TABLE OF CONTENTS

WANG 3300

OBJECT TAPE LOADING PROCEDURE

Revised: July 15, 1972

WANG 3300 OBJECT TAPE LOADING PROCEDURE

TABLE OF CONTENTS

PART 1 PAPER TAPE AND CASSETTE TAPE INSTRUCTIONS

Figure A—1 is a photograph of a Wang 3315 Teletype terminal. The paper tape reader and punch units are located to the left of the keyboard; the rotary power switch is located on the lower right-hand edge of the terminal.

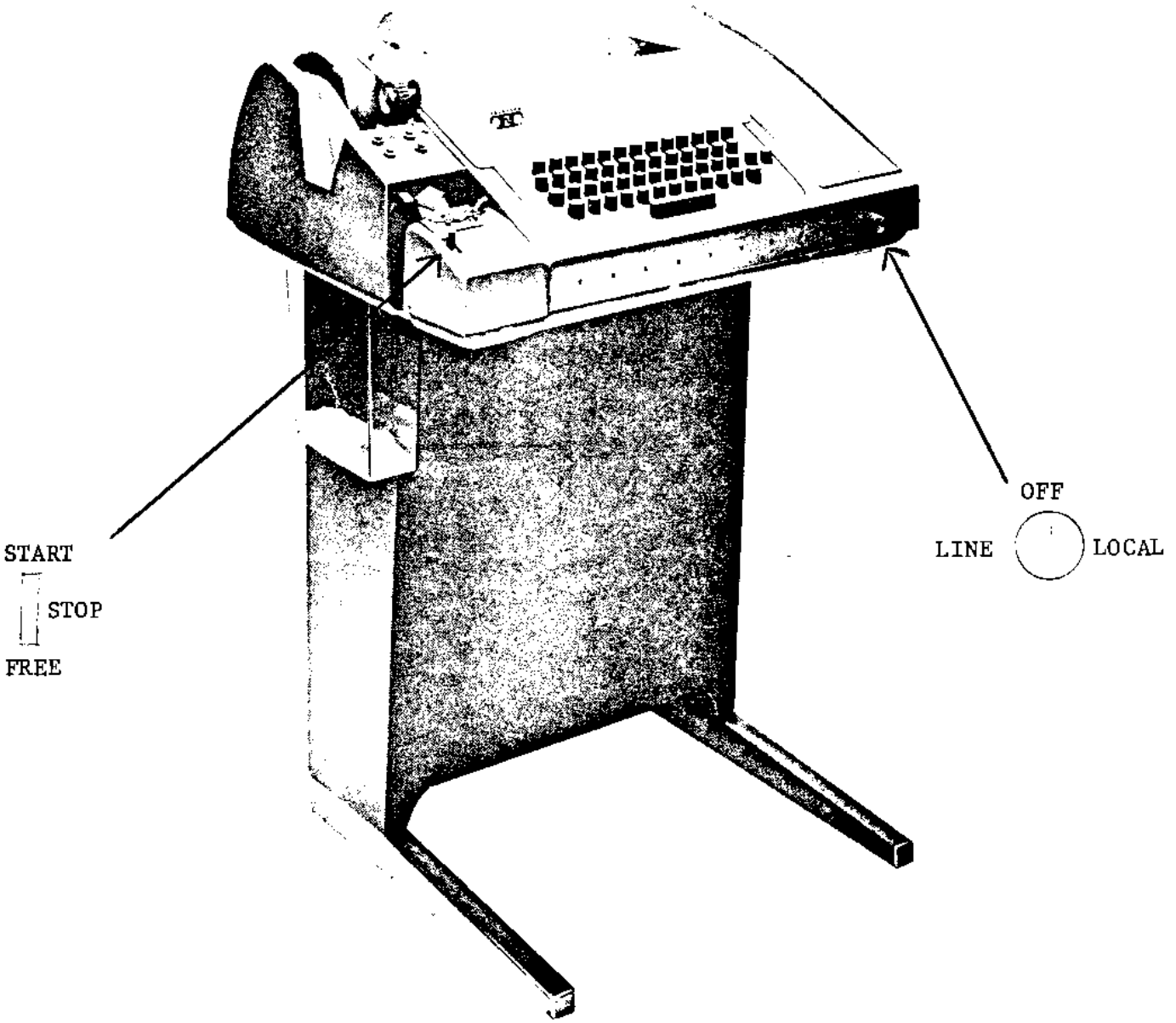


FIGURE A-1: WANG 3315 TELETYPE TERMINAL

Step #1. POWERING UP THE SYSTEM

Make sure the rotary power switch on all Teletype terminals is OFF. On the rear of the 3300 chassis (Figure 1-2), turn the power ON/OFF switch to the ON position.

Step #2. INITIALIZING THE SYSTEM.

1. The Console Registers.

The front panel of the 3300 computer (Fig. 1–1) contains five 8-bit registers labeled Z,A,B,C, and MEMORY. The bit locations in each register are labeled 80, 40, 20, 10, 8, 4, 2, 1, respectively. There is also a 7-bit STATUS register whose bit locations are labeled Ca, P,Z,D,V,1 &digit zero, respectively. All bit locations on the panel are indicator lights which are ON when a bit has been set and OFF when a bit has been cleared.

2. The Control Buttons.

Along the bottom of the control panel are four groups of square buttons. The left-most group contains seven selector buttons labeled B, C, Z, A, S, M, and CORE, respectively. The first six are used to store information into any of the six registers shown on the panel; the button marked CORE is used to store or display the location in memory whose address appears in the B and C registers. The next group contains eight buttons, called Data Bit Entry Switches, labeled 80, 40, 20, 10, 8, 4, 2, 1. All addresses and instruction codes are keyed into the computer using these switches. When an entry switch is in the "in" position (depressed), it is ON and represents a 1-bit. When a switch is in the "out" position (not depressed), it is OFF and represents a 0-bit. The next group to the right consists of three buttons labeled RUN, DISPLAY, and ENTER. These switches specify what mode of operation the computer is in (i.e., about to RUN a program, DISPLAY the contents of a register or of a memory location, or about to ENTER an instruction code or an address). The last group contains four switches, labeled LOAD, STEP, EXQ, and GO. These are generally used for stopping and starting programs while the computer is in the RUN mode (i.e., the RUN switch is depressed).

3. Initializing the System.

To initialize the computer console, all the Data Bit Entry Switches must be placed in the OFF or "out" position, and the ENTER switch should be depressed placing the computer in the Data Entry mode. Following this, the B and C buttons should be pressed (they will automatically return to their "out" positions). If this has been done correctly, all bits in the B and C registers will be cleared. This, in effect, sets the computer memory address to location 0000.

To initialize the Wang 3315 Teletype terminal, the OFF button on the paper tape punch unit should be depressed, and the three-way switch on the tape reader should be placed in the STOP position (Fig. A–1). The rotary power switch should be set to its OFF position.

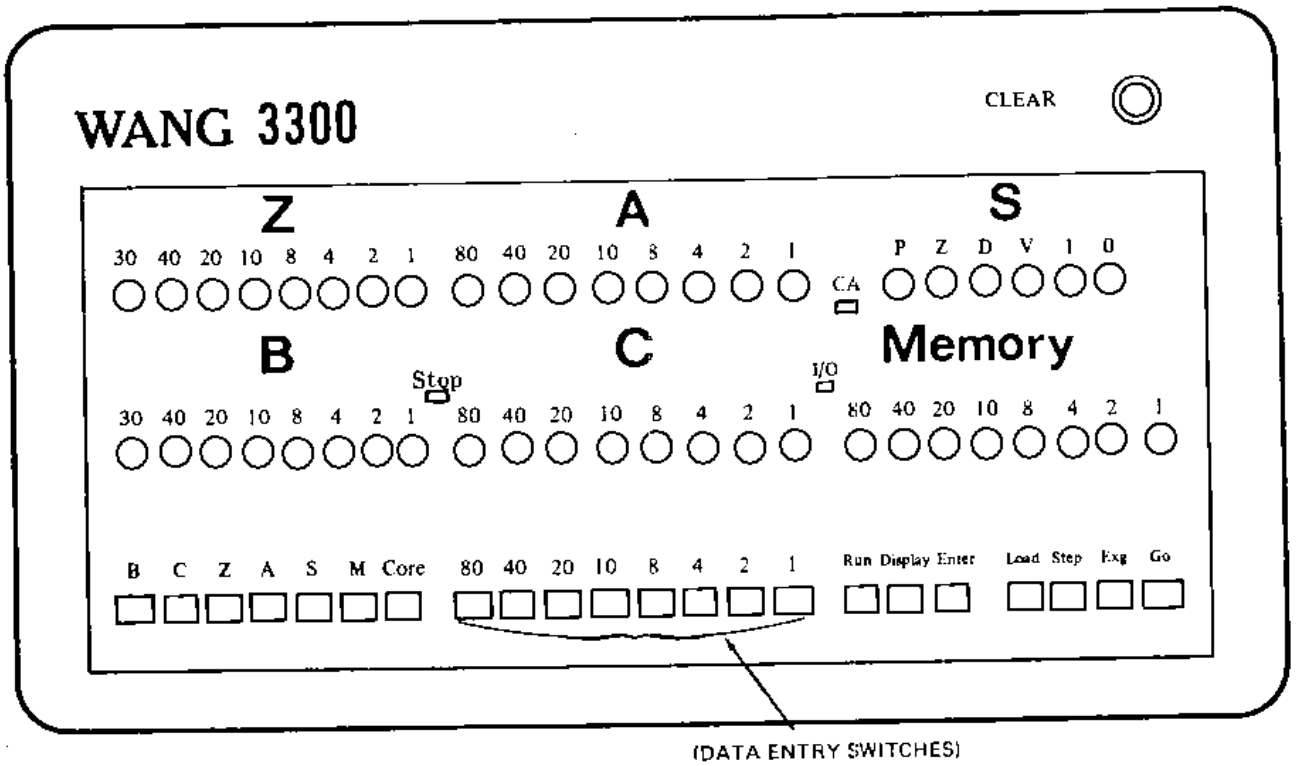


FIGURE 1-1: WANG 3300 COMPUTER CONSOLE

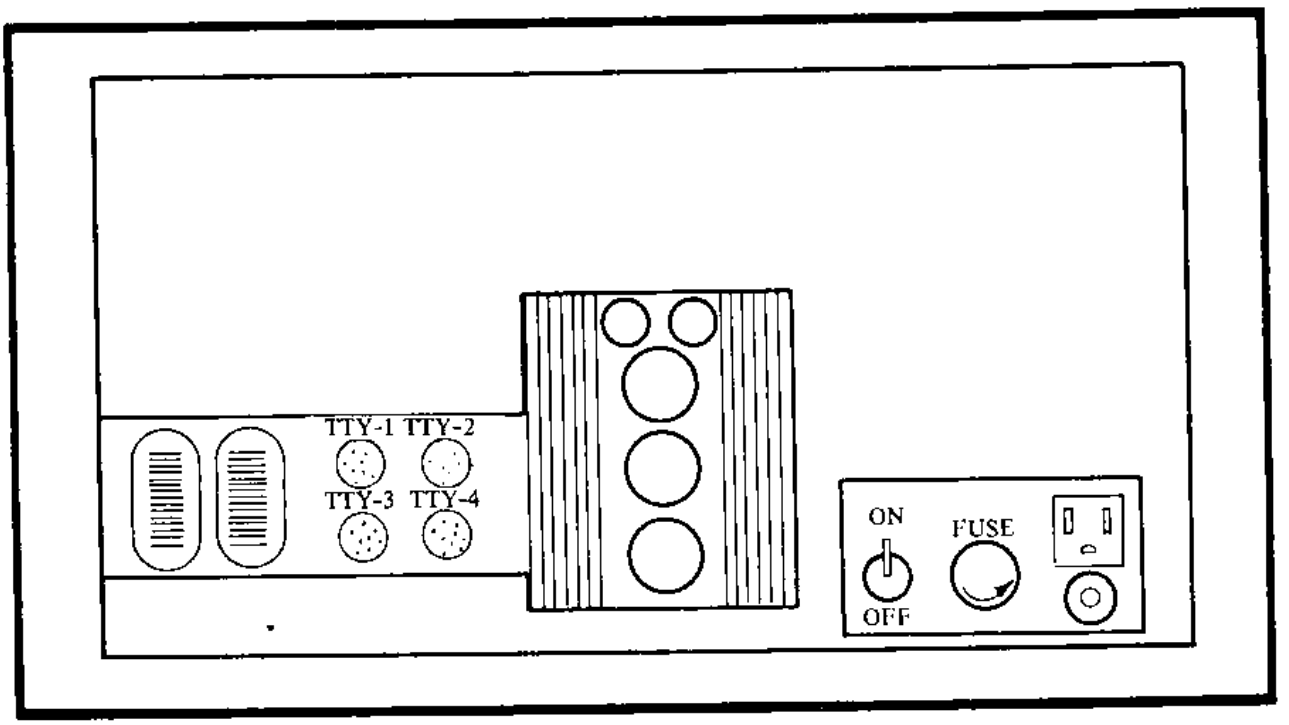


FIGURE 1-2: REAR OF WANG 3300 COMPUTER CHASSIS

Step #3. LOADING THE BOOTSTRAP.

The BOOTSTRAP program is a sequence of instruction codes and memory addresses which must be manually keyed into core memory using the buttons along the bottom of the 3300 control panel (Fig. 1–1). The code for each instruction and address in the BOOTSTRAP must be set up on the Data Bit Entry Switches.

1. Placing Information Into the Registers.

To enter data into registers or memory, the computer is set to Enter mode; i.e., the ENTER button is depressed and left in this position. Then, suppose the Entry Switches labeled 8, 4, and 1 were depressed (each switch will remain depressed until pushed a second time). Pushing the B button would place that bit configuration in the B register; i.e., the 8, 4, and 1 bits of the B register would go ON. Similarly, pushing the C, Z, A, S, and M buttons would place the 8, 4, 1 configuration in the C, Z, A, S, and MEMORY registers, respectively. The action caused by pushing the CORE button is less direct. When CORE is pressed, the computer interprets the contents of the B and C registers as the address of a location in core memory. It first moves the present contents of that location into the MEMORY register. It then places the bit configuration which is set on the Entry Switches into that location and increments the address in the B and C registers by one. The noticeable changes on the console when CORE is depressed occur in the C (and possibly the B) register and in the MEMORY register.

2. Notation Used in the BOOTSTRAP Instruction Codes.

All instruction codes and addresses in the BOOTSTRAP are expressed in hexadecimal notation. The hexadecimal number system contains sixteen digits labeled 0 through 9 and A through F. Some codes in the BOOTSTRAP have familiar representations such as 31 (read as "three-one," not thirty-one); others may be unfamiliar such as 0D (read as "zero-D"), or FE (read as "F-E"). The following table may be helpful in setting up codes on the Entry Switches. Darkened blocks represent switches or indicator lights in the ON position.

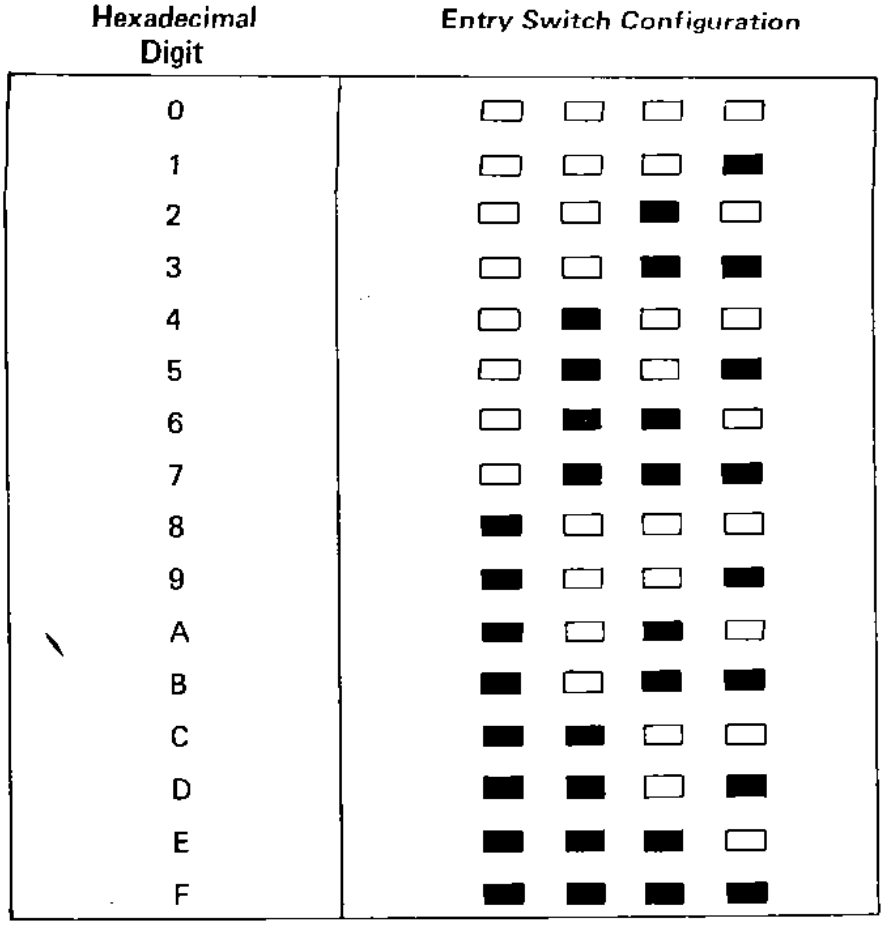


FIGURE A-2: ENTRY SWITCH CONFIGURATIONS

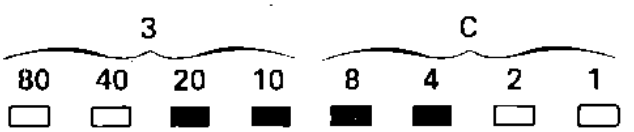
3. Setting Up Codes On the Entry Switches.

All instruction codes in the BOOTSTRAP, 3C for example, are set on the Entry Switches in the following manner:

i) the switch configuration (Fig. A–2) for the left-hand digit, **3**, is set on the leftmost four Entry Switches by depressing the 20 and the 10 switches;

ii) the configuration for the right-hand digit, **C**, is set on the rightmost four Entry Switches by depressing the 8 and the 4 switches.

The Entry Switches will then have the following configuration:



To reinitialize the Entry Switches, simply push each depressed switch and it will return to its OUT position.

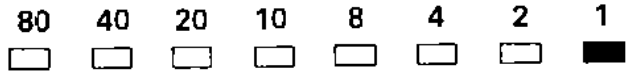
4. Keying the BOOTSTRAP Program Into the Computer.

The teletype BOOTSTRAP program is listed on pages 15, 16 at the end of this section. Column one shows the core locations for the instruction codes and addresses in column two. The last column gives the corresponding Entry Switch configurations.

The first instruction must be placed into core at location 0000. After initializing the system, this address will appear in the B and C registers and the computer will be in Data Entry mode. The BOOTSTRAP can be loaded by carrying out the following simple procedure:

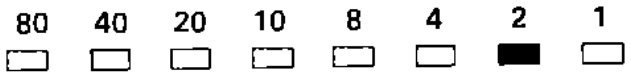
a) set up the first two digits (OD) in column 2 on the Entry Switches by depressing the 8, 4, and 1 switches (see steps i and ii).

b) press CORE; the code OD will be placed in core at location 0000, and the C register will be incremented to read 01.

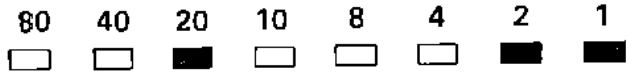


c) set up the next two digits (00) on the Entry Switches by returning the previously set switches to their OUT positions.

d) press CORE; the code (00) will be placed in core at location 0001 and the C register will be incremented to 02.



Repeat Steps C and D for each of the remaining instructions in the BOOTSTRAP program. When the final instruction code (61)* has been entered, the C register should read 23.



5. Reviewing the BOOTSTRAP Instruction Codes.

It is a good idea to review the instructions which have just been entered, and to reenter those which are not correct.

e) With the ENTER button depressed, return all Entry Switches to their OUT positions, and press the B button and then the C button to clear the B and C registers. This will reset the memory address back to 0000.

f) Depress the DISPLAY button, placing the computer in Display mode.

g) Press CORE once. The contents of location 0000 (i.e., OD) will be displayed in the MEMORY register, and the C register will be incremented to 01.

h) Continue pressing CORE, checking the instruction codes displayed in MEMORY against the list in Fig. 3. If an erroneous instruction code is discovered, that code must be reentered according to the following procedure:

* 61 for TTYLOAD; 44 for TTLOAD

6. Correcting Erroneous Instructions.

i) Return all Data Entry Switches to their OUT positions.

j) Depress the ENTER button.

k) Check the listing in the BOOTSTRAP to determine the core location of the instruction to be corrected, this location must be placed in the B and C registers according to step "l". Fig. A–2 (page 7) may be helpful here.

l) Set up the first two digits (00) of the location on the Entry Switches, and press the B button. Set up the last two digits of the location on the Entry Switches and press the C button. The B and C registers should now contain the correct location.

m) Set up the correct instruction code on the Entry Switches and press CORE.

n) Return all Entry Switches to their OUT positions. To continue reviewing the BOOTSTRAP instructions, press DISPLAY and continue pressing CORE as in Step g, page 8.

PART 2: PAPER TAPE LOADING INSTRUCTIONS

Step #4. LOADING THE TELETYPE LOADER

If the TCBOOT program is resident in the 3300 computer, then TTYLOAD can be loaded directly from cassette by following the instructions in PART 3, Step #5.

1. Clearing the B and C Registers.

When the BOOTSTRAP has been successfully checked the B and C registers must be reset to location 0000 by performing steps a and b below.

a) Depress ENTER.

b) Return all Entry Switches to their OUT positions, and press the B and then the C buttons.

2. Mounting the TTY LOADER Tape.

On the Teletype to be used in loading the system, turn the rotary switch to LINE (Fig. A-1). The three-way switch on the tape reader should be moved to FREE. The short TELETYPE LOADER tape should be placed in the paper tape reader unit. The plastic cap on the reader unit (Fig. A-1) should be raised, and the tape placed beneath it with the label facing upward and pointing toward the front of the terminal. The small sprocket holes directly behind the tape label should catch on the sprocket teeth beneath the plastic cap. Finally, the cap should be lowered and locked in its original position, and the three-way switch returned to the STOP position.

3. Executing the BOOTSTRAP Program.

On the 3300 computer console:

c) Depress the RUN button.

d) Press the CLEAR button.

e) Press the GO button.

On the Teletype tape reader unit:

f) Move the three-way switch from STOP to START. The TELETYPE LOADER tape will be read by the computer. When the trailer code at the end of the tape reaches the plastic cap on the reader, the computer will stop automatically, but the reader must be halted manually, according to step g.

g) Move the three-way switch to FREE.

4. Correcting a Faulty Load.

If the tape is not read, or if it stops before the trailer code near the end of the tape is reached, then the BOOTSTRAP should be rechecked (Steps e-h, page 8) and another attempt made to load the TELETYPE LOADER. Normally, only the contents of the last three core locations in the BOOTSTRAP (0020 – 0022) will have been altered.

Step #5. LOADING A 3300 OBJECT PROGRAM PAPER TAPE.

In order to load an object program paper tape, the TTYLOAD program must be resident in the 3300 computer. TTYLOAD can be loaded by following the instructions given in Step #4 above or by loading file #5, #6, or #7. from the Software User's System Cassette.

1. Mounting an Object Tape

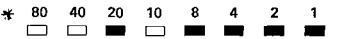
The object program tape should be placed in the tape reader in the same manner as the TELETYPE LOADER tape. (See page 9, step 2.) Make sure the teletype reader switch is in the "STOP" position.

2. Keying In the Starting Location of the Loader.

Before the 3300 object program tape can be loaded into the computer, the starting location of the LOADER program must be placed in the B and C registers; i.e, location 2F40. On the computer console:

a) Depress the ENTER button.

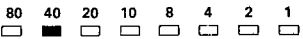
b) Set up *2F on the Entry Switches, i.e.,



and press the B button.

* The loader is generally loaded in the last page of available computer memory although it may be loaded in any page by the Bootstrap. The starting address of the loader will be XX40. Generally XX = 0F for a 4K system
XX = 1F for an 8K system
XX = 2F for a 12K system.

c) Set up 40 on the Entry Switches, i.e.,



and press the C button.

3. Executing the Loader Program.

d) Return all Entry Switches to their OUT positions.

e) Depress the RUN button.

f) Press CLEAR.

g) Press GO.

The TTY LOADER program will automatically start the tape reader. If the reader stops before reaching the trailer code at the end of the tape, a loading error has occurred and Step #5 should be repeated. When the entire tape is read, it should stop at the beginning of the blank trailer code, and all indicator lights in the A register should be off. This indicates a good load. If the load does not terminate in this manner, then repeat Step #5, page 10.

PART 3: CASSETTE TAPE LOADING INSTRUCTIONS

Step #4: LOADING THE TAPE CASSETTE BOOTSTRAP

1. Clearing the B and C Registers

When the BOOTSTRAP has been successfully checked, the B and C registers must be reset to location 0000 by performing steps a and b below.

a) Depress ENTER

b) Return all Entry Switches to their OUT position, and press the B and then the C buttons.

2. Mounting a TCBOOT Tape

Turn the rotary switch on the Teletype to LINE (Fig. A-1). The three-way switch on the tape reader should be set to FREE. The TCBOOT#1 or TCBOOT#2 tape for your system should be placed in the paper tape reader unit. The plastic cap on the reader should be raised and the tape placed beneath it with the label facing upward and pointing toward the front of the terminal. The small sprocket holes should catch on the sprocket teeth beneath the plastic cap. Finally, the cap should be lowered and locked into position, and the three-way switch returned to the STOP position.

3. Executing the BOOTSTRAP Program

On the 3300 computer console:

c) Press *RUN*, *CLEAR*, and *GO*

On the Teletype tape reader unit:

d) Move the three-way switch from STOP to START. The TCBOOT tape will be read by the computer. When the trailer code at the end of the tape reaches the plastic cap on the reader, the computer will stop automatically, but the reader must be halted manually according to step e.

e) Move the three-way switch to FREE.

4. Correcting a Faulty Load

If the tape is not read or if it stops before the trailer code is reached, then the BOOTSTRAP should be rechecked (Steps e - h page 8).

Step #5 LOADING A 3300 CASSETTE OBJECT PROGRAM

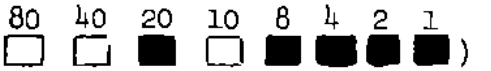
1. Loading The TCLOAD Program

a. Place the Operating System cassette into either the LEFT-HAND or the RIGHT-HAND port (depending on which TC BOOT has been loaded) of the unit directly connected to the 3300 computer. The LEFT-HAND port is considered to hold the NO. 1 CASSETTE, the RIGHT-HAND port holds the NO. 2 CASSETTE.

b. On the 3300 computer console,

Depress ENTER.

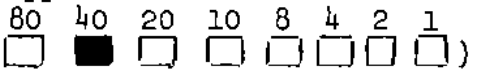
c. Set up 2F



on the entry switches.

d. Press B (2F appears in B).

e. Set up 40



f. Press C (40 appears in C).

g. Return all entry switches to their OUT position.

h. Press RUN, CLEAR, and GO

The computer will read the TCLOAD program from the first file on the Software User's Operating System cassette. It will halt with the execution address of TCLOAD (i.e. XY00 $_{16}$) in the B and C registers, and the file number 03_{16} in the A register.

2. The Software User's System Cassette

The Software User's system cassette contains seven program files:

i. files #01_{16} and #02_{16} contain the TCLOAD program (execution address = XY00_{16}) and the tape parameters.

ii. file #03_{16} contains the TCEDIT program (execution address = 0200).

iii. file #04_{16} contains the TCASMB program (execution address = 0200) and a copy of TCBOOT #1 (execution address = 0240).

iv. file #05_{16} contains CAP (execution address = 0900) and TTYLOAD (execution address = 0F40).

v. file #06_{16} contains CAP (execution address = 3900) and TTYLOAD (execution address = 3F40).

vi. file #07_{16} contains CAP (execution address = 5900) and TTYLOAD (execution address = 5F40).

3. Loading the System Cassette

Files 03_{16} through 07_{16} can be loaded into the computer by following the instructions given below. It is assumed that TCBOOT is resident in the 3300.

i. Load the TCLOAD program

j. Place the file number of the desired program into the A register.

i.e. for TCASMB:

i. depress ENTER

ii. set up an 04_{16} 80 40 20 10 8 4 2 1 (□ □ □ □ □ ■ □ □)
on the ENTRY SWITCHES

iii. press the A button

k. Press RUN, CLEAR, and GO

l. The selected file will be read and the number of the next sequential file will appear in the A register.

For 8-bit TTYLOAD, TCBOOT #1 & #2

TELETYPE BOOTSTRAP LISTING

				Core Location	Instruction Codes	Entry Switch Configuration							
						80	40	20	10	8	4	2	1
BOOT	OFS	X'FF'	....	0000	0D	□	□	□	□	■	■	□	■
				0001	00	□	□	□	□	□	□	□	□
	LZI	X'20'	....	0002	1B	□	□	□	■	■	□	■	■
				0003	20	□	□	■	□	□	□	□	□
	LAI	X'83'	....	0004	1A	□	□	□	■	■	□	■	□
				0005	83	■	□	□	□	□	□	■	■
	CIOJ		....	0006	3C	□	□	■	■	■	□	□	□
				0007	08	□	□	□	□	■	□	□	□
	STI	X'B0'	....	0008	30	□	□	■	■	□	□	□	□
				0009	B0	■	□	■	■	□	□	□	□
	JMP	*-2	....	000A	46	□	■	□	□	□	■	■	□
				000B	08	□	□	□	□	■	□	□	□
	LA*	BUFF	....	000C	C3	■	■	□	□	□	□	■	■
				000D	1E	□	□	□	■	■	■	■	□
	CI	0	....	000E	19	□	□	□	■	■	□	□	■
				000F	00	□	□	□	□	□	□	□	□
	STS	X'01'	....	0010	14	□	□	□	■	□	■	□	□
				0011	01	□	□	□	□	□	□	□	■
	JEQ	BOOT	....	0012	27	□	□	■	□	□	■	■	■
				0013	00	□	□	□	□	□	□	□	□
	ONS	X'01'	....	0014	0C	□	□	□	□	■	■	□	□
				0015	01	□	□	□	□	□	□	□	■
	UA+	LADR	....	0016	CE	■	■	□	□	■	■	■	□
				0017	20	□	□	■	□	□	□	□	□
	INC	RCNT	....	0018	42	□	■	□	□	□	□	■	□
				0019	22	□	□	■	□	□	□	■	□
	JNZ	BOOT+2	....	001A	25	□	□	■	□	□	■	□	■
				001B	02	□	□	□	□	□	□	■	□
	HLTJ	*	....	001C	00	□	□	□	□	□	□	□	□
				001D	1C	□	□	□	■	■	■	□	□
BUFF	DAC	X'0220'	....	001E	02	□	□	□	□	□	□	■	□
				001F	20	□	□	■	□	□	□	□	□
LADR	DAC	X'2F40'	....	0020	** 2F	□	□	■	■	■	■	■	■
				0021	40	□	■	□	□	□	□	□	□
RCNT	DC	X'61'	....	0022	61	□	■	■	□	□	□	□	■

** NOTE: (1) TTYLOAD is generally loaded into the last page of available memory although it may be loaded into any page. The starting address of where the loader is to be entered is contained in locations 20 and 21. Therefore, location 0020 = 0F for a 4K system
= 1F for an 8K system
= 2F for a 12K system, etc.

(2) TCBOOT is always loaded at 2F40.

(3) Locations 0020, 0021, and 0022 must be restored if the BOOTSTRAP is used more than once.

TELETYPE BOOTSTRAP LISTING

				Core Location	Instruction Codes	Entry Switch Configuration							
						80	40	20	10	8	4	2	1
BOOT	OFS	X'FF'	....	0000	0D								
				0001	00								
	LZI	X'20'	....	0002	1B								
				0003	20								
	LAI	X'83'	....	0004	1A								
				0005	83								
	CIOJ		....	0006	3C								
				0007	08								
	STI	X'B0'	....	0008	30								
				0009	B0								
	JMP	*-2	....	000A	46								
				000B	08								
	LA*	BUFF	....	000C	C3								
				000D	1E								
	CI	0	....	000E	19								
				000F	00								
	STS	X'01'	....	0010	14								
				0011	01								
	JEQ	BOOT	....	0012	27								
				0013	00								
	ONS	X'01'	....	0014	0C								
				0015	01								
	UA+	LADR	....	0016	CE								
				0017	20								
	INC	RCNT	....	0018	42								
				0019	22								
	JNZ	BOOT+2	....	001A	25								
				001B	02								
	HLTJ	*	....	001C	00								
				001D	1C								
BUFF	DAC	X'0220'	....	001E	02								
				001F	20								
LADR	DAC	X'2F40'	....	0020	** 2F								
				0021	40								
RCNT	DC	X'44'	....	0022	44								

** NOTE: (1) The loader is generally loaded into the last page of available memory although it may be loaded into any page. The starting address of where the loader is to be entered is contained in locations 20 and 21. Therefore,

location 0020 = 0F for a 4K system
= 1F for an 8K system
= 2F for a 12K system, etc.

(2) Locations 0020, 0021, and 0022 must be restored if the BOOTSTRAP is used more than once.

WANG 3300

4-BIT TELETYPE LOADER

TTLOAD

Revised: January 4, 1972

TABLE OF CONTENTS

(Teletype Tape Read Object-Program Loader--4-bit format)

TTLOAD is used with the Teletype reader to load 3300 object program paper tapes which are in 4-bit TTLOAD format. TTLOAD transmits a tape-on and tape-off character to automatically start and stop the tape. The loader reads over leader tape and reads blocks of object program in a scatter load fashion until an end of tape character is encountered. Parity is checked on each data frame. A checksum is calculated and checked with the checksum supplied at the end of each block. Three special control characters are also checked for:

STX - Start of Record $(42)_{16}$

ETX - End of Block $(43)_{16}$

EOT - End of Tape $(44)_{16}$

(See Input Requirements, page 21, which contains a description of OBJECT TAPE FORMAT).

PROGRAM RESTRICTIONS:

(a) Program Size: BC hex bytes; 188 decimal bytes.

(b) Loading Address: xx40; it can be loaded on any page as long as it is begun at in-page byte 40. It is generally loaded into the last page in memory, xF40 (where x = 0 for 4K memory, x = 1 for 8K memory, etc.).

(c) Timing: Limited by TAPE READER SPEED.

(d) Minimum Memory Configuration: 4K.

(e) Peripherals: 33ASR Teletype TBE or any model which can transmit tape-on, tape-off characters.

OPERATING PROCEDURE:

(1) Load TTLOAD into the highest page, xF40 (or any other page), of available memory using TTLOAD BOOTSTRAP (4-bit, page 16).

(2) Start executing at xF40 (i.e., B = xF_{16}, C = 40_{16}).

(3) Depress RUN; depress CLEAR.

(4) Turn the teletype to ON-LINE. Mount an object program tape on the teletype tape reader; set reader switch to STOP.

(5) Press GO.

(6) The tape will be read and loaded.

(7) Repeat steps (4) and (5) for each tape to be loaded.

NORMAL TERMINATION:

After each tape is read: Z-Register = 66_{16}
A-Register = 00_{16}

ERROR TERMINATION:

Several errors can occur during loading which will stop tape motion and cause the program to halt. In each case, the A-Register will contain the last character read and the Z-Register will contain an in-page byte address which can be used to determine the type of error encountered.

(1) Z-Register = $6E_{16}$ Indicates the program was looking for one of the control characters and did not find it. The A-Register contains the character actually read.

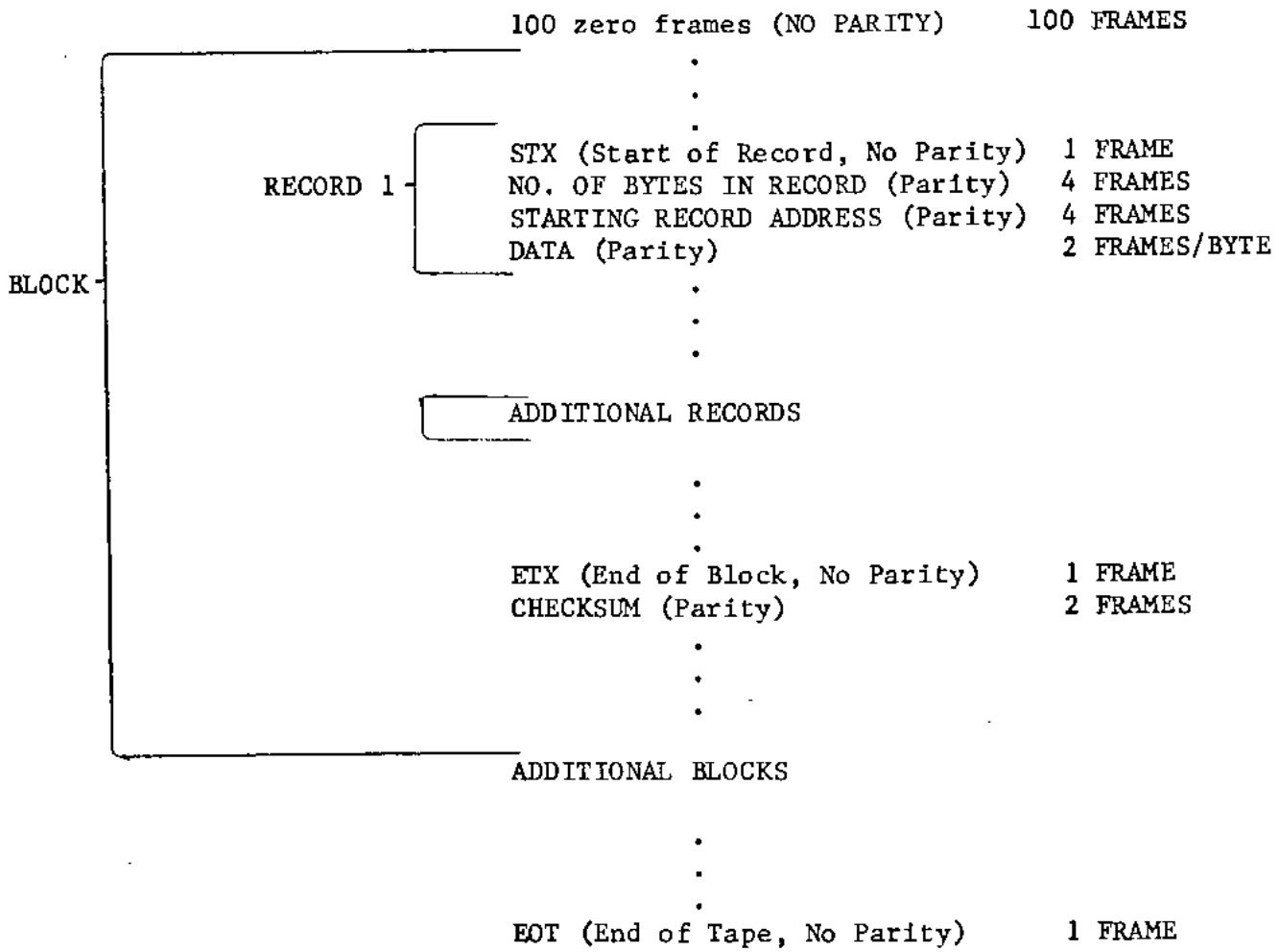
(2) Z-Register = $9C_{16}$ The checksum is not correct. The A-Register contains the checksum read from the tape. The calculated checksum can be found at $xFF6_{16}$.

(3) Z-Register = EE_{16} Incorrect parity; the A-Register contains the character read with the '80' bit on.

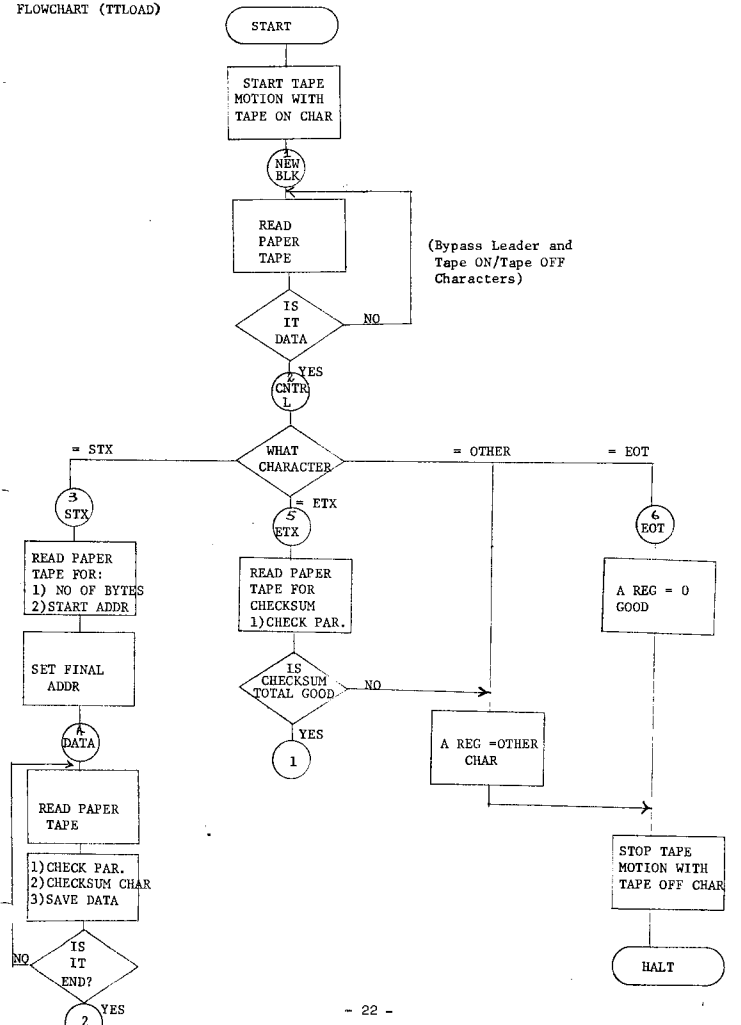
INPUT REQUIREMENTS:

Format of Object Tape (4-bit format):

The object tape is formatted with odd parity as follows:



FLOWCHART (TTLOAD)



WANG 3300

8-BIT TELETYPE LOADER

TTYLOAD

January 4, 1972

WANG 3300 8-BIT TELETYPE LOADER (TTYLOAD)

TABLE OF CONTENTS

TTYLOAD

(Teletype Tape Read Object-Program Loader--8-bit format)

TTYLOAD is used with the Teletype reader to load 3300 object program paper tapes which are in 8-bit TTYLOAD format. TTYLOAD transmits a tape-on and tape-off character to automatically start and stop the tape. The loader reads over leader tape and reads blocks of object program in a scatter load fashion until an end of tape character is encountered. Parity is checked on each data frame. A checksum is calculated and checked with the checksum supplied at the end of each block. Four special control characters are also checked for:

SOH - Start of Tape $(01)_{16}$

STX - Start of Record $(02)_{16}$

ETX - End of Block $(03)_{16}$

EOT - End of Tape $(04)_{16}$

(See Input Requirements, page 27, which contain a description of OBJECT TAPE FORMAT).

PROGRAM RESTRICTIONS:

(a) Program Size: 9E hex bytes; 158 decimal bytes.

(b) Loading Address: xx40; it can be loaded on any page as long as it is begun at in-page byte 40. It is generally loaded into the last page in memory, xF40 (where x = 0 for 4K memory, x = 1 for 8K memory, etc.).

(c) Timing: Limited by TAPE READER SPEED.

(d) Minimum Memory Configuration: 4K.

(e) Peripherals: 33ASR Teletype TBE or any model which can transmit tape-on, tape-off characters.

OPERATING PROCEDURE:

(1) Load TTYLOAD into the highest page, xF40 (or any other page), of available memory using TTYLOAD BOOTSTRAP (8-bit, page 15).

(2) Start executing at xF40 (i.e., B = xF_{16}, C = 40_{16}).

(3) Depress RUN; press CLEAR.

(4) Turn the teletype to ON-LINE. Mount an object program tape on the teletype tape reader; set reader switch to STOP.

(5) Press GO.

(6) The tape will be read and loaded.

(7) Repeat steps (4) and (5) for each tape to be loaded.

NORMAL TERMINATION:

After each tape is read: Z-Register = 64_{16}
A-Register = 00_{16}

ERROR TERMINATION:

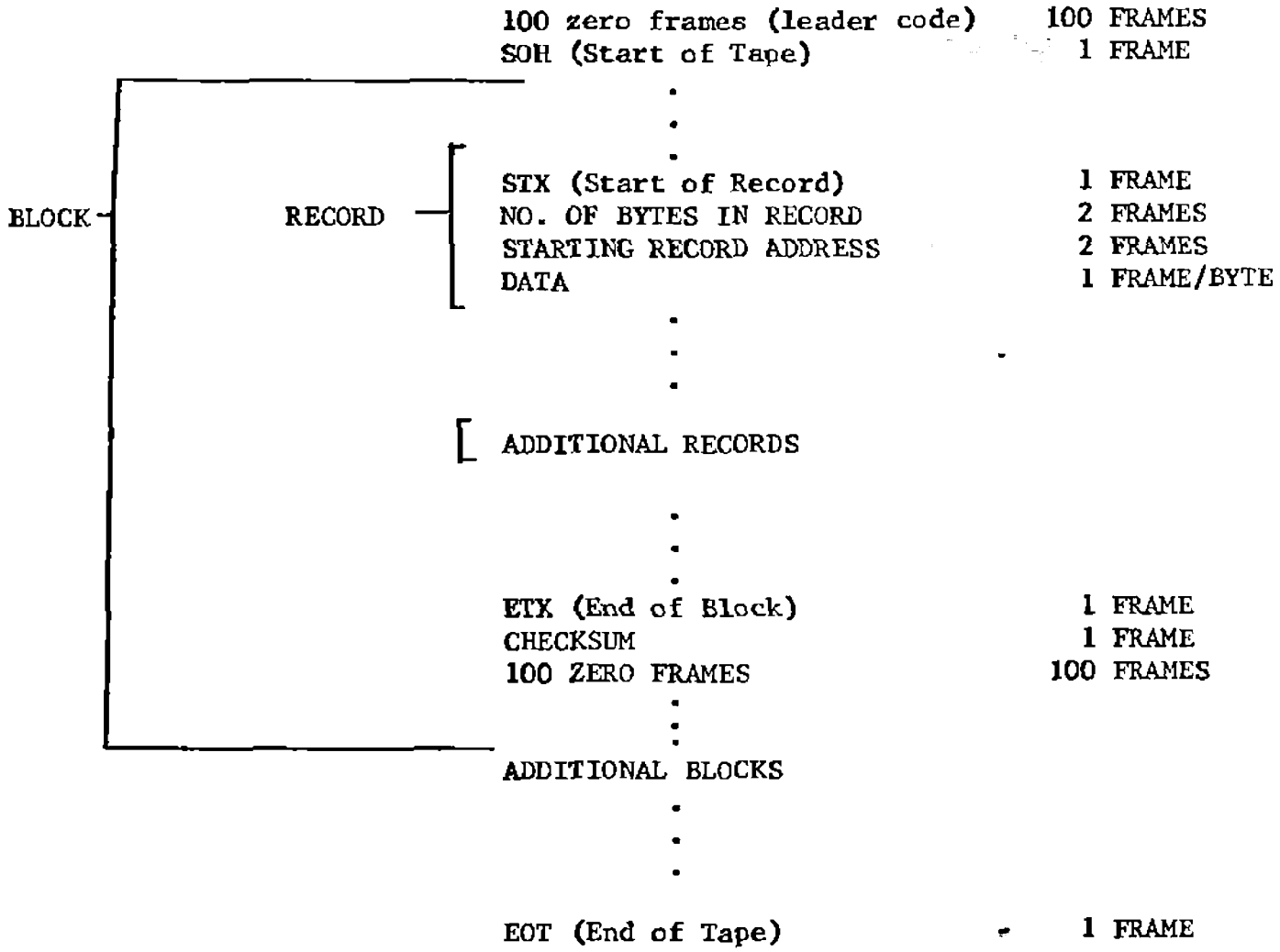
Several errors can occur during loading which will stop tape motion and cause the program to halt. In each case, the A-Register will contain the last character read and the Z-Register will contain an in-page byte address which can be used to determine the type of error encountered.

(1) Z-Register = $6C_{16}$ Indicates the program was looking for one of the control characters and did not find it. The A-Register contains the character actually read.

(2) Z-Register = $9A_{16}$ The checksum is not correct. The A-Register contains the checksum read from the tape. The calculated checksum can be found at $xFF6_{16}$.

INPUT REQUIREMENTS:

Format of Object tape (8-bit format)



WANG 3300

TAPE CASSETTE BOOTSTRAP

TCBOOT #1, TCBOOT #2

July 15, 1972

WANG 3300 TAPE CASSETTE BOOTSTRAP

TABLE OF CONTENTS

TCBOOT

(Tape Cassette Bootstrap Program)

TCBOOT is used with the Wang model 3320 Dual Tape Cassette unit to load the Tape Cassette Loader program (TCLOAD) from cassette tape. Two TCBOOT programs are supplied with each 3320 system; TCBOOT #1 loads TCLOAD from the left-hand port, while TCBOOT #2 loads TCLOAD from the right-hand port. TCBOOT is loaded starting at location $2F40_{16}$ by the Teletype BOOTSTRAP program (TTYBOOT). Each TCBOOT program contains a 1-byte constant at location $2F43_{16}$ specifying the maximum amount of core memory contained in the 3300 system being used. This constant is of the form XE_{16} where X represents the last core module present in the 3300. There are, therefore, fourteen versions each of TCBOOT #1 and TCBOOT #2, ranging from 12K systems through 64K systems.

TCBOOT starts the cassette and reads TCLOAD into the last page of memory as specified by the constant at location $2F43_{16}$. Checksumming is not performed but a system halt occurs if an end of tape, an end of block, or a tape error is detected. When TCLOAD is loaded, TCBOOT performs an automatic jump to the starting location of TCLOAD ($XF00_{16}$).

PROGRAM RESTRICTIONS

1. Program Size: 92_{16} bytes (146_{10} bytes)
2. Loading Address: $2F40_{16}$
3. Minimum Memory Configuration: 12K
4. Peripherals: 33ASR(TBE) Teletype, 3320 Dual Tape Cassette unit.

OPERATING PROCEDURE

1. Load TCBOOT #1 or #2 into location $2F40_{16}$ using TTYBOOT.
2. Place the Software User's system cassette into the corresponding tape cassette port.
3. Set up $2F40_{16}$ in the B and C registers and press RUN, CLEAR, and GO
4. The first two files of the cassette, containing TCLOAD and the tape parameters, will be read into the 3300.

```
DIAG PG AD  HEX   T  C   X D   M  MREF  STMT SOURCE STATEMENT                                              SEQ

                                           2 *     EQUATES
                                  0001     3 @TCRDY EQU  X'01'
                                  0002     4 @TCERR EQU  X'02'
                                  0004     5 @TCEOT EQU  X'04'
                                  0008     6 @TCFPN EQU  X'08'
                                  0010     7 @TCEOB EQU  X'10'
                                  0080     8 @TCBOT EQU  X'80'
                                  0000     9 @MODFY EQU  X'00'
     2F 40                                10        ORG  X'2F40'
     2F 40  0000  00 001 1 01  00         11 TCBOOT OFS  X'FF'
     2F 42  1A7E  00 011 0 10  7E         12        LAI  X'7E'          LAST CORE MODULE IN COMPUTER
     2F 44  CAC2  11 001 0 10  C2 2FC2    13        UA   RESET
     2F 46  CAC4  11 001 0 10  C4 2FC4    14        UA   TCMOD3
     2F 48  CAC6  11 001 0 10  C6 2FC6    15        UA   TCMOD4
     2F 4A  CAC8  11 001 0 10  C8 2FC8    16        UA   TCMOD5
     2F 4C  1801  00 011 0 00  01         17        AI   1
     2F 4E  CACA  11 001 0 10  CA 2FCA    18        UA   TCMOD1
     2F 50  CACC  11 001 0 10  CC 2FCC    19        UA   TCMOD2
     2F 52  CACE  11 001 0 10  CE 2FCE    20        UA   TCLUNT
     2F 54  CAD0  11 001 0 10  D0 2FD0    21        UA   TCLOAD
     2F 56  D2C0  11 010 0 10  C0 2FC0    22        LZ   TCUNIT
     2F 58  1A80  00 011 0 10  80         23        LAI  X'80'
     2F 5A  3080  00 110 0 00  80         24        STI  @TCBOT
     2F 5C  005C  00 000 0 00  5C 2F5C    25 TCBHLT HLTJ *              BAD LOAD
     2F 5E  3060  00 111 1 00  60 2F60    26        CIOJ                START READ
     2F 60  92C2  10 110 0 10  C2 2FC2    27 SETPTR DL   RESET
     2F 62  BA82  10 111 0 10  82 2F82    28        DU   TCBPTR
     2F 64  4586  01 001 1 10  86 2F86    29        JST  TCRD2          READ BYTES 1,2
     2F 66  4586  01 001 1 10  86 2F86    30        JST  TCRD2          READ BYTES 3,4
     2F 68  BA84  10 111 0 10  84 2F84    31        DU   TCBCTR
     2F 6A  4586  01 001 1 10  86 2F86    32        JST  TCRD2          READ BYTES 5,6
     2F 6C  4586  01 001 1 10  86 2F86    33        JST  TCRD2          READ BYTES 7,8
     2F 6E  4586  01 001 1 10  86 2F86    34        JST  TCRD2          READ TEXT PTR
     2F 70  19C8  00 011 0 01  C8         35        CI   X'C8'
     2F 72  255C  00 100 1 01  5C 2F5C    36        JNE  TCBHLT
     2F 74  4586  01 001 1 10  86 2F86    37 TCB1   JST  TCRD2          READ TEXT
     2F 76  BE82  10 111 1 10  82 2F82    38        DU*  TCBPTR
     2F 78  1CFE  00 011 1 00  FF         39.       DLI  -2
     2F 7A  A484  10 101 0 10  84 2F84    40        DAM  TCBCTR
     2F 7C  2574  00 100 1 01  74 2F74    41        JNZ  TCB1
                                          42 *      MEANINGFUL DATA LOADED
     2F 7E  4586  01 001 1 10  86 2F86    43 TCB2   JST  TCRD2          READ OTHER FILLER
     2F 80  467E  01 000 1 10  7E 2F7E    44        JMP  TCB2
                                          45 *
     2F 82  2F82                          46 TCBPTR DAC  *
     2F 84  2F84                          47 TCBCTR DAC  *
                                          48 *
     2F 86  2F86                          49 TCRD2  DAC  *
     2F 88  4592  01 001 1 10  92 2F92    50        JST  TCRD1
     2F 8A  CA8F  11 001 0 10  8F 2F8F    51        UA   *+5
     2F 8C  4592  01 001 1 10  92 2F92    52        JST  TCRD1
     2F 8E  1B00  00 011 0 11  00         53        LZI  @MODFY
     2F 90  4786  01 000 1 11  86 2F86    54        JMP* TCRD2
                                          55 *
     2F 92  2F92                          56 TCRD1  DAC  *
     2F 94  D2C0  11 010 0 10  C0 2FC0    57        LZ   TCUNIT
```

- 31.8 -

```
DIAG  PG AD  HEX   T  C   X D  M  MREF  STMT SOURCE STATEMENT                                    SEQ
      2F 96  31E9  00 110 0 01 E9        58       SFI  @TCEOB+@TCERR+@TCEOT
      2F 98  45A2  01 000 1 10 A2 2FA2    59       JMP  TCREOB
      2F 9A  31FE  00 110 0 01 FE         60       SFI  *TCRDY
      2F 9C  4596  01 000 1 10 96 2F96    61       JMP  *-6
      2F 9E  3AA0  00 111 0 10 A0 2FA0    62       RODJ                 A=BYTE READ
      2F A0  4792  01 000 1 11 92 2F92    63       JMP* TCRD1
                                          64 *
      2F A2  31F9  00 110 0 01 F9    *    65 TCREOB SFI  @TCERR+@TCEOT
      2F A4  00A4  00 000 0 00 A4 2FA4    66       HLTJ *
      2F A6  3284  10 110 0 10 84 2F84    67       DL   TCBCTR
      2F A8  25SC  00 100 1 01 5C 2F5C    68       JNZ  TCBHLT
      2F AA  32C0  10 110 0 10 C0 2FC0    69       DL   TCUNIT
      2F AC  3ACE  10 111 0 11 CE 2FCE    70       DU*  TCLUNT      SET TC UNIT INTO REGULAR LOADER
      2F AE  C2C4  11 000 0 10 C4 2FC4    71       LA   TCMOD3      SET RELOCATABLE ADDRESSES
      2F B0  C4CA  11 001 0 11 CA 2FCA    72       UA*  TCMOD1
      2F B2  C4CC  11 001 0 11 CC 2FCC    73       UA*  TCMOD2
      2F B4  C2CA  11 000 0 10 CA 2FCA    74       LA   TCMOD1
      2F B6  C4C4  11 001 0 11 C4 2FC4    75       UA*  TCMOD3
      2F B8  C4C6  11 001 0 11 C6 2FC6    76       UA*  TCMOD4
      2F BA  C4C8  11 001 0 11 C8 2FC8    77       UA*  TCMOD5
      2F BC  1C02  00 011 1 00 02         78       DLI  2
      2F BE  4700  01 000 1 11 00 2F00    79       JMP* TCLOAD
      2F C0  40C1                         80 TCUNIT DC   X'40C1'      TAPE CASSETTE UNIT
                                          81 ***    THE FOLLOWING DATA MUST DIFFER WITH MAXIMUM MEMORY SIZE
      2F C2  7EC8                         82 RESET  DC   X'7EC8'
      2F C4  7EF6                         83 TCMOD3 DC   X'7EF6'
      2F C6  7EF8                         84 TCMOD4 DC   X'7EF8'
      2F C8  7EFA                         85 TCMOD5 DC   X'7EFA'
      2F CA  7FF6                         86 TCMOD1 DC   X'7FF6'
      2F CC  7FF8                         87 TCMOD2 DC   X'7FF8'
      2F CE  7FFE                         88 TCLUNT DC   X'7FFE'
      2F D0  7FC0                         89 TCLOAD DC   X'7F00'
      2F D2                        C000   90        END    END CARD GENERATED BY ASSEMBLER
```

DEFN	SYMBOL	VALUE	REFERENCES
0009	*MODFY	0000	0053
0008	@TCBOT	0080	0024
0007	@TCECH	0010	0058
0005	@TCEOT	0004	0058 0065
0004	@TCERR	0002	0058 0065
0006	@TCFRN	0008	
0003	@TCRDY	0001	0060
0082	RESET	2FC2	0013 0027
0027	SETPTR	2F69	
0047	TCBCTR	2F84	0031 0040 0067
0025	TCBHLT	2F5C	0036 0068
0011	TCBOUT	2F40	
0045	TCBPTR	2F42	0028 0038
0037	TCB1	2F74	0041
0043	TCB2	2F7E	0044
0045	TCLOAD	2FD0	0021 0079
0055	TCLUNT	2FCE	0020 0070
0086	TCMOD1	2FCA	0018 0072 0074
0087	TCMOD2	2FCC	0019 0073
0043	TCMOD3	2FC4	0014 0071 0075
0044	TCMOD4	2FC6	0015 0076
0045	TCMOD5	2FC8	0016 0077
0056	TCRD1	2F92	0050 0052 0063
0049	TCRD2	2F86	0029 0030 0032 0033 0034 0037 0043 0054
0065	TCREOM	2FA2	0059
0040	TCUNIT	2FC0	0022 0057 0069

NORMAL TERMINATION

If the Software User's System cassette loads correctly, the system will halt, displaying an 03_{16} in the A register and an $XF00_{16}$ in the B and C registers.

INPUT REQUIREMENTS

The only valid input to TCBOOT is the 3300 Tape Cassette Loader program (TCLOAD). The loader must be the first file on a cassette and the tape must be rewound before a correct load is possible.

WANG 3300

TAPE CASSETTE LOADER

TCLOAD

July 15, 1972

WANG 3300 TAPE CASSETTE LOADER (TCLOAD)

TABLE OF CONTENTS

TCLOAD

(Cassette Tape Object Program Loader)

TCLOAD is used with the Wang model 3320 Dual Tape Cassette unit to load 3300 object program files. TCLOAD automatically starts and stops the 3320 cassette drive. The loader program reads blocks of 256 bytes from cassette, and calculates and verifies the checksum for each block. Start of record byte, file number and record number are also verified. Input tapes for TCLOAD must conform to the input format specifications described in the section on INPUT REQUIREMENTS FOR TCLOAD.
To facilitate the loading of object program files, TCLOAD utilizes a method of automatic read retry. If any input block is not read correctly, TCLOAD reads the second copy of the block. If this block fails, TCLOAD makes three more attempts to achieve a successful read of either block, before a system halt occurs.

PROGRAM RESTRICTIONS

a. Program Size: 134_{16} bytes (308_{10} bytes)

b. Loading Address: $XYC8_{16}$ through $(XY+1)FF_{16}$
(XY is the contents of location $2F43_{16}$ in the TCBOOT program. For example, if location $2F43_{16}$ = 7E, the TCLOAD program will be loaded in locations $7EC8_{16}$ through $7FFF_{16}$).

c. Execution Address: $(XY+1)00_{16}$ (i.e. if location 2F43 = 7E, the execution address would be $7F00_{16}$)

d. Minimum System Configuration: 4K

e. Peripherals: 33ASR(TBE) Teletype, Wang 3320 Dual Tape Cassette Unit.

OPERATING PROCEDURE

1. Load the appropriate version of TCBOOT. Location $2F43_{16}$ will be set to XE_{16}; this address can be altered as desired.

2. Mount the Wang Software User's System cassette on the 3320 cassette unit, and execute the TCBOOT program. The system will halt with an 03_{16} in the A register and the starting address of the loader $(XY+1)00_{16}$ in the B and C registers.

3. To use TCLOAD, place the file number of the desired input file into the A register, and press RUN, CLEAR, and GO. (See part 3 of the OBJECT TAPE LOADING PROCEDURE)

NORMAL TERMINATION

If a good load has occurred, the 3300 will halt with the file number of the next sequential file displayed in the A register and the starting location ($(XY+1)00_{16}$) of the loader in the B and C registers.

ERROR TERMINATION

There are two error halts built into the TCLOAD program:

1. REREAD FAILED. If TCLOAD fails to successfully read either copy of a particular input block after four attempts, the system will halt at location $(XY+1)C8_{16}$.

2. FILE NOT FOUND. If the requested file is not found, the system will halt at location $(XY+1)CA_{16}$; the A register will contain the file number of the last file read.

INPUT REQUIREMENTS FOR TCLOAD

The format of the physical record block for object programs which can be read by TCLOAD is as follows:

1. The physical block is fixed at 256 bytes, defined by the statement

```
BLOCK    DC    X'0100'    256 BYTES
```

2. Control information occurs at the front of each physical block.

   a. byte #1 Tape file data BOF/EOF as follows
      X'AA' = beginning of record
      X'F0' = beginning of file
      X'0F' = end of file
      X'FF' = self contained file (i.e., file contained within a single physical block)

   b. byte #2 Check sum byte

   c. byte #3,4 Record length (meaningful text)

d. byte #5 Control byte check sum

e. byte #6 File number

f. byte #7,8 Record number (within file)

g. byte #9,10 Record text pointers

h. byte #11,N Record text characters

In reading a tape file, a check of byte #1 permits the loader to process the data according to the following rules:

Byte #1 = X'F0' Beginning of File record.

a. File number must be 1 greater than prior record.
b. Record number must be equal to 0001.
c. Text contains text information of initial data in file.

Byte #1 = X'AA' Beginning of Record.

a. File number must be identical to prior record.
b. Record number must be 1 greater than prior record.
c. Text contains text information.

Byte #1 = X'0F' End of File record.

a. File number must be identical to prior record.
b. Record number must be 1 greater than prior record.
c. Text contains text information of last data in file.

Byte #1 = X'FF' Beginning/End of File record.

a. File number must be 1 greater than prior record.
b. Record number must be 0001
c. Text data for file totally within current record.

3. The TCLOAD program expects redundant recording of each physical block on the cassette tape. To further insure the integrity of all data output to tape, TCLOAD performs the following functions:

a. Check sum control of all data contained within each physical block,
b. Verification of the start of record byte.
c. Verification of file number and record number according to the rules established for each start of record byte.

```
DIAG PG AD  HEX   T C  X D  M  MREF  STMT SOURCE STATEMENT                                    5E1
                                        2 AA      ORG   X'6F00'
6F 00                                   3         DC    C'TCLOAD AUTO-JUMP'
6F 00  5443404F41442041
       55544F2040554D50
6F 10  3030303333732                    4         DC    C'00-0372'
04 00                                   5 CPARGN  ORG   X'0460'
6F 00                                   6         ORG   AA
                                        7 *       TAPE CASSETTE LOAD PROGRAM
                                        8 *       TAPE FORMAT
                                        9 *       BYTE 1          START OF RECORD BYTE
                                       10 *       BYTE 2          CHECK-SUM BYTE
                                       11 *       BYTE 3+4        NUMBER OF TEXT BYTES
                                       12 *       BYTE 5          CONTROL BYTE CHECK-SUM
                                       13 *       BYTE 6          FILE NUMBER
                                       14 *       BYTE 7,8        RECORD NUMBER
                                       15 *       BYTE 9,10       STARTING CORE MEMORY ADDRESS
                                       16 *       BYTE 11-N       DATA BYTES
                                       17 *
                                       18 **      STATUS EQUATES
                                0001   19 *TCRDY  EQU   X'01'
                                0002   20 *TCERR  EQU   X'02'
                                0004   21 *TCEOT  EQU   X'04'
                                0008   22 *TCFPN  EQU   X'08'
                                0010   23 *TCEOB  EQU   X'10'
                                0080   24 *TCHOT  EQU   X'80'
                                0000   25 *MODFY  EQU   X'00'
                                0020   26 *C      EQU   X'20'       CARRY BIT
                                       27 **
                                       28 *       LOADER LOADS ALL RECORDS WITHIN FILE SPECIFIED
                                       29 *       A=FILE WANTED
                                       30 *       END OF PAGE=TPIN*TPOUT  (COMPILED AS TAPE DRIVE 1)
                                       31 **      LOAD=GOOD
                                       32 *       HALT B*C=TCLOAD  Z=TAPE  A=NEXT FILE IN SEQUENCE
                                       33 **      LOAD=BAD
                                       34 *        HALT 7  TAPE ERROR
                                       35 *       HALT 8  REREAD FAILED
                                       36 *       HALT 9  FILE NOT FOUND
```

SFQ

```
DIAG  PG AD  HEX   T  C   X D   M  MREF  STMT SOURCE STATEMENT
                                          38 *     USES TAPE CASSETTE 1 (ADDR 40-C1)  A=FILE WANTED
6F 00  0000  00 001 1 01  00              39 TCLOAD OFS  X'FF'
6F 02  CAAF  11 001 0 10  AF 6FAF         40        UA   TCLBA+1     SAVE FILE WANTED
6F 04  1AFF  00 011 0 10  FF              41        LAI  X'FF'
6F 06  CA87  11 001 0 10  87 6F87         42        UA   TCBLK+1     SET TCBLK=FF
6F 08  1400  00 011 0 10  00              43        LAI  0
6F 0A  CA4B  11 001 0 10  4B 6F4B         44        UA   TCLB+1      CLEAR RECORD WANTED INDICATOR
6F 0C  1AFF  00 011 0 10  FF              45        LAI  X'FF'
6F 0E  CA87  11 001 0 10  87 6F87         46        UA   TCBLK+1     TCBLK=FF FOR BLOCK 1
6F 10  C2E8  11 000 0 10  E8 6FE8         47 TCL0   LA   TCRTRY+1
6F 12  CAEA  11 001 0 10  EA 6FEA         48        UA   TCRTRY      SET RETRY COUNTER
6F 14  1CFF  00 011 1 00  FF              49 TCL1   DLI  X'FF'
6F 16  BAEE  10 111 0 10  EE 6FEE         50        DU   TCCT34      BYTE COUNT=MAX
6F 18  02FE  11 010 0 10  FE 6FFE         51        LZ   TPIN
6F 1A  1A80  00 011 0 10  80              52        LAI  X'80'
6F 1C  3C1E  00 111 1 00  1E 6F1E         53        CIOJ              START READ FUNCTION
6F 1E  4FF6  01 001 1 11  F6 6FF6         54        JST* $TCRD2
6F 20  BAEC  10 111 0 10  EC 6FEC         55        DU   TCCT12      SAVE BYTES 1,2  START CHECK-SUM
6F 22  DA35  11 011 0 10  35 6F35         56        UZ   TCLA+1
6F 24  4FF8  01 001 1 11  F8 6FF8         57        JST* $TCRCS
6F 26  BAEE  10 111 0 10  EE 6FEE         58        DU   TCCT34      SAVE BYTES 3,4  NUMBER OF BYTES
6F 28  4FF6  01 001 1 11  F6 6FF6         59        JST* $TCRD2
6F 2A  BAF0  10 111 0 10  F0 6FF0         60        DU   TCCT56      SAVE BYTES 5,6
6F 2C  0A0F  00 001 1 01  0F              61        OFS  @C
6F 2E  BAEC  10 011 0 10  EC 6FEC         62        AMC  TCCT12      CHECK-SUM FILE NUMBER
6F 30  4FF8  01 001 1 11  F8 6FF8         63        JST* $TCRCS
6F 32  BAF2  10 111 0 10  F2 6FF2         64        DU   TCCT78      SAVE BYTES 7,8 RECORD NUMBER
6F 34  1400  00 011 0 10  00              65 TCLA   LAI  @MODFY      ADDR=1ST BYTE READ
6F 36  1AAA  00 011 0 01  AA              66        CI   X'AA'
6F 38  274A  00 100 1 11  4A 6F4A         67        JEQ  TCLB        'AA'
6F 3A  1A0F  00 011 0 01  0F              68        CI   X'0F'
6F 3C  274A  00 100 1 11  4A 6F4A         69        JEQ  TCLB        EOF
6F 3E  1AFF  00 011 0 01  FF              70        CI   X'FF'
6F 40  2772  00 100 1 11  72 6F72         71        JEQ  TCL5        BEF
6F 42  1AF0  00 011 0 01  F0              72        CI   X'F0'
6F 44  2772  00 100 1 11  72 6F72         73        JEQ  TCL5        BOF
                                          74 *     IGNORE THE REST OF THIS RECORD
6F 46  4FF6  01 001 1 11  F6 6FF6         75 TCL2   JST* $TCRD2      IGNORE THE REST
6F 48  6546  01 000 1 10  46 6F46         76        JMP  TCL2
                                          77 *
6F 4A  1400  00 011 0 10  00              78 TCLB   LAI  @MODFY      ADDR=WANTED INDICATOR
6F 4C  2746  00 100 1 11  46 6F46         79        JZ   TCL2        NOT WANTED
```

```
DIAG  PG AD  HEX   T  C   X D  M  MREF  STMT SOURCE STATEMENT                                    SEQ
                                          61 *--------------------READ TEXT POINTER INTO COPE----------
      6F 4E  4FF8  01 001 1 11 F8 6FF8   62        JST* $TCRCS
      6F 50  BAF4  10 111 0 10 F4 6FF4   83        DU   TCTPTR
      6F 52  C2EC  11 000 0 10 EC 6FEC   84 ---   LA   TCCT12------ TEST-----------------------------
      6F 54  92F0  10 010 0 10 F0 6FF0   85        C    TCCT56        CONTROL BYTE CHECK-SUM
      6F 56  2546  00 100 1 01 46 6F46   86        JNE  TCL2          *IF IT FAILS IGNORE THE REST OF THE RECORD
                                          87 *--------------------READ DATA INTO TEXT AREA-------------
      6F 58  4FF8  01 001 1 11 F8 6FF8   88 TCL3   JST* $TCRCS
      6F 5A  DEF4  11 011 1 10 F4 6FF4   89        UZ*  TCTPTR        SAVE 1ST BYTE
      6F 5C  C465  11 001 0 10 65 6F65   90 ------  UA   TCL3A+1------ SAVE 2ND BYTE-----------------
      6F 5E  1CFF  00 011 1 00 FF        91        OLI  -1
      6F 60  AAEF  10 101 0 10 EE 6FEE   92        DAM  TCCT34
      6F 62  275E  00 100 1 11 6E 6F6E   93 ------ JZ   TCL4-----------------------------------------
      6F 64  1A00  00 011 0 10 00        94 TCL3A  LAI  @MODFY        ADDR=ODD BYTE
      6F 66  DFF4  11 001 1 10 F4 6FF4   95        UA*  TCTPTR        SAVE ODD BYTE
      6F 68  1CFF  00 011 1 00 FF        96 ------  OLI  -1
      6F 6A  AAEE  10 101 0 10 EE 6FEE   97        DAM  TCCT34
      6F 6C  2558  00 100 1 01 58 6F58   98        JNZ  TCL3          LOOP FOR MORE DATA
      6F 6E  4FF8  01 001 1 11 F8 6FF8   99 TCL4-- JST* $TCRCS-------READ/CHECK-SUM FILLER BYTES-------
      6F 70  466E  01 000 1 10 6E 6F6E  100        JMP  TCL4
                                         101 *
                                         102 *  ---- BEGINNING OF FILE ----------------------------
      6F 72  C2F1  11 000 0 10 F1 6FF1  103 TCL5   LA   TCCT56+1      PICK-UP FILE READ AND THEN TEST VS.
      6F 74  92AF  10 010 0 10 AF 6FAF  104        C    TCL8A+1        FILE WANTED
      6F 76  2546  00 100 1 10 46 6F46  105 ------ JLT  TCL2 --------IGNORE LOWER NUMBERED FILES-------
      6F 78  24CA  00 100 1 00 CA 6FCA  106        JGT  TCLE2         ERROR HIGHER NUMBERED FILES
      6F 7A  42A8  01 000 0 10 A8 6FA8  107        INC  TCL8+1        SET RECORD WANTED INDICATOR
      6F 7C  466A  01 000 1 10 4A 6F4A  108 ------ JMP  TCL8----------------------------------------
                                         109 *
      6F 7E  31F9  00 110 0 01 F9       110 TCREOB SFI  @TCERR+@TCEOT
      6F 80  34C8  00 111 0 00 C8 6FC8  111 TCL7-- TIAJ TCLE1 ------A=I/O STATUS-------------------
      6F 82  C248  11 000 0 10 48 6F48  112        LA   TCL8+1        RECORD WANTED?
      6F 84  2710  00 100 1 11 10 6F10  113        JZ   TCL0           NO*READ ANOTHER
      6F 86  1CFF  00 011 0 10 FF       114 TCBLK  LAI  X'FF'
      6F 88  2732  00 100 1 11 82 6F82  115        JZ   TCL9
      6F 8A  42A7  01 000 0 10 A7 6FA7  116        INC  TCBLK+1       SET BLK=00
      6F 8C  C2EC  11 000 0 10 EC 6FEC  117 ------ LA   TCCT12------PROCESS BLOCK 1----------------
      6F 8E  92E0  10 010 0 10 E0 6FE0  118        C    TCCT12+1       VERIFY CHECK-SUM
      6F 90  2514  00 100 1 01 14 6F14  119        JNE  TCL1          *ERROR
      6F 92  1A00  00 011 0 10 C0       120        LAI  X'C0'  ------ OK
      6F 94  02FE  11 010 0 10 FE 6FFE  121        LZ   TPIN
      6F 96  3098  00 111 1 00 98 6F98  122        CIOJ               SPACE A RECORD
      6F 98  3810  00 111 0 00 10       123 ------ STI  @TCEOB--------------------------------------
      6F 9A  2498  01 000 1 10 98 6F98  124        JMP  *-2
      6F 9C  1CFF  00 011 0 10 FF       125 TCL8   LAI  X'FF'
      6F 9E  C487  11 001 0 10 87 6F87  126 ------ UA   TCBLK+1------ SET TCBLK=FF-------------------
      6F A0  C435  11 000 0 10 35 6F35  127        LA   TCLA+1
      6F A2  1A00  00 011 0 01 F0       128        CI   X'F0'
      6F A4  2710  00 100 1 11 10 6F10  129 ------ JEQ  TCL0 --------BOF-----------------------------
      6F A6  1AAA  00 011 0 01 AA       130        CI   X'AA'
      6F A8  2710  00 100 1 11 10 6F10  131        JEQ  TCL0          TEXT READ ANOTHER
      6F AA  4FCC  01 001 1 10 CC 6FCC  132 ------ JST  TCHALT-------------------------------------
      6F AC  42AF  01 000 0 10 AF 6FAF  133        INC  TCL8A+1       BUMP FILE WANTED TO SHOW GOOD LOAD
      6F AE  1CC0  00 011 1 00 00       134 TCL8A  DLI  @MODFY        ADDR=FILE WANTED
      6F B0  47FA  01 000 1 11 FA 6FFA  135        JMP* TCLJST    ****TCLOAD AUTOMATED EXIT/OR HLTJ TCLOAD
                                         136 *      LST BLOCKS IN ERROR
```

```
                                                                                                        SEQ
DIAG  PG AD  HEX    T  C    X D    M  MREF   STMT SOURCE STATEMENT
      6F B2  C2EC   11 000 0 10   EC 6FEC    137  TCL9   LA    TCCT12
      6F B4  92ED   10 010 0 10   ED 6FED    138         C     TCCT12+1
      6F B6  279C   00 100 1 11   9C 6F9C    139         JEQ   TCL8        2ND BLOCK O.K.
      6F B8  4ECC   01 001 1 10   CC 6FCC    140         JST   TCHALT
      6F BA  4EDC   01 001 1 10   DC 6FDC    141         JST   TCBACK
      6F BC  4EDC   01 001 1 10   DC 6FDC    142         JST   TCBACK
      6F BE  4ECC   01 001 1 10   CC 6FCC    143         JST   TCHALT
      6F C0  1AFF   00 011 0 10   FF         144         LAI   X'FF'
      6F C2  C387   11 001 0 10   87 6F87    145         OA    TCBLK+1     SET TCBLK=FF
      6F C4  42EA   01 000 0 10   EA 6FEA    146         INC   TCRTRY      BUMP RETRY COUNTER
      6F C6  2514   00 100 1 01   14 6F14    147         JNZ   TCL1        TRY READ AGAIN
      6F C8  00C8   00 000 0 00   C8 6FC8    148  TCLE1  HLTJ  *           BAD LOAD  REREAD FAILED
      6F CA  00CA   00 000 0 00   CA 6FCA    149  TCLE2  HLTJ  *           BAD LOAD  FILE NOT FOUND A=LAST FILE READ
                                             150  *
```

```
DIAG  PG AD  HEX   T  C   X D  M  MREF  STMT SOURCE STATEMENT                SEQ
      6F CC  6FCC                          152 TCHALT DAC  *
      6F CE  D2FE  11 010 0 10  FE 6FFE    153         LZ   TPIN
      6F D0  1A00  00 011 0 10  00         154         LAI  X'00'
      6F D2  3C04  00 111 1 00  D4 6FD4    155         CIOJ
      6F D4  D2FF  11 010 0 10  FF 6FFF    156         LZ   TPOUT
      6F D6  31FE  00 110 0 01  FE         157         SFI  @TCRDY
      6F D8  46D6  01 000 1 10  D6 6FD6    158         JMP  *-2
      6F DA  47CC  01 000 1 11  CC 6FCC    159         JMP* TCHALT
                                           160 **
      6F DC  6FDC                          161 TCBACK DAC  *
      6F DE  1AE0  00 011 0 10  E0         162         LAI  X'E0'
      6F E0  D2FE  11 010 0 10  FE 6FFE    163         LZ   TPIN
      6F E2  3CE4  00 111 1 00  E4 6FE4    164         CIOJ
      6F E4  3910  00 110 0 00  10         165         STI  @TCE08
      6F E6  46E4  01 000 1 10  E4 6FE4    166         JMP  *-2
      6F E8  47DC  01 000 1 11  DC 6FDC    167         JMP* TCBACK
                                           168 **
      6F EA  FCFC                          169 TCRTRY DC   X'FCFC'      RETRY COUNTER
      6F EC  0000                          170 TCCT12 DC   X'0000'      CHECK-SUM
      6F EE  0000                          171 TCCT34 DC   X'0000'      NUMBER BYTES
      6F F0  0000                          172 TCCT56 DC   X'0000'      FILE NUMBER
      6F F2  0000                          173 TCCT7A DC   X'0000'      RECORD NUMBER
      6F F4  0000                          174 TCTPTR DC   X'0000'      TEXT PTR  FROM BYTES 9,10
      6F F6  6EC8                          175 $TCRD2 DAC  TCRD2                    *********
      6F F8  6EE4                          176 $TCRCS DAC  TCRCS                    *********
                                           177 **
      6F FA  0400                          178 TCLJST DAC  CPARGN       TCLOAD AUTOMATED EXIT/OR HLTJ TCLOAD
      6F FC  4B00  01 000 1 10  00 6F00    179 TCLJMP JMP  TCLOAD
      6F FE                                180         ORG  AA+254
      6F FE  40                            181 TPIN    DC   X'40'
      6F FF  C1                            182 TPOUT   DC   X'C1'
      6E C5                                183         ORG  AA-X'38'
      6E C8  6EC8                          184 TCRD2   DAC  *           READ 2 BYTES
      6E CA  4ED4  01 001 1 10  D4 6ED4    185         JST  TCRD1
      6E CC  CAD1  11 001 0 10  D1 6ED1    186         DA   *+5
      6E CE  4ED4  01 001 1 10  D4 6ED4    187         JST  TCRD1
      6E D0  1B00  00 011 0 11  00         188         LZI  @MODFY      Z=EVEN BYTE  A=ODD BYTE
      6E D2  47C8  01 000 1 11  C8 6EC8    189         JMP* TCRD2
                                           190 **
      6E D4  6ED4                          191 TCRD1   DAC  *           READ 1 BYTE
      6E D6  D3F8  11 010 0 11  F8 6EF8    192         LZ*  ATPIN
      6E D8  31E9  00 110 0 01  E9         193         SFI  @TCE0B+@TCERR+@TCEOT    1) IDLE
      6E DA  47F6  01 000 1 11  F6 6EF6    194         JMP* KREOB                   2) LOOP
      6E DC  31FF  00 110 0 01  FF         195         SFI  @TCRDY                  3)   FOR
      6E DE  46D6  01 000 1 10  D6 6ED6    196         JMP  *-6                     4)    READ
      6E E0  D3F2  00 111 0 10  F2 6EF2    197         RDDJ         A=BYTE READ
      6E E2  47D4  01 000 1 11  D4 6ED4    198         JMP* TCRD1
                                           199 **
      6E E4  6FE4                          200 TCRCS   DAC  *           READ 2 CHARACTERS**ACCUMULATE CHECK-SUM
      6E E6  4EC8  01 001 1 10  C8 6EC8    201         JST  TCRD2
      6E E8  C5FA  00 000 1 01  EA 6EEA    202         XZAJ
      6E EA  B00F  00 001 1 01  0F         203         OFS  @C
      6E EC  96FA  10 011 0 11  FA 6EFA    204         AMC* ACCT12
      6E EE  D5F0  00 000 1 01  F0 6EF0    205         XZAJ
      6E F0  B00F  00 001 1 01  0F         206         OFS  @C
      6E F2  96FA  10 011 0 11  FA 6EFA    207         AMC* ACCT12
```

```
DIAG  PG AD  HEX   T  C   X D   M  MREF  STMT SOURCE STATEMENT                                SEQ
      AE F4  47E4  01 000 1 11  E4 6EE4   208          JMP* TCRCS                          ********
      AE F6  6F7E                         209 KRF0H    DAC  TCKE0B                        ********
      AE F8  4FFE                         210 ATPIN    DAC  TPIN                          ********
      AE FA  6FEC                         211 ACCT12   DAC  TCCT12
      AE FC                         0000  212          END      END CARD GENERATED BY ASSEMBLER
```

CROSS REFERENCE LISTING

DEFN	SYMBOL	VALUE	REFERENCES
0176	$TCRCS	6FF8	0057 0063 0082 0088 0099
0175	$TCRD2	6FF6	0054 0059 0075
0026	BC	0020	0061 0203 0206
0025	BMODFY	0000	0065 0078 0094 0134 0188
0024	$TC40T	0040	
0023	$TCE04	0010	0123 0165 0193
0021	$TCE0T	0004	0110 0193
0027	$TE249	0002	0110 0193
0022	$TCFPN	0005	
0019	$TCRDY	0001	0157 0195
0032	AA	6F00	0006 0180 0183
0211	ACCT12	6FFA	0204 0207
0210	APPIN	6FF8	0192
0075	CHANGE	0080	0178
0209	CHECK	6FF6	0154
0181	TCHECK	6FDC	0141 0142 0167
0114	TCHLK	6FA6	0042 0046 0116 0126 0145
0170	TCCT12	6FEC	0055 0062 0084 0117 0118 0137 0138 0211
0171	TCCT34	6FEE	0050 0058 0092 0097
0172	TCCT56	6FF0	0060 0085 0103
0173	TCCT78	6FF2	0064
0182	TCHELT	6FCC	0132 0140 0143 0159
0045	TCLA	6F34	0056 0127
0078	TCLB	6F4A	0044 0067 0069 0107 0108 0112
0145	TCLC1	6FC4	0111
0149	TCLE2	6FDA	0106
0179	TCLJMP	6FFC	
0178	TCLJST	6FFA	0135
0037	TCLO40	6F60	0179
0047	TCLO	6F10	0113 0129 0131
0048	TCL1	6F14	0119 0147
0075	TCL2	6F46	0076 0079 0086 0105
0048	TCL3	6F5A	0098
0084	TCL3A	6F64	0040
0094	TCL4	6F6E	0093 0100
0103	TCL5	6F72	0071 0073
0111	TCL7	6F80	
0124	TCL8	6F90	0139
0131	TCL83	6F92	0040 0104 0133
0147	TCL9	6F96	0115
0230	TCWCE	6FE4	0176 0208
0191	TCH01	6F04	0185 0187 0198
0196	TCH02	6F08	0175 0189 0201
0210	TCH03	6F7E	0234
0144	TCH04	6F14	0047 0048 0146
0217	TCHTH	6FF4	0063 0069 0095
0181	TPIN	6FFE	0051 0121 0153 0163 0210
0149	TPOUT	6FFF	0156

4. The contents of the record text area (bytes 11 - N) is not of direct concern to TCLOAD except for the number of text bytes and check sum verification.

WANG 3300

SYMBOLIC TAPE EDITOR

(TELETYPE AND CASSETTE VERSION)

USER'S MANUAL

(TTYEDIT, TCEDIT)

Revised July 15, 1972

TABLE OF CONTENTS

TABLE OF CONTENTS

ABSTRACT

The Symbolic Editor (TTYEDIT) is a 3300 Utility Program that is used to generate or correct paper tape containing 3300 Source Assembly language programs or other teletype line files. It is used with the 3315 teletype terminal for both reading and punching source tapes and keyboard/printing operations.

Two versions of TTYEDIT exist. One is a 4K, 4-bit version (the tape is in 4-bit format and must be loaded with a 4-bit loader) for use with the 4K, 4-bit Assembler. The other is an 8K, 8-bit version (the tape is in 8-bit format and must be loaded with an 8-bit loader) for use with the 8K, 8-bit Assembler.

A similar Symbolic Editor (TCEDIT) which utilizes the 3320 cassette tapes to read or write source language program files is also available. TCEDIT requires a minimum 12K system and is in 8-bit format only. This version of the Symbolic Editor is loaded via the appropriate version of TCBOOT#1 or TCBOOT#2.

I. OPERATION OF TTYEDIT AND OF TCEDIT

A. INTRODUCTION

TTYEDIT is one of the standard utility programs for the WANG 3300 digital computer system. It is designed to help the user to prepare and edit line-file paper tapes using the 3315 Teletype terminal. TTYEDIT contains a number of features which simplify the process of listing, modifying, adding to, and copying source program lines. The tape cassette version (TCEDIT) of the Symbolic Editor further simplifies these functions by greatly speeding up reading and writing of files of source program lines.

While TTYEDIT and TCEDIT have been specifically designed for the preparation of assembly source language program tapes for later execution by the WANG 3300 assembly program (see the ASSEMBLER User's Manual), they may be used with any files consisting of keyboard entered information-- such as BASIC data files.

B. LINE AND RECORD FORMATS

The basic unit with which the Symbolic Editor user works is the teletype source line. This is defined to be any string of ASCII teletype characters terminated by a carriage return and line feed, which marks the record end on tape. For tapes read by either TTYEDIT or by TCEDIT, a line with no characters except a carriage return and line feed will be considered a null line; when read, it will be deleted and ignored. Similarly, single carriage return or line feed characters at the beginning of a line will also be deleted.

A teletype line contains from 1 to 72 characters, lines may vary in length, but they may not exceed 72 characters. Since one space is required for the colon output and one for the carriage return, the user may effectively add a 70 character line to workspace at one time.

Teletype source lines are separated by 2 rubout characters when written onto paper tape; rubout characters in source lines are ignored when the tape is read.

C. WORKSPACE

"WORKSPACE: is defined to be an area within the computer which is 1,878 * characters in size and consists of a variable number of lines. It is within this workspace that the actual editing

*5,974 characters for the 8K version of TTYEDIT.

(insertion, deletion, movement) and listing occurs. There are a set of commands which control the reading of records from the old source file tape and the writing of records onto the new source file tape; another set of commands allow the examination and changing of the status of the workspace.

TTYEDIT (TCEDIT) rules allow the user to write lines from the workspace onto the new file tape or to read lines into the first 1,476 ** characters of the workspace. Note that the workspace cannot be completely filled by reading; this is done to guarantee that there will always be some slack space in which the user may make additions to the present contents of the workspace. In reading and writing into and from the workspace, the original sequence of the files is not disturbed; interchange of source line sequence can only be accomplished by use of the move command (/M).

D. TYPING RULES

All TTYEDIT (TCEDIT) commands and additions to the source file are entered via the teletype keyboard; all editing of output is written onto the keyboard, providing a record of the entire editing process. When TTYEDIT (TCEDIT) wishes to obtain the next user teletype input, it outputs a colon (:) to the keyboard and waits for a user response.

A user response (entry) is always terminated by a carriage return, which causes TTYEDIT (TCEDIT) to ignore further keyboard input and service the entry. When the entry has been serviced, TTYEDIT (TCEDIT) issues another colon and the user may make another entry. If a user command causes TTYEDIT (TCEDIT) to output onto the teletype, the system places a space in column 1; thus, a record of the editing session always shows user inputs (lines starting with a colon) and TTYEDIT (TCEDIT) responses (lines without a colon).

In general, once a valid entry has been make, it cannot be revoked by the user. The exceptions to this are the listing command and the status command. During these commands, the user may interrupt by striking the ESCape key; this will terminate the listing in progress and request the next user input. TCEDIT allows the ESC key to be struck at any time.

There are two classes of typewriter input to TTYEDIT (TCEDIT): functional commands always consisting of a slash (/) followed by a single letter and, optionally, other special fields (see Part II, Page 52). Spaces anywhere in a function command are ignored, but all special punctuation, if any, must appear in the correct form. Any input which is not in the format of

** 5,572 characters for the 8K version of TTYEDIT.
5,572 characters for the 12K version of TCEDIT.

a command is considered to be a source line and will be accepted as such. There are no restrictions on the contents of source lines, except that they end with a carriage return and not begin with a slash. Teletype source lines may only be entered immediately after an insert (/I), or an append (/A) command, or a previously entered source line.

E. ERROR CORRECTION

If an error is detected prior to the typing of a carriage return, the entire line may be deleted by striking the RUBOUT character; TTYEDIT (TCEDIT) will reply with a colon and the line will be ignored and another line may be typed.

The user may correct single characters in a source line being typed by using the backarrow (←) character. This action 'crosses out' the most recently typed character. Several characters may be crossed out in succession by repetitive use of the backarrow as shown in the following example:

```
:TAG    JST    NXIT←←←←EXIT        LEAVE
```

4 backarrows to delete last 4 characters

```
:TAG    JST    EXIT        LEAVE
```

←——line as it will be processed by the system

F. TERMINOLOGY

One of the more attractive features of TTYEDIT (TCEDIT) is the flexibility it gives to the user in identifying the elements with which he wishes to deal. There are three classes of identifiers which are recognized by TTYEDIT (TCEDIT):

1. absolute workspace line numbers;
2. input source file line numbers; and
3. symbolic tags.

An absolute workspace line number refers to the position of a source line with reference to the present contents of the workspace; it is an unsigned integer with a value of 1 to 255.

Thus, the following command/response:

```
:/L 16
 JOE    INC   PTR+2    MODIFY ADDRESS
```

This command lists the sixteenth line currently in the workspace.

The following command would delete the twelfth line of the workspace:

```
:/D 12
```

This command causes all subsequent lines to be "moved up" one in absolute line number order. Note that now the command

```
:/L 15
```

would list the line that was 16th (prior to the deletion) but which would now be "moved up" one in absolute line number order and is now the 15th line.

An input source file line number refers to the sequential position of records (lines) on the old input source tape. The count is set to zero when TTYEDIT (TCEDIT) starts and is maintained as records are read into workspace from the input tape. It is reset to zero every time a /E command is entered, which punches all of workspace, end of tape, and trailer code. (Note: It does not punch leader code.) With TCEDIT, after a /P 2/i command has been entered, a/E command will write the contents of workspace and end of file data to cassette 2; start of file data is not written. The format of a source file line number is a slash (/) followed by an unsigned integer of from one to five digits. The following command would read up to the 263rd line of the old source file.

```
:/R /263
```

A symbolic tag is defined to be a string of from 1 to 4 letters and digits not beginning with a digit and separated by at least one blank character or a backslash character from the rest of the line. The characters "@", "#", and "$" are considered letters for this definition. A source line is considered to have a tag if the first character of the line is a legal tag character; note that source lines need not begin with tags.

Symbolic tags may be used for references alone or with an integer offset. An offset is a signed integer of from 1 to 5 digits. For source lines presently in workspace, references may be made with either positive or negative offsets. The commands:

```
:/D 16
:/D JOE
:/D JOE-0002
:/D JOE+14
```

are all legal, provided the referenced lines are in workspace. For lines on the old source files, a symbolic tag may be used (if they do not already appear in the workspace) with a positive offset, but not a negative offset. The following commands are legal:

```
:/R JOE
:/R JOE+267
```

The following command is not legal (because it has a negative offset):

```
:/R JOE-200
```

We will use the term "internal reference" to mean a reference to a line in workspace, either by a symbolic tag form or by an absolute workspace line number form. Similarly, the term "external reference" will be used for a reference to a line on the old source file, either by symbolic tag or sequential line number.

G. FUNCTIONAL FEATURES

TTYEDIT and TCEDIT may be considered to perform four classes of functions:

1) controlling the movement of the input and output tape files.
2) examining the contents of the workspace;
3) modifying the contents of the workspace; and
4) setting tabs for the automatic alignment of fields.

1.A. Controlling Tape Movement Using TTYEDIT. (/R, /W, /E)

The purpose of an editing program is to fetch lines of an input file, allow the user to modify or add to the lines, and write the corrected lines onto an output file.

The TTYEDIT user explicitly brings records into the workspace by issuing /R commands, using external references. When this command is given, TTYEDIT will read lines into workspace until workspace is full or the requested line has been entered. If workspace fills before the referenced line has been read, an error message will be issued. Generally, when the user wishes to read the entire tape file, he would enter a dummy external reference (i.e., a tag not on the tape file).

The TTYEDIT user explicitly writes lines onto the output file by issuing /W commands specifying the internal reference of the last line to be written from workspace. If no internal reference line number is supplied, the entire buffer will be written.

When the last source line has been read and/or all editing has been accomplished, the user may enter a /E (for "end") command. This command will cause the entire contents of workspace to be punched followed by an EOT (end of tape) character and 50 frames of zero trailer code. The EOT character is a special character recognized by TTYEDIT and the 3300 assembler, ASMB, as the end of tape. To facilitate off-line tape, preparation, TTYEDIT will also treat carriage return/line feed characters followed by at least one zero frame as end of tape. Therefore, the insertion of zero frames (by use of the HERE IS key on the teletype) after a line will let a user segment source tape prepared off-line. The /E command will also set the external line number source file count back to zero.

For the /W and /E commands, an additional procedure is required to allow the user to manually turn the teletype paper tape punch on and off. The procedure is as follows:

a) Type in the /W or /E (including carriage return).

b) The program will now pause. The user may wish to turn the teletype to local, turn the punch ON, press HERE IS (to punch some zero frames) since /W and /E commands do not punch leader code, turn to on-line and press carriage return. If no zero frames are to be punched, the punch may be turned on without the teletype being turned to local.

c) The tape punching operation will be completed and the program will pause again.

d) The user will turn off the tape punch and enter another carriage return to signify the punch is off.

1.B Controlling Tape Movement Using TCEDIT. (/P i[/j], /R, /W, /E)

The TCEDIT program offers the user the capability of accessing cassette drives one and two in addition to the paper tape reader/punch unit on the teletype. The /P command, whose format is shown below, specifies the desired device.

FORMAT: /P i[/j]

where: i = the device number
device 0 paper tape reader/punch
device 1 cassette drive 1
device 2 cassette drive 2

j = a positive integer designating a specific file on cassette drive 1 or 2.

TCEDIT uses device 1 strictly for input (/R) while device 2 is only for output (/W, /E). When TCEDIT is initially loaded the device number is automatically set to zero. To access cassette number 1 in order to read in source lines from file j, the command /P1/j must be entered. A carriage return following this command is not necessary; as soon as j has been entered, TCEDIT will position tape 1 to the beginning of file j. The use of the /R command is identical to the description given in section 1.A. for TTYEDIT. Once a /P1/j command has been executed and file j has been read, a /P1('file' omitted) command will position the cassette to the start of the file immediately following file j; the tape will not be rewound before being repositioned.

To access cassette 2 for outputting source lines to file j, the command /P2/j must be entered. Again, the tape is automatically positioned to the end of the file previous to file j; no carriage return is required. The use of the /W and /E commands is exactly as described in section 1.A. with the following two exceptions:

a. The initial output command (/W, /E) given for file j automatically writes start of file data on the cassette tape.

b. The /E command automatically writes end of file data on the tape.

Once a /P2/j command has been executed and file j has been written and ended (/E) a subsequent /P2 (/'file' omitted) will reposition the tape to the start of the next sequential file without rewinding the tape first.

To access the paper tape reader/punch unit for either input (/R) or output (/W, /E) purposes, the command /P0 must be entered. No carriage return is required, and the file designation /j is omitted.

2. Examining the Contents of Workspace (/L, /S)

The user may list a single line, several lines, or the entire contents of workspace by issuing a /L command with one, two, or no internal references, respectively.

The following command will list the line with tag JOE:

:/L JOE or :/L 10

The following command will list a group of lines:

```
:/L JOE+6, JOE+24      or    :/L 10,20
```

The following command will list the entire contents of workspace:

```
:/L
```

The listing process may be stopped at any time by striking the ESCape key; the EDIT program will then return to entry mode. If symbolic tag references are used, they must be present in workspace or an error message is issued.

The status of WORKSPACE may be displayed by a /S command. This produces a print out of the external sequence number of the last source line read, the amount of workspace available, the first and last lines in the workspace and all tags with their absolute internal value.

Example:

```
:/S
LAST LINE 00650          (last external input line read)
FREE SPACE 00960         (no. of characters left in buffer)
01  JOE   AI     -1      (first line)
05  ALPHA                (line with tag)
11  BETA                 (line with tag)
25        LAI    0       (last line)
```

3. Modifying the Contents of Workspace (/I, /D, /M, /A)

Operations allowed on lines in workspace include insertion of new lines from the typewriter, deletion of lines, changing of lines, movement of lines from one part of workspace to another, and adding lines.

The /I command, used with an internal reference, allows the insertion of any number of typewriter lines after the referenced line. In order to insert lines at the beginning of workspace, the absolute line number 0 is allowed by this command. All lines entered after a /I command are considered as typewriter source lines until another function command (slash, letter) entry is made.

If the number of insertions is such that the workspace becomes full, an error message is issued; the user may either write lines, or make deletions to give himself more room.

The /D command, used with one or two internal references, allows the deletion of one or a group of source lines presently in workspace. For example:

```
:/D JOE+1          or      :/D 12
:/D JOE,JOE+10     or      :/D 10, 20
```

The /M command, used with two or three internal references, allows the user to move one or a group of lines in workspace to a position just after the destination reference. The following command will move the line at JOE plus 2 immediately after JOE. (Effectively, JOE+1 and JOE+2 are interchanged.)

```
:/M JOE+2,JOE
```

The following command will move the referenced block of nine lines to a position immediately after the line with the tag JOE:

```
:/M 16,24,JOE
```

Rules for reference entry for these commands is similar to those of the (/L) command; in addition, the destination of the move (/M) command must be outside the block being moved.

The /A command allows the user to append lines at the end of the last line in the workspace.

4. Tabbing

The /T command is used to set up tabs for tabulated formatting. For example, the following command would set tabs at columns 6, 12, 20, and 30 on the teletype where positions are numbered 0 through 71:

```
:/T 6,12,20,30
```

Tabs greater than 71 but less than 255 are legal and will allow the user to tab to the next lines on the teletype for those teletypes with an automatic carriage return feature.

If a source line is to be output in tabbed format, the backslash (\) character is used to indicate 'space to the next tab setting'. If already past the last tab setting, the backslash is ignored. For example, the following source line will be entered:

```
:TAG\JMP\TAG2\JUMP TO TAG2
```

The above line will be listed in tabbed format as follows if the tabs are set at 6, 12, and 14:

```
TAG   JMP   TAG2JUMP TO TAG2
```

The commands /L, /W, and /E automatically provide tabbed formatting unless nontabbed formatting is specified by typing a backslash in the command immediately following the /L, /W or /E. For example:

```
:TAG\JMP\TAG2
:/L\ TAG
 TAG\JMP\TAG2
```

When punching tapes, the nontabbed format has the advantage of compressing the source lines thus producing a shorter tape.

The 8K 3300 Assembler, ASMB, is written to accept backslash characters as field separators. The Symbolic Editor initially has tabs set at 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50; these are standard tab settings for the 3300 Assembler. Hence, for preparing source tapes for the 3300 Assembler, the tabs need not be set.

NOTE: The 4K 3300 Assembler will not accept backslash characters as field separators.

II. TTYEDIT COMMAND REPERTOIRE

Below are given the legal formats for all TTYEDIT and TCEDIT commands. All commands begin with a slash (/) and are followed by a single letter and, sometimes, additional arguments. Spaces anywhere in a command line are ignored, but all punctuation rules must be observed. Characters enclosed in brackets ([,]) are optional.

Special definitions used in the following descriptions are:

'external reference' { /'integer' | 'tag' | 'tag' + 'integer' }

'internal reference' { 'integer' | 'tag' | 'tag' + 'integer' | 'tag' - 'integer' }

'buffer count' 'integer'

'unit' { '0' | '1' | '2' }

'file' 'positive integer'

A. /A command

Format: /A

Function: Appends subsequently entered lines after last line in workspace until a new command is entered.

Errors: Workspace

B. /D command

Format: /D 'internal reference'

/D 'internal reference', 'internal reference'

Function: Deletes referenced line or group of lines from workspace.

Errors: Reference not in workspace; second reference not after first.

C. /I command

Format: /I 'internal reference'

Function: Allows entry of teletype source lines immediately after referenced line, until another command is entered.

Errors: Reference not in workspace; workspace full (after additions).

D. /E command

Format: /E[\]

Function: TTYEDIT and TCEDIT with /P0 selected. Writes all lines remaining in workspace onto output file; then writes an 'EOT' character followed by 50 tape feed characters. Last line will restart at zero. (A carriage return is entered before and after punching to signify the tape punch has been turned on and off. The backslash means nontabbed format.

Function: TCEDIT with /P2/'file' selected. Writes all lines remaining in workspace onto file designated by 'file'; file control byte are included. Last line will restart at zero. The backslash means non-tabbed format.

Errors: TTYEDIT or TCEDIT with /P0 selected:
None
TCEDIT with /P2/'file' selected:
I/O ERR=3 Logical record greater than tape cassette buffer.
I/O ERR=4 Cassette file protected
I/O ERR=7 Tape error

E. /L command

Format: /L[\]

/L[\] 'internal reference'

/L[\] 'internal reference' , 'internal reference'

Function: (Format 1) Lists entire contents of workspace;
(Format 2) lists referenced line;
(Format 3) lists group of referenced lines.

The backslash indicates non-tabbed format.

F. /M command

Format: /M 'internal reference' , 'internal reference'

/M 'internal reference' , 'internal reference' , 'internal ref.

Function: (Format 1) Moves line or
(Format 2) group of lines to the last internal reference indicated.

Errors: Reference not in workspace; second reference not after first; third reference between first two.

G. /R command

Format: /R 'external reference'

Function: Reads the old source file until the referenced line is in workspace.

Errors: TTYEDIT or TCEDIT with /P0 selected:
Workspace, reference not in workspace.
TCEDIT with /P1/'file' selected:
I/O ERR=5 Read/reread failed

H. /S command

Format: /S

Function: Display status of workspace including last source line read, space available in workspace, the first and last line in workspace, and all lines with tags.

Errors: None.

I. /T command

Format: /T 'integer' , 'integer' , ... , 'integer'

where 0<integer<256.

Function: Set up to 10 tabs.

Errors: More than 10 tabs specified. A tab greater than 255 specified.

J. /W command

Format: /W[\] 'internal reference'

Function: TTYEDIT or TCEDIT with /P0 selected.
Writes (punches) the specified number of lines up to the internal reference onto the new source file paper tape. If the 'internal reference' line number is omitted, all lines in the buffer will be punched and removed. (A carriage return is entered before and after punching to signify that the tape punch has been turned on and off.) The backslash indicates non-tabbed format.

Function: TCEDIT with /P2/'file' selected.
Writes the specified number of lines up to the internal reference onto the file specified by 'file'. If the internal reference is omitted, all lines in the workspace will be written and removed. The first /W command issued for the specified file writes start of file data on the cassette. The /W command does not write end of file data on the cassette. The backslash indicates non-tabbed format.

Errors: TTYEDIT or TCEDIT with /P0 selected:
None

TCEDIT with /P2/'file' selected:
I/O ERR=3 Logical record greater than tape cassette buffer.
I/O ERR=4 Cassette file protected.
I/O ERR=7 Tape error.

K. /P command (TCEDIT only)

Format: /P 'unit' ['file']

Function: Select an input/output device.

'unit' is 0 (paper tape reader/punch), 1 (cassette #1), or 2 (cassette #2).

'unit' = 0: /P0 with no carriage return causes the I/O commands (/R, /W, /E) to access the paper tape reader/punch. The /'file' is not used when 'unit' is zero.

'unit' = 1 /P1/'file' with no carriage return positions cassette #1 to the start of the file specified by 'file'. The input command (/R) will subsequently read source lines from the indicated 'file'.

'unit' = 2 /P2/'file' with no carriage return positions cassette #2 to the end of the file previous to that specified by 'file'. The output commands (/W, /E) will subsequently write source lines to cassette #2.

Errors:

I/O ERR=1	Tape cassette unit not available
I/O ERR=2	File not found
I/O ERR=6	File/record out of sequence
I/O ERR=8	End of tape

APPENDIX A

EXAMPLES OF THE USE OF TTYEDIT AND TCEDIT

I. COPYING A SOURCE FILE.

A. TTYEDIT or TCEDIT with /PO selected.

Default values will be used throughout. A read of a non-existent line number will force a tape copy.

```
:/R /9999
REF. NOT ON FILE
:/E
(CR) (After punch turned ON)
(CR) (After punch turned OFF)
```

NOTE: For tapes larger than the capacity of workspace, several /R /9999, /W sequences would be required before the final /R /9999, /E

B. TCEDIT with /P1['file'] selected

Default values are used. A read of a non-existent line will cause the entire file to be read up to the capacity of workspace.

```
:/P1/1
:/R/9999
REF. NOT ON FILE
:/P2/1
:/E
```

NOTE: For files larger than the capacity of workspace, several /R/9999 , /W sequences may be required before the final /R/9999, /E.

II. CORRECTING ONE LINE OF A SOURCE FILE.

A. TTYEDIT or TCEDIT with /PO selected.

Again, default values are used. The desired line is read, changed, and the remainder of the file is read.

```
:/R JOE+3
:/L JOE+3
     INC   PTR
:/D JOE+3
:/A
:     INC   PNTR
:/R /9999
REF. NOT ON FILE
:/E
(CR) (After punch turned ON)
(CR) (After punch turned OFF)
```

III. CREATING A SOURCE FILE.

A. TTYEDIT or TCEDIT with /P0 selected.

Default values are used.

```
:/I 0          (or /A--begin insertion of empty buffer)
:*     THIS IS THE PROGRAM WHICH
:*     MULTIPLIES A FACTOR RECEIVED AS INPUT IN THE Z AND A
     .
     .
     .
:      INC    PTR
WORKSPACE
:/W
(CR) (After punch turned ON)
(CR) (After punch turned OFF)


:/I 0          (or /A)
:      JMP   *-4
     .
     .
     .
:      END
:/W
(CR) (After punch turned ON)
(CR) (After punch turned OFF)
```

NOTE: Alternatively, /A could be used instead of /I 0.

B. TCEDIT with /P1[/'file'] or /P2[/'file'] selected

Default values are used.

```
:/A
:*     THIS IS THE PROGRAM WHICH
:*     MULTIPLIES A FACTOR RECEIVED AS INPUT IN THE Z AND A.
     .
     .
     .
:      INC    PTR
WORKSPACE
:/P2/1
:/W
:/A
:      JMP   *-4
     .
     .
     .
:      END
:/E
```

or

```
:/R /9999
REF. NOT ON FILE
:/L JOE+3
     INC   PTR
:/D JOE+3
:/I JOE+2
:    INC   PNTR
:/E
(CR) (After punch turned ON)
(CR) (After punch turned OFF)
```

B. TCEDIT with /P2[/'file'] selected

Default values are used. The desired line is read, changed, and the remainder of the file is read

```
:/P1/1
:/R JOE+3
:/L JOE+3
     INC   PTR
:/D JOE+3
:/A
:    INC   PNTR
:/R/9999
REF. NOT ON FILE
:/P2/1
:/E
```

or

```
:/P1/1
:/R/9999
REF. NOT ON FILE
:/L JOE+3
     INC   PTR
:/D JOE+3
:/I JOE+2
:    INC   PNTR
:/P2/1
:/E
```

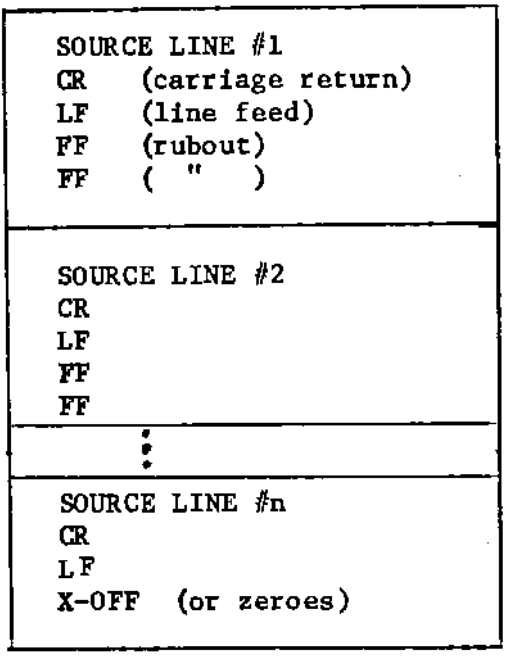
APPENDIX B

ERROR MESSAGES

	MESSAGE	MEANING
1.	WORKSPACE	Workspace is full; some lines must be deleted or written before insertion may continue.
2.	REFERENCE NOT IN WORKSPACE	Internal line number or tag not in workspace buffers.
3.	REFERENCE NOT ON FILE	External line number or tag not on input file.
4.	INVALID	The user command is invalid and/or the argument of the user command is invalid.
5.	I/O ERR=1	Tape cassette unit is not available.
6.	I/O ERR=2	The requested file can not be found.
7.	I/O ERR=3	The size of the current logical record is larger than the tape cassette buffer (256 bytes)
8.	I/O ERR=4	The output cassette is file protected.
9.	I/O ERR=5	Read/reread failure.
10.	I/O ERR=6	File/record out of sequence.
11.	I/O ERR=7	Tape hardware error.
12.	I/O ERR=8	End of tape.

APPENDIX C

PAPER TAPE FORMAT



INPUT REQUIREMENTS FOR TCEDIT

The format of the physical record block for object programs which can be read by the IOCS (Input Output Cassette System) package in TCEDIT is as follows:

1. The physical block is fixed at 256 bytes, defined by the statement

```
BLOCK    DC    X'0100'    256 BYTES
```

2. Control information occurs at the front of each physical block.

   a. byte #1 Tape file data BOF/EOF as follows
      X'AA';X'AB' = beginning of record
      X'F0' X'F1' beginning of file
      X'0E' X'0F' end of file
      X'FE' X'FF' self contained file (i.e., file contained within a single physical block)

   b. byte #2 Check sum byte

   c. byte #3,4 Record length (meaningful text)

   d. byte #5 00

   e. byte #6 File number

   f. byte #7,8 Record number (within file)

   g. byte #9,N Record text characters

In reading a tape file, a check of byte #1 permits IOCS to process the data according to the following rules:

Byte #1 = [X'F0', X'F1']* Beginning of File record.

   a. File number must be 1 greater than prior record.
   b. Record number must be equal to 0001.
   c. Text contains text information of initial data in file.

Byte #1 = [X'AA', X'AB']* Beginning of Record.

   a. File number must be identical to prior record.
   b. Record number must be 1 greater than prior record.
   c. Text contains text information.

Byte #1 = [X'0E', X'0F']* End of File record.

a. File number must be identical to prior record.
b. Record number must be 1 greater than prior record.
c. Text contains text information of last data in file.

Byte #1 = [X'FE', X'FF']* Beginning/End of File record.

a. File number must be 1 greater than prior record.
b. Record number must be 0001.
c. Text data for file totally within current record.

* 1st byte in brackets is byte number one in the first record of the redundant record pair.

2nd byte in brackets is byte number one in the second record of the redundant record pair.

3. The IOCS package provides for redundant recording of each physical block on the cassette tape. To further insure the integrity of all data output to tape, the IOCS package performs the following functions:

a. Check sum control of all data contained within each physical block.
b. Verification of the start of record byte.
c. Verification of file number and record number according to the rules established for each start of record byte.

4. The contents of the record text area (bytes 9 - N) are not of direct concern to the IOCS package except for the number of text bytes and check sum verification.

5. Text material within each physical record contains one or more complete logical records. A logical record is a number of text bytes followed by a carriage return character.

APPENDIX D

SYMBOLIC EDITOR TTYEDIT

LOADING AND OPERATING INSTRUCTIONS

GENERAL

The Symbolic Editor, TTYEDIT, is a program which is used to prepare and correct symbolic source tapes used as input to the Wang 3300 Assembler, ASMB. The user can create a new source tape by typing in a group of lines which are buffered by TTYEDIT, editing them, and punching them out. When all lines have been punched, end of tape (control X-OFF or zeroes) and trailer code can be punched to complete the tape.

Similarly, an old source tape can be modified by reading it in, editing it, and punching it again.

LOADING PROCEDURE

TTYEDIT is loaded with the standard Wang 3300 TTYLOAD (for 8-bit TTYEDIT) or TTLOAD (for 4-bit TTYEDIT) loader. Steps 1 and 2 of the loading procedure may be omitted if the loader is already present in core memory.

1. Key in the 3300 BOOTSTRAP (enter via the computer console). (See section on 3300 System loading procedures.)

2. Using the BOOTSTRAP, load the 3300 loader program (TTYLOAD or TTLOAD). The loading address of the loader should be $(XF40)_{16}$, where X=0 for a 4K system, X=1 for an 8K system, etc.

   (See section on 3300 System loading procedures.)

3. Mount the TTYEDIT tape in the teletype tape reader; set the reader switch to the STOP position, and start the loader at XF40.

The tape will be read and loaded. If loading is successful, the entire tape will be read and the loader will halt with a zero displayed in the A register.

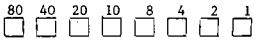
OPERATING PROCEDURE

1. Starting the Program.

   TTYEDIT is always started or restarted at location $(00BA)_{16}$

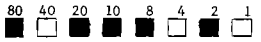
as follows:

(a) Turn the teletype to 'ON-LINE', turn the punch 'OFF'.
(b) Depress ENTER.
(c) Set up 00 on the Data Entry Switches, i.e.,



and press the B button.

(d) Set up BA on the Data Entry Switches, i.e.,



and press the C button.

(e) Depress RUN.
(f) Press CLEAR.
(g) Press GO.

2. Entering a Command.

The teletype will type out a colon and the user may enter his first command. Thereafter, after each command is completed, a colon will be typed indicating a new command may be entered. (With certain commands, such as /I and /A, a new command line terminates the operation.)

3. Clear Workspace.

When TTYEDIT is first loaded and executed, the buffer will be empty and ready for input lines. If the user restarts TTYEDIT with data lines already in the buffer, he must go through the following procedure to remove them:

(a) Enter a /S command to see what lines are in the buffer and to determine the highest internal line number.

(b) Delete all lines currently in the buffer with the following command:

/D 1,XX (where XX is the highest line number)

(c) Enter a /E to reset the external input tape source line count to 0. (The teletype will also type out trailer code, which has no affect.)

The /E command will also do one other important clearing function. When a line from an old source tape is read, the first character of the next line is also read and saved (since it takes two characters to stop the teletype reader). If an end of tape is encountered, the saved character will be a null trailer code character and will not be saved. If, however, the user does not read the old source tape in completely, and wishes to initialize TTYEDIT, the saved character from the next line will still be in the system; a /E command will clear it.

APPENDIX E

SYMBOLIC EDITOR TCEDIT

LOADING AND OPERATING INSTRUCTIONS

GENERAL

The Symbolic Editor, TCEDIT, is a program which is used to prepare and correct symbolic source files on cassette. These files can subsequently be used as input to the Wang 3300 Assembler, TCASMB. The user can create a new source file by typing in a group of lines, editing them, and writing them out to cassette or to paper tape. When all lines have been written to cassette, end of file data is automatically added to the tape.

Similarly, an old source file can be read from cassette, modified, and recopied.

LOADING PROCEDURE

TCEDIT is loaded using the standard Wang 3300 TCLOAD program which is in turn loaded by the appropriate version of TCBOOT#1 or TCBOOT#2. The user should refer to the PRELIMINARY OPERATING MANUAL FOR THE WANG 3320 DUAL CASSETTE UNIT for detailed operating instructions for this device.

1. Key in the teletype BOOTSTRAP program (see Step #3, page 6).

2. Using the BOOTSTRAP, load the appropriate version of TCBOOT#1 or TCBOOT#2, by following the instructions in Step #4, page 9. The TCBOOT tape replaces the teletype loader tape in that section.

3. Place the TCEDIT/TCASMB cassette in the left port of the 3320 Dual Cassette unit (the right port if TCBOOT#2 was loaded in Step 2), and press REWIND.

4. Start the TCBOOT program at 2F40.
The TCLOAD program will be read from the cassette and the 3300 will halt with an 03 in the A-register. A correct load will cause an 04 to appear in the A-register when the 3300 halts.

OPERATING PROCEDURE

1. Starting the Program

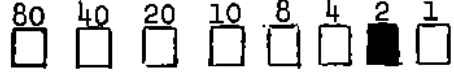
TCEDIT is always started or restarted at location $(0200)_{16}$ as follows:

(a) Turn the teletype to 'ON-LINE,' turn the punch 'OFF'.

(b) Depress ENTER on the 3300 console.

(c) Set up 02 on the Data Entry Switches

i.e. 80 40 20 10 8 4 2 1



and press B.

(d) Set up 00 on the Data Entry Switches

i.e. 80 40 20 10 8 4 2 1

and press C.

(e) Press RUN

(f) Press CLEAR

(g) Press GO

2. Entering a Command

The Teletype will type out a colon and the user may enter his first command. After each command is completed, a colon will be typed indicating that a new command can be entered. With certain commands, such as /I and /A, a new command line terminates the operation.

3. Clear Workspace

When TCEDIT is first loaded and executed, workspace will be empty and ready for input lines. If the user restarts TCEDIT with data lines already in the workspace, he can use the following procedure to remove them:

(a) Enter a /S command to see what lines are in the workspace and to determine the highest internal line number.

(b) Delete all lines currently in the buffer with the following command:

/D 1,XX (where XX is the highest line number)

(c) Enter a /PO command to access the paper tape punch unit.

(d) Enter a /E to reset the external input tape source line count to 0. (The teletype will also type out trailer code, which has no affect).

INTERRUPTING PROGRAM EXECUTION

The TCEDIT program contains an additional feature which is not present in TTYEDIT. The ESC key on the Teletype can be used to interrupt the TCEDIT program at any time; the interrupt feature in TTYEDIT is limited to the /L and /S commands. When the ESC key is struck, the system responds with an "E" error message and a colon to signify that it is ready to accept new input from the user.

```
i.e.  :/P2/1
      :/W
            (ESC key struck)
      I/O ERR = E
      :
```

WANG 3300

ASSEMBLER MANUAL

(ASMB4)
(ASMB8)
(TCASMB)

Revised: July 15, 1972

TABLE OF CONTENTS

TABLE OF CONTENTS

ABSTRACT

The WANG 3300 Assembler, ASMB, translates symbolic source language instructions into an object program ready for execution on the computer.

A 4-bit version, ASMB4, operates with the minimum WANG 3300 configuration (4K core memory and 1 teletype). An 8-bit version, ASMB8, is available for larger machines which provides a larger symbol table. A second 8-bit version, TCASMB, is available for systems having 12K or more core memory, one Teletype, and a model 3320 Dual Cassette Unit. TCASMB reads symbolic source files on cassette produced by TCEDIT. The source language is read from the symbolic source tape three times. On pass 1 a symbol table is generated and printed out, along with basic syntax errors. On pass 2 a source and object code assembly listing is printed. On pass 3 an object code paper tape which is compatible with the 3300 teletype loader is punched.

I. INTRODUCTION.

The WANG 3300 Assembler, ASMB, is a standard software module of the WANG 3300 digital computer system. It translates symbolic source language instructions into an object program compatible with the WANG 3300 teletype loader.

The 4-bit version, ASMB4, is a minimum assembler; that is, it operates with a minimum WANG 3300 system (4K core memory, 1 teletype). The assembler tape is in 4-bit format and ASMB4 punches object tapes in 4-bit format; these tapes are loaded with TTLOAD (4-bit teletype loader). ASMB4 allows for a maximum of 64 symbols.

The 8-bit version of the assembler, ASMB8, is for use with machines with at least 8K of core memory; a larger symbol table is provided for. ASMB8 allows for 64 + 682 (no. of 4K-core memory modules greater than 8K). The assembler tape is in 8-bit format and ASMB8 punches object tapes in 8-bit format; these tapes are loaded with TTYLOAD (8-bit Teletype loader).

A second 8-bit version of the assembler, TCASMB, operates with machines having at least 12K of core and one model 3320 Dual Cassette Device. TCASMB allows for 64 + 682 (no. of 4K core memory modules greater than 12K). The assembler tape is in 8-bit format and punches object paper tapes in 8-bit format. These tapes are loaded with TTYLOAD (8-bit Teletype loader).

ASMB is a three-pass assembler. During the first pass, the source language symbolic paper tape or cassette tape file is read and the symbol table is created. The second pass reads the source tape again, generates an assembly listing on the teletype of the object code and source lines. Pass 3 punches the object code on paper tape. Since additional syntax errors are detectable during Pass 2, it is advisable that the user select Pass 2 before selecting Pass 3.

If a given source tape is known to be error free and no listing is desired, the user may progress directly from Pass 1 to Pass 3 punch phase. (See Appendix E, page 104).

At the end of the assembly, the program is ready to be loaded from the object tape and executed.

II. GENERAL INFORMATION.

A. Symbolic Instructions.

There are three parts to a symbolic instruction.

1. The tag field which contains a symbol of from 1 to 4 characters that names the instruction. Other instructions may refer to this instruction by referencing the symbol. Use of the field in machine instructions is optional; requirements for assembler instructions are given in Section III.

2. The operation field which contains the machine instruction code or assembler instruction code. This field is required. An asterisk (*), plus (+), or minus (-) may appear in the operation field immediately to the right of the machine instruction code and specify indirect addressing, auto-incrementing and auto-decrementing, respectively.

3. The operand field which contains the address portion of machine instructions or whatever is specified for assembler instructions. This field is generally required, but it may be omitted in certain instructions. A null operand field is indicated by a comma.

An optional field is also available.

4. The comments field which is used for remarks. This field does not affect the execution of the program.

B. Assembler Character Set.

Source statements are written using the following characters:

Letters:	A through Z, and $, #, @ (Note: $, #, @ are usually referred to as 'rational characters'.)
Digits:	0 through 9
Special Characters:	+, -, ', *, blank \ (backslash) for 8-bit ASMB, only

Characters that appear in a character constant and comment fields are not restricted to the assembler character set. Any ASCII character msy appear.

C. Symbolic Line Format.

Symbolic instructions are typed one per line in the following format:

The tag field, if present, starts in column 1; otherwise, column 1 is blank. This tag field is a symbol of from 1 to 4 nonblank characters. At least one blank column separates the tag field from the operation field, the operation field from the operand field, and the operand field from the comment field. (In the 8-bit version, ASMB8, fields may also be separated by a backslash character (\).)

A remarks line, indicated by an asterisk (*) in column 1, allows the entire statement field to be used for comments. A remarks line is listed but does not affect the object code. A remarks line may appear anywhere in the source program.

Note: The tag, operation, and operand field must not have embedded blanks.

To obtain an aligned listing in ASMB4, sufficient spaces should be inserted between fields such that the operation field begins in column 6 and the operand field begins in column 11. In the 8-bit version, ASMB8, an aligned listing is automatically produced if the fields are separated by backslash characters (\).

A carriage return/line feed combination, though not counted as a character in determining the line length, is considered an integral part of a source line and, in fact, is the delimiter between source lines.

D. Assembler Language Structure.

1) TAG FIELD

The tag field (usually optional) is a symbol (from 1 to 4 non-blank characters, the first character being a letter). Note, backslash (\) is an illegal tag character in ASMB8.

2) OPERATION FIELD

The operation field (required) is a mnemonic operation code representing a machine instruction or an assembly instruction. These are names of 3 or 4 letters optionally followed by an *, + or - to indicate indirect addressing, auto-incrementing and auto-decrementing of the address (operand) field. No embedded blanks are permitted. A complete list of operand field mnemonic names is presented in APPENDIX A, page 96.

3) OPERAND FIELD

The operand field is composed of an expression which in turn is composed of a term or an arithmetic combination of terms. It defines either the address for memory reference and jump instructions, or special symbols, addresses, masks, and shift counts for other machine and assembler instructions.

Examples:

```
TAG     OPERATION       OPERAND         COMMENTS
        ORG             X'2000'
FIX     LZ+              BUF1           FIND NEXT NON-ZERO DATA
        JZE              FIX
FIX1    UZ              CNT2+2          STORE IT
        TZAJ                            ,TRANSFER TO A REGISTER
        ...
        ...
        ...
```

For ASMB8 and TCASMB, the following examples are also valid:

```
\ORG\X'2000'
FIX\LZ+\BUF1\FIND NEXT NON-ZERO DATA
\JZE\FIX
FIX1\UZ\CNT2+2\STORE IT
\TZAJ\\,TRANSFER TO A REGISTER
```

4) SYMBOL

A symbol is a character string of from 1 to 4 (maximum) characters in length. The first character must be a letter. The other characters may be letters, digits, or rational characters. A symbol appearing in the tag field of a symbolic instruction is assigned a value by the assembler. The value assigned to each symbol naming machine instruction, storage areas and constants is the left-most byte. A symbol defined by an EQU instruction is assigned the value of the expression in the operand field.

5) TERMS

A term represents a numerical value. A term may be a symbol, a location counter reference, or a self-defining term.

6) LOCATION COUNTER REFERENCE

The assembler maintains a location counter. This counter points to the next available memory location (byte address).

The location counter is referred to by using an asterisk (*). For example, JMP *+4.

7) SELF-DEFINING TERMS

A self-defining term is one whose value is inherent in the term. For example, the decimal self-defining term, 32, represents the value 32. There are four types of self-defining terms: decimal, hexadecimal, binary, and character.

A. Decimal Self-Defining Term

A decimal self-defining term is an unsigned decimal integer written as a sequence of from 1 to 5 digits. A decimal constant can not exceed a value that requires more than 16 bits; therefore, the maximum decimal integer is 65,535. If it exceeds this amount, an error diagnostic will result and the expression will be set to zero. Some examples of decimal self-defining terms are: 16, 132, and 0015.

When terms are expressed in decimal mode, they will be converted to their 16-bit binary equivalent.

B. Hexadecimal Self-Defining Term

A hexadecimal self-defining term consists of a string of hexadecimal digits enclosed by apostrophes and preceded by the letter X (i.e., X'A1E9'). Hex constants are a maximum of 2 bytes long, except when used with the DC assembler instruction, but are otherwise subject to constraints similar to those in the section on binary constants, e.g.,

TERM	UNDERSTOOD AS	LENGTH, BYTES
X'1'	X'01'	1
X'12A'	X'012A'	2
X'ABC123'	X'C123'	2

C. Binary Self-Defining Term

A binary self-defining term is written as a sequence of 1's and 0's enclosed in single quotation marks and preceded by the letter B (i.e., B'10110101').

A binary constant should not exceed 16 bits in length, or a truncation diagnostic will be generated, and the term value taken as the equivalent of the low-order 16 bits. For constants of 16 or less bits, the length will be taken as the next whole byte and will be padded from the left with zeroes, e.g.,

TERM	UNDERSTOOD AS	LENGTH, BYTES
B'01'	B'00000001'	1
B'10000000'	B'10000000'	1
B'100000001'	B'0000000100000001'	2
B'1111011111111011'	B'1011111011111011'	2

D. Character Self-Defining Term

This term is used to define the ASCII code equivalent of characters. A character self-defining term consists of a string of characters enclosed in single quotation marks and preceded by the letter C (i.e., C'AB').

Character constants are normally taken to be one or two bytes long except when used with a DC assembler instruction. In an arithmetic expression are truncated if necessary to a maximum of 2 bytes (with a diagnostic), the value of which is the (hex) ASCII equivalent of the last two characters in the string, i.e.,

```
TAG1   LAI   C'AB'
```

is illegal; the result would be A-register = X'42'. Note that

```
TAG2   DL    X1
X1     DC    C'AB'
```

is acceptable; the result would be X'4142' in the Z- and A-registers.

Length of Self-Defining Terms

In a DC (define constant) statement, character or hex constants may be as long as desired (up to the maximum length of a source line, defined as 50 contiguous characters including blanks). Elsewhere the length of the constant must be one byte except where the term represents a full 16-bit computer address, such as with the DAC and EQU assembler instructions.

On truncation of constants: A constant used in a machine instruction is always truncated to one byte right-justified, e.g.,

```
LAI   C'ABCD'
```

is the same as LAI C'D'.

The same is true of addresses. To say the following means JMP X'42' (on the current page):

```
JMP   X'A542'
```

If A5 is the current page number, this is acceptable; but, if not, strange results will ensue.

8. EXPRESSIONS

An expression is a term or an arithmetic combination of terms, with allowed arithmetic operators: +, -, *, ↑. The following are examples of arithmetic expressions:

```
ALPH+12
*+X'AB'
*
0-BETA+B'101'
*↑2
A+B↑X'01'
2↑3↑B'0101'
```

where: * = current program counter location
+ = add
- = subtract
↑ = multiply

The rules for coding expressions are:

1) The allowable arithmetic operators are +, -, *, ↑(Shift N key on teletype).
2) An expression can not contain two successive terms or operators. The expression may begin with an operator (except ↑); otherwise, + is assumed.

A. Evaluation of Expressions

1. Every term is converted to its binary equivalent (module 2).

2. Arithmetic operations are performed from left to right, including multiplication. For example:

CODED FORM	ALGEBRAIC
X↑Y	XY
2+X↑Y	Y(2+X)
Y↑2+X	2Y+X
X↑2+Y	2X+Y
2+X↑Y+3	(2+X)Y+3

3. Expressions are computed to 16 bits. Negative values are in 2's complement form.

4. Parentheses are not permitted.

III. ASSEMBLER INSTRUCTIONS ("PSEUDO-OPS")

Assembler instructions are requests to the assembler to perform certain operations during assembly.

The following is a list of assembler instructions:

Symbol Definition Instruction

EQU - Equate Symbol

Data Definition Instructions

DAC - Define Address Constant

DC - Define Constant

DS - Define Storage

Program Control Instructions

ORG - Set Location Counter

ALN - Align Location Counter on Even Boundary

APG - Set Absolute Page

END - End Assembly

1. EQU - Equate Symbol

The EQU instruction defines a symbol by assigning to it the value of the expression in the operand field. The format of the EQU instruction statement is as follows:

Tag	Operation	Operand
Symbol	EQU	Expression

Any symbol in the operand field must be previously defined. That is, any symbol in the operand of an EQU instruction must have appeared in the tag field of a prior statement.

Examples of the EQU instruction are:

```
TAPE   EQU   5

MASK   EQU   B'10101010'

ADDR   EQU   ALPH-BETA+X'12'

BITS   EQU   ALPH↑BETA
```

(Note -- ALPH and BETA must have been previously defined.)

The EQU symbol can be used to perform an additional function which is applicable only with the TCASMB program. If the line

Tag	Operation	Operand
'' (two single quotes)	EQU	*

is entered anywhere in a program, TCASMB will construct a "common" symbol table containing all those symbols which are previous to the EQU line. These symbols will be saved in the computer after the punch phase has been completed, and can thereby be referenced by subsequent programs. The common symbol table can be removed by initiating a pass '0'. The following example illustrates the use of this feature.

```
*
* PAGE ZERO DATA
*
      ORG   X'0000'
ZER   DC    X'0000'
LIM   DC    X'0064'
NUM   DC    X'02'
"     EQU   *
      END
```

```
*
* SAMPLE PROGRAM
*
      ORG   X'0100'
STRT  DL    ZER
      DU    CNT
      DU    ADDR
      ONS   08
BEG   LA    NUM
      DAM   ADDR
      INC   CNT+1
      DL    LIM
      DCM   CNT
      JNE   BEG
      DL    ADDR
      HLTJ  STRT
ADDR  DS    2
CNT   DS    2
      END
```

2. DAC - Define Address Constant

The DAC instruction defines a 16-bit address from the expression in the operand field. An address constant is usually used by an indirect memory reference instruction. In this way, any memory location may be accessed. An address constant has a storage length of two bytes and is aligned on an even boundary. If the DAC assembler instruction occurs on an odd boundary, a byte of zero is generated and the location counter is incremented by 1 before the address constant is stored. A blank operand field generates two zero bytes on an even boundary.

The format of the DAC instruction statement is as follows:

Tag	Operation	Operand
Blank or Any Symbol	DAC	Expression or Blank

Examples of valid DAC instructions are:

```
ADDR    DAC    ALPH
TWLV    DAC    12
```

3. DC - Define Constant

The DC instruction defines constant data. The format of the DC instruction statement is as follows:

Tag	Operation	Operand
Blank or Any Symbol	DC	Format described below

There are three types of constants that may appear in the operand of a DC instruction. Each is specified by a single letter code. The constant types and their machine format are the following:

Code	Type of Constant	Machine Format
C	Character	8-bit ASCII for each character
X	Hexadecimal	4-bit code for each hex digit
B	Binary	binary format

A. Character Constant

A character constant is a string of characters enclosed in single quotes and preceded by the letter C; for example, C'HI THERE'. Characters that appear in a character constant are not restricted to the assembler character set. Any character that can be represented can appear. A single quote in a character constant is represented by a pair of quotes. Each character of the string is assembled as its eight bit ASCII equivalent. The storage length of a character constant is equal to the number of characters in the string. Example: DC 'ERROR'

B. Hexadecimal Constant

A hexadecimal constant is a string of hexadecimal digits enclosed in quotes and preceded by the letter X (i.e., X'A5B'). Each hexadecimal digit is assembled as its 4-bit hexadecimal equivalent. If the hexadecimal has an odd number of terms, a leading 0 is supplied.

```
Examples:       DC   X'12F4E9'
                DC   10X'40F0'
```

C. Binary Constant

A binary constant is a string of 0's and 1's, enclosed in quotes and preceded by the letter B (i.e., B'101'). Eight contiguous 0's and 1's are assembled into one byte. If the length of the string is not divisible by 8, leading 0's are added to fill the constant to the boundary of the next available byte. The storage length, therefore, of a binary constant is always one or two bytes.

```
Examples:    DC    B'10101101'
             DC    10B'10101010'
```

D. Replication Factor

Any unsigned decimal integer, less than or equal to 65,535, preceding a constant is a replication factor. The total storage length is the storage length of the constant multiplied by the replication factor. An example of the use of a replicating factor is the following which sets 100 bytes to X'00'.

```
             DC    100X'00'
```

If the replication factor is not followed by a constant, a constant of X'00' is implied. That is, n bytes of storage are set to zero, where n is the replication factor. A replication factor of 0 is allowed, but object code is not generated. Factors greater than 65,535 are set equal to the difference between 65,536 and the factor.

4. DS - Define Storage

The DS instruction reserves the number of storage bytes as specified by the operand. The format of the DS instruction statement is as follows:

Tag	Operation	Operand
Blank or Any Symbol	DS	Same as DC Operand Format

The storage length reserved is the same as if the constant had been generated. Object code is not generated. The location counter is incremented by the storage length of the constant.

Examples of valid DS instructions are the following:

```
TABX    DS     10
TABZ    DS     100B'101'
```

The first example reserves 10 bytes of storage; the second example, 100 bytes of storage.

5. APG - SET ABSOLUTE PAGE

The assembler performs a validity check on memory address references. A memory reference must address the current page (the page the memory instruction reference is on) or one of the absolute pages, page 0 or page 1. The APG instruction indicates to the assembler the absolute page the program is using. This will eliminate absolute page reference error diagnostics. The format of the APG instruction is as follows:

Tag	Operation	Operand
Blank	APG	An expression which evaluates to 0 or 1

The assembler initially assumes the absolute page is page 0. If the operand evaluates to less than zero or greater than 1, a diagnostic is generated and the instruction is ignored.

6. END - End Assembly

The END instruction must be the last instruction statement in the assembly and it terminates the input of source statements for the assembly. The operand may be used to indicate the program's entry point. The format of the END instruction statement is as follows:

Tag	Operation	Operand
Blank or Symbol	END	Blank or an Expression

7. ORG - Set Location Constant

The ORG instruction sets the location counter to the value specified by the expression in the operand. The format of the ORG instruction statement is as follows:

Tag	Operation	Operand
Blank or Symbol	ORG	Blank or an Expression

The assembler assumes an initial location counter of 0. If the operand is blank, the location counter is set to the highest value that has occurred during the assembly. Any symbol in the operand field must be previously defined.

Examples of valid ORG instructions are the following:

```
ORG  X'100'

ORG  256

ORG  ALPH+10

ORG  ALPH+10↑X'02'
```

Note: ALPH must have been previously defined.

8. ALN - Align Location Counter on Even Boundary

The ALN instruction sets the location counter to an even boundary. If the location counter value is odd, a zero byte is generated and the location counter is incremented by 1; otherwise, the ALN instruction results in no operation. The format of the ALN instruction statement is as follows:

Tag	Operation	Operand
Blank	ALN	Blank

IV. MACHINE INSTRUCTIONS.

The 73 valid machine instruction mnemonics are listed in APPENDIX A.

The assembly of machine instructions involves the following functions:

1) If there is a symbol in the tag field, the symbol is given the value of the location counter.

2) The mnemonic machine instruction code is translated into 8 bits and stored in the left half of the 16-bit instruction word.

3) The operand, if present, is evaluated. For most instructions, the shift group excepted, the low-order 8 bits of the expressions value are logically ORed with the right half of the instruction word. For shift instructions, 1 is subtracted from the value of the expression and the low-order 2 bits ORed into the right half of the instruction word.

   For certain instructions, the operand portion of the instruction is a mask which specifies which bits will be tested. In some instructions, one-bits indicate tested bits; in others, zero-bits indicate tested bits. To allow consistant assembly language programming, the operand portion of those instructions which have zero test bit indication will be automatically complemented by the assembler such that one bits may always be used in the source code to indicate tested bits. These instruction are OFS, SFA, SFG, SFI, SMA, and SFS.

4) If indirect addressing or auto-indexing are specified, the appropriate bits are set.

5) For memory reference instructions, the absolute/current page bit is inserted.

6) The final 16-bit instruction word is assigned to the location specified by the location counter.

7) The location counter is incremented by 2.

The format of a machine instruction is as follows:

Tag	Operation	Operand
Blank or Symbol	Mnemonic Machine Instruction Code	Blank or Expression

Note that any machine instructions ending in "J" (i.e., LZAJ, XZAJ, CIOJ, etc.) do not require an operand. If no operand is present, the assembler assumes it to be "*+2", i.e., the address of the next sequential instruction. However, if a comment is desired on the line with no operand, a comma (,) must be inserted before the comment or the comment will be regarded as an operand. In addition, if no operand is desired, the trailing "J" may be dropped from the operator.

For example,

```
CIO             ,COMMENT
```

is interpreted by the assembler as the following:

```
CIOJ  *+2       COMMENT
```

V. ASSEMBLER OUTPUTS.

The assembler has two forms of output:

1) A printed listing summarizing the generated object code together with the associated source statements and any assembly diagnostics, followed by a listing of all symbolic names used in the assembly.

2) An object code tape which may be loaded into the Wang 3300 using the LOADER.

A. Output Listings

The assembler output listing consists of two parts:

1) A symbol table listing with tag error diagnostics on the first pass.
2) A listing of source statements side by side with generated object code and detected assembler diagnostics on the second pass.

   In Pass 2, the source statement fields are automatically aligned if they are separated by backslash characters; this only applies to ASMB8 and TCASMB (backslashes are illegal field separators in ASMB4). The backslash character is interpreted as 'space to the next tab setting'. The source statement is printed beginning in column 22; tabs are set at columns 27, 32, 37, 42, 47, 52, 57, 62, and 67.

B. Error Diagnostics

When violations of assembly rules are detected by the assembler, an appropriate error flag is inserted in the diagnostic field of the assembly listing. The diagnostic field has space for up to five error diagnostics. A complete list of error diagnostics and their associated meaning is found in APPENDIX C, page 101.

APPENDIX A

3300 INSTRUCTION MNEMONICS

Table A.1 lists the machine instruction codes. A description of the meaning of each column in Table A.1 follows:

MNEMONIC: Machine instruction mnemonic code.

DESCRIPTION: Functional description of instruction.

IND: An I in this column indicates indirect addressing is valid. (Indirect addressing is coded by immediately following the mnemonic code with an asterisk (*), e.g., LA* .)

AUTO: An A in this column indicates auto-indexing is valid. (Auto-indexing is coded by immediately following the mnemonic code with either a + or a -, e.g., LA+ .)

OPERAND: This column designates the valid operand format for the corresponding instruction:

ADDR - operand must be a memory address reference.

DATA - operand must be a data item of 8 bits or less (-256 to 255). In some instructions, the normal sense of the 0's and 1's is reversed. To compensate, the assembler takes the 1's complement of the data item. This is indicated in the table by an *.

SHIFT - operand must be a data item of 2 bits or less (1 to 4). The assembler subtracts 1 from the operand to agree with the actual machine instruction format.

JUMP - operand must be an in-page address.

BLANK - operand may be omitted. For instructions that accept an in-page jump address and the operand is null (blank), the assembler inserts an address of *+2.

TABLE A.1

MNEMONIC	DESCRIPTION	IND	AUTO	OPERAND
AC	ADD WITH CARRY	I	A	ADDR
ADD	ADD	I	A	ADDR
AMC	ADD TO MEMORY	I	A	ADDR
C	COMPARE	I	A	ADDR (EVEN)
DAM	DOUBLE ADD TO MEMORY	I	A	ADDR (EVEN)
DCM	DOUBLE COMPARE WITH MEMORY	I	A	ADDR (EVEN)
DL	DOUBLE LOAD	I	A	ADDR (EVEN)
DU	DOUBLE UNLOAD	I	A	ADDR (EVEN)
LA	LOAD A	I	A	ADDR
LZ	LOAD Z	I	A	ADDR
UA	UNLOAD A	I	A	ADDR
UAH	UNLOAD A HIGH DIGIT	I	A	ADDR
UZ	UNLOAD Z	I	A	ADDR
XMA	EXCHANGE MEMORY AND A	I	A	ADDR
BA	BOOLEAN ADD	I		ADDR
BO	BOOLEAN OR	I		ADDR
BX	BOOLEAN EXCLUSIVE OR	I		ADDR
INC	INCREMENT	I		ADDR
JEI	JUMP AND ENABLE INTERRUPT	I		ADDR
JMP	JUMP	I		ADDR
JST	JUMP AND STORE LOCATION	I		ADDR
AI	ADD IMMEDIATE			DATA
BAI	BOOLEAN AND IMMEDIATE			DATA
BOI	BOOLEAN OR IMMEDIATE			DATA
BXI	BOOLEAN EXCLUSIVE OR IMMEDIATE			DATA

TABLE A.1

MNEMONIC	DESCRIPTION	IND	AUTO	OPERAND
CI	COMPARE IMMEDIATE			DATA
DLI	DOUBLE LOAD IMMEDIATE			DATA
LAI	LOAD A IMMEDIATE			DATA
LZI	LOAD Z IMMEDIATE			DATA
ONS	ON STATUS			DATA
SAA	SKIP IF ANY A			DATA
SIA	SKIP IF TRUE A			DATA
SIG	SKIP IF GENERAL TRUE			DATA
SII	SKIP IF TRUE I/O			DATA
SIS	SKIP IF TRUE STATUS			DATA
ICH	INITIATE CHANNEL			DATA
ECH	END CHANNEL			DATA
OFS	OFF STATUS			DATA *
SFA	SKIP IF FALSE A			DATA *
SFG	SKIP IF GENERAL FALSE			DATA *
SFI	SKIP IF FALSE I/O			DATA *
SFS	SKIP IF FALSE STATUS			DATA *
SMA	SKIP IF MIXED A			DATA *
RT	ROTATE			SHIFT
RTC	ROTATE WITH CARRY			SHIFT
SH	SHIFT			SHIFT
SHC	SHIFT WITH CARRY			SHIFT
AKIJ	ACKNOWLEDGE INTERRUPT & JUMP			JUMP/BLANK
CIOJ	CONTROL I/O AND JUMP			JUMP/BLANK

TABLE A.1

MNEMONIC	DESCRIPTION	IND	AUTO	OPERAND
DNJ	DOUBLE NOT AND JUMP			JUMP/BLANK
DSIJ	DISABLE INTERRUPT AND JUMP			JUMP/BLANK
LZAJ	LOAD FROM Z AND A AND JUMP			JUMP/BLANK
HLTJ	HALT AND JUMP			JUMP/BLANK
NJ	NOT A AND JUMP			JUMP/BLANK
OMNJ	ON MASK AND JUMP			JUMP/BLANK
PARJ	ADD ODD PARITY TO A AND JUMP			JUMP/BLANK
RDDJ	READ DATA AND JUMP			JUMP/BLANK
SBCJ	SHIFT BINARY DOUBLE WITH CARRY & JUMP			JUMP/BLANK
SBJ	SHIFT BINARY DOUBLE AND JUMP			JUMP/BLANK
SDJ	SHIFT DECIMAL DOUBLE AND JUMP			JUMP/BLANK
TASJ	TRANSFER A TO STATUS AND JUMP			JUMP/BLANK
TSAJ	TRANSFER STATUS TO A AND JUMP			JUMP/BLANK
TZAJ	TRANSFER Z TO A AND JUMP			JUMP/BLANK
WRDJ	WRITE DATA AND JUMP			JUMP/BLANK
XZAJ	EXCHANGE Z AND A AND JUMP			JUMP/BLANK
JZA	JUMP ON Z AND A			DATA/BLANK
JEQ	JUMP EQUAL			JUMP
JGT	JUMP GREATER THAN			JUMP
JLT	JUMP LESS THAN			JUMP
JNC	JUMP STATUS C BIT OFF			JUMP
JNE	JUMP NOT EQUAL			JUMP
JNZ	JUMP STATUS Z BIT OFF			JUMP
JZ	JUMP STATUS Z BIT ON			JUMP

APPENDIX B

3300 ASSEMBLER INSTRUCTION (Pseudo-Ops)

Table A.2 lists the assembler instruction codes. A complete description of the meaning and use of each assembler instruction may be found in Section III.

Table A.2

MNEMONIC	DESCRIPTION
ALN	ALIGN ON EVEN BOUNDARY
APG	SET ABSOLUTE PAGE
DAC	DEFINE ADDRESS
DC	DEFINE CONSTANT
DS	DEFINE STORAGE
END	END ASSEMBLY
EQU	EQUATE SYMBOL
ORG	SET LOCATION COUNTER

APPENDIX C

3300 ASSEMBLER ERROR CODES

ERROR	EXPLANATION
A	Tag required and not present, e.g., the EQU statement requires a symbolic tag. The statement is ignored.
C	Bad constant, such as X'AQB35'. Operand part of instruction will be set to zero.
E	An indirect (+, -, or *) reference is to a pointer on an odd location. The assembler will calculate the address properly, but if execution is attempted a machine error will result.
F	Symbol table overflow. Too many tag names were defined. Some must be removed and the program re-assembled. (ASMB4 allows for a maximum of 64 symbols; ASMB8 allows for the number of symbols shown by the formula on page 74).
I	An auto-indexing or indirect reference was attached to an op code for which it is not permitted. Instruction assembled into a halt.
M	Multiply defined symbol. The second definition of the symbol is ignored; all references to a symbol use its value at initial definition.
O	Illegal operation code. Assembles into a halt instruction.
Q	Memory reference has been made to a location not on current or absolute page, though the symbol referenced does exist in memory. Operand part of instruction will be set to zero.
R	Operand required and not present. For assembler instructions (pseudo-ops), the instruction is ignored. In machine instructions, the operand part of the instruction is set to zero.
S	Shift attempted greater than 4 bits or less than 1. The instruction assembles properly, but will cause a machine error if execution is attempted.
T	Bad term such as "TAGN DU RUMPLESTILSKIN". Such an operand cannot exist, and there is no reasonable truncation.
U	Reference has been made to a symbolic tag which is nowhere defined.
V	Value of operand is invalid. Occurs only in APG statements. Statement is ignored.
X	Two operators in expression have been encountered in succession, i.e., AP3+-AP4.

ERROR	EXPLANATION
Z	The value of an operand has been truncated. The representation of the operand was too long to fit into the operand part of an instruction and is truncated from the right, e.g.,

```
LAI   X'345'  means   LAI  X'45'
DLI   C'AB'   means   DLI  C'B'
```

I/O ERR=5 Read/reread failed

Notice that even if some of the diagnostics are set, one may proceed to Pass 3. It is then necessary to exercise care in resetting these memory locations by hand to the proper numbers before execution is attempted.

There is one error for which there is no diagnostic and for which the programmer is responsible:

The location counter (in ASMB) has a maximum value of X'FFFF', or nominally 64K bytes. If a programmer has available an 8K machine, it will still assemble 64K programs and will attempt to load them; the code in excess of core size will be lost, however. (The computer does not wrap-around). The location counter is also "circular" in the sense that 2 bytes beyond location X'FFFF' is location X'0001'. It is the programmer's responsibility to be aware of this.

APPENDIX D

OUTPUT LISTING

During Pass 1, any errors encountered cause the source line to be listed. The format of the listing is:

Columns	1- 5	diagnostic message area
"	6	blank
"	7-10	statement number (assembler generated)
"	11	blank
"	12-15	location counter
"	16-21	blanks
"	22-71	source line

At the end of Pass 1, the symbol table is listed. The format of the listing is:

Columns	1- 4	symbol
"	5- 8	blanks
"	9-12	value of symbol

The format of the source listing printed during Pass 2 is:

Columns	1- 5	diagnostic message area
"	6	blank
"	7-10	statement number
"	11	blank
"	12-15	location
"	16	blank
"	17-20	data (4 hex digits)
"	21	blank
"	22-71	source line

APPENDIX E

OPERATING INSTRUCTIONS

A. General

The Wang 3300 Assembler (ASMB4, ASMB8, TCASMB) is a program which translates a symbolic source program tape into a form suitable for execution on the Wang 3300 and punches out this form on a binary object paper tape.

The four-bit version (ASMB4) will accept as input, paper tape source files in four-bit format only, while ASMB8 will accept only paper tape source files in eight-bit format. The cassette version (TCASMB) will accept as input only those cassette source files created by the cassette version of the Symbolic Tape Editor (TCEDIT).

The assembler generates the object tape in two or three 'passes' or readings of the symbolic source tape. During Pass 1, the first reading, the assembler builds a symbol table of the relative addresses of the (programmer supplied) tag names, and detects simple syntax errors in the source program. During Pass 2 and Pass 3, the actual assembly (translation) takes place. The difference in these two passes is that Pass 2 lists the source text, object code, the statement number, and further error diagnostics on the teletype, while Pass 3 punches a binary object tape. For that reason, and since Pass 3 ignores errors completely, it is desirable to run Pass 2 before running Pass 3, even though it is possible to skip Pass 2.

B.1. Preparation of the Symbolic Source Paper Tape

A symbolic source paper tape (an ASCII coded paper tape of the symbolic source program) must initially be generated for use as input to the Wang 3300 Assembly program. This symbolic tape is generated at the same time as the program is typed on the teletype simply by turning the paper tape punch to "ON", while the teletype machine is off-line (on "LOCAL"). The symbolic tape must follow the format rules specified in the Assembler Manual.

After each line, a combination of "carriage return" and "line feed" and "rubout" and "rubout" (CR/LR/FF/FF) must be typed. Additional CR/LF's will be ignored by the assembler but individual carriage returns or line feeds will not be ignored. One source line, therefore, is defined as the character string between CR/LF's. This line should not exceed 50 characters in length including blanks. Characters in excess of this length will be read and ignored.

Leader/trailer code is an important part of the input tape and should be punched "before" and "after" the symbolic code on the tape. The assembler will accept two types of leader/trailer code:

1) Rubout characters (all 8 holes punched in the tape--an X'FF'.). This makes an acceptable leader/trailer code (to the machine) but it is physically very easy to tear and cannot be written on.

2) A much more suitable code is generated by pushing the "HERE IS" button down two or three times. (Each 'HERE IS' punches 20 blank frames.)

This symbolic tape is the only tape that should be used as input to the assembler and forms the input for all three passes of the assembly program.

It is generally more convenient to prepare new source tapes and edit old source tapes with the Wang 3300 Symbolic Editor Program (TTYEDIT). With this program, blocks of lines can be typed or read from tape and modified before they are punched.

B.2. Preparation of the Symbolic Source Cassette Tape

A 3321 cassette tape containing one or more symbolic source files must initially be prepared using the Tape Cassette Symbolic Editor program (TCEDIT). Instructions in the use of TCEDIT are found in the section entitled WANG 3300 SYMBOLIC EDITOR.

C.1. Use of the ASMB4 and ASMB8 Assembler Programs to Produce an Object (Binary) Tape

A summary of instructions for operation of ASMB4 and ASMB8 follows in a flow diagram (Figures 1 and 2, page 110).

Notes on specific operations referred to in Figures 1 and 2.

a) Load ASMB4 (ASMB8). See Wang 3300 Object Tape Loading Procedure.

Remember ASMB4 is 4-bit and must be loaded via TTLOAD; ASMB8 is 8-bit and must be loaded via TTYLOAD. (TTYLOAD or TTLOAD should be put in the last page of available core, xF40-- x = 0 for 4K machine, x = 1 for 8K machine, etc.).

b) Instructions regarding the paper tape punch refer to the buttons on the punch console on the teletype. For the reader, it is assumed that there is a three-position switch, FREE/STOP/START. When running the assembler, the switch should be set to the "STOP" position; the tape should not be started manually.

c) Mount the source tape on the teletype tape reader.

d) Start ASMB at location 0200. That is,

1) Turn the teletype to 'ON-LINE'; turn the punch 'OFF'.
2) Depress ENTER on the computer console.
3) Set up 02 on the Data Entry Switches, i.e.,

and press the B button.

4) Set up 00 on the Data Entry Switches, i.e.,

and press the C button.

5) Depress RUN.
6) Press CLEAR.
7) Press GO.

8) For ASMB8 only, press '1' on the teletype for Pass 1.

e) The source tape will now be read. At the end of Pass 1, the user should remount the source tape.

For ASMB4:

Press GO. Press '2' on the teletype for Pass 2 (Pass 2 is optional).

For ASMB8:

Press '2' on the teletype for Pass 2 (Pass 2 is optional).

f) At the end of Pass 2, remount the source tape.

For ASMB4:

Turn the paper tape punch 'ON'. Press GO. Press '3' on the teletype for Pass 3.

For ASMB8:

Press '3' on the teletype for Pass 3 and follow the punching instructions whenever they are typed out.

g) Assembling another source tape:

For ASMB4:

Follow steps c) through f).

For ASMB8:

Mount the source tape. Press '1' on the teletype for Pass 1. Follow step e) through f) for ASMB8.

C.2. Use of the TCASMB Assembler Program to Produce an Object (Binary) Tape

A summary of instructions for the operation of TCASMB follows in a flow diagram (Figure 3, page 112).

Notes on specific operations referred to in Figure 3.

a) Load TCBOOT. See Wang 3300 Object Tape Loading Procedure.

b) Load TCASMB. See Wang 3300 Object Tape Loading Procedure.

c) Instructions regarding the paper tape punch refer to the buttons on the punch console on the Teletype.

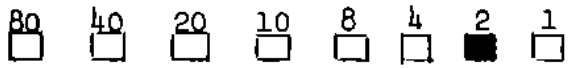
d) Mount the source cassette on either port of the 3320 unit.

e) Start TCASMB at location 0200. That is,

1) Turn the Teletype to 'ON-LINE'; turn the punch 'OFF'.

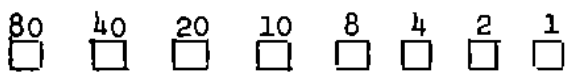
2) Depress ENTER on the computer console.

3) Set up 02 on the Data Entry Switches, i.e.,



and press the B button.

4) Set up 00 on the Data Entry Switches, i.e.,



and press the C button.

5) Press RUN, CLEAR, and GO.

f) The TCASMB program will type a series of questions on the Teletype. The following example illustrates these questions. Boxed-in characters represent responses typed on the Teletype by the user. There are five possible passes which can be performed by TCASMB: these are indicated by user responses of "0", "1", "2", "3", and "4".

Pass 0: Pass 0 initializes the assembler. Both the ordinary symbol table and the common symbol table are removed. The system responds with a request for new input tape and file numbers.

Pass 1: Pass 1 reads the indicated source input file and forms the ordinary symbol table and the common symbol table if one is present in the user's program.

Pass 2: Pass 2 reads the indicated input source file and prints the output listing (APPENDIX D).

Pass 3: Pass 3 reads the indicated input source file and punches the object tape.

Pass 4: Pass 4 initializes everything but the common symbol table if one is present. It then requests tape and file numbers for a new input source file. When these have been supplied, the system automatically reads the indicated file and performs pass 1 on that file.

EXAMPLE

```
WANG 3300 ASSEMBLER--INPUT:
TAPE/FILE? 2/3
PASS? 1
STRT    002A
BEG     0032
ZER     0042
LIM     0044
ADDR    0046
CNT     0048

PASS? 2

                    *
                    * SAMPLE PROGRAM
                    *
0004 002A 0000      ORG  X'002A'
0005 002A B242 STRT DL   ZER
0006 002C BA48      DU   CNT
0007 002E BA46      DU   ADDR
0008 0030 0C08      ONS  08
0009 0032 1A02 BEG  LAI  X'02'
0010 0034 AA46      DAM  ADDR
0011 0036 4249      INC  CNT+1
0012 0038 B244      DL   LIM
0013 003A A248      DCM  CNT
0014 003C 2532      JNE  BEG
0015 003E B246      DL   ADDR
0016 0040 002A      HLTJ STRT
0017 0042 0000 ZER  DC   X'0000'
0018 0044 0064 LIM  DC   X'0064'
0019 0046 0000 ADDR DS   2
0020 0048 0000 CNT  DS   2
0021 004A 0000      END
```

```
PASS?3

PUNCH PHASE
SET TAPE READER TO FREE
TURN TAPE PUNCH ON
PRESS CARRIAGE RETURN

*2B:H:F*FBH2D"H%22F*D#
PASS?4
TAPE/FILE? 2/2
ZER     0000
LIM     0002
NUM     0004
"       0005

PASS?0
TAPE/FILE? 2/2

PASS?
```

D. Error Facility

Error diagnostics appear as a 1-letter code immediately preceding the statement number in which the error occurs. A complete table of the error diagnostics is found in the Assembler Manual (APPENDIX C, page 101). Some errors will occur only in Pass 1, some only in Pass 2, and some in both passes. The message I/O ERR = E occurs only TAPE/FILE? with TCASMB when the "ESC" key is struck.

E. Interrupting Execution of TCASMB

The user can interrupt TCASMB during any phase of its operation simply by pressing the "ESC" key on the Teletype. TCASMB will respond with the message:

I/O ERR = E
TAPE/FILE?

FIGURE 1

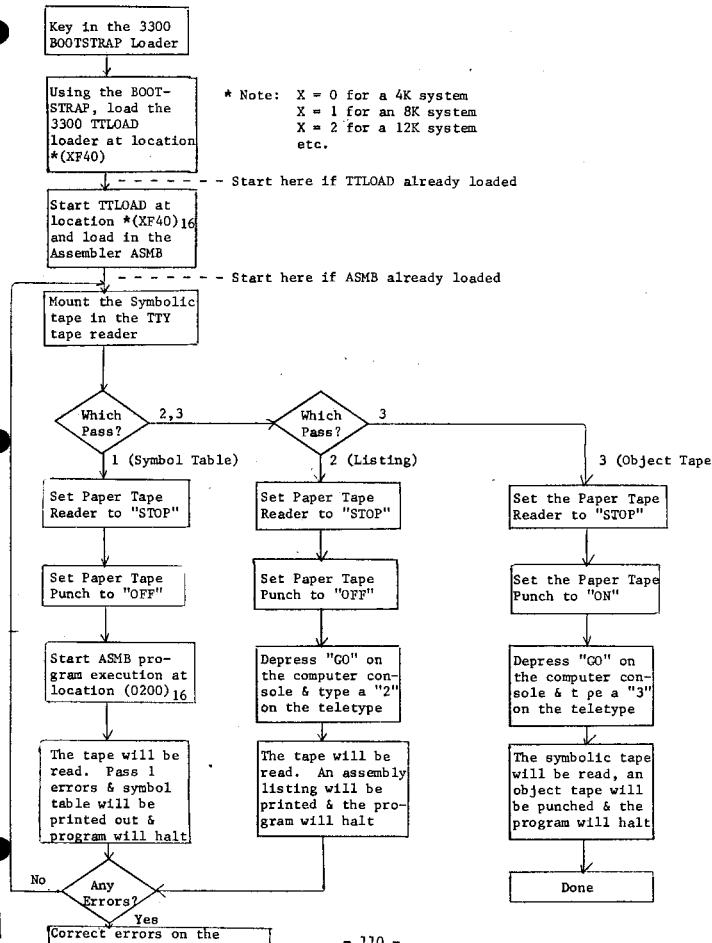

Key in the 3300 BOOTSTRAP Loader
Using the BOOT-STRAP, load the 3300 TTLOAD loader at location *(XF40)
* Note: X = 0 for a 4K system
X = 1 for an 8K system
X = 2 for a 12K system
etc.
Start here if TTLOAD already loaded
Start TTLOAD at location *(XF40)16 and load in the Assembler ASMB
Start here if ASMB already loaded
Mount the Symbolic tape in the TTY tape reader
Which Pass?
2,3
Which Pass?
3
1 (Symbol Table)
2 (Listing)
3 (Object Tape)
Set Paper Tape Reader to "STOP"
Set Paper Tape Reader to "STOP"
Set the Paper Tape Reader to "STOP"
Set Paper Tape Punch to "OFF"
Set Paper Tape Punch to "OFF"
Set the Paper Tape Punch to "ON"
Start ASMB program execution at location (0200)16
Depress "GO" on the computer console & type a "2" on the teletype
Depress "GO" on the computer console & t pe a "3" on the teletype
The tape will be read. Pass 1 errors & symbol table will be printed out & program will halt
The tape will be read. An assembly listing will be printed & the program will halt
The symbolic tape will be read, an object tape will be punched & the program will halt
No
Any Errors?
Yes
Done
Correct errors on the symbolic tape; start again

FIGURE 2

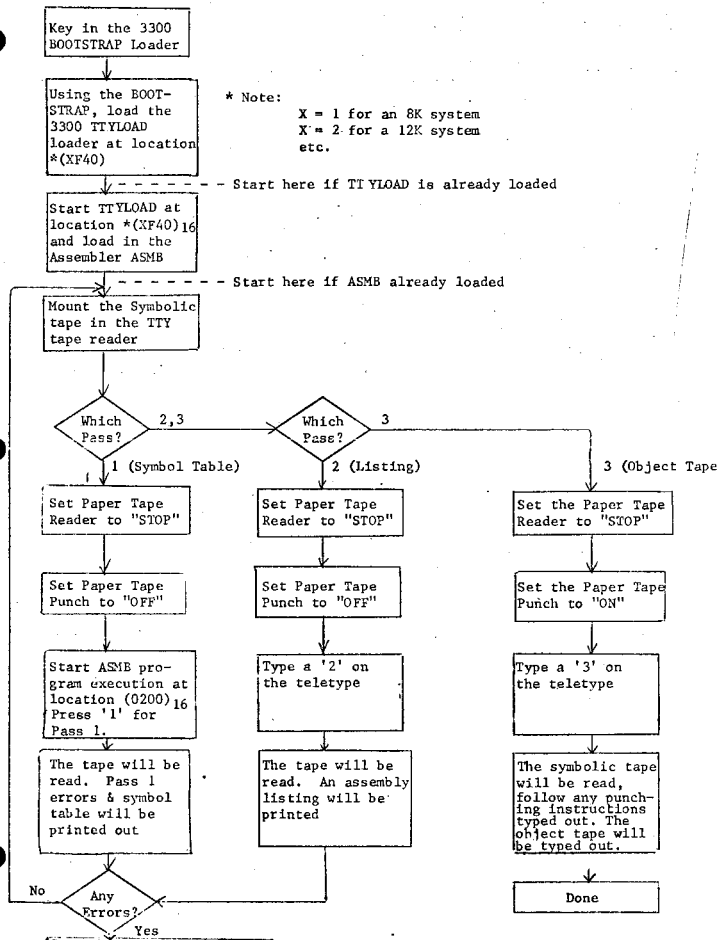
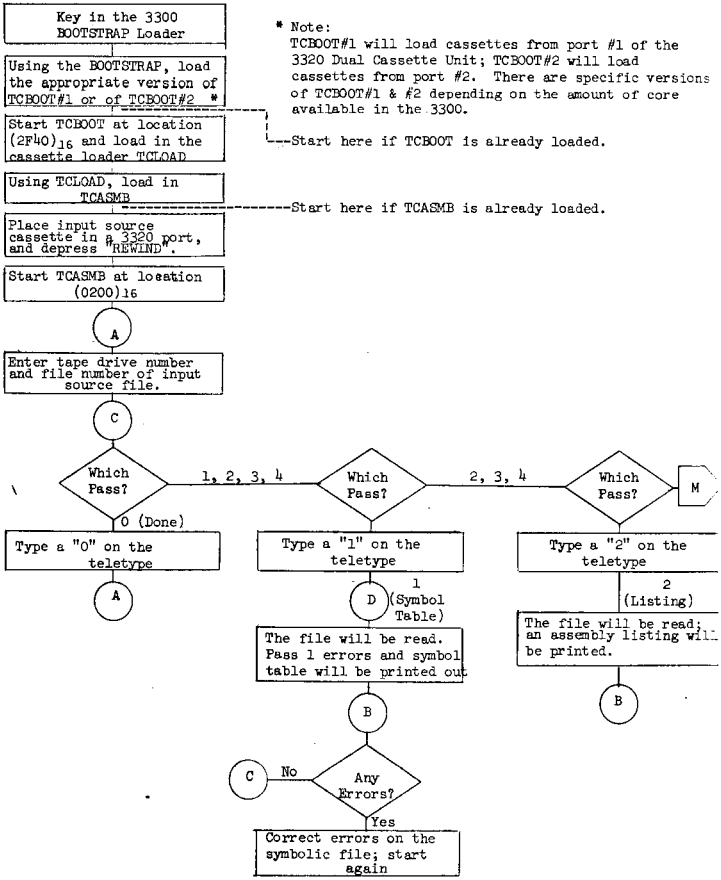
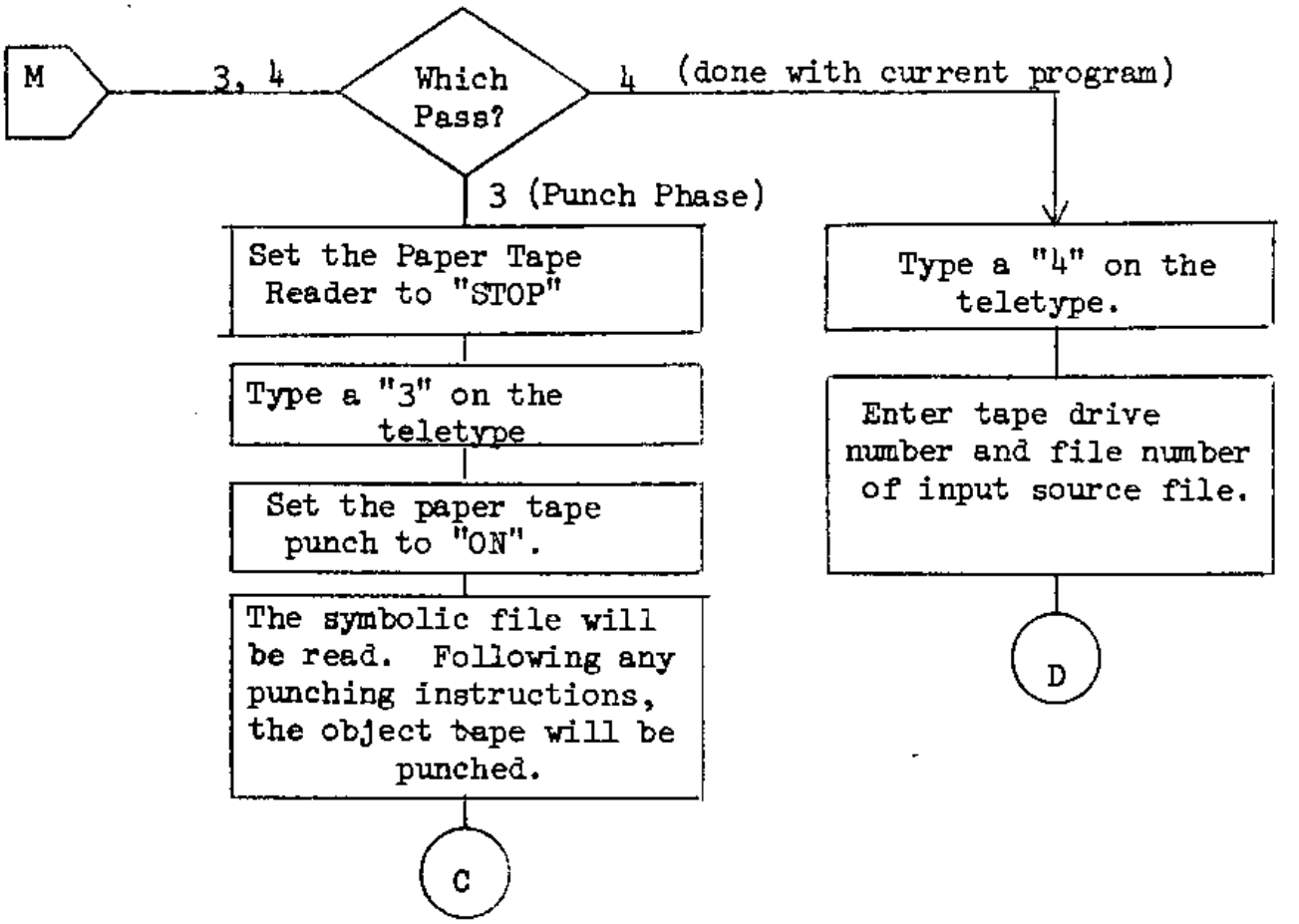

Key in the 3300 BOOTSTRAP Loader
Using the BOOT-STRAP, load the 3300 TTYLOAD loader at location *(XF40)
* Note:
X = 1 for an 8K system
X = 2 for a 12K system
etc.
Start here if TTYLOAD is already loaded
Start TTYLOAD at location *(XF40)16 and load in the Assembler ASMB
Start here if ASMB already loaded
Mount the Symbolic tape in the TTY tape reader
Which Pass?
2,3
Which Pass?
3
1 (Symbol Table)
2 (Listing)
3 (Object Tape)
Set Paper Tape Reader to "STOP"
Set Paper Tape Reader to "STOP"
Set the Paper Tape Reader to "STOP"
Set Paper Tape Punch to "OFF"
Set Paper Tape Punch to "OFF"
Set the Paper Tape Punch to "ON"
Start ASMB program execution at location (0200)16 Press '1' for Pass 1.
Type a '2' on the teletype
Type a '3' on the teletype
The tape will be read. Pass 1 errors & symbol table will be printed out
The tape will be read. An assembly listing will be printed
The symbolic tape will be read, follow any punching instructions typed out. The object tape will be typed out.
No
Any Errors?
Yes
Done
Correct errors on the symbolic tape; start again

FIGURE 3


Key in the 3300 BOOTSTRAP Loader
Using the BOOTSTRAP, load the appropriate version of TCBOOT#1 or of TCBOOT#2 *
Start TCBOOT at location $(2F40)_{16}$ and load in the cassette loader TCLOAD
Using TCLOAD, load in TCASMB
Place input source cassette in a 3320 port, and depress "REWIND".
Start TCASMB at location $(0200)_{16}$
A
Enter tape drive number and file number of input source file.
C
Which Pass?
1, 2, 3, 4
0 (Done)
Type a "0" on the teletype
A
Which Pass?
2, 3, 4
Type a "1" on the teletype
1 (Symbol Table)
D
The file will be read. Pass 1 errors and symbol table will be printed out
B
C
No
Any Errors?
Yes
Correct errors on the symbolic file; start again
Which Pass?
M
Type a "2" on the teletype
2 (Listing)
The file will be read; an assembly listing will be printed.
B
* Note:
TCBOOT#1 will load cassettes from port #1 of the 3320 Dual Cassette Unit; TCBOOT#2 will load cassettes from port #2. There are specific versions of TCBOOT#1 & #2 depending on the amount of core available in the 3300.
Start here if TCBOOT is already loaded.
Start here if TCASMB is already loaded.


M
3, 4
Which Pass?
4 (done with current program)
3 (Punch Phase)
Set the Paper Tape Reader to "STOP"
Type a "3" on the teletype
Set the paper tape punch to "ON".
The symbolic file will be read. Following any punching instructions, the object tape will be punched.
C
Type a "4" on the teletype.
Enter tape drive number and file number of input source file.
D

WANG 3300

CHECKOUT ASSISTANCE PROGRAM

(CAP)

Revised: January 4, 1972

TABLE OF CONTENTS

ABSTRACT

CAP is an on-line debug program for the WANG 3300 System. It provides assistance to the user in checking out his machine language program while using a minimum amount of memory. CAP allows the user to examine and modify both the registers and memory, to insert breakpoints, to search memory, and to write paper tapes.

I. INTRODUCTION.

The Checkout Assistance Program (CAP) is one of the standard software modules of the WANG 3300 digital computer system. CAP is a source program which allows the user to check out his machine language program with the aid of his teletype. Core storage may be examined and modified, and program execution may be interrupted by the user's insertion of breakpoints.

There are both 4-bit and 8-bit versions of CAP. The 4-bit version is for the 4K, 4-bit Assembler Package; it is origined at X'0900'. The 4-bit version must be loaded with the 4-bit TTLOAD; it punches tapes in 4-bit format. The standard 8-bit version of CAP is for the 8K, 8-bit Assembler Package; it is origined at X'1900'. Versions of the 8-bit CAP origined at X'0900', X'2900', X'3900', X'4900', and X'5900' are available on special request.

CAP is loaded into locations X'x900' through X'xF3F'. Locations X'xF40' through X'xFFFF' are reserved for the LOADER. The program to be checked out (the user's program) is loaded elsewhere in memory. Control is transferred to CAP which then begins a dialogue with the user. Control is released to the user program by the JT (jump to) command. Control is returned to CAP only if a breakpoint halt is encountered. A halt or loop in the user program requires the user to manually transfer to CAP.

The user program must observe certain restrictions:

1. The user program must fit into memory without using any of the space reserved by CAP. CAP uses locations X'x900 through X'xF3F' plus locations FE and FF in each of the absolute pages.

2. The user must be cautious when checking out programs that contain I/O interrupts. When initiated from start location, CAP disables the I/O interrupt and, thereafter, uses the teletype I/O on a non-interrupt basis, assuming disabled interrupt.

II. CONVERSATIONAL RULES.

The user communicates with CAP via the teletype; all commands are entered via the keyboard, and all outputs are printed there. When CAP is ready to accept the next user command request, it types a colon (:) and waits for a user command entry. A user response is

always terminated by a carriage return, which causes CAP to ignore any further input until the command has been serviced. When the requested command has been performed, CAP issues another colon (:) and the user may make another entry.

In general, once a valid command has been entered, it cannot be removed by the user. The exception to this is during the display or search of memory, or a breakpoint/continue type of trace. Striking the ESCape key during these operations terminates the listing in progress and returns control to CAP.

CAP commands consist of two letters followed by arguments. All arguments of CAP are hexadecimal numbers. The following terminology will be used for this section:

A 'byte' is an argument of two hexadecimal digits.

An 'address' is an argument of one to four hexadecimal digits.

A 'value' is an argument of even number of hexadecimal digits.

If CAP detects an error, it will space over to the digit in error and type a ":", followed by the work "ERROR", followed by an error number. For example, the user typed the following:

```
:XZ
  :ERROR 4          would be CAP's reply
                    (e.g., means an invalid command)
```

III. FUNCTIONAL FEATURES.

CAP performs several different classes of functions for the user:

- examination of memory or registers
- modification of memory or registers
- insertion or deletion of breakpoints
- searching memory for a specified value
- transferring control to user program
- writing of paper tape

A. Examination of Memory or Registers (XM, XR)

The XM (examine memory) command allows the user to display the value of one or more bytes. The format of the XM command is:

```
:XM 'address 1' [, 'address 2' ]
```

The contents of memory from 'address 1' to 'address 2' are displayed. If 'address 2' is not supplied, the byte at 'address 1' is displayed.

For example, if the user typed the following followed by a carriage return, CAP would display memory from 03AC to 1257 (see Appendix B for memory display format).

```
:XM 03AC,1257
```

The XR (examine registers) command allows the user to display the value of the Z, A, and S registers at the time CAP was entered. The format of the XR command is the following:

```
:XR
```

(See Appendix B for register display format)

B. Modification Functions (CM, CA, CZ, CS, MP)

The CM (change memory) command allows the user to modify the value of one or more bytes of memory. The format of the CM command is:

```
:CM 'address', 'value'
```

The contents of memory beginning at location 'address' are changed to 'value'. One or more bytes are changed depending on the length of 'value'. If 'value' is two hexadecimal digits, one byte is changed; if 'value' is four hexadecimal digits, two bytes are changed, etc.

For example, if the user types the following followed by a carriage return, CAP changes the contents of location 1CFE to 2F, 1CFF to 34, etc. (note: spaces are ignored):

```
:CM 1CFE, 2F34 6A7C
```

After the changes to memory have been made, CAP types out the changed memory locations in the format of the XM command.

On breakpoint entrance, CAP saves the Z, A, and S registers and, when returning via JT to the user program, restores these registers. The CZ (change Z register) command allows the user to change the Z register save area. When CAP returns to the user program, it restores the value of Z. The format of the CZ command is:

```
:CZ 'byte'
```

For example, if the user types the following followed by a carriage return, when CAP returns to the user program, register Z contains the value 2F:

```
:CZ 2F
```

Similarly, the CA (change A register) command and the CS (change S register) command have both format and function similar to the CZ command.

The MP (move page command) allows the user to move the contents of one page (256 bytes) of memory to another page of memory. The format of the MP command is:

```
:MP 'page address 1' , 'page address 2'
```

For example, the following would transfer the contents of locations X'0200' through X'02FF' to locations X'1900' through X'19FF'. (Note, the contents of locations X'0200' through X'02FF' are not modified.)

```
:MP 0200, 1900
```

C. Breakpoint Control (BH, BC, BR)

Breakpoints are used to interrupt the execution of a user program and control is passed to CAP. An active breakpoint in the user program is a JST* FE instruction (location FE contains the address of CAP's breakpoint entry). Breakpoints are inserted in the user program by BH (breakpoint halt) commands or BC (breakpoint continue) commands.

A breakpoint has two states:

1. An active state, where the breakpoint instruction exists in the user program.
2. An inactive state, where the breakpoint does not exist in the user program but is in the CAP breakpoint table.

The following occurs when a user program reaches a breakpoint:

1. Control is passed to CAP.
2. The program counter and Z, A, and S registers are displayed. If a memory dump is requested with the breakpoint, it is printed.
3. All inactive breakpoins are made active (the user instruction is saved and replaced with the breakpoint instruction).
4. If the enter breakpoint was established through a BC (breakpoint continue) command, it is made inactive, i.e., the breakpoint instruction is replaced with the original

user instruction. Control is now passed back to the user program at the breakpoint location. This, in effect, produces a selective trace. Note: If there is only 1 breakpoint in a program and it was inserted by a BC command, it will only be active once.

5. If the breakpoint was established through a BH (breakpoint halt) command, the breakpoint remains active, i.e., the breakpoint instruction is not removed and the user command mode is entered.

The format of BC (breakpoint continue) command is:

:BC 'address' [, 'address 1' [, 'address 2']]

The user instruction at location 'address' is saved and a JMP* FE instruction inserted in its place. The breakpoint is saved by CAP in its place. The breakpoint is saved by CAP in the breakpoint table with the breakpoint continue designation. (CAP allows up to 10 breakpoints.)

The 'address 1' and 'address 2' are optional and, if they are entered, a dump of memory will be printed from locations 'address 1' to 'address 2' when the breakpoint is executed.

The format of BH (breakpoint halt) command is:

:BH 'address' [, 'address 1' [, 'address 2']]

The function of the BH inatruction is similar to the BC command, except the breakpoint is always active and control is not passed back to the user program. That is, the breakpoint instruction in the user program is never restored, except explicitly by a BR (breakpoint restore) command. The 'address 1' and 'address 2' are optional and specify the range of a breakpoing dump similar to BC. CAP restores a breakpoint by replacing the user instruction and deleting the breakpoint from the breakpoint table.

The format of the BR (breakpoint restore) command is:

:BR ['address']

The breakpoint at location 'address' is restored. The user instruction is replaced and the breakpoint is removed from the breakpoint table. If a BR command is issued without an 'address' argument, all breakpoints are restored.

D. Memory Searching (SB, SW)

The SB (search byte) and SW (search word) commands allow the user to search core for one or two bytes, respectively, with a given value. The search starts and ends at user specified locations.

The format of an SB (search byte) command is:

```
:SB 'byte' , 'address 1' , 'address 2'
```

The format of an SW (search word) command is:

```
:SW 'word' , 'address 1' , 'address 2'
```

A successful search results in the display of all locations with the specified value.

E. Control Release (JT)

The JT (jump to) command allows the user to transfer control from CAP to a specified location in the user program. The user program maintains control until a breakpoint is encountered. The breakpoint transfers control back to CAP. A user program loop or halt requires a manual transfer of control back to CAP.

The format of a JT (jump to) command is:

```
:JT 'address'
```

Control is transferred to the location 'address' in the user program.

F. Paper Tape Operations (WR)

The WR (write command) allows the user to write a record from core onto tape.

The format of the WR command is:

```
:WR 'address' , 'address'
```

A standard core image logical record is written from the first 'address' to the second 'address'; the generated paper tape may be loaded with the Wang 3300 Teletype LOADER. (8-bit for the 8-bit CAP, 4-bit for 4-bit CAP.)

APPENDIX A

CAP COMMAND REPERTOIRE

Below are given the legal formats for all CAP commands. All commands are two letters followed by additional arguments.

Special definitions used in the below description are:

'byte'	Two hexadecimal digits.
'address'	One to four hexadecimal digits which are less than the starting location of CAP.
'value'	An even number of hexadecimal digits.
'word'	Four hexadecimal digits.
'pg. address'	One to four hexadecimal digits which specify the start of a page.

Brackets indicate optional arguments.

BC -- Breakpoint Continue Command

Format: BC 'address' [, 'address 1' [, 'address 2']]

Function: Insert breakpoint continue at location 'address'. Optionally, a memory dump is made from 'address 1' to 'address 2' (if they are entered) when the breakpoint occurs.

BH -- Breakpoint Halt Command

Format: BH 'address' [, 'address 1' [, 'address 2']]

Function: Insert breakpoint halt at location 'address'. Optionally, a memory dump is made from 'address 1' to 'address 2' (if they are entered) when the breakpoint occurs.

BR -- Breakpoint Restore Command

Format: BR ['address']

Function: Restore breakpoint at location 'address'; if no 'address' is apecified, all breakpoints are restored.

CA -- Change A-Register Command

Format: CA 'byte'

Function: Change the A-register to the value of 'byte'.

CM -- Change Memory Command

Format: CM 'address' , 'value'

Function: Change the contents of memory beginning at location 'address' to 'value'.

CS -- Change S-Register Command

Format: CS 'byte'

Function: Change the S-register to the value of 'byte'.

CZ -- Change Z-Register Command

Format: CZ 'byte'

Function: Change the Z-register to the value of 'byte'.

JT -- Jump to User Command

Format: JT 'address'

Function: Transfer control to the user program at location 'address'

MP -- Move Page Command

Format: MP 'pg. address 1' , 'pg. address 2'

Function: Move the contents of the page at 'pg. address 1' to the page specified by 'pg. address 2'.

SB -- Search Memory for Byte

Format: SB 'byte' , 'address 1' , 'address 2'

Function: All memory locations with the value 'byte' starting with location 'address 1' to location 'address 2' are displayed.

SW -- Search Memory for Word

Format: SW 'word' , 'address 1' , 'address 2'

Function: All memory locations with the value 'word' starting with location 'address 1' to location 'address 2' are displayed.

WR -- Write Record to Paper Tape

Format: WR 'address 1' , 'address 2'

Function: A standard core image logical record is written on the paper tape from location 'address 1' to location 'address 2'.

XM -- Examine Memory

Format: XM 'address 1' [, 'address 2']

Function: Display the contents of the bytes of memory from location 'address 1' to 'address 2'.

XR -- Examine Registers

Format: XR

Function: Display the contents of the Z, A, and S registers.

APPENDIX B

CAP DISPLAY FORMATS

There are four display formats used by CAP:

1. Memory Display Format (after CM and XM commands)

```
AAAA - xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
BBBB -
etc.
```

AAAA is the start of the memory display.
BBBB is AAAA+10_{16} and the x's represent the contents of the memory locations in hexadecimal.

2. Register Display Format (after an XR command)

```
Z = xx  A = xx  S = xx
```

3. Breakpoint Display Format (after an XR command)

```
BP - AAAA  Z = xx  A = xx  S = xx
```

4. Address Display Format (after SB and SW commands)

```
AAAA
BBBB
etc.
```

AAAA and BBBB are the addresses where the searched for byte(s) can be found.