 16D.
90 RUN 16
```

This program occupies 390D = 186H bytes, or 1.52 pages in the text area. This program could also be stored in the same 2716 EPROM on pages 194 and 195 by turning the PROGRAM POWER switch ON, and issuing the Basic direct command 'SAVE 194'. Basic would then reply, 'SAVED ON PAGES %C2 TO %C3' after programming is complete. The actual program text would then span addresses C200H thru C386H on the 2716. The programs are designed to be resident at pages 16D and 18D, so to run them, the system power would be turned OFF, the 2716 would be removed from the 32K Bytesaver card and inserted in SCC socket ROM 2 (which spans pages 16D thru 23D). System power would then be turned ON, and after several 'Return' keystrokes, Control Basic would prompt: 'OK >'. Issuing the direct command 'RUN 16' would then begin execution of the linked Basic programs.

Note that these programs could also be run while resident on the 32K Bytesaver card by changing program line 80 in the first program segment to '80 RUN 194', and program line 90 in the second program segment to '90 RUN 192'. Program execution would then commence after issuing a 'RUN 192' direct command. Also note that these two heavily remarked and un-abbreviated programs make very

poor use of the Basic text editing area. Refer to the Basic programs presented in Section 5 for good examples of efficient text area usage.

The inconvenience of linking together 2-page text area segments in EPROM memory to form larger system programs may be easily overcome by adding more RAM memory to the Basic text editing area. Cromemco's 4KZ static RAM card mapped into addresses 2000H thru 2FFFH would expand the text area from 2201H thru 23FFH (511 bytes) to 2201H thru 2FFFH (3582 bytes), or approximately a seven-fold increase. Cromemco's 16KZ dynamic RAM card, mapped into addresses 0000H thru 3FFFH would expand the text area to 2201H thru 3FFFH (7678 bytes), or roughly a fifteen-fold increase. If more memory is added to the development system to expand the editing area, two restrictions must be observed:

(1) The additional memory must be RAM, and it must overlap, or be contiguous with SCC RAM memory, which ends at address 23FFH.

(2) The end of the text area is automatically set to 2400H by an SCC Basic initial start. The contents of memory location 21FDH contains this value in low-high Intel style, i.e., (21FDH), (21FEH) = 00H, 24H. The values in these locations may be changed to point to the new end of text either with the Basic 'PUT' direct command/statement, or with the Basic 'LOCK' direct command. For example, if using the 4KZ RAM card, the end of text area would be specified with either a 'PUT(%21FD)=%3000' command or program statement, or a 'LOCK %30' command. If using the 16KZ RAM card, the corresponding commands would be 'PUT(%21FD)=%4000' and 'LOCK %40'.

With an enlarged Basic text area, more than one 32K Bytesaver-resident 2716 EPROM may be programmed sequentially by a single Basic 'SAVE' command. Note that the 2716's must maintain their relative positions if moved to other sockets to maintain the

correct program text sequence. It is anticipated then that in the typical small, dedicated SCC application, programs would be developed in Control Basic and the program text committed to one or two 2716 EPROMs loaded into SCC sockets ROM 1 and ROM 2 (which allows for 4 KBytes of program text). The Control Basic interpreter/SCC Monitor would reside in SCC sockets ROM 0 and ROM 1, and SCC RAM memory would be used only for Basic variables, input line, stack (in memory area 2000H thru 21FEH), and array @(I) storage (in memory area 2201H thru 23FFH). Notice that with all of the program text in 2716 EPROM and none in the text area, array @(I) would start at 2203H, leaving room for over 250 two-byte array values. In addition to Basic text, ROM 1 and ROM 2 could also contain small hand-assembled Z80 routines which are CALLed by the Basic program text where high processing speed is required, and any part of SCC RAM memory not used by array @(I) could be used for scratchpad memory either by the Basic program or by assembly routines.

3.4 CDOS DEVELOPMENT SYSTEMS

For maximum performance from an SCC system, both in terms of execution speed and code density, source programs would typically be written directly in Z80 assembly language or FORTRAN IV. The final system object code would ordinarily populate SCC sockets ROM 0 thru ROM 3, which provide 8 KBytes of firmware code space, and the SCC's 1 KByte of RAM would be used for the stack and scratch pad memory. Cromemco supplies software support for these languages on either 5" or 8" floppy diskettes. The Cromemco

Relocatable Macro Assembler package (model FDA-S on the small disk and FDA-L on the large disk) provides a two-pass Z80 assembler which reads source code from a disk file, assembles it, and produces a relocatable object file and listing. In the same package, program LINK may be used to load the assembled code into memory, and program DEBUG may be used to trace, insert break-points, disassemble, patch or program 2716 EPROMs with the object code. Cromemco's FORTRAN IV software package (models FDF-S and FDF-L) provides a complete implementation of ANSI standard FORTRAN X3.9-1966, except for complex data types. Cromemco FORTRAN IV operates as a part of CDOS (Cromemco Disk Operating System) to provide disk file access. Since this FORTRAN IV produces relocatable code, FORTRAN modules may be linked with code produced by the Cromemco Relocatable Macro Assembler, and the Assembler can access scientific and arithmetic routines located in the FORTRAN IV library. This library is searched by the linker to resolve any undefined subroutine calls, therefore, only the subroutines and system routines required to run Cromemco FORTRAN programs are loaded before execution. The user is referred to the Cromemco Macro Assembler Instruction Manual, part number 023-0039, and the Cromemco FORTRAN IV Instruction Manual, part number 023-0038, for full details on these languages. It is the intent of this subsection only to provide the information necessary to modify the SCC card to make it suitable for use in a Cromemco CDOS system, so that these two disk software packages (and any of the other Cromemco supplied disk software packages, such as COBOL, Structured FORTRAN, 16K Extended Basic, Trace System Simulator,

Word Processing System and Data Base Management) may be used.

The minimum hardware requirements for a CDOS system are: a powered S-100 bus, at least 32 KBytes of contiguous RAM starting at address 0000H, a Cromemco ZPU or modified SCC card (see below), a Cromemco 4FDC Floppy Disk Controller card, one to four Cromemco supplied floppy disk drives, and an RS-232/20 mA current loop terminal. The SCC cannot directly replace a Cromemco ZPU card in a CDOS system as I/O port numbers 00H thru 09H are used by both the SCC and the 4FDC card, and also because of the memory conflict between the SCC on-board memory and the 32 KBytes of external memory. Therefore, the SCC I/O ports must be re-addressed, and a short bootstrap program must be placed in SCC socket ROM 0 which initializes the 5501, disables the SCC on-board memory, then jumps to the CDOS Monitor program, RDOS. These steps are explained in detail later below.

There are two levels of I/O addressing on the SCC; base address and function address. When the Z80A addresses an I/O port with address lines A7(MSB) thru A0(LSB), then lines A7 thru A4 define the SCC <u>base address</u> (the port number MSD), while address lines A3 thru A0 (the port number LSD) are used to define the SCC <u>function address</u>. In its factory shipped condition, the SCC responds to a base address of 0000, and as a consequence all SCC port numbers lie between <u>0</u>0H and <u>0</u>FH. Changing the base address to 0010, for example, would cause the SCC to respond to I/O port numbers between <u>2</u>0H and <u>2</u>FH. The SCC base address is sensed by IC24, and a high at its output pin 5 is fed as an enable to the

function decoding ROM, IC25. Address lines A3 thru A0 feed this ROM as inputs. These lines define the I/O port number LSD, which may also be properly termed an 'offset' from the base address, where an offset of +0 from the base address is the base address itself, an offset of +1 from the base address is one more than the base address, and so on. Since this is a four bit value, all offsets must range from +0 to +15. The function decoding ROM accepts the offset as an address input, and its 8-bits of data output are used to selectively enable circuitry on the SCC corresponding to addressed I/O port, and disable those not addressed. Thus there are two methods of re-assigning SCC I/O port numbers; changing the SCC base address thru a hardware modification to the input lines which feed IC24, and changing the ROM data in IC25 thereby changing the SCC I/O circuitry which is enabled by each offset address from +0 to +15. Changing the base address by a hardware fix has the disadvantage of being 'permanent', and the offsetting advantage of keeping all SCC I/O ports. If the base address is changed, Cromemco recommends its value be set to 0010 to avoid conflicts with all other Cromemco product base addresses. Changing the function addresses by replacing ROM IC25 has the advantage of being 'temporary', as the original ROM may easily be re-inserted to restore the SCC to its factory shipped condition. The disadvantage of replacing ROM IC25 and leaving the base address set to 0000 is that offsets +0 to +9 may not be used by the SCC; since they are used by the 4FDC card. Thus only offsets +10 thru +15 may be used, meaning only six SCC I/O ports may be used. Since twelve SCC I/O ports are used on the

factory shipped SCC, six I/O ports must be permanently disabled. Information will now be presented on how to change the SCC I/O base address, and how to change the ROM contents to alter the function addressing. Modifying the function addressing is discussed first.

Cromemco supplies a ROM replacement part for IC25, which should be referred to when ordering as part number 74904-1. Inserting this part into socket IC25 generates the following function addresses:

FUNCTION ADDRESS PORT NUMBER CHANGES

PORT NUMBER	INPUT FUNCTION	OUTPUT FUNCTION
0AH	J1 PARALLEL INPUT	J1 PARALLEL OUTPUT
0BH	J2 PARALLEL INPUT	J2 PARALLEL OUTPUT
0CH	SERIAL STATUS	BAUD RATE
0DH	SERIAL RECEIVER DATA	SERIAL XMITTER DATA
0EH	NOT USED	COMMAND REGISTER
0FH	INTERRUPT ADDRESS	INTERRUPT MASK

Notice that TIMER 1 thru TIMER 5 are disabled, and additionally parallel ports IN 03H and OUT 03H (and thus also interrupt source RS7) are disabled. The user may optionally wish to program a 74S288 with the ROM pattern shown below to achieve the same results instead of ordering part number 74904-1 from Cromemco:

74S288 ROM DATA FOR 4FDC COMPATIBILITY

ADDRESS	DATA
00 - 0F	FC FC FC FC FC FC FC FC FC FC 7E 7D 18 00 FC 10
10 - 1F	FC FC FC FC FC FC FC FC FC FC 7E 7D 28 30 20 40

The SCC I/O base address is changed from 0000 to 0010 by inverting line A5 before it is fed to IC24 as an input (see Figure 27). An unused section of IC18, a 74LS04, is used to perform the inversion.

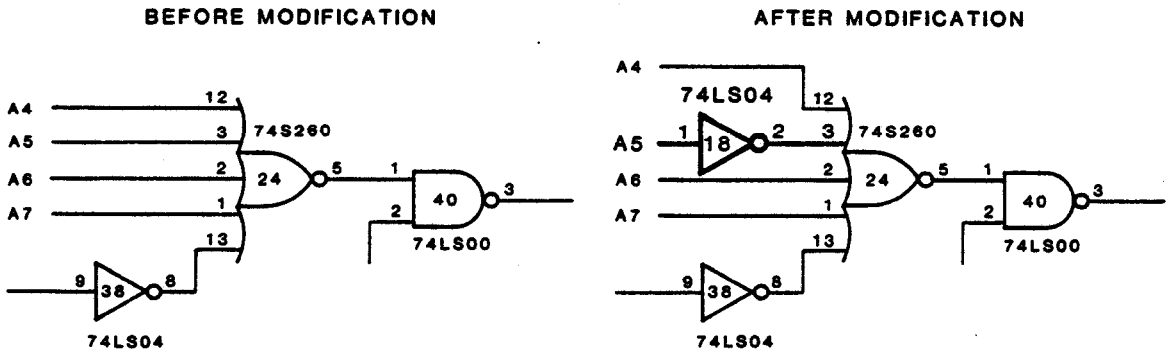


Figure 27: SCC BASE ADDRESS SCHEMATIC MODIFICATION

To make the base address hardware modification, turn the SCC card so that the component side is facing upward, then carefully cut the leftmost foil trace which runs from IC24, under capacitor C14, to IC34. It is advisable to temporarily remove C14 while cutting the trace to provide more working space. After the trace has been cut, turn the board over so that the solder side is

facing upward, then install an insulated jumper wire from IC34 pin 5 (which connects to address line A5) to IC18 pin 1, then install another insulated jumper wire from IC18 pin 2 to IC24 pin 3 (see Figure 28). If this hardware modification is made, update the SCC Schematic Diagram as per Figure 27.

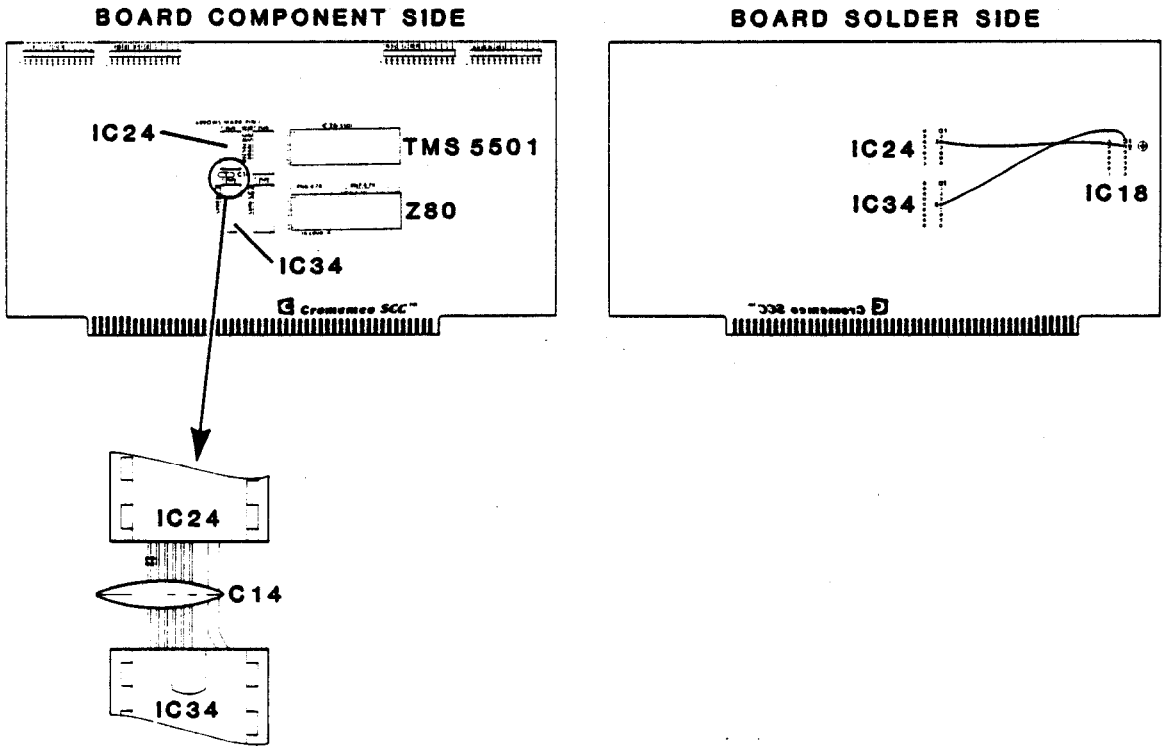


Figure 28: SCC BASE ADDRESS HARDWARE MODIFICATIONS

Either the function address ROM change, or the base address hardware modification described above, make the SCC hardware compatible with the 4FDC card. To now make it compatible with the CDOS operating system, which requires at least 32 KBytes of RAM starting at address 0000H, it will also be necessary to disable the SCC on-board memory after a POC or RESET by executing the bootstrap program listed below. IMPORTANT NOTE: Be sure to cut

the trace labeled 'DISABLE' which allows SCC memory to be disabled by a logic 1 bit D7 at port OUT 0AH. The program has three functions:

(1) It initializes the 5501.

(2) It loads a short program into external RAM.

(3) It jumps to, and executes the program which was loaded in external RAM. The program disables the SCC on-board memory and jumps to the RDOS program.

Note that the program is written assuming a function address change with BAUD RATE port number 0CH, COMMAND REGISTER port number 0EH, and INTERRUPT MASK port number 0FH--the new function addresses assigned by the IC25 replacement ROM. If a base address hardware modification is made instead, the SCC I/O port number EQUate statements would be changed in the source code to reflect the new base address. In this case, the BAUD RATE port number would be equated to 20H, the COMMAND REGISTER port number to 21H, and the INTERRUPT MASK port number to 23H.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                    0001 ;
                    0002 ;  THIS PROGRAM MAKES THE SCC COMPATIBLE WITH
                    0003 ;  THE HARDWARE REQUIREMENTS OF CDOS. IT ASSUMES
                    0004 ;  ROM 74903-1 IS INSERTED IN SCC SOCKET IC25,
                    0005 ;  THAT THE 'DISABLE' TRACE HAS BEEN CUT, AND
                    0006 ;  THAT AT LEAST 32 KBYTES OF RAM STARTS AT 0000H.
                    0007 ;  THE PROGRAM EXECUTES AFTER ANY POC OR RESET.
                    0008 ;
      (000C)        0009 BAUD:   EQU  0CH              ;BAUD RATE PORT
      (000E)        0010 COMREG: EQU  0EH              ;COMMAND REG. PORT
      (2400)        0011 DEST:   EQU  2400H            ;PROGRAM DESTINATION
      (000F)        0012 INTMSK: EQU  0FH              ;INTERRUPT MASK PORT
      (0C00)        0013 RDOS:   EQU  0C00H            ;RDOS START ADDRESS
                    0014                               ;
0000'               0015 BOOT:   ORG  0                ;POC OR RESET ADDR
```

```
                    0016                                    ;
                    0017 ;
                    0018 ;  INITIALIZE THE 5501
                    0019 ;
0000  3E09          0020            LD    A,00001001B       ;INE & RESET 5501
0002  D30E          0021            OUT   COMREG,A          ;TO COMMAND REG.
0004  3E00          0022            LD    A,00000000B       ;MASK OFF ALL 5501
0006  D30F          0023            OUT   INTMSK,A          ;INTERRUPT SOURCES
0008  3E01          0024            LD    A,00000001B       ;ASSUME TTY--110 BAUD,
000A  D30C          0025            OUT   BAUD,A            ;TWO STOP BITS
                    0026 ;
                    0027 ;  MOVE PROGRAM BELOW TO EXTERNAL RAM
                    0028 ;
000C  211A00        0029            LD    HL,PGMBEG         ;LOAD HL, DE AND BC
000F  110024        0030            LD    DE,DEST           ;IN PREPARATION FOR
0012  010700        0031            LD    BC,PGMEND-PGMBEG  ;A BLOCK MOVE
0015  EDB0          0032            LDIR                    ;BLOCK MOVE PROGRAM
                    0033 ;
                    0034 ;  JUMP TO AND EXECUTE OFF-BOARD PROGRAM
                    0035 ;
0017  C30024        0036            JP    DEST
                    0037 ;
                    0038 ;  THIS IS THE PROGRAM WHICH IS MOVED TO,
                    0039 ;  THEN EXECUTED IN OFF-BOARD RAM
                    0040 ;
001A  3E80          0041 PGMBEG:    LD    A,10000000B       ;SET MEMORY DSBL BIT
001C  D30A          0042            OUT   0AH,A             ;SEND TO PORT OUT 0AH
001E  C3000C        0043            JP    RDOS              ;JUMP TO RDOS
      (0021)        0044 PGMEND:    EQU   $
```

The object code is used to program the first 21H bytes of a 2716 EPROM, and the EPROM is inserted in SCC socket ROM 0. Since the program begins at address 0000H, it is automatically executed after any POC or RESET. After execution, all SCC memory is disabled, and it remains disabled until bit D7 of port OUT 0AH is reset to logic 0. Note that external memory may overlap SCC memory during execution of this bootstrap program as conflicts are prevented by the SCC circuitry. The program is exited by a jump to RDOS. The user is then expected to make several 'Return' keystrokes which enables RDOS to determine the terminal baud rate, and RDOS replies with the message, 'CROMEMCO RDOS1', followed by a

semi-colon prompt for an RDOS command. At this point, the user would typically load the CDOS system from disk using the 'B' (for Boot) RDOS command. Refer to the Cromemco Disk Operating System User's Manual, part number 023-0036, for full details on this, and other CDOS commands and features.

Section 4

Theory of Operation

This section describes the SCC internal logic and the SCC S-100 bus interface. Please refer to the SCC Block Diagram, Figure 1, and to the SCC Schematic Diagram while reading this section.

4.1 SCC FUNCTIONAL DESCRIPTION

The major SCC internal subsystems are topically grouped and discussed in the following paragraphs.

POWER SUPPLIES: The unregulated +8 VDC, +18 VDC and -18 VDC power lines from the S-100 bus are converted to regulated voltages of +5 VDC @ 1.75 Amps (max), +12 VDC @ 100 mAmps (max) and -5 VDC @ 50 mAmps (max) by IC41, IC42, and IC17 respectively.

CLOCKS: An 8.000 MHz crystal controls internal SCC timing functions. Derived signal IC32 pin 9 supplies a 4.000 MHz square wave to the Z80A and to S-100 bus line ϕ, while its complement at IC32 pin 8 provides a clock reference to other SCC devices. IC12 and its associated circuitry, use this signal to supply a 2.000 MHz, two phase clock to the 5501 at its ϕ_1 and ϕ_2 clock inputs, and to also supply a stable 2.000 MHz square wave to S-100 bus line CLOCK.

DATA BUS INTERFACE: The Z80A data bus lines D0 - D7 drive the S-100 Data Out (DO) bus lines DO0 - DO7 thru tri-state driver IC49, and receive data from the S-100 Data In (DI) bus thru receiver

IC50. Driver IC49 may be disabled during DMA by forcing S-100 bus line $\overline{\text{DODSB}}$ active low. IC11 pin 3 automatically disables the DO driver whenever the DI receiver IC50 is enabled, thus the eight DO bus lines may be connected to their corresponding DI lines to form a single bi-directional data bus, if desired. The DI receiver is controlled by IC28 pin 8, which enables the DI receiver during memory reads to addresses in the range 2400H thru FFFFH (external memory), during I/O reads to port numbers not assigned to the SCC, and during INTA cycles which are granted to devices other than the SCC's 5501.

ADDRESS BUS INTERFACE: The Z80A address lines A0 - A15 drive S-100 bus lines A0 - A15 thru buffer/drivers IC47 and IC48. These devices may be tri-stated during DMA by forcing S-100 bus line $\overline{\text{ADSB}}$ active low. The ten low order lines A0 - A9 are pulled-up to +5 VDC thru a resistor network to meet a memory timing requirement of the 4045 RAM chip. The Z80A address lines also drive SCC on-board RAM and EPROM memory chips, plus memory chip select and I/O function address decoding circuitry. These topics are covered in later paragraphs.

Z80A CONTROL OUTPUT LINES: The Z80A's system control, CPU control and bus control output lines (e.g., $\overline{\text{WR}}$, $\overline{\text{HALT}}$, $\overline{\text{BUSAK}}$) are logically combined to generate S-100 bus 8080-like control signals. Some of these S-100 bus signals are simply inverted Z80A signals; sM1, sHLTA, and pHLDA are derived from $\overline{\text{M1}}$, $\overline{\text{HALT}}$ and $\overline{\text{BUSAK}}$ respectively. S-100 bus lines $\overline{\text{MREQ}}$ and $\overline{\text{RFSH}}$ are buffered versions of Z80A output lines of the same name. The remaining S-100 bus control lines are

derived from Z80A outputs as follows:

$$sINP = \overline{\overline{IORQ} + \overline{RD}}$$

$$sOUT = \overline{\overline{IORQ} + \overline{WR}}$$

$$s\overline{WO} = \overline{sOUT + (\overline{RFSH} \cdot \overline{RD} \cdot MREQ)}$$

$$sMEMR = \overline{\overline{MREQ} + \overline{RD}}$$

$$sINTA = (\text{External Priority}) \cdot (\overline{\overline{M1} + \overline{IORQ}})$$

$$pDBIN = RD + sINTA$$

$$p\overline{WR} = \overline{WR} \text{ (delayed)}$$

The final signal of this group is pSYNC. This S-100 bus line signals the beginning of a new machine cycle. Line pSYNC is clocked high by a falling edge of either $\overline{MREQ}$ or $\overline{IORQ}$, and it is clocked back low by the next rising edge of clock line ϕ.

In the above group, the leading edge of signal $p\overline{WR}$ is delayed so that signal sOUT has time to propagate and stabilize before the pWR low pulse is used as a data strobe. All of these lines are connected to the S-100 bus thru tri-state drivers IC51 and IC52. Thus, they may all be disabled during DMA by forcing S-100 bus lines $\overline{SDSB}$ and $\overline{CDSB}$ active low.

Z80A CONTROL INPUT LINES: Two of the Z80A control inputs, $\overline{NMI}$ and $\overline{BUSRQ}$, are buffered versions of S-100 bus lines $\overline{NMI}$ and $\overline{HOLD}$, respectively. Z80A pin $\overline{RESET}$ is forced active low by any one of three sources: an internal POC (which also drives S-100 bus line $\overline{POC}$),an S-100 bus line $\overline{RESET}$, and SCC connector J3 pin 13. The Z80A maskable $\overline{INT}$ pin is forced active low by either a low on S-100 bus line $\overline{INT}$, or by a high from the 5501 interrupt request pin

INT.

The Z80A $\overline{\text{WAIT}}$ input is controlled by several different sources. A low level on either S-100 bus line RDY or line XRDY forces input $\overline{\text{WAIT}}$ active low, and thus causes the Z80A to 'idle' one or more wait states of 250 nSec each. A short low pulse on either RDY or XRDY latches $\overline{\text{WAIT}}$ active low in IC6 and IC7 until clock line ϕ goes low. This allows a single wait state to be generated by inverting S-100 bus line pSYNC and feeding it to either RDY or XRDY. Multiple wait states may be generated by holding RDY or XRDY low, thereby defeating the latch reset function. An on-board PROM memory read cycle automatically generates a single wait state. IC5 pin 6 forces $\overline{\text{WAIT}}$ active low, thereby generating one wait state, between the falling edge of MREQ and the falling edge of clock ϕ, if an on-board EPROM socket is accessed. If an 'M1 WAITS ONLY' jumper is inserted, the wait state will be inserted only during M1 cycles (opcode fetch) to SCC EPROM, and not during other SCC EPROM memory read cycles. The 5501 also generates wait states when it is accessed thru IC22 pin 12. Finally, SCC connectors J1 pin 11 and J2 pin 11 drive $\overline{\text{WAIT}}$ thru IC22 to synchronize the Z80A to slower peripheral devices.

MEMORY SELECT CIRCUITRY: Memory chip select strobes are generated by IC33, a one-of-ten decoder. Inputs A, B and C are driven by A11, A12 and A13, respectively, and are used to select 2 KBytes blocks of memory with the $\overline{0}$, $\overline{1}$, $\overline{2}$, $\overline{3}$ and $\overline{4}$ outputs. Input D is used as an active low enable signal. IC24 pin 6 decodes the enable when A14 and A15 are low, $\overline{\text{M1}}$ or $\overline{\text{MREQ}}$ is active, $\overline{\text{IORQ}}$ is not

active, and the SCC MEMORY DISABLE option is not active. The $\overline{0}$, $\overline{1}$, $\overline{2}$ and $\overline{3}$ outputs of IC33 select SCC sockets ROM 0 (IC46), ROM 1 (IC45), ROM 2 (IC44) and ROM 3 (IC43), respectively. IC23 combines output $\overline{4}$ with A10 to select the 1 KByte of SCC RAM memory, IC34 and IC35. Another section of IC23 detects all EPROM references and drives the Z80A $\overline{WAIT}$ input thru IC10 pin 2. IC30 pin 11 goes low when either SCC EPROM or RAM are accessed, and this level is used to control the SCC Data In bus receiver, IC50, thru control logic IC28. When the SCC MEMORY DISABLE trace is cut, then all SCC on-board memory is disabled by a logic 1 port OUT 0AH bit D7, which disables all chip selects from IC24. Note that SCC RAM selection occurs slightly later than the RAM read/write strobes. This allows 'early write' cycles in which the RAM Data Out drivers are inhibited.

I/O SELECT CIRCUITRY: I/O select strobes are generated by IC25, a 32 x 8 custom PROM. The A, B, C and D inputs are driven by address lines A0, A1, A2 and A3 respectively, and the E input is driven by Z80A control line $\overline{WR}$. The $\overline{CS}$ input is driven active low by IC40 and IC24 when A4 - A7 are low, $\overline{IORQ}$ is active, and $\overline{M1}$ is inactive. Output Y1 is an active high strobe for I/O port 0BH, and Y2 is an active high strobe for I/O port 0AH. Y3 is an active low enable for the 5501. Y4 - Y7 select a 5501 function by driving its A0 - A3 address lines as follows:

Y4	Y5	Y6	Y7	5501 FUNCTION
0	0	0	0	RECEIVER DATA
0	0	0	1	PARALLEL INPUT DATA
0	0	1	0	INTERRUPT ADDRESS
0	0	1	1	SERIAL STATUS
0	1	0	0	COMMAND REGISTER
0	1	0	1	BAUD RATE
0	1	1	0	XMITTER DATA
0	1	1	1	PARALLEL OUTPUT DATA
1	0	0	0	INTERRUPT MASK
1	0	0	1	TIMER 1
1	0	1	0	TIMER 2
1	0	1	1	TIMER 3
1	1	0	0	TIMER 4
1	1	0	1	TIMER 5
1	1	1	0	NO FUNCTION
1	1	1	1	NO FUNCTION

IC25 output line Y8 goes low when an on-board I/O function is selected, and it is used to disable the DI bus receiver, IC50, thru logic IC28. The port OUT 0AH write strobe is generated by IC7 pin 6 from signals $\overline{WR}$, $\overline{IORQ}$ and Y2. Data is loaded into IC2 on the leading edge of this strobe. The trailing edge of the strobe may be used by a peripheral device to load the data into another latch or FIFO as required. The port IN 0AH read strobe is generated by IC7 pin 6 from signals $\overline{RD}$, $\overline{IORQ}$ and Y2. This signal may be tied to GATE DATA to ensure input data is stable during the read cycle. Ports OUT 0BH and IN 0BH operate in a similar fashion.

THE 5501 INTERFACE: The 5501 requires an 8080-like set of signals for proper operation. Thus, the SCC generates a two phase 12 volt clock, a SYNC pulse, and multiplexes status information to feed the 5501 data bus. Furthermore, the SCC synchronizes its I/O timing to that of the slower 5501 by use of the Z80A $\overline{WAIT}$ line.

The 2.000 MHz two phase clock frequency is derived from the 4.000 MHz clock line IC32 pin 8 at IC12 pins 8 and 9. IC12 pin 9 drives the 5501 ϕ_2 input thru inverter IC27 pin 6 with active pull-up to +12 volts thru Q4. Input ϕ_1 is generated when ϕ_2 and the 4 MHz input clock are low at IC30 pin 6.

The 5501 SYNC input pulse is triggered by the falling edge of ϕ if IORQ is active. SYNC falls at the rising edge of the next ϕ clock. The IC9 latch prevents multiple SYNC pulses during one IORQ. During SYNC, 5501 data bus lines D0 and D1 are driven by open collector NAND gates IC21 pins 6 and 3 to generate 8080-like status bits. D0 is pulled low if the INTA signal from IC11 pin 6 is inactive. D1 is pulled low if the Z80A forces its control line $\overline{WR}$ active low. Bus conflicts are prevented by IC9 pin 6 which turns off IC20 during SYNC.

The 5501 CE input is driven by Y3 from IC25. If CE is driven high, the 5501 proceeds to perform a read or write operation as specified by status bits D0 and D1 during SYNC, and by the address information at inputs A0 - A5 which are defined by IC25 outputs Y4 - Y7. If the CE input is forced low, no 5501 operation takes place, excepting INTA cycles which proceed automatically if D0 is high during SYNC. As a cycle begins and Y3 of IC25 goes low, the Z80A is immediately forced into a wait state by IC11 pin 11. The Z80A then waits for IC11 pin 13 to go low at the end of SYNC, which in turn releases the Z80A $\overline{WAIT}$ line. This insures that the 5501 and the Z80A are synchronized during I/O cycles. The same events occur during INTA, except that the Z80A $\overline{WAIT}$ is initiated

by IC11 pin 6 instead of by Y3 from IC25.

When the 5501 SERIAL STATUS port is read, IC5 pin 8 goes low enabling IC8 to place 5501 D3 data on the D6 line, and D4 data on the D7 line. Also, IC23 pin 11 does high disabling D6 and D7 data from the 5501. These bits are buffer flags for the UART, and are assigned according to the Cromemco convention D7 = TBE and D6 = RDA. The 5501 parallel output port is buffered by IC13, and the parallel input port is buffered by IC14. The output buffer may be disabled by forcing the OUTPUT ENABLE pin of SCC connector J3 active high. The 5501 SENS input is driven thru inverter IC10 pin 13 from the $\overline{\text{INT}}$ pin of SCC connector J3, and may optionally serve as the lowest priority 5501 interrupt source instead of TIMER 5. Serial output data from the 5501 is converted from TTL levels to RS-232/ 20 mA current loop by transistors Q1 and Q2, while serial input data is converted from RS-232/20 mA current levels to TTL levels by transistor Q3 and associated circuitry.

INTERRUPT PRIORITY: The SCC includes daisy chain interrupt priority circuitry which is design compatible with all other Cromemco products which support this feature. During an INTA cycle, if $\overline{\text{PRIORITY IN}}$ is forced low by some other higher priority card, the SCC $\overline{\text{PIORITY OUT}}$ line automatically is forced low thereby disabling all lower priority cards in the daisy chain. The SCC then disables any interrupt acknowledge response from its own 5501, and examines the S-100 bus for a response. During an INTA cycle, if the SCC 5501 has issued an interrupt request and no higher priority board has forced the $\overline{\text{SCC PRIORITY}}$ IN line low,

then the SCC $\overline{\text{PRIORITY OUT}}$ line goes low, and the 5501 controlled data bus is examined for an interrupt response.

4.2: THE SCC S-100 BUS

Physically, the S-100 bus is realized as a set of 100-contact edge connectors mounted to a common mother board and wired in parallel. The modules that plug into the edge connectors of the S-100 bus are circuit cards that measure 5" by 10" (see Figure 29).

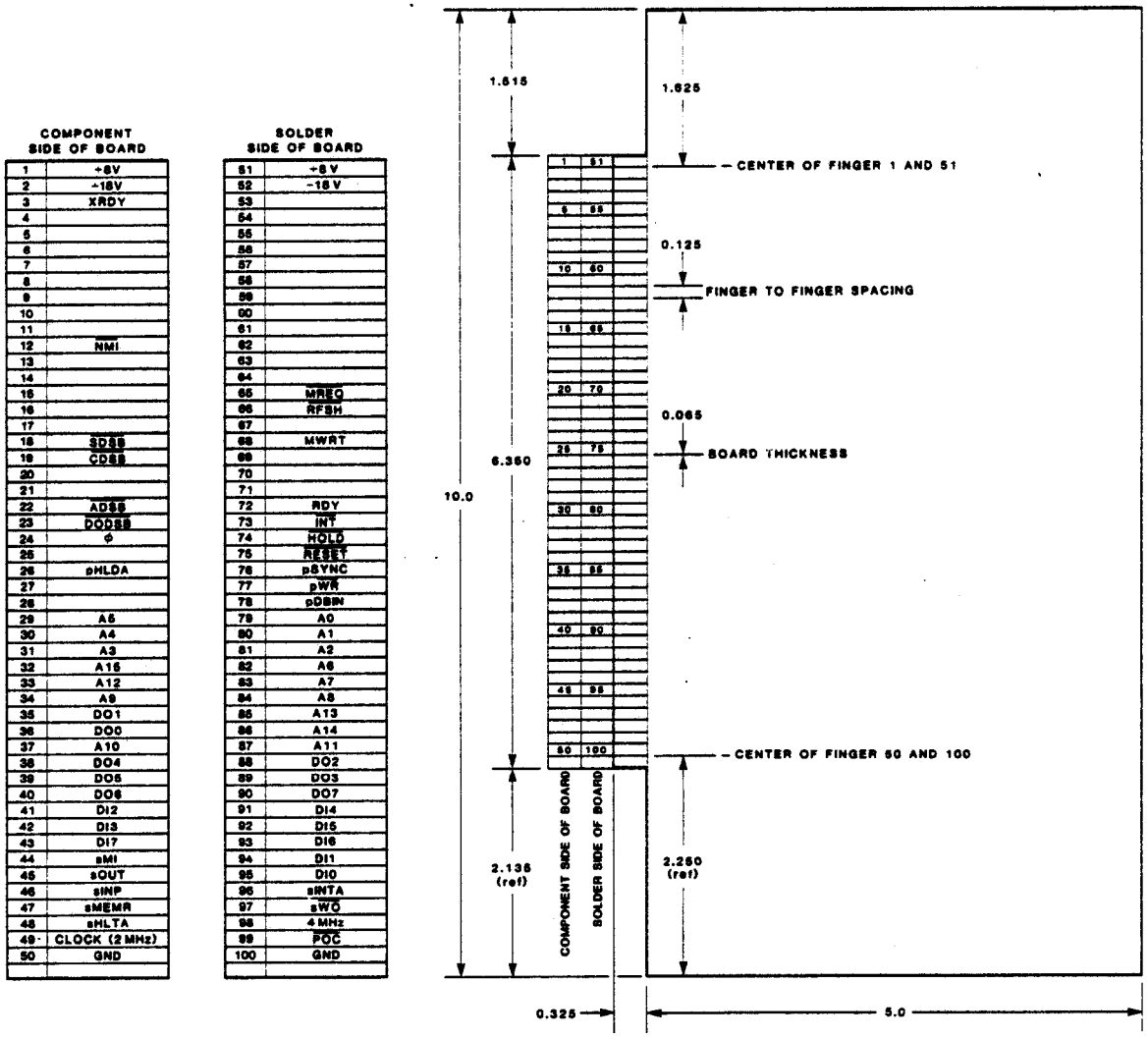


Figure 29: S-100 BUS CARD DIMENSIONS

The S-100 bus was originally designed for use with a CPU module using the 8080 microprocessor, and consequently, the bus signal definitions closely follow those of an 8080 system. The Z80A microprocessor control lines differ quite dramatically from the 8080 lines, but the SCC card is designed to supply the important '8080-like' S-100 bus functions. The S-100 bus line functions supported by the SCC guarantee the card's compatibility with the entire Cromemco S-100 bus product line.

A complete listing of the S-100 bus signals provided by the SCC is shown below. Table entry '---' indicates that the SCC is not connected to the corresponding S-100 bus pin number.

THE SCC S-100 BUS

1.	+8 VOLTS	26.	pHLDA	51.	+8 VDC	76.	pSYNC
2.	+18 VOLTS	27.	---	52.	-18 VDC	77.	p$\overline{WR}$
3.	XRDY	28.	---	53.	---	78.	pDBIN
4.	---	29.	A5	54.	---	79.	A0
5.	---	30.	A4	55.	---	80.	A1
6.	---	31.	A3	56.	---	81.	A2
7.	---	32.	A15	57.	---	82.	A6
8.	---	33.	A12	58.	---	83.	A7
9.	---	34.	A9	59.	---	84.	A8
10.	---	35.	DO1	60.	---	85.	A13
11.	---	36.	DO0	61.	---	86.	A14
12.	$\overline{NMI}$	37.	A10	62.	---	87.	A11
13.	---	38.	DO4	63.	---	88.	DO2
14.	---	39.	DO5	64.	---	89.	DO3
15.	---	40.	DO6	65.	$\overline{MREQ}$	90.	DO7
16.	---	41.	DI2	66.	$\overline{RFSH}$	91.	DI4
17.	---	42.	DI3	67.	---	92.	DI5
18.	$\overline{SDSB}$	43.	DI7	68.	MWRT	93.	DI6
19.	$\overline{CDSB}$	44.	sM1	69.	---	94.	DI1
20.	---	45.	sOUT	70.	---	95.	DI0
21.	---	46.	sINP	71.	---	96.	sINTA
22.	$\overline{ADSB}$	47.	sMEMR	72.	$\overline{RDY}$	97.	s$\overline{WO}$
23.	$\overline{DODSB}$	48.	sHLTA	73.	$\overline{INT}$	98.	4MHZ
14.	ϕ	49.	CLOCK (2MHZ)	74.	$\overline{HOLD}$	99.	$\overline{POC}$
25.	---	50.	GND	75.	$\overline{RESET}$	100.	GND

The signals of the S-100 bus can be grouped in four functional categories: 1) power supply, 2) address, 3) data, and 4) clock and control signals.

S-100 POWER SUPPLY

+8 Volts	Pins 1 and 51
+18 Volts	Pin 2
-18 Volts	Pin 52
Ground	Pins 50 and 100

Three unregulated D.C. power supply voltages appear on the S-100 bus: +8 volts, +18 volts and -18 volts. The main power supplies are unregulated, so power supply regulation must be performed on each individual circuit card, usually by three terminal regulator IC's.

Distributed power supply regulation has several advantages over a single, centrally regulated supply:

-Each card is individually protected from voltage overload. Faulty regulation in one master supply cannot destroy the entire computer system.

-The heat produced by voltage regulation is thermally distributed through a larger physical volume.

-Voltage drops along the bus do not influence the voltage on the card circuitry itself.

-Initial cost of the computer mainframe is lower. Regulation circuitry is purchased only as additional cards are added to the system.

An S-100 bus mainframe capable of accepting a full 21 cards (like the Cromemco Z-2, Z-2D and SYSTEM THREE) typically has a power supply current capacity of 30 amps at +8 volts and 15 amps at +18 and -18 volts.

S-100 ADDRESS SIGNALS

A0	PIN 79	A8	PIN 84
A1	PIN 80	A9	PIN 34
A2	PIN 81	A10	PIN 37
A3	PIN 31	A11	PIN 87
A4	PIN 30	A12	PIN 33
A5	PIN 29	A13	PIN 85
A6	PIN 82	A14	PIN 86
A7	PIN 83	A15	PIN 32

There are 16 address lines on the S-100 bus allowing the direct addressing of 65,536 words of memory space. Tri-state TTL drivers are used to drive the address bus. S-100 bus control line $\overline{\text{ADSB}}$ can be used to disable the address drivers to allow DMA operations when other cards need to take control of the address bus.

S-100 DATA SIGNALS

DI0	PIN 95	DO0	PIN 36
DI1	PIN 94	DO1	PIN 35
DI2	PIN 41	DO2	PIN 88
DI3	PIN 42	DO3	PIN 89
DI4	PIN 91	DO4	PIN 38
DI5	PIN 92	DO5	PIN 39
DI6	PIN 93	DO6	PIN 40
DI7	PIN 43	DO7	PIN 90

Although the S-100 bus is based on the 8080 microprocessor which has an 8-bit bi-directional data bus, the S-100 has two directional data busses, each eight bits wide. The data input bus is called the DI bus, and the data output bus is called the DO bus. The S-100 bus provides control line $\overline{\text{DODSB}}$ to disable the DO

bus (DO DISABLE) for DMA operations.

S-100 CLOCK AND CONTROL SIGNALS

XRDY	PIN 3	MREQ	PIN 65
NMI	PIN 12	RFSH	PIN 66
SDSB	PIN 18	MWRT	PIN 68
CDSB	PIN 19	RDY	PIN 72
ADSB	PIN 22	INT	PIN 73
DODSB	PIN 23	HOLD	PIN 74
ϕ	PIN 24	RESET	PIN 75
pHLDA	PIN 26	pSYNC	PIN 76
sM1	PIN 44	pWR	PIN 77
sOUT	PIN 45	pDBIN	PIN 78
sINP	PIN 46	sINTA	PIN 96
sMEMR	PIN 47	sWO	PIN 97
sHLTA	PIN 48	4MHZ	PIN 98
CLOCK (2 MHZ)	PIN 49	POC	PIN 99

There are two clock signals on the S-100 bus: ϕ (pin 24) and 2 MHz CLOCK (pin 49). The 2 MHz CLOCK line is always a 2 MHz signal regardless of the processor clock frequency. ϕ provides a single phase clock at the processor clock frequency. The rising edge of ϕ defines the beginning of each CPU T-cycle. All clock and control signals on the S-100 bus are standard TTL levels.

Most control signals on the S-100 bus are functionally equivalent to control signals used with the 8080 microprocessor. Those prefixed with a lower case 'p', such as pHLDA, pSYNC, pWR and pDBIN, serve the same function as the corresponding control signals for the 8080 microprocessor. Similarly, S-100 bus signals prefixed with an 's' are functionally equivalent to the corresponding outputs of the 8080 status latch. These signals include sM1, sOUT, sINP, sMEMR, sHLTA, sINTA and sWO. In Cromemco Z80A systems, line 4MHZ is used to indicate 4 MHz operation (logic

1) or 2 MHz operation (logic 0).

Four of the S-100 control lines are dedicated to tri-stating bus drivers (e.g., during DMA operations). $\overline{\text{ADSB}}$ is used to disable the address bus; $\overline{\text{DODSB}}$ is used to disable the Data Output bus; $\overline{\text{SDSB}}$ is used to disable the status lines (those prefixed with an 's'); and $\overline{\text{CDSB}}$ is used to disable the clock and control signals.

Three of the S-100 control signals shown are used primarily with the Z80A CPU. These are $\overline{\text{NMI}}$ (Non-Maskable Interrupt), $\overline{\text{MREQ}}$, and $\overline{\text{RFSH}}$. The functions of these signals on the S-100 bus are the same as the corresponding lines of the Z80A microprocessor.

The remaining lines are used primarily in S-100 systems with an operator's front panel. MWRT (pin 68) is used to indicate a memory write operation and is used in conjunction with front panel memory deposit. XRDY is an alternate to RDY to avoid bus conflicts when both front panel circuitry and other circuitry need control of the processor $\overline{\text{WAIT}}$ line.

Finally, there is the $\overline{\text{POC}}$ (Power-On Clear) signal that remains at logic 0 when power is first turned on, and then transitions to logic 1 approximately 100 milliseconds later to indicate that power is on and the power supply voltages have stabilized.

4.3 SCC S-100 BUS WAVEFORMS

The following figures show the behavior of the SCC controlled S-100 bus during each possible type of CPU cycle. These figures are patterned after those appearing in the Z80A Technical Manual which is included with the SCC documentation. Waveforms for an opcode fetch cycle are shown in Figures 30 and 31; memory read and write cycles are shown in Figures 32 and 33; I/O cycles are shown in Figures 34 and 35. Figure 36 shows a hold request and a hold acknowledge sequence. Figure 37 shows an interrupt request followed by an interrupt acknowledge cycle. Figure 38 shows the response to an NMI, and Figure 39 shows an interrupt-driven exit from a Z80A halted state.

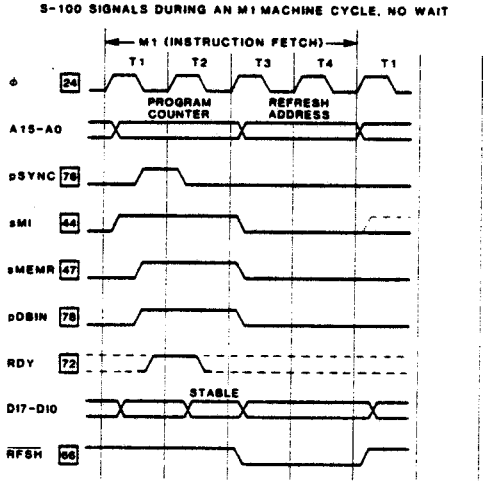


Figure 30

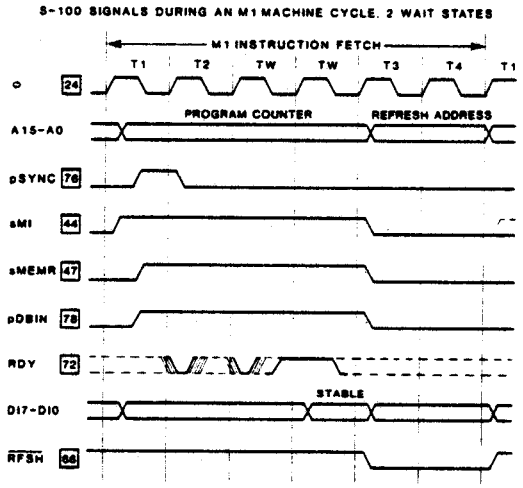


Figure 31

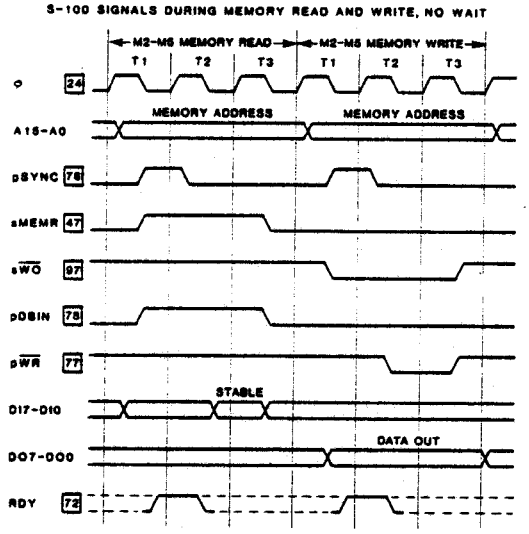


Figure 32

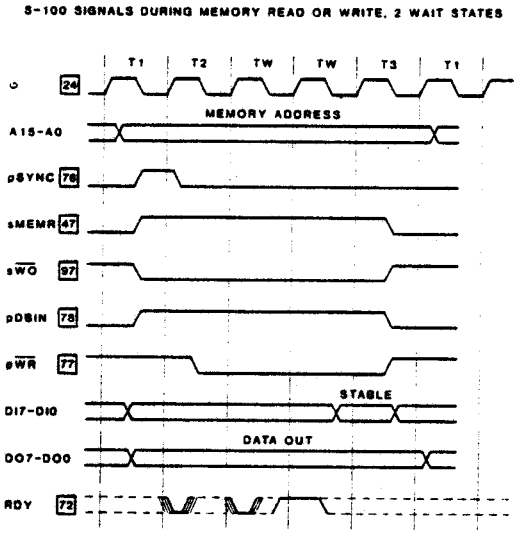


Figure 33

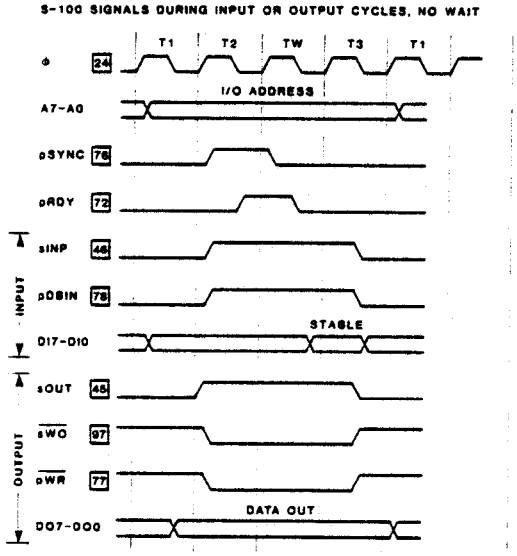


Figure 34

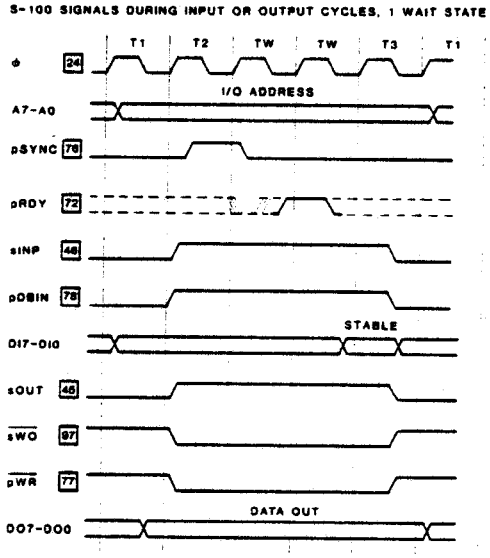


Figure 35

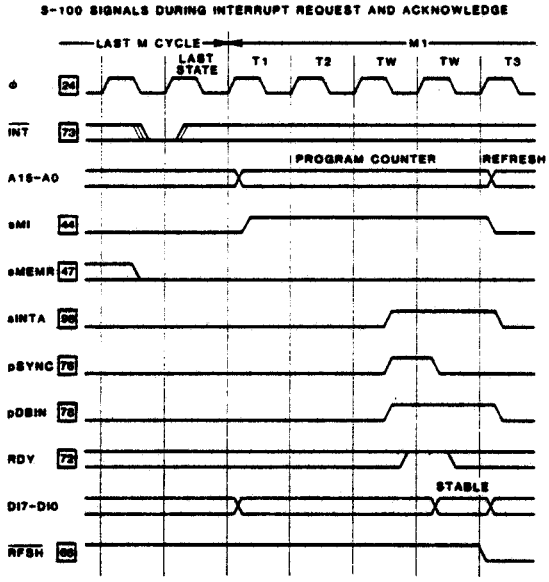


Figure 36

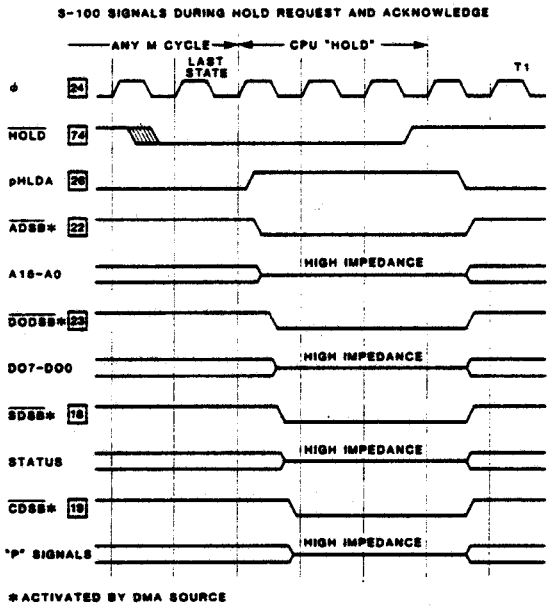


Figure 37

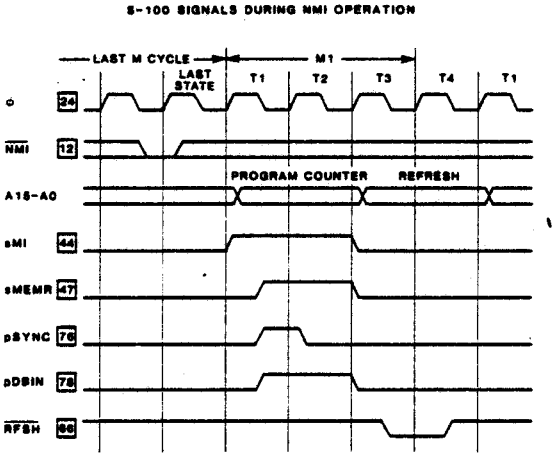


Figure 38

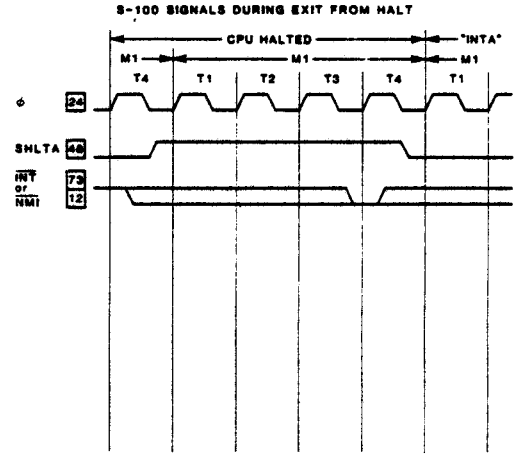


Figure 39

Section 5

A Simple SCC Example System

This section brings many of the topics discussed in previous sections together in a concrete SCC example system. The example system is simple, yet complete. The system hardware consists of a powered S-100 bus, an SCC card, a TTY, an external parallel data input source, and a RESET pushbutton switch. The system firmware consists of a model MCB-216 ROM chip set, and a 2716 programmed with an executive Control Basic routine and three short Z80 assembly language routines. The example system services interrupt requests from one of the SCC interval timers, and also from the TTY keyboard via source RDA.

In overview, the system monitors a parallel 8-bit input data source connected to SCC port IN 0AH at connector J1. The system samples parallel data byte X approximately once every 60 seconds. Value Y is then determined from sample X by the linear relationship Y = A*X + B, where A = +1.2 and B = -10.8 by assumption. The value Y is then formatted and output to the TTY printer. In addition, if any key is pressed on the TTY keyboard, a sample X is to be immediately taken, and the corresponding function value Y is output to the TTY printer.

A schematic/block diagram of the system hardware is shown in Figure 40. The system firmware uses the Control Basic AUTORUN command, thus program execution begins automatically after power is applied, or after S-100 bus line $\overline{\text{RESET}}$ is forced active low.

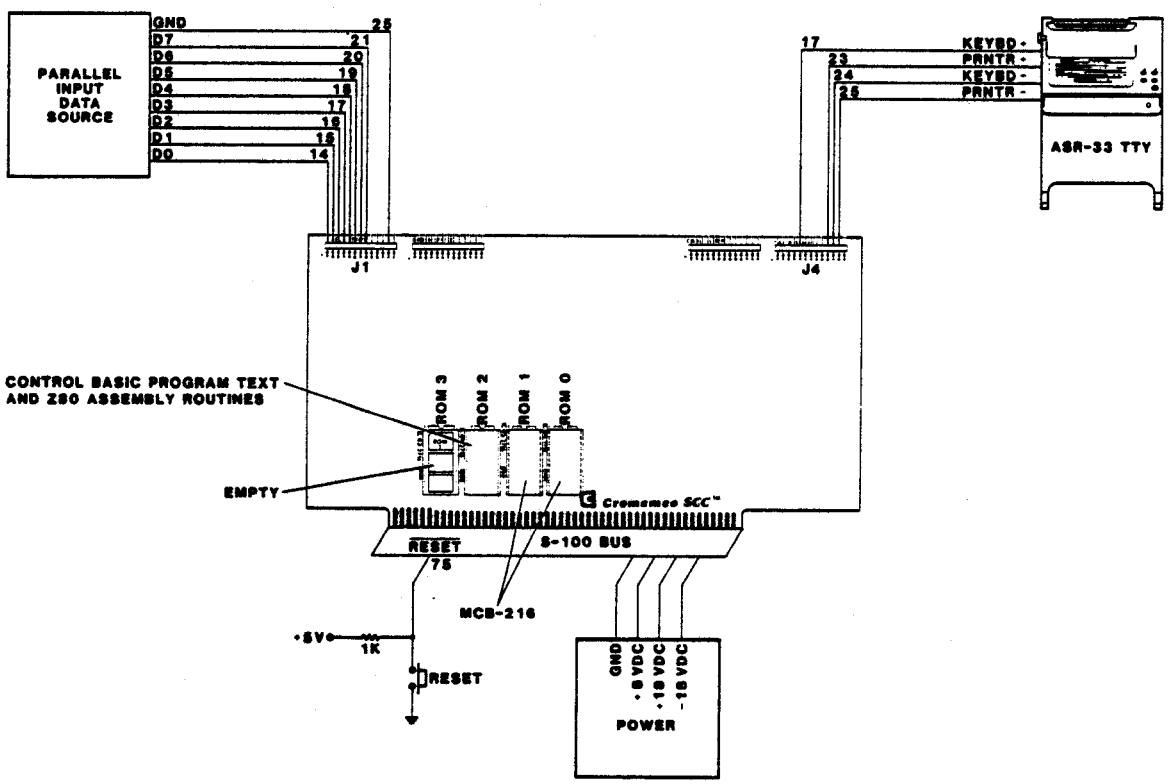


Figure 40: EXAMPLE SYSTEM HARDWARE

Two Control Basic features should be discussed before the system 3CB text is presented. The first concerns the AUTORUN command. If the first line in a Basic program stored on page 10H is the AUTORUN command, then the program on page 10H will run automatically after any RESET or POC, and the serial baud rate will also automatically be set to 9600 baud. The second feature concerns interrupt servicing. When operating in IM 0, the restart service routine starting addresses are 00H, 08H, 10H, 18H, 20H, 28H, 30H and 38H which correspond to RST 00H thru RST 38H. This address range is occupied by one of the MCB-216 ROMs, however, whose contents may not be altered by the user. Consequently, model MCB-216 ROMs have been programmed at each RST service address with JP instructions to fixed locations in SCC socket ROM

2. The programmed jumps are tabulated below:

RESTART ADDRESS	SCC ROM 2 JUMP ADDRESS
RST 00H (TIMER 1)	NOT AVAILABLE
RST 08H (TIMER 2)	JP 10F2H
RST 10H (INT)	JP 10F4H
RST 18H (TIMER 3)	JP 10F6H
RST 20H (RDA)	JP 10F8H
RST 28H (TBE)	JP 10FAH
RST 30H (TIMER 4)	NOT AVAILABLE
RST 38H (TIMER 5/PI7)	JP 10FEH

Note that interrupt source TIMER 1 cannot be used because address 0000H is required for an initial power-on jump to the Control Basic cold start location, and TIMER 4 cannot be used because address 0030H is used as a break-point trap by the SCC Monitior program. The other interrupt vectors jump to page 10H addresses which are separated by only two bytes. It is then the responsibility of the programmer to insert two byte relative jump instructions at these locations.

The system Control Basic text (shown below) consists of two segments. The first segment is a short program loaded on page 10H which issues the AUTORUN command, and then RUNs the other segment which is loaded on page 11H. This second segment is the Control Basic system executive routine. The executive routine is loaded on page 11H instead of page 10H because there is not sufficient memory space between the end of the page 10H Basic program and the first interrupt vector address (10F2H) to contain it. Both fully annotated and compacted versions of each program segment are presented below, along with their corresponding memory storage

requirements. Note in the abbreviated program texts, that a large line number occupies the same memory space as a small one, except when it is used as an argument of a Basic statment, and that spaces appearing after a line number but before the statement are not stored.

PAGE 10H CONTROL BASIC TEXT (26 BYTES)

```
100 AUTORUN
200 RUN %11
```

COMPACTED VERSION OF THE SAME PROGRAM (23 BYTES)

```
100 AUTORUN;RUN%11
```

PAGE 11H CONTROL BASIC TEXT (880 BYTES)

```
100 REM
110 REM    >>>  SYSTEM EXECUTIVE ROUTINE  <<<
120 REM
130 REM
140 REM :: CALL ASSY INITIALIZATION ROUTINE
150 CALL %1080
160 REM :: INITIALIZE 3CB VARIABLES; M (MINUTE COUNT),
170 REM :: T (COUNTDOWNS RAM ADDR), K (KEYDOWN FLAG ADDR)
180 M=0
190 T=%23F0
200 K=%23F2
210 REM :: CHECK KEYDOWN FLAG (NON-ZERO -> TTY KEY PRESSED)
220 IF GET(K)=0 GOTO 260
230 REM :: TTY KEY PRESSED.  RESET FLAG & PRINT READING.
240 PUT(K)=0
250 GOSUB 1000
260 REM :: CHECK TIME COUNT (COUNTDOWNS = 6000 -> MINUTE UP)
270 IF (GET(T)+%100*GET(T+1))<6000 GOTO 210
280 REM :: MINUTE IS UP.  RESET TIME COUNT, UPDATE AND
290 REM :: PRINT MINUTE COUNT, PRINT READING.
300 PUT(T)=0,0
310 M=M+1
320 IF (M=32767) M=0
330 PRINT 'MINUTE:',#5,M,'    ',
340 GOSUB 1000
350 REM :: LOOP BACK TO CHECK KEYDOWN
360 GOTO 220
1000 REM
1010 REM :: SUBROUTINE SCALE, FORMAT & PRINT READING
1020 REM
```

```
1030 X=IN(%0A)
1040 Y=12*X-108
1050 PRINT 'READING:',#5,Y/10,'.',#1,ABS(Y-10*(Y/10))
1060 RETURN
```

COMPACTED VERSION OF THE SAME PROGRAM (227 BYTES)

```
1 C.%1080;M=0;T=%23F0;K=T+2
2 IF G.(K)#0 PUT(K)=0;GOS.5
3 IF (G.(T)+G.(T+1)*%100)<6000 G.2
4 PUT(T)=0,0;M=(M+1)*(M#32766);P.'MINUTE:',#5,M,'    ',;GOS.5;G.2
5 Y=12*IN(%A)-108;P.'READING:',#5,Y/10,'.',#1,ABS(Y-10*(Y/10));R.
```

The first Control Basic program segment is self explanatory. In the second segment, the executive routine begins by calling an assembly language routine with a starting address of 1080H which initializes the SCC system. A full description of this routine appears later along with its listing. Then variable M, used to count the number of elapsed minutes, is initialized to zero. Variables T and K are then equated to SCC RAM address locations to save program text space, since they are referenced more than once later in the text. RAM locations T, T+1 and K are used to pass data between the two interrupt service routines and the Control Basic program. The RAM memory area 21FFH thru 23FFH is reserved for Control Basic text editing and array @(I) storage, but since the system Basic program text is stored in EPROM, and array @(I) is not used by the program, this entire RAM area may be used as scratchpad or for data passing. Locations T and T+1 hold a two byte 'COUNTDOWNS' value in low-high Intel style, which represents a whole number of elapsed 10 millisecond intervals. This number is incremented by the TIMER 2 service routine once every 10 milliseconds. Location K holds a one byte 'KEYDOWN' flag which is set to a non-zero value by the RDA service routine whenever any

TTY key is pressed.

After these initialization tasks are completed, the executive routine then checks the 'KEYDOWN' flag, and if a TTY key has been pressed, the flag is reset to 00H by a PUT statement, and a Basic subroutine is called which reads the parallel input data, scales it, and prints it to the TTY. Locations T and T+1 are then sampled to determine if one minute has elapsed, which corresponds to 6000 'COUNTDOWNS' of 10 milliseconds each. Note that the GET function evaluates to a one byte value, thus contents (T) and (T+1) must be scaled and added together to determine the actual two byte 'COUNTDOWNS' value. If one minute has elapsed, then locations T and T+1 are zeroed by a PUT statement, and the minute count M is incremented. The Control Basic programmer must guard against expressions which exceed 32767 in absolute value, as program execution will terminate with the error message 'HOW?' in such cases. Thus, in this program M is reset to zero when it reaches the increment statement with a current value of 32766. The current minute count is then formatted and printed to the TTY, followed by a call to the subroutine which prints out the scaled reading value.

This subroutine begins by reading SCC input port IN 0AH. The read value, X, may vary from 0 to 255 decimal, and thus the value Y = 1.2*X - 10.8 would vary from -10.8 to 295.2 in steps of 1.2. To calculate these decimal values using integer arithmetic, a value of Y ten times larger than the desired Y value is first calculated. Since this value must lie between -108 and 2952, no

overflow is possible. The whole number part is then obtained by computing Y mod 10 (no rounding). The decimal point is printed out as a separate ASCII character, and the decimal part is then found by calculating Y minus ten times (Y mod 10). This digit is then printed out 'close packed' to the period character to create a standard form decimal number.

The system Z80 assembly language listings appear below. The assembly routines consist of a system initialization subroutine, a TIMER 2 interrupt service routine, an RDA interrupt service routine, and two programmed relative jump instructions. The assembly object code resides at 1080H thru 10B6H (54 bytes), the SCC ROM 2 page 10H area lying between the short AUTORUN Control Basic program and the RST service jump area 10F0H thru 10FFH.

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15

                        0001 ;
                        0002 ;    EXAMPLE SYSTEM Z80 ASSEMBLY ROUTINES STORE ON
                        0003 ;  PAGE 10H IN SCC SOCKET ROM 1.  THE INITIALIZATION
                        0004 ;  ROUTINE IS CALLED BY THE 3CB EXECUTIVE.  THE
                        0005 ;  TIMER 2 INTERRUPT SERVICE ROUTINE IS REACHED BY
                        0006 ;  A RST 08H WHICH JUMPS TO ROM 1 ADDRESS 10F2H.
                        0007 ;  THIS ROUTINE PROGRAMS A RELATIVE JUMP TO THE
                        0008 ;  TIMER 2 SERVICE STARTING ADDRESS AT THIS LOCATION.
                        0009 ;  LIKEWISE, RDA IS REACHED BY A RST 20H, A JUMP TO
                        0010 ;  10F8H, AND A RELATIVE JUMP TO THE RDA SERVICE
                        0011 ;  STARTING ADDRESS.
                        0012 ;
                        0013 ;    NOTE THAT THE ASSEMBLY ROUTINES ARE SANDWICHED
                        0014 ;  BETWEEN THE CONTROL BASIC 'AUTORUN'-'RUN %11'
                        0015 ;  PROGRAM AND THE RELATIVE JUMP INSTRUCTIONS ON
                        0016 ;  PAGE 10H.  THE ASSEMBLY LISTING IS ARRANGED SO
                        0017 ;  THAT ALL PAGE 10H BYTES OCCUPIED BY ASSEMBLY
                        0018 ;  OBJECT CODE APPEAR IN SEQUENTIAL ADDRESS ORDER.
                        0019                                    ;
(0000)                  0020 BAUD:   EQU  00H                   ;BAUD RATE PORT
(0002)                  0021 COMREG: EQU  02H                   ;COMMAND REG. PORT
(0003)                  0022 INTMSR: EQU  03H                   ;INTERRUPT MASK PORT
(23F2)                  0023 K:      EQU  23F2H                 ;TTY KEYDWN FLAG ADDR
```

```
                (009C)      0024 TENMS:  EQU  9CH              ;10 MSEC DELAY COUNT
                (23F0)      0025 T:      EQU  23F0H            ;COUNTDOWNS ADDR
                (0006)      0026 TIMER2: EQU  06H              ;TIMER 2 PORT
                            0027                               ;
                            0028 ;
                            0029 ;    SCC SYSTEM INITIALIZATION SUBROUTINE
                            0030 ;
                            0031         ORG  1080H            ;PAGE 10H START ADDR
                            0032                               ;
1080  3E01                  0033         LD   A,00000001B      ;TTY 110 BAUD,
1082  D300                  0034         OUT  BAUD,A           ;TWO STOP BITS
1084  3E09                  0035         LD   A,00001001B      ;INTA ENABLE,
1086  D302                  0036         OUT  COMREG,A         ;RESET 5501
1088  3E12                  0037         LD   A,00010010B      ;UNMASK ONLY SOURCES
108A  D303                  0038         OUT  INTMSK,A         ;RDA AND TIMER 2
108C  210000                0039         LD   HL,0             ;CLEAR HL
108F  22F023                0040         LD   (T),HL           ;ZERO COUNTDOWNS
1092  22F223                0041         LD   (K),HL           ;RESET KEYDWN FLAG
1095  3E9C                  0042         LD   A,TENMS          ;LOAD AND
1097  D306                  0043         OUT  TIMER2,A         ;START TIMER 2
1099  ED46                  0044         IM   0                ;(EMPHASIS ONLY)
109B  FB                    0045         EI                    ;ENABLE Z80 INTERRUPTS
109C  C9                    0046         RET                   ;RETURN TO 3CB EXEC
                            0047 ;
                            0048 ;  TIMER 2 SERVICE ROUTINE
                            0049 ;
                            0050                               ;
109D  F5                    0051 SRVTM2: PUSH AF               ;SAVE AF ON STACK
109E  E5                    0052         PUSH HL               ;SAVE HL ON STACK
109F  3E9C                  0053         LD   A,TENMS          ;LOAD AND START
10A1  D306                  0054         OUT  TIMER2,A         ;TIMER 2 AGAIN
10A3  2AF023                0055         LD   HL,(T)           ;GET COUNTDOWNS
10A6  23                    0056         INC  HL               ;INCREMENT
10A7  22F023                0057         LD   (T),HL           ;RESTORE
10AA  E1                    0058         POP  HL               ;RESTORE HL
10AB  F1                    0059         POP  AF               ;RESTORE AF
10AC  FB                    0060         EI                    ;RE-ENABLE INTERRUPTS
10AD  C9                    0061         RET                   ;CONTINUE 3CB PROGRAM
                            0062                               ;
                            0063 ;
                            0064 ;  RDA SERVICE ROUTINE
                            0065 ;
                            0066                               ;
10AE  F5                    0067 SRVRDA: PUSH AF               ;SAVE AF ON STACK
10AF  3EFF                  0068         LD   A,-1             ;SET KEYDWN FLAG TRUE
10B1  32F223                0069         LD   (K),A            ;
10B4  F1                    0070         POP  AF               ;RESTORE AF
10B5  FB                    0071         EI                    ;RE-ENABLE INTERRUPTS
10B6  C9                    0072         RET                   ;CONTINUE 3CB PROGRAM
                            0073                               ;
                            0074 ;
                            0075 ;  PROGRAMMED RELATIVE JUMPS
                            0076 ;
```

```
                    0077                    ;
                    0078        ORG  10F2H  ;RST 08H VECTORED HERE
                    0079                    ;
10F2  18A9          0080        JR   SRVTM2 ;JR TO TIMER 2 SERVICE
                    0081                    ;
                    0082        ORG  10F8H  ;RST 20H VECTORED HERE
                    0083                    ;
10F8  18B4          0084        JR   SRVRDA ;JR TO RDA SERVICE
                    0085                    ;
                    0086        END
```

The initialization subroutine is called by the Control Basic executive, and it sets the TTY baud rate and stop bit number, enables the automatic 5501 INTA response with bit INE, and resets the 5501. It masks off all 5501 interrupt sources except RDA and TIMER 2, zeroes the common memory data locations T, T+1 and K, and starts TIMER 2 with its first 10 millisecond DELAY COUNT. Interrupts are then enabled, and the routine returns to the 3CB executive. Notice that register SP is not initialized as this function is automatically performed during the Control Basic cold start procedure.

Every 10 milliseconds TIMER 2 reaches timeout, which interrupts the Control Basic program and causes the TIMER 2 service routine to be executed. The service routine re-starts TIMER 2 on a new 10 millisecond interval, and increments the two byte 'COUNTDOWNS' value at RAM addresses T and T+1, re-enables interrupts and RETurns to the Control Basic program, continuing from the point of interruption.

Pressing any key on the TTY keyboard sends a data byte to the 5501 serial receiver over SCC connector J4, and thus generates an RDA condition. The RDA condition then interrupts the Control

Basic program and causes the RDA service routine to be executed. The service routine sets the one byte 'KEYDWN' flag at RAM address to non-zero data 0FFH, re-enables interrupts, and RETurns to the Control Basic program, continuing from the point of interruption.

The last portion of the listing consists of the two relative jump instructions to the TIMER 2 and RDA service routines. Note that if the service routines are displaced from these jump instructions by more than 128 locations, three byte JP instructions would be used instead of the two byte JR instructions.

The Control Basic program text and the Z80 assembly language object code is used to program a 2716 EPROM which is then inserted in SCC socket ROM 2. If it is assumed that a Cromemco 32K Bytesaver card is loaded with an erased 2716 EPROM which spans the address range C000H thru C7FFH (pages 192 thru 199), then the first Control Basic segment text would be entered into the text editing area, and then committed to EPROM firmware with a 'SAVE 192' direct command. The first page (192) of this 2716 EPROM then corresponds to page 10H after down-loading into SCC socket ROM 2. The text editing area would then be cleared with the 'NEW' direct command, and the Control Basic executive program text entered and committed to EPROM with a 'SAVE 193' direct command. The 51 bytes of assembly object code would be assembled by hand, then programmed into EPROM with the SCC Monitor 'SM' (for Substitute Memory) command. To do this, SCC Monitor would be entered from

Control Basic by issuing the 'QUIT' direct command. Example dialogue between the user and SCC Monitor to commit the first eight bytes of object code to EPROM locations C180H thru C187H (which correspond to addresses 1080H thru 1087H when the 2716 EPROM is moved down to SCC socket ROM 2) follows. The characters the user types are underlined.

```
OK
>QUIT <CR>                    (Leave Control Basic)

CROMEMCO SCCMON               (Enter SCC Monitor)
:SM C180 <CR>
C180:  FF.3E  FF.01  FF.D3  FF.00  FF.3E  FF.09  FF.D3  FF.01<CR>
```

Programming a 2716 EPROM in this manner results in the following overall system memory map after the programmed EPROM has been moved down to SCC socket ROM 2:

EXAMPLE SYSTEM MEMORY MAP

LOGICAL ADDRESS	PHYSICAL ADDRESS	CONTENTS
0000H - 0422H	SCC ROM 0	SCC MONITOR PROGRAM
0423H - 0FFFH	SCC ROM 0, ROM 1	CONTROL BASIC INTERPRETER
1000F - 1022H	SCC ROM 2	PAGE 10H BASIC AUTORUN
1080H - 10AAH	SCC ROM 2	ASSEMBLY ROUTINES
10F2H - 10F3H	SCC ROM 2	RST 08H RELATIVE JUMP
10F8H - 10F9H	SCC ROM 2	RST 20H RELATIVE JUMP
1100H - 1321H	SCC ROM 2	PAGE 11H BASIC EXECUTIVE
2000H - 21FEH	SCC RAM	BASIC VARIABLES AND STACK
21FFH - 23FFH	SCC RAM	COMMON RAM MEMORY AREA

Note that the un-programmed ROM 2 bytes contain data 0FFH, and that SCC socket ROM 3 is not used in the example system.

With the system firmware in place and the external parallel

data source attached to SCC connector J1, program execution would begin automatically after power is applied or after the RESET pushbutton switch is pressed. An example of the program output to the TTY printer is shown below. The example output assumes parallel data byte X = 10H is read at minute 1000, X = 20H is read at minute 1001, and that between minute 1000 and 1001 a TTY key is pressed which causes data X = 14H to be immediately sampled.

SAMPLE TTY OUTPUT

```
  :
  :
MINUTE: 1000   READING:    8.4
READING:   13.2
MINUTE: 1001   READING:   27.6
  :
  :
```

APPENDIX A

This appendix codifies all SCC I/O port descriptions for convenient reference. Also note the quick-reference SCC I/O PORT SUMMARY on the manual overleaf.

SERIAL STATUS (IN 00H)

TBE	RDA	IPG	TBE	RDA	SRV	ORE	FME

BIT	LABEL	FUNCTION
D7	TBE	Transmitter Buffer Empty. The Transmitter Buffer is a temporary storage register which buffers the data byte sent to TRANSMITTER DATA (port OUT 01H). A logic 1 TBE indicates that the Transmitter Buffer is ready to accept a new data byte. TBE goes to logic 1 immediately after the buffer contents are transferred to the serial transmitter, thus the new data byte may be loaded into the buffer while transmission of the previous byte is still in progress. TBE is cleared to logic 0 after the Transmitter Buffer is unloaded, and it is set to logic 1 after a 5501 RESET issued thru COMMAND REGISTER bit RES. A TBE condition generates a maskable interrupt request. Note that TBE may still be sampled from the SERIAL STATUS port even though it is masked-off (disabled) as an interrupt source. TBE status is also available at bit D4 (see below).
D6	RDA	Read Data Available. A logic 1 RDA indicates that a data byte is available and may be read from RECEIVER DATA (port IN 01H). If a new data byte has arrived in its entirety before the previous contents of RECEIVER DATA has been read, the new byte will overwrite the old, and the OVERRUN error flag will be set to logic 1. The RDA flag is reset to logic 0 by either inputting RECEIVER DATA by executing an IN A,(01H) instruction, or by a 5501 RESET. An RDA condition generates a maskable interrupt request. Note that RDA may still be sampled from the SERIAL STATUS port even though it is masked-off as an interrupt source. RDA status is also available at bit D3 (see below).

D5 IPG — Interrupt Pending. A logic 1 IPG indicates that one or more of the eight possible prioritized interrupt request sources has become active, and that the corresponding interrupt is enabled. The logical states of bit IPG and 5501 pin INT are identical. IPG is reset to logic 0 after an RST instruction has been gated onto the data bus in response to an INTA (Interrupt Acknowledge) cycle, or after INTERRUPT ADDRESS port IN 03H is read, assuming in both cases that no further enabled interrupt requests are pending.

D4 TBE — Transmitter Buffer Empty (identical to bit D7 described above).

D3 RDA — Read Data Available (identical to bit D6 described above).

D2 SRV — Serial Receive. A logic 1 SRV indicates a high level (spacing) on the serial data input line, and a logic 0 SRV indicates a low level (spacing) on the serial data input line. SRV goes to logic 1 when no data is being received, and is provided for break detection and test purposes.

D1 ORE — Overrun Error. A logic 1 ORE indicates that a new byte has overwritten a previous byte without the previous byte being read. ORE is reset to logic 0 after the SERIAL STATUS port is read, or by a 5501 RESET.

D0 FME — Frame Error. A logic 1 FME indicates an error in one or both stop bits which 'framed' the last received data byte. FME remains at logic 1 until a valid character is received.

BAUD RATE (OUT 00H)

STOP BITS	9600	4800	2400	1200	300	150	110

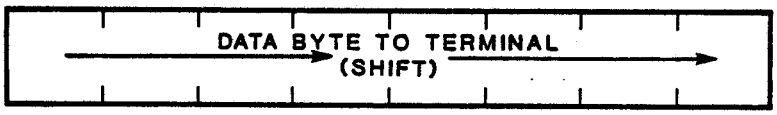
A logic 0 output to bit D7 (STOP BITS) selects two stop bits, a logic 1 to bit D7 selects one stop bit. A logic 1 in any bit position D6 thru D0 selects a baud rate, with the correspondence tabulated below. If more than one of these bits is set to logic 1, the highest baud rate selected will result. If no bits D6 thru D0 are set to logic 1, the serial transmitter is disabled. If COMMAND REGISTER bit D4 (High Baud, or HBD) is set to logic 1, the baud rate will undergo an eightfold increase. The bit assignments follow:

BIT	FUNCTION	
D7	STOP BITS	
D6	9,600 BAUD	76,800 BAUD
D5	4,800 BAUD	38,400 BAUD
D4	2,400 BAUD	19,200 BAUD
D3	1,200 BAUD	9,600 BAUD
D2	300 BAUD	2,400 BAUD
D1	150 BAUD	1,200 BAUD
D0	110 BAUD	880 BAUD
	(HBD = LOGIC 0)	(HBD = LOGIC 1)

RECEIVER DATA (IN 01H)

The Z80A reads this port to input a serial data byte from the terminal device. An RDA condition notifies the Z80A that this port may be read, and status bits ORE and FME may be sampled to verify the data integrity. Eight data bits are received from the terminal device, and the direction of shift is illustrated above.

TRANSMITTER DATA (OUT 01H)



The Z80A loads this port to output a serial data byte to the terminal device. A TBE condition notifies the Z80A that this port may be loaded. Eight data bits are sent to the terminal device, and the direction of shift is illustrated above.

COMMAND REGISTER (OUT 02H)

NOT USED	NOT USED	TB5	HBD	INE	RS7	BRK	RES

BIT	LABEL	FUNCTION
D7		Not used. Either a logic 0 or a logic 1 may be output to D7.

D6		Not used. Either a logic 0 or a logic 1 may be output to D6.
D5	TB5	Test Bit D5. TB5 should be reset to logic 0 for normal operation. Setting TB5 to logic 1 disables the internal interrupt priority logic and enables the 5501 internal clock making its INT pin a TTL level clock of 1562.5 Hz if HBD = logic 0, or 12.5 KHz if HBD = logic 1. TB5 is provided for test purposes.
D4	HBD	High Baud. Setting HBD to logic 1 octuples the internal clock frequency. This causes each TIMER to decrement its DELAY COUNT once every 8 uSec instead of once every 64 uSec, and serial transmit and receive data rate to octuple.
D3	INE	INTA Enable. The 5501 monitors the data bus at SYNC of each machine cycle to determine if the Z80A is entering an Interrupt Acknowledge (INTA) cycle. Resetting INE to logic 0 causes the 5501 to ignore the INTA cycle. Setting INE to logic 1 causes the 5501 to accept the INTA, and gate one-of-eight RST instructions onto the data bus to service the interrupt. The RST instruction gated onto the bus will correspond to the highest priority interrupt source currently requesting service.
D2	RS7	RST 7 Select. The lowest priority interrupt (level 7) source is selected with this control bit. Setting RS7 to logic 1 defines bit D7 (MSB) of SCC port IN 0AH to be auxiliary INT input PI7; a low-to-high transition on PI7 activates a level 7 interrupt request. Resetting RS7 to logic 0 defines TIMER 5 as the level 7 interrupt source; TIMER 5 reaching DELAY COUNT zero activates a level 7 interrupt request.
D1	BRK	Break. Setting BRK to logic 1 latches the serial transmitter output XMT in the low (spacing) state. Resetting BRK to logic 0 allows the serial transmitter to operate normally. RES overrides BRK.
D0	RES	5501 Reset. Setting RES to logic 1 results in the following: (1) Flags RDA, SBD, FBD and ORE are reset to logic 0; the serial receiver goes into the 'search for new character' mode; the current contents of RECEIVER DATA, port IN 01H, is not affected and contains the last character received.

(2) Flag TBE is set to logic 1; the serial transmitter output goes high (marking). RES overrides BRK if both are set to logic 1 in the same command byte.

(3) All interval timers are cleared, and all active interrupt requests are cleared except for TBE.

(4) If TBE interrupt requests are enabled (unmasked), IPG is set to logic 1. If TBE interrupt requests are disabled (masked), IPG is reset to logic 0.

(5) RES is automatically reset to logic 0 after steps (1) thru (4) are completed. Outputting a logic 0 RES bit causes no response from the 5501.

INTERRUPT ADDRESS (IN 03H)

1	1	L_2	L_1	L_0	1	1	1

This port is polled by the Z80A to determine the highest unmasked (enabled) interrupt source currently requesting service. The 5501 may also be programmed to place the contents of this port on the data bus in response to an INTA cycle thru COMMAND REGISTER bit INE. After this port is either read by the Z80A, or after it is gated onto the data bus in response to an INTA cycle, the corresponding interrupt request is cleared. The INTERRUPT ADDRESS contents are then replaced with the address of the next highest priority enabled interrupt currently requesting service. If INTERRUPT ADDRESS port IN 03H is read with no interrupt pending, its contents will be 0FFH. SERIAL STATUS bit IPG (Interrupt Pending) should then be used to differentiate between 'no 5501 interrupts pending' (an interrupt from S-100 bus line $\overline{INT}$, for example) and 'TIMER 5/PI7' conditions when interrupt polling.

INTERRUPT SOURCE	INTERRUPT ADDRESS D7	D6	D5	D4	D3	D2	D1	D0	PRIORITY LEVEL	OPCODE
TIMER 1	1	1	0	0	0	1	1	1	0 (HIGHEST)	0C7H (RST 00H)
TIMER 2	1	1	0	0	1	1	1	1	1	0CFH (RST 08H)
INT	1	1	0	1	0	1	1	1	2	0D7F (RST 10H)
TIMER 3	1	1	0	1	1	1	1	1	3	0DFH (RST 18H)
RDA	1	1	1	0	0	1	1	1	4	0E7H (RST 20H)
TBE	1	1	1	0	1	1	1	1	5	0EFH (RST 28H)
TIMER 4	1	1	1	1	0	1	1	1	6	0F7H (RST 28H)
TIMER 5/PI7	1	1	1	1	1	1	1	1	7 (LOWEST)	0FFH (RST 30H)

NO REQUEST 1 1 1 1 1 1 1 1 7 (LOWEST) 0FFH (RST 38H)

INTERRUPT MASK (OUT 03H)

TIMER 5 OR PI7	TIMER 4	TBE	RDA	TIMER 3	$\overline{\text{INT}}$	TIMER 2	TIMER 1

BIT	MASK REQUEST	UN-MASK REQUEST	INTERRUPT SOURCE
D7	0	1	TIMER 5 OR PI7
D6	0	1	TIMER 4
D5	0	1	TBE
D4	0	1	RDA
D3	0	1	TIMER 3
D2	0	1	INT (J3 PIN 23)
D1	0	1	TIMER 2
D0	0	1	TIMER 1

Each bit position in this output port selectively masks (disabled) or un-masks (enables) interrupt requests emanating from one of the eight 5501 interrupt sources. A logic 0 bit masks an interrupt request; a logic 1 bit un-masks it. Note that interrupt requests are latched even when the corresponding INTERRUPT MASK bits are reset to logic 0. In such cases, an interrupt request will be retained until its corresponding mask bit is programmatically changed to logic 1 (it will then become enabled, and will be prioritized with other enabled requests), or until COMMAND REGISTER bit RES is set to logic 1 which clears all interrupt requests. An un-masked interrupt request is prioritized with all other enabled requests, and if it is the highest priority currently requesting service, its encoded address is placed in INTERRUPT ADDRESS port IN 03H. In this case it sets SERIAL STATUS bit IPG to logic 1, and requests servicing by forcing Z80A input $\overline{\text{INT}}$ active low. A masked interrupt request cannot be sensed by the Z80A unless it is either an RDA or TBE, which are always available as SERIAL STATUS bits.

PARALLEL DATA (IN 04H)

D7	D6	D5	D4	D3	D2	D1	D0

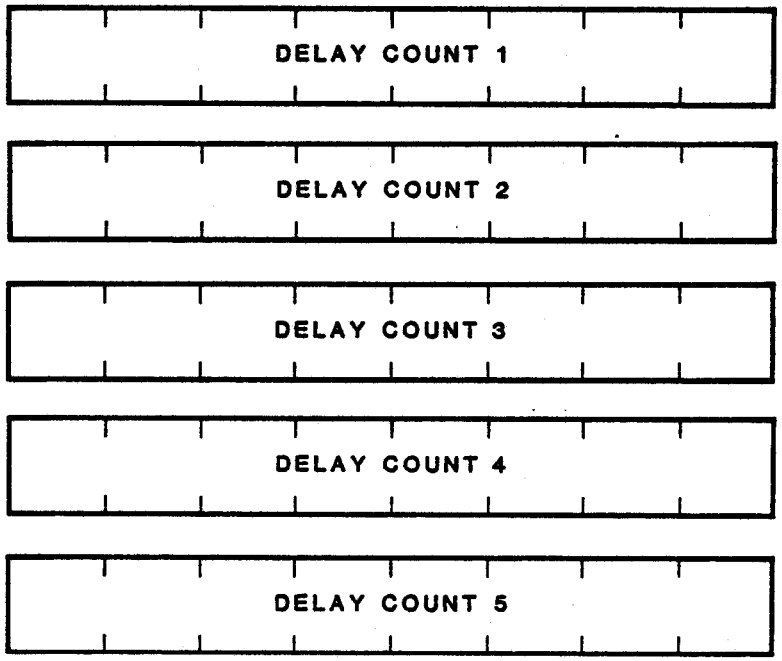
Inputs parallel data from SCC connector J3. MSB bit D7 of this parallel input port may be defined as auxiliary interrupt

input source PI7 by setting COMMAND REGISTER bit RS7 to logic 1. If RS7 is reset to logic 0, D7 functions as an ordinary MSB data bit.

PARALLEL DATA (OUT 04H)

Outputs parallel data to SCC connector J3.

TIMER 1 (OUT 05H)
TIMER 2 (OUT 06H)
TIMER 3 (OUT 07H)
TIMER 4 (OUT 08H)
TIMER 5 (OUT 09H)



The Z80A outputs a one byte DELAY COUNT to any of the ports listed above to start its associated interval timer. A timer decrements its DELAY COUNT once every 64 uSec (if COMMAND REGISTER bit HBD is logic 0), or once every 8 uSec (if HBD is logic 1). A timer issues an interrupt request upon timeout. The resulting timing interval from loading to count zero is plus 0 uSec, and minus 64 uSec (HBD=logic 0) or 8 uSec (HBD=logic 1). The Z80A has no way to detect this event unless the timer interrupt request is enabled thru INTERRUPT MASK port OUT 03H. An enabled timer interrupt request is prioritized along with other enabled

requests, and if it is the highest priority currently requesting service, its encoded address is placed in INTERRUPT ADDRESS port IN 03H. In this case SERIAL STATUS bit IPG is set to logic 1, and a Z80A service request is made by forcing Z80A pin $\overline{INT}$ active low. Loading DELAY COUNT zero causes an immediate interrupt request, and loading a new DELAY COUNT while a timer is counting re-initializes the timer without generating an interrupt request. A 5501 RESET from COMMAND REGISTER bit RES clears all five timers and clears any pending timer interrupt requests.

PARALLEL DATA (IN 0AH)

D7	D6	D5	D4	D3	D2	D1	D0

Inputs parallel data from SCC connector J1.

PARALLEL DATA (OUT 0AH)

D7	D6	D5	D4	D3	D2	D1	D0

Outputs parallel data to SCC connector J1. MSB bit D7 controls SCC on-board memory enable/disable if the SCC solder trace labeled 'DISABLE' is cut. In this case, D7 = logic 1 disables SCC memory; D7 = logic 0 enables SCC memory. If the trace is not cut, SCC memory is always enabled, and D7 functions as an ordinary MSB data bit.

PARALLEL DATA (IN 0BH)

D7	D6	D5	D4	D3	D2	D1	D0

Inputs parallel data from SCC connector J2.

PARALLEL DATA (OUT 0BH)

D7	D6	D5	D4	D3	D2	D1	D0

Outputs parallel data to SCC connector J2.

Appendix B

ASCII CHARACTER SET
(7-BIT CODE)

00H	NUL	(^@)	20H	SPACE		40H	@		60H	'	
01H	SOH	(^A)	21H	!		41H	A		61H	a	
02H	STX	(^B)	22H	"		42H	B		62H	b	
03H	ETX	(^C)	23H	#		43H	C		63H	c	
04H	EOT	(^D)	24H	$		44H	D		64H	d	
05H	ENG	(^E)	25H	%		45H	E		65H	e	
06H	ACK	(^F)	26H	&		46H	F		66H	f	
07H	BEL	(^G)	27H	'		47H	G		67H	g	
08H	BS	(^H)	28H	(		48H	H		68H	h	
09H	HT	(^I)	29H	)		49H	I		69H	i	
0AH	LF	(^J)	2AH	*		4AH	J		6AH	j	
0BH	VT	(^K)	2BH	+		4BH	K		6BH	k	
0CH	FF	(^L)	2CH	,		4CH	L		6CH	l	
0DH	CR	(^M)	2DH	-		4DH	M		6DH	m	
0EH	SO	(^N)	2EH	.		4EH	N		6EH	n	
0FH	SI	(^O)	2FH	/		4FH	O		6FH	o	
10H	DLE	(^P)	30H	0		50H	P		70H	p	
11H	DC1	(^Q)	31H	1		51H	Q		71H	q	
12H	DC2	(^R)	32H	2		52H	R		72H	r	
13H	DC3	(^S)	33H	3		53H	S		73H	s	
14H	DC4	(^T)	34H	4		54H	T		74H	t	
15H	NAK	(^U)	35H	5		55H	U		75H	u	
16H	SYN	(^V)	36H	6		56H	V		76H	v	
17H	ETB	(^W)	37H	7		57H	W		77H	w	
18H	CAN	(^X)	38H	8		58H	X		78H	x	
19H	EM	(^Y)	39H	9		59H	Y		79H	y	
1AH	SUB	(^Z)	3AH	:		5AH	Z		7AH	z	
1BH	ESC	(^[)	3BH	;		5BH	[		7BH	{	
1CH	FS	(^\)	3CH	<		5CH	\		7CH	|	
1DH	GS	(^])	3DH	=		5DH	]		7DH	}	
1EH	RS	(^^)	3EH	>		5EH	^		7EH	~	
1FH	VS	(^_)	3FH	?		5FH			7FH	DEL	

Note: (^A) means Control-A, (^B) means Control-B, etc.

SCC PARTS LIST

Integrated Circuits		Part No.
IC1	74LS373	010-0102
IC2	74LS273	010-0107
IC3	74LS373	010-0102
IC4	74LS273	010-0107
IC5	74LS22	010-0105
IC6-7	74LS10	010-0063
IC8	74LS367	010-0108
IC9	74LS00	010-0069
IC10	74LS04	010-0066
IC11	74LS00	010-0069
IC12	74LS74	010-0055
IC13	74LS240	010-0038
IC14	74LS244	010-0100
IC15	74LS367	010-0108
IC16	74LS74	010-0055
IC17	LM320-5/7905	012-0000
IC18	74LS04	010-0066
IC19-20	74LS367	010-0108
IC21	74LS03	010-0067
IC22	7407	010-0104
IC23	74LS32	010-0058
IC24	74LS260	010-0094
IC25	74904	010-0109
IC26	5501	011-0005
IC27	7406	010-0028
IC28	74LS51	010-0106
IC29	74LS08	010-0064
IC30	74LS32	010-0058
IC31	74LS04	010-0066
IC32	74LS74	010-0058
IC33	74LS42	010-0057
IC34-35	4045	011-0012
IC36	Z80-A	011-0010
IC37-38	74LS04	010-0066
IC39	74LS08	010-0064
IC40	74LS00	010-0069
IC41	LM340-5/7805	012-0001
IC42	7812	012-0002
IC43-46	2716	NOT SUPPLIED
IC47-51	74LS244	010-0100
IC52	74LS367	010-0108
IC53	74LS11	010-0062

Transistors/Diodes		Part No.
Q1	2N3906	009-0002
Q2-3	2N3904	009-0001
Q4-5	2N3906	009-0002
Q6	2N3904	009-0001
D1-2	1N914	008-0002

Capacitors		Part No.
C1	.1 UF DISK	004-0030
C2	.05 UF DISK	004-0027
C3-C6	.1 UF DISK	004-0030
C7	300 PF DISK	004-0015
C8-14	.1 UF DISK	004-0030
C15-16	150 PF DISK	004-0011
C17-18	.1 UF DISK	004-0030
C19-21	10 UF TANT.	004-0032
C22	30 PF DISK	004-0003
C23-24	.1 UF DISK	004-0030
C25	.05 UF DISK	004-0027
C26	10 UF TANT.	004-0032
C27	.1 UF DISK	004-0030
C28	10 UF TANT.	004-0032
C29	6.8 UF TANT.	004-0034
C30	6.8 UF TANT.	004-0034
C31-39	.1 UF DISK	004-0030

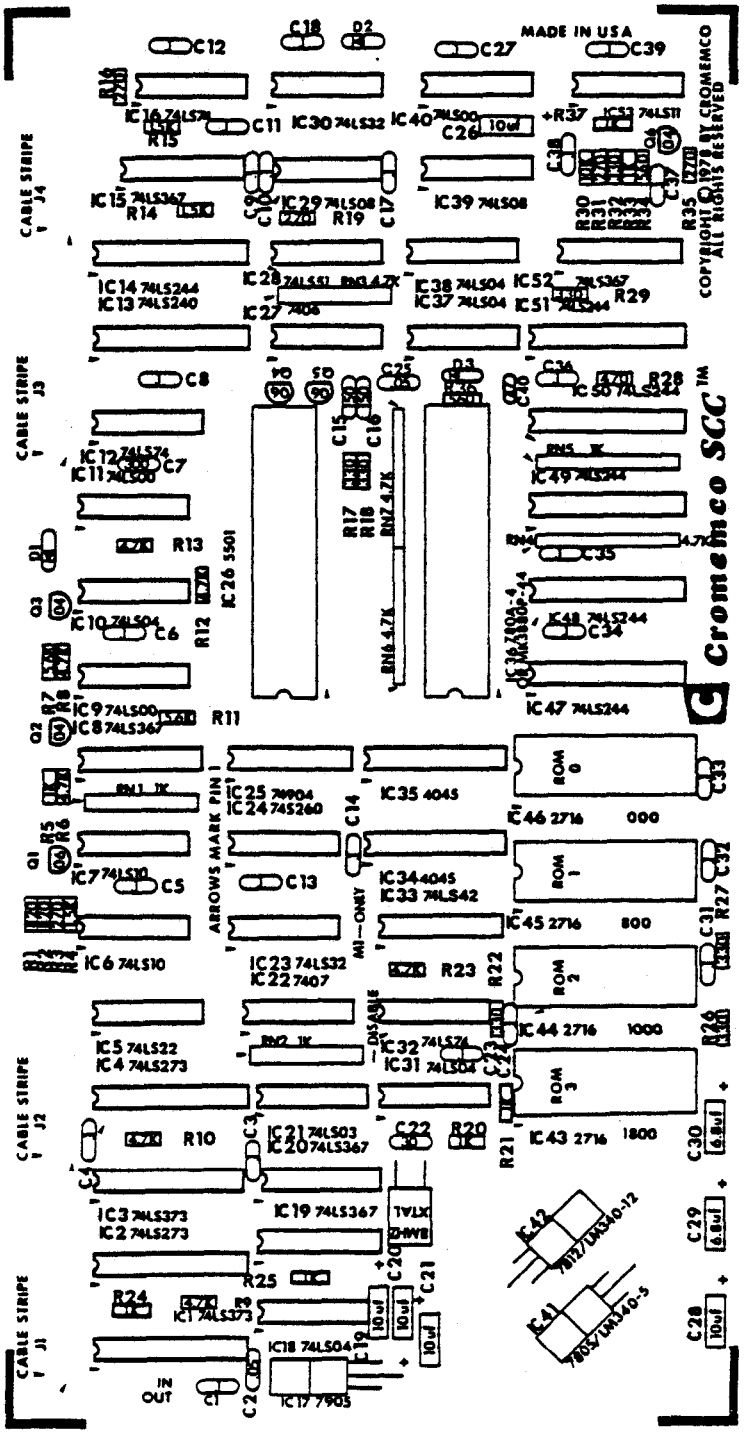
Resistors		Part No.
R1-2	820	001-0017
R3	270	001-0011
R4	3.3 K	001-0041
R5	1 K	001-0018
R6	4.7 K	001-0024
R7	5.6 K	001-0026
R8-10	4.7 K	001-0024
R11	5.6 K	001-0026
R12-13	4.7 K	001-0024
R14-15	1.5 K	001-0020
R16	270	001-0011
R17-18	330	001-0012
R19	270	001-0011
R20-21	1 K	001-0018
R22	330	001-0012
R23-25	4.7 K	001-0024
R26-27	330	001-0012
R28	470	001-0014
R29	330	001-0012
R30	10 K	001-0030
R31	270	001-0011
R32	330	001-0012
R33	1 K	001-0018
R34	560	001-0015
R35	270	001-0011

Resistor Networks		Part No.
RN1-2	1K, 8 PIN	003-0007
RN3	4.7K, 8 PIN	003-0009
RN4-5	1K, 10 PIN	003-0011

SCC PARTS LIST

Miscellaneous	Part No.
2 EA. SOCKETS, 40 PIN	017-0006
4 EA. SOCKETS, 24 PIN	017-0005
11 EA. SOCKETS, 20 PIN	017-0004
2 EA. SOCKETS, 18 PIN	017-0003
7 EA. SOCKETS, 16 PIN	017-0002
24 EA. SOCKETS, 14 PIN	017-0001
4 EA. CONNECTOR STRIPS, 26 PIN, RIGHT ANGLE	017-0012
1 EA. CRYSTAL, 8 MHZ	026-0001
4 EA. 6-32X3/8 SCREW	015-0006
4 EA. 6-32 HEX NUTS	015-0013
1 EA. LARGE HEAT SINK	021-0017

Documentation	Part No.
SCC INSTRUCTION MANUAL	023-0050


MADE IN USA
COPYRIGHT © 1978 BY CROMEMCO
ALL RIGHTS RESERVED
Cromemco SCC TM
CABLE STRIPE J4
CABLE STRIPE J3
CABLE STRIPE J2
CABLE STRIPE J1
ARROWS MARK PIN 1
M1—ONLY
DISABLE
IN
OUT
ROM 0
ROM 1
ROM 2
ROM 3

LIMITED WARRANTY

Cromemco, Inc. ("Cromemco") warrants this product against defects in material and workmanship to the original purchaser for ninety (90) days from the date of purchase, subject to the following terms and conditions.

What Is Covered By This Warranty

During the ninety (90) day warranty period Cromemco will, at its option, repair or replace this Cromemco product or repair or replace with new or used parts any parts or components, manufactured by Cromemco, which prove to be defective, provided the product is returned to an Authorized Cromemco Dealer as set forth below.

How To Obtain Warranty Service

You should immediately notify IN WRITING your Authorized Cromemco Dealer or Cromemco Inc of problems encountered during the warranty period. In order to obtain warranty service, first obtain a return authorization number by contacting the Authorized Cromemco Dealer from whom you purchased the product. Then attach to the product:

1. Your name, address and telephone number,
2. the return authorization number,
3. a description of the problem, and
4. proof of the date of retail purchase.

Ship or otherwise return the product, transportation and insurance costs prepaid, to the Authorized Cromemco Dealer. If you are unable to receive warranty repair from the Authorized Cromemco Dealer from whom you purchased the product, you should contact Cromemco Customer Support at: Cromemco, Inc., 280 Bernardo Ave., Mountain View, Ca. 94043.

What Is Not Covered By This Warranty

Cromemco does not warrant any products, components or parts not manufactured by Cromemco.

This warranty does not apply if the product has been damaged by accident, abuse, misuse, modification or misapplication; by damage during shipment; or by improper service. This product is not warranted to operate satisfactorily with peripherals or products not manufactured by Cromemco. Transportation and insurance charges incurred in transporting the product to and from the Authorized Cromemco Dealer or Cromemco are not covered by this Warranty.

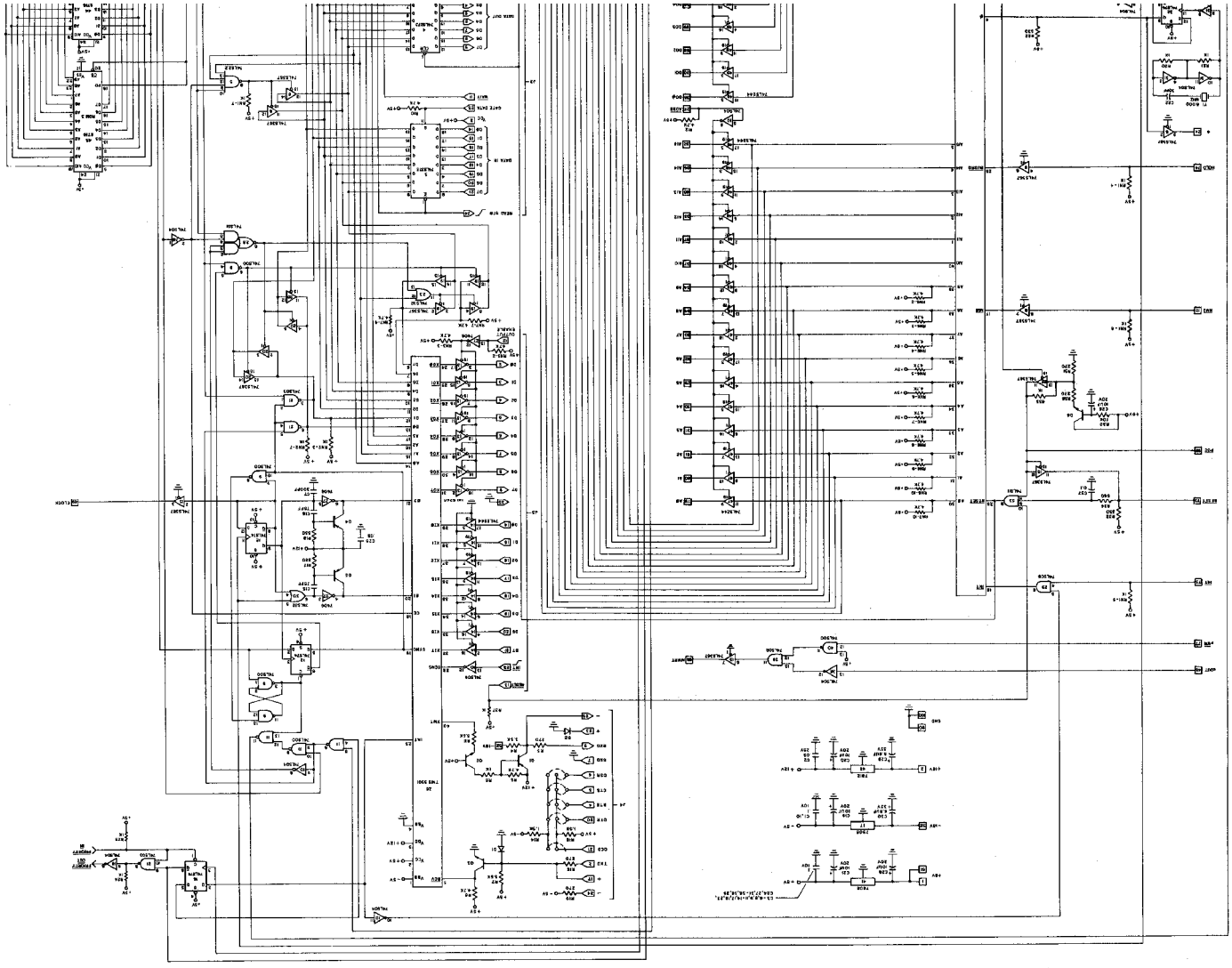
THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. ANY IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF PURCHASE OF THIS PRODUCT. CROMEMCO SHALL NOT BE LIABLE FOR INCIDENTAL AND/OR CONSEQUENTIAL DAMAGES FOR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY, EVEN IF CROMEMCO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOFTWARE, TECHNICAL INFORMATION AND FIRMWARE IS LICENSED "AS IS" AND WITH ALL FAULTS. THE AGENTS, DEALERS, AND EMPLOYEES OF CROMEMCO ARE NOT AUTHORIZED TO MAKE MODIFICATIONS TO THIS WARRANTY, OR ADDITIONAL WARRANTIES BINDING ON CROMEMCO ABOUT OR FOR PRODUCTS SOLD OR LICENSED BY CROMEMCO. ACCORDINGLY, ADDITIONAL STATEMENTS WHETHER ORAL OR WRITTEN EXCEPT SIGNED WRITTEN STATEMENTS FROM AN OFFICER OF CROMEMCO DO NOT CONSTITUTE WARRANTIES AND SHOULD NOT BE RELIED UPON.

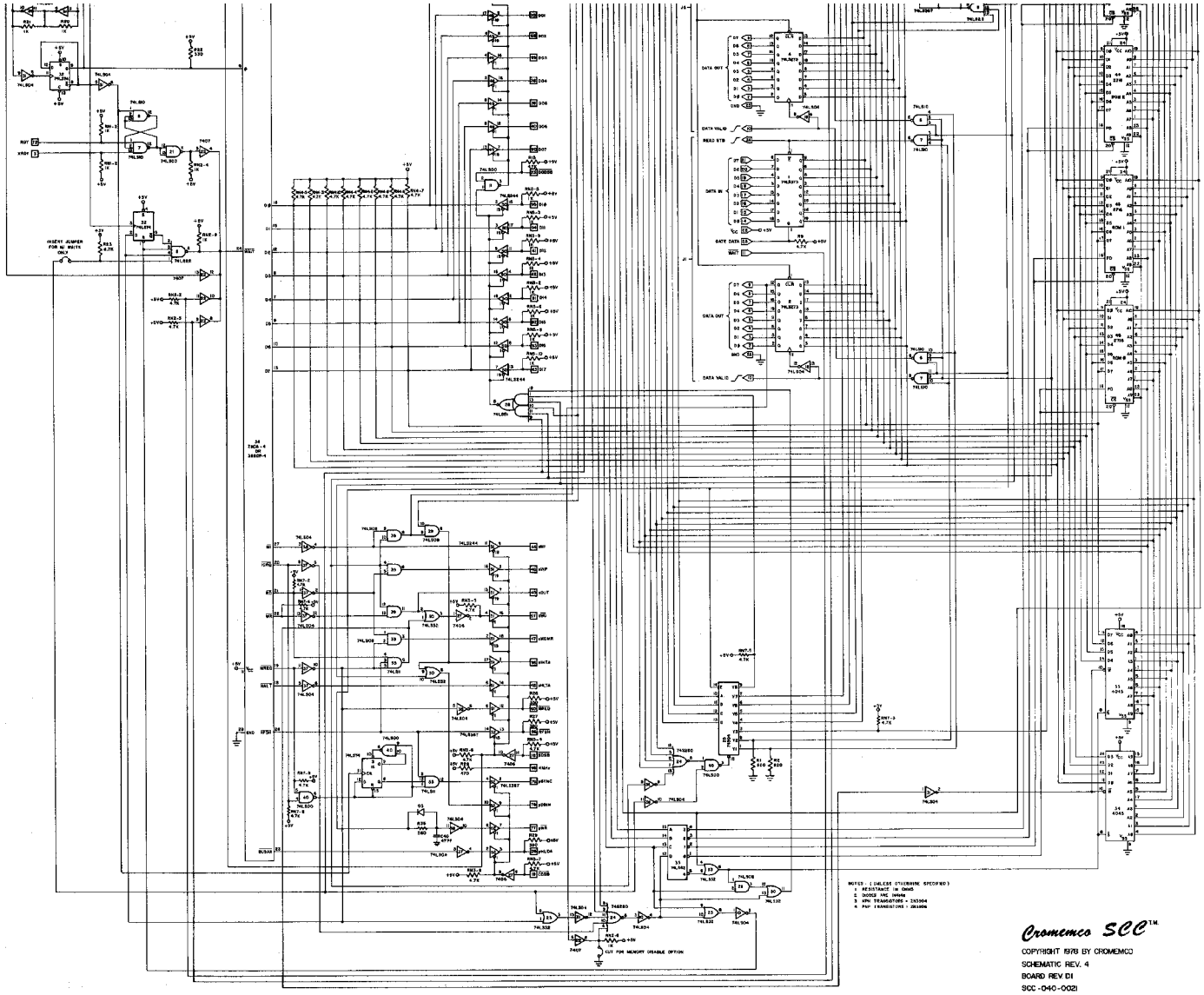
SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES OR LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

THIS WARRANTY AND THE STATUTE OF LIMITATIONS SHALL RUN CONCURRENTLY WITH ANY ACCEPTANCE PERIOD. THIS WARRANTY IS NOT TRANSFERABLE. NO SUIT, LITIGATION, OR ACTION SHALL BE BROUGHT BASED ON THE ALLEGED BREACH OF THIS WARRANTY OR IMPLIED WARRANTIES MORE THAN ONE YEAR AFTER THE DATE OF PURCHASE IN THOSE JURISDICTIONS ALLOWING SUCH A LIMITATION, OTHERWISE NO SUCH ACTION SHALL BE BROUGHT MORE THAN ONE YEAR AFTER THE EXPIRATION OF THIS WARRANTY.

THIS WARRANTY SHALL NOT BE APPLICABLE TO THE EXTENT THAT ANY PROVISION OF THIS WARRANTY IS PROHIBITED BY ANY FEDERAL, STATE OR MUNICIPAL LAW WHICH CANNOT BE PREEMPTED. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

5/81





Cromemco SCC™

SCHEMATIC REV. 4

BOARD REV 01

SCC-040-0021