FR FAMILY
F²MC FAMILY
IN CONFORMANCE WITH μITRON 3.0 SPECIFICATIONS
SOFTUNE REALOS
CONFIGURATOR MANUAL
PREFACE

Objectives and Intended Readership

Softune REALOS (hereafter called REALOS/FR) is a real-time operating system that runs on Fujitsu FR/FMC-16/FMC-8L family controllers and microcontrollers. The Softune REALOS/FR kernel specifications comply with the µITRON 3.0 specifications. The Softune REALOS/907/896 kernel specifications comply with the µITRON 2.0 specifications.

This manual is intended for engineers who develop products using Softune REALOS. The manual describes the functions and operation of the Softune REALOS configurator. Read this manual as a reference.

Trademarks

- “TRON” is an abbreviation for “The Real-time Operating system Nucleus.”
- ITRON is an abbreviation of Industrial TRON.
- µITRON is an abbreviation of Micro Industrial TRON.
- Microsoft, Windows, Windows NT, and MS-DOS are registered trademarks of Microsoft Corporation in the United States and other countries.

Other system names and product names in this manual are the trademarks of their respective companies or organizations. The symbols™ and ® are sometimes omitted in the text.
## Configuration of This Manual

This manual consists of 3 parts.

### PART I  Configurator

**Chapter 1  Notes on Use**

This chapter contains notes concerning the creation of application systems using the Configurator.

**Chapter 2  Overview of the Configurator**

This chapter provides an overview of the configurator.

**Chapter 3  Configuration Definition Statements**

This chapter describes the structure of and lists the configuration definition statements and explains them in detail.

### PART II  Non-Windows Version

**Chapter 4  Non-Windows Version Operations**

This chapter explains the operation of non-Windows versions.

### PART III  Softune Workbench Version of the Configurator

**Chapter 5  Information for Softune Workbench Users**

This chapter provides an overview of the Workbench version of the configurator and contains notes on its use.

**Chapter 6  Operation of the Workbench Version of the Configurator**

This chapter explains how to operate the Softune Workbench version of the configurator.

**Chapter 7  Definition Dialogs of the Workbench Version of the Configurator**

This chapter explains the settings procedure and configuration of the Workbench configurator in the definition dialog.

**Chapter 8  Limitations of the Workbench Version of the Configurator**

This chapter explains the limitations on the Softune Workbench version of the configurator.

## Related Manuals

*Softune Assembler Manual*

*Softune Linkage Kit Manual*
1. The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.

2. The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.

3. The contents of this document may not be reproduced or copied without the permission of FUJITSU LIMITED.

4. FUJITSU semiconductor devices are intended for use in standard applications (computers, office automation and other office equipments, industrial, communications, and measurement equipments, personal or household devices, etc.).

   **CAUTION:**
   Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, sea floor repeaters, vehicle operating controls, medical devices for life support, etc.) are requested to consult with FUJITSU sales representatives before such use. The company will not be responsible for damages arising from such use without prior approval.

5. Any semiconductor devices have inherently a certain rate of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.

6. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Control Law of Japan, the prior authorization by Japanese government should be required for export of those products from Japan.
Softune REALOS manual set

The REALOS manual set consists of the three volumes listed below. A first-time user of REALOS should first read the “FR Family Softune REALOS User’s Guide”.

  - This manual contains the following items: product installation procedure, application system creation procedure, notes on the system, and debugging macro commands used to debug applications on REALOS with the debugger.

  - This manual contains information required to create user programs.

- **“Softune REALOS Configurator Manual”**
  - This manual explains the REALOS configurator functions and how to use them.

- **“Softune REALOS/FR/907 Analyzer Manual”**
  - Please refer to “Kernel Manual”. 
READING THIS MANUAL

■ Page Layout

In this manual, an entire section is presented on a single page or spread whenever possible. The reader can thus view a section without having to flip pages.

The content of each section is summarized immediately below the title. You can obtain a rough overview of this product by reading through these summaries.

Also, higher level section headings are given in lower sections so that you can know to which section the text you are currently reading belongs without referring to the table of contents or the cover of each chapter.

■ Products names

Product names in this manual are abbreviated as follows:

Microsoft® Windows® 95 operating system: Windows 95
Microsoft® Windows NT® Workstation operating system Version 4.0: Windows NT 4.0

■ Notation Used in This Manual

The following notation is used in this manual to represent definition statement description formats, etc.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>Indicates that the elements within the square brackets “[ ]” may be omitted.</td>
</tr>
<tr>
<td>{ }</td>
<td>Indicates a choice of one of the elements within the brackets “{ }”.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Indicates that the preceding item may be written repeatedly.</td>
</tr>
</tbody>
</table>

**Note:** Indicates an item that requires special attention. Always read these notes carefully.
CONTENTS

PART 1 CONFIGURATOR ................................................................................................................. 1

CHAPTER 1 NOTES ON USE ....................................................................................................... 3
  1.1 Notes on the Creation of Configuration Data ................................................................. 4
  1.2 Notes on the Configuration Process .................................................................................. 6
  1.3 Notes on the Softune Workbench Version of the Configurator ............................... 7

CHAPTER 2 OVERVIEW OF THE CONFIGURATOR ................................................................. 9
  2.1 What Is the Configurator? ............................................................................................... 10
  2.2 Positioning of the Configurator ....................................................................................... 11
  2.3 Configurator Functions .................................................................................................. 12
    2.3.1 Defining Configuration Data ..................................................................................... 13
    2.3.2 Executing the Configuration Process ........................................................................ 14
    2.3.3 Creating Absolute Format Object Files with the Softune Workbench Version .... 16
  2.4 Configuration Data (FR Family) .................................................................................... 17
    2.4.1 CPU Definition (FR) ................................................................................................ 18
    2.4.2 Linkage Definition (FR) .......................................................................................... 19
    2.4.3 System Definition (FR) ............................................................................................ 20
    2.4.4 Memory Definition (FR) .......................................................................................... 21
    2.4.5 System Call Definition (FR) ...................................................................................... 22
    2.4.6 Task Definition (FR) ................................................................................................. 24
    2.4.7 Common Stack Definition (FR) ................................................................................ 26
    2.4.8 Semaphore Definition (FR) ....................................................................................... 28
    2.4.9 EventFlag Definition (FR) ......................................................................................... 30
    2.4.10 Mailbox Definition (FR) .......................................................................................... 31
    2.4.11 Variable-size MemoryPool Definition (FR) ............................................................ 32
    2.4.12 Fixed-size MemoryPool Definition (FR) ............................................................... 34
    2.4.13 Vector Definition (FR) ............................................................................................ 36
  2.5 Configuration Date (F²MC-16 Family) ............................................................................ 37
    2.5.1 CPU Definition (F²MC-16) ...................................................................................... 38
    2.5.2 Linkage Definition (F²MC-16) .................................................................................. 39
    2.5.3 System Definitions (F²MC-16) .................................................................................. 40
    2.5.4 Memory Definitions (F²MC-16) ............................................................................... 42
    2.5.5 Task Definitions (F²MC-16) ...................................................................................... 44
    2.5.6 Common Stack Definitions (F²MC-16) ..................................................................... 46
    2.5.7 Semaphore Definitions (F²MC-16) .......................................................................... 47
    2.5.8 EventFlag Definitions (F²MC-16) .......................................................................... 48
    2.5.9 Mailbox Definitions (F²MC-16) ............................................................................... 50
    2.5.10 Fixed-Size MemoryPool Definitions (F²MC-16) ...................................................... 52
    2.5.11 Vector Definitions (F²MC-16) ............................................................................... 54
### CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS ................................................................. 69

3.1 Overview of the Configuration Definition Statements .......................................................... 70

3.2 Definition Statement Explanation .......................................................................................... 74

3.2.1 CPU ....................................................................................................................................... 76
3.2.2 RELOCATE_FILE .................................................................................................................... 80
3.2.3 LIBRARY_FILE ...................................................................................................................... 81
3.2.4 ABSOLUTE_FILE ................................................................................................................ 82
3.2.5 MAP_FILE ............................................................................................................................. 83
3.2.6 CYC_HDR_NUMBER ............................................................................................................. 84
3.2.7 ALM_HDR_NUMBER ............................................................................................................. 85
3.2.8 EXC_HDR_ENTRY ................................................................................................................... 86
3.2.9 CPU_EXTHDR_ENTRY .......................................................................................................... 87
3.2.10 SYS_EXTHDR_ENTRY .......................................................................................................... 88
3.2.11 SCAL_EXC_ENTRY .............................................................................................................. 90
3.2.12 PRIORITY_LEVEL ............................................................................................................... 92
3.2.13 INCLUDE ............................................................................................................................. 94
3.2.14 ILM ..................................................................................................................................... 95
3.2.15 CODE ................................................................................................................................. 96
3.2.16 SYS_CLOCK ........................................................................................................................ 98
3.2.17 CONTEXT_REGISTER ....................................................................................................... 100
3.2.18 REGISTER_BANK ............................................................................................................. 101
3.2.19 wai_tsk ............................................................................................................................. 102
3.2.20 SYS_STK_SIZE ................................................................................................................... 103
3.2.21 ROM ................................................................................................................................... 104
3.2.22 RAM ................................................................................................................................... 105
3.2.23 KNL_ROM .......................................................................................................................... 106
3.2.24 KNL_RAM .......................................................................................................................... 107
3.2.25 USR_SECTION .................................................................................................................... 108
3.2.26 SYSTEM_CALL ................................................................................................................... 110
3.2.27 TSK .................................................................................................................................... 112
3.2.28 COM_STACK ...................................................................................................................... 120
3.2.29 SEM_ICNT ........................................................................................................................... 122
3.2.30 SEM ................................................................................................................................... 124
3.2.31 FLG_BIT ................................................................................................................ ............. 128
3.2.32 FLG_PATTERN............................................................................................................ ...... 129
3.2.33 FLG.................................................................................................................... ................. 130
3.2.34 MBX.................................................................................................................... ................ 134
3.2.35 MPL .................................................................................................................... ................ 136
3.2.36 MPF .................................................................................................................... ................ 140
3.2.37 EIT_ENTRY.............................................................................................................. .......... 142
3.2.38 INIT_HDR_ENTRY......................................................................................................... .... 144
3.2.39 CALL_ENTRY ............................................................................................................. ....... 145
3.2.40 RESET_ENTRY ............................................................................................................ ..... 146

PART 2 NON-WINDOWS VERSION................................................................................................... ........ 147

CHAPTER 4 OPERATION OF THE NON-WINDOWS VERSION ......................................................... 149
4.1 Configurator Startup Command.............................................................................................. ...... 150
4.2 Option for the Monitor Debugger ........................................................................................... ....... 152

PART 3 SOFTUNE WORKBENCH VERSION OF THE CONFIGURATOR............................................... 155

CHAPTER 5 INFORMATION FOR SOFTUNE WORKBENCH USERS................................................ 157
5.1 Outline of the Workbench Version of the Configurator ................................................................. 158

CHAPTER 6 OPERATION OF THE WORKBENCH VERSION OF
THE CONFIGURATOR ..................................................................................................... 163
6.1 Starting the Workbench Version of the Configurator .................................................................. 164
6.1.1 Starting the Configurator with a New Configuration File....................................................... 165
6.1.2 Starting the Configurator by Using an Existing Configuration File........................................ 170
6.2 Editing the Configuration File............................................................................................ ............ 172
6.3 Making a Project.......................................................................................................... ................. 174

CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF
THE CONFIGURATOR ...................................................................................................... 179
7.1 Definition Dialog Organization ................................................................................................. 180
7.1.1 Types of Dialog......................................................................................................... ............ 182
7.1.2 List of File Names...................................................................................................... ........... 184
7.1.3 File Selection .......................................................................................................... .............. 185
7.1.4 Miscellaneous Lists ..................................................................................................... ......... 186
7.2 System Definition Dialog........................................................................................................ 188
7.3 Memory Definition Dialog..................................................................................................... 194
7.4 System Call Definition Dialog.............................................................................................. 198
7.5 Task Definition Dialog........................................................................................................ 200
7.5.1 Task Detail Definition Dialog ............................................................................................ 204
7.5.2 Common Stack Definition ................................................................................................. 207
7.6 Semaphore Definition Dialog............................................................................................... 208
7.7 EventFlag Definition Dialog ................................................................................................. 212
FIGURES

Figure 2.2  Configurator Positioning ........................................................................................................... 11
Figure 2.3.1  Defining Configuration Data .................................................................................................. 13
Figure 2.3.2  Executing the Configuration Process ................................................................................... 14
Figure 2.3.3  Creating Absolute Format Object Files (Softune Workbench Version) .............................. 16
Figure 2.4.11 Vanable-size MemoryPool Total Size/Number of Blocks ................................................ 32
Figure 2.4.12 Vanable-size MemoryPool Total Size/Number of Blocks ................................................ 34
Figure 2.5.10 Total size of fixed-size memorypool/block size ................................................................ 53
Figure 3.2.16 Status Transition of the System Clock Handler .................................................................. 98
Figure 3.2.28 Example Where the Common Stack Cannot Function Properly .................................... 121
Figure 4.1a  Configuration Definition Statements .................................................................................... 152
Figure 4.1b  Output Example of an Interrupt Vector Table ........................................................................ 152
Figure 4.1c  Output Example of a Jump Table ........................................................................................ 153
Figure 5.1a  [Create New Project] dialog.(FR) ......................................................................................... 158
Figure 5.1b  Example of the System Definition Dialog (FR) ................................................................. 159
Figure 5.1c  Example of Displaying Help Information about a System Call with a Single Operation .... 160
Figure 5.1d  Relationship of Softune Workbench and REALOS .............................................................. 161
Figure 6.1.1a  Example of the Softune Workbench Main Window (FR) .................................................. 165
Figure 6.1.1b Example of the [New] Dialog of Softune Workbench ....................................................... 166
Figure 6.1.1c Example of the [Create New Project] Dialog of Softune Workbench ............................ 166
Figure 6.1.1d Example of the [Create Configuration File] Dialog ....................................................... 167
Figure 6.1.1e Example of the [System] Dialog (FR) .............................................................................. 167
Figure 6.1.1f Example of the Warning Message .................................................................................... 168
Figure 6.1.2a Example of the [Create Configuration File] Dialog ....................................................... 170
Figure 6.1.2b Example of the [Set Configuration File] Dialog .............................................................. 171
Figure 6.1.2c Example of the Warning Message Dialog ....................................................................... 171
Figure 6.2a Example of the Softune Workbench Project Window .......................................................... 172
Figure 6.2b Example of the [Set Configuration File] Dialog .................................................................. 173
Figure 6.3a Example of the [Add Member] Menu Item of Softune Workbench ..................................... 174
Figure 6.3b Example of the [Add Member] Dialog ............................................................................... 175
Figure 6.3c Example of the [Build] Menu Item of Softune Workbench ............................................... 176
Figure 6.3d Example of Compilation and Linkage Execution Results .................................................. 177
Figure 7.1a Example of the Wizard Dialogs ......................................................................................... 180
Figure 7.1b Example of the Tabbed Dialogs ...................................................................................... 180
Figure 7.1.2 Example of a List of File Names .................................................................................. 184
Figure 7.1.4 Example of a Miscellaneous List ..................................................................................... 186
### TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.4.5</td>
<td>List of System Calls</td>
<td>22</td>
</tr>
<tr>
<td>Table 3.1a</td>
<td>List of Configuration Definition Names</td>
<td>70</td>
</tr>
<tr>
<td>Table 3.1b</td>
<td>List of Configuration Definition Names [16]</td>
<td>71</td>
</tr>
<tr>
<td>Table 3.1c</td>
<td>List of Configuration Definition Names [8L]</td>
<td>73</td>
</tr>
<tr>
<td>Table 3.2.16a</td>
<td>Generating the System Clock Handler Return [16]</td>
<td>98</td>
</tr>
<tr>
<td>Table 3.2.16b</td>
<td>Generating the System Clock Handler Return [8L]</td>
<td>99</td>
</tr>
<tr>
<td>Table B.a</td>
<td>FR Family Kernel Code Section and Kernel Data Section</td>
<td>235</td>
</tr>
<tr>
<td>Table B.b</td>
<td>F²MC-16 Family Kernel Code Section and Kernel Data Section</td>
<td>235</td>
</tr>
<tr>
<td>Table B.c</td>
<td>F²MC-8L Family Kernel Code Section and Kernel Data Section</td>
<td>235</td>
</tr>
<tr>
<td>Table B.d</td>
<td>Formulas for Calculating Kernel Management Areas for the FR Family</td>
<td>238</td>
</tr>
<tr>
<td>Table B.e</td>
<td>Formulas for Calculating Kernel Management Areas for the F²MC-16 Family</td>
<td>240</td>
</tr>
<tr>
<td>Table B.f</td>
<td>Formulas for Calculating Kernel Management Areas for the F²MC-8L Family</td>
<td>241</td>
</tr>
<tr>
<td>Table B.g</td>
<td>Kernel and Task Memory Size Calculation (FR Family)</td>
<td>242</td>
</tr>
<tr>
<td>Table B.h</td>
<td>Kernel and Task Memory Size Calculation (F²MC-16 Family)</td>
<td>243</td>
</tr>
<tr>
<td>Table B.i</td>
<td>Kernel and Task Memory Size Calculation (F²MC-8L Family)</td>
<td>244</td>
</tr>
</tbody>
</table>
PART 1  CONFIGURATOR

Part I provides an overview of the configurator and explains the configuration definition statements that are common to both the Windows/non-Windows versions.

Chapter 1  Notes on Use
Chapter 2  Overview of the Configurator
Chapter 3  Configuration Definition Statements
CHAPTER 1   NOTES ON USE

This chapter contains notes concerning the creation of application systems using the configurator.

1.1 Notes on the Creation of Configuration Data
1.2 Notes on the Configuration Process
1.3 Notes on the Softune Workbench Version
### Notes on the Creation of Configuration Data

**Note:**

- **Exception handler**
  
  It is necessary to define an exception handler in order to execute suitable exception processing and then continue normal processing if a CPU exception occurs in the target system.

  (->See the definition statement “EXC_HDR_ENTRY”.)

  If a CPU exception is generated without an exception handler having been defined, the user’s “system down” processing may be executed and the target system may not be able to continue with its processing correctly.

- **Interrupt handlers**

  Interrupt handlers must be defined for all interrupts that are generated in the target system.

  (->See the definition statement “EIT_ENTRY”.)

  If an interrupt for which no interrupt handler is defined occurs, the target system cannot continue its processing correctly.

- **System stack size**

  The system stack is used by the kernel and by the interrupt handler. Define the system stack size with plenty of space.

  (->See the definition statement “SYS_STK_SIZE”.)

  If the system stack size is insufficient, the target system will not operate correctly.

- **Task stack size**

  Define the task stack size with plenty of space.

  (->See the definition statements “TSK” and “COM_STACK”.)

  If the task stack size is insufficient, the target system will not operate correctly.
1.1 Notes on the Creation of Configuration Data

• Common stack

When two or more tasks that share the same common stack are simultaneously in the READY, WAIT, or other non-DORMANT state, the target system cannot operate normally.

Simultaneously operating tasks must use different common stacks or refrain from using common stacks.

(->See the definition statement “COM_STACK”.)

• Section arrangement

If areas or addresses designated by the KNL_RAM, KNL_ROM, and USR_SECTION definition statements overlap, the sections may be arranged incorrectly.

After configuring, verify the output map file or check whether the sections are arranged properly, which include standard in the linkage kit.
This section contains notes on the configuration process.

**Notes on the Configuration Process**

**Note:**

- **CTRL-C**
  
  Do not forcibly terminate the configuration process by pressing CTRL-C (pressing the “C” key while holding down the “CTRL” key).

  If CTRL-C is pressed, the Configurator may not operate correctly.

- **Configurator startup name**

  To start the configurator from the DOS prompt in Windows 95/NT, execute “frcfs.exe”.

  ( -> See Appendix A, “Notes on Using The configurator.”)

- **Using file names and directory names**

  Do not use 2-byte characters in a file name or directory name used by the configurator.

  If 2-byte characters are specified in a file name or directory name, the configurator may not operate correctly.

  When installing the configurator, do not use 2-byte characters in the installation directory name. If the configurator is installed using 2-byte characters in the directory name, it may not operate correctly.
1.3 Notes on the Softune Workbench Version of the Configurator

This section contains notes on the Softune Workbench version of the configurator.

Notes on the Softune Workbench version of the configurator

Note:

- REALOS configurator add-in module for Softune Workbench

  Softune REALOS Configurator is a REALOS function expansion module for Softune Workbench. The module is in the form of a dynamic link library (DLL) form, allowing it to be linked dynamically with Softune Workbench.

  Softune Workbench itself has no REALOS support function. If, however, Softune REALOS is installed, Softune Workbench is automatically expanded so that the REALOS support function can be used.

- Before using the Softune Workbench version of the configurator

  Before using the Softune Workbench version of the configurator, read Chapters 5 and 6 in Part III.

- Setting ROM and RAM

  If a new configuration file is created by the Workbench version of the configurator, set the ROM/RAM area with the link option setting from [Project]-[Tool Option Setting] in Softune Workbench.

- Setting the relocatable object file and library file

  If the configurator is activated and saved using an existing configuration file when a new project is created, the relocatable object file and library file in the existing configuration file are not registered in Softune Workbench.

  Register the source file or library file from [Project]-[Member Addition] in Softune Workbench.

- Comment lines in an existing configuration file

  If a configuration file containing comment lines is loaded and then saved, the comment lines do not remain in the configuration file.
CHAPTER 2  OVERVIEW OF THE CONFIGURATOR

This chapter provides an overview of the configurator.

2.1  What Is the Configurator?
2.2  Positioning of the Configurator
2.3  Configurator Functions
2.4  Configuration Data (FR Family)
2.5  Configuration Data (F²MC-16 Family)
2.6  Configuration Data (F²MC-8L Family)
CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.1 What Is the Configurator?

The configurator is a tool that is used to construct REALOS/FR systems. After assembling or compiling the tasks, handlers, etc., the user uses the configurator to define the REALOS/FR system configuration data. Using this system configuration data, the configurator allocates the initial data and work areas that are needed in order for the REALOS/FR system to run, and also conditionally assembles the kernel source code in order to create a kernel that is optimized for the user application system.

- What is the Configurator?

The configurator creates an absolute format object file by linking together kernel objects, user relocatable objects defined by the user, and libraries.

This data that is essential to system construction is called “configuration data”, the data format that is used to store configuration data in files is called the “configuration definition statement”, and a file in which configuration definition statements are written is called a “configuration file”.

The configurator can read INCLUDE files. Just like configuration files, INCLUDE files contain configuration data that is necessary for system construction.
2.2 Positioning of the Configurator

Executing the configurator creates an absolute format object file.

- Positioning the Configurator

Executing the configurator creates an absolute format object file consisting of the kernel and a user program linked into one. (See Figure 2.2.)

Figure 2.2 Configurator Positioning
CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.3 Configurator Functions

The configurator functions can be broadly divided into two types:

- **Configuration data definition**
- **Configuration execution**

**Configurator Functions**

- **Configuration data definition**
  
  With the Softune Workbench version, it is easy to define the configuration data (from the Windows environment). The defined configuration data is saved as a text file, so it can be edited with a text editor, etc.

  For details, see Section 2.3.1 Defining Configuration Data.

- **Configuration execution**

  The Configurator generates the optimum kernel for the user system and various required data in accordance with the user-defined configuration data. It also links the kernel to a user application file. Therefore, an absolute object file generated by the Configurator can be incorporated into the system. For details see Section 2.3.2 Configuration Execution, Section 2.3.3 Absolute Object File Generation with Softune Workbench Version.
2.3 Configurator Functions

2.3.1 Defining Configuration Data

Configuration data can be easily defined by using the Softune Workbench version of the configurator.

- Defining Configuration Data

When the Softune Workbench version is used, it is easy to define configuration data (from the Windows environment). The defined configuration data is saved as a text file, so it can be edited with a text editor, etc., (Figure 2.3.1).

For the configuration data format, see Chapter 3 Configuration Definition Statements.

For the Softune Workbench version, see Part III, Softune Workbench Version.

![Figure 2.3.1 Defining Configuration Data](image)
2.3.2 Executing the Configuration Process

The configurator generates an absolute format object file on the basis of the configuration data.

- Executing the Configuration Process

The configurator creates an absolute format object file by linking the kernel object, the user relocatable object, and the user library that were all created on the basis of data described in the configuration file. (See Figure 2.3.2.)

The configurator starts up the assembler and the linker. Therefore, before starting up the configurator, it is necessary to define the directory where the tools are stored in the environment variable PATH.

Figure 2.3.2 Executing the Configuration Process

[Diagram showing the process of executing the configuration process with labels for User relocatable object, User library, Configuration file, Assembler, Linker, Kernel data file, Kernel source file, Kernel object file, Librarian, Embedded version, Evaluation version, and Absolute object file, Map file.]
2.3 Configurator Functions

The configurator has two operating modes. For details on the differences among the three modes and how to specify each of the modes, see Section 4.1, “Configurator Startup Command.”
CHAPTER 2  OVERVIEW OF THE CONFIGURATOR

2.3.3 Creating Absolute Format Object Files with the Softune Workbench Version

The configurator section is executed automatically when an absolute format object file is created by the Windows version of the configurator.

Creating Absolute Format Object Files with the Softune Workbench Version

The MS-DOS prompt version of the configurator (frcfs.exe) is automatically executed when an absolute format object file is generated by the Softune Workbench version of the configurator. Because the MS-DOS prompt version of the configurator starts the assembler, when an absolute format object file is generated, the directory where the various tools are stored must be defined in the PATH environment variable.

For details on the Softune Workbench version of the configurator, see Part III, “Softune Workbench Version of the Configurator.”

![Diagram of the process of creating absolute format object files with the Softune Workbench version.](image-url)

**Figure 2.3.3 Creating Absolute Format Object Files (Softune Workbench Version)**
2.4 Configuration Data (FR Family)

The configurator constructs a system based on the configuration data that was input from the configuration file. This section describes the rules, notes, restrictions, etc., that apply to the definition of configuration data.

Configuration Data

There are 13 types of configuration data definitions in the FR:

- CPU definition
- Linkage definition
- System definition
- Memory definition
- System call definition
- Task definition
- Common stack definition
- Semaphore definition
- Eventflag definition
- Mailbox definition
- Variable-size memorypool definition
- Fixed-size memorypool definition
- Vector definition

For details on each definition, see Sections 2.4.1, “CPU Definition,” to 2.4.13, “Vector Definition.”
CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.4.1 CPU Definition (FR)

The CPU definition specifies the target CPU using its MB number.

CPU definition

Specify the target CPU of the program that performs configuration using its MB number. An error occurs if a CPU that cannot be selected is specified.
2.4.2 Linkage Definition (FR)

The linkage definition defines the following four items:

- Relocatable object names
- Library names
- Absolute object name
- Map file output

## Linkage Definition

- **Relocatable object names**
  
  Specify the names of the user’s relocatable objects for the relocatable object names.
  The file names can be specified either with an absolute path specification or a relative path specification.

- **Library names**
  
  For the library name specification, specify the names of the libraries that the user program uses.
  The file names can be specified either with an absolute path specification or a relative path specification.

- **Absolute object name**
  
  Specify the name of the absolute object that was output when the configuration process was executed.

- **Map file output**
  
  Specify whether or not a map file is to be output. The map file is output to the current directory under the file name created by adding the extension ".mp1" to the absolute object name.
  
  For example, if the absolute object name is "realos.abs", the map file name will be "realos.mp1".
  
  The map file can be used to check the list of options that have been sent to the linker by the configurator and the list of sections that have been allocated.
2.4.3 System Definition (FR)

The system definition defines the following four items:

- Number of cyclic activation handlers/number of alarm handlers
- Exception handler definition
- Priority level
- INCLUDE file names

### System Definition

- **Number of cyclic activation handlers/number of alarm handlers**
  
  Specify the number of cyclic activation handlers and alarm handlers to be used by the user. Because the amount of memory used increases as this number increases, specify the required number. (The amount of memory used for the cyclic activation handlers and alarm handlers is listed below.)

  The range of values that can be specified is from 0 to 32767. If this specification is omitted, 0 is assumed.

<table>
<thead>
<tr>
<th>Object name</th>
<th>Amount of memory used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclic activation handler</td>
<td>28 x number of handlers</td>
</tr>
<tr>
<td>Alarm handler</td>
<td>28 x number of handlers</td>
</tr>
</tbody>
</table>

  (Unit: Bytes)

- **Exception handler definition**
  
  Specify the system call and CPU exception handler entry by means of a label or address. If this specification is omitted, the exception handler is not embedded.

- **Priority level**
  
  Specify the minimum priority for tasks used by the user system.

  The priority level that can be specified ranges from 8 to 32. If this specification is omitted, “32” is assumed to have been specified when the configuration process is executed.

  If the priority level decreases, the amount of memory used increases. Use the setting that is appropriate for the system.

- **INCLUDE file names**
  
  Specify the names of the configuration files that are to be included.

  Use INCLUDE file names when splitting the configuration file into two or more files.

  INCLUDE files cannot be nested.
2.4 Configuration Data (FR Family)

2.4.4 Memory Definition (FR)

The memory definition defines the following three items.

- **System stack**
- **Memory type and range**
- **Section names/addresses**

### Memory Definition

- **System stack**
  
  The system stack is used by the kernel and interrupt handlers.
  
  Specifiable sizes are multiples of four bytes. The minimum is 256 bytes.
  
  If this specification is omitted, the default is 1024 bytes.

- **Memory type and range**
  
  Set the ROM and RAM areas that can be allocated for the target. Based on this setting, the linker allocates the area automatically.
  
  Kernel code (including data such as initial values) is located contiguously at the address defined as the kernel data address.
  
  Data managed by the system (such as OS data and memory pools) is located contiguously at the address defined as the kernel data address.
  
  If the kernel code address and kernel data address are omitted, location is done automatically based on the specified ROM and RAM areas.
  
  Specify an address that is a multiple of four as the start address.

- **Section names/addresses**
  
  Specify all of the section names used by the user program and the addresses where those sections are to be located. If the address specification is omitted, each section is located after the previously specified section. Therefore, the address specification cannot be omitted for the first section that is specified.
  
  It is essential to note that the linker returns an error when a nonexistent section is defined.
  
  - For the kernel data addresses, see Appendix B.
The system call definition defines the system calls that are used by the user program.

**System Call Definition**

The system call definition defines the system calls that are used by the user program. Define all of the system calls that are to be used. Defining the system calls that are to be used makes it possible to optimize the kernel code size, since no unnecessary modules are embedded.

Table 2.4.5 lists the system calls that can be selected.

**Table 2.4.5 List of System Calls**

<table>
<thead>
<tr>
<th>Function name</th>
<th>System call name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK</td>
<td>sta_tsk ext_tsk ter_tsk dis_dsp ena_dsp chg_pri rot_rdq rel_wai get_tid ref_tsk</td>
</tr>
<tr>
<td>TSYNC</td>
<td>sus_tsk rsm_tsk frsm_tsk slp_tsk tslp_tsk wup_tsk can_wup</td>
</tr>
<tr>
<td>SEM</td>
<td>sig_sem wai_sem preq_sem ref_sem</td>
</tr>
<tr>
<td>FLG</td>
<td>set_flg clr_flg wai_flg pol_flg ref_flg</td>
</tr>
<tr>
<td>MBX</td>
<td>snd_msg rcv_msg prcv_msg ref_mbx</td>
</tr>
<tr>
<td>INT</td>
<td>ret_int loc_cpu unl_cpu chg_ilm ref_ilm</td>
</tr>
<tr>
<td>MPL</td>
<td>get_blk pget_blk rel_blk ref_mpl</td>
</tr>
<tr>
<td>MPF</td>
<td>get_blf pget_blf tget_blf rel_blf ref_mpf</td>
</tr>
<tr>
<td>TIM</td>
<td>set_tim get_tim dly_tsk def_cyc act_cyc ref_cyc def_alm ref_alm ret_trmr</td>
</tr>
<tr>
<td>SYS</td>
<td>get_ver ref_sys</td>
</tr>
</tbody>
</table>
The task definition defines the following eight items:

- Task name
- Task entry
- Task startup priority
- Task stack size
- Task attributes
- Task start code
- Task ID
- Extended information
- Object wait function with task time-out

**Task Definition**

- **Task name**
  
  An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here is as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.

  When writing code that references an ID number, using this object name is recommended.

- **Task entry**
  
  Specify the task entry with a label or an address.

- **Task startup priority**
  
  Specify the priority of the task when it is started up. Specify a priority that is higher than the minimum priority specified by the priority level. If this specification is omitted, the minimum priority is assumed.

  However: $8 \leq \text{Minimum priority}$
2.4 Configuration Data (FR Family)

- **Task stack size**
  
  Specify either the stack size used by the task, or a common stack name. Specify the size as a multiple of four bytes, with a minimum of 96 bytes. If this specification is omitted, a specification of “1024 bytes” is assumed.

  The stack size for a user task written with C language functions can be roughly determined by using the Softune C analyzer.

  However, it may be necessary to increase the specified amount (96 bytes) used for the task context.

  For details on common tasks, see Section 2.4.7, “Common Stack Definition.”

- **Task attributes**

  Specify the state at system startup of the task being defined. If this specification is omitted, “DORMANT” is assumed.

- **Task start code**

  Specify the task start code. This value is passed to the task when the task is started up. If this specification is omitted, “0” is assumed.

- **Task ID**

  Object IDs are used to specify individual objects.

  Note that ID numbers must be consecutive.

  The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

- **Extended information**

  32 bits of data that the user can use in any manner desired can be specified, either by a numeric value or a label. This information can be referenced by system call “ref_tsk”.

- **Object wait function with task time-out**

  Specify whether the object wait function with a task time-out is allowed. If this specification is omitted, the object wait function with a task time-out cannot be used.

  The object wait system call with a time-out function (twai_sem, twai_flg, trcv_msg, and tget_blf) can be used only by a task for which the use of the object wait function with a task time-out is allowed.

  This rule, however, does not apply to the tsip_tsk and dly_tsk system calls.
2.4.7 Common Stack Definition (FR)

The common stack definition defines a stack that can be shared by multiple tasks.

- **Common Stack Definition**
  
  By specifying a common stack, it becomes possible for multiple tasks to use the same stack. A common stack is created by specifying a stack name and size. A task can use a common stack by specifying the name of the common stack in place of the stack size.

  Specify the size as a multiple of four bytes, with a minimum of 96 bytes. If this specification is omitted, a specification of “1024 bytes” is assumed.

  **Note:**

  When multiple tasks that are sharing a stack (using the same common stack) are in a state other than the DORMANT state (READY, WAIT, etc.) simultaneously, the target system will not operate correctly.

  Tasks that operate simultaneously must either use different common stacks or else must not use common stacks.

[Example where correct operation is possible]

```
Task A
READY

Task B
DORMANT

Using the common stack

Common stack I
```

[Example where correct operation is not possible]

```
Task A
READY

Task B
WAIT

Using the common stack

Common stack I
```
CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.4.8 Semaphore Definition (FR)

The semaphore definition defines the following five items:

- Semaphore name
- Semaphore initial count
- Semaphore count maximum
- Semaphore ID
- Extended information

Semaphore Definition

- Semaphore name

An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.

When writing code that references an ID number, using this object name is recommended.

- Semaphore initial count

Specify the semaphore count at system startup as a value in the range from 0 to 32767. Specify a value for the initial count that is equal to or less than the semaphore count maximum. If this specification is omitted, “1” is assumed.

- Semaphore count maximum

Specify the maximum semaphore count that the system can use as a value in the range from 0 to 32767. An error results if a count value that is greater than the maximum specified here is used. If this specification is omitted, “32767” is assumed.

- Semaphore ID

Object IDs are used to specify individual objects.

Note that ID numbers must be consecutive.

The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.
2.4 Configuration Data (FR Family)

Extended information

32 bits of data that the user can use in any manner desired can be specified, either by a numeric value or a label. This information can be referenced by system call “ref_sem”.

<table>
<thead>
<tr>
<th>ID number</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
</tbody>
</table>

- ● ID is being used
- ○ ID is not being used

The example of incorrect usage would generate an error since ID number 3 is not being used.
2.4.9 EventFlag Definition (FR)

The eventflag definition defines the following four items:

- Eventflag name
- Eventflag initial pattern
- Eventflag ID
- Extended information

### EventFlag Definition

**Eventflag name**

An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.

When writing code that references an ID number, using this object name is recommended.

**Eventflag initial pattern**

Specify the eventflag bit pattern at system startup as a 32-bit value in the range from 0 to 0xFFFFFFFF. If this specification is omitted, “0” is assumed.

**Eventflag ID**

Object IDs are used to specify individual objects.

Note that ID numbers must be consecutive.

The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>- - - - - - - ○</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>- - - - - - - ○</td>
</tr>
</tbody>
</table>

- ● ID is being used
- ○ ID is not being used

The example of incorrect usage would generate an error since ID number 3 is not being used.

**Extended information**

32 bits of data that the user can use in any manner desired can be specified, either by a numeric value or a label. This information can be referenced by system call “ref_flg”.

---

30
2.4.10 Mailbox Definition (FR)

The mailbox definition defines the following three items:

- **Mailbox name**
  - An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.
  - When writing code that references an ID number, using this object name is recommended.

- **Mailbox ID**
  - Object IDs are used to specify individual objects.
  - Note that ID numbers must be consecutive.
  - The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

ID is being used
ID is not being used

The example of incorrect usage would generate an error since ID number 3 is not being used.

- **Extended information**
  - 32 bits of data that the user can use in any manner desired can be specified, either by a numeric value or a label. This information can be referenced by system call "ref_mbx".
CHAPTER 2  OVERVIEW OF THE CONFIGURATOR

2.4.11 Variable-size MemoryPool Definition (FR)

The Variable-size memorypool definition defines the following four items:

- Variable-size Memorypool name
- Variable-size Memorypool total size/number of blocks
- Variable-size Memorypool ID
- Extended information

**Variable-size MemoryPool Definition**

- **Variable-size Memorypool name**
  
  An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.

  When writing code that references an ID number, using this object name is recommended.

- **Variable-size Memorypool total size/number of blocks**
  
  Assuming "m" as the size of one variable-size memorypool block, "n" as the number of blocks, and "M" as the total size of the variable-size memorypool, then $n = M/m$. (See Figure. 2.4.11.) However, "m" must be a multiple of four bytes, with a minimum of 20 bytes.

  **Figure 2.4.11 Variable-size MemoryPool Total Size/Number of Blocks**

  In addition to the variable-size memorypool area, the memory block management area is required as an area used for variable-size memorypool management. The variable-size memorypool area and the memory block management area are automatically allocated from the area specified by the definition statement “KNL_RAM”.

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Area of m bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Area of $M = m \times n$ bytes</td>
</tr>
<tr>
<td>Block n</td>
<td></td>
</tr>
</tbody>
</table>

Example of variable-size memorypool with block size $m$ and number of blocks $n$
Variable-size Memorypool ID

Object IDs are used to specify individual objects.

Note that ID numbers must be consecutive.

The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

The example of incorrect usage would generate an error since ID number 3 is not being used.

Extended information

32 bits of data that the user can use in any manner desired can be specified, either by a numeric value or a label. This information can be referenced by system call “ref_mpl”.
2.4.12 Fixed-size MemoryPool Definition (FR)

The fixed-size memorypool definition defines the following four items:

- Fixed-size memorypool name
- Number of fixed-size memorypool blocks and block size
- Fixed-size memorypool ID
- Extended information

Fixed-size memorypool definition

- Fixed-size memorypool name
  
  An object ID number is assigned for an object name as a variable that can be referenced externally. By establishing the name defined here as an external reference variable and using it in a user program, it is possible to code programs without being aware of object ID numbers.

  When an ID number is referenced, the use of this object name is recommended in coding.

- Number of fixed-size memorypool blocks/block size

  Assuming m as the size of one block in the fixed-size memorypool, n as the number of blocks, and M as the total size of the fixed-size memorypool, then \( M = m \times n \). (See Figure 2.4.12.) However, m must be a multiple of four bytes. The minimum value is 20 bytes.

Overall fixed-size memorypool

- Area of m bytes
- Area of \( M = m \times n \) bytes

Example of fixed-size memorypool with block size m and number of blocks n

Figure 2.4.12 Variable-size MemoryPool Total Size/Number of Blocks

The fixed-size memorypool area is automatically allocated from the area specified by the KNL_RAM definition statement.
2.4 Configuration Data (FR Family)

- **Fixed-size memorypool ID**
  
  Object IDs are used to identify individual objects.

  Note that ID numbers must be consecutive.

  The range of numbers that can be specified for IDs is 1 to 32767. If this specification is omitted, an available ID is automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>32767</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example of correct usage:</td>
<td>●●●●●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●●●●●</td>
</tr>
<tr>
<td>Example of incorrect usage:</td>
<td>●●●●●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●●●●●</td>
</tr>
</tbody>
</table>

  ● ID is being used
  ○ ID is not being used

The example of incorrect usage generates an error because ID number 3 is not being used.

- **Extended information**

  Any 32-bit information that the user can use can be specified by a numeric value or a label. This information can be referenced by system call ref_mpf.
The vector definition defines the following three items:

- Vector number
- Entry
- Reset vector entry

**Vector Definition**

- Vector number
  Specify the vector number of the EIT vector being defined.
  The selectable vector numbers range from 16 to 62 and from 65 to 255.

- Entry
  Specify the label or address for the EIT vector entry.

- Reset vector entry
  Specify the entry name of the reset vector.
  If this specification is omitted, the reset vector name is “_sys_entry.”
2.5 Configuration Date (F²MC-16 Family)

The configurator configures systems based on configuration data input from configuration files. This section explains the rules for defining configuration data in the F²MC-16 family, along with precautionary information and restrictions.

- Configuration data

There are eleven types of configuration data definitions in the F²MC-16 family:

- CPU definition
- Linkage definition
- System definition
- Memory definition
- Task definition
- Common stack definition
- Semaphore definition
- Eventflag definition
- Mailbox definition
- Fixed-size memorypool definition
- Vector definition

For the details of each definition, see Sections 2.5.1, “CPU Definition,” to 2.5.11, “Vector Definition.”
The CPU definition specifies the target CPU using its MB number.

- **CPU definition**
  
  Specify the target CPU of the program that performs configuration using its MB number.
  
  An error occurs if a CPU that cannot be selected is specified.
2.5.2 Linkage Definition (F²MC-16)

Linkage definitions define the following four items.

- Relocatable object names
- Library names
- Absolute object names
- Map file output designations

**Linkage Definition**

- Relocatable object names
  Relocatable object names designate user-relocatable objects. When designating file names, either relative paths or absolute paths may be used.
  Programs that reserve areas for tasks, memorypools, semaphores, eventflags or vector tables are automatically generated and embedded into execute files, and therefore need not be created.

- Library names
  Library names designate libraries for use in user programs. When designating file names, either relative paths or absolute paths may be used.

- Absolute object names
  Absolute object names designate absolute objects output from Configurator. When designating file names, either relative paths or absolute paths may be used.

- Map file output designations
  Map file output designations refer to the output of map files, which are files in which the absolute object name extension has been changed to “.mp1”
  For example, the map file name corresponding to the absolute object name "realos.abs" would be "realos.mp1."
  The map file can be used to check the list of options that have been sent to the linker by the configurator and the list of sections that have been allocated.
2.5.3 System Definitions (F²MC-16)

F²MC-16 system definitions include definitions of the following 12 items:
- Number of cyclic activation handlers
- Priority level
- System call exception handling definition
- CPU exception handler entry definition
- Include file name
- Number of register banks
- F²MC-16F dedicated code selection
- wai_tsk embedding
- System clock handler return method
- Context register

System Definitions

- Number of cyclic activation handlers and number of alarm handlers
  Specify the number of cyclic activation handlers and alarm handlers to be used by the user. Because the amount of memory used increases as this number increases, specify only the required number. (The amount of memory used for the cyclic activation handlers and alarm handlers is listed below.)

  The allowable range is 0 to 127. If this specification is omitted, the default is 0. Because these definition statements involve the use of timers, extra memory is required for timer operation.

    | Object name               | Required memory                     |
    |---------------------------|-------------------------------------|
    | Timer                     | 4                                   |
    | Cyclic activation handler | (12 + 1 bit) x number of handlers   |
    | Alarm handler             | 2 + 14 x number of handlers         |

- Priority level
  Specify the lowest priority level assigned to a task to be used in the user system.

  The allowable range is 1 to 16. If this specification is omitted, the default is 16.

  Since a smaller priority level number requires more memory, use a priority level that is appropriate for the system.

- CPU exception handler entry definition
  Specify the entry of the CPU exception handler.

  If this specification is omitted, an CPU exception handler definition is not embedded.

- System call exception handling definition
  Specify the entry of the system call exception handler.

  If this specification is omitted, the system call exception handler definition is not embedded.

  You can also specify whether to trigger a system call exception for every error.

- Include file name
  Specify the name of the configuration file to be included.

  Include file nesting is not allowed.
2.5 Configuration Date (F²MC-16 Family)

- **Kernel operation interrupt level**
  Specify the kernel operation interrupt enable level.
  Specify an interrupt level that is lower than that used in applications.
  The allowable range is 0 to 6. Normally, 6 is used. If this specification is omitted, the default is 6.

- **Number of register banks**
  Reserve a specified number of register banks, starting with bank 0.
  The allowable range is 0 to 31. If this specification is omitted, the default is 31.
  This setting specifies how much of the RAM area for register banks is actually used for register banks. Automatic allocation is not performed by the linker for the area reserved by this setting.
  16 x (1 + specified number) bytes of memory is required.

- **F²MC-16F dedicated code selection**
  Specify output of F²MC-16F dedicated code when an F²MC-16F family CPU is used.

- **wai_tsk embedding**
  Specify whether to embed the wai_tsk system call function.
  When wai_tsk is to be embedded, the system clock handler return method must be defined.

- **System clock handler return method**
  Specify the interface for the user-specified interrupt handler for the system clock. If this specification is omitted, a system clock interrupt handler is not embedded.

- **Context register**
  Specify whether to perform context switching by register bank switching or by saving the context to and restoring it from the stack. For context switching by saving the context to and restoring it from the stack, specify the memory register handled as the context. (Select all memory registers if you select context switching by saving the context to and restoring it from the stack using the C compiler.)
  When switching from one task to another, the kernel saves the PS, PC, PCB, DTB, ADB, DPR, A, and T registers. Note that the memory register to be saved is only the one specified here.
  When context switching by register bank switching is selected, the kernel does not save any memory registers. However, register bank switching has the same effect as saving all memory registers (RW0 to RW7). Note that a single register bank cannot be shared by multiple tasks (excluding stack-sharing tasks).
Memory Definitions include definitions of the following three items:

- System stack
- Memory type and range
- Section name/address

Memory Definitions

- System stack
  The system stack is used by the kernel and interrupt handlers.
  As the size, specify an even number from 0x20 to 0xfffe.
  If this specification is omitted, the default size is 256 bytes.

- Memory type and range
  Specify the ROM and RAM areas that can be allocated by the target. Automatic allocation is performed by the linker according to this setting.

  Kernel code (including data such as initial values) is placed at consecutive addresses that have been defined as kernel code addresses.

  Data managed by the system (such as operating system data and memory pools) is placed at consecutive addresses that have been defined as kernel data addresses.

  If kernel code addresses and kernel data addresses are not defined, code and data are automatically placed according to the specification of the ROM and RAM areas.

  Use an even-numbered address as the start address.

- Section name/address
  Specify the names of all sections used in user programs, as well as the addresses where those sections are to be placed. If addresses are omitted, sections are placed after the previously specified sections. When you specify sections for the first time, however, their addresses must also be specified.

  Note that definition of nonexistent sections will result in a linker error.

Reference: For an explanation of kernel data addresses, see Appendix B, “Allocating Kernel Code and Kernel Data.”
2.5.5 Task Definitions (F²MC-16)

Task definitions include definitions of the following 10 items:

- Task name
- Task ID
- Task entry
- Task start priority level
- Task stack size
- RP register
- Data bank register
- Additional data bank register
- Direct page register
- Condition code register

### Task Definitions

- **Task name**
  
  Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.

  Fujitsu recommends using these object names when referencing ID numbers.

- **Task ID**
  
  Object IDs are used to identify individual objects.

  Note that ID numbers must be assigned sequentially.

  Assign IDs in the range from 1 to 255. If this specification is omitted, unused IDs are automatically assigned.

  ![ID number usage example](image)

  - **Proper use**
    - ID number used
    - ID number not used
  - **Improper use**
    - ID number used
    - ID number not used

  In the incorrect use example, an error will result because ID number 3 is not used.

- **Task entry**
  
  Specify the entry of a task using a label or address.
2.5 Configuration Date (F^MC-16 Family)

- **Task start priority level**
  Specify the priority level for starting a task. Specify a priority level that is higher than the lowest priority level. If this specification is omitted, the lowest priority level is assigned.

```
<table>
<thead>
<tr>
<th>Task priority</th>
<th>Start priority level</th>
<th>Lowest priority level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>16</td>
</tr>
</tbody>
</table>
```

- **Task stack size**
  Specify the size of the stack area used for a target task. If this specification is omitted, the default is a size of 0x100 bytes.

  If labels are defined for stack sharing, any task name or entry name that duplicates a label will result in an error.

  If C language functions are used to describe a user stack, the approximate stack size can be calculated using Softune C Analyzer.

  Note that the specified stack size must include the task context size.

  For an explanation of common stacks, see Section 2.5.6, “Common Stack Definitions.”

- **RP register**
  Specify the RP register used for a task. The allowable range is 0 to 31. However, do not specify bank 1 since it is used by the kernel. If this specification is omitted, register bank 0 is used for the task.

  If register bank switching is specified for the context register setting in the system definition, set different values for individual tasks (excluding stack-sharing tasks).

- **Data bank register**
  Specify data bank registers. The allowable range is 0x00 to 0xff. The default is 0.

- **Additional data bank register**
  Specify additional data bank registers. The allowable range is 0x00 to 0xff. The default is 0.

- **Direct page register**
  Specify direct page registers. The allowable range is 0x00 to 0xff. The default is 0x00.

- **Condition code register**
  Specify condition code registers. The allowable range is 0x00 to 0xff. The default is 0x40. (This default value is used unless there is a special case.)


CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.5.6 Common Stack Definitions (F²MC-16)

Common stack definitions define stacks shared by multiple tasks.

- **Common Stack Definitions**

  Specifying a common stack allows multiple tasks to use the same stack.

  To create a common stack, specify the name and size of the stack. To allow a task to use a common stack, specify the common stack name instead of a task stack size.

  Specify a size of 0x0E to 0xFFFE bytes. If this specification is omitted, the default is 256 bytes.

  If multiple tasks using the same common stack enter any mode other than DORMANT mode (such as READY or WAIT), the target system may not operate properly.

  Tasks that operate at the same time must use different common stacks, or must avoid using any common stacks.

---

[Sample of proper operation] [Sample of improper operation]
2.5.7 Semaphore Definitions (F²MC-16)

Semaphore definitions include definitions of the following four items:

- Semaphore initial count (general)
- Semaphore initial count (individual)
- Semaphore name
- Semaphore ID

**Semaphore Definitions**

- Semaphore initial count (general)
  Define the fixed value of the semaphore initial count for all semaphore IDs.
  The allowable range for the initial count value is 0 to 255.

- Semaphore initial count (individual)
  Specify a semaphore count at system startup in the range from 0 to 255. If this specification is omitted, the default is 1.
  This setting is ignored when a fixed value has been defined for the semaphore initial count.

- Semaphore name
  Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.
  Fujitsu recommends using these object names when referencing ID numbers.

- Semaphore ID
  Object IDs are used to identify individual objects.
  Note that ID numbers must be assigned sequentially.
  Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Improper use</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

- ●: ID number used
- ○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
2.5.8 Eventflag Definitions (F²MC-16)

Eventflag definitions include definitions of the following five items:

- Eventflag control method
- Eventflag initial pattern (fixed value)
- Eventflag initial pattern
- Eventflag name
- Eventflag ID

**Eventflag Definitions**

- **Eventflag control method**
  
  Specify the eventflag control method. Either 1-bit or 16-bit control can be selected.

- **Eventflag initial pattern (fixed value)**
  
  Define the fixed value of the eventflag initial pattern in the bit width selected for the eventflag control method. Before this setting can be defined, the eventflag control method must have already been defined.

  When selecting 1-bit eventflag control, specify 0 or 1. When selecting 16-bit eventflag control, specify a value in the range from 0 to 0xffff.

- **Eventflag initial pattern**
  
  Define an eventflag initial pattern at system startup in the bit width selected for the eventflag control method.

  When selecting 1-bit eventflag control, specify 0 or 1. When selecting 16-bit eventflag control, specify a value in the range from 0 to 0xffff. The default is 0.

  Specifying an initial pattern in any bit width other than that of the selected eventflag control method will result in an error. This setting is ignored when a fixed value has been defined for the eventflag initial pattern.

- **Eventflag name**
  
  Assign each object name an object ID number as a variable to which an external reference can be made. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.

  For references to ID numbers in coding, using object names is recommended.

- **Eventflag ID**
  
  Object IDs are used to identify individual objects.

  Note that ID numbers must be assigned sequentially.

  Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.
2.5 Configuration Date (F²MC-16 Family)

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Improper use</td>
<td>●</td>
<td>●</td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

●: ID number used
○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
CHAPTER 2  OVERVIEW OF THE CONFIGURATOR

2.5.9 Mailbox Definitions (F²MC-16)

Mailbox definitions include definitions of the following two items:

- Mailbox name
- Mailbox ID

Mailbox Definitions

- Mailbox name
  Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.
  Fujitsu recommends using these object names when referencing ID numbers.

- Mailbox ID
  Object IDs are used to identify individual objects.
  Note that ID numbers must be assigned sequentially.
  Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Improper use</td>
<td>●</td>
<td>●</td>
<td>O</td>
<td>●</td>
<td>O</td>
<td>-</td>
</tr>
</tbody>
</table>

- ●: ID number used
- O: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.


CHAPTER 2  OVERVIEW OF THE CONFIGURATOR

2.5.10  Fixed-size MemoryPool Definitions (F²MC-16)

Fixed-size memorypool definitions include definitions of the following three items:

- Fixed-size memorypool name
- Fixed-size memorypool ID
- Fixed-size memorypool size/block size

Fixed-size memorypool Definitions

- Fixed-size memorypool name
  Assign each object name an object ID number as a variable to which an external reference can be made. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.
  For references to ID numbers in coding, using object names is recommended.

- Fixed-size memorypool ID
  Object IDs are used to identify individual objects.
  Note that ID numbers must be assigned sequentially.
  Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Improper use</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

●: ID number used  
○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
Fixed-size memorypool size/block size

If the size of each block in a Fixed-size memorypool is \( m \), the block count is \( n \), and the total memorypool size for each ID is \( M \), then \( M = m \times n \) (see Figure 2.5.10).

Define the total size \( M \) of a Fixed-size memorypool for each ID that does not exceed 64K bytes.

**Figure 2.5.10  Total size of Fixed-size memorypool/block size**

In addition to the Fixed-size memorypool area, a Fixed-size memory block management area is required. The Fixed-size memorypool area and fixed size memory block management area are automatically allocated to the areas specified by the KNL_RAM definition statement.
2.5.11 Vector Definitions (F²MC-16)

Vector definitions include definitions of the following three items:

- Initialization handler entry
- Interrupt handler entry
- Reset vector entry

### Vector Definitions

- **Initialization handler entry**
  Define the entry of the initialization handler called when the operating system starts.
  The default name for the initialization handler is `_inihdr`.

- **Interrupt handler entry**
  Specify a label name or address as the entry name of the interrupt handler. Specify the interrupt number as an IRQn number.
  Specify a value in the range from 11 to 41 as the interrupt number.

- **Reset vector entry**
  Specify the entry name of the reset vector.
  The default name of the reset vector is `_sys_entry`. 
2.6 Configuration Data (F²MC-8L Family)

The configurator configures systems based on configuration data, which is input from configuration files. This section describes the rules for defining configuration data for the F²MC-8L family as well as provides precautionary information and limitations.

Definitions of Configuration Data

There are ten types of configuration data definitions.

- CPU definition
- Linkage definition
- System definition
- Memory definition
- Task definition
- Common stack definition
- Semaphore definition
- Eventflag definition
- Mailbox definition
- Vector definition

Section 2.6.1, “CPU Definition,” to Section 2.6.10, “Vector Definition,” describe each definition.
2.6.1 CPU Definition (F²MC-8L)

The CPU definition specifies the target CPU using its MB number.

- CPU Definition

  Specify the target CPU of the program that performs configuration using an MB number. Specifying a CPU that cannot be selected causes an error.
2.6.2 Linkage Definition (F²MC-8L)

The linkage definitions include definitions of the following four items:

- Relocatable object names
- Library names
- Absolute object names
- Map file output

### Linkage Definitions

- **Relocatable object names**
  Specify as a relocatable object name the name of a user-relocatable object. Either the absolute path or a relative path can be used to specify a file name.

- **Library names**
  Specify as a library name the name of a library to be used by the user program. Either the absolute path or a relative path can be used to specify a file name.

- **Absolute object names**
  Specify the names of absolute objects to be output when configuration is performed.

- **Map file output**
  Specify whether to output a map file. The map file, which is the absolute object name with its extension changed to .mp1, is output to the current directory.
  
  For example, the map file name corresponding to the absolute object name realos.abs would be realos.mp1.
  
  The map file can be used to check the list of options that have been sent to the linker by the configurator and the list of sections that have been allocated.
2.6.3 System Definitions (F²MC-8L)

F²MC-8L system definitions include definitions of the following seven items:

- Number of cyclic activation handlers
- Priority level
- Method for returning from system clock handler
- wai_tsk embedding
- Exception processing embedding
- Include file
- Number of register banks

## System Definitions

### Number of cyclic activation handlers

Specify the number of cyclic activation handlers to be used by the user. Because the amount of memory used increases as this number increases, specify only the required number. The amount of memory used for the cyclic activation handlers is listed below.

Because these definition statements involves the use of timers, also take into consideration the extra memory required for the timers.

The allowable range is 0 to 127. If this specification is omitted, the default is 0.

<table>
<thead>
<tr>
<th>Object name</th>
<th>Amount of memory used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer</td>
<td>4</td>
</tr>
<tr>
<td>Cyclic activation handler</td>
<td>$(8 \text{ bytes} + 1 \text{ bit}) \times \text{number of handlers}$ (unit: bytes)</td>
</tr>
</tbody>
</table>

Seven and fewer bits are rounded up.

### Exception processing embedding

Specify whether to embed exception processing in the kernel. If a name for exception processing is specified, the exception handler is called when a system call execution error occurs. Otherwise, an exception handler is not called.

If an exception handler is embedded, the following error handing routines are installed:

- Reserved function error
- Parameter error
- Illegal object definition
- Context error
- Run-time error
- Forcible wait state cancellation
- Results of processing

### Priority level

Specify the lowest priority level to be used by the user system.

The allowable range is 1 to 4. If this specification is omitted, the default is 4.

Since a smaller priority level number requires more memory, use a priority level that is appropriate for the system.
2.6 Configuration Data (F²MC-8L Family)

- Include file
  Specify the names of the configuration files that are to be included.
  Include files cannot be nested.

- System clock handler return method
  Specify the interface for the system clock interrupt handler specified by the user. If this definition is omitted, interrupt handling is terminated by the ret instruction.

- Number of register banks
  Reserve a specified number of register banks, starting with bank 0.
  The allowable range is 0 to 31. If this specification is omitted, the default is 31.
  Since this definition specifies the size of the RAM area to be used for register banks, automatic allocation is not performed by the linker for the area reserved by this setting.
  The amount of memory required is $8 \times (1 + \text{specified number})$ (unit: bytes).

- wai_tsk embedding
  Specify whether to embed the wai_tsk system call function.
  If embedding of wai_tsk is specified, also define the system clock handler return method.
2.6.4 Memory Definitions (F²MC-8L)

Memory definitions include definitions of the following three items:

- System stack
- Memory type and area
- Section name/address

### Memory Definitions

- **System stack**
  
The system stack is used by the kernel and interrupt handlers.
  
The allowable range is 0x20 to 0x2000.
  
If this specification is omitted, the default is 40 bytes.

- **Memory type and area**
  
Specify ROM and RAM areas that can be allocated by the target. Automatic allocation is performed by the linker according to this setting.

  Kernel code (including data such as initial values) is placed at consecutive addresses that have been defined as kernel code addresses.

  Data managed by the system, such as operating system data and memory pools, is placed at consecutive addresses that have been defined as kernel data addresses.

  If kernel code addresses or kernel data addresses are not defined, code and data are automatically placed according to the specification of the ROM and RAM areas.

  Specify an even-numbered address as the start address.

- **Section name/address**
  
Specify as section names the names of all sections used by the user program. Specify as addresses the addresses at which those sections are to be placed. If the address corresponding to a section is not specified, the section is placed immediately after the section specified last. Accordingly, the address for the first specified section must be specified.

  Note that definition of nonexistent sections will result in a linker error.

**Reference:** For an explanation of kernel data addresses, see Appendix B, “Allocating Kernel Code and Kernel Data.”
Task definitions include definitions of the following six items:

- Task name
- Task ID
- Task entry
- Task start priority level
- Task stack size
- RP registers

### Task Definitions

- **Task name**
  
  Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.

  Fujitsu recommends using these object names when referencing ID numbers.

- **Task ID**
  
  Object IDs are used to identify individual objects.

  Note that ID numbers must be assigned sequentially.

  Assign IDs in the range from 1 to 255. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Incorrect use</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

- **ID number**
  - ●: ID number used
  - ○: ID number not used

  In the incorrect use example, an error will result because ID number 3 is not used.

- **Task entry**
  
  Specify the entry of a task using a label or address.
2.6 Configuration Data (F²MC-8L Family)

- **Task start priority level**
  Specify the priority level for starting a task. Specify a priority level that is higher than the lowest priority level in the system definition. If this specification is omitted, the lowest priority level is assigned.

- **Task stack size**
  Specify the size of the stack or the name of the common stack used by a task. The allowable range is 0x000E to 0x2000. If this specification is omitted, the default is 40 bytes.

  If C language functions are used to describe a user stack, the approximate stack size can be calculated using Softune C Analyzer.

  If this is done, take into consideration the stack to be used as the task context.

  For an explanation of common stacks, see Section 2.6.6, “Common Stack Definition.”

- **RP register**
  Specify the registers to be used by a task. The allowable range is 1 to 31. However, do not specify bank 0 since it is used by the kernel. If this specification is omitted, the default is 1.
CHAPTER 2 OVERVIEW OF THE CONFIGURATOR

2.6.6 Common Stack Definitions (F²MC-8L)

Common stack definitions define stacks used shared by multiple tasks.

- Common Stack Definitions

  Specifying a common stack allows multiple tasks to use the same stack.

  To create a common stack, specify the name and size of the stack. To allow a task to use a common stack, specify the common stack name instead of a task stack size.

  Specify a value between 0x0E and 0x2000 bytes for the common stack size. If this specification is omitted, the default is 40 bytes.

  If multiple tasks using the same common stack enter any mode other than DORMANT mode (such as READY or WAIT mode), the target system may not operate properly.

  Tasks that operate at the same time must use different common stacks, or must avoid using any common stacks.

[Example of proper operation]  [Example of improper operation]
2.6.7 Semaphore Definitions (F²MC-8L)

The Semaphore definitions include definitions of the following four items:

- Semaphore initial count (general)
- Semaphore initial count (individual)
- Semaphore name
- Semaphore ID

Semaphore Definitions

- Semaphore initial count (general)
  Specify the semaphore initial count value only when the same fixed value is specified for all semaphore IDs.
  The allowable range is 0 to 255.

- Semaphore initial count (individual)
  Specify a value between 0 and 255 for the semaphore count value at system startup. If this specification is omitted, the default is 1.
  This setting is ignored when a semaphore initial count value has been specified.

- Semaphore name
  Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.
  Fujitsu recommends using these object names when referencing ID numbers.

- Semaphore ID
  Object IDs are used to identify individual objects.
  Note that ID numbers must be assigned sequentially.
  Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Incorrect use</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

- ●: ID number used
- ○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
2.6.8 Eventflag Definitions (F²MC-8L)

Eventflag definitions include definitions of the following two items:

- Eventflag name
- Eventflag ID

Eventflag Definitions

- Eventflag name

Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.

Fujitsu recommends using these object names when referencing ID numbers.

- Eventflag ID

Object IDs are used to identify individual objects.

Note that ID numbers must be assigned sequentially.

Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Incorrect use</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

●: ID number used
○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
2.6.9 Mailbox Definitions (F²MC-8L)

The mailbox definitions include definitions of the following two items:

- Mailbox name
- Mailbox ID

- Mailbox Definitions

  - Mailbox name
    
    Assign each object name an object ID number so that it can be handled as an external reference variable. In user programs, the use of defined names as external reference variables enables programs to be coded without having to consider object ID numbers.
    
    Fujitsu recommends using these object names when referencing ID numbers.

  - Mailbox ID
    
    Object IDs are used to identify individual objects.
    
    Note that ID numbers must be assigned sequentially.
    
    Assign IDs in the range from 1 to 127. If this specification is omitted, unused IDs are automatically assigned.

<table>
<thead>
<tr>
<th>ID number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct use</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Incorrect use</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

●: ID number used
○: ID number not used

In the incorrect use example, an error will result because ID number 3 is not used.
2.6.10 Vector Definitions (F²MC-8L)

Vector definitions include definitions of the following four items:

- Initialization handler entry
- Call vector entry
- Interrupt handler entry
- Reset vector entry

**Initialization handler entry**
Define the entry name of the initialization handler called when the operating system starts.
The default name for the initialization handler is _inihdr.

**Call vector entry**
Specify the entry name and vector number of the call vector.
Specify a value in the range from 2 to 7.

**Interrupt handler entry**
Specify a label name or address as the entry name of the interrupt handler. Specify the interrupt number as an IRQn number.
Specify a value in the range from 0 to 11 as the interrupt number.

**Reset vector entry**
Specify the entry name of the reset vector.
The default name of the reset vector is _sys_entry.
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

This chapter describes the structure of and lists the configuration definition statements and explains them in detail.

3.1 Overview of the Configuration Definition Statements
3.2 Definition Statement Explanation
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.1 Overview of the Configuration Definition Statements

Configuration definition statements are the data formats used when the configurator stores configuration data in configuration files.

### Overview of the Configuration Definition Statements

Configuration definition statements consist of the configuration definition name and parameters. Tables 3.1a, 3.1b and 3.1c list the configuration definition names.

#### Table 3.1a List of Configuration Definition Names [FR]

<table>
<thead>
<tr>
<th>Configuration data name</th>
<th>Definition name</th>
<th>Description of definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU definition</td>
<td>CPU</td>
<td>CPU type</td>
</tr>
<tr>
<td>Linkage definition</td>
<td>RELOCATE_FILE</td>
<td>Relocatable object name</td>
</tr>
<tr>
<td></td>
<td>LIBRARY_FILE</td>
<td>Library file name</td>
</tr>
<tr>
<td></td>
<td>ABSOLUTE_FILE</td>
<td>Absolute object name</td>
</tr>
<tr>
<td></td>
<td>MAP_FILE</td>
<td>Map file output specification</td>
</tr>
<tr>
<td>System definition</td>
<td>CYC_HDR_NUMBER</td>
<td>Number of cyclic activation handlers</td>
</tr>
<tr>
<td></td>
<td>ALM_HDR_NUMBER</td>
<td>Number of alarm handlers</td>
</tr>
<tr>
<td></td>
<td>EXC_HDR_ENTRY</td>
<td>Exception handler entry</td>
</tr>
<tr>
<td></td>
<td>PRIORITY_LEVEL</td>
<td>Priority level</td>
</tr>
<tr>
<td></td>
<td>INCLUDE</td>
<td>INCLUDE file name</td>
</tr>
<tr>
<td>Memory definition</td>
<td>SYS_STK_SIZE</td>
<td>System stack size</td>
</tr>
<tr>
<td></td>
<td>ROM</td>
<td>Code address range area name</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>Data address range area name</td>
</tr>
<tr>
<td></td>
<td>KNL_ROM</td>
<td>Kernel code allocation address</td>
</tr>
<tr>
<td></td>
<td>KNL_RAM</td>
<td>Kernel data allocation address</td>
</tr>
<tr>
<td></td>
<td>USR_SECTION</td>
<td>Section name and section location</td>
</tr>
<tr>
<td>System call definition</td>
<td>SYSTEM_CALL</td>
<td>Embedded system call</td>
</tr>
<tr>
<td>Task definition</td>
<td>TSK</td>
<td>Task definition</td>
</tr>
<tr>
<td></td>
<td>COM_STACK</td>
<td>Task common stack definition</td>
</tr>
<tr>
<td>Semaphore definition</td>
<td>SEM</td>
<td>Semaphore definition</td>
</tr>
<tr>
<td>Eventflag definition</td>
<td>FLG</td>
<td>Eventflag definition</td>
</tr>
<tr>
<td>Mailbox definition</td>
<td>MBX</td>
<td>Mailbox definition</td>
</tr>
<tr>
<td>Memorypool definition</td>
<td>MPL</td>
<td>Variable-size memorypool definition</td>
</tr>
<tr>
<td></td>
<td>MPF</td>
<td>Fixed-size memorypool definition</td>
</tr>
<tr>
<td>Vector definition</td>
<td>EIT_ENTRY</td>
<td>Vector definition</td>
</tr>
<tr>
<td></td>
<td>MPF</td>
<td>Fixed-size memorypool definition</td>
</tr>
</tbody>
</table>
### 3.1 Overview of the Configuration Definition Statements

#### Table 3.6b List of Configuration Definition Names [16]

<table>
<thead>
<tr>
<th>Configuration data name</th>
<th>Definition name</th>
<th>Description of definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector definition</td>
<td>EIT_ENTRY</td>
<td>Vector definition</td>
</tr>
<tr>
<td></td>
<td>RESET_ENTRY</td>
<td>Reset vector entry</td>
</tr>
<tr>
<td>CPU definition</td>
<td>CPU</td>
<td>CPU type</td>
</tr>
<tr>
<td>Linkage definition</td>
<td>RELOCATE_FILE</td>
<td>Relocatable object name</td>
</tr>
<tr>
<td></td>
<td>LIBRARY_FILE</td>
<td>Library file name</td>
</tr>
<tr>
<td></td>
<td>ABSOLUTE_FILE</td>
<td>Absolute object name</td>
</tr>
<tr>
<td></td>
<td>MAP_FILE</td>
<td>Map file output specification</td>
</tr>
<tr>
<td>System definition</td>
<td>CYC_HDR_NUMBER</td>
<td>Number of cyclic activation handlers</td>
</tr>
<tr>
<td></td>
<td>ALM_HDR_NUMBER</td>
<td>Number of alarm handlers</td>
</tr>
<tr>
<td></td>
<td>CPU_EXTHDR_ENTRY</td>
<td>CPU exception handler entry</td>
</tr>
<tr>
<td></td>
<td>SYS_EXTHDR_ENTRY</td>
<td>System call exception handler entry</td>
</tr>
<tr>
<td></td>
<td>PRIORITY_LEVEL</td>
<td>Priority level</td>
</tr>
<tr>
<td></td>
<td>INCLUDE</td>
<td>INCLUDE file</td>
</tr>
<tr>
<td></td>
<td>ILM</td>
<td>Interrupt level at kernel operation</td>
</tr>
<tr>
<td></td>
<td>CODE</td>
<td>F*MC-16F code output selection</td>
</tr>
<tr>
<td></td>
<td>SYS_CLOCK</td>
<td>System clock handler return method</td>
</tr>
<tr>
<td></td>
<td>CONTEXT_REGISTER</td>
<td>Context register</td>
</tr>
<tr>
<td></td>
<td>REGISTER_BANK</td>
<td>Number of register banks</td>
</tr>
<tr>
<td></td>
<td>wai_tsk</td>
<td>wai_tsk installation</td>
</tr>
<tr>
<td>Memory definition</td>
<td>SYS_STK_SIZE</td>
<td>System stack size</td>
</tr>
<tr>
<td></td>
<td>ROM</td>
<td>Code address range</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>Data address range</td>
</tr>
<tr>
<td></td>
<td>KNL_ROM</td>
<td>Kernel code location address</td>
</tr>
<tr>
<td></td>
<td>KNL_RAM</td>
<td>Kernel data location address</td>
</tr>
<tr>
<td></td>
<td>USR_SECTION</td>
<td>Section name and section location address</td>
</tr>
<tr>
<td>Task definition</td>
<td>TSK</td>
<td>Task definition</td>
</tr>
<tr>
<td></td>
<td>COM_STACK</td>
<td>Task common stack specification</td>
</tr>
<tr>
<td>Semaphore definition</td>
<td>SEM_ICNT</td>
<td>Semaphore initial count value definition</td>
</tr>
<tr>
<td></td>
<td>SEM</td>
<td>Semaphore definition</td>
</tr>
</tbody>
</table>
### Table 3.6b List of Configuration Definition Names [16] (Continued)

<table>
<thead>
<tr>
<th>Configuration data name</th>
<th>Definition name</th>
<th>Description of definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventflag definition</td>
<td>FLG_BIT</td>
<td>Selection of eventflag management method</td>
</tr>
<tr>
<td></td>
<td>FLG_PATTERN</td>
<td>Setting of eventflag initial pattern value</td>
</tr>
<tr>
<td></td>
<td>FLG</td>
<td>Eventflag definition</td>
</tr>
<tr>
<td>Mailbox definition</td>
<td>MBX</td>
<td>Mailbox definition</td>
</tr>
<tr>
<td>Memorypool definition</td>
<td>MPL</td>
<td>Memorypool definition</td>
</tr>
<tr>
<td>Vector definition</td>
<td>EIT_ENTRY</td>
<td>Interrupt handler entry</td>
</tr>
<tr>
<td></td>
<td>INIT_HDR_ENTRY</td>
<td>Initialization handler entry</td>
</tr>
<tr>
<td></td>
<td>RESET_ENTRY</td>
<td>Reset vector entry</td>
</tr>
</tbody>
</table>
### Table 3.1c List of Configuration Definition Names [8L]

<table>
<thead>
<tr>
<th>Configuration data name</th>
<th>Definition name</th>
<th>Description of definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU definition</td>
<td>CPU</td>
<td>CPU type</td>
</tr>
<tr>
<td>Linkage definition</td>
<td>RELOCATE_FILE</td>
<td>Relocatable object name</td>
</tr>
<tr>
<td></td>
<td>LIBRARY_FILE</td>
<td>Library file name</td>
</tr>
<tr>
<td></td>
<td>ABSOLUTE_FILE</td>
<td>Absolute object name</td>
</tr>
<tr>
<td></td>
<td>MAP_FILE</td>
<td>Map file output specification</td>
</tr>
<tr>
<td>System definition</td>
<td>CYC_HDR_NUMBER</td>
<td>Number of cyclic activation handlers</td>
</tr>
<tr>
<td></td>
<td>SCAL_EXC_ENTRY</td>
<td>System call optimization/exception handler name</td>
</tr>
<tr>
<td></td>
<td>PRIORITY_LEVEL</td>
<td>Priority level</td>
</tr>
<tr>
<td></td>
<td>INCLUDE</td>
<td>INCLUDE file</td>
</tr>
<tr>
<td></td>
<td>SYS_CLOCK</td>
<td>System clock handler return method</td>
</tr>
<tr>
<td></td>
<td>REGISTER_BANK</td>
<td>Number of register banks</td>
</tr>
<tr>
<td></td>
<td>wai_tsk</td>
<td>wai_tsk installation</td>
</tr>
<tr>
<td>Memory definition</td>
<td>SYS_STK_SIZE</td>
<td>System stack size</td>
</tr>
<tr>
<td></td>
<td>ROM</td>
<td>Code address size area name</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>Data address size area name</td>
</tr>
<tr>
<td></td>
<td>KNL_ROM</td>
<td>Kernel code allocation address</td>
</tr>
<tr>
<td></td>
<td>KNL_RAM</td>
<td>Kernel data allocation address</td>
</tr>
<tr>
<td></td>
<td>USR_SECTION</td>
<td>Section name and section location address</td>
</tr>
<tr>
<td>Task definition</td>
<td>TSK</td>
<td>Task definition</td>
</tr>
<tr>
<td></td>
<td>COM_STACK</td>
<td>Task common stack specification</td>
</tr>
<tr>
<td>Semaphore definition</td>
<td>SEM_ICNT</td>
<td>Semaphore initial count value definition</td>
</tr>
<tr>
<td></td>
<td>SEM</td>
<td>Semaphore definition</td>
</tr>
<tr>
<td>Eventflag definition</td>
<td>FLG</td>
<td>Eventflag definition</td>
</tr>
<tr>
<td>Mailbox definition</td>
<td>MBX</td>
<td>Mailbox definition</td>
</tr>
<tr>
<td>Vector definition</td>
<td>EIT_ENTRY</td>
<td>Interrupt handler entry</td>
</tr>
<tr>
<td></td>
<td>CALL_ENTRY</td>
<td>Call vector entry</td>
</tr>
<tr>
<td></td>
<td>INIT_HDR_ENTRY</td>
<td>Initialization handler entry</td>
</tr>
<tr>
<td></td>
<td>RESET_ENTRY</td>
<td>Reset vector entry</td>
</tr>
</tbody>
</table>
3.2 Definition Statement Explanation

This section provides details on the definition statement formats.

Definition Statement Formats

**Definition Name**

<table>
<thead>
<tr>
<th>Format</th>
<th>Definition Name</th>
<th>Parameter List</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FR]</td>
<td>definition-name</td>
<td>parameter-1, parameter-2, ..., parameter-n</td>
<td>definition-name parameter-1, parameter-2, ..., parameter-n</td>
</tr>
<tr>
<td>[16]</td>
<td>definition-name</td>
<td>parameter-1, parameter-2, ..., parameter-n</td>
<td>definition-name parameter-1, parameter-2, ..., parameter-n</td>
</tr>
<tr>
<td>[8L]</td>
<td>definition-name</td>
<td>parameter-1, parameter-2, ..., parameter-n</td>
<td>definition-name parameter-1, parameter-2, ..., parameter-n</td>
</tr>
</tbody>
</table>

**Definition Name**

[Explanation]
This section provides a description of the definition statement, including how to specify the parameters, the ranges of values that can be specified, and the keywords that can be selected.

[Example]
This section provides examples of how the definition statement is described in a file. Refer to these examples when describing the definition statement in a configuration file.

Configuration file

```
... definition-name parameter-list ... 
```

**Reading the Definition Statements**

- If a given definition statement is “Nonomissible”, that definition statement must always be specified at least once. If a definition statement is “Omissible”, the default value for that statement is assumed to have been specified.
- The definition name can be described in either upper-case or lower-case letters.
- Only one definition statement can be specified on one line, and lines cannot be continued.
3.2 Definition Statement Explanation

• Any characters that follow a “#” symbol in a definition statement are regarded as comments.

• The definition name and the parameters can be separated by any number of spaces or tab characters. (The “  ” symbol represents any number of spaces or tab characters.)

• Parameters are delimited by commas (“,”). Spaces or tab characters before and after these commas are ignored.

[Example]
TSK tsk_1, tsk_1_entry, ,0x1000,READY,3,,0xFFFFFFFF

• Parameters that are enclosed in square brackets (“[” and “]”) may be omitted.

[Example]
When defining a common stack, the following three specifications have the same meaning:

COM_STACK name
COM_STACK name,
COM_STACK name,1024

• Parameters that are enclosed in rounded brackets (“{” and “}”) indicate that one of the elements enclosed in the brackets is to be selected.

• The vertical bar (“|”) is used to delimit the elements in the list of choices. (The selected element can be described in either upper- or lower-case letters.)

[Example]

MAP_FILE  {LIST | NOLIST}

• Parameters that begin with “0x” are regarded to be values specified in hexadecimal notation.

• Parameters that begin with “0” are regarded to be values specified in octal notation.

• Parameters that begin with other values are regarded to be specified in decimal notation. (For example, “0400”, “256”, and “0x100” all represent the decimal value 256.)

• If an address, value, etc., that exceeds 32 bits is specified, only the least significant 32 bits are valid. (For example, the value “0x10001234” is regarded as “0x1234”.)

• If an external reference symbol, such as an entry, contains a character that cannot be used with the assembler/linker, an error is generated by the assembler/linker.

The following types of characters can be used in external reference symbols:

- Letters
- Numbers
- Underscores (“_”)

Note: However, the first character cannot be a number.
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.1 CPU

[FR] CPU cpu [Optional: x]
[8L] CPU cpu [, [mode]] [Optional: x]

CPU

[Explanation]
Specify the target CPU by inputting an MB number.
An error occurs if an unavailable parameter is designated or if this definition statement is used to define a duplicate target CPU.

[Example]
Specifying "MB91V100" as the target CPU:

```
Configuration file

CPU    MB91V100
```

CPU [16]

[Explanation]
This definition statement defines the mode register at CPU address 0xFFFFDF.

- cpu: This parameter specifies the target CPU series name as an MB number (MB90700/MB90750H/MB90220/MB90670 ...).
- mode: This parameter selects the microcomputer chip mode from single, internal ROM, and external ROM (SINGLE/IN_ROM/OUT_ROM).
- bus: This parameter selects the bus mode from 8-bit and 16-bit (8 bit/16 bit/no setting).
- multi: This parameter selects whether the bus is a multiplex bus (MULTI/NOMULTI/no setting).
- exec_ram: This parameter selects whether the built-in RAM program is used (EXEC_RAM/RAM/no setting).

Omit parameters that are not related to the target CPU.
An error occurs if any parameter that cannot be selected is specified or a target CPU is defined twice.
No definition statement can be omitted.
3.2 Definition Statement Explanation

[Example]

Setting an MB90550 CPU:
- CPU series name : MB90550
- Chip mode : External ROM
- Bus mode : 16 bits
- Multiplex bus : Omitted
- Built-in RAM : Omitted

```
Configuration file
   ...
   CPU MB90550,OUT_ROM
   ...
```

Setting an MB90753H CPU:
- CPU series name : MB90750H
- Chip mode : Internal ROM
- Bus mode : 8 bits
- Multiplex bus : NOMULTI
- Built-in RAM : Omitted

```
Configuration file
   ...
   CPU MB90750H,IN_ROM,8BIT,NOMULTI
   ...
```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

Setting an MB90223 CPU:
- CPU series name: MB90220
- Chip mode: External ROM
- Bus mode: 16 bits
- Multiplex bus: Omitted
- Built-in RAM: Usable

Configuration file

```
CPU MB90220,OUT_ROM,16BIT,,EXEC_RAM
```

Setting an MB90678 CPU:
- CPU series name: MB90675
- Chip mode: Single
- Bus mode: 8 bits
- Multiplex bus: Omitted
- Built-in RAM: Omitted

Configuration file

```
CPU MB90675,SINGLE,8BIT
```

In addition to the above settings, settings for each series are possible.

**CPU [8L]**

[Explanation]

This definition statement defines the reset mode at CPU address 0xFFFFD.

- **cpu**: This parameter specifies the target CPU series name using an MB number (MB89120/MB89620/MB89820/MB89920, ...).
- **mode**: This parameter selects the microcomputer chip mode from single chip mode and external bus allowable mode (SINGLE/BUS/no setting).

An error occurs if any parameter that cannot be selected is specified or a target CPU is defined twice.
[Example]

Setting an MB89144 CPU:
- CPU series name: MB89140
- Chip mode: Single

```
Configuration file

CPU MB89140,SINGLE
```

Setting an MB89629R CPU:
- CPU series name: MB89620R
- Chip mode: External bus allowable mode

```
Configuration file

CPU MB89620R,BUS
```

Setting an MB89875 CPU:
- CPU series name: MB89870
- Chip mode: Omitted (single)

```
Configuration file

CPU MB89870
```
3.2.2 RELOCATE_FILE

[FR] RELOCATE_FILE  filename  [Optional: x]
[16] RELOCATE_FILE  filename  [Optional: x]
[8L] RELOCATE_FILE  filename  [Optional: x]

RELOCATE_FILE

[Explanation]
This definition statement specifies the name of a user relocatable object to be linked to the system during the linking process. The specified file name cannot include a comma (",") however.
This definition statement can be defined as many times as necessary.

[Example]
Linking the relocatable objects “test1.obj” and “test2.obj”:

Configuration file

```
RELOCATE_FILE  test1.obj
RELOCATE_FILE  test2.obj
```

3.2 Definition Statement Explanation

3.2.3 LIBRARY_FILE

<table>
<thead>
<tr>
<th>FR</th>
<th>LIBRARY_FILE</th>
<th>filename</th>
<th>[Optional: ○ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>LIBRARY_FILE</td>
<td>filename</td>
<td>[Optional: ○ ]</td>
</tr>
<tr>
<td>8L</td>
<td>LIBRARY_FILE</td>
<td>filename</td>
<td>[Optional: ○ ]</td>
</tr>
</tbody>
</table>

**LIBRARY_FILE**

[Explanation]

This definition statement specifies the name of a user library to be linked to the system during the linking process. The specified file name cannot include a comma (",") however.

This definition statement can be defined as many times as necessary.

If this definition statement is omitted, no libraries are linked.

[Example]

Linking the libraries “realos.lib” and “userlib.lib”:

```
LIBRARY_FILE
realos.lib
```

Configuration file

```
3.2.4 ABSOLUTE_FILE

[FR] ABSOLUTE_FILE  filename  [Optional: o ]
[16] ABSOLUTE_FILE  filename  [Optional: o ]
[8L] ABSOLUTE_FILE  filename  [Optional: o ]

ABSOLUTE_FILE

[Explanation]
This definition statement specifies the name of the absolute object to be output during the linking process.

If this definition statement is used twice to define an absolute object name, the definition that appears last is valid.

If this definition statement is omitted, “realos.abs” is used for the name of the absolute object that is output.

[Example]
Specifying “realos01.abs” as the name of the absolute object to be output during the linking process:

Configuration file

```
<table>
<thead>
<tr>
<th>Configuration file</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ _ _ _ _ _</td>
</tr>
<tr>
<td>_ _ _ _ _ _</td>
</tr>
<tr>
<td>_ _ _ _ _ _</td>
</tr>
<tr>
<td>ABSOLUTE_FILE      _ realos01.abs</td>
</tr>
<tr>
<td>_ _ _ _ _ _</td>
</tr>
<tr>
<td>_ _ _ _ _ _</td>
</tr>
</tbody>
</table>
```

3.2.5 MAP_FILE

[FR] MAP_FILE {LIST I NOLIST} [Optional: ○ ]
[16] MAP_FILE {LIST I NOLIST} [Optional: ○ ]
[8L] MAP_FILE {LIST I NOLIST} [Optional: ○ ]

MAP_FILE

[Explanation]
This definition statement specifies whether the Linker should output a map file.

- LIST: The Linker outputs a map file.
- NOLIST: The Linker does not output a map file.

If "LIST" is specified, the name of the map file that is output by the linker is the name of the absolute object with the extension changed to ".mp1".

If this definition statement is used twice to specify whether the linker should output a map file, the definition that appears last is valid.

If this definition statement is omitted, a map file is output.

The map file can be used to check the list of options that have been sent to the linker by the configurator and the list of sections that have been allocated.

[Example]
Outputting a map file:

```
Configuration file

MAP_FILE LIST
```

Not outputting a map file:

```
Configuration file

MAP_FILE NOLIST
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.6 CYC_HDR_NUMBER

- CYC_HDR_NUMBER number
- Optional: o

[Explanation]
This statement defines the number of cyclic activation handlers used by the system.

The number of cyclic activation handlers can be specified within the following ranges:

- [FR] 0 to 32767
- [16] 0 to 127
- [8L] 0 to 127

If this definition statement is used twice to define the number of cyclic handlers, the definition that appears last is valid.

If this definition statement is omitted, the number of cyclic activation handlers is 0.

When installing cyclic activation handlers for the FMC-16 and FMC-8L, define the system clock handler return method.

For the amount of memory used in this definition, see Sections 2.4.3, 2.5.3, and 2.6.3, “System Definition.”

[Example]
Defining the number of cyclic activation handlers as “5”:

```
Configuration file

... ...

CYC_HDR_NUMBER  5

... ...
```
### 3.2.7 ALM_HDR_NUMBER

<table>
<thead>
<tr>
<th>[FR]  ALM_HDR_NUMBER</th>
<th>number</th>
<th>[Optional: ○ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16] ALM_HDR_NUMBER</td>
<td>number</td>
<td>[Optional: ○ ]</td>
</tr>
<tr>
<td>[8L]</td>
<td></td>
<td>[Optional: -   ]</td>
</tr>
</tbody>
</table>

**[Explanation]**

This definition statement defines the number of alarm handlers used by the system. The number of alarm handlers can be specified within the following ranges:

- [FR] 0 to 32767
- [16] 0 to 127

If this definition statement is used twice to define the number of alarm handlers, the definition that appears last is valid.

If this definition statement is omitted, the number of alarm handlers is 0.

**[Example]**

Defining the number of alarm handlers as “5”:

```
Configuration file

   ...
   ...
   ALM_HDR_NUMBER  5
   ...
   ...
```

When installing alarm handlers, define the system clock handler return method.

For the amount of memory used in this definition, see Sections 2.4.3, 2.5.3, and 2.6.3, “System Definition.”
EXC_HDR_ENTRY

[Explanation]

This definition statement defines, by means of a label or address, the entry for the exception handler that is called when a CPU exception or a system call exception (an error that occurs during execution of a system call) occurs.

If this definition statement is used twice to define the exception handler, the definition that appears last is valid.

If this definition statement is omitted, no exception handler is embedded.

[Example]

Defining the entry “exc_entry” for an exception handler:

```
Configuration file

EXC_HDR_ENTRY exc_entry
```

Note:

- Exception handler

It is necessary to define an exception handler in order to execute suitable exception processing and then continue normal processing if a CPU exception occurs in the target system.

If a CPU exception is generated without an exception handler having been defined, the user’s “system down” processing may be executed and the target system may not be able to continue with its processing correctly.
3.2 Definition Statement Explanation

3.2.9 CPU_EXTHDR_ENTRY

[FR] [Optional: - ]
[16] CPU_EXTHDR_ENTRY entry [Optional: ○ ]
[8L] [Optional: - ]

CPU_EXTHDR_ENTRY

[Explanation]
Defines the name of the exception handler to be called when a CPU exception occurs.
If more than one exception handler is defined using this definition statement, the last definition will be valid.
If this definition statement is omitted, no CPU exception handler will be installed.

[Example]
To define the entry "_cpu_exception" as the CPU exception handler:

```
Configuration file

CPU_EXTHDR_ENTRY _cpu_exception
```

CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.10 SYS_ExTHDR_ENTRY

[FR] SYS_ExTHDR_ENTRY entry[, [ {1 I 2} ]]

[Optional: - ]

[16] SYS_ExTHDR_ENTRY entry[, [ {1 I 2} ]]

[Optional: - ]

[8L]

[Optional: - ]

### SYS_ExTHDR_ENTRY

[Explanation]

Defines the name of the exception handler to be called when a system call exception occurs (when an error occurs during execution of a system call).

Either the number 1 or 2 should be selected (if this statement is omitted, the default value 1 is designated).

- 1 Perform exception handling for all errors
- 2 Perform exception handling for all errors except for time-outs (E_TMOOUT) and polling failures (E_PLFAIL).

If more than one exception handler is defined using this definition statement, the last definition will be valid.

If this definition statement is omitted, no exception handler will be installed.

[Example]

To define the entry "_scall_exception" as the system exception handler, with exception processing for all errors:

```
Configuration file

SYS_ExTHDR_ENTRY _scall_exception,1
```

- To define the entry "_scall_exception" as the system exception handler, with exception processing for all errors except for time-outs (E_TMOOUT) and polling failures (E_PLFAIL):

```
Configuration file

SYS_ExTHDR_ENTRY _scall_exception,2
```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.11 SCAL_EXC_ENTRY

[FR] [Optional: - ]
[16] [Optional: - ]
[8L] SCAL_EXC_ENTRY (NORMALIZE_2 | NORMALIZE_A [, [entry]] | OPTIMIZE) [Optional: o ]

SCAL_EXC_ENTRY

[Explanation]
This definition statement defines the entry name of the exception handler that will be called if optimization processing and exception processing (for an error at system clock execution) for the ret_int system call occurs.

- NORMALIZE_2: This parameter selects two-error processing.
- NORMALIZE_A: This parameter selects all-error processing.
- OPTIMIZE : This parameter selects optimization processing.

Processing of the system call (ret_int) to be returned from the interrupt handler is optimized. Alternatively, whether to install kernel error processing is selected and whether to install exception processing is set. If optimization is specified, no error handling is performed.

If NORMALIZE_A is selected, the all-error processing listed below is performed.

If NORMALIZE_2 is selected, the two-error processing listed below is performed.

If an entry name is set, it is used as the entry name of the exception handler that is called when exception processing occurs.

<table>
<thead>
<tr>
<th>NORMALIZE_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Reserved function error class processing</td>
</tr>
<tr>
<td>- Parameter error class processing</td>
</tr>
<tr>
<td>- Context error processing</td>
</tr>
<tr>
<td>- Incorrect object error class processing</td>
</tr>
<tr>
<td>- Execution-time error class processing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NORMALIZE_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Wait status forcible release error class processing</td>
</tr>
<tr>
<td>- Processing execution result error class processing</td>
</tr>
</tbody>
</table>

If this definition statement is used twice to define the exception handler name, the definition that appears last is valid.

If this definition statement is omitted, NORMALIZE_2 error processing is performed. The system call exception handler will not be installed.
3.2 Definition Statement Explanation

[Example]
For optimization

Selecting all-error processing and defining entry name scall_exception for the system exception handler:

```
Configuration file
SCAL_EXC_ENTRY  OPTIMIZE
```

Selecting two-error processing and defining entry name scall_exception for the system exception handler:

```
Configuration file
SCAL_EXC_ENTRY  NORMALIZE_A, scall_exception
```

Selecting all-error processing:

```
Configuration file
SCAL_EXC_ENTRY  NORMALIZE_A
```

Selecting two-error processing and defining entry name scall_exception for the system exception handler:

```
Configuration file
SCAL_EXC_ENTRY  NORMALIZE_2, scall_exception
```

Selecting two-error processing:

```
Configuration file
SCAL_EXC_ENTRY  NORMALIZE_2
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.12 PRIORITY_LEVEL

- PRIORITY_LEVEL

[FR] PRIORITY_LEVEL level [Optional: ☐]
[16] PRIORITY_LEVEL level [Optional: ☐]
[8L] PRIORITY_LEVEL level [Optional: ☐]

[Explanation]
This definition statement defines the task priority level.
The task priority level can be specified within the following ranges:

  [FR]  8 to 32
  [16]  1 to 16
  [8L]  1 to 4

If this definition statement is used twice to define the task priority level, the definition that appears last is valid.

If this definition statement is omitted, the priority level is as follows:

  [FR]  32
  [16]  16
  [8L]  4

Before task definition, set PRIORITY_LEVEL.

  [FR]  8 x task priority level
  [16]  2 x task priority level
  [8L]  4 x task priority level

If the priority level decreases, the amount of memory used increases. Set an appropriate level.
[Example]
Defining the priority level as 8 [example for FR]

Configuration file

```

PRIORITY_LEVEL  8
```

In this example, task priorities lower than task priority 8 (9 to 32) cannot be used. Therefore, a priority within the range specified by this priority level (1 to 8) must be specified when task startup priority in a task definition and the task priority in a program are changed.
3.2.13 INCLUDE

[FR] INCLUDE filename [Optional: ]

[16] INCLUDE filename [Optional: ]

[8L] INCLUDE filename [Optional: ]

**Include**

**[Explanation]**

This definition statement specifies an INCLUDE file in which data for a user module is defined. INCLUDE files cannot be nested.

This definition statement can be defined as many times as necessary.

If this definition statement is omitted, no INCLUDE files are loaded.

**[Example]**

Specifying that the INCLUDE file "usrdef.rcf" is to be loaded:

```
Configuration file

  ...
  ...

  INCLUDE usrdef.rcf

  ...
  ...
```
3.2.14 ILM

ILM

[Explanation]
Defines allowable interrupt levels during kernel execution.

This statement is used to define the scope of interrupts that are allowed during system calls issued by tasks. Handler interrupts levels for system calls issued by handlers remain unchanged, and are not affected by interrupt levels set using this statement.

The allowable range of interrupt levels is 0 to 6.

If more than one kernel operation interrupt level is defined using this definition statement, the last definition will be valid.

If this definition statement is omitted, the interrupt level will be level 6.

For most normal uses, interrupt level 6 is used.

[Example]
To set the kernel operation interrupt level to level 5

```
Configuration file

ILM 6
```

CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.15 CODE

[FR] [Optional:  -  ]
[16] CODE { 16F I 16L } [Optional:  ○  ]
[8L] [Optional:  -  ]

CODE

[Explanation]
Performs output definitions for F²MC-16F dedicated code
This statement selects output file codes for the use of a 16F microcontroller core.

• 16F  16F dedicated code output
• 16L  16L/16LX/16/16H code output

This setting has no significance unless a 16F microcontroller is used.
If more than one definition is made using this definition statement, the last definition will be valid.
If this definition statement is omitted, 16F code output will be selected.

[Example]
To designate 16L/16LX/16/16H microcontroller dedicated output code

```
Configuration file

 CODE 16L
```
SYS_CLOCK

[Explanation]
This definition statement defines the system clock handler return method.

- RET
- RETP (16 only)
- ret_int

If this definition statement is used twice to define the system clock handler, the definition that appears last is valid.

If this definition statement is omitted, no system clock is installed for an FMC-16. For an FMC-8L, processing terminates with the ret instruction.

Specify this setting according to the method of creating the system clock handler. For details, see the “Softune REALOS Kernel Manual”.

Table 3.2.16a Generating the System Clock Handler Return [16]

<table>
<thead>
<tr>
<th>System clock handler generation method</th>
<th>Part a</th>
<th>Part b</th>
<th>Part c</th>
<th>Pre-processing</th>
<th>Post-processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>C language or assembly language</td>
<td>Call</td>
<td>ret</td>
<td>Ret_int</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Calip</td>
<td>retp</td>
<td>Ret_int</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assembly language</td>
<td>jmp[p]</td>
<td>ret_int</td>
<td>–</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### 3.2 Definition Statement Explanation

#### Table 3.2.16b  Generating the System Clock Handler Return [8L]

<table>
<thead>
<tr>
<th>System clock handler generation method</th>
<th>Part a</th>
<th>Part b</th>
<th>Part c</th>
<th>Pre-processing</th>
<th>Post-processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly language</td>
<td>call</td>
<td>Ret</td>
<td>Ret_int</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>jmp</td>
<td>Retp_int</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>C language</td>
<td>Function call</td>
<td>Ret</td>
<td>Ret_int</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[Example]

Designating RET as the system clock handler return method

```plaintext
Configuration file

```

```plaintext
:::

SYS_CLOCK RET

:::
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.17  CONTEXT_REGISTER

[FR] [Optional:  -  ]
[16] CONTEXT_REGISTER    number  [Optional:  0  ]
[8L] [Optional:  -  ]

- CONTEXT_REGISTER

[Explanation]
This definition statement defines from bit information the memory register used as the context.
The context register can be specified within the range from 0 to 0xff.

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
RW7 & RW6 & RW5 & RW4 & RW3 & RW2 & RW1 & RW0
\end{array}
\]

\[=> \text{0xff}\]

If 0 is defined, register bank switching is used for context switching.
If any value other than 0 is specified, stack switching (context switching for general stack saving and recovery) is used for the specified register.
If this statement is used twice to define the number of context registers, the definition that appears last is valid.
If this definition statement is omitted, register bank switching is used.
For the context register, see Sections 2.5.3, "System Definition."

[Example]
Using memory register RW7 as the context

```
Configuration file

  .
  .
 CONTEXT_REGISTER    0x80
  .
  .
```

Note: To describe a task using the C compiler, specify 0 or 0xff.
3.2 Definition Statement Explanation

3.2.18 REGISTER_BANK

[FR] [Optional: - ]
[16] REGISTER_BANK number [Optional: o ]
[8L] REGISTER_BANK number [Optional: o ]

■ REGISTER_BANK

[Explanation]

This definition statement defines the register bank to be used. Specify this definition statement so that it includes the entire register bank area specified in the task definition. An error occurs if the defined value is lower than the specified register bank.

Specify the register bank as 0 to 31.

For an F^2MC-16, register bank 1 is used by the kernel and the register bank 1 area is automatically reserved during linking. Note that even if register bank 1 is specified in the configuration file, a user can use only register bank 0.

For an F^2MC-8L, register bank 0 is used by the kernel and the register bank 0 area is automatically reserved during linking.

Specifying register bank 0 has the same effect as specifying register bank 1.

Since this definition statement specifies 0 to n, banks cannot be specified individually.

If this definition statement is omitted, register bank 31 is assumed.

If this statement is used twice to define the register bank, the definition that appears last is valid.

<table>
<thead>
<tr>
<th>Amount of memory used [16]</th>
<th>16 x (1 + number) bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of memory used [8L]</td>
<td>8 x (1 + number) bytes</td>
</tr>
</tbody>
</table>

[Example]

Defining register banks

Register banks 0 to 5 are used. (Used memory: 96 bytes for an F^2MC-16, 48 bytes for an F^2MC-8L)

Configuration file

```
....
REGISTER_BANK 5
....
```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.19 wai_tsk

[FR]         [Optional: - ]
[16]  wai_tsk  {ON I OFF}   [Optional: o ]
[8L]  wai_tsk  {ON I OFF}   [Optional: o ]

wai_tsk

[Explanation]
This definition statement defines a wai_tsk system call function.
• ON   This parameter installs the wai_tsk system call function.
• OFF  This parameter does not install the wai_tsk system call function.

If this definition statement is used twice to define wai_tsk, the definition that appears last
is valid.
If this definition statement is omitted, the system call function is installed.
To install wai_tsk, define the system clock handler return method.

[Example]
Installing wai_tsk

Configuration file

```
  ...
  wai_tsk  ON
  ...
```
3.2 Definition Statement Explanation

### 3.2.20 SYS_STK_SIZE

<table>
<thead>
<tr>
<th>[FR] SYS_STK_SIZE size</th>
<th>[Optional: ○ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16] SYS_STK_SIZE size</td>
<td>[Optional: ○ ]</td>
</tr>
<tr>
<td>[8L] SYS_STK_SIZE size</td>
<td>[Optional: ○ ]</td>
</tr>
</tbody>
</table>

**SYS_STK_SIZE**

[Explanation]

This definition statement defines the system stack size.

The system stack size can be specified as a byte count in the following ranges:

- **[FR]** Multiple of four bytes. The minimum is 256 bytes.
- **[16]** Even value from 0x020 to 0xfffe
- **[8L]** Value from 0x0020 to 0x2000

If this definition statement is used twice to specify the system stack size, the definition that appears last is valid.

If this definition statement is omitted, the byte counts are as follows:

- **[FR]** 1024 bytes
- **[16]** 256 bytes
- **[8L]** 40 bytes

See Appendix B for the system stack calculation method.

[Example]

Defining the system stack size as 0x1000 bytes:

```
Configuration file

SYS_STK_SIZE 0x1000
```

Note:

- **System stack size**
  
The system stack is used by the kernel and by the interrupt handler. Define the system stack size with plenty of space.

  If the system stack size is insufficient, the target system will not operate correctly.
3.2.21 ROM

This definition statement defines the area that can be allocated by the target microcontroller.

**FR**
- top_address : 0x00000000 to 0xffffffff  Address that is a multiple of four
- bottom_address : 0x00000003 to 0xffffffff + 1  Address that is a multiple of four

**16**
- top_address : 0x000000 to 0xfffffe  Even number address
- bottom_address : 0x000001 to 0xffffff + 1  Even number address

**8L**
- top_address : 0x0000 to 0xfffc  Even number address
- bottom_address : 0x0001 to 0xffff + 1  Even number address

- territory_name : Area name

This definition statement can be specified two or more times. It cannot be omitted.

**Example**

Locating the ROM area at addresses 0x100000 to 0x10faff and naming the area "ROMarea"

```
ROM 0x100000,0x10faff,ROMarea
```

Locating the ROM area at addresses 0x100000 to 0x107fff and at addresses 0xff3000 to 0xffffff

```
ROM 0x100000,0x107fff,test
ROM 0xff3000,0xffffff,rom_test
```
3.2 Definition Statement Explanation

### 3.2.22 RAM

- **[FR]** RAM `top_address, bottom_address, territory_name`  
  - Optional: x
- **[16]** RAM `top_address, bottom_address, territory_name`  
  - Optional: x
- **[8L]** RAM `top_address, bottom_address, territory_name`  
  - Optional: x

**RAM**

[Explanation]

This definition statement defines the area that can be located by the target microcontroller.

- **[FR]** `top_address`: 0x00000000 to 0xfffffffc Address that is a multiple of four
  - `bottom_address`: 0x00000003 to 0xffffffff + 1 Address that is a multiple of four
- **[16]** `top_address`: 0x000000 to 0xfffffe Even number address
  - `bottom_address`: 0x000001 to 0xffffff + 1 Even number address
- **[8L]** `top_address`: 0x0000 to 0xfffc Even number address
  - `bottom_address`: 0x0001 to 0xffff + 1 Even number address
  - `territory_name`: Area name

This definition statement can be used two or more times. It cannot be omitted.

[Example]

Locating the RAM area at addresses 0x000000 to 0x003fff and naming the area name “RAMarea”

```
Configuration file
  ...
  RAM 0x000000,0x003fff,RAMarea
  ...
```

Locating the RAM area at addresses 0x000000 to 0x000fff and at addresses 0x020000 to 0x02ffff

```
Configuration file
  ...
  RAM 0x000000,0x000fff,test
  RAM 0x020000,0x02fff,ram_test
  ...
```
3.2.23 KNL_ROM

- [FR] KNL_ROM address [Optional: ○ ]
- [16] KNL_ROM address [Optional: ○ ]
- [8L] KNL_ROM address [Optional: ○ ]

**Explanation**

This definition statement defines the address at which the kernel code is located.

- [FR] address: 0x00000000 to 0xffffffff Address that is a multiple of four
- [16] address: 0x000000 to 0xfffffe Even number address
- [8L] address: 0x0000 to 0xfffe Even number address

If this definition statement is used twice to define the address where the kernel code is located, the definition that appears last is valid.

This definition statement cannot be omitted.

**Example**

Locating the kernel code in addresses 0x000c0000

```
$ Configuration file
  ...
  ...
  KNL_ROM 0x000c0000
  ...
  ...
```

---

106
3.2 Definition Statement Explanation

3.2.24 KNL_RAM

[FR]  KNL_RAM address  [Optional: ○ ]
[16]  KNL_RAM address  [Optional: ○ ]
[8L]  KNL_RAM address  [Optional: ○ ]

**[FR]**  KNL_RAM address  
address: 0x00000000 to 0xffffffff  Address that is a multiple of four
address: 0x000000 to 0xfffffe  Even number address
address: 0x0000 to 0xfffe  Even number address

**[16]**  KNL_RAM address  
address: 0x00000000 to 0xffffffff  Address that is a multiple of four
address: 0x000000 to 0xfffffe  Even number address
address: 0x0000 to 0xfffe  Even number address

**[8L]**  KNL_RAM address  
address: 0x00000000 to 0xffffffff  Address that is a multiple of four
address: 0x000000 to 0xfffffe  Even number address
address: 0x0000 to 0xfffe  Even number address

This definition statement defines the address at which the kernel data is located.

If this definition statement is used twice to define the address where the kernel data is located, the definition that appears last is valid.

This definition statement cannot be omitted.

**[Example]**

Locating the kernel code in addresses 0x00001000

```
Configuration file

::

:: KNL_RAM  0x00001000
::
::
```
3.2.25 USR_SECTION

[FR] USR_SECTION section_name [, [address I territory_name]] [Optional: o ]
[16] USR_SECTION section_name [, [address I territory_name]] [Optional: o ]
[8L] USR_SECTION section_name [, [address I territory_name]] [Optional: o ]

USR_SECTION

[Explanation]

This definition statement defines the address at which a section used by the user program is located. If the address is omitted, the section is located immediately after the previously defined section. However, if no sections have been defined, the location address cannot be omitted.

This definition statement can be specified as many times as necessary.

If this definition statement is omitted, all sections are located automatically.

Although the EIT table is automatically generated by the configurator, it can be located anywhere by using this definition statement.

If the “R_eit” section is not specified, it is automatically located by the linker function at an available address.

[Example]

Locating the segments named code1 and code2 from the area CODE

```
Configuration file

... 
ROM 0x20000,0x2ffff,CODE
USR_SECTION code1+code2, CODE
...
```

Locating the sections named code3 and code4 starting at address 0x1000

```
Configuration file

... 
USR_SECTION code3, 0x1000
USR_SECTION code4, 
...
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.26 SYSTEM_CALL

[FR] SYSTEM_CALL {TSK I TSYNC I SEM I FLG I MBX I INT I MPL I MPF I TIM I SYS} [Optional: ○ ]
SYSTEM_CALL systemcall_name

[16] [Optional: - ]
[BL] [Optional: - ]

■ SYSTEM_CALL

[Explanation]
This definition statement defines the system calls used by the system. System calls can be specified either by their function names or by their system call names. When a function name is specified, all system calls belonging to that function are regarded as having been specified.

When a system call name is specified, only that system call is defined.

- **TSK:** Defines all of the system calls in the task management functions so that they can be used.

  The system calls “sta_tsk”, “ext_tsk”, “ter_tsk”, “dis_dsp”, “ena_dsp”, “chg_pri”, “rot_rdq”, “rel_wai”, “get_tid”, and “ref_tsk” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **TSYNC:** Defines all of the system calls in the task association synchronization functions so that they can be used.

  The system calls “sus_tsk”, “rsm_tsk”, “frsm_tsk”, “slp_tsk”, “tslp_tsk”, “wup_tsk”, and “can_wup” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **SEM:** Defines all of the semaphore-related system calls so that they can be used.

  The system calls “sig_sem”, “wai_sem”, “twai_sem”, “preq_sem”, and “ref_sem” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **FLG:** Defines all of the eventflag-related system calls so that they can be used.

  The system calls “set_flg”, “clr_flg”, “wai_flg”, “twai_flg”, “pol_flg” and “ref_flg” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **MBX:** Defines all of the mailbox-related system calls so that they can be used.

  The system calls “snd_msg”, “rcv_msg”, “trcv_msg”, “prcv_msg”, “trcu_msg” and “ref_mbx” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **INT:** Defines all of the system calls in the interrupt management functions so that they can be used.

  The system calls “ret_int”, “loc_cpu”, “unl_cpu”, “chg_ilm”, and “ref_ilm” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)
3.2 Definition Statement Explanation

- **MPL:** This parameter defines all system calls to the fixed-size memory pool so that they can be used.
  
The system calls "get_blf", "pget_blf", "tget_blf", "rel_blf", and "ref_mpf" can be used. (The effect is the same as if all of the listed system calls had been defined.)

- **MPF:** This parameter defines all system calls to the fixed-size memory pool so that they can be used.
  
The system calls "get_blf", "pget_blf", "tget_blf", "rel_blf", and "ref_mpf" can be used. (The effect is the same as if all of the listed system calls had been defined.)

- **TIM:** Defines all of the system calls in the time management functions so that they can be used.
  
The system calls “set_tim”, “get_tim”, “dly_tsk”, “def_cyc”, “act_cyc”, “ref_cyc”, “def_alm” “ref_alm”, and “ret_tmr” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

- **SYS:** Defines all of the system calls in the system management functions so that they can be used.
  
The system calls “get_ver” and “ref_sys” can be used. (The effect is the same as if all of the listed system calls had been defined individually.)

This definition statement can be defined as many times as necessary.

If this definition statement is omitted, all system calls are selected in the target system.

**[Example]**

Defining all of the system calls in the task management functions and the system calls “snd_msg” and “rcv_msg” for use:

```
Configuration file

SYSTEM_CALL TSK
SYSTEM_CALL snd_msg
SYSTEM_CALL rcv_msg
```

...
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.27 TSK

[, [stacd] [, [ID] [, [exinf] [, [tmo]]]]]]]

[16] TSK  name, [tskid], entry [, [itskpri] [, [stksz]

[8L] TSK  name, [tskid], entry [, [itskpri] [, [stksz]
[, [RP_reg]]]]

TSK

[Explanation]

This definition statement defines tasks.

- **name:** This parameter specifies the label under which the task ID is to be stored. (This parameter must not be omitted.)

- **entry:** This parameter specifies the task entry by means of either a label name or an address. (This parameter must not be omitted.)

- **itskpri:** This parameter specifies the numeric value for the task startup priority (the priority level when the task is started up).

  If this parameter is omitted, the priority level (the minimum priority) specified in the system definition is assumed.

  The task startup priority must be equal to or higher than the priority level.

- **stack:** This parameter specifies either the size of the stack area used by the task, or else the name of the common stack. (See the description of the definition statement in Section 2.2.17, “COM_STACK.”)

  If this parameter is omitted, the stack size is assumed to be 1024 bytes.

  When specifying a stack size, specify a size that is a multiple of four.

  The stack size can be roughly determined by using the “Maximum stack usage calculation tool (musc),” which is a standard tool provided with the FR C compiler.

  However, it is necessary to increase the portion (96 bytes) used for the task context information.

- **tskstat:** This parameter specifies the status of the task at system startup.

  - **DORMANT:** The initial status of the task is “DORMANT”.

  - **READY:** The initial status of the task is “READY”.

  If this parameter is omitted, the initial status of the task is “DORMANT”.

- **stacd:** This parameter specifies as a numeric value the start code that is passed to the task when it is executed. This code is valid only when the initial status is “READY”; this code is ignored if the initial status is “DORMANT”.

  If this parameter is omitted, “0” is assumed.
3.2 Definition Statement Explanation

- **ID**: This parameter specifies the task ID number as a numeric value. (ID: 1 to 32767)
  
  If the task ID is omitted, an available number is assigned automatically.
  
  Note that ID numbers must be specified consecutively.

- **exinf**: This parameter specifies extended information, by means of either a numeric value or a label name.
  
  If this parameter is omitted, “0” is assumed.

- **tmo**: This parameter specifies whether use of the object wait system call with a task time-out is to be allowed.
  
  - **NOTMO**: Use of the object wait system call with a time-out is not allowed.
  
  - **TMO**: Use of the object wait system call with a time-out is allowed.

  If this specification is omitted, "NOTMO" is assumed.

  When TMO is specified, the usable system calls are twai_sem, twai_flg, trcv_msg, and tget_blf. This definition places no restrictions on tsip_tsk and dly_tsk.

**[Example]**

Defining a task with the following parameters:

- **Task name**: “tsk_1”
- **Entry**: “tsk_1_entry”
- **Startup priority**: Default value
- **Stack size**: Default value
- **Initial status**: Default value
- **Start code**: Default value
- **Task ID**: Automatic assignment
- **Extended information**: Default value
- **Time-out function**: Default value

```plaintext
Configuration file

TSK tsk_1, tsk_1_entry
```

113
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

Defining a task with the following parameters:

- Task name: “tsk_2”
- Entry: “tsk_2_entry”
- Startup priority: 3
- Stack size: 0x1000
- Initial status: READY
- Start code: 3
- Task ID: Automatic assignment
- Extended information: 0xFFFFFFFF
- Time-out function: Allowed

Configuration file

```
TSK tsk_2,tsk_2_entry,3,0x1000,READY,3,,0xFFFFFFFF,TMO
```

Note:

- Task stack size
Define the task stack size with plenty of space.
If the task stack size is insufficient, the target system will not operate correctly.

■ TSK [16]

[Explanation]

This definition statement defines a task.
This definition statement cannot be omitted.

Set the priority level before defining the task.

- name: This parameter specifies the label indicating where the task ID is stored. (Required)
- tskid: This parameter specifies the task ID number (ID = 1 to 255). If the task ID is omitted, an available number is automatically assigned. Specify ID numbers consecutively.
- entry: This parameter specifies the task entry with the label name. (Required)
- itskpri: This parameter specifies the task activation priority (priority at task activation) with a numeric value. If the task activation priority is omitted, the lowest priority level (PRIORITY_LEVEL) specified by the system definition is used. The task activation priority must be higher than the priority level.
- stksz: This parameter specifies the size of the stack area to be used by the task as 0x000e to 0xfffe, or specifies the common stack name and stack name. If the stack size is omitted, 0x100 bytes are used.
- RP_reg: This parameter specifies RP registers 0 to 31. Do not specify register bank 1
3.2 Definition Statement Explanation

because it is used by the kernel. If RP_reg is omitted, 0 is assumed.

- **DTB:** This parameter specifies data bank registers 0x00 to 0xff. If DTB is omitted, 0 is assumed.
- **ADB:** This parameter specifies additional data bank registers 0x00 to 0xff. If ADB is omitted, 0 is assumed.
- **DPR:** This parameter specifies direct page registers 0x00 to 0xff. If DPR is omitted, 0 is assumed.
- **CCR:** This parameter specifies condition code registers 0x00 to 0xff. If CCR is omitted, 0x40 (interrupt allowable, user stack usable) is assumed.

[Example]

Defining a task with the following parameters:

- **Task name:** “tsk_1”
- **Task ID:** 3
- **Entry:** “tsk_1_entry”
- **Initial priority:** 5
- **Stack size:** 0x1000
- **RP register:** 0
- **Data bank register:** 0x10
- **Additional data bank register:** 0x50
- **Direct page register:** 0x80
- **Condition code register:** 0x40

```
TSK tsk_1,3,tsk_1_entry,5,0x1000,0x10,0x50,0x80,0x40
```

Configuration file
Defining a task with the following parameters:

- Task name: “tsk_2”
- Task ID: Automatic
- Entry: “tsk_2_entry”
- Initial priority: Omitted
- Stack size: tsk_1
- RP register: Omitted
- Data bank register: Omitted
- Additional data bank register: Omitted
- Direct page register: Omitted
- Condition code register: Omitted

```
Configuration file

TSK tsk_2,,tsk_2_entry,,tsk_1
```

Defining a task with the following parameters:

- Task name: “tsk_3”
- Task ID: Automatic
- Entry: “tsk_3_entry”
- Initial priority: Omitted
- Stack size: Omitted
- RP register: Omitted
- Data bank register: Omitted
- Additional data bank register: Omitted
- Direct page register: Omitted
- Condition code register: 0x44

```
Configuration file

TSK tsk_3,,tsk_3_entry,,,,,,,0x44
```

116
3.2 Definition Statement Explanation

**TSK [8L]**

[Explanation]
This definition statement defines a task.
This definition statement cannot be omitted.
Set the priority level before defining the task.

- **name**: This parameter specifies the label indicating where the task ID is stored. (Required)
- **tskid**: This parameter specifies the task ID number (ID = 1 to 127). If the task ID is omitted, an available number is automatically assigned. Specify ID numbers consecutively.
- **entry**: This parameter specifies the task entry with the label name. (Required)
- **itskpri**: This parameter specifies the task activation priority (priority at task activation) with a numeric value. If the task activation priority is omitted, the lowest priority level (PRIORITY_LEVEL) specified by the system definition is used. The task activation priority must be higher than the priority level.
- **stksz**: This parameter specifies the sizes of the stack area to be used by the task as 0x000e to 0x2000, or specifies the common stack name and stack name. If the stack size is omitted, 40 bytes are used.
- **RP_reg**: This parameter specifies RP registers 1 to 31. Do not specify register bank 0 because it is used by the kernel. If RP_reg is omitted, 1 is assumed.

[Example]
Defining a task with the following parameters:

- Task name: “tsk_1”
- Task ID: 1
- Entry: “tsk_1_entry”
- Initial priority: 4
- Stack size: 0x40
- RP register: 1

```
TSK tsk_1, 1, tsk_1_entry, 4, 0x40, 1
```

Configuration file
Defining a task with the following parameters:

- Task name: “tsk_2”
- Task ID: Automatic
- Entry: “tsk_2_entry”
- Initial priority: Omitted
- Stack size: Omitted (40)
- RP register: Omitted (1)

```
Configuration file

TSK tsk_2,,tsk_2_entry
```

Defining a task with the following parameters:

- Task name: “tsk_3”
- Task ID: Automatic
- Entry: “tsk_3_entry”
- Initial priority: Omitted
- Stack size: Omitted (40)
- RP register: 6

```
Configuration file

TSK tsk_3,,tsk_3_entry,,,6
```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.28 COM_STACK

<table>
<thead>
<tr>
<th>FR</th>
<th>COM_STACK name [,size]</th>
<th>Optional: o</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>COM_STACK name [,size]</td>
<td>Optional: o</td>
</tr>
<tr>
<td>8L</td>
<td>COM_STACK name [,size]</td>
<td>Optional: o</td>
</tr>
</tbody>
</table>

**COM_STACK**

[Explanation]

This definition statement defines task common stacks.

- **name:** This parameter specifies the name of the common stack. (This parameter must not be omitted.)
- **size:** This parameter specifies the size of the common stack.

For F²MC-16, the common stack size can be specified from 14 to 0xfffe. For F²MC-8L, the common stack size can be specified from 0x000e to 0x2000.

If this parameter is omitted, the following sizes are assumed:

- [FR] 0x400 bytes
- [16] 0x100 bytes
- [8L] 40 bytes

When a common stack is used, one stack can be shared by more than one task.

An error is generated if the same common stack name is specified twice.

**Note:**

When multiple tasks that are sharing a stack (using the same common stack) are in a state other than the DORMANT state (READY, WAIT, etc.) simultaneously, the target system will not operate correctly.

Tasks that operate simultaneously must either use different common stacks or else must not use...
common stacks. (See Figure 2.2.17.)

Figure 3.2.28 Example Where the Common Stack Cannot Function Properly

In this example, task A and task B are both in the READY state, and are both using common stack I. As a result, the two tasks destroy each other’s stack, with the result that the system will not operate properly.

[Example]

Setting up the common stack “com_stack” for use by the tasks “tsk_1” and “tsk_2”:

```
Configuration file

    ...
    COM_STACK  com_stack,0x2000
    TSK tsk_1,tsk_1_entry,5,com_stack,READY
    TSK tsk_2,tsk_2_entry,5,com_stack,DORMANT
    ...
    ...
```
3.2.29 SEM_ICNT

**[FR]**

**[16]** SEM_ICNT icnt **[Optional: ○]**

**[8L]** SEM_ICNT icnt **[Optional: ○]**

---

**SEM_ICNT**

**[Explanation]**

This definition statement is defined when the same initial count value, which is fixed, is set for each semaphore ID.

The initial count value can be 0 to 255.

If this definition statement is omitted, specify an initial count for each ID.

**[Example]**

Defining a semaphore initial count value as the fixed value 128

```
Configuration file

SEM_ICNT  128
```

---

122
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.30  SEM


SEM [FR]

[Explanation]
This definition statement defines semaphores.

- name: This parameter specifies the label under which the semaphore ID is to be stored. (This parameter must not be omitted.)
- isemcnt: This parameter specifies the count (number of resources) at system startup as a numeric value ranging from 0 to 32767.
  If this parameter is omitted, “1” is assumed.
- maxcnt: This parameter specifies the maximum value for the semaphore count as a numeric value.
  If this parameter is omitted, “32767” is assumed.
- ID: This parameter specifies as a numeric value the semaphore ID number. (ID: 1 to 32767)
  If this parameter is omitted, an available number is assigned automatically. Note that ID numbers must be specified consecutively.
- exinf: This parameter specifies extended information, by means of either a numeric value or a label name.
  If this parameter is omitted, “0” is assumed.

[Example]
Defining a semaphore with the following parameters:

- Semaphore name: “sem_1”
- Initial count: 10
- Maximum count: 20
- Semaphore ID: 3
- Extended information: 0xFFFF

Configuration file

```
SEM sem_1,10,20,3,0xFFFF
```

124
3.2 Definition Statement Explanation

Defining a semaphore with the following parameters:

• Semaphore name: “sem_2”
• Initial count: Default value
• Maximum count: Default value
• Semaphore ID: Automatic assignment
• Extended information: Default value

Configuration file

SEM sem_2

Defining a semaphore with the following parameters:

• Semaphore name: “sem_3”
• Initial count: 10
• Maximum count: 20
• Semaphore ID: Automatic assignment
• Extended information: 0xFFFFFFFF

Configuration file

SEM sem_3,10,20,,,0xFFFFFFFF
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

SEM [16][8L]

[Explanation]
This definition statement defines a semaphore.

- **name**: This parameter specifies the label indicating where the semaphore ID is stored. (Required)
- **semid**: This parameter specifies the semaphore ID number as a numeric value (ID = 1 to 127). If the semaphore ID is omitted, an available number is automatically assigned. Specify ID numbers consecutively.
- **isemcnt**: This parameter specifies the count value as a numeric from value 0 to 255 at system activation. If this parameter is omitted, 1 is assumed for the FMC-16 and 0 is assumed for the FMC-8L. If the SEM_ICNT definition statement is specified, count values at system activation are ignored.

[Example]
Defining a semaphore with the following parameters:

- Semaphore name: “sem_1”
- Semaphore ID: 3
- Initial count: 10

Configuration file

```
SEM sem_1,3,10
```

Defining a semaphore with the following parameters:

- Semaphore name: “sem_2”
- Semaphore ID: Automatic
- Initial count: Omitted (0)

Configuration file

```
SEM sem_2
```
3.2 Definition Statement Explanation

Defining a semaphore with the following parameters:

- Semaphore name: "sem_3"
- Semaphore ID: Automatic
- Initial count: 2 (fixed)

Configuration file:

```
SEM_ICNT 2
SEM       sem_2
```
3.2.31 FLG_BIT

[FR] [16] FLG_BIT { 16BIT I 1BIT} [Optional: o ]
[8L] [Optional: - ]

FLG_BIT

[Explanation]
Defines an eventflag type.

• 16BIT
• 1BIT

This definition should be made before eventflags are defined. Eventflag type definition should not be changed during the course of eventflag definition.

If this definition statement is omitted, the value will be “1BIT.”

[Example]
To designate 16-bit control pattern

Configuration file

FLG_BIT 16BIT
FLG flg_1,1,0xffff

To designate 1-bit control pattern

Configuration file

FLG_BIT 1BIT
FLG flg_1,1,1
### 3.2.32 FLG_PATTERN

**[FR]** [Optional: - ]

**[16]** FLG_PATTERN pattern [Optional: ○ ]

**[8L]** [Optional: - ]

#### FLG_PATTERN

[Explanation]

Defines whether an initial pattern value is to be fixed or variable by ID for the eventflag definition statement (FLG).

Be sure to define the eventflag type (FLG_BIT) before making this definition.

If the FLG_BIT statement designates the “1-BIT” type, the allowable range is 0 to 1. If the FLG_BIT statement designates the “16-BIT” type, the allowable range is 0 to 0xffff.

Values outside the allowable range will result in an error.

If this definition statement is omitted, the initial pattern should be designated with the eventflag definition statement (FLG).

[Example]

To set the eventflag initial pattern value to “1”

```plaintext

Configuration file

FLG_BIT       1BIT
FLG_PATTERN   1

```

To set the eventflag initial pattern value to “0x8000”

```plaintext

Configuration file

FLG_BIT       16BIT
FLG_PATTERN   0x8000

```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.33 FLG

[16] FLG name [, [flgid] [, [iflgptn]]]  [Optional: o ]
[8L] FLG name [, flgid]  [Optional: o ]

This definition statement defines eventflags.

- **name**: This parameter specifies the label under which the eventflag ID is to be stored. (This parameter must not be omitted.)
- **iflgptn**: This parameter specifies the initial pattern of the eventflag as a numeric value. For the FR, the initial pattern value can be 0 to 0xffffffff. For the F^2MC-16, the initial pattern value is specified as 0 or 1 when the FLG_BIT definition is “1 bit.” If the FLG_BIT definition is “16 bits,” the initial pattern value can be 0 to 0xffff. If any value other than these values is specified, an error occurs. If a fixed initial pattern (FLG_PATTERN) is defined, a defined initial pattern value is ignored. If this parameter is omitted, 0 is assumed.
- **flgid**: This parameter specifies the ID number of the eventflag as a numeric value. (1 to 32767 for the FR, 1 to 127 for the F^2MC-16 and F^2MC-8L) If this parameter is omitted, an available number is automatically assigned. Specify ID numbers consecutively.
- **exinf**: This parameter specifies extended information, by means of either a numeric value or a label name. If this parameter is omitted, “0” is assumed.

**Example for FR**

Defining an eventflag with the following parameters:

- Eventflag name:   “flg_1”
- Initial pattern:  5
- Eventflag ID:  3
- Extended information:  0xFFFF

```plaintext
FLG flg_1,5,3,0xFFFF
```

Configuration file
3.2 Definition Statement Explanation

Defining an eventflag with the following parameters:
- Eventflag name: “flg_2”
- Initial pattern: Default value
- Eventflag ID: Automatic assignment
- Extended information: Default value

```
Configuration file

FLG flg_2
```

Defining an eventflag with the following parameters:
- Eventflag name: “flg_3”
- Initial pattern: 0x0E0E
- Eventflag ID: Automatic assignment
- Extended information: 0xFFFFFFFF

```
Configuration file

FLG flg_3,0x0E0E,,0xFFFFFFFF
```

[Example for F^2MC-16]

Defining an eventflag with the following parameters:
- Eventflag management method: 16 bits
- Eventflag name: “flg_1”
- Eventflag ID: 3
- Initial pattern: 5

```
Configuration file

FLG_BIT 16BIT
FLG flg_1,3,0x0005
```

```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

Defining an eventflag with the following parameters:

- Eventflag management method: 1 bits
- Eventflag name: “flg_2”
- Eventflag ID: Automatic
- Initial pattern: Default value (0)

[Configuration file]

```
FLG_BIT 1BIT
FLG flg_2
```

[Example for F²MC-8L]

Defining an eventflag with the following parameters:

- Eventflag name: “flg_1”
- Eventflag ID: 3

[Configuration file]

```
FLG flg_1,3
```

Defining the eventflag with the following parameters:

- Eventflag name: “flg_2”
- Eventflag ID: Automatic

[Configuration file]

```
FLG flg_2
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.34 MBX

[16] MBX name [, mbxid] [Optional: ○ ]
[8L] MBX name [, mbxid] [Optional: ○ ]

[Explanation]

This definition statement defines mailboxes.

- **name:** This parameter specifies the label under which the mailbox ID is to be stored. (This parameter must not be omitted.)
- **mbxid:** This parameter specifies the mailbox ID number as a numeric value. (1 to 32767 for the FR, 1 to 127 for the FMC-16 and FMC-8L)

  If this parameter is omitted, an available number is automatically assigned.

  Specify ID numbers consecutively.

- **exinf:** This parameter specifies extended information, by means of either a numeric value or a label name.

  If this parameter is omitted, “0” is assumed.

[Example]

Defining a mailbox with the following parameters:

- Mailbox name: “mbx_1”
- Mailbox ID: 3
- Extended information: 0xFFFF

```
Configuration file

...  
MBX  mbx_1,3,0xFFFF
  ...
```

Defining a mailbox with the following parameters:

- **Mailbox name:** "mbx_2"
- **Mailbox ID:** Automatic assignment
- **Extended information:** Default value

Configuration file

```plaintext
MBX mbx_2
```

Defining a mailbox with the following parameters:

- **Mailbox name:** “mbx_3"
- **Mailbox ID:** Automatic assignment
- **Extended information:** 0xFFFFFFFF

Configuration file

```plaintext
MBX mbx_3,,0xFFFFFFFF
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.35  MPL

[FR] MPL  name, mplsiz, blksiz [, [ID] [, [exinf]]]  
[16] MPL  name, [mplid], mplsiz, blk  
[8L]  

MPL [FR]

[Explanation]

This definition statement defines memorypools.

• name:  This parameter specifies the label under which the memorypool ID is to be stored. (This parameter must not be omitted.)

• mplsiz:  This parameter specifies the size of the entire memorypool as a number of bytes. (This parameter must not be omitted.)

  Specify the memorypool size so that it is a multiple of four bytes.

• blksiz:  This parameter specifies the block size as a numeric value. (This parameter must not be omitted.)

  Specify the block size so that it is a multiple of four bytes, with a minimum of 20 bytes.

  If a block size that is greater than the size of the entire memorypool (mplsiz) is specified, an error results.

• ID:  This parameter specifies as a numeric value the memorypool ID number. (ID: 1 to 32767)

  If this parameter is omitted, an available number is assigned automatically.

  Note that ID numbers must be specified consecutively.

• exinf:  This parameter specifies extended information, by means of either a numeric value or a label name.

  If this parameter is omitted, “0” is assumed.

[Example]

Defining a memorypool with the following parameters:

• Memorypool name:  "mpl_1"
• Memorypool size:  2048
• Block size:  256
• Memorypool ID:  3
• Extended information:  0xFFFF

Configuration file

...  
MPL  mpl_1,2048,256,3,0xFFFF  
...
3.2 Definition Statement Explanation

Defining a memorypool with the following parameters:

- Memorypool name: “mpl_2”
- Memorypool size: 2048
- Block size: 256
- Memorypool ID: Automatic assignment
- Extended information: Default value

```
MPL mpl_2, 2048, 256
```

Defining a memorypool with the following parameters:

- Memorypool name: “mpl_3”
- Memorypool size: 2048
- Block size: 256
- Memorypool ID: Automatic assignment
- Extended information: 0xFFFFFFFF

```
MPL mpl_3, 2048, 256, 0xFFFFFFFF
```

### MPL [16]

[Explanation]

This definition statement defines a fixed-size memorypool.

- **name:** This parameter specifies the label indicating where the fixed-size memorypool ID is stored. (Required)
- **mplid:** This parameter specifies the fixed-size memorypool ID as a numeric value (1 to 127). If the fixed-size memorypool ID is omitted, an available number is automatically assigned. Specify ID numbers consecutively. (Optional)
- **mplsiz:** This parameter specifies the size of a single fixed-size memorypool as a byte count. Specify the size of a single fixed-size memorypool as an even number byte value in the range from 0x0004 to 0xfffe. (Required)
- **blk:** This parameter specifies the number of blocks in the range from 0x0001 to 0x3fff. Define this parameter so that the entire size of the fixed-size memorypool for each ID is within 64K bytes. (Required)
[Example]

Defining a fixed-size memorypool with the following parameters:

- Fixed-size memorypool name: "mpl_1"
- Fixed-size memorypool ID: 3
- Fixed-size memorypool size: 2048
- Number of blocks: 10

```
Configuration file

MPL mpl_1,3,2048,10
```

Defining a fixed-size memorypool with the following parameters:

- Fixed-size memorypool name: "mpl_1"
- Fixed-size memorypool ID: Automatic
- Fixed-size memorypool size: 1024
- Number of blocks: 20

```
Configuration file

MPL mpl_1,,1024,20
```
CHAPTER 3 CONFIGURATION DEFINITION STATEMENTS

3.2.36 MPF

[FR] MPF name, mpfcnt, blksiz [, [ID] [, [exinf]]]  
[Optional: ○ ]  
[16] [Optional: - ]  
[8L] [Optional: - ]

**MPF [FR]**

[Explanation]

This definition statement defines a fixed-size memorypool.

- **name**: This parameter specifies the label indicating where the fixed-size memorypool ID is stored. (Required)
- **mpfcnt**: This parameter specifies the number of blocks in the entire fixed-size memorypool. (Required)
- **blksiz**: This parameter specifies the block size as a numeric value. (Required)
  
  Specify the block size as a byte count that is a multiple of four. The minimum is 20 bytes. The size of the entire fixed-size memorypool is blksiz x mpfcnt.

- **ID**: This parameter specifies the ID number of the fixed-size memorypool as a numeric value (ID = 1 to 32767).
  
  If this parameter is omitted, an available number is automatically assigned.
  
  Specify ID numbers consecutively.

- **exinf**: This parameter specifies the extended information as a numeric value or label.
  
  If this parameter is omitted, 0 is assumed.

[Example]

Defining a memorypool with the following parameters:

- **Fixed-size memorypool name**: “mpf_1”
- **Number of blocks in the entire fixed-size memorypool**: 10
- **Block size**: 256
- **Fixed-size memorypool ID**: 3
- **Extended information**: 0xFFFF

Configuration file

```
MPL mpf_1,10,256,3,0xFFFF
```

140
3.2 Definition Statement Explanation

Defining a fixed-size memorypool with the following parameters:

- Fixed-size memorypool name: "mpl_2"
- Number of blocks in the entire fixed-size memorypool: 10
- Block size: 256
- Fixed-size memorypool ID: Automatic
- Extended information: Default value

```
MPF mpf_2,10,256
```

Defining a fixed-size memorypool with the following parameters:

- Fixed-size memorypool name: "mpl_3"
- Number of blocks in the entire fixed-size memorypool: 10
- Block size: 256
- Fixed-size memorypool ID: Automatic
- Extended information: 0xFFFFFFFF

```
MPF mpf_3,10,256,,0xFFFFFFFF
```
CHAPTER 3  CONFIGURATION DEFINITION STATEMENTS

3.2.37  EIT_ENTRY

[F] EIT_ENTRY  eit_no, entry  [Optional:  ]
[16] EIT_ENTRY  eit_no, entry  [Optional:  ]
[8L] EIT_ENTRY  eit_no, entry  [Optional:  ]

EIT_ENTRY

[Explanation]
This definition statement defines EIT vectors.

[FR]
- eit_no: This parameter specifies the vector number. (This parameter must not be omitted.)
  Excluding those reserved for the system, the following vector numbers may be specified: 16 to 62, and 65 to 255.
  Vector numbers 0 to 15, 63 and 64 are reserved by the system and cannot be defined by users.
- entry: This parameter specifies the EIT vector entry by means of either a label name or an address. (This parameter must not be omitted.)

[16] [8L]
- eit_no: This parameter defines the IRQ n number.
- entry: This parameter specifies the interrupt vector entry with a label or an address value.

Define the timer handler for the kernel system clock here.

For the F^2MC-16, 11 to 41, excluding system-reserved values, can be assigned as interrupt numbers. Since interrupt numbers 8, 9, and 10 are reserved by the system, a user cannot define them.

For the F^2MC-8L, interrupt numbers can be assigned 0 to 11.

[Example]
Defining an EIT vector with the following parameters:
- Vector number: 28
- Entry: “int_28_entry”

Configuration file

```
EIT_ENTRY  28, int_28_entry
```
Defining an interrupt handler with the following parameters:

- Interrupt number: 7
- Entry: "sci0_tx_hdr"

Configuration file

```
...  
EIT_ENTRY 7, sci0_tx_hdr  
...  
```

**Note:**

- Interrupt handlers

  Interrupt handlers must be defined for all interrupts (vector numbers 16 to 62) that are generated in the target system.

If an interrupt for which no interrupt handler is defined is generated, the user’s “system down” processing may be executed and the target system may not be able to continue with its processing correctly.
3.2.38 INIT_HDR_ENTRY

[FR] [Optional: - ]
[16] INIT_HDR_ENTRY entry [Optional: o ]
[8L] INIT_HDR_ENTRY entry [Optional: o ]

- INIT_HDR_ENTRY

[Explanation]

Defines the entry for the initialization handler called at operating system startup.

If more than one initialization handler definition is made using this definition statement, the last definition will be valid.

If this definition statement is omitted, the initialization handler name will be "_inihdr."

Note that initialization handlers must be created by the user.

The initialization handler performs operations including startup of initialization tasks.

[Example]

To define an initialization handler with the entry "_inithdr"

- Configuration file
  
  ...  
  INIT_HDR_ENTRY _inithdr
  ...  

3.2.39 CALL_ENTRY

[FR] [Optional: - ]
[16] [Optional: - ]
[8L] CALL_ENTRY entry, call_no [Optional: ○ ]

### CALL_ENTRY

**[Explanation]**

This definition statement defines a call vector.

- **entry:** This parameter specifies the label of the call vector. (Required)
- **call_no:** This parameter specifies the call vector number. (Required)

Call vector numbers can be from 2 to 7.

If this definition statement is omitted, a user does not define the call vector.

**[Example]**

Defining a call vector with the following parameters:

- **Vector number:** 7
- **Entry:** “cycle_set"

```
CALL_ENTRY cycle_set,7
```

Defining a call vector with the following parameters:

- **Vector number:** 2
- **Entry:** “sta_tsk"

```
CALL_ENTRY sta_tsk,2
```
3.2.40 RESET_ENTRY

[FR] RESET_ENTRY entry  [Optional:  o  ]
[16] RESET_ENTRY entry  [Optional:  o  ]
[8L] RESET_ENTRY entry  [Optional:  o  ]

■ RESET_ENTRY

[Explanation]
This definition statement defines a reset vector.
If this statement is used twice to define a reset vector, the definition that appears last is valid.
If this definition statement is omitted, the reset vector name is “_sys_entry”.

[Example]
Defining the entry name of a reset vector as “_sys_entry”

```
Configuration file

RESET_ENTRY     _sys_entry
```

PART 2  NON-WINDOWS VERSION

Part 2 describes how to operate the non-Windows version of the configurator.

Chapter 4  Operation of the Non-Windows Version
CHAPTER 4  OPERATION OF THE NON-WINDOWS VERSION

This chapter explains the operation of the non-Windows version of the configurator. This chapter explains the startup method and options. The non-Windows version of the configurator is the configurator used under the DOS prompt.

4.1 Configurator Startup Command
4.2 Option for the Monitor Debugger
4.1 Configurator Startup Command

The following is the startup statement for the configurator:


- Configurator Startup Command
  - Description
    The configurator startup command starts the REALOS configurator.
    
    One of the two operating modes may be specified for the configurator as follows:
    
    • When the -a option is specified:
      After assembling a kernel source file, the resulting kernel object file is linked with user relocatable object files. When the FR is used, the kernel object file is generated under $RTOS\rtos\911\obj, where it also remains after execution. When the FMC-16 or FMC-8L is used, a library file is generated.
      
      When the configuration file is modified, the kernel object file must be generated again.
    
    • When the -a option is not specified:
      A kernel object file under $RTOS\rtos\911\obj is linked with user object files. Since this method does not assemble the kernel source file, a system can be built quickly.
      
      When the embedded version is used, the -a option must be specified to generate a kernel object file in advance.

    Reference: $RTOS indicates a directory defined by the RTOS environment variable.

- Options
  -V
    Displays the version of the configurator.
  
  -XV
    Does not display the version of the configurator.
  
  -help
    Displays help.
  
  -a
    Assembles a kernel source file, and links the generated kernel object file and user object files.
  
  -jmp [16]
    See Section 4.2, “Option for the Monitor Debugger.”
4.1 Configurator Startup Command

-\textbf{f}

Specifies a configuration file.
Whenever this option is specified, a configuration file must be specified.
The suffix of the configuration file name cannot be omitted.

-GHS [FR]
Specify this option when using a tool supplied by GHS.

- \textbf{Startup examples}

\% frcfs
The configurator displays the version and help, then terminates.

\% frcfs -V -f realos.rcf
After displaying the version, the configurator reads configuration file realos.rcf, and
performs configuration.

\textbf{Note:} Since the configurator activates the assembler and linker during internal processing, the
directory holding each of these tools must be defined in the \texttt{PATH} environment variable in
advance.
Before the configurator can start, the \texttt{RTOS} environment variable must have been defined
on the host machine.
When executing configuration, the configurator reads files under the directory defined by
the \texttt{RTOS} environment variable. Therefore, before starting the configurator, be sure to
define the directory in which \texttt{REALOS} is installed. The configurator uses an intermediate
file that is created under the directory defined by the \texttt{TMP} environment variable. If the
configurator is executed when this environment variable is not defined, the intermediate
file is created under the current directory.
The -jmp option outputs an interrupt vector table as a jump table. This function is available only for the F²MC-16 family.

### Option for the Monitor Debugger

The configurator usually creates a vector table for vector addresses specified with the EIT_ENTRY and RESET_ENTRY configuration definition statements.

When the -jmp option is specified, this vector table is created as a jump table.

The vector addresses specified with the configuration definition statements shown in Figure 4.2a are usually output as the vector table shown in Figure 4.2b.

#### Figure 4.2a Configuration Definition Statements

```assembly
.SECTION INTVEC, CODE, ALIGN=1
.DATA.L r97_dispatch ; (IRQ42/ICR15)
.DATA.L 0 ; (IRQ41/ICR15)

.DATA.L uartTXint ; (IRQ13/ICR 1)
.DATA.L 0 ; (IRQ12/ICR 0)

.DATA.L uartRXint ; (IRQ11/ICR 0)
.DATA.L r97_cpu_expt ; (IRQ10/------)
.DATA.L 0 ; (IRQ9 /------)
.DATA.E _sys_entry ; (IRQ8 /------)
.DATA.B 0 x 88
.END
```

**Note:** The vector address in IRQ42 is always registered by the configurator. The vector address in IRQ10 is registered only when the CPU exception function has been included.

#### Figure 4.2b Output Example of an Interrupt Vector Table
4.2 Option for the Monitor Debugger

Note: The vector address in IRQ42 is always registered by the configurator. The vector address in IRQ10 is registered only when the CPU exception function has been included.

Figure 4.2c Output Example of a Jump Table

```
.SECTION INTVEC, CODE, ALIGN=1
jmp r97_dispatch ; (IRQ42/ICR15)
jmp 0 ; (IRQ41/ICR15)
:
:
jmp 0 ; (IRQ14/ICR1)
jmp uartTXint ; (IRQ13/ICR 1)
jmp 0 ; (IRQ12/ICR 0)
jmp uartRXint ; (IRQ11/ICR 0)
jmp r97_cpu_expt ; (IRQ10/-----)
jmp 0 ; (IRQ9 /-----)
jmp _sys_entry ; (IRQ8 /-----)
.END
```
PART 3  SOFTUNE WORKBENCH VERSION OF THE CONFIGURATOR

Part 3 describes how to operate the Softune Workbench version (hereafter Workbench version) of the configurator, and also explains the definition dialogs for creating configuration files.

Chapter 5  Information for Softune Workbench Users
Chapter 6  Operation of the Workbench Version of the Configurator
Chapter 7  Definition Dialogs in the Workbench Version of the Configurator
Chapter 8  Limitations of the Workbench Version of the Configurator
CHAPTER 5  INFORMATION FOR SOFTUNE WORKBENCH USERS

This chapter provides information you need to know before using the Workbench version of the configurator as well as outlines the Workbench version of the configurator.

5.1 Outline of the Workbench Version of the Configurator
5.1 Outline of the Workbench Version of the Configurator

This section outlines and provides notes on the Workbench version of the configurator.

- **Outline of the Workbench Version of the Configurator**
  - **REALOS configurator add-in module for Softune Workbench**
    
    The REALOS configurator add-in module is a REALOS function expansion module for 
    Softune Workbench. The module is in the form of a dynamic link library (DLL) form, allowing 
    it to be linked dynamically with Softune Workbench. Softune Workbench itself does have no 
    REALOS support function. If, however, the REALOS configurator add-in is installed, Softune 
    Workbench is automatically expanded so that the REALOS support function can be used.
  - **Added functions**
    
    The following functions are added:
    - A choice of whether to include the REALOS kernel (REALOS(ABS)) is added to [Project 
      Type] of [Create New Project].

    Figure 5.1a shows an example of the [Create New Project] dialog.

![Figure 5.1a](image_url)
5.1 Outline of the Workbench Version of the Configurator

- REALOS configuration setting screens are added. (System, memory, system call, task, semaphore, event flag, mailbox, memory pool, vector, and CPU setting dialogs are added.)

Figure 5.1b shows an example of the system definition dialog.

![Figure 5.1b Example of the System Definition Dialog (FR)](image)

- Linkage with the make function is enabled. When the REALOS configuration is modified, the system can be rebuilt automatically just by pressing the make button.
CHAPTER 5 INFORMATION FOR SOFTUNE WORKBENCH USERS

- Help information about the REALOS configurator is added to Softune Workbench help.
  - Pressing the F1 key with a system call name selected on the edit screen displays help information about the REALOS system call.

Figure 5.1c shows an example of displaying help information about a system call (ext_tsk) with a single operation.

![Figure 5.1c Example of Displaying Help Information about a System Call with a Single Operation](image)

**Note:** The REALOS configurator add-in module (SiRcf.DLL file) is supplied with the REALOS product. If just Softune Workbench itself is installed, the REALOS support function cannot be used. When the Softune REALOS product is purchased, Softune Workbench functions are expanded automatically just by installing Softune REALOS. This means that when REALOS is revised in the future by adding or modifying configuration functions or by modifying the contents of help files, only the REALOS product needs to be revised. In this way, Softune Workbench functions will be modified automatically.

Figure 5.1d shows the relationships between Softune Workbench and REALOS.
5.1 Outline of the Workbench Version of the Configurator

- External reference symbol file

Based on the object file names defined in the configuration file (tasks, semaphores, event flags, mailboxes, and variable-size and fixed-size memory pools), the configurator creates an external reference symbol file.

The external reference symbol file is created under the name “R_project-file-name.h” in the current directory in which a project has been created with Softune Workbench.

When the external reference symbol file name (for example, R_project-file-name.h) is included in a user program, symbols can be referenced.

**Note:** An external reference symbol file is created each time [Project]-[Make or Build] is performed with Softune Workbench.
CHAPTER 6    OPERATION OF THE WORKBENCH VERSION OF THE CONFIGURATOR

This chapter explains how to operate the Workbench version of the configurator. In the Workbench version of the configurator, editing configuration data is simple because related configuration data is grouped into several dialogs, making the data visually easy to understand. The subsections entitled “Related Definition Names” that appear in the explanations indicate definition statements that set and change the configuration file.

6.1 Starting the Workbench Version of the Configurator
6.2 Editing the Configuration File
6.3 Making a Project
6.1 Starting the Workbench Version of the Configurator

There are two methods for starting the Workbench version of the configurator.

- Starting the Workbench Version of the Configurator
  
The method for starting the configurator to create a new project depends on whether a new configuration file or an existing configuration file will be used.
6.1 Starting the Workbench Version of the Configurator

6.1.1 Starting the Configurator with a New Configuration File

This section explains the startup procedure used when a new configuration file is created.

- **When Starting the Configurator with a New Configuration File**
  
  First, select [New] from the file menu of the Softune Workbench main window. Figure 6.1.1a shows an example of the Softune Workbench main window.

  ![Figure 6.1.1a Example of the Softune Workbench Main Window (FR)](image)

Figure 6.1.1a Example of the Softune Workbench Main Window (FR)
CHAPTER 6  OPERATION OF THE WORKBENCH VERSION OF THE CONFIGURATOR

Next, from the [New] dialog, select [Project file].

Figure 6.1.1b shows an example of the [New] dialog of Softune Workbench.

![Figure 6.1.1b Example of the [New] Dialog of Softune Workbench](image)

Select [REALOS(ABS)] in [Project Type], and specify a project name in the [Project Name] field.

Figure 6.1.1c shows an example of the [Create New Project] dialog of Softune Workbench.

![Figure 6.1.1c Example of the [Create New Project] Dialog of Softune Workbench](image)

Clicking the [OK] button opens the [Create Configuration File] dialog.

Figure 6.1.1d shows an example of the [Create Configuration File] dialog.
6.1 Starting the Workbench Version of the Configurator

Figure 6.1.1d Example of the [Create Configuration File] Dialog

Select [New making], then click the [OK] button. The [System] dialog opens. (When [New making] is selected, a wizard dialog is opened.)

Figure 6.1.1e shows an example of the [System] dialog.

Figure 6.1.1e Example of the [System] Dialog (FR)

Enter information in the fields in the dialog, and click the [Next] button to continue making settings for the configurator.

When the settings are complete, click the [Finish] button. A configuration file is created.
**Note:** When the [Finish] button is clicked after configuration data has been set under [New making], the warning message shown in Figure 6.1.1f appears. Select the linker tab from the [Setup Tool Option] dialog in the project menu of Softune Workbench, select [Placement/Connection] from the category, and set the ROM and RAM areas.

![Figure 6.1.1f Example of the Warning Message](image)

**Figure 6.1.1f Example of the Warning Message**
CHAPTER 6  OPERATION OF THE WORKBENCH VERSION OF THE CONFIGURATOR

6.1.2  Starting the Configurator by Using an Existing Configuration File

This section explains the startup procedure used when an existing configuration file is read.

When Starting the Configurator by Using an Existing Configuration File

Open the [Create Configuration File] dialog shown in Figure 6.1.1d by following the same steps used to create a new configuration file.

From the [Create Configuration File] dialog, select [An existing Configuration file].

Figure 6.1.2a shows an example of the [Create Configuration File] dialog.

![Create Configuration File Dialog](image)

Figure 6.1.2a  Example of the [Create Configuration File] Dialog

After entering a file name, click the [OK] button. The [Set Configuration File] dialog (system definition) opens. (When [An existing Configuration file] is selected, a tabbed dialog opens.)

Figure 6.1.2b shows an example of the [Set Configuration File] dialog.
6.1 Starting the Workbench Version of the Configurator

Open a dialog that you want to modify. (Clicking a tab changes the display to an associated setting screen.) Edit the settings. After editing is complete, click the [OK] button. The configuration file is edited.

**Note:** When editing is performed with [An existing Configuration file] selected, the message shown in Figure 6.1.2c appears, since the relocatable object file (user object file) and library file defined in the configuration file are not reflected in Softune Workbench. Select [Add Member] from the Softune Workbench project menu, and add the user object file or library file.

![Figure 6.1.2b Example of the [Set Configuration File] Dialog](image)

![Figure 6.1.2c Example of the Warning Message Dialog](image)
6.2 Editing the Configuration File

This section explains how to edit the configuration file after opening an existing project file.

When an existing project file is opened, the configuration file (sample.rcf) is added to the project window, as shown in Figure 6.2a.

Figure 6.2a shows an example of the Softune Workbench project window.
6.2 Editing the Configuration File

To edit the configuration file, double-click its name. The [Set Configuration File] dialog opens. Use this dialog for configurator editing.

Figure 6.2b shows an example of the [Set Configuration File] dialog.

Open the dialog you want to modify. After editing is complete, click the [OK] button. The configuration file is edited.
6.3 Making a Project

This section explains the procedure for assembling a kernel source file, then compiling and linking it with user source files to create an absolute object file.

Making a Project

After specifying configuration data, select [Add Member] from the project menu of Softune Workbench. Figure 6.3a shows an example of the [Add Member] menu item of Softune Workbench.

Figure 6.3a  Example of the [Add Member] Menu Item of Softune Workbench
6.3 Making a Project

The [Add Member] dialog opens. Select and register user source files or library files you want to link with the kernel source.

Figure 6.3b shows an example of the [Add Member] dialog.

![Add Member Dialog Example](image)

**Figure 6.3b** Example of the [Add Member] Dialog
CHAPTER 6  OPERATION OF THE WORKBENCH VERSION OF THE CONFIGURATOR

After completing registration, select [Make] or [Build] from the project menu of Softune Workbench. Compilation, assembling, and linkage are performed.

Figure 6.3c shows an example of the [Build] menu item of Softune Workbench.

![Example of the [Build] Menu Item of Softune Workbench](image)

Softune Workbench assembles the kernel and compiles and links user source files. Figure 6.3d shows the execution results.
6.3 Making a Project

Figure 6.3d Example of Compilation and Linkage Execution Results
CHAPTER 7  DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

This chapter explains how to set definition dialogs in the Workbench version of the configurator and also explains the organization of these dialogs.

7.1 Definition Dialog Organization
7.2 System Definition Dialog
7.3 Memory Definition Dialog
7.4 System Call Definition Dialog
7.5 Task Definition Dialog
7.6 Semaphore Definition Dialog
7.7 EventFlag Definition Dialog
7.8 Mail Box Definition Dialog
7.9 Variable-size MemoryPool Definition Dialog
7.10 Fixed-size MemoryPool Definition Dialog
7.11 Vector Definition Dialog
7.12 CPU Setup Dialog
7.13 Debug Setup Dialog
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.1 Definition Dialog Organization

This section explains the operations and functions of the following five buttons that are common to all definition dialogs:

- [Next] button
- [Back] button
- [Finish] button
- [OK] button
- [Cancel] button

### Common buttons

The dialogs that appear when a new configuration file is created (called wizard dialogs) have the four buttons shown below.

Figure 7.1a shows an example of the wizard dialogs.

![Figure 7.1a](example.png)

**Figure 7.1a Example of the Wizard Dialogs**

- [Back] button: Opens the previous dialog. The data specified in the current dialog is discarded.
- [Next] button: Opens the next dialog. The data specified in the current dialog is saved.
- [Finish] button: The data set in all dialogs is reflected in the configuration file.
- [Cancel] button: Any specified data is discarded.

The dialogs that appear when an existing configuration file is loaded or when the configurator is opened from the project window (called tabbed dialogs) have the two buttons shown below.

Figure 7.1b shows an example of the tabbed dialogs.

![Figure 7.1b](example.png)

**Figure 7.1b Example of the Tabbed Dialogs**

- [OK] button: The current settings as edited with the definition dialogs are reflected in the configuration data.
- [Cancel] button: Clicking the [Cancel] button discards the changes.
CHAPTER 7  DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.1.1  Types of Dialog

There are 13 types of dialog:

- System definition dialog
- System_1 definition dialog
- System_2 definition dialog
- Memory definition dialog
- System call definition dialog
- Semaphore definition dialog
- Mailbox definition dialog
- Task definition dialog
- Eventflag definition dialog
- Variable-size memorypool definition dialog
- Vector definition dialog
- CPU setup dialog

## Types of Dialog

- System definition dialog (FR, 8L)
  Defines information related to the overall system.

- System_1 definition dialog (16)
  Defines system information. The information includes the number of cyclic activation handlers, number of alarm handlers, the CPU exception handler entry, the number of priority levels, whether to include the debugger, system call exception processing, and include names.

- System_2 definition dialog (16)
  Defines system information. The information includes kernel operation interrupt levels, the number of register banks, FMC-16F-dedicated code output, wait function embedding, system clock handler settings, and the context switching method.

- Memory definition dialog (FR, 16, 8L)
  Defines information related to memory.

- System call definition dialog (FR)
  Selects system calls used.

- Task definition dialog (FR, 16, 8L)
  Defines information related to specific tasks.

- Semaphore definition dialog (FR, 16, 8L)
  Defines information related to specific semaphores.

- Eventflag definition dialog (FR, 16, 8L)
  Defines information related to specific eventflags.

- Mailbox definition dialog (FR, 16, 8L)
  Defines information related to specific mailboxes.

- Variable-size memorypool definition dialog (FR)
  Defines information related to specific variable-size memorypools.
7.1 Definition Dialog Organization

- Fixed-size memorypool definition dialog (FR, 16)
  Defines information related to specific fixed-size memorypools.
- Vector definition dialog (FR, 16, 8L)
  Defines interrupt handler and vector information.
- CPU setup dialog (16, 8L)
  Defines CPU information.
CHAPTER 7  DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.1.2  List of File Names

A list of file names is used to edit file names.

■ List of File Names

If configuration data uses multiple file names, a list of file names, buttons for manipulating the list, and an edit box for inputting a file name are displayed. Figure 7.1.2 shows an example of a list of file names in the system definition dialog.

![Example of a List of File Names](image)

Figure 7.1.2  Example of a List of File Names
7.1.3 File Selection

Input files are selected with a file selection dialog box.

- File Selection

In the definition windows, the file selection dialog box is used to specify input or output files.

When the [Browse] button of [Setting of include file] field in the system definition dialog box is clicked, the file selection dialog box opens. Figure 7.1.3 shows an example of file selection dialog box.

Once a file name is input in the "[File name]" field and the "[OK]" button is clicked, the file name that was input is regarded as having been specified.

Only those files that match the wildcard specification in the "[File list]" field are displayed in the file list. When a file is selected from the file list by using the mouse, the name of that file is reflected in the "[File name]" field. Double-clicking a file in the file list causes that file to be regarded as having been specified immediately without the "OK" button being pressed.

To change the wildcard specification in the "[File list]" field, select a new file type from the drop-down list box.

![File selection dialog box display example](image)

Figure 7.1.3  File selection dialog box display example
An information list is used to edit multiple items of information.

- **Miscellaneous Lists**
  A list of multiple information items, buttons for manipulating the list, and edit boxes for entering the individual items are displayed. Figure 7.1.4 shows an example of the mailbox definition dialog.

![Example of a Miscellaneous List](image)

If the list is too long to fit vertically in the list box, a scroll bar is displayed. When a line is selected, the contents of the line are reflected in the edit boxes of the associated items. The [Add] button is used to add information specified for each item to the list box.

The [Upd] button updates the selection in the list box the with information specified for each item.

The [Del] button deletes the selection from the list box. When the last line in the list box is deleted, the preceding line is selected automatically.
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.2 System Definition Dialog

The system definition dialog defines the system-related information items listed below. Two types of system definition dialog are provided for the F₂MC-16.

- Number of cyclic activation handlers (common)
- Number of alarm handlers (FR, 16)
- CPU exception handler entry name (16)
- System call exception process (16)
- Number of register banks to be reserved (8L)
- F₂MC-16F dedicated code output (16)
- Embedding of the wai_tsk function (16, 8L)
- Context switching method (16)
- Specification of ret_int system call optimization (8L)
- Exception handler entry name (FR)
- Priority level (common)
- Kernel operation interrupt level (16)
- System call exception process (16)
- INCLUDE file (common)

System Definition Dialog

Figures 7.2a, 7.2b, 7.2c, 7.2d, and 7.2e show examples of system definition dialogs.

Figure 7.2a Example of the System Definition Dialog (FR)
Figure 7.2b  Example of the System_1 Definition Dialog (16)
CHAPTER 7  DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

Figure 7.2c  Example of the System_2 Definition Dialog (16)

Figure 7.2d  Example of the Save and Restore Register Selection Dialog (16)
Figure 7.2e  Example of the System Definition Dialog (8L)

- Setting items [common]
  [Cyclic Handler Counts] is specified with a numeric value.
  [Priority Level] is specified with a numeric value.
  [Include file Setting] specifies an include file name. For details on the operation of this list box, see Section 7.1.2, "List of File Names."
  For details of the definitions, see Sections 2.4.3, 2.5.3, and 2.6.3, "System Definitions."

- Setting items [8L]
  [Register Bank Secure Count] is specified with a numeric value. The specified value is passed to the linker.
  For [System Clock Handler Setting] and [wai_tsk embedding], make a selection as necessary.
  For [ret_int System Call Optimizing], make a selection as necessary. In the [Exception Handler] field, enter a label.
  When [OPTIMIZE] is selected, no entry in [Exception Handler] is allowed. (The default is NORMALIZE_2.)

- Setting items [16]
  [F2MC-16F Dedicated Code Output] is valid only when a 16F family device is selected as the CPU.
  [CPU Exception Handler Entry] specifies a label with a character string.
  For [Exception Execution Method] of [System Call Exception Process], select a method from
the drop-down list box as necessary.
Select [WAI_TSK Embedding] as necessary.

[Alarm Handler Count] is specified with a numeric value.

When the check box of [System Clock Handler Embedding] of [System Clock Handler Setting] is checked, the [Return Method] button is displayed.

Specify a desired return method from the drop-down list box as necessary.

If [Escape and Return in Stack] is selected in [Context Switching Method], the [Register Selection] button is enabled. Clicking this button opens the [Escaped and Returned Register Selection] dialog.

In this dialog, select the registers to be used, then click the [Close] button. The selected registers are set.

[Kernel Operation Interrupt Level] is specified with a numeric value. Normally, 6 is specified.

[Register Bank Secure Count] is specified with a numeric value. The specified value is passed to the linker.

- Setting items [FR]
  [Alarm Handler Count] is specified with a numeric value.
  [Exception Handler Entry] specifies a label with a character string, or specifies an address with a numeric value.

**Note:** Exception handler
If a CPU exception is generated without an exception handler having been defined, processing cannot continue properly.

### Definition Names Related to the System Definition Dialog

- **[Common]** CYC_HDR_NUMBER, PRIORITY_LEVEL, INCLUDE
- **[8L]** SCAL_EXC_ENTRY, SYS_CLOCK, REGISTER_BANK, WAI_TSK
- **[16]** ALM_HDR_NUMBER, CPU_EXTHDR_ENTRY, SYS_EXTHDR_ENTRY, ILM, CODE, SYS_CLOCK, CONTEXT_REGISTER, REGISTER_BANK, WAI_TSK
- **[FR]** ALM_HDR_NUMBER, EXC_HDR_ENTRY
The memory definition dialog defines the following information items concerning memory:

- System stack size (common)
- Kernel code address (common)
- Kernel data address (common)

Figures 7.3a, 7.3b, and 7.3c show examples of the memory definition dialogs.
7.3 Memory Definition Dialog

Figure 7.3b  Memory Definition Dialog (16)

Figure 7.3c  Memory Definition Dialog (8L)
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

- Setting item [common]

  [System Stack Size] is specified with a numeric value.

  The specification of [System stack] may be omitted. The default size is 0x40 for the 8L, 0x40
  for the 16, and 0x400 for the FR.

  [Kernel Code Address] is specified with a numeric value.

  [Kernel Data Address] is specified with a numeric value.

  [OS Data Size] indicates the data area used by the operating system.

  [Variable Memorypool Size] (for the FR) indicates the total size of the memory pool
  management area and memory pool.

  [Fixed Memorypool Size] (for the FR,16 only) indicates the total size: the block size
  multiplied by the number of blocks.

  [Task Stack Size] indicates the total size of the task stack and common stack.

  The following four information items are also calculated and displayed according to the
  settings:

  • OS data size
  • Variable memory pool size (for the FR)
  • Fixed memory pool size (for the FR,16 only)
  • Task stack size

  For details, see Appendix B, "Kernel Code and Kernel Data."

  For details of the definitions, see Sections 2.4.4, 2.5.4, and 2.6.4, "Memory Definitions."

**Note:** System stack size

The system stack is used by the kernel and interrupt handler. Define the system stack
size with a sufficient margin. If a system stack shortage occurs, the target system will not
operate normally.

### Definition Names Related to the Memory Definition Dialog

<table>
<thead>
<tr>
<th>Common</th>
<th>SYS_STK_SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>8L</td>
<td>KNL_ROM, KNL_RAM</td>
</tr>
<tr>
<td>16</td>
<td>KNL_ROM, KNL_RAM, MPL</td>
</tr>
<tr>
<td>FR</td>
<td>KNL_ROM, KNL_RAM, MPL</td>
</tr>
</tbody>
</table>
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.4 System Call Definition Dialog

The system call definition dialog is used to select system calls that are to be embedded in an absolute format object. By embedding only the necessary system calls, the size of the kernel code can be reduced.

- System Call Definition Dialog

Figure 7.4a shows an example of the system call definition dialog. Figure 7.4b shows an example of the task management function dialog.

Figure 7.4a Example of the System Call Definition Dialog (FR)

Specifies a desired system call names displayed by selecting system call types from the drop-down list box. Click system call names to select whether or not to embed these system calls in the object.

Figure 7.4b Example of the Task Management Function Definition Dialog (FR)

Clicking each button opens the system call dialog of the corresponding function. Click system call names to select whether or not to embed these system calls in the object.
7.4 System Call Definition Dialog

Note: ret_int in the interrupt management function dialog and ret_tmr of the time management function are always selected.

Note that in the evaluation version, regardless of the settings in this dialog, all system calls are always embedded in the object.

For details about definition, see Section 2.4.5, “System Call Definitions.”

Definition Name Related to the System Call Definition Dialog

[FR] SYSTEM_CALL
7.5 Task Definition Dialog

The task definition dialog defines the information items below.

- Task Definition Dialog

  Figures 7.5a, 7.5b, and 7.5c show examples of the task definition dialog.

![Figure 7.5a Example of the Task Definition Dialog (FR)]
7.5 Task Definition Dialog

Figure 7.5b  Example of the Task Definition Dialog (16)
To make a task information addition or modification, first click the respective button. When the task detail definition dialog opens, the dialog can be used to make a definition or modification.

If defines a common stack, clicking the [Common Stack] button of the dialog opens the common stack definition dialog.

If the task is created by C source, can use to aim at the almost stack size with the bellow list displaying automatically analyzed stack size. This displaying stack size is including task context area.

**Note:** Priority Level
Ensure that each tasks activation priority level specify a higher value than a number of setting on [Priority Level].

### Definition Names Related to the Task Definition Dialog

[Common] TSK, COM_STACK
7.5.1 Task Detail Definition Dialog

The task detail definition dialog defines the following task-related information items:

- Name (common)
- Entry (common)
- Stack (common)
- DTB (16)
- DPR (16)
- Initial state (FR)
- Extended information (FR)
- ID number (common)
- Activation priority (common)
- RP register (16/8L)
- ADB (16)
- CCR (16)
- Start code (FR)
- Time-out (FR)

### Task Detail Definition Dialog

Figures 7.5.1a, 7.5.1b, and 7.5.1c show examples of the task detail definition dialog.

![Task Detail Definition Dialog](image)

**Figure 7.5.1a Example of the Task Detail Definition Dialog (FR)**
Setting items [common]

[Name] specifies a character string for the task name.

[Id Number] () specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.

[Entry] specifies a character string for the label name.

[Task Priority] (activation priority) specifies a numeric value. If this item is omitted, the priority level specified in the system definition dialog (system_1 definition for 16) is used as the minimum priority, and that value is displayed in the list box.

[Stack] specifies a numeric value for the stack size, or specifies a common stack name with a character string or from the drop-down list box.

Setting items [8L]

[RP Register] specifies a numeric value. The default is 1.

For details, see Section 2.6.5, "Task Definition."
Setting items [16]

[RP Register] specifies a numeric value. The default is 0.

[DTB] specifies a numeric value. The default is 0.

[ADB] specifies a numeric value. The default is 0.

[DPR] specifies a numeric value. The default is 0.

[CCR] specifies a numeric value. The default is 0x40. (Set the I flag to 1, and the S flag to 0).

For details, see Section 2.5.5, "Task Definition."

Setting items [FR]

[Task Status] (initial state) specifies a task state at system activation.

[Start Code] specifies a numeric value. The default is 0.

[Ex-Info] (extended information) specifies a numeric value or a character string that serves as a label name. The default is 0.

[Time Out] specifies whether to use a system call with a time-out function.

For details, see Section 2.4.6, "Task Definition."

Note: Task stack size

Define a task stack size with a sufficient margin. If an insufficient stack size is specified, the target system may malfunction.

Definition Names Related to the Task Detail Definition Dialog

[Common] TSK, COM_STACK
7.5 Task Definition Dialog

7.5.2 Common Stack Definition

The common stack definition dialog defines a common stack. The size of memory can be suppressed by sharing the common stack among tasks without using the common stack at the same time.

Common Stack Definition

Figure 7.5.2 shows an example of the common stack definition dialog.

Figure 7.5.2   Example of the Common Stack Definition Dialog (Common)

Setting items [common]

Clicking the [Common stack] button of the task definition dialog opens the common stack definition dialog.

[Name] specifies a character string for the label name.

[Size] specifies a numeric value for the stack size.

For details of definition, see Section 2.4.7, “Common Stack Definition,” Section 2.5.6, “Common Stack Definition,” and Section 2.6.6, “Common Stack Definition.”

Note: Common stack

If multiple tasks sharing the same common stack are placed in a state (such as the ready and wait states) other than the dormant state, the target system malfunctions. Ensure that tasks operating simultaneously use a separate common stack or do not use a common stack.

Definition Names Related to the Common Stack Definition Dialog

[Common] TSK, COM_STACK
Semaphore Definition Dialog

The semaphore definition dialog defines the following semaphore information items used in user programs:

- Name (common)
- ID number (common)
- Initial count (common)
- Maximum count value (FR)
- Extended information (FR)

Figures 7.6a, 7.6b, and 7.6c show examples of the semaphore definition dialog.
7.6 Semaphore Definition Dialog

Figure 7.6b  Example of the Semaphore Definition Dialog (16)

Figure 7.6c  Example of the Semaphore Definition Dialog (8L)
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

- Setting items [common]
  
  [Name] specifies a character string for the semaphore name.

  [Id Number] specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.

  [Initial Count] specifies a numeric value for the system activation count value. The default is 1.

- Setting items [FR]
  
  [Max Count] (maximum count value) specifies a numeric value. The default is 32767.

  [Ex-Info] (extended information) specifies a numeric value or a character string that serves as a label name. The default is 0.

For details of definition, see Section 2.4.8, “Semaphore Definition,” Section 2.5.7, “Semaphore Definition,” and Section 2.6.7, “Semaphore Definition.”

- Definition Names Related to the Semaphore Definition Dialog

  [Common] SEM
7.7 EventFlag Definition Dialog

The eventflag definition dialog defines the following eventflag information items used in user programs:

- Name (common)
- ID number (common)
- Initial pattern (16, FR)
- Extended information (FR)

EventFlag Definition Dialog

Figures 7.7a, 7.7b, and 7.7c show examples of the eventflag definition dialog.

![EventFlag Definition Dialog Example](image)

Figure 7.7a Example of the EventFlag Definition Dialog (FR)
7.7 EventFlag Definition Dialog

**Figure 7.7b**  Example of the EventFlag Definition Dialog (16)

**Figure 7.7c**  Example of the EventFlag Definition Dialog (8L)
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

- Setting items [common]
  [Name] specifies a character string for the event flag name.
  [Id Number] specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.

- Setting items [16]
  [Initial Pattern] specifies a numeric value. The default is 0.
  [Eventflag Type Definition] (bit length specification) specifies whether to use the 1-bit event flag function or the 16-bit event flag function.

- Setting items [FR]
  [Initial Pattern] specifies a numeric value. The default is 0.
  [Ex-Info] specifies a numeric value or a character string that serves as a label name. The default is 0.

For details of definition, see Section 2.4.9, “EventFlag Definition,” Section 2.5.8, “EventFlag Definition,” and Section 2.6.8, “EventFlag Definition.”

Definition Names Related to the EventFlag Definition Dialog

[Common] FLG
CHAPTER 7 DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.8 Mailbox Definition Dialog

The mailbox definition dialog defines the following mailbox information items used in user programs:

- Name (common)
- ID number (common)
- Extended information (FR)

Mailbox Definition Dialog

Figures 7.8a, 7.8b, and 7.8c show examples of the mailbox definition dialog.

Figure 7.8a  Example of the Mailbox Definition Dialog (FR)
7.8 Mailbox Definition Dialog

Figure 7.8b  Example of the Mailbox Definition Dialog (16)

Figure 7.8c  Example of the Mailbox Definition Dialog (8L)
CHAPTER 7 DEFINITION DIALOOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

- Setting items [common]
  - [Name] specifies a character string for the mailbox name.
  - [Id Number] specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.

- Setting items [FR]
  - [Ex-Info] (extended information) specifies a numeric value or character string that serves as a label name. The default is 0.

For details of definition, see Section 2.4.10, “Mailbox Definition,” Section 2.5.9, “Mailbox Definition,” and Section 2.6.9, “Mailbox Definition.”

Definition Names Related to the Mailbox Definition Dialog

- [Common] MBX
7.9 Variable-Size MemoryPool Definition Dialog

The variable-size memorypool definition dialog defines the following variable-size memorypool information items used in user programs:

- Name (FR)
- Memorypool size (FR)
- ID number (FR)
- Block size (FR)
- Extended information (FR)

Figure 7.9 shows an example of the variable-size memorypool definition dialog.

- Setting items [FR]
  
  [Name] specifies a character string for the variable-size memory pool name.
  
  [Memorypool Size] specifies a numeric value.
  
  [Block Size] specifies a numeric value.
  
  [Id Number] specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.
  
  [Ex-Info] (extended information) specifies a numeric value or character string that serves as a label name. The default is 0.

For details of definition, see Section 2.4.11, “Variable-Size MemoryPool Definition.”

Definition Names Related to the Variable-Size MemoryPool Definition Dialog

[FR] MPL
CHAPTER 7  DEFINITION DIALOGS IN THE WORKBENCH VERSION OF THE CONFIGURATOR

7.10 Fixed-Size MemoryPool Definition Dialog

The fixed-size memorypool definition dialog defines the following fixed-size memorypool information items used in user programs:

- Name (FR, 16)
- Memorypool size (FR, 16)
- ID number (FR, 16)
- Number of blocks (FR, 16)

Fixed-Size MemoryPool Definition Dialog

Figures 7.10a and 7.10b show examples of the fixed-size memorypool definition dialog.

![Figure 7.10a](image_url)

Figure 7.10a  Example of the Fixed-Size MemoryPool Definition Dialog (FR)
7.10 Fixed-Size MemoryPool Definition Dialog

- Setting items [FR]
  - [Name] specifies a character string for the fixed-size memory pool name.
  - [Id Number] specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.
  - [Block Count] (number of blocks) specifies a numeric value.
  - [Block Size] specifies a numeric value.
  - [Ex-Info] (extended information) specifies a numeric value or character string that serves as a label name. The default is 0.

- Setting items (16)
  - [name] specifies a character string for the fixed-size memory pool name.
  - [Id Number] (ID number) specifies a numeric value. If this item is omitted, an available number is assigned, and AUTO is displayed in the list box.
  - [Block Size] specifies a numeric value.
  - [Block Count] (number of blocks) specifies a numeric value.

For details of definition, see Section 2.4.12, “MemoryPool Definition,” and Section 2.5.10, “MemoryPool Definition.”

- Definition Names Related to the Variable-Size MemoryPool Definition Dialog

  [16] MPL
  [FR] MPL
The vector definition dialog defines the following interrupt handler and vector information items used in user programs:

- Reset vector name (common)
- Initialization handler (16, 8L)
- Call vector setting (8L)
- Vector number (common)
- Entry (common)

The system clock timer handler of the kernel is also defined using this dialog.

Figures 7.11a, 7.11b, and 7.11c show examples of the vector definition dialog.
7.11 Vector Definition Dialog

Figure 7.11b  Example of the Vector Definition Dialog (16)

Figure 7.11c  Example of the Vector Definition Dialog (8L)
Setting items [common]

[Number] specifies a numeric value for the vector number.
[Entry] specifies a character string for the label name.
[Reset Vector Entry] specifies a character string for the reset vector entry name.

Setting items [8L]

[Initialization Handler Entry] specifies a character string for the initialization handler entry name.
[Definition Name] specifies a desired number and entry name displayed by selecting call vector definition or interrupt handler definition from the drop-down list box.

The list box displays vector numbers and their corresponding entry names. When a line is selected in the list box, the contents of the line are reflected as number and entry data in the edit box.

Setting items [16]

[Initialization Handler Entry] specifies a character string for the initialization handler entry name.

For details of definition, see Section 2.4.13, “Vector Definition,” Section 2.5.11, “Vector Definition,” and Section 2.6.10, “Vector Definition.”

Note: Interrupt handler (FR only)
Interrupt handlers need to be defined for all interrupts generated on the target system. If an interrupt is generated, and no interrupt handler is defined for the interrupt, the user-created system shutdown processing may be executed, and the target system may not continue processing normally.

Definition Names Related to the Vector Definition Dialog

- [Common] EIT_ENTRY, RESET_ENTRY
- [8L] CALL_ENTRY, INIT_HDR_ENTRY
- [16] INIT_HDR_ENTRY
7.12 CPU Setup Dialog

The CPU setup dialog defines the following CPU information items used in user programs:

- Chip mode (16, 8L)
- Bus mode (16)
- Multiplex bus (16)
- Built-in RAM (16)

The CPU setup dialog defines the following CPU information items used in user programs:

- Chip mode (16, 8L)
- Bus mode (16)
- Multiplex bus (16)
- Built-in RAM (16)

CPU Setup Dialog

Figures 7.12a and 7.12b show examples of the CPU setup dialog.

![CPU Setup Dialog](image)

Figure 7.12a Example of the CPU Setup Dialog (16)
[CPU Mode Setting] is used to make a desired selection for each of [Chip Mode], [Bus Mode], [Multiplex Bus], and [Inner RAM].

Depending on the CPU, some items may not be selectable, in which case those items are dimmed.

### Definition Names Related to the CPU Setup Dialog

- [8L] CPU
- [16] CPU
The Debug Seeing Dialog defines the following debug modules information items used in user programs:

- Kernel debug type (FR,16)
- Debug module (FR,16)
- Trace buffer size (FR,16)

Figures 7.13a and 7.13b show examples of the debug setting dialog.
7.13 Debug Setting Dialog

Figure 7.13b  Example of the Debug Setting Dialog (16)

- Setting items [FR, 16]

  [Kernel Debug Type] specifies a kernel debugging types of depending on debugging software.

  - Standard
    Use the empty debugging hook module.

  - Break Function
    Task dispatching break and system call break functions are allowed.

  - Break & Trace (FR)
    Task dispatching break, system call break, and task trace without time stamp functions are allowed.

  - REALOS Analyzer
    Ensure to select this If use the REALOS Analyzer.

  - Other
    If use the other debugging software excepting Softune Workbench and REALOS Analyzer. In this case, select debugging module files by [Add] button.

  [Debug Module] specifies debugging module files that displays default file names of the place of installed Softune Workbench or REALOS. Click file names to select whether or not to embed these files int the system. If use the other placed files or another debugging software, add files and delete default displaying debugging module files.

  For details of the debugging module files, see "REALOS Analyzer Manual"
[Trace Buffer Size] specifies trace buffer size that is defined in assemble file as the symbol of “TRC_DATA_NUM”.

**Note:** If don’t embed debugging module files in the system, select “Standard” of [Kernel Debug Type], or select “Other” and delete all selected files that entirely take debugging codes off.
This chapter describes the limitations on the workbench version of the configurator.

8.1 Maximum Number of Definable Objects (FR)
CHAPTER 8  LIMITATIONS ON THE WORKBENCH VERSION OF THE CONFIGURATOR

8.1 Maximum Number of Definable Objects (FR)

With FR, the number of objects that can be defined with the Workbench version of the configurator is limited.

- Maximum Number of Definable Objects (FR)
  
  The maximum number of definable entries of each type of object (task, semaphore, event flag, mailbox, and memory pool) is 1024. (More than 1024 entries can be set using the terminal version of the configurator.)
The appendixes contain notes on using the Softune REALOS configurator, information on kernel data allocation, configurator error messages, and a dialog list.

Appendix A  Notes on Using the Configurator
Appendix B  Kernel Code and Kernel Data Allocation
Appendix C  Configurator Error Messages
Appendix D  Dialog List
APPENDIX

Appendix A  Notes on Using the Configurator

This appendix contains notes on using the configurator.

- **Notes on Using the Embedded Version**
  
  If the configurator file is modified, a kernel object file needs to be generated by specifying the -a option again.

- **Notes on Using the Evaluation Version**
  
  The definition statement SYSTEM_CALL cannot be used. If the definition statement is used, a warning message is displayed.

  **Note**: With the evaluation version, a kernel is generated so that all system calls can be used.

- **Notes on Using the Configurator on Windows 95 or Windows NT**
  
  The following statement is used to activate the configurator for the MS-DOS prompt on Windows 95 or Windows NT:

  ```
  ```

  When the configurator is used on the MS-DOS prompt screen, all options described in Section 4.1, “Configurator Activation Commands,” can be specified.

  Since the internal process activates the assembler and linker when the configurator is used on the MS-DOS prompt screen, define in the PATH environment variable the directory that contains the tools before using the configurator.
Appendix B Kernel Code and Kernel Data Allocation

This appendix describes kernel codes and kernel data, and also describes the method of calculating the sizes of memory areas defined in KNL_ROM and KNL_RAM.

Kernel Code and Kernel Data

Usually, the kernel code section and kernel data section are automatically allocated in contiguous areas by specifying the KNL_ROM or KNL_RAM definition statement.

The tables below list the kernel code sections and kernel data sections.

**Table B.a FR Family Kernel Code Section and Kernel Data Section**

<table>
<thead>
<tr>
<th>Kernel code section</th>
<th>inidata, oscode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel data section</td>
<td>knldatal, sstack, knldata2, R_stk*, mplmem, mplctl, mpfmem</td>
</tr>
</tbody>
</table>

**Table B.b F2MC-16 Family Kernel Code Section and Kernel Data Section**

<table>
<thead>
<tr>
<th>Kernel code section</th>
<th>R97_TBL, T97_ILV, R97_CODE*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel data section</td>
<td>R97_SSTK, T97_DATA, R97_STK*, R97_POOL*</td>
</tr>
</tbody>
</table>

**Table B.c F2MC-8L Family Kernel Code Section and Kernel Data Section**

<table>
<thead>
<tr>
<th>Kernel code section</th>
<th>R97_CODE, R97_ITB_SEG, R97_ITB_SEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel data section</td>
<td>R97_SSTK, R97_<em>_DATA, R97_STK</em></td>
</tr>
</tbody>
</table>

**Note:** When KNL_ROM is not specified, all kernel code sections need to be allocated with USR SECTION.

When KNL_RAM is not specified, all kernel data sections need to be allocated with USR SECTION.
APPENDIX

Kernel code allocation

In the memory area defined with the KNL_ROM definition statement, the configurator allocate code and initialized data items required for REALOS operation.

Kernel code allocation on each CPU when a KNL_ROM definition is made is indicated below.

FR kernel code allocation

```
0x00000000
    KNL_ROM
    inidata  Kernel initialization data(*1)
    oscode   Operating system code size(*2)
0xFFFFFFFF
```

*1: Kernel initialization data estimation

\[
\text{inidata} = 40 + 28 \times \text{(Number of tasks)} + 16 \times \text{(Number of semaphores)} + 16 \times \text{(Number of eventflags)} + 12 \times \text{(Number of mailboxes)} + 28 \times \text{(number of variable-size memorypools)} + 24 \times \text{(number of fixed-size memorypools)} + 8 \times \text{(Number of alarm handlers)} + 8 \times \text{(Number of cyclic activation handlers)}
\]

*2: The operating system code size is about 3 K bytes to 9 K bytes.

FMC-16 kernel code allocation

```
0x000000
    KNL_ROM
    R97_TBL
    R97_ILV
    R97_CODE*
0xFFFFFFFF
```
F²MC-8L kernel code allocation

![Diagram showing kernel code allocation]

- KNL_ROM
- R97_CODE
- R97_ITB_SEG
- R97_INST_SEG
APPENDIX

Kernel code allocation

In the memory area defined with the definition statement KNL_RAM, the configurator allocates various data items required for REALOS operation.

Kernel data allocation in each CPU when a KNL_RAM definition is made is indicated below.

FR kernel data allocation

<table>
<thead>
<tr>
<th>Management area name</th>
<th>Memory size used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>108 + 8 x (priority level) + 44 x (number of tasks)</td>
</tr>
<tr>
<td>Semaphore</td>
<td>16 x (number of semaphores)</td>
</tr>
<tr>
<td>Eventflag</td>
<td>16 x (number of eventflags)</td>
</tr>
<tr>
<td>Mailbox</td>
<td>20 x (number of mailboxes)</td>
</tr>
<tr>
<td>Variable-size memorypool</td>
<td>20 x (number of variable-size memorypools)</td>
</tr>
<tr>
<td>Fixed-size memorypool</td>
<td>24 x (number of fixed-size memorypools)</td>
</tr>
<tr>
<td>Alarm handler</td>
<td>28 (number of alarm handlers)</td>
</tr>
<tr>
<td>Cyclic activation handler</td>
<td>28 x (number of cyclic activation handlers)</td>
</tr>
</tbody>
</table>

*1: Kernel management area (FR family)

Table B.d indicates the formulas for calculating kernel management areas for the FR family.

Table B.d  Formulas for Calculating Kernel Management Areas for the FR Family
(Unit: Bytes)

*2: Finding a system stack size

The system stack size depends heavily on the user system status (methods of using interrupts and programming interrupt handlers). The user should therefore set a system stack size that matches the user's system, ignoring the default.
Appendix B Kernel Code and Kernel Data Allocation

The system stack area is used for user interrupt handler processing, kernel system call processing, and system clock processing. The size used by each type of processing is as follows:

(a) Kernel system call processing: Approx. 100 bytes

(b) User interrupt handler processing: Size used by the user interrupt handlers

Note: Add (a) when a handler issues a system call.

(c) System clock processing: Approx. 80 bytes + Size used by the user timer handler

Note: Add (a) when a handler issues a system call.

However, the greatest value among (a), (b), and (c) should not be simply selected as a system stack size. Since these types of processing may be nested, consider this before determining a system stack size.

For example, suppose that a task issues a system call, and that an interrupt is generated while the kernel is executing system call processing. Further, suppose that the user interrupt handler is activated, then issues another system call. In such a case, the required size is (a) + (b).

Consequently, a system stack size can be found as follows:

System stack = (a) + ((b) x (maximum level of interrupt nesting)) + (c)

*3 Formula for calculating task stack sizes

The total of task stack sizes is sum of the stack sizes of tasks not using common stacks + sum of the sizes of common stacks defined. (Common stacks not in use are not included.)

*4 Variable-size memorypool area, fixed-size memorypool area

Total of variable-size memorypool sizes or fixed-size memorypool sizes

*5 Variable-size memorypool management area

<table>
<thead>
<tr>
<th>Memorypool name</th>
<th>Memorypool size</th>
<th>Block size</th>
<th>Number of blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpl_1</td>
<td>$M_1$</td>
<td>$m_1$</td>
<td>$k_1$</td>
</tr>
<tr>
<td>mpl_2</td>
<td>$M_2$</td>
<td>$m_2$</td>
<td>$k_2$</td>
</tr>
<tr>
<td>...</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>mpl_n</td>
<td>$M_n$</td>
<td>$m_n$</td>
<td>$k_n$</td>
</tr>
</tbody>
</table>

(However, $k_x = M_x \div m_x$)

When there are $n$ variable-size memorypools as indicated above, the size ($T$) of a variable-size memorypool management area is $T = ((k_1 + 1) + (k_2 + 1) + \ldots + (k_n + 1)) \times 4.$
Table B-e indicates the formulas for calculating kernel management areas for the F²MC-16 family.

### Table B.e  Formulas for Calculating Kernel Management Areas for the F²MC-16 Family

(Unit: Bytes)

<table>
<thead>
<tr>
<th>Management area name</th>
<th>Memory size used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>$4 + 12 + \text{(number of tasks)} + 4 \times \text{(priority level)}$</td>
</tr>
<tr>
<td>Evenflag(*1)</td>
<td>1 bit [\begin{align*}((4 + 1) \text{ bits}) \times \text{(number of eventflags)}\end{align*}]</td>
</tr>
<tr>
<td></td>
<td>16 bits [6 \times \text{(number of eventflags)}]</td>
</tr>
<tr>
<td>Semaphore</td>
<td>$5 \times \text{(number of semaphores)}$</td>
</tr>
<tr>
<td>Mailbox</td>
<td>$6 \times \text{(number of mailboxes)}$</td>
</tr>
<tr>
<td>Variable-size memroypool</td>
<td>$8 \times \text{(number of memory pools)}$</td>
</tr>
<tr>
<td>When wai_tsk or 16-bit eventflag is incorporated</td>
<td>$2 \times \text{(number of tasks)}$</td>
</tr>
<tr>
<td>Time management</td>
<td>Common [2 \times \text{(when system clock or timer handler is incorporated)}]</td>
</tr>
<tr>
<td></td>
<td>System clock [6 \times \text{(when system clock is incorporated)}]</td>
</tr>
<tr>
<td></td>
<td>Timer(*2) [4]</td>
</tr>
<tr>
<td></td>
<td>Cyclic activation handler [((12 + 1) \text{ bits}) \times \text{(number of handlers)}]</td>
</tr>
<tr>
<td></td>
<td>Alarm handler [2 + 14 \times \text{(number of handlers)}]</td>
</tr>
</tbody>
</table>

*1: Select 1 bit or 16 bits according to the eventflag management method.

*2: To be added when a cyclic activation handler or wai_tsk is incorporated

**Note:** If the number of bits is less than a byte, the value is rounded upward to a byte. When the number of objects of each type is 0, or when the number of handlers is 0, 0 bytes should be used, regardless of the formula.
Table B-1 indicates the formulas for calculating kernel management areas for the F²MC-8L family.

Table B.1  Formulas for Calculating Kernel Management Areas for the F²MC-8L Family

(1) Management area name  Memory size used

<table>
<thead>
<tr>
<th>Management area name</th>
<th>Memory size used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>14 + (9 to 11)(*1) x (number of tasks) + 4 x (priority level)</td>
</tr>
<tr>
<td>Evenflag</td>
<td>(4 + 1) bits) x (number of eventflags)</td>
</tr>
<tr>
<td>Semaphore</td>
<td>5 x (number of semaphores)</td>
</tr>
<tr>
<td>Mailbox</td>
<td>4 x (number of mailboxes)</td>
</tr>
<tr>
<td>System clock</td>
<td>4 (when system clock is incorporated)</td>
</tr>
<tr>
<td>Cyclic activation handler</td>
<td>4 + ((8 + 1) bits) x (number of cyclic activation handlers)</td>
</tr>
<tr>
<td>When wai_tsk is incorporated</td>
<td>4 (0 bytes when cyclic activation handler is incorporated)</td>
</tr>
</tbody>
</table>

*1: Usually, 9 bytes are used. When wai_tsk is incorporated, 11 bytes are used.

Note: If the number of bits is less than a byte, the value is rounded upward to a byte. When the number of objects of each type is 0, or when the number of handlers is 0, 0 bytes should be used, regardless of the formula.
Examples of Memory Size Calculations

- Example of memory size calculation (FR family)

Table B.9 shows an example of kernel and task memory size calculation for the FR family.

**Table B.9 Kernel and Task Memory Size Calculation (FR Family)**

<table>
<thead>
<tr>
<th>Example of kernel memory size calculation</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting condition</td>
<td></td>
</tr>
<tr>
<td>Fixed value</td>
<td>204 bytes</td>
</tr>
<tr>
<td>Priority level</td>
<td>8 x 8 64 bytes</td>
</tr>
<tr>
<td>Number of tasks</td>
<td>16 pcs. x 44 704 bytes</td>
</tr>
<tr>
<td>Number of eventflags</td>
<td>8 pcs. x 16 128 bytes</td>
</tr>
<tr>
<td>Number of mailboxes</td>
<td>2 pcs. x 20 40 bytes</td>
</tr>
<tr>
<td>Number of cyclic activation handlers</td>
<td>3 pcs. x 28 84 bytes</td>
</tr>
<tr>
<td>When system clock is used</td>
<td></td>
</tr>
<tr>
<td>When system call is issued within handler</td>
<td></td>
</tr>
<tr>
<td>System stack</td>
<td>100 bytes</td>
</tr>
<tr>
<td>Kernel system call processing (fixed value)</td>
<td>380 bytes</td>
</tr>
<tr>
<td>Size used by user interrupt handlers</td>
<td>100 bytes</td>
</tr>
<tr>
<td>When system call is issued within handler (fixed value)</td>
<td>100 bytes</td>
</tr>
<tr>
<td>When system clock processing is incorporated into system</td>
<td>80 bytes</td>
</tr>
<tr>
<td>System stack = 100 + 100 + 100 + 80</td>
<td>3136 bytes</td>
</tr>
<tr>
<td>User stack</td>
<td></td>
</tr>
<tr>
<td>Operating system area (fixed)</td>
<td>96 bytes</td>
</tr>
<tr>
<td>Stack used by user</td>
<td>100 bytes</td>
</tr>
<tr>
<td>User stack = (96 + 100) x 16 (number of tasks)</td>
<td>4740 bytes</td>
</tr>
<tr>
<td>Total</td>
<td>4740 bytes</td>
</tr>
</tbody>
</table>
## Example of memory size calculation (F²MC-16 family)

Table B.h shows an example of kernel and task memory size calculation for the F²MC-16 family.

### Table B.h  Kernel and Task Memory Size Calculation (F²MC-16 Family)

<table>
<thead>
<tr>
<th>Example of kernel memory size calculation</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting condition</td>
<td></td>
</tr>
<tr>
<td>Number of cyclic activation handlers</td>
<td>4 each</td>
</tr>
<tr>
<td>Priority level</td>
<td>6 levels</td>
</tr>
<tr>
<td>System clock</td>
<td></td>
</tr>
<tr>
<td>Number of tasks</td>
<td>16</td>
</tr>
<tr>
<td>Number of eventflags</td>
<td>8</td>
</tr>
<tr>
<td>Number of mailboxes</td>
<td>2</td>
</tr>
<tr>
<td>System stack</td>
<td></td>
</tr>
<tr>
<td>Suppose that an area of 14 bytes is used for one multilevel interrupt, and that a system call can be issued from the area. Then, 14 bytes are required. An additional area is required when a user stack is used within a handler. Stack used within a handler: 8 bytes</td>
<td></td>
</tr>
<tr>
<td>System stack area = 14 used when system calls are issued) x 4 levels + (14 used within stack handler at handler activation) x 4 levels</td>
<td>144 bytes</td>
</tr>
<tr>
<td>Register bank area</td>
<td></td>
</tr>
<tr>
<td>One bank is used for the kernel. One bank is used for the handler. One bank is used for each task.</td>
<td></td>
</tr>
<tr>
<td>Register bank area = (1 kernel + 1 handler + number of tasks) x 16</td>
<td>288 bytes</td>
</tr>
<tr>
<td>User stack area</td>
<td></td>
</tr>
<tr>
<td>Operating system control area</td>
<td>12</td>
</tr>
<tr>
<td>User stack</td>
<td>32</td>
</tr>
<tr>
<td>(Operating system control area + user stack) x 16 stacks</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
Table B.i  Kernel and Task Memory Size Calculation (F²MC-8L Family)

<table>
<thead>
<tr>
<th>Example of kernel memory size calculation</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting condition</td>
<td>Subtotal</td>
</tr>
<tr>
<td>Number of cyclic activation handlers</td>
<td>4</td>
</tr>
<tr>
<td>Number of eventflags</td>
<td>3</td>
</tr>
<tr>
<td>Number of mailboxes</td>
<td>3</td>
</tr>
<tr>
<td>Number of cyclic activation handlers</td>
<td>3</td>
</tr>
<tr>
<td>Priority level</td>
<td>4 levels</td>
</tr>
<tr>
<td>System stack</td>
<td></td>
</tr>
</tbody>
</table>
| Suppose that an area of 12 bytes (PC, PS, A, T, IX, and EP) is used for one multilevel interrupt, and that a system call can be issued from the areas. Then, 14 bytes are required. Because there are two levels, a system stack area of 52 bytes is required in total. An additional area is required when a user stack is used within a handler. Stack used within a handler: 8 bytes
| Interrupt: 2 levels
| System stack area = ((handler activation stack (12) + stack used for system calls (14)) x 2 levels + stack used within handler (8)) | 60 bytes |
| Register bank area                      |          |
| One bank is used for kernel. One bank is used for handler. One bank is used for each task (4 banks). Register bank area = (1 kernel + 1 handler + number of tasks) x 8 | 48 bytes |
| User stack area                         |          |
| User stack                              | 15       |
| When multilevel interrupts are accepted (One task must not be interrupted.) (User stack + interrupt stack) x 3 tasks + user stack | 240 bytes |
| Total                                   |          |
|                                         | 480 bytes |
Task Stack

[Task stack session]
The configurator creates task stack sessions in the temporary file for each task ID using the following rules:

[FR]
- Task stack section name: R_stkXXXX
- Symbol name: R_XXXXstk
  XXXX indicates a four-digit hexadecimal character string.

[16, 8L]
- Task stack section name: R97_STKXX
- Symbol name: r97_XXstk
  XX indicates a two-digit hexadecimal character string.

These names can be used to initialize task stack regions. Task stack sections are output to R_wdata.asm for the FR and to R97_data.asm for the FMC_16.

[Example of output for the FR]
The following is output when the task ID is decimal 10 and the size of stack region is 0x400:

```assembly
DATA_SECTION R_stk000A,4
.export R_000Astk
.res.b 0x400
R_000Astk
SECTION_END
```

R_wdata.asm
APPENDIX

(Task stack size)
The task stack size is defined using rules similar to those for the task stack section name.

[FR]
- Task stack size: R_TSKSKXXXX
  XXXX indicates a four-digit hexadecimal character string.

[16, 8L]
- Task stack size: R97_STKXX
  XX indicates a two-hexadecimal character string.

Task stack sections are output to “R_id.asm” for the FR and “R97_id.asm” for the F^2MC_16.

[Example of output for the FR]
The following is output when the task ID is 1 and the size of stack region is 0x500:

```
R_id.asm

   ...
   ...
   .export R_TSKSK0001
R_TSKSK0001 .equ 0x500

   ...
```

Appendix C Configurator Error Messages

This appendix describes the levels and display format of messages that are output during configuration.

### Configurator Error Message Levels

The error messages output by the configurator when configuration is executed are classified into three levels on the basis of severity.

- **Warning**
  
  This type of message is less serious than an error, and the results of output can be used with almost no problem being caused.

  In some cases, however, processing not intended by the user may be performed.

  After checking a warning message, determine whether the results of output are usable.

- **Error**
  
  This type of message allows processing to be continued, but configuration is not executed. The cause of an error must be corrected, then configuration must be reexecuted.

  Usually, errors occur when a file is read.

- **Fatal error**
  
  This type of message indicates an error that prevents processing from continuing. A fatal error may be caused by an incorrect specification by the user or by execution environment problem.

In addition, error messages are output by the assembler and linker executed within the configurator. For these messages, see the *Softune Assembler Manual* and the *Softune Linkage Kit Manual*. 

## Display Format of Configurator Error Messages

Error messages are output in the following format.

```
*** file name (line number) XnnnT: message text (auxiliary-message)
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file-name (line-number)</td>
<td>Configuration file name and line number that encountered the error. This information is output if an error occurs when a configuration file is read.</td>
</tr>
<tr>
<td>X</td>
<td>An error level is indicated using one of the following three characters: W: Warning message E: Error F: Fatal error</td>
</tr>
<tr>
<td>nnnn</td>
<td>Error number. Error numbers and error levels are related as follows: 1000 to 1999: W 4000 to 4999: E 9000 to 9999: F</td>
</tr>
<tr>
<td>T</td>
<td>The tool ID is represented by the following letter: M: Configurator</td>
</tr>
<tr>
<td>message-text</td>
<td>Error message text (The user can choose between Japanese and English.)</td>
</tr>
<tr>
<td>auxiliary-message</td>
<td>More detailed information about the error. Information such as the symbol name that caused the error is displayed. This information may be included in the error message text.</td>
</tr>
</tbody>
</table>

**Note:** An error may occur when the assembler or linker has been activated by the configurator. In this case, a tool ID of A or L may be output.

For details of this type of error, see the *Softune Assembler Manual* and the *Softune Linkage Kit Manual.*
### Warning Message

| W1001M | Can not use "definition name" on evaluation version |

The definition name identified by `definition-name` cannot be used with the evaluation version. This error occurs only when the evaluation version is used.
APPENDIX
■ Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4024M</td>
<td>Illegal character <em>(parameter name)</em></td>
</tr>
<tr>
<td>E4026M</td>
<td>Specified value is out of range <em>(value)</em></td>
</tr>
<tr>
<td>E4100M</td>
<td>Nest level exceeds limit <em>(MAX nesting level)</em></td>
</tr>
<tr>
<td>E4110M</td>
<td>Unknown define name <em>(definition name)</em></td>
</tr>
</tbody>
</table>

The parameter identified by *parameter-name* includes an unusable character.
This error occurs when, for example, a nonnumeric character is specified in a parameter where a numeric value must be specified, or a numeric value is specified in a parameter where a label must be specified.

A value within the value range indicated by *value* is specifiable.
This error occurs when, for example, a value greater than 32767 is specified as an object ID.

The INCLUDE file nesting level exceeded the level indicated by *nesting-level*.

The definition name indicated by *definition-name* cannot be used.
This error occurs when, for example, an unsupported definition name is used.
Appendix C Configurator Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4111M</td>
<td>Too long line (MAX value char)</td>
</tr>
<tr>
<td>E4120M</td>
<td>parameter name is too long</td>
</tr>
<tr>
<td>E4121M</td>
<td>Too many definition name (MAX value)</td>
</tr>
<tr>
<td>E4123M</td>
<td>Too many Parameters (parameter)</td>
</tr>
</tbody>
</table>

A line must not be longer than the value indicated by `value`. Correct the line so that it is not longer than the value indicated by `value`.

The parameter identified by `parameter-name` is too long.

The number of definitions identified by `definition-name` must not exceed `value`. This error occurs when, for example, an attempt is made to define more than 32767 tasks.

The parameter identified by `parameter` and subsequent parameters are not required.
### APPENDIX

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4130M</td>
<td>Multiple definition (definition name)</td>
</tr>
</tbody>
</table>

A definition statement that must not be specified more than once was specified more than once.

This error occurs when, for example, a CPU definition is made more than once, or the same common stack name, or task, semaphore, eventflag, mailbox, or memorypool ID is specified more than once.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4131M</td>
<td>Parameter not defined (parameter name)</td>
</tr>
</tbody>
</table>

A parameter that must always be specified is was not specified.

The parameter identified by `parameter-name` must not be omitted.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4132M</td>
<td>Illegal parameter (parameter)</td>
</tr>
</tbody>
</table>

A parameter that cannot be specified was specified.

This error mainly occurs when, for example, a character string that cannot be selected is specified in a selection item.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4133M</td>
<td>Symbol is already defined (symbol name)</td>
</tr>
</tbody>
</table>

A symbol that is already defined was defined again.

This error occurs when, for example, an attempt is made to define a duplicate task, semaphore, eventflag, mailbox, or memorypool name.
Appendix C Configurator Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4136M</td>
<td>Illegal size or address (value)</td>
</tr>
<tr>
<td>E4140M</td>
<td>Isemcnt is bigger than maxcnt (MAX value)</td>
</tr>
<tr>
<td>E4141M</td>
<td>Blksiz is bigger than mplsiz (MAX size)</td>
</tr>
</tbody>
</table>

The size or address indicated by `value` is incorrect.
When specifying a size or address, specify a multiple of 4 (or, for the last address, a multiple of 4 less 1).

A value greater than the maximum allowable value was specified for the initial semaphore count.
Decrease the initial count, or increase the maximum allowable value.

A value greater than the memorypool size was specified for the memorypool block size.
Decrease the block size, or increase the memorypool size.
APPENDIX

 Fatal Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9000M</td>
<td>Environment variable not found (environment variable name)</td>
</tr>
</tbody>
</table>

The environmental variable identified by `environmental-variable-name` is not defined.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9001M</td>
<td>Insufficient memory</td>
</tr>
</tbody>
</table>

Memory for executing a program is insufficient.
The program may be made executable by the following:

- Decrease the number of tasks, the number of semaphores, the number of eventflags, the number of mailboxes, and the number of memory pools.
- Shorten task names, semaphore names, eventflag names, mailbox names, memorypool names, and task entry symbol names.
- Specify no task extension information, semaphore extension information, eventflag extension information, mailbox extension information, and memorypool extension information.

If memory is still insufficient, increase the amount of memory installed on the personal computer when using the personal computer version, or use the workstation version. When using the workstation version, contact the system administrator and increase the system memory space.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9002M</td>
<td>Not configurated</td>
</tr>
</tbody>
</table>

Configuration was not executed.
This error mainly occurs when configuration fails due to a configuration file read error.
## Appendix C Configurator Error Messages

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9003M</td>
<td>Kernel source file not found</td>
</tr>
<tr>
<td>F9011M</td>
<td>Input file is not found (file name)</td>
</tr>
<tr>
<td>F9016M</td>
<td>File read error (file name)</td>
</tr>
<tr>
<td>F9017M</td>
<td>File write error (file name)</td>
</tr>
</tbody>
</table>

The kernel source file is not stored under the predetermined directory ($RTOS\knl). This error occurs when the evaluation version is used and configuration is executed with specifying the `-a` option.

The specified input file cannot be found.

Read permission has not been granted for the file identified by `file-name`, or there is a hardware problem.

The disk where the file identified by `file-name` is to be written is full. Create a free space on the disk, then reexecute configuration. This error also occurs when there is a file or directory that has the same name and write permission has not been granted.
### APPENDIX

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9022M</td>
<td>Unknown option name (option)</td>
</tr>
<tr>
<td>F9023M</td>
<td>Illegal option parameter (parameter)</td>
</tr>
<tr>
<td>F9030M</td>
<td>Missing input file name</td>
</tr>
</tbody>
</table>

A nonexistent option was specified.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9801M</td>
<td>definition name is not defined</td>
</tr>
</tbody>
</table>

The definition identified by definition-name was not made.
This error occurs when a definition statement that must not be omitted is not specified.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9802M</td>
<td>definition name (name) is not defined</td>
</tr>
</tbody>
</table>

name is not defined in the definition identified by definition-name.
This error occurs mainly when a common stack is specified in a task definition without a task common stack being defined.
### Appendix C Configurator Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9803M</td>
<td>definition name (ID: number) is not defined</td>
</tr>
</tbody>
</table>

The definition data corresponding to definition-name(ID: number) is not defined.
This error occurs mainly when successive ID numbers are not used in task, semaphore, eventflag, mailbox, or memorypool definitions.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9804M</td>
<td>PRIORITY_LEVEL is higher than tsk_priority (more than priority level)</td>
</tr>
</tbody>
</table>

A priority lower than the priority level specified in the definition statement PRIORITY_LEVEL was specified as a task activation priority.
For a task activation priority, specify a priority not lower than the priority level specified in the definition statement PRIORITY_LEVEL.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9805M</td>
<td>EIT vector No. vector_number is system reserve</td>
</tr>
</tbody>
</table>

The EIT vector identified by vector-number is reserved for the system.
This error occurs when a CPU definition statement is omitted. This error also occurs when an MB911V100 or MB91103 is specified, and an EIT vector other than vector numbers 16 to 62 and 65 to 255 is defined.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9897M</td>
<td>Error in assembler (file name)</td>
</tr>
</tbody>
</table>

An error occurred with the assembler.
This error occurs mainly when a character that is not allowed is specified in a task entry or extended information.
### APPENDIX

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9898M</td>
<td>Error in linker</td>
</tr>
<tr>
<td>F9899M</td>
<td><code>tool name</code> is not found</td>
</tr>
<tr>
<td>F9990M</td>
<td>File I/O error (<code>file name</code>, <code>identification information</code>)</td>
</tr>
<tr>
<td>F9993M</td>
<td>Can not create directory (<code>directory name</code>)</td>
</tr>
</tbody>
</table>

An error occurred with the linker.
Correct the cause of the error according to the map file.

The directory where the assembler and linker are stored cannot be found using the PATH environmental variable.
Define in the PATH environmental variable the directory where the assembler and linker are stored.

An error occurred during file input-output.

The directory identified by `directory-name` cannot be created.
The disk may have no free space, or there may be a file or directory that has the same name and write permission has not been granted.
### Appendix C Configurator Error Messages

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F9994M</td>
<td>Can not create file (file name)</td>
</tr>
<tr>
<td>F9995M</td>
<td>Can not close file (file name)</td>
</tr>
<tr>
<td>F9996M</td>
<td>Can not open file (file name)</td>
</tr>
<tr>
<td>F9997M</td>
<td>Error message not found</td>
</tr>
</tbody>
</table>

The file identified by `file-name` cannot be created.
The disk may have no free space, or there may be a file or directory that has the same name and write permission has not been granted.

The file identified by `file-name` cannot be closed.
The disk may have no free space.

The file identified by `file-name` cannot be opened.
The disk may have no free space, or there may be a file or directory that has the same name and write permission has not been granted.

The message file (frcfs.msg) used by the configurator may have been destroyed. Reinstall Softune REALOS.
If this error still occurs after reinstallation, contact your Fujitsu representative immediately.
APPENDIX

<table>
<thead>
<tr>
<th>F9998M</th>
<th>Can not open message file <em>(file name)</em></th>
</tr>
</thead>
</table>

The message file used by the configurator cannot be opened.
Store frcf.s.msg under the directory *(RTOS\syslib)*.

<table>
<thead>
<tr>
<th>F9999M</th>
<th>Internal error <em>(identification information)</em></th>
</tr>
</thead>
</table>

When this error occurs, contact your Fujitsu representative immediately.
Appendix D Dialog List

This appendix describes the following dialogs output when a dangerous operation or incorrect operation is performed:

- Warning dialog
- Error dialog

### Warning Dialog

The warning dialog is output when setting data may be lost.

Figure D.a shows an example of warning dialog output.

![Example of the Warning Dialog](image)

**Figure D.a Example of the Warning Dialog**

W1100M: Target MCU is different.
New project: MB91101
Configuration file: MB91107
Target MCU set by a new project is made effective.

The target MCU differs from the MCU (CPU) set in the configuration file. Target MCU set by a new project becomes effective.

W1101M The relocatable object file and library file in the loaded configuration file are not registered. Register the members.

When an existing configuration file is read, the relocatable object file and library file set in the configuration file are not registered. Open a dialog by first selecting [Project] and then [Member Setting] in Softune Workbench, then setting the members.
APPENDIX

W1102M: Enter a file name.

When an existing configuration file is read, this message is output if you click the [OK] button without setting the file name in the file input edit box.
Set the configuration file.

W1103M: Section allocation has changed. Okay to proceed with initialization?

This message is output if, after a kernel code address or kernel data address has been set, another kernel code address or kernel data address is set. When you click the [OK] button, the new setting is reflected in Softune Workbench.

W1104M: Set a ROM or RAM area by setting the link option after selecting [Project] then [Tool Option Setting].

This message is output when you click [Complete] in the configurator setting dialog after creating a new configuration file.
From the tool option setting dialog of the project menu in Softune Workbench, select the linker tab. Then, select [Allocation or Linkage] from the category, and set a ROM or RAM area.
Error Dialog

The error dialog is output, for example, for an incorrect specification.

Figure D.b shows an example of error dialog output.

Figure D.b  Example of the Error Dialog

E4200M: The stack name is registered in the task definition.

When a stack name is set in the task definition dialog, this message is output if an attempt is made to delete or change the stack name in the common stack definition dialog.

E4201M: Execution is disabled because memory is insufficient. Close unnecessary Windows applications, then reexecute.

This message is output if a settings attempt is made when memory is insufficient. Close unnecessary Windows applications, then retry.

E4202M: A set MB number is invalid. The target MCU set in the new project is made effective.

This message is output when the MCU (CPU) set in the existing configuration file is not registered as a target MCU.
INDEX

A
absolute object name ............................................ 39
ABSOLUTE_FILE ................................................. 82
ALM_HDR_NUMBER ........................................... 85

B
button .................................................................. 180

C
CALL_ENTRY ..................................................... 145
CODE ................................................................. 96
common stack definition .................................... 26, 46, 64, 207
common stack definition dialog ........................... 207
COM_STACK...................................................... 120
configuration ...................................................... 6, 14
configuration data ............................................. 4, 13, 17, 37
configuration definition statement ......................... 70
configuration file ............................................... 172
configurator ..................................................... 228, 229
CPU_EXTHDR_ENTRY........................................ 87
CYC_HDR_NUMBER ........................................... 84

D
Debug Setting Dialog ........................................... 230
definition name related to the CPU setup dialog. 229
definition name related to the eventflag definition dialog 214
definition name related to the mailbox definition dialog 218
definition name related to the memory definition dialog ............................................. 218
definition name related to the semaphore definition dialog ........................................... 196
definition name related to the system call definition dialog ........................................... 199
definition name related to the system definition dialog ................................................. 192
definition name related to the task definition dialog ...................................................... 202
definition name related to the variable-size memorypool definition dialog ................. 220, 223
definition name related to the vector definition dialog ................................................... 226
definition names related to the common stack definition dialog .............................. 207
definition names related to the system definition dialog ............................................. 192
definition names related to the task detail definition dialog ........................................ 206
definition of configuration data ............................... 55
definition statement ............................................... 74
definition statement format .................................... 74
dialog ................................................................... 182

E
EIT_ENTRY ......................................................... 142
eventflag .......................................................... 48, 66
eventflag definition ............................................. 30, 48, 66, 129
eventflag definition dialog .................................... 212, 214
eventflag type ...................................................... 128
exception handler name ........................................ 87
EXC_HDR_ENTRY ............................................... 86

F
F2MC-16F dedicated code .................................... 96
file name .............................................................. 184
file selection ......................................................... 185
fixed-size memorypool definition ......................... 34, 52
fixed-size memorypool definition dialog ................ 222
FLG ..............................................................129, 130
FLG_BIT ................................................................... 128
FLG_PATTERN .....................................................129

I
ILM ......................................................................... 95
INCLUDE ............................................................... 94
INIT_HDR_ENTRY ............................................... 144
initial pattern ..................................................... 129
initialization handler ............................................. 144
interrupt level, allowable ........................................ 95

K
kernel ..................................................................... 95
KNL_RAM ............................................................ 107
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL_ROM</td>
<td>106</td>
</tr>
<tr>
<td>library name</td>
<td>39, 57</td>
</tr>
<tr>
<td>LIBRARY_FILE</td>
<td>81</td>
</tr>
<tr>
<td>linkage definition</td>
<td>19, 39, 57</td>
</tr>
<tr>
<td>List</td>
<td>186</td>
</tr>
<tr>
<td>mailbox definition</td>
<td>31, 50, 67</td>
</tr>
<tr>
<td>mailbox definition dialog</td>
<td>216, 218</td>
</tr>
<tr>
<td>map file output</td>
<td>39, 57</td>
</tr>
<tr>
<td>MAP_FILE</td>
<td>83</td>
</tr>
<tr>
<td>MBX</td>
<td>134</td>
</tr>
<tr>
<td>memory</td>
<td>42, 60</td>
</tr>
<tr>
<td>memory definition</td>
<td>21, 42, 60</td>
</tr>
<tr>
<td>memory definition dialog</td>
<td>194, 196</td>
</tr>
<tr>
<td>monitor debugger</td>
<td>152</td>
</tr>
<tr>
<td>MPF</td>
<td>140</td>
</tr>
<tr>
<td>MPL</td>
<td>136, 137</td>
</tr>
<tr>
<td>object</td>
<td>234</td>
</tr>
<tr>
<td>object file</td>
<td>16</td>
</tr>
<tr>
<td>PRIORITY_LEVEL</td>
<td>92</td>
</tr>
<tr>
<td>project</td>
<td>174</td>
</tr>
<tr>
<td>RAM</td>
<td>105</td>
</tr>
<tr>
<td>REGISTER_BANK</td>
<td>101</td>
</tr>
<tr>
<td>relocatable object name</td>
<td>39</td>
</tr>
<tr>
<td>RELOCATE_FILE</td>
<td>80</td>
</tr>
<tr>
<td>RESET_ENTRY</td>
<td>146</td>
</tr>
<tr>
<td>ROM</td>
<td>104</td>
</tr>
<tr>
<td>SCAL_EXC_ENTRY</td>
<td>90</td>
</tr>
<tr>
<td>SEM</td>
<td>124, 126</td>
</tr>
<tr>
<td>semaphore</td>
<td>47</td>
</tr>
<tr>
<td>semaphore definition</td>
<td>28, 47, 65</td>
</tr>
<tr>
<td>semaphore definition dialog</td>
<td>208, 210</td>
</tr>
<tr>
<td>SEM_ICNT</td>
<td>122</td>
</tr>
<tr>
<td>SYS_CLOCK</td>
<td>98</td>
</tr>
<tr>
<td>SYS_EXTHDR_ENTRY</td>
<td>88</td>
</tr>
<tr>
<td>SYS_STK_SIZE</td>
<td>103</td>
</tr>
<tr>
<td>system call definition</td>
<td>22</td>
</tr>
<tr>
<td>system call definition dialog</td>
<td>198, 199</td>
</tr>
<tr>
<td>system call exception</td>
<td>88</td>
</tr>
<tr>
<td>system definition</td>
<td>20, 40, 58</td>
</tr>
<tr>
<td>system definition dialog</td>
<td>188, 192</td>
</tr>
<tr>
<td>SYSTEM_CALL</td>
<td>110</td>
</tr>
<tr>
<td>task definition</td>
<td>24, 44, 62</td>
</tr>
<tr>
<td>task definition dialog</td>
<td>200, 202</td>
</tr>
<tr>
<td>task detail definition dialog</td>
<td>204, 206</td>
</tr>
<tr>
<td>TSK</td>
<td>112, 114, 117</td>
</tr>
<tr>
<td>USR_SECTION</td>
<td>108</td>
</tr>
<tr>
<td>variable-size memorypool definition</td>
<td>32</td>
</tr>
<tr>
<td>variable-size memorypool definition dialog</td>
<td>220, 223</td>
</tr>
<tr>
<td>vector definition</td>
<td>36, 54, 68</td>
</tr>
<tr>
<td>vector definition dialog</td>
<td>224, 226</td>
</tr>
<tr>
<td>wai_tsk</td>
<td>102</td>
</tr>
</tbody>
</table>