Chapter 13 ROM programming

This chapter describes how to store a program into a ROM in the stand-alone system. Before reading this chapter, it is recommended to thoroughly read Chapters, “6 Programming on i8086” and “12 Linker lld”.

13.1 Structure of ROM program

This section first describes segment placements of the ROM program, and then explains how to relocate the segments using the linker lld.

13.1.1 Segments

The LSI C-86 places programs and data in segments shown in Table 13.1.

	Segment name	Group name	Class name
Program with S or D model	TEXT	CGROUP	CODE
Program with P or L model	x_TEXT		CODE
Initialized data	DATA	DGROUP	DATA
Uninitialized data	BSS	DGROUP	DATA
Initialized far data	x_n_DATA		FAR_DATA
Uninitialized far data	x_n_BSS		FAR_BSS

Table 13.1: Segments generated by compiler

When storing these segments into a ROM, TEXT and x_TEXT are placed in the ROM since they are program codes. Since BSS and x_n_BSS are variable data, they must be placed in a RAM.
However, since DATA and x_n_DATA are initialized, it is necessary to store them in the ROM. Additionally, since data placed in the ROM is not changed during execution of the program, data placed in DATA and x_n_DATA is not a variable, but functions as if it is a constant. Great care should be taken on this point.

However, there are some exceptions. DATA segment containing initialized near data is automatically copied from the ROM to the RAM as the program execution is started.
Since DATA and BSS form a single DGROUP, they are gathered and placed in the RAM. Therefore, the near DATA segment is handled as an initialized variable.
This special DATA segment process is executed by the ghost segment function of the lld and initial setting module "crom" (described later) for the ROM program.

13.1.2 Segment placement

At the beginning of the initial setting module crom.p86, the following segments are declared.

```
TEXT          CSEG                        ; code
_GHOST_DATA   CSEG                  PARA  ; data in ROM, copied to RAM
FAR_DATA      DSEG  'FAR_DATA'      PARA  ; far data is placed in ROM
RESETVEC      CSEG  'RESETVEC'            ; JMPF

DATA          DSEG                  PARA  ; data in RAM
BSS           DSEG                        ; bss in RAM
ENDBSS        DSEG                        ; end of bss
XSTACK        DSEG                  STACK ; stack base
ENDSTK        DSEG                        ; stack top

FAR_BSS       DSEG  'FAR_BSS'       PARA  ; far bss
END_FAR_BSS   DSEG  'END_FAR_BSS'   PARA  ; end of far bss
```

The order of this declaration specifies the segment placements. It is necessary to put crom at the beginning during linking.

The following describes overview of each segment.

The near program codes are gathered in the TEXT segment.
The linker llb outputs the contents of the DATA segment to the next _GHOST_DATA segment. Data output to this segment is copied to the DATA segment on the RAM during initial setting of the program.
For details of the ghost segment, see section 12.4, Ghost segment.

The class of the TEXT and _GHOST_DATA segments is 'CODE'. Therefore, all far functions are gathered after the _GHOST_DATA segment.

Since the FAR_DATA segment has the class 'FAR_DATA', all of far data in the program are gathered to this segment.

The RESETVEC segment contains the JMPF command (jump between segments) that jumps to the beginning of the TEXT segment.
Initialized near data is gathered and placed in the DATA segment. However, the linker lld actually outputs the data to the _GHOST_DATA segment.

Uninitialized near data is gathered to the BSS segment. ENDBSS is a marker showing the end of BSS.
The XSTACK segment is a program stack. The default size is set to 500 bytes. However, by changing the contents of crom.p86 or specifying the size during linking, the stack size can be changed. ENDSTACK is a marker showing the top of the stack.

All of uninitialized far data is gathered to FAS_BSS. END_FAR_BASS is a marker showing the end of FAR_BSS.

13.1.3 Segment re-placement

Segments to be placed in the ROM are those from TEXT to RESETVEC. Segments after DATA should be placed in the RAM.
To re-place the segments, -T option provided by the linker lld is used. When no segment name is put on -T, the entire object is re-placed. Therefore, to give ROM addresses, it is recommended to omit the segment name as shown below.

```
-T F0000
```

To specify a RAM address, DATA is given as a segment name.

```
-TDATA 10000
```

As a result, all segments after DATA are re-placed.

13.1.4 Jump commands at reset

i8086 starts execution from an address of FFFF : 0000 at system reset. Therefore, a segment jump command is generally put at FFFF : 0000.

The segment jump command that jumps to the beginning of the TEXT segment is placed in the RESETVEC segment. When this segment is re-placed to FFFF : 0000 using -TRESETVEC FFFF0, a jump command at system reset is generated. If this jump command is not necessary, it is also not necessary to give this option. If this option is not given, the RESETVEC segment is laced at a position after the segment with a class name of FAR_DATA, that is, at the end of the ROM area.

When using -TRESETVEC FFFF0, it is also absolutely necessary to specify -TDATA *seg*. Otherwise, all segments after the segment that specifies the -T option are re-placed. Thus, all segments after the DATA segment are placed after RESETVEC.

Additionally, it is recommended not to re-place segments unless .HEX files are created. If .COM files or .EXE files are created, this may result in a huge file.

13.2 Initial setting module crom

What the ROM program must do first is several initialization processes. The module crom.p86 performs the initialization processes. (Depending on the type of installer, crom.p86 is stored in instpath\src\lib\crom.p86 under the directory instpath.)

The crom is a prototype in which general processes are described. In most cases, the crom cannot be used as it is. It is absolutely necessary that the user must change the contents of this file corresponding to the target system.