DATA GENERAL
CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

PROGRAM

Binary Loader

TAPES

Special Format: 091-000004-03

ABSTRACT

The Binary Loader is a routine used to load the absolute binary tapes produced as output by the Assembler.

093-000003-03

1. REQUIREMENTS

1.1 Memory

1K or larger alterable memory.

1.2 Equipment

Teletype ASR or paper tape reader.

1.3 External Subroutines

None.

1.4 Other

None.

2. OPERATING PROCEDURE

2.1 Calling Sequence

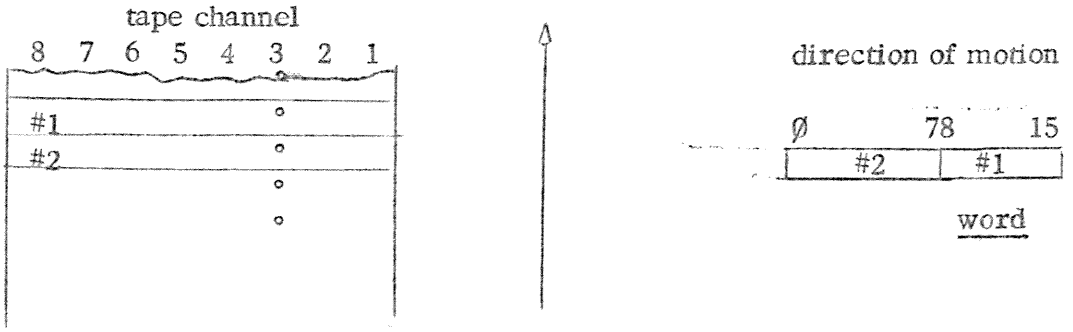
The Binary Loader must be loaded using the Bootstrap Loader and the special format tape, 091-000004.

The Binary Loader is started by entering SX777 in the data switches and depressing START. "X" represents the two most significant octal digits of the highest memory address available. For example, X = Ø7 for a 4K system and 17 for an 8K system. "S" represents bit Ø of the data switches and should be set if input is to be via the paper tape reader and reset if via the teletype.

2.2 Input Format

The input to the Loader is an absolute binary tape. The tape is punched in blocks separated by null (all zero) characters. The Loader reads two tape characters to form a 16-bit word.

The format is as follows:



In other words, the first tape character forms bits 8-15 of the data word, and the second tape character forms bits $\emptyset$-7 of the data word.

The first non-null tape character indicates the start of a new block. Four different block types, data, multiple data, start, and error, are defined. The block type is determined by the first word of the block. A description of each block type follows.

The first word, WC, of a Data Block is in the range $\emptyset < WC \leq 2\emptyset_8$. Its format is:

word	
1	-WC
2	ADDRESS
3	CHECKSUM
4	DATA WD 1
5	DATA WD 2
"	"
"	"
"	"
3+N	DATA WD N

WC = N

The two's complement of WC is given in the first word. Normally sixteen data words will be punched per data block, but the .END and .LOC pseudo-ops to the Assembler may cause short blocks to be punched. The second word contains the address at which the first data word is to be loaded. Subsequent data words are loaded in sequentially ascending locations. The third word contains a checksum. This number is computed so that the binary sum of all words in the block should give a zero result. The remaining words are the data to be loaded.

The first word, WC, of a Multiple Data Block is in the range

$$2\emptyset_8 < WC \leq 77777_8$$

Its format is:

word	
1	-WC
2	ADDRESS
3	CHECKSUM
4	DATA WD

where again the two's complement of WC is given in the first word. This block type is used to indicate that 16_{10} or more data words, all identical to the one data word punched, are to be loaded sequentially into memory locations beginning at the absolute address, ADDRESS. In this case, the number of identical data words, n, is given by the formula

$$n = WC - 1$$

i.e. if the first word of the block is -17_{10}, the data is to be repeated 16_{10} times (note that WC is the absolute value of the first word). The checksum is computed in the same manner as an ordinary Data Block.

The first word of a Start Block is ØØØØØ1. Its format is:

word	
1	ØØØØØ1
2	S ADDRESS
3	CHECKSUM

The second word uses bit Ø as a flag. If S = 1, the loader will HALT after loading. If S = Ø, the loader will transfer control after loading to the address in bits 1-15 of the second word. The checksum is the same as that for a Data Block.

The first word of an Error Block is greater than +1. Its format is:

word	
1	>1
2	
.	IGNORED
.	
.	
N	

The last byte of an error block is a rubout (377).
An error block is ignored in its entirety by the Loader.

The binary tape to be loaded must be mounted in the input device selected by bit Ø of the data switches before starting the Loader.

2.3 Output Format

The output is a loaded routine ready for execution. If no starting address was given, the Loader will HALT at location XX741. Otherwise, control will be transferred to the loaded routine.

2.4 Error Returns

Two error conditions will cause the Loader to HALT at location XX727.

The first is a binary tape that attempts to overwrite the Loader. This is a fatal error, and the user must reassemble with a lower origin before loading will be successful.

The second is a checksum failure over the last block read. The binary tape should be repositioned to the beginning of the last block read and CONTINUE depressed. If this second attempt fails, the binary tape should be assumed to be incorrectly punched. The user must either reassemble to obtain a new binary tape, or he must proceed with the loading from the next block and after loading key in from the console the sixteen words of the block in error.

2.5 State of Active Registers upon Exit

If a checksum error occurs, ACØ will contain the incorrect checksum.

If a binary tape attempted to overwrite the Loader, AC3 will contain the address which would have been overwritten.

2.6 Cautions to User

If possible, the user should write routines which do not destroy locations above XX635 (the start of the Loader). If he adheres to this practice, the Bootstrap and Binary Loaders will always be intact and need never be re-loaded. Note that although the Loader will not load data above XX635, the user can write in this area during execution.

3. DISCUSSION

3.1 Algorithms

The binary loader reads in a frame of information at a time from the input device using a GTCHR routine. Once the start of a block has been detected (a non-null frame), the Loader assembles two frames at a time to construct a complete 16-bit word. The type of block is determined, *i.e.* start, data, multiple data, or error, and control is transferred to an appropriate processing routine. A start block terminates the loading process by causing control to be transferred to the starting address or causing the Loader to HALT.

3.2 Limitations and Accuracy

The Binary Loader will not permit itself to be overwritten.

3.3 Size and Timing

The Loader is 12Ø (octal) words in length, 116 of which immediately precede the Bootstrap Loader and the remaining two of which follow the Bootstrap.

The speed of the Loader is limited by the speed of the input device.

3.4 References

See write-up 093-000002 for a description of the Bootstrap Loader.

3.5 Flow Diagrams

Not applicable.

4. EXAMPLES AND APPLICATIONS

None.

5. PROGRAM LISTING

A list of the Binary Loader follows. It has been originated at 3635 (a 2K system) for illustrative purposes only.

```
                ;
                ; PREAMBLE FOR NEW BOOT PROGRAM
                ;

        000777          .LOC 777        ;ANY NON PAGE ZERO WILL DO

        000027          GET=27

000777  000001          000001          ;TAPE SYNCHRONIZER
01000   177754          BEG-END-2       ;NEGATIVE WORD COUNT FOR PREAMBLE

01001   020421  BEG:    LDA 0,C4K       ;MEMORY SIZING INCREMENT
01002   176221          ADCZR 3,3,SKP   ;FORM HIGHEST ADDRESS
01003   116403  LOOP:   SUB 0,3         ;DECREMENT
01004   055400          STA 3,0,3       ;STORE ADDRESS
01005   031400          LDA 2,0,3       ;GET IT BACK
01006   172414          SUBN 3,2,SZR    ;SAME ?
01007   000774          JMP LOOP        ;NO - NO MEMORY
01010   004027          JSR GET         ;GET
01011   044411          STA 1,C4K       ;SAVE COUNT OF BINLOADER
01012   133000          ADD 1,2         ;FORM FIRST ADDRESS
01013   151400          INC 2,2         ;INCREMENT ADDRESS
01014   004027          JSR GET         ;GET
01015   045000          STA 1,0,2       ;SET INTO MEMORY
01016   010404          ISZ C4K         ;BUMP COUNT
01017   000774          JMP .-4         ;GO BACK
01020   063077          HALT            ;WHOA FAT HIPPO
01021   001000          JMP 0,2
01022   004000  C4K:    4000
01023   000756  END:    JMP BEG         ;GETS CONTROL HERE
```

```
                        ;START
                        ;BINARY BLOCK LOADER

                        ;SUBROUTINE TO ASSEMBLE A WORD INTO AC2. THIS WORD IS
                        ;ADDED INTO THE CHECKSUM HELD IN AC0

01024 177636                    BUILD-BEND-1    ;MINUS WORD COUNT FOR BIN LOADER

17636 01025 054512      BUILD:  STA 3,TEMP1     ;SAVE THE RETURN
      01026 004407              JSR GTCHR       ;GET CHARACTER INTO AC3
      01027 171300              MOVS 3,2        ;AND SAVE IN THE LH OF AC2
      01030 004405              JSR GTCHR       ;GET THE NEXT CHARACTER
      01031 173300              ADDS 3,2        ;AND BUILD IN AC2
      01032 143000              ADD 2,0         ;ADD INTO CHECKSUM
      01033 002534              JMP @TEMP1      ;AND RETURN
      01034 000004      DIFF:   4

                        ;SUBROUTNE TO GET A CHARACTER INTO AC3
                        ;IF SWITCH0=0, USE TELETYPE, ELSE USE PTR
      01035 054503      GTCHR:  STA 3,TEMP2     ;SAVE THE RETURN
      01036 034503              LDA 3,SAVE      ;GET THE SWITCH WORD
      01037 175103              MOVL 3,3,SNC    ;AND TEST BIT 0
      01040 000405              JMP .+5         ;A 0, USE THE TTI
      01041 063612              SKPDN PTR       ;A 1, USE THE PTR
      01042 000777              JMP .-1
      01043 074512              DIAS 3,PTR      ;READ INTO AC3 AND START
      01044 002474              JMP @TEMP2      ;RETURN

      01045 063610              SKPDN TTI       ;WAIT FOR TTI FLAG
      01046 000777              JMP .-1
      01047 074510              DIAS 3,TTI
      01050 002470              JMP @TEMP2      ;EXIT

                        ;START OF THE LOADER
      01051 062677      START:  IORST
      01052 060477              READS 0         ;READ SWITCHES
      01053 040466              STA 0,SAVE      ;AND SAVE THE WORD
      01054 060110              NIOS TTI        ;START BOTH READERS
      01055 060112              NIOS PTR
```

AAA

```
                        ;READ IN A BLOCK
01056 004757    BLOCK1: JSR GTCHR       ;GET A CHARACTER
01057 171383            MOVS 3,2,SNR    ;AND TEST IT FOR ZERO
01060 000776            JMP BLOCK       ;YES, STILL IN LEADER
01061 004754            JSR GTCHR       ;OK, BUILD A WORD
01062 173300            ADDS 3,2        ;IN AC2
01063 141000            MOV 2,0         ;SET INTO THE CHECKSUM
01064 145000            MOV 2,1         ;SET THE COUNTER
01065 004740            JSR BUILD       ;GO GET THE ADDRESS
01066 050477            STA 2,ADDRS     ;AND STORE IT
01067 004736            JSR BUILD       ;READ THE CHECKSUM WORD
01070 125113            MOVL# 1,1,SNC   ;TEST THE COUNT
01071 000426            JMP TEST        ;IT IS >0, IE A START OR IGNORE
01072 044453            STA     1,COUNT ;BLOCK

                        ;READ IN THE DATA BLOCK
01073 030445            LDA 2,TEMP2     ;SEE IF STORAGE
01074 034443            LDA 3,DIFF
01075 172433            SUB 3,2
01076 034467            LDA 3,ADDRS     ;ADDRESS IS TOO BIG
01077 136400            SUB     1,3
01100 172023            ADCZ 3,2,SNC
01101 000414            JMP CHKER       ;YES, HALT THE LOADER
01102 030441            LDA     2,C2@
01103 147033            ADDZ# 2,1,SNC
01104 010436            ISZ COUNT
01105 147022            ADDZ    2,1,SZC ;REPEAT BLOCK?
01106 125113    STORE1: MOVL#   1,1,SNC
01107 004716            JSR BUILD
01110 052455            STA 2,@ADDRS
01111 010454            ISZ ADDRS
01112 010453            ISZ     COUNT
01113 000773            JMP STORE
01114 101004            MOV 0,0,SZR     ;NOW, TEST THE CHECKSUM
01115 063077    CHKER1: HALT            ;CHECKSUM ERROR, AC0=VALUE
01116 000743            JMP BLOCK       ;GO READ IN A BLOCK
```

```
                        ;START BLOCK OR IGNORE BLOCK
01117 125224  TEST:   MOVZR 1,1,SZR
01120 000411          JMP IGNOR         ;AN IGNORE BLOCK
01121 101004          MOV 0,0,SZR       ;TEST THE CHECK SUM
01122 000773          JMP CHKER         ;ERROR
01123 033442          LDA 2,ADDRS       ;GET THE ADDRESS
01124 052677          IORST             ;DO A RESET
01125 151113          MOVL# 2,2,SNC     ;TEST BIT 0
01126 001000          JMP 0,2           ;0=START THE PROGRAM
01127 063077          HALT              ;0, HALT
01130 000777          JMP .-1
                        ;IGNORE ERROR MESSAGES BY READING UNTIL
                        ;A RUBOUT

01131 004704  IGNOR:  JSR GTCHR         ;GET INTO AC3
01132 020404          LDA 0,C377
01133 116404          SUB 0,3,SZR
01134 000775          JMP IGNOR
01135 000721          JMP BLOCK         ;OK, GO INTO BLOCK MODE
01136 000377  C377:   377
01137 000000  TEMP1:  0
01140 000000  TEMP2:  0
01141 000000  SAVE:   0
01142 000000  COUNT:  0
01143 000020  C20:    20                ; REPEAT BLOCKS HAVE WD > 20(OCTAL)

      001165          .LOC .+21         ;SKIP BOOTSTRAP (OLD NOVA)
01165 000000  ADDRS:  0
77777 01166 000603    BEND:   JMP START

                      .END
```

Kor. 177777