User’s Manual

CC78K0R Ver. 2.00

C Compiler

Operation

Target Device
78K0R Microcontrollers

Document No. U18549EJ1V0UM00 (1st edition)
Date Published October 2007

[MEMO]

[MEMO]

INTRODUCTION

The purpose of this manual is to enable complete understanding of the functions and operation of the CC78K0R (78K0R Microcontroller C Compiler).

This manual does not explain how to write CC78K0R source programs. Therefore, before reading this manual, please read **"CC78K0R Ver. 2.20 C Compiler Language User's Manual (U18548E)"** (hereafter called the "Language manual").

[Target Devices]

Software for 78K0R microcontrollers can be developed by using the CC78K0R. To use this software, the RA78K0R (78K0R Microcontroller Assembler Package) (sold separately) and the target model's device file are required.

[Target Readers]

This manual is written for users who have the knowledge gained from reading through the user's manual for the device once and have software programming experience. However, since knowledge about C compilers and the C language is not particularly needed, first-time users of C compilers can use this manual.

[Organization]

The organization of this manual is described below.

CHAPTER 1 OVERVIEW

This chapter describes the role and position of the CC78K0R in microcontroller development.

CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION

This chapter describes how to install the CC78K0R, the file names of the supplied programs, and the operating environment for programs.

CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING

This chapter uses sample programs to describe how to run the CC78K0R and presents examples showing the processes from compiling to linking.

CHAPTER 4 CC78K0R FUNCTIONS

This chapter describes optimization methods and ROMization functions in the CC78K0R.

CHAPTER 5 COMPILER OPTIONS

This chapter describes the functions of the compiler options, specification methods, and prioritization.

CHAPTER 6 C COMPILER OUTPUT FILES

This chapter describes the output of various list files output by the CC78K0R.

CHAPTER 7 USING C COMPILER

This chapter introduces techniques to aid in the skillful use of the CC78K0R.

CHAPTER 8 STARTUP ROUTINES

The CC78K0R provides startup routines as samples. This chapter describes the uses of the startup routines and provides suggestions on how to improve them.

CHAPTER 9 ERROR MESSAGES

This chapter describes the error messages output by the CC78K0R.

APPENDIXES

The appendices provide and a sample programs, a list of the use-related cautions, a command options, and an index.

[How to Read This Manual]

First, those who want to see how to actually use CC78K0R, read **CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING**.

Users with a general knowledge of C compilers or users who have read the Language manual can skip **CHAPTER 1 OVERVIEW**.

[Related Documents]

The table below shows the documents (such as user's manuals) related to this manual. The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Documents related to development tools (user's manuals)

Document Name		Document No.
CC78K0R Ver. 2.00 C Compiler	Operation	This document
	Language	U18548E
RA78K0R Ver. 1.20 Assembler Package	Operation	U18547E
	Language	U18546E
SM+ System Simulator	Operation	U18010E
PM+ Ver. 6.30 Project Manager		U18416E
ID78K0R-QB Ver. 3.20 Integrated Debugger	Operation	U17839E

[Conventions]

The meanings of the symbols used in this manual are explained.

RTOS:	Real-time OS for 78K0R Microcontroller RX78K0R
…:	Repeat in the same format.
[ ]:	Characters enclosed in these brackets can be omitted.
「 」:	Characters enclosed in these brackets are as shown (character string).
“ ”:	Characters enclosed in these brackets are as shown (character string).
‘ ’:	Characters enclosed in these brackets are as shown (character string).
Boldface:	Characters in bold face are as shown (character string).
_ :	Underlining at important locations or in examples is the input character sequence.
Δ :	At least one space
⋮ :	Indicates an omission in a program description
( ) :	Characters between parentheses are as shown (character string).
/ :	Delimiter
\\:	Backslash

[File Name Conventions]

The conventions for specifying the input files that are designated in the command line are shown below.

(1) Specifying disk file names

```
[drive-name]  [\]  [[path-name]...]  primary-name  [.[file-type]]
    <1>        <2>  <3>               <4>            <5>
```

<1> Specifies the name of the drive (A: to Z:) storing the file.

<2> Specifies the name of the root directory.

<3> Specify the subdirectory name.

Specify a character string of a length allowed by the OS.

Characters that can be used:

All the characters allowed by the OS, except parentheses (()), semicolons (:), and commas (,).

Note that a hyphen (-) cannot be used as the first character of a path name.

<4> Primary name

Specify a character string of a length allowed by the OS.

Characters that can be used:

All the characters allowed by the OS, except parentheses (( )), semicolons (:), and commas (,).

Note that a hyphen (-) cannot be used as the first character of a path name.

<5> File type

Specify a character string of a length allowed by the OS.

Characters that can be used:

All the characters allowed by the OS, except parentheses (( )), semicolons (:), and commas (,).

Example: `C:\Program Files\NEC Electronics Tools\CC78K0R\V1.00\smp78k0r\cc78k0r`

Remarks **1.** A space cannot be specified before and after ‘:’, ‘.’, or ‘\’.

2. Uppercase and lowercase letters are not distinguished (not case-sensitive).

(2) Specifying device file names

The following logical devices are available.

Logical Device	Description
CON	Output to the console.
PRN	Output to the printer.
AUX	Output to an auxiliary output device.
NUL	Dummy output (nothing is output.)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table No.   Title , Page

CHAPTER 1   OVERVIEW

The 78K0R C compiler CC78K0R translates C source programs written in ANSI-C[Note] or the C language for the 78K0R into the machine language for the 78K0R.

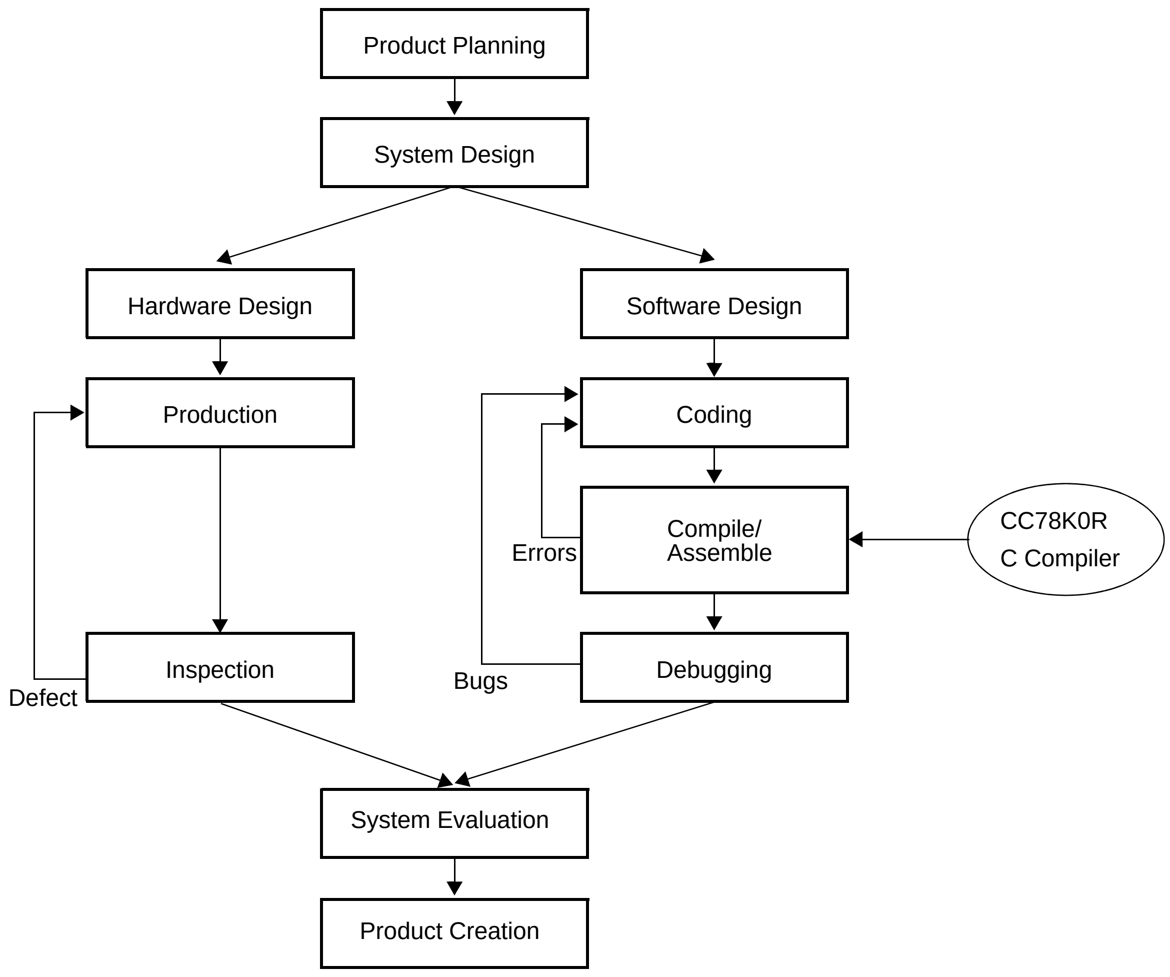
The CC78K0R can be started up on Windows® by using PM+ supplied with the assembler package for the 78K0R. When PM+ is not used, the compiler is started up on the command prompt.

Note ANSI-C is the C language that conforms to the standard set by the American National Standards Institute.

1.1 Role of CC78K0R

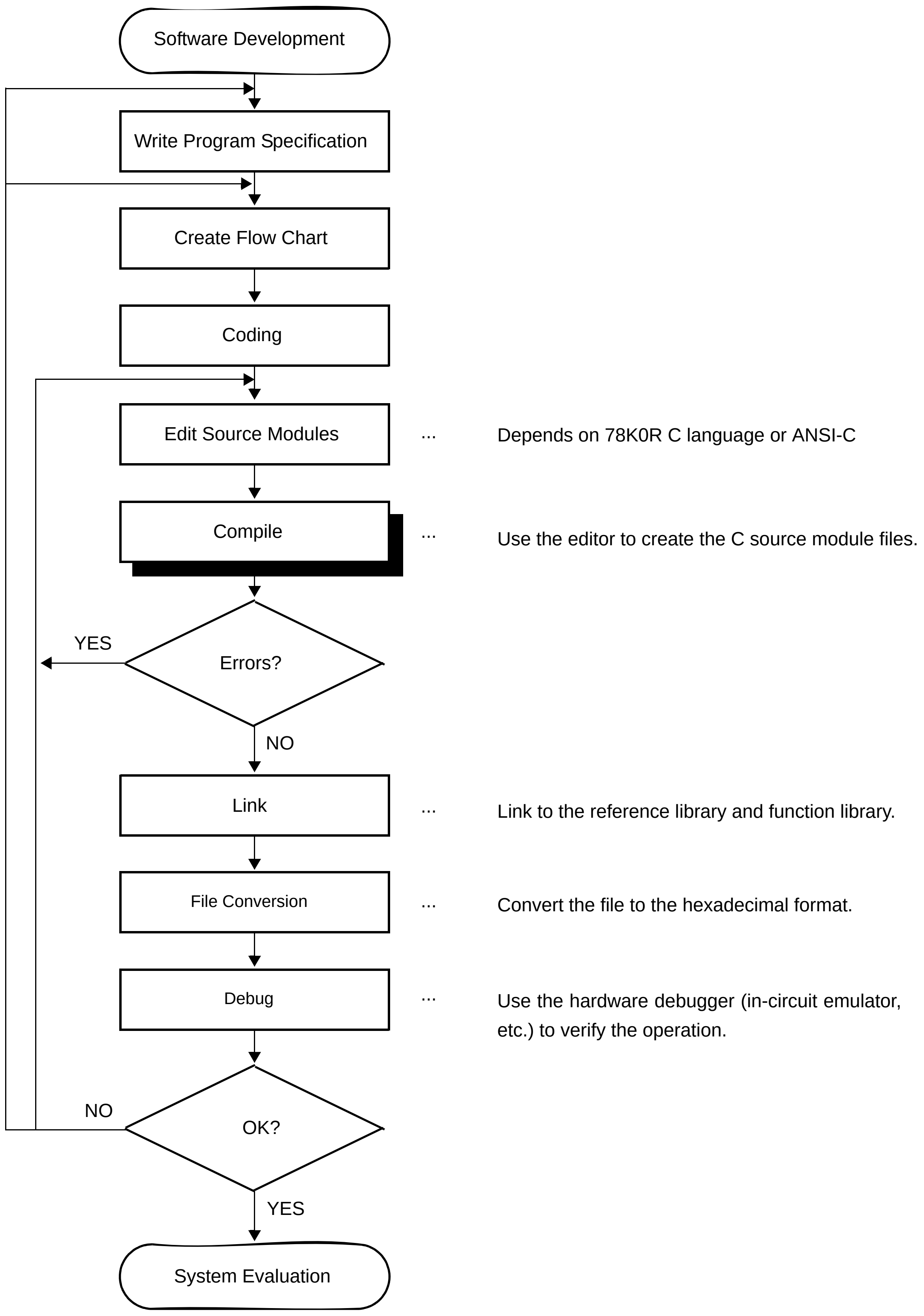
The position of CC78K0R in product development is shown below.

Figure 1-1 Development Process



The software development process is shown below.

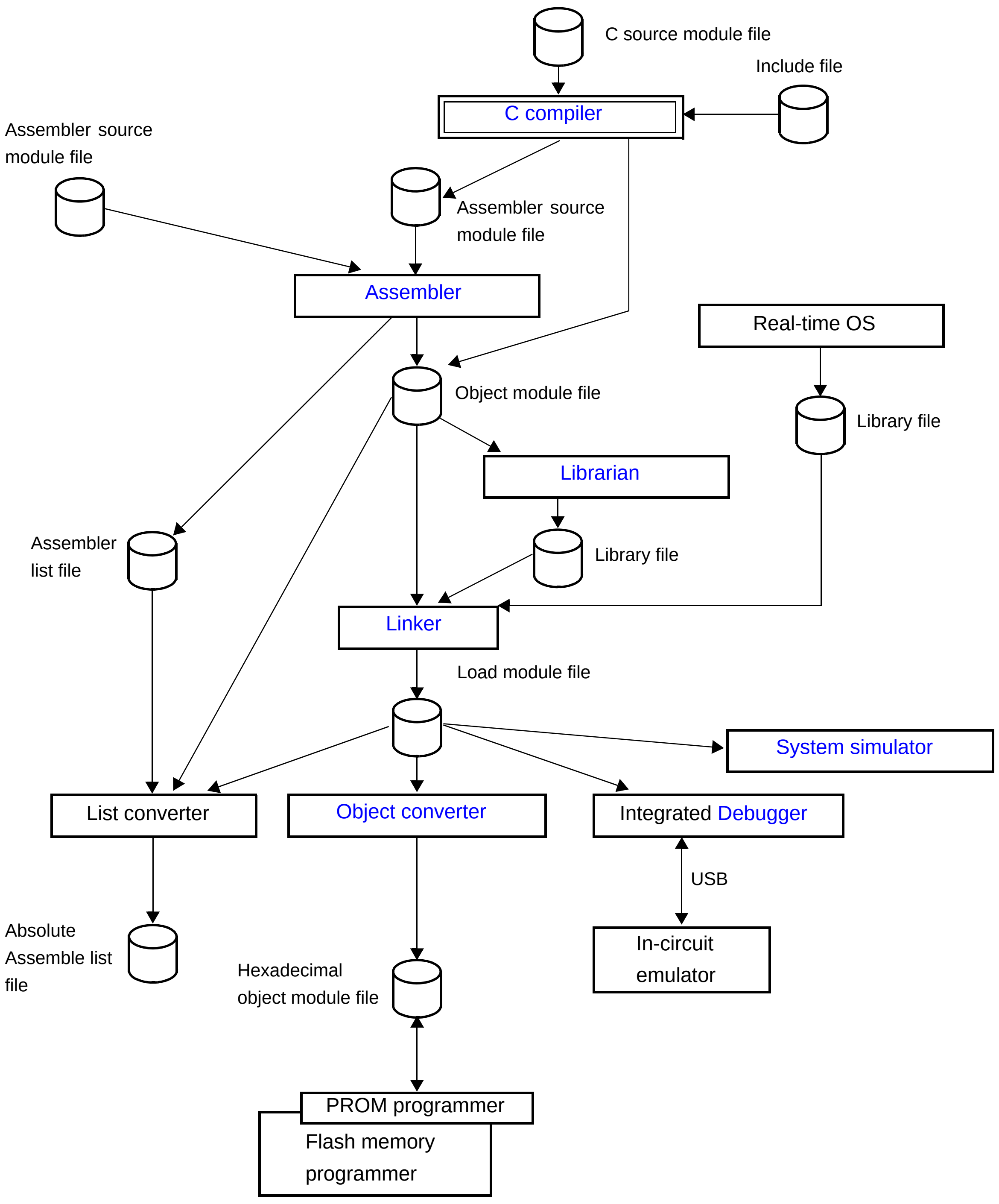
Figure 1-2 Software Development Process


Software Development
Write Program Specification
Create Flow Chart
Coding
Edit Source Modules
Compile
Errors?
YES
NO
Link
File Conversion
Debug
OK?
NO
YES
System Evaluation
··· Depends on 78K0R C language or ANSI-C
··· Use the editor to create the C source module files.
··· Link to the reference library and function library.
··· Convert the file to the hexadecimal format.
··· Use the hardware debugger (in-circuit emulator, etc.) to verify the operation.

1.2 Development Procedure Using CC78K0R

The development procedure using CC78K0R is shown below.

Figure 1-3 Program Development Procedure Using CC78K0R



1.2.1 Using editor to create source module files

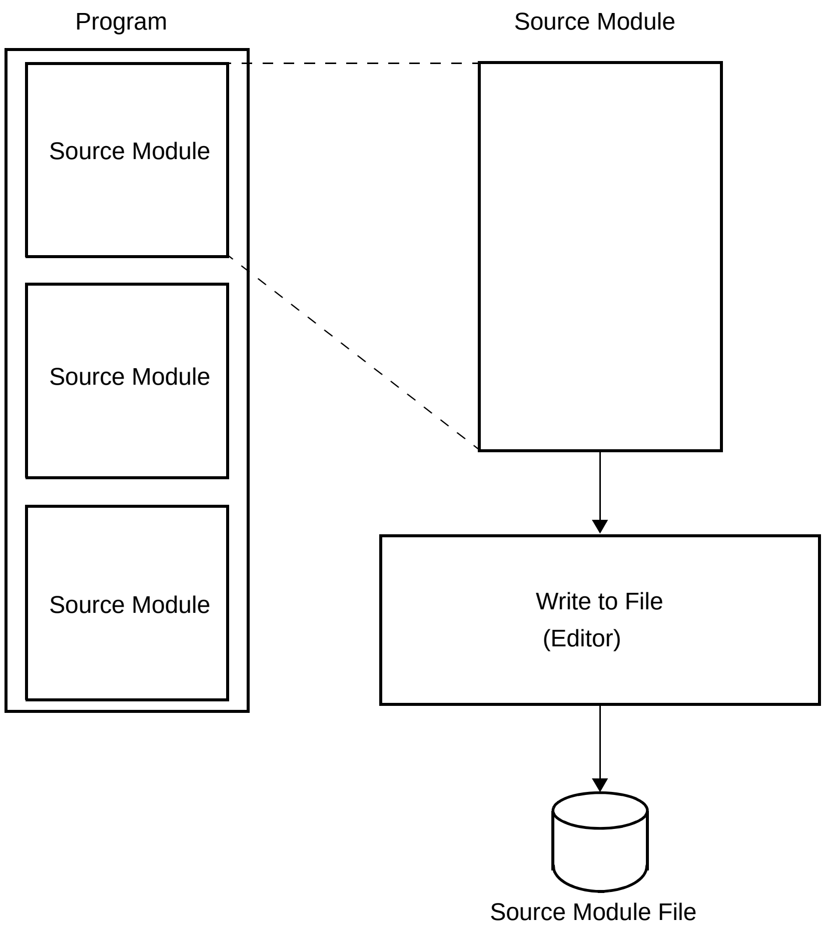
1 program is divided into several functional modules.

1 module is the coding unit and becomes the input unit to the C compiler. A module that is the input unit to the C compiler is called a C source module.

After each C source module is coded, use the editor to save the source module to a file. A file created in this way is called a C source module file.

The C source module files become the CC78K0R input files.

Figure 1-4 Creating Source Module Files



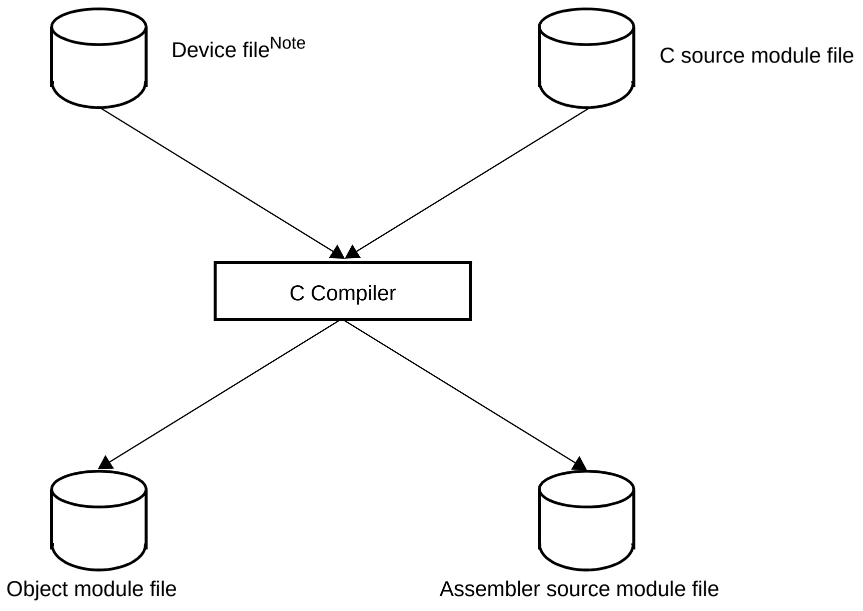
1.2.2 C compiler

The C compiler translates C language into a machine language, taking the device file and C source module file as input.

If it finds a description error in the C source module file, the C compiler outputs a compilation error. If no compilation error occurs, an object module file is output.

In addition, an assembler source module file can also be output so that the program can be modified and checked at the assembly language level. To output an assembler source module file, specify the -a option or -sa option when compiling (for details of options, refer to "CHAPTER 5 COMPILER OPTIONS".).

Figure 1-5 C Compiler Function



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

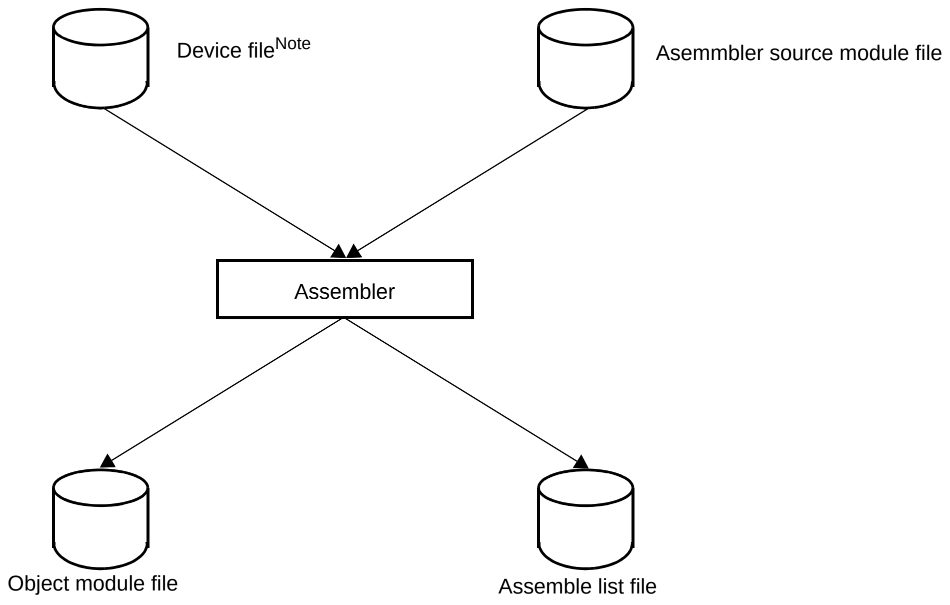
http://www.necel.com/micro/ods/eng/

1.2.3 Assembler

Assembly is executed by using the assembler included in the RA78K0R assembler package (sold separately).

The assembler inputs the device file and assembler source module file and translates the source module file from the assembly language to a machine language. If a description error is found in the assembler source module, an assembly error is output. If no assembly error occurs, an object module file containing machine language information and location information that indicates to which address of memory each machine language code is to be allocated is output. In addition, information during assembly is also output as an assemble list file.

Figure 1-6 Assembler Function



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

1.2.4 Linker

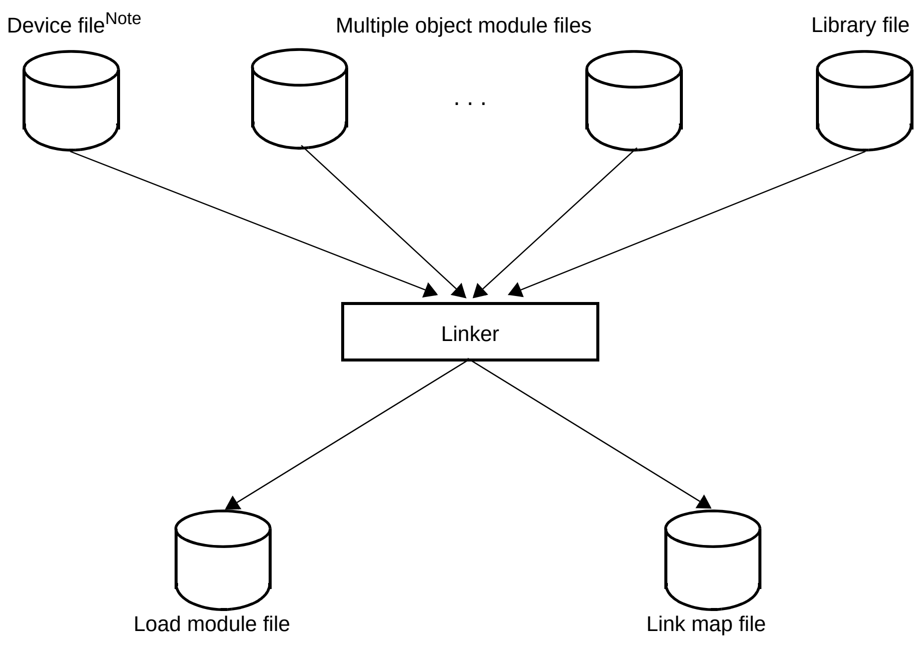
Linking is performed by using the linker included in the RA78K0R Assembler Package (sold separately).

The linker inputs the device file and multiple object module files output by the C compiler or object module files output by the assembler, and links them to the library files (even if there is 1 object module, linking must be performed).

1 load module file is output.

In this case, the linker determines the location addresses of relocatable segments in the input module. This determines the values of relocatable symbols and external reference symbols, and embeds the correct values in the load module file. The linker outputs the linking information as a link map file.

Figure 1-7 Linker Function



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

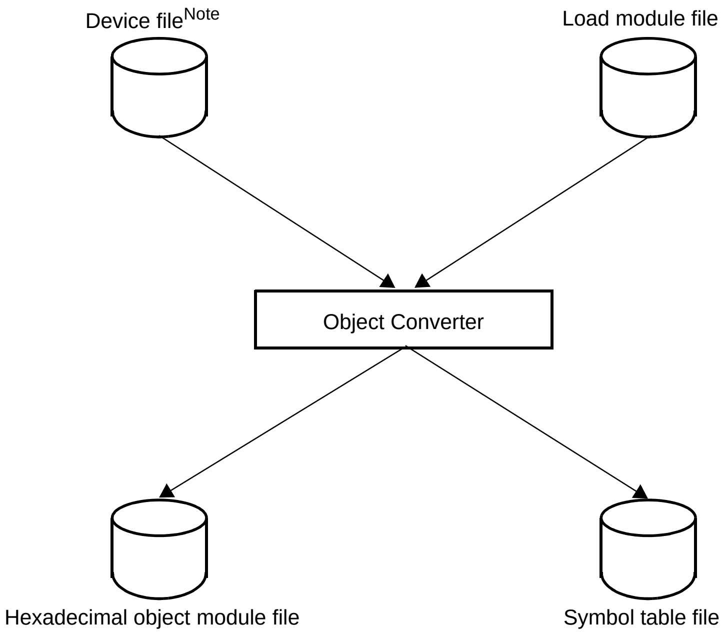
http://www.necel.com/micro/ods/eng/

1.2.5 Object converter

The object converter uses the converter included in the RA78K0R Assembler Package (sold separately).

The object converter inputs the device file and load module file output by the linker and converts it into an Intel HEX format object module file. The object converter also outputs information upon file conversion as a symbol table file.

Figure 1-8 Object Converter Function



Note Obtain the device file by downloading it from the Version-up Service, which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

1.2.6 Librarian

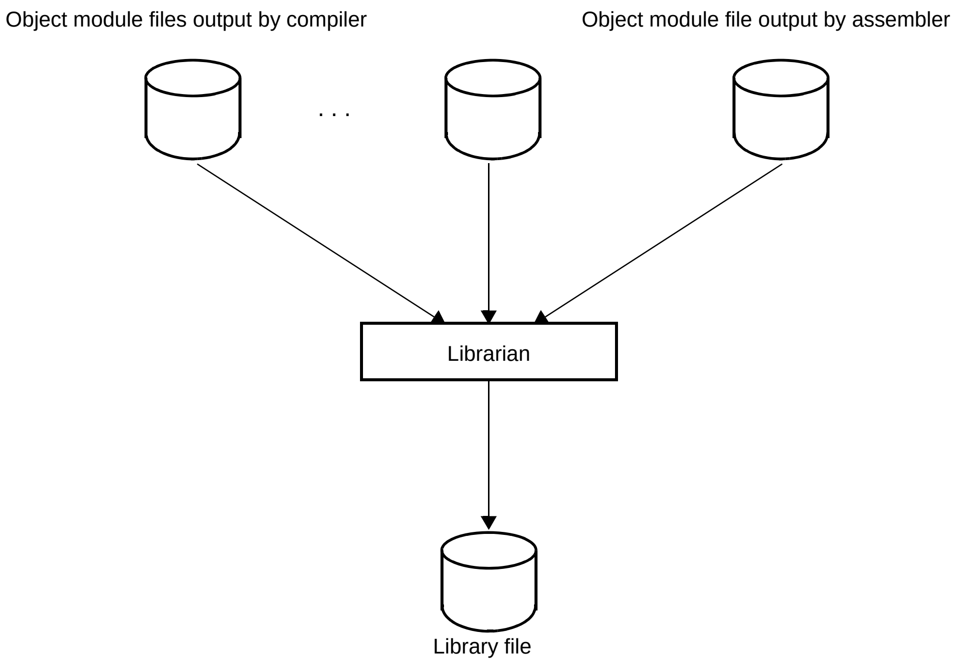
Clearly defined modules having a general interface are formed into a library for convenience. By creating the library, many object modules form 1 file and become easy to handle.

The linker has functions to extract only the needed modules from the library file and link them. Therefore, if multiple modules are registered in 1 library file, the names of the module files needed when linking no longer have to be individually specified.

The librarian uses the librarian included in the RA78K0R Assembler Package (sold separately).

The librarian inputs object module files that are output from the C compiler or assembler, and then outputs a library file.

Figure 1-9 Librarian Function


Object module files output by compiler
Object module file output by assembler
. . .
Librarian
Library file

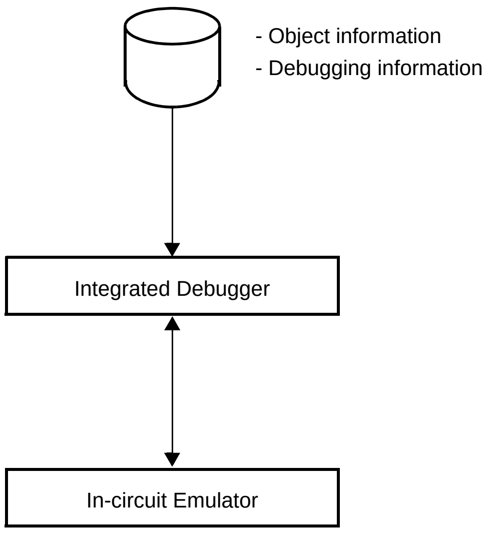
1.2.7 Debugger

Source debugging using a GUI becomes possible by loading the load module files output by the linker into the IE (in-circuit emulator) by using the ID78K0R-QB (78K0R integrated debugger).

To debug, the -g option specifying the output of debugging information is specified when the target source program is compiled (-g is the default option). By making this specification, the symbols and line numbers needed in debugging are added to the object module. For information on the compiler options, see "CHAPTER 5 COMPILER OPTIONS".

The debugger and the IE are packaged separately (sold separately).

Figure 1-10 Debugger Function



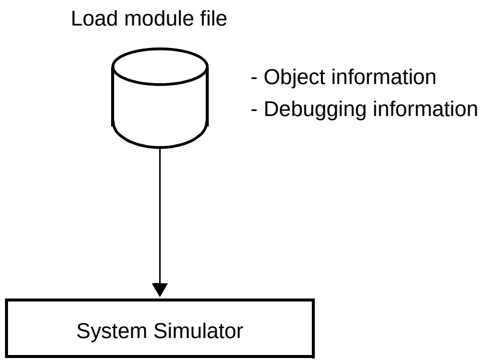
1.2.8 System simulator

Source debugging using a GUI becomes possible by downloading the load module files output from the linker by using the SM+ for 78K0R (78K0R system simulator).

SM+ for 78K0R is software that simulates a load module file on the host machine, in the same manner as operating with the ID78K0R-QB.

In addition to simulating machine instructions in the SM+ for 78K0R, the on-chip peripherals for the devices and the interrupts can be simulated. Since external parts and procedures are provided to construct dummy target systems, the programs including the operation of the target system are debugged at an early stage independent of hardware development.

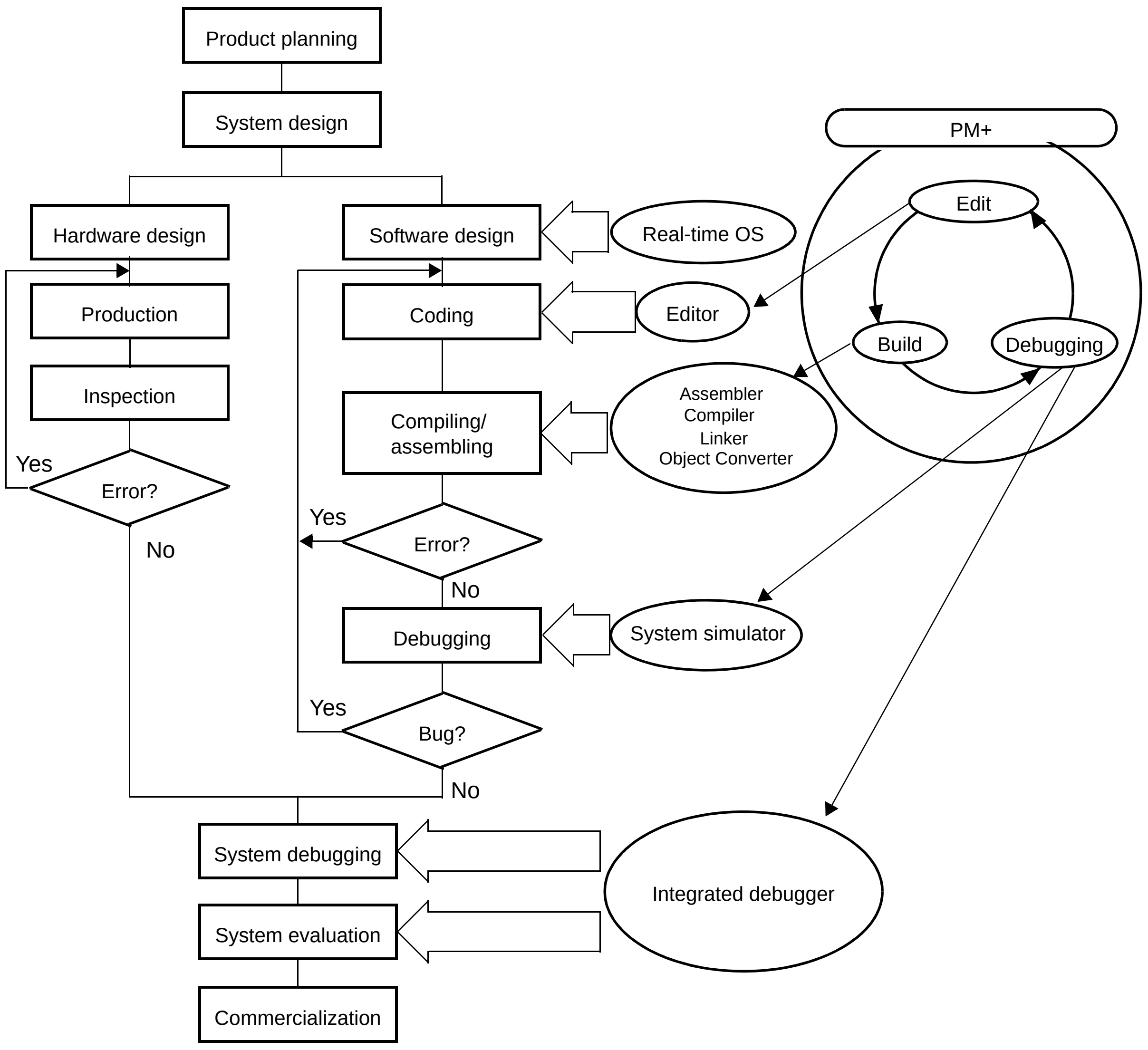
Figure 1-11 System Simulator Function



1.2.9 PM+

PM+ provides an integrated development environment that allows users to develop programs efficiently. Using PM+, a series of works required upon user program development, such as starting the editor, builder and debugger, can be performed.

Figure 1-12 PM+ Function



CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION

This chapter explains the procedure to install the files stored in the supply media of the CC78K0R to the user development environment (host machine) and the procedure to uninstall them from the user development environment.

2.1 Host Machines and Supply Medium

The CC78K0R supports the development environments listed below.

Table 2-1 Supply Medium and Recording Formats for CC78K0R

OS	Supply Medium
Windows (2000/XP Professional/XP Home Edition)	CD-ROM

2.2 Installation

The procedure for installing to the host machine the files provided in the CC78K0R's supply media is described below.

(1) Starting up Windows

Power on the host machine and peripherals and start Windows.

(2) Set supply media

Set the CC78K0R's supply media in the appropriate drive (CD-ROM drive) of the host machine. The setup programs will start automatically. Perform the installation by following the messages displayed in the monitor screen.

Caution If the setup program does not start automatically, execute INSTALL.EXE in the root.

(3) Confirmation of files

Using Windows Explorer, etc., check that the files contained in the CC78K0R's supply media have been installed to the host machine.

For the details of each folder, refer to "2.4 Folder Configuration".

2.3 Installation of Device Files

Obtain the device file by downloading it from the Online Delivery Service (ODS), which can be accessed from the following Website.

http://www.necel.com/micro/ods/eng/

Use the device file installer to install the device files. The device file installer is installed at the same time as the CC78K0R.

2.4 Folder Configuration

The standard folder displayed during installation is "Program Files\NEC Electronics Tools" of the Windows system. The configuration under the install folder is as follows. Note that the drive and install folder can be changed during installation. When performing make operation with PM+, perform installation of tools (CC78K0R, RA78K0R) to the same drive and folder.

The descriptions in this manual assume installation to the standard folder with "Program Files\NEC Electronics Tools", which is the default program name, according to the setup program default directions.

Figure 2-1 Folder Configuration

```
Program Files\NEC Electronics Tools\
    │
    ├── CC78K0R\Vx.xx\bin\ ...............  Folder storing C Compiler
    │
    ├── CC78K0R\Vx.xx\doc\ ...............  Folder storing user's manual and supplementary explanations
    │
    ├── CC78K0R\Vx.xx\hlp\ ...............  Folder storing on-line help file
    │
    ├── CC78K0R\Vx.xx\inc78k0r\ ......  Folder storing header files
    │
    ├── CC78K0R\Vx.xx\lib78k0r\ .......  Folder storing library files, object files for startup routines
    │
    ├── CC78K0R\Vx.xx\smp78k0r\CC78K0R\ .....  Folder storing  files for verifying installation
    │
    └── CC78K0R\Vx.xx\src\cc78k0r\
                  │
                  ├── bat\ ......................  Folder storing batch files
                  │
                  ├── lib\ .......................  Folder storing library files, object files for startup routines
                  │
                  └── src\ ......................  Folder storing source files
```

2.5 File Organization

The table below lists the contents of each folder.

The folder structure and file organization are the ones obtained when the installer was used.

Table 2-2 File Organization

Folder Name	File Name	Description
CC78K0R\V*x.xx*\bin\	cc78k0r.exe	Compiler
	cc78k0r.msg	Message file
	*.hlp	Help message file
	*.dll	DLL files
CC78K0R\V*x.xx*\hlp\	cc78k0rp.chm	On-line help file
CC78K0R\V*x.xx*\inc78k0r\	*.hNote 1	Header files for standard library
CC78K0R\V*x.xx*\lib78k0r\ (For link)Note2, 3	cl0r*.lib	Libraries (runtime and standard libraries)
	s0r*.rel	Object files for startup routines
CC78K0R\V*x.xx*\smp78k0r\CC78K0R\	prime.c	Source program for verifying installation
	sample.bat	Batch files for verifying installation
	readme.doc	Explanation of files for verifying installation
	lk78k0r.dr	Link directive file for reference
CC78K0R\V*x.xx*\src\cc78k0r\bat\Note4	mkstup.bat	Assemble batch files for startup routines
	reprom.bat	For updating rom.asm
	*.batNote5	Batch files for updating standard functions (partial)
CC78K0R\Vx.xx\src\cc78k0r\lib\ (For modifications)Note2	cl0r*.lib	Libraries (runtime and standard libraries)
	s0r*.rel	Object files for startup routines
CC78K0R\V*x.xx*\src\cc78k0r\src\	cstart*.asmNote 4	Source files for startup routines
	rom.asm	Source files for ROMization routine
	*.asmNote 5	Source files for standard functions (partial)

Remark *: Alphanumeric symbols

Note 1 Refer to CC78K0R C Compiler Language User's Manual.

Note 2 To modify the startup routine, modify the source file below the CC78K0R\V*x.xx*\src\cc78k0r\lib folder. The file assembled with a batch file is stored in the CC78K0R\Vx.xx\src\cc78k0r\lib folder. Copy the file to the CC78K0R\Vx.xx\lib78k0r folder and link it with the user program.

Note 3 Refer to "2.5.1 Library files".

Note 4 The batch files contained in this folder are provided for use in command prompt, so they cannot be used in PM+. Use these files only when modification of source files is required.

Note 5 Refer to the contents in Table 8-1.

Note 6 * = B | E | N (B: when the boot area is specified, E: when the flash area is specified, N: when the standard libraries are not used)

Note 7 Refer to the contents in Table 8-3.

2.5.1 Library files

The library file consist of standard libraries, runtime libraries, and startup routines.

The table below lists the folder contents.

Table 2-3 Library Files

Folder Name	File Name			File Role
	Normal	Boot Area	Flash Area	
lib78k0r\	cl0rm.lib cl0rl.lib cl0rmf.lib cl0rlf.lib cl0rxm.lib Note3 cl0rxl.lib Note3	cl0rm.lib cl0rl.lib cl0rmf.lib cl0rlf.lib cl0rxm.lib Note3 cl0rxl.lib Note3	cl0rme.lib cl0rle.lib cl0rmfe.lib cl0rlfe.lib cl0rxme.lib Note3 cl0rxle.lib Note3	Library (runtime and standard libraries) Note 1
	s0rm.rel s0rml.rel s0rl.rel s0rll.rel	s0rmb.rel s0rmlb.rel s0rlb.rel s0rllb.rel	s0rme.rel s0rmle.rel s0rle.rel s0rlle.rel	Object files for startup routines Note 2

Note 1 The rule for naming libraries is given below.

```
lib78k0r\cl0r<mul><model><float><flash>.lib
```

<mul>

None:	Multiplier not used
x:	Multiplier used

<model>

m:	Small model or medium model
l:	Large model

<float>

None:	Standard library and runtime library (floating point library is not used)
f:	For floating point library

<flash>

None:	For normal/boot area
e:	For flash memory area

Note 2 The rule for naming startup routines is given below.

```
lib78k0r\s0r<model><lib><flash>.rel
```

<model>

m:	Medium model (can also be used for specifying the small model)
l:	Large model

<lib>

None:	When standard library functions are not used
l:	When standard library functions are used

<flash>

None:	Normal
b:	For boot area
e:	For flash memory area

Note 3 The CC78K0R libraries are compatible with the following multiplier devices.
However, if an interrupt occurs while computation is in progress, some of the computation results are disabled from being interrupted so that they are not corrupted.
Refer to the CC78K0R C Compiler Language User's Manual in regards to library functions and interrupt disable times.

[Special function register]

Function	Reserved Words	Addresses	Size
Multiplication input data A	MULA	FFFF0H	16bit
Multiplication input data B	MULB	FFFF2H	16bit
Multiplication result data	MULOH, MULOL	FFFF4H, FFF6H	16bit x 2

<Register configuration>

<Multiplier A> <Multiplier B> <Product>

MULA (bits 15 to 0) * MULB (bits 15 to 0) = MULOH (upper) (bits 15 to 0), MULOL (lower) (bits 15 to 0)

2.6 Uninstallation

The procedure for uninstalling the files installed to the host machine is described below.

(1) Windows startup

Power on the host machine and peripherals and start Windows.

(2) Deletion of CC78K0R

Open "Add/Remove Programs" or "Add or Remove Programs" on the Control Panel and select "NEC EL CC78K0R V*x.xx*".

(3) Confirmation of files

Using Windows Explorer, etc., check that the files installed to the host machine have been uninstalled.

For the details of each folder, refer to "2.4 Folder Configuration".

2.7 Environment Settings

2.7.1 Host machine

Machine in which Windows 2000/XP can operate

- Windows 2000/XP
- Command prompt in Windows 2000/XP

2.7.2 Environment variables

Set the following environment variables for command prompt operation.

Table 2-4 Environment Variables

Environment Variable	Description
PATH	Specifies the folder where the compiler is located.
TMP	Specifies the folder where temporary files are created.
LANG78K	Specifies the kanji code (2-byte code) in the source files. sjis: Shift JIS (Default) euc: EUC none: No 2-byte codes
INC78K0R	Specifies the folder where the standard header files of the C compiler are located.
LIB78K0R	Specifies the folder where the C compiler's libraries are located.

[Specification Example]

```
PATH=%PATH%;C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\bin
set TMP=C:\tmp
set LANG78K=sjis
set INC78K0R=C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r
set LIB78K0R=C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r
```

CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING

This chapter uses the CC78K0R and the RA78K0R Assembler Package to describe the procedure from compiling to linking.

By actually performing the processes from compiling to linking of the “prime.c” sample program following the execution procedure given in this chapter, you can become familiar with the operations of compiling, assembling, and linking (see "APPENDIX A SAMPLE PROGRAMS" for information about the sample program).

How to execute on the PM+ and how to execute from the command line is described (for information on installation, see "2.2 Installation").

3.1 PM+

This section describes the user interface when the CC78K0R is started in PM+ included in the RA78K0R Assembler Package.

If the CC78K0R is started from PM+, cc78k0rp.dll included in CC78K0R is referenced.

3.1.1 Position of cc78k0rp.dll (tools DLL)

The tools DLL file, such as the cc78k0rp.dll file, is needed to run the Windows version of the 78K0R C compiler (CC78K0R) from PM+.

3.1.2 Execution environment

This environment conforms to PM+.

3.1.3 CC78K0R option setting menu

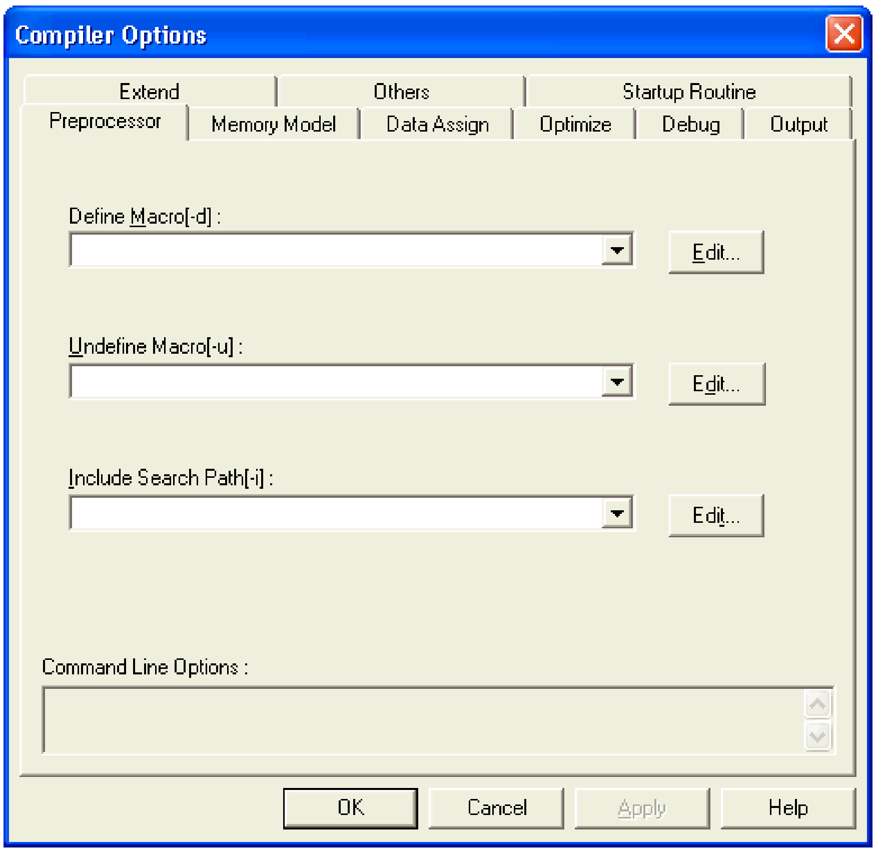
(1) Option menu items

The item [Compiler Options] is added to the [Tools] menu in PM+ by the tools DLL file included in the CC78K0R C Compiler Package.

(2) [Compiler Options] dialog box

Select the [Compiler Options] menu under [Tools] in PM+ to call the option setting function for the tools DLL and open the [Compiler Options] dialog box.

Figure 3-1 [Compiler Options] Dialog Box



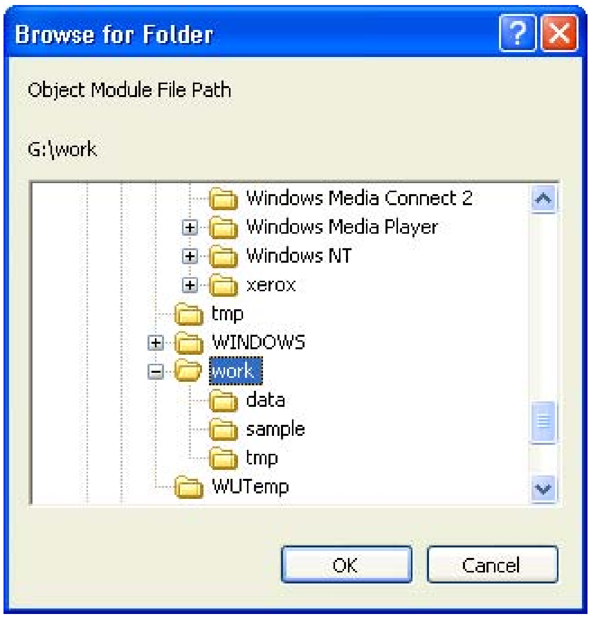
(a) [Browse for Folder] dialog box

In the [Compiler Options] dialog box, when the [Browse] button is clicked for the following path settings, the following dialog box appears.

Only the folders can be specified in this dialog box.

- Object module file output path under the [Output] tab
- Assembler module file output path under the [Output] tab
- Error list file output path under the [Output] tab
- Cross-reference list file output path under the [Output] tab
- Preprocessor list file output path under the [Output] tab
- Temporary file path under the [Others] tab

Figure 3-2 [Browse for Folder] Dialog Box



Browse for Folder

Object Module File Path

G:\work

(b) [ParameterFile] dialog box

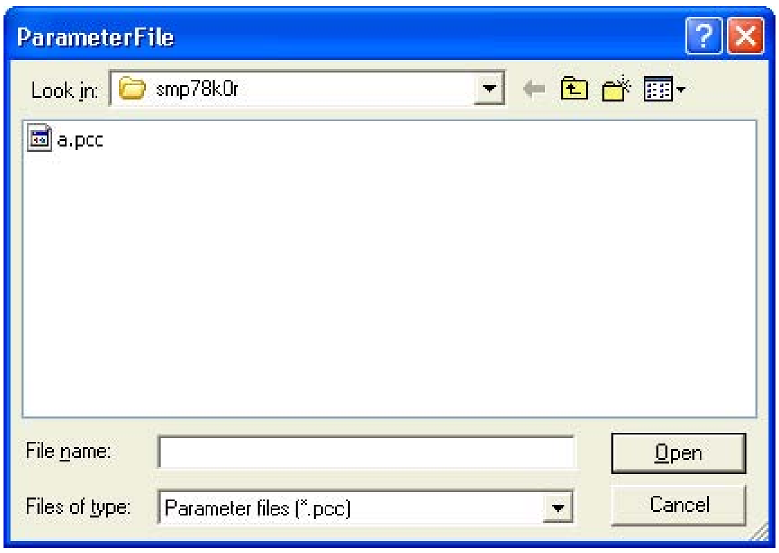
When the [Browse] button is clicked for the following path settings, the following dialog box appears.

- Parameter file under the [Others] tab

This dialog box displays the following.

Current folder:	Project file folder
File type:	Parameter files (*.pcc)

Figure 3-3 [ParameterFile] Dialog Box



ParameterFile

Look in: smp78k0r

a.pcc

File name:

Files of type: Parameter files (*.pcc)

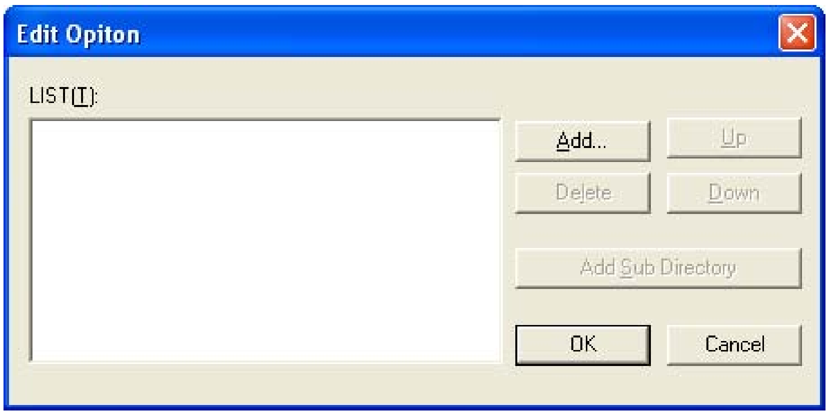
(c) [Edit Option]dialog box

In the [Compiler Options] dialog box, when the [Edit...] button is clicked for the following path settings, the following dialog box appears.

- Define macro under the [Preprocessor] tab
- Undefine macro under the [Preprocessor] tab
- Include search path under the [Preprocessor] tab

Items are edited in list format in the [Edit Option] dialog box.

Figure 3-4 [Edit Option] Dialog Box



The [Edit Option] dialog box is described below.

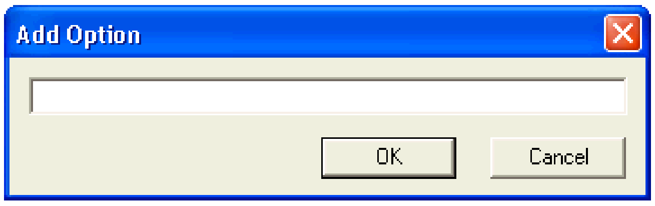
- [Add...] button

  Adds a list item.

  If the item to be added is a file or folder, the corresponding [Browse for Folder] dialog box opens.

  In all other cases, the [Add Option] dialog box opens. Specify details of the item to be added in this box.

Figure 3-5 [Add Option] Dialog Box



- [Delete] button

  Deletes the selected list item.

- [Up] button

  Moves the selected list item up.

- [Down] button

  Moves the selected list item down.

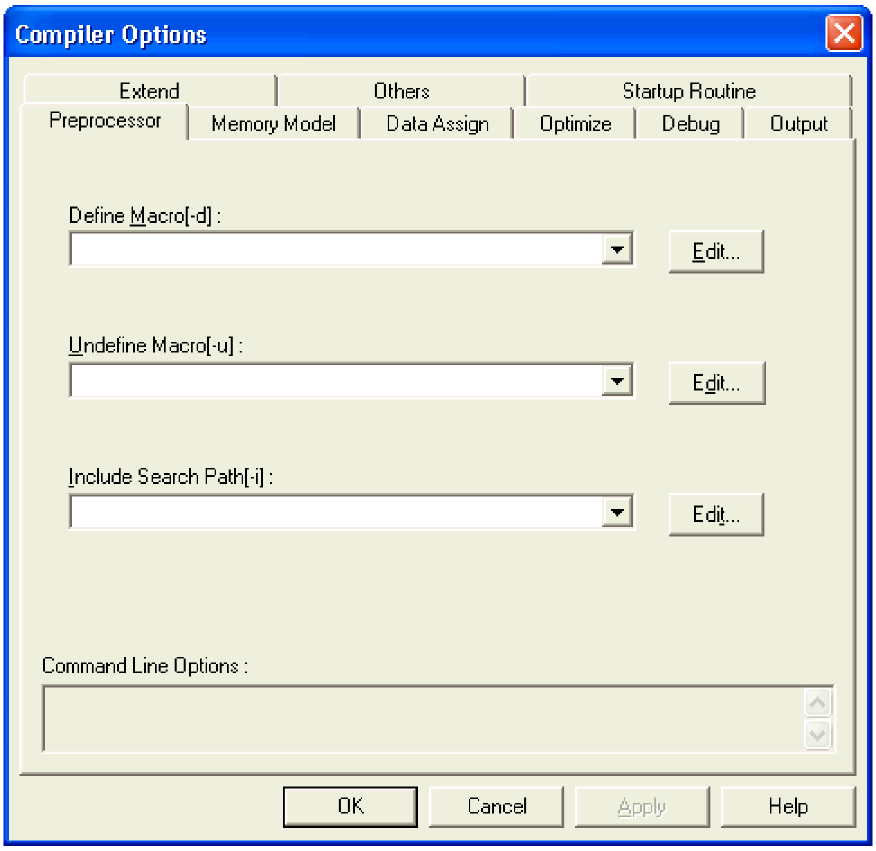
- [Add Sub Directory] button

  A subdirectory can be added to the selected list item when the item is specified as Include Search Path[-i](I) under the [Preprocessor] tab.

3.1.4 Description of each part of [Compiler Options] dialog box

Each part of the [Compiler Options] dialog box is described.

Figure 3-6 [Compiler Options] Dialog Box



- Setting of compiler options

  The compiler options are divided into the following 9 options and set respectively.

  Each setting screen is displayed by clicking the corresponding tab at the top of the dialog box.

  [Preprocessor] tab (default)

  [Memory Model] tab

  [Data Assign] tab

  [Optimize] tab

  [Debug] tab

  [Output] tab

  [Extend] tab

  [Others] tab

  [Startup Routine] tab

- Command Line Options

  The option character string currently set is displayed.

  The option character string entered is reflected and displayed in real time.

  Nothing can be input in this display area.

  Even though the default option of the CC78K0R is the "specified" state (i.e., a check box is selected, etc.), nothing is displayed in this area by default.

  Options that do not fit in the option character display area can be checked by scrolling with the scroll bar.

- [OK] button

  The settings edited in this dialog box are set, and the [Compiler Options] dialog box closes.

  If [Special Compiler Options] is selected in the Project Window, the options are set for the source file. If [Compiler Options] is selected in the [Tools] menu, the options are set for all of the source files.

- [Cancel] button

  The options are not set, and the dialog box closes.

  The ESC key has the same effect as the [Cancel] button no matter where the focus is in the dialog box.

- [Apply] button

  This button is effective only when option settings have been changed.

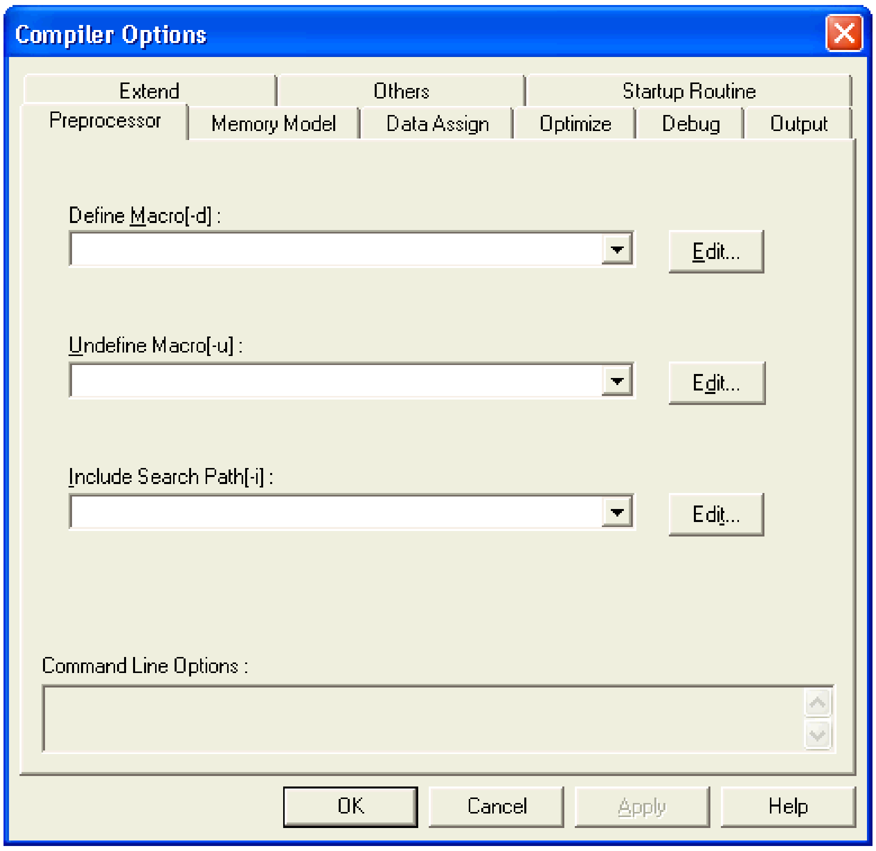
  The edited contents in this dialog box are applied and the [Compiler Options] dialog box remains displayed.

- [Help] button

  The help file for this dialog box opens.

[Preprocessor] tab

Figure 3-7 [Compiler Options] Dialog Box (When [Preprocessor] Tab Is Selected)



- Define Macro[-d]

  Input into this combo box a macro name and a definition name of the macro definition to be enabled.

  For the macro name, 30 macro definitions can be performed at once by delimiting with ",".

  Up to 256 characters can be input for specifying a defined macro name.

  Up to 7,709 characters can be input into this combo box.

  Can be specified using the [Edit...] button. (Opens the [Edit Option]dialog box.)

  An error message will appear if a defined macro name is specified twice.

- Undefine Macro[-u]

  Input into this combo box a macro name of the macro definition to be disabled.

  For the macro name, 30 macro definitions can be invalidated at once by delimiting with ",".

  Up to 256 characters can be input for specifying an undefined macro name.

  Up to 7,709 characters can be input into this combo box.

  Can be specified using the [Edit...] button. (Opens the [Edit Option]dialog box.)

  An error message will appear if an undefined macro name is specified twice.

- Include Search Path[-i]

  The folder that contains include files is input to the combo box.

  64 folders can be specified at once by delimiting with ",".

  Up to 259 characters can be input for specifying an include file path.

  Up to 16,639 characters can be input into this combo box.

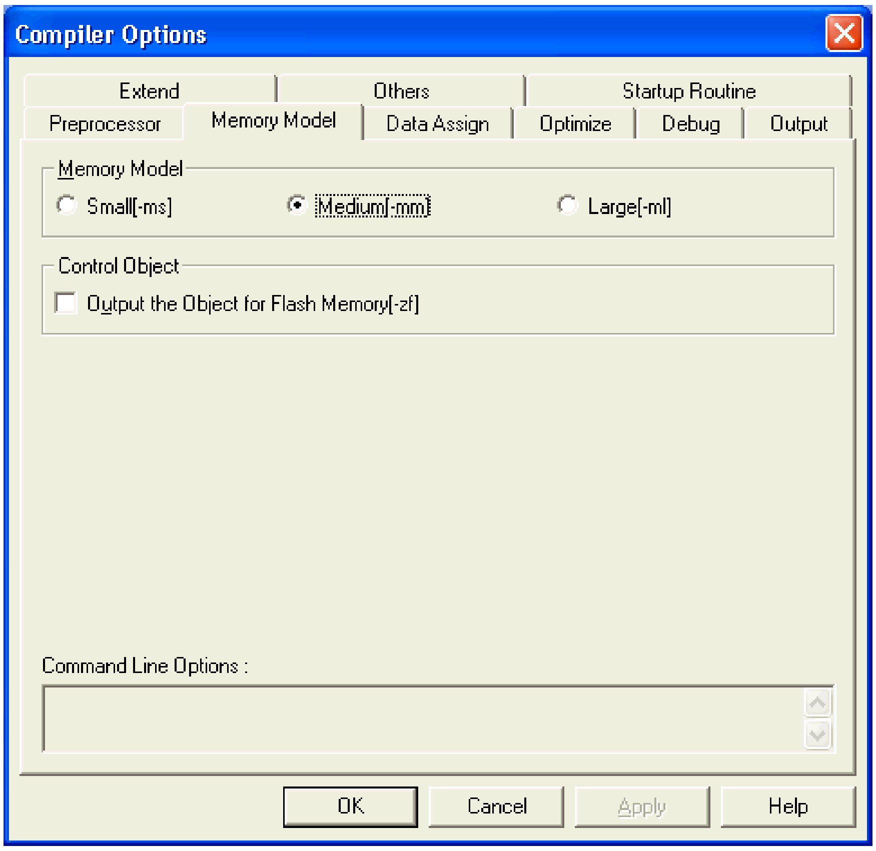
  Can be specified using the [Edit...] button. (Opens the [Edit Option]dialog box.)

  Unexisted path cannot be specified.

  An error message will appear if the same include file path is specified twice.

[Memory Model] tab

Figure 3-8 [Compiler Options] Dialog Box (When [Memory Model] Tab Is Selected)



Memory Model
Small[-ms] [✓ Medium[-mm]] Large[-ml]

Control Object
☐ Output the Object for Flash Memory[-zf]

Command Line Options :

Caution The settings for the [Memory Model] tab cannot be performed if special compiler options are set per source file.

- Memory Model

  Select the memory model type used for compilation by selecting the appropriate radio button (Small[-ms] / Medium[-mm] / Large[-ml]).

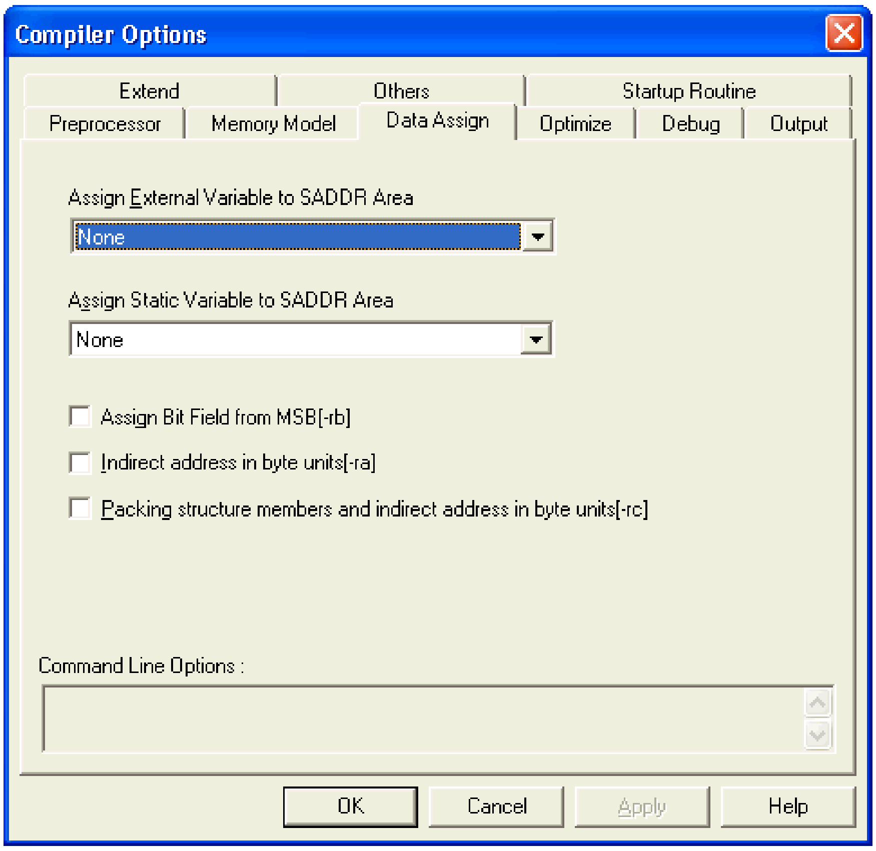
  “Medium[-mm]“ is selected by default.

- Control Object

Output the Object for Flash Memory[-zf]	Select this check box to output object from flash.

[Data Assign] tab

Figure 3-9 [Compiler Options] Dialog Box (When [Data Assign] Tab Is Selected)



Assign External Variable to SADDR Area: None

Assign Static Variable to SADDR Area: None

☐ Assign Bit Field from MSB[-rb]

☐ Indirect address in byte units[-ra]

☐ Packing structure members and indirect address in byte units[-rc]

Command Line Options :

- Assign External Variable to SADDR Area

  The type of an external variable to be assigned to the saddr area is selected in the drop-down list box (None / only char size[-rd1] / char,short,int size[-rd2] / char,short,int,long size[-rd4] / struct,union,array[-rdm] / char size and struct,union,array[-rd1m] / char,short,int size and struct,union,array[-rd2m] / char,short,int,long size and struct,union,array[-rd4m]).

  “None“ is selected by default.

  Caution   This area cannot be performed if special compiler options are set per source file.

- Assign Static Variable to SADDR Area

  The type of a static variable to be assigned to the saddr area is selected in the drop-down list box (None / only char size[-rs1] / char,short,int size[-rs2] / char,short,int,long size[-rs4] / struct,union,array[-rsm] / char size and struct,union,array[-rs1m] / char,short,int size and struct,union,array[-rs2m] / char,short,int,long size and struct,union,array[-rs4m]).

  “None“ is selected by default.

- Assign Bit Field from MSB[-rb]

  Select this check box when a member of the bit field structure is allocated from the MSB.

- Indirect address in byte units[-ra]

  Select this check box to perform indirect reference in 1-byte units.

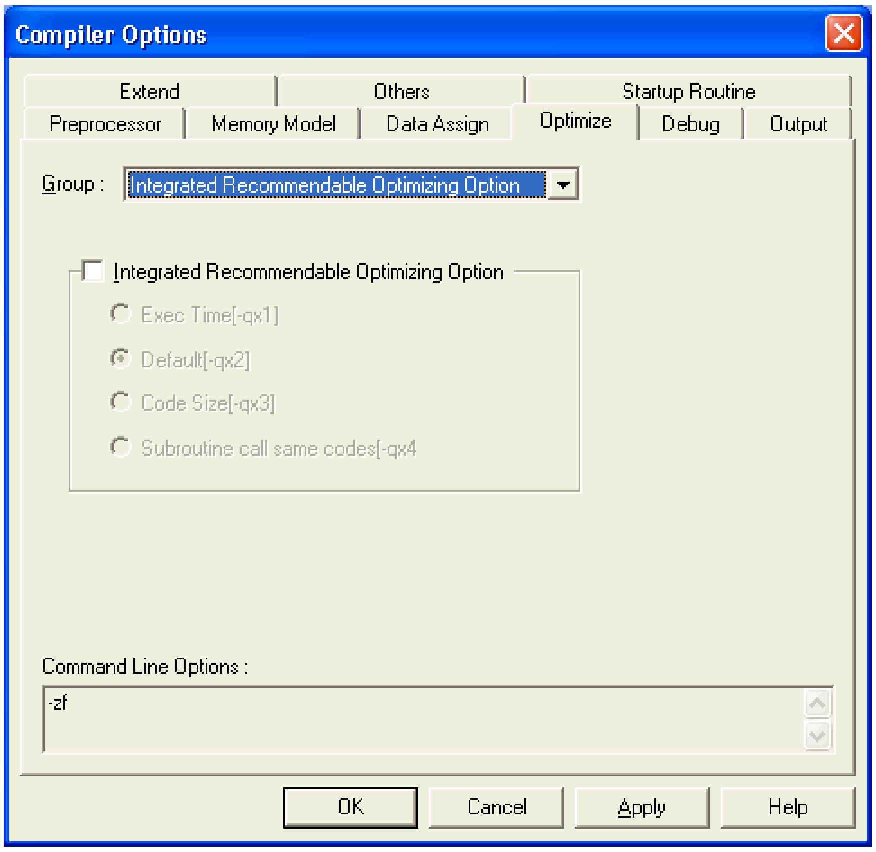
- Packing structure members and indirect address in byte units[-rc]

  Select this check box to pack a structure and perform indirect reference in 1-byte units.

[Optimize] tab

(1) When "Integrated Recommendable Optimizing Option" is selected in the [Group] drop-down list box

Figure 3-10 [Compiler Options] Dialog Box (When [Integrated Recommendable Optimizing Option] Is Selected)



Group : Integrated Recommendable Optimizing Option

☐ Integrated Recommendable Optimizing Option

○ Exec Time[-qx1]
● Default[-qx2]
○ Code Size[-qx3]
○ Subroutine call same codes[-qx4

Command Line Options :

-zf

- Integrated Recommendable Optimizing Option

  The "Integrated Recommendable Optimizing Option" integrates optimization options according to purpose, instead of specifying them individually. Accordingly this option makes the optimization option easier to set.

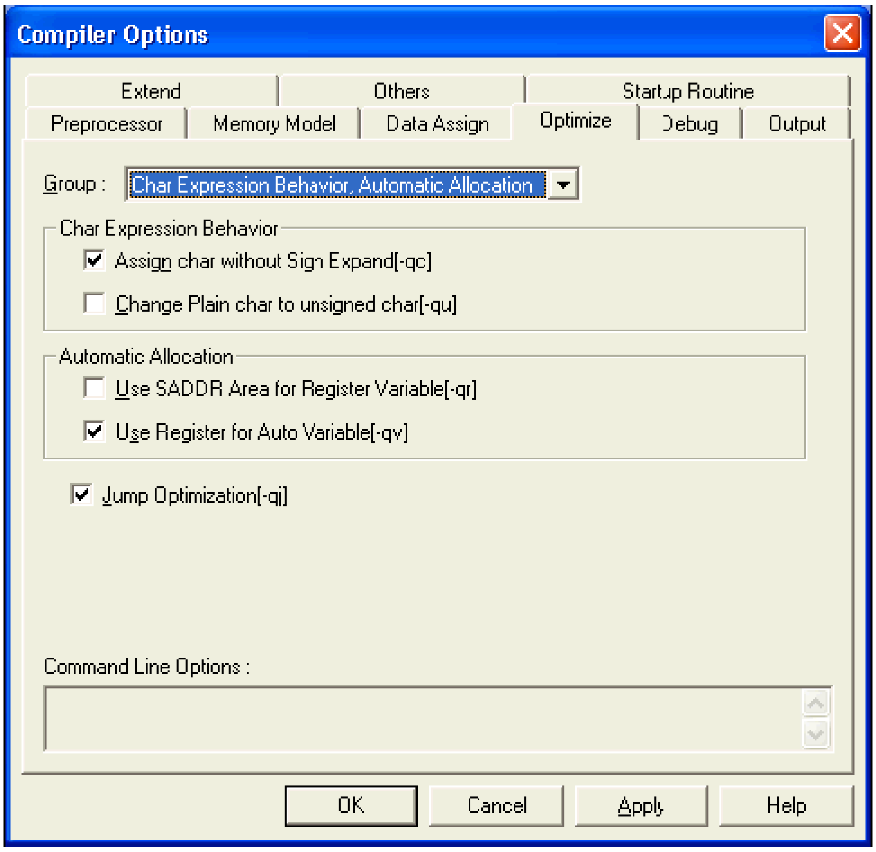
  Select this check box to perform optimization and select the integrated recommendable optimizing option by selecting the appropriate radio button (Exec Time[-qx1] / Default[-qx2]).

  Their meanings are as follows.

Exec Time[-qx1]	Select this option when the efficiency of executing speed is important.
Default[-qx2]	Select this option when both the efficiency of executing speed and the efficiency of object code size are equally important (selected by default).
Code Size[-qx3]	Select this option when the efficiency of object code size is important.
Subroutine call same codes[-qx4]	Select this option when subroutine call same codes.

(2) When "Char Expression Behavior, Automatic Allocation" is selected in the [Group] drop-down list box

Figure 3-11 [Compiler Options] Dialog Box (When [Char Expression Behavior, Automatic Allocation] Is Selected)



Group : Char Expression Behavior, Automatic Allocation

Char Expression Behavior

[✓] Assign char without Sign Expand[-qc]

[ ] Change Plain char to unsigned char[-qu]

Automatic Allocation

[ ] Use SADDR Area for Register Variable[-qr]

[✓] Use Register for Auto Variable[-qv]

[✓] Jump Optimization[-qj]

Command Line Options :

- Char Expression Behavior

Assign char without Sign Expand[-qc]	Select this check box to perform calculations including char without sign extension (selected by default).
Change Plain char to unsigned char[-qu]	Select this check box to regard the char as a unsigned char.

- Automatic Allocation

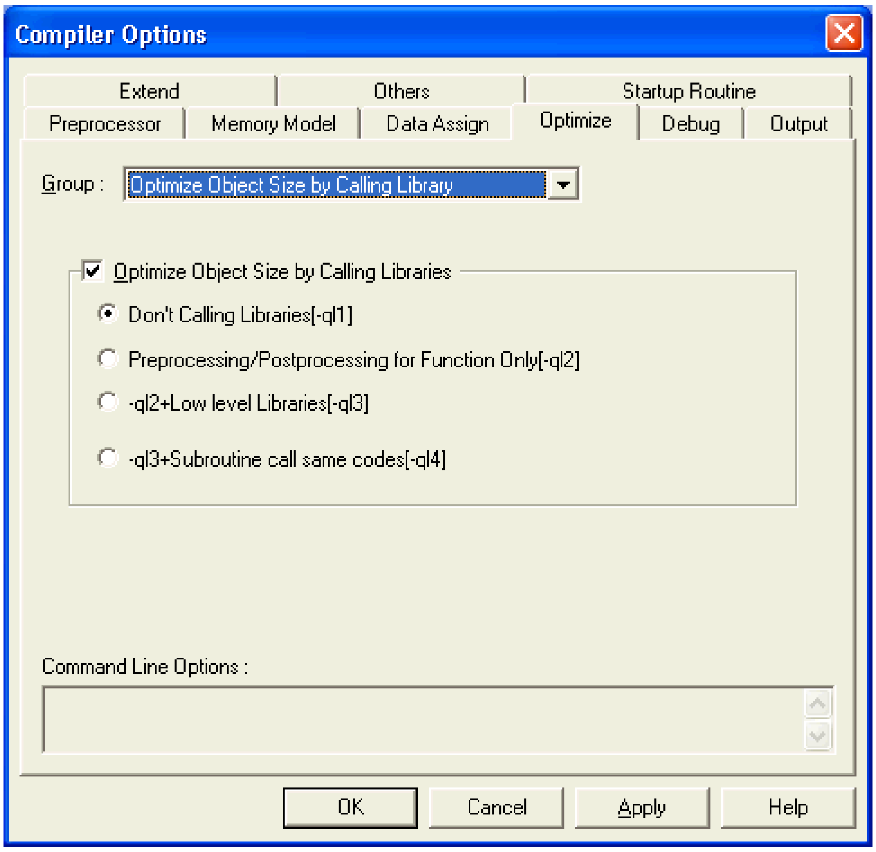
Use SADDR Area for norec+Register Variable[-qr]	Select this check box to add a register variable to a register and assigns it to the saddr area and select the variable to be assigned by selecting the appropriate radio button (norec Parameter/ Auto Variable[-qr1] / -qr1+Register Variable[-qr2]). If this check box is selected, "-qr + Register Variable[-qr2]" is selected by default.
Use Register for Auto Variable[-qv]	Selecet the check box to assign an auto variable automatically to a register or the saddr area (selected by default).

- Jump Optimization[-qj]

  Selecet the check box to optimize jump instructions (selected by default).

(3) When "Optimize Object Size by Calling Library" is selected in the [Group] drop-down list box

Figure 3-12 [Compiler Options] Dialog Box (When [Optimize Object Size by Calling Library] Is Selected)



Group : Optimize Object Size by Calling Library

[✓] Optimize Object Size by Calling Libraries

[●] Don't Calling Libraries[-ql1]

[ ] Preprocessing/Postprocessing for Function Only[-ql2]

[ ] -ql2+Low level Libraries[-ql3]

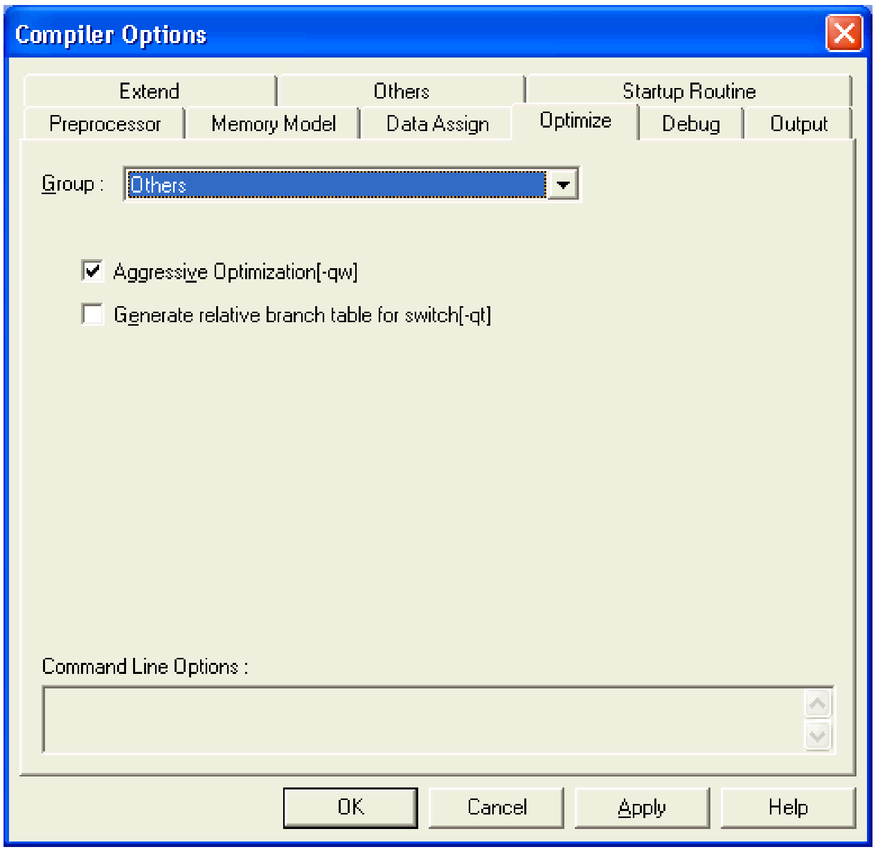
[ ] -ql3+Subroutine call same codes[-ql4]

Command Line Options :

- Optimize Object Size by Calling Libraries
  Select this check box to perform optimization that gives priority to the code size and replace the standard code pattern with a library (selected default). Specify using radio buttons the range to be replaced by a library (for the case when selecting “Don’t Calling Libraries[-ql1]”, “Preprocessing for Function Only[-ql2]”, “-ql2+Low level Libraries[-ql3]“, or “-ql3+Subroutine call same codes[-ql4]”).
  “Don’t Calling Libraries[-ql1]“ is selected by default.
  When the number n of -qln becomes greater, the object code size becomes smaller, and accordingly the executing speed becomes slower.

(4) When "Others" is selected in the [Group] drop-down list box

Figure 3-13 [Compiler Options] Dialog Box (When [Others] Is Selected)



Group : Others

[✓] Aggressive Optimization[-qw]

[ ] Generate relative branch table for switch[-qt]

Command Line Options :

- Aggressive Optimization[-qw]

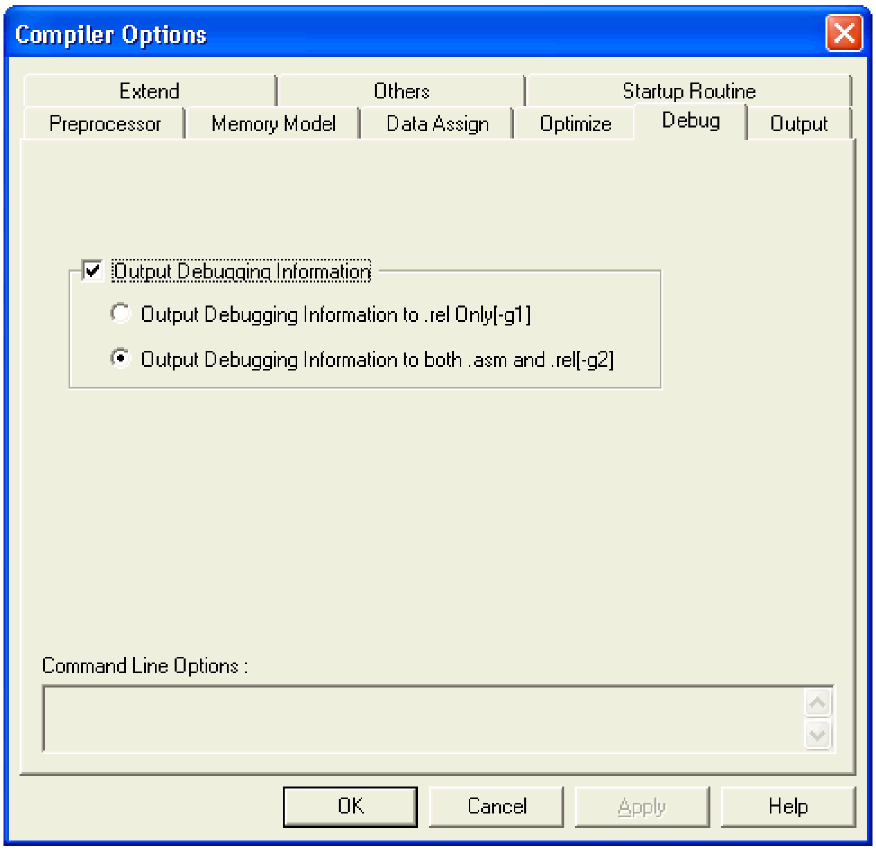
  Select this check box to perform aggressive optimization (reshuffling the execution order in an expression) (selected by default)。

- Generate relative branch table for switch[-qt]

  Select this check box to make the branch table in a switch statement of a function allocated to the far area, into of a relative branch.

[Debug] tab

Figure 3-14 [Compiler Options] Dialog Box (When [Debug] Tab Is Selected)



[✓] Output Debugging Information

( ) Output Debugging Information to .rel Only[-g1]

(●) Output Debugging Information to both .asm and .rel[-g2]

Command Line Options :

- Output Debugging Information

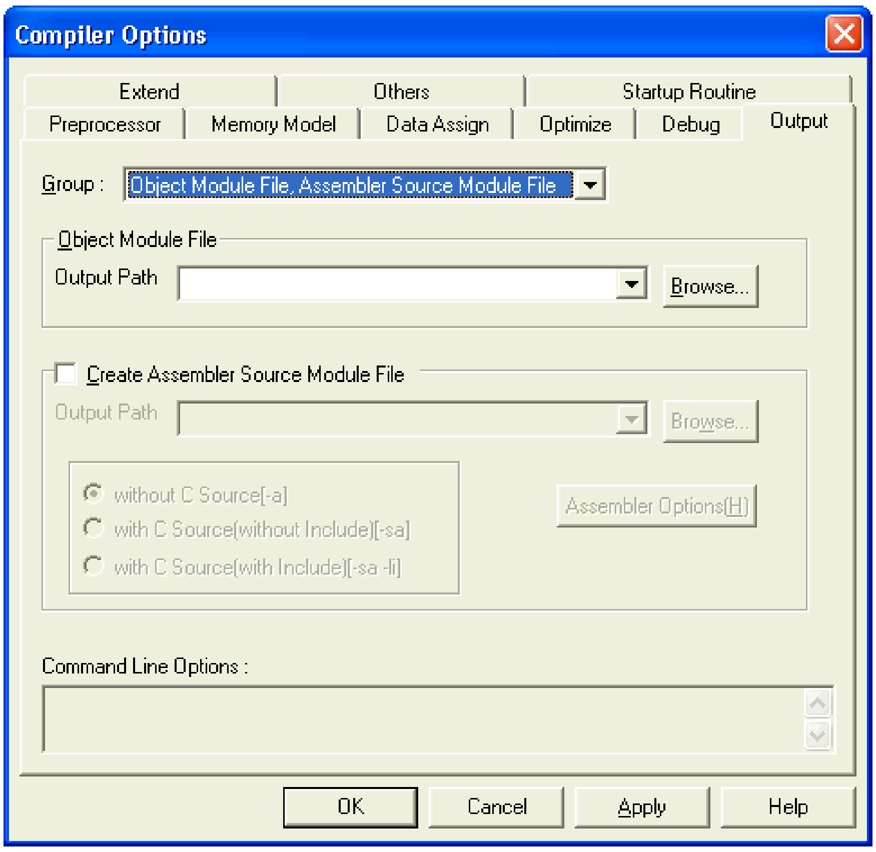
  Select this check box when outputting debug information (selected by default if "Debug Build" is selected in PM+). Specify using radio buttons the file to which debug information is to be output (“Output Debugging Information to .rel Only[-g1]” or “Output Debugging Information to both .asm and .rel[-g2]”).

  If this check box is selected, "Output Debugging Information to both .asm and .rel[-g2]" is selected by default.

[Output] tab

(1) When "Object Module File, Assembler Source Module File" is selected in the [Group] drop-down list box

Figure 3-15 [Compiler Options] Dialog Box (When [Object Module File, Assembler Source Module File] Is Selected)



- Object Module File

  To specify an object module file output path, input the path name in the combo box.

  Up to 259 characters can be input into this combo box.

  Specification is also possible using the [Browse...] button (Opens the [Browse for Folder] dialog box).

  When universal options are specified in PM+, processing is always performed assuming that the path name is specified.

  When the source file is specified, processing is performed as a path name if a path exists, and as a file name if no path exists.

- Create Assembler Source Module File

  Select this check box when outputting assembler source module files. Specify using radio buttons whether to add C sources to the assembler source module files, whether to add include files (“without C source[-a]”, “with C Source(without Include)[-sa]”, or ““with C Source(with Include)[-sa -li]”).

  If this check box is selected, "without C source[-a]" is selected by default.

To specify the output path of the assembler source module file, input the path name in the combo box. To specify a source file name, append the extension "asm".

Up to 259 characters can be input into this combo box.

Specification is also possible using the [Browse..] button (Opens the [Browse for Folder] dialog box).

When univarsal options are specified in PM+, processing is always performed assuming that the path name is specified.

When the source file is specified, processing is performed as a path name if a path exists, and as a file name if no path exists.

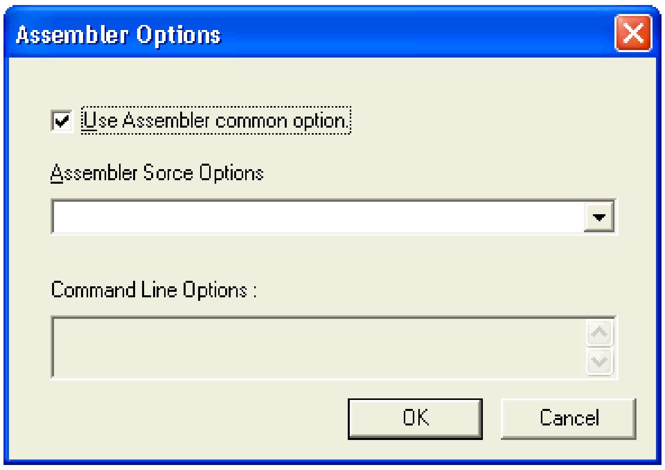
- [Assembler Options[H]] button

  Specify assembler options for the assembler source module file.

  If no option is specified, processing is performed assuming that all assembler options have been specified.

  When the [Assembler Options[H]] button under the [Output] tab in the [Compiler Options] dialog box is clicked, the following dialog box appears.

Figure 3-16 [Assembler Options] Dialog Box



- Use Assembler common option

  Select this check box to enable all the options set in the [Assembler Options] dialog box (selected by default).

- Assembler Source Options

  To enable options for the output assembler source file of the C compiler, input a character string including the option name in the combo box.

  Up to 259 characters can be input into this combo box.

  Caution Do not describe chip type specification (-c), device file specification (-y), and parameter file specification (-f) because they are set separately with this tools DLL.

- Command Line Options

  This edit box is a read-only box.

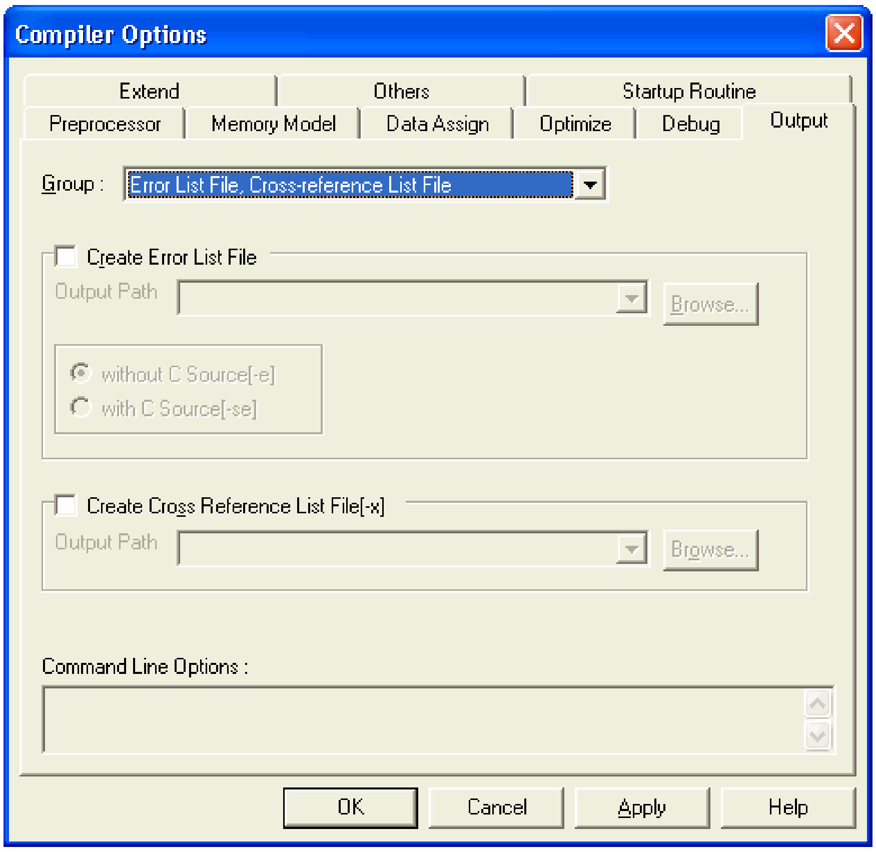
  The option character strings that are currently set are displayed.

  All of the assembler common options and assembler source options are targets.

  Option character strings that are specified with radio buttons, check boxes, or combo boxes in the option setting dialog boxes are displayed in this edit box.

(2) When "Error List File, Cross-reference List File" is selected in the [Group] drop-down list box

Figure 3-17 [Compiler Options] Dialog Box (When [Error List File, Cross-reference List File] Is Selected)



Group : Error List File, Cross-reference List File

Create Error List File

Output Path

without C Source[-e]

with C Source[-se]

Create Cross Reference List File[-x]

Output Path

Command Line Options :

- Create Error List File

  Select this check box to output the error list file. Specify using radio buttons whether to add C sources to the error list file, whether to add include files (“without C Source[-e]” or “with C Source[-se]”).

  If this check box is selected, “without C Source[-e]” is selected by default.

  To specify the error list file output path, input the path name in the combo box.

  Up to 259 characters can be input into this combo box.

  Specification is also possible using the [Browse...] button (Opens the [Browse for Folder] dialog box).

  When universal options are specified, processing is always performed assuming that the path name is specified.

  When the source file is specified, processing is performed as a path name if a path exists, and as a file name if no path exists.

- Create Cross Reference List File[-x]

  Select this check box to output the cross-reference list file.

  To specify the cross-reference list file output path, input the path name in the combo box.

  Up to 259 characters can be input into this combo box.

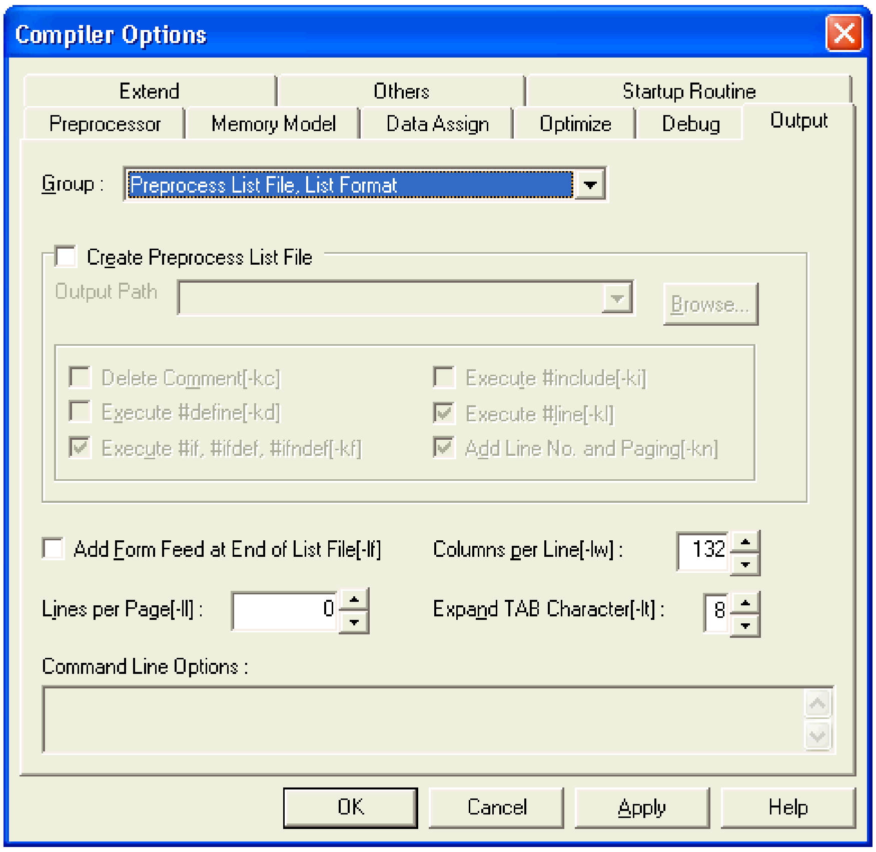
  Specification is also possible using the [Browse...] button (Opens the [Browse for Folder] dialog box).

  When universal options are specified, processing is always performed assuming that the path name is specified.

  When the source file is specified, processing is performed as a path name if a path exists, and as a file name if no path exists.

(3) When "Preprocess List File, List Format" is selected in the [Group] drop-down list box

Figure 3-18 [Compiler Options] Dialog Box (When [Preprocess List File, List Format] Is Selected)



Group : Preprocess List File, List Format

[ ] Create Preprocess List File

Output Path :

[ ] Delete Comment[-kc]
[ ] Execute #define[-kd]
[✓] Execute #if, #ifdef, #ifndef[-kf]
[ ] Execute #include[-ki]
[✓] Execute #line[-kl]
[✓] Add Line No. and Paging[-kn]

[ ] Add Form Feed at End of List File[-lf]

Columns per Line[-lw] : 132

Lines per Page[-ll] : 0

Expand TAB Character[-lt] : 8

Command Line Options :

- Create Preprocess List File

  Select this check box to output the preprocess list file and validate the specification for the following preprocess list files.

Delete Comment[-kc]	Select this check box to delete comments.
Execute #define[-kd]	Select this check box to execute #define.
Execute #if, #ifdef, #ifndef[-kf]	Select this check box to execute #if, #ifdef, #ifndef (selected by default).
Execute #include[-ki]	Select this check box to execute #include.
Execute #line[-kl]	Select this check box to execute #line (selected by default).
Add Line No. and Paging[-kn]	Select this check box to add Line No. and Paging (selected by default).

To specify the preprocess list file output path, input the path name in the combo box.

Up to 259 characters can be input into this combo box.

Specification is also possible using the [Browse...] button (Opens the [Browse for Folder] dialog box).

When universal options are specified, processing is always performed assuming that the path name is specified.

When the source file is specified, processing is performed as a path name if a path exists, and as a file name if no path exists.

- Add Form Feed at End of List File[-lf]

  Select this check box to add the new page break code at the end of each list file.

- Columns per Line[-lw]

  Specify the number of characters in 1 line of each type of list file.

  The specifiable number of characters is 0 and 72 to 132.

  132 is set by default.

- Lines per Page[-ll]

  Specify the number of lines on 1 page of each type of list file.

  The specifiable number of lines is 0 and 20 to 32,767.

  0 is set by default.

- Expand TAB Character[-lt]

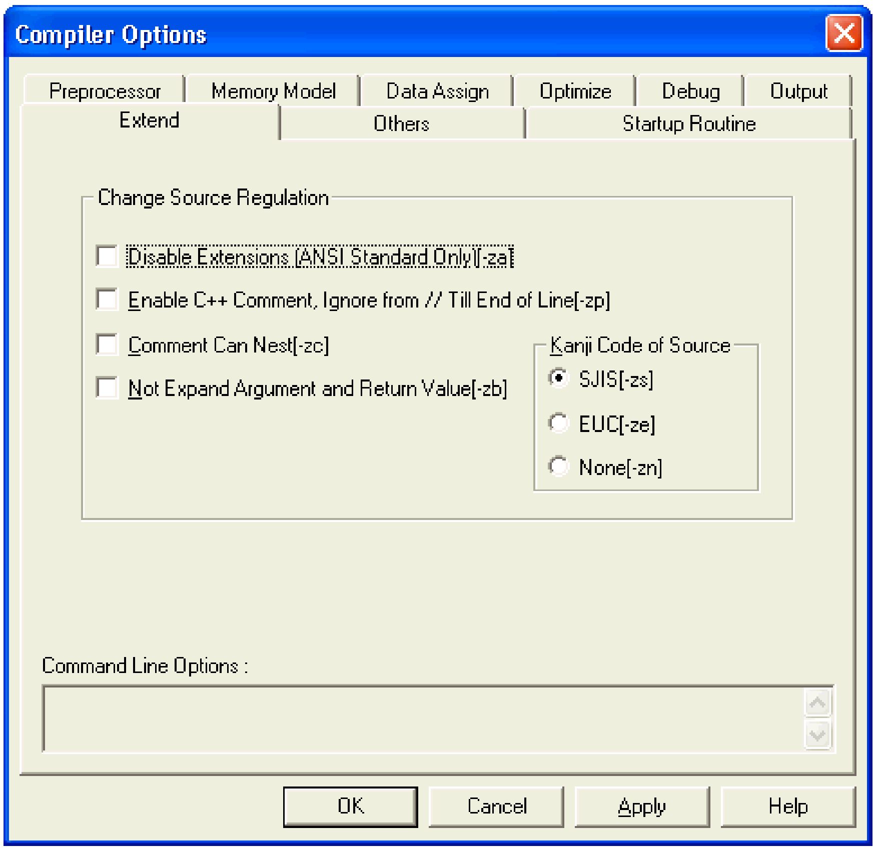
  Specify the range for the tab stop position of each type of list file.

  The specifiable range for the tab stop position is 0 to 8.

  0 is set by default.

[Extend] tab

Figure 3-19 [Compiler Options] Dialog Box (When [Extend] Tab Is Selected)



Change Source Regulation

☐ Disable Extensions (ANSI Standard Only)[-za]

☐ Enable C++ Comment, Ignore from // Till End of Line[-zp]

☐ Comment Can Nest[-zc]

☐ Not Expand Argument and Return Value[-zb]

Kanji Code of Source

● SJIS[-zs]

○ EUC[-ze]

○ None[-zn]

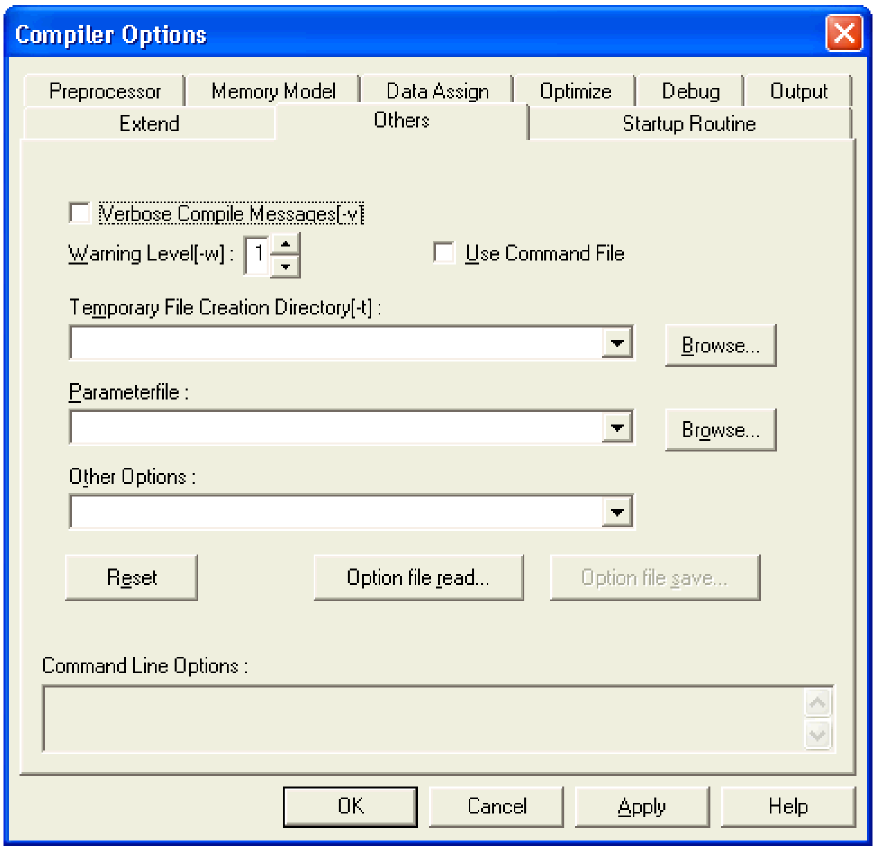
Command Line Options :

- Change Source Regulation

Disable Extensions (ANSI Standard Only)[-za]	Select this check box to disable functions not prescribed by ANSI and when enabling specific functions of ANSI.
Enable C++ Comment, Ignore from // Till End of Line[-zp]	Select this check box when the portion after "//" until the line return is interpreted as a comment.
Comment Can Nest[-zc]	Select this check box to allow nested comments.
Not Expand Argument and Return Value[-zb]	Select this check box when not expanding char/ unsigned char type arguments or return values to be int.
Kanji Code of Source	Select the type of kanji code (2-byte code) used in the comment of the source by selecting the appropriate radio button (SJIS[-zs] / EUC[-ze] / None[-zn]). "SJIS[-zs]" is selected by default.

[Others] tab

Figure 3-20 [Compiler Options] Dialog Box (When [Others] Tab Is Selected)



- Verbose Compile Messages[-v]

  Select this check box to output the execution state of the compilation to the console.

- Warning Level[-w]

  Specify the warning level.

  The specifiable range for the level is 0 to 2.

Level	Description
0	Do not output warning messages.
1 (selected by default)	Output normal warning messages.
2	Output detailed warning messages.

- Use Command File

  Select this check box to create the command file.

  The option character string is output to the command file, so awareness of restrictions on the length of the option character string is not required.

  Caution This check box cannot be performed if special compiler options are set per source file.

- Temporary File Creation Directory[-t]

  Input the folder in which to store the temporary files in the combo box.

  Only one folder can be specified in this combo box.

  Up to 259 characters can be input into this combo box.

  Specification is also possible using the [Browse...] button (Opens the [Browse for Folder] dialog box).

- Parameterfile

  Input into this combo box parameter file to which the name of an input file and options are to be input.

  Only one folder can be specified in this combo box.

  Up to 259 characters can be input into this combo box.

  Specification is also possible using the [Browse...] button (Opens the [ParameterFile] dialog box).

- Other Options

  If a compiler option other than the various option specification items must be specified, input that option in the combo box.

  Up to 259 characters can be input into this combo box.

- [Reset] button

  Clicking this button sets the default option settings.

- [Option file read...] button

  Clicking this button causes the option information file containing the option settings to be read.

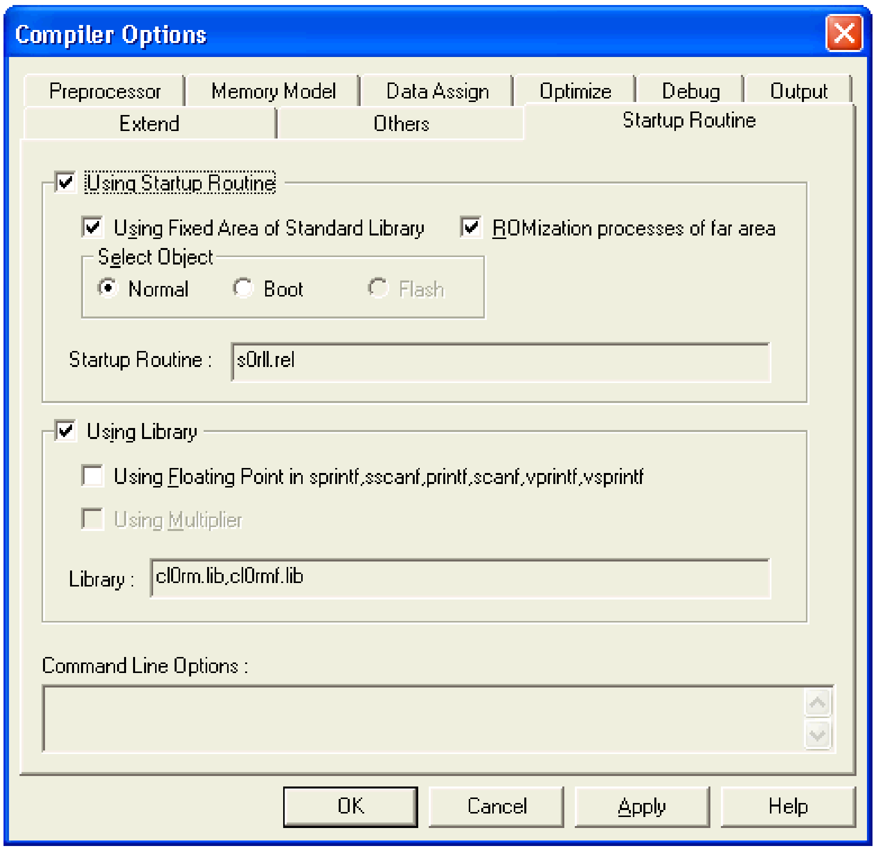
- [Option file save...] button

  Option settings are saved as an option information file.

  This button is enabled only when information has been set with the [OK] button or the [Apply] button.

[Startup Routine] tab

Figure 3-21 [Compiler Options] Dialog Box (When [Startup Routine] Tab Is Selected)



[✓] Using Startup Routine

[✓] Using Fixed Area of Standard Library [✓] ROMization processes of far area

Select Object: [● Normal] ○ Boot ○ Flash

Startup Routine : s0rll.rel

[✓] Using Library

[ ] Using Floating Point in sprintf,sscanf,printf,scanf,vprintf,vsprintf

[ ] Using Multiplier

Library : cl0rm.lib,cl0rmf.lib

Command Line Options :

Caution The settings for the [Startup Routine] tab cannot be performed if special compiler options are set per source file.

- Using Startup Routine
  Select this check box to use the standard startup routine provided for this C compiler (selected by default).

Using Fixed Area of Standard Library	Select this check box to use the fixed area used by the standard library (selected by default).
ROMization processes of far area	Select this check box to perform ROMization processing for the far area (selected by default).
Select Object	Select the desired startup routine for the normal, boot, or flash area by selecting the appropriate radio button (Normal / Boot / Flash). If the [Output the Object for Flash Memory[-zf]] check box under the [Memory Model] tab is not selected, the startup routine for the normal or boot areas can be selected, and if the check box is selected, only the startup routine for the flash area can be selected.
Startup Routine	Indicates the file name of the startup routine to be used.

- Using Library

  Select this check box to use the standard library provided for this C compiler (selected by default).

Using Floating Point in sprintf,sscanf,printf,scanf,vprintf,vsprintf	Select this check box to use the sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points.
Using Multiplier	Select this check box when using a device that has a multiplier and the multiplier is used (selected by default if a device that has a multiplier has been specified). Caution Product types that do not have a multiplier cannot be selected.
Library	Displays the file name of the library to be used.

3.2 Procedure (When Not Using Self Rewrite Mode)

3.2.1 MAKE from PM+

The MAKE method using PM+ is described below.

PM+ is a software program used for the integrated management of tools as the core of the development environment. Using PM+ enables handling application programs and environment settings as projects. Program creation using an editor, source management, compilation, and debugging can be performed as a continuous series of operations.

(1) Starting up PM+

When a development tool packages are correctly installed, the [NEC Electronics Tools] menu is created in the Programs folder displayed from the [Start] button, and PM+ and other programs are registered in this menu.

Click [PM+] from the menu to start up PM+.

(2) Creating project

Register a project first to start a series of development operations using PM+.

To register a project, first create the workspace in which that project is managed. For the procedure to create a workspace, refer to the PM+ User's Manual.

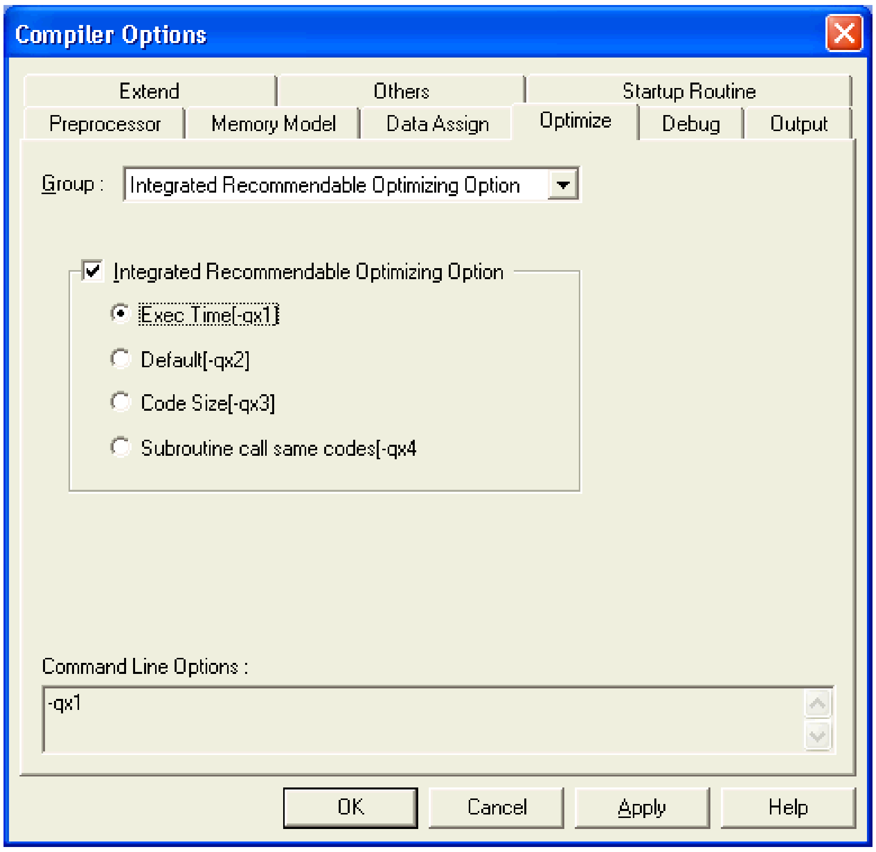
(3) Setting compiler and linker options

A minimum number of options are set for build in the MAKE file created automatically upon completion of project creation. Project-specific options are set in the [Tools] menu.

If the [Compiler Options] in the [Tools] menu is selected, the [Compiler Options] dialog box appears.

An example changing the Optimize option from default [-qcjlvw] to "Exec Time[-qx1]" is shown below.

Figure 3-22 [Compiler Options] Dialog Box (When [Optimize] Tab Is Selected)



Group : Integrated Recommendable Optimizing Option

[✓] Integrated Recommendable Optimizing Option

[● Exec Time[-qx1]]

Default[-qx2]

Code Size[-qx3]

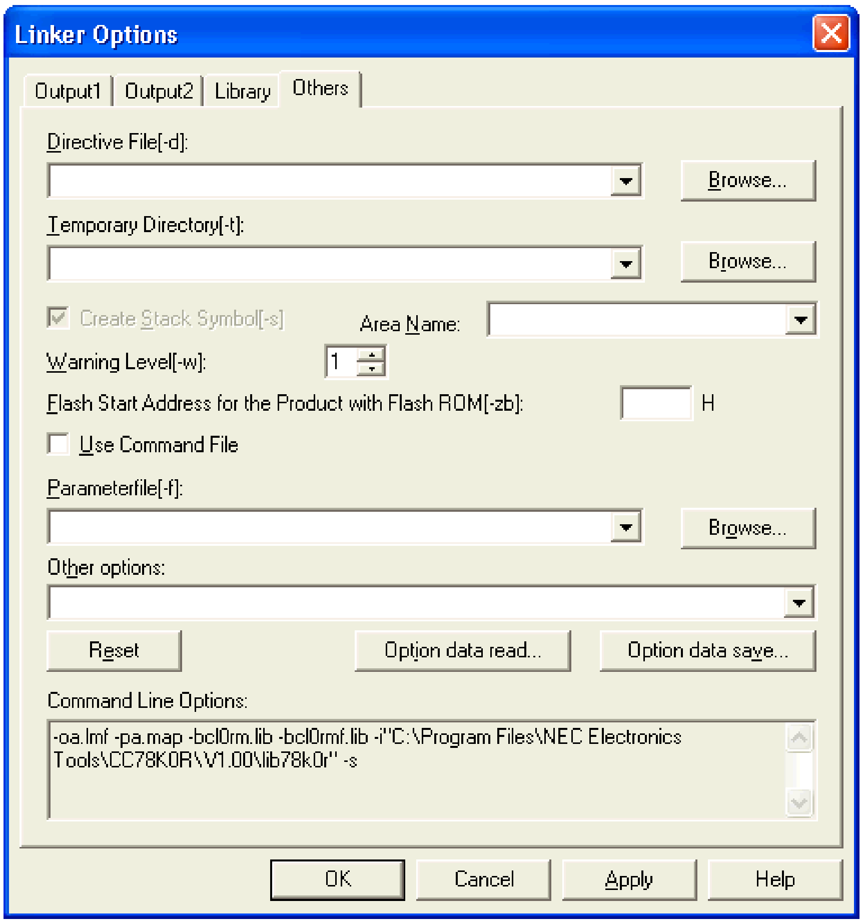
Subroutine call same codes[-qx4

Command Line Options :

```
-qx1
```

If "Using Startup Routine" is selected in the [Startup Routine] tab of the [Compiler Options] dialog box, the standard startup routine for this C compiler gets linked before all sources (not displayed to the [Linker Options] dialog box).
When "Using Library" is selected, the standard library for this C compiler gets linked behind all libraries.
If C source is included in the source file settings, stack symbol automatic generation option -s is automatically specified to the linker.
The name of the startup routine file does not affect the load module file name.

Figure 3-23 [Linker Options] Dialog Box



```
-oa.lmf -pa.map -bcl0rm.lib -bcl0rmf.lib -i"C:\Program Files\NEC Electronics
Tools\CC78K0R\V1.00\lib78k0r" -s
```

(4) Building project

Projects are built with the set options.

Building of an entire project is done by selecting [Build] from the [Build] menu, or by clicking the [Build] button on the tool bar. PM+ MAKE is started up by the automatically created MAKE file.

Upon completion of build, a message dialog box appears. Check that build has been completed normally.

Caution The contents displayed in the [OutPut] window during build are saved as the "Project file name + .plg" file name to the project folder.

3.2.2 Compiling to linking in command line (for command prompt)

(1) When parameter file is not used

The command below is used to start the CC78K0R, assembler, and linker in a command line.

Assembling is not needed when there is no assembler description in C source. In this case, link the object module file output from a C compiler (Δ : space).

```
>[path-name]cc78k0r[Δoption]ΔC-source-name[Δoption]
>[path-name]ra78k0r[Δoption]Δassembler-source-name[Δoption]
>[path-name]lk78k0r[Δoption]Δobject-module-name[Δoption]
```

Caution To link libraries created by users, be sure to specify the libraries attached to the CC78K0R and the floating point libraries at the end of the library list.

To use the sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points, specify the floating point libraries attached to the CC78K0R and the libraries attached to the compiler, in this order.

To use the sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points, specify the libraries attached to the CC78K0R and the floating point libraries attached to the compiler, in this order.

Also, specify the startup routine attached to the CC78K0R before the user programs.

The library and object module file specification order during linking is shown below.

(Library specification order)

When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points

(i) User program library file (specified with the -b option)

(ii) Library file attached to C compiler (specified with the -b option)

(iii) Floating point library file attached to C compiler (specified with the -b option)

When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points

(i) User program library file (specified with the -b option)

(ii) Floating point library file attached to C compiler (specified with the -b option)

(iii) Library file attached to C compiler (specified with the -b option)

(Specification order of other files)

(i) Object file of startup routine attached to CC78K0R

(ii) Object module file of user program

The following shows an example of linking C source s1.c and assembler source s2.asm.

```
C>cc78k0r -cf1166a0 s1.c -e -a
-i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r"
-y"C:\Program Files\NEC Electronics Tools\dev"
C>ra78k0r -cf1166a0 s2.asm -e
-y"C:\Program Files\NEC Electronics Tools\dev"
C>lk78k0r s0rll.rel s01.rel s2.rel
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rxm.lib"
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rm.lib" -s
-osample.lmf -y"C:\Program Files\NEC Electronics Tools\dev"
```

Remark When specifying multiple compiler options, delimit between compiler options by a space. It does not matter whether a description is written in uppercase or lowercase (non case sensitive). For detailed information, see "CHAPTER 5 COMPILER OPTIONS".
The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see "CHAPTER 5 COMPILER OPTIONS" and RA78K0R Assembler Package Operation User's Manual.

(2) When parameter file is used

When multiple options are input in starting a C compiler, assembler, or linker, the same specification may be repeated several times if sufficient information for startup has not been specified in the command line. In such cases, a parameter file should be used.

Specify the parameter file specification option (-f) in the command line when using a parameter file.

The following shows the startup method for a compiler, assembler, and linker by using a parameter file.

```
>[path-name]cc78k0rΔ-fparameter-file-name
>[path-name]ra78k0rΔ-fparameter-file-name
>[path-name]lk78k0rΔ-fparameter-file-name
```

The following shows a usage example.

```
C>cc78k0r -fpara.pcc
C>ra78k0r -fpara.pra
C>lk78k0r -fpara.plk
```

Parameter files are created by an editor. All options and output file names that should be specified in a command line can be written.

The following shows examples of creating parameters by the editor.

<Contents of para.pcc>

```
-cf1166a0 s1.c -e -a
-i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r"
-y"C:\Program Files\NEC Electronics Tools\dev"
```

<Contents of para.pra>

```
-cf1166a0 s2.asm -e -y"C:\Program Files\NEC Electronics Tools\dev"
```

<Contents of para.plk>

```
s0rll.rel s1.rel s2.rel
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rxm.lib"
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rm.lib" -s
-osample.lmf -y"C:\Program Files\NEC Electronics Tools\dev"
```

The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see "CHAPTER 5 COMPILER OPTIONS" and RA78K0R Assembler Package Operation User's Manual.

3.3 Procedure (When Using Self Rewrite Mode)

This function is available only for the device having the flash memory self rewriting function.

3.3.1 Compiling to linking via PM+

PM+ is used to illustrate the make technique.

Be sure to execute compiling to linking in the following order.

(1) Compiling to linking program for boot area

(a) Creating a project
Create a project for the boot area and register the source file.

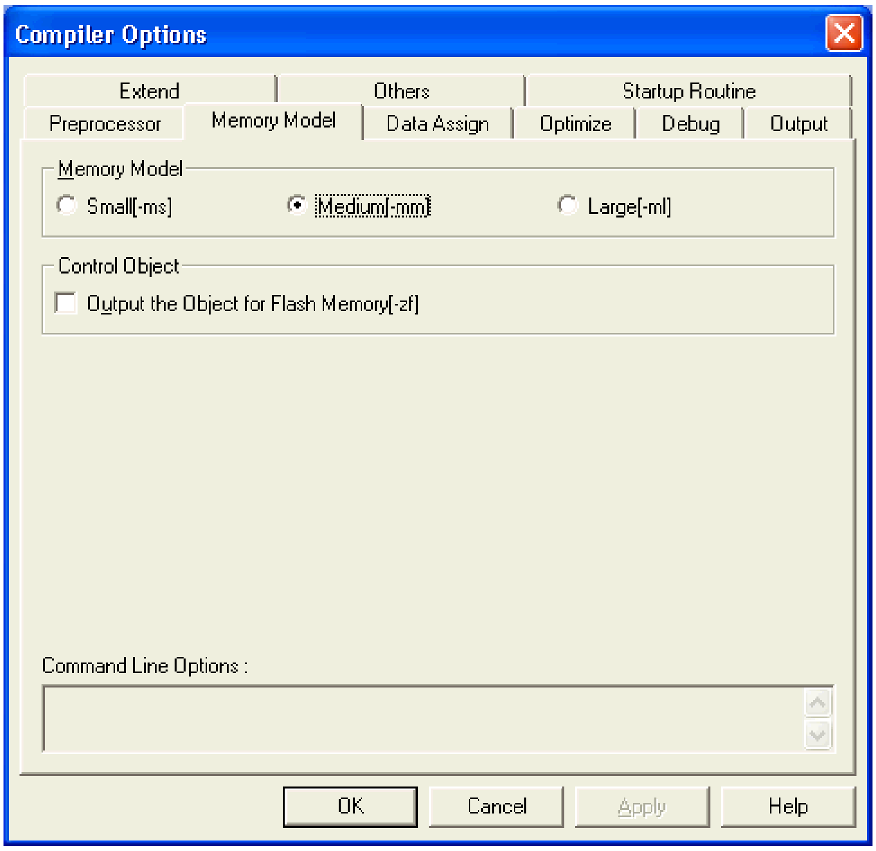
(b) Compiler, linker, and object converter options settings

Only the minimum options required for build are set in MAKE file automatically created when project creation is ended. Project-specific options are set with the [Tools] menu.

Selecting [Compiler Options] in the [Tools] menu displays the [Compiler Options] dialog box.

(i) Setting compiler option

Do not specify the [Output the Object for Flash Memory[-zf]] check box under the [Memory Model] tab.

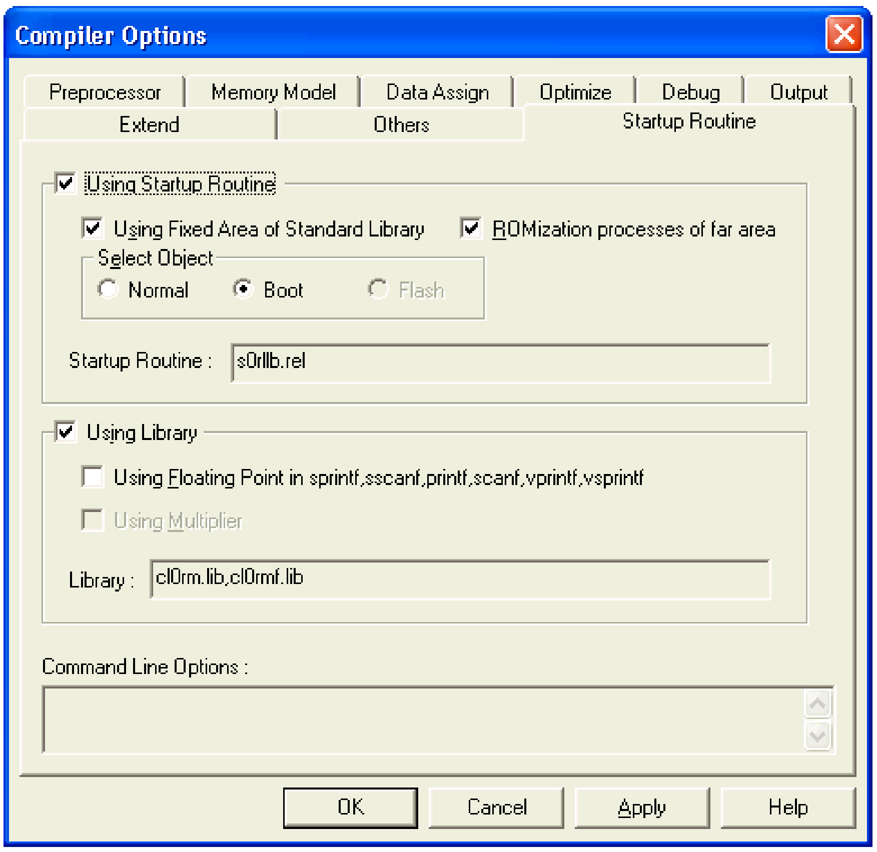


Memory Model
Small[-ms] [✓ Medium[-mm]] Large[-ml]

Control Object
☐ Output the Object for Flash Memory[-zf]

Command Line Options :

Select [Boot] radio button in the [Select Object] box under the [Startup Routine] tab.



[✓] Using Startup Routine

[✓] Using Fixed Area of Standard Library [✓] ROMization processes of far area

Select Object: Normal [✓ Boot] Flash

Startup Routine : s0rllb.rel

[✓] Using Library

[ ] Using Floating Point in sprintf,sscanf,printf,scanf,vprintf,vsprintf

[ ] Using Multiplier

Library : cl0rm.lib,cl0rmf.lib

Command Line Options :

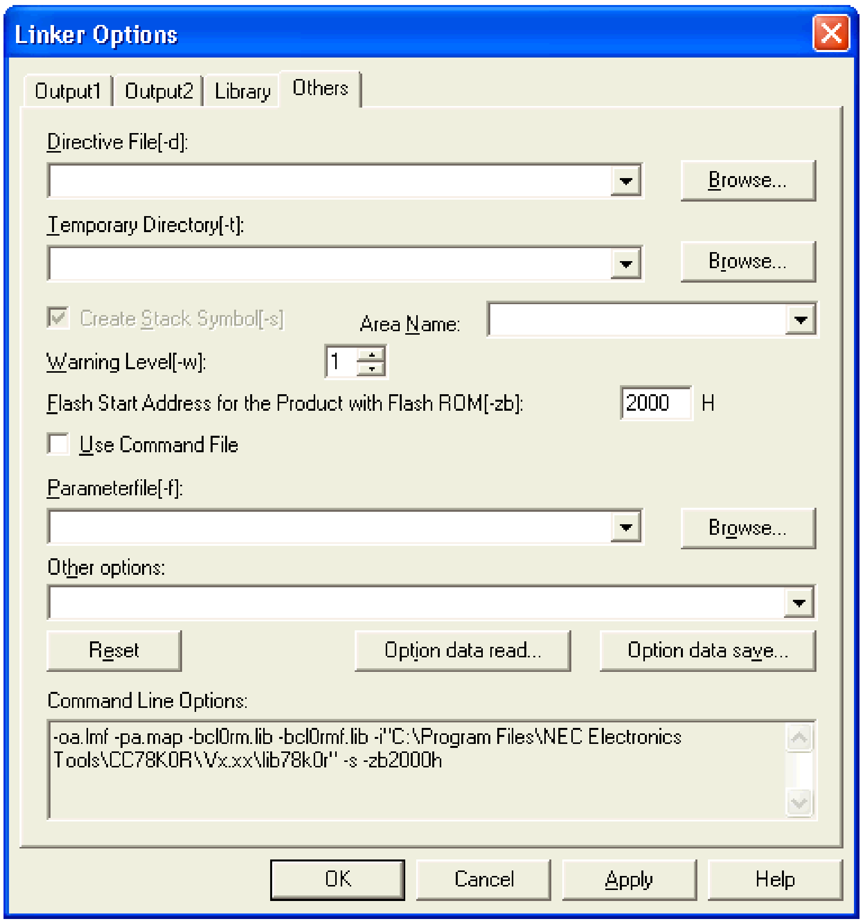
(ii) Setting linker option

Specify "Flash Start Address for the Product with Flash ROM[-zb]" and then click the [OK] button.

Since "Using Startup Routine" and "Using Library" check boxes are selected under the [Startup Routine] tab, it is not necessary to specify the startup routine and library in the [Linker Options] dialog box.

Also, since the C source (boot.c) is included in the source file specification, "Create Stack Symbol[-s]" option is automatically set.

Remark For information about the linker options, refer to RA78K0R Assembler Package Operation User's Manual.



[✓] Create Stack Symbol[-s]

Warning Level[-w]: 1

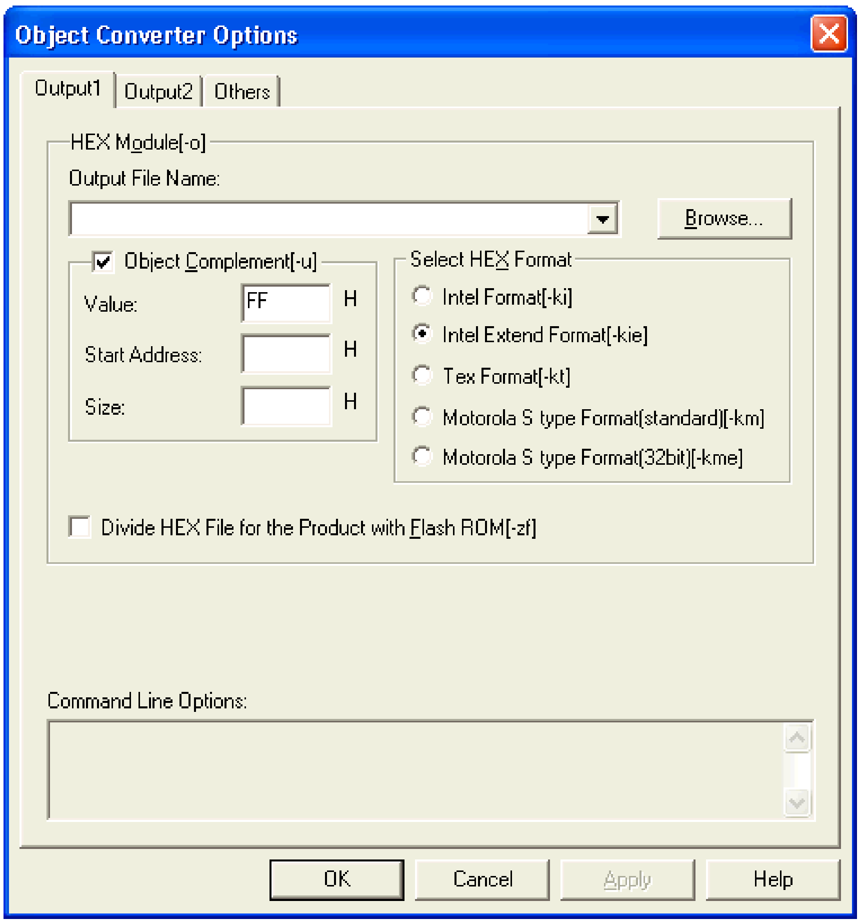
Flash Start Address for the Product with Flash ROM[-zb]: 2000 H

Command Line Options:

```
-oa.lmf -pa.map -bcl0rm.lib -bcl0rmf.lib -i"C:\Program Files\NEC Electronics
Tools\CC78K0R\Vx.xx\lib78k0r" -s -zb2000h
```

(iii) Setting object converter option

Do not specify the [Divide HEX File for Product with Flash ROM[-zf]] check box.


Object Converter Options

Output1 | Output2 | Others

HEX Module[-o]

Output File Name:

Browse...

[✓] Object Complement[-u]

Value: FF H

Start Address: H

Size: H

Select HEX Format

( ) Intel Format[-ki]

(●) Intel Extend Format[-kie]

( ) Tex Format[-kt]

( ) Motorola S type Format(standard)[-km]

( ) Motorola S type Format(32bit)[-kme]

[ ] Divide HEX File for the Product with Flash ROM[-zf]

Command Line Options:

OK | Cancel | Apply | Help

Caution After the program for boot area is compiled and object-converted, write in the HEX file (e.g. boot.hex) with a flash programmer. After writing, be sure to save the load module file (e.g. boot.lmf) and HEX file created in the above procedure. Do not build the program for boot area again.

(c) Building project

Projects are built with the set options.

Build of an entire project is done by selecting [Build] from the [Build] menu, or by clicking the [Build] button on the tool bar. PM+ MAKE is started up by the automatically created MAKE file.

Upon completion of build, a message dialog box appears. Check that build has been completed normally.

Caution The contents displayed in the [OutPut] window during build are saved as the "Project file name + .plg" file name to the project folder.

(2) Compiling to linking program for flash area

(a) Creating a project
Create a project for the flash area and register the source file.

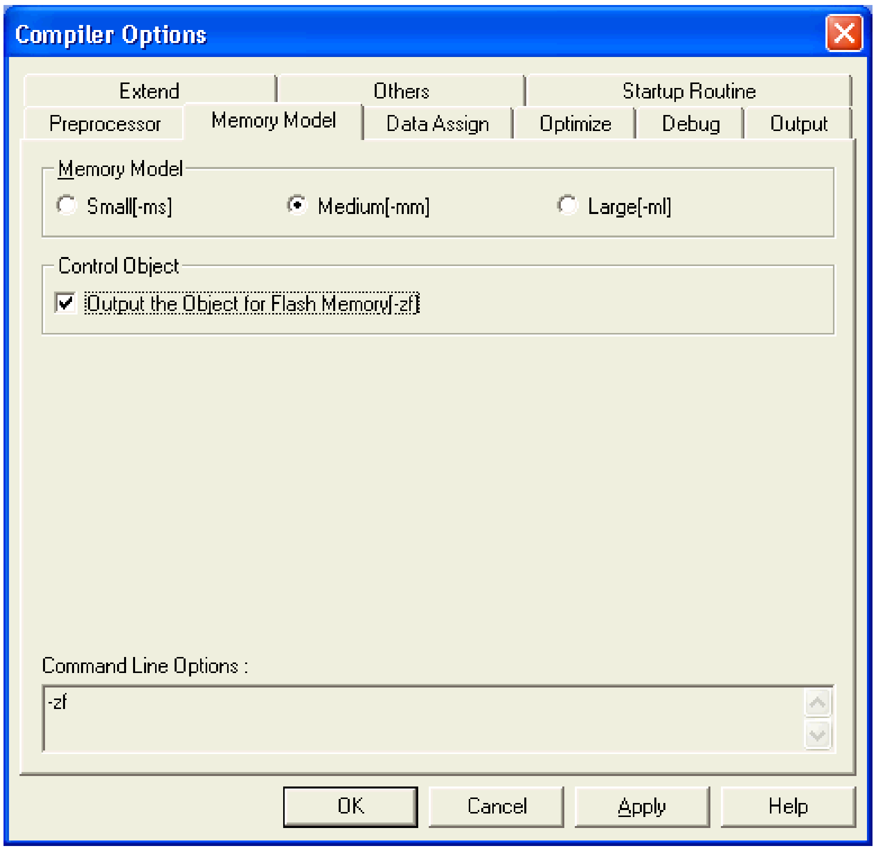
(b) Compiler, linker, and object converter option settings

Only the minimum options required for build are set in MAKE file automatically created when project creation is ended. Project-specific options are set with the [Tools] menu.

Selecting [Compiler Options] in the [Tools] menu displays the [Compiler Options] dialog box.

(i) Setting compiler option

Specify the [Output the Object for Flash Memory[-zf]] check box under the [Memory Model] tab.


Compiler Options

Memory Model: Small[-ms] [✓ Medium[-mm]] Large[-ml]

Control Object: [✓ Output the Object for Flash Memory[-zf]]

Command Line Options :

-zf

Select [Flash] radio button in the [Select Object] box under the [Startup Routine] tab.

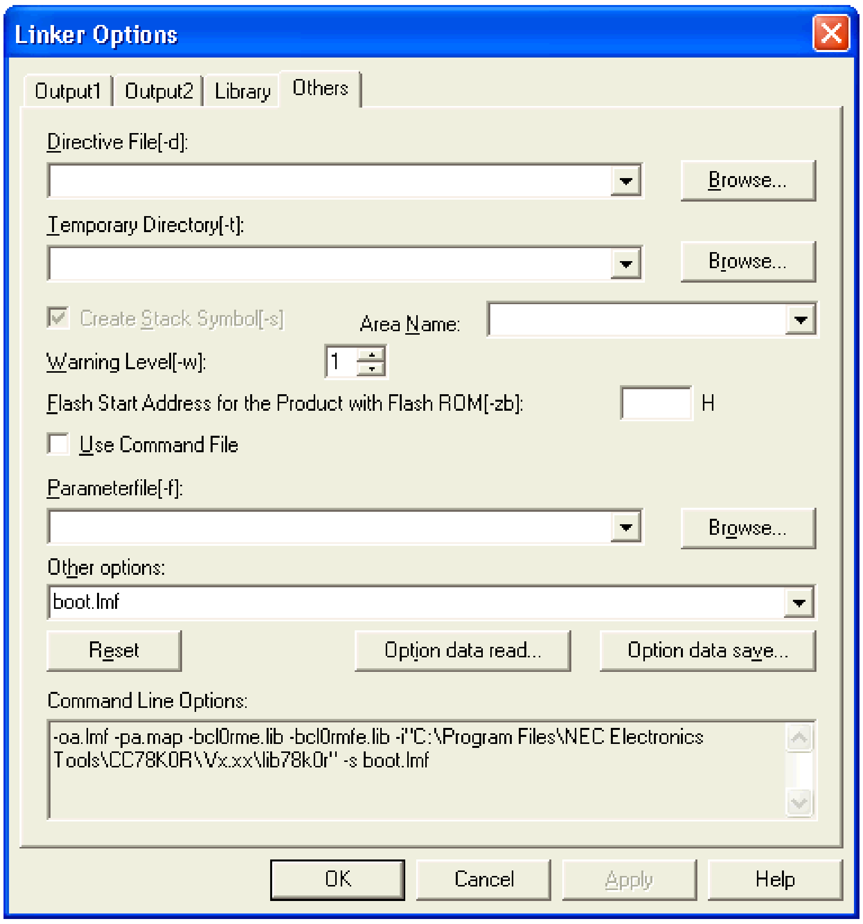
(ii) Setting linker option

Specify the boot area load module file that was created in the "Other options" combo box.

Since the "Using Startup Routine" and "Using Library" check boxes are selected under the [Startup Routine] tab in the [Compiler Options] dialog box, it is not necessary to specify the startup routine and library in the [Linker Options] dialog box.

Also, since the C source (flash.c) is included in the source file specification, "Create Stack Symbol[-s]" option is automatically set.

Remark For information about the linker options, refer to RA78K0R Assembler Package Operation User's Manual.



Directive File[-d]:

Temporary Directory[-t]:

☑ Create Stack Symbol[-s] Area Name:

Warning Level[-w]: 1

Flash Start Address for the Product with Flash ROM[-zb]: H

☐ Use Command File

Parameterfile[-f]:

Other options: boot.lmf

Command Line Options:

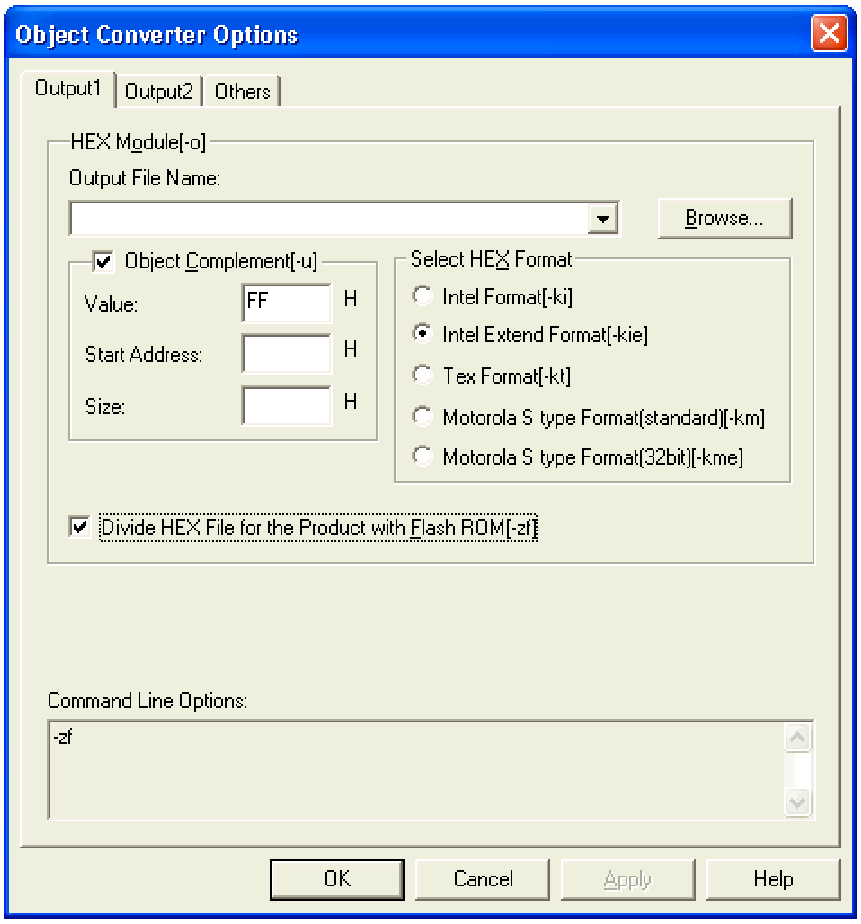
-oa.lmf -pa.map -bcl0rme.lib -bcl0rmfe.lib -i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r" -s boot.lmf

(iii) Setting object converter option (for flash area)

Be sure to specify the [Divide HEX File for Product with Flash ROM[-zf]] check box.

By specifying the -zf option, the HEX file for boot area (e.g. flash.hxb) and the HEX file for flash area (e.g. flash.hxf) are output.

The flash.hxb and the boot.hex that is generated when the program for boot area is built have the same contents. However, when the HEX file for boot area is already written and the program for flash area is built again, it is recommended to confirm that there is no difference in the saved boot.hex and the created flash.hxb.


Object Converter Options

Output1 | Output2 | Others

HEX Module[-o]

Output File Name:

[✓] Object Complement[-u]

Value: FF H

Start Address: H

Size: H

Select HEX Format

○ Intel Format[-ki]

● Intel Extend Format[-kie]

○ Tex Format[-kt]

○ Motorola S type Format(standard)[-km]

○ Motorola S type Format(32bit)[-kme]

[✓] Divide HEX File for the Product with Flash ROM[-zf]

Command Line Options:

```
-zf
```

(c) Building project

Projects are built with the set options.

Build of an entire project is done by selecting [Build] from the [Build] menu, or by clicking the [Build] button on the tool bar. PM+ MAKE is started up by the automatically created make file.

Upon completion of build, a message dialog box appears. Check that build has been completed normally.

Caution The contents displayed in the [OutPut] window during build are saved as the "Project file name + .plg" file name to the project folder.

3.3.2 Compiling to linking in command line (for command prompt)

(1) When parameter file is not used

The command below is used to start the CC78K0R, assembler, and linker in a command line.

Assembling is not needed when there is no assembler description in the C source. In this case, link the object module file output from a C compiler (Δ: space).

```
>[path-name]cc78k0r[Δoption]ΔC-source-name[Δoption]
>[path-name]ra78k0r[Δoption]Δassembler-source-name[Δoption]
>[path-name]lk78k0r[Δoption]ΔObject-module-name,etc.[Δoption]
```

The following shows examples of compiling and linking the C source for boot area and the C source for flash area.

(a) Compiling to linking, object-converting program for boot area

<Example 1: Compiling program for boot area>

```
C>cc78k0r -cf1166a0 boot.c
-i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r"
-y"C:\Program Files\NEC Electronics Tools\dev"
```

<Example 2: Linking program for boot area>

```
C>lk78k0r s0rllb.rel boot.rel
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rxm.lib"
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rm.lib"
-s -oboot.lmf -zb2000h -y"C:\Program Files\NEC Electronics Tools\dev"
```

<Example 3: Object-converting program for boot area>

```
C>oc78k0r boot.lmf -oboot.lmf -y"C:\Program Files\NEC Electronics Tools\dev"
```

Caution After the program for boot area is compiled and object-converted, write in the HEX file (e.g. boot.hex) with a flash programmer. After writing, be sure to save the load module file (e.g. boot.lmf) and the HEX file created in the above procedure. Do not build the program for boot area again.

(b) Compiling to linking program for flash area

<Example 1: Compiling program for flash area>

```
C>cc78k0r -cf1166a0 flash.c -zf
-i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r"
-y"C:\Program Files\NEC Electronics Tools\dev"
```

<Example 2: Linking program for flash area>

```
C>lk78k0r boot.lmf s0lle.rel flash.rel
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rxm.lib"
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rm.lib"
-s -oflash.lmf -y"C:\Program Files\NEC Electronics Tools\dev"
```

<Example 3: Object-converting program for flash area>

```
C>oc78k0r flash.lmf -oflash.lmf -y"C:\Program Files\NEC Electronics Tools\dev"
```

Caution By specifying the -zf option when object-converting, the HEX file for boot area (e.g. flash.hxb) and the HEX file for flash area (e.g. flash.hxf) are output. The flash.hxb and the boot.hex that is generated when the program for boot area is built have the same contents. However, when the HEX file for boot area is already written and the program for flash area is built again, it is recommended to confirm that there is no difference in the saved boot.hex and the created flash.hxb.

Remark When specifying multiple compiler options, delimit between compiler options by a space. It does not matter whether a description is written in uppercase or lowercase (non case sensitive). For detailed information, see "CHAPTER 5 COMPILER OPTIONS".

The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see "CHAPTER 5 COMPILER OPTIONS" and RA78K0R Assembler Package Operation User’s Manual.

Caution When linking a library created by a user or a floating-point library, be sure to specify the library attached to the CC78K0R at the end of the library line. When linking a program for flash area and a program for boot area, specify the load module file for boot area in the beginning, and specify the startup routine for flash area before the user program.

The following shows the library and object module file specification orders when linking.

(Library specification order)

- When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points

  (i) User program library file (specified with the -b option)
  (ii) Library file attached to the CC78K0R (specified with the -b option)
  (iii) Floating point library file attached to the CC78K0R (specified with the -b option)

- When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points

  (i) User program library file (specified with the -b option)
  (ii) Floating point library file attached to the CC78K0R (specified with the -b option)
  (iii) Library file attached to the CC78K0R (specified with the -b option)

Caution Specify the library for boot area when linking the program for boot area, and the library for flash area when linking the program for flash area.

(Specification order of other files)

  (i) Load module file for boot area of user program
  (ii) Startup routine object module file for flash area attached to the CC78K0R
  (iii) Object module file for flash area of user program

(2) When parameter file is used

When multiple options are input in starting a C compiler, assembler, or linker, the same specification may be repeated several times if sufficient information for startup has not been specified in the command line. In such cases, a parameter file should be used.

Specify the parameter file specification option (-f) in the command line when using a parameter file.

The following shows the startup method for a compiler, assembler, and linker by using a parameter file.

```
>[path-name]cc78k0rΔ-fparameter-file-name
>[path-name]ra78k0rΔ-fparameter-file-name
>[path-name]lk78k0rΔ-fparameter-file-name
```

The following shows a usage example.

```
C>cc78k0r -fpara.pcc
C>lk78k0r -fpara.plk
```

Parameter files are created by Editor. All options and output file names that should be specified in a command line can be written.

The following shows examples of creating parameters by Editor.

<Contents of para.pcc>

```
-cf1166a0 boot.c
-i"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r"
-y"C:\Program Files\NEC Electronics Tools\dev"
```

<Contents of para.pra>

```
s0rllb.rel boot.rel
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rxm.lib"
-b"C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r\cl0rm.lib"
-s -oboot.lmf -zb2000h
-y"C:\Program Files\NEC Electronics Tools\dev"
```

Remark The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see "CHAPTER 5 COMPILER OPTIONS" and RA78K0R Assembler Package Operation User’s Manual.

3.4 I/O Files of C Compiler

The CC78K0R inputs the C source module files written in the C language. These are converted into machine language and output as object module files.

After compiling, the assembler source module files are output so that the user can check and revise the contents at the assembly language level. Based on the compiler options, the list files such as the preprocess list, cross-reference list, and error list are output.

If there is a compiler error, the error message is output to the console and the error list file. If errors occur, various files other than an error list file cannot be output.

The CC78K0R I/O files are shown below.

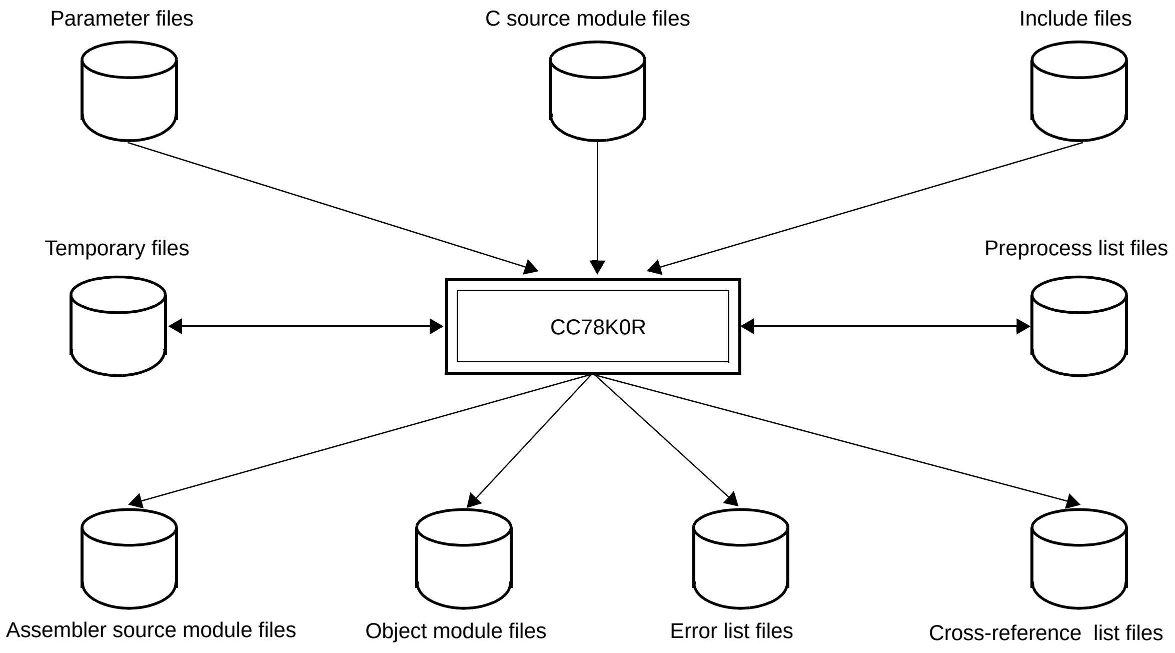
Table 3-1 C Compiler I/O Files

Type	File Name	Description	Default File Type
Input Files	C source module file	- Source file written in the C language (File created by the user)	c
	Include file	- File referenced by a C source module file (File written in the C language) - File created by the user	h
	Parameter file	- File created by the user when the user wants to specify multiple commands that cannot be specified in the command line when the C compiler is run	pcc
Output Files	Object module file	- Binary image file containing machine language information, relocatable information related to the location address of the machine language, and symbol information	rel
	Assembler source module file	- ASCII image file of the object code output by the compiler	asm
	Preprocess list file	- List file output by the preprocess instructions such as #include - ASCII image file	ppl
	Cross-reference list file	- List file containing the function name and variable name information used in the C source module file	xrf
	Error list file	- List file containing the source file and compiler error messages	ecc cer her er^{Note}
I/O File	Temporary file	- Intermediate file for compiling - The file is renamed to an appropriate name when compiling ends without error and is deleted when compiling ends in error.	$nn (file name fixed)

Note The following 4 file types are available for error list files.

File Types	Description
cer	Error list files with C source corresponding to *.c' files (output by specifying the -se option)
her	Error list files with C source corresponding to *.h' files (output by specifying the -se option)
er	Error list files with C source corresponding to files other than the above (output by specifying the -se option)
ecc	Error list files without C source corresponding to all of the source files (output by specifying the -se option)

Figure 3-24 C Compiler I/O Files



Remark If there are compiling errors, a variety of files other than the error list and cross reference files cannot be output.
A temporary file is renamed to an appropriate name when the compiling ends without error. If compiling ends in error, the temporary files are deleted.

3.5 Execution Start and End Messages

3.5.1 Execution start message

When the CC78K0R starts, the execution start message is displayed on the console.

```
78K0R C Compiler Vx.xx  [xx xxx xxxx]
     Copyright(C) NEC Electronics Corporation xxxx, xxxx
```

3.5.2 Execution end message

If compiler errors were not detected in the compilation result, the CC78K0R outputs the following message to the console and returns control to the operating system.

```
Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 0 warning(s) found.
```

If compiler errors were detected in the compilation result, the CC78K0R outputs the error messages and the number of errors to the console and returns control to the operating system.

```
prime.c( 18 ) : CC78K0R warning W0745 : Expected function prototype
prime.c( 20 ) : CC78K0R warning W0745 : Expected function prototype
prime.c( 26 ) : CC78K0R warning W0622 : No return value
prime.c( 37 ) : CC78K0R warning W0622 : No return value
prime.c( 44 ) : CC78K0R warning W0622 : No return value

Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 5 warning(s) found.
```

If a fatal error was detected where the compiling process cannot continue during compilation, the compiler outputs a message to the console, stops compilation, and returns control to the operating system.

An example that outputs an error is shown below.

```
78K0R C Compiler Vx.xx  [xx xxx xxxx]
    Copyright(C) NEC Electronics Corporation xxxx, xxxx

CC78K0R error F0018 : Option is not recognized '-s'
Please enter 'CC78K0R  --', if you want help messages.
Program aborted.
```

In this example, since a nonexistent compiler option (-s) was input, an error results and the compiler stops.

If the CC78K0R outputs error messages and stops the compilation, find the sources of these error messages in "CHAPTER 9 ERROR MESSAGES" and correct.

CHAPTER 4 CC78K0R FUNCTIONS

4.1 Optimization Method

Optimization is performed to create efficient object module files in the CC78K0R.

The table below lists the supported optimization methods.

Table 4-1 Optimization Methods

Phase	Contents	Example
Syntax		
(1)	Execute during constant computations compilation	`a = 3 * 5 ;     --> a = 15 ;`

(2)	True or false decision based on partial evaluation of a logical expression	`0 && ( a || b ) --> 0`
		`1 || ( a && b ) --> 1`

(3)	Offset calculations of pointers, arrays, etc.	Calculate the offsets during compilation.
Code Generator		
(4)	Register management	Effectively use unused registers.
(5)	Use the special instructions of the target CPU.	`a = a + 1 ;     --> Use the inc instruction.` Use the move instruction to substitute array elements.
(6)	Use short instructions.	If there is an instruction with the same operation, use the instruction with fewer bytes. `mov a , #0      --> clrb a`
(7)	Change long jump instructions to short jump instructions.	The intermediate code that was output is reprocessed.
Optimizer		

(8)	Delete common partial expressions.	`a = b + c ;     --> a = b + c ;`
		`d = b + c + e ;     d = a + e ;`

(9)	Move outside an instruction loop.	`for ( i = 0 ; i < 10 ; i++ )`
		`{`
		`        :`
		`    a = b + c ;`
		`        :`
		`}`
		↓
		`a = b + c ;`
		`for ( i = 0 ; i < 10 ; i++ )`
		`{`
		`        :`
		`}`

(10)	Delete unused instructions.	`a = a ;                               --> Delete`
		`After a = b ; , a is not referenced --> Delete`
		(a is an automatic variable)

Table 4-1 Optimization Methods

Phase	Contents	Example

(11)	Delete copies.	`a = b ;`
		`c = a + d ;      --> c = b + d ;`
		a is not referenced any more (a is an automatic variable).

(12)	Change the calculation order in an expression.	The calculation whose result remains in the register as valid before other calculations is executed.
(13)	Memory device allocation (temporary variables)	Variables used locally are allocated to registers.
(14)	Peephole optimization	Replacement of special patterns `Examples  a * 1 --> a , a + 0 --> a`
(15)	Decrease the strength of the calculation.	`Examples  a * 2 --> a + a , a << 1`
(16)	Memory device allocation (register variables)	Data is allocated to rapidly accessible memory. Examples Registers, saddr (only when the -qr option is specified)
(17)	Jump optimization (the -qj option)	Consecutive jump instructions are combined into 1 instruction.
(18)	Register allocation (the -qv, -qr, -rd, -rs options)	Variables are automatically allocated to registers.

Remark (1) to (7), (14), and (15) are performed regardless of the optimization option specifications.

The optimizations in (8) to (13), (17), and (18) are performed when optimization options are specified.

(16) is performed when there are register declarations in the C source program. However, the saddr area is only allocated when the -qr option is specified.

For information about the optimization options, see "CHAPTER 5 COMPILER OPTIONS".

4.2 ROMization Function

ROMization means that the initial values, such as the initial values of external variables, are placed in the ROM. These values are copied to RAM when the system is executed.

The CC78K0R provides startup routines with the processing of programs in ROM as samples. For ROMization, using the startup routines in ROM eliminates the problem of describing ROMization processes for startup.

For information about the startup routines, see "8.3 Startup Routines".

How to store a program on ROM is described below.

4.2.1 Linking

During linking, the startup routine, object module files, and libraries are linked. The startup routine initializes the object program.

- s0r*.rel

  Startup routine (when stored on ROM)

  The copy routine for the initialization data is included, and the beginning of the initial data is indicated.

  The label "_@cstart" (symbol) is added to the start address.

- cl0r*.lib

  Library attached to CC78K0R.

  The library files of the CC78K0R include the following 2 libraries.

  (1) Runtime library

  "@@" is added to the symbol head of the runtime library name. For the special library cprep, cdisp, however, "_@" is added to the symbol head.

  (2) Standard library

  "_" is added to the symbol head of the standard library name.

- *.lib

  Library created by a user.

  "_" is added to the symbol head.

  Caution The CC78K0R provides various kinds of startup routines and libraries. For details of startup routine, refer to "CHAPTER 8 STARTUP ROUTINES". For details of libraries, refer to "2.5.1 Library files".

CHAPTER 5 COMPILER OPTIONS

When the CC78K0R is started, the compiler options can be specified. The compiler options provide instructions for the CC78K0R operation and indicate the information required beforehand in program execution.

The compiler options are not only specified individually, but multiple options can also be simultaneously specified. The user selects the compiler options to match the objectives and to perform the tasks efficiently.

5.1 Specifying Compiler Options

Compiler options can be specified in the following ways.

- Specified in the command line when the CC78K0R starts.
- Specified in the [Compiler Options] dialog box of PM+.
- Specified in the parameter file.

For the specification methods for the compiler options described above, see "CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING".

Specify the suboption or file name after a compiler option without inserting a blank, such as a space. Spaces are required between the compiler options.

Uppercase characters and lowercase characters are not distinguished for the compiler options.

<Example>

```
cc78k0rΔ-cf1166a0Δprime.cΔ-aprime.asmΔ-qx2
```

Remark Δ: blanks such as spaces

5.2 Prioritization

For the compiler options shown in the following table, the prioritization is explained in a case where two or more options along the vertical axis and options along the horizontal axis are simultaneously specified.

Table 5-1 Prioritization of Compiler Options

	-no	-p	-np	-d	-u	-a	-e	-x	-sa
-r	NG	-	-	-	-	-	-	-	-
-q	NG	-	-	-	-	-	-	-	-
-g	NG	-	-	-	-	-	-	-	-
-k	-	Δ	NG	-	-	-	-	-	-
-d	-	-	-	-	OK	-	-	-	-
-u	-	-	-	OK	-	-	-	-	-
-sa	-	-	-	-	-	NG	-	-	-
-lw	-	Δ	-	-	-	Δ	Δ	Δ	-
-ll	-	Δ	-	-	-	Δ	Δ	Δ	-
-lt	-	Δ	-	-	-	Δ	Δ	Δ	-
-lf	-	Δ	-	-	-	Δ	Δ	Δ	-
-li	-	-	-	-	-	-	-	-	Δ

[Location marked by NG]

If an option in the horizontal axis is specified, the option in the vertical axis becomes invalid.

<Example>

```
C>cc78k0r -cf1166a0 -e sample.c -no -rd -g
```

The -rd and -g options become invalid.

[Location marked by Δ]

If an option in the horizontal axis is not specified, the option in the vertical axis becomes invalid.

<Example>

```
C>cc78k0r -cf1166a0 -e sample.c -p -k
```

Since the -p option is specified, the -k option is valid.

[Location marked by OK]

The option specified last out of an option in the horizontal axis and an option in the vertical axis has priority.

<Example>

```
C>cc78k0r -cf1166a0 -e sample.c -utest -dtest=1
```

Since the -d option is specified last, the -u option becomes invalid, and the -d option has priority.

As with the -o and -no options, the option specified last has priority even if n can be added before the option name.

<Example>

```
C>cc78k0r -cf1166a0 -e sample.c -o -no
```

Since the -no option is specified last, the -o option becomes invalid, and the -no option has priority.

Options not described in Table 5-1 are not particularly affected by other options. However, if the help specification option (--/-?-h) was specified, all of the option specifications become invalid.

The help specification option (--/-?-h) cannot be specified in PM+. To reference help in PM+, click the [Help] button in each option dialog box of PM+.

5.3 Types

Compiler options are categorized into the following types.

Table 5-2 List of Compiler Options

Types	Option	Description
Device type specification	-c	Specifies the type of target device.
Object module file creation specification	-o	Specifies the output of the object module files.
	-no	
Memory assignment specification	-r	Specifies the method of memory assignment.
	-nr	
	-rd	Specifies the automatic assignment of an external variable/external static variable (except for the const-type variable) to the saddr area.
	-nr	
	-rs	Specifies the automatic assignment of a static auto variable to the saddr area.
	-nr	
Optimization specification	-q	Specifies optimization types.
	-nq	
Debugging information output specification	-g	Specifies the output of the C source level debugging information.
	-ng	
Preprocess list file creation specification	-p	Specifies the output of the preprocess list files.
	-k	Specifies processing for the preprocess lists.
Preprocess specification	-d	Performs macro definitions.
	-u	Invalidates macro definitions.
	-i	Reads from the folder that is specified as the include file.
Assembler source module file creation specification	-a	Specifies the output from the assembler source module files.
	-sa	
Error list file creation specification	-e	Specifies the output from the error list files.
	-se	
Cross-reference list file creation specification	-x	Specifies the output from the cross reference list files.

Table 5-2 List of Compiler Options

Types	Option	Description
List format specification	-lw	Specifies number of characters for 1 line of each list file.
	-ll	Specifies number of lines for 1 page of each list file.
	-lt	Changes the expanded number of characters for each list file tab.
	-lf	Adds the page break code at the end of the list files.
	-li	Adds the C source of the include files to the assembler source module file with C source comments.
Warning output specification	-w	Specifies whether a warning message is output to the console.
Execution state display specification	-v	Specifies whether the execution status of compilation is output to the console.
	-nv	
Parameter file specification	-f	Inputs input file names and options from specified files.
Temporary file creation folder specification	-t	Specifies the drive and folder where the temporary files are created.
Help specification	--	Outputs help messages to the console.
	-?	
	-h	
Function expansion specification	-z	Enables extended functions.
	-nz	
Device file search path	-y	Specifies paths that search device files.
Memory model specification	-m	Specifies the memory model used for compilation.

5.4 Descriptions

This section describes each compiler option in detail.

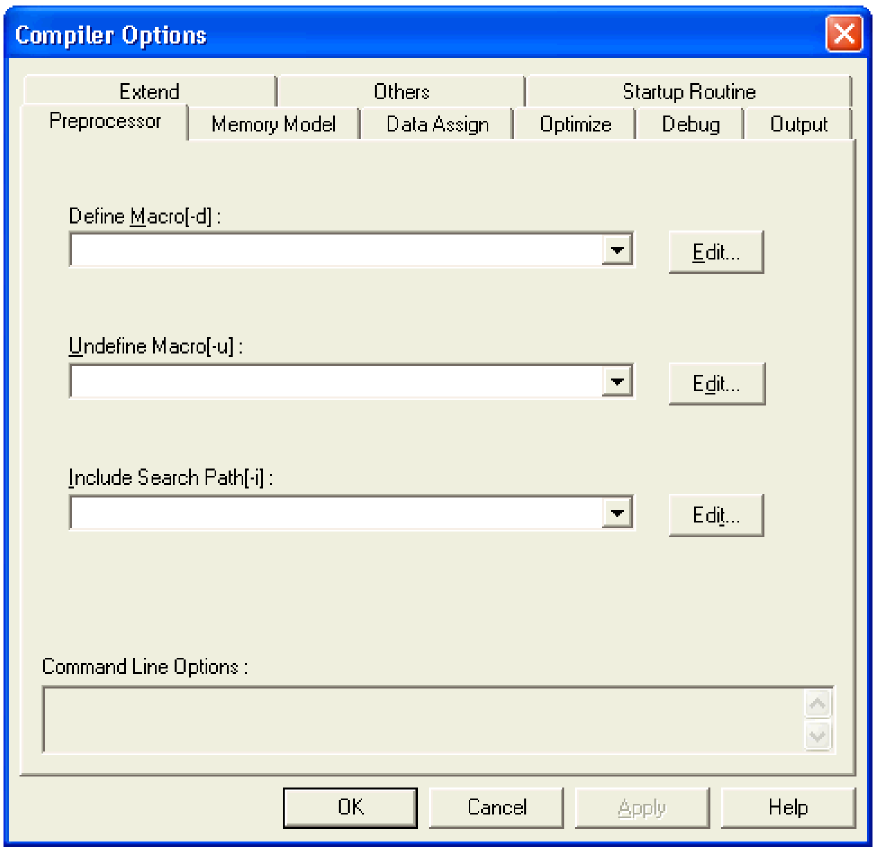
This example illustrates starting the CC78K0R in the command line. To start in PM+, specify the command, device type specification, and options left out of the C source in the <Compiler Options> dialog box.

[Example: (In command line)]

```
C>cc78k0r -cf1166a0 prime.c -g
```

[Example: (When using PM+)]

Figure 5-1 [Compiler Options] Dialog Box



Device type specification

(1) -c

[Description format]

```
-cdevice-type
```

- Interpretation when omitted

  Specification of this option cannot be omitted.

[Function]

- The -c option specifies the target device designated for compilation.

[Application]

- Be sure to specify this option. The CC78K0R compiles for the specified target device and generates the object code for it.

[Description]

- For the target devices that can be specified by the -c option and the corresponding device type, refer to the user's manual of the device used or "Device Files Operating Precautions".
- When CC78K0R is used, device files are required.

[Caution]

- The -c option cannot be omitted. However, if the following description is in the C source, the specification can be omitted from the command line.

```
#pragma pc (device-type)
```

- If different devices were specified in the C source and the command line, the device in the command line has priority.
- It is not necessary for this option to be set by the compiler option when PM+ is used, because the setting of this option is determined by the project setting.

[Use Example]

- To specify in the command line that the uPD78F1166_A0 is to be the target device, describe as:

```
C>cc78k0r -cf1166a0 prime.c
```

- Specify the target device (uPD78F1166_A0) in the C source (prime.c) and start the compiler.

```
#pragma pc ( f1166a0 )

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void ) {
        int     i , prime , k , count ;
            :
}
```

This allows the target device specification to be omitted from the command line.

```
C>cc78k0r prime.c
```

- Specify different devices in C source (prime.c) and the command line, and then start the compiler.

<C source>

```
#pragma pc ( f1166a0 )

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void ) {
        int     i , prime , k , count ;
            :
}
```

<Command line>

```
C>cc78k0r -cf1176 prime.c
```

The target device specified in the command line is given priority, so the compiler runs as follows.

```
78K0R C Compiler Vx.xx  [xx xxx xxxx]
    Copyright(C) NEC Electronics Corporation xxxx, xxxx

sample\prime.c ( 1 )  : CC78K0R warning W0832 : Duplicated chip specifier
sample\prime.c ( 18 ) : CC78K0R warning W0745 : Expected function prototype
sample\prime.c ( 20 ) : CC78K0R warning W0745 : Expected function prototype
sample\prime.c ( 26 ) : CC78K0R warning W0622 : No return value
sample\prime.c ( 37 ) : CC78K0R warning W0622 : No return value
sample\prime.c ( 44 ) : CC78K0R warning W0622 : No return value

Target chip : uPD78F1176
Device file : Vx.xx

Compilation complete, 0 error(s) and 6 warning(s) found.
```

Object module file creation specification

(1) -o/-no

[Description formats]

```
-o[output-file-name]
-no
```

- Interpretation when omitted

  -o*input-file-name*.rel

[Function]

- The -o option specifies the output of the object module file. In addition, the output destination or output file name is specified.
- The -no option specifies not to output the object module file.

[Application]

- If you want to change the output destination or the output file name of the object module file, specify the -o option.
- If only the output of the assembler source module file is the target for compilation, specify the -no option. Consequently, the compilation time is reduced.

[Description]

- If the output file name is omitted when the -o option is specified, the object module file name becomes "*input-file-name*.rel".
- If the extension for the output file name is omitted when the -o option is specified, object module file *output-file-name*.rel will be output.
- If there is a compilation error even when the -o option is specified, the object module file is not output.
- If the drive name is omitted when the -o option is specified, the object module file is output to the current drive.
- If both the -o and -no options are simultaneously specified, the last specified one is valid.

[Cautions]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Object Module File" area under the [Output] tab.
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the [Output Path] combo box under the [Output] tab.

[Use Example]

- The -no option that is specified first is ignored, the -o option that is specified second is valid, so the object module file (prime.o) will be output.

```
C>cc78k0r -cf1166a0 prime.c -no -o
```

Memory assignment specification

(1) -r/-nr

[Description formats]

```
-rprocess-type (Multiple specifications are possible)
-nr
```

- Interpretation when omitted

  -nr

[Function]

- The -r option specifies how to assign a program to the memory.
- The -nr option invalidates the -r option.

[Application]

- If you want to specify how to assign a program to the memory, specify the -r option.

[Description]

- The process types that can be specified by the -r option are shown below.

  Process type specification cannot be omitted. Otherwise, Fatal error (F0012) will occurs.

Process Type	Function
a	Performs indirect reference in 1-byte units.
b	Assigns a bit field from the most significant bit (MSB).
d[*n*][m] (*n* = 1, 2, 4)	Assigns an external variable/external static variable (except for the const-type variable) automatically to the saddr area, irrespective of whether there is an sreg declaration or not. For details, see "(2) -rd/-nr".
s[*n*][m] (*n* = 1, 2, 4)	Assigns a static auto variable automatically to the saddr area, irrespective of whether there is an sreg declaration or not. For details, see "(3) -rs/-nr".
c	Performs indirect reference in 1-byte units. Packs a structure and aligns the structure members to 1 byte.

Remark Multiple process types can be specified.

- When the -nr option is specified, the process types are interpreted as follows.

Process Type	Function
a	Does not perform indirect reference in 1-byte units.
b	Assigns a bit field from the least significant bit (LSB).
d	Does not automatically assign any variable to the saddr area.
s	Does not automatically assign any variable to the saddr area.
c	Does not perform indirect reference in 1-byte units. Does not pack a structure and does not align the structure members to 1 byte.

[Use Example]

- To allocate the external variable or external static variable, and static auto variable automatically to the saddr area, regardless of whether sreg has been declared, describe as:

```
C>cc78k0r -cf1166a0 -rds
```

(2) -rd/-nr

[Description formats]

```
-rd[n][m] (n = 1, 2, 4)
-nr
```

- Interpretation when omitted

  -nr

[Function]

- The -rd option specifies the automatic assignment of an external variable/external static variable (except for the const-type variable) to the saddr area.
- The -nr option invalidates the -rd option.

[Application]

- If you want to automatically assign an external variable/external static variable (except for the const-type variable) to the saddr area irrespective of whether there is an sreg declaration or not, specify the -rd option.

[Description]

- Variables to be assigned change depending on the value of *n* and the specification of m.

Specification of *n*, m	Variable Types to Be Assigned
n	When *n* = 1: char, unsigned char When *n* = 2: char, unsigned char, short, unsigned short, int, unsigned int, enum, near pointer When *n* = 4: char, unsigned char, short, unsigned short, int, unsigned int, enum, long, unsigned long, pointer
m	Structure, Union, Array
Omitted	All variables

- The sreg-declared variable is automatically assigned to the saddr area irrespective of the -rd option specification.
- The variable that is referenced by means of an extern declaration is processed as are to be assigned to the saddr area.
- The variable assigned to the saddr area by specifying this option is handled in a similar way to an sreg variable.

[Use Example]

- To allocate the char or unsigned char type external variable or external static variable automatically to the saddr area, regardless of whether sreg has been declared, describe as:

```
C>cc78k0r -cf1166a0 -rd1
```

(3) -rs/-nr

[Description formats]

```
-rs[n][m] (n = 1, 2, 4)
-nr
```

- Interpretation when omitted

  -nr

[Function]

- The -rs option specifies the automatic assignment of a static auto variable to the saddr area.
- The -nr option invalidates the -rs option.

[Application]

- If you want to automatically assign a static auto variable to the saddr area irrespective of whether there is an sreg declaration or not, specify the -rs option.

[Description]

- Variables to be assigned change depending on the value of *n* and the specification of m.

Specification of *n*, m	Variable Types to Be Assigned
n	When *n* = 1: char, unsigned char When *n* = 2: char, unsigned char, short, unsigned short, int, unsigned int, enum, near pointer When *n* = 4: char, unsigned char, short, unsigned short, int, unsigned int, enum, long, unsigned long, pointer
m	Structure, Union, Array
Omitted	All variables

- The sreg-declared variable is automatically assigned to the saddr area irrespective of the -rs option specification.
- The static auto variable that is assigned to the saddr area by specifying this option is handled in a similar way to an sreg-declared auto variable.

[Use Example]

- To allocate the char or unsigned char type static auto variable automatically to the saddr area, regardless of whether sreg has been declared, describe as:

```
C>cc78k0r -cf1166a0 -rs1
```

Optimization specification

(1) -q/-nq

[Description formats]

```
-q[optimization-type] (Multiple specifications are possible)
-nq
```

- Interpretation when omitted

  When the -qr option is not specified, objects for real-time OS is output and register declarations (when saddr area is used) are ignored.

[Function]

- The -q option specifies calling the optimization phase to generate efficient objects.
- The -nq option invalidates the -q option.

[Application]

- If you want to improve the execution speed of the objects and reduce the code size, specify the -q option. If the -q option is specified and you want to perform multiple optimizations simultaneously, specify the optimization types consecutively. For details, see **[Description]**.

[Description]

- The table below lists the optimization types that can be specified by the -q option.

Optimization Type	Process Description
No specification	When the -qr option is not specified, objects for real-time OS is output and register declarations (when saddr area is used) are ignored.
u	Regards the char with no qualifier as a unsigned char to improve code efficiency

Optimization Type	Process Description
c	Performs calculations including char without sign extension.

Calculation Target	Calculation Result
unsigned char type variable and unsigned char type variable	unsigned char type
unsigned char type variable and signed char type variable	unsigned char type
signed char type variable and signed char type variable	signed char type
Constants from -128 to 255 and unsigned char type variable	unsigned char type
Constants from -128 to 127 and signed char type variable	signed char type
Constants from 0 to 255 with suffix U and signed char type variable	unsigned char type

Optimization Type	Process Description
r	Adds a register variable to a register and assigns it to the saddr area.
j	Optimizes jump instructions.
x[*n*] (*n* = 1 - 4)	Assigns the optimization options automatically according to the priority of speed/code size. The assigned option differs depending on the value of *n* as follows. If *n* is omitted, it is interpreted as *n* = 2. 1: Speed precedence. Regarded as the -qcjvw option specification. 2: Default. Regarded as the -qcjlvw option specification. 3: Code size precedence. Regarded as the -qcjl3vw option specification. 4: Code size precedence. Regarded as the -qcjl4vw option specification.
w	Performs aggressive optimization. Reshuffles the execution order in an expression.
v	Assigns an argument and automatic variable automatically to a register or the saddr area.
l[*n*] (*n* = 1 - 4)	Performs optimization based on the priority of code size and replaces the standard code pattern with a library. If this option is not specified, the code is optimized based on the priority of speed. The scope changes depending on the value of *n* as follows. If *n* is omitted, it is interpreted as *n* = 1. 1: No replacement 2: Executes the only the processes before/after a function 3: Executes the processes before/after a function, uses a lower level library 4: Executes the processes before/after a function and subroutine call same codes When the -mm or -ml option is specified, the code size per file does not exceed 32KB If the code size exceeds 32KB, an error message (F0925) will be output.

Optimization Type	Process Description
t	Makes the branch table in a switch statement of a function allocated to the far area, into of a relative branch. If the branch distance exceeds 64 KB, an error message (F0924) will be output.

- Multiple optimization types can be specified.
- If the -q option or optimization types are omitted, the optimization is identical to when the -qcjlvw option is specified.
- To delete a portion of the default options specify the options other than the options you want to delete (Example -qr is specified -> Deletes -qcjlvw).
- If both the object module file and the assembler source module file are not output, the -q option other than -qu becomes invalid.
- If both the -q and -nq options are simultaneously specified, the last specified one is valid.
- If several -q options are simultaneously specified, the last specified one is valid.

[Use Example]

- Regarding char without a qualifier as an unsigned char enhances code efficiency.

```
C>cc78k0r -cf1166a0 prime.c -qu
```

- The -qc option that is specified first is ignored, the -qr option that is specified second is valid, and arguments of norec, auto variables, and register variables are allocated to the saddr area.

```
C>cc78k0r -cf1166a0 prime.c -qc -qr
```

- To validate both the -qc and -qr options, describe as:

```
C>cc78k0r -cf1166a0 prime.c -qcr
```

Debugging information output specification

(1) -g/-ng

[Description formats]

```
-g[n] (n = 1, 2)
-nq
```

- Interpretation when omitted

  -g2

[Function]

- The -g option specifies the addition of debugging information to the object module file.
- The -ng option invalidates the -g option.

[Application]

- If the -g option is not specified, the line numbers and symbol information needed in the object module file to be input to the debugger are not output. Therefore, in source level debugging, all of the modules to be linked are compiled by specifying the -g option.

[Description]

- The operation differs depending on the value of *n* as follows.

Value of *n*	Function
Omitted	Regarded as *n* = 2.
1	Adds debug information (information starting with $DGS or $DGL) to the object module file only. No debug information is added to the assembler source module file. This option makes it easier to reference an assembler file. Source debugging of object files is available since debug information is added to them.
2	Adds debug information to the object module file and the assembler source module file.

- If both -g and -ng are simultaneously specified, the last specified one is valid.
- If both the object module file and the assembler source module file are not output, the -g option becomes invalid.

[Use Example]

- To add debug information in the object module file (prime.o), describe as:

```
C>cc78k0r -cf1166a0 prime.c -g
```

Preprocess list file creation specification

(1) -p

[Description format]

```
-p[output-file-name]
```

- Interpretation when omitted

  None (no file is output)

[Function]

- The -p option specifies the output of the preprocess list file. In addition, the output destination or output file name is specified. If the -p option is omitted, no preprocess list file is output.

[Application]

- If you want to output the source file after preprocess processing is executed according to the -k option process type, or want to change the output destination or the output file name of the preprocess list file, specify the -p option.

[Description]

- If the output file name is omitted when the -p option is specified, the preprocess list file name becomes "*input-file-name*.ppl".
- If the extension for the output file name is omitted when the -p option is specified, preprocess list file *output-file-name*.ppl will be output.
- If the drive name is omitted when the -p option is specified, the preprocess list file is output to the current drive.

[Cautions]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Create Preprocess List File" area under the [Output] tab.
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the [Output Path] combo box under the [Output] tab.

[Use Example]

- To output the preprocess list file (sample.ppl), describe as:

```
C>cc78k0r -cf1166a0 prime.c -psample.ppl
```

(2) -k

[Description format]

```
-k[process-type] (Multiple specifications are possible)
```

- Interpretation when omitted

  -kfln

[Function]

- The -k option specifies the processing for the preprocess list.

[Application]

- This option is specified when comments are deleted and definition expansions are referenced when the preprocess list file is output.

[Description]

- The process types that can be specified by the -k option are listed below.

Process Type	Description
Omitted	Same as specifying -kfln
c	Delete comments
d	#define expansion
f	Conditional compilations of #if, #ifdef, and #ifndef
i	#include expansion
l	#line processing
n	Line number and paging processing

Remark Multiple process types can be specified.

- If the -p option is not specified, the -k option becomes invalid.
- If several -k options are simultaneously specified, the last specified one is valid.

[Use Example]

- To perform deletion of comments, line number processing and page processing when the preprocess list file (prime.ppl), describe as:

```
C>cc78k0r -cf1166a0 prime.c -p -kcn
```

<Output example>

```
/*
78K0R C Compiler Vx.xx Preprocess List      Date:xx xxx xxxx    Page:1

Command         : -cf1166a0 prime.c -p -kcn
In-file         : prime.c
PPL-file        : prime.ppl
Para-file       :
*/

        1 : #define TRUE    1
        2 : #define FALSE   0
        3 : #define SIZE    200
        4 :
        5 : char    mark [ SIZE + 1 ] ;
        6 :
        7 : main ( )
        8 : {
                :
/*
Target chip     : uPD78F1166_A0
Device file     : Vx.xx
*/
```

Preprocess specification

(1) -d

[Description format]

```
-dmacro-name[=definition-name][,macro-name[=definition-name]]...
(Multiple specifications are possible)
```

- Interpretation when omitted

  Only the macro definitions in a C source module file are valid.

[Function]

- The -d option specifies the same macro definition as the #define statement in the C source.

[Application]

- Specify this option when you want to validate the special macro definition.

[Description]

- By delimiting each definition by a comma ",", 30 macro definitions are made at one time.
- Spaces are not allowed before and after "=" and ",".
- If the definition name is omitted, the compiler presumes that "macro name=1" was defined.
- If the same macro name was specified in both the -d and -u options, the last specified one is valid.

[Use Example]

- This is an example where the following codes are defined in the C source (prime.c).

```
#define TEST    1
#define TIME    10
```

```
C>cc78k0r -cf1166a0 prime.c -dTEST,TIME=10
```

(2) -u

[Description format]

```
-umacro-name[,macro-name]... (Multiple specifications are possible)
```

- Interpretation when omitted
  A macro definition specified with -d is valid.

[Function]

- The -u option disables macro definitions similar to the #undef statement in the C source.

[Application]

- Specify this option to invalidate the macro name defined by the -d option.

[Description]

- By delimiting each macro name by a comma ",", 30 macro definitions can be disabled at one time.
  Spaces are not allowed before and after a comma ",".
- A macro definition that can be disabled by the -u option is one that has been defined by the -d option.
  A macro name defined by #define in a C source module file or a system macro name of the CC78K0R cannot be disabled by the -u option.
- If the same macro name is specified by both the -d and -u options, the last specified one is valid.

[Use Example]

- The -d option that is specified first is ignored and the -u option that is specified second is valid, the macro definition for TEST thus becomes invalid.

```
C>cc78k0r -cf1166a0 prime.c -dTEST,TIME=10 -uTEST
```

(3) -i

[Description format]

```
-ifolder[,folder]... (Multiple specifications are possible)
```

- Interpretation when omitted

  The compiler considers that the following folders were specified.

  (i) Folder with source file[Note 1]

  (ii) Folder specified by environment variable INC78K0R

  (iii) C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\inc78k0r[Note 2]

[Function]

- The -i option specifies input of the include files specified by the #include statement in the C source from the specified folder.

[Application]

- Specify this option when you want to search for the include files from a certain folder.

[Description]

- By using a comma "," to delimit, 64 folders can be specified at one time.
- Spaces cannot be inserted before and after a comma ",".
- If multiple folders are specified after -i, or if the -i option is specified multiple times, the files specified by #include are searched for in the specified order.
- The search sequence is as follows.

  (i) Folder with source file[Note 1]

  (ii) Folder specified with the -i option

  (iii) Folder specified with environment variable INC78K0R

  (iv) C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\inc78k0r[Note 2]

Note 1 If the include file name is specified with " " (double quotation marks) in the #include statement, folders with source files are searched first. If the include file name is specified with < >, search is not performed.

Note 2 This is an example of when the CC78K0R is installed to C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*.

[Use Example]

- To input the include file that is specified in an #include statement in the C source (prime.c) from folder b: and b:\sample, describe as:

```
C>cc78k0r -cf1166a0 prime.c -ib:,b:\sample
```

Assembler source module file creation specification

(1) -a

[Description format]

```
-a[output-file-name]
```

- Interpretation when omitted
  No assembler source module file is output.

[Function]

- The -a option specifies the output of the assembler source module file. In addition, the output destination or output file name is specified.

[Application]

- If you want to change the output destination or the output file name of the assembler source module file, specify the -a option.

[Description]

- A disk file name or device file name can be specified as the file name.
- If the output file name is omitted when the -a option is specified, the assembler source module file name becomes "*input-file-name*.asm".
- If the extension for the output file name is omitted when the -a option is specified, assembler source module file *output-file-name*.asm will be output.
- If the drive name is omitted when the -a option is specified, the assembler source module file is output to the current drive.
- If both the -a and -sa options are simultaneously specified, the -sa option is ignored.

[Caution]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Create Assembler Source Module File" area under the [Output] tab, and select "without C Source[-a]".
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the [Output Path] combo box under the [Output] tab. To specify a file name, append the extension "asm".

[Use Example]

- To output the assembler source module file (sample.asm), describe as:

```
C>cc78k0r -cf1166a0 prime.c -asample.asm
```

(2) -sa

[Description format]

```
-sa[output-file-name]
```

- Interpretation when omitted

  No assembler source module file is output.

[Function]

- The -sa option adds the C source as a comment to the assembler source module file. In addition, the output destination or output file name is specified.

[Application]

- If you want to output the assembler source module file and the C source module file together, specify the -sa option.

[Description]

- A disk file name or device file name can be specified as the file name.
- If the output file name is omitted when the -sa option is specified, the assembler source module file name becomes "*input-file-name*.asm".
- If the extension for the output file name is omitted when the -sa option is specified, assembler source module file *output-file-name*.asm will be output.
- If the drive name is omitted when the -sa option is specified, the assembler source module file is output to the current drive.
- If both the -sa and -a options are simultaneously specified, the -sa option is ignored.
- The C source in an include file is not added to the comments in the output assembler source module. However, if the -li option is specified, the C source in the include file is also added to the comments.

[Caution]

- To change the output destination when using PM+, specify the new output destination in the "Output Path" combo box in the "Create Assembler Source Module File" area under the [Output] tab, and select either "with C Source[without Include][-sa]" or "with C Source[with Include][-sa -li]".
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output Path" combo box under the [Output] tab. To specify a file name, append the extension "asm".

[Use Example]

- To add the C source (prime.c) as a comment to the assembler source module file (prime.asm), describe as:

```
C>cc78k0r -cf1166a0 prime.c -sa
```

<Output example>

```
; 78K0R C Compiler Vx.xx Assembler Source   Date:xx xxx xxxx    Time:xx:xx:xx

; Command       : -cf1166a0 prime.c -sa
; In-file       : prime.c
; Asm-file      : prime.asm
; Para-file     :

$PROCESSOR ( f1166a0 )
$DEBUG
$NODEBUGA
$KANJICODE SJIS
$TOL_INF        03FH , 100H , 00H , 00H , 00H

$DGS    FIL_NAM , .file ,       037H , 0FFFEH , 03FH , 067H , 01H , 00H
$DGS    AUX_FIL , prime.c
$DGS    MOD_NAM , prime ,       00H , 0FFFEH , 00H , 077H , 00H , 00H
          :
        EXTRN   _@RTARG0
        EXTRN   @@isrem
        PUBLIC  _printf
        PUBLIC  _putchar
        PUBLIC  _mark
        PUBLIC  _main
          :
@@CODEL CSEG
_main :
$DGL    1 , 19
        push    hl                       ; [ INF ] 1 , 1
        subw    sp , #08H                ; [ INF ] 2 , 1
        movw    hl , sp                  ; [ INF ] 3 , 1
??bf_main :
; line   9 :    int i , prime , k , count ;
; line  10 :
; line  11 :    count = 0 ;
$DGL    0 , 4
        clrw    ax                       ; [ INF ] 1 , 1
        movw    [ hl ] , ax     ; count ; [ INF ] 1 , 1
; line  12 :
; line  13 :    for ( i = 0 ; i <= SIZE ; i++ )
$DGL    0 , 6
        movw    [ hl + 6 ] , ax ; i     ; [ INF ] 2 , 1
?L0003 :
        movw    ax , [ hl + 6 ] ; i     ; [ INF ] 2 , 1
        cmpw    ax , #0C8H      ; 200   ; [ INF ] 3 , 1
        or1     CY , a.7                 ; [ INF ] 2 , 1
        skc                              ; [ INF ] 2 , 1
        bnz     $?L0004                  ; [ INF ] 2 , 4
          :

; *** Code Information ***
;
;
```

```
; $FILE C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\smp78k0r\cc78k0r\prime.c
;
; $FUNC main ( 8 )
;       bc = ( void )
;       CODE SIZE = 117 bytes , CLOCK_SIZE = 86 clocks , STACK_SIZE = 16 bytes
;
; $CALL printf ( 18 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $CALL putchar ( 18 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $CALL putchar( 20 )
;       bc = ( int : ax )
;
; $CALL printf ( 25 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $FUNC printf ( 31 )
;       bc = ( pointer s : ax , int i : [ sp + 4 ] )
;       CODE SIZE = 22 bytes , CLOCK_SIZE = 20 clocks , STACK_SIZE = 14 bytes
;
; $FUNC putchar ( 41 )
;       bc = ( char c : x )
;       CODE SIZE = 16 bytes , CLOCK_SIZE = 16 clocks , STACK_SIZE = 6 bytes

; Target chip : uPD78F1166_A0
; Device file : Vx.xx
```

Error list file creation specification

(1) -e

[Description format]

```
-e[output-file-name]
```

- Interpretation when omitted
  No error list file is output.

[Function]

- The -e option specifies the output of the error list file. In addition, the output destination or output file name is specified.

[Application]

- If you want to change the output destination or the output file name of the error list file, specify the -e option.

[Description]

- A disk file name or device file name can be specified as the file name.
- If the output file name is omitted when the -e option is specified, the error list file name becomes "*input-file-name*.ecc".
- If the extension for the output file name is omitted when the -e option is specified, error list file *output-file-name*.ecc will be output.
- If the drive name is omitted when the -e option is specified, the error list file is output to the current drive.
- If the -w0 option is specified, warning messages are not output.

[Cautions]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Create Error List File" area under the [Output] tab and select "without C Source[-e]".
- When individual compileroptions are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output Path" combo box under the [Output] tab.

[Use Example]

- To output the error list file (prime.ecc), describe as:

```
C>cc78k0r -cf1166a0 prime.c -e
```

<Output example>

```
prime.c ( 18 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 20 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 26 ) : CC78K0R warning W0622 : No return value
prime.c ( 37 ) : CC78K0R warning W0622 : No return value
prime.c ( 44 ) : CC78K0R warning W0622 : No return value

Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 5 warning(s) found.
```

(2) -se

[Description format]

```
-se[output-file-name]
```

- Interpretation when omitted
  No error list file is output.

[Function]

- The -se option adds the C source module file to the error list file. In addition, the output destination or output file name is specified.

[Application]

- If you want to output the error list file and the C source module file together, specify the -se option.

[Description]

- A disk file name or device file name can be specified as the file name.
- If the output file name is omitted when the -se option is specified, the error list file name becomes *input-file-name*.cer.
- If the extension for the output file name is omitted when the -se option is specified, error list file *output-file-name*.cer will be output.
- If the drive name is omitted when the -se option is specified, the error list file is output to the current drive.
- The folder and the file name cannot be specified for include files.
  If the file type of the include file is "H", the error list file with the file type of "her" is output to the current drive.
  It the file type of the include file is "C", the error list file with the file type of "cer" is output. In all other cases, the error list file with the "er" file type is output.
- If there weren't any errors, the C source is not added. In this case, the error list file is not created for the include file.
- If the -w0 option is specified, warning messages are not output.

[Cautions]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Create Error List File" area under the [Output] tab and select "with C Source[-se]".
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the [Output Path] combo box under the [Output] tab.

[Use Example]

- To add the C source module file (prime.c) to the error list file (prime.cer), describe as:

```
C>cc78k0r -cf1166a0 prime.c -se
```

<Output example>

```
/*
78K0R C Compiler Vx.xx Error List       Date:xx xxx xxxx    Time:xx:xx:xx

Command         : -cf1166a0  prime.c  -se
In-file         : prime.c
Err-file        : prime.cer
Para-file       :
*/

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void )
{
            :
        prime = i + i + 3 ;
        printf ( "%6d" , prime ) ;
*** CC78K0R warning W0745 : Expected function prototype
        count++ ;
        if ( ( count%8 ) == 0 ) putchar ( '\n' ) ;
*** CC78K0R warning W0745 : Expected function prototype
        for ( k = i + prime ; k <= SIZE ; k += prime )
            :
}
```

Cross-reference list file creation specification

(1) -x

[Description format]

```
-x[output-file-name]
```

- Interpretation when omitted

  No cross-reference list file is output.

[Function]

- The -x option specifies the output of the cross-reference list file. In addition, the output destination or output file name is specified. The cross-reference list file is valuable for checking the referencing frequency, definition, and referenced point of a symbol.

[Application]

- If you want to output the cross-reference list file or want to change the output destination or the output file name of the cross-reference list file, specify the -x option.

[Description]

- A disk file name or a device file name can be specified as the file name.
- If the output file name is omitted when the -x option is specified, the cross-reference list file name becomes "*input-file-name*.xrf".
- If the extension for the output file name is omitted when the -x option is specified, cross-reference list file *output-file-name*.xrf will be output.
- Even if an internal error other than C0101 or a compilation error with the number F0024 or a number starting from E occurs, a cross-reference list file is created. However, the contents of the file are not guaranteed.

[Cautions]

- To change the output destination when using PM+, specify the new output destination in the [Output Path] combo box in the "Create Cross Reference List File[-x]" area under the [Output] tab.
- When individual compiler options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the [Output Path] combo box under the [Output] tab.

[Use Example]

- To output the cross-reference list file (prime.xrf), describe as:

```
C>cc78k0r -cf1166a0 prime.c -x
```

<Output example>

```
78K0R C Compiler Vx.xx Cross reference List    Date:xx xxx xxxx    Page:1

Command   : -cf1166a0 prime.c -x
In-file   : prime.c
Xref-file : prime.xrf
Para-file :

ATTRIB MODIFY TYPE    SYMBOL          DEFINE  REFERENCE

EXTERN NEAR   array   mark                5      29      31      37
EXTERN FAR    func    printf              7      33      40
REG1          pointer s                   7      13
PARAM
REG1          int     i                   7      12
PARAM
REG1          int     j                   9      12
REG1          pointer ss                 10      13
EXTERN FAR    func    putchar            16      35
REG1          char    c                  16      19
PARAM
REG1          char    d                  18      19
EXTERN FAR    func    main               22
REG1          int     i                  24      28      28      28      29
    30      30      30      31      32      32
                                                 36
REG1          int     prime              24      32      33      36      36
REG1          int     k                  24      36      36      36      37
REG1          int     count              24      26      34      35      40
              #define TRUE                1      29
              #define FALSE               2      37
              #define SIZE                3       5      28      30      36


 Target chip : uPD78F1166_A0
 Device file : Vx.xx
```

List format specification

(1) -lw

[Description format]

```
-lw[number-of-characters]
```

- Interpretation when omitted

  -lw132 (For console output, this becomes 80 characters)

[Function]

- The -lw option specifies the number of characters in 1 line of each type of list file.

[Application]

- If you want to change the number of characters in 1 line of each list file, specify the -lw option.

[Description]

- The range of the number of characters that can be specified by the -lw option is as follows and does not include terminators (CR, LF).

  72 ≤ number of characters printed in 1 line ≤ 132

- If the number of characters is omitted, the number of characters in 1 line becomes 132 characters (If output to the console, there is a maximum of 80 characters).
- If the list file specification specifies nothing, the -lw option is invalid.

[Use Example]

- To set the number of characters on 1 line of the cross-reference list file (prime.xrf) to 72 characters, describe as:

```
C>cc78k0r -cf1166a0 prime.c -x -lw72
```

(2) -ll

[Description format]

```
-ll[number-of-lines]
```

- Interpretation when omitted

  There is no page break

[Function]

- The -ll option specifies the number of lines on 1 page of each type of list file.

[Application]

- If you want to change the number of lines in 1 page in each type of list file, specify the -ll option.

[Description]

- The range of the number of lines that can be specified by the -ll option is as follows.

  20 $\leq$ number of lines printed on 1 page $\leq$ 65535

- If -ll0 is specified, there is no page break.
- If the number of lines is omitted, there is no page break.
- If the list file specification specifies nothing, the -ll option is invalid.

[Use Example]

- To set the number of lines on 1 page of the cross-reference list file (prime.xrf) to 20 lines, describe as:

```
C>cc78k0r -cf1166a0 prime.c -x -ll20
```

(3) -lt

[Description format]

```
-lt[number-of-characters]
```

- Interpretation when omitted
  -lt8

[Function]

- The -lt option indicates the basic number of characters for outputting a horizontal tabulation (HT) code in the source module file, replacing it with several blanks (spaces) in each list (tabulation processing).

[Application]

- If few characters are specified in 1 line in each list by the -lw option, few blanks will result from an HT code, so specify the -lt option to reduce the number of characters.

[Description]

- The range of the number of characters that can be specified by the -lt option is as follows.

  0 ≤ number of specifiable characters ≤ 8

- If the -lt0 is specified, the tabulation processing is not performed, and the tab codes are output.
- If the number of characters is omitted, the number of expansion characters of a tab becomes 8 characters.
- If the list file specification specifies nothing, the -lt option is invalid.

[Use Example]

- If the -lt option is omitted, the compiler presumes that the -lt8 option was specified and the number of blanks to be output from the HT code is set to 8.

```
C>cc78k0r -cf1166a0 prime.c -p
```

- The number of blanks to be output from the HT code is set to 1.

```
C>cc78k0r -cf1166a0 prime.c -p -lt1
```

(4) -lf

[Description format]

```
-lf
```

- Interpretation when omitted

  The new page break code will not be added.

[Function]

- The -lf option specifies adding the new page break code at the end of each list file.

[Description]

- If the list file specification specifies nothing, the -lf option is invalid.

[Use Example]

- To add the new page break code at the end of the assembler source module file (prime.asm), describe as:

```
C>cc78k0r -cf1166a0 prime.c -a -lf
```

(5) -li

[Description format]

```
-li
```

- Interpretation when omitted

  No C sources of the include file will be added.

[Function]

- The -li option adds the C source of the include file to the assembler source module file with C source comments.

[Description]

- If the -sa option is not specified, this option is ignored.

[Use Example]

- To add the C source file of the include file to the assembler source module file (prime.asm) with C source comments, describe as:

```
C>cc78k0r -cf1166a0 prime.c -sa -li
```

Warning output specification

(1) -w

[Description format]

```
-w[level]
```

- Interpretation when omitted

  -w1

[Function]

- The -w option specifies whether a warning message is output to the console.

[Application]

- This option specifies whether to output warning messages to the console.

  Detailed messages can also be output.

[Description]

- The levels of the warning message are given below.

Level	Description
0	Do not output warning messages.
1	Output normal warning messages.
2	Output detailed warning messages.

- If the -e or -se option is specified, the warning messages are output to the error list file.
- Level 0 indicates not to output warning messages to the console and the error list file (when -e or -se is specified).

[Use Example]

- If the -w option is omitted, the compiler presumes that the -w1 option was specified and outputs normal warning messages.

```
C>cc78k0r -cf1166a0 prime.c
```

Execution state display specification

(1) -v/-nv

[Description formats]

```
-v
-nv
```

- Interpretation when omitted

  -nv

[Function]

- The -v option outputs the execution state of the current compilation to the console.
- The -nv option invalidates the -v option.

[Application]

- Specify this option to check the execution status of compilation.

[Description]

- The phase name and function names in the process are output.
- If both the -v and -nv options are simultaneously specified, the last specified one is valid.

[Use Example]

- To output the current status of compilation to the console, describe as:

```
C>cc78k0r -cf1166a0 prime.c -v
```

Parameter file specification

(1) -f

[Description format]

```
-ffile-name
```

- Interpretation when omitted

  The input of an option and an input file name is possible only from a command line.

[Function]

- The -f option specifies the input of the options or input file name from the specified file.

[Application]

- When sufficient information for starting the CC78K0R cannot be specified in a command line because multiple options are input while compiling, specify the -f option.
- When specifying options repeatedly for compilation, describe the options in the parameter file and specify the -f option.

[Description]

- Parameter file nesting is not allowed.
- The number of characters that can be described in a parameter file is not limited.
- Spaces and tabs delimit the options or input file names.
- The options or input file names described in the parameter file are expanded at the location of the parameter file specification in the command line.
- The prioritization of the expanded options is that the last specified one is valid.
- Characters described after the ";" and "#" are interpreted as comments until the end of the line.

[Use Example]

- Contents of the parameter file (prime.pcc)

```
; parameter file
prime.c -cf1166a0 -aprime.asm -e -x
```

The parameter file (prime.pcc) is used in the compilation.

```
C>cc78k0r -fprime.pcc
```

Temporary file creation folder specification

(1) -t

[Description format]

```
-tfolder
```

- Interpretation when omitted

  The temporary files are created in the drive folder specified by the environment variable TMP. If not specified, the files are created in the current drive and current folder.

[Function]

- The -t option specifies the drive and folder where the temporary files are created.

[Application]

- The location for creating the temporary files can be specified.

[Description]

- Even if there are temporary files that have been created previously, if a file is not protected, it is overwritten the next time it is created.
- A temporary file expands in memory to the required memory size.
  If the required memory size is no longer available, the temporary file is created in the specified folder and the memory contents are written to the file. Accesses to subsequent temporary files are to files not in memory.
- The temporary files are deleted when compilation ends. By pressing CTRL-C, the files are deleted when compilation stops.

[Use Example]

- To output the temporary files to the tmp folder, describe as:

```
C>cc78k0r -cf1166a0 prime.c -ttmp
```

Help specification

(1) --/-?/-h

[Description formats]

```
--
-?
-h
```

- Interpretation when omitted

  Nothing is displayed

[Function]

- The --, -?, and -h options display brief explanations of the options and the help messages such as the default options on the console (valid only in the command line[Note]).

  Note Do not specify this option in PM+. To reference help in PM+, click the [Help] button in the [Compiler Options] dialog box.

[Application]

- The option and its description are displayed. Refer to them when running the CC78K0R.

[Description]

- If the --, -?, or -h option is specified, all of the other compiler options become invalid.
- When viewing the continuation of a displayed help message, press the [Enter] key. To exit the display before the end, press any character other than the [Enter] key, and then press the [Enter] key.

[Use Example]

- To display the help messages on the console, describe as:

```
C>cc78k0r -h
```

Function expansion specification

(1) -z/-nz

[Description formats]

```
-ztype (Multiple specifications are possible)
-nz
```

- Interpretation when omitted

  -nz

[Function]

- The -z option enables extended functions.
- The -nz option invalidates the -z option.
- Types must not be omitted, otherwise, Fatal error (F0012) will occur.

[Application]

- The functions for processing by the following type specifications are available for the 78K0R expansion functions.

[Description]

- The type specifications of the -z option are as follows.

Type Specification	Description
p	The characters after "//" until the line return are interpreted as a comment.
c	Nested comments "/* */" are allowed.
s[Note]	Interprets the kanji (2-byte character) code in comments as SJIS.
e[Note]	Interprets the kanji code in comments as EUC.
n[Note]	Interprets comments as not containing kanji codes (2-byte codes).
b	char-/unsigned char-type argument and return value are not int-extended.
a	Functions not in the ANSI standard are illegal. The ANSI-compliant portion of the functions are valid. Specifically, the following tasks are performed. - The following are no longer reserved words. callt, sreg, bit, boolean, #asm, #endasm - The trigraph sequence (3-character representation) becomes valid. - The compiler-defined macro __STDC__ is 1. - Data allocation to the last one byte of a 64 KB boundary area is enabled by performing a relational expression for the far pointer for three bytes. - The following warning is output for a int type bit field. (CC78K0R warning W0787: Bit field type is not int) - If -w2 is specified, the following warnings are output for the -qc, -zp, -zc options. (CC78K0R warning W0029: '-QC' option is not portable) (CC78K0R warning W0031: '-ZP' option is not portable) (CC78K0R warning W0032: '-ZC' option is not portable) - If -w2 is specified, the following warning is output for each #pragma statement. (CC78K0R warning W0849: #pragma statement is not portable) - If -w2 is specified, the following warning is output for an __asm statement and the assemble output is performed. (CC78K0R warning W0850: Asm statement is not portable) - If -w2 is specified, the following error is output for an #asm to #endasm block. (CC78K0R error E0801: Undefined control, etc.)
f	Outputs object from flash.

Note s, e, and n cannot be specified simultaneously.

[Use Example]

- The characters after "//" until the line return in the C source (prime.c) are interpreted as a comment. Also, nested comments "/* */" are allowed.

```
C>cc78k0r -cf1166a0 prime.c -zpc
```

Device file search path

(1) -y

[Description format]

```
-yfolder
```

- Interpretation when omitted

  Normal search path only

  Remark The normal search paths are as follows.

  (i) <..\..\..\dev> (for the path where cc78k0r.exe started)

  (ii) Path where the CC78K0R started

  (iii) Current folder

  (iv) PATH environment variable

[Function]

- The -y option first searches the path specified as the search path for reading device files. If it does not exist, the normal paths are searched.

[Application]

- If the device file is not installed in the normal search path, but in a special folder, the path is specified by this option.

[Caution]

- When using PM+, a folder is determined when registering a device file at "Device Name" in the [Project Setup] dialog box. Therefore, it is not necessary to specify this option when setting options with this compiler.

[Use Example]

- To search "C:\tmp\dev" first to read the device file, describe as:

```
C>cc78k0r -cf1166a0 -yC:\tmp\dev
```

Memory model specification

(1) -m

[Description format]

```
-mtype
```

- Interpretation when omitted
  -mm

[Function]

- The -m option specifies the memory model used for compilation.
- Multiple models cannot be specified at the same time.
- Types must not be omitted; otherwise, the fatal error (F0012) will occur.

[Application]

- By specifying a memory model, whether each function and variable is allocated in the near or far area is specified.
- If a __near or __far qualifier is described for functions or variables in a C source, specification of the near or far area that is specified by the __near or __far qualifier takes precedence.

[Description]

- The following items are available for specifying the memory model with the -m option.

Type Specifications	Memory Model	Explanation
s	Small model	Considers the memory to consist of a code portion 64 KB (max.) and a data portion of 64 KB (max.), 128 KB in total, and specifies the near or far area.
m	Medium model	Considers the memory to consist of a code portion of 1 MB (max.) and a data portion 64 KB (max.), 1 MB in total, and specifies the near or far area.
l	Large model	Considers the memory to consist of a code portion of 1 MB (max.) and a data portion of 1 MB (max.), 1 MB in total, and specifies the near or far area.

Remark Even if a memory model that consists of a data portion or code portion of 64 KB (max.) is specified, functions and variables for which the __far qualifier is specified can be allocated to the space of 1 MB (max.).
Memory model specification specifies the location of functions or variables for which the __far qualifier is not specified.

[Use Example]

- To use the small model for the memory model during compilation, describe as:

```
C>cc78k0r -cf1166a0 prime.c -ms
```

CHAPTER 6 C COMPILER OUTPUT FILES

This chapter describes the files that the CC78K0R outputs.

The CC78K0R outputs the following files.

- Object Module File
- Assembler Source Module File
- Error List File
- Preprocess List File
- Cross-reference List File

6.1 Object Module File

The object module file is a binary image file containing C source program compilation results.

If the debug data output option (-g) has been specified, the object module file will also contain debug data.

6.2 Assembler Source Module File

The assembler source module file is an ASCII image list of C source program compilation results, and is a source module file in assembly language that corresponds to the target C source program.

It can also include the C source program to this file as comments by setting the assembler source module file creation specification option (-sa).

[Output format]

```
    ; 78K0R C Compiler V(1)x.xx Assembler Source    Date:(2)xx xxx xx
Time:(3)xx:xx:xx

    ; Command   : (4)-cf1166a0 prime.c –sa
    ; In-file   : (5)prime.c
    ; Asm-file  : (6)prime.asm
    ; Para-file : (7)

    $PROCESSOR ( (8)f1166a0 )
(9) $DEBUG
(10)$NODEBUGA
(11)$KANJICODE  SJIS
(12)$TOL_INF    03FH , 100H , 00H , 00H , 00H

(13)$DGS    FIL_NAM , .file ,       034H , 0FFFEH , 03FH , 067H , 01H , 00H
    :
(14)        EXTRN   _@RTARG0
                :
    ; line (15)1        : (16)#define  TRUE    1
    ; line (15)2        : (16)#define  FALSE   0
    ; line (15)3        : (16)#define  SIZE    200
                :
(14)_main :
(17)$DGL    1 , 14
(14)        push    hl                          ; (21) [ INF ] 1 , 1
(14)        subw    sp , #08H                   ; (21) [ INF ] 2 , 1
(14)        movw    ax , sp                     ; (21) [ INF ] 2 , 1
(14)        movw    hl , ax                     ; (21) [ INF ] 1 , 1
                :
(18)??bf_main :
                :
    ; (22)*** Code Information ***
    ;
    ; (23)$FILE   C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\CC78K0R
\prime.c
    ;
    ; (24)$FUNC   main ( 8 )
    ; (25)        bc = ( void )
    ; (26)        CODE SIZE = 116 bytes , CLOCK_SIZE = 86 clocks , STACK_SIZE
= 16 bytes
    ;
    ; (27)CALL    printf ( 18 )
    ; (28)        bc = ( pointer:ax , int : [ sp + 2 ] )
    ;
    ; (27)$CALL   putchar ( 20 )
    ; (28)        bc = ( int : ax ) ;
    ;
    ; (27)$CALL   printf ( 25 )
    ; (28)        bc = ( pointer : ax , int : [ sp + 2 ] )
    ;
    ; (24)$FUNC   printf ( 31 )
    ; (25)        bc = ( pointer s : ax , int i : [ sp + 4 ] )
    ; (26)       CODE SIZE = 23 bytes , CLOCK_SIZE = 22 clocks , STACK_SIZE =
14 bytes
    ;
    ; (24)$FUNC   putchar ( 41 )
    ; (25)        bc = ( char c : x )
    ; (26)       CODE SIZE = 16 bytes , CLOCK_SIZE = 18 clocks , STACK_SIZE =
6 bytes

    ; Target chip : (19)uPD78F1166_A0
    ; Device file : (20)Vx.xx
```

Item Number	Description	Number of Columns	Format
(1)	Version number	4 (fixed)	Displayed in "*x.yz*" format
(2)	Date	11 (fixed)	System date (Displayed in "*DD Mmm YYYY*" format)
(3)	Time	8 (fixed)	System time (Displayed in "*HH:MM:SS*" format)
(4)	Command line	-	Outputs the command line contents following "CC78K0R". Contents after column 80 are output beginning at column 15 on the next line. A semicolon (;) is output to column 1. One or more white-space characters or tabs are replaced by a single white-space character.
(5)	C source module file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".c" is attached as the file type (extension). Contents after column 80 are output beginning at column 15 on the next line. A semicolon (;) is output to column 1.
(6)	Assembler source module file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".asm" is attached as the file type (extension). Contents after column 80 are output beginning at column 15 on the next line. A semicolon (;) is output to column 1.
(7)	Parameter file contents	-	Outputs the parameter file contents. Contents after column 80 are output beginning at column 15 on the next line. A semicolon (;) is output to column 1. One or more white-space characters or tabs are replaced by a single white-space character.
(8)	Device type	Maximum 6 (variable)	This character string is specified via the -c option.
(9)	Debug data	Maximum 8 (variable)	Outputs DEBUG control. Output is either $DEBUG or $NODEBUG.
(10)	Debug information control of assembler	9 (fixed)	Outputs NODEBUGA control. Output is $NODEBUGA.

Item Number	Description	Number of Columns	Format
(11)	Kanji type information	Maximum 15 (variable)	Outputs the kanji code (2-byte code) type. Output is $KANJICODE SJIS, $KANJICODE EUC, or $KANJICODE NONE.
(12)	Tool information	37 (fixed)	Outputs tool information, version number, error information, specified options, etc. (information starts with $TOL_INF).
(13)	Symbol information	-	Outputs symbol information (information starts with $DGS). This information is output only when the debug data output option has been specified. Even then, it is not output if the -g1 option has been specified.
(14)	Assembler source	-	Outputs an assembler source file containing the compilation results.
(15)	Line number	4 (fixed)	Outputs the C source module file's line numbers as right-aligned decimal value with zeros suppressed.
(16)	C source	-	This is the input C source image. Contents after column 80 are output beginning at column 16 on the next line. A semicolon (;) is output to column 1.
(17)	Line number information	-	Line number for line number entry (information starts with $DGL) This information is output only when the debug data output option has been specified. Even then, it is not output if the -g1 option has been specified.
(18)	Labels for symbol information creation	Maximum 34 (variable)	Outputs function label information (information starts with ??). This information is output only when the debug data output option has been specified.
(19)	Target device for this compiler	Maximum 15 (variable)	Displays the target device as specified via command line option (-c) or the source file.
(20)	Device file version	6 (fixed)	Displays the version number of the input device file.
(21)	Size, clock	-	Outputs size and clock for output instructions. (Information starting with ;[INF]). The number of clocks when accessing the internal RAM area or SFR area, or when not accessing for data, is output. For the conditional branch instruction, the number of clocks when the condition is established is output. Hazards are not considered. Note, therefore, that the output clock count is different from the actual clock count. It is just a reference value.
(22)	Function information (start)	-	Indicates start of function information.
(23)	Function information (file name)	-	Outputs target source file name with full path. (Information starting with ;$FILE).

Item Number	Description	Number of Columns	Format
(24)	Function information (definition function)	-	Outputs function name and defined line number as decimal code. (Information starting with ;$FUNC).
(25)	Function information (return value, argument of definition function)	-	Outputs the definition function's return value register and argument information (register or stack position).
(26)	Function information (definition function's size, clock, stack)	-	Outputs the size, clock, and maximum consumption stacks calculated statically for the definition function. Only the stack size used by a function itself is shown here. If a function calls another function, the stack size used by the called function is not added to the stack size of the calling function. CLOCK_SIZE is the result to which the number of clocks in item (21) is added.
(27)	Function information (call function)	-	Outputs the function name and function call line number as decimal code. (Information starting with ;$CALL).
(28)	Function information (Call function's return value, argument)	-	Outputs return value register and argument information during function call (register or stack position).

6.3 Error List File

An error list file contains messages regarding any errors and warnings that occurred during compilation.

The C source program can be added to the error list by specifying a compiler option. An error list file that contains a C source program can be used as a C source module file by revising the C source program and deleting comments, such as the list header.

6.3.1 Error list file with C source

[Output format]

```
/*
78K0R C Compiler V(1)x.xx Error List    Date:(2)xx xxx xx   Time:(3)xx:xx:xx

Command          : (4)-cf1166a0 prime.c -se
C-file           : (5)prime.c
Err-file         : (6)prime.cer
Para-file        : (7)
*/

(8)#define      TRUE    1
(8)#define      FALSE   0
(8)#define      SIZE    200

(8)char     mark [ SIZE + 1 ] ;

(8)void main ( void ) {
(8)     int      i , prime , k , count ;
(8)     cont = 0 ;
  *** CC78K0R error (9)E0711 : (10)Undeclared 'cont' ; function 'main'
(8)     for ( i = 0 ; i <= SIZE ; i++ )
(8)             mark [ i ] = TRUE ;
(8)     for ( i = 0 ; i <= SIZE ; i++ ) {
(8)             if ( mark [ i ] ) {
                        prime = i + i + 3 ;
                        printf ( "%6d" , prime ) ;
  *** CC78K0R warning (9)W0745 : (10)Expected function prototype
           :
/*
 (11)Target chip : uPD78F1166_A0
 (12)Device file : Vx.xx
Compilation complete, (13)1 error(s) and (14)5 warning(s) found.
*/
```

Item Number	Description	Number of Columns	Format
(1)	Version number	4 (fixed)	Displayed in "*x.yz*" format
(2)	Date	11 (fixed)	System date (Displayed in "*DD Mmm YYYY*" format)
(3)	Time	8 (fixed)	System time (Displayed in "*HH:MM:SS*" format)
(4)	Command line	-	Outputs the command line contents following "CC78K0R". Contents after column 80 are output beginning at column 13 on the next line. One or more white-space characters or tabs are replaced by a single white-space character.
(5)	C source module file name	Number of characters enabled by OS (variable)	Outputs the specified file name. If the file type is omitted, ".c" is attached as the file type (extension). Contents after column 80 are output beginning at column 13 on the next line.
(6)	Error list file name	Number of characters enabled by OS (variable)	Outputs the specified file name. If the file type is omitted, ".cer" is attached. Contents after column 80 are output beginning at column 13 on the next line.
(7)	Parameter file contents	-	Outputs the parameter file contents. Contents after column 80 are output beginning at column 13 on the next line. One or more white-space characters or tabs are replaced by a single white-space character.
(8)	C source	-	This is the input C source image. Contents after column 80 are not wrapped to the next line.
(9)	Error message number	5 (fixed)	Outputs error numbers in the "*#nnnn*" format. "F" is output if "#" is an abort error, "E" if it is a fatal error, "C" if is an Internal error, and "W" if it is a warning. "nnnn" (the error number) is displayed as a 4-digit decimal number (no zero suppression).
(10)	Error message	-	See "CHAPTER 9 ERROR MESSAGES". Contents after column 80 are not wrapped to the next line.
(11)	Target device for this compiler	Maximum 15 (variable)	Displays the target device as specified via command line option (-c) or the source file.
(12)	Device file version	6 (fixed)	Displays the version number of the input device file.
(13)	Number of errors	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.
(14)	Number of warnings	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.

6.3.2 Error list file with error message only

[Output format]

```
(1)prime.c ( (2)18 ) : CC78K0R warning (3)W0745 : (4)Expected function prototype
(1)prime.c ( (2)20 ) : CC78K0R warning (3)W0745 : (4)Expected function prototype
(1)prime.c ( (2)26 ) : CC78K0R warning (3)W0622 : (4)No return value
(1)prime.c ( (2)37 ) : CC78K0R warning (3)W0622 : (4)No return value
(1)prime.c ( (2)44 ) : CC78K0R warning (3)W0622 : (4)No return value

Target chip : (7)uPD78F1166_A0
Device file : (8)Vx.xx

Compilation complete, (5)0 error(s) and (6)5 warning(s) found.
```

Item Number	Description	Number of Columns	Format
(1)	C source module file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".c" is attached as the file type (extension).
(2)	Line number	5 (fixed)	Outputs a right-aligned decimal value with zeros suppressed.
(3)	Error message number	5 (fixed)	Outputs the error message number in "#*nnnn*" format. "F" is output if "#" is an abort error, "E" if it is a fatal error, "C" if is an internal error, and "W" if it is a warning. "nnnn" (the error number) is displayed as a 4-digit decimal number (no zero suppression).
(4)	Error message	-	See "CHAPTER 9 ERROR MESSAGES".
(5)	Number of errors	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.
(6)	Number of warnings	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.
(7)	Target device for this compiler	Maximum 15 (variable)	Displays the target device as specified via command line option -c or the source file.
(8)	Device file version	6 (fixed)	Displays the version number of the input device file.

6.4 Preprocess List File

The preprocess list file is an ASCII image file that contains results of C source program preprocessing only.

When specifying the -k option, a preprocess list file can be used as a C source module file unless "n" has been specified as the processing type. When the -kd option is specified, the list with #define expansion is output.

[Output format]

<When PAGEWIDTH=80>

```
/*
78K0R C Compiler V(1)x.xx Preprocess List  Date:(2)xx xxx xx   Page:(3)xxx

Command :       (4)-cf1166a0 prime.c -p -lw80
In-file :       (5)prime.c
PPL-file :      (6)prime.ppl
Para-file :     (7)
*/

    (8)1 :  (9)#define  TRUE    1
    (8)2 :  (9)#define  FALSE   0
    (8)3 :  (9)#define  SIZE    200
    (8)4 :  (9)
    (8)5 :  (9)char     mark [ SIZE + 1 ] ;
    (8)6 :  (9)
/*
    (10)Target chip : uPD78F1166_A0
    (11)Device file : Vx.xx
*/
```

Item Number	Description	Number of Columns	Format
(1)	Version number	4 (fixed)	Displayed in "*x.yz*" format
(2)	Date	11 (fixed)	System date (Displayed in "*DD Mmm YYYY*" format)
(3)	Number of pages	4 (fixed)	Outputs a right-aligned decimal number with zeros suppressed.
(4)	Command line	-	Outputs the command line contents following "CC78K0R". Contents that exceed the line length are output beginning at column 13 on the next line. One or more white-space characters or tabs are replaced by a single white-space character.
(5)	C source module file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".c" is attached as the file type (extension). Contents that exceed the line length are output beginning at column 13 on the next line.
(6)	Preprocess list file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".ppl" is attached. Contents that exceed the line length are output beginning at column 13 on the next line.

Item Number	Description	Number of Columns	Format
(7)	Parameter file contents	-	Outputs the parameter file contents. Contents that exceed the line length are output beginning at column 13 on the next line. A semicolon ";" is output to column 1. One or more white-space characters or tabs are replaced by a single white-space character.
(8)	Line number	5 (fixed)	Outputs a right-aligned decimal value with zeros suppressed.
(9)	C source	-	This is the input C source. Contents that exceed the line length are output beginning at column 9 on the next line.
(10)	Target device for this compiler	Maximum 15 (variable)	Indicates the target device that is specified by a command line option or in a source file
(11)	Device file version	6 (fixed)	Displays the version number of the input device file.

6.5 Cross-reference List File

Cross-reference list files contain lists of identifiers such as declarations, definitions, referenced functions, and variables. They also include other information, such as attributes and line numbers. These are output in the order they are found.

[Output format]

<When PAGEWIDTH=80>

```
78K0R C Compiler V(1)x.xx Cross reference List  Date:(2)xx xxx xxxx
Page:(3)xxx

Command         : (4) -cf1166a0 prime.c -x -lw80
In-file         : (5)prime.c
Xref-file       : (6)prime.xrf
Para-file       : (7)
Inc-file        : [ n ] (8)

(9)ATTRIB    (10)MODIFY  (11)TYPE    (12)SYMBOL  (13)DEFINE  (14)REFERENCE

EXTERN      NEAR        array       mark        5           14  16  22
EXTERN      FAR         func        main        7
AUTO1                   int         i           9           13  13  13  14
                                                            15  15  15  16
                                                            17  17  21
AUTO1                   int         prime       9           17  18  21  21
AUTO1                   int         k           9           21  21  21  22
AUTO1                   int         count       9           11  19  20  25
  :
/*(15)Target chip : uPD78F1166_A0
  (16)Device file : Vx.xx */
```

Item Number	Description	Number of Columns	Format
(1)	Version number	4	Displayed in "*x.yz*" format
(2)	Date	11 (fixed)	System date (Displayed in "*DD Mmm YYYY*" format)
(3)	Number of pages	4 (fixed)	Outputs a right-aligned decimal number with zeros suppressed.
(4)	Command line	-	Outputs the command line contents following "CC78K0R". Contents that exceed the line length are output beginning at column 13 on the next line. One or more white-space characters or tabs are replaced by a single white-space character.
(5)	C source module file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".c" is attached as the file type (extension). Contents that exceed the line length are output beginning at column 13 on the next line.
(6)	Cross-reference list file name	Number of characters enabled by OS	Outputs the specified file name. If the file type is omitted, ".xrf" is attached. Contents that exceed the line length are output beginning at column 13 on the next line.

Item Number	Description	Number of Columns	Format
(7)	Parameter file contents	-	Outputs the parameter file contents. Contents that exceed the line length are output beginning at column 13 on the next line. One or more white-space characters or tabs are replaced by a single white-space character.
(8)	Include file	Number of characters enabled by OS	Outputs the file name specified in the C source. "n" is a number starting with "1" that indicates the include file number. Contents that exceed the line length are output beginning at column 13 on the next line. This line is not output when there is no include file.
(9)	Symbol attribute	6 (fixed)	Displays the symbol attributes. An external variable is displayed as EXTERN, an external static variable as EXSTC, an internal static variable as INSTC, an auto variable as AUTOnn, a register variable as REGnn (where nn is the scope value, a numerical value that begins with "1"), an external typedef declaration as EXTYP, an internal typedef declaration as INTYP, a label as LABEL, a structure or union tag as TAG, a member as MEMBER, and a function parameter as PARAM.
(10)	Symbol qualifier attributes	6 (fixed)	Displays the symbol qualifier attributes (left-aligned). A const variable is displayed as CONST, a volatile variable as VLT, a callt function as CALLT, an sreg-bit variable as SREG, an sfr variable as RWSFR, a read-only sfr variable as ROSFR, a write-only sfr variable as WOSFR, an interrupt function as VECT, functions and variables allocated in near area as NEAR, functions and variables allocated in far area as FAR.
(11)	Symbol type	7 (fixed)	Displays the symbol type. Types include char, int, short, long, and field. "u" is added at the start for unsigned type. Additional types include void, float, double, ldouble (long double), func, array, pointer, struct, union, enum, bit, inter, and #define.
(12)	Symbol name	15 (fixed)	If the symbol name exceeds 15 characters and fit into a line, that name is output as it is. If it exceeds 15 characters and one line, the excess is output from column 23 on the next line and items (13) and (14) are output from column 39 on the next line.
(13)	Symbol definition line number	8 (fixed)	This outputs the line number and file name defined for the symbol, and is displayed as: line number (5-digit): include file number (2-digit)
(14)	Symbol reference line number	8 (fixed)	This outputs the line number and file name that reference the symbol, and is displayed as: line number (5-digit): include file number (2-digit) If the line contents exceed the line length, the remaining contents are output beginning at column 48 of the next line.
(15)	Target device for this compiler	Maximum 15 (variable)	Displays the target device as specified via command line option -c or the source file.
(16)	Device file version	6 (fixed)	Displays the version number of the input device file.

CHAPTER 7   USING C COMPILER

This chapter introduces methods for efficiently using the CC78K0R.

7.1   Efficient Operation (EXIT Status Function)

When the compilation ends, the CC78K0R returns the top error level generated during compilation to the operating system as the EXIT status.

The EXIT status is shown below.

Table 7-1 EXIT Status

Processing	EXIT Statuses
Normal operation	0
WARNING occurs	0
FATAL ERROR occurs	1
ABORT	2

If PM+ is not used and the CC78K0R is started in the command line, efficient operation can be further improved by using the status in a batch file.

[Use Example]

```
cc78k0r -cf1166a0 %1
IF ERRORLEVEL 1 GOTO ERR
cc78k0r -cf1166a0 %2
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
ERR
echo Some error found.
EXIT
```

[Description]

- When the C source passed to %1 was compiled, a fatal error was generated. Essentially, the process continues after an error message was output. But using the 1 returned in the EXIT status, execution can be stopped without processing the next C source in %2.

7.2 Setting Up Development Environment (Environment Variables)

The CC78K0R supports the following environment variables.

Table 7-2 Environment Variables

Environment Variables	Description
PATH	Search path for executable forms
INC78K0R	Search path for include files
TMP	Search path for temporary files
LANG78K	Type of kanji code (2-byte code) (can be specified by the -zs, -ze, or -zn option) (sjis: shift JIS code, euc: EUC code, none: no 2-byte codes)
LIB78K0R	Search path for libraries

[Use Example]

<When using command prompt>

```
; AUTOEXEC.BAT
PATH C:\Program Files\NEC Electronics
Tools\CC78K0R\Vx.xx\bin;C:\bat;C:\cc78k0r;C:\tool
VERIFY ON
BREAK ON
SET INC78K0R=C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r
SET LIB78K0R=C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\lib78k0r
SET TMP=C:\tmp
SET LANG78K=sjis
```

[Description]

- Executable files are searched in the sequence of C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\bin, C:\bat, C:\cc78k0r, C:\tool by path specification.
- Include files are searched from C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\inc78k0r.
  If no setting is made, search is performed from C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\inc78k0r (if the CC78K0R is installed to C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*).
- Library files are searched from C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\lib78k0r during linking.
  If no setting is made, search is performed from C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*\lib78k0r (if the CC78K0R is installed to C:\Program Files\NEC Electronics Tools\CC78K0R\V*x.xx*).
- Temporary files are created in C:\tmp.
- Shift JIS code is used as kanji code.

[Caution]

Do not set environment variables when using PM+.

7.3 Interrupting Compilation

If compiling was started from the command line, the compilation can be interrupted by the command key input (CTRL-C). If 'break on' was specified, control returns to the operating system unrelated to the timing of the key input. And for 'break off,' control returns to the operating system only when the screen is displayed. Then all of the open temporary files and output files are deleted.

If you want to stop a build (make) in PM+, select [Stop build] in the [Run] menu in the PM+, or click the [Stop Build] button in the tool bar. When building in PM+, command key input is not accepted.

CHAPTER 8 STARTUP ROUTINES

To execute a C language program, a program is needed to activate ROMization for inclusion in the system and the user program (main function). This program is called the startup routine.

To execute a program written by a user, a startup routine must be created for that program. The CC78K0R provides the object files of the startup routines that include the processing required before program execution and the source files (assembly source) of the startup routines that the user can adapt to the system. By linking the object file of the startup routine to the user program, an executable program can be created even if the user does not describe the execution preprocess.

This chapter describes the contents, uses, and improvements of the startup routines.

8.1 File Organization

The files related to a startup routine are stored in the folder src\cc78k0r of the C compiler package.

```
Program Files\NEC Electronics Tools\
    |
    |---- CC78K0R\Vx.xx\bin\
    |
    |---- CC78K0R\Vx.xx\doc\
    |
    |---- CC78K0R\Vx.xx\hlp\
    |
    |---- CC78K0R\Vx.xx\inc78k0r\
    |
    |---- CC78K0R\Vx.xx\lib78k0r\
    |
    |---- CC78K0R\Vx.xx\smp78k0r\CC78K0R\
    |
    |---- CC78K0R\Vx.xx\src\cc78k0r\
                  |
                  |-- bat\
                  |                -->   Folders that contain files related to startup routines
                  |-- lib\
                  |
                  |-- src\
```

The contents of the folders under src\cc78k0r are shown next.

8.1.1 “bat” folder contents

A batch file in this folder cannot be used in PM+.

Use these batch files only when the source, such as for a startup routine, must be modified.

Table 8-1 “bat” Folder Contents

Batch File Name	Description
mkstup.bat	Assemble batch file for startup routine
reprom.bat	Batch file for updating rom.asm[Note 1]
repgetc.bat	Batch file for updating getchar.asm
repputc.bat	Batch file for updating putchar.asm
repputcs.bat	Batch file for updating _putchar.asm
repselo.bat	Batch file for updating setjmp.asm and longjmp.asm (the compiler reserved area is saved)[Note 2]
repselon.bat	Batch file for updating setjmp.asm and longjmp.asm (the compiler reserved area is not saved)[Note 2]
repvect.bat	Batch file for updating vect*.asm

Note 1 Since ROMization routines are in the library, the library is also updated by this batch file.

Note 2 The setjmp and longjmp that save the compiler reserved area (saddr area secured for KREGxx, etc.), and the setjmp and longjmp that do not save the compiler reserved area (only the registers are saved) are created.

8.1.2 “lib” folder contents

The lib folder contains the object files that were assembled from the source files of startup routines and libraries. This object file can be linked with programs for any 78K0R target device. If the code modification is not especially needed, link the default object file as is. This object file is overwritten when batch file mkstup.bat, which is provided by the CC78K0R, is executed.

Table 8-2 “lib” Folder Contents

File Name			File Role
Normal	Boot Area	Flash Area	
cl0rm.lib cl0rl.lib cl0rmf.lib cl0rlf.lib cl0rxm.lib cl0rxl.lib	cl0rm.lib cl0rl.lib cl0rmf.lib cl0rlf.lib cl0rxm.lib cl0rxl.lib	cl0rme.lib cl0rle.lib cl0rmfe.lib cl0rlfe.lib cl0rxme.lib cl0rxle.lib	Library (runtime and standard libraries)
s0rm.rel s0rml.rel s0rl.rel s0rll.rel	s0rmb.rel s0rmlb.rel s0rlb.rel s0rllb.rel	s0rme.rel s0rmle.rel s0rle.rel s0rlle.rel	Object files for startup routines

For details on the file contents, refer to "2.5.1 Library files".

8.1.3 “src” folder contents

The src folder contains the assembler sources of the startup routines, ROM routines, error processing routines, and standard library functions (a portion). If the source must be modified to conform to the system, the object files for linking can be created by modifying this assembler source and using a batch file in the bat folder to assemble.

Table 8-3 “src” Folder Contents

Startup Routine Source File Name	Description
cstart.asm[Note]	Source file for startup routine (when standard library is used)
cstartn.asm[Note]	Source file for startup routine (when standard library is not used)
rom.asm	Source file for ROMization routine
_putchar.asm	_putchar function
putchar.asm	putchar function
getchar.asm	getchar function
longjmp.asm	longjmp function
setjmp.asm	setjmp function
vectxx.asm	Vector source for each interrupt (xx: vector address)
def.inc	For setting library according to type
macro.inc	Macro definition for each typical pattern
vect.inc	Start address of flash memory area branch table
library.inc	Selection of library assigned to boot area explicitly

Note A file name with "n" added is a startup routine that does not have standard library processing. Use only if the standard library will not be used.
cstartb*.asm is a startup routine for boot area and cstarte*.asm is a startup routine for flash area.

8.2 Batch File Description

8.2.1 Batch files for creating startup routines

The mkstup.bat in the bat folder is used to create the object file of a startup routine.

The assembler in the RA78K0R Assembler Package is required for mkstup.bat. Therefore, if PATH is not specified, specify it and run.

How to use this file is described next.

[How To Use]

- Execute the following command line in the src\cc78k0r\bat folder containing mkstup.bat.

```
mkstup device-typeNote
```

Note   Refer to the user's manual of the device used or "Device Files Operating Precautions".

[Use Example]

- The startup routine to be used is created when the target device is the uPD78F1166_A0.

```
mkstup f1166a0
```

The mkstup.bat batch file is stored in the form that overwrites the object file of the startup routine in the lib folder at the same level as the bat folder as shown below.

The startup routine that is required to link the object file is output to each folder.

The names of the object files created in lib are shown below.

```
┌── bat ── mkstup.bat
│
└── lib ── s0rm.rel
           s0rmb.rel
           s0rme.rel
           s0rml.rel
           s0rmlb.rel
           s0rmle.rel
           s0rl.rel
           s0rlb.rel
           s0rle.rel
           s0rll.rel
           s0rllb.rel
           s0rlle.rel
```

8.3 Startup Routines

8.3.1 Overview of startup routines

A startup routine makes the preparations needed to execute the C source program written by the user. By linking to a user program, a load module file that achieves the objective can be created.

(1) Function

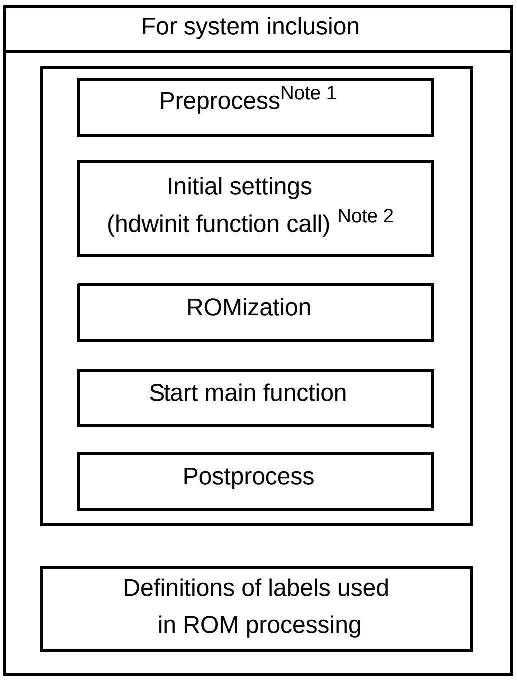
Memory initialization, ROMization for inclusion in the system, and the starting and ending processes for the C source program are performed.

- ROMization

  The initial values of the external variables, static variables, and sreg variables defined in the C source program are located in ROM. However, the variable values cannot be rewritten; only placed in ROM as is. Therefore, the initial values located in ROM must be copied to RAM. This process is called a ROMization. When a program is written to ROM, it can be run by a microcontroller.

(2) Configuration

The figure below shows the programs related to the startup routines and their configurations.


For system inclusion
Preprocess Note 1
Initial settings (hdwinit function call) Note 2
ROMization
Start main function
Postprocess
Definitions of labels used in ROM processing

Note 1 If the standard library is used, the processing related to the library is performed first. Files that do not have an “n” appended at the end of the name in the startup routine source file are processed in relation to the standard library. Files with the appended “n” are not processed.

Note 2 The hdwinit function is a function created when needed by the user as the function to initialize a peripheral device (sfr). By creating the hdwinit function, the timing of the initial settings can be sped up (the initial settings can be made in the main function). If the user does not create the hdwinit function, the process returns without doing anything.

cstart.asm and cstartn.asm have nearly identical contents.

The table below shows the differences between cstart.asm and cstartn.asm.

Type of Startup Routine	Uses Library Processing
cstart.asm	Yes
cstartn.asm	No

(3) Uses of startup routines

Thetable below lists the names of the object files for the source files provided by the CC78K0R.

File Type	Source File	Object File
Startup routine	cstart*.asm^Note1, 2	s0r*.rel^Note2, 3, 4
ROM file	rom.asm	Included in library

Note 1 *: If the standard library is not used, "n" is added. If used, the character is not added.

Note 2 *: "b" is startup routine for boot area, and "e" is that for flash area.

Note 3 *: If a fixed area in the standard library is used, "l" is added.

Note 4 *: if the small model or medium model is specified, "m" is added. if the large model is specified, "l" is added.
if the small model or medium model is specified, use the startup routine that "l" is added when variables are allocated in the far area.

Remark rom.asm defines the label indicating the final address of the data copied by ROMization.
The object of the rom.asm is included in the library.

8.3.2 Description of sample program (cstart.asm)

This section uses cstart.asm and rom.asm as examples to describe the contents of the startup routines. A startup routine consists of the preprocessing, initial settings, ROMization processing, starting the main function, and postprocessing.

Remark cstart is called in the format added _@ to its head.

(1) Preprocessing

Preprocessing in cstart.asm is described.

<cstart.asm preprocessing>

```
        NAME    @cstart

$INCLUDE ( def.inc )                                                    ; (1)
$INCLUDE ( macro.inc )
                                                                        ; (2)
BRKSW     EQU   1 ; brk , sbrk , calloc , free , malloc , realloc function use
EXITSW    EQU   1 ; exit , atexit function use
RANDSW    EQU   1 ; rand , srand  function use
DIVSW     EQU   1 ; div function use
LDIVSW    EQU   1 ; ldiv function use
FLOATSW   EQU   1 ; floating point variables use
STRTOKSW  EQU   1 ; strtok function use

        PUBLIC  _@cstart , _@cend                                       ; (3)

$_IF ( BRKSW )
        PUBLIC  _@BRKADR , _@MEMTOP , _@MEMBTM
          :
$ENDIF
        EXTRN   _main , _@STBEG , _hdwinit , _@MAA                      ; (4)
$_IF ( EXITSW )
        EXTRN   _exit
$ENDIF
                                                                        ; (5)
        EXTRN   _?R_INIT , _?RLINIT , _?R_INIS , _?DATA , _?DATAL , _?DATS
@@DATA  DSEG    BASEP ; near                                            ; (6)

$_IF ( EXITSW )
_@FNCTBL :      DS    4 * 32
_@FNCENT :      DS    2
   :
_@MEMTOP :      DS    32
_@MEMBTM :
$ENDIF
```

(1) Including include files

def.inc -->	For setting library according to the type.
macro.inc -->	Macro definition for each typical pattern.

(2) Library switch

If standard libraries in comments are not used, by changing the EQU definition to 0, the space secured for the processing of unused libraries and for use by the library can be conserved. The default is set to use everything (In a startup routine without library processing, this processing is not performed).

(3) Symbol definitions

The symbols used when using the standard library are defined.

(4) External reference declaration of symbol for stack resolution

- The public symbol (_@STBEG) for stack resolution is an external reference declaration. _@STBEG has the value of the last address in the stack area + 1.
- _@STBEG is automatically generated by specifying the symbol generation option (-s) for stack resolution in the linker. Therefore, always specify the -s option when linking. In this case, specify the name of the area used in the stack. If the name of the area is omitted, the RAM area is used, but the stack area can be located anywhere by creating a link directive file. For memory mapping, refer to the user's manual of the target device.

  An example of a link directive file is shown below. The link directive file is a text file created by the user in an ordinary editor (for details about the description method, refer to RA78K0R Assembler Package Operation User's Manual).

  <Example when -sSTACK is specified in linking>
  Create lk78k0r.dr (link directive file). Since ROM and RAM are allocated as default operations by referencing the memory map of the target device, it is not necessary to specify ROM and RAM allocations unless they should be changed. For link directive, refer to lk78k0r.dr in the smp78k0r\cc78k0r folder.

```
               First address  Size
                       |        |
memory  SDR     : ( 0xFFE20h , 0000098h )
memory  STACK   : ( 0xxxxxxh , 0xxxxxxh ) <-- Specify the first
                                              addressand size here,
                                              then specify lk78k0r.dr
                                              by the -d linker option.
                                              (Example: -dlk78k0r.dr)
merge @@INIS    : = SDR
merge @@DATS    : = SDR
merge @@BITS    : = SDR
```

(5) External reference declaration of label for ROMization processing

The label for ROMization processing is defined in the postprocessing section.

(6) Securing area for standard library

The area used when using the standard library is secured.

(2) Initial settings

The initial settings in cstart.asm are described.

<Initial settings in cstart.asm>

```
@@VECT00        CSEG    AT      0                                   ; (1)
        DW      _@cstart

@@LCODE CSEG    BASE
_@cstart :
        SEL     RB0                                                 ; (3)
        MOV     A , #_@MAA                                          ; (2)
        MOV1    CY , A.0
        MOV1    MAA , CY
        MOVW    SP , #LOWW _@STBEG      ; SP <-stack begin address  ; (4)
        CALL    !!_hdwinit                                          ; (5)
          :
$_IF ( BRKSW OR EXITSW OR RANDSW OR FLOATSW )
        CLRW    AX
$ENDIF
          :
```

(1) Reset vector setting

The segment of the reset vector table is defined as follows. The first address of the startup routine is set.

```
@@VECT00        CSEG    AT      0000H
                DW      _@cstart
```

(2) Mirror area setting

The mirror area is set.

For the mirror area, refer to the user's manual of the target device.

(3) Register bank setting

Register bank RB0 is set as the work register.

(4) Stack pointer (SP) setting

_@STBEG is set in the stack pointer.

_@STBEG is automatically generated by specifying the symbol generation option (-s) for stack resolution in the linker.

(5) Hardware initialization function call

The hdwinit function is created when needed by the user as the function for initializing a peripheral device (SFR). By creating this function, initial settings can be made to match the user's objectives.

If the user does not create the hdwinit function, the process returns without doing anything.

(3) ROMization processing

The ROMization processing in cstart.asm is described.

<ROMization processing>

```
; copy external variables having initial value
$_IF ( _ESCOPY )
        MOV     ES , #HIGHW _@R_INIT
$ENDIF
        MOVW    HL , #LOWW _@R_INIT
        MOVW    DE , #LOWW _@INIT
        BR      $LINIT2
LINIT1 :
$_IF ( _ESCOPY )
        MOV     A , ES : [ HL ]
$ELSE
        MOV     A , [ HL ]
$ENDIF
        MOV     [ DE ] , A
        INCW    HL
        INCW    DE
LINIT2 :
        MOVW    AX , HL
        CMPW    AX , #LOWW _?R_INIT
        BNZ     $LINIT1
```

In ROMization processing, the initial values of the external variables and the sreg variables stored in ROM are copied to RAM. The variables to be processed have the 4 types (a) to (d) shown in the following example.

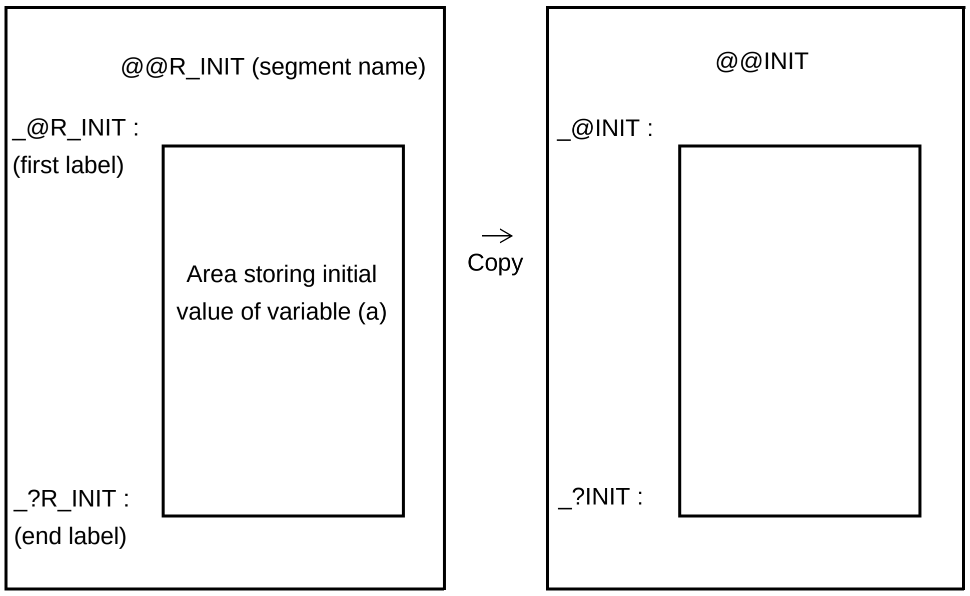
<Example>

```
char    c = 1 ;             (a) External variable with initial value
int     i ;                 (b) External variable without initial valueNote
__sreg  int     si = 0 ;    (c) sreg variable with initial value
__sreg  char    sc ;        (d) sreg variable without initial valueNote

void main ( void ) {
            :
}
```

Note The external variables without initial value and sreg variables without initial value are not copied, and zeros are written directly to RAM.

- The figure below shows the ROMization processing for (a) External variable with initial value.
  The initial value of the variable (a) is placed in @@R_INIT segment in the ROM by the CC78K0R. The ROMization processing copies this value to the @@INIT segment in RAM (the same processes are performed for the variable (c)).



- The first and end labels in the @@R_INIT segment are defined by _@R_INIT and _?R_INIT. The first and end labels in the @@INIT segment are defined by _@INIT and _?INIT.
- The variables (b) and (d) are not copied, but zeros are directly placed in the segment determined by the RAM. The tables below show the segment names of the ROM and RAM areas where the variables (a) to (d) are placed, and the first and end labels of the initial values in each segment.

<ROM Area for Initial Values>

Variable Type	Segment	First Label	End Label
External variable with initial value (a) (when allocated in near area)	@@R_INIT	_@R_INIT	_?R_INIT
External variable with initial value (a) (when allocated in far area)	@@RLINIT	_@RLINIT	_?RLINIT
sreg variable with initial value (c)	@@R_INIS	_@R_INIS	_?R_INIS

<RAM Area for Initial Values (Copy Destination)>

Variable Type	Segment	First Label	End Label
External variable with initial value (a) (when allocated in near area)	@@INIT	_@INIT	_?INIT
External variable with initial value (a) (when allocated in far area)	@@INITL	_@INITL	_?INITL
External variable without initial value (b) (when allocated in near area)	@@DATA	_@DATA	_?DATA
External variable without initial value (b) (when allocated in far area)	@@DATAL	_@DATAL	_?DATAL
sreg variable with initial value (c)	@@INIS	_@INIS	_?INIS
sreg variable without initial value (d)	@@DATS	_@DATS	_?DATS

(4) Starting main function and postprocessing

Starting the main function and postprocessing in cstart.asm are described.

<Starting main function and postprocessing>

```
            CALL    !!_main         ; main ( ) ;            ; (1)
$_IF ( EXITSW )
            CLRW    AX
            CALL    !!_exit         ; exit ( 0 ) ;          ; (2)
$ENDIF
            BR      $$
;
_@cend :
                                                            ; (3)
@@R_INIT    CSEG    UNIT64KP
_@R_INIT :
@@RLINIT    CSEG    UNIT64KP
_@RLINIT :
@@R_INIS    CSEG    UNIT64KP
_@R_INIS :
@@INIT      DSEG    BASEP
_@INIT :
@@INITL     DSEG    UNIT64KP
_@INITL :
@@DATA      DSEG    BASEP
_@DATA :
@@DATAL     DSEG    UNIT64KP
_@DATAL :
@@INIS      DSEG    SADDRP
_@INIS :
@@DATS      DSEG    SADDRP
_@DATS :
@@CALT      CSEG    CALLT0
@@CNST      CSEG    MIRRORP
@@CNSTL     CSEG    PAGE64KP
@@BITS      BSEG
;
            END
```

(1) Starting main function

The main function is called.

(2) Starting exit function

The exit function is called if needed.

(3) Definitions of segments and labels used in ROMization processing

The segments and labels used in each variable (a) to (d) (see "(3) ROMization processing") in ROMization processing are defined. A segment indicates the area that stores the initial value of each variable. A label indicates the first address in each segment.

The ROMization processing file rom.asm is described. The relocatable object file of rom.asm is in the library.

```
        NAME    @rom
;
        PUBLIC  _?R_INIT , _?RLINIT , _?R_INIS
        PUBLIC  _?INIT , _?INITL , _?DATA , _?DATAL , _?INIS , _?DATS
;
@@R_INIT        CSEG    UNIT64KP                        ; (1)
_?R_INIT :
@@RLINIT        CSEG    UNIT64KP
_?RLINIT :
@@R_INIS        CSEG    UNIT64KP
_?R_INIS :
@@INIT          DSEG    BASEP
_?INIT :
@@INITL         DSEG    UNIT64KP
_?INITL :
@@DATA          DSEG    BASEP
_?DATA :
@@DATAL         DSEG    UNIT64KP
_?DATAL :
@@INIS          DSEG    SADDRP
_?INIS :
@@DATS          DSEG    SADDRP
_?DATS :
;
        END
```

(1) Definition of labels used in ROMization processing

The labels used for each variable (a) to (d) (see "(3) ROMization processing") in ROMization processing, are defined. These labels indicate the last address of the segment storing the initial value of each variable.

8.3.3 Revising startup routines

The startup routines provided by the CC78K0R can be revised to match the target system actually being used. The essential points about revising these files are explained in this section.

(1) When revising startup routine

The essential points about revising a startup routine source file are described. After revising, use the batch file mkstup.bat in the src\cc78k0r\bat folder to assemble the revised source file (cstart*.asm) (*: alphanumeric symbols).

- Symbols used in standard library functions

  If the library functions listed in the table below are not used, the symbols corresponding to each function in the startup routine (cstart.asm) can be deleted. However, since the exit function is used in the startup routine, _@FNCTBL and _@FNCENT cannot be deleted (if the exit function is deleted, these symbols can be deleted). The symbols in the unused library functions can be deleted by changing the library switch.

Library Function Name	Symbols Used
brk sbrk malloc calloc realloc free	_errno _@MEMTOP _@MEMBTM _@BRKADR
exit	_@FNCTBL _@FNCENT
rand srand	_@SEED
div	_@DIVR
ldiv	_@LDIVR
strtok	_@TOKPTR
atof strtod Mathematical function Floating-point runtime library	_errno

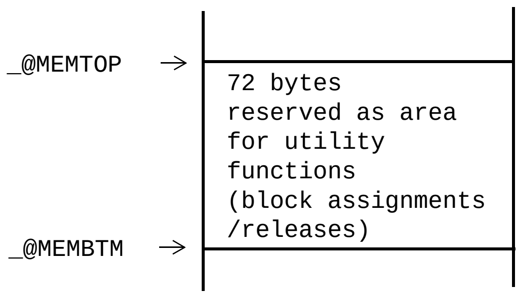
- Areas that are used for utility functions (block assignments/releases)

  If the size of the area used by a utility function (block assignment/release) is defined by the user, this is explained in the following example.

  [Example]

  If you want to reserve 72 bytes for use by utility functions (block assignments/releases), make the following changes to the initial settings of the startup routine.

```
_@MEMTOP :   DS   73
_@MEMBTM :
```



_@MEMTOP →
72 bytes reserved as area for utility functions (block assignments /releases)
_@MEMBTM →

Add one byte to the area size to be secured and then specify the value in the startup routine. In the above example, 73 bytes are secured in the startup routine, but up to 72 bytes can actually be secured for utility functions.

If the specified size is too big to be stored in the RAM area, errors may occur when linking.
In this case, decrease the size specified as shown below, or avoid by correcting the link directive file. For correction of the link directive file, see "(2) Link directive file".

[Example]

To decrease the specified size

```
_@MEMTOP :   DS   72        →  Change to 40
```

(2) Link directive file

How to create a link directive file is explained. Specify a file created using the -d option when linking to match the actual target system. Heed the following cautions when creating the file (for the detailed description method for a link directive, see RA78K0R Assembler Package Operation User's Manual).

- The CC78K0R sometimes uses a portion of the short direct address area (saddr area) in the following compiler-specific objectives.

  Specifically, this is the 44-byte area of FFEB8H to FFEDFH.

  (a) register variable when the -qr option is specified [FFEB4H to FFEC3H]

  (b) Arguments or automatic variables of norec function [FFEC4H to FFED3H]

  (c) Segment information [FFED4H to FFED7H]

  (d) Arguments of runtime library [FFED8H to FFEDFH]

  (e) Standard library task (part of the area (a) and (b)).

  Caution If the user does not use the standard library, the area (e) is not used.

The following shows an example of changing RAM size with the link directive file (lk78k0r.dr). When changing memory size, do not overlap another area. Refer to the memory map of the target device to be used when changing memory size.

<lk78k0r.dr>

```
              First address Size
memory RAM : ( 0FCF00h , 002F20h )-> Make this size larger.
memory SDR : ( 0FFE20h , 000098h )   (also change the first address if necessary)
merge @@INIS : = SDR              -> Specifies the location of the segment.
merge @@DATS : = SDR              -> Specifies the location of the segment.
merge @@BITS : = SDR              -> Specifies the location of the segment.
```

If you want to change the location of the segment, add a merge statement. If the function to revise the compiler output section name was used, the segment can be independently located (refer to CC78K0R C Compiler Language User's Manual).

If the result of changing the location of a segment does not provide enough memory for the location, change the corresponding memory statement.

(3) When using RTOS

The RX78K0R and CC78K0R provide sample programs for initialization routines (assembler format). When using the RX78K0R and CC78K0R in combination, initialization routines for both tools must therefore be modified.

For the method for modifying initialization routines, refer to the RX78K0R Functions user's manual.

8.4 ROMization Processing in Startup Module for Flash Area

The startup modules for flash differ with the ordinary startup modules in the following points.

Table 8-4 ROM Area Section for Initialization Data

Variable Type	Segment	First Label	End Label
External variable with initial value (a) (when allocated in near area)	@ER_INIT CSEG UNIT64KP	E@R_INIT	E?R_INIT
External variable with initial value (a) (when allocated in far area)	@ERLINIT CSEG UNIT64KP	E@RLINIT	E?RLINIT
sreg variable with initial value (c)	@ER_INIS CSEG UNIT64KP	E@R_INIS	E?R_INIS

Table 8-5 RAM Area Section for Copy Destination

Variable Type	Segment	First Label	End Label
External variable with initial value (a) (when allocated in near area)	@EINIT DSEG BASEP	E@INIT	E?INIT
External variable with initial value (a) (when allocated in far area)	@EINITL DSEG UNIT64KP	E@INITL	E?INITL
External variable without initial value (b) (when allocated in near area)	@EDATA DSEG BASEP	E@DATA	E?DATA
External variable without initial value (b) (when allocated in far area)	@EDATAL DSEG UNIT64KP	E@DATAL	E?DATAL
sreg variable with initial value (c)	@EINIS DSEG SADDRP	E@INIS	E?INIS
sreg variable without initial value (d)	@EDATS DSEG SADDRP	E@DATS	E?DATS

- In the startup module, the following labels are added at the head of each segment in ROM area and RAM area.

  E@R_INIT, E@R_INIS, E@INIT, E@DATA, E@INIS, E@DATS, E@INITL, E@DATAL

  Furthermore, the following labels are added if the large model is specified or variables are allocated in the far area.

  E@RLINIT, E@INITL, E@DATAL

- In the terminal module, the following labels are added at the terminal of each segment in ROM area and RAM area.

  E?R_INIT, E?R_INIS, E?INIT, E?DATA, E?INIS, E?DATS, E?RLINIT, E?INITL, E?DATAL

- The startup module copies the contents from the first label address of each segment in ROM area to the end label address -1, to the area from the first label address of each segment in RAM area.
- Zeros are embedded from E@DATA to E?DATA, and from E@DATS to E?DATS.

- Furthermore, zeros are embedded from E@DATAL to E?DATAL if the large model is specified or variables are allocated in the far area.

CHAPTER 9 ERROR MESSAGES

This chapter explains the causes of error messages output by the CC78K0R.

9.1 Error Message Format

The error message format is as follows.

```
Source-file-name (line-number) : Error-message
```

<Examples>

```
prime.c ( 8 )   : CC78K0R error E0712 : Declaration syntax
prime.c ( 8 )   : CC78K0R error E0301 : Syntax error
prime.c ( 8 )   : CC78K0R error E0701 : External definition syntax
prime.c ( 19 )  : CC78K0R warning W0745 : Expected function prototype
```

However, the C0101, C0103, and C0104 internal errors are output in the following format.

```
[xxx.c <yyy> zzz] CC78K0R error C0101 : Internal error
[xxx.c <yyy> zzz] CC78K0R error C0103 : Intermediate file error
[xxx.c <yyy> zzz] CC78K0R error C0104 : Illegal use of register
```

Remark	*xxx*.c:	source file name
	yyy:	line number
	zzz:	message

9.2 List of Error Messages

It is necessary to understand the format of an error number before using the list of error messages.

The error number indicates the type of error message and the CC78K0R processing for the error.

The error number format is as follows.

```
F/E/C/Wxxxx
```

(1) Abort error (F*xxxx*)

Compilation is always stopped if this error occurs. The object module file and assembler source file are not output.

(2) Fatal error (E*xxxx*)

If more than a specific number of this error occurs, compilation is stopped. The object module file and assembler source file are not output.

(3) Internal error (C*xxxx*)

Compilation is always stopped if this error occurs. The object module file and assembler source file are not output.

(4) Warning (W*xxxx*)

Compilation continues.

Remark *xxxx* (4-digit number):

Type	Description
From 0001	Error messages for a command line
From 0101	Error messages for an internal error and memory
From 0201	Error messages for a character
From 0301	Error messages for configuration element
From 0401	Error messages for conversion
From 0501	Error messages for an expression
From 0601	Error messages for a statement
From 0701	Error messages for a declaration and function definition
From 0801	Error messages for a preprocessing directive
From 0901	Error messages for fatal file I/O and running on an illegal operating system

Caution If the file name contains a syntax error, the file name is added to the message.

An error message is added, changed, and deleted according to the language specification of the C compiler being developed.

9.2.1 Error messages for a command line

Table 9-1 Error Messages for Command Line <from 0001>

Error Number	Error Message	
F0001	Message	Missing input file
	Cause	The input source file name was not specified.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the file name correctly.
F0002	Message	Too many input files
	Cause	Multiple input source file names are specified.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the file name correctly.
F0003	Message	Unrecognized string
	Cause	An item other than an option was specified on the interactive command line.
F0004	Message	Illegal file name *file name*
	Cause	Either the format, characters, or number of characters in the specified file name are incorrect.
F0005	Message	Illegal file specification
	Cause	An illegal file name was specified.
F0006	Message	File not found
	Cause	The specified input file does not exist.
F0007	Message	Input file specification overlapped *file name*
	Cause	Duplicate input file names were specified.
F0008	Message	File specification conflicted *file name*
	Cause	Duplicate I/O file names were specified.
F0009	Message	Unable to make file *file name*
	Cause	Since the specified output file already exists as a read-only file, it cannot be created.
F0010	Message	Directory not found
	Cause	A drive or folder not existed is included in the output file name.
F0011	Message	Illegal path
	Cause	An illegal path name was specified in the option setting the path name in the parameter.
F0012	Message	Missing parameter '*option*'
	Cause	A required parameter is not specified.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the parameter correctly.

Table 9-1 Error Messages for Command Line <from 0001>

Error Number	Error Message	
F0013	Message	Parameter not needed '*option*'
	Cause	An unnecessary option parameter was specified.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the parameter correctly.
F0014	Message	Out of range '*option*'
	Cause	The specified value of the option parameter is out of range.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the value correctly.
F0015	Message	Parameter is too long
	Cause	The number of characters in the option parameter exceeded the limit.
F0016	Message	Illegal parameter '*option*'
	Cause	There is a syntax error in the option parameter.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the option correctly.
F0017	Message	Too many parameters
	Cause	The total number of option parameters exceeds the limit.
F0018	Message	Option is not recognized '*option*'
	Cause	An incorrect option was specified.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the option correctly.
F0019	Message	Parameter file nested
	Cause	The -f option was specified in the parameter file.
	Action by User	Since a parameter file cannot be specified in a parameter file, correct them so that there is no nesting.
F0020	Message	Parameter file read error
	Cause	The parameter file read failed.
F0021	Message	Memory allocation failed
	Cause	Memory allocation failed.
W0022	Message	Same category option specified – ignored '*option*'
	Cause	Conflicting options had duplicate specifications.
	Program Processing	The option specified later is validated and processing continues.
W0023	Message	Incompatible chip name
	Cause	The device type in the command line and the device type in the source differ.
	Program Processing	The device type in the command line has priority.

Table 9-1 Error Messages for Command Line <from 0001>

Error Number	Error Message	
F0024	Message	Illegal chip specifier on command line
	Cause	The device type in the command line is incorrect.
W0029	Message	'-QC' option is not portable
	Cause	The -qc option does not conform to the ANSI standard (For details about -qc, see "CHAPTER 5 COMPILER OPTIONS").
W0031	Message	'-ZP' option is not portable
	Cause	The -zp option does not conform to the ANSI standard (For details about -zp, see "CHAPTER 5 COMPILER OPTIONS").
W0032	Message	'-ZC' option is not portable
	Cause	The -zc option does not conform to the ANSI standard (For details about -zc, see "CHAPTER 5 COMPILER OPTIONS").
F0033	Message	Same category option specified '*option*'
	Cause	Conflicting options had duplicate specifications.
	Action by User	"Please enter 'cc78k0r--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and correct the input.
W0067	Message	'*Option*' option deleted - ignored
	Cause	The deleted option was specified. '*option*' is ignored.
W0068	Message	'*Option 1*' option deleted - regarded as '*option 2*'
	Cause	'*Option 1*' was deleted, so '*option 2*' is enabled.

9.2.2 Error messages for an internal error and memory

Table 9-2 Error Messages for Internal Error and Memory <from 0101>

Error Number	Error Message	
C0101	Message	Internal error
	Cause	An internal error occurred.
	Action by User	Contact support.
E0102	Message	Too many errors
	Cause	The total number of FATAL errors exceeded 30.
	Program Processing	Processing continues, but subsequent error messages are not output. The previous errors may have caused many errors. First, remove these previous errors.
C0103	Message	Intermediate file error
	Cause	The intermediate file contains errors.
	Action by User	Contact support.
C0104	Message	Illegal use of register
	Cause	The register is incorrectly used.
E0105	Message	Register overflow : simplify expression
	Cause	The expression is too complex and no more usable registers remain.
	Action by User	Simplify the complex expression causing the error.
C0106	Message	Stack overflow '*overflow cause*'
	Cause	The stack overflowed. The cause of the overflow is the stack or heap.
	Action by User	Contact support.
E0108	Message	Compiler limit : too much automatic data in function
	Cause	The area allocated for the automatic variables of the function exceeded the limit of 64 KB.
	Action by User	Decrease the variables so that 64 KB is not exceeded.
E0109	Message	Compiler limit : too much parameter of function
	Cause	The area allocated for the parameters of the function exceeded the limit of 64 KB.
	Action by User	Decrease the parameters so that 64 KB is not exceeded.
E0110	Message	Compiler limit : too much code defined '*section name*' in file
	Cause	The area allocated for 'section name' exceeded the limit.
E0111	Message	Compiler limit : too much global data defined in file
	Cause	The area allocated for the global variables in the file exceeded the limit of 64 KB.

Table 9-2 Error Messages for Internal Error and Memory <from 0101>

Error Number	Error Message	
E0113	Message	Compiler limit: too many local labels
	Cause	Number of local labels in 1 function exceeds the process limit.
	Action by User	The function itself is too large. Divide it.
E0116	Message	Compiler limit : too many function definitions in file
	Cause	The number of function definitions in one file exceeds the processing limit.
	Action By User	Separate the function definitions into several files so as not to exceed the processing limit.
E0117	Message	Compiler limit : too many source lines in file
	Cause	The number of source lines in one file exceeds the processing limit.
	Action By User	Separate the source lines to several files so as not to exceed the processing limit.

9.2.3 Error messages for a character

Table 9-3 Error Messages for Character <from 0201>

Error Number	Error Message	
E0201	Message	Unknown character '*hexadecimal number*'
	Cause	Characters having the specified internal code cannot be recognized.
E0202	Message	Unexpected EOF
	Cause	The file ended while the function was operating.
W0203	Message	Trigraph encountered
	Cause	A trigraph sequence (3-character representation) appeared.
	Action by User	If the -za option was specified, since trigraph sequences are valid, this warning is not output.

9.2.4 Error messages for configuration element

Table 9-4 Error Messages for Configuration Element <from 0301>

Error Number	Error Message	
E0301	Message	Syntax error
	Cause	A syntax error occurred.
	Action by User	Make sure there are no description errors in the source.
E0303	Message	Expected identifier
	Cause	An identifier is required for the goto statement.
	Action by User	Correctly describe the identifier to be specified for the goto statement.
W0304	Message	Identifier truncate to '*identifier*'
	Cause	The specified identifier is too long. The character number of the identifier (including '_') exceeds 250.
	Action by User	Shorten the length of the identifier.
E0305	Message	Compiler limit : too many identifiers with block scope
	Cause	There are too many symbols having block scope in 1 block.
E0306	Message	Illegal index, indirection not allowed
	Cause	An index is used in an expression that does not take a pointer value.
E0307	Message	Call of non-function '*variable name*'
	Cause	The variable name is used as a function name.
E0308	Message	Improper use of a typedef name
	Cause	The typedef name is improperly used.
W0309	Message	Unused '*variable name*'
	Cause	The specified variable is declared in the source, but is never used.
W0310	Message	'*Variable name*' is assigned a value which is never used
	Cause	The specified variable is used in an assignment statement, but is never used otherwise.
E0311	Message	Number syntax
	Cause	The constant expression is illegal.
E0312	Message	Illegal octal digit
	Cause	This is illegal as an octal digit.
E0313	Message	Illegal hexadecimal digit
	Cause	This is illegal as a hexadecimal digit.
E0314	Message	Too big constant
	Cause	The constant is too large and cannot be represented.
E0315	Message	Too small constant
	Cause	The constant is too small and cannot be represented.

Table 9-4 Error Messages for Configuration Element <from 0301>

Error Number	Error Message	
E0316	Message	Too many character constants
	Cause	The character constant exceeds 2 characters.
E0317	Message	Empty character constant
	Cause	The character constant ' ' is empty.
E0318	Message	No terminated string literal
	Cause	There is no double quote " " at the end of the string.
E0319	Message	Changing string literal
	Cause	A character string literal is rewritten.
W0320	Message	No null terminator in string literal
	Cause	The null character is not added to the character string literal.
E0321	Message	Compiler limit : too many characters in string literal
	Cause	The number of characters in the character string literal exceeded 509.
E0322	Message	Ellipsis requires three periods
	Cause	The C compiler detected "..", but "..." is required.
E0323	Message	Missing '*delimiter*'
	Cause	The delimiter is incorrect.
E0324	Message	Too many }'s
	Cause	The "{" and "}" are incorrectly paired.
E0325	Message	No terminated comment
	Cause	The comment is not terminated by "*/".
E0326	Message	Illegal binary digit
	Cause	This is illegal as a binary digit.
E0327	Message	Hex constants must have at least one hex digit
	Cause	At least 1 hexadecimal digit is required in a hexadecimal constant representation.
W0328	Message	Unrecognized character escape sequence '*character*'
	Cause	The escape sequence cannot be correctly recognized.
E0329	Message	Compiler limit : too many comment nesting
	Cause	The number of nesting levels of comments exceeded the limit of 255.
W0332	Message	Non-supported keyword found-ignored '*function attributes*' in this file
	Cause	A keyword not supported is detected. Function attributes in this file are ignored.
W0340	Message	Unreferenced label '*label name*'
	Cause	The specified label has been defined, but has not been referenced even once.

Table 9-4 Error Messages for Configuration Element <from 0301>

Error Number	Error Message	
E0342	Message	'*function qualifier*' keyword is not allowed
	Cause	This function qualifier cannot be used.

9.2.5 Error messages for conversion

Table 9-5 Error Messages for Conversion <from 0401>

Error Number	Error Message	
W0401	Message	Conversion may lose significant digits
	Cause	A long was converted into int. Be careful the value may be lost.
E0402	Message	Incompatible type conversion
	Cause	An illegal type conversion took place in the assignment statement.
E0403	Message	Illegal indirection
	Cause	The * operator is used in an integer type expression.
E0404	Message	Incompatible structure type conversion
	Cause	The types on both sides of an assignment statement to a structure or structure pair differ.
E0405	Message	Illegal lvalue
	Cause	This is an illegal left value.
E0406	Message	Cannot modify a const object '*variable name*'
	Cause	A variable with the const attribute is rewritten.
E0407	Message	Cannot write for read/only sfr '*SFR name*'
	Cause	Tried to write to a read-only sfr.
E0408	Message	Cannot read for write/only sfr '*SFR name*'
	Cause	Tried to read a write-only sfr.
E0409	Message	Illegal SFR access '*SFR name*'
	Cause	Illegal data was read from or written to an sfr.
W0410	Message	Illegal pointer conversion
	Cause	A pointer and an object other than a pointer are converted.
W0411	Message	Illegal pointer combination
	Cause	Different types are mixed in the same pointer combination.
W0412	Message	Illegal pointer combination in conditional expression
	Cause	Different types in a pointer combination are used in a conditional expression.
W0413	Message	Illegal structure pointer combination
	Cause	Pointers to structures with different types are mixed.
E0414	Message	Expected pointer
	Cause	A pointer is required.
W0415	Message	Conversion may lose significant digits for far pointer
	Cause	An attempt was made to convert a far pointer into a near pointer or int. Note that the values may be lost.

Table 9-5 Error Messages for Conversion <from 0401>

Error Number	Error Message	
W0416	Message	Illegal type and size (far/near) pointer combination
	Cause	Different types or different sizes (far or near pointer) are used together in the same pointer combination.
W0417	Message	Illegal type and size (far/near) pointer combination in conditional expression
	Cause	Different types or different sizes (far or near pointer) are used in a conditional expression between pointers.
W0418	Message	Illegal structure and size (far/near) pointer combination
	Cause	Pointers to structures with different types or different sizes (far or near pointer) are used together.

9.2.6 Error messages for an expression

Table 9-6 Error Messages for Expression <from 0501>

Error Number	Error Message	
E0501	Message	Expression syntax
	Cause	The expression contained a syntax error.
E0502	Message	Compiler limit : too many parentheses
	Cause	The nesting of parentheses in the expression exceeded 32.
W0503	Message	Possible use of '*variable name*' before definition
	Cause	The variable is used before a value is assigned to it.
W0504	Message	Possibly incorrect assignment
	Cause	The main operators in conditional expressions, such as if, while, and do statements, are assignment operators.
W0505	Message	Operator '*operator*' has no effect
	Cause	The operator has no effect in the program. This is probably due to a description error.
E0507	Message	Expected integral index
	Cause	Only an integer type expression is allowed in the index of an array.
E0508	Message	Too many actual arguments
	Cause	The number of arguments specified in a function call is more than the number of parameters specified in the list of argument types or the function definition.
E0509	Message	Too few actual arguments
	Cause	The number of arguments specified in a function call is fewer than the number of parameters specified in the list of argument types or the function definition.
W0510	Message	Pointer mismatch in function '*function name*'
	Cause	The given arguments have different pointer types than the arguments specified in the list of argument types or the function definition.
W0511	Message	Different argument types in function '*function name*'
	Cause	The argument types given in the function call do not match the list of argument types or the function definition.
E0513	Message	Illegal structure / union member '*member name*'
	Cause	A member that is referenced in the structure and not defined is indicated.
E0514	Message	Expected structure / union pointer
	Cause	The expression before the '->' operator is not a pointer to a structure or a union, but is the name of a structure or a union.
	Action by User	Make the expression before the '->' operator a pointer to a structure or a union.

Table 9-6 Error Messages for Expression <from 0501>

Error Number	Error Message	
E0515	Message	Expected structure/union name
	Cause	The expression before the "." operator is not the name of a structure or a union, but is a pointer to a structure or a union.
	Action by User	Make the expression before the "." operator a structure or a union variable.
E0516	Message	Zero sized structure '*structure name*'
	Cause	The size of the structure is zero.
E0517	Message	Illegal structure operation
	Cause	An operator that cannot be used in a structure is used.
E0518	Message	Illegal structure/union comparison
	Cause	2 structures or unions cannot be compared.
E0519	Message	Illegal bit field operation
	Cause	There is an illegal description for a bit field.
E0520	Message	Illegal use of pointer
	Cause	The only operators that can be used on pointers are addition, subtraction, assignment, relational, indirection (*), and member reference (->).
E0521	Message	Illegal use of floating
	Cause	An operator that cannot be used on floating-point variables is used.
W0522	Message	Ambiguous operators need parentheses
	Cause	2 shift, relational, and bit logical operators appear continuously without parentheses.
E0523	Message	Illegal bit, boolean type operation
	Cause	An illegal operation is performed on bit or boolean type variables.
E0524	Message	'&' on constant
	Cause	A constant address is not obtained.
E0525	Message	'&' requires lvalue
	Cause	The '&' operator can only be used in an expression assigned to the left value.
E0526	Message	'&' on register variable
	Cause	The address of a register variable is not obtained.
E0527	Message	'&' on bit, boolean ignored
	Cause	The address of a bit field, or bit or boolean type variable is not obtained.
W0528	Message	'&' is not allowed array/function, ignored
	Cause	The & operator does not have to be applied to an array name or function name.
E0529	Message	Sizeof returns zero
	Cause	The value of the sizeof expression becomes zero.

Table 9-6 Error Messages for Expression <from 0501>

Error Number	Error Message	
E0530	Message	Illegal sizeof operand
	Cause	The operand of the sizeof expression must be an identifier or a type name.
E0531	Message	Disallowed conversion
	Cause	Illegal casting occurred.
	Action by User	Check for illegal casting. This error occurs when a constant is cast to a pointer, or when an address is outside the range of the memory model.
E0532	Message	Pointer on left, needs integral right : '*operator*'
	Cause	Since the left operand is a pointer, the right operand must be an integral value.
E0533	Message	Invalid left-or-right operand : '*operator*'
	Cause	The left or right operand is illegal for the operator.
E0534	Message	Divide check
	Cause	The divisor of the / operation or % operation is zero.
E0535	Message	Invalid pointer addition
	Cause	2 pointers are not added.
E0536	Message	Must be integral value addition
	Cause	Only integral values can be added to a pointer.
E0537	Message	Illegal pointer subtraction
	Cause	The subtraction between pointers must be for pointers having the same type.
E0538	Message	Illegal conditional operator
	Cause	The conditional operator is not correctly described.
E0539	Message	Expected constant expression
	Cause	A constant expression is required.
W0540	Message	Constant out of range in comparison
	Cause	The constant partial expression is compared to a value outside of the range permitted by the type of the other partial expression.
E0541	Message	Function argument has void type
	Cause	The argument of the function has the void type.
E0546	Message	Too few actual argument for inline function '*function name*'
	Cause	The number of arguments specified in the function call of a function expanded inline is less than the number of parameters provided in the specifications.

9.2.7 Error messages for a statement

Table 9-7 Error Messages for Statement <from 0601>

Error Number	Error Message	
E0602	Message	Compiler limit : too many characters in logical source line
	Cause	The number of characters in a logical source line exceeded 2,048.
E0603	Message	Compiler limit : too many labels
	Cause	The number of labels exceeded 33.
E0604	Message	Case not in switch
	Cause	The case statement is not described in the correct position.
E0605	Message	Duplicate case '*label name*'
	Cause	The same case label is described two or more times in a switch statement.
E0606	Message	Non constant case expression
	Cause	Something other than an integral constant is specified in a case statement.
E0607	Message	Compiler limit : too many case labels
	Cause	The number of case labels in the switch statement exceeded 257.
E0608	Message	Default not in switch
	Cause	The default statement is not described in the correct position.
E0609	Message	More than one 'default'
	Cause	The default statement is described multiple times in the switch statement.
E0610	Message	Compiler limit : block nest level too depth
	Cause	The block nesting exceeded 45.
E0611	Message	Inappropriate 'else'
	Cause	There is no correspondence between if and else.
W0613	Message	Loop entered at top of switch
	Cause	A while, do, or for is specified immediately after the switch statement.
W0615	Message	Statement not reached
	Cause	The statement is never reached.
E0617	Message	Do statement must have 'while'
	Cause	A while is required at the end of a do.
E0620	Message	Break/continue error
	Cause	The positions of the break and continue statements are incorrect.
E0621	Message	Void function '*function name*' cannot return value
	Cause	A function declared as void returns a value.

Table 9-7 Error Messages for Statement <from 0601>

Error Number	Error Message	
W0622	Message	No return value
	Cause	A function that should return a value does not return a value.
	Action by User	If a value must be returned, add a return statement. If a value does not have to be returned, give the function the void type.
E0623	Message	No effective code and data, cannot create output file
	Cause	Since the code and data are not valid, the output file cannot be created.

9.2.8 Error messages for a declaration and function definition

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0701	Message	External definition syntax
	Cause	The function is not correctly defined.
E0702	Message	Too many callt functions
	Cause	There are too many declarations of the callt function. A maximum of 32 callt functions can be declared.
	Action by User	Decrease the number of callt function declarations.
E0703	Message	Function has illegal storage class
	Cause	The function is specified with an illegal storage class.
E0704	Message	Function returns illegal type
	Cause	The return value of the function is an illegal type.
E0706	Message	Parameter list error
	Cause	The function parameter list contains errors.
E0707	Message	Not parameter '*character string*'
	Cause	Something other than a parameter is declared in a function definition.
E0708	Message	Illegal parameters in rtos_task function
	Cause	The RTOS task function parameters are illegal.
	Action by User	Specify just one parameter. A parameter whose size exceeds 4 bytes cannot be specified.
E0710	Message	Illegal storage class
	Cause	The auto and register declarations are outside the function or the boolean variable is defined inside the function.
E0711	Message	Undeclared '*variable name*'; function '*function name*'
	Cause	An undeclared variable is used.
E0712	Message	Declaration syntax
	Cause	The declaration statement does not match the syntax.
E0713	Message	Redefined '*variable name*'
	Cause	Two or more of the same variables are defined.
	Action by User	Set the variable definition once.
W0714	Message	Too many register variables
	Cause	There are too many declarations of register variables.
	Action by User	Decrease the number of register variables. For the number that can be used, refer to CC78K0R C Compiler Language User's Manual.
E0715	Message	Too many sreg variables
	Cause	There are too many declarations of sreg variables.

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0718	Message	Too many bit, boolean type variables
	Cause	There are too many bit and boolean type variables.
	Action by User	Decrease the number of bit, boolean, and __boolean type variables. For the number that can be used, refer to CC78K0R C Compiler Language User's Manual.
E0719	Message	Illegal use of type
	Cause	An illegal type name is used.
E0720	Message	Illegal void type for '*identifier*'
	Cause	The identifier is declared by void.
W0721	Message	Illegal type for register declaration
	Cause	A register declaration is specified with an illegal type.
	Program Processing	The register declaration is ignored and processing continues.
E0722	Message	Illegal keyword for rtos_task function
	Cause	An illegal qualifier was specified for the RTOS task function.
E0724	Message	Structure redefinition
	Cause	The same structure is redefined.
W0725	Message	Illegal zero sized structure member
	Cause	The area taken as a structure member is not secured.
E0726	Message	Function cannot be structure/union member
	Cause	A function cannot be a member of a structure or a union.
E0727	Message	Unknown size structure/union '*name*'
	Cause	Structures or unions have undefined sizes.
E0728	Message	Compiler limit : too many structure/union members
	Cause	The members in a structure or union exceeded 256.
E0729	Message	Compiler limit : structure/union nesting
	Cause	The nesting of structures or unions exceeded 15.
E0730	Message	Bit field outside of structure
	Cause	A bit field is declared outside of the structure.
E0731	Message	Illegal bit field type
	Cause	A type other than an integral type is specified in a bit field type.
E0732	Message	Too long bit field size
	Cause	The number of bit specifications in a bit field declaration exceeds the number of bits in that type.

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0733	Message	Negative bit field size
	Cause	The number of bit specifications in a bit field declaration is negative.
E0734	Message	Illegal enumeration
	Cause	The enumeration type declaration does not match the syntax.
E0735	Message	Illegal enumeration constant
	Cause	The enumeration constant is illegal.
E0736	Message	Compiler limit : too many enumeration constants
	Cause	The number of enumeration constants exceeded 255.
E0737	Message	Undeclared structure / union / enum tag
	Cause	A tag is not declared.
E0738	Message	Compiler limit : too many pointer modifying
	Cause	The number of indirection operators (*) exceeded 12 in a pointer definition.
E0739	Message	Expected constant
	Cause	A variable is used in the index in an array declaration.
E0740	Message	Negative subscript
	Cause	The specification of the size of an array is negative.
E0741	Message	Unknown size array '*array name*'
	Cause	The size of an array is undefined.
	Action by User	Specify the size of the array.
E0742	Message	Compiler limit : too many array modifying
	Cause	The array declaration exceeds 12 dimensions.
E0743	Message	Array element type cannot be function
	Cause	An array of functions is not allowed.
W0744	Message	Zero sized array '*array name*'
	Cause	The number of elements of the defined array is zero.
W0745	Message	Expected function prototype
	Cause	The function prototype is not declared.
E0747	Message	Function prototype mismatch
	Cause	The function prototype declaration contains errors.
	Action by User	Check whether the parameter and return value types match the function.
W0748	Message	A function is declared as a parameter
	Cause	A function is declared as an argument.
W0749	Message	Unused parameter '*parameter name*'
	Cause	The parameter is not used.

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0750	Message	Initializer syntax
	Cause	The initialization does not match the syntax.
E0751	Message	Illegal initialization
	Cause	The constant of an initial value setting does not match the type of the variable.
W0752	Message	Undeclared initializer name '*name*'
	Cause	The initializer name is not declared.
E0753	Message	Cannot initialize static with automatic
	Cause	The static variable cannot be initialized using an automatic variable.
E0756	Message	Too many initializers '*array name*'
	Cause	There are more initial values than elements in the declared array.
E0757	Message	Too many structure initializers
	Cause	There are more initial values than members in the declared structure.
E0758	Message	Cannot initialize a function '*function name*'
	Cause	The function cannot be initialized.
E0759	Message	Compiler limit : initializers too deeply nested
	Cause	The depth of the nesting of initialized elements exceeded the limit.
W0760	Message	Double and long double are treated as IEEE 754 single format
	Cause	double and long double are handled as IEEE 754 single-precision formats.
W0761	Message	Cannot declare sreg with const or function
	Cause	sreg cannot be declared with a const declaration or function.
	Program Processing	An sreg declaration is ignored.
W0762	Message	Overlapped memory area '*variable name 1*' and '*variable name 2*'
	Cause	The variable name 1 and variable name 2 areas for which absolute address allocation specification is performed overlap.
W0763	Message	Cannot declare const with bit, boolean
	Cause	bit and boolean type variables cannot have const declarations.
	Program Processing	A const declaration is ignored.
W0764	Message	'*Variable name*' initialized and declared extern-ignored extern
	Cause	An externally referenced variable without a body was initialized.
	Program Processing	The extern declaration is ignored.
E0765	Message	Undefined static function '*function name*'
	Cause	There was a reference to a function whose body is not in the same file and was declared static.

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0769	Message	__far is not allowed for callt/interrupt function
	Cause	The __far qualifier must not be used for the callt and interrupt functions.
E0770	Message	Parameters are not allowed for interrupt function
	Cause	An interrupt function cannot have arguments.
E0771	Message	Interrupt function must be void type
	Cause	An interrupt function must have the void type.
E0772	Message	Callt are not allowed for interrupt function
	Cause	An interrupt function cannot be declared callt.
E0773	Message	Cannot call interrupt function
	Cause	An interrupt function cannot be called.
E0774	Message	Interrupt function can't use with the other kind interrupts
	Cause	An interrupt function cannot be used in other types of interrupts.
E0775	Message	Cannot call rtos_task function
	Cause	RTOS task cannot be called.
E0776	Message	Cannot call ret_int/_kernel_int_entry
	Cause	System call ret_int/_kernel_int_entry cannot be called from a function.
E0777	Message	Cannot allocate rtos_system_call
	Cause	The RTOS system call function must not be allocated.
E0778	Message	Cannot call ext_tsk in interrupt function
	Cause	System call ext_tsk cannot be called in the interrupt function/interrupt handler.
E0780	Message	Zero width for bit field '*member name*'
	Cause	Member name is specified to the member whose bit specification number of bit field declaration is 0.
W0787	Message	Bit field type is not int
	Cause	Type other than int is specified for bit field type.
E0788	Message	Cannot allocate a __flash function '*function name*'
	Cause	One of the __flash functions cannot be allocated.
E0789	Message	'-ZF' option did not specify - cannot allocate an EXT_FUNC function '*function name*'
	Cause	Flash memory area object creation option -zf is not specified. It cannot be allocated to the function specified in the #pragma EXT_FUNC.
E0790	Message	Callt/__interrupt are not allowed for EXT_FUNC function '*function name*'
	Cause	Callt/__interrupt declarations are not allowed for the function specified in the #pragma EXT_FUNC.

Table 9-8 Error Messages for Declaration and Function Definition <from 0701>

Error Number	Error Message	
E0791	Message	'-ZF' option specified - cannot allocate a callt function '*function name*'
	Cause	Flash memory area object creation option -zf was specified. A callt function cannot be allocated.
E0799	Message	Cannot allocate '*variable name*' out of '*address range*'
	Cause	Address specification for variable names for which absolute address allocation specification is performed exceed the specifiable address range.

9.2.9 Error messages for a preprocessing directive

Table 9-9 Error Messages for Preprocessing Directive <from 0801>

Error Number	Error Message	
E0801	Message	Undefined control
	Cause	A symbol starting with # cannot be recognized as a keyword.
E0802	Message	Illegal preprocess directive
	Cause	The preprocess directive is illegal.
	Action by User	Check if the preprocess directive (such as #pragma) is written in front of the header of the file and if there is any error.
E0803	Message	Unexpected non-whitespace before preprocess directive
	Cause	A character other than a whitespace character precedes the preprocess directive.
W0804	Message	Unexpected characters following '*preprocess directive*' directive - newline expected
	Cause	Extra characters follow the preprocess directive.
E0805	Message	Misplaced else or elif
	Cause	The #if, #ifdef, and #ifndef do not correspond to #else and #elif.
E0806	Message	Misplaced endif
	Cause	The #if, #ifdef, and #ifndef do not correspond to #endif.
E0807	Message	Compiler limit : too many conditional inclusion nesting
	Cause	The nesting of conditional compiling exceeded 255.
E0810	Message	Cannot find include file '*file name*'
	Cause	The include file was not found.
	Action by User	Specify the path in which an include file exists or specify a path using the -i option for the environmental variable INC78K0R.
E0811	Message	Too long file name '*file name*'
	Cause	The file name is too long.
E0812	Message	Include directive syntax
	Cause	The file name in the definition of the #include statement is not correctly enclosed by " " or < >.
E0813	Message	Compiler limit : too many include nesting
	Cause	The nesting of the include files exceeded 50.
E0814	Message	Illegal macro name
	Cause	The macro name is illegal.
E0815	Message	Compiler limit: too many macro nesting
	Cause	The number of nesting macros exceeds 200.

Table 9-9 Error Messages for Preprocessing Directive <from 0801>

Error Number	Error Message	
W0816	Message	Redefined macro name '*macro name*'
	Cause	The macro name is redefined.
W0817	Message	Redefined system macro name '*macro name*'
	Cause	The system macro name is redefined.
E0818	Message	Redeclared parameter in macro '*macro name*'
	Cause	The same identifier appears in the parameter list in the macro definition.
W0819	Message	Mismatch number of parameter '*macro name*'
	Cause	The number of parameters when referencing differs from the number of parameters defined by #define.
E0821	Message	Illegal macro parameter '*macro name*'
	Cause	The description enclosed by parentheses ( ) in the function format macro is illegal.
E0822	Message	Missing) '*macro name*'
	Cause	The right parenthesis ")" was not found in the same line as the #define definition in the function format macro.
E0823	Message	Too long macro expansion '*macro name*'
	Cause	The actual argument during macro expansion is too long.
W0824	Message	Identifier truncate to '*macro name*'
	Cause	The macro name is too long.
	Program Processing	It is shortened to the displayed '*macro name*'.
W0825	Message	Macro recursion '*macro name*'
	Cause	The #define definition becomes recursive.
E0826	Message	Compiler limit : too many macro defines
	Cause	The number of macro definitions exceeded 10,000.
E0827	Message	Compiler limit : too many macro parameters
	Cause	1 macro definition had over 31 calling parameters.
E0828	Message	Not allowed #undef for system macro name
	Cause	The system macro name is specified by #undef.
W0829	Message	Unrecognized pragma '*character string*'
	Cause	This character string is not supported.
	Action by User	Check that the keywords are correct. This warning occurs if an incorrect segment was specified in the #pragma section.
E0830	Message	No chip specifier : #pragma pc ( )
	Cause	There is no device specifier.

Table 9-9 Error Messages for Preprocessing Directive <from 0801>

Error Number	Error Message	
E0831	Message	Illegal chip specifier : #pragma pc (*device type*)
	Cause	The device specifier is illegal.
W0832	Message	Duplicated chip specifier
	Cause	The device specifier is duplicated.
E0833	Message	Expected #asm
	Cause	There is no #asm.
E0834	Message	Expected #endasm
	Cause	There is no #endasm.
W0835	Message	Too many characters in assembler source line
	Cause	A line in the assembler source is too long.
W0836	Message	Expected assembler source
	Cause	There is no assembler source between #asm and #endasm.
W0837	Message	Output assembler source file, not object file
	Cause	There is a #asm block or __asm statement. Assembler source file is output instead of the object file.
	Action by User	Specify the -a or -sa compiler option in order to output the #asm and __asm statement description to the object file, and then assemble the output assembler file.
E0838	Message	Duplicated pragma VECT or INTERRUPT or RTOS_INTERRUPT '*character string*'
	Cause	The #pragma VECT '*character string*', INTERRUPT '*character string*', or RTOS_INTERRUPT ’*character string*’ is duplicated.
E0839	Message	Unrecognized pragma VECT or INTERRUPT or RTOS_INTERRUPT '*character string*'
	Cause	There is an unrecognized #pragma VECT '*character string*', INTERRUPT '*character string*', or RTOS_INTERRUPT ’*character string*’.
W0840	Message	Undefined interrupt function '*function name*'- ignored BANK or SP_SWITCH or LEAFWORK specified
	Cause	The save destination is specified for an undefined interrupt function.
	Program Processing	Register bank specifications, stack switching specifications specifications are ignored.
E0842	Message	Unrecognized pragma SECTION '*character string*'
	Cause	There is an unrecognized #pragma SECTION '*character string*'.
E0843	Message	Unspecified start address of '*section name*'
	Cause	The correct starting address is not specified after AT in the #pragma section.
E0845	Message	Cannot allocate '*section name*' out of '*address range*'
	Cause	The specified section cannot be placed at the specified starting address.

Table 9-9 Error Messages for Preprocessing Directive <from 0801>

Error Number	Error Message	
W0846	Message	Rechanged section name '*section name*'
	Cause	The same section name is duplicated and its specification is changed.
	Program Processing	The section name specified last is valid and processing continues.
E0847	Message	Different BANK or SP_SWITCH specified on same interrupt function '*function name*'
	Cause	A different register bank specification or stack switching specification is specified for an interrupt function with the same name.
W0849	Message	#pragma statement is not portable
	Cause	The #pragma statement does not conform to ANSI.
W0850	Message	Asm statement is not portable
	Cause	The ASM statement does not conform to ANSI.
W0851	Message	Data aligned in '*area name*'
	Cause	The segment area or structure tag is data aligned. The area name is a segment name or a structure tag.
W0852	Message	Module name truncate to '*module name*'
	Cause	The specified module name is too long.
	Program Processing	It is shortened to the displayed '*module name*'.
E0853	Message	Unrecognized pragma NAME '*module name*'
	Cause	Unrecognizable characters are in the '*module name*'.
E0854	Message	Undefined rtos_task '*character string*'
	Cause	The body of RTOS task is not defined.
E0855	Message	Cannot assign rtos_interrupt_handler to non-maskable and software interrupt
	Cause	The non-maskable interrupt and software interrupt cannot be specified in the RTOS_INTERRUPT handler.
W0856	Message	Rechanged module name '*module name*'
	Cause	Duplicate module names are specified.
W0857	Message	Section name truncate to '*section name*'
	Cause	The specified section name is too long.
	Program Processing	It is shortened to the displayed '*section name*'. Make the section name 8 or fewer characters.
E0858	Message	Unrecognized pragma 'pragma *character string*' '*illegal character string*'
	Cause	There is an unrecognized #pragma 'pragma *character string*', '*illegal character string*'.
E0859	Message	Cannot allocate EXT_TABLE out of 0xc0-0xff80
	Cause	The start address of the flash area branch table must be within 0xc0 to 0xff80.

Table 9-9 Error Messages for Preprocessing Directive <from 0801>

Error Number	Error Message	
E0860	Message	Redefined #pragma EXT_TABLE
	Cause	The #pragma EXT_TABLE is redefined.
E0861	Message	No EXIT_TABLE specifier
	Cause	Flash area branch table start address is not specified.
	Program Processing	Specify the -zf option only when the self-rewriting function is used in flash memory products with a self-rewriting function.
E0862	Message	Illegal EXT_FUNC id specifier : out of 0x0-0xff
	Cause	The ID value of the function in the flash memory area that are specified by #pragma EXT_FUNC must be 0x80-0xff80.
E0863	Message	Redefined #pragma EXT_FUNC name '*function name*'
	Cause	The function name specified by the #pragma EXT_FUNC is redefined.
E0864	Message	Redefined #pragma EXT_FUNC id '*ID value*'
	Cause	The ID value specified by the #pragma EXT_FUNC is redefined.
E0865	Message	Out of range - cannot allocate an EXT_FUNC function '*function name*'
	Cause	Address of the flash memory area branch table exceeds the specifiable address range. A function specified by the #pragma EXT_FUNC cannot be allocated.
E0866	Message	#pragma section found after C body. cannot include file containing #pragma section and without C body at the line
	Cause	There is #pragma section syntax after C body description. Subsequent files that contain #pragma section syntax and no C body (including external reference declarations of variables and functions) cannot be included.
E0867	Message	#pragma section found after C body. cannot specify #include after #pragma section in this file
	Cause	There is #pragma section syntax after C body description. Hereafter, #include syntax cannot be described.
E0868	Message	#include found after C body. cannot specify #pragma section after #include directive
	Cause	There is #include syntax after C body description. Hereafter, #pragma section syntax cannot be described.
W0869	Message	'*Section name*' section cannot change after C body
	Cause	Specified section cannot be changed after C body description.
W0870	Message	Data aligned before '*variable name*' in '*section name*'
	Cause	Data alignment is done before '*variable name*' is allocated in '*section name*'.
W0871	Message	Data aligned after '*variable name*' in '*section name*'
	Cause	Data alignment is done after '*variable name*' is allocated in '*section name*'.
E0899	Message	Character string specified by #error is output
	Cause	An #error character string was specified.

9.2.10 Error messages for fatal file I/O and running on an illegal operating system

Table 9-10 Error Messages for Fatal File I/O and Running on an Illegal Operating System <from 0901>

Error Number	Error Message	
F0901	Message	File I/O error
	Cause	A physical I/O error was generated during file input/output.
	Action by User	If an intermediate file is the cause, increase the conventional memory, or use EMS or XMS memory.
F0902	Message	Cannot open '*file name*'
	Cause	The file cannot be opened.
	Action by User	Check if a device file is installed in an ordinary search path. The path can be specified by the -y option. Refer to the description about the search path in "5.4 Device file search path".
F0903	Message	Cannot open overlay file '*file name*'
	Cause	The overlay file cannot be opened.
F0904	Message	Cannot open temp
	Cause	The input temporary file cannot be opened.
F0905	Message	Cannot create '*file name*'
	Cause	A file create error was generated.
F0906	Message	Cannot create temp
	Cause	A create error was generated in an output temporary file.
	Action by User	Check if the environmental variable TMP is specified.
F0907	Message	No available data block
	Cause	A temporary file cannot be created because the drive file does not have sufficient capacity.
F0908	Message	No available directory space
	Cause	A temporary file cannot be created because of insufficient folder area on the drive.
F0909	Message	R/O : read/only disk
	Cause	A temporary file cannot be created because the drive is read only.
F0910	Message	R/O file : read/only, file opened read/only mode
	Cause	A write error was generated by a temporary file for the following reasons. - A file with the same name as a temporary file already exists on the drive and it has the read-only attribute. - The output temporary file is opened with the read-only attribute because of internal conflicts.

Table 9-10 Error Messages for Fatal File I/O and Running on an Illegal Operating System <from 0901>

Error Number	Error Message	
F0911	Message	Reading unwritten data, no available directory space
	Cause	An I/O error was generated for the following reasons. - EOF was passed and input proceeded. - The temporary file cannot be created because of insufficient folder area on the drive.
F0912	Message	Write error on temp
	Cause	A write error was generated to the output temporary file.
	Action by User	A complex source expression (such as too deep nesting) may be the cause. Contact support.
F0914	Message	Insufficient memory in hostmachine
	Cause	The CC78K0R cannot start because of insufficient memory.
	Action by User	Increase the free area in the conventional memory.
W0915	Message	Asm statement found. skip to jump optimize this function '*function name*'
	Cause	#asm block or __asm statement was detected. This function does not have jump optimization. Perform the W0837 response.
E0922	Message	Heap overflow : please retry compile without -QJ
	Cause	A memory overflow was generated in jump optimization. Recompile without specifying -qj.
F0923	Message	Illegal device file format
	Cause	A device file in an old format was referenced.
F0924	Message	Out of range. please retry compile without -QT
	Cause	The code size that optimization can be performed exceeds 32 KB.
	Action by User	Recompile without specifying -qt.
F0925	Message	Out of range. please retry compile without -QL4
	Cause	The branch distance of the branch table in a switch statement exceeds 64 KB.
	Action by User	Separate the file so that the code size does not exceed 32KB, or recompile without specifying -ql4.

9.3 List of PM+ Error Messages

Table 9-11 PM+ Error Messages

Error Type	Error Message	
!	Message	Out of range. The range of columns is from 72 to 132.
	Cause	A value out of the specifiable range is described for the number of characters per line.
	Action by User	Specify a value in the specifiable range and retry the execution.
	Button	[OK] ... Close the message box.
!	Message	Out of range. The range of lines is from 0, and 20 to 32,767.
	Cause	A value out of the specifiable range is described for the number of lines per page.
	Action by User	Specify a value in the specifiable range and retry the execution.
	Button	[OK] ... Close the message box.
!	Message	Out of range. The range of TAB character is from 0 to 8.
	Cause	A value out of the specifiable range is described for the tab stop position.
	Action by User	Specify a value in the specifiable range and retry the execution.
	Button	[OK] ... Close the message box.
!	Message	Out of range. The range of warning level is from 0 to 2.
	Cause	A value out of the specifiable range is described for the warning level.
	Action by User	Specify a value in the specifiable range and retry the execution.
	Button	[OK] ... Close the message box.
!	Message	Cannot find folder. Willyou create?
	Cause	A non-existing folder is described.
	Action by User	Click the [OK] button to create a new folder. Click the [Cancel] button to cancel the folder creation.
	Button	[OK] ... Creates a folder and closes the message box. [Cancel] ... Closes the message without creating a new folder.
!	Message	Cannot find drive. Make sure the drive.
	Cause	The specified drive is not found.
	Action by User	Specify the correct drive name.
	Button	[Retry] ... Retries accessing the drive. [Cancel] ... Accessing is ignored.

Table 9-11 PM+ Error Messages

Error Type	Error Message	
!	Message	Illegal File name.
	Cause	The file name includes characters whose use are not allowed by the OS or the CC78K0R.
	Action by User	Do not use characters that cannot be handled by the OS, or "#" or "," which cannot be handled by the CC78K0R.
	Button	[OK] ... Close the message box.
!	Message	Unable to create the falder.
	Cause	Folder creation was rejected by the OS because of a shortage of available disk space, read-only, etc.
	Action by User	Check the available disk space and whether write is permitted.
	Button	[OK] ... Close the message box.
!	Message	Ignored Options.
	Cause	Option information in the project file includes a combination of options that causes warning or an error in the CC78K0R.
	Action by User	Check whether the option specification contradicts.
	Button	[OK] ... Close the message box.
!	Message	Can't read Option Information.
	Cause	Valid option information is not included in the file specified by option information read specification.
	Action by User	Check whether the specified option information file is of the 78K0R, or whether the specified option information file is not destroyed.
	Button	[OK] ... Close the message box.
!	Message	Cannot find path or file. Make sure the path or filename.
	Cause	A non-existing path or file was specified, where a path or file name that actually exists, such as an include file, must be specified.
	Action by User	Check the target file name and the path for it, and specify the correct name or path.
	Button	[OK] ... Close the message box.
X	Message	Cannot find %s shown in environment variable PATH.
	Cause	cc78k0r.exe is not found.
	Action by User	Check whether the CC78K0R was installed normally, or whether the PATH environment variable was set correctly.
	Button	[OK] ... Close the message box.

Table 9-11 PM+ Error Messages

Error Type	Error Message	
!	Message	Multiple Include Search Path definition.
	Cause	The same include file path was specified twice.
	Action by User	Do not specify the same include file path twice.
	Button	[OK] ... Close the message box.
!	Message	Too many characters for Include Search Path.
	Cause	The include file path whose length exceeds the specifiable range was specified.
	Action by User	Specify the correct path name.
	Button	[OK] ... Close the message box.
!	Message	Too many Include Search Path. Up to 64 can be specified for Include Search Path.
	Cause	The number of specified include file paths exceeds the specifiable number.
	Action by User	Keep the number of specified paths 64 or fewer.
	Button	[OK] ... Close the message box.
!	Message	Multiple define definition.
	Cause	The same defined macro was specified twice.
	Action by User	Do not specify the same defined macro twice.
	Button	[OK] ... Close the message box.
!	Message	Too many characters for macro Definition. Up to 256 characters can be described for a macro name.
	Cause	The length of characters used for specifying the defined macro name exceeds the specifiable range.
	Action by User	Keep the number of characters used for macro name specification 256 or fewer.
	Button	[OK] ... Close the message box.
!	Message	Too many macro for macro Definition. macro Definition and macro Undefinition can be specified to 30 in all.
	Cause	The number of defined and undefined macros that can be specified exceeds the specifiable range.
	Action by User	Keep the number of defined and undefined macros 30 or fewer in total.
	Button	[OK] ... Close the message box.
!	Message	Multiple undefine definition.
	Cause	The same undefined macro was specified twice.
	Action by User	Do not specify the same undefined macro twice.
	Button	[OK] ... Close the message box.

Table 9-11 PM+ Error Messages

Error Type	Error Message	
!	Message	Too many characters for undefine Definition. Up to 256 characters can be described for a macro name.
	Cause	The length of characters used for specifying the undefined macro name exceeds the specifiable range.
	Action by User	Keep the number of characters used for macro name specification 256 or fewer.
	Button	[OK] ... Close the message box.
!	Message	Too many macro for macro Undefinition. macro Definition and macro Undefinition can be specified to 30 in all.
	Cause	The number of defined and undefined macros that can be specified exceeds the specifiable range.
	Action by User	Keep the number of defined and undefined macros 30 or fewer in total.
	Button	[OK] ... Close the message box.
!	Message	Can't set options correctly to (source name)
	Cause	When an attempt was made to reflect a common option in a special option, a contradiction was found in specification, or the specification exceeds the quantitative limits.
	Action by User	Option specification that cannot be reflected is ignored. Check the special option settings, as necessary.
	Button	[OK] ... Close the message box.
!	Message	Only fin of an extension is effective.
	Cause	An extension other than "fin" was specified for the function information file name.
	Action by user	Specify "fin" as an extension for the function information file name.
	Button	[OK] ... Close the message box.
!	Message	There is options that specification different from whole options cannot be done. The options was set automatically.
	Cause	There are options for which the same specification must be made by a common option and special option.
	Action by user	Specification of the common option is automatically reflected in that of the special option. (Options to be reflected) -ms, -mm, -mc, -ml, -zf, -rd
	Button	[OK] ... Close the message box.

APPENDIX A   SAMPLE PROGRAMS

This chapter introduces sample programs for the CC78K0R.

A.1   C Source Module File

```
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark [ SIZE + 1 ] ;

void main ( void ) {
        int     i , prime , k , count ;

        count = 0 ;

        for ( i = 0 ; i <= SIZE ; i++ )
                mark [ i ] = TRUE ;
        for ( i = 0 ; i <= SIZE ; i++ ) {
                if ( mark [ i ] ) {
                        prime = i + i + 3 ;
                        printf ( "%6d" , prime ) ;
                        count++ ;
                        if ( ( count%8 ) == 0 ) putchar ( '\n' ) ;
                        for ( k = i + prime ; k <= SIZE ; k += prime )
                                mark [ k ] = FALSE ;
                }
        }
        printf ( "\n%d primes found." , count ) ;
}

void printf ( char *s , int i ) {
        int     j ;
        char    *ss ;

        j = i ;
        ss = s ;
}

void putchar ( char c ) {
        char    d ;
        d = c ;
}
```

A.2 Execution Example

```
C>cc78K0R -cf1166a0 prime.c -a –p -x -e -ng
```

```
78K0R C Compiler Vx.xx [xx xxx xxxx]
Copyright(C) NEC Electronics Corporation xxxx, xxxx

prime.c ( 18 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 20 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 26 ) : CC78K0R warning W0622 : No return value
prime.c ( 35 ) : CC78K0R warning W0622 : No return value
prime.c ( 41 ) : CC78K0R warning W0622 : No return value

Target chip : uPD78F1166_A0
Device file : Vx.xx

Compilation complete, 0 error(s) and 5 warning(s) found.
```

A.3 Output List

A.3.1 Assembler source module file

```
; 78K0R C Compiler Vx.xx Assembler Source   Date:xx xxx xxxx    Time:xx:xx:xx
; Command       : -cf1166a0 prime.c -a -p -x -e -ng
; In-file       : prime.c
; Asm-file      : prime.asm
; Para-file     :

$PROCESSOR ( f1166a0 )
$NODEBUG
$NODEBUGA
$KANJICODE EUC
$TOL_INF        03FH , 100H , 02H , 00H , 00H

        EXTRN   _@RTARG0
        EXTRN   @@isrem
        PUBLIC  _mark
        PUBLIC  _main
        PUBLIC  _printf
        PUBLIC  _putchar

@@CNST  CSEG    MIRRORP
L0011 : DB      '%6d'
        DB      00H
L0017 : DB      ' '
        DB      0AH
        DB      '%d primes found.'
        DB      00H

@@DATA  DSEG    BASEP
_mark : DS      ( 201 )
        DS      ( 1 )

; line  5
; line  8

@@CODE  CSEG    BASE
_main :
        push    hl                                  ; [ INF ] 1 , 1
        subw    sp , #08H                           ; [ INF ] 2 , 1
        movw    hl , sp                             ; [ INF ] 3 , 1
; line  11
        clrw    ax                                  ; [ INF ] 1 , 1
        movw    [ hl ] , a            ; count       ; [ INF ] 1 , 1
; line  13
        movw    [ hl + 6 ] , ax       ; i           ; [ INF ] 2 , 1
L0003 :
        movw    ax , [ hl + 6 ]       ; i           ; [ INF ] 2 , 1
        cmpw    ax , #0C8H            ; 200         ; [ INF ] 3 , 1
        orl     CY , a.7                            ; [ INF ] 2 , 1
        skc                                         ; [ INF ] 2 , 1
        bnz     $L0004                              ; [ INF ] 2 , 4
; line  14
        movw    ax , [ hl + 6 ]       ; i           ; [ INF ] 2 , 1
        movw    bc , ax                             ; [ INF ] 1 , 1
        mov     _mark [ bc ] , #01H   ; 1           ; [ INF ] 4 , 1
        movw    ax , [ hl + 6 ]       ; i           ; [ INF ] 2 , 1
        incw    ax                                      ; [ INF ] 1 , 1
        movw    [ hl + 6 ] , ax         ; i             ; [ INF ] 2 , 1
        br      $L0003                                  ; [ INF ] 2 , 4
L0004 :
; line  15
        clrw    ax                                      ; [ INF ] 1 , 1
        movw    [ hl + 6 ] , ax         ; i             ; [ INF ] 2 , 1
L0006 :
        movw    ax , [ hl + 6 ]         ; i             ; [ INF ] 2 , 1
        cmpw    ax , #0C8H              ; 200           ; [ INF ] 3 , 1
        or1     CY , a.7                                ; [ INF ] 2 , 1
        skc                                             ; [ INF ] 2 , 1
        bnz     !L0007                                  ; [ INF ] 2 , 4
; line  16
        movw    ax , [ hl + 6 ]         ; i             ; [ INF ] 2 , 1
        addw    ax , #loww ( _mark )                    ; [ INF ] 3 , 1
        movw    de , ax                                 ; [ INF ] 1 , 1
        mov     a , [ de ]                              ; [ INF ] 2 , 2
        cmp0    a                                       ; [ INF ] 1 , 1
        bz      $L0015                                  ; [ INF ] 2 , 4
; line  17
        movw    ax , [ hl + 6 ]         ; i             ; [ INF ] 2 , 1
        addw    ax , ax                                 ; [ INF ] 1 , 1
        addw    ax , #03H               ; 3             ; [ INF ] 3 , 1
        movw    [ hl + 4 ] , ax         ; prime         ; [ INF ] 2 , 1
; line  18
        push    ax                                      ; [ INF ] 1 , 1
        movw    ax , #loww ( L0011 )    ; 0             ; [ INF ] 3 , 1
        call    !!_printf                               ; [ INF ] 4 , 3
        pop     ax                                      ; [ INF ] 1 , 1
; line  19
        movw    ax , [ hl ]             ; count         ; [ INF ] 1 , 1
        incw    ax                                      ; [ INF ] 1 , 1
        movw    [ hl ] , ax             ; count         ; [ INF ] 1 , 1
; line  20
        movw    _@RTARG0 , ax                           ; [ INF ] 2 , 1
        movw    ax , #08H               ; 8             ; [ INF ] 3 , 1
        call    !!@@isrem                               ; [ INF ] 3 , 3
        or      a , x                                   ; [ INF ] 2 , 1
        bnz     $L0012                                  ; [ INF ] 2 , 4
        movw    ax , #0AH               ; 10            ; [ INF ] 3 , 1
        call    !!_putchar                              ; [ INF ] 4 , 3
L0012 :
; line  21
        movw    ax ,  [ hl + 6 ]        ; i             ; [ INF ] 2 , 1
        xchw    ax , bc                                 ; [ INF ] 1 , 1
        movw    ax , [ hl + 4 ]         ; prime         ; [ INF ] 2 , 1
        addw    ax , bc                                 ; [ INF ] 1 , 1
        movw    [ hl + 2 ] , ax         ; k             ; [ INF ] 2 , 1
L0014 :
        movw    ax , [ hl + 2 ]         ; k             ; [ INF ] 2 , 1
        cmpw    ax , #0C8H              ; 200           ; [ INF ] 3 , 1
        or1     CY , a.7                                ; [ INF ] 2 , 1
        skc                                             ; [ INF ] 2 , 1
        bnz     $L0015                                  ; [ INF ] 2 , 4
; line  22
        movw    ax , [ hl + 2 ]         ; k             ; [ INF ] 2 , 1
        movw    bc , ax                                 ; [ INF ] 1 , 1
        mov     _mark [ bc ] , #00H     ; 0             ; [ INF ] 4 , 1
        movw    ax , [ hl + 2 ]         ; k             ; [ INF ] 2 , 1
        xchw    ax , bc                                 ; [ INF ] 1 , 1
        movw    ax , [ hl + 4 ]         ; prime         ; [ INF ] 2 , 1
        addw    ax , bc                                 ; [ INF ] 1 , 1
        movw    [ hl + 2 ] , ax         ; k             ; [ INF ] 2 , 1
        br      $L0014                                  ; [ INF ] 2 , 4
L0015 :
; line  24
        movw    ax , [ hl + 6 ]         ; i             ; [ INF ] 2 , 1
        incw    ax                                      ; [ INF ] 1 , 1
        movw    [ hl + 6 ] , ax         ; i             ; [ INF ] 2 , 1
        br      $L0006                                  ; [ INF ] 2 , 4
L0007 :
; line  25
        movw    ax , [ hl ]             ; count         ; [ INF ] 1 , 1
        push    ax                                      ; [ INF ] 1 , 1
        movw    ax , #loww ( L0017 )    ; 0             ; [ INF ] 3 , 1
        call    !!_printf                               ; [ INF ] 4 , 3
        pop     ax                                      ; [ INF ] 1 , 1
; line  26
        addw    sp , #08H                               ; [ INF ] 2 , 1
        pop     hl                                      ; [ INF ] 1 , 1
        ret                                             ; [ INF ] 1 , 6
; line  29
_printf :
        push    hl                                      ; [ INF ] 1 , 1
        push    ax                                      ; [ INF ] 1 , 1
        subw    sp , #04H                               ; [ INF ] 2 , 1

        movw    hl , sp                                 ; [ INF ] 3 , 1
; line  33
        movw    ax , [ hl + 12 ]        ; i             ; [ INF ] 2 , 1
        movw    [ hl + 2 ] , ax         ; j             ; [ INF ] 2 , 1
; line  34
        movw    ax , [ hl + 4 ]         ; s             ; [ INF ] 2 , 1
        movw    [ hl ] , ax             ; ss            ; [ INF ] 1 , 1
; line  35
        addw    sp , #04H                               ; [ INF ] 2 , 1
        pop     hl                                      ; [ INF ] 1 , 1
        ret                                             ; [ INF ] 1 , 6
; line  38
_putchar :
        push    hl                                      ; [ INF ] 1 , 1
        movw    hl , ax                                 ; [ INF ] 1 , 1
; line  40
        mov     a , l                                   ; [ INF ] 1 , 1
        mov     h , a                                   ; [ INF ] 1 , 1
; line  41
        pop     hl                                      ; [ INF ] 1 , 1
        ret                                             ; [ INF ] 1 , 6
        END


; *** Code Information ***
;
; $FILE /auto/proj/cmp/cc.new/work/egashira/cc78sk0r/src/test/prime2.c
;
; $FUNC main ( 8 )
;       bc = ( void )
;       CODE SIZE = 155 bytes , CLOCK_SIZE = 117 clocks , STACK_SIZE = 16 bytes
;
; $CALL printf ( 18 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
```

```
; $CALL putchar ( 20 )
;       bc = ( int : ax )
;
; $CALL printf ( 25 )
;       bc = ( pointer : ax , int : [ sp + 2 ] )
;
; $FUNC printf ( 29 )
;       bc = ( pointer s : ax , int i : [ sp + 4 ] )
;       CODE SIZE = 18 bytes , CLOCK_SIZE = 16 clocks , STACK_SIZE = 8 bytes
;
; $FUNC putchar ( 38 )
;       bc = ( char c : x )
;       CODE SIZE = 6 bytes , CLOCK_SIZE = 11 clocks , STACK_SIZE = 2 bytes

; Target chip : uPD78F1166_A0
; Device file : Vx.xx
```

A.3.2 Preprocess list file

```
/*
78K0R C Compiler Vx.xx Preprocess List          Date:xx xxx xxxx    Page:1
Command         : -cf1166a0 prime.c -a -p -x -e -ng
In-file         : prime.c
PPL-file        : prime.ppl
Para-file       :
*/

        1 :   #define TRUE    1
        2 :   #define FALSE   0
        3 :   #define SIZE    200
        4 :
        5 :   __far char  mark [ SIZE + 1 ] ;
        6 :
        7 :   void main ( void )
        8 :   {
        9 :       int i , prime , k , count ;
        10 :
        11 :      count = 0 ;
        12 :
        13 :      for ( i = 0 ; i <= SIZE ; i++ )
        14 :          mark [ i ] = TRUE ;
        15 :      for ( i  = 0 ; i <= SIZE ; i++ ) {
        16 :          if ( mark [ i ] ) {
        17 :              prime = i + i + 3 ;
        18 :              printf ( "%6d", prime ) ;
        19 :              count++ ;
        20 :              if ( ( count%8 ) == 0 ) putchar ( '\n' ) ;
        21 :                  for ( k = i + prime ; k <= SIZE ; k += prime )
        22 :                      mark [ k ] = FALSE ;
        23 :              }
        24 :      }
        25 :      printf ( "\n%d primes found." , count ) ;
        26 :  }
        27 :
        28 :  printf ( char *s , int i )
        29 :  {
        30 :      int     j ;
        31 :      char    *ss ;
        32 :
        33 :      j = i ;
        34 :      ss = s ;
        35 :  }
        36 :
        37 :  putchar ( char c )
        38 :  {
        39 :      char    d ;
        40 :      d = c ;
        41 :  }


/*
Target chip : uPD78F1166_A0
Device file : Vx.xx
*/
```

A.3.3 Cross-reference list file

```
78K0R C Compiler Vx.xx Cross reference List     Date:xx xxx xxxx    Page:1

Command     : -cf1166a0 prime.c -a -p -x -e –ng
In-file     : prime.c
Xref-file   : prime.xrf
Para-file   :

ATTRIB  MODIFY  TYPE    SYMBOL  DEFINE  REFERENCE

EXTERN  FAR     array   mark      5      14  16  22
EXTERN  FAR     func    main      7
AUTO1           int     i         9      13  13  13  14  15  15  15  16  17  17
                                         21
AUTO1           int     prime     9      17  18  21  21
AUTO1           int     k         9      21  21  21  22
AUTO1           int     count     9      11  19  20  25
EXTERN  FAR     func    printf  28       18  25
EXTERN  FAR     func    putchar 37       20
PARAM           pointer s       28       34
PARAM           int     i       28       33
AUTO1           int     j       30       33
AUTO1           pointer ss      31       34
REG1            char    c       37       40
PARAM
REG1            char    d       39       40
                #define TRUE      1      14
                #define FALSE     2      22
                #define SIZE      3       5  13  15  21


 Target chip : uPD78F1166_A0
 Device file : Vx.xx
```

A.3.4 Error list file

```
prime.c ( 18 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 20 ) : CC78K0R warning W0745 : Expected function prototype
prime.c ( 26 ) : CC78K0R warning W0622 : No return value
prime.c ( 35 ) : CC78K0R warning W0622 : No return value
prime.c ( 41 ) : CC78K0R warning W0622 : No return value


Target chip : uPD78F1166_A0
Device file : Vx.xx
Compilation complete, 0 error(s) and 5 warning(s) found.
```

APPENDIX B LIST OF USE-RELATED CAUTIONS

This chapter indicates cautions related to the use of the CC78K0R.

Table B-1 List of Use-related Cautions

Number	Cautions
1	[Specification of options] - When several specifications have been made for an option that does not allow multiple specifications, the last specification takes priority (is valid). - The type specification following the -c option must not be omitted. If it is omitted, an abort error will occur. If the -c option is not specified, be sure to enter #pragma pc (type) in the C source module file instead. During compilation, if the specified option is different from the option in the C source, the specified option takes priority. A warning message is output at that time. - If the help option has been specified, all other options are ignored.
2	[File output destinations] Only disk-type files can be specified as the output destination for object module files.
3	[Error messages] When a syntax error has been found in a file, an error message is attached to the file name. If a device file has been specified at a prohibited location, the specified character string is output by itself. In all other cases, the drive, path, and file extension must be attached.
4	[Source file names] In the CC78K0R, the part except the source file name extension (primary name) is used as the module name by default. Therefore, some restrictions apply to the source file names that can be used. - Regarding the length of the file name, configure the file name with a primary name and extension within the range allowed by the OS, and separate the primary name and the extension with a dot (.). - The characters that can be used for the primary name and the extension consist of the characters allowed by the OS, except parentheses (( )), semicolons (;), and commas (,). Note that a hyphen (-) cannot be used as the first character of a file name or file name. Do not specify file names that include a space or 2-byte characters. - An error is output during linking for files that have the same name as the first 8 characters of the primary name. - Sharp symbol (#) cannot be used for file names and path names in parameter files.
5	[Include files] It is not possible to define functions (excluding declarations) in an include file and then expand the file within the C source. When definitions are made within an include file, problems such as incorrect display of definition lines during source debugging may occur.

Table B-1 List of Use-related Cautions

Number	Cautions
6	[Use of output assembler source] When a C source program contains descriptions that use assembly language such as #asm blocks or __asm statements, the load module file creation procedure sequence is compile, assemble, and then link. Be careful about the following points when using the assembler by outputting the assembler source to perform assembly without outputting direct objects by the CC78K0R, such as when descriptions using assembly language are used. - If symbols need to be defined in the #asm block (part between #asm and #endasm) and the __asm statement, use a symbol of 8 or less characters beginning with the strings ?L@ (for example, ?L@01, ?L@sym, etc.). However, these symbols cannot be defined externally (PUBLIC declaration). It is not possible to define segments in the #asm block and the __asm statement. If a symbol beginning with the strings ?L is not used, the Fatal error (F2114) is output during assembly. - When using variables that are extern-ed in the #asm block in C source, EXTRN is not generated if there are no references in other C descriptions, and a link error is output. Therefore, perform EXTRN in the #asm block if no referencing is done in C. - If the C source contains #asm blocks and __asm statements, specify the -a or -sa option to enable assembly descriptions, and assemble the output assembler source. When using PM+, either specify the -a/-sa options through individual option specification for sources for which only assembler source files are output, or specify the -a/-sa options through universal option specification. - When using PM+, the RA78K0R is started regardless of compiler options -o/-no when compiler option -a or -sa is specified. - When changing the segment name using the #pragma section directive, do not specify a segment having the same name as the primary name of the source file name. Otherwise, error (F2106) is output during assembly.

Table B-1 List of Use-related Cautions

Number	Cautions

7	[Creating link directive file]
	When an area outside of the ROM or RAM area of the target device is used when linking the objects created by the CC78K0R, or when you want to place the code or data at any specified address, create a link directive file and specify the -d option when linking.
	For information about creating link directive files, see RA78K0R Assembler Package Operation User's Manual and lk78k0r.dr (in the smp78k0r folder) equipped with the CC78K0R.
	
	<Example: When you want to place external variables without initial values (except sreg variables) from a certain C source file to the external memory.>
	
	(1) Change the section name for the external variables without initial value at the beginning of the C source.
	`#pragma section @@DATAL EXTDATA`
	`:`
	
	Caution Initialization of the changed segment and ROMization should be performed by changing the startup routine.
	
	(2) Create a link directive file.
	<lk78k0r.dr>
	`memory  EXTRAM  : ( 040000h , 1000h )`
	`merge   EXTDATA : = EXTRAM`
	
	Heed the following points when creating a link directive file.
	- When using the -s automatic generation option for stack symbols while linking, it is recommended to secure the stack area by the memory directive of the link directive file and specify its name explicitly. If the area name is omitted, it is used as the stack area in the RAM (except for the SFR area).
	
	<Example: When added to the link directive file lk78k0r.dr>
	`memory  EXTRAM  : ( 040000h , 1000h )`
	`memory  STK     : ( 0FB000H , 100H )`
	`merge   EXTDATA : = EXTRAM`
	
	(Command line)
	`> lk78k0r s0rml.rel prime.rel -bcl0rm.lib -sSTK -dlk78k0r.dr`
	- The following error may be output when linking in the defined memory area.
	`"*** RA78K0R error E3206: Segment 'xxx' can't allocate to memory-ignored."`
	
	[Cause]
	Because of insufficient space in the defined memory area, the indicated segment cannot be located.
	
	[Response]
	The response action is roughly divided into the following 3 steps.
	(i) Examine the size of a segment that cannot be located (refer to the .map file).
	(ii) Based on the size of the segment examined in step (i), increase the size of the area where the segment is located in the directive file.
	(iii) Specify directive file specification option -d and link.
	
	However, based on the type of the segment marked by an error in step (i), the method used to examine the segment size differs in the following way.
	- When the segment is automatically generated during compilation
	Examine the size of the segment by the map file that is linked and created.
	- When the segment is created by the user
	Examine the size of the segment that is not located by the assemble list file (.prn).

Table B-1 List of Use-related Cautions

Number	Cautions
8	[When using va_start macro] The operation of va_start macro defined in stdarg.h is not guaranteed (because the offset of the first argument differs depending on the function). Choose a macro as follows. - When the first argument is specified, use the va_starttop - When the second and subsequent arguments are specified, use the va_start macro.
9	[Startup routines and libraries] - Use the provided startup routines and libraries with the same versions as the files in the executable form (cc78k0r.exe). - (b) For the floating point support functions sprintf, vprintf, and vsprintf, if the result value of a conversion that is specified with the conversion specifiers "%f", "%e", "%E", "%g" or "%G" is smaller than the precision, the value is rounded down. "%f" conversion is executed even if the result value of conversion that is specified with "%g"/"%G" is greater than the precision. For functions sscanf and scanf, if no effective character is read during conversion that is specified with the conversion specifiers "%f", "%e", "%E", "%g", or "%G", +0 is regarded as the conversion result. If the conversion result is "±", ±0 is regarded as the conversion result. [Prevention method] None

Table B-1 List of Use-related Cautions

Number	Cautions

10	[When performing ROMization]
	ROMization consists in placing initial values such as those of external variables that have an initial value in ROM, and then copying these values to RAM during system operation. In CC78K0R, a code is generated by default for ROMization. Therefore, it is necessary to perform linking with the startup routine including ROMization during linking.
	The startup routine for the small model and medium model does not include ROMization processing for the far area. When variables are allocated in the far area using the __far qualifier or the like, use the startup routine for the large model.
	The following startup routines, all including ROMization processing, are provided by the CC78K0R.
	If the flash memory self rewrite mode for is used, refer to “8.3.3 (3) When using RTOS“.
	Startup routines:
	- When not using C standard library area: s0rm.rel, s0rl.rel
	- When using C standard library area: s0rml.rel, s0rll.rel
	
	<Usage example>
	`C>lk78k0r s0rl.rel sample.rel -s -bcl0rxm.lib -bcl0rm.lib -osample.lmf`
	
	`sample.rel:   Object module file of user program`
	`s0rl.rel:     Startup routine`
	`cl0rxm.lib:   Library that uses a multiplier`
	`cl0rm.lib:    Runtime library, standard library`
	
	The -s option is a stack symbol (_@STBEG, _@STEND) automatic generation option.
	
	Caution 1: Be sure to link the startup routine at the beginning.
	Caution 2: When creating a library, create it separately from the library provided by the CC78K0R, and specify it prior to the compiler library during linking.
	Caution 3: Do not add user functions to the CC78K0R library.
	Caution 4: When using a floating point library (cl0r*f.lib), it is necessary to link the startup routine including the ROMization processing to both the standard library and the floating point library.
	
	- When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points
	<Example>
	`-bmylib.lib -bcl0rmf.lib -bcl0rm.lib`
	
	- When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points
	<Example>
	`-bmylib.lib -bcl0rm.lib -bcl0rmf.lib`

11	[Stack area symbol generation (-s)] In CC78K0R, the user cannot secure a stack area. To secure a stack area, specify the -s option during linking. When using PM+, the -s option is automatically attached when the source file specification includes the C source.

Table B-1 List of Use-related Cautions

Number	Cautions
12	[ROM code] When ordering ROM code, specify the -r or -u object converter options, such as -r, -u0FFH (do not cancel the specification). <Example> `-r -u0FFH` `-r:` Sort HEX file contents by order of addresses. `-u`*`complement-value`*`:` Fill empty space in ROM code with the specified complement value.
13	[Help specification option] In PM+, compiler options --, -?, and -h, which display option descriptions, are ignored. For help, click the [Help] button in the [Option Setup] dialog box of each tool.
14	[-ll option specification] When using PM+, the maximum number that can be specified for the -ll option is 32,767. If a number that exceeds 32,767 is specified, specify the -ll option with another option.

Table B-1 List of Use-related Cautions

Number	Cautions

15	[When using PM+]
	(a) Parameter file created by user
	When PM+ is specified for the parameter file created by the user, those contents are loaded to the parameter file created by PM+. When creating a parameter file, be careful about the following points. Otherwise, an error will occur during build execution.
	- Specify a file with the same name as the parameter file created by PM+.
	- Do not describe the device type specification option (-c), device file search path specification option (-y), and source file.
	- No validity check is performed for the options described in the parameter file created by the user.
	(b) [Assembler Options] dialog box
	Do not specify the -c, -f, and -y options and the source file, or an error will occur during build execution.
	No validity check is performed for the options specified in the [Assembler Options] dialog box, so an error will occur during build execution in case of description errors.
	(c) Include file dependence relationship
	During checking of dependence relationships of include files during MAKE file creation with PM+, condition statements such as #if are ignored. Therefore, include files not required for build are mistaken as required files. If described as comments or character strings, they are correctly judged as without dependence relationship.
	<Example>
	`#if       0`
	`#include  "header1.h" /* Dependence relationship judged to exist */`
	`#else                 / * ! zero */`
	`#include  "header2.h"`
	`#endif`
	`/*`
	`#include  "header3.h"`
	`*/`
	header1.h is judged as required for build during the dependence relationship check. If the header1.h file exists, header1.h gets registered to ProjectWindow of PM+.
	[Prevention method]
	None. However, this has no effect on the build processing.
	(d) Project-related file settings
	The CC78K0R attribute startup routines and standard libraries can be added and deleted from the [Project] menu of PM+ or from "Add Project-Related File" displayed by right-clicking in the Project window.
	Perform the CC78K0R attribute startup routine and standard library settings from the [Startup Routine] tab in the [Compiler Options] dialog box.

16	[Prototype declaration]
	An error (E0301, E0701) will occur if a function prototype declaration does not contain a function type specification.
	<Example>
	`f ( void ) ;    /* E0301 : Syntax error */`
	`                /* E0701 : External definition syntax */`
	[Prevention method]
	Describe the function type.
	<Example>
	`int     f ( void ) ;`

Table B-1 List of Use-related Cautions

Number	Cautions

17	[Error message output]
	If there is a spelling error in the keyword at the beginning of a line outside the function, the display position of the error line may be offset and an inappropriate error output.
	
	<Example>
	`    extren int i ; /* extern spelling error. No error will occur here. */`
	`    /* comment */`
	`    void   f ( void ) ;`
	`    [EOF]          /* Error such as E0712 */`
	
	[Prevention method]
	None

18	[Description of comments in preprocessing directive]
	In the description of preprocessing directives, an error (E0803, E0814, E0821, etc.) will occur when a comment is described at the same line as a function type macro either before or within a preprocessing directive.
	
	<Example>
	`    /* com1 */      #pragma         sfr                     /* E0803 */`
	`    /* com2 */      #define         ONE     1               /* E0803 */`
	`    #define         /* com3 */      TWO     2               /* E0814 */`
	`    #ifdef          /* com4 */      ANSI_C                  /* E0814 */`
	
	`    /* com5 */      #endif`
	`    #define         SUB ( p1 ,  /* com6 */ p2 )  p2 = p1    /* E0821 */`
	
	[Prevention method]
	Describe the comment after the preprocessing directive.
	
	<Example>
	`    #pragma sfr                                             /* com1 */`
	`    #define ONE     1                                       /* com2 */`
	`    #define TWO     2                                       /* com3 */`
	`    #ifdef  ANSI_C                                          /* com4 */`
	
	`    #endif                                                  /* com5 */`
	`    #define SUB ( p1 ,  p2 )  p2 = p1                       /* com6 */`

Table B-1 List of Use-related Cautions

Number	Cautions

19	[Use of tag for structure, union, or enum]
	If the tag (for a structure, union, or enum) is used before defining it in a function prototype declaration, a warning will occur if condition (a) below is fulfilled, and an error will occur if condition (b) below is fulfilled.
	(a) If the tag is used in an argument declaration and a pointer to a structure or union is defined, the warning (W0510) will occur when a function is called.
	<Example>
	`void    func ( int , struct st ) ;`
	
	`struct  st {`
	`        char    memb1 ;`
	`        char    memb2 ;`
	`} st [ ] = {`
	`        { 1 , 'a' } , { 2 , 'b' }`
	`} ;`
	
	`void    caller ( void ) {`
	`                    /* W0510 Pointer mismatch */`
	`        func ( sizeof ( st ) / sizeof ( st [ 0 ] ) , st ) ;`
	`}`
	(b) An error (E0737) will occur if the tag is used in a return value type declaration of an argument declaration, and a structure, union, or enum type is specified.
	<Example>
	`                    /* E0737 Undeclared structure/union/enum tag */`
	`void    func1 ( int , struct st ) ;`
	`                    /* E0737 Undeclared structure/union/enum tag */`
	`struct  st      func2 ( int ) ;`
	`struct  st {`
	`        char    memb1 ;`
	`        char    memb2 ;`
	`} ;`
	
	[Prevention method]
	Define the tag of the structure, union, or enum beforehand.

20	[Initialization of array, structure, or union in function]
	Arrays, structures, and unions using something other than a static variable address, constant, or character string cannot be initialized.
	<Example>
	`void    f ( void ) ;`
	`void    f ( void ) {`
	`        char    *p , *p1 , *p2 ;`
	`        char    *ca [ 3 ] = { p , p1 , p2 } ; /* Error( E0750 ) */`
	`}`
	
	[Prevention method]
	Describe an assignment statement and use it instead of initialization.
	<Example>
	`void    f ( void ) ;`
	`void    f ( void ) {`
	`        char    *ca [ 3 ] ;`
	`        char    *p , *p1 , *p2 ;`
	`        ca [ 0 ] = p ;  ca [ 1 ] = p1 ; ca [ 2 ] = p2 ;`
	`}`

Table B-1 List of Use-related Cautions

Number	Cautions

21	[extern callt function]
	If the address of an extern callt function is referenced by initializing the function table, etc., and the callt function is called by the same module, the assemble list is illegal and an error will occur during assembly.
	<Example>
	`callt   extern  void    funca ( void ) ;`
	`callt   extern  void    funcb ( void ) ;`
	`callt   extern  void    funcc ( void ) ;`
	
	`static  void    ( * const func [ ] ) ( ) = {`
	`        funca , funcb , funcc`
	`} ;`
	`void    func2 ( void ) {`
	`        funcc ( ) ;`
	`        funcb ( ) ;`
	`        funca ( ) ;`
	`}`
	
	[Prevention method]
	Separate the function table and function call module.

22	[Functions returning a structure]
	When a function returns a structure, an interrupt is generated in the process of returning a return value. If there is a call of the same function during interrupt servicing, the return value is illegal after the interrupt servicing ends.
	<Example>
	`struct  str {`
	`        char    c ;`
	`        int     i ;`
	`        long    l ;`
	`} st ;`
	
	`struct  str     func ( ) {`
	`        /*  Interrupt occurrence */`
	`          :`
	`}`
	
	`void    main ( ) {`
	`        st = func ( ) ; /* Interrupt occurrence */`
	`}`
	
	If the func function is called at the interrupt destination during the above servicing, st may be corrupted.
	
	[Prevention method]
	None

Table B-1 List of Use-related Cautions

Number	Cautions

23	[Union initialization]
	An error (E0750) will occur when, during initialization of unions having structures, unions, or arrays as members, the initializer syntax is specified with nesting.
	<Example>
	`    struct  Ss {`
	`            int     d1 , d2 ;`
	`    } ;`
	`    union   Au {`
	`            struct  Ss t1 ;`
	`    } u = { { 1 , 2 } } ;   /* E0750  Initializer syntax */`
	[Prevention method]
	Do not specify the initializer of a union with nesting.
	<Example>
	`    struct  Ss {`
	`            int     d1 , d2 ;`
	`    } ;`
	`    union   Au {`
	`            struct  Ss t1 ;`
	`    } u = { 1 , 2 } ;`

24	[Kanji code (2-byte code) classification] To use a source containing EUC code, set the environmental variable LANG78K to euc, or specify the -ze option.
25	[Section start address specification] The size of the section whose start address is specified with the #pragma section directive is always an even number.

APPENDIX C COMMAND OPTIONS

In this chapter the program options are summarized in table format.

Use this when developing programs.

This option table can be used as an option index.

Table C-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
Device type specification	`-cdevice-type`	Specifies the type of target device.	Independent	Specification of this option cannot be omitted.
Object module file creation specification	`-o[output-file-name]`	Specifies the output of the object module files.	If -o and -no are specified simultaneously, the last one specified is enabled.	-o*input-file-name*.rel
	`-no`	Specifies not to output the object module file.		
Memory assignment specification	`-rprocess-type` (Multiple specifications are possible)	Specifies the method of memory assignment.	If -r and -nr, -rd and -nr, -rs and -nr are specified simultaneously, the last ones specified are enabled.	-nr
	`-rd[n][m] (n = 1, 2, 4)`	Specifies external variables/external static variables are automatically assigned to the saddr area.		
	`-rs[n][m] (n = 1, 2, 4)`	Assigns a static auto variable automatically to the saddr area.		
	`-nr`	The -r, -rd, -rk and -rs options are disabled.		
Optimization specification	`-q[optimization-type]` (Multiple specifications are possible)	Specifies calling the optimization phase to generate efficient objects.	If -q and -nq are specified simultaneously, the last one specified is enabled.	-qcjlvw
	`-nq`	Invalidates the -q option.		
Debugging information output specification	`-g[n] (n = 1, 2)`	Specifies the output of the source level debugging information.	If -g and -ng are specified simultaneously, the last one specified is enabled.	-g2
	`-nq`	Invalidates the -g option.		
Preprocess list file creation specification	`-p[output-file-name]`	Specifies the output of the preprocess list files.	If -p is not specified, then -k is disabled.	None (no file is output)
	`-k[process-type]` (Multiple specifications are possible)	Specifies processing for the preprocess lists.		-kfln

Table C-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
Preprocess specification	`-dmacro-name[=definition-name][,macro-name[=definition-name]]...` (Multiple specifications are possible)	Specifies processing which is compatible for text that is defined in the C source.	Independent	Only the macro definitions in a C source module file are valid.
	`-umacro-name[,macro-name]...` (Multiple specifications are possible)	Disables macro definitions similar to the #undef statement in the C source.	Independent	A macro definition specified with -d is valid.
	`-ifolder[,folder]...` (Multiple specifications are possible)	Specifies input of the include files specified by the #include statement in the C source from the specified folder.	Independent	1. Folder with source file 2. Folder specified by environment variable INC78K0R 3. C:\Program Files\NEC Electronics Tools\CC78K0R\Vx.xx\inc78k0r
Assembler source module file creation specification	`-a[output-file-name]`	Specifies the output of the assembler source module file.	If -a and -sa are specified simultaneously, then -sa is disabled.	No assembler source module file is output.
	`-sa[output-file-name]`	Adds the C source as a comment to the assembler source module file.		
Error list file creation specification	`-e[output-file-name]`	Specifies the output of the error list file.	Independent	No error list file is output.
	`-se[output-file-name]`	Adds the C source module file to the error list file.	Independent	
Cross-reference list file creation specification	`-x[output-file-name]`	Specifies the output of the cross-reference list file.	Independent	No cross-reference list file is output.

Table C-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
List format specification	`-lw[`*`number-of-characters`*`]`	Specifies the number of characters in 1 line of each type of list file.	Independent	-lw132 (For console output, this becomes 80 characters)
	`-ll[`*`number-of-lines`*`]`	Specifies the number of lines on 1 page of each type of list file.	Independent	There is no page break
	`-lt[`*`number-of-characters`*`]`	The -lt option indicates the basic number of characters for outputting a horizontal tabulation (HT) code in the source module file, replacing it with several blanks (spaces) in each list (tabulation processing).	Independent	-lt8
	`-lf`	Specifies adding the new page break code at the end of each list file.	Independent	The new page break code will not be added.
	`-li`	Adds the C source of the include file to the assembler source module file with C source comments.	Independent	No C sources of the include file will be added.
Warning output specification	`-w[`*`level`*`]`	Specifies the output of warning messages to the console.	Independent	-w1
Execution state display specification	`-v`	Specifies whether or not the current compilation execution status is output to the console.	If -v and -nv are specified simultaneously, the last one specified is enabled.	-nv
	`-nv`	Invalidates the -v option.		
Parameter file specification	`-f`*`file-name`*	Specifies the input of the options or input file name from the specified file.	Independent	The input of an option and an input file name is possible only from a command line.

Table C-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
Temporary file creation folder specification	`-t`*`folder`*	Creates temporary files in specified drives and folders.	Independent	The temporary files are created in the drive folder specified by the environment variable TMP. If not specified, the files are created in the current drive and current folder.
Help specification	`--/-?/-h`	The --, -?, and -h options display brief explanations of the options and the help messages such as the default options on the console (valid only in the command line).	All other options are disabled.	Nothing is displayed
Function expansion specification	`-z`*`type`* (Multiple specifications are possible)	Enables extended functions.	If -z and -nz are specified simultaneously, the last one specified is enabled.	-nz
	`-nz`	Invalidates the -z option.		
Device file search path	`-y`*`folder`*	Specifies paths that search device files.	Independent	Normal search path only
Memory model specification	`-m`*`type`*	Specifies the memory model used for compilation.	Independent	-mm

INDEX

For further information, please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706