



Supplement to
8002A
μProcessor Lab System Users Manual
070-2701-02
6500/1
EMULATOR SPECIFICS
Option 24

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Serial Number _____

WARRANTY

Tektronix warrants that this product, excluding customer-supplied equipment, is free from defects in materials and workmanship. The warranty period is ninety (90) days from the date of shipment. Tektronix will, at its option, repair or replace the product if Tektronix determines it is defective within the warranty period. CRTs are warranted for one (1) year. During the nine (9) months following expiration of the product warranty, Tektronix will replace defective CRTs at no charge for the material.

In the forty-eight (48) contiguous United States, the District of Columbia, and in other areas where Tektronix normally offers on-site service for this product, Tektronix will provide this service at no charge during the product warranty period described above. In areas where Tektronix does not offer on-site service for this product, warranty service will be provided at no charge if the product is returned, freight prepaid, to a service center designated by Tektronix.

Tektronix may use the original vendor's service organization to service any product that is supplied but not manufactured by Tektronix.


Tektronix is not obligated to furnish service under this warranty:

- a. to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair, or service the product;
- b. to repair damage resulting from improper use or from connecting the product to incompatible equipment;
- c. if personnel other than Tektronix representatives modify the hardware or software.

There is no implied warranty of fitness for a particular purpose. Tektronix is not liable for consequential damages.

Copyright © 1980 Tektronix, Inc. All rights reserved.
Contents of this publication may not be reproduced in any form without the written permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered by U.S. and foreign patents and/or pending patents.

TEKTRONIX, TEK, SCOPE-MOBILE, and  are registered trademarks of Tektronix, Inc. TELEQUIPMENT is a registered trademark of Tektronix U.K. Limited.

Printed in U.S.A. Specification and price change privileges are reserved.

PREFACE

These 6500/1 Emulator Specifics are a supplement to the 8002A System User's Manual. Insert this supplement into your 8002A System User's Manual, at the end of Section 9, Emulator Specifics.

You may want to note in your table of contents for Section 9 that the 6500/1 Emulator Specifics begin on page 9-234.

CONTENTS

Section 9	EMULATOR SPECIFICS	
	6500/1 Emulator Specifics	9-234
	Demonstration Run	9-234
	Emulator-Specific Commands and Displays	9-248
	SET—Setting 6500/1 Commands and Displays	9-248
	RESET—Resetting 6500/1 Emulator Status	9-249
	DSTAT—Sample 6500/1 Emulator Status Display	9-249
	DISM—Sample 6500/1 Disassembled Code	9-251
	TRACE—Sample 6500/1 TRACE Display	9-251
	BKPT—Sample 6500/1 Breakpoint Display	9-253
	6500/1 Service Calls	9-254
	SVC Demonstration	9-255
	6500/1 Design Features	9-257

TABLES

Table No.		Page
9-23	6500/1 SET Command Parameters	9-248
9-24	6500/1 Service Calls	9-254

ILLUSTRATIONS

Fig. No.		Page
9-15	6500/1 Status Register	9-250
9-16	6500/1 SVC Demonstration Program (Sheet 1 of 2)	9-255
9-17	6500/1 SVC Demonstration Program (Sheet 2 of 2)	9-256
9-18	6500/1 Memory Allocation	9-258

6500/1 EMULATOR SPECIFICS

This subsection explains the features of the 8002A system that are unique to the 6500/1 emulator. The 6500/1 Demonstration Run is designed to be used with Section 1, the Learning Guide of this manual; the rest of this subsection contains reference material.

Demonstration Run

Introduction

This Demonstration Run shows you how to assemble, load, execute, and monitor the execution of a simple program. Designed to be used with the Learning Guide's Demonstration Run, this demonstration uses a 8002A system configured with a 6500/1 emulator. (The Learning Guide's Demonstration Run uses a 8002A system configured with an 8080A emulator.) This demonstration uses the following conventions:

1. Underlined—Any underlined characters in a command line must be entered from your system terminal. Those characters not underlined are system output.
2. All addresses are in hexadecimal notation unless otherwise specified.
3. <CR>—Command lines are terminated by a carriage return unless otherwise noted. When a carriage return is to be entered, the symbol <CR> is used.
4. <LF>—When a linefeed is to be entered, the symbol <LF> is used.

Please note: Your 6500/1 displays may differ from the ones shown in this demonstration. However, the differences, if any, will be minor. They should not affect your grasp of the basic concepts presented here.

Starting the 8002A System

Verify that your system is turned on and ready to use. Refer to the Learning Guide Demonstration Run for instructions.

Look at Your Master Disc; Make a Backup Disc

(Make sure your system disc is in drive 0.) List the contents of your master disc with the following LDIR command line:

```
> LDIR 0 <CR>
      |
      v
    disc drive
```

```
6500/1 TEKDOS V3.X
```

```
FILE NAME  BLKS
```

```
TEKDOS      16
.           173
COPYSYS     1
NEWDISC     1
DEMO%S     1
```

} file names and the number of
blocks each file occupies

```
TOTAL FILES      5
TOTAL BLOCKS USED 192
BLOCKS AVAILABLE 112
TOTAL BAD BLOCKS 0
```

← "BAD" means blocks not usable
by TEKDOS

You may want to make a copy of your master disc before you start using your 8002A, in case you accidentally erase something on your master disc. Insert a blank disc in drive 1, then enter the following command line to format and verify the new disc, and copy the files from your master disc to your backup disc:

```
> NEWDISC <CR>
```

```
* NEWDISC--FORMATS, VERIFIES, AND CREATES A NEW SYSTEM DISC
```

```
FORMAT 1
*FORMAT* EOJ } format blank disc
VERIFY 1
*VERIFY* EOJ } verify new disc
COPYSYS 0 1 ← execute COPYSYS command
               file: copy TEKDOS from
               disc in drive 0 to disc
               in drive 1
```

```
* COPYSYS FOR TEKDOS V3.X
COPY TEKDOS/0 TEKDOS/1 } copy TEKDOS
*COPY* EOJ
COPY /0 /1 } copy control files
*COPY* EOJ
COPY COPYSYS/0 COPYSYS/1 } copy command file "COPYSYS"
*COPY* EOJ
REN 1 6500 TEKDOS V3.X } name new disc
*RENAME* EOJ
*END OF COPYSYS*
COPYSYS COMPLETED
```

Enter the following LDIR command line to confirm that the contents of your system disc in drive 0 have been copied to the disc in drive 1:

```
> LDIR 1 <CR>

6500/1 TEKDOS V3.X

FILE NAME  BLKS
TEKDOS      16
.           173
COPYSYS     1
NEWDISC     1

TOTAL FILES      4
TOTAL BLOCKS USED 191
BLOCKS AVAILABLE 113
TOTAL BAD BLOCKS 0
```

Examine the Demonstration Run Program

The Demonstration Run Program. List the contents of the system disc in drive 0 again with the following command line:

```
> LDIR 0 <CR>

6500/1 TEKDOS V3.X

FILE NAME  BLKS
TEKDOS      16
.           173
COPYSYS     1
NEWDISC     1
DEMO%S      1 ← Demonstration Run program
                    source code file

TOTAL FILES      5
TOTAL BLOCKS USED 192
BLOCKS AVAILABLE 112
TOTAL BAD BLOCKS 0
```

Enter the following command line to copy the source code file from the disc in drive 0 to the disc in drive 1:

```
> COPY DEMO%S/0 DEMO%S/1 <CR>
```

Enter the following command line to copy the source code file DEMO%S onto the system terminal (TEKDOS assumes the file is to be copied to the system terminal when no device or other file is specified):

```
> COPY DEMO%S <CR>
      |
      |
      | file to be copied
```

```

TITLE    "6500/1 DEMONSTRATION RUN PROGRAM"
ORG      800H      ; START AT ADDRESS 800
DEMO    LDX      #0      ; INITIALIZE TABLE POINTER
        LDA      #0      ; CLEAR ACCUMULATOR
        LDY      #5      ; SET PASS COUNTER
LOOP    CLC      ; CLEAR CARRY BIT
        ADC      500H,X  ; ADD BYTE FROM TABLE
        INX      ; POINT TO NEXT BYTE
        DEY      ; DECREMENT PASS COUNTER
        BNE      LOOP   ; IF NOT 5 PASSES, LOOP
        STA      04F7H  ; IF 5 PASSES, EXIT
        NOP      ; FROM USER PROGRAM
        ORG      40H    ; SET UP SERVICE CALL
        WORD     420H  ; TELLING TEKDOS TO
        BYTE     1AH   ; EXIT FROM USER PROGRAM
        END      DEMO  ; END OF SOURCE FILE CODE

*COPY* EOJ
```

Explanation of Demonstration Run Program Source Code. The source code in the file DEMO%S contains two kinds of statements: TEKDOS Assembler directives and 6500/1 assembly language instructions. The assembler directives allow the TEKDOS Assembler to convert source code into executable machine language code (object code). The assembler directives are explained in the Learning Guide section of this manual.

The following paragraphs discuss the 6500/1 instructions that constitute the 6500/1 Demonstration Run program. For more information about the 6500/1 instruction set and addressing modes, refer to the 6500/1 Assembler Specifics section of the 8002A Assembler Manual.

What the Demonstration Run Program Does. The Demonstration Run program adds five numbers from a table in program memory and leaves the sum in the accumulator. The program's steps are illustrated in the flowchart in Fig. 1-5, in the Learning Guide section of this manual.

Prepare Registers. The LDX #0 and LDA #0 instructions clear the X register and the accumulator. The LDY #5 instruction sets up the Y register for use as a pass counter. The CLC instruction clears the carry bit in preparation for the ADC (add with carry) instruction.

Add Byte from Table. The ADC 500H,X instruction adds the byte at location 500H, indexed by the X register, to the contents of the accumulator.

Point to New Byte. The INX instruction increments the X register, which then points to the next byte in the table.

Reduce Pass Counter. The DEY instruction decrements the Y register. Notice that since the DEY instruction occurs just after the ADC instruction, the pass counter is decremented after each addition is performed.

Pass Counter=0? The BNE instruction checks to see if the Y register is zero. If the Y register does not contain zero, then the program returns to LOOP and adds another number into the accumulator. The instructions between CLC and BNE constitute the loop in this program. Since the Y register originally held the number 5, the instructions within the loop will be executed five times.

Exit. When the pass counter reaches zero, the program does not jump in response to the BNE instruction, but instead proceeds to the STA 04F7H and NOP instructions. These instructions are used by the 8002A μ Processor Lab to exit from the user program. (The exit is a service call. See Section 6 of this manual for a full explanation of service calls.)

Run the Demonstration Run Program

Assemble the Source Code. Enter the following command line to assemble the source code and store the assembler output in an object file and a list file:

```
> ASM DEMO;O DEMO;A DEMO%S <CR>
```

```
Tektronix      6500 ASM V3.X
**** Pass 2
   16 Source Lines      16 Assembled Lines   47047 Bytes available
   >>> No assembly errors detected <<<
*ASM* EOJ
```


- Check that memory was filled with zeros.

```
> DUMP 800 8FF <CR>
```

upper address
 lower address

```
0800=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0810=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0820=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0830=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0840=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0850=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0860=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0870=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08A0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08B0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08C0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08D0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08E0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
08F0=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

address of first byte in block
 values of bytes in block
 ASCII characters corresponding to data

- Load the object code file into memory.

```
> LOAD DEMO;0 <CR>
```

object code file

```
TRANSFER ADDRESS: 0800
*LOAD* EOJ
```

● Check memory contents again.

> DUMP 800 8FF <CR>

0800=A2	00	A9	00	A0	05	E8	7D	00	05	E8	88	D0	F9	8D	F7
0810=04	EA	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0820=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0830=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0840=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0850=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0860=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0870=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0880=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0890=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08A0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08B0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08C0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08D0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08E0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08F0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

address of first byte in block
 values of bytes in block
 ASCII characters corresponding to data

● Dump to a line printer.

> DUMP 800 8FF LPT1 <CR>

lower address
 upper address
 DUMP display device (line printer)

0800=A2	00	A9	00	A0	05	18	7D	00	05	E8	88	D0	F9	8D	F7
0810=04	EA	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0820=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0830=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0840=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0850=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0860=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0870=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0880=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0890=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08A0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08B0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08C0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08D0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08E0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
08F0=00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

- Invoke the Debug system.

```
> DEBUG <CR>
```

- Disassemble object code.

```
> DISM 800 811 <CR>
```

lower address upper address

```

LOC  INST  MNEM OPER
0800 A200  LDX  #00
0802 A900  LDA  #00
0804 A005  LDY  #05
0806 18    CLC
0807 7D0005 ADC  0500,X
080A E8    INX
080B 88    DEY
080C DOF8  BNE  F8
080E 8DF704 STA  04F7
0811 EA    NOP
  
```

Execute the Program.

- Choose the emulation mode.

```
> EMULATE 0 <CR>
```

emulation mode 0

```
*EMULATE* EOJ
```

- Set values to the table in memory.

```
> EXAM 500 <CR>
```

address

```
0500=00-01 20-02 20-03 20-04 20-05 20 <CR>
```

When 0500=00 appears, type 0102030405.
 The system fills in the rest.
 The second carriage return
 terminates the EXAM command.

- Set a breakpoint to suspend execution after the last addition.

```
> BKPT 504 R <CR>
      |
      | "Read" parameter
      |
      |
      | address
```

- Start program execution.

```
> GO 800 <CR>
```

LOC	INST	MNEM	OPER	X/Y	I/PC	EADD	SR	RA	RX	RY	SP	TC	IOS
0807	7D0005	ADC	0500,X	04		0504	24	0F	04	01	10	21AD	80
0807	BREAK												

|
RA=contents of the
accumulator

In the above BKPT display line, notice that the accumulator contains the hexadecimal sum of the numbers in the memory table: $1 + 2 + 3 + 4 + 5 = 0F$.

You have successfully assembled, loaded, and executed the Demonstration Run program. Next you will use some system commands to monitor the program's execution.

Monitor Program Execution.

The following command lines let you monitor program execution. As the program proceeds, you can watch the changes in the emulator's registers, and observe the effect of each instruction.

● **Trace all instructions.**

> TRACE ALL <CR>

> GO 800 <CR>

LOC	INST	MNEM	OPER	X/Y	I/PC	EADD	SR	RA	RX	RY	SP	TC	IOS
0800	A200	LDX	#00				26	0F	00	01	10	43CC	80
0802	A900	LDA	#00				26	00	00	01	10	1C9E	80
0804	A005	LDY	#05				24	00	00	05	10	3EBC	80
0806	18	CLC					24	00	00	05	10	178E	80
0807	7D0005	ADC	0500,X	00		0500	24	01	00	05	10	39AD	80
080A	E8	INX					24	01	01	05	10	127F	80
080B	88	DEY					24	01	01	04	10	349D	80
080C	DOF8	BNE	F8		080C	0806	24	01	01	04	10	0D6E	80
0806	18	CLC					24	01	01	04	10	3497	80
0807	7D0005	ADC	0500,X	01		0501	24	03	01	04	10	2F8C	80
080A	E8	INX					24	03	02	04	10	085D	80
080B	88	DEY					24	03	02	03	10	2A7C	80
080C	DOF8	BNE	F8		080C	0806	24	03	02	03	10	034E	80
0806	18	CLC					24	03	02	03	10	9366	80
0807	7D0005	ADC	0500,X	02		0502	24	06	02	03	10	256D	80
080A	E8	INX					24	06	03	03	10	478C	80
080B	88	DEY					24	06	03	02	10	205D	80
080C	DOF8	BNE	F8		080C	0806	24	06	03	02	10	427C	80
0806	18	CLC					24	06	03	02	10	5638	80
0807	7D0005	ADC	0500,X	03		0503	24	0A	03	02	10	1B4E	80
080A	E8	INX					24	0A	04	02	10	3D6D	80
080B	88	DEY					24	0A	04	01	10	163F	80
080C	DOF8	BNE	F8		080C	0806	24	0A	04	01	10	385D	80
0806	18	CLC					24	0A	04	01	10	2678	80
0807	7D0005	ADC	0500,X	04		0504	24	0F	04	01	10	112E	80
0807	BREAK												

breakpoint notice

the contents of the Y register, the pass counter

the contents of the X register, the table pointer

the contents of the accumulator

● Set a breakpoint after a specific instruction.

> TRACE ALL <CR>

> TRACE <CR>

TRACE ALL,,0000,FFFF

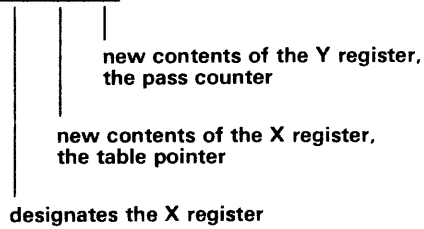
> BKPT 804 <CR>

> GO 800 <CR>

LOC	INST	MNEM	OPER	X/Y	I/PC	EADD	SR	RA	RX	RY	SP	TC	IOS
0800	A200	LDX	#00				26	0F	00	01	10	21E1	80
0802	A900	LDA	#00				26	00	00	01	10	43FF	80
0804	A005	LDY	#05				24	00	00	05	10	1CD1	80
0804	BREAK												

● Set new values to pass counter and table pointer; check results.

> SET RX 02 02 <CR>



> DSTAT <CR>

PC=	BP=	R	M=	SR	RA	RX	RY	SP	TC	IOS	
0806	0504	R	0804	M=0	24	00	02	02	10	3EEF	80

|
 Y register contents
 (pass counter)

|
 X register contents
 (table pointer)

- Resume program execution.

> GO <CR>

```

LOC  INST  MNEM OPER  X/Y I/PC EADD SR RA RX RY SP TC  IOS
0806  18     CLC
0807  7D0005 ADC  0500,X 02      0502 24 03 02 02 10 17C0 80
080A  E8     INX
080B  88     DEY
080C  D0F8   BNE  F8      080C 0806 24 03 03 01 10 34CF 80
0806  18     CLC
0807  7D0005 ADC  0500,X 03      0503 24 07 03 01 10 0DA1 80
080A  E8     INX
080B  88     DEY
080C  D0F8   BNE  F8      080C 0806 26 07 04 00 10 2AB0 80
*DOS* EOJ

```

Notice that the program performs two passes through the loop, and that the program adds the third and fourth numbers in the table ($3 + 4 = 7$).

The program encounters no more breakpoints, so execution is uninterrupted. The *DOS* EOJ (End Of Job) message and the prompt character > appear when the service call causes an exit to TEKDOS. The service call also takes you out of the Debug system. When you see the prompt, the program has finished execution and you may enter your next TEKDOS command.

Turning Off the 8002A μ Processor Lab and Peripherals

Refer to the Learning Guide, Section 1 of this manual, for a procedure for turning off the 8002A μ Processor Lab and peripherals.

You have now successfully assembled, loaded, executed, and monitored the execution of the Demonstration Run program. The remainder of this subsection contains 6500/1 emulator-specific reference material.

Emulator-Specific Commands and Displays

SET—Setting 6500/1 Register Values

The SET command can be used to change the values in the 6500/1 registers, beginning with the first register specified. Use the TEKDOS register designator to name the register in a command line. When setting more than one register with a single SET command, set the registers in the order in which they're listed below.

Table 9-23
6500/1 SET Command Parameters

Register	TEKDOS Register Designator
status register	SR
accumulator register	RA
index register X	RX
index register Y	RY
stack pointer	SP
timer/counter	TC
I/O status register	IOS

You must invoke the Debug system before you enter a SET command.

NOTE

Bits 4 and 5 of the status register cannot be altered with a SET command.

Example. Suppose the DSTAT command returns the following emulator status:

```
> DSTAT

PC=0000                SR RA RX RY SP TC  IOS
M=0 24 00 00 00 10 0000 00
```

To reassign values to the accumulator, the X register, and the Y register, use the following command line.

```
> SET RA 98 99 99
```

Since the SET command does not produce a display line, use the DSTAT command to check the results of the SET command.

> DSTAT

```

                SR RA RX RY SP TC   IOS
PC=0000        M=0 24 98 99 99 10 0000 00

```

RA now contains 98, RX contains 99, and RY contains 99.

RESET—Resetting 6500/1 Emulator Status

The RESET command causes the following values to be altered:

- I/O ports PA, PB, PC, and PD (memory locations 0080, 0081, 0082, and 0083) are each loaded with the value FF.
- Bit 2 of the status register is set, which disables interrupts.
- The stack pointer is decremented by three.
- The I/O status register is cleared.

DSTAT—Sample 6500/1 Emulator Status Display

The DSTAT command sends program execution status and breakpoint status to the assigned Debug output device or file. All values in the DSTAT display line are hexadecimal. You must invoke the Debug system before you enter a DSTAT command.

This is an example of a DSTAT display line for the 6500/1 emulator:

> DSTAT

```

                SR  RA  RX  RY  SP  TC  IOS
PC=0807 BP=0504 R 0806 M=0  24  00  02  03  10  3FDA  A0

```

The DSTAT display includes:

PC= the program counter, which contains the address of the next instruction to be executed by the emulator; in this example, address 0807. The GO command can be used to change the contents of the program counter.

BP= the breakpoints and breakpoint parameters, if any. In this example, one breakpoint is at address 0504, when TEKDOS reads from that address; and a second breakpoint is at address 0806, when TEKDOS reads from or writes to that address.

R or W breakpoint parameter:
 R = Break on Read
 W = Break on Write
 Blank = Break on Read or Write

The BKPT and CLBP commands can be used to change the breakpoint status.

- M= the emulation mode. In this example, emulation mode 0. The EMULATE command can be used to change the emulation mode.
- SR= the contents of the status register. In this example, the status register contains 24. The SET command can be used to change this register. See Fig. 9-15.
- RA= the contents of the accumulator. In this example, the accumulator contains 00. The SET command can be used to change this value.
- RX= the contents of the X register. In this example, the X register contains 02. The SET command can be used to change this value.

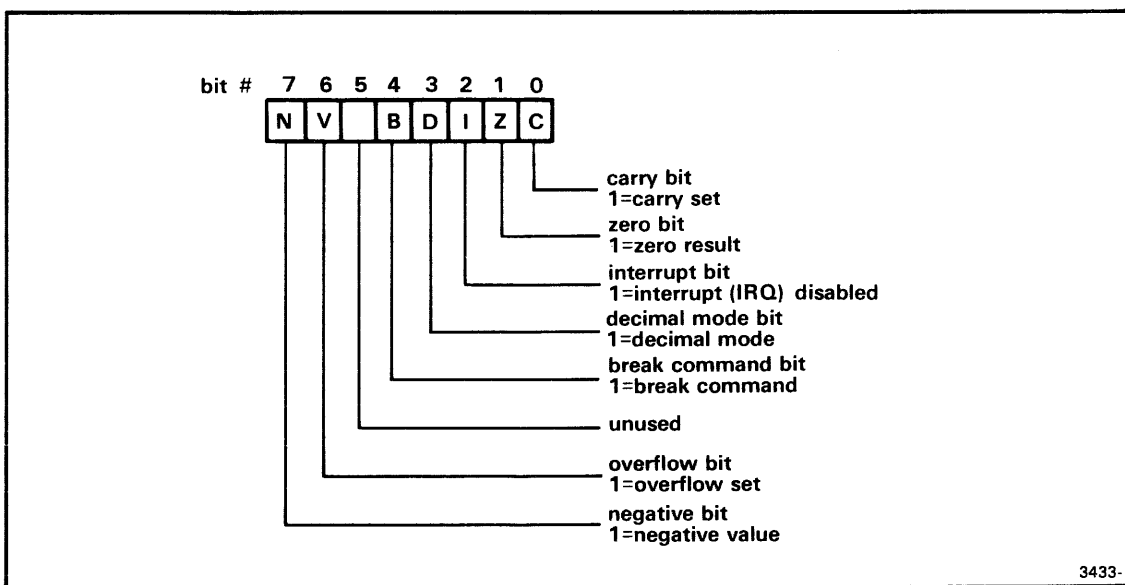


Fig. 9-15. 6500/1 Status Register.

This figure shows the bit configuration for the 6500/1 status register.

- RY= the contents of the Y register. In this example, the Y register contains 03. The SET command can be used to change this value.
- SP= the stack pointer, which contains the memory address of the stack; in this example, 10. The contents of the stack pointer can be changed using the SET command.
- TC= the contents of the timer/counter. In this example, the timer/counter contains 3FDA. The SET command can be used to alter the timer/counter.
- IOS= the contents of the I/O status register. In this example, the I/O status register contains A0. The SET command can be used to change the I/O status register.

DISM—Sample 6500/1 Disassembled Code

Use the DISM command to verify that the hexadecimal values in memory correspond to the proper assembly language instructions. DISM displays object code, assembly language mnemonics, and operands on the assigned Debug output device or file. You must first invoke the Debug system in order to enter a DISM command.

The following is a sample of 6500/1 DISM command output:

```

> DEBUG
> DISM 800 810

LOC      INST      MNEM      OPER
0800     A200     LDX      #00
0802     A900     LDA      #00
0804     A005     LDY      #05
0806     18      CLC
0807     7D0005  ADC      0500,X
080A     E8      INX
080B     88      DEY
080C     D0F8     BNE      F8
080E     8DF704  STA      04F7
    
```

} address or data being operated on
} symbolic names for the instructions
} hexadecimal machine language instructions
} memory addresses

TRACE—Sample 6500/1 Trace Display

The TRACE command lets you monitor program execution. To begin program execution, you enter the GO command. When trace conditions have been set, this GO command enables TRACE lines to be sent to the assigned Debug output device or file. All values in the TRACE display lines are hexadecimal. You must invoke the Debug system before you enter a TRACE command.

Any one of the following conditions will stop both the TRACE display and program execution:

- you press the space bar to suspend the TRACE display and program execution;
- you press the ESC key to suspend program execution;
- the program triggers a breakpoint;
- the program reaches a Service Call which exits from or aborts the program.

The TRACE SHORT format on the 6500/1 emulator is the same as the TRACE LONG format.

NOTE

The emulator runs slower than normal processing speeds when TRACE conditions have been set.

The following is an example of 6500/1 TRACE command output:

- > DEBUG
- > TRACE ALL
- > GO 800

LOC	INST	MNEM	OPER	X/Y	I/PC	EADD	SR	RA	RX	RY	SP	TC	IOS
0800	A200	LDX	#00				26	0F	00	01	10	43CC	80
0802	A900	LDA	#00				26	00	00	01	10	1C9E	80
0804	A005	LDY	#05				24	00	00	05	10	3EBC	80
0806	18	CLC					24	00	00	05	10	178E	80
0807	7D0005	ADC	0500,X	00		0500	24	01	00	05	10	39AD	80

BKPT—Sample 6500/1 Breakpoint Display

The BKPT command sets a breakpoint, which suspends program execution when TEKDOS either reads from or writes to the specified address. BKPT sends a trace line to the assigned Debug output device or file. The GO command resumes program execution after the breakpoint. All values in the BKPT display are hexadecimal. You must invoke the Debug system before you enter a BKPT command. See the 6500/1 Design Features, elsewhere in these 6500/1 Emulator Specifics, for additional information on the BKPT command.

NOTE

The emulator runs slightly slower than normal processing speeds when a breakpoint has been set.

The following is an example of 6500/1 BKPT command output:

```
> DEBUG
> BKPT 504
> GO 800
```

```
LOC INST  MNEM OPER  X/Y I/PC EADD SR RA RX RY SP TC  IOS
0807 7D0005 ADC  0500,X 04      0504 24 OF 04 01 10 21AD A0
0807 BREAK
```

The effective address in the trace line is calculated as follows:

- For indexed addressing, X/Y contains the contents of either the X or Y register. The effective address of the previous instruction is the sum of the register contents and the operand.
- For relative branch instructions, I/PC contains the contents of the program counter. The effective address of the next instruction to be executed is the sum of the program counter and the operand.
- For indirect instructions, I/PC contains the contents of the indirect address.

6500/1 Service Calls

Service calls (SVCs) allow you to use additional 8002A system capabilities while your program is running on the emulator processor.

TEKDOS allows you to define six different service calls at one time. Each of the SVCs is associated with an address and a set of instructions. Any of the following 6500/1 instructions may be used to generate an SVC:

ADC	DEC	ORA
AND	EOR	ROL
ASL	INC	ROR
BIT	LDA	SBC
CMP	LDX	STA
CPX	LDY	STX
CPY	LSR	STY

To generate a 6500/1 service call, the instruction must have one of the six SVC addresses in its operand field, and must be followed by a NOP instruction to allow time for the SVC to occur.

Table 9-24 lists the code for each 6500/1 SVC. In this table and in the SVC demonstration program shown in Fig. 9-16, the STA instruction is used to generate service calls; however, any instruction in the above list may be used.

Table 9-24
6500/1 Service Calls

SVC Number	6500/1 Service Calls	SRB Pointer
1	STA 04F7H NOP	40, 41
2	STA 04F6H NOP	42, 43
3	STA 04F5H NOP	44, 45
4	STA 04F4H NOP	46, 47
5	STA 04F3H NOP	48, 49
6	STA 04F2H NOP	4A, 4B

The operand of the instruction sends TEKDOS to a specific memory address called the SRB (Service Request Block) pointer. That specific memory address tells TEKDOS where to find the data (stored in the SRB) that, in turn, tells TEKDOS which service to perform. Section 6 of this manual explains service calls, service request blocks, and SRB pointers in detail.

SVC Demonstration

Figures 9-16 and 9-17 are an assembler listing of a 6500/1 program that uses four SVC functions: Assign Channel, Read ASCII, Write ASCII, and Abort. The program's algorithm is explained in Section 6, Service Calls. Section 6 also demonstrates the program, using a version of the program written in 8080A assembly language. You can perform a parallel demonstration with the 6500/1 emulator using the program in Figs. 9-16 and 9-17.

```

                ORG 40H          ;BEGINNING OF SRB VECTOR
                BYTE HI(SRB1FN),LO(SRB1FN)
                BYTE HI(SRB2FN),LO(SRB2FN)
                BYTE HI(SRB3FN),LO(SRB3FN)
                BYTE HI(SRB4FN),LO(SRB4FN)
                BYTE HI(SRB5FN),LO(SRB5FN)
;
                END OF SRB VECTOR
                ORG 800H        ; SET UP SRB AREAS

;
                SRB1 = ASSIGN "CONI" TO CHANNEL 0
SRB1FN  BYTE 10H              ; ASSIGN
SRB1CH  BYTE 00H             ; TO CHANNEL 0
SRB1ST  BLOCK 01H           ; STATUS RETURNED HERE
SRB1BY  BLOCK 01H           ; RESERVED
SRB1CO  BLOCK 01H           ; RESERVED
SRB1LE  BYTE 05H            ; LENGTH OF "CONI"+<CR>
SRB1HI  BYTE HI(CONI)       ; POINTER TO
SRB1LO  BYTE LO(CONI)       ; "CONI"+<CR>
;
                END OF SRB1

;
                SRB2= ASSIGN "LPT1" TO CHANNEL 1
SRB2FN  BYTE 10H              ; ASSIGN
SRB2CH  BYTE 01H             ; TO CHANNEL 1
SRB2ST  BLOCK 01H           ; STATUS RETURNED HERE
SRB2BY  BLOCK 01H           ; RESERVED
SRB2CO  BLOCK 01H           ; RESERVED
SRB2LE  BYTE 05H            ; LENGTH OF "LPT1"+<CR>
SRB2HI  BYTE HI(LPT1)       ; POINTER TO
SRB2LO  BYTE LO(LPT1)       ; "LPT1"+<CR>
;
                END OF SRB2

;
                SRB3 = READ ASCII LINE FROM CONI (CHANNEL 0)
SRB3FN  BYTE 01H              ; READ ASCII
SRB3CH  BYTE 00H             ; FROM CHANNEL 0
SRB3ST  BLOCK 01H           ; STATUS RETURNED HERE
SRB3BY  BLOCK 01H           ; RESERVED
SRB3CO  BLOCK 01H           ; BYTE COUNT RETURNED HERE
SRB3LE  BYTE 00H            ; 256 BYTES IN OUR BUFFER
SRB3HI  BYTE HI(BUFFER)     ; POINTER TO
SRB3LO  BYTE LO(BUFFER)     ; OUR BUFFER
;
                END OF SRB3

;
                SRB4 = WRITE ASCII LINE TO LPT1 (CHANNEL 1)
SRB4FN  BYTE 02H              ; WRITE ASCII
SRB4CH  BYTE 01H             ; TO CHANNEL 1
SRB4ST  BLOCK 01H           ; STATUS RETURNED HERE
SRB4BY  BLOCK 01H           ; RESERVED
SRB4CO  BLOCK 01H           ; BYTE COUNT RETURNED HERE
SRB4LE  BLOCK 01H           ; FILLED IN BY PROGRAM
SRB4HI  BYTE HI(BUFFER)     ; POINTER TO
SRB4LO  BYTE LO(BUFFER)     ; OUR BUFFER
;
                END OF SRB4

```

3433-2

Fig. 9-16. 6500/1 SVC Demonstration program (sheet 1 of 2).

```

;      SRB5 = ABORT (CLOSE ALL CHANNELS AND TERMINATE)
SRB5FN  BYTE  1FH          ; ABORT
SRB5CH  BLOCK 01H         ; RESERVED
SRB5ST  BLOCK 01H         ; RESERVED
SRB5BY  BLOCK 01H         ; RESERVED
SRB5CO  BLOCK 01H         ; RESERVED
SRB5LE  BLOCK 01H         ; RESERVED
SRB5HI  BLOCK 01H         ; RESERVED
SRB5LO  BLOCK 01H         ; RESERVED
;
;
BUFFER  BLOCK 100H        ; OUR I/O AREA
CONI    ASCII "CONI"      ; ASCII OF "CONI"
        BYTE 0DH          ; + <CR>
LPT1    ASCII "LPT1"      ; ASCII OF "LPT1"
        BYTE 0DH          ; + <CR>
;
;      END OF DATA DEFINITIONS
;
;
;      BEGINNING OF EXECUTABLE CODE
START   ORG 0B00H         ; ENTRY POINT INTO PROGRAM
        STA 04F7H         ; CALL SVC1
        NOP              ; TO ASSIGN "CONI"
        LDA SRB1ST        ; CHECK THE STATUS TO SEE
        CMP #00H          ; IF ALL WENT WELL
        BNE ABORT         ; NO? STOP EVERYTHING
        STA 04F6H         ; YES? CALL SVC2
        NOP              ; TO ASSIGN "LPT1"
        LDA SRB2ST        ; CHECK THE STATUS TO SEE
        CMP #00H          ; IF ALL WENT WELL
        BNE ABORT         ; NO? STOP EVERYTHING
LOOP    STA 04F5H         ; CALL SVC3
        NOP              ; TO READ A "CONI" LINE
        LDA SRB3ST        ; INTO BUFFER
        CMP #00H          ; ALL OK?
        BNE ABORT         ; NO? STOP EVERYTHING
        LDA SRB3CO        ; TRANSFER I/O COUNT
        STA SRB4LE        ; TO BUFFER LENGTH FOR WRITE
        STA 04F4H         ; CALL SVC4
        NOP
        LDA SRB4ST        ; CHECK TO SEE IF
        CMP #00H          ; ALL IS OK
        BEQ LOOP          ; YES? BACK TO READ ANOTHER LINE
        ; NO? FALL THRU TO TERMINATION
;
ABORT   STA 04F3H         ; CALL SVC5
        NOP              ; TO DO THE ABORT
        JMP $             ; SHOULD NEVER REACH HERE
        END START        ; ASSEMBLER DIRECTIVE (ENTRY POINT)

```

3433-3

Fig. 9-17. 6500/1 SVC Demonstration program (sheet 2 of 2).

This program shows the use of four 8002A service calls. The program's algorithm is explained in Section 6 of this manual, Service Calls. You may load this program into 8002A program memory using the TEKDOS Editor, the TEKDOS Assembler, and the LOAD command. The program accepts a line of ASCII characters from the system terminal; then, after a carriage return, it prints the line on a line printer. Terminate the program by entering a CTRL-Z when the program is waiting for input.

6500/1 Design Features

6500/1 Address Space. Figure 9-18 shows the memory allocation in the 6500/1. Refer to this figure as you read the following discussion. The 6500/1 can access memory locations between 0800 and 0FFF. You may also use locations 0400 to 07FF of 8002A program memory when using emulation modes 0 or 1. Locations 04F0—04F7, however, are reserved for SVC's.

In addition, locations 0—3F and 80—8F can be accessed only while operating within the Debug system. The following considerations apply:

- 80, 81, 82, and 83 are I/O Port A, Port B, Port C, and Port D, respectively.
- 84—87 contain the timer/counter. To assign a new value to the timer/counter, you must use the SET command.
- 88—8E are not accessible to TEKDOS.
- 8F contains the I/O status register. To assign a new value to this location use the SET command.

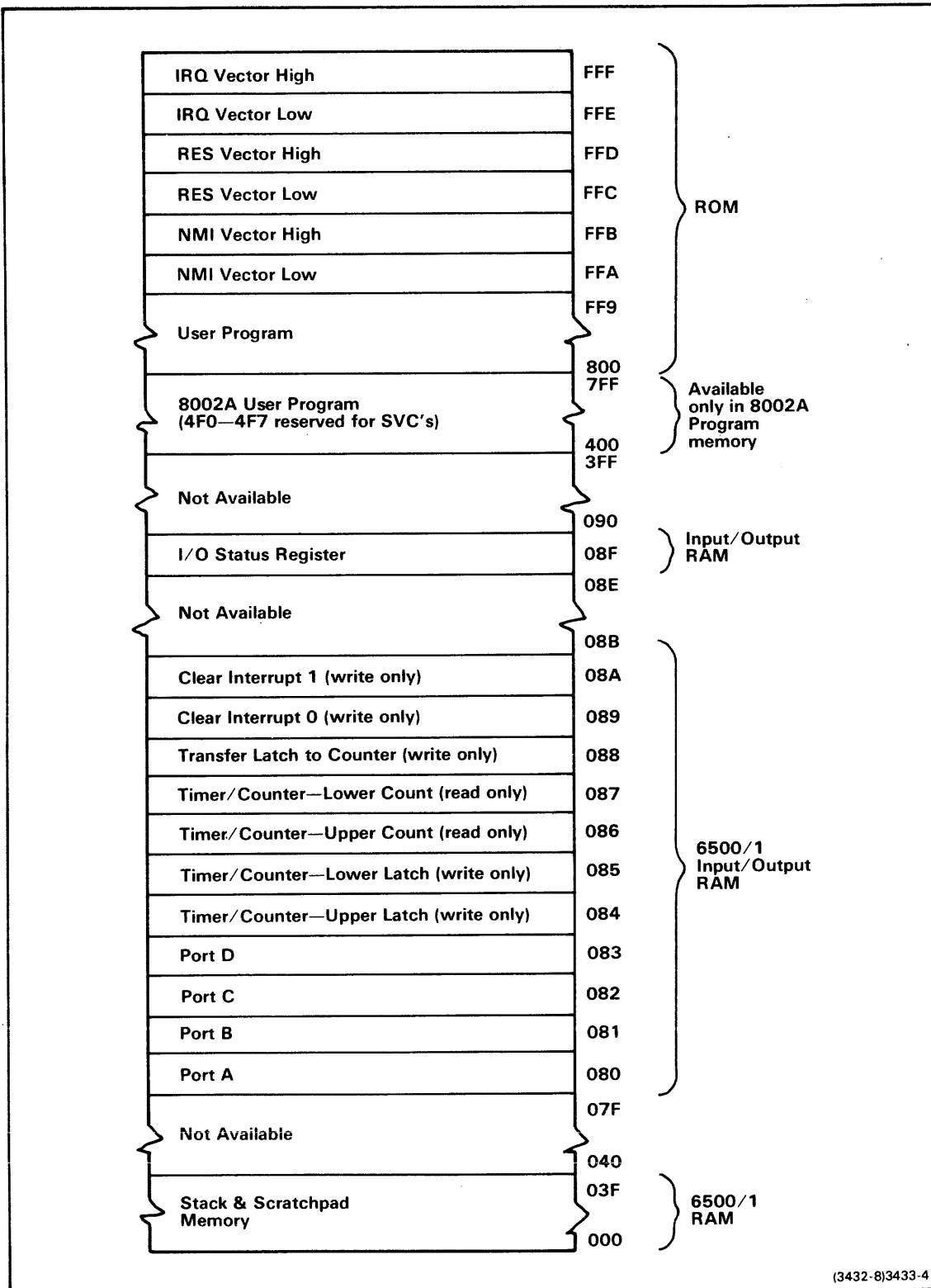
NOTE

Any attempt to enter the GO command with an address parameter less than 400H will cause TEKDOS to display the following error message:

```
***ERROR 37 INVALID GO ADDRESS
```

Programs may be executed in this address range only if the following conditions are met:

- *TRACE is off.*
- *No breakpoints are set between 0 and 3F.*
- *The address is accessed by a microprocessor jump instruction.*



(3432-8)3433-4

Fig. 9-18. 6500/1 Memory Allocation.

Addresses 400H—7FFH may not be accessed by the 6500/1 microcomputer. However, you may use this address range to develop your program on the 8002A.

6500/1 Breakpoint Features.

- A breakpoint placed immediately after a conditional jump instruction will cause a break whether the condition is met or not.
- An unexpected BKPT or RTPA break will occur when all of the following conditions are true:

— The absolute indexed address mode is used with one of the following instructions:

ADC	CMP	EOR	LDY
AND	CPX	LDA	ORA
BIT	CPY	LDX	SBC

— The effective address (sum of the operand and index register) crosses a page boundary.

— A BKPT or RTPA break is set exactly one page preceding the effective address.

When all of these conditions are true, a break message is displayed on the line of the absolute indexed instruction.

Clocking the Emulator. The 8002A does not support the resistor-only mask option for clocking the 6500/1 emulator in emulation mode 1.

Edge Detect Capabilities. To use the edge detect capabilities of the 6500/1 microcomputer, follow this sequence:

1. Set PA0 and PA1 high by loading 1's in bits 0 and 1 of location 80H.
2. Clear the edge detect flags by writing to locations 089H and 08AH. You can accomplish this by using the STA 089H and STA 08AH instructions.
3. Enable PA0 and PA1 (edge detect interrupt) by setting bits 2 and 3 of the I/O status register. Use the SET command to alter the I/O status register.

Timer/Counter Trace Display. The timer/counter and IOS bit 7 (the timer/counter underflow bit) are not accurately displayed in the trace line when operating in Timer Mode 0 or 1.

Status Register. If you set the zero and negative flag bits of the status register without using the BIT instruction, and the TRACE command is active, the following error message will appear:

```
***SR NOT RESTORED
```

The system will then wait for you to change one of these two bits. You can use the SET command to accomplish this. To continue, enter the GO command.

NOTE

The 6500/1 emulator will not execute properly when the following conditions occur simultaneously:

- *The TRACE command is active.*
- *Status register bit 3 = 0.*
- *The stack pointer is less than or equal to 1.*