Synertek

Systems Corporation

CP110 SUPERJOLT REFERENCE MANUAL

CP110

Single Board Computer

(Superjolt Series)

Synertek Systems Corp.
Sunnyvale, California

MAN-A-260007 Rev A

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

SECTION 1

SUPER JOLT START-UP

HOOKING UP THE POWER SUPPLY

The proper power supply is an important part of your SUPER JOLT system. Be sure to check your power source and have identified the proper connections and have tested the power supply to be certain that all voltages are proper and that none have overvoltage surges when TURNING ON power.

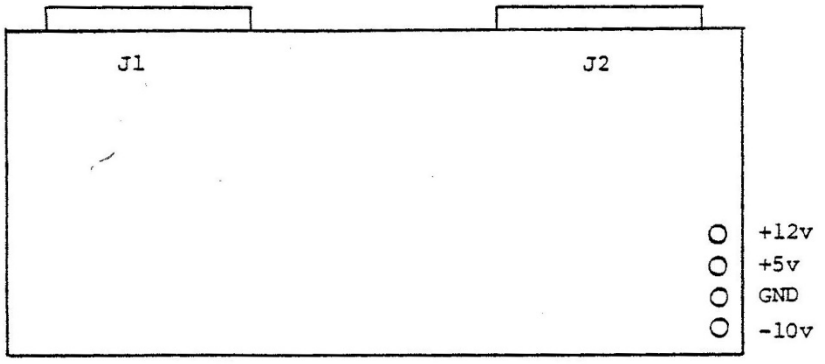
Your SUPER JOLT card requires a minimum of two voltages and one ground connection. The card, as delivered will perform with +5v, GND, and -10v if you are using the TTY current loop. If you plan to use the EIA terminal connection or 2708 PROMS you will need to add the +12v supply. Power supply requirements are listed below.

POWER SUPPLY REQUIREMENTS

	TYPICAL		MAXIMUM	
Voltage	W/ROMs *	W/O ROMs *	W/ROMs	W/O ROMs
+12v±5%	70 ma	4.4 ma	110 ma	6 ma
+5v±5%	550 ma	520 ma	880 ma	850 ma
-10v±5%	30 ma	30 ma	40 ma	40 ma

* SW101 TINY BASIC/Resident Assembler ROMs

All power connections are hooked up as indicated below and as marked on the P.C. card.



HOOKING UP A TERMINAL TO THE JOLT

Types of Terminals: The following is a list of qualifications a terminal must have to run on a SUPER JOLT system with DEMON™ installed:

A. Character Set: Must transmit and receive the standard ASCII character set. (64, 96, or 128 character).

B. Mode: Mode of transmission is bit serial full duplex (full duplex is where the keyboard sends only to the computer and the computer sends only to the printer).

C. Transmission Rate: Transmission rate can be anything from 110 to 300 baud (10 to 30 characters per second) the SUPER JOLT will synchronize to the baud rate of the terminal.

D. Bit Serial Format: Start bit, seven data bits, one parity bit (this bit is ignored by DEMON™ on receive and set to a "1" on transmit), and one, one and a half, or two stop bits.

E. Electrical Interface: Any one of three types of electrical interfaces can be used with the SUPER JOLT

1. RS-232C (EIA)
2. 20 milliamp current loop interface
3. TTL logic interface

HOOKING UP THAT ELECTRICAL INTERFACE

E.I.A.: The first type of interface we will discuss is the E.I.A. or RS-232C standard, here after refered to as E.I.A.. Table 1 shows the complete standard for signal assignment on the 25 Pin Cannon connector which is the standard connector for terminal-modem computer hookup with E.I.A.. Figure 1 shows the connection of a typical E.I.A. terminal to the SUPER JOLT CPU. When using the SUPER JOLT with an E.I.A. interface equipped terminal, connections should be made in this fashion. Note that the D.S.R. signal from the CPU is required only on some terminals.

TTY CURRENT LOOP

The most common example of a current loop interfaced terminal is the model ASR33 Teletype™. Before hooking up your Teletype™ or other current loop type terminal

RS-232 STANDARD SIGNALS AT THE TERMINAL

* Signals Commonly Used

PIN	FUNCTION
*1	Protective ground
*2	Transmitted data
*3	Received data
4	Request to send
5	Clear to send
6	Data set ready
*7	Signal ground
8	Data carrier detector
9	Reserved for data set testing
10	Reserved for data set testing
11	Unassigned
12	Unassigned
13	Unassigned
14	Unassigned
15	Unassigned
16	Unassigned
17	Unassigned
18	Unassigned
19	Unassigned
20	Data terminal ready
21	Unassigned
22	Ring indicator
23	Unassigned
24	Unassigned
25	Unassigned

TABLE 1

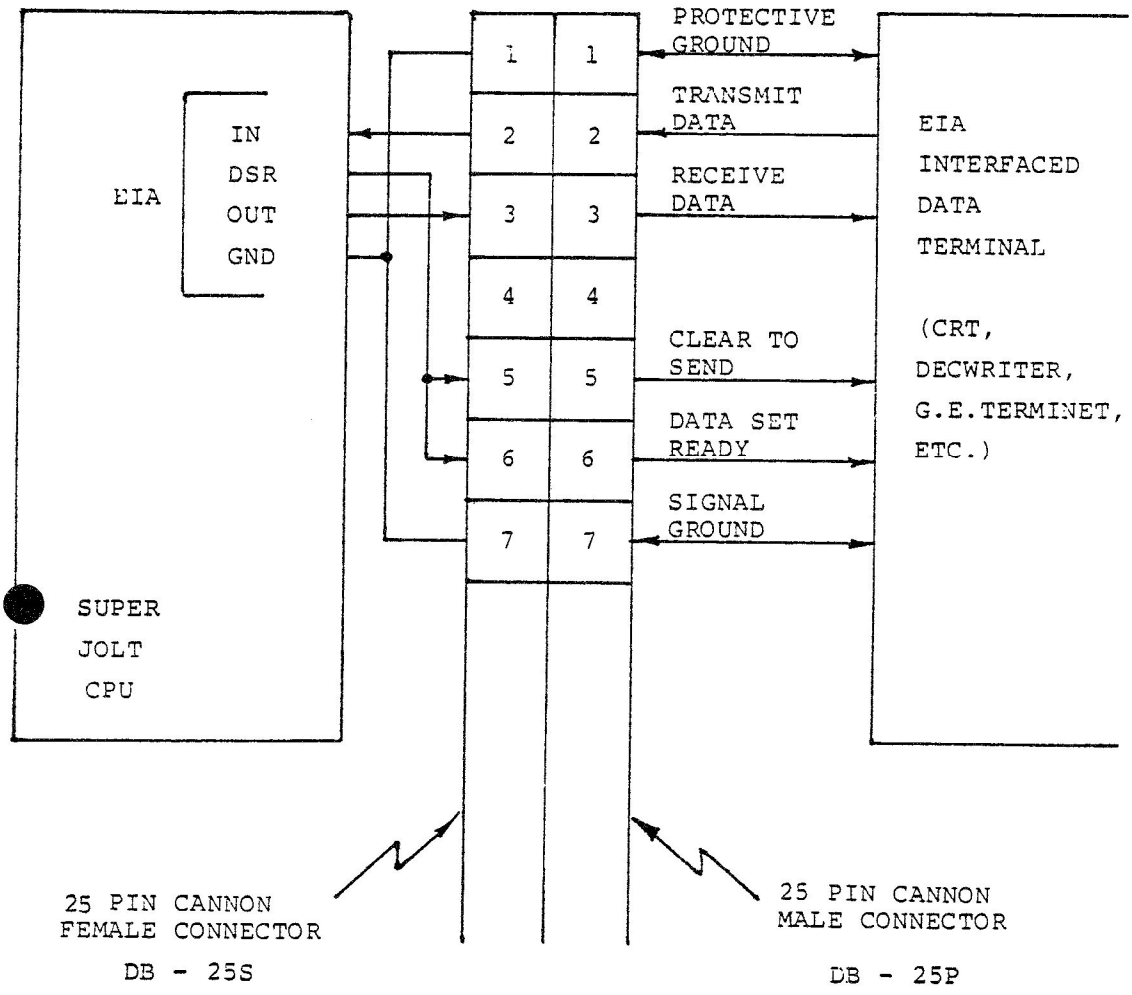


FIGURE 1. SUPER JOLT HOOKED TO AN EIA EQUIPPED TERMINAL

refer to its maintanence manual and be certain that it is set up for 20 milliamp current loop operation and identify the four interface wires shown in Figure 2 for the Teletype™. If your Teletype™ also has a paper tape reader, then check to see that it is an automatic reader as SUPER JOLT does not supply a "reader run" relay control circuit.

Unlike RS-232C, the 20 milliamp current loop interface has no standard connector. Every computer and terminal manufacturer has their own type of connector. For interfacing the SUPER JOLT you should select a connector that mates with the one on your particular model Teletype™.

Figure 2 shows a typical teletype hookup to SUPER JOLT using the 20 milliamp current loop interface.

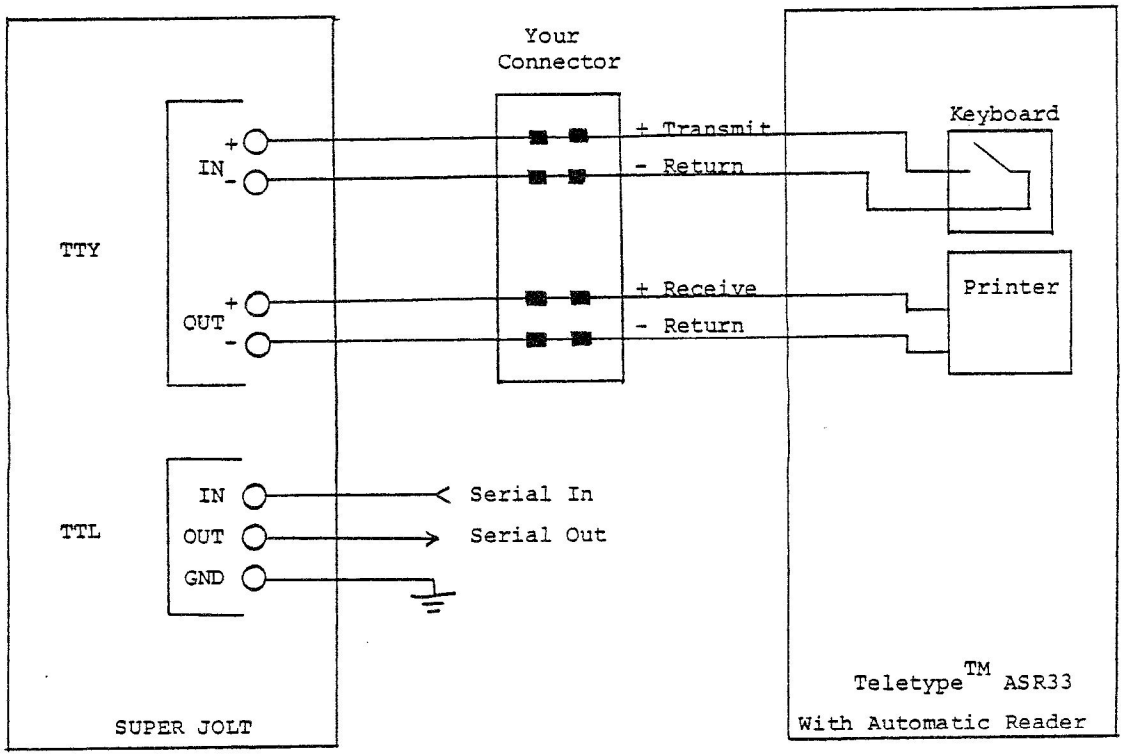


FIGURE 2. SUPER JOLT HOOKED TO A 20 ma CURRENT LOOP TELETYPE™ OR TTL INTERFACE

TTL

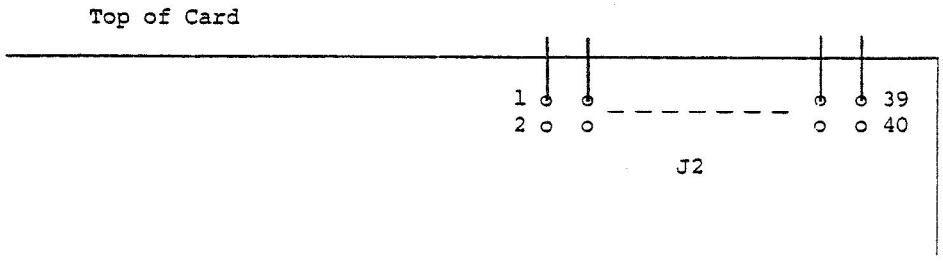
The third type of serial interface to the SUPER JOLT is "TTL" which is a direct logic level interface. Most commercial terminals do not use this type of interface, however some homemade terminals and inexpensive terminals use this method to save costs in interfacing (i.e. the T.V. typewriter). Interfacing this type of equipment to the SUPER JOLT CPU is provided for. See figure 2 for the proper connections.

READY TO RUN?

Your SUPER JOLT card comes ready to power-up automatically to the on-board debugger-monitor DEMON™ using a TTY. Refer to section 4 on using DEMON™. If you wish to power-up to your application program in 2708 PROM refer to section 3 for selection of the proper on-board jumpers. Many other jumper options are available concerning negative power input, 6530 on-board RAM, terminal input and address line tristate control. The user should become very familiar with the function of each option in order to most effectively use the SUPER JOLT card. Again, if you have a TTY (20 ma current loop) and a power supply your SUPER JOLT is ready-to-use with no jumper changes. Power-up or reset will cause the 6502 CPU to execute the DEbug MONitor.

HOOKING UP THE I/O

SUPER JOLT is designed with two types of connectors. The first, called J1, is on the upper left and is used mainly for system expansion such as RAM, PROM, I/O etc. J1 should only be used to expand to other available SUPER JOLT cards or for your own special card designs. The second, called J2, is on the upper right and contains all on-board I/O lines. Table 2 list J2 I/O pin #'s and software address assignments and Table 3 list available mating connectors. Pin numbering on both connectors is illustrated as follows:



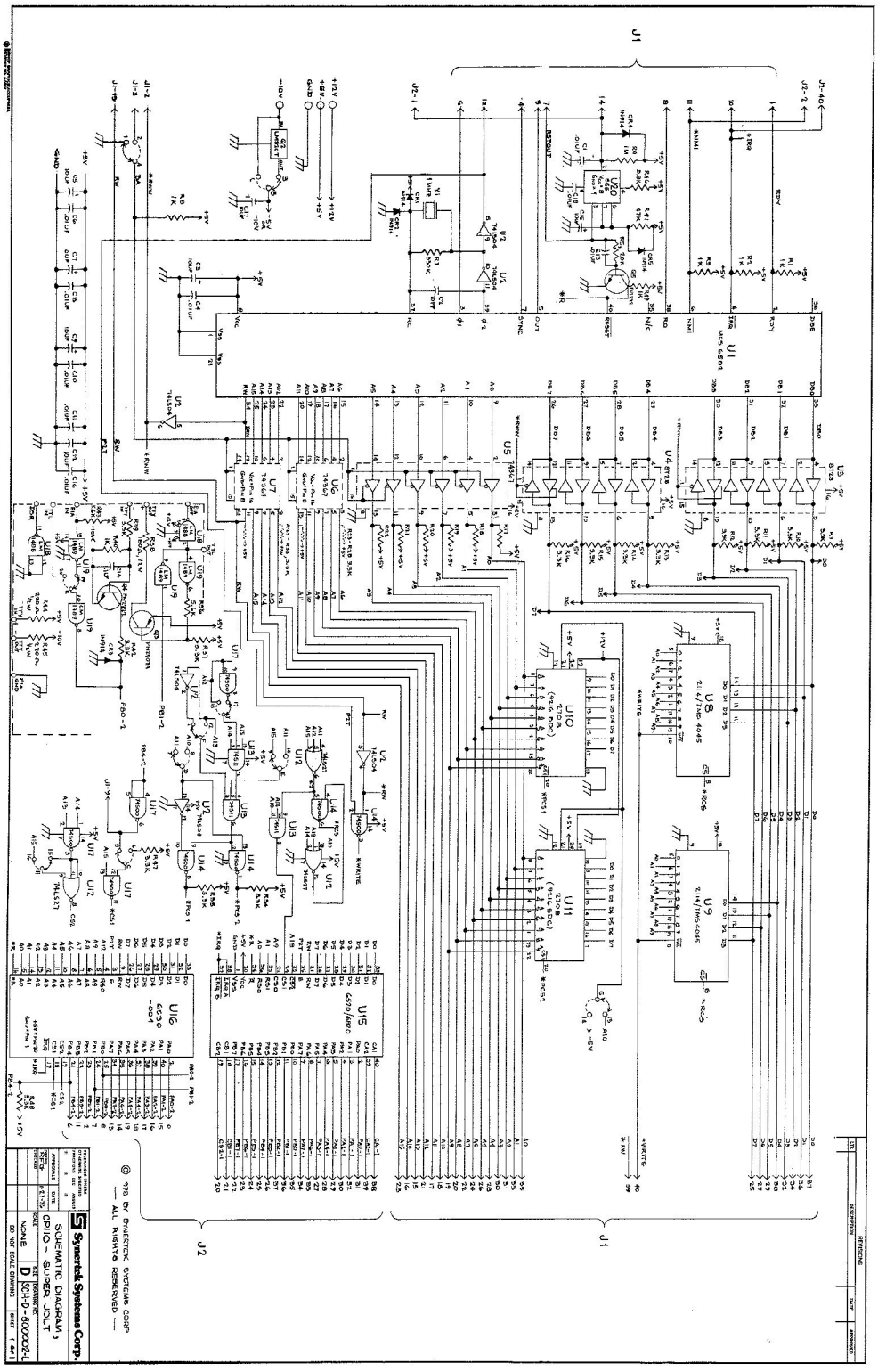
PORT #1	Bit #	NAME	PIN #	ADDRESS
	0	PA0-1	31	
	1	PA1-1	32	6520 PIA
	2	PA2-1	30	4600 Direction/Data
	3	PA3-1	29	4601 Control
	4	PA4-1	28	
	5	PA5-1	27	
	6	PA6-1	33	
	7	PA7-1	34	
	Interrupt Input	CA1-1	38	
	Output Control	CA2-1	39	

PORT #2	BIT #	NAME	PIN #	
	0	PB0-1	35	
	1	PB1-1	36	6520 PIA
	2	PB2-1	37	4602 Direction/Data
	3	PB3-1	26	4603 Control
	4	PB4-1	25	
	5	PB5-1	24	
	6	PB6-1	23	
	7	PB7-1	22	
	Interrupt Input	CB1-1	21	
	Output Control	CB2-1	20	

PORT #3	BIT #	NAME	PIN #	
	0	PA0-2	10	
	1	PA1-2	15	6530 I/O
	2	PA2-2	16	6E00 Data
	3	PA3-2	17	6E01 Direction
	4	PA4-2	18	
	5	PA5-2	19	
	6	PA6-2	14	
	7	PA7-2	13	

PORT #4	BIT #	NAME	PIN #	
	2	PB2-2	12	6E02 Data
	3	PB3-2	11	6E03 Direction

TABLE 2. SUPER JOLT I/O ASSIGNMENTS ON J2



APPLICATION	MANUFACTURER	PART NUMBER
FLAT CABLE CONNECTORS*	T & B ANSLEY SPECTRA-STRIP 3M	609-4000 802-140 3417-0000
P.C.B. MOUNTING CONNECTORS	AMP	86418-2
*RECOMMENDED FLAT CABLING	T & B ANSLEY SPECTRA-STRIP 3M	171-40 MANY TYPES 3302/40, 3365/40, 3469/40, 3476/40

TABLE 3. J1 & J2 CONNECTOR PART NUMBERS

SECTION 2

SUPER JOLT MODULE

General

The SUPER JOLT CPU card is a complete microcomputer on a single printed circuit board. When connected to a terminal, the CPU card provides everything necessary to begin writing, debugging and executing microcomputer programs. The salient features of the SUPER JOLT CPU card are:

- A Synertek SY6502 NMOS microprocessor
- 1024 bytes of program RAM, and 64 bytes of interrupt vector RAM
- 1024 bytes of mask programmed ROM containing DEMON™, a powerful debug monitor
- Sockets for 2048 bytes PROM memory
- 28 programmable I/O lines
- Crystal controlled clock
- Serial I/O ports for use with a teleprinter current loop drive/receiver, EIA standard driver/receiver, or TTL
- Expandable address and data buses
- Buffered CPU address and data lines
- Hardware interrupts
- Control panel interface lines available on card connector
- Optional ROM resident TINY BASIC and Assembler

The CPU card was designed to be a general purpose microcomputer with provisions for expanding memory and interfacing to serial or parallel I/O devices. System expansion may be accomplished through the use of standard SUPER JOLT support cards

CPU

The SY6502 CPU chip is a parallel 8-bit microprocessor with 16 address lines and an internal oscillator. The data bus (DO-D7) is bi-directional and will drive one TTL (1.6 ma, 130 pf) load directly. The 64K byte (2^{16}) address space is used to address program memory and to select I/O devices for communication with the CPU. The address will also drive one TTL (1.6 ma, 130 pf) load directly. On-board address and data buffers expand the drive capability to 48 ma.

The internal oscillator operates in a "free run" mode based on a crystal oscillator frequency of 1 MHZ. The crystal provides a very stable clock which allows for accurate and repeatable programmed timing loops.

The RESET input to the CPU is pulled to logic ground by a 555 timer circuit on the printed circuit board. The CPU normally fetches a new program count vector from hex locations FFFC and FFFD upon activation of the RESET line, but these locations are in the interrupt vector RAM and therefore volatile. Hardware on the CPU board causes the CPU to begin executing the monitor program by forcing the effective sixteenth bit of the address bus (A15) to a logic ZERO during reset. As a result, the RESET function on the SUPER JOLT CPU card cause the debug monitor (DEMON^{TM}) to begin executing. This can be altered by changing the various on-board jumpers. (see section 3)

There are two interrupt inputs to the CPU. One interrupt is maskable under program control ($\overline{IRQ}$) and the other ($\overline{NMI}$) is not.

A READY control line provides for asynchronous operation with slow memory or I/O devices.

The address bus (A0-A15), the data bus (D0-D7), the two phase clock (PIT,P2T), the reset line (*RESET), the interrupt lines (*IRZ, and *NMI) and the ready line (RDY) are all available at the edge connector of the CPU card.

A more detailed description of the CPU inputs and outputs may be found in the SY6500 hardware manual available from Synertek Inc.

PROGRAM RAM

There are 1024 bytes of program RAM provided on the CPU card. The program RAM is hardwired addressed as the first 1024 bytes of the CPU's 64K of memory address space. It may become necessory to remove these RAM's from their sockets if a 4K memory card is also hardwired in this address space. The program RAM on the CPU card uses 2114 4K static RAMs.

MONITOR ROM AND INTERRUPT VECTOR RAM

The monitor ROM is located in the last 1K bytes of the lower half of memory space (first 32K bytes). The interrupt vector RAM is located in the last 64 bytes of the 64K memory address space.

The monitor ROM and interrupt vector RAM as well as additional I/O are implemented with a single SY6530 chip.

PROGRAMMABLE USER I/O

The programmable I/O lines available from the CPU card are provided by a Peripheral Interface Adapter (PIA) and the 6530 multi-function chip.

The PIA has two 8-bit I/O ports with two interrupt-causing control lines each. A data direction register for each port determines whether each I/O line is an input or an output. A detailed description of the PIA chip may be found in the SY6500 microcomputer family Hardware Manual.

The 6530 ROM chip provides 10 additional I/O lines that may also be specified as input or output lines under program control. There are eight I/O lines from one port on the 6530 and two lines from the second port. These I/O lines may be used in conjunction with DEMON™ for interfacing a high speed paper tape reader to the CPU card. In the paper tape reader application, the eight I/O lines from the second port are used to accomplish the handshake control between the reader and the CPU card.

The PIA is hardwired addressed as location 4600_{16} to 4603_{16} in the memory address space. Memory addresses from 4000_{16} to $5C03_{16}$ are allocated for PIA devices so that the SUPER JOLT system may be easily expanded to accomodate up to eight PIA chips. For a complete illustration of memory allocation refer to section 3.

STANDARD INTERFACE CIRCUITS

The SUPER JOLT CPU card provides direct interfacing with a 20 ma current loop RS232C interface requires +12v and -10v. Both interfaces are wired in parallel on the input and output thereby allowing both interfaces to be used simultaniously. The TTL input must be jumpered as an option in place of the EIA input. For further assistance in connecting the SUPER JOLT CPU to a terminal refer to section 1.

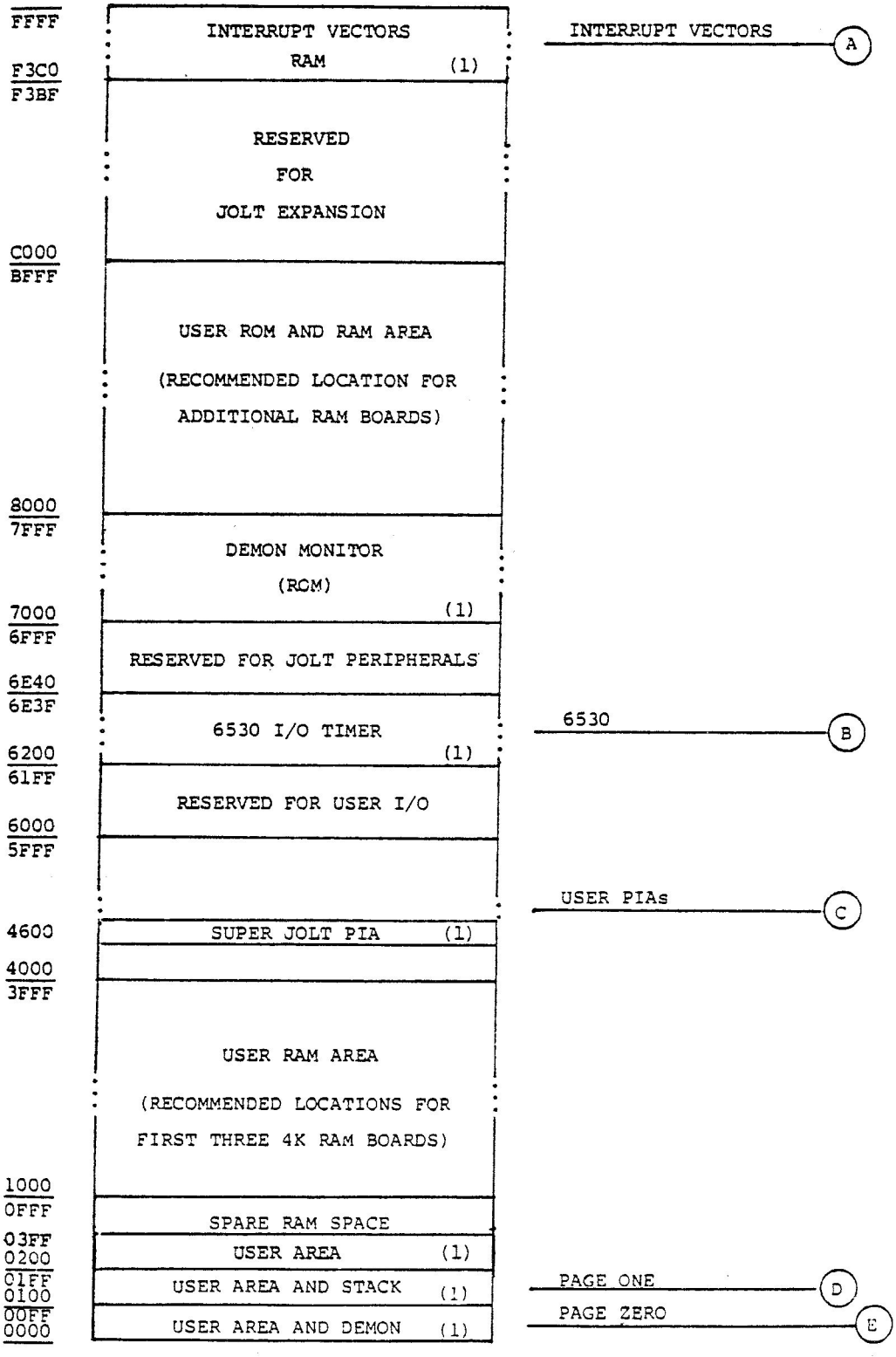
SECTION 3

MEMORY MAP AND ON-BOARD JUMPER OPTIONS

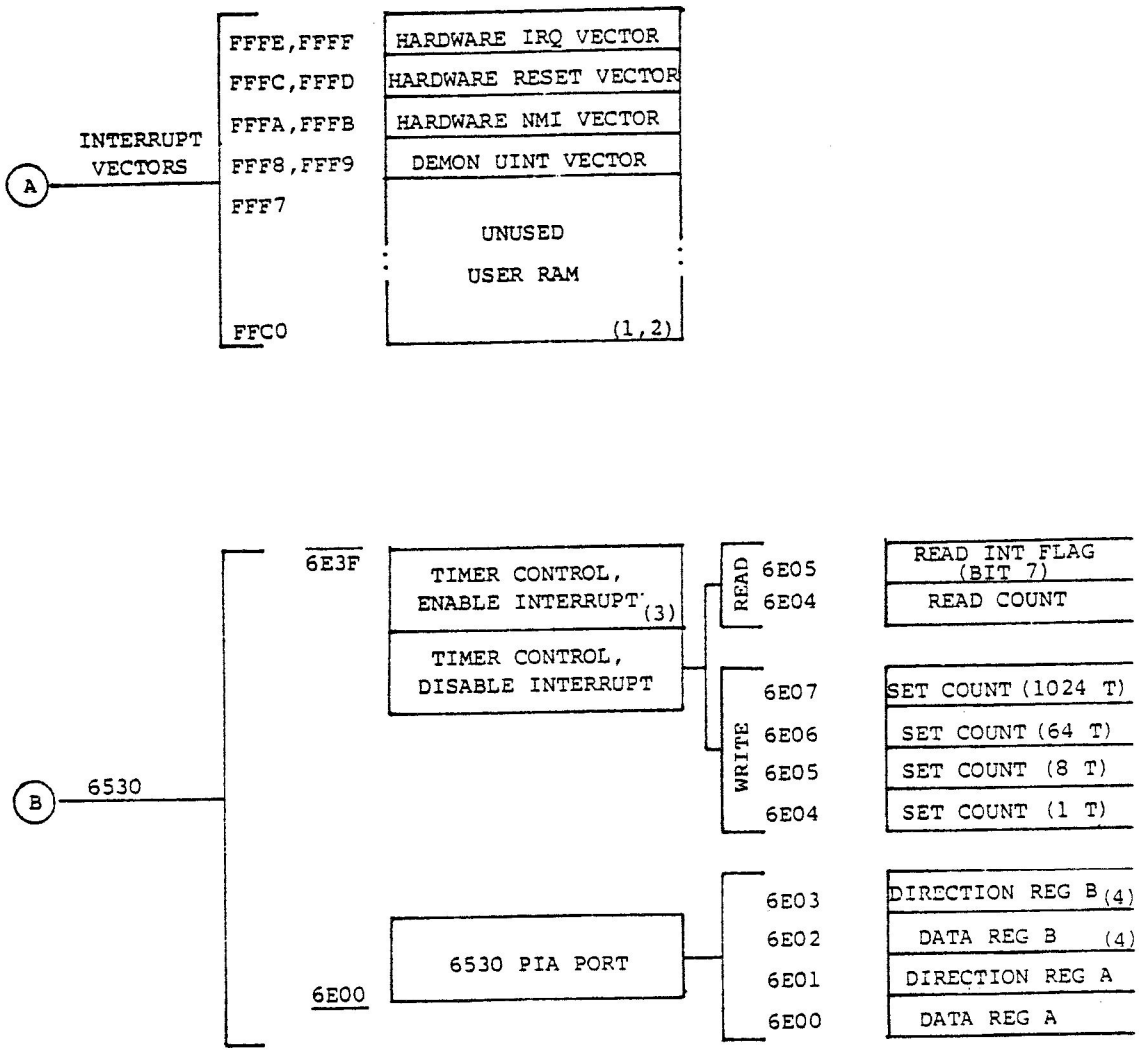
SUPER JOLT SYSTEM MEMORY MAP

The memory map on the following pages explains what functions have been assigned to each segment of the SUPER JOLT address space. It is recommended that users respect this space allocation when adding memory and peripherals to their SUPER JOLT systems. Space has been reserved for 32K bytes of user RAM or ROM, seven additional PIA devices, and up to 512 user I/O devices registers. Other areas are reserved for JOLT expansion, i.e., new SUPER JOLT peripherals and memory options will use these spaces. Users are advised to not use SUPER JOLT expansion space unless absolutely necessary.

Note that some areas used by the SUPER JOLT CPU board and PIA boards have more space indicated than there are registers or locations in the device occupying them. This is because these devices do not decode all address bits, or use some of the address bits for special functions. For example, the 6530 timer determines the time scale and interrupt enable/disable by the address used to access it. Thus, these "partly filled" areas are actually entirely used and are not available for other uses.



(1) Standard on Superjolt CPU Card

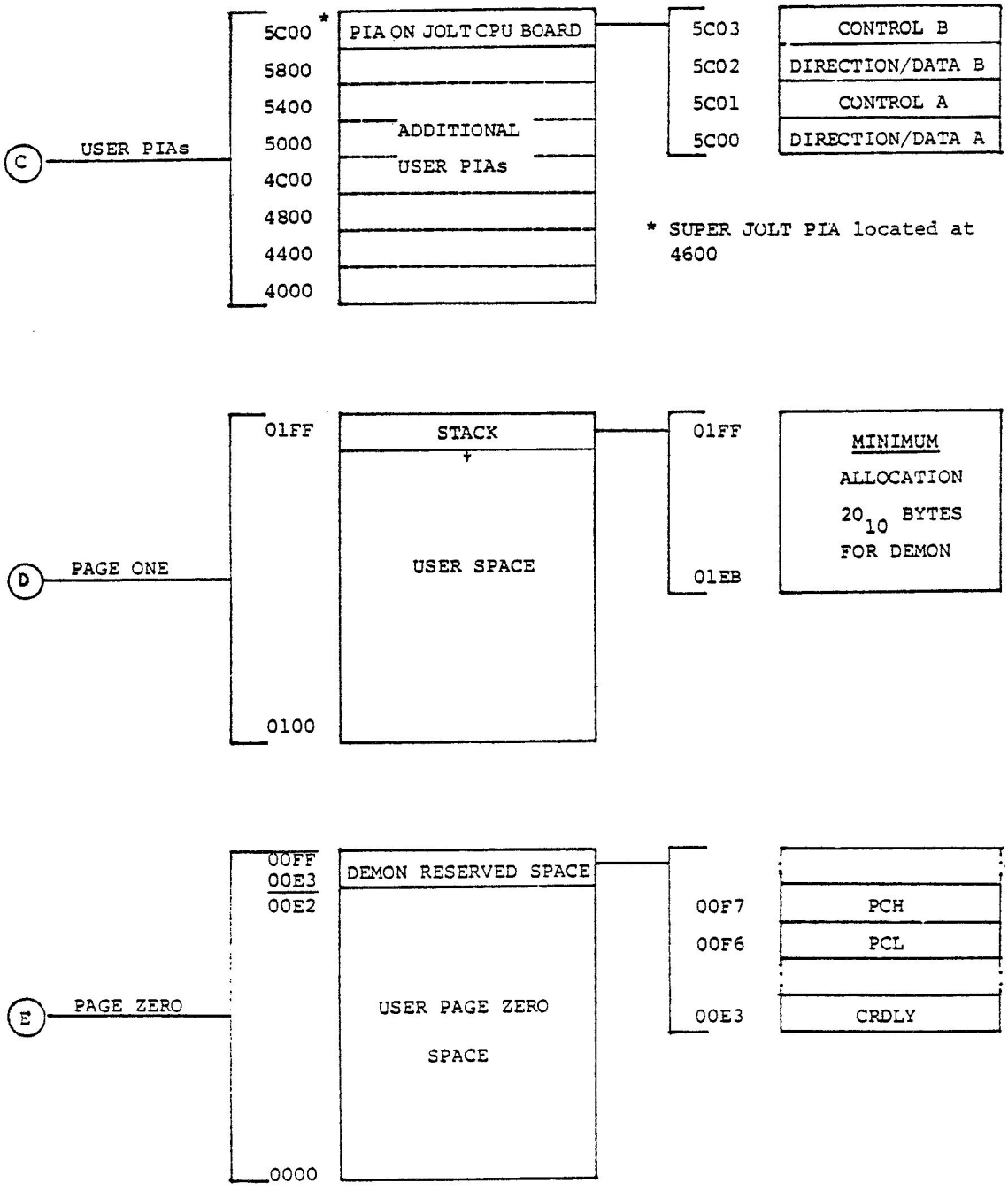


(1) Standard on JOLT CPU board.

(2) Available to user—not used by DEMON.

(3) To get enable-interrupt address, add 0008_{16} to disable-interrupt address with corresponding functions.

(4) Reserved for DEMON use—TTY control and reset functions.


C
USER PIAs
5C00 *
5800
5400
5000
4C00
4800
4400
4000
PIA ON JOLT CPU BOARD
ADDITIONAL
USER PIAs
5C03
5C02
5C01
5C00
CONTROL B
DIRECTION/DATA B
CONTROL A
DIRECTION/DATA A
* SUPER JOLT PIA located at 4600
D
PAGE ONE
01FF
0100
STACK
USER SPACE
01FF
01EB
MINIMUM
ALLOCATION
20_{10} BYTES
FOR DEMON
E
PAGE ZERO
00FF
00E3
00E2
0000
DEMON RESERVED SPACE
USER PAGE ZERO
SPACE
00F7
00F6
00E3
PCH
PCL
CRDLY

JUMPER	POSITION	APPLICABLE ADDRESS OPTION	DESCRIPTION
A	1 2		ADDRESS BUFFERS ALWAYS ENABLED ADDRESS BUFFERS TRI-STATE EXTERNALLY CONTROLLED
B	3 4		-10v POWER INPUT (ON-BOARD REGULATED TO -5v) -5v POWER INPUT (BY-PASSES REGULATOR)
C (1)	5 6		AUTO POWER-ON TO "DEMON" - ENABLED AUTO POWER-ON TO "DEMON" - DISABLED
D	7 8	E,F,G,H A,B,C,D	A11 CONTROL'S ROM SOCKET SELECTION A10 CONTROL'S PROM SOCKET SELECTION
E	9 10	E,F,G,H A,B,C,D	A15 ROM ADDRESS ENABLE A11 PROM ADDRESS ENABLE
F	11 12	C,D,G,H A,B,C,D	$\overline{\text{A13}}$ PROM/ROM ADDRESS ENABLE A13 PROM/ROM ADDRESS ENABLE
G	13 14	E,F,G,H A,B,C,D	A10 ROM ADDRESS ENABLE -5v PROM ADDRESS ENABLE
H (1)	15 16	 A,E	ENABLES 16 BYTE RAM ON 6530 DISABLES 64 BYTE RAM ON 6530
J	17 18	B,D,F,H A,C,E,G	A12 PROM/ROM ADDRESS ENABLE A12 PROM/ROM ADDRESS ENABLE
K	19 20		EIA INPUT TTL INPUT

NOTE: (1) - The purpose of option H-16 is to allow PROM or RAM memory to exist at high address locations (such as F---) while still using the 6530 I/O - timer and "DEMON" subroutines, however, use of this option disables the Auto Power-On to "DEMON" (Jumper C must be in position 6)

EXAMPLES		FUNCTION
C-6	H-16	PWR. ON TO PROM/ROM MEMORY AT RESET VECTOR
C-6	H-15	PWR. on to 6530 RAM (not useful)
C-5	H-16	AUTO PWR.-ON DISABLED, 6530 RAM DISABLED
C-5	H-15	AUTO PWR.-ON, 6530 RAM ENABLED

SUPER JOLT ON-BOARD JUMPER OPTIONS

ADDRESS BITS →	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			ADDRESS OPTION
START ADDRESS																			
F800	1	1	1	1	1	0	-	-	-	-	-	-	-	-	-	-	(1Kx8)	2708 PROM-1	A
FC00	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-		2708 PROM-2	
E800	1	1	1	0	1	0	-	-	-	-	-	-	-	-	-	-		2708 PROM-1	B
EC00	1	1	1	0	1	1	-	-	-	-	-	-	-	-	-	-		2708 PROM-2	
D800	1	1	0	1	1	0	-	-	-	-	-	-	-	-	-	-		2708 PROM-1	C
DC00	1	1	0	1	1	1	-	-	-	-	-	-	-	-	-	-		2708 PROM-2	
C800	1	1	0	0	1	0	-	-	-	-	-	-	-	-	-	-		2708 PROM-1	D
CC00	1	1	0	0	1	1	-	-	-	-	-	-	-	-	-	-		2708 PROM-2	
F000	1	1	1	1	0	-	-	-	-	-	-	-	-	-	-	-	(2Kx8)	9216 ROM-1	E
F800	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-2	
E000	1	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-1	F
E800	1	1	1	0	1	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-2	
D000	1	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-1	G
D800	1	1	0	1	1	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-2	
C000	1	1	0	0	0	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-1	H
C800	1	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-		9216 ROM-2	
7000	0	1	1	1	X	X	-	-	-	-	-	-	-	-	-	-	1Kx8	6530 "DEMON" ROM	
0000	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-		1Kx8 RAM	
FFC0	1	1	1	1	X	X	1	1	1	1	-	-	-	-	-	-		64x8 6530 RAM	
4600	0	1	0	0	0	1	1	X	X	X	X	X	X	X	-	-		6520/6820 PIA	
6E00	0	1	1	0	X	X	1	0	0	0	-	-	-	-	-	-		6530 I/O TIMER	

(1) Legend for Above:

- Address lines decoded inside memory or I/O chip
0 Address line logic state for valid enable
1 Address line logic state for valid enable
X Indicates address line not used in decoding

(2) How to use: Select one (1) address option from A-H, then use jumper option chart to determine the required on-board jumpers.

(3) Address Decoding equations on-board (inside) 6530 Chip

RSO = PIN 4 CSI = PIN 18 CS2 = PIN 19

ROM ENABLE = $\overline{RSO} \cdot CSI \cdot CS2$
= $\overline{A15} \cdot A14 \cdot A13 \cdot A12$ (for Super Jolt)

RAM ENABLE = $RSO \cdot \overline{CSI} \cdot CS2 \cdot A9 \cdot A8 \cdot A7 \cdot A6$
= $\overline{A15} \cdot A14 \cdot A13 \cdot A12 \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot A6$

I-O/TIMER ENABLE = $\overline{RSO} \cdot CSI \cdot CS2 \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6}$
= $A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot A9 \cdot A8 \cdot A7 \cdot A6$

SUPER JOLT MEMORY - I/O DECODING MAP

SECTION 4

DEMON MONITOR CHECKOUT

You are now ready to check out your JOLT DEMON monitor. The instructions which follow assume that your JOLT is connected to a suitable power supply and a teletype or other serial computer terminal. A detailed description of the DEMON monitor starts on page 4-28. Here is a summary of its features:

DEMON is the DEbug MONitor program for the JOLT Microcomputer. It is supplied in read-only memory (ROM) as part of the 6530 multi-function chip on the JOLT CPU board. Because the DEMON code is non volatile, it is available at system power-on and cannot be destroyed inadvertantly by user programs. Furthermore, the user is free to use only those DEMON capabilities which he needs for a particular program. Both interrupt types, interrupt request (IRQ) and non-maskable interrupt (NMI) may be set to transfer control to DEMON or directly to the user's program.

DEMON communicates with the user via a serial full-duplex port (using ASCII codes) and automatically adjusts to the speed of the user's terminal. Any speed--even non-standard ones--can be accommodated. If the user's terminal has a long carriage return time, DEMON can be set to perform the proper delay. Commands typed at the terminal can direct DEMON to start a program, display or alter registers

and memory locations, set breakpoints, and load or punch programs. If available in the system configuration, a high-speed paper tape reader may be used to load programs through a parallel port on the 6530 chip. Programs may be punched in either of two formats--hexadecimal (assembler output) or BNPF (which is used for programming read-only memories). On loading or modifying memory, DEMON performs automatic read-after-write verification to insure that addressed memory exists, is read/write type, and is responding correctly. Operator errors and certain hardware failures may thus be detected using DEMON.

DEMON also provides several subroutines which may be called by user programs. These include reading and writing characters on the terminal, typing a byte in hexadecimal, reading from high-speed paper tape, and typing a carriage-return, line-feed sequence with proper delay for the carriage of the terminal being used. Program tapes loaded by DEMON may also specify a start address so that programs may be started with a minimum of operator action.

CHECKOUT INSTRUCTIONS

( ) 1. Turn power on, or if the power is on, perform a RESET operation. Type a carriage-return on the terminal. DEMON should respond with:

```
* 7052  30  18  FF  01  FF
```

(Exact values may vary, although the first and last values should be as shown). If no response or a garbled response occurs, RESET and try again. In case of continued trouble, refer to the diagnostic section of the CPU Assembly Manual.

The "* 7052 30 18 FF 01 FF" printout is DEMON's standard breakpoint message format. It consists of an asterisk "*" to identify the breakpoint printout, followed by the CPU register contents in this order: PC, P, A, X, Y, and S, i.e., Program Counter, Processor Status, Accummulator, X index, Y index and Stack Pointer. Note that all DEMON inputs and outputs are in base 16 which is referred to as hexadecimal, or just hex. In hexadecimal, the "digits" are 0,1,2...,A,B,C,D,E,F. After printing the CPU registers, DEMON is ready to receive commands from you, the operator. DEMON indicates this "ready" status by typing the prompting character "." on a new line.

( ) 2. DEMON's response to RESET is to wait for a carriage-return and then print the user's registers. DEMON uses this carriage return-character to measure the terminal line speed. If you have a settable-rate terminal, change the

rate (any speed between 10 and 30 cps will work) and repeat Step 1. DEMON should respond at the new terminal speed.

( ) 3. The user's CPU registers may also be displayed with the R command. Type an R. The monitor should respond as above, but without the asterisk. Presence of the asterisk indicates that an interrupt or break instruction caused the printout.

```
.R  7052  30  18  FF  01  FF
```

( ) 4. Displayed values may be modified using the Alter (:) command. To modify register contents, type a colon (:) followed by the new values. For example:

```
.R  7052  30  18  FF  01  FF
.:  0100  00  00  00  00  FF
.R  0100  00  00  00  00  FF
```

Notice that DEMON automatically types spaces to separate data fields. (Note: Characters typed by you, the user, are underlined in this document for clarity. Everything else is typed by the computer.) Examine your registers (R command) to verify the changes.

Memory may be examined and modified, as above, using the M and : commands. Try this:

```
.M  0100  00  66  23  EE  01  A2  41  6E
```

The memory command (M) causes DEMON to type the contents of the first eight bytes of memory. (Memory data will be random on startup). Alter and verify these bytes using the Alter command, as above:

```
.M   0100  00  66  23  EE  01  A2  41  6E
.:   0100  00  01  02  03  04  05  06  07
```

If only part of a line is to be altered, items to be left unchanged can be skipped over by typing blanks, and carriage-return (↵). Try this:

```
.M   0100  00  01  02  03  04  05  06  07
.:   0100  FF  __  FF  FF  ↵
.M   0100  FF  01  FF  FF  04  05  06  07
```

( ) 5. Try to alter a location in DEMON ROM:

```
.M   7000  85  F9  A9  23  D0  58  A9  16
.:   7000  00?
```

DEMON verifies all changes to memory. Since locations 7000 through 7007 are in read-only memory, alteration is not possible. DEMON signals write failure with a question mark. Similarly, the monitor will notify you of an attempt to alter a non-existant location:

```
.M   9000  90  90  90  90  90  90  90  90
.:   9000  00?
```

Note that attempts to read non-existant memory will normally yield the high-order byte of the address read.

( ) 6. There are three hardware facilities which may be used to stop a running (or run-away) program without the program itself calling DEMON. These are the hardware inputs RESET,

IRQ, and NMI. To test this feature enter the following program at location 0000:

location	contents	instruction
0000	4C	LOOP JMP LOOP
0001	00	
0002	00	

(Use the M and : commands.)

Now, set the program counter (PC) to this location using the R and : commands. Finally, tell DEMON to start executing your program using the GO (G) command:

```
.M 0000 FF 11 11 11 91 91 71 91
.: 0000 4C 00 00 ↓
.M 0000 4C 00 00 11 91 91 71 91
.R 0000 30 00 00 00 FF
.: 0000 ↓
.G
```

The computer should now be executing the program. It will continue to run until interrupted. Using the interrupt request line (IRQ), interrupt the processor. It should respond with:

```
*  0000 30 00 00 00 FF
```

Try the same experiment with non-maskable interrupt (NMI). The result should be the same except for a "#" character preceeding, which identifies the NMI printout. Finally, try it with RESET. RESET, however, forces JOLT to branch to DEMON, loosing the old PC and other register contents. Thus NMI is the preferred means for manually interrupting program execution. IRQ may also be

used unless it is required for other functions such as peripheral interrupts.

( ) 7. Use M and : to enter the following test program called CHSET because it prints the character-set on the terminal. Note that Alter (:) commands may be repeated without intervening M commands to set sequential locations:

```
                        ;CHECKOUT PROGRAM -- PRINT THE CHARACTER SET ON USER TERMINAL
                        CRLF    =$728A          ;ADDRESS OF DEMON CRLF ROUTINE
                        WRT     =$72C6          ;ADDRESS OF DEMON WRITE ROUTINE
                        ;
0000                            *=0             ;VARIABLE STORAGE IN PAGE ZERO
0000                    CHAR    *=*+1           ;STORAGE FOR CHARACTER
                        ;
0001                            *=$0100         ;PROGRAM STARTS ON PAGE ONE
                        ;
0100  20 8A 72          CHSET   JSR CRLF        ;DO CARRIAGE RETURN & LINE FEED
0103  A9 20                     LDA #$20        ;FIRST CHAR IS A SPACE
0105  85 00                     STA CHAR        ;INITIALIZE
                        ;
0107  A5 00             LOOP    LDA CHAR        ;GET CHARACTER
0109  C9 6C                     CMP #$6C        ;CHECK FOR LIMIT
010B  F0 08                     BEQ DONE        ;DONE IF 60
                        ;
010D  20 C6 72                  JSR WRT         ;PRINT CHAR
0110  E6 00                     INC CHAR        ;NEXT CHAR CODE
0112  4C 07 01                  JMP LOOP        ;CONTINUE
                        ;
0115  00                DONE    BRK             ;STOP & RETURN TO DEMON MONITOR
                        ;
0116  4C 00 01                  JMP CHSET       ;DO IT AGAIN
```

```
.M  0100  20  8D  72  20  EC  72  8D  26
.:  0100  20  8A  72  A9  20  85  00  A5
.:  0108  00  C9  60  F0  08  20  C6  72
.:  0110  E6  00  4C  07  01  00  4C  00
.:  0118  01  ↓
```

Now run the program. Do this by setting the PC to 0100 and using the G command. The listing should look like this:

```
.R  0000 30 00 00 00 FF
.:  0100 ↓
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
```

The program may be continued, causing it to execute again, by typing G:

```
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
```

The CHSET program uses two DEMON monitor functions: CRLF is the DEMON function which causes a carriage-return and line-feed to be typed on the terminal. WRT is the routine which prints the character whose code is in the A register at the time of the call.

( ) 8. Save the CHSET program on paper tape (if your

terminal has a punch) as follows: First, punch some leader tape with the terminal in local mode. Then return to line mode and enter:

.WH 0100 0118 ↓

Turn the punch on after typing the second address, but before typing carriage-return. Then type carriage-return to punch the tape. When punching stops, turn the terminal back to local and type:

;00

and some blank trailer. This is a zero-length record which terminates your tape. See Appendix III for more information on tape formats.

( ) 9. Try re-loading your program using the LH command:

.LH

Now start the reader to load the program. The tape will be read and printed simultaneously. Stop the tape when the end is reached. (Before loading, you may wish to destroy the program in memory to verify that loading from tape actually works.)

( ) 10. Use the M and : commands to load the following program:

```
                        ;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
                        ;

                        ;
0000                            *=0             ;VARIABLE STORAGE IN PAGE ZERO
0000                    BINARY  *=*+1           ;STORAGE FOR CHAR DURING DISSECTION
0001                    CCUNT   *=*+1           ;COUNT OF BITS REMAINING TO PRINT
                        ;
0002                            *=$0100         ;PROGRAM BEGINS ON PAGE ONE
                        ;
                        CRLF    =$728A          ;DEMON CRLF ROUTINE
                        WRT     =$7206          ;DEMON WRITE ROUTINE
                        RCT     =$72E9          ;DEMON READ ROUTINE
                        SPACE   =$7377          ;DEMON SPACE ROUTINE
                        ;
0100 20 8A 72           PBIN    JSR CRLF        ;PRINT CARRIAGE RETURN & LINE FEED
0103 20 E9 72                   JSR RDT         ;GET A CHARACTER
0106 85 00                      STA BINARY      ;SAVE FOR DISSECTION
0108 20 77 73                   JSR SPACE       ;PRINT A SPACE
                        ;
010B A9 08                      LCA #8          ;INITIALIZE BIT COUNT
010D 85 01                      STA COUNT
010F A9 30              PBLCOP  LDA #'0         ;ASSUME ZERO: LOAD ASCII "0"
0111 06 00                      ASL BINARY      ;C=NEXT BIT
0113 BC 02                      BCS PRINT       ;PRINT ZERO
                        ;
0115 A9 31                      LDA #'1         ;LOAD ASCII "1"
                        ;
0117 20 06 72           PRINT   JSR WRT         ;PRINT BINARY DIGIT
011A 06 01                      DEC COUNT       ;COUNT BIT PRINTED
0110 10 F1                      BPL PBLOOP      ;DO NEXT BIT

011E 40 00 01                   JMP PBIN        ;DO IT ALL AGAIN
```

```
.M  0100  20  8D  72  A9  20  85  00  A5
.:  0100  20  8A  72  20  E9  72  85  00
.:  0108  20  77  73  A9  08  85  01  A9
.:  0110  30  06  00  B0  02  A9  31  20
.:  0118  C6  72  C6  01  10  F1  4C  00
.:  0120  01  ↓
```

The purpose of this program is to print the binary representation of an ASCII input character on the terminal. Run the program by starting it at location 0100. Try typing some characters:

```
.R  0116  33  60  00  00  FF
.:  0100  ↓
.G
U  101010101
B  101111011
1  110011101
```

There is obviously something wrong with the program. Bits which should be printed as 1's are 0's and vice versa. (Refer to your 6500 reference card for character codes.) Looking at the program, the problem is that the branch after PBLOOP goes the wrong way! It should be BCC, Branch if Carry Clear (or alternatively, the 1 and 0 loads could be interchanged). Thus, when a one-bit is shifted out of the character, a one should be printed.

Patch the program and try again (the code for BCC is 90).

```
.M   0113  B0  02  A9  31  20  C9  72  C6
.:   0113  90  ↓
.R   7052  31  FC  FF  01  FF
.:   0100  ↓
.G
U  010101010
B  010000100
1  001100010
```

There is, alas, still an error--one too many bits is being printed. The cause of this is a little less obvious. (Do you see it?) To investigate the problem, set a breakpoint at location 011E. Do this by replacing the instruction there with a BRK (code of 00). Then run the program:

```
.M   011E  4C  00  01  EF  4C  00  01  00
.:   011E  00  ↓
.R   7052  31  FC  FF  01  FF
.:   0100  ↓
.G
U  010101010
 *   011F  B0  00  00  AA  FF
```

Once the break has occurred, you are free to investigate the state of the program using DEMON. In particular, check the value in location COUNT:

```
.M   0000  00  FF  1B  2E  31  EA  FO  FA
```

Aha! Although COUNT starts out with a value of 8, it is going one step too far (FF is minus 1). This is because the test instruction, BPL PBLOOP is testing to see whether the count is

greater than or equal to zero. Replace it with BNE (code D0), replace your breakpoint with the original contents at that location, and try the program again.

```
.M 011C 10 F1 00 00 01 EF 4C
.: 011C D0 __ 4C ↓
.R 011F B0 00 00 AA FF
.: 0100 ↓
.G
U 01010101
B 01000010
1 00110001
I 01001001
W 01010111
O 01001111
R 01010010
K 01001011
S 01010011
```

```
                ;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
                ;

                ;
0000                    *=0             ;VARIABLE STORAGE IN PAGE ZERO
0000            BINARY  *=*+1           ;STORAGE FOR CHAR DURING DISSECTION
0001            CCUNT   *=*+1           ;COUNT OF BITS REMAINING TO PRINT
                ;
0002                    *=$0100         ;PROGRAM BEGINS ON PAGE ONE
                ;
                CRLF    =$728A          ;DEMON CRLF ROUTINE
                WRT     =$7206          ;DEMON WRITE ROUTINE
                RCT     =$72E9          ;DEMON READ ROUTINE
                SPACE   =$7377          ;DEMON SPACE ROUTINE
                ;
0100 20 8A 72   PBIN    JSR CRLF        ;PRINT CARRIAGE RETURN & LINE FEED
0103 20 E9 72           JSR RDT         ;GET A CHARACTER
0106 85 00              STA BINARY      ;SAVE FOR DISSECTION
0108 20 77 73           JSR SPACE       ;PRINT A SPACE
                ;
010B A9 08              LDA #8          ;INITIALIZE BIT COUNT
010D 85 01              STA COUNT
010F A9 30      PBLCOP  LDA #'0         ;ASSUME ZERO: LOAD ASCII "0"
0111 06 00              ASL BINARY      ;C=NEXT BIT
0113 90 02              BCC PRINT       ;PRINT ZERO
                ;
0115 A9 31              LDA #'1         ;LOAD ASCII "1"
0117 20 06 72   PRINT   JSR WRT         ;PRINT BINARY DIGIT
011A 06 01              DEC COUNT       ;COUNT BIT PRINTED
011CD0 F1               BNE PBLCCP      ;DO NEXT BIT
                ;
011E 4C 00 01           JMP PBIN        ;DO IT ALL AGAIN
```

CORRECTED PBIN PROGRAM

( ) 11. Save the corrected program using the WH command. Before punching the terminating record (as above in step 8), turn off the punch and set the PC to the start address of the program (0100). Then punch locations 00F6 and 00F7 on the tape, then the terminator (;00), and finally, some trailer:

```
.R   7052  30  37  FF  01  FF
.:   0100  ↓
.WH  00F6  00F7  ↓
;0200F6000101A2
.;00
```

The resulting tape can be loaded and then started as follows:

```
.LH
  ⋮ (program loads in)
.G
```

Locations 00F6 and 00F7 contain the starting address for programs. You may assemble and load your starting address into these locations to make tapes which can be started with a minimum of operator action. The carriage-return delay time may also be set in this manner. See Appendix III.

( ) 12. It is also possible to punch BNPF-format tapes using DEMON. BNPF is the format used by some ROM programmers. The command is similar to that for writing hex tapes:

```
.WB  0100  0127  ↓
```

This command would punch the corrected PBIN program in BNPF

format. Try punching a BNPF tape. (Note that DEMON will not load tapes in this format--use hex format (WH) for saving programs for later loading into your JOLT.)

( ) 13. If you have a high-speed paper tape reader attached to your JOLT system, you can use it to load programs in hex format. The H command switches the load device to and from the high speed reader. If you have a high speed reader, try loading a tape as follows:

.H
.LH

Note that control will not return to the user terminal until a terminator record (;00) is read.

THIS COMPLETES STEP-BY-STEP CHECKOUT
OF THE DEMON MONITOR

HOW TO HAND-ASSEMBLE JOLT PROGRAMS

If you do not use an assembler to convert your JOLT programs to machine language (hexadecimal), you will have to convert your programs manually. Here is a suggested procedure.

The procedure consists of four steps:

STEP 1: Decide which variables and subroutines are to be placed in page zero and assign fixed locations to them.

STEP 2: Look up each instruction in the 6502 code chart and record the operation code in hexadecimal, noting how many bytes of memory are required by each instruction.

STEP 3: Determine the location in hexadecimal of each labelled instruction or variable.

STEP 4: Fill in all remaining values, using the locations determined in Step 3.

When writing a program for hand assembly, it is desirable to split your code into small routines which can be assembled separately. Since you will be loading and debugging your program by hand, you should leave some space for changes so that complete reassembly is not required to fix small problems.

By way of illustration, the PBIN program (used in the Monitor Checkout section) will be hand-assembled:

```
;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
;
;
        *=0                 ;VARIABLE STORAGE IN PAGE ZERO
BINARY  *=*+1               ;STORAGE FOR CHAR DURING DISSECTION
COUNT   *=*+1               ;COUNT OF BITS REMAINING TO PRINT
;
        *=$0100             ;PROGRAM BEGINS ON PAGE ONE
;
CRLF    =$728A              ;DEMON CRLF ROUTINE
WRT     =$72C6              ;DEMON WRITE ROUTINE
RDT     =$72E9              ;DEMON READ ROUTINE
SPACE   =$7377              ;DEMON SPACE ROUTINE
;
PBIN    JSR CRLF            ;PRINT CARRIAGE RETURN & LINE FEED
        JSR RDT             ;GET A CHARACTER
        STA BINARY          ;SAVE FOR DISSECTION
        JSR SPACE           ;PRINT A SPACE
;
        LDA #8              ;INITIALIZE BIT COUNT
        STA COUNT
;
PBLOOP  LDA #'0             ;ASSUME ZERO: LOAD ASCII "0"
        ASL BINARY          ;C=NEXT BIT
        BCC PRINT           ;PRINT ZERO
;
        LDA #'1             ;LOAD ASCII "1"
;
PRINT   JSR WRT             ;PRINT BINARY DIGIT
        DEC COUNT           ;COUNT BIT PRINTED
        BNE PBLOOP          ;DO NEXT BIT
;
        JMP PBIN            ;DO IT ALL AGAIN
```

Step 1

Decide which variables and subroutines are to be placed in page zero and assign fixed locations to them.

Page zero contains locations 0000 to 00FF. The variables that are to reside in page zero must be identified prior to assembling the rest of the program since the mode and length of some instructions depend on whether their operands are in page zero. The sample program has two variables in page zero. They are simply assigned locations sequentially:

```
Loc  Contents  Instruction
                                  .
                                  .
                                  .
0000  XX       BINARY *=*+1          ;STORAGE FOR CHAR DURING DISSECTION
0001  XX       COUNT  *=*+1          ;COUNT OF BITS REMAINING TO PRINT
                                  .
                                  .
                                  .
```

The program does not specify initial values of these locations, so the contents position is marked with X's, indicating that no values will have to be loaded there. In this example, there are no subroutines or other instructions to be assembled in page zero. It will be more convenient for hand assembly if such code, when it occurs, is placed after the variables. Then the position of variables will not depend on the length of preceding instructions.

Step 2

Look up each instruction in the 6502 code chart and record the operation code in hexadecimal, noting how many bytes of memory are required by each instruction.

The length and code of each instruction is determined by its mode. Some instructions, such as JSR and BNE, have only one possible mode, and thus present no difficulty. The mode for other instructions depends on the operand. For example, immediate mode is indicated by a pound sign (#) followed by a value. Instructions of this type use the operation code from the immediate columns of the code chart. The value following the pound sign is put in the second byte of the instruction. For example:

Contents	Instruction
A9 08	LDA #8
A9 31	LDA #'1 ;ASCII "1"

Instructions which have a zero page mode may be assembled in that mode if the operand is in fact in page zero:

```
85 01          STA COUNT
```

The same operation with a non-zero page operand would occupy three bytes:

```
8D __ __       STA ADDR
```

Since symbols other than page zero (and certain pre-assigned addresses like WRT) do not have locations yet, we must leave blank spots to fill in later. Do mark the correct number of spaces for the unknown bytes, since the length of instructions determines the position of instructions which follow. Similarly, branch instructions will have their second bytes blank at this point:

```
D0 __          BNE PBLOOP
```

Thus far, the partially assembled program looks like this:

```
Location    Contents        Instruction
                            ;CHECKOUT PROGRAM -- PRINT BINARY

                                    *=0
0000        XX              BINARY  *=*+1
0001        XX              COUNT   *=*+1

                                    *=$0100

                            CRLF    =$728A
                            WRT     =$72C6
                            RDT     =$72E9
                            SPACE   =$7377

            20 8A 72        PBIN    JSR CRLF
            20 E9 72                JSR RDT
            85 00                   STA BINARY
            20 77 73                JSR SPACE

            A9 08                   LDA #8
            85 01                   STA COUNT

            A9 30           PBLOOP  LDA #'0
            06 00                   ASL BINARY
            90 __                   BCC PRINT

            A9 31                   LDA #'1

            20 C6 72        PRINT   JSR WRT
            C6 01                   DEC COUNT
            D0 __                   BNE PBLOOP

            4C __ __                JMP PBIN
```

Step 3

Determine the location in hexadecimal of each labelled instruction.

It is now possible to fill in the location column, because the length of each instruction is known. Count in hex (0,1,2,..., 9,A,B,C,D,E,F) and write in the locations (of the first bytes)

of instructions and variables which have labels:

Location	Contents	Instruction
		;CHECKOUT PROGRAM -- PRINT BINARY
		*=0
0000	XX	BINARY *=*+1
0001	XX	COUNT *=*+1
		*=$0100
		CRLF =$728A
		WRT =$72C6
		RDT =$72E9
		SPACE =$7377
0100	20 8A 72	PBIN JSR CRLF
	20 E9 72	JSR RDT
	85 00	STA BINARY
	20 77 73	JSR SPACE
	A9 08	LDA #8
	85 01	STA COUNT
010F	A9 30	PBLOOP LDA #'0
	06 00	ASL BINARY
	90 __	BCC PRINT
	A9 31	LDA #'1
0117	20 C6 72	PRINT JSR WRT
	C6 01	DEC COUNT
	D0 __	BNE PBLOOP
	4C __ __	JMP PBIN

Step 4

Fill in the remaining values, using the locations determined in Step 3.

Locations of variables not already entered may now be filled in. Be sure to enter the low half first and the high half second.

For example, location PBIN is at address 0100. It is recorded as:

```
        4C 00 01                JMP PBIN
```

Branches can now be completed by counting the number of bytes from the instruction to the target address. When going forward, count beginning with the first byte following the instruction, up to but not including the first byte at the target address. Thus, the boxed bytes are all that are counted in this example:

```
        90 __                   BCC PRINT
        [A9] [31]               LDA #'1
        20 C6 72    PRINT       JSR WRT
```

When branching backwards, count those bytes from the end of the branch instruction itself (counting both bytes) to and including the first byte of the instruction at the target address. Thus:

```
                                STA COUNT
010F    [A9] [30]       BPLOOP  LDA #'0
        [06] [00]               ASL BINARY
        [90] [02]               BCC PRINT

        [A9] [31]               LDA #'1

0117    [20] [C6] [72]  PRINT   JSR WRT
        [C6] [01]               DEC COUNT
        [D0] [__]               BNE PBLOOP

     4C 00 01                   JMP PBIN
```

Although you could count in hexadecimal, it is easier to count in decimal (base 10). When counting in decimal, count up whether going forward or backward, and look up the correct hexadecimal value on the Branch Chart shown on the next page and also in Appendix V. (If you do count in hexadecimal, backward counts need to be negated. Do this by subtracting the count from 100 hexadecimal. Forward hexadecimal counts may be used without modification.)

The assembly is now complete and ready to be loaded into JOLT.

CHART OF BRANCHES: DECIMAL TO HEXADECIMAL

FORWARD MSD →	0	1	2	3	4	5	6	7	
↓LSD↓									
0	—	16	32	48	64	80	96	112	—
1	1	17	33	49	65	81	97	113	F
2 ←	2	18	34	50	66	82	98	114	E
3	3	19	35	51	67	83	99	115	D
4	4	20	36	52	68	84	100	116	C
5	5	21	37	53	69	85	101	117	B
6	6	22	38	54	70	86	102	118	A
7	7	23	39	55	71	87	103	119	9
8	8	24	40	56	72	88	104	120	8
9	9	25	41	57	73	89	105	121	7
A	10	26	42	58	74	90	106	122	6
B	11	27	43	59	75	91	107	123	5
C	12	28	44	60	76	92	108	124	4
D	13	29	45	61	77	93	109	125	3
E	14	30	46	62	78	94	110	126	2
F	15	31	47	63	79	95	111	127	→ 1
—	16	32	48	64	80	96	112	—	0
									↑LSD↑
	F	E	D	C	B	A	9	8	← MSD BACKWARD

```
Location    Contents        Instruction

                            ;CHECKOUT PROGRAM -- PRINT BINARY

                                    *=0
0000        XX              BINARY  *=*+1
0001        XX              COUNT   *=*+1

                                    *=$0100

                            CRLF    =$728A
                            WRT     =$72C6
                            RDT     =$72E9
                            SPACF   =$7377

0100        20 8A 72        PBIN    JSR CRLF
            20 E9 72                JSR RDT
            85 00                   STA BINARY
            20 77 73                JSR SPACE

            A9 08                   LDA #8
            85 01                   STA COUNT

010F        A9 30           PBLOOP  LDA #'0
            06 00                   ASL BINARY
            90 02                   BCC PRINT

            A9 31                   LDA #'1

0117        20 C6 72        PRINT   JSR WRT
            C6 01                   DEC COUNT
            D0 F1                   BNE PBLOOP

            4C 00 01                JMP PBIN
```

NOTE, HINTS, & RECOMMENDATIONS FOR USING YOUR JOLT MICROCOMPUTER

Storage Allocation

Some care in selecting locations for programs will save programming time and memory space. Page zero storage (0000 to 00FF) is a special resource in your system since it can be used for indirect references (to tables or routines) and since direct references to page zero locations require shorter instructions (2 instead of 3 bytes) for most instructions. Therefore, you will probably want to give priority to putting variables and data in page zero. Be sure to avoid locations at the high end of the page, however, since these are used by DEMON (00E3 to 00FF).

Code and data may also be placed in page one (0100 to 01FF). Be careful, however, to leave sufficient space for the stack (which, with DEMON's initialization, fills from the high end of the page downward, from location 01FF towards location 0100). You should allow at least three bytes for each level of nested subroutine call or interrupt possible in your program, plus space for all data you push onto the stack, plus an additional 20_{10} bytes for DEMON. Failure to leave enough space may cause part of your program or data to be overwritten by the stack, with unpredictable results.

DETAILED DESCRIPTION OF DEMON

DEMON Commands[†]

Command	Description
↓	Set line speed. After RESET, a carriage return is typed to allow DEMON to measure the line speed.
.R	Display user registers. The format is: PC P A X Y S where: PC is the program counter P is the processor status A is the A (accumulator) register X is the X (index) register Y is the Y (index) register S is the stack pointer low byte (high byte is always 01)
.G	Go. Begin execution at user PC location (see R command).
.M addr	Memory examine. DEMON will display the eight bytes beginning at address addr.
.: ADDR data	Alter registers or memory. DEMON allows the user to alter registers (if R command precedes) or memory (if M command precedes). Values for registers or memory locations which are not to be changed need not be typed

[†]Characters typed by the user are underlined. All other characters are typed by the computer. ↓ means carriage return.

—these fields may be skipped by typing spaces instead of data. The remainder of the fields in a line may be left unchanged by typing carriage return. The : command may be repeated to alter subsequent memory locations without the necessity of typing intervening M commands. Note that DEMON automatically types spaces to separate data fields.

.LH Load Hexadecimal. DEMON responds with carriage return, line-feed and loads data in assembler output format from the terminal or high-speed paper tape reader. The format is:

Zero or more leading characters except ";" (usually blank leader)

Any number of records of the form:

;ccaaaadddd....ddssss

where:

cc is the number of bytes in the record in hex

aaaa is the hex address to store the first byte of data

dddd....dd is the data (two hex digits per byte)

ssss is the check-sum, which is the arithmetic sum, to 16 bits, of all the count, address and data bytes represented by the record

A terminating record of zero length, either: ;00 or ;↓

Note that read-after-write and check-sum tests are performed. An error will result in a "?" being typed at the point the error occurred. Data from records with bad check-sums is deposited in memory as received, prior to the error stop.

.H — High-speed/low-speed reader switch. This command switches the load device from the user's terminal to the high-speed reader or vice versa.

.WH addl addh↓ — Write Hexadecimal. An assembler-format tape is generated at the user's terminal. Format is as described above in the LH command description. Note that the address range must be specified with the lower address first. As in the Alter command, DEMON types the space between the address fields.

.WB addl addh↓ — Write BNPF. A BNPF format tape is generated at the user's terminal. Format is one or more records as follows:

```
aaaa BddddddddF BddddddddF BddddddddF BddddddddF
```

where:

aaaa is the address of the first of the four bytes specified in the record. (Note: BNPF conventions require that the letter "B" never occur in the address field. Blanks are substituted by DEMON.)

B is the letter "B", meaning begin data.

dddddddd is eight data bits—P for logical true, N for logical false.

F is the letter "F", meaning finish.

Note that the BNPF format is output as multiples of four bytes. Thus, a multiple of four bytes will always be punched even if a non-multiple of four bytes is specified.

↓↓ Cancel Command. While typing any command, its further effect may normally be terminated by typing one or two carriage returns, as required. During alter (:), carriage return means that no further bytes (or registers) are to be altered.

DEMON Interrupt and Breakpoint Action

BRK

The BRK instruction causes the CPU to interrupt execution, save PC and P registers on the stack, and branch through a vector at locations FFFE and FFFF. DEMON initializes this vector to point to itself on RESET. Unless the user modifies this vector, DEMON will gain control when a BRK instruction is executed, print an asterisk "*" and the registers (as in R command), and wait for user commands. Note that after a BRK which vectors to DEMON, the user's PC points to the byte following the BRK; however, users who choose to handle BRK instructions themselves

should note that BRK acts as a two-byte instruction, leaving the PC (on return via RTI) two bytes past the BRK instruction.

IRQ

Interrupt Request is also vectored through location FFFE. The CPU traps (as with BRK) through this vector when IRQ goes low, provided interrupts are not inhibited. Since this vector is the same as for BRK, DEMON examines the BRK bit in the P register after this type of interrupt. If a BRK did not cause the interrupt, then DEMON will pass control through the UINT vector. Users should normally put the address of their interrupt service routine in the UINT vector location. If an IRQ occurs and UINT has not been set by the user, DEMON reports the unexpected interrupt in the same way as an NMI (see below).

NMI

Non-Maskable Interrupts vector through location FFFA. DEMON initializes this vector at RESET to point to itself. If an NMI occurs, a pound-sign character (#) precedes the asterisk and CPU registers printout. This action is the same for IRQ's if the user has not set this vector to point to his own routine.

RESET or POWER-UP

On RESET or POWER-UP, DEMON takes control, initializes itself and the system, sets defaults for interrupt vectors and waits for a carriage-return input from the user to determine terminal line speed. After carriage-return is typed, control is passed to the user as in BRK.

DEMON Monitor Calls and Special Locations

Call	Address	Action	Arg.	Result	Notes
JSR WRT	72C6	Type of character	A	None	A,X cleared Y preserved
JSR RDT	72E9	Read a character	None	A	X cleared Y not preserved
JSR CRLF	728A	Type CR-LF and delay	None	None	A,X cleared Y preserved
JSR SPACE	7377	Type a space character	None	None	A,X,Y preserved
JSR WROB	72B1	Type a byte in hex	A	None	A,X cleared Y preserved
JSR RDHSR	733D	Read a character from high-speed paper tape reader	None	X--char read A--char trimmed to 7 bits	Y preserved

Function	Locations	Notes
Start Address	00F6,00F7	Set with hex tape on load
CR-LF Delay	00E3	Set on load or with user program (in bit times, minimum of 1. Zero means 256 bits-time delay).
UINT	FFF8	User IRQ vector
NMI Vector	FFFA	Hardware NMI vector
RESET Vector	FFFC	Hardware RESET vector
IRQ Vector	FFFE	Hardware IRQ vector

6502 Processor

1. Addresses for the 6502 processor are always stored low-order byte first, high-order byte second. Thus the lower part of an address is in the location having the lower-numbered address.

2. BRK acts as a two-byte instruction. When entered via BRK, DEMON adjusts the PC so as to make BRK in effect, operate as a one-byte instruction. Users who elect to handle BRK themselves (by changing the hardware IRQ interrupt vector) should be aware of this difference and program accordingly.

3. Certain undefined operation codes will cause the 6502 CPU to "hang-up". When in this "hung-up" state, the CPU can only be stopped with reset. NMI will not work. All other undefined codes may have unpredictable effects. Undefined codes should be avoided.

4. Attempting to read non-existent memory locations will usually yield the high-order part of the address as data. However, this is not true in all cases (indexed loads may respond differently), and should not be relied upon.

The JOLT CPU Board

1. User PIA's are not fully address decoded, which means that each PIA uses 1K of address space. Thus, each PIA register appears every 4 locations in the 1K space used by that PIA. See the JOLT memory map in Appendix III.

2. Unless debouncing is provided for an NMI button, several interrupts can occur when this button is pressed. The result is that DEMON is interrupted in the process of servicing the original interrupt, and the users CPU registers are lost. With proper debouncing, however, CPU registers printed by DEMON after NMI will correctly reflect the state of the machine when the first interrupt occured.

DEMON Memory Usage

DEMON uses the top 29_{10} bytes of page zero (locations 00E3 through 00FF). The user is advised to avoid these locations, except as noted above, if return to DEMON or use of DEMON subroutines is required before RESETing the processor. DEMON also uses the hardware stack when it is in control. Provided the user does not alter the stack pointer during a break, and provided the stack does not overflow, DEMON will restore the stack to its original status before returning to the user's program. The user is advised to use page 1 (the stack page) cautiously, leaving at least 20_{10} bytes for DEMON use during a break or when using other DEMON functions.

SY6502 INSTRUCTION SET SUMMARY

The following symbols are used in this summary:

A	Accumulator
X, Y	Index Registers
M	Memory
P	Processor Status Register
S	Stack Register
L, LOC	Absolute Location
Z	Zero-Page Location
*	Affected
-	Not Affected
+	Sum
$\wedge$	Logical AND
-	Difference
$\forall$	Logical Exclusive Or
$\uparrow$	Transfer From Stack
$\downarrow$	Transfer To Stack
$\rightarrow$	Transfer To
$\leftarrow$	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
#	Immediate Addressing Mode

SY6502 INSTRUCTION SET SUMMARY

Addressing Modes

Mode	Description	# Bytes	Example		
IMPLIED	The operation performed is implied by the instruction.	1*	TAX	AA	Code for transfer A to X
ACCUMULATOR	The operation is performed upon the A register.	1	ROL A	2A	Code for rotate left A
IMMEDIATE	The data accessed is in the second byte of the instruction.	2	LDA #3	A9 03	Code for load A immediate Constant to use
ZERO PAGE	The address within page zero of the data accessed is in the second byte of the instruction.	2	LDA Z	A5 75	Code for load A zero page Low part of address on page zero
ZERO PAGE INDEXED BY X	The second byte of the instruction plus the contents of the X register (without carry) is the address on page zero of the data accessed.	2	LDA Z,X	B5 75	Code for zero page indexed by X Base address on page zero
ZERO PAGE INDEXED BY Y	The second byte of the instruction plus the contents of the Y register (without carry) is the address on page zero of the data accessed.	2	LDX Z,Y	B6 75	Code for zero page indexed by Y Base address on page zero
ABSOLUTE	The address of the data accessed is in the second and third bytes of the instruction.	3	LDA L	AD 47 02	Code for load A absolute Low part of address High part of address

*Except BRK which is two bytes when not using DEMON.

Mode	Description	# Bytes	Example		
INDEXED BY X	The address in the second and third bytes of the instruction, plus the contents of the X register is the address of the data accessed.	3	LDA L,X	BD 47 02	Code for load A indexed by X Low part of base address High part of base address
INDEXED BY Y	The address in the second and third bytes of the instruction, plus the contents of the Y register is the address of the data accessed.	3	LDA L,Y	B9 47 02	Code for load A indexed by Y Low part of base address High part of base address
INDIRECT PRE-INDEXED BY X	The second byte of the instruction plus the contents of the X register (without carry) is the address on page zero of the two-byte address of the data accessed.	2	LDA (Z,X)	A1 75	Code for load A, indirect pre-indexed by X Base address on page zero
INDIRECT POST-INDEXED BY Y	The contents of the page zero two-byte address specified by the second byte in the instruction, plus the contents of the Y register is the address of the data accessed.	2	LDA (Z),Y	B1 75	Code for load A, indirect post-indexed by Y Base address of page zero
RELATIVE BRANCH	The second byte of the instruction contains the offset (in bytes) to branch address.	2	BEQ LOC	F0 07	Code for branch if equal Seven bytes ahead
INDIRECT JUMP	The address in the second and third bytes of the instruction is the address of the address to which the jump is made.	3	JMP (LOC)	6C 47 02	Code for jump indirect Low part of indirect address High part of indirect address

SY6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(L,X)	(L),Y	REL	IND	N	Z	C	I	D	V	B
ADC	A + M + C → A, C Add memory to accumulator with carry			69	65	75		7D	70	79	61	71			*	*	*	-	-	*	-
AND	A ∧ M → A "AND" memory with accumulator			29	25	35		2D	3D	39	21	31			*	*	-	-	-	-	-
ASL	C ← 7 6 5 4 3 2 1 0 ← 0 Shift left one bit (memory or accumulator		0A		06	16		0E	1E						*	*	*	-	-	-	-
BCC	Branch on C = 0 Branch on carry clear												90		-	-	-	-	-	-	-
BCS	Branch on C = 1 Branch on carry set												B0		-	-	-	-	-	-	-
BEQ	Branch on Z = 1 Branch on result zero												F0		-	-	-	-	-	-	-
BIT	A ∧ M, M_7 → N, M_6 → V Test bits in memory with accumulator				24			2C							M_7	*	-	-	-	M_6	-
BMI	Branch on N = 1 Branch on result minus												30		-	-	-	-	-	-	-
BNE	Branch on Z = 0 Branch on result not zero												D0		-	-	-	-	-	-	-
BPL	Branch on N = 0 Branch on result plus												10		-	-	-	-	-	-	-
BRK	Forced interrupt PC↓ P↓ Force break	00													-	-	-	1	-	-	1

SY6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(L,X)	(L),Y	REL	IND	N	Z	C	I	D	V	B
BVC	Branch on V = 0 Branch on overflow clear												50		-	-	-	-	-	-	-
BVS	Branch on V = 1 Branch on overflow set												70		-	-	-	-	-	-	-
CLC	0 → C Clear carry flag	18													-	-	0	-	-	-	-
CLD	0 → D Clear decimal mode flag	D8													-	-	-	-	0	-	-
CLI	0 → I Clear interrupt disable flag	58													-	-	-	0	-	-	-
CLV	0 → V Clear overflow flag	B8													-	-	-	-	-	0	-
CMP	A - M Compare memory and accumulator			C9	C5	D5		CD	DD	D9	C1	D1			*	*	*	-	-	-	-
CPX	X - M Compare memory and index X			E0	E4			EC							*	*	*	-	-	-	-
CPY	Y - M Compare memory and index Y			C0	C4			CC							*	*	*	-	-	-	-
DEC	M - 1 → M Decrement memory by one				C6	D6		CE	DE						*	*	-	-	-	-	-
DEX	X - 1 → X Decrement index X by one	CA													*	*	-	-	-	-	-

SY6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(L,X)	(L),Y	REL	IND	N	Z	C	I	D	V	B
DEY	Y - 1 → Y Decrement index Y by one	88													*	*	-	-	-	-	-
EOR	A ⊻ M → A "Exclusive-Or" memory with accumulator			49	45	55		4D	5D	59	41	51			*	*	-	-	-	-	-
INC	M + 1 → M Increment memory by one				E6	F6		EE	FE						*	*	-	-	-	-	-
INX	X + 1 → X Increment Index X by one	E8													*	*	-	-	-	-	-
INY	Y + 1 → Y Increment index Y by one	C8													*	*	-	-	-	-	-
JMP	(PC + 1) → PCL (PC + 2) → PCH Jump to new location							4C						6C	-	-	-	-	-	-	-
JSR	PC + 2 ↓, (PC + 1) → PCL (PC + 2) → PCH Jump to new location saving return address							20							-	-	-	-	-	-	-
LDA	M → A Load accumulator with memory			A9	A5	B5		AD	BD	B9	A1	B1			*	*	-	-	-	-	-
LDX	M → X Load index X with memory			A2	A6		B6	AE		BE					*	*		-	-	-	-
LDY	M → Y Load index Y with memory			A0	A4	B4		AC	BC						*	*	-	-	-	-	-

SY6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(Z,X)	(Z),Y	REL	IND	N	Z	C	I	D	V	B
LSR	0 → 7 6 5 4 3 2 1 0 → C Shift right one bit (memory or accumulator)		4A		46	56		4E	5E						0	*	*	-	-	-	-
NOP	No Operation	EA													-	-	-	-	-	-	-
ORA	A V M → A "OR" memory with accumulator			09	05	15		0D	1D	19	01	11			*	*	-	-	-	-	-
PHA	A ↓ Push accumulator on stack	48													-	-	-	-	-	-	-
PHP	P ↓ Push processor status on stack	08													-	-	-	-	-	-	1
PLA	A ↑ Pull accumulator from stack	68													*	*	-	-	-	-	-
PLP	P ↑ Pull processor status from stack	28													From Stack						
ROL	M or A 7 6 5 4 3 2 1 0 ← C ← Rotate one bit left (memory or accumulator)		2A		26	36		2E	3E						*	*	*	-	-	-	-
ROR	Rotate One Bit Right (Memory or Accumulator)		6A		66	76		6E	7E						*	*	*	-	-	-	-
RTI	P ↑ PC ↑ Return from interrupt	40													From Stack						
RTS	PC ↑, PC + 1 → PC Return from subroutine	60													-	-	-	-	-	-	-

SY6502 INSTRUCTION SET SUMMARY

Instr	Description	Mode													Condition Codes						
		IMP	ACC	IMM	Z	Z,X	Z,Y	ABS	L,X	L,Y	(L,X)	(L),Y	REL	IND	N	Z	C	I	D	V	B
SBC	A - M - $\overline{C}$ → A Note: $\overline{C}$ = Borrow Subtract memory from accumulator with borrow			E9	E5	F5		ED	FD	F9	E1	F1			*	*	*	-	-	*	-
SEC	1 → C Set carry flag	38													-	-	1	-	-	-	-
SED	1 → D Set decimal mode flag	F8													-	-	-	-	1	-	-
SEI	1 → I Set interrupt disable flag	78													-	-	-	1	-	-	-
STA	A → M Store accumulator in memory				85	95		8D	9D	99	81	91			-	-	-	-	-	-	-
STX	X → M Store index X in memory				86		96	8E							-	-	-	-	-	-	-
STY	Y → M Store index Y in memory				84	94		8C							-	-	-	-	-	-	-
TAX	A → X Transfer accumulator to index X	AA													*	*	-	-	-	-	-
TAY	A → Y Transfer accumulator to index Y	A8													*	*	-	-	-	-	-
TSX	S → X Transfer stack pointer to index X	BA													*	*	-	-	-	-	-
TXA	X → A Transfer index X to accumulator	8A													*	*	-	-	-	-	-
TXS	X → S Transfer index X to stack pointer	9A													-	-	-	-	-	-	-
TYA	Y → A Transfer index Y to accumulator	98													*	*	-	-	-	-	-

CHART OF BRANCHES: DECIMAL TO HEXADECIMAL

FORWARD MSD →	0	1	2	3	4	5	6	7	
↓LSD↓									
0	—	16	32	48	64	80	96	112	—
1	1	17	33	49	65	81	97	113	F
2	2	18	34	50	66	82	98	114	E
3	3	19	35	51	67	83	99	115	D
4	4	20	36	52	68	84	100	116	C
5	5	21	37	53	69	85	101	117	B
6	6	22	38	54	70	86	102	118	A
7	7	23	39	55	71	87	103	119	9
8	8	24	40	56	72	88	104	120	8
9	9	25	41	57	73	89	105	121	7
A	10	26	42	58	74	90	106	122	6
B	11	27	43	59	75	91	107	123	5
C	12	28	44	60	76	92	108	124	4
D	13	29	45	61	77	93	109	125	3
E	14	30	46	62	78	94	110	126	2
F	15	31	47	63	79	95	111	127	1
—	16	32	48	64	80	96	112	—	0
									↑LSD↑
	F	E	D	C	B	A	9	8	← MSD BACKWARD

Forward Branches

Count in decimal from the end of the branch instruction to target address. Do not count bytes in either the branch or target instruction. Find the count in the center of the chart. Use the Most-Significant-Digit at the top of the column, and the Least-Significant-Digit at the left of the row.

Reverse Branches

Count in decimal from the end of the branch to the beginning of the target instruction. Count all bytes in both instructions. Find the count in the center of the chart. Use the Most-Significant-Digit at the bottom of the column, and the Least-Significant-Digit at the right of the row.

Example

Forward 10 gives 0A. Backward 10 gives F6.

Chart Idea Credit

Ray Boaz, Homebrew Computer Club Newsletter, Volume 1, Number 7, September 1975.

```
CARD # LOC     CODE        CARD
     1                    ;
     2                    ;  DEMON DEBUG MONITOR
     3                    ;
     4                    ; PROMPTING CHARACTER IS A PERIOD
     5                    ;
     6                    ;
     7                    ; DISPLAY COMMANDS
     8                    ;
     9                    ; .R               DISPLAY REGISTERS (PC,P,A,X,Y,SP)
    10                    ; .M  ADDR         DISPLAY MEMORY (8 BYTES BEGINNING AT ADDR)
    11                    ;
    12                    ; ALTER COMMAND (:)
    13                    ; .:  DATA         ALTERS PREVIOUSLY DISPLAYED ITEM OR NEXT ITEM
    14                    ;
    15                    ; PAPER TAPE I/O COMMANDS
    16                    ; .LH              LOAD HEX TAPE
    17                    ; .WB  ADDR1 ADDR2 WRITE BNPF TAPE (FROM LOW ADDR1 TO HIGH ADDR2)
    18                    ; .WH  ADDR1 ADDR2 WRITEHEXF TAPE (FROM LOW ADDR1 TO HIGH ADDR2)
    19                    ;
    20                    ; CONTROL COMMANDS
    21                    ; .G               GO, CONTINUE EXECUTION FROM CURRENT PC ADDRESS
    22                    ; .H               TOGGLES HIGH-SPEED-READER OPTION
    23                    ;                       (IF ITS ON, TURNS IT OFF; IF OFF, TURNS ON)
    24                    ;
    25                    ; BRK AND NMI ENTRY POINTS TO DEMON
    26                    ;   DEMON IS NORMALLY ENTERED WHEN A 'BRK' INSTRUCTION IS
    27                    ;   ENCOUNTERED DURING PROGRAM EXECUTION. AT THAT
    28                    ;   TIME CPU REGISTERS ARE OUTPUT: * PC P A X Y SP
    29                    ;   AND CONTROL IS GIVEN TO THE KEYBOARD.
    30                    ; USER MAY ENTER DEMON BY PROGRAMMED BRK OR INDUCED NMI. NMI
    31                    ;   ENTRIES CAUSE A '#' TO PRECEDE THE '*' IN THE CPU REGISTER
    32                    ;   PRINTOUT FORMAT.
    33                    ;
    34                    ; NON-BRK IRQ (EXTERNAL DEVICE) INTERRUPT HANDLING
    35                    ;   A NON-BRK IRQ INTERRUPT CAUSES AN INDIRECT JUMP TO THE ADDRESS
```

```
CARD # LOC     CODE        CARD
   37                    ;   LOCATED  AT  'UNIT' (FFF8). THIS LOCATION CAN BE SET
   38                    ;   USING THE ALTER COMMAND, OR LOADER AUTOMATICALLY IN PAPER TAPE
   39                    ;   FORM WITH THE LH CMD IF THE USER ASSIGNS HIS IRQ INTERRUPT
   40                    ;   VECTOR TO $FFF8 IN THE SOURCE ASSEMBLY PROGRAM.
   41                    ; IF NOT RESET BY THE USER, UNIT IS SET TO CAUSE EXTERNAL
   42                    ;   DEVICE INTERRUPTS TO ENTER DEMON AS NMI'S. I.E.,
   43                    ;   IF A NMI OCCURS WITHOUT AN INDUCED NMI SIGNAL, IT IS
   44                    ;   AN EXTERNAL DEVICE INTERUPT.
   45                    ;
   46                    ; SETTING AND RESETTING PROGRAM BREAKPOINTS
   47                    ; BREAKPOINTS ARE SET AND RESET USING THE MEMORY DISPLAY
   48                    ;   AND ALTER COMMANDS. BRK HAS A '00' OPERATION CODE.
   49                    ; TO SET A BREAK POINT SIMPLY DISPLAY THE MEMPOY LOCATION
   50                    ;   (FIRST INSTRUCTION BYTE)  AT WHICH THE BREAKPOINT IS
   51                    ;   TO BE PLACED THEN ALTER THE  LOCATION TO '00'. THERE IS
   52                    ;   NO LIMIT TO THE NUMBER OF BREAKPOINTS THAT CAN BE
   53                    ;   ACTIVE AT ONE TIME.
   54                    ; TO RESET A BREAKPOINT, RESTORE THE ALTERED MEMORY LOCATION
   55                    ;   TO ITS ORIGINAL VALUE.
   56                    ; WHEN AND IF A BREAKPOINT IS ENCOUNTERED DURING EXECUTION,
   57                    ;   THE BREAKPOINT DATA PRECEDED BY AN '*' IS DISPLAYED.
   58                    ;
   59                    ;
   60                    ;
   61                    MDBK      =%00010110          ; X,X,POR,DATA-AVAIL,GOT-DATA,SERIAL-OUT,IN
   62                    DAVAIL    =$08
   63                    GOTDAT    =$04
   64                    IOBASE    =$6E00
   65                    MPA       =IOBASE+0
   66                    MDA       =IOBASE+1
   67                    MPB       =IOBASE+2
   68                    MDB       =IOBASE+3
   69                    MCLKIT    =IOBASE+4
   70                    MCLKRD    =IOBASE+4
   71                    MCLKIP    =IOBASE+5
   72                    UNIT      =$FFF8
   73                    NCMDS     =7
   74                    MP0       =$7000
   75                    MP1       =$7100
   76                    MP2       =$7200
   77                    MP3       =$7300
   78                    ;
   79                    ; ZERO PAGE MONITOR RESERVE AREA
   80                    ;
   81                    CRDLY     =227           ;DELAY FOR CR IN BIT TIMES
```

```
CARD # LOC      CODE        CARD
   83                      WRAP      =228
   84                      DIFF      =229
   85                      HSPTR     =231
   86                      HSROP     =232
   87                      PREVC     =233
   88                      MAJORT    =234
   89                      MINORT    =235
   90                      ACMD      =236
   91                      TMP0      =238
   92                      TMP2      =240
   93                      TMP4      =242
   94                      TMP6      =244
   95                      PCL       =246
   96                      PCH       =247
   97                      FLGS      =248
   98                      ACC       =249
   99                      XR        =250
  100                      YR        =251
  101                      SP        =252
  102                      SAVX      =253
  103                      TMPC      =254
  104                      TMPC2     =255
  105                      RCNT      =TMPC
  106                      LCNT      =TMPC2
  107                      ;
  108                      ;    64 BYTE RAM MONITOR RESERVE AREA
  109                      ;
  110                      RAM64     =$FF00
  111  0000                          *=RAM64
```

```
CARD # LOC      CODE        CARD
  113                       ;
  114                       ;
  115                       ; DEMON PAGE ZERO (RELATIVE)
  116  FF00                        *=MP0
  117                       ;
  118  7000  85 F9          NMINT    STA   ACC            ; SAVE A
  119  7002  A9 23                   LDA   #'#            ; SET A=# TO INDICATE NMINT ENTRY
  120  7004  D0 55                   BNE   B3
  121                       ;
  122  7006  A9 16          RESET    LDA   #MDBK          ; INIT DIR REG, PCR TC 1 RELOCATES
  123                       ;
  124  7008  8D 03 6E                STA   MDB
  125                       ;
  126  700B  A2 08                   LDX   #8             ; X=0
  127  700D  BD F7 73       R1       LDA   INTVEC-1,X     ; INITILIZE INT VECTORS
  128  7010  9D F7 FF                STA   UNIT-1,X
  129  7013  CA                      DEX
  130  7014  D0 F7                   BNE   R1
  131                       ;
  132  7016  86 EA                   STX   MAJORT         ; INIT MAJOR T COUNT TO ZERO
  133  7018  86 E7                   STX   HSPTR          ; CLEAR HSPTR FLAGS
  134  701A  86 E8                   STX   HSROP
  135  701C  CA                      DEX                  ; X=FF
  136  701D  9A                      TXS                  ; SP=FF
  137                       ;
  138                       ;                            ; COMPUTE BIT TIME CONSTANT, X=FF
  139                       ;
  140  701E  A0 01                   LDY   #1             ; SET TO MEASURE 2 BITS
  141  7020  84 E3                   STY   CRDLY          ; INIT CR DELAY TIME PARM
  142  7022  AD 02 6E       R0       LDA   MPB            ; WAIT FOR START
  143  7025  4A                      LSR   A
  144  7026  90 FA                   BCC   R0
  145                       ;
  146  7028  8E 04 6E       R2       STX   MCLKIT         ; START CLOCK INITIALLY WITH FF
  147  702B  AD 05 6E       R3       LDA   MCLKIP
  148  702E  10 04                   BPL   R4
  149  7030  E6 EA                   INC   MAJORT         ; COUNT MAJOT T
  150  7032  D0 F4                   BNE   R2             ; GO RESTART CLOCK WITH X=FF
  151                       ;
  152  7034  98             R4       TYA
  153  7035  4D 02 6E                EOR   MPB
  154  7038  29 01                   AND   #1
  155  703A  F0 EF                   BEQ   R3             ; WAIT FOR Y BIT 0 AND SERIAL-IN NOT EQU
  156  703C  88                      DEY
  157  703D  10 EC                   BPL   R3             ; LOOP UNITL START OF BIT 2
  158                       ;
  159  703F  AD 04 6E                LDA   MCLKRD
  160  7042  49 FF                   EOR   #$FF           ; COMPLEMENT RESIDUE
  161  7044  4A             R5       LSR   A              ; HALVE IT
  162  7045  46 EA                   LSR   MAJORT         ; HALVE MAJOR
  163  7047  90 02                   BCC   R6
  164  7049  09 80                   ORA   #$80           ; PROPAGATE HO TO LO
```

```
CARD #  LOC    CODE        CARD
  166   704B   C8          R6        INY
  167   704C   F0 F6                  BEQ   R5
  168   704E   85 E8                  STA   MINCRT
  169                      ;
  170   7050   58                     CLI                    ; ENABLE INTS
  171   7051   00                     BRK                    ; ENTER DEMON BY BRK
  172                      ;
  173   7052   85 F9       INTRQ     STA   ACC              ; SAVE ACC
  174   7054   68                     PLA                    ; FLAGS TO A
  175   7055   48                     PHA                    ; RESTORE STACK STATUS
  176   7056   29 10                  AND   #$10             ; TEST BRK FLAG
  177   7058   F0 27                  BEQ   BX               ; USER INTERRUPT
  178                      ;
  179   705A   0A                     ASL   A                ; SET A=SPACE (10 X 2 = 20)
  180   705B   85 FE       B3        STA   TMPC             ; SAVE INT TYPE FLAG
  181   705D   D8                     CLD                    ; CLEAR DECIMAL MODE
  182   705E   4A                     LSR   A                ; # IS ODD, SPACE IS EVEN
  183                      ;                               ; SET CY FOR PC BRK CORRECTION
  184   705F   86 FA                  STX   XR               ; SAVE X
  185   7061   84 FB                  STY   YR               ; Y
  186   7063   68                     PLA
  187   7064   85 F8                  STA   FLGS             ; FLAGS
  188   7066   68                     PLA
  189   7067   69 FF                  ADC   #$FF             ; CY SET TO PC-1 FOR BRK
  190   7069   85 F6                  STA   PCL
  191   706B   68                     PLA
  192   706C   69 FF                  ADC   #$FF
  193   706E   85 F7                  STA   PCH
  194   7070   BA                     TSX
  195   7071   86 FC                  STX   SP               ; SAVE ORIG SP
  196                      ;
  197   7073   20 8A 72    B5        JSR   CRLF
  198   7076   A6 FE                  LDX   TMPC
  199                      ;
  200   7078   A9 2A                  LDA   #'*
  201   707A   20 C0 72               JSR   WRTWO
  202   707D   A9 52                  LDA   #'R              ; SET FOR R DISPLAY TO PERMIT
  203   707F   D0 16                  BNE   S0               ;  IMMEDIATE ALTER FOLLOWING BREAKPOINT.
  204                      ;
  205   7081   A5 F9       BX        LDA   ACC
  206   7083   6C F8 FF               JMP   (UNIT)           ; CONTROL TO USER IRQ SERVICE ROUTINE
  207                      ;
  208   7086   A9 00       START     LDA   #0               ; NEXT COMMAND FROM USER
  209   7088   85 E7                  STA   HSPTR            ; CLEAR H. S. PAPER TAPE FLAG
  210   708A   85 E4                  STA   WRAP             ; CLEAR ADDRESS WRAP-AROUND FLAG
  211   708C   20 8A 72               JSR   CRLF
  212   708F   A9 2E                  LDA   #'.              ; TYPE PROMPTING '.'
  213   7091   20 C6 72               JSR   WRDC
  214   7094   20 E9 72               JSR   RDDC             ; READ CMD. CHAR RETURNED IN A
  215                      ;
  216   7097   A2 06       S0        LDX   #NCMDS-1         ; LOOK-UP CMD
```

```
CARD # LOC    CODE        CARD
  218  7099  DD 06 71  S1        CMP   CMDS,X
  219  709C  D0 19              BNE   S2
  220                    ;
  221  709E  A5 FD              LDA   SAVX          ; SAVE PREVIOUS CMD
  222  70A0  85 E9              STA   PREVC
  223  70A2  86 FD              STX   SAVX          ; SAVE CURRENT CMD INDEX
  224  70A4  A9 71              LDA   #MP1/256      ; JUMP INDIRECT TO CMD MODE
  225  70A6  85 ED              STA   ACMD+1        ;   ALL CMD CODE BEGINES ON MP1
  226  70A8  BD 0D 71           LDA   ADRS,X
  227  70AB  85 EC              STA   ACMD
  228  70AD  E0 03              CPX   #3            ; IF :, R CR M (C, 1, CR 2) SPACE  2
  229  70AF  B0 03              BCS   IJMP
  230  70B1  20 74 73           JSR   SPAC2
  231                    ;
  232  70B4  6C EC 00  IJMP      JMP   (ACMD)
  233                    ;
  234  70B7  CA        S2        DEX
  235  70B8  10 DF              BPL   S1            ; LOOP FOR ALL COMMANDS
  236                    ;
  237  70BA  A9 3F     ERROPR    LDA   #'?           ; OPERATOR ERR, TYPE '?', RESTART
  238  70BC  20 C6 72           JSR   WROC
  239  70BF  90 C5              BCC   START         ; JMP START (WROC RETURNS CY=C)
  240                    ;
  241  70C1  38        DCMP      SEC                 ; TMP2-TMP0 DOUBLE SUBTRACT
  242  70C2  A5 F0              LDA   TMP2
  243  70C4  E5 EE              SBC   TMP0
  244  70C6  85 E5              STA   DIFF
  245  70C8  A5 F1              LDA   TMP2+1
  246  70CA  E5 EF              SBC   TMP0+1
  247  70CC  A8                 TAY                 ; RETURN HIGH ORDER PART IN Y
  248  70CD  05 E5              ORA   DIFF          ; OR LO FOR EQU TEST
  249  70CF  60                 RTS
  250  70D0  A5 EE     PUTP      LDA   TMP0          ; MOVE TMP0 TO PCH,PCL
  251  70D2  85 F6              STA   PCL
  252  70D4  A5 EF              LDA   TMP0+1
  253  70D6  85 F7              STA   PCH
  254  70D8  60                 RTS
  255                    ;
  256                    ;
  257  70D9  A9 00     ZTMP      LDA   #0            ; CLEAR REGS
  258  70DB  95 EE              STA   TMP0,X
  259  70DD  95 EF              STA   TMP0+1,X
  260  70DF  60                 RTS
  261                    ;
  262                    ;  READ AND STORE BYTE. NO STORE IF SPACE OR RCNT=0.
  263                    ;
  264  70E0  20 B3 73  BYTE      JSR   RD0B          ; CHAR IN A, CY=0 IF SP
  265  70E3  90 10              BCC   BY3           ; SPACE
  266                    ;
  267  70E5  A2 00              LDX   #0            ; STORE BYTE
  268  70E7  81 EE              STA   (TMP0,X)
  269                    ;
```

```
CARD #  LCC     CODE        CARD
  271   70E9  C1 EE          CMP   (TMP0,X)      ; TEST FOR VALID WRITE (RAM)
  272   70EB  F0 05          BEQ   BY2
  273   70ED  68             PLA                 ; ERR, CLEAR JSR ADR IN STACK
  274   70EE  68             PLA
  275   70EF  4C BA 70       JMP   ERROPR
  276                      ;
  277   70F2  20 7C 72  BY2  JSR   DADD          ; INCR CKSUM
  278   70F5  20 97 73  BY3  JSR   INCTMP        ; GO INCR TMP0 ADR
  279   70F8  C6 FE          DEC   RCNT
  280   70FA  60             RTS
  281                      ;
  282   70FB  A9 F8     SETR LDA   #FLGS          ; SET TO ACCESS REGS
  283   70FD  85 EE          STA   TMP0
  284   70FF  A9 00          LDA   #0
  285   7101  85 EF          STA   TMP0+1
  286   7103  A9 05          LDA   #5
  287   7105  60             RTS
  288                      ;
  289   7106  3A        CMDS .BYTE ':'
  290   7107  52             .BYTE 'R'
  291   7108  4D             .BYTE 'M'
  292   7109  47             .BYTE 'G'
  293   710A  48             .BYTE 'H'
  294   710B  4C             .BYTE 'L'
  295   710C  57             .BYTE 'W'           ; W MUST BE LAST CMD IN CHAIN
  296   710D  3A        ADRS .BYTE ALTER-MP1
  297   710E  14             .BYTE DSPLYR-MP1
  298   710F  1C             .BYTE DSPLYM-MP1
  299   7110  5C             .BYTE GO-MP1
  300   7111  6F             .BYTE HSP-MP1
  301   7112  74             .BYTE LH-MP1
  302   7113  C2             .BYTE WO-MP1
```

```
CARD # LOC    CODE        CARD
  304                     ;
  305                     ;
  306                     ; NOTE -- ALL CMD CODE MUST BEGIN ON MP1
  307                     ;
  308                     ; DISPLAY REG CMD  - PC,A,P,X,Y, AND SP
  309                     ;
  310  7114  20 A6 72  DSPLYR   JSR   WRPC          ; REITE PC
  311  7117  20 FB 70           JSR   SETR
  312  711A  D0 07              BNE   M0           ; USE DISPLAY
  313                     ;
  314  711C  20 A4 73  DSPLYM   JSR RDOA           ; READ MEM ADR INTO TMP0
  315  711F  90 16              BCC   ERRS1
  316  7121  A9 08              LDA   #8
  317  7123  85 FE      M0      STA   TMPC
  318  7125  A0 00              LDY   #0
  319  7127  20 77 73  M1       JSR   SPACE         ; TYPE 8 BYTES OF MEMPOY
  320  712A  B1 EE              LDA   (TMP0),Y      ; (TMP0) PRESERVER FOR POSS ALTER
  321  712C  20 B1 72           JSR   WROB
  322  712F  C8                 INY                 ; INCR INDEX
  323  7130  C6 FE              DEC   TMPC
  324  7132  D0 F3              BNE   M1
  325  7134  4C 86 70  BEQS1    JMP   START
  326                     ;
  327  7137  4C BA 70  ERRS1    JMP   ERROPR
  328                     ;
  329                     ;  ALTER LAST DISPLAYED ITEM ( DR IN TMP0)
  330                     ;
  331  713A  C6 E9      ALTER   DEC   PREVC         ; R INDEX = 1
  332  713C  D0 0D              BNE   A3
  333                     ;
  334  713E  20 A4 73           JSR   RDOA          ; CY=C IF SP
  335  7141  90 C3              BCC   A2            ; SPACE
  336  7143  20 D0 70           JSR   PUTP          ; ALTER PC
  337  7146  20 FB 70  A2       JSR   SETR          ; ALTER R'S
  338  7149  D0 05              BNE   A4            ; JMP A4 (SETR RETURNS ACC = 5)
  339  714B  20 9A 72  A3       JSR   WROA          ; ALTER M, TYPE ADR
  340  714E  A9 08              LDA   #8
  341                     ;
  342  7150  85 FE      A4      STA   RCNT
  343  7152  20 77 73  A5       JSR   SPACE         ; PRESERVES Y
  344  7155  20 E0 70           JSR   BYTE
  345  7158  D0 F8              BNE   A5
  346  715A  F0 D8      A9      BEQ   BEQS1
  347                     ;
  348  715C  A5 FC      GO      LDX   SP
  349  715E  9A                 TXS                 ; ORIG OR NEW SP VALUE TO SP
  350  715F  A5 F7              LDA   PCH
  351  7161  48                PHA
  352  7162  A5 F6              LDA   PCL
  353  7164  48                PHA
  354  7165  A5 F8              LDA   FLGS
  355  7167  48                PHA
```

```
CARD #  LOC    CODE        CARD
  357   7168   A5 F9             LDA   ACC
  358   716A   A6 FA             LDX   XR
  359   716C   A4 FB             LDY   YR
  360   716E   40                RTI
  361                     ;
  362   716F   E6 E8     HSP     INC   HSROP          ; TOGGLE BIT C
  363   7171   4C 86 70           JMP START
  364                     ;
  365   7174   20 E9 72  LH      JSR   RDOC           ; READ SECOND COMMAND CHAR
  366   7177   20 8A 72           JSR   CRLF
  367   717A   A6 E8              LDX   HSROP          ; ENABLE PTR OPTION IF SET
  368   717C   86 E7              STX   HSPTR
  369   717E   20 E9 72  LH1     JSR   RDOC
  370   7181   C9 3B              CMP   #';            ; FIND NEXT RECORD MARK
  371   7183   D0 F9              BNE   LH1
  372                     ;
  373   7185   A2 04              LDX   #4
  374   7187   20 D9 70           JSR   ZTMP           ; CLEAR CKSUM REGS TMP4
  375   718A   20 B3 73           JSR   RDOB
  376   718D   D0 C6              BNE   LH2
  377                     ;
  378   718F   A2 00              LDX   #0             ; CLEAR HS RDR FLAG
  379   7191   86 E7              STX   HSPTR
  380   7193   F0 9F              BEQ   BEQS1          ; FINISHED
  381                     ;
  382   7195   85 FE     LH2     STA   RCNT           ; RCNT
  383   7197   20 7C 72           JSR   DADD           ; ROD LNGH TO CKSUM
  384   719A   20 B3 73           JSR   RDOB           ; SA HO TC TMP0
  385   719D   85 EF              STA   TMP0+1
  386   719F   20 7C 72           JSR   DADD           ; ADD TO CKSUM
  387   71A2   20 B3 73           JSR   RDOB
  388   71A5   85 EE              STA   TMP0
  389   71A7   20 7C 72           JSR   DADD
  390                     ;
  391   71AA   20 E0 70  LH3     JSR   BYTE           ; BYTE SUB/R DECRE RCNT ON EXIT
  392   71AD   D0 F8              BNE   LH3
  393   71AF   20 A4 73           JSR   RDOA           ; CKSUM FROM HEX TO TMP0
  394   71B2   A5 F2              LDA   TMP4
  395   71B4   85 F0              STA   TMP2
  396   71B6   A5 F3              LDA   TMP4+1
  397   71B8   85 F1              STA   TMP2+1
  398   71BA   20 C1 70           JSR   DCMP
  399   71BD   F0 BF              BEQ   LH1
  400   71BF   4C BA 70  ERRP1   JMP   ERROPR
  401                     ;
  402   71C2   20 E9 72  WO      JSR   RDOC           ; RD 2ND CMD CHAR
  403   71C5   85 FE              STA   TMPC
  404   71C7   20 77 73           JSR   SPACE
  405   71CA   20 A4 73           JSR   RDOA
  406   71CD   20 87 73           JSR   T2T2           ; SA TO TMP2
  407   71D0   20 77 73           JSR   SPACE          ; SPACE BEFORE NEXT ADDRESS
  408   71D3   20 A4 73           JSR   RDOA
```

```
CARD # LOC     CODE        CARD
  410  71D6  20 87 73          JSR   T2T2          ; SA TO TMP0, BA TO TMP2
  411  71D9  20 E9 72          JSR   RDOC          ; DELAY FOR FINAL CR
  412  71DC  A5 FE             LDA   TMPC
  413                    ;
  414  71DE  C9 48             CMP   #'H
  415  71E0  D0 59             BNE   WB
  416                    ;
  417  71E2  A6 E4      WH0    LDX   WRAP          ; IF ADDR HAS WRAPPED AROUND
  418  71E4  D0 52             BNE   BCCST         ; THEN TREMINATE WRITE OPERATION
  419                    ;
  420  71E6  20 8A 72          JSR   CRLF
  421  71E9  A2 18             LDX   #24
  422  71EB  86 FE             STX   RCNT          ; RCNT=24
  423  71ED  A2 04             LDX   #4            ; CLEAR CKSUM
  424  71EF  20 D9 70          JSR   ZTMP
  425                    ;
  426  71F2  A9 38             LDA   #'8
  427  71F4  20 C6 72          JSR   WROC          ; WR RCD MARK
  428                    ;
  429  71F7  20 C1 70          JSR   DCMP          ; EA-SA (TMP0+2-TMP0) DIFF IN LOC DIFF,+1
  430  71FA  98                TYA                 ; MS BYTE OR DIFF
  431  71FB  D0 0A             BNE   WH1
  432  71FD  A5 E5             LDA   DIFF
  433  71FF  C9 17             CMP   #23
  434  7201  B0 04             BCS   WH1           ; DIFF GT 24
  435  7203  85 FE             STA   RCNT          ; INCR LAST TCNT
  436  7205  E6 FE             INC   RCNT
  437  7207  A5 FE      WH1    LDA   RCNT
  438  7209  20 7C 72          JSR   DADD         ; ADD TO CKSUM
  439  720C  20 B1 72          JSR   WROB          ; RCD CNT IN A
  440  720F  A5 EF             LDA   TMP0+1        ; SA HO
  441  7211  20 7C 72          JSR   DADD
  442  7214  20 B1 72          JSR   WROB
  443  7217  A5 EE             LDA   TMP0          ; SA LO
  444  7219  20 7C 72          JSR   DADD
  445  721C  20 B1 72          JSR   WROB
  446                    ;
  447  721F  A0 00      WH2    LDY   #0
  448  7221  B1 EE             LDA   (TMP0),Y
  449                    ;                         ; INC CKSUM, PRESERVES A
  450  7223  20 7C 72          JSR   DADD
  451  7226  20 B1 72          JSR   WROB
  452  7229  20 97 73          JSR   INCTMP        ; INC SA
  453  722C  C6 FE             DEC   RCNT
  454  722E  D0 EF             BNE   WH2           ; LOOP FOR UP TO 24 BYTES
  455                    ;
  456  7230  20 9E 72          JSR   WROA4         ; WRITE CKSUM
  457                    ;
  458  7233  20 C1 70          JSR   DCMP
  459  7236  B0 AA             BCS   WH0           ; LOOP WHILE EA GT OR = SA
  460  7238  4C 66 70  BCCST   JMP   START;
```

```
CARD # LOC    CODE        CARD
  462                     ;
  463  723B  E6 FD        WB        INC   SAVX          ; SAVX TC = NCMDS FOR ASCII SUB/R
  464  723D  A5 E4        WB1       LDA   WRAP          ; IF ADDR HAS WRAPPED AROUND
  465  723F  D0 F7                  BNE   BCCST         ; THEN TERMINATE WRITE OPERATION
  466                     ;
  467  7241  A9 04                  LDA   #4
  468  7243  85 EC                  STA   ACMD
  469  7245  20 8A 72               JSR   CRLF
  470  7248  20 9A 72               JSR   WR0A          ; OUTPUT HEX ADR
  471                     ;
  472  724B  20 77 73     WBNPF     JSR   SPACE
  473  724E  A2 09                  LDX   #9
  474  7250  86 FE                  STX   TMPC          ; LOOP CNT = 9
  475  7252  A1 E5                  LDA   (TMPC-9,X)
  476  7254  85 FF                  STA   TMPC2         ; BYTE TO TMPC2
  477  7256  A9 42                  LDA   #'B
  478  7258  D0 08                  BNE   WBF2          ; WRITE B
  479                     ;
  480  725A  A9 50        WBF1      LDA   #'P
  481  725C  C5 FF                  ASL   TMPC2
  482  725E  B0 02                  BCS   WBF2
  483  7260  A9 4E                  LDA   #'N
  484                     ;
  485  7262  20 C6 72     WBF2      JSR   WR0C          ; WRITE N OR P
  486  7265  C6 FE                  DEC   TMPC
  487  7267  D0 F1                  BNE   WBF1          ; LOOP
  488  7269  A9 46                  LDA   #'F
  489  726B  20 C6 72               JSR   WR0C          ; WRITE F
  490                     ;
  491  726E  20 97 73               JSR   INCTMP
  492                     ;
  493  7271  C6 EC                  DEC   ACMD          ; TEST FOR MULTIPLE OF FOUR
  494  7273  D0 D6                  BNE   WBNPF
  495                     ;
  496  7275  20 C1 70               JSR   DCMP
  497  7278  B0 C3                  BCS   WB1           ; LOOP WHILE EA GT OR = SA
  498  727A  90 BC                  BCC   BCCST
  499                     ;
  500  727C  48           DADD   PHA                    ; SAVE A
  501  727D  18                     CLC
  502  727E  65 F2                  ADC   TMP4
  503  7280  85 F2                  STA   TMP4
  504  7282  A5 F3                  LDA   TMP4+1
  505  7284  69 00                  ADC   #0
  506  7286  85 F3                  STA   TMP4+1
  507  7288  68                     PLA                 ; RESTORE A
  508  7289  60                     RTS
  509                     ;
  510  728A  A2 0D        CRLF      LDX   #$0D
  511  728C  A9 0A                  LDA   #$0A
  512  728E  20 C0 72               JSR   WRTWO
  513  7291  A6 E3                  LDX   CRDLY         ; BIT-TIME COUNT FOR DELAY
```

```
CARD # LOC    CODE       CARD
  515  7293  20 1D 73  CR1       JSR   DLY2          ; DELAY OF ONE BIT-TIME
  516  7296  CA                  DEX
  517  7297  D0 FA               BNE   CR1
  518  7299  60                  RTS
  519                    ;
  520                    ;  WRITE ADR FROM TMP0 STORES
  521                    ;
  522  729A  A2 01       WRCA      LDX   #1
  523  729C  D0 0A               BNE   WRDA1
  524  729E  A2 05       WRDA4     LDX   #5
  525  72A0  D0 06               BNE   WRDA1
  526  72A2  A2 07       WRDA6     LDX   #7
  527  72A4  D0 02               BNE   WRDA1
  528  72A6  A2 09       WRPC      LDX   #9
  529  72A8  B5 ED       WRDA1     LDA   TMP0-1,X
  530  72AA  48                  PHA
  531  72AB  B5 EE               LDA   TMP0,X
  532  72AD  20 B1 72            JSR   WRDB
  533  72B0  68                  PLA
  534                    ;
  535                    ;  WRITE BYTE - A = BYTE
  536                    ;  UNPACK BYTE INTO TWO ASCII CHARS, A=BYTE;  X,A=CHARS
  537                    ;
  538  72B1  48          WRDB      PHA
  539  72B2  4A                  LSR   A
  540  72B3  4A                  LSR   A
  541  72B4  4A                  LSR   A
  542  72B5  4A                  LSR   A
  543  72B6  20 58 73            JSR   ASCII         ; CONVERT TO ASCII
  544  72B9  AA                  TAX
  545  72BA  68                  PLA
  546  72BB  29 0F               AND   #$0F
  547  72BD  20 58 73            JSR   ASCII
  548                    ;
  549                    ;  WRITE 2 CHARS - X,A = CHARS
  550                    ;
  551  72C0  48          WRTWO     PHA
  552  72C1  8A                  TXA
  553  72C2  20 C6 72            JSR   WRT
  554  72C5  68                  PLA
  555                    ;
  556                    ;  WRITE SERIAL OUTPUT
  557                    ;  A=CHAR TO BE OUTPUT
  558                    ;
  559  72C6  20 1D 73    WRT       JSR   DLY2
  560  72C9  A2 09               LDX   #9
  561                    WRDC      =WRT
  562  72CB  49 FF               EOR   #$FF          ; COMPLEMENT A
  563  72CD  38                  SEC
  564                    ;
  565  72CE  20 DA 72    WRT1      JSR   OUT
  566  72D1  20 1D 73            JSR   DLY2
```

```
CARD #  LOC   CODE        CARD
   568  72D4  4A                  LSR   A
   569  72D5  CA                  DEX
   570  72D6  D0 F6               BNE   WRT1
   571  72D8  F0 3F               BEQ   RDT5
   572                    ;                                ; USE   BNE?
   573  72DA  48          OUT     PHA                      ; SAVE A
   574  72DB  AD 02 6E            LDA   MPB                ; OUTPUT BIT FROM Y
   575  72DE  29 FD               AND   #%11111101
   576  72E0  90 02               BCC   OUT1
   577  72E2  09 02               ORA   #%00000010
   578  72E4  8D 02 6E  OUT1      STA   MPB
   579  72E7  68                  PLA                      ; RESTORE A
   580  72E8  60                  RTS
   581                    ;
   582                    ;  OUTPUT RETURNS CHAR IN A
   583                    ;
   584  72E9  A5 E7       RDT     LDA   HSPTR              ; TEST FRO PTR OPTION
   585  72EB  4A                  LSR   A
   586  72EC  B0 4F               BCS   RDHSR
   587                    RDOC    =RDT
   588  72EE  A2 08               LDX   #8
   589                    ;
   590  72F0  AD 02 6E  RDT1      LDA   MPB
   591  72F3  4A                  LSR   A                  ; WAIT FOR START BIT
   592  72F4  90 FA               BCC   RDT1
   593                    ;
   594  72F6  20 20 73            JSR   DLY1
   595  72F9  20 DA 72            JSR   OUT                ; ECHO START BIT
   596                    ;
   597  72FC  20 1D 73  RDT2      JSR   DLY2
   598  72FF  AD 02 6E            LDA   MPB                ; CY = NEXT BIT
   599  7302  4A                  LSR   A
   600  7303  20 DA 72            JSR   OUT                ; ECHO
   601                    ;
   602  7306  08                  PHP                      ; SAVE BIT
   603  7307  98                  TYA                      ; Y CONTAINS CHAR BEING FORMED
   604  7308  4A                  LSR   A
   605  7309  28                  PLP                      ; RECALL BIT
   606  730A  90 02               BCC   RDT4
   607  730C  09 80               ORA   #$80               ; ADD IN NEXT BIT
   608  730E  A8          RDT4    TAY
   609  730F  CA                  DEX
   610  7310  D0 EA               BNE   RDT2               ; LOOP FOR 8 BITS
   611  7312  49 FF               EOR   #$FF               ; COMPLEMENT DATA
   612  7314  29 7F               AND   #$7F               ; CLEAR PARITY
   613                    ;
   614  7316  20 1D 73            JSR   DLY2
   615  7319  18          RDT5    CLC
   616  731A  20 DA 72            JSR   OUT                ; AND DELAY 2 HALF-BIT-TIMES
   617                    ;
   618  731D  20 20 73  DLY2      JSR   DLY1
```

```
CARD #  LOC   CODE       CARD
  620   7320  48         DLY1    PHA                     ; SAVE FLAGS AND A
  621   7321  08                 PHP
  622   7322  8A                 TXA                     ; SAVE X
  623   7323  48                 PHA
  624   7324  A6 EA              LDX   MAJORT
  625   7326  A5 EB              LDA   MINORT
  626                    ;
  627   7328  8D 04 6E   DL2      STA   MCLKIT
  628                    ;
  629   732B  AD 05 6E   DL3      LDA   MCLKIP
  630   732E  10 FB              BPL   DL3
  631   7330  CA                 DEX
  632   7331  08                 PHP
  633   7332  AD 04 6E           LDA   MCLKRD            ; RESET TIMER INT FLAG
  634   7335  28                 PLP
  635   7336  10 F3              BPL   DL3
  636                    ;                               ; RESTORE REGS
  637   7338  68                 PLA
  638   7339  AA                 TAX
  639   733A  28                 PLP
  640   733B  68                 PLA
  641   733C  60         DLX     RTS
  642                    ;
  643   733D  AD 02 6E   RDHSR    LDA   MPB               ; LOOP ON DATA AVAIL
  644   7340  29 08              AND   #DAVAIL
  645   7342  F0 F9              BEQ   RDHSR
  646                    ;
  647   7344  AE 00 6E           LDX   MPA               ; READ DATA
  648   7347  AD 02 6E           LDA   MPB               ; SEND GOT-DATA PULSE
  649   734A  09 04              ORA   #GOTDAT
  650   734C  8D 02 6E           STA   MPB
  651   734F  29 FB              AND   #%11111011
  652   7351  8D 02 6E           STA   MPB
  653   7354  8A                 TXA
  654   7355  29 7F              AND   #$7F
  655   7357  60                 RTS
  656                    ;
  657   7358  18         ASCII    CLC
  658   7359  69 06              ADC   #6
  659   735B  69 F0              ADC   #$F0
  660   735D  90 02              BCC   ASC1
  661   735F  69 06              ADC   #$06
  662                    ;
  663   7361  69 3A      ASC1    ADC   #$3A
  664   7363  48                 PHA                     ; TEST FOR LETTER B IN ADR DURING WBNPF
  665   7364  C9 42              CMP   #'B
  666   7366  D0 0A              BNE   ASCX
  667   7368  A5 FD              LDA   SAVX
  668   736A  C9 07              CMP   #NCMDS
  669   736C  D0 04              BNE   ASCX              ; NOT WB CMD
  670   736E  68                 PLA
  671   736F  A9 20              LDA   #'
```

```
CARD #  LOC   CODE        CARD
  673   7371  48                ASCX    PHA
  674   7372  68                        PLA
  675   7373  60                        RTS
  676                           ;
  677   7374  20 77 73          SPAC2   JSR   SPACE
  678   7377  48                SPACE   PHA                   ; SAVE A,X,Y
  679   7378  8A                        TXA
  680   7379  48                        PHA
  681   737A  98                        TYA
  682   737B  48                        PHA
  683   737C  A9 20                     LDA   #' '
  684   737E  20 C6 72                  JSR   WRT             ; TYPE SP
  685   7381  68                RESTORE PLA                   ; RESTORE A,X,Y
  686   7382  A8                        TAY
  687   7383  68                        PLA
  688   7384  AA                        TAX
  689   7385  68                        PLA
  690   7386  60                        RTS
  691                           ;
  692   7387  A2 02              T2T2    LDX   #2
  693   7389  B5 ED              T2T21   LDA   TMP0-1,X
  694   738B  48                        PHA
  695   738C  B5 EF                     LDA   TMP2-1,X
  696   738E  95 ED                     STA   TMP0-1,X
  697   7390  68                        PLA
  698   7391  95 EF                     STA   TMP2-1,X
  699   7393  CA                        DEX
  700   7394  D0 F3                     BNE   T2T21
  701   7396  60                        RTS
  702                           ;
  703                           ;  INCREMENT (TMP0,TMP0+1) BY 1
  704   7397  E6 EE              INCTMP  INC   TMP0            ; LOW BYTE
  705   7399  F0 01                     BEQ   INCT1
  706   739B  60                        RTS
  707                           ;
  708   739C  E6 EF              INCT1   INC   TMP0+1          ; HIGH BYTE
  709   739E  F0 01                     BEQ   SETWRP
  710   73A0  60                        RTS
  711                           ;
  712   73A1  E5 E4              SETWRP  INC WRAP              ; POINTER HAS WRAPPED AROUND SET FLAG
  713   73A3  60                        RTS
  714                           ;
  715                           ;  READ HEX ADR, RETURN HO IN TMP0 , LO IN TMP0+1 AND CY=1
  716                           ;    IF SP CY=0
  717                           ;
  718   73A4  20 B3 73          RDOA    JSR   RDOB            ; READ TWO CHAR BYTE
  719   73A7  90 02                     BCC   RDOA2           ; SPACE
  720                           ;
  721   73A9  85 EF                     STA   TMP0+1
  722   73AB  20 B3 73          RDOA2   JSR   RDOB
  723   73AE  90 02                     BCC   RDEXIT          ; SP
  724   73B0  85 EE                     STA   TMP0
```

```
CARD # LOC     CODE        CARD
  726  73B2  60          RDEXIT   RTS
  727                    ;
  728                    ;  READ HEX BYTE AND RETURN IN A, AND CY=1
  729                    ;   IF SP CY =0
  730                    ;   Y REG IS PRESERVED
  731                    ;
  732  73B3  98          RDOB     TYA                 ; SAVE Y
  733  73B4  48                   PHA
  734  73B5  A9 00                LDA   #0            ; SET DATA = 0
  735  73B7  85 EC                STA   ACMD
  736  73B9  20 E9 72             JSR   RDOC
  737  73BC  C9 0D                CMP   #$0D          ; CR?
  738  73BE  D0 06                BNE   RDOB1
  739  73C0  68                   PLA                 ; YES - GO TO START
  740  73C1  68                   PLA                 ; CLEANING STACK UP FIRST
  741  73C2  68                   PLA
  742  73C3  4C E6 70             JMP   START
  743                    ;
  744  73C6  C9 20       RDOB1    CMP   #' '          ; SPACE
  745  73C8  D0 0A                BNE   RDOB2
  746  73CA  20 E9 72             JSR   RDOC          ; READ NEXT CHAR
  747  73CD  C9 20                CMP   #' '
  748  73CF  D0 0F                BNE   RDOB3
  749  73D1  18                   CLC                 ; CY=0
  750  73D2  90 12                BCC   RDOB4
  751                    ;
  752  73D4  20 EB 73    RDOB2    JSR   HEXIT         ; TO HEX
  753  73D7  0A                   ASL   A
  754  73D8  0A                   ASL   A
  755  73D9  0A                   ASL   A
  756  73DA  0A                   ASL   A
  757  73DB  85 EC                STA   ACMD
  758  73DD  20 E9 72             JSR   RDOC          ; 2ND CHAR ASSUMED HEX
  759  73E0  20 EB 73    RDOB3    JSR   HEXIT
  760  73E3  05 EC                ORA   ACMD
  761  73E5  38                   SEC                 ; CY=1
  762  73E6  AA          RDOB4    TAX
  763  73E7  68                   PLA                 ; RESTORE Y
  764  73E8  A8                   TAY
  765  73E9  8A                   TXA                 ; SET Z & N FLAGS FOR RETURN
  766  73EA  60                   RTS
  767                    ;
  768  73EB  C9 3A       HEXIT    CMP   #$3A
  769  73ED  08                   PHP                 ; SAVE FLAGS
  770  73EE  29 0F                AND   #$0F
  771  73F0  28                   PLP
  772  73F1  90 02                BCC   HEX09         ; 0-9
  773  73F3  69 08                ADC   #8            ; ALPHA ADD 8+CY=9
  774  73F5  60          HEX09    RTS
  775                    ;
  776  73F6                       *=MP3+$F8
  777                    ;
```

```
CARD #  LOC     CODE        CARD
  779   73F8  00 70      INTVEC   .WORD NMINT          ; DEFAULT USER IRQ TO NMINT
  780   73FA  00 70               .WORD NMINT
  781   73FC  06 70               .WORD RESET
  782   73FE  52 70               .WORD INTRQ
  783                     ;
  784                               .END


NUMBER OF ERRORS =    0,  NUMBER OF WARNINGS =    0
```

SYMBOL TABLE

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
ACC	00F9	98	118 173 205 357
ACMD	00EC	90	225 227 232 468 493 735 757 760
ADRS	710D	296	226
ALTER	713A	331	298
ASCII	7358	657	643 647
ASCX	7372	674	666 669
ASC1	7361	663	660
A2	7146	337	335
A3	714B	339	332
A4	7150	342	338
A5	7152	343	345
A9	715A	346	****
BCCST	7238	460	418 465 498
BEQS1	7134	325	346 380
BX	7081	205	177
BYTE	70E0	264	344 391
BY2	70F2	277	272
BY3	70F5	278	265
B3	7058	180	120
B5	7073	197	****
CMDS	7106	289	218
CRDLY	00E3	81	141 513
CRLF	728A	510	197 211 366 420 469
CR1	7293	515	517
DADD	727C	500	277 383 386 389 438 441 444 450
DAVAIL	0008	62	644
DCMP	70C1	241	398 429 458 496
DIFF	00E5	84	244 248 432
DLX	733C	641	****
DLY1	7320	620	594 618
DLY2	731D	618	515 559 566 597 614
DL2	7328	627	****
DL3	7328	629	630 635
DSPLYM	711C	314	298
DSPLYR	7114	310	297
ERROPR	70BA	237	275 327 400
ERRP1	71BF	400	****
ERRS1	7137	327	315
FLGS	00F8	97	187 282 354
GO	715C	348	299
GOTDAT	0004	63	649
HEXIT	73EB	768	752 759
HEX09	73F5	774	772
HSP	716F	362	300
HSPTR	00E7	85	133 209 368 379 584
HSROP	00E8	86	134 362 367
IJMP	70B4	232	229
INCTMP	7397	704	278 452 491
INCT1	739C	708	705
INTRQ	7052	173	782
INTVEC	73F8	779	127

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES								
IOBASE	6E00	64	65	66	67	68	69	70	71		
LCNT	00FF	106	****								
LH	7174	365	301								
LH1	717E	369	371	399							
LH2	7195	382	376								
LH3	71AA	391	392								
MAJORT	00EA	88	132	149	162	624					
MCLKIP	6EC5	71	147	629							
MCLKIT	6E04	69	146	627							
MCLKRD	6E04	70	159	633							
MDA	6E01	66	****								
MDB	6E03	68	124								
MDBK	0015	61	122								
MINORT	00EB	89	168	625							
MPA	6E00	65	647								
MPB	6E02	67	142	153	574	578	590	598	643	648	650 652
MP0	7000	74	116								
MP1	7100	75	224	296	297	298	299	300	301	302	
MP2	7200	76	****								
MP3	7300	77	776								
M0	7123	317	312								
M1	7127	319	324								
NCMDS	00C7	73	216	668							
NMINT	7000	118	779	780							
OUT	72DA	573	565	595	600	616					
OUT1	72E4	578	576								
PCH	00F7	96	193	253	350						
PCL	00F6	95	190	251	352						
PREVC	00E9	87	222	331							
PUTP	70D0	250	336								
RAM64	FF00	110	111								
RCNT	00FE	105	279	342	382	422	435	436	437	453	
RDEXIT	73B2	726	723								
RDHSR	733D	643	586	645							
RDOA	73A4	718	314	334	393	405	408				
RDOA2	73A8	722	719								
RDOB	73B3	732	264	375	384	387	718	722			
RDOB1	73C5	744	738								
RDOB2	73D4	752	745								
RDOB3	73E0	759	748								
RDOB4	73E6	762	750								
RDOC	72E9	587	214	365	369	402	411	736	746	758	
RDT	72E9	584	587								
RDT1	72F0	590	592								
RDT2	72FC	597	610								
RDT4	730E	608	606								
RDT5	7319	615	571								
RESET	7006	122	781								
R0	7022	142	144								
R1	700D	127	130								
R2	7023	146	150								
R3	7023	147	155	157							
R4	7034	152	148								
R5	7044	161	167								

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES									
R6	704B	166	163									
SAVX	00FD	102	221	223	463	667						
SETR	70FB	282	311	337								
SETWRP	73A1	712	709									
SP	00FC	101	195	348								
SPACE	7377	678	319	343	404	407	472	677				
SPAC2	7374	677	230									
START	7086	208	239	325	363	460	742					
S0	7097	216	203									
S1	7099	218	235									
S2	7087	234	219									
TMPC	00FE	103	105	180	198	317	323	403	412	474	486	
TMPC2	00FF	104	106	476	481							
TMP0	00EE	91	243	246	250	252	258	259	268	271	283	285
			320	385	388	440	443	448	475	529	531	693
			696	704	708	721	724					
TMP2	00F0	92	242	245	395	397	695	698				
TMP4	00F2	93	394	396	502	503	504	506				
TMP6	00F4	94	****									
T2T2	7387	692	406	410								
T2T21	7389	693	700									
UNIT	FFF8	72	128	206								
WB	723B	463	415									
WBF1	725A	480	487									
WBF2	7262	485	478	482								
WBNPF	7248	472	494									
WB1	723D	464	497									
WH0	71E2	417	459									
WH1	72C7	437	431	434								
WH2	721F	447	454									
WO	71C2	402	302									
WRAP	00E4	83	210	417	464	712						
WRCA	729A	522	339	470								
WROA1	72A8	529	523	525	527							
WROA4	729E	524	456									
WRCA6	72A2	526	****									
WROB	7281	538	321	439	442	445	451	532				
WRCC	72C6	561	213	238	427	485	489					
WRPC	72A6	528	310									
WRT	72C6	559	553	561	684							
WRTWO	72C0	551	201	512								
WRT1	72CE	565	570									
XR	00FA	99	184	358								
YR	00FB	100	185	359								
ZTMP	7009	257	374	424								

```
CARD #  LOC   CODE        CARD
     1                   ;MEMORY ADDRESS TEST
     2                   ;FOR EACH LOC IN TEST RANGE
     3                           ;CLEAR WHOLE RANGE
     4                   ;       SET LOC TO $FF
     5                   ;       VERIFY WHOLE RAGE $00 EXCEPT (LCC)
     6                   ;       VERIFY (LCC) TO BE $FF
     7                   ;BREAK TO MONITOR ON ERROR WITH LOC IN (C,1)
     8                   ;PRINT "*" ON COMPLETION OF PASS & REPEAT
     9                   ;
    10  0CCC                    *=$00C0             ;PAGE C
    11                  ;
    12                  WRT     =$72D2
    13  0000            LOC     *=*+2               ;TEST CELL ADDR
    14  0002            LOW     *=*+2                ;LOWER LIMIT OF TEST
    15  0004            HIGH    *=*+2               ;UPPER LIMIT OF TEST+1
    16  0006            PTR     *=*+2               ;POINTER TO CELL UNDER TEST
    17                  ;
    18  0C08                    *=$0C10             ;START ADDR
    19                  ;
    20  0010  A9 0D     MAC     LDA #$0D            ;TYPE CR
    21  0C12  2C C2 72          JSR WRT
    22  0C15  A9 CA             LDA #$0A            ;& LF
    23  0017  2C C2 72          JSR WRT
    24                  ;
    25  0C1A  2C 68 CC          JSR RSTLOC          ;LOC=LOW

    26  0C1D  2C 71 CC          JSR RSTPTR          ;PTR=LOW

    27  002C  A2 00             LDX #0
    28                  ;
    29                  ;CLEAR MEMORY AREA UNDER TEST
    30  0022  A9 00     ML1     LDA #0
    31  0024  81 C6             STA (PTR,X)         ;STORE ZERO
    32  0026  2C 7A 00          JSR INCPTR          ;INCREMENT & TEST

    33  C029  DC F7             BNE ML1             ;NEXT LOC
    34                  ;
    35                  ;PUT $FF IN SELEXTED CELL
    36  C02B  A9 FF     TEST    LDA #$FF
    37  002D  81 00             STA (LOC,X)
    38                  ;VERIFY ALL CELLS ZERO EXCEPT (LCC)
    39  002F  20 71 0C          JSR RSTPTR          ;PTR=LOW

    40                  ;
    41  0C32  A1 C6     VLOOP   LDA (PTR,X)         ;GET CELL
    42  C034  F0 17             BEQ NEXTC           ;OK IF ZERO
    43  0036  A4 C6             LDY PTR             ;NOT ZERO--IS THIS (LOC)?
    44  0038  C4 0C             CPY LOC
    45  CC3A  FC 01             BEC CK1
    46  003C  CC                BRK                 ;NOT (LCC)
    47                  ;
    48  CC3D  A4 C7     OK1     LDY PTR+1
```

```
CARD #  LOC   CODE        CARD
   49   003F  C4 01             CPY LOC+1
   50   0041  F0 01             BEQ CK2
   51   0043  00                BRK                 ;NOT (LOC)
   52                    ;
   53   0044  C9 FF      CK2     CMP #$FF            ;IS (LOC)--IS DATA OK?
   54   0046  F0 01             BEQ OK3
   55   0048  CC                BRK                 ;WRONG DATA
   56                    ;
   57   0049  A9 00      OK3     LDA #0              ;RESET (LOC)
   58   004B  81 CC             STA (LOC,X)
   59                    ;
   60   004D  20 7A 00   NEXTC   JSR INCPTR          ;NEXT CELL

   61   005C  D0 E0             BNE VLOOP           ,IF NOT AT LIMIT
   62                    ;
   63   0052  A5 00             LDA LOC             ;PRINT STAR EVERY PAGE BOUNDARY
   64   0054  D0 07             BNE NOSTAR
   65   0056  A9 2A             LDA #'*
   66   0058  20 C2 72          JSR WRT
   67   005B  A2 00             LDX #0              ;FIX X AFTER MON CALL
   68                    ;
   69   005D  20 8B 00   NOSTAR  JSR INCLOC          ;NEXT LOC

   70   006C  D0 C9             BNE TEST
   71                    ;
   72   0062  20 68 00          JSR RSTLOC          ;PASS COMPLETE

   73   0065  40 10 C0          JMP MAD             ;NEXT PASS
   74                    ;
   75                    ;RESET LOC TO LOW
   76   C068  A5 02      RSTLOC LDA LOW
   77   C06A  85 CC             STA LOC
   78   006C  A5 C3             LDA LOW+1
   79   006E  85 C1             STA LOC+1
   80   007C  6C                RTS
   81                    ;
   82                    ;RESET PTR TO LOW
   83   0071  A5 C2      RSTPTR LDA LOW
   84   0073  85 06             STA PTR
   85   0075  A5 C3             LDA LOW+1
   86   0077  85 07             STA PTR+1
   87   0079  6C                RTS
   88                    ;
   89                    ;INCREMENT PTR & CHECK FOR LIMIT
   90   007A  E6 06      INCPTR INC PTR             ;INCREMENT
   91   007C  DC C2             BNE INC1
   92                    ;
   93   007E  E6 07             INC PTR+1
   94                    ;
   95   0080  A5 04      INC1   LDA HIGH            ;CHECK
   96   0082  C5 C6             CMP PTR
   97   0084  D0 C4             BNE IPRET          ;NOT AT LIMIT
```

```
CARD #  LOC   CODE        CARD
   98                    ;
   99   0086  A5 C5             LDA HIGH+1
  100   0088  C5 07             CMP PTR+1          ;Z=1 IF AT LIMIT
  101                    ;
  102   008A  60         IPRET  RTS
  103                    ;
  104                    ;INCREMENT LOC & CHECK FOR LIMIT
  105   008B  E6 CC      INCLOC INC LOC            ;INCR
  106   008D  D0 02             BNE INC2
  107                    ;
  108   008F  E6 01             INC LOC+1
  109                    ;
  110   0091  A5 C4      INC2   LDA HIGH           ;CHECK
  111   0093  C5 00             CMP LOC
  112   0095  D0 04             BNE ILRET
  113   0097  A5 05             LDA HIGH+1
  114   0099  C5 01             CMP LOC+1         ;Z=1 IF AT LIMIT
  115                    ;
  116   009B  6C         ILRET  RTS


NUMBER OF ERRORS =   0,  NUMBER OF WARNINGS =   0


        SYMBOL TABLE

  SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

  HIGH     00C4          15     95   99  110  113
  ILRET    009B         116    112
  INCLOC   008B         105     69
  INCPTR   007A          90     32   60
  INC1     008C          95     91
  INC2     0091         110    106
  IPRET    008A         102     97
  LOC      00CC          13     37   44   49   58   63   77   79  105  108  111
                                114
  LOW      00C2          14     76   78   83   85
  MAD      C01C          20     73
  ML1      0022          30     33
  NEXTC    004D          60     42
  NOSTAR   005D          69     64
  CK1      003D          48     45
  CK2      0044          53     50
  CK3      0049          57     54
  PTR      0006          16     31   41   43   48   84   88   90   93   96  100
  RSTLOC   00E8          76     25   72
  RSTPTR   0071          83     26   39
  TEST     002B          36     70
  VLOOP    0032          41     61
  WRT      72C2          12     21   23   66
```

SECTION 5

ROM RESIDENT SOFTWARE (OPTIONAL)

Resident software is available for your SUPER JOLT card in the form of two mask-programmed ready-only memories (ROMs).

ROM OPERATION INSTRUCTIONS

These two proprietary 2Kx8 mask ROMs (AM9216's) contain the 6502 Resident Assembler Program (RAP) and JOLT TINY BASIC. These ROMs are designed for immediate use on the CP110 SUPER JOLT Card.

Program Location:

TINY BASIC	C000 - C8FF
RAP	C900 - CFFF

Ram Requirements:

Both TINY BASIC and RAP assume RAM at page 0 and 1 (0000 - 01FF). Both programs will, upon execution, determine the extent of RAM memory, beginning at 0200 and continuing to write/read memory until the read fails, indicating the end of RAM memory. This RAM address space information is used by the programs as the total extent of RAM available.

Page 0 Vector Initialization:

Prior to running either TINY BASIC or RAP, two branch vectors must be entered at location 0000. Use DEMON™ to first desplay, then alter these locations as follows:

```
.M 0000 XX XX XX XX XX XX XX XX
.: 0000 4C E9 72 4C C6 72
```

The branch at 0000 (4CE972) is a JMP to the DEMON™ read character routine (RDT). The branch at 0003 (4CC672) is a JMP to the DEMON™ write character routine (WRT).

Start Addresses:

TINY BASIC	C000
RAP	C900

High Speed Paper Tape Input Option (TINY BASIC Only):

TINY BASIC may be made to run using the high speed paper tape input option. First use DEMON™ to initialize page 0 as follows:

```
.: 0000 4C 06 00 4C C6 72 A6 E8
.: 0008 86 E7 20 1D 73 4C E9 72
```

Use DEMON's™ "H" command, to set the high-speed reader mode. Then transfer control to TINY BASIC at C000. Input will be read in from the high speed reader

source, but will not be echoed on the printer device. Following completion of the read-in, if control is not returned to the terminal, restart BASIC at C003, (warm start) to preserve the data in memory.

Resident Assembler Test Program:

Users should note that the .ORG $1000 on the RAP test program should be changed to .ORG $200 for SUPER JOLT usage. This can be done dynamically by stopping the test run just prior to the read-in of the .ORG $1000, typing in the .ORG $200, skipping past the .ORG $1000 on the paper tape, and continuing.

JOLT TINY BASIC

The JOLT TINY BASIC interpretive program is a subset of Dartmouth BASIC that resides in 2,304 bytes of program memory. The language consists of 12 statements, four expressions and two machine language subroutine calls.

PRINT: print-list

This statement prints values of the expressions and/or the contents of the strings in the print-list. The items may be expressions or alphanumeric strings enclosed in quotation marks.

INPUT: input-list

This statement checks to see if the current line is exhausted. If it is, a question mark is prompted with an X-ON control character, and a new line is read in. Then or otherwise, the input list is scanned for an expression which is evaluated. The value thus derived is stored in the first variable in the input-list.

LET variable = expression:

This statement assigns the value of the expression to the variable. The long form of this statement executes slightly faster than the short form (variable = expression).

GOTO expression:

The GOTO statement premits changes in the sequence of program execution to the line number derived by the evaluation of the expression in the GOTO statement. This permits one to compute the line number of the next statement on the basis of program parameters during program execution.

GOSUB expression:

The GOSUB statement is like the GOTO statement, except that TINY BASIC remembers the line number of the GOSUB statement, so that the next occurrence of a RETURN statement will result in execution proceeding from the starement following the GOSUB. Subroutines called by GOSUB statements may be nested to any depth.

RETURN:

The RETURN statement transfers execution control to the line following the most recent unRETURNed GOSUB.

IF expression rel expression THEN statement:

The IF statement compares the expressions according to one of six relational operators. If the relationship is TRUE, the statement is executed; if FALSE, the associated statement is skipped. The six relational operators are:

"=","<",">","<=",">=","> <".

END:

The END statement must be the last executable statement to terminate a problem at any time or to clear out any saved GOSUB line numbers.

REM comments:

The REM statement permits comments to be interspersed in the program.

CLEAR:

The CLEAR statement formats the user program space, deleting any previous programs.

RUN:

The RUN statement is used to begin program execution at the first (lowest) line number.

LIST:

The LIST statement causes part or all of the user program to be listed. If no parameters are given, the whole program is listed.

EXPRESSIONS:

An expression is the combination of one or more numbers or variables joined by operators, and possibly grouped by parentheses. There are four operators:

+(add), - (sub), *(multiply), and /(divide).

RESIDENT ASSEMBLER PROGRAM

The JOLT Resident Assembler Program (RAP) is a 1.75K byte program designed for use on CP110 SUPER JOLT systems equipped with at least 4K bytes of RAM memory. RAP processes source statements producing an output listing on teletype-like like devices. The assembly process is performed in one pass, reading source input, printing the listing and generation object code continuously until all processing is complete. Source input is accepted by the assembler either by directly typing input at the keyboard or by reading a previously prepared punched paper tape.

The assembler stores the generated object code directly into JOLT memory. There it can be executed immediately after assembly or punched out in hex format using

the DEMON™ monitor.

Rap is compatible with the Synertek Cross Assembler with the following exceptions:

- Expressions and *(used for current program counter) are not allowed.
- The OPT and PAGE pseudo operations are not implemented.
- Octal and binary numbers are not implemented.
- ORG is used instead of *= to origin program.
- RES is used for reserving storage.

Input Line Format

Source input is free format where each statement can be composed of the following optional fields:

- Label - If present, must begin in column one and be terminated by a space.
- Operation - If present, must be preceded by a space and must be one of the SY 65XX mnemonics defined in the RAP Manual.
- Operands - If present, must be preceded by a space and follow one of the forms found in the RAP manual.
- Comments - if present, must have as its first character a semicolon (;) and if not in column one, must be preceded by a space.
- Carriage return - All lines are terminated by a carriage return.
- Expressions and * (used for current program counter) are not allowed.

A name is any alphanumeric symbol. Names are used as labels or in operand fields.

The first character of a name can be any assembler recognized name character or a letter. The second and following characters of a name can include numbers. A name is terminated by a blank or a carriage return. Names have no restrictions on length other than they must fit on one line.

Numbers are an unsigned string of hexaadecimal or decimal characters. Hexidecimal numbers are preceded by a $ and can contain the numbers 0 through 9 and the hex characters A through F. Decimal numbers contain only the decimal characters 0 through 9.

ASCII strings have two forms. The short form is used in conjunction with immediate operands and consists of a single quote followed by a single ASCII character. The long form is used to define large strings of ASCII data. This form is only valid in the BYTE Pseudo operation. Carriage returns are not allowed within the string.

All symbols that are intended to be used as an 8-bit immediate operand must be

defined (appear in a label field) before they are used. Sixteen bit operands and forward relative branches are inserted at the time of their definition by the assembler. This means that the value appearing in the assembly listing of previously undefined symbols is not the ultimate value used at the time the assembly is complete.

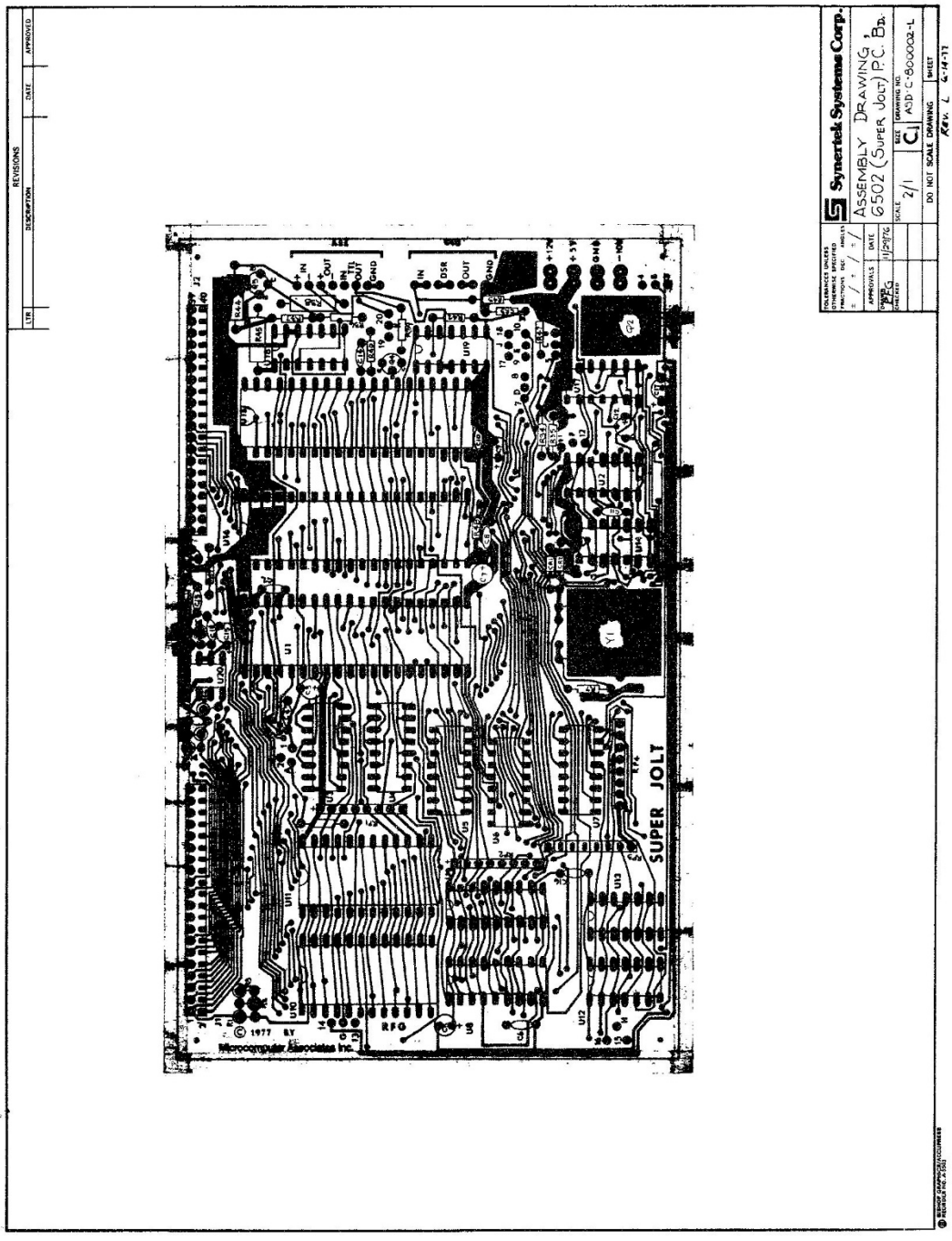
The assembler detects errors during the assembly process and outputs an appropriate error code after the printed hex output on the listing. Recognized errors include: Branch address out of range, undefined operation code, size of operand value exceed 8-bits, multiple appearances of the symbol in the label field, and improper format in the operand field. Additionally, the location field is offset in the table dump for undefined symbols.

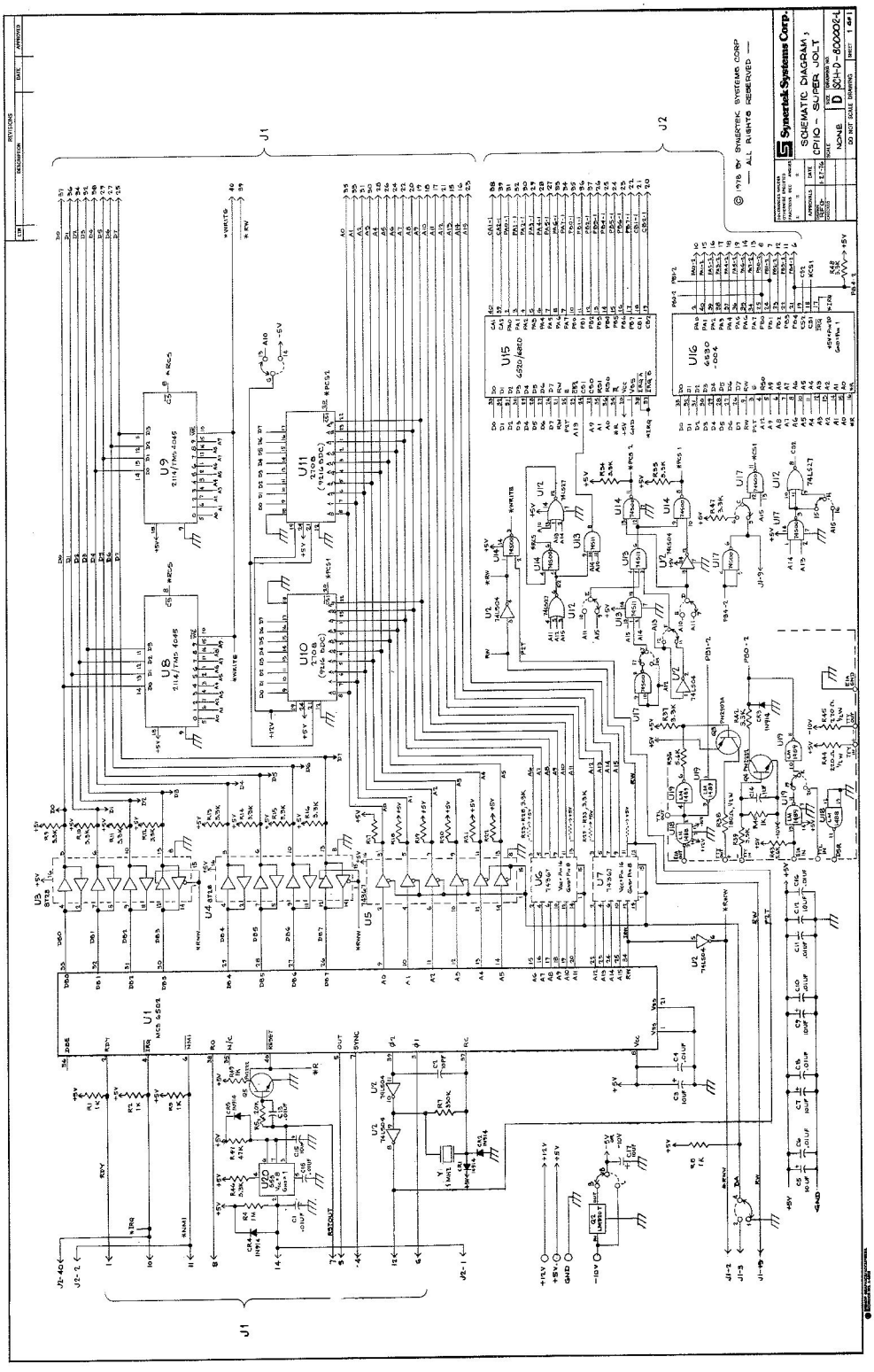
The reverse-slash character may be used to delete the current source line during an assembly run. Reverse-slash is obtained by the Shift and L key combination. When used, the delete line causes the current line to be ignored and terminated, a carriage return will be effected and the assembler will position the carriage to accept the next line of source data.

The user may at any time, reorigin the assembler to correct areas of code already assembled. For example, if half way through an assembly, the user realized that an instruction was omitted, say five lines ahead of its current position, he may stop the assembler, reorigin back to the location at which the instruction should be inserted, insert the instruction and then repeat the assembly process from that point forward.

SECTION 6

SUPERJOLT SCHEMATIC & ASSEMBLY DRAWING




Synertek Systems Corp.
SCHEMATIC DIAGRAM; CPU10 - SUPER-KIM
SCM-D-8000024-4
© 1978 BY SYNERTEK SYSTEMS CORP — ALL RIGHTS RESERVED —

Synertek Systems Corporation

P.O. BOX 552 SANTA CLARA, CALIFORNIA 95052 TEL. (408) 988-5600 TWX: 910-338-0135