1.	IDENTIFICATION
1.1	Digital-7-12-I
1.2	F. F. Loader
1.3	January 7, 1965

2. . ABSTRACT

The F. F. Loader will load a binary object tape produced by the PDP-7 Assembly System and will normally be punched by the assembly system at the beginning of such a tape.

During read-in, a computed checksum will be compared against a checksum read from tape. The loader will halt if the checksums differ, displaying (in the AC) a word in which zeros indicate incorrect bits. When read-in is completed, the loader will execute the instructions which were punched in the start block on the object tape. The address following the last constant stored by the loader (normally the first free location in memory) will be in the AC at this time.

3. REQUIREMENTS

3.1 Memory Locations

17600-17755 (4k; 7600-7755)

3.2 Subprograms

Read-In Mode Loader (Digital-7-12-I)

4. USAGE

All library programs are written for an 8K machine. However, the program library is completely compatible with machines having larger and smaller memories. That is, any PDP-4 may be operated as an 8K machine (remembering only that programs stored in the upper half of an 8K memory will actually be stored in locations 10000_8 lower in a 4K machine). Consequently, switch settings and memory addresses are specified for an 8K machine.

A binary tape of the F. F. Loader (supplied separately or preceding an object program) can be loaded by the Read-In Mode Loader (RIM) by placing the tape in the reader and depressing the START key with 17770 in the ADS (if RIM loader is not in memory, load RIM tape by depressing READ-IN switch on console).

The F. F. Loader starting address is 17600. The loader assumes an RSB instruction has been given to the tape reader before entry at 17600. To start the loader after positioning the tape before the start block, an RSB must be placed in 17577 and the loader started there.

When the loader is punched on an object tape by the Assembler, a jump to 17600 is automatically executed after the loader is placed in memory. The remainder of the object tape will be loaded automatically. If the loader is not punched on the object tape, a JMP 17600 replaces it, causing the RIM loader to transfer control directly to the F. F. Loader (which must be in memory, see above).

If START followed by an address was used to terminate the source program, the F. F. Loader will transfer control to that address immediately after the tape is loaded. If PAUSE was used followed by an address, depressing CONTINUE will cause the control transfer. If either START or PAUSE are used with no ensuing address, the computer will halt after loading the program. To execute the program, the user will have to place the starting address in the address switches and depress START. In all cases, the AC will contain the location following the last constant stored.

6. DESCRIPTION

A tape in the format produced by the PDP-7 Assembler, Digital-7-3-S, (as described in the Assembler Output chapter) can be loaded by the F. F. Loader. The data block body contains the information to be loaded in F. F. binary. Interpretation of this information is determined by the loader codes discussed below. Where to load the data generated, when to cease loading, and where to transfer machine control is determined by the data block heading, the termination block, and the start block, respectively (see Digital-7-3-S). This information to the loader is punched in normal binary format.

F. F. Binary

F. F. Binary code is read in alphanumeric mode; that is, all eight bits on each line are taken as data. Each three lines of tape supply 24 bits of information to the loader. Of these, 18 form a data word and the remaining 6 instruct the loader how to handle the data word.

To enable the loader to detect the end of a block of data, the number of words in each block is complemented and punched in the block heading. It is then indexed each time a word is read. Reading continues until this count is 0. Thus, if the block consists of n words, 3n lines of tape will be read, eight bits per line, to obtain the data. A word (three lines) of F. F. Binary may be interpreted directly from the tape by the user if desired. Hold the tape with three holes to the right of the feed holes and five on the left.

Reading as the loader does, from title to termination block, the first line is the eight least significant bits in the data word, the rightmost bit being bit 17. The second line is the next eight bits in the data word with bit 9 on the right. The third line from left to right contains two code bits not used by the F. F. Loader, followed by the four code bits which tell the F. F. Loader how to handle the data word. The rightmost two bits are bits 0 and 1 of the data word.

The data words in any block are loaded into memory from the highest location loaded by that block downward. The block heading contains the highest location loaded as the address of a DAC instruction. This address is decremented by one whenever a data word is loaded into memory. It will always indicate the next location to be loaded by the F. F. Loader. It is referred to as the current address indicator (CAI) in the code descriptions.

The F. F. Loader handling codes (four bits) and their meanings are:

Code	Meaning
0	Storage word. Deposit the word in the address indicated by the CAI and decrement the CAI by one.
1	Storage word with undefined symbol address. The address part of the word is the location of a register which now contains the value of the symbolic address which was undefined at assembly time (see code 3). Add the contents of the register address to the instruction part of the word to obtain the storage word. The storage word is then treated as in code 0.
2	Reference to an undefined symbol. An undefined symbol occurred in an arithmetic operation. The word is the temporary location which contains the value of the undefined symbol (see code 3). The values of undefined symbols obtained from their temporary locations are combined until a word with code 0, 1, or 5 is encountered. The symbol value is added if bit 0 of the data word is 0, subtracted if bit 0 is 1. The accumulated undefined symbol values form the storage word processed when codes 0, 1, or 5 are encountered.
3	Definition word. The data is a DAC instruction whose address is the temporary location assigned to store the definition of a previously undefined symbol which was defined at this point in the program. Deposit the contents of the CAI, plus one, into the addressed register. If the symbol was not defined equal to the value just deposited (e.g., a comma definition was not used), a word of type 4 will follow to correct the definition.
4	Undefined symbol's value. This word replaces the value defined in code 3 above. Deposit the value in the register indicated by the preceding code 3 word.
5	Constant. Search the constants' table for the constant. Insert the word at the end of the table if it is not encountered. Then add the address of the register in which the constant is stored to the next storage word (the word which used the literal as an address).
6	First three characters of a symbol.
7	Second three characters of a symbol.
10	Value of a symbol.
11-17	Unused.

Words with code 6, 7, and 10 are ignored during normal loading of a program. However, these words may be read by DDT-7 or the Assembler when punched in their respective formats (see PDP-7 Assembler, Digital-7-3-S).

As statements which have undefined symbol addresses are encountered by the Assembler, they generate code 1 words on the object tape. To avoid loading temporary symbol storage over another program or segment in memory, these registers are allocated sequentially beginning with the last address assignment (or 22 if no address assignment has occurred). The only difficulty which may be encountered occurs when the number of unique undefined symbols used since the last address assignment exceeds the number of locations since the last address assignment (or since location 22, if no assignment has been made), resulting in erroneous loading. In normal use, suchan event will not occur, but the user should be aware of the possibility.

Operating Characteristics

When the Assembler punches a data block of an object tape, it calculates a checksum (the sum of all words in the block excluding the checksum) and punches it in the heading. The F. F. Loader recalculates this sum on loading and compares it to the sum punched in the heading. If the two differ after loading a block of data, the computer will halt with a number in the AC whose 0 bits indicate where the computed checksum differs from the punched value. If repeated loadings will cause the same difference to appear in the AC lights, the object tape is probably faulty and should be reassembled. If the difference varies, the computer or reader may be causing the difficulty. Depressing CONTINUE will cause the loader to ignore the checksum; however, the recommended action is to reload the tape.

During the loading process memory locations 7 and 10 are used for storage. Consequently these locations must not be used in the assembled program. When loading is completed, location 7 contains the first free address following the constants' storage area.

8. TAPE FORMAT

FIO DEC symbolic, ASCII symbolic

9. EXECUTION TIME

n.a.

10. PROGRAM

10.4 Program Listing

```
ASCII
F.F.LOADER    1-11-65
TEM1= 17777       RIB= 17762       TEM2= RIB        CKS= 17775
CONEND= 7         INDEX = 10       WORD= 10
/READ START BLOCK
17600/
FUNNY,   JMS R1B            /READ 2 WORDS TO OBTAIN INSTR GEN BY PAUSE
         DAC DONE 1         /OR START
         JMS R1B
         DAC DONE 2
         JMS R1B  /READ ADDRESS WHEN START OR PAUSE ENCOUNTERED
         DAC CONEND                   /CONSTANTS AREA SETUP
         DAC CONBEG
         ISZ CONEND         /READY FOR FIRST CONSTANT
/READ DATA BLOCK HEADING
BLOCK,   JMS R1B  /READ STARTING ADDRESS
         DAC CA1  /PLACE IN CURRENT ADDRESS INDICATOR
         SPA      /SKIP IF NEG, IE SKIP (TERMIN BLOCK)
         JMP DONE           /END OF LOADING
         JMS R1B  /READ WORD COUNT FOR THIS BLOCK
         DAC TEM1
         ADD CAI
         DAC CKS  /CHECKSUM
         JMS R1A  /READ CHECKSUM
         DZM WORD           /CLEAR ACCUMULATED WORD
CONT,    ISZ TEM1           /FINISHED LOADING THIS BLOCK?
         JMP CONT1          /NO
         LAC CKS  /YES-CHECK CHECKSUM
         CMA
         SZA
         HLT      /CHECK SUM ERROR
         RSB
         JMP BLOCK          /CONTINUE
/BREAK DOWN A F.F.BINARY WORD
CONT1,   JMS R1A  /READ FIRST LINE OF A F.F.BINARY WORD
         DAC TEM2
         JMS R1A  /READ SECOND LINE OF A F.F.BINARY WORD
         RTL
         RTL
         RTL
         RTL
         ADD TEM2
         DAC TEM2           /DATA WORD, BITS 2-17
         JMS R1A  /READ THIRD LINE OF A F.F.BINARY WORD
         RTR
         ADD CDSP
         DAC R1A  /SET UP DISPATCH FOR WORD TYPE
         AND CCMA
         RAR
         ADD TEM2          /FORM WHOLE DATA WORD
         XCT I R1A         /DISPATCH TO (DSP+LOADER CODE) WITH DATA WORD
                           /IN AC
         JMP CONT          /GET NEXT F.F.BINARY WORD


/DISPATCH TABLE FOR MAGIC CODES
DSPTCH,  JMP CODE0         /STORAGE WORD
         JMP CODE1         /WORD WHOSE ADDRESS POINTS TO AN UNDEFINED
         JMP CODE2         /ADDED OR SUBTRACTED VALUE
         JMP CODE3         /ADDRESS FOR AN UNDEFINED
         XCT CODE4         /VALUE FOR AN UNDEFINED
         JMP CONST         /CONSTANT, CODE5
         NOP      /FIRST SYM WORD, CODE6
         NOP      /SECOND SYM WORD, CODE7
         NOP      /SYM CODE10
/START LOADED PROGRAM
DONE,    LAC CONEND        /PLACE FIRST FREE ADDRESS IN AC
         XX       /TRANSFER CONTROL OR HALT
         XX
/STORAGE WORD, UNDEFINED ADDRESS
CODE1,   AND IMSK          /MASK UNDEF. ADDRESS POINTER
         ADD I TEM2        /ADD VALUE OF UNDEF. ADDRESS AS A DAC
/STORAGE WORD
CODE0,   ADD WORD          /ACCUMULATED UNDEF. SYMB, VALUES
CAI,     XX       /CURRENT ADDRESS INDICATOR
         CLC      /-1, TWO'S COMP.
         TAD CAI  /DECREMENT ADDRESS POINTER
         DAC CAI
         JMP CONT-1       /GET NEXT F.F.BINARY WORD
/FOR-UNDEFINED SYMBOL
CODE2,   CLAVSPA  /ADD IF BIT0=0
CCMA,    CMA      /SUBTRACT IF BIT 0=1
         XOR I TEM2        /IF BIT 0 WAS 0, SAME AS TEM2, IF BIT 0
                           /WAS 1, CMA TEM2
         ADD WORD          /ADD VALUE TO WORD
CODE2A,  DAC WORD
         JMP CONT          /GET NEXT F.F.BINARY WORD
/POINTER TO WHERE AN UNDEFINED WILL BE STORED
```

```
CODE3,   DAC CODE4          /SAVE DATA WORD, TEMP, LOCATION FOR UNDEFINED
         LAC CAI
         TAD CCMA           /ADD 1 TO CURRENT ADDRESS INDICATOR
CODE4,   XX      /PUT CURRENT ADDRESS INDICATOR INTO TEMP. LOCATION
         JMP CONT           /GET NEXT F.F.BINARY WORD


/READ 1 ALPHA WORD
R1A,     0
         RSF
         JMP .-1
         RRB
         RSA
         DAC TEM
         ADD CKS
         DAC CKS  /ACCUMULATE CHECK SUM
         LAC TEM
         CLL
         JMP I R1A
/SOME CONSTANT STORAGE
CDSP,    DSPTCH
IMSK,    760000
TEM,     0
CONBEG,  0
/SEARCH AND STORE A CONSTANT, CODE5
CONST,   ADD WORD           /ADD ANY UNDEF. SYMBOL VALUES TO CONSTANT AND
         DAC I CONEND       /DEPOSIT IN END OF CONSTANT TABLE
         LAC CONBEG
         DAC INDEX          /DEPOSIT BEGINNING ADDRESS OF TABLE IN INDEX
         LAC I CONEND       /LOAD CURRENT CONSTANT VALUE
CON1,    SAD I INDEX        /COMPARE TO CONSTANTS IN TABLE PREVIOUSLY
         JMP FIND
         SAD I INDEX        /C (INDEX) IS AUTOMATICALLY INCREMENTED
         JMP FIND           /BEFORE EACH TEST
         SAD I INDEX
         JMP FIND
         SAD I INDEX
         JMP FIND
         SAD I INDEX
         JMP FIND
         JMP CON1
FIND,    LAC INDEX
         SAD CONEND
         ISZ CONEND         /IF NEW CONSTANT, INCREMENT POINTER TO END
                            /OF CONSTANT TABLE
         JMP CODE2A         /JUMP TO CODE 2A, PLACE VALUE OF CONSTANT IN WOR
PAUSE
```