

— CONTENTS —

INTRODUCTION	1
LOADING ADDRESS	2
RELATIONSHIP BETWEEN THE EXECUTION ADDRESS AND LOADING ADDRESS	3
SYSTEM FILE GENERATION	6
OFFSET	7
LOADING ADDRESS AND EXECUTION ADDRESS (ORG STATEMENT)	8
SYMBOL TABLE	9
LINK/T COMMAND	10
LINK MESSAGE EXAMPLES	12
ERROR MESSAGES	13

CONTENTS

1	INTRODUCTION
2	LOADING ADDRESS
3	RELATIONSHIP BETWEEN THE EXECUTION ADDRESS AND LOADING ADDRESS
6	SYSTEM FILE GENERATION
7	OFFSET
8	LOADING ADDRESS AND EXECUTION ADDRESS (ORG STATEMENT)
9	SYMBOL TABLE
10	LINK \T COMMAND
12	LINK MESSAGE EXAMPLES
13	ERROR MESSAGES

INTRODUCTION

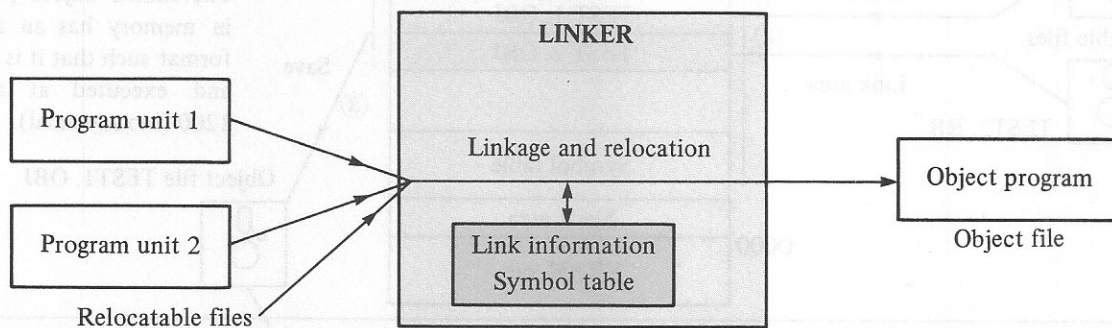
The linker for the SHARP MZ-80 series inputs relocatable files output by the assembler or the BASIC compiler and outputs object files.

Relocatable files are not programs which are directly executable by the CPU, but are files which contain information used to keep programs relocatable. They also contain global symbols in ASCII code which are declared to link two or more program units.

The linker fetches relocation information and loads object programs into the link area in main memory while adding the programmer-specified loading address to the relocatable addresses. When two or more relocatable program units are loaded, units are appended to the first program unit (file), if the loading address is specified for the first unit.

The linkage operation itself is described in detail in Section 2.3, "Linker" of the System Command manual. However, the programmer does not need to be aware of details of the linkage operation details.

When outputting the object program (object file), it is necessary to specify the loading address and the execution address.

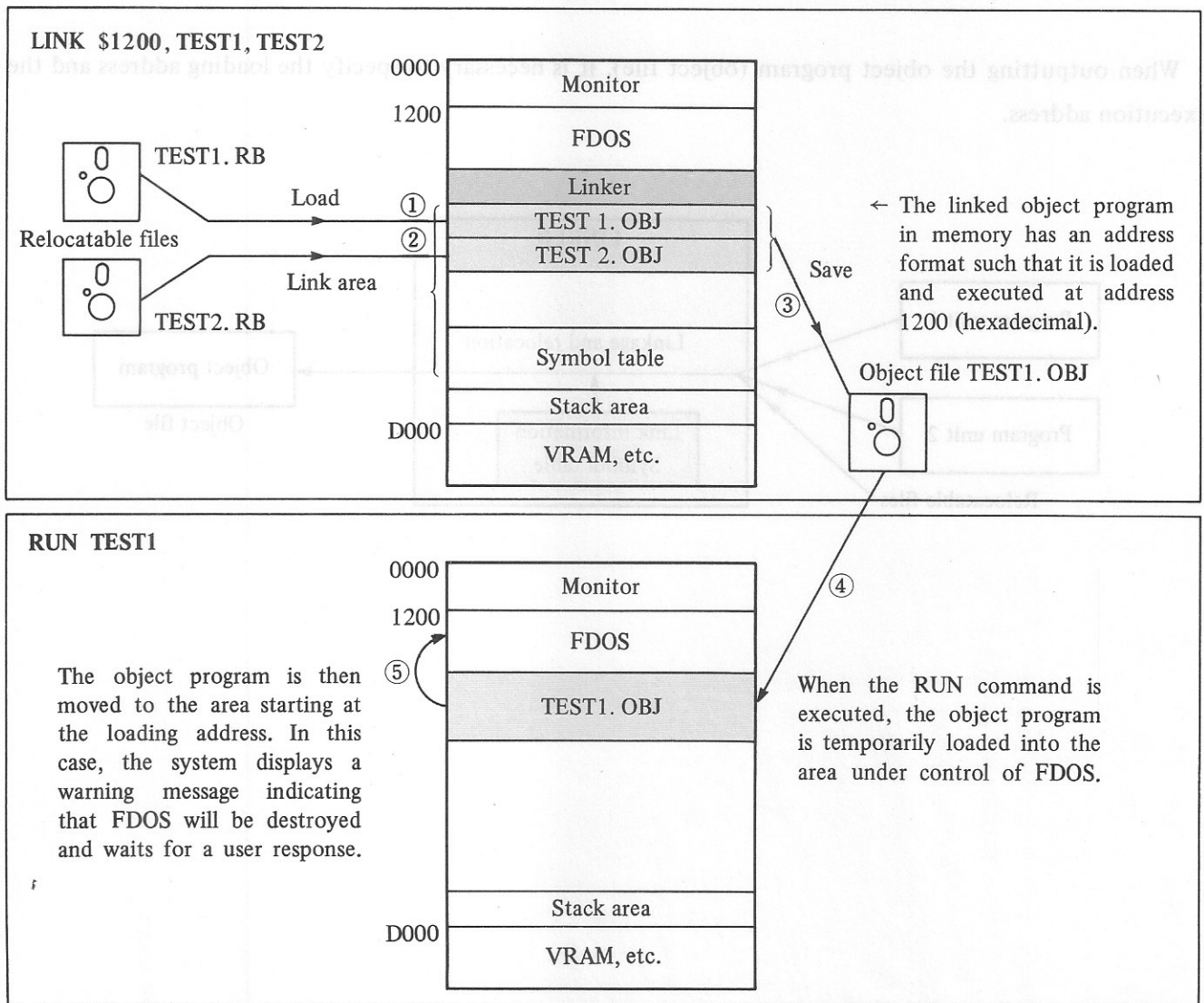


LOADING ADDRESS

The loading address specifies the address at which the object program is to be loaded. When this address is not specified, FDOS assumes the starting address which can be managed by FDOS as loading address.

LINK TEST1, TEST2	Links TEST1 and TEST2 and assigns the loading address to the beginning of the area managed by FDOS.
LINK \$1200, TEST1, TEST2	Links TEST1 and TEST2 and assigns the loading address to 1200H.

The figure below shows the flow of files from the time they are linked by the linker until they are executed with the RUN command. Numbers ① through ⑤ in the figure denote the processing sequence.



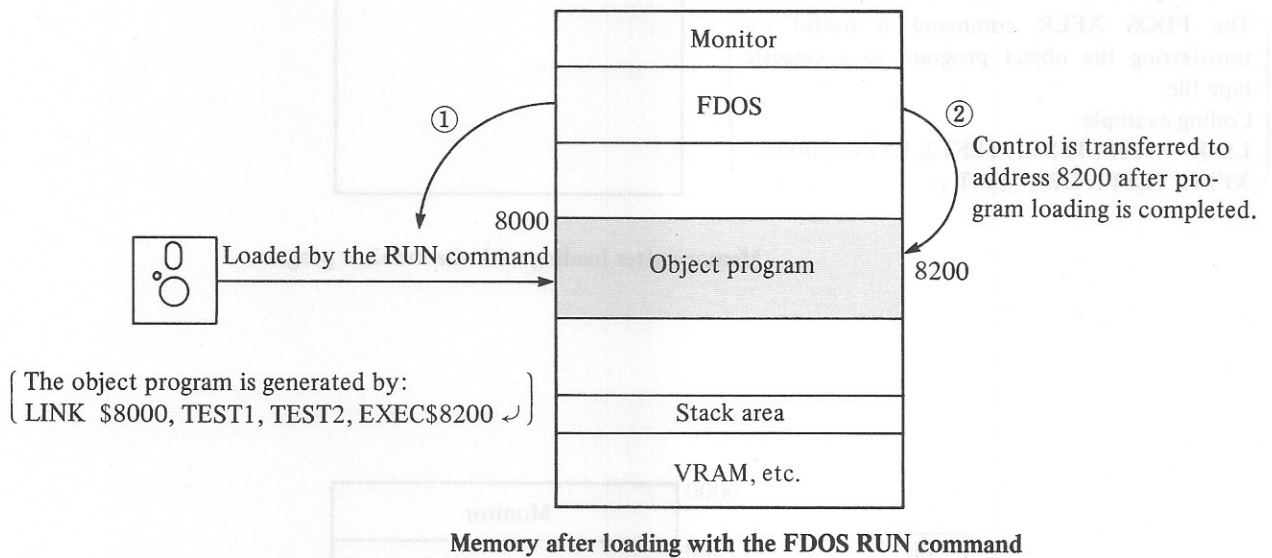
RELATIONSHIP BETWEEN THE EXECUTION ADDRESS AND LOADING ADDRESS

The programmer may specify the execution address as well as the loading address when outputting an object file through the linker.

`LINK $8000, TEST1, TEST2, EXEC$8200`

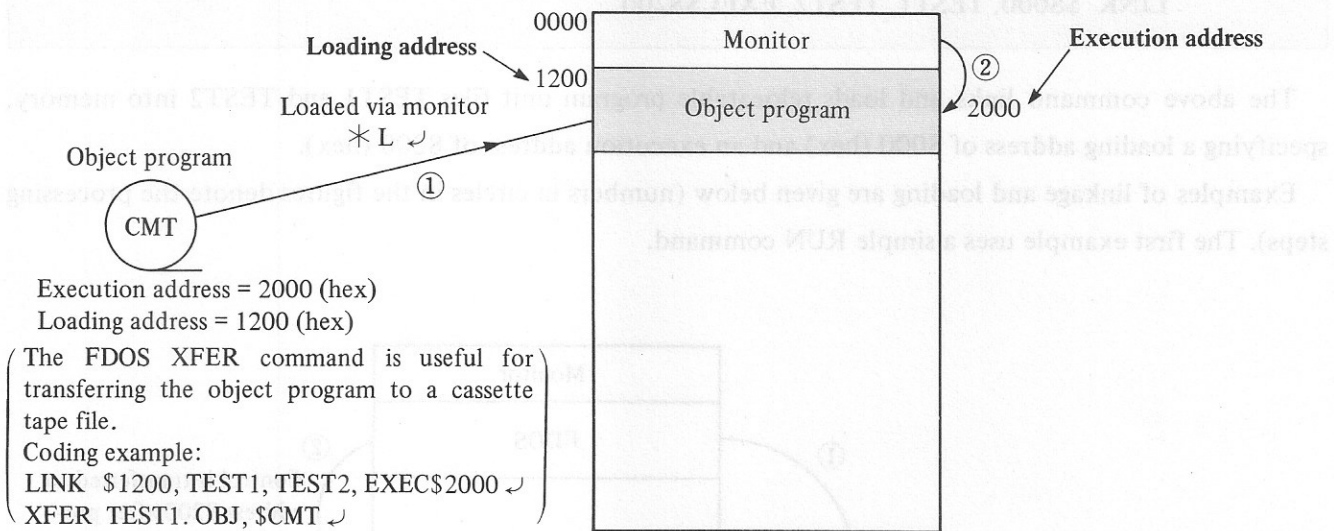
The above command links and loads relocatable program unit files TEST1 and TEST2 into memory, specifying a loading address of 8000 (hex) and an execution address of 8200 (hex).

Examples of linkage and loading are given below (numbers in circles in the figures denote the processing steps). The first example uses a simple RUN command.

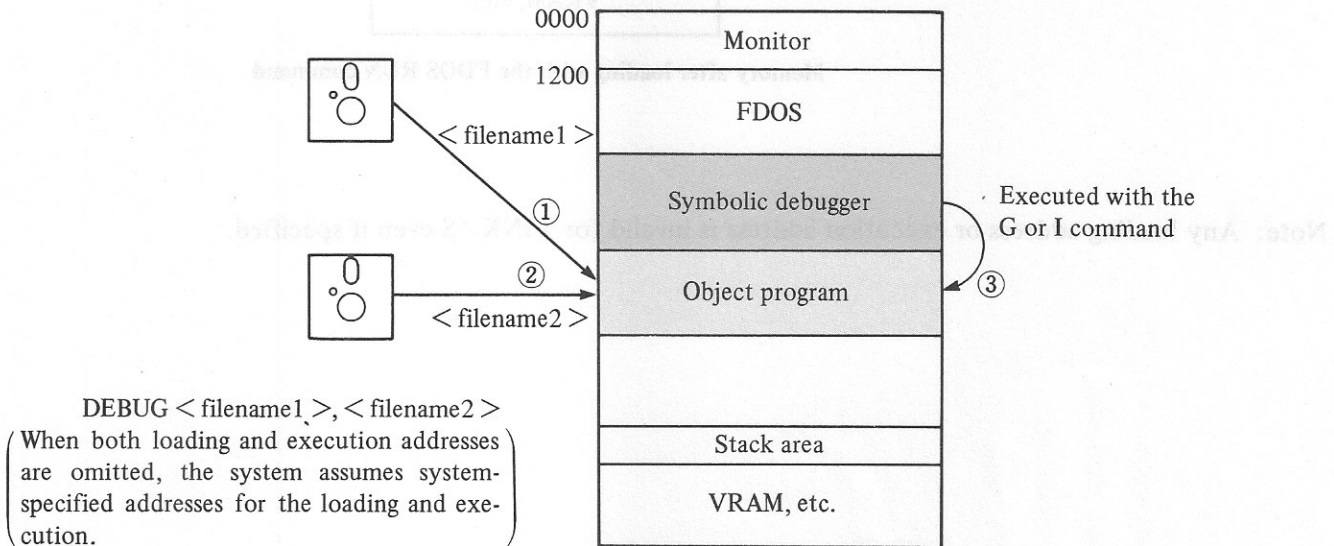


Note: Any loading address or execution address is invalid for LINK/S even if specified.

When the monitor is used to load the object program, its starting address in memory is designated by the loading address. The program counter is set to the address designated by the execution address after the object program is loaded. The figure below shows how an object program with a loading address of 1200 and an execution address of 2000 is loaded and how control is transferred.



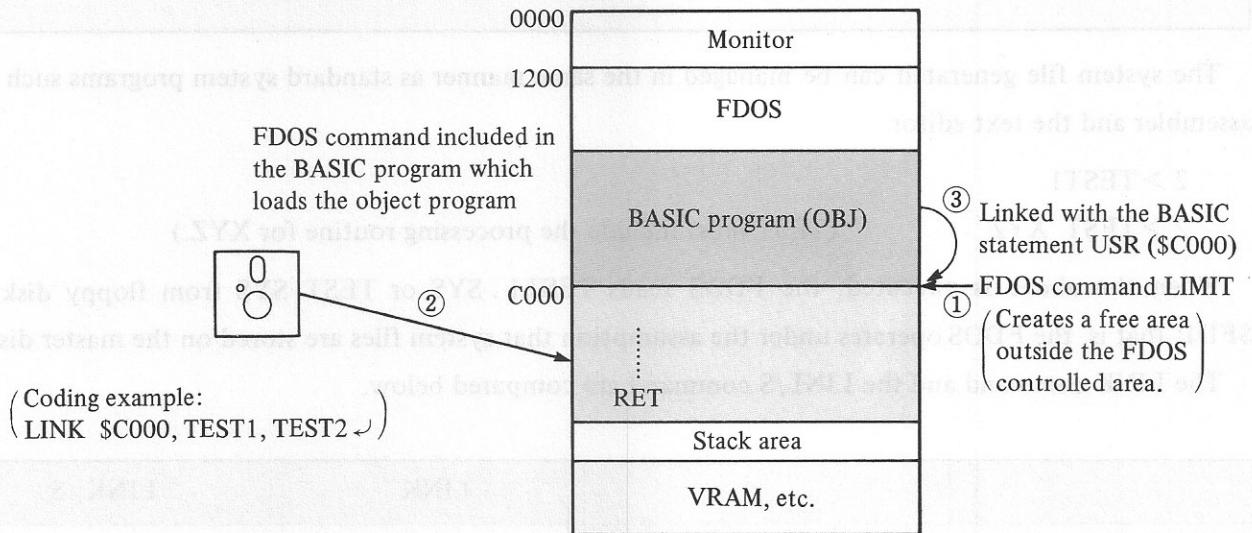
Memory after loading with the monitor program



Memory after loading with the symbolic debugger

Subroutine programs created with the assembler and BASIC programs created with the BASIC compiler may be linked using a library (see the "Programming Utility" manual) or the BASIC USR statement. Here, an example is given of linking an object program with a BASIC program using the USR statement.

The figure below shows how an object program is loaded and linked with a BASIC program. The area in memory which is managed by FDOS is reduced with the FDOS LIMIT command to create a free area. The object program is loaded into this free area with the FDOS or BASIC LOAD statement. The BASIC program can then call the object program as a subroutine using the USR() statement.



Memory after loading with an FDOS command in a BASIC program

SYSTEM FILE GENERATION

Multiple relocatable files can be linked into a system file for management by the FDOS.

LINK /S TEST1, \$FD1; FDOSEQU . LIB	Generates system file TEST . SYS by linking TEST . RB and FDOSEQU . LIB.
LINK /S TEST1, TEST2, \$FD1; TEST/O	Generates system file TEST . SYS in \$FD1 by linking TEST1 . RB and TEST2 . RB.

The system file generated can be managed in the same manner as standard system programs such as the assembler and the text editor.

2 > TEST1

2 > TEST XYZ (TEST must include the processing routine for XYZ.)

When the above is executed, the FDOS reads TEST1 . SYS or TEST . SYS from floppy disk drive \$FD1; that is, the FDOS operates under the assumption that system files are stored on the master diskette.

The LINK command and the LINK/S command are compared below.

	LINK	LINK /S
File generated	Object file	System file
Global switches/T,/P	Valid	Valid
Table size specification (TBL\$)	Valid	Valid
Loading address specification	Valid	Invalid
Execution address specification (EXEC\$)	Valid	Invalid
Offset specification	Valid	Invalid
ORG command	Available	Unavailable

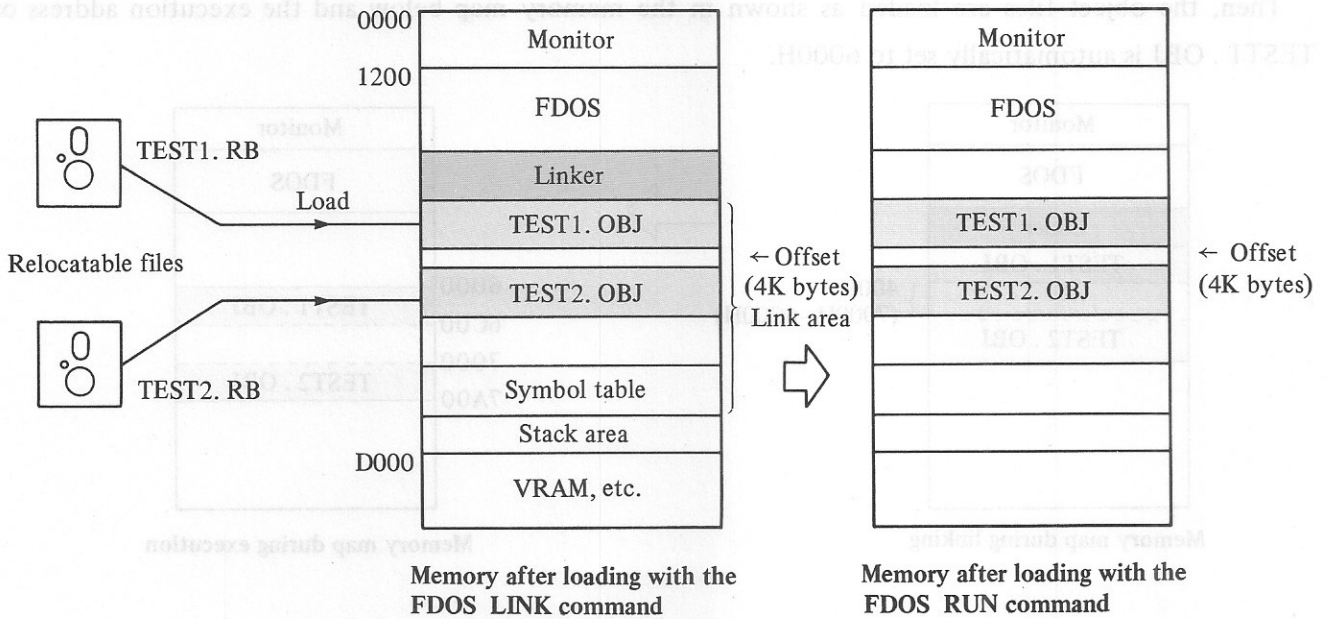
OFFSET

The programmer can specify an offset to reserve a free area between two object program units.

LINK TEST1, \$1000, TEST2

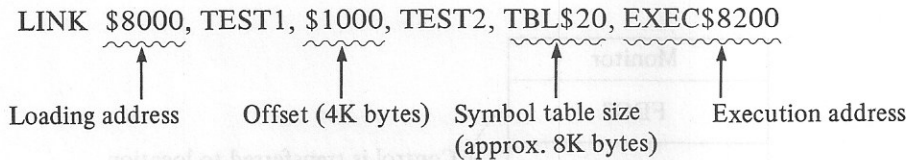
Links TEST1 and TEST2 so that the object program is loaded at the area equivalent to 1000 (hex) addresses reserved between them.

Execution of the above command is illustrated below.



Note that the loading address and offset are carefully distinguished in the following command:

A 4-digit hexadecimal number preceded by a \$ symbol in the first argument position is always interpreted as the loading address.



Note: Any loading address or execution address is invalid for LINK/S even if specified.

LOADING ADDRESS AND EXECUTION ADDRESS (ORG STATEMENT)

Although a loading address can be specified with the linker, it can also be specified with the ORG assembler directive during assembly. Assume that there are two relocatable files.

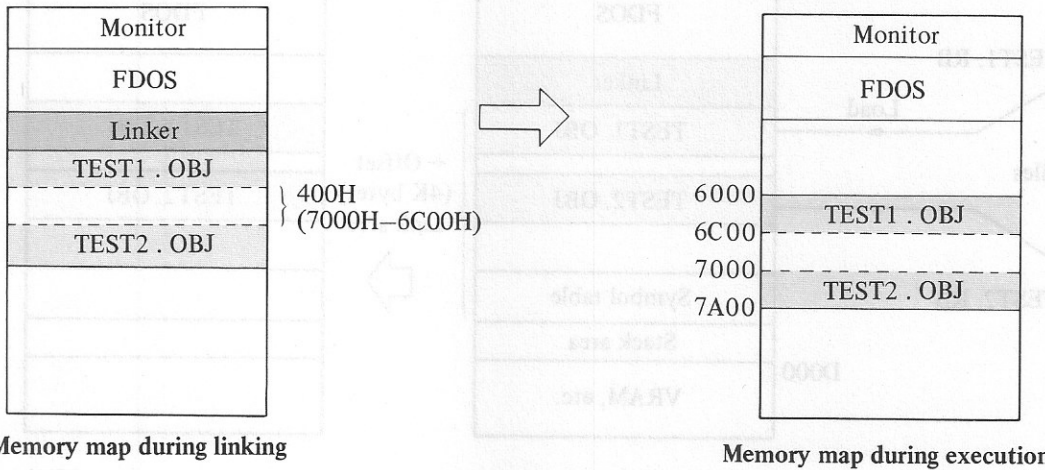
TEST1: Assembled with loading address 6000H specified. The object file will be loaded in the area from 6000H through 6C00H.

TEST2: Assembled with loading address 7000H specified. The object file will be loaded in the area from 7000H through 7A00H.

These are linked as follows.

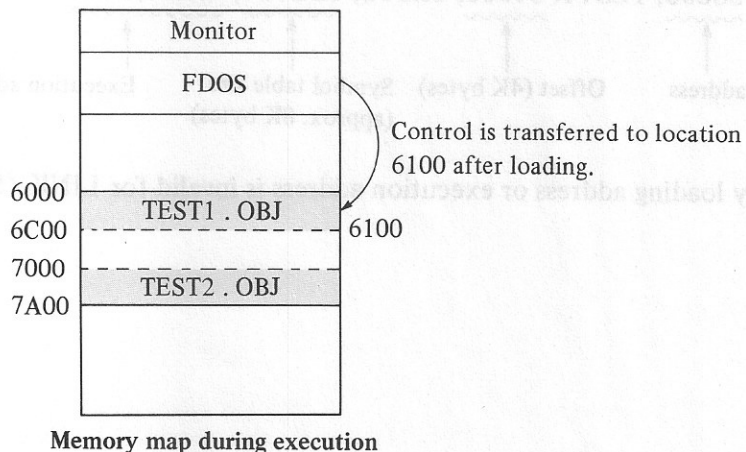
LINK TEST1, TEST2

Then, the object files are loaded as shown in the memory map below and the execution address of TEST1 . OBJ is automatically set to 6000H.



The loading addresses specified during assembly are valid even if the loading addresses and offsets are specified in the LINK command. However, when no loading address is specified for TEST2 during assembly, the offset specified in the LINK command is valid. The execution address specified in the LINK command is valid.

LINK \$5000, TEST1, \$3000, TEST2, EXEC\$6100



Loading addresses specified during assembly are invalid when the LINK/S command is used to generate a system file.

SYMBOL TABLE

Information referred to as symbols in the linker and symbolic debugger indicates globally declared labels (that is, label symbols defined by the ENT or EQU assembler directive) in the source program. This information is stored in the relocatable file by the assembler for use in linking with other relocatable programs.

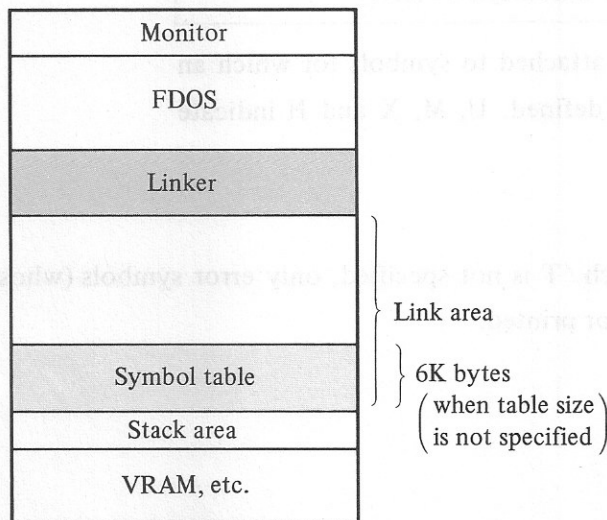
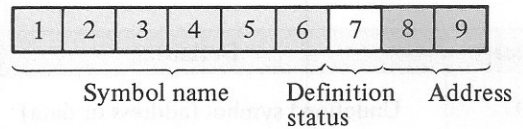
The linker loads label symbols into the symbol table while inputting program units in the relocatable files. The symbol table is placed at the end of the link area; its size is set to approximately 6K bytes by the linker unless otherwise specified by the programmer. The programmer can specify an area of more than 6K bytes for the symbol table area using the LINK command as follows:

LINK TEST1, TEST2, TBL\$20

This command links TEST1 and TEST2 and specifies a symbol table size of 2000H (approximately 8K bytes).

TBL\$20 in the above command specifies that a symbol table of approximately 8K bytes is to be created. In other words, the programmer can reserve a symbol table area in 256-byte units. As shown in the memory map, the symbol table is constructed at the end of the link area.

Each symbol table entry is 9 bytes long. The format of the symbol table entry is shown at right. Section 2.3, "Linker" in the System Command manual describes how the linker uses this 9-byte information to link relocatable program units.



Linker memory map

LINK/T COMMAND

The LINK/T command is used to display the contents of the symbol table after program linking is completed. It displays a symbol name, its absolute address (in hexadecimal representation) and the definition status for each symbol table entry. The user can detect symbol definition errors by checking the definition status.

The LINK/T command has two basic formats:

LINK/T TEST1, TEST2	Links TEST1 and TEST2 and displays the symbol table on the CRT screen.
LINK/T/P TEST1, TEST2	Links TEST1 and TEST2 and prints the symbol table on the printer.

— The photo at right shows link and symbol table information displayed on the CRT screen with the LINK/T command for the three program units shown on Page 12. Undefined symbols are labeled "U".

```

2>LINK/T UNIT-#1,UNIT-#2,UNIT-#3
Linking UNIT-#1.RB
  Top asm.bias $4A00
  End asm.bias $4A1A
Linking UNIT-#2.RB
  Top asm.bias $4A1A
  End asm.bias $4A2B
Linking UNIT-#3.RB
  Top asm.bias $4A2B
  End asm.bias $4A42
Save UNIT-#1.OBJ
Loading address $4A00
Execute address $4A00
Bytesize $0042

Symbol table
COUNT 4A03 X COUNT0 D 0050
COUNT1 D 0000 COUNT2 D 0000
PEND 4A13 M START 0000
TMDLYL 5E83 U
2>
  
```

— Symbol definition messages are listed below.

Message	Definition
U	Undefined symbol (address or data)
M	Multi-defined symbol (address or data)
X	Cross-defined symbol (address or data)
H	Half-defined symbol (data)
D	EQU-defined symbol (data)

No message is attached to symbols for which an address has been defined. U, M, X and H indicate error conditions.

— If global switch/T is not specified, only error symbols (whose definition messages are U, M, X or H) are displayed or printed.

The listing below shows a printout of link and symbol table information. The symbol table entries have been sorted as may be seen from this listing.

```

Linking M-LANG#1.RB
  Top asm.bias $4A00
  End asm.bias $5678
Linking M-LANG#2.RB
  Top asm.bias $5678
  End asm.bias $5B14
Linking MONEQU.LIB
  Top asm.bias $5B14
  End asm.bias $5B14
Save M-LANG.OBJ
  Loading address $4A00
  Execute address $4A00
  Bytesize $1114

```

Symbol table			
&MSG	57C2	&NL	57B4
1HEX0	574D	1SET	57F7
4HEX0	5776	4SET	5832
?FEED	52D6	?TABP	5374
@ERR3 D	0005	@ERR4 D	0006
BDRIVE	4CF9	BELL	0A80
BPFLG	5A57	BPSIM	5A40
BRKEY	0527	BRST8 D	0039
BWRIT	50A4	CLBF0	4CED
CLBF3	4CF0	CLBP	5678
COMMON	5511	COMPL	4BFC
CONT9	5635	CR D	00D8
DI	0E0E	DM D	00DC
ERCODE	4CF2	ERJMP	4CF5
ERTRK	4CF3	ESCPRT	52DA
GET1	56DF	GET1K	5741
GET44	572C	GETKY	0610
HS D	00DD	IBUFE	1180
LIST	4A9F	LISTA	529C
LISTS	529A	LOAD	4ABD
LPNT	5A3D	MAIN1	4A3A
MEMRY	4BA1	MES0	5914
MES11	5990	MES12	59A8
MES15	59E3	MES16	59E4
MES19	59F3	MES2	593D
MES22	59FE	MES23	5A05
MES3	5944	MES4	594E
MES7	5972	MES8	597E
MTFG	4CEC	MTOFF	4D71
NLMSG	57BF	PMSGX	5279
POTFF D	00FF	PRNT	063C
PROTC	57CF	PRTAB	52AE
PUSHR	0DF1	PWORK	58C6
REGST	4C07	RETRY	4CF1
SAVE	4ABA	SCOMP	56B9
SEEK	4D5A	SFREE D	0000
SPROG	56C1	STAFG	5222
TYPE0	577F	TYPE1	5782
VERIFY	5182	VRFCNT	4CF8
WRK0	5A4E	WRK1	5A4F
WRK4	5A52	WRK5	5A53
X2HEX	5765	XFER	4B30
XGET22	571D	XGET4	5734
ZAF	5A5B	ZAFC	5A63
ZDE	5A5F	ZDEC	5A67
ZIR	5A73	ZIX	5A6F
ZSP	5A6D		
&PRNT	57A1	&PRNTS	5796
2HEX0	5760	2SET	5823
0000T	5888	??KEY	0D77
@ERR1 D	0003	@ERR2 D	0004
ACCUM	4BF7	ARST7 D	0038
BKTBL	5A19	BPDUM	5A58
BREAD	4F9B	BREAK	4C7C
BUFFR	5A74	BUSY	4F3A
CLBF1	4CEE	CLBF2	4CEF
CLEAR	4A82	CMD	5A3F
COMPR	5688	CONT	5565
CTBL	58E6	CURSOL	5375
DR D	00DB	EFREE D	FFF0
ERJPAD	4CF6	ERSECT	4CF4
FCMD	4CEB	FDERR	4AB5
GET2	5716	GET4	5731
GETL	0BE5	GOTO	4CB0
JRTBL	58B2	LFLG	5A5A
LISTM	5263	LISTN	5256
LOOK0	586B	LOOK1	5878
MAIN2	4A48	MELDY	0AA3
MES1	5933	MES10	598A
MES13	59C1	MES14	59D8
MES17	59E7	MES18	59ED
MES20	59F7	MES21	59F9
MES24	5A0B	MES25	5A13
MES5	5959	MES6	5966
MES9	5985	MSG	06B5
MTON	4D44	NL	0757
PNL	5251	POTFE D	00FE
PRNTS	063A	PROG	4C01
PTAB0	58BD	PTAB1	58C3
RCLB	4F13	READY	4CFC
RSTRT	552B	SACC	569B
SCR D	00DA	SEARCH	583D
SKPBL	57E7	SOUND	568E
START	4A00	TR D	00D9
TYPE2	578C	TYPE3	578F
WARN	5A48	WRITE	4C13
WRK2	5A50	WRK3	5A51
WRK6	5A54	X1HEX	5752
XGET1	56E2	XGET2	5719
XTEMP	09BE	X80TB	589D
ZBC	5A5D	ZBCC	5A65
ZHL	5A61	ZHLC	5A69
ZIY	5A71	ZPC	5A6B

(Note: This listing is not related to the programs on page 12.)

LINK MESSAGE EXAMPLES

First program unit loaded (UNIT-#1)

```

TMDLYH :   LD      HL, START
COUNT :   ENT
           DEC     HL
           LD      A, H
           CP      COUNT0
           JR      NZ, COUNT
           LD      A, L
           CP      COUNT1
           JR      NZ, COUNT
           CP      COUNT2
           JR      NZ, COUNT
           RET
PEND :     ENT
           DEFM    'TMDLYH '
           DEFB    0DH
COUNT1 :  EQU     00H
COUNT0 :  EQU     50H
           END
    
```

Second program unit loaded (UNIT-#2)

```

TMDLYL :   LD      HL, START
LOOP1 :    DEC     H
           LD      A, H
           CP      COUNT
           JR      NZ, LOOP
           RET
PEND :     ENT
           DEFM    'TMDLYL '
           DEFB    0DH
START :    EQU     1000H
COUNT :   EQU     00H
           END
    
```

Third program unit loaded (UNIT-#3)

```

INPUT :    CALL    001BH
           CALL    TMDLYL
           CALL    001BH
           LD      HL, START
           CP      0DH
           JR      Z, END
           LD      (HL), A
           INC     HL
           JR      INPUT
END :      JP      0000H
COUNT2 :  EQU     12
           END
    
```

Refer to photo on page 10.

"START" X

START is not defined as an address in the first program, but is defined as data in the second or subsequent program with the START: EQU statement.

Note:

The EQU statement should be placed at the beginning of the program unit.

"COUNT2" H

COUNT2 is not defined as data in the first program, but is defined as data in the third program with the COUNT2: EQU statement.

"COUNT1" D

COUNT1 is defined as data (D indicates no error condition).

"COUNT" X

COUNT is defined as an address in the first program while it is simultaneously defined as data in the second program.

"PEND" M

PEND is defined as an address in the first program while it is simultaneously defined as an address in the second program (duplicated definition).

"TMDLYL" U

TMDLYL is neither defined as an address nor declared with the ENT directive in any other external program unit.

ERROR MESSAGES

The error messages issued by the linker are described in the System Command manual. Here, only error messages which require particular attention are described.

no memory space

Indicates that the symbol table is full; that is, that there are too many symbols to be cataloged. The symbol table size is set to approximately 6K bytes by the linker unless specified by the programmer. It is necessary to specify the TBL\$ argument in the LINK command to increase or decrease the symbol table size.

memory protection

Indicates that the link area is inadequate, that is, that the linked data has reached the symbol table area located at the end of the link area. In this case, MLINK command is available.

il data

Indicates that the data read from the specified relocatable file has an illegal link format. This condition may be caused by a hardware read error in the floppy disk drive or by an assembly error in the source program.

ERROR MESSAGES

The error messages issued by the linker are described in the System Command manual. Here, only error messages which require particular attention are described.

no memory space

Indicates that the symbol table is full; that is, that there are too many symbols to be cataloged. The symbol table size is set to approximately 6K bytes by the linker unless specified by the programmer. It is necessary to specify the TBLS argument in the LINK command to increase or decrease the symbol table size.

memory protection

Indicates that the link area is inadequate, that is, that the linked data has reached the symbol table area located at the end of the link area. In this case, MLINK command is available.

il data

Indicates that the data read from the specified relocatable file is not in legal link format. This condition may be caused by a hardware read error in the floppy disk drive or by an assembly error in the source program.