Data General

SOFTWARE

User's Manual

PROGRAM BINARY LOADER TAPES ABSOLUTE BINARY: 091-000004

093-000003-06

ABSTRACT

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

Ordering No. 093-000003

Printed in the United States of America
Rev. 06, January 1973

NOTICE

Original Release - June, 1969
First Revision - Unknown
Second Revision - Unknown
Third Revision - November, 1970
Fourth Revision - October, 1971
Fifth Revision - October, 1972
Sixth Revision - January, 1973

This revision of the Binary Loader manual, 093-000003-06, corrects an error in the starting address on page 1. Also, the pages of the manual have been renumbered to start with 1. The revision supersedes manual number 093-000003-05. Since Revision 6 follows closely upon Revision 5, the changes incorporated in Revision 5 as well as the Revision 6 change are listed at the back of the manual.

1.0 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.

2.0 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 SXX777 in the data switches and depressing START. "XX" represent the two most significant octal digits of the highest memory address available, for example, XX = ∅7 for a 4K system and 17 for an 8K system. "S" represents bit ∅ of the data switches and should be set if input is to be via the paper tape reader and reset if via the teletype.

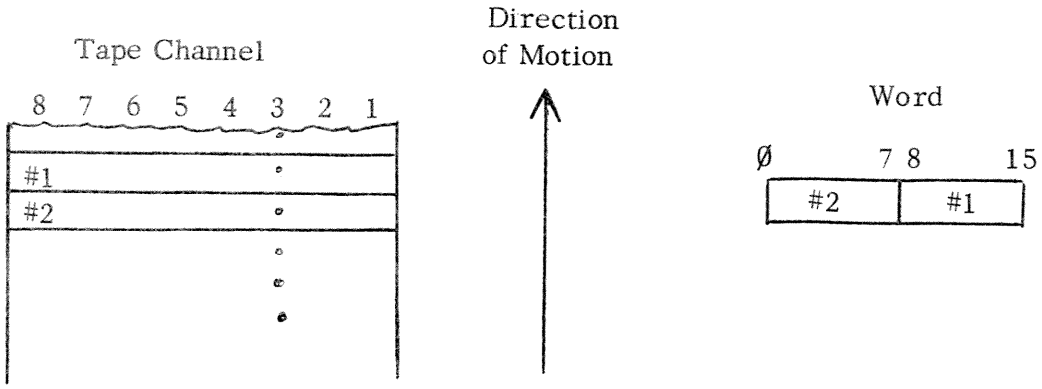
2.2 Input Format

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:

---

* unless the tape was produced under the Stand-alone, Disk, or Real Time Disk Operating Systems.



In other words, the first tape character forms bits 8-15 of the data word, and the second tape character forms bits Ø-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 :

* blocks produced under SOS, DOS, or RDOS are not separated by nulls.

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

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 (177).
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. If the tape was produced by the stand-alone assembler or debugger, 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 reloaded. Note that although the Loader will not load data above XX635, the user can write in this area during execution.

---

* If the tape was produced under SOS, DOS, or RDOS, it must be repositioned at the beginning of the first block; RESET and START must then be depressed.

3.0 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.

4.0 EXAMPLES AND APPLICATIONS

None.

5.0 PROGRAM LISTING

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

```
  0001   .MAIN
01
02                ;
03                ; PREAMBLE FOR NEW BOOT PROGRAM
04                ;
05
06        000610          .LOC 610          ;ANY NON PAGE ZERO WILL DO
07
08        000027          GET=27
09
10  00610 000001          000001            ;TAPE SYNCHRONIZER
11  00611 177754          BEG-END-2         ;NEGATIVE WORD COUNT FOR PREAMBLE
12
13  00612 020421  BEG:    LDA 0,C4K         ;MEMORY SIZING INCREMENT
14  00613 176221          ADCZR 3,3,SKP     ;FORM HIGHEST ADDRESS
15  00614 115400  LOOP:   SUB 0,3           ;DECREMENT
16  00615 055400          STA 3,0,3         ;STORE ADDRESS
17  00616 031400          LDA 2,0,3         ;GET IT BACK
18  00617 172414          SUB# 3,2,SZR      ;SAME ?
19  00620 000774          JMP LOOP          ;NO - NO MEMORY
20  00621 004027          JSR GET           ;GET
21  00622 044411          STA 1,C4K         ;SAVE COUNT OF BINLOADER
22  00623 133000          ADD 1,2           ;FORM FIRST ADDRESS
23  00624 151400          INC 2,2           ;INCREMENT ADDRESS
24  00625 004027          JSR GET           ;GET
25  00626 045000          STA 1,0,2         ;SET INTO MEMORY
26  00627 010404          ISZ C4K           ;BUMP COUNT
27  00630 000774          JMP .-4           ;GO BACK
28  00631 063077          HALT              ;WHOA FAT HIPPO
29  00632 001000          JMP 0,2
30  00633 004000  C4K:    4000
31  00634 000756  END:    JMP BEG           ;GETS CONTROL HERE
```

```
A 0002   .MAIN
01                ;START
02                ;BINARY BLOCK LOADER
03
04                ;SUBROUTINE TO ASSEMBLE A WORD INTO AC2, THIS WORD IS
05                ;ADDED INTO THE CHECKSUM HELD IN AC0
06
07 00535 177636           BUILD-BEND-1      ;MINUS WORD COUNT FOR BIN LOADER
08
09 00536 054512 BUILD:    STA 3,TEMP1       ;SAVE THE RETURN
10 00537 004407           JSR GTCHR         ;GET CHARACTER INTO AC3
11 00540 171300           MOVS 3,2          ;AND SAVE IN THE LH OF AC2
12 00541 004405           JSR GTCHR         ;GET THE NEXT CHARACTER
13 00542 173300           ADDS 3,2          ;AND BUILD IN AC2
14 00543 143000           ADD 2,0           ;ADD INTO CHECKCUM
15 00544 002504           JMP @TEMP1        ;AND RETURN
16 00545 000004 DIFF:     4
17
18                ;SUBROUTNE TO GET A CHARACTER INTO AC3
19                ;IF SWITCH0=0, USE TELETYPE, ELSE USE PTR
20 00546 054503 GTCHR:    STA 3,TEMP2       ;SAVE THE RETURN
21 00547 034503           LDA 3,SAVE        ;GET THE SWITCH WORD
22 00550 175103           MOVL 3,3,SNC      ;AND TEST BIT 0
23 00551 000405           JMP .+5           ;A 0, USE THE TTI
24 00552 063612           SKPDN PTR         ;A 1, USE THE PTR
25 00553 000777           JMP .-1
26 00554 074512           DIAS 3,PTR        ;READ INTO AC3 AND START
27 00555 002474           JMP @TEMP2        ;RETURN
28
29 00556 063610           SKPDN TTI         ;WAIT FOR TTI FLAG
30 00557 000777           JMP .-1
31 00560 074510           DIAS 3,TTI
32 00561 002470           JMP @TEMP2        ;EXIT
33
34                ;START OF THE LOADER
35 00562 062677 START:    IORST
36 00563 060477           READS 0           ;READ SWITCHES
37 00564 040466           STA 0,SAVE        ;AND SAVE THE WORD
38 00565 060110           NIOS TTI          ;START BOTH READERS
39 00566 060112           NIOS PTR
40
41
```

```
A 0003   .MAIN
01
02
03                      ;READ IN A BLOCK
04 00667 004757 BLOCK:  JSR GTCHR          ;GET A CHARACTER
05 00670 171305         MOVS 3,2,SNR       ;AND TEST IT FOR ZERO
06 00671 000776         JMP BLOCK          ;YES, STILL IN LEADER
07 00672 004754         JSR GTCHR          ;OK, BUILD A WORD
08 00673 173300         ADDS 3,2           ;IN AC2
09 00674 141000         MOV 2,0            ;SET INTO THE CHECKSUM
10 00675 145000         MOV 2,1            ;SET THE COUNTER
11 00676 004740         JSR BUILD          ;GO GET THE ADDRESS
12 00677 050477         STA 2,ADDRS        ;AND STORE IT
13 00700 004736         JSR BUILD          ;READ THE CHECKSUM WORD
14 00701 125113         MOVL# 1,1,SNC      ;TEST THE COUNT
15 00702 000426         JMP TEST           ;IT IS >0, IE A START OR IGNORE
16 00703 044450         STA      1,COUNT   ;BLOCK
17
18                      ;READ IN THE DATA BLOCK
19 00704 030445         LDA 2,TEMP2        ;SEE IF STORAGE
20 00705 034740         LDA 3,DIFF
21 00706 172400         SUB 3,2
22 00707 034467         LDA 3,ADDRS        ;ADDRESS IS TOO BIG
23 00710 136400         SUB      1,3
24 00711 172023         ADCZ 3,2,SNC
25 00712 000414         JMP CHKER          ;YES, HALT THE LOADER
26 00713 030441         LDA      2,C20
27 00714 147033         ADDZ# 2,1,SNC
28 00715 010436         ISZ COUNT
29 00716 147022         ADDZ     2,1,SZC   ;REPEAT BLOCK?
30 00717 125113 STORE:  MOVL#    1,1,SNC
31 00720 004716         JSR BUILD
32 00721 052455         STA 2,@ADDRS
33 00722 010454         ISZ ADDRS
34 00723 010430         ISZ      COUNT
35 00724 000773         JMP STORE
36 00725 141004         MOV 0,0,SZR        ;NOW, TEST THE CHECKSUM
37 00726 063077 CHKER:  HALT               ;CHECKSUM ERROR, AC0=VALUE
38 00727 000740         JMP BLOCK          ;GO READ IN A BLOCK
39
```

47737

```
A 0004   .MAIN
01                      ;START BLOCK OR IGNORE BLOCK
02 00730 125224 TEST:    MOVZR 1,1,SZR
03 00731 000411          JMP IGNOR          ;AN IGNORE BLOCK
04 00732 101004          MOV 0,0,SZR        ;TEST THE CHECK SUM
05 00733 000773          JMP CHKER          ;ERROR
06 00734 030442          LDA 2,ADDRS        ;GET THE ADDRESS
07 00735 062677          IORST              ;DO A RESET
08 00736 151113          MOVL# 2,2,SNC      ;TEST BIT 0
09 00737 001000          JMP 0,2            ;0=START THE PROGRAM
10 00740 063077          HALT               ;1, HALT
11 00741 000777          JMP .-1
12                      ;IGNORE ERROR MESSAGES BY READING UNTIL
13                      ;A RUBOUT
14
15 00742 004704 IGNOR:   JSR GTCHR          ;GET INTO AC3
16 00743 020404          LDA 0,C377
17 00744 116404          SUB 0,3,SZR
18 00745 000775          JMP IGNOR
19 00746 000721          JMP BLOCK          ;OK, GO INTO BLOCK MODE
20 00747 000377 C377:    377
21 00750 000000 TEMP1:   0
22 00751 000000 TEMP2:   0
23 00752 000000 SAVE:    0
24 00753 000000 COUNT:   0
25 00754 000020 C20:     20                 ; REPEAT BLOCKS HAVE WD > 20(OCTAL)
26
27       000776          .LOC .+21          ;SKIP BOOTSTRAP (OLD NOVA)
28 00776 000000 ADDRS:   0
29 00777 000653 BEND:    JMP START
30
31                       .END
```

CHANGE FROM REVISION 5 TO REVISION 6 OF THE BINARY LOADER MANUAL

Page 1	The starting address is corrected to SXX777.

CHANGES FROM REVISION 4 TO REVISION 5

The following describes the substantive changes in revision 5 of the Binary Loader manual. Typographical corrections are not given.

Pages 2 and 3 (Pages 1 and 2 of Rev. 6)	Binary format for tapes punched by stand-alone software differs from that produced by software supported by either the Stand-alone, Disk, or Real Time Disk Operating Systems. Binary blocks produced under these operating systems are not separated by all-zero characters.
Page 5 (Page 4 of Rev. 6)	The recovery procedure for checksums detected during the reading of binary tapes lacking interblock nulls requires that the tapes be repositioned at the beginning of the first block; RESET and START must then be depressed.

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code