<u>U T I L I T I E S</u> <u>M A N U A L</u>

John Burnett & Charles Wagner
August, 1970

FAIRCHILD SYSTEMS TECHNOLOGY DIVISION

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

FIGURES

SECTION 1

UTILITY PROGRAMS

1.1 BASIC TAPE LOADER

The primary purpose of the Basic Tape Loader, BTL, is to load the Tape Object Loader. Since BTL must be entered into the memory manually, it is desirable that it be as short as possible. For this reason the format of the tape that BTL loads is quite specialized; any tape in this format is frequently referred to as a bootstrap tape.

The tape is a sequence of 8-bit characters. It takes four of these characters to form a single 24-bit word. The first character forms bits 5-0 of the word, the second forms 11-6, etc. The format of these 8-bit characters is:

BITS	DESCRIPTION
7,6	FLAGS
5-0	DATA

The flag value for the first, second and third characters loaded must be '00'. For all but the last word to be loaded, the fourth character must have the flag value '11' or '01'. The last word to be loaded must have the flag value '10' for its fourth character.

The first word loaded from a BTL formatted tape must have the value:

```
BRU  xxxxx
```

'xxxx' is the address BTL will transfer control to when loading is complete. The second word must have the value:

```
DATA  yyyy
```

'yyyy' is the address of the first word to be loaded -1. That is, the third word on the tape will be loaded at yyyy+1, the fourth at yyyy+2, etc.

Note that it is possible to load any program that adheres to these conventions. The program BTLTG (1.6) will convert any object tape to a format acceptable to BTL. The octal representation of BTL is:

LOC	CODE	(INTERPRETATION)
07760	05007772	(LDX 0)
07761	16007773	(STX 0)
07762	06020420	(FEED PTR)
07763	06601420	(RDTT)
07764	03147763	(BOI LT or bit equal, i.e. BUSY or Not Available)
07765	07032006	(LDS)
07766	02207762	(BAT A=0, i.e. BZ)
07767	15047773	(STE* indirect)
07770	36007773	(AOM)
07771	02047762	(BAT A=odd, i.e. BO)

1.2 TAPE OBJECT LOADER

The paper tape object loader, TOL, can be used to load paper tape programs when no relocation is required. TOL ignores record types 0 and 3[1] so that programs that were assembled relocatable will be loaded as if they had been assembled absolute.

The type 2 record[1] will cause interrupt locations to be loaded and, like all the other loaders, is used to specify the entry point of the loaded program. The latter is done by a type 2 record[1] whose identifier is MAINxx; xx may be anything. Failure to find such a record will result in an unpredictable transfer when loading is complete. Finding more than one causes the last encountered to be used.

TOL is loaded by BTL and occupies locations 101_8 thru 177_8. It may be re-entered for further loading by executing a BRU 102B providing it has not been destroyed.

1.2.1 TOL Operating Instructions

BTL (1.0) must be present in core memory.

1. Place TOL bootstrap tape in teletype paper tape reader. Move switch to START.
2. Lift SIC switch.
3. Enter 01007760_8 into CPU switch register.
4. Lift LDC switch.
5. Push SIC switch to the down position.
6. Push START to load TOL.

The execution of TOL is shown in the following flow chart:

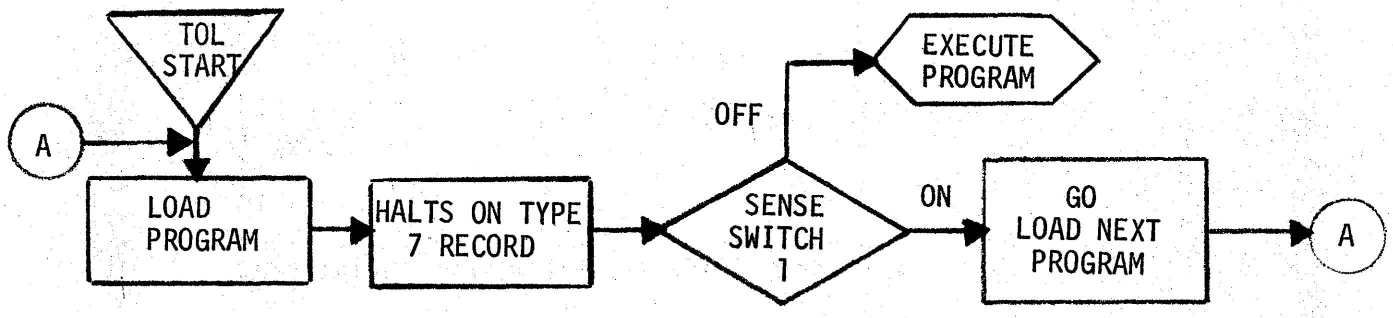

Figure 1.2.1
TOL Execution

[1] All record formats of records output by the DOPSY Assembler are explained in the DOPSY Assembler User's Manual.

TOL allows multiple assemblies to be loaded by halting when a type 7 record (END) is encountered. When START is pressed, the setting of switch 1 determines whether to continue loading or to enter the loaded program.

1.3 BASIC CARD LOADER

The Basic Card Loader, BCL, is used for loading the Card Object Loader, COL, into core. BCL is comprised of one alpha mode card which contains code to read four other alpha mode cards into core locations 100B and following. When loading of these cards is completed, BCL transfers control to location 114B. The contents of the BCL card is given in Appendix D.

The bootstrap procedure for using BCL and COL is as follows:

1. Place the binary bootstrap card (Appendix B), BCL, and COL in the card reader followed by the object deck to be loaded, and press the START button on the card reader. The card reader READY light should become lighted.

2. Lift the SIC switch on the CPU.

3. Enter 01000100_8 into the CPU switch register.

4. Lift the LDC switch.

5. Return the SIC switch to down position.

6. Press LOAD CDR and START to initiate actual loading of BCL, COL and the object program.

Appendix A gives the 029 characters and the octal code produced when read in the BCD mode. All BCD cards associated with BCL should be punched according to this table.

1.4 CARD OBJECT LOADER

The Card Object Loader, COL, is a six-card program that can be used to load card object programs when no relocation is required. COL ignores record types 0 and 3[1] so that programs that were assembled relocatable will be loaded as if they had been assembled absolute.

The type 2 record[1] will cause interrupt locations to be loaded and, like all the other loaders, is used to specify the entry point of the loaded program. The loading of interrupts, however, is limited to those in the range of 1 through 47_8. The entry point is specified by a type 2 record[1] whose identifier is MAINxx; xx may be anything. Failure to read such a record will result in an unpredictable transfer when loading is complete, while reading more than one causes the last one read to be used.

[1]All record formats of records output by the DOPSY Assembler are explained in the DOPSY Assembler User's Manual.

The loaded program is entered when a type 7 record[1] (END) is read. If multiple assemblies are to be loaded, all but the last must have their type 7 records[1] removed.

The format of card object programs loaded by COL is octal; that is, it requires eight columns for one 24-bit word. The maximum number of words loaded per card is eight. (Columns 17 through 80)

COL is loaded by its first two cards (BCL). The actual COL program is contained in four cards, 16 words per card (64 columns). It loads into locations 100_8 - 177 and uses 50_8 through 74_8 as an input buffer. COL can be re-entered for further loading by executing a BRU 114B, providing it has not been destroyed.

1.4.1 COL Operation

The following diagram shows the format of a typical card deck loaded from the card reader by COL:

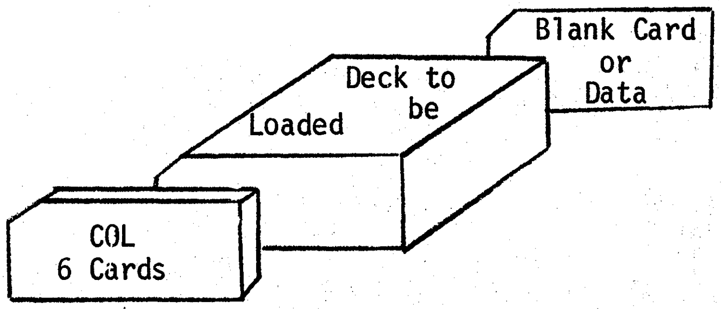

Figure 1.4.1
Card Deck Format Using COL

The detailed loading procedure is described in paragraph 1.3.

1.5 RELOCATING OBJECT LOADER

The relocating object loader, ROL, will load a collection of object programs into core memory. The source of these programs may be paper tape and/or cards; the programs may be absolute or relocatable. In the latter case they are 'stacked' in memory one after the other and the necessary address modification is done to assure that the programs will run correctly. In the former case, however, the program is loaded in the same locations that it was assembled for; absolute loads in no way effect where following relocatable loads will be loaded. Absolute loads should be used with caution since overlaying a relocatable program could destroy some linked-list used by ROL in linking PROC and CALL directives.

[1]All record formats of records output by the DOPSY Assembler are explained in the DOPSY Assembler User's Manual.

The entry point to the main program is specified with a MAINPR PROC record. This is the program to which control is relinquished when loading is complete. If a DEBUG PROC record is encountered, however, control will go to DEBUG and the user will have to enter his program from the DEBUG procedure.

The calling sequence from the loader for the user's program is:

```
          CALL    BSM    MAINPR
                  ---               OVERLAY RETURN
                  ---               NEW LOAD RETURN
```

If the user does not CLEAR (see Operation) the loader or destroy it in his program, he can return to CALL+1 or CALL+2. Returning to CALL+1 saves the previous symbol table to allow overlays to use procedures from previous loads; returning to CALL+2 will initiate for a new, independent load.

ROL is loaded by COL, TOL or BTL and occupies locations 7000_8 to 7777_8. A symbol table is built beginning at 7000_8 and extends downward in memory.

1.5.1 <u>ROL Operation</u>

The following diagram shows the format of a typical card deck loaded from the CR by ROL.

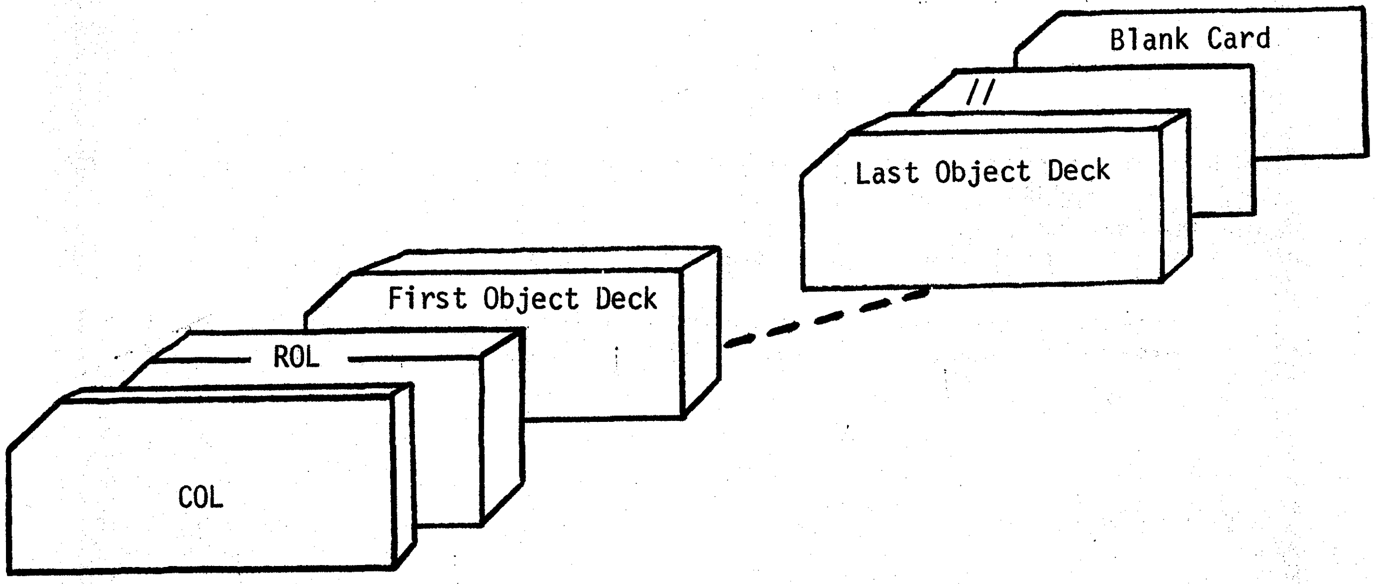

Figure 1.5.1.1

Card Deck Format Using ROL

To load the deck requires the same procedure as that for COL (see 1.3). ROL may also be loaded from paper tape either by TOL or BTL. The latter will load faster since the tape is shorter. See BTLTG (1.6) for converting the ROL object tape to the bootstrap format. See BTL (1.0) or TOL (1.1) for instructions on their usage.

The execution of the relocating loader is shown in the following flow chart.

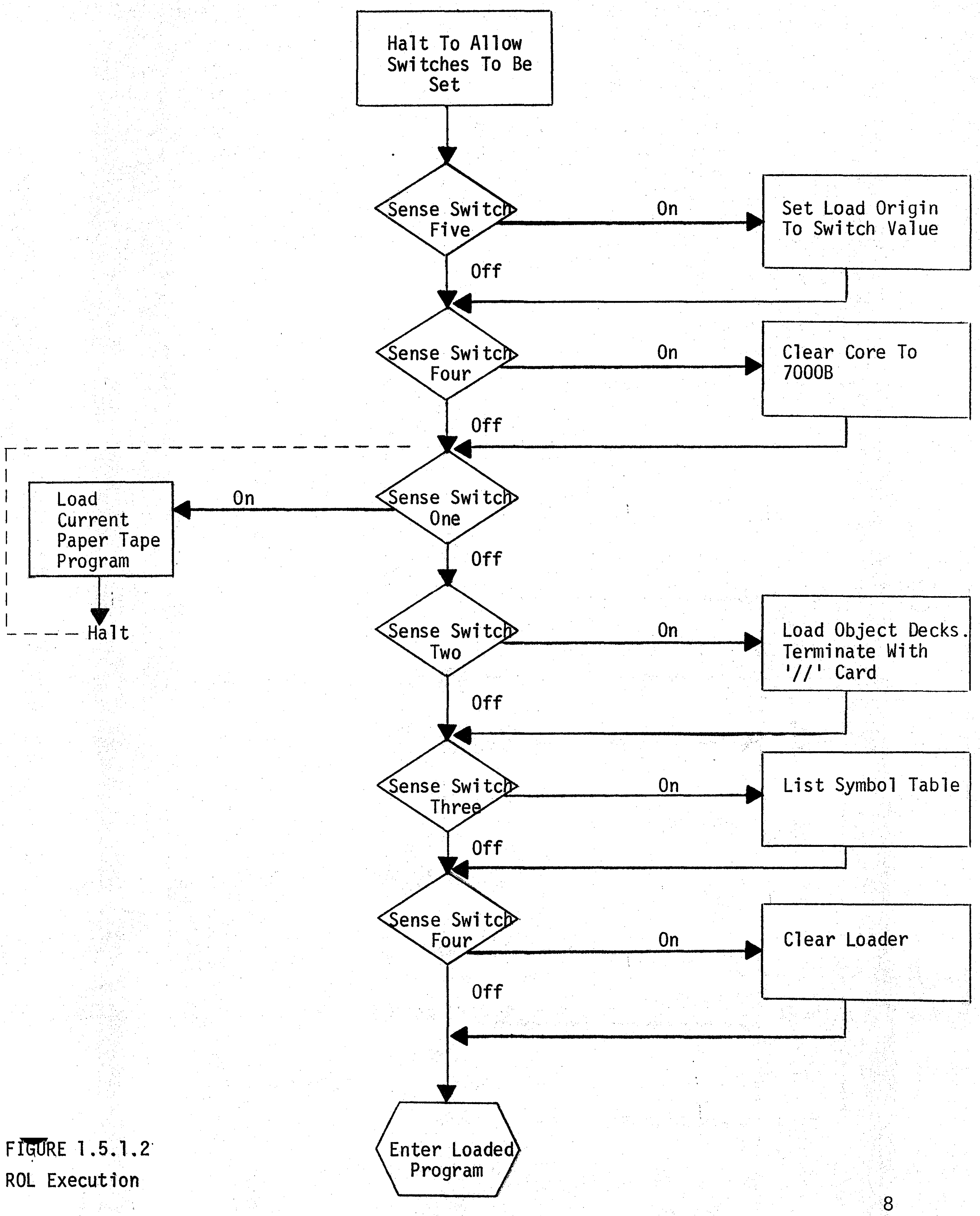

FIGURE 1.5.1.2
ROL Execution

1.5.2 Switch Settings

The use of the sense switches by ROL is shown in the following table:

Sense Switch	Meaning
1	Load paper tape.
2	Load cards.
3	List symbol table.
4	Clear core.
5	Loading origin is specified in console switch register; 200B is assumed otherwise.

1.5.3 Symbol Table Output

The first line of symbol table output is the address plus one of the highest location of the loaded program. This value is also transmitted to the loaded program in index register zero.

Subsequent lines contain the name of a called or declared PROC entry, followed by a code, followed by the address of the PROC entry. This line has the following format: SSSSSS X   AAAAAA

SSSSSS - the sumbol

Code X -	U	CALL with no corresponding PROC
	N	PROC with no corresponding CALL
	D	Duplicate PROC entry
	?	Combination of N and D
	0, 1, 2	ROL has operated incorrectly. Take a core dump and report condition to Field Service.

AAAAAA - Address of PROC entry.

If an undefined entry is called, the machine will halt. Note that the 'N' warning is frequently produced by PROC statements that are used to establish interrupt routine linkages and does not constitute an error.

1.5.4 Messages

TEXT	ACTION
PROGRAM TOO BIG[1]	Reload using overlay.
SEQUENCE ERROR	Card in error (will be reread when START is pressed).
PARITY ERROR	Reload necessary.
MAINPR MISSING	Reload necessary.

[1]Allowable program size depends on: a) loading point, b) number of external references and c) size of core.

1.6 BASIC TAPE LOADER TAPE GENERATOR

The basic tape loader tape generator (BTLTG) can be used to convert paper tape programs from standard assembler format to the BTL binary format (see section 1.0). BTLTG operates like TOL (see section 1.1); it loads a program to be converted and punches out a BTL formatted tape. Record types 0 and 3 are ignored; programs that were assembled relocatable will be loaded and converted as if they had been assembled absolute.

BTLTG does not load interrupts; it uses type 2 records only to establish the entry point to the program. The last occurrence of a type 2 record whose identifier is DEBUxx will be taken as entry point, if present; otherwise the last occurrence of a type 2 record whose identifier is MAINxx will be taken; if neither is present, the first type 2 record encountered will be taken.

BTLTG allows multiple assemblies to be loaded by halting when a type 7 record (END) is encountered. When START is pressed, switch one is tested; if it is ON, loading continues; if it is OFF, the program is punched out. If no type 2 records were present, the tape cannot be punched; BTLTG will halt with 70707070B in the A register. If type 2 records were present, but no type 1 records were loaded, BTLTG will halt with 07070707B in the A register. After punching is complete, BTLTG is ready to restart.

BTLTG is loaded by BTL and occupies locations 220B through 436B. It may be restarted by executing a BRU 221B providing it has not been destroyed. It will convert any program which does not load in the area it occupies. (BTLTG can be loaded in relocatable form to convert programs which load in the area it normally occupies).

1.6.1 BTLTG Operating Instructions

BTL (1.0) must be present in core memory.

1. Place BTLTG bootstrap tape in teletype paper tape reader. Move switch to START.
2. Enter 01007760_8 into CPU switch register. Press LDC.
3. Press START to load BTLTG. CPU halts when loaded.
4. Place paper tape to be converted in teletype paper tape reader. Move switch to START.
5. Set console switch 1 ON.
6. Press START to load tape. CPU halts when tape is loaded.
7. Repeat steps 4-6 for additional tapes if required. Then go to step 8.
8. Set console switch 1 OFF. Turn on teletype paper tape punch. Press START. Converted tape will be punched.

The execution of BTLTG is shown in the following flow chart:

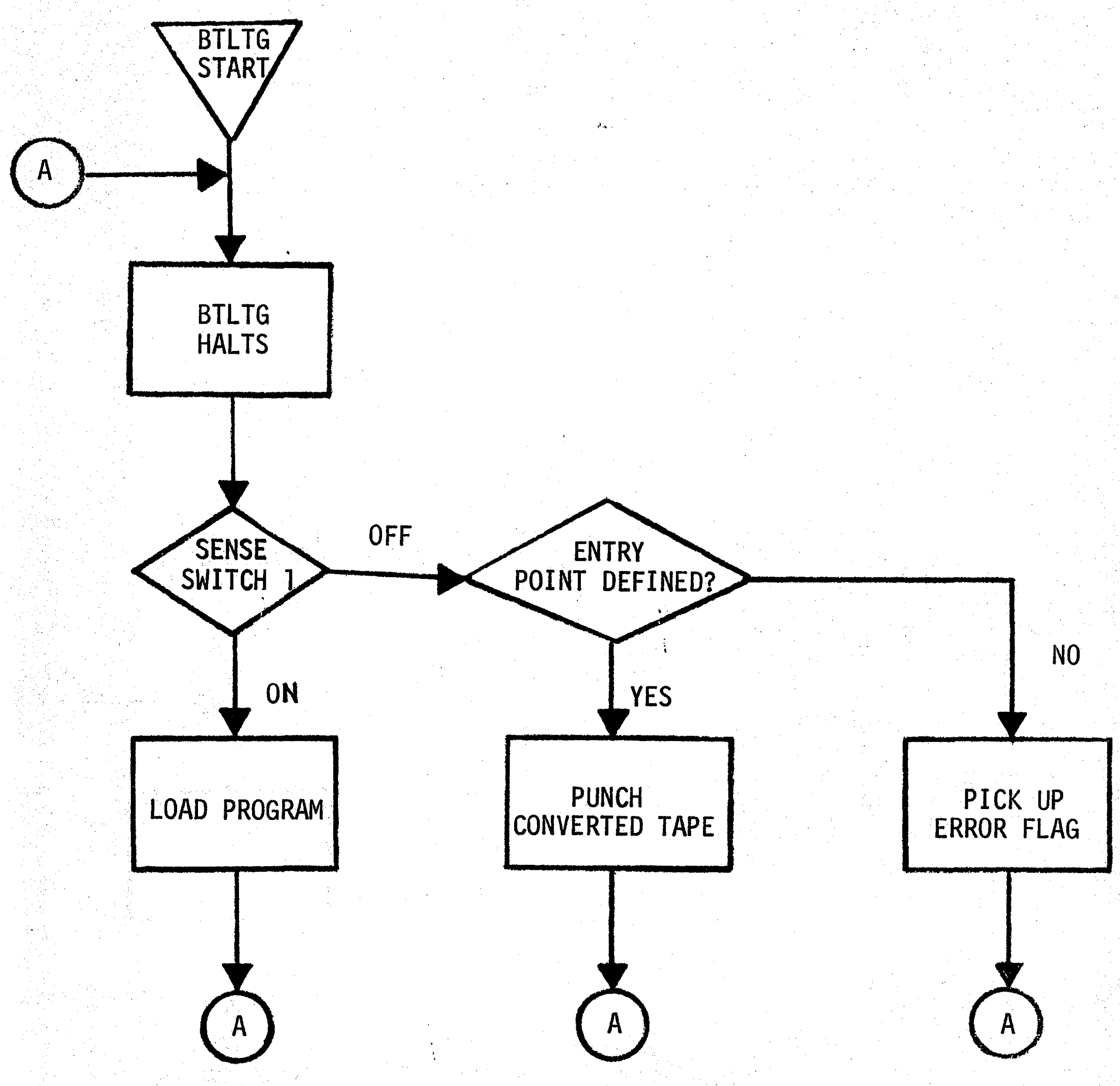

FIGURE 1.6.1
BTLTG Execution

1.7 DEBUG AID

The debug aid, DEBUG, is a procedure that a user can load with his program to aid him in testing and debugging. All information required by DEBUG must be entered through the teletype keyboard. The format and action of these commands is discussed after the following general information.

In discussing each of the commands the following notation is used:

CA means Current Address
(CA) means the contents of the Current Address

Whenever DEBUG is ready to accept a command, it types an '#'. The command will be accepted for processing when a command character is typed. Until this time a command may be cancelled by pressing the CTRL and L keys. Illegal characters cause a '?' to be typed and the command must be started over.

COMMAND CHARACTER	DESCRIPTION
↑	-- Types (CA-1) and sets CA = CA-1.
LF	-- Types (CA+1) and sets CA = CA+1.
C	-- This command is used after an ADDRESS STOP to resume execution of the stopped program. The first instruction executed is located at the STOP address plus one.
R	-- Types the contents of the A and E registers. After an ADDRESS STOP, these are the values in the registers after executing the instruction at the STOP address.
D	-- Returns control to the automatic restart routine for reloading the monitor.
N	-- NOTE - allows the user to type a message on the teletype. All printing characters can be typed; the CTRL L key combination terminates the command.
X	-- Types the contents of each of the eight index registers. After an ADDRESS STOP, these are the values in the registers after executing the instruction at the STOP address.

The following commands can be preceded by an address 'a'. If 'a' is absent the CA is used.

aG	-- Transfer control to the location addressed by 'a' and sets CA=A.
aE	-- Allows octal information to be entered at location 'a' and sets CA=a. In this mode the legal characters are CR, LF, CTRL L and the octal digits 0, 1, 2, 3, 4, 5, 6, and 7. Other characters type a '?' and allow the user to retype the information to be entered into the current location.

CR, LF and CTRL L terminate input as follows:

CR - sets (CA) to the value typed and initializes for a new command.
LF - sets (CA) to the value typed, sets CA = CA+1, and initializes to alter the contents of the new CA.
CTRL L - initializes for a new command.

The following commands may be preceded by a quantity of the form 'a, b'; either or both may be omitted. 'a' is an address and if absent is assumed to be CA, while the value assumed for an absent 'b' is dependent on the command.

1.	a,b	
2.	,b	CA assumed
3.	a	standard 'b' assumed
4.		assumed CA and standard 'b'

a,bL The contents of locations 'a' through 'b' are listed on the teletype. The value assumed for 'b' is 'a'. Output terminates when the value of the contents of 'b' are typed or when CTRL L is pressed. CA always references the last location listed.

a,bA This command will cause control to return to DEBUG when the instruction at location 'a' has been executed 'b' times. If 'b' is not specified it is assumed to be 1. The instruction at 'a' must not be a transfer instruction; ie, BRU, BOS, BSM, BSZ, BOI, BAT.

1.8 FILE COMPARE

The File Compare routine, FCOMP, provides the user with the capability of comparing two disc files.

The command for invoking FCOMP is

```
// UTILITY 'FCOMP' 'FILE1' 'FILE2'
```

If FILE2 is omitted, FILE1 will be compared with working storage. Both files must meet the following conditions:

(1) They must appear in the file directory under the same job number.
(2) They must be of the same type.
(3) They must be the same length.

If a set of records fail the comparison, they are output to POD (the record from FILE1 is first), followed by a record containing:

(1) The word from FILE1 that failed (in ASCII)
(2) The word from FILE2 that failed (in ASCII)
(3) The word number within the record of the failing word (1-20, Base 10)
(4) The record number within the file of the failing record (1-N, Base 10).

If console switch one is off, the program halts before continuing to check the file.

ERROR MESSAGES:

ERROR-- IN FILE SPECIFICATION

One or the other of the input files are not in the directory under the current job number.

ERROR-- IN FILE TYPE

One file does not match the other as to file type.

ERROR-- EOF ON X. where X = 1 or 2

If the two files are not of the same length, this message will occur after the beginnings of the two files have been compared.

ERROR-- MISSING PARAMETER

Both file names, FILE1 and FILE2, were omitted.

ERROR-- INVALID FILE NAME

One or the other of the file names start with a blank or an illegal name ($DIRCT or $ARR) was specified.

ERROR-- SYSTEM-2

FCOMP was not able to output a "no-compare" record.

ERROR-- SYSTEM-4

Working Storage could not be opened.

ERROR-- SYSTEM-20

Phase 2 of FCOMP (FCOMP1) could not be loaded.

1.9 LIST CARDS

The List Cards routine, LISTC, provides the user with the capability of listing cards, including monitor commands (//) on the line printer.

The command for invoking LISTC is:

```
// UTILITY 'LISTC'
```

The card deck should be placed in the card reader and followed by a card containing dollar signs ($) in columns 1, 2, and 3 and two blank cards. The card reader must be ready and the line printer on line.

1.10 CARD-TO-TAPE

The Card-to-Tape routine, CRDTAP, provides the capability for the writing of cards on magnetic tape, one card (20 words) per block[1]. The resulting tape will be in a format acceptable by the FST-1 Assembler and the DOPSY Disk Operating System as standard input. Multiple files may be created on a single tape by first positioning the tape to the point at which writing is to begin.

At the end of the operation, CRDTAP writes a Tape Mark on the tape and leaves it positioned at that point so that further files may be written if desired.

The card deck should be placed in the card reader followed by a card containing dollar signs ($) in columns 1, 2, 3, and two blank cards. The card reader should be placed in "ready" condition and the tape drive placed in "remote".

The program is invoded via the command:

```
// UTILITY 'CRDTAP'
```

1.11 OCTAL DUMP

The Octal Dump routine, OCTAL, is a stand alone routine which can be loaded into core using COL (section 1.4) and will allow the user to dump core onto the line printer or teletypewriter, depending on the program version available, in groups of up to 1777B words.

The program prints directions for use on the output device used for the core dump.

1.12 TAPE-TO-LINE PRINTER

The Tape-to-Line Printer routine, TAPLP, reads a record at a time from a magnetic tape file and causes the records to be printed on the line printer. The records on the tape are assumed to be formatted and coded such that they can be transmitted directly to the line printer without conversion. A tape produced either by CRDTAP, the FST-1 Assembler, or the DOPSY Disk Operating System will be in the format acceptable to TAPLP. Any file on a multi-file tape may be listed by first positioning the tape to the proper file.

The command for invoding the program is:

```
// UTILITY 'TAPLP'
```

[1]A block is a physical record existing on tape.

SECTION II

OPERATING PROCEDURES

2.1 SYSTEM DISC RECONSTITUTION

2.1.1 Dopsy Loading Procedure

Order of card decks:

(1) DICL - Disc Initialization Card Loader
(2) $ARR (8, 12 or 16K system) - Automatic Restart Routine
(3) Deck of Assignment Cards
(4) OPEN
(5) NSOPEN
(6) CLOSE
(7) DFILEN
(8) SWAPOUT
(9) SRCH
(10) GET
(11) PUT
(12) GETW
(13) PUTW
(14) READ
(15) WRITE
(16) ADRXLATE
(17) SCAN
(18) INREC
(19) OUTREC
(20) GFREC
(21) PFREC
(22) CRIO
(23) TTRIO
(24) LPIO
(25) TTPIO
(26) DISCIO
(27) MTIO
(28) BINDEC
(29) BINOCT
(30) OCTBIN
(31) CRASC
(32) ASCBIN
(33) TVECT
(34) MONMP
(35) DEBUG
(36) $JOB
(37) $SET
(38) $RENAME
(39) $UPDATE
(40) $DELETE
(41) $ASSIGN
(42) $EXEC
(43) $CREA2
(44) $CREATE
(45) $FDUMP
(46) $UTILITY
(47) $MTAP
(48) $MONMP
(49) TWO CARD BOOT[1]

[1]See Appendices B and C.

OPERATOR FUNCTIONS

(A) Start with Deck 1 (DICL) in the card reader hopper.
(B) Check the disc 'write inhibit' switch and turn the switch off (down).
(C) Load the command register with 01000100_8.
(D) Depress CPU RESET.
(E) Depress LOAD CR (one card will be read).
(F) Depress card reader START and wait for READY indication.
(G) Depress CPU START. Cards should start loading.
(H) On first CPU halt, set console switches 2, 3, and 4 and depress CPU START.
(I) The teletype will output the core map and then halt:

Typical Map:

```
01503
MAINPR  N     00305
SWAPOU        00571
DISCIO        00655
DINT    N     00754
BINDEC        01040
ADRXLA        01073
WRITE         01162
PUTW          01177
CRIO          01222
CRINT   N     01275
TTPIO         01343
TTPINT  N     01432
```

(J) Set console switches 2 and 6 and depress CPU start. The disc will be erased (about 25 seconds) and CPU will stop.
(K) Set only console switch 2 and depress start. Cards will be read and then the teletype will respond with an asterisk (*) indicating the monitor is loaded.
(L) Type the system job I.D.

    // JOB '←←←←'

followed by a carriage return.
(M) Optional: To verify that DICL is correct, type // FDUMP DIRECTORY and verify that the following routines are listed on the teletype:

```
$DIRCT
$MONMP
$CREAT
$EXEC
$CREA2
$UPDAT
$PATCH
$SET
$DELET
$ASSIG
$FDUMP
$RENAM
$NOTE
$JOB
$DEBUG
```

(N) If step L is correct or assumed to be okay, type // SET CR. The card reader will read the control card for $ARR and the teleprinter will type PASSWORD.

(O) The operator responds by depressing control, shift and N simultaneously on the teletype keyboard.

(P) The card reader will start reading cards and after $MONMP is created, instructions for loading the 2 card boot will be printed and the CPU will then halt.

(Q) The operator loads the 2 card boot by setting 01000100_8 in the command register, depressing CPU RESET, depressing LOAD CR and CPU START.

The teletype will respond with the system monitor asterisk. DOPSY has now been loaded.

Error recovery - none.

'Error in record sequence' when $MONMP is created: Replace complete DOPSY deck with a new copy.

2.1.2 Utilities Loading Procedure

Order of card decks:

(1) $NOTE
(2) $DUMP
(3) $EDIT
(4) PCNVT
(5) LISTC
(6) $DEBUG
(7) FCOMP
(8) FCOMP1
(9) $PATCH
(10) DBUP
(11) CRDTAP
(12) TAPLP

OPERATOR FUNCTIONS

(A) With the disc monitor in core (verify by typing control L on teletype --- an asterisk should be returned), type

```
          // JOB '←←←←'
```

(B) Put cards in card reader and depress card reader START.

(C) Type // SET CR.

Cards will be read and all utility programs automatically loaded.

Error recovery. Type TTK. Reload card deck and type // SET CR to repeat.

2.1.3 Assembler Loading Procedure

Order of card decks:

(1) $ASM
(2) $ASM1

Put cards in reader and depress card reader START.

```
Type // JOB '←←←←'
     // SET CR
```

Assembler decks will automatically load.

2.1.4 FST-1 Diagnostic Loading Procedure

Card decks:

(1) SPUD
(2) MEMDI
(3) SPARWR
(4) DISCT
(5) *MTCC1
(6) *MTCC2
(7) *MTCC4
(8) DSKDIA
(9) TTYDIA
(10) LPDIA
(11) CRDIA
(12) MGTDIA

Insert cards in card reader and depress card reader START.

```
Type // JOB 'DIAG'
     // SET CR
```

If the diagnostics are being loaded with the complete FST-1 Software Package, the // JOB 'DIAG' instruction is on a card and the operator need only type

```
          // SET CR
```

The magnetic tape diagnostics are in assembly language and the assembler must be on the disc. This will also verify that the assembler words. The Delete function is also exercised here.

2.2 SYSTEM TAPE CREATION (DBUP)

The purpose of 'DBUP' is to allow the user to create a magnetic tape copy of the system disc. This can then be booted back to restore the system. (The magnetic tape heads should be cleaned before using this program either to copy or restore a disc).

USE

1. Creating a Back-up Tape

    a. Program call command
    // UTILITY 'DBUP'

    b. Program will unwind the magnetic tape to the beginning of tape marker (BOT) and write a boot record. Then the disc will be copied from the beginning of working storage.

    c. At the end of the routine the magnetic tape will be spaced forward. This allows the user to put a BOT foil marker on the tape at that point (i.e. half way down the tape). This facilitates the making of multiple copies.

    d. The program will try to recover from disc and magnetic tape read and write errors. If recovery is not possible, the program indicates the error condition and terminates.

2. Booting and reloading the disc from magnetic tape.

    a. Position the magnetic tape at load point BOT.
    b. Press the magnetic tape boot button on the FST-1 console panel. --the first tape record should be read.
    c. Put OCTAL 01000100 (BRU to 100B) in the switch register - SWITCHES 18 and 6 up. (Lowest order switch is switch 0).
    d. Press "RESET"
    e. Put "SIC" up and press "LDC".
    f. Put "SIC" down and press "START".
    g. Program should start reading magnetic tape and writing to the system disc. At the end of the program, TTY will print * and magnetic tape will unwind.
    h. System disc and core monitor will be in the same state as when the copy was made.

ERROR MESSAGES:

	MESSAGE	ACTION
1.	WRITE RING MISSING	Put write enable ring in tape reel and reload program.
2.	EOT PASSED	End of tape marker passed while writing. Mount new tape and re-load program.
3.	MAG TAPE OPERATION ERROR	Program not able to write to magnetic tape without write error. Clean tape unit. Mount new tape and reload program.
4.	DISC OPERATION ERROR	An error condition has been found on reading or writing to disc. Reload program.

NOTE: During the reloading of the disc, the program may loop trying to recover from a Magnetic Tape read error. The effect will be a continuous rocking of the magnetic tape reel. Recovery is sometimes possible by pushing first the RESET and then the START buttons on the magnetic tape unit. If a solid, irrecoverable ERROR condition exists, clean the tape unit and BOOT a different magnetic tape copy of the disc files.

Appendix A

029 CHARACTERS EQUIVALENCE TO SIX-BIT OCTAL CODE
(For Cards Read in BCD Mode)

OCTAL	029 CHAR	OCTAL	029 CHAR
00	!	40	- (Minus)
01	1	41	J
02	2	42	K
03	3	43	L
04	4	44	M
05	5	45	N
06	6	46	O
07	7	47	P
10	8	50	Q
11	9	51	R
12	0	52	11-0
13	#	53	$
14	@	54	*
15	'	55	)
16	=	56	;
17	"	57	¬
20	SPACE	60	&
21	/	61	A
22	S	62	B
23	T	63	C
24	U	64	D
25	V	65	E
26	W	66	F
27	X	67	G
30	Y	70	H
31	Z	71	I
32	0-8-2	72	12-0
33	,	73	. (Period)
34	%	74	<
35	_ (Underline)	75	(
36	>	76	+
37	?	77	| (Vertical Line)

Appendix B

CARD NO. 1: BINARY BOOTSTRAP CARD (B.B.C.)

The Binary Bootstrap Card loads the Card Reader Bootstrap Card, which is used to load systems program. (The code on the B.B.C. card is force-loaded into memory location 100B). RUNning from 100B, 24B words (one alpha-mode card) are read into 50B from Card No. 2 and the Card Reader Bootstrap program is entered at 55B. (See Appendix C).

```
LOC         CODE                  INST       COMMENTS

100       24000114        ST      LDA        DCBP
  1       06403440        RD      ARD        CDRD
  2       03140101                BOI        BSY, NA→ RD
  3       06000040        RD2     STST       CDRD
  4       03140103                BOI        BSY, NA→ RD2
  5       06010040                ETST       CDRD
  6       03140111                BOI        <, B→ HLT
  7       03200100                BOI        =  →  ST

110       01000055                BRU        55B
  1       00000100        HLT     BAH
  2       00000024        DCBA    DATA       BLOCK LENGTH
  3       00000050        DCBB    DATA       MEM LOCATION
114       00000112        DCBP    DATA       DCBA
```

Appendix C

CARD NO. 2: CARD READER BOOTSTRAP CARD

```
00000  00000040                    ORG       40B
00040  00000000   MOVED            DATA      0
00040  00000050                    ORG       50B
00040  00000000   PARX             EQU       7
00040  00000000   SLX              EQU       6
00050  00000051   DDCB             DATA      *+1,140B,60B,406B,0
00051  00000140
00052  00000060
00053  00000406
00054  00000000
00055  05700000   START            LDX       PARX,0
00056  05600007                    LDX       SLX,7
00057  24700064   MOVER            LDA       ARREAD,7
00060  14700040                    STA       MOVED,7
00061  11700001                    ATX       7,1
00062  03300057                    BLE       MOVER
00063  01000040                    BRU       MOVED
00064  06611470   ARREAD           RDS       70B
00065  24000050                    LDA       DDCB
00066  06401470                    RD        70B
00067  03140041                    BOI       3,41B
00070  06000070                    STST      70B
00071  03140044                    BOI       3,44B
00072  03400100                    BOI       10B,100B
00073  01000040                    BRU       MOVED
00073  00000000                    END
```

Appendix D

BCL LISTING

```
                                PAGE
00000  00000050                 CRG    50B
00050  00000004    CDCNT        DATA   4
00051  00000114    EPNT         DATA   114B
00052  00000020    WDCNT        DATA   16
00053  00000100    LDPNT        DATA   100B
00054  00000052    DCBP         DATA   WDCNT
00055  24000054    READ         LDA    DCBP
00056  06403440                 ARD    40B
00057  03140056                 BOI    3,*-1
00060  06000040                 STST   40B
00061  03140060                 BOI    3,*-1
00062  06010040                 ETST   40B
00063  03200055                 BOI    4,READ
00064  03140050                 BOI    3,CDCNT      Halt ON DCB,MEMORY OVERFLOW
00065  37000050                 SOM    CDCNT
00066  24000052                 LDA    WDCNT
00067  20000053                 ADD    LDPNT
00070  14000053                 STA    LDPNT
00071  24000050                 LDA    CDCNT
00072  02500055                 BNEZ   READ
00073  01040051                 BRU*   EPNT
00073  00000000                 END
```