R. MOORE

FP6000 PROGRAM REPORT 105

GENERAL PURPOSE LOADER

J. M. Chapm...,
Ferranti Electronics
November, 1963

CONTENTS

1.0 Introduction

The General Purpose Loader forms the load and go pass of compiling processes and is automatically included in programs produced by the compilers. The program's request slip and the loader program are read by EXECUTIVE which then transfers control to the loader to read in and assemble the program. When a program has been successfully read in, the loader will clear itself out of store and enter the program at location 20.

The output of compilers is, in general, in the form of blocks which are identified by their leading character. A block may contain up to 128 characters although 80 is regarded as a normal maximum.

Blocks of semi-compiled program contain sequences of five-characters, of which the first is a label and the other four together constitute a word. The label informs the loader what it is supposed to do with the word.

The Loading Process

While it is loading a program, the loader keeps a "Transfer Address" (T), together with 15 relativizers corresponding to the various program and data areas. The loader provides for any or none of these relativizers to be added to the word, and for loading it at location T. After loading, a number DT is added to T. Normally DT is set at 1 but it may be reset at any value by the compiler. In addition T itself may be reset relative to any of the program or data areas.

1.2 The Relativizers

The relativizers and their symbolic names are:-

0	Always 0	
1	Lower Work Space	(RLW)
2	Lower Variables	(RLV)
3	Spare	
4	Constants (RCN) (Start of non erasible memory)	(RLP)
5	Literals	(RLT)
6	Spare	
7	Upper Preset	(RUP)
8	Spare	

9	Upper work space (RUW) (End of non erasible memory)
10	Spare
11	Program (RPR) (set by title blocks)
12	Special Rel. #1 (RR1) (By convention the I/O Package)
13	Special Rel. #2 (RR2) (By convention the Arithmetic Package)
14	Special Rel. #3 (RR3)
15	Special Rel. #4 (RR4)

The initial values of these relativizers are set by a block of the consolidated leader. Relativizers 1 - 10 are incremented by an end of section block. Relativizers 12 - 15 are set from the cue table.

1.3 Nomenclature

Symbols used in this report (other than relativizer names given above) are:-

T	The transfer address setting
DT	The transfer address increment
B	The octal value of the lead character of a block
L	The octal value of the label character of a five character sequence
N	The contents of the 24 bit word in a five character sequence
W	The word count of a block

For further clarification, values to be interpreted as octal will be preceded by *, otherwise numbers may be assumed expressed in decimal.

2.0 Blocks of Information

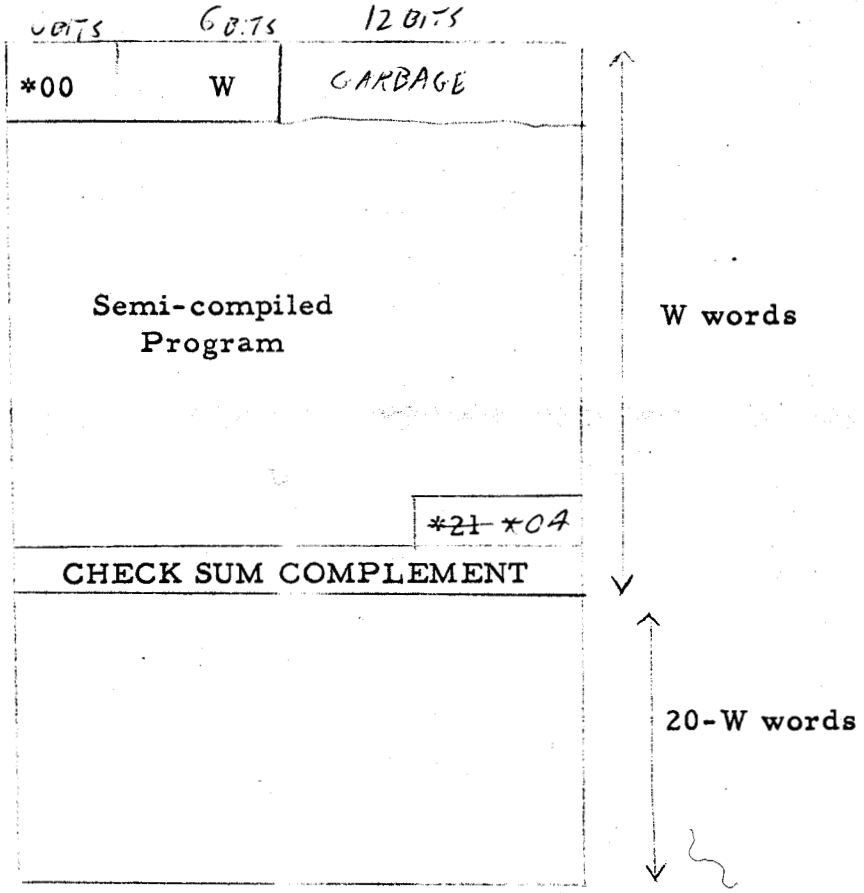
A block of information consists of a group of up to 80 characters. This will be contained on one punched card, a strip of paper tape terminated by 'newline' or one 20 word block on magnetic tape. A block is identified by its leading character and a block format is peculiar to its type.

The block types and their functions are:-

B	Graphic	Function
*00	0	Object Program (semi compiled)
*01	1	Object Program (binary)
*02	2	Entry Point
*03	3	Title of Program Segment
*04	4	End of Program Segment
*05 - *12		Illegal
*13	;	Pause
*14	<	Terminator of consolidated Leader
*15	=	Relativizer settings in Consolidated Leader
*16	>	Parameters (consolidated Leader)
*17	?	Cue Entry (consolidated Leader)
*20	Space	Ignored (corresponds to blank card)
*21 - *32		Ignored (reserved for future use)
*33	+	Title of ordinary leader (always ignored)
*34	,	End of ordinary leader (always ignored)
*35 - *36		Ignored (reserved for future use)
*37	/	Cue entry in ordinary leader (always ignored)
*40 - *72		Request slip or comments (always ignored)
*73 - *77		Ignored

2.1 Object Program Block (Semi-compiled)

A block of object program, when placed in an input/output buffer appears as follows:



The form of the semi-compiled program is given in section 3.

The end of block marker (*21) may occur anywhere in the word immediately before the check sum.

The first W words of the buffer are check-summed using function 127 before the block is scanned.

Semi-compiled program is always loaded temporarily with a parameter S added to the transfer address.

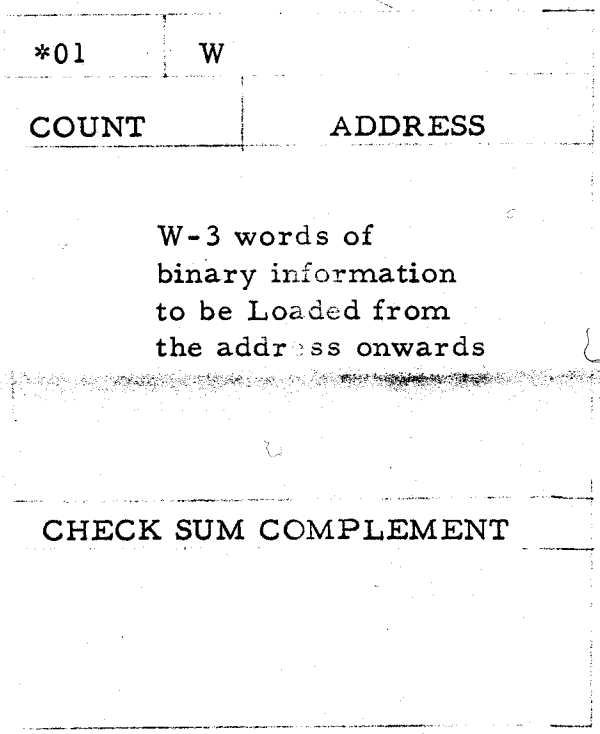
2.2 Object program block (binary program)

This block contains program which is absolute form (i.e. no relativizers have to be added to it, and it may be loaded under control of EXECUTIVE without recourse to the loader which itself is in this form).

Binary program is always loaded at the address specified and is

not stored temporarily with a displacement of S as is the case with semi-compiled program.

The layout of a binary block is as follows:-



The check sum is performed as for semi-compiled program.

2.3 Entry Point block

An entry point block consists of an entry and possibly a message to be printed by EXECUTIVE on the monitor typewriter.

EXECUTIVE will type the message.
It is laid out as follows:

*02 -ve if message	W	ENTRY POINT
count		ZERO
		CHARACTERS OF MESSAGE (IF ANY)
		CHECK SUM COMPLEMENT

If no message is to be typed, the program is entered as indicated without suspension. If a message is typed the program will be automatically suspended.

2.4 Title of a program segment

A title of a program segment must follow either a consolidated leader or an end of segment block. If it occurs immediately after a block of semi-compiled or binary program then the loader will be suspended and the EXECUTIVE will type the message:-

LOADER ERROR #05

The block consists of

(a) Identifier B=*03

(b) Octal field giving type

(c) Alphanumeric field giving name

In both cases the fields are terminated by spaces.

When the loader detects such a block it will scan the cue-list and if either the name does not appear or else a segment of this name has already been read, the loader will continue reading ignoring all blocks until the next "title of segment" block is found.

If the name is present and a previous segment has not been read in, the value of relativizer #11 (RPR) is set to the value indicated by the cue name and the cue list marked that the segment has been accepted.

2.5 End of segment block

An end of segment block terminates a sequence of semi-compiled program. It contains a sequence of numbers which are treated as increments for relativizers 1 - 10.

The block consists of:

(a) Identifier B=*04

(b) Ten Octal fields corresponding to relativizer increments, these are:

Increment for RLW
Increment for RLV
Zero
Increment for RCN
Increment for RLT
Zero
Increment for RUP
Zero
Increment for RUW
Zero

The increments are added to the various relativizers and the cue-list examined. If all of the preset type (i.e. constants, literals, program etc.) have been accepted then the wind-up phase is entered, this is described in section 4.

If any preset segment is yet to come, the loader will return to searching for a title.

2.6 Pause block

On encountering "pause" the loader switches the peripheral off line to give the operator an opportunity to replace the tape or make some other manual adjustment. On pressing "engage" the reading continues as before. The remainder of such a block is always ignored.

2.7 Consolidated Leaders

Block identifiers *14 to *17 indicate leader information - this is more fully described in section 5. Only the first leader is recognized - any subsequent leaders will be ignored.

2.8 Other block identifiers

Codes *05 - *12 will cause EXECUTIVE to type

LOADER ERROR #01

Codes *20 - *77 will be ignored whenever they occur.

3.0 Semi-compiled program

Semi-compiled program consists essentially of a field of characters of which the first one is a label (L), normally a field consists of five characters, the last four of which form a 24 bit word (N). The labels and the corresponding interpretation of the rest of the field are:-

L	Function
*00	N is mask for following word
*01	Add N to T
*02	Set DT at N
*03	Ignore next N characters
*04	End of block
*05	Set T at N (S not added by loader)
*06	
*07	
*10	Set RR1 from Cue table
*11	Set RR2 from Cue table
*12	Set RR3 from Cue table
*13	Set RR4 from Cue table
*14-*17	Illegal
*20	Ignore L only
*21-*37	Illegal
@ *40	Load word at T, add DT to T
A *41	Add RLW to N, Load at T, add DT to T
B *42	Add RLV to N, Load at T, add DT to T
C *43	Add Rel. 3 to N, Load at T, add DT to T
D *44	Add RCN to N, Load at T, add DT to T
E *45	Add RLT to N, Load at T, add DT to T
F *46	Add Rel. 6 to N, Load at T, add DT to T
G *47	Add RUP to N, Load at T, add DT to T
H *50	Add Rel. 8 to N, Load at T, add DTto T
I *51	Add RUW to N, Load at T, add DT to T
J *52	Add Rel. 10 to N, Load at T, add DT to T
K *53	Add RPR to N, Load at T, add DT to T
L *54	Add RR1 to N, Load at T, add DT to T
M *55	Add RR2 to N, Load at T, add DT to T
N *56	Add RR3 to N, Load at T, add DT to T
O *57	Add RR4 to N, Load at T, add DT to T
P *60	Set T at N
Q *61	Set T at N+RLW
R *62	Set T at N+RLV
S *63	Set T at N+Relativizer 3
T *64	Set T at N+RCN
*65	Set T at N+RLT
*66	Set T at N+Relativizer 6
*67	Set T at N+RUP

*70	Set T at N+Relativizer 8
*71	Set T at N+RUW
*72	Set T at N+Relativizer 10
*73	Set T at N+RPR
*74	Set T at N+RR1
*75	Set T at N+RR2
*76	Set T at N+RR3
*77	Set T at N+RR4

3.1 Normal loading

Usually a word to be loaded needs to have one of the relativizers added to it and for it to be stored in one of the data areas. For example the instruction:-

ADX 6 TIME

would require a transfer address setting (suppose it is instruction #203)

e.g. L = *73 N = 203

TIME would be identified by the compiler as Lower Variable #14, and ADX as numeric function code 001.

Thus the semi-compiled version of this instruction would be:

L = *42 N = 001 6 0 14

And if RPR had been set at 316 and RLV at 89, the instruction would be loaded at 519 as (001 6 0 103)

Note that the addition of the relativizer is to the least significant 15 bits only and no further carry is propagated.

3.2 Masked loading

Occasionally a compiler is not able to determine an address at the time of compilation, as is the case of forward references to instructions. The compiler then will compile the function part of the instruction as usual and will later arrange to insert the address when it is known.

For example consider the instruction:

(200) BNZ 6 END

When the compiler reaches this instruction it has not yet located the label END. It would thus compile

L = *73 N = 200
L = *40 N = (052 6 0 0)

Later in the compilation it will identify END as say 563 and will output

L = *73 N = 200
L = *00 N = *77777
L = *53 N = 563

The masking operation replaces those bits of the original stored word with corresponding bits from the new relativized word. The masking is performed on the next load operation only and this operation must occur in the same block as the mask setting.

3.3 Ignoring characters

The label L = *03 is designed to permit compilers to reproduce source program and comments for listing purposes and for the loader to ignore such information.

3.4 Transfer Address Increments

The transfer address is normally incremented by unity after each load operation. The transfer address increment DT may be reset at any time to any value (in particular to -1 for loading backwards). However, such a setting will not carry from one program segment to another and DT is reset to 1 automatically when a title block is read. In additional label L = *01 can cause T to be incremented by N (note that any increment at the previous load is not ignored).

3.5 End of block marker

All blocks of semi-compiled program must be terminated by an end of block marker (L = *04). This causes all subsequent information in the block to be ignored.

3.6 Special Relativizers

Four special relativizers, labelled RR1 - RR4 are provided to permit use of general purpose routines with multiple entry points. By convention the first two are used for the I/O Package and an appropriate arithmetic package.

The values of these relativizers are set by labels L = *10 - *13. Following the label is a sequence of characters terminated by space (*20), giving the name of the cue. The cue list is searched and the relativizer set to the corresponding value. If the name is not in the cue list, the loader will be suspended after typing

LOADER ERROR #02

3.7 Other labels

The label L = *20 (corresponding to space) is always ignored and the next character in sequence taken as the label.

Any labels not allocated a meaning will cause the loader to be suspended after typing

LOADER ERROR #03

consolidated lead r).

The normal entry point is assumed to be in location 20.

4.0 Entering the program

Whenever the loader comes across an end of segment block, it examines the cue list to determine whether all the preset segments that are required, have in fact been read. If some are still to come the loader will switch to searching for the next title block. If all segments have been read the following action is taken:-

(a) Program is moved down S locations (N.B. S is a parameter supplied by the compiler as part of the consolidated leader).

(b) The area immediately above the program area is cleared for S locations. (N.B. The address at which clearing will commence is taken as the original setting of RUW in the consolidated leader).

(c) The area (if any) between location 30 and the start of the preset information is cleared. (N.B. The address, up to which the store will be cleared will be given by the original setting of RCN in the consolidated leader).

(d) The program is entered at location 20.

4.1 Setting of Entry Points

Locations 20-29 are reserved for program entry points. These locations should be set by the compiler in absolute form using the binary load facilities (Block type *01) or else it should arrange to set the transfer address using label *05 to ensure that the word is stored in the area 20-29 which is not affected by the final shift.

An entry point is normally an unconditional branch to an appropriate location in the program area.

The normal entry point is assumed to be in location 20. If this location is not set by the compiler the loader will type

LOADING COMPLETE

on reaching this point and suspend itself. The operator should then type (for example)

GO 0#NAME AT 75

to activate the program.

5.0 The Consolidated Leader

Leader information produced whenever a compilation is concluded. At this stage the program may or may not be complete. If it is not complete the leaders must be processed by the consolidation phase which produces a consolidated leader.

A consolidated leader consists of blocks containing:

(a) Relativizers

(b) Parameters

(c) Cues

(d) Terminator

If multiple copies of (a), (b) and (d) are included in the program, the first only is read, all others being completely ignored.

Cues may be read in any order and they are added to the cue list when read. All cues following the terminator are ignored.

When the terminator block is read the loader checks that at least one each of (a) and (b) have been read. Otherwise it will type:-

LOADER ERROR #06

and suspend itself.

All incomplete leader information is always ignored.

5.1 Relativizers

A relativizer block consists of the first character (B=*15) followed by ten octal fields giving the initial settings of relativizers 1-10. Fields are separated by at least one space.

When this block is read, Relativizers 4 and 9 are stored for future use to determine areas to be cleared.

5.2 Parameters

The Parameter block consists of the first character (B=*16) followed by two octal fields giving the values of the various parameters these are:

Parameter #0	S	(Temporary displacement of program whilst loading)
Parameter #1	Instruction (This is an instruction which is obeyed immediately before the shifting operations described in section 4.0, . It is usually either a null order or else a command to release the input peripheral).	

5.3 Cue Block

A cue block consists of:

(a) Block identifier (B=*17)

(b) Single Octal field giving type and value

(c) Alphanumeric field giving the name

The octal and alphanumeric fields are terminated by at least one space character.

The type and value of a cue is specified by the cue "data word" which is split into the 9 most significant and the 15 least significant bits. The 15 least significant bits determine the "cue value" and the 9 most significant define the "cue type".

5.4 Cue Types

The type bits are numbered from B0 to B8 (B0 being the "sign" bit of the data word. Meaning is allocated as follows:

B0	=1	for preset areas (program, constants etc)
	=0	otherwise
B1	=1	for areas which must be in lower memory
	=0	otherwise
B2	=1	Spare

B3-5	=0	Blank Cue (Value =0)
	=1	Program Cue
	=2	Entry Cue
	=3	Area Cue
	=4	Peripheral Cue
	=5-7	Spare
B6-8	Spare	

(c) Blank cue and spare cue

5.5 Cue Values

The value part of a cue is used as follows:

Program Cues	in ordinary leaders	=	length of program segment
	in consolidated leaders	=	starting address of segment
Entry Cues	in ordinary leaders	=	address of entry relative to start of segment
	in consolidated leaders	=	address of entry
Area Cues	in ordinary leader	=	size of area
	in consolidated leader	=	starting address of area
Peripheral Cues	least significant 6 bits used only Peripheral type (0-7) followed by Unit number (0-7)		

6.0 Reading and writing modes

The modes of reading by the loader, and writing by the compiler are defined below to ensure compatibility.

6.1 Paper Tape

The loader reads paper tape always in mode 10. This has the following properties:

(a) Binary mode

(b) 80 characters or to newline

(c) Blank tape and erase ignored

(d) TC3 to TC1 ignored

(e) TC4 switches off line

A compiler should punch a block of N characters using mode 8 and follow this with newline and blank tape. To produce this the compiler should output the two words:-

*77762076

*20762000

in mode 0, using a control word count of 7. This gives newline followed by 3 blank characters.

Leader information which may be typed will be accepted by the loader but it should be typed in strict adherence to the restrictions given in section 5 and use only characters in the restricted binary set.

6.2 Punched Cards

Punched cards will always be read as alphanumeric characters (mode 0) and should be punched in the same way.

6.3 Magnetic Tape

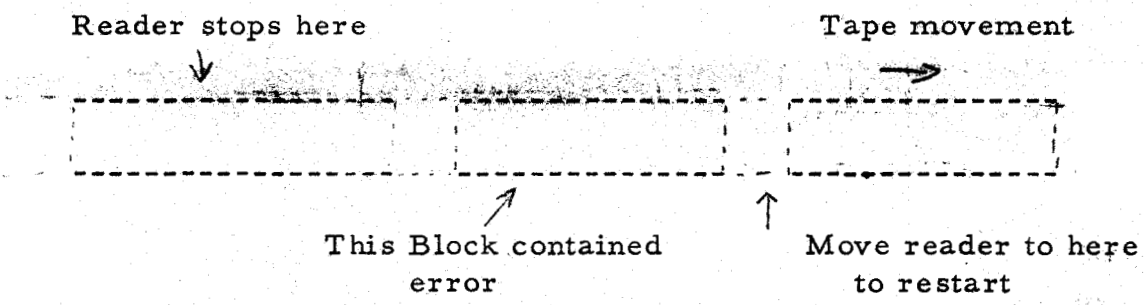
Magnetic tape is read in mode 0 using a standard block length of 20 words. It is assumed that the information was put on the tape by a card-tape or paper tape-magnetic tape transcription program which would write on tape in mode 1.

7.0 Error Messages

When the loader detects an error in the data presented to it (e.g. an illegal identifier) it will print a message of the form:-

LOADER ERROR #03

and will be suspended with the input peripheral disengaged. To restart, the operator should move back two blocks, that is reinput the last two cards in the output stack or move the paper tape back to the blank section before last:-



The peripheral should then be re-engaged and a message typed on the console typewriter

GO 0#NAME

where "NAME" is the name of the program being loaded.

7.1 Error message Summary

The error messages and their meanings are:-

01	Illegal Block Identifier (see section 2.8)
02	Use of Cue name not in table (see section 3.6)
03	Illegal label used in semi-compiled program (see section 3.7)
04	Check Sum Error (see section 2.1)
05	Title block in middle of program segment (see section 2.4)
06	Incomplete Consolidated Leader (see section 5.0)