Publication Number 4200113A

SC/MP Kit Users Manual

March 1976

2900 Semiconductor Drive
Santa Clara, California 95051

PREFACE

This manual is intended to assist the SC/MP Kit user in the assembly of the kit and in the operation of the KITBUG firmware included with the kit. The assembly procedures assume familiarity with basic electronic assembly techniques and tools. The operating instructions for KITBUG are fully described and require no particular prior experience. The actual use of SC/MP Kit implies some familiarity with electronic interface requirements and techniques and with computer programming. An application example that may be of assistance in understanding the use and operation of SC/MP Kit is included as appendix C of this manual. Listed below are additional sources of interfacing and programming information supplied with the kit.

- SC/MP Technical Description
- SC/MP Programming and Assembler Manual
- Data Sheets for each integrated circuit provided with the kit

CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Chapter 1

SC/MP KIT DESCRIPTION

1.0 INTRODUCTION

The SC/MP Demonstration Kit includes all the integrated circuits and discrete components required to build a small and completely functional microprocessor system. The kit includes the following items.

- SC/MP microprocessor chip
- 256 bytes (8 bits/byte) of read/write memory (RAM) for storage of user programs
- 512 bytes of preprogrammed read-only memory (ROM) containing a debug program and Teletype® input/output routines
- An 8-bit buffer between the outputs of the memory devices and the SC/MP chip inputs
- Interface circuitry to provide the level conversions and drive requirements for a serial input/output interface to a Teletype®
- Voltage regulator and crystal to meet the SC/MP power and timing requirements
- Printed circuit board on which the components can be mounted
- Mounting sockets for SC/MP and ROM chips
- 72-pin edge connector socket
- All required discrete components

The kit allows both microprocessor veterans and newcomers to build and exercise a viable microprocessor system. The kit is a valuable aid in understanding the functions and capabilities of the SC/MP microprocessor and should also prove useful in developing basic system concepts. Using the kit, small programs can be developed and entered into memory via the Teletype® (TTY) keyboard: the programs then can be executed and their operation monitored by the KITBUG program. Thus, the kit provides a simple and effective way of familiarizing users with the characteristics of the SC/MP instruction set.

Additionally, the SC/MP Kit is ideally suited for quickly implementing a variety of simple "real-life" applications and demonstrations. For example, the input/output control signals and control-oriented instruction set allow the kit to function as a program-controlled timer and to operate external lights, switches, and controls. Photographic lights, lawn sprinkler systems, and alarm and security systems are just a few examples of possible applications.

1.1 FUNCTIONAL OVERVIEW

A block diagram of the kit is shown in figure 1-1. The paragraphs that follow provide a general functional overview of the major elements of the kit. Neither assembly nor operation of the kit require a detailed understanding of the hardware configurations or interrelationship; however, for detailed descriptions of SC/MP and various system configurations refer to the SC/MP Technical Description (Publication Number 4200079). The printed circuit board supplied with the kit provides the interconnections between the components of the kit; assembly of the kit is described in chapter 2. The KITBUG program provides the basic routines necessary to operate the system; KITBUG operation is described in chapter 4.

1.1.1 SC/MP Microprocessor

The SC/MP microprocessor is, of course, the heart (or, more accurately, the brain) of the kit. SC/MP, under the control of KITBUG or your own program, provides the data manipulation and the sequencing and control required for all kit operations.

The FLAG 0 output from SC/MP is used to transmit data to the TTY and the SENSE B input to SC/MP is used to receive data from the TTY. These serial data transfers are accomplished under control of the KITBUG program. A variety of other input, output and control signals are provided by SC/MP and are available for use by your programs. A complete description of SC/MP is provided in the SC/MP Data Sheet and the SC/MP Technical Description.

1.1.2 Memory Data Transfers

The memory provided with the kit consists of 512 bytes of read-only memory (ROM) and 256 bytes of read/write memory (RAM). Transfers of data from ROM or to and from read/write memory are accomplished by sending out an address (from SC/MP) to select the desired memory location and then sending a read (NRDS) or write (NWDS) data strobe signal to indicate the direction of data flow and to synchronize the transfer of the data. The parallel transfers of data between SC/MP and memory are accomplished via the 8-bit bidirectional SC/MP data lines.

®Teletype Corporation Trademark.

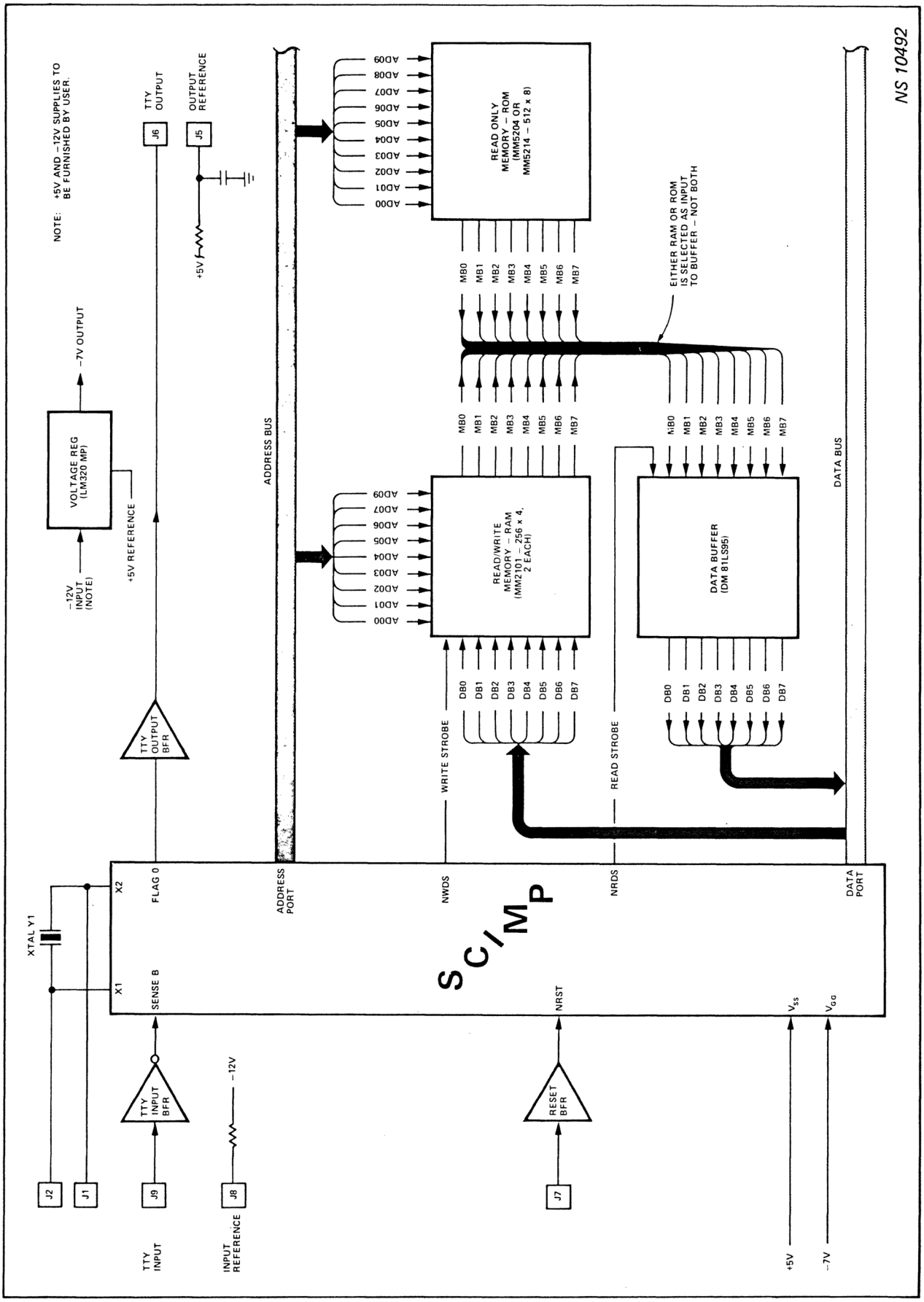


Figure 1-1. SC/MP Kit Block Diagram

Although SC/MP sends out a 16-bit address during memory operations, not all of these bits are actually needed to address the memory supplied with the kit. Table 1-1 defines the address bits that are used in the kit and the corresponding hexadecimal values for these addresses. Note that since not all of the bits are used, a particular memory location can be specified by several different 16-bit addresses. For example, RAM location $0FFF_{(16)}$ can also be addressed as $FAFF_{(16)}$, $26FF_{(16)}$ and so on.

The write strobe signal (NWDS) is used as the read/write control signal for RAM. When NWDS is low, it indicates that the data on the data lines is to be written into the RAM location specified by the address bus.

For read operations, NWDS remains high. The address bits select a location in RAM or ROM and the data from the specified memory location (of either RAM or ROM) is presented to the inputs of the data buffer. The read data strobe signal (NRDS) is then sent out by SC/MP to gate the data through the buffer and into SC/MP.

1.1.3 Timing, Power, and Reset

All of the timing requirements of the SC/MP kit are met by a 1.000-MHz crystal, which is connected to the X1 and X2 inputs to the SC/MP microprocessor chip.

The components of the kit require three regulated voltages: +5V (V_{SS}), -7V (V_{GG}), and -12V. The user provides the +5V and -12V power: the -7V is derived from the voltage regulator included with the kit. The +5V and -12V provided by the user must meet the following specifications:

+5V ± 5% @ 350 milliamperes
-12V ± 5% @ 200 milliamperes

NOTE

If additional circuits are added to the kit, the power requirements also increase.

The reset circuit consists of a resistor-capacitor network and two serially connected inverters; in figure 1-1, these components are represented by the reset buffer. When J7 is grounded, all system operations are aborted. When the ground is removed, the low-to-high transition initializes (zeros) all SC/MP registers and the next instruction is fetched from location 0001_{16} in ROM (the beginning of the KITBUG program).

Table 1-1. SC/MP Kit Memory Addressing

Address Bits	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
ROM Selected	X	X	X	X	X	X	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	
	0-F				0, 1, 4, 5, 8, 9, C, or D				0-F				0-F				Hexadecimal Values
RAM Selected	X	X	X	X	X	X	1	X	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	
	0-F				2, 3, 6, 7, A, B, E, or F				0-F				0-F				Hexadecimal Values

X = not used in standard kit configuration

1.2 THE PRINTED CIRCUIT BOARD

The printed circuit board has plated-through holes that accept the leads from the integrated circuits and discrete components supplied with the kit. The board, whose dimensions are 4.375 inches by 4.862 inches, also provides room for some additional components to expand the system as described in chapter 3. The traces on the board interconnect the components. The board is equipped with a 72-pin edge connector to allow mounting in standard card cages. Table 1-2 lists some sources of compatible card cages, extender cards, mating connectors and wire-wrap breadboard cards.

All of the SC/MP signals are available at "stake holes" on the card: the desired signals can be easily wired to the edge-connector pins, thus allowing complete flexibility in designating the connections to the card-edge. The stake holes are located along each side of the SC/MP chip next to the holes in which the chip itself is mounted and can be seen in figure 2-1. On the kit schematic (figure 2-2), the stake holes are indicated by the small square boxes located on the SC/MP signal lines adjacent to each SC/MP pin. The TTY interface signals, the Reset input signal, and the power inputs are already wired to the card-edge pins. The pin assignments for these signals are shown in figure 2-2.

Table 1-2. Sources of Accessory Equipment*

Equipment	Source	Part Number
72-contact Edge Connector	Augat Robinson-Nugent Stanford Applied Eng. National Connector Cinch Winchester Elco Viking	14005-17P3 EC-721 CDP7000-72 900100-36 50-72C-30 HW36C0111 00-6307-072-309-001 3VH36/1JND5
13-connector Card Cage with Backplane	Augat Robinson-Nugent Scanbe	8170-MG1 MECA-1
9-connector Card Cage with Backplane	Augat	8170-MG10
6-connector Card Cage with Backplane	Augat	8170-MG8
3-connector Card Cage with Backplane	Augat	8170-MG6
Extender Card	Augat Robinson-Nugent	8136-MG13 EB-72
Universal wire-wrap Card with Terminals	Augat Robinson-Nugent	8136-UMG1 UNI-24
High-density wire-wrap Card with Terminals	Augat Robinson-Nugent	8136-MG15
Universal wire-wrap Card without Terminals	Robinson-Nugent	(Special)

*The accessory equipment listed in Table 1-1 has not necessarily been evaluated by National Semiconductor.

Chapter 2

KIT ASSEMBLY AND CHECKOUT

2.1 TOOL AND MATERIAL REQUIREMENTS

The SC/MP demonstration kit can be assembled with soldering equipment and very simple tools. Recommended tools and materials for the assembly process are listed in table 2-1. Some general recommendations and precautions are listed below.

- Review data sheets for all integrated devices supplied with the kit to verify pin-outs and pin orientation.
- To avoid unnecessary component replacement, ensure that polarized capacitors are installed with the correct polarity; also, check that each integrated-circuit module is properly oriented (pin #1 in square hole) before soldering component leads. (Refer to "Stuffing Procedures" for detailed information.)
- Do not use a high-powered soldering iron or gun; excessive heat may lift a soldering pad or, worse yet, it can damage the board or components. (Refer to table 2-1 for proper soldering equipment.)
- If a soldered component must be changed, use a suction device or wooden toothpick to remove solder from component-mounting holes. *Do not use a sharp metal object to remove solder; the plated-through conductor can be permanently damaged by such means.*
- After soldering, remove excess flux from the soldered areas.

Table 2-1. Recommended Tools and Soldering Equipment for Assembly of SC/MP Kit

Item	Use	Specification	Recommendation
Soldering Tool	Soldering/Desoldering	Wattage: Not more than 40 Tip Temp: 600°F maximum	Weller Soldering Station W-TCP-L, or equivalent components
Desoldering Aid	To remove molten solder	Suction Device	Soldapuldt or equivalent
Solder	Component installation, component replacement, and miscellaneous wiring	Resin (flux) core, high tin content (60% tin/40% lead); 18 gauge (SWG) preferred	Commercial
Resin (flux) Solvent	Removal of excess flux from soldered area	Must not dissolve or otherwise affect board material or conductor bonding agent	Freon, Acetone, and/or Isopropyl Alcohol (100% dry)
Continuity Checker	To check solder connections	Must not exceed 1.5 volts	
Long Nose Pliers	Component installation, component replacement, and miscellaneous wiring		
Diagonals	Component installation, component replacement, and miscellaneous wiring		

CAUTION

The MOS devices (SC/MP, RAM, and ROM) can be damaged by contact with an electrostatic or high-voltage charge. To guard against this, the following handling precautions are recommended.

- MOS devices should be stored or transported in conductive material so that all exposed leads are shorted together. Styrofoam or plastic trays must not be used.
- A grounded bench surface should be used, and soldering equipment or any other apparatus used in assembling the kit should be grounded.
- Nylon clothing should not be worn while handling MOS devices, and you should ground yourself prior to handling the devices.

First Band (1st Digit)		Second Band (2nd Digit)	
Black	0	Black	0
Brown	1	Brown	1
Red	2	Red	2
Orange	3	Orange	3
Yellow	4	Yellow	4
Green	5	Green	5
Blue	6	Blue	6
Violet	7	Violet	7
Gray	8	Gray	8
White	9	White	9

Third Band (Multiplier)

Gold	0.1
Black	1
Brown	10
Red	100
Orange	1,000
Yellow	10,000
Green	100,000
Blue	1,000,000

2.2 STUFFING PROCEDURES

2.2.1 Component Count and Identification

Upon receipt of the SC/MP demonstration kit, the user first should verify that all components specified in the parts list (appendix A) are included. Then, each component should be identified by part number, value (resistance or capacitance), or any other specified parameter.

Capacitors can be identified by case markings that usually consist of the value in "μf" plus the voltage rating (if applicable); polarized capacitors are generally marked with a "+" at one end of the case to indicate mounting orientation. Two small capacitors, C6 and C7 (0.1 μf, 50V), are included with the kit. These capacitors are not polarized and typical case markings are ".1Z," "104Z," or "CK104" which indicates 0.1 microfarad. Resistors are identified by a standard color code, where the first, second, and third bands of the code define the resistor value. For convenience, the color code is listed below.

NOTE

The "first" band is the band that is nearest to the end of the resistor.

Referring to the preceding color code, R1—a 10K resistor, is coded with a brown band to designate the first digit as '1', a black band to identify '0' as the second digit, and an orange band to specify a multiplier of 1000; thus, a 10 x 1000, or 10K. Most resistors have a fourth band to designate tolerance; that is, gold for 5%, silver for 10%. All integrated circuits are identified by case markings.

2.2.2 Mounting Components on Board

Once the parts are counted and identified, they can be mounted on the board and soldered in place. It is recommended that components be installed in the following sequence:—first, all discrete parts (capacitors and resistors); next, all integrated circuits; and last, crystal Y1. (A piece of double-sided foam tape is supplied with the kit and should be placed between the printed circuit board and the case of the crystal to prevent accidental grounding of the crystal.) This order of assembly allows the board to be relatively flat during all soldering operations. The layout of the printed circuit board and mounted components are shown in figure 2-1. The figure provides explicit detail of where each component goes. To guarantee a successful job, the "DOs" and "DON'Ts" listed below must be followed to the letter. Figure 2-2 is a schematic diagram of the assembled kit.

NOTES:

1. Check Device Data Sheets to verify pin-outs and Pin 1 designations for ICs,
2. Capacitors C9-C14 are optional (not supplied with kit). See Chapter 3.

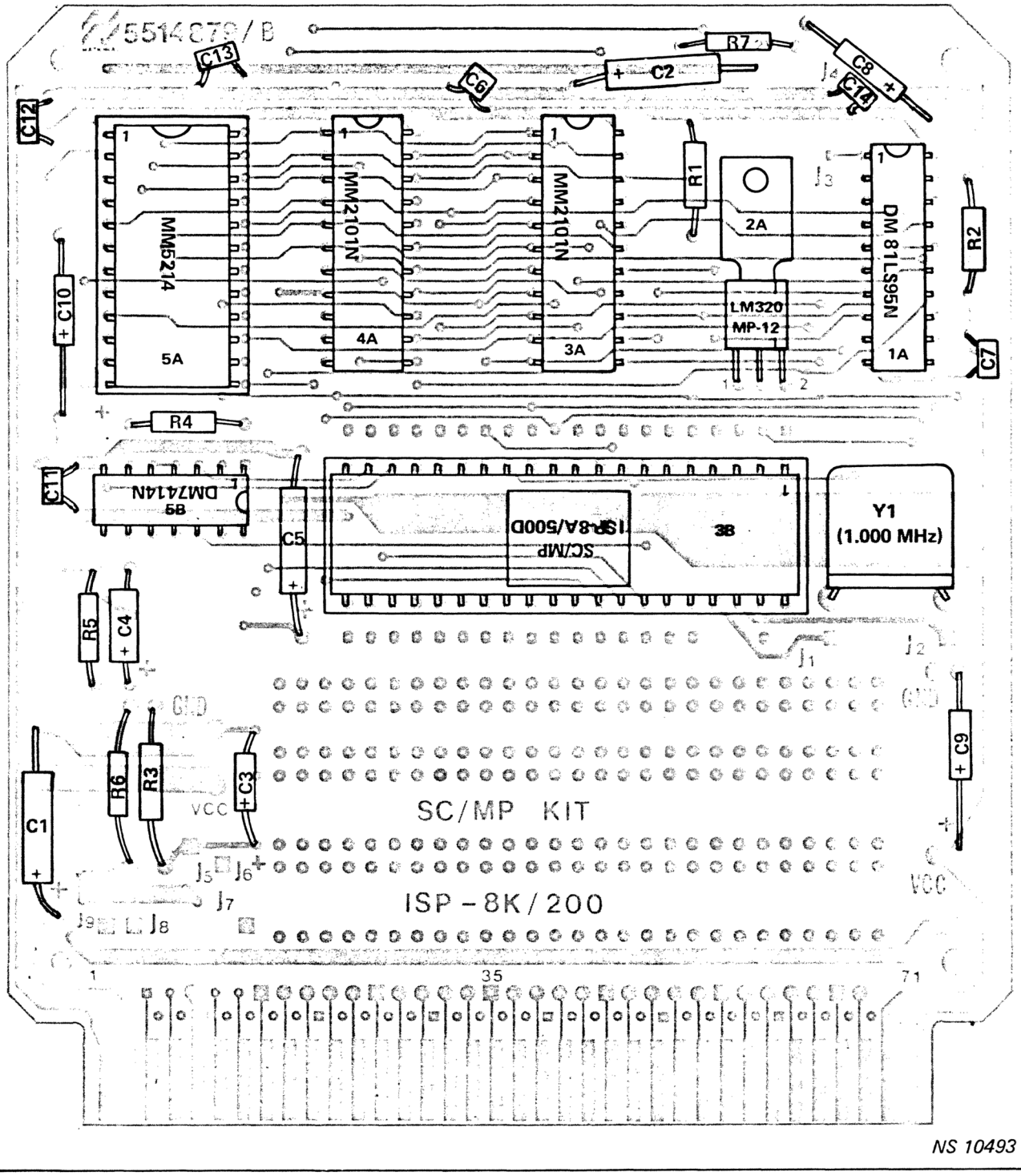


Figure 2-1. SC/MP Kit Component Locations

- DO make certain that polarized capacitors are properly mounted. If a "+" does not appear on the case, the lead with the solder "blob" is positive. Where applicable, the "+" symbol is shown in figure 2-1.

- DO NOT solder an integrated circuit until the pin orientation matches that shown in figure 2-1—Pin #1 goes in the hole with the square pad. Pin #1 can be identified usually by a recessed dot adjacent to this particular pin; there is no identifying mark for other pins.

CAUTION

ONCE AN INTEGRATED CIRCUIT IS SOLDERED INTO PLACE, IT IS VERY DIFFICULT TO DESOLDER THE COMPONENT WITHOUT PIN DAMAGE, BOARD DAMAGE, OR BOTH. REFER TO DEVICE DATA SHEETS; THEN CHECK AND RECHECK PIN ORIENTATION BEFORE YOU BEGIN SOLDERING.

NOTE

An alternate method of mounting the integrated circuits on the printed circuit board is to use sockets instead of soldering the ICs directly to the board. Sockets are available from a variety of sources to accommodate all standard ICs. The sockets must be the "solder-tail" type intended for installation on printed circuit boards (as opposed to wire-wrap boards). The sockets must be soldered to the printed circuit board in the same positions (and using the same techniques and precautions) as the ICs would have occupied; using sockets for mounting ICs permits the easy insertion, removal, and replacement of ICs without resoldering of connections. In some applications or environments, this may be a preferred method. Sockets for the SC/MP chip and for the ROM are included with the kit. The ROM socket lets you easily substitute your own ROM or PROM for the one containing KITBUG.

- DO NOT solder from the component side; once the leads are inserted in the proper holes (figure 2-1), turn the card over and fill the pad with molten solder. Then, neatly trim the excess lead length as flush as possible with the soldered connection. (Note: There is no need to trim the integrated circuit or socket leads.)

2.2.3 Board Cleanup and Pre-Power Verification

After all components are mounted and soldered in place, excess flux should be removed with any one of the cleaning agents called out in table 2-1. Visually examine each connection for solder bridges, cold joints, and so forth; if a connection is in doubt, use a continuity checker (must not exceed 1.5 volts) to test for opens and shorts. Perform a final check to ensure that all components are positioned properly as to polarity and pin orientation.

2.3 COMPONENT REPLACEMENT

A defective component can be replaced using the following procedure.

1. Cut leads as flush as possible on component side of board. Apply sufficient heat to melt solder and then, with appropriate tools (table 2-1), remove lead stubs and vacuum solder from connection pads.

2. Clean holes so that new component can be installed without the use of force; install replacement part in accordance with the procedures in paragraph 2.2.2. Clean and dress connections as indicated in paragraph 2.2.3.

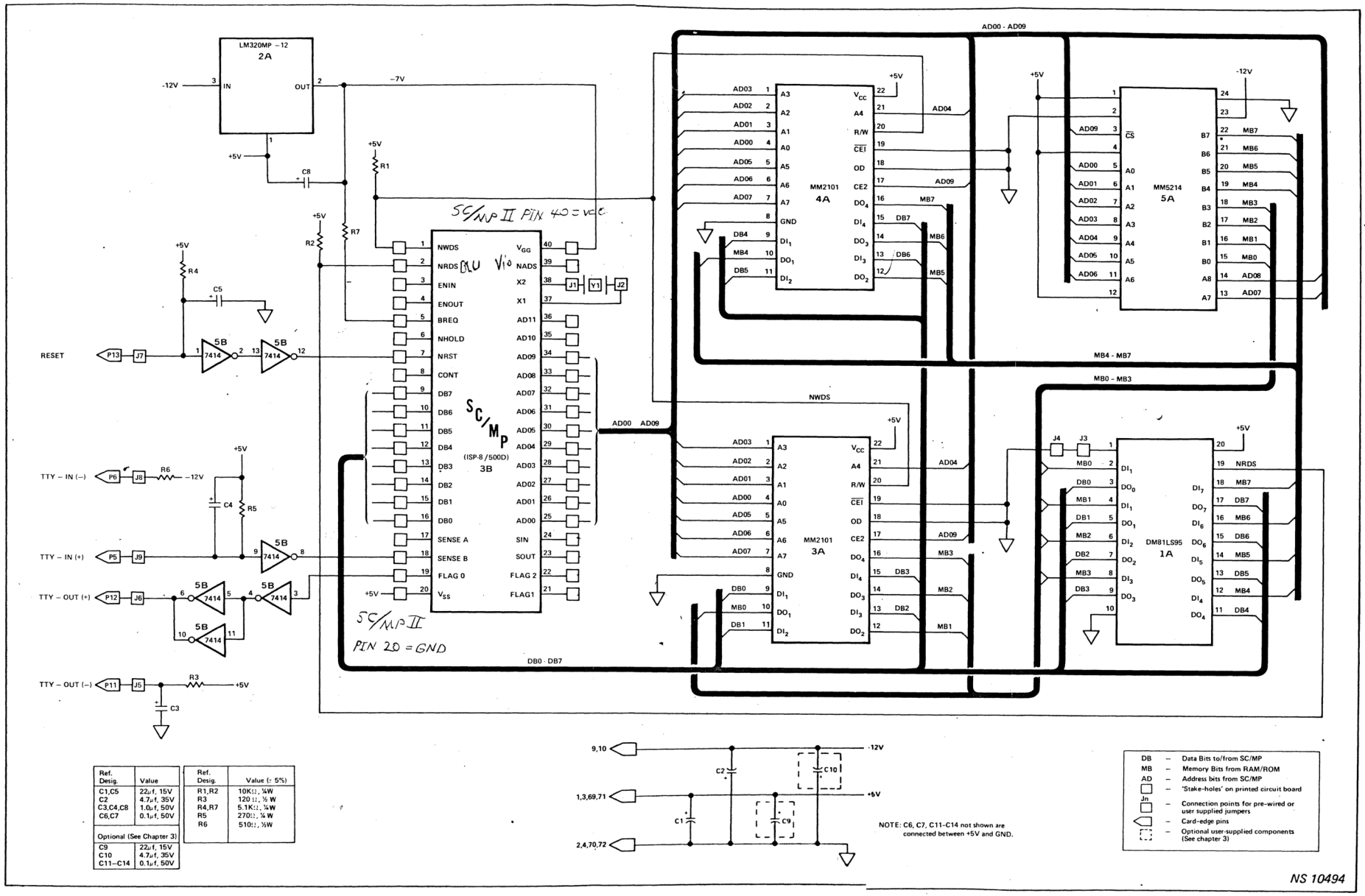
2.4 CONNECTING POWER

Edge-connector pins are provided on the printed circuit board for connection of input power. The +5V power must be connected to edge-pins 1, 3, 69, and 71. The -12V power must be connected to edge-pins 9 and 10. The recommended method of connecting power to the board is to use the standard 72-pin edge-card connector socket provided with the kit. If two separate supplies are used, both must be referenced to a common ground. The ground connections for the printed circuit board are edge-pins 2, 4, 70, and 72.

2.5 TELETYPE SETUP AND SYSTEM CONNECTION

The SC/MP Kit is designed to operate with a standard Teletype® model ASR 3320/JC or TU (with or without a paper tape reader/punch) without XON, XOFF, and with the automatic answerback option disabled.

The TTY must be set to operate in the full-duplex mode with a 20-milliampere current loop interface. Instructions for TTY setup and connection to the kit are provided below. Figure 2-3a is a top view of the TTY showing the location of the assemblies that are referred to in these



Ref. Desig.	Value
C1,C5	22μf, 15V
C2	4.7μf, 35V
C3,C4,C8	1.0μf, 50V
C6,C7	0.1μf, 50V
Optional (See Chapter 3)	
C9	22μf, 15V
C10	4.7μf, 35V
C11–C14	0.1μf, 50V

Ref. Desig.	Value (± 5%)
R1,R2	10KΩ, ¼W
R3	120 Ω, ½ W
R4,R7	5.1KΩ, ¼W
R5	270Ω, ¼ W
R6	510Ω, ½W

Figure 2-2. SC/MP Kit Schematic

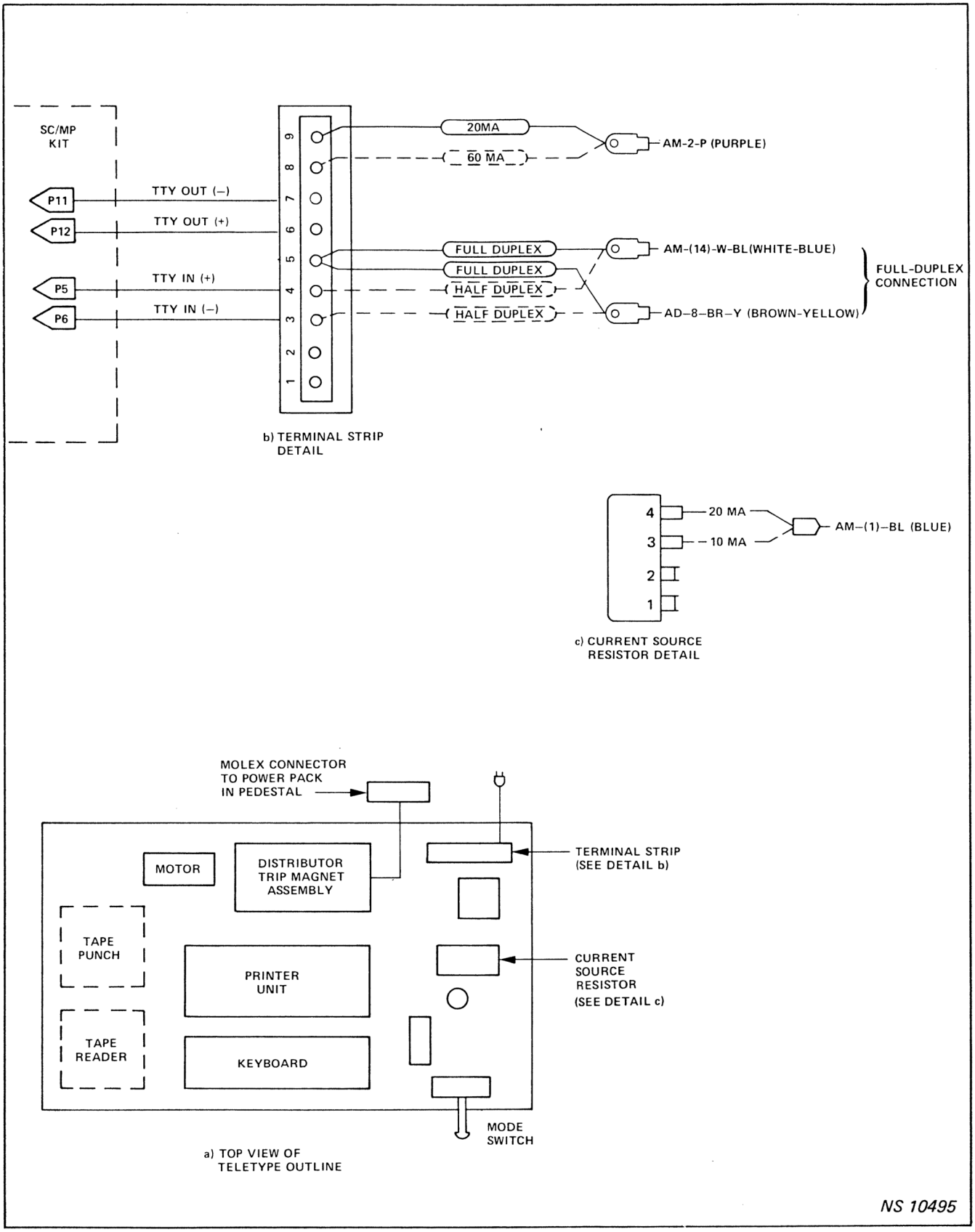


Figure 2-3. Teletype Connections

instructions. Figures 2-3b and 2-3c show the details of the terminal strip and current source resistor. In these figures, the dotted lines indicate the connections for half-duplex and 60-milliampere current loop operation. This is the configuration in which the TTY is normally shipped from Teletype Corporation. The solid lines indicate the desired connections for full-duplex and 20-milliampere current loop operation. Ensure that power is removed from the TTY before performing the following steps:*

1. To set TTY current source to 20 milliamperes, move blue wire from terminal 3 to terminal 4 of the current source resistor.

2. To set receive current to 20 milliamperes, move purple wire from pin 8 to pin 9 on the terminal strip located at rear of TTY.

3. To configure TTY for full-duplex, move white-blue wire from pin 4 to pin 5 on the terminal strip, and move brown-yellow wire from pin 3 to pin 5.

4. To disable the auto-answerback option, lift the print station paper cover and locate the cavity behind the keyboard. Directly beneath the carriage is a set of nine codebars. At the front of this assembly is a tie-bar. (See figure 2-4.) The auto-answerback is disabled by placing a clip over the tie-bar so that the third slot from the right is covered. On some models, one of these copper-colored clips already may be placed over the second slot; if so, move it to the third slot. If no clip is provided, it can be obtained from your local Teletype® dealer.

5. Connect TTYOUT (+) from kit edge-card pin 12 to pin 6 on the TTY terminal strip (figure 2-3b).

6. Connect TTYOUT (-) from kit edge-card pin 11 to pin 7 on the TTY terminal strip.

7. Connect TTYIN (+) from kit edge-card pin 5 to pin 4 on the TTY terminal strip.

8. Connect TTYIN (-) from kit edge-card pin 6 to pin 3 on the TTY terminal strip.

NOTE

Cable length from TTY to SC/MP Kit should not exceed 12 feet. Recommended cable type is standard twisted-pair, 22 AWG.

*If the TTY is obtained from National Semiconductor Corporation (order number IMP-00/810) steps 1-4 have already been accomplished.

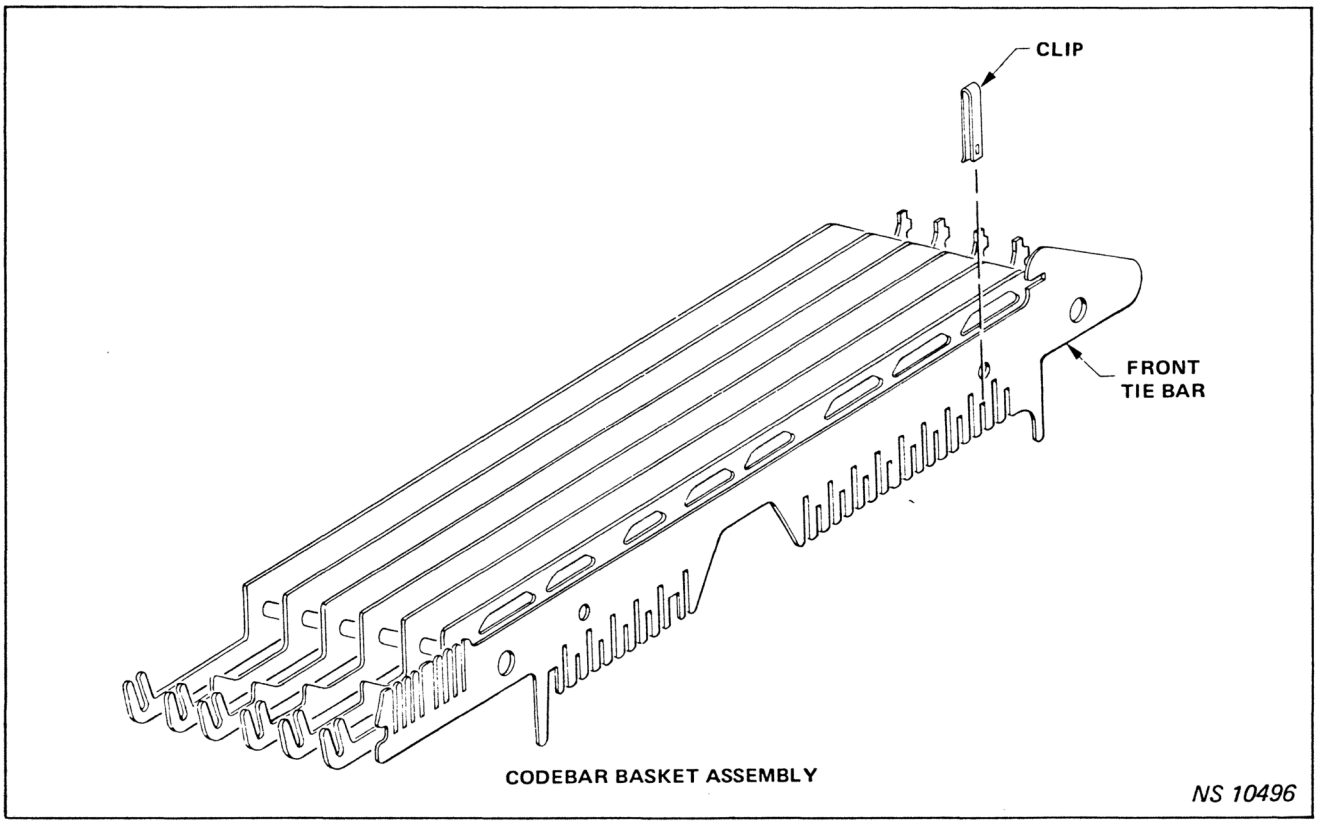


Figure 2-4. Disabling TTY Auto-Answerback

2.6 RESET SWITCH

The SC/MP Reset Signal (NRST) is available at card-edge pin 13 (and at J7) and allows the direct connection of a simple momentary-contact switch. All that is required to reset SC/MP is to ground pin 13 momentarily. A schematic representation of a reset switch is shown in figure 2-5.

2.7 SYSTEM START-UP

Once the TTY is set up and connected, the system is ready for startup and operation.

1. Apply power to the kit. If separate switches are used for the -12V and +5V supplies, turn the -12V supply on first and then, the +5V supply. This sequence ensures proper initialization of the SC/MP chip. If both supplies are operated by the same switch, it may be necessary to use the NRST signal to initialize SC/MP.

2. Turn the TTY mode switch (at the right front of the TTY) to LINE.

3. Press the Carriage Return key on the TTY. KITBUG will print a question mark and then a hyphen to indicate that it is awaiting a command. See chapter 4 for KITBUG operating instructions.

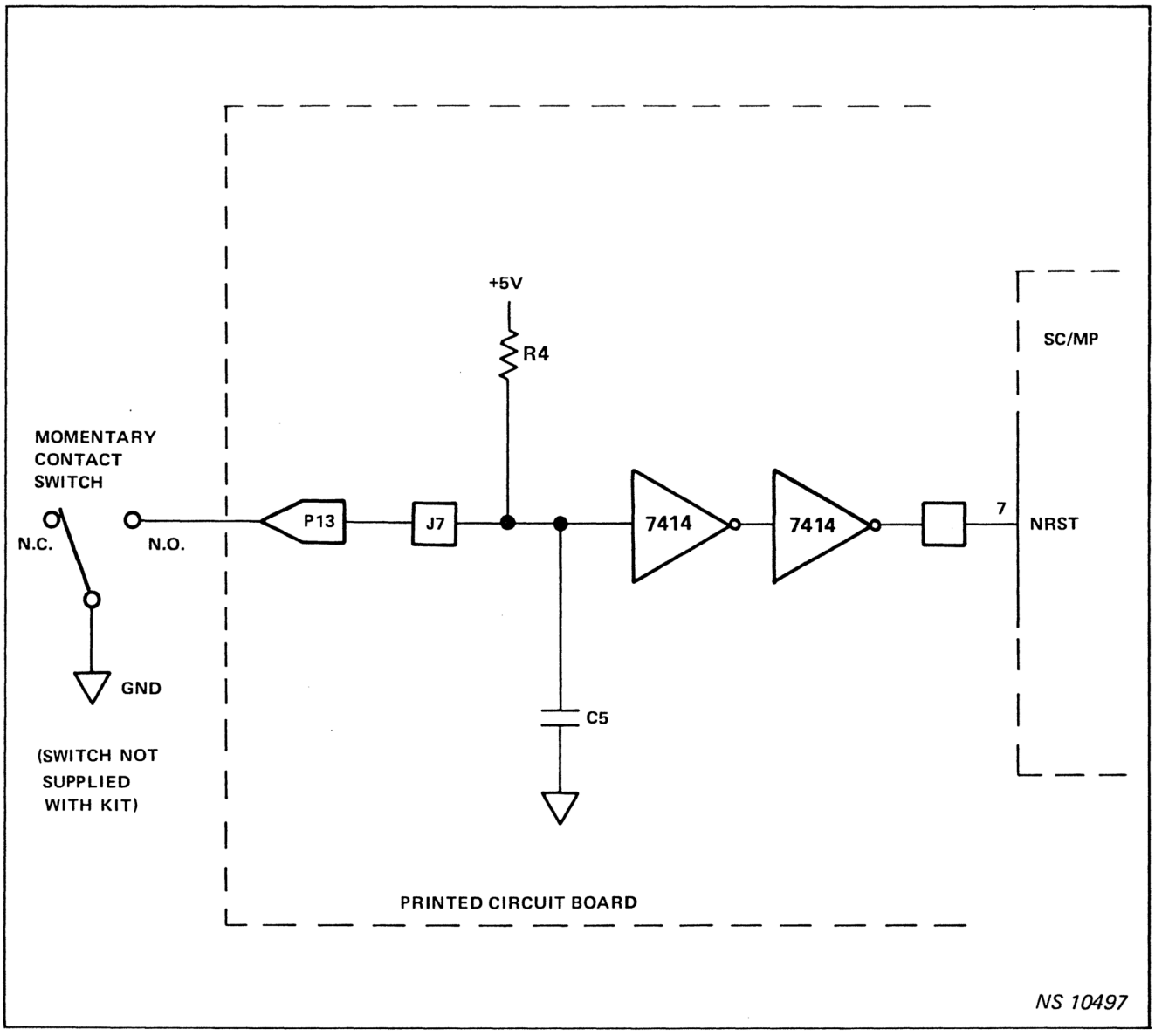


Figure 2-5. Reset Switch

Chapter 3

KIT EXPANSION GUIDELINES

3.0 INTRODUCTION

Although the SC/MP Kit is not intended to serve as the basis of a large or complex system, some expansion of capabilities can be accomplished. The paragraphs that follow describe some of the considerations and precautions that must be observed when expanding the kit.

Space is provided on the printed circuit board for additional components. The plated-through holes in the unused portion of the board are spaced to accommodate most standard sizes of ICs. It may be advantageous to use sockets as described in section 2.2.2 to mount additional ICs.

3.1 POWER AND SIGNAL LOADING

As additional components are added to the kit, power requirements increase accordingly and must be met by the user. The requirements for the basic kit configuration are defined in section 1.1.3.

The SC/MP signals are available at the plated-through "stake-holes" located adjacent to the holes in which the SC/MP chip is mounted. If these signals are used in an expanded configuration of the kit, care must be taken that the loading and fanout capabilities of the SC/MP chip are not exceeded. The electrical characteristics of the SC/MP signals are defined in the SC/MP Data Sheet. Typically, where more than one TTL load must be serviced, it will be necessary to provide buffering for the SC/MP signals.

3.2 DECOUPLING CAPACITORS

As components are added to the kit it may be necessary to provide additional decoupling capacitors. Mounting holes for these capacitors are provided in the power planes at intervals along the periphery of the printed circuit board.

The values and electrical locations for these optional user-supplied capacitors (C9-C14) are shown on the kit schematic (figure 2-2). The corresponding physical locations are shown in figure 2-1.

3.3 ADDRESS CONSIDERATIONS

If additional memory is incorporated into the kit, care must be taken to ensure that there is no conflict with the addresses assigned to the existing RAM and ROM. Section 1.1.2 and table 1-1 describe the existing addressing scheme. Typically, it will be necessary to provide some address decoding circuitry to allow conflict-free operation of additional memory.

3.4 EXTERNAL CONTROL OF DATA BUFFER

The DM81LS95 buffer supplied with the kit has two control inputs. In the standard kit configuration, the Read Data Strobe (NRDS) Signal from SC/MP is used as one control input (pin 19 on the DM81LS95). The second control input (pin 1) is continuously enabled by a connection to ground.

If a particular application should require external control of the data buffer, the unused control input can be enabled by cutting the trace between J4 and J3 on the printed circuit board. This cuts the connection to ground, and an external signal can then be applied to J3. Refer to the data sheet for the DM81LS95 for details on the use and the effect of the control inputs.

Chapter 4

USING KITBUG

4.0 INTRODUCTION

The preceding chapters told you how to put your SC/MP Kit together. This chapter will tell you how to put it to work. What kind of work? That's up to you—SC/MP will do whatever it has been instructed to do. The instructions are provided by you in the form of a program that you have put into read/write memory. All you need now is some method for getting your program into memory. It would also be helpful if there were some convenient way of checking out your program to make sure that it is doing what you intended it to do. That's where KITBUG comes in—and that's what this chapter is about.

4.1 THE KITBUG PROGRAM

The KITBUG program has been encoded into Read-Only Memory (ROM) devices that are supplied with each SC/MP Kit. KITBUG, as its name implies, is intended primarily to assist you in the checkout of your programs. To accomplish this, KITBUG enables you to perform the following operations.

- Initiate execution of your program at any point desired.
- Establish breakpoints within your program to allow execution of selected program segments.
- Examine the contents of memory and SC/MP registers to determine if your program is producing the expected results.
- Change the contents of any memory location to make corrections to your program.
- Change the contents of the SC/MP registers to set up conditions your program requires.

The KITBUG Program is used to enter your program into memory via the keyboard of a TTY. Part of the KITBUG program consists of the input/output subroutines required to allow communications between SC/MP and a TTY.

NOTE

A complete listing for KITBUG is provided in appendix B.

4.2 HOW KITBUG WORKS

The KITBUG program is located at the bottom of memory (beginning at location 000). Thus, whenever the SC/MP Kit is powered up or reset, control is automatically given to KITBUG. Since only one program at a time can be run in SC/MP, the KITBUG Program must provide some orderly method of transferring control from itself to your program and then back again. To do this, KITBUG uses an area of read/write memory to store information about the operating requirements for your program. KITBUG also uses SC/MP Pointer Register P3 to store a pointer that your program can use to return control to KITBUG. Thus, when control is transferred to your program from KITBUG, the SC/MP registers, such as the Program Counter and the Pointer Registers (P1 and P2) are set to the initial values that you specify as being required by your program. And, when your program transfers control back to KITBUG, the current contents of the SC/MP registers are copied out to memory. This provides an image of the state of the SC/MP registers at the time of the transfer of control. Using KITBUG, you can then inspect this image and the memory locations of your program to check the operation of your program.

The memory locations used to store the register-image are listed in table 4-1. These locations can be examined using the Type command and can be set to any values you require using the Modify command. Note that PC, P1 and P2 require two consecutive 8-bit memory locations since they are 16-bit registers. P3 is not referenced since it is used by KITBUG.

Table 4-1. Memory Locations Used for SC/MP Register Image

Memory Location* (Hex)	Register
0FF7	PC: Program Counter (bits 8-15)
0FF8	PC: Program Counter (bits 0-7)
0FF9	P1: Pointer Register 1 (bits 8-15)
0FFA	P1: Pointer Register 1 (bits 0-7)
0FFB	P2: Pointer Register 2 (bits 8-15)
0FFC	P2: Pointer Register 2 (bits 0-7)
0FFD	AC: Accumulator
0FFE	EX: Extension Register
0FFF	SR: Status Register

*See footnote on following page.

4.3 CONFIGURATION REQUIREMENTS

As mentioned in the preceding paragraph, the ROM containing the KITBUG Program is located at the bottom of memory.* The KITBUG Program is 512 bytes long and thus occupies the memory range from $0000_{(16)}$ through $01FF_{(16)}$.

KITBUG uses 20 bytes at the upper boundary of read/write memory to maintain the image of the SC/MP registers and for temporary storage of internally required information. The 256 bytes of RAM supplied with the kit must therefore be located in the address range of $0F00_{(16)}$ through $0FFF_{(16)}$, and locations $0FEC_{(16)}$ through $0FFF_{(16)}$ must be reserved for use by KITBUG. One final consideration, if you want your program to be able to return control to KITBUG, *Pointer Register P3 should not be used by your program since P3 is used by KITBUG to store the pointer that allows transfer of control back to KITBUG.*

4.4 COMMUNICATING WITH KITBUG

The TTY provides the communication link between you and the KITBUG Program. Whenever control is transferred to KITBUG (by your program, at powerup, or by reset), a hyphen (-) is printed at the TTY. The hyphen is a "prompt character" and indicates that KITBUG is waiting for you to enter a command via the TTY keyboard. The commands recognized by KITBUG are described in the paragraphs that follow. Each command that you enter must be terminated by pressing the carriage return key. In the descriptions that follow, the symbol (CR) is used to represent pressing the carriage return key.

4.4.1 Format of Entries

The KITBUG commands consist of a single letter (T, M, or G) followed by a three- or four-digit hexadecimal number that represents a memory address. The valid hexadecimal digits are 0 through 9 and A, B, C, D, E, and F. As a command is being entered, KITBUG checks each character to ensure that it is a legal character. A legal character is defined as one of the three command letters or a hexadecimal digit; furthermore, a character must be entered in appropriate sequence. For example, if the first character entered in response to the prompt character is '9', it is considered illegal since KITBUG requires that one of the three command letters (T, M, or G) be entered at that position. Note that a "space" is also an illegal character.

When an illegal character is detected, KITBUG immediately prints a question mark (?) at the TTY and then prompts for a new command. You can use this feature to abort a command. Simply type any illegal character and you are given a fresh start.

When you are entering the numeric values required by the KITBUG commands, KITBUG uses only the number of digits that it requires. For example, if 12340124 were typed as an address, the value 0124 (the last four digits entered) would be accepted by KITBUG. Therefore, if you make a mistake during an entry, simply continue and type the correct information on the same line; KITBUG ignores the erroneous part of the entry.

One final note on entry formats. When entering numeric values, leading zeros can be omitted. Thus, if you enter 124 to specify an address, KITBUG supplies a leading zero and treats the entry as 0124. Note that to correct an error, as explained in the preceding paragraph, the leading zero(s) must be entered because KITBUG uses the last *four* digits entered.

4.5 THE KITBUG COMMANDS

KITBUG recognizes three commands: T (Type), M (Modify), and G (Go). Additionally, Modify can be used to simulate a fourth command (Halt). Descriptions of each command and examples of their use are provided in the paragraphs that follow.

4.5.1 The Type Command

The Type command allows you to examine the contents of any location in memory by causing the contents of the specified locations to be printed at the TTY. The contents of memory are not altered. The format for the command is

T⟨address⟩

where ⟨address⟩ is a hexadecimal number indicating the address of the memory location from which the printout is to begin.

The contents of each memory location, beginning at ⟨address⟩ is printed on a separate line preceded by the address for that location.

**The addresses defined in table 4-1 and paragraph 4.3 are the actual values generated and used by KITBUG. However, because not all of the 16 address bits are physically used in the kit, a memory location can be specified by several different address values. See section 1.1.2 for a discussion of this characteristic.*

Example: (Note: user entries are underlined)

```
-T330 (CR)
address   0330  AB
          0331  26   } contents
          0332  0C
          0333  A2   (BREAK Key pressed)
-
```

The printout continues until an input from the TTY keyboard is recognized. When any keyboard input is detected during printout, the Type command is aborted and the prompt character (-) is printed. (It may be necessary to press repeatedly a key before it is detected by KITBUG.) KITBUG is then ready to accept another command.

4.5.2 The Modify Command

The Modify command allows you to scan the contents of memory and selectively modify the contents of any location. The format for the command is

M ⟨address⟩

where ⟨address⟩ is a hexadecimal number indicating the memory location where the scanning is to begin. As with the Type command, the contents of each memory location, beginning at ⟨address⟩ will be printed preceded by the address of the memory location. However, after the contents of each location are printed, KITBUG waits for you to enter a new value (a 2-digit hexadecimal number) to replace the current contents of that location. To skip the location and leave it unchanged, simply press Carriage Return; KITBUG then prints the contents of the next location.

Example:

```
                                  Comments
-MFFD (CR)
0FFD 0A 09 (CR)      (change contents of AC to 09)
0FFE 14 (CR)         (leave contents of EX unchanged)
0FFF 30 3132 (CR)    (change contents of SR to 32)
0000 08 X ?          (terminate the command)
-
```

The command is terminated by entering any illegal character; in the above example, the illegal letter X is used. KITBUG prints a question mark, indicating that an illegal character was entered, and then prompts for the next command. Note that the memory locations being scanned and modified in this example are those locations where the image of SC/MP registers are maintained. In the example the contents of location 0FFF were changed from 30 to 32. The value 31 is ignored by KITBUG since it uses only the number of digits it requires–in this case two (see section 4.4.1).

NOTE

After the contents of memory location 0FFF were modified in the example above, note that the address of the next memory location is 0000 instead of 1000. This is due to the "wraparound" addressing characteristics of SC/MP and occurs on all operations–KITBUG commands and user programs alike. See the SC/MP Technical Description for a discussion of this characteristic.

4.5.3 The Go Command

The Go command transfers control from KITBUG to your program. The format for the command merely consists of the letter 'G', followed by carriage return ((CR)).

When the command is executed, KITBUG loads the SC/MP registers with the values stored in the register-image area of memory. Thus, control is transferred to your program beginning at the point indicated by the contents of memory locations FF7 and FF8 (Program Counter), and the other SC/MP registers are set to whatever initial values your program requires.

You can, therefore, begin execution at any point in your program by using the Modify command to set the contents of locations FF7 and FF8 to the desired starting point and then by using the Go command, initiate execution of your program. See paragraph 4.6 for an example of the use of the Go command.

Your program will then have control of SC/MP until you force control back to KITBUG. This can be done by using the reset signal, by removing and then re-applying power, or by providing a special instruction within your program to transfer control back to KITBUG. Using the reset signal causes all SC/MP registers to be cleared. Removing and re-applying power causes the registers to be cleared and also results in the loss of data stored in read/write memory. Using the special instruction described in the following paragraphs effects transfer of control without loss of information.

4.6 TRANSFERRING CONTROL BACK TO KITBUG

When the Go command is given to KITBUG, control is transferred to your program and remains there. Since part of a program debugging procedure usually involves running selected segments of your program, you will usually want to be able to transfer control back to KITBUG after the selected segment has been run. This can be accomplished by inserting a special instruction in your program at the point where you want control returned to KITBUG.

When KITBUG executes the Go command, it stores an address in SC/MP Pointer Register 3 (P3) that indicates the entry point for KITBUG. By inserting an XPPC P3 (Exchange Program Counter with P3—opcode 3F) Instruction at the desired point in your program, control can be returned automatically to KITBUG when that point is reached. Care must be taken that your program does not alter the contents of P3 since that would break the link back to KITBUG.

The following example causes the execution of a program segment that begins at address 0F50 and ends at 0F80. When the segment has been executed, control is returned to KITBUG and the Type command could then be used to examine memory to determine the results obtained.

Example:

```
-MF81 (CR)
0F81 23 3F (CR)      Insert XPPC P3 (3F) instruction
                     to provide return to KITBUG
0F82 06 X ?
-MFF7 (CR)
0FF7 0F (CR)         }
                     } Set Program Counter to 0F50
0FF8 1A 50 (CR)      }
0FF9 48 X ?
-G (CR)              Transfer control to your program
-                    Program segment completed
```

Note that if location 0F81 had previously contained an instruction that was part of your program, it would be destroyed when the XPPC P3 instruction was inserted.

NOTE

In many applications it may be desirable to have your program run continuously in a loop. In this case, it may be easier to use the Reset Signal to return control to KITBUG rather than using the XPPC instruction. The only disadvantage of using the Reset Signal is that it causes all SC/MP registers to be cleared to zero: thus, the register image maintained in RAM will also contain all zeros after KITBUG has resumed control.

Appendix A

SC/MP KIT PARTS LIST

Item	Description	Reference Designation	Quantity
1	Printed Circuit Board	-	1
2	I.C. ISP-8A/500D (SC/MP Chip)	3B	1
3	I.C. MM5214 (ROM) See Note	5A	1
4	I.C. MM2101N (RAM)	3A, 4A	2
5	I.C. DM81LS95N (8-bit Buffer)	1A	1
6	I.C. DM7414N (Hex-Schmitt Trigger)	5B	1
7	I.C. Socket; 40-pin (for SC/MP)	3B	1
8	I.C. Socket; 24-pin (for ROM)	5A	1
9	LM320MP - 12, voltage regulator	2A	1
10	Crystal, 1.000 MHz	Y1	1
11	Capacitor, 22 μf, 15V	C1, C5	2
12	Capacitor, 4.7 μf, 35V	C2	1
13	Capacitor, 1.0 μf, 50V	C3, C4, C8	3
14	Capacitor, 0.1 μf, 50V	C6, C7	2
15	Resistor, 10K Ω, 1/4W, 5%	R1, R2	2
16	Resistor, 120 Ω, 1/2W, 5%	R3	1
17	Resistor, 5.1K Ω, 1/4W, 5%	R4, R7	2
18	Resistor, 270 Ω, 1/4W, 5%	R5	1
19	Resistor, 510 Ω, 1/2W, 5%	R6	1
20	Foam Tape (for mounting crystal)	-	-
21	72-pin Card-Edge Connector Socket	-	1

Note: ROM contains the KITBUG Program. In some kits, the MM5244 is substituted for MM5214. These two devices are functionally equivalent and pin compatible.

Appendix B

KITBUG PROGRAM LISTING

```
                        .TITLE  KITBUG,' P00937A  12/1/75'
                ;*********************************************************************
                ;*
                ;*
                ;*
                ;*
                ;*
                ;*
                ;*
                ;*
                ;*
                ;*********************************************************************
                ;
        0001    P1      =       1
        0002    P2      =       2
        0003    P3      =       3
                ;
        FFFF    EXOFF   =       -1


                        .PAGE   'STACK ASSIGNMENTS'
                        .LOCAL
                ;
                ; FIXED STACK ASSIGNMENTS
                ;
0000                    .=0FFF
0FFF            STACK:
        0000    SR      =       .-STACK
0FFF                    .=.-1
        FFFF    EX      =       .-STACK
0FFE                    .=.-1
        FFFE    AC      =       .-STACK
0FFD                    .=.-2
        FFFC    PT2     =       .-STACK
0FFB                    .=.-2
        FFFA    PT1     =       .-STACK
0FF9                    .=.-2
        FFF8    PC      =       .-STACK
                ;
        0FF6    P2ADR   =       .-1


                        .PAGE   'DEBUG ENTRY AND EXIT'
                        .LOCAL
                ;
                ; ON A SOFTWARE HALT, HARDWARE USES THE FOLLOWING WORDS
                ; TO SAVE THE ENVIROMENT.
                ;
0FF7                    .=0
0000 08                 NOP
0001 901D       START:  JMP     ENTER
                ;
                ; DEBUG EXIT - RESTORE ENVIROMENT AND GO.
                ;
0003 C0FA       EXIT:   LD      STACK+EX        ; RESTORE E REG
0005 01                 XAE
0006 C0F2               LD      STACK+PT1       ; RESTORE P1
0008 35                 XPAH    P1
0009 C0F0               LD      STACK+PT1+1
000B 31                 XPAL    P1
000C C0EE               LD      STACK+PT2       ; RESTORE P2
000E 36                 XPAH    P2
000F C0EC               LD      STACK+PT2+1
```

```
 61 0011 32            XPAL    P2
 62 0012 C0E4          LD      STACK+PC        ; PUT DESIRED PC IN P3
 63 0014 37            XPAH    P3
 64 0015 C0E2          LD      STACK+PC+1
 65 0017 33            XPAL    P3
 66 0018 C7FF          LD      @EXOFF(P3)      ; ADD EXIT OFFSET TO PC
 67 001A C0E4          LD      STACK+SR        ; RESTORE SR
 68 001C 07            CAS
 69 001D C0DF          LD      STACK+AC
 70 001F 3F            XPPC    P3
 71             ;
 72             ; DEBUG ENTRY POINT
 73             ;
 74 0020 C8DC  ENTER:  ST      STACK+AC
 75 0022 06            CSA
 76 0023 C8DB          ST      STACK+SR
 77 0025 01            XAE                     ; SAVE EXTENSION REGISTER
 78 0026 C8D7          ST      STACK+EX
 79 0028 36            XPAH    P2              ;   POINTER
 80 0029 C8D1          ST      STACK+PT2
 81 002B 32            XPAL    P2
 82 002C C8CF          ST      STACK+PT2+1
 83 002E 35            XPAH    P1              ;   STACK
 84 002F C8C9          ST      STACK+PT1
 85 0031 31            XPAL    P1
 86 0032 C8C7          ST      STACK+PT1+1
 87 0034 37            XPAH    P3
 88 0035 C8C1          ST      STACK+PC
 89 0037 33            XPAL    P3
 90 0038 C8BF          ST      STACK+PC+1


 91                    .PAGE   'MAIN COMMAND LOOP'
 92                    .LOCAL
 93             ;
 94             ; THIS CODE INITIALIZES POINTER REGISTERS AND
 95             ; PROMPTS FOR AND GETS THE NEXT COMMAND.
 96             ;
 97             ; ON EXIT, E HOLDS THE COMMAND CHARACTER
 98             ;
 99 003A C4F6  CMDLP:  LDI     L(P2ADR)
100 003C 32            XPAL    P2
101 003D C40F          LDI     H(P2ADR)
102 003F 36            XPAH    P2
103 0040 C401          LDI     H(PUTC)         ; PRINT CR-LF
104 0042 37            XPAH    P3
105 0043 C4C4          LDI     L(PUTC)-1
106 0045 33            XPAL    P3
107 0046 C40D          LDI     0D
108 0048 3F            XPPC    P3              ; PRINT CR
109 0049 C40A          LDI     0A
110 004B 3F            XPPC    P3              ; PRINT LF
111 004C C42D          LDI     '-'
112 004E 3F            XPPC    P3
113 004F C401          JS      P3,GECO         ; GET COMMAND CHARACTER
    0051 37C4
    0053 8533
    0055 3F


114                    .PAGE   'GO'
115                    .LOCAL
116             ;
117             ; RESTORE MACHINE STATE AND TRANSFER CONTROL
118             ; TO SPECIFIED ADDRESS.
119             ;
120             ; G ADDRESS
121             ;
```

```
0056 40     GO:     LDE
0057 E447           XRI     'G'
0059 9C07           JNZ     $SKIP
005B 3F             XPPC    P3              ; CALL GECO
005C E40D           XRI     0D
005E 98A3           JZ      EXIT
0060 906A           JMP     ERROR
0062        $SKIP:


                    .PAGE   'TYPE'
                    .LOCAL
                ;
                ; TYPE OR MODIFY MEMORY.
                ;
0062 40     TYPE:   LDE                     ; CHECK FOR TYPE COMMAND, IF
0063 E454           XRI     'T'             ;   NOT 'T', SKIP COMMAND.
0065 9809           JZ      $2
0067 40     MOD:    LDE
0068 E44D           XRI     'M'
006A 9C60           JNZ     $SKIP
006C C400           LDI     0
006E 9002           JMP     $1
0070 C401   $2:     LDI     1
0072 CEFF   $1:     ST      @-1(P2)         ; SAVE FLAG FOR TYPE OR MODIFY
0074 C400           JS      P3,GHEX         ; GET ADDRESS
0076 37C4
0078 DF33
007A 3F
007B E40D           XRI     0D              ; CHECK TERMINATOR
007D 9C4D           JNZ     ERROR
007F C601           LD      @1(P2)          ; PUT STARTING ADDRESS IN STACK
0081 35             XPAH    P1
0082 C601           LD      @1(P2)
0084 31             XPAL    P1
0085 C401   $4:     LDI     H(PUTC)         ; PRINT CR-LF
0087 37             XPAH    P3
0088 C4C4           LDI     L(PUTC)-1
008A 33             XPAL    P3
008B C40D           LDI     0D
008D 3F             XPPC    P3              ; PRINT CR
008E C40A           LDI     0A
0090 3F             XPPC    P3              ; PRINT LF
0091 35             XPAH    P1              ; PRINT HIGH BYTE
0092 01             XAE                     ; READ AND RESTORE BYTE FROM P1
0093 40             LDE
0094 35             XPAH    P1
0095 C401           LDI     H(PHEX2)
0097 37             XPAH    P3
0098 C443           LDI     L(PHEX2)-1
009A 33             XPAL    P3
009B 40             LDE
009C 3F             XPPC    P3              ; CALL PHEX2
009D 31             XPAL    P1              ; PRINT LOW BYTE
009E 01             XAE
009F 40             LDE
00A0 31             XPAL    P1
00A1 40             LDE
00A2 3F             XPPC    P3              ; CALL PHEX1
00A3 C501           LD      @1(P1)
00A5 3F             XPPC    P3              ; PRINT 2-DIGIT HEX FOLLOWED BY
                                            ;   BLANK  (PHEX1)
00A6 C200           LD      (P2)            ; CHECK TYPE OR MODIFY FLAG
00A8 9CDB           JNZ     $4
00AA C401           JS      P3,GECO
00AC 37C4
00AE 8533
00B0 3F
```

```
182 00B1 E40D          XRI     0D
183 00B3 98D0          JZ      $4
184 00B5 E415          XRI     015             ; 0D XOR 018 (CAN)
185 00B7 9881  LOOP1:  JZ      CMDLP
186 00B9 C400          JS      P3,GHEX2
    00BB 37C4
    00BD DB33
    00BF 3F
187 00C0 E40D          XRI     0D
188 00C2 9C08          JNZ     ERROR
189 00C4 C601          LD      @1(P2)
190 00C6 C601          LD      @1(P2)
191 00C8 C9FF          ST      -1(P1)
192 00CA 90B9          JMP     $4
193 00CC       $SKIP:


194                    .PAGE   'ERROR PROCESSING'
195                    .LOCAL
196            ;
197            ; PRINT CARRAIGE RETURN , LINE FEED AND LOOP
198            ; TO THE TOP OF THE COMMAND LOOP.
199            ;
200 00CC C401  ERROR:  LDI     H(PUTC)         ; PRINT LINE FEED
201 00CE 37            XPAH    P3
202 00CF C4C4          LDI     L(PUTC)-1
203 00D1 33            XPAL    P3
204 00D2 C40A          LDI     0A
205 00D4 3F            XPPC    P3
206 00D5 C43F          LDI     '?'
207 00D7 3F            XPPC    P3
208 00D8 C400          LDI     0
209 00DA 90DB          JMP     LOOP1


210                    .PAGE     'HEX NUMBER INPUT'
211                    .LOCAL
212            ;
213            ; GHEX GETS A 16-BIT VALUE AND PUSHES IT TO THE STACK.
214            ; GHEX2 ASSUMES THE FIRST CHAR IS IN THE E REGISTER.
215            ; ONLY THE LAST 4 INPUT DIGITS ARE SAVED.
216            ;
217            ; RETURNS VALUE IN TOP 2 WORDS OF STACK AND TERMINATOR
218            ; IN THE AC AND EX REGISTERS.
219            ;
220 00DC C401  GHEX2:  LDI     1
221 00DE 9002          JMP     $6
222 00E0 C400  GHEX:   LDI     0               ; RESET GHEX2 FLAG
223 00E2 CAFB  $6:     ST      -5(P2)
224 00E4 C485          LDI     L(GECO)-1       ; SAVE RETURN ADDRESS AND SET UP
225 00E6 33            XPAL    P3              ;   TO GECO
226 00E7 CEFD          ST      @-3(P2)         ; STORE RETURN ADDRESS TO LEAVE ROOM
227 00E9 C401          LDI     H(GECO)         ;   FOR RESULT
228 00EB 37            XPAH    P3
229 00EC CEFF          ST      @-1(P2)
230 00EE C2FF          LD      -1(P2)
231 00F0 9C01          JNZ     $1
232 00F2 3F            XPPC    P3
233 00F3 C400  $1:     LDI     0               ; INITIALIZE RESULT TO 0
234 00F5 CA03          ST      3(P2)
235 00F7 CA02          ST      2(P2)
236 00F9 40    $LOOP:  LDE
237 00FA 03            SCL
238 00FB FC3A          CAI     '9'+1           ; CHECK FOR 0-9
239 00FD 940F          JP      $2              ; NOT 0-9, TOO LARGE
240 00FF 03            SCL
241 0100 FCF6          CAI     '0'-'9'-1       ; CHECK FOR 0-9
242 0102 9419          JP      $3              ; IF POSITIVE, NUMBER IS
```

```
                                                    ;   IN RANGE AND CONVERTED.
0104 C601  $RET:    LD      @1(P2)                  ; NUMBER IS NOT A HEX DIGIT,
0106 37             XPAH    P3                      ;   RETURN
0107 C601           LD      @1(P2)
0109 33             XPAL    P3
010A 40             LDE
010B 3F             XPPC    P3
010C 90D2           JMP     GHEX
010E 03     $2:     SCL
010F FC0D           CAI     'F'+1-'9'-1             ; CHECK FOR DIGITS A-F.
0111 94F1           JP      $RET                    ; NUMBER TOO LARGE
0113 03             SCL
0114 FCFA           CAI     'A'-'F'-1
0116 9402           JP      $4                      ; DIGIT BETWEEN A&F
0118 90EA           JMP     $RET
011A 02     $4:     CCL
011B F40A           ADI     10                      ; ADJUST DIGIT VALUE FOR 10-16
011D CAFF   $3:     ST      -1(P2)                  ; SAVE ADJUSTED DIGIT
011F C404           LDI     4                       ; SET UP BIT COUNTER FOR
0121 CAFE           ST      -2(P2)                  ;   SHIFT.
0123 02     $5:     CCL                             ; SHIFT HEX DIGIT LEFT ONE
0124 C203           LD      3(P2)                   ;   DIGIT, ONE BIT EACH
0126 F203           ADD     3(P2)                   ;   TIME THROUGH LOOP.
0128 CA03           ST      3(P2)
012A C202           LD      2(P2)
012C F202           ADD     2(P2)
012E CA02           ST      2(P2)
0130 BAFE           DLD     -2(P2)
0132 9CEF           JNZ     $5
0134 02             CCL
0135 C203           LD      3(P2)                   ; ADD CURRENT DIGIT INTO
0137 F2FF           ADD     -1(P2)                  ;   NUMBER
0139 CA03           ST      3(P2)
013B 3F             XPPC    P3                      ; GET NEXT CHAR
013C 90BB           JMP     $LOOP                   ; AND LOOP


                    .PAGE   'HEX NUMBER OUTPUT'
                    .LOCAL
                ;
                ; PRINT HEX NUMBER WITH TRAILING BLANK (PHEX1) OR
                ; WITHOUT IT (PHEX2).  NUMBER TO BE PRINTED IS
                ; IN AC.
                ;
013E CEFF   PHEX1:  ST      @-1(P2)                 ; SAVE AC
0140 C420           LDI     020                     ; SET FLAG TO PRINT BLANK AFTER
0142 9004           JMP     $1                      ;   NUMBER
0144 CEFF   PHEX2:  ST      @-1(P2)                 ; SAVE AC
0146 C400           LDI     0                       ; CLEAR FLAG TO PRINT BLANK
0148 CEFF   $1:     ST      @-1(P2)                 ;   AFTER NUMBER
014A C4C4           LDI     L(PUTC)-1               ; LOAD ADDRESS OF PUTC TO P3
014C 33             XPAL    P3                      ;   AND SAVE RETURN ADDRESS
014D CEFF           ST      @-1(P2)
014F C401           LDI     H(PUTC)
0151 37             XPAH    P3
0152 CEFF           ST      @-1(P2)
0154 C402           LDI     2                       ; SET FLAG FOR 1ST NUMBER
0156 CEFF           ST      @-1(P2)
0158 C204           LD      4(P2)                   ; GET ORIGINAL VALUE
015A 1C             SR                              ; SHIFT TO LOW 4 BITS
015B 1C             SR
015C 1C             SR
015D 1C             SR
015E 02     $5:     CCL                             ; CONVERT TO ASCII
015F F4F6           ADI     -10
0161 9404           JP      $2                      ; NUMBER IS A THRU F
0163 F43A           ADI     '0'+10
0165 9002           JMP     $3
```

```
309 0167 F440  $2:     ADI     'A'-1           ; THE -1 TAKES CARE OF CARRY IN
310 0169 3F    $3:     XPPC    P3              ; PRINT NUMBER
311 016A BA00          DLD     (P2)
312 016C 9806          JZ      $4
313 016E C204          LD      4(P2)           ; GET ORIGINAL NUMBER
314 0170 D40F          ANI     0F              ; MASK 2ND DIGIT
315 0172 90EA          JMP     $5
316 0174 C203  $4:     LD      3(P2)           ; CHECK FOR PRINTING BLANK
317 0176 9801          JZ      $6
318 0178 3F            XPPC    P3              ; IF NOT 0, PRINT BLANK
319 0179 C201  $6:     LD      1(P2)           ; RESTORE RETURN ADDRESS
320 017B 37            XPAH    P3
321 017C C202          LD      2(P2)
322 017E 33            XPAL    P3
323 017F C604          LD      @4(P2)          ; RESTORE STACK AND AC
324 0181 C601          LD      @1(P2)
325 0183 3F            XPPC    P3              ; RETURN
326 0184 90B8          JMP     PHEX1


327                    .PAGE   'GECO'
328                    .LOCAL
329            ;
330            ; GECO IS USED FOR KEYBOARD INPUT SO IT ECHOS THE
331            ; CHARACTER BUT DOES NOT ENABLE THE READER RELAY.
332            ;
333 0186 C408  GECO:   LDI     8               ; SET COUNT = 8
334 0188 CAFF          ST      -1(P2)
335 018A 06    $2:     CSA                     ; WAIT FOR START BIT
336 018B D420          ANI     020
337 018D 9CFB          JNZ     $2              ; NOT FOUND
338 018F C457          LDI     87              ; DELAY 1/2 BIT TIME
339 0191 8F04          DLY     4
340 0193 06            CSA                     ; IS START BIT STILL THERE?
341 0194 D420          ANI     020
342 0196 9CF2          JNZ     $2              ; NO
343 0198 06            CSA                     ; SEND START BIT (NOTE THAT
344 0199 DC01          ORI     1               ;   OUTPUT IS INVERTED)
345 019B 07            CAS
346 019C C47E  $LOOP:  LDI     126             ; DELAY 1 BIT TIME
347 019E 8F08          DLY     8
348 01A0 06            CSA                     ; GET BIT (SENSEB)
349 01A1 D420          ANI     020
350 01A3 9802          JZ      $3
351 01A5 C401          LDI     1
352 01A7 CAFE  $3:     ST      -2(P2)          ; SAVE BIT VALUE (0 OR 1)
353 01A9 1F            RRL                     ; ROTATE INTO LINK
354 01AA 01            XAE
355 01AB 1D            SRL                     ; SHIFT INTO CHARACTER
356 01AC 01            XAE                     ; RETURN CHAR TO E
357 01AD 06            CSA                     ; ECHO BIT TO OUTPUT
358 01AE DC01          ORI     1
359 01B0 E2FE          XOR     -2(P2)
360 01B2 07            CAS
361 01B3 BAFF          DLD     -1(P2)          ; DECREMENT BIT COUNT
362 01B5 9CE5          JNZ     $LOOP           ; LOOP UNTIL 0
363 01B7 06            CSA                     ; SET STOP BIT
364 01B8 D4FE          ANI     0FE
365 01BA 07            CAS
366 01BB 8F08          DLY     8
367 01BD 40            LDE                     ; AC HAS INPUT CHARACTER
368 01BE D47F          ANI     07F
369 01C0 01            XAE
370 01C1 40            LDE
371 01C2 3F            XPPC    P3              ; RETURN
372 01C3 90C1          JMP     GECO
```

```
373                     .PAGE   'PUTC'
374                     .LOCAL
375             ;
376             ; PUT CHARACTER  IN AC TO TTY.  ALL REGS SAVED.
377             ; IF INPUT DETECTED, CONTROL PASSES TO PROMPT.
378             ; NOTE: TTY LOGIC LEVELS ARE INVERTED FOR OUTPUT
379             ;
380 01C5 01    PUTC:   XAE
381 01C6 C4FF          LDI     255
382 01C8 8F17          DLY     23
383 01CA 06            CSA                     ; SET OUTPUT BIT TO LOGIC 0
384 01CB DC01          ORI     1               ;   FOR START BIT. (NOTE INVERSION)
385 01CD 07            CAS
386 01CE C409          LDI     9               ; INITIALIZE BIT COUNT
387 01D0 CAFF          ST      -1(P2)
388 01D2 C48A  $1:      LDI     138             ; DELAY 1 BIT TIME
389 01D4 8F08          DLY     8
390 01D6 BAFF          DLD     -1(P2)          ; DECREMENT BIT COUNT.
391 01D8 9810          JZ      $EXIT
392 01DA 40            LDE                     ; PREPARE NEXT BIT
393 01DB D401          ANI     1
394 01DD CAFE          ST      -2(P2)
395 01DF 01            XAE                     ; SHIF DATA RIGHT 1 BIT
396 01E0 1C            SR
397 01E1 01            XAE
398 01E2 06            CSA                     ; SET UP OUTPUT BIT
399 01E3 DC01          ORI     1
400 01E5 E2FE          XOR     -2(P2)
401 01E7 07            CAS                     ; PUT BIT TO TTY
402 01E8 90E8          JMP     $1
403 01EA 06    $EXIT:  CSA                     ; SET STOP BIT
404 01EB D4FE          ANI     0FE
405 01ED 07            CAS
406 01EE D420          ANI     020             ; CHECK FOR KEYBOARD INPUT
407 01F0 9803          JZ      $2              ; ATTEMPTED INPUT (NOTE THAT
408                                            ;   INPUT IS NOT INVERTED)
409 01F2 3F            XPPC    P3              ; RETURN
410 01F3 90D0          JMP     PUTC
411 01F5 C400  $2:      JS LDI  P3,CMDLP
    01F7 37C4
    01F9 3933
    01FB 3F
412       0000          .END

   ******       0 ERRORS IN ASSEMBLY   ******
$1&     $1(     $1)     $1+     $2&     $2(     $2)     $2*     $2+     $3(
0072    00F3    0148    01D2    0070    010E    0167    018A    01F5    011D

$3)     $3*     $4&     $4(     $4)     $5(     $5)     $6(     $6)     $EXIT+
0169    01A7    0085    011A    0174    0123    015E    00E2    0179    01EA

$LOOP(  $LOOP*  $RET(   $SKIP%  $SKIP&  AC      CMDLP   ENTER   ERROR   EX
00F9    019C    0104    0062    00CC    FFFE    003A    0020    00CC    FFFF

EXIT    EXOFF   GECO    GHEX    GHEX2   GO      LOOP1   MOD     P1      P2
0003    FFFF    0186    00E0    00DC    0056    00B7    0067    0001    0002

P2ADR   P3      PC      PHEX1   PHEX2   PT1     PT2     PUTC    SR      STACK
0FF6    0003    FFF8    013E    0144    FFFA    FFFC    01C5    0000    0FFF

START   TYPE
0001    0062


   FCB3    08E0
```

Appendix C

APPLICATION EXAMPLE

C.1 SOFTWARE "ONE-SHOT"

The following program is intended for use with the SC/MP Kit. The program simulates a retriggerable one-shot. A momentary contact switch is used to "fire the one-shot" (begin the program). The switch is connected to the SENSE A input to SC/MP; SENSE A is the interrupt input. When the interrupt (switch closure) is detected, the FLAG 1 output from SC/MP is set to a logic '1' and is used to drive an LED indicator through a transistor. (The hardware for this demonstration circuit is shown schematically in figure C-1.) A Delay Instruction (DLY) is then used to generate a delay of approximately 4 seconds. After 4 seconds, the LED will be turned off by setting the FLAG 1 output to '0' (zero). If the switch is held down, or depressed again before the LED is turned off, the LED remains lit; it is turned off approximately 4 seconds after the last switch opening.

Table C-1 is an assembler listing for the program showing the memory locations, assembler mnemonic, and machine language format (in hexadecimal) for each instruction in the program.

Using KITBUG and the TTY, the program could be entered into memory using the Modify Command of KITBUG and then could be executed using the Go command. Table C-2 shows the printout of this program that would be obtained using the KITBUG Type Command.

Note that this program (and any program utilizing interrupts) uses Pointer Register P3. Therefore, a program-controlled transfer back to KITBUG cannot be accomplished (see section 4.6 for a discussion of transfer of control between KITBUG and application programs).

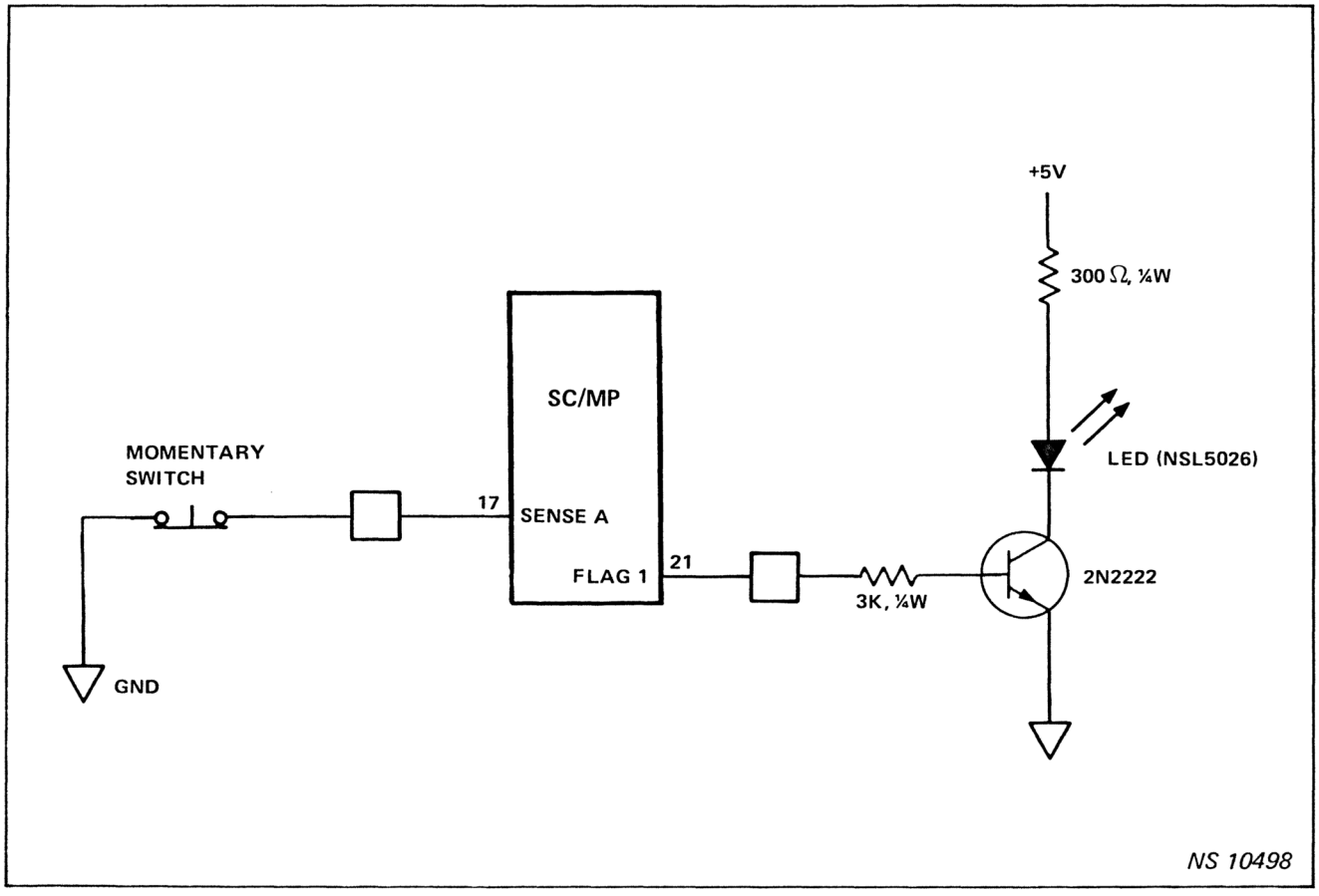


Figure C-1. 'One-Shot' Schematic

Table C-1. "One-Shot" Assembler Listing

Memory Address	Machine Language Code (Hex)		Assembler Opcode Mnemonics	Assembler Operand ↓	Comments
0F00	04		DINT		;DISABLE INTERRUPT
0F01	C41C		LDI	L(GO)-1	
0F03	33		XPAL	3	
0F04	C40F		LDI	H(GO)	
0F06	37		XPAH	3	;SET UP INTERRUPT ;POINTER
0F07	C400	OVER:	LDI	0	
0F09	C81E		ST	COUNT	;ZERO LOOP CNTR
0F0B	05		IEN		;ENABLE INTERRUPTS
0F0C	C01B	SEARCH:	LD	COUNT	
0F0E	98FC		JZ	SEARCH	;WAIT FOR INTERRUPT
0F10	C4FF		LDI	X'FF	
0F12	8FFF	LOOP:	DLY	X'FF	
0F14	B813		DLD	COUNT	
0F16	9CFA		JNZ	LOOP	
0F18	C400		LDI	0	
0F1A	07		CAS		;TURN OFF LED
0F1B	90EA		JMP	OVER	
			;INTERRUPT SERVICE		
0F1D	C402	GO:	LDI	2	
0F1F	07		CAS		;TURN ON LED
0F20	C40F		LDI	15	
0F22	C805		ST	COUNT	
0F24	05		IEN		
0F25	3F		XPPC	3	;RETURN TO MAIN PROG
0F26	90F5		JMP	GO	
		;DATA AREA			
	0F29	COUNT:	.=.+1		
	0000		.END		

Table C-2. Printout of "One-Shot" Program Using Type Command

```
-TF00 (CR)
0F00 04
0F01 C4
0F02 1C
0F03 33
0F04 C4
0F05 0F
0F06 37
0F07 C4
0F08 00
0F09 C8
0F0A 1E
0F0B 05
0F0C C0
0F0D 1B
0F0E 98
0F0F FC
0F10 C4
0F11 FF
0F12 8F
0F13 FF
0F14 B8
0F15 13
0F16 9C
0F17 FA
0F18 C4
0F19 00
0F1A 07
0F1B 90
0F1C EA
0F1D C4
0F1E 02
0F1F 07
0F20 C4
0F21 0F
0F22 C8
0F23 05
0F24 05
0F25 3F
0F26 90
0F27 F5
```