

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Note that the following URLs in this document are not available:

<http://www.necel.com/>

<http://www2.renesas.com/>

Please refer to the following instead:

Development Tools | <http://www.renesas.com/tools>

Download | http://www.renesas.com/tool_download

For any inquiries or feedback, please contact your region.

<http://www.renesas.com/inquiry>

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.



User's Manual

CC78K0S Ver. 2.00

C Compiler

Operation

Target Device
78K0S Microcontrollers

Document No. U17416EJ2V0UM00 (2nd edition)

Date Published August 2007

© NEC Electronics Corporation 2005

Printed in Japan

[MEMO]

Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

PC/AT is a trademark of International Business Machines Corporation.

i386 is a trademark of Intel Corporation.

- **The information in this document is current as of August, 2007. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

INTRODUCTION

The purpose of this manual is to enable complete understanding of the functions and operation of the CC78K0S (78K0S Microcontrollers C Compiler).

This manual does not explain how to write CC78K0S source programs. Therefore, before reading this manual, please read “**CC78K0S C Compiler Language User’s Manual (U17415E)**” (hereafter called the “Language manual”).

[Target Devices]

Software for 78K0S Microcontrollers can be developed by using the CC78K0S. To use this software, the RA78K0S (78K0S Microcontrollers 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 CC78K0S in microcontroller development.

CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION

This chapter describes how to install the CC78K0S, 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 CC78K0S and presents examples showing the processes from compiling to linking.

CHAPTER 4 CC78K0S FUNCTIONS

This chapter describes optimization methods and ROMization functions in the CC78K0S.

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 CC78K0S.

CHAPTER 7 USING C COMPILER

This chapter introduces techniques to aid in the skillful use of the CC78K0S.

CHAPTER 8 STARTUP ROUTINES

The CC78K0S 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 CC78K0S.

APPENDIXES

The appendices provide and a sample program, a list of the cautions encountered during use, lists of the restrictions related to the CC78K0S and an index.

[How to Read This Manual]

First, those who want to see how to actually use CC78K0S, 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.
CC78K0S Ver. 2.00 C Compiler	Operation	This document
	Language	U17415E
RA78K0S Ver. 2.00 Assembler Package	Operation	U17391E
	Language	U17390E
	Structured assembly language	U17389E
SM+ System Simulator	Operation	U18601E
	User Open Interface	U18212E
SM78K Series Ver. 2.52 System Simulator	Operation	U16768E
PM+ Ver. 6.30 Project Manager		U18416E
ID78K0S-NS Ver. 2.52 Integrated Debugger	Operation	U16584E
ID78K0S-QB Ver. 3.00 Integrated Debugger	Operation	U17287E

[Conventions]

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

RTOS:	Real-time OS for 78K0 Microcontrollers RX78K0
...:	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).
<u> </u> :	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] [N] [[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:\nectools32\smp78k0S\CC78k0S\prime.C

- 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.)

[MEMO]

CONTENTS

CHAPTER 1 OVERVIEW ...	13
1.1 Role of CC78K0S ...	13
1.2 Development Procedure Using CC78K0S ...	15
1.2.1 Using editor to create source module files ...	16
1.2.2 C compiler ...	17
1.2.3 Assembler ...	18
1.2.4 Linker ...	19
1.2.5 Object converter ...	20
1.2.6 Librarian ...	21
1.2.7 Debugger ...	22
1.2.8 System simulator ...	23
1.2.9 PM+ ...	24
CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION ...	25
2.1 Host Machines and Supply Media ...	25
2.2 Installation ...	26
2.3 Installation of Device Files ...	27
2.4 Folder Configuration ...	28
2.5 Uninstallation Procedure ...	29
2.6 Environment Settings ...	30
2.6.1 Host machine ...	30
2.6.2 Environment variables ...	30
2.6.3 File organization ...	31
2.6.4 Library files ...	32
CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING ...	35
3.1 PM+ ...	35
3.1.1 Position of cc78k0sp.dll (tools DLL) ...	35
3.1.2 Execution environment ...	35
3.1.3 CC78K0S option setting menu ...	36
3.1.4 Description of each part of < Compiler Options > dialog box ...	39
3.2 Procedure ...	59
3.2.1 MAKE from PM+ ...	59
3.2.2 Compiling to linking in command line (for DOS prompt) ...	61
3.3 I/O Files of C Compiler ...	63
3.4 Execution Start and End Messages ...	65
3.4.1 Execution start message ...	65
3.4.2 Execution end message ...	65
CHAPTER 4 CC78K0S FUNCTIONS ...	66
4.1 Optimization Method ...	66
4.2 ROMization Function ...	68
4.2.1 Linking ...	68
CHAPTER 5 COMPILER OPTIONS ...	69
5.1 Specifying Compiler Options ...	69
5.2 Prioritization ...	70
5.3 Types ...	72
5.4 Descriptions ...	74
CHAPTER 6 C COMPILER OUTPUT FILES ...	119
6.1 Object Module File ...	119
6.2 Assembler Source Module File ...	120
6.3 Error List File ...	125
6.3.1 Error list file with C source ...	125

6.3.2 Error list file with error message only ...	127
6.4 Preprocess List File ...	128
6.5 Cross-Reference List File ...	130
CHAPTER 7 USING C COMPILER ...	133
7.1 Efficient Operation (EXIT Status Function) ...	133
7.2 Setting Up Development Environment (Environment Variables) ...	134
7.3 Interrupting Compilation ...	135
CHAPTER 8 STARTUP ROUTINES ...	136
8.1 File Organization ...	136
8.1.1 bat folder contents ...	137
8.1.2 src folder contents ...	138
8.2 Batch File Description ...	139
8.2.1 Batch files for creating startup routines ...	139
8.3 Startup Routines ...	140
8.3.1 Overview of startup routines ...	140
8.3.2 Description of sample program (cstart.asm) ...	142
8.3.3 Revising startup routines ...	149
CHAPTER 9 ERROR MESSAGES ...	152
9.1 Error Message Format ...	152
9.2 Types of Error Messages ...	153
9.3 List of Error Messages ...	154
9.3.1 Error message for a command line ...	155
9.3.2 Error message for an internal error and memory ...	159
9.3.3 Error message for a character ...	160
9.3.4 Error message for configuration element ...	161
9.3.5 Error message for conversion ...	164
9.3.6 Error message for an expression ...	165
9.3.7 Error message for a statement ...	169
9.3.8 Error message for a declaration and function definition ...	171
9.3.9 Error message for a preprocessing directive ...	177
9.3.10 Error message for fatal file I/O and running on an illegal operating system ...	181
APPENDIX A SAMPLE PROGRAMS ...	183
A.1 C Source Module File ...	183
A.2 Execution Example ...	184
A.3 Output List ...	185
A.3.1 Assembler source module file ...	185
A.3.2 Preprocess list file ...	190
A.3.3 Cross-reference list file ...	191
A.3.4 Error list file ...	192
APPENDIX B LIST OF USE-RELATED CAUTIONS ...	193
APPENDIX C COMMAND OPTIONS ...	203
APPENDIX D INDEX ...	207

LIST OF FIGURES

Figure No. Title, Page

1-1	Development Process ...	13
1-2	Software Development Process ...	14
1-3	Program Development Procedure Using CC78K0S ...	15
1-4	Creating Source Module Files ...	16
1-5	C Compiler Function ...	17
1-6	Assembler Function ...	18
1-7	Linker Function ...	19
1-8	Object Converter Function ...	20
1-9	Librarian Function ...	21
1-10	Debugger Function ...	22
1-11	Simulator Function ...	23
1-12	PM+ Function ...	24
2-1	Folder Configuration ...	28
3-1	< Compiler Options > Dialog Box ...	36
3-2	< Browse for Folder > Dialog Box ...	37
3-3	< ParameterFile > Dialog Box ...	37
3-4	< Edit Option > Dialog Box ...	38
3-5	< Add Option > Dialog Box ...	38
3-6	< Compiler Options > Dialog Box ...	39
3-7	< Compiler Options > Dialog Box (When << Preprocessor >> Tab Is Selected) ...	41
3-8	< Compiler Options > Dialog Box (When << Memory Model >> Tab Is Selected) ...	42
3-9	< Compiler Options > Dialog Box (When << Data Assign >> Tab Is Selected) ...	43
3-10	< Compiler Options > Dialog Box (When << Integrated Recommendable Optimizing Option >> Is Selected) ...	44
3-11	< Compiler Options > Dialog Box (When << Char Expression Behavior, Automatic Allocation >> Is Selected) ...	45
3-12	< Compiler Options > Dialog Box (When << Optimize Object Size by Calling Library >> Is Selected) ...	46
3-13	< Compiler Options > Dialog Box (When << Others >> Is Selected) ...	47
3-14	< Compiler Options > Dialog Box (When << Debug >> Tab Is Selected) ...	48
3-15	< Compiler Options > Dialog Box (When << Object Module File, Assembler Source Module File >> Is Selected) ...	49
3-16	< Assembler Options > Dialog Box ...	50
3-17	< Compiler Options > Dialog Box (When << Error List File, Cross-reference List File >> Is Selected) ...	51
3-18	< Compiler Options > Dialog Box (When << Preprocess List File, List Format >> Is Selected) ...	52
3-19	< Compiler Options > Dialog Box (When << Extend >> Tab Is Selected) ...	54
3-20	< Compiler Options > Dialog Box (When << Others >> Tab Is Selected) ...	55
3-21	< Compiler Options > Dialog Box (When << Startup Routine >> Tab Is Selected) ...	57
3-22	< Compiler Options > Dialog Box (When << Optimize >> Tab Is Selected) ...	59
3-23	< Linker Options > Dialog Box ...	60
3-24	C Compiler I/O Files ...	64
5-1	< Compiler Options > Dialog Box ...	74
8-1	Overview of Startup Routines ...	140
8-2	ROMization Processing ...	146

LIST OF TABLES

Table No.	Title	Page
2-1	Supply Media and Recording Formats for C Compiler ...	25
2-2	Environment Variables ...	30
2-3	File Organization (* = Alphanumeric Symbols) ...	31
2-4	Library Files ...	32
3-1	C Compiler I/O Files ...	63
4-1	Optimization Methods ...	66
5-1	Prioritization of Compiler Options ...	70
5-2	List of Compiler Options ...	72
5-3	Process Types Specifiable by -r Option ...	79
5-4	Interpretation When Specifying NR ...	80
5-5	Maximum (-rd) Range of Variables ...	81
5-6	Maximum (-rk) Range of Variables to be Assigned ...	82
5-7	Maximum (-rs) Range of Variables to be Assigned ...	83
5-8	Optimization Types ...	84
5-9	Operation Differs Depending on Value of n ...	87
5-10	Process Types of -k Option ...	89
5-11	Warning Message Levels ...	109
5-12	Type Specifications of -z Option ...	114
6-1	Description of Output Items (assembler source module file) ...	122
6-2	Description of Output Items (error list file with C source) ...	126
6-3	Description of Output Items (error list file with error message only) ...	127
6-4	Description of Output Items (preprocess list file) ...	128
6-5	Description of Output Items (cross-reference list file) ...	131
8-1	bat Folder Contents ...	137
8-2	src Folder Contents ...	138
8-3	Differences Between Startup Routine Sources ...	141
8-4	Correspondence Between Source Files and Object Files ...	141
8-5	ROM Area for Initial Values ...	146
8-6	RAM Area for Initial Values (Copy Destination) ...	146
8-7	Symbols Used in Library Functions ...	149
9-1	Error Message for a Command Line < from 0001 > ...	155
9-2	Error Message for an Internal Error and Memory < from 0101 > ...	159
9-3	Error Message for a Character < from 0201 > ...	160
9-4	Error Message for Configuration Element < from 0301 > ...	161
9-5	Error Message for Conversion < from 0401 > ...	164
9-6	Error Message for an Expression < from 0501 > ...	165
9-7	Error Message for a Statement < from 0601 > ...	169
9-8	Error Message for a Declaration and Function Definition < from 0701 > ...	171
9-9	Error Message for a Preprocessing Directive < from 0801 > ...	177
9-10	Error Message for Fatal File I/O and Running on an Illegal Operating System < from 0901 > ...	181
B-1	List of Use-related Cautions ...	193
C-1	Compiler Options ...	203

CHAPTER 1 OVERVIEW

The CC78K0S C compiler program translates C source programs written in ANSI-C^{Note} or the C language for the 78K0S Series into the machine language for the 78K0S Series.

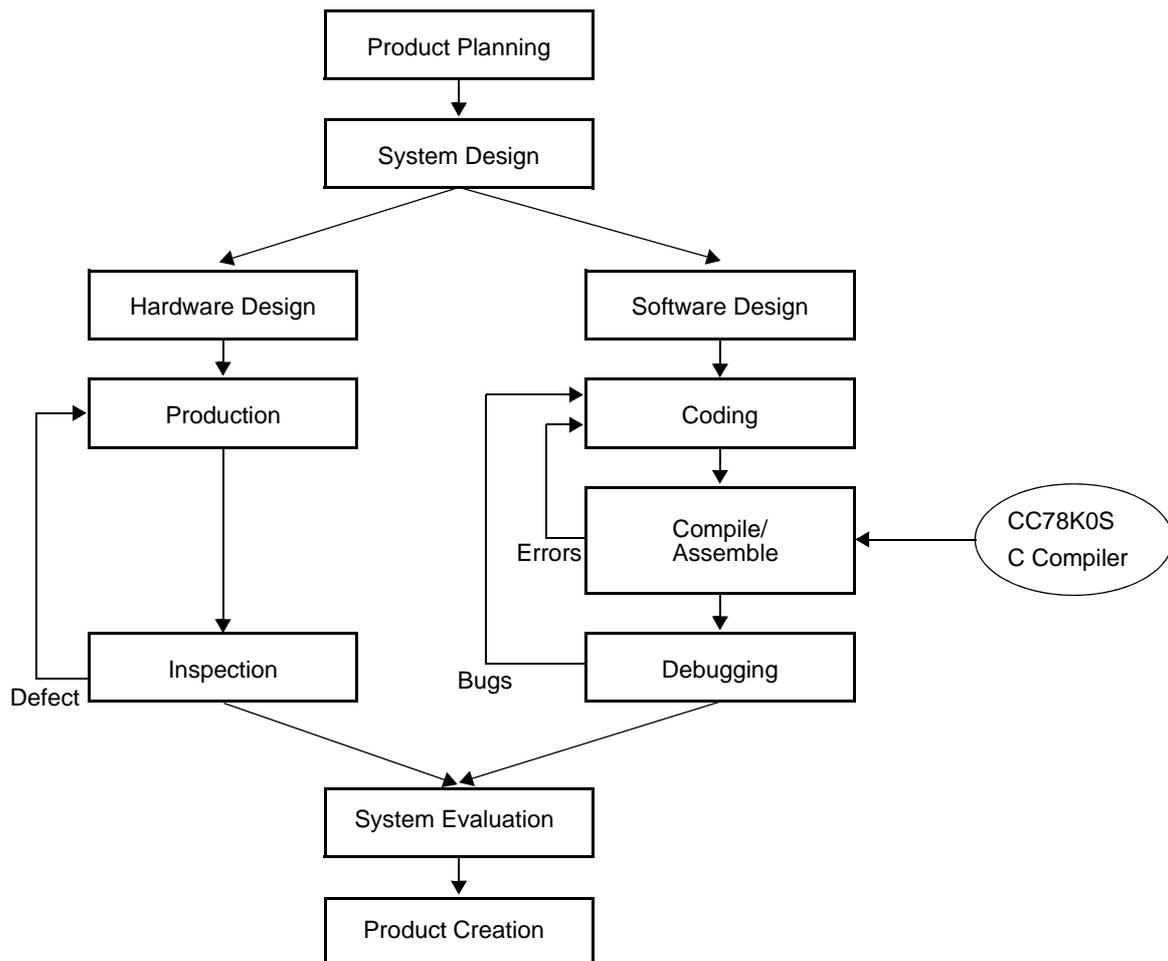
The CC78K0S can be started up on Windows by using PM+ supplied with the assembler package for the 78K0S Series. When PM+ is not used, the compiler is started up on the DOS.

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

1.1 Role of CC78K0S

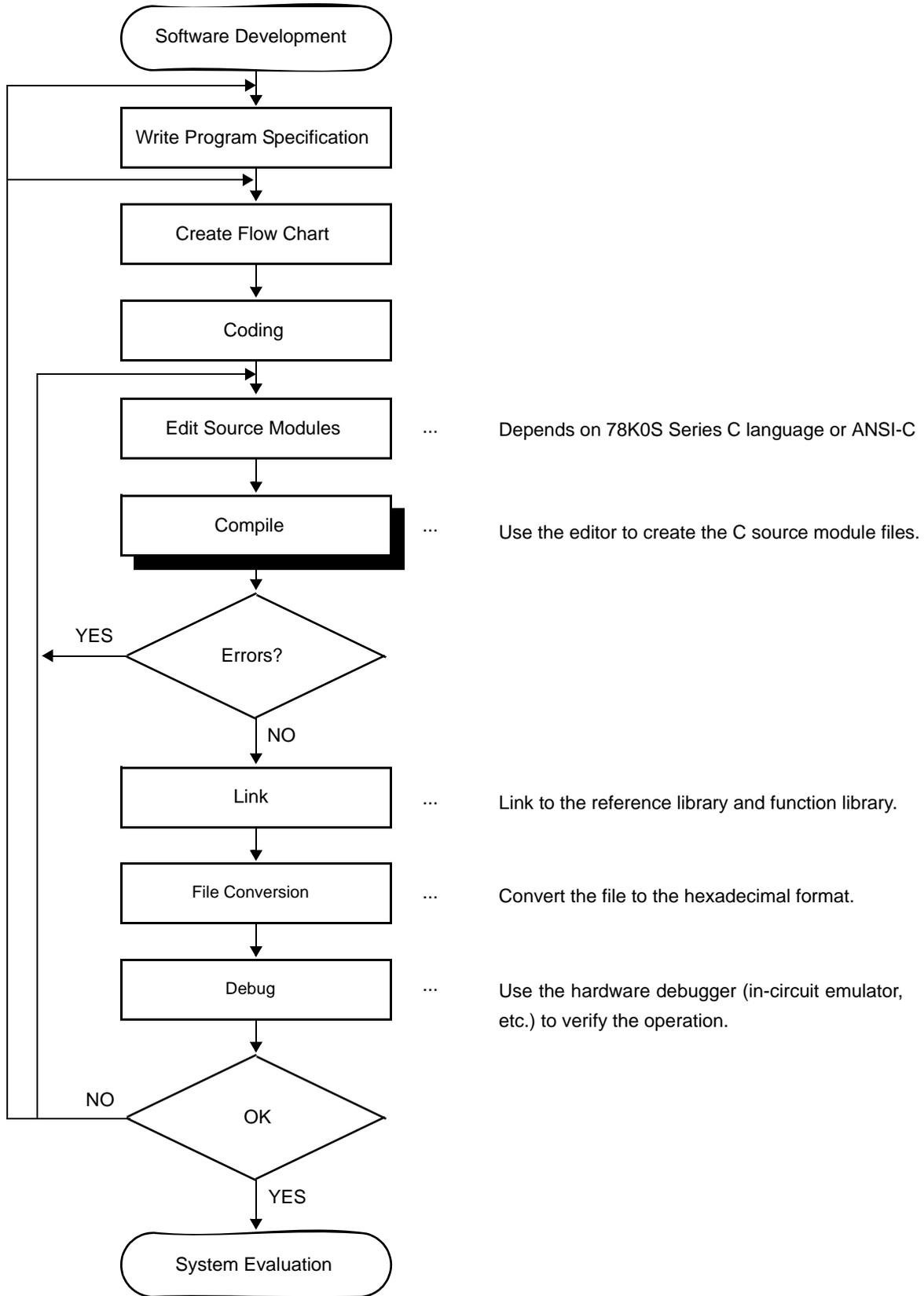
The position of CC78K0S in product development is shown below.

Figure1-1 Development Process



The software development process is shown below.

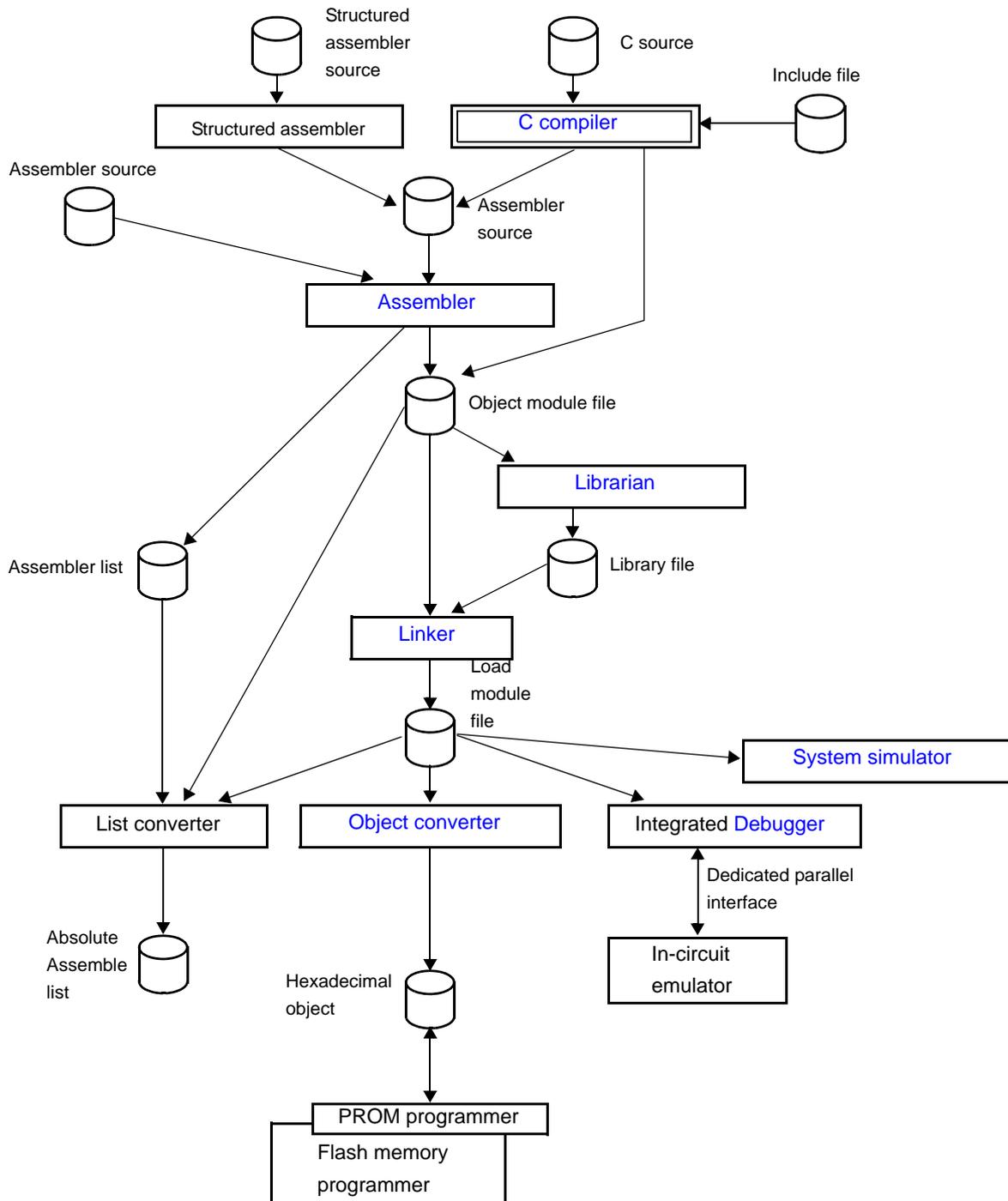
Figure1-2 Software Development Process



1.2 Development Procedure Using CC78K0S

The development procedure using CC78K0S is shown below.

Figure1-3 Program Development Procedure Using CC78K0S



1.2.1 Using editor to create source module files

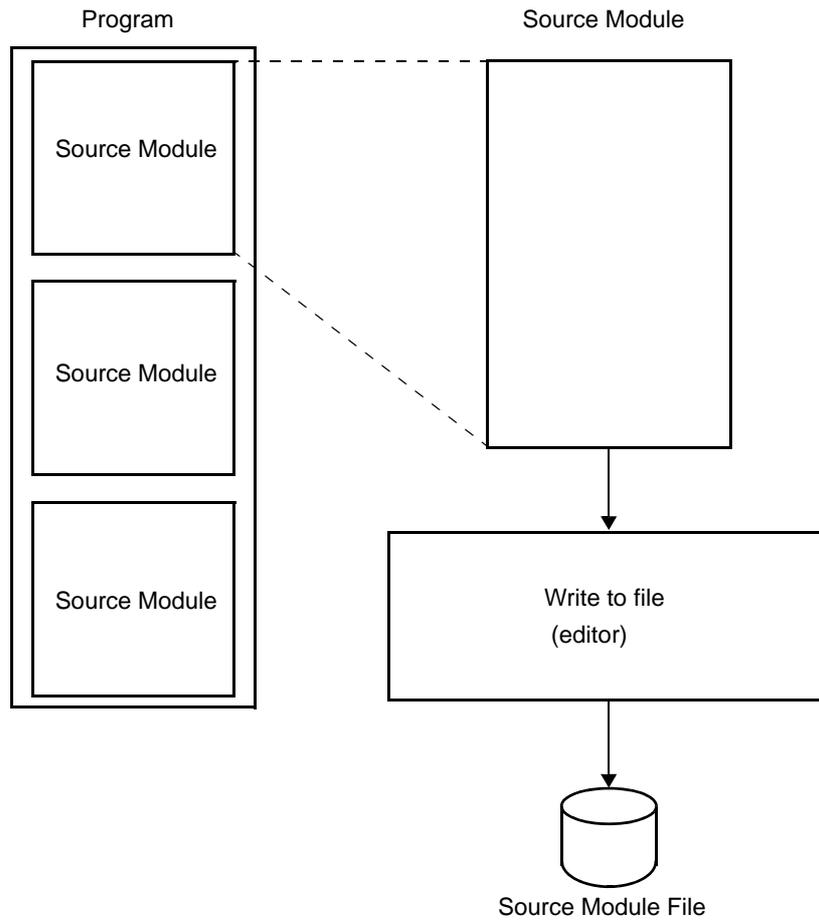
1 program is divided into several functional modules.

1 module is the coding unit and becomes the input unit to the 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 CC78K0S input files.

Figure1-4 Creating Source Module Files

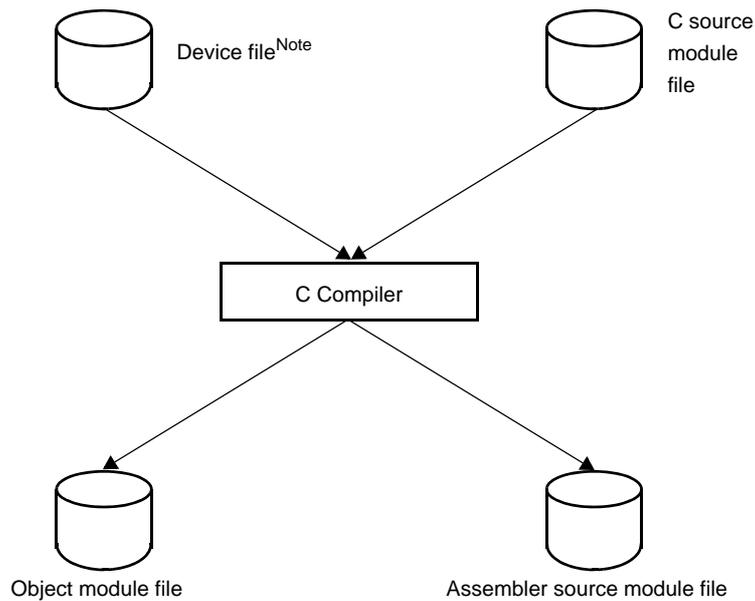


1.2.2 C compiler

The C compiler translates C language into a machine language, taking a 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").

Figure1-5 C Compiler Function



Note 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/tool/DeviceFile/list.html>

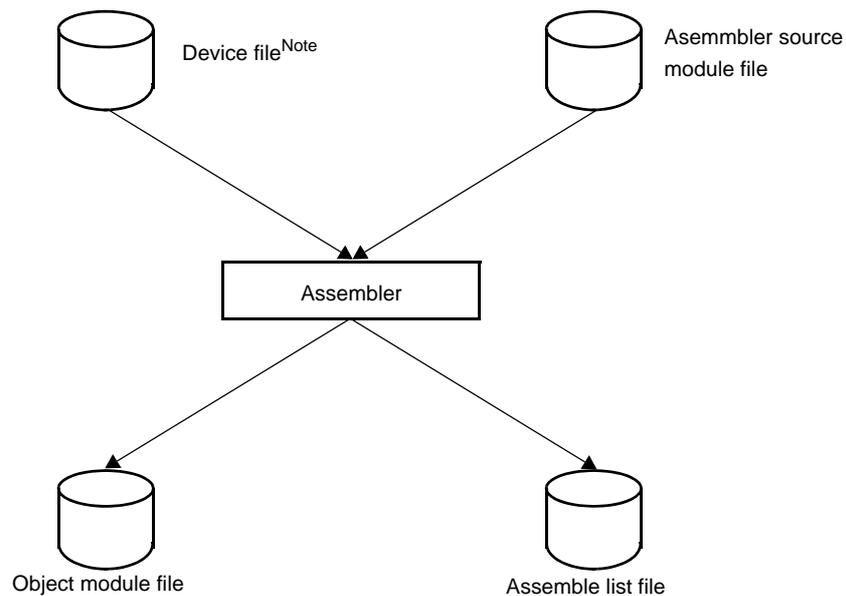
1.2.3 Assembler

Assembly is executed by using the assembler included in the RA78K0S assembler package (sold separately).

The assembler is a program that inputs an 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 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.

Figure1-6 Assembler Function



Note 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/tool/DeviceFile/list.html>

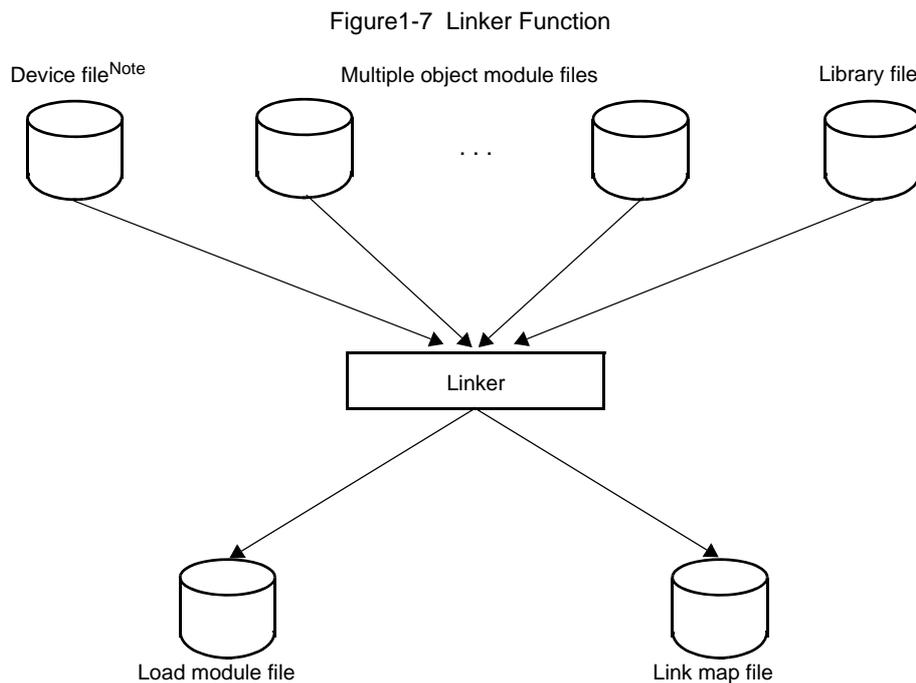
1.2.4 Linker

Linking is performed by using the linker included in the RA78K0S Assembler Package (sold separately).

The linker inputs multiple object module files output by the 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.



Note 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/tool/DeviceFile/list.html>

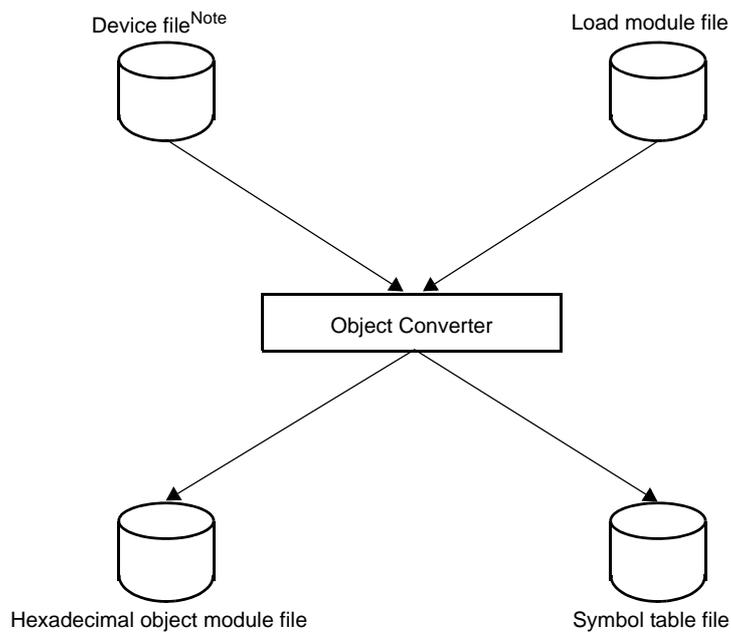
1.2.5 Object converter

The object converter uses the converter included in the RA78K0S Assembler Package (sold separately).

The object converter inputs a load module file output by the linker and converts its file format. The result is output as an intel-standard hexadecimal object module file.

Symbol information is output as a symbol table file.

Figure1-8 Object Converter Function



Note 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/tool/DeviceFile/list.html>

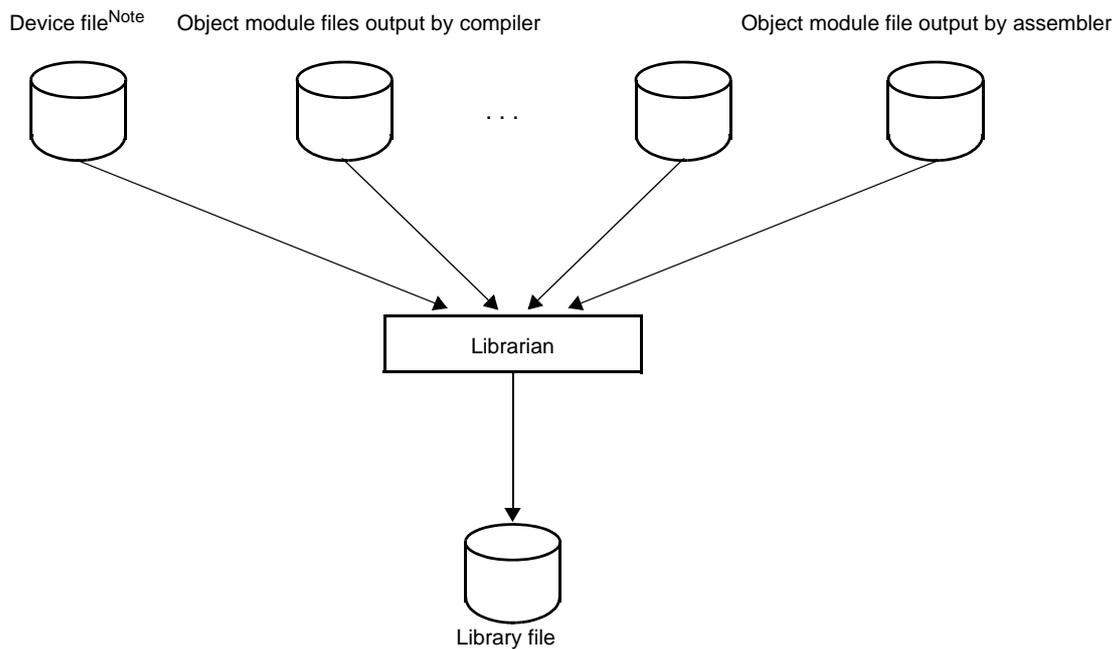
1.2.6 Librarian

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 is used to create and update library files. The librarian uses the librarian included in the RA78K0S Assembler Package (sold separately).

Figure1-9 Librarian Function



Note 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/tool/DeviceFile/list.html>

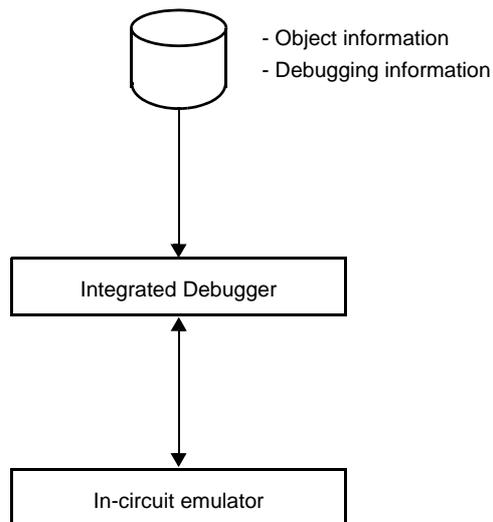
1.2.7 Debugger

Source debugging using a graphical user interface becomes possible by loading the load module files output by the linker into the IE (in-circuit emulator) by using the ID78K0S-NS (78K0S Series 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).

Figure1-10 Debugger Function

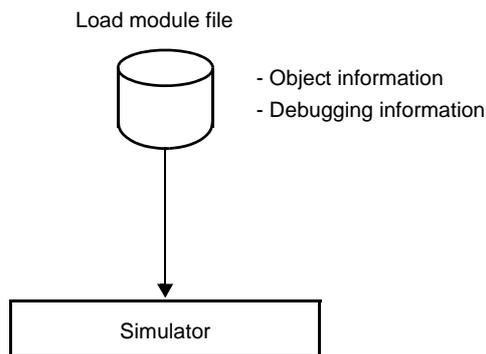


1.2.8 System simulator

Source debugging using a graphical user interface becomes possible by downloading the load module files output from the linker by using the SM78K0S (78K0S Series system simulator).

SM78K0S is software that has the same operating image as the ID78K0S-NS and performs simulations on the host machine. In addition to simulating machine instructions in the SM78K0S, 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.

Figure1-11 Simulator Function

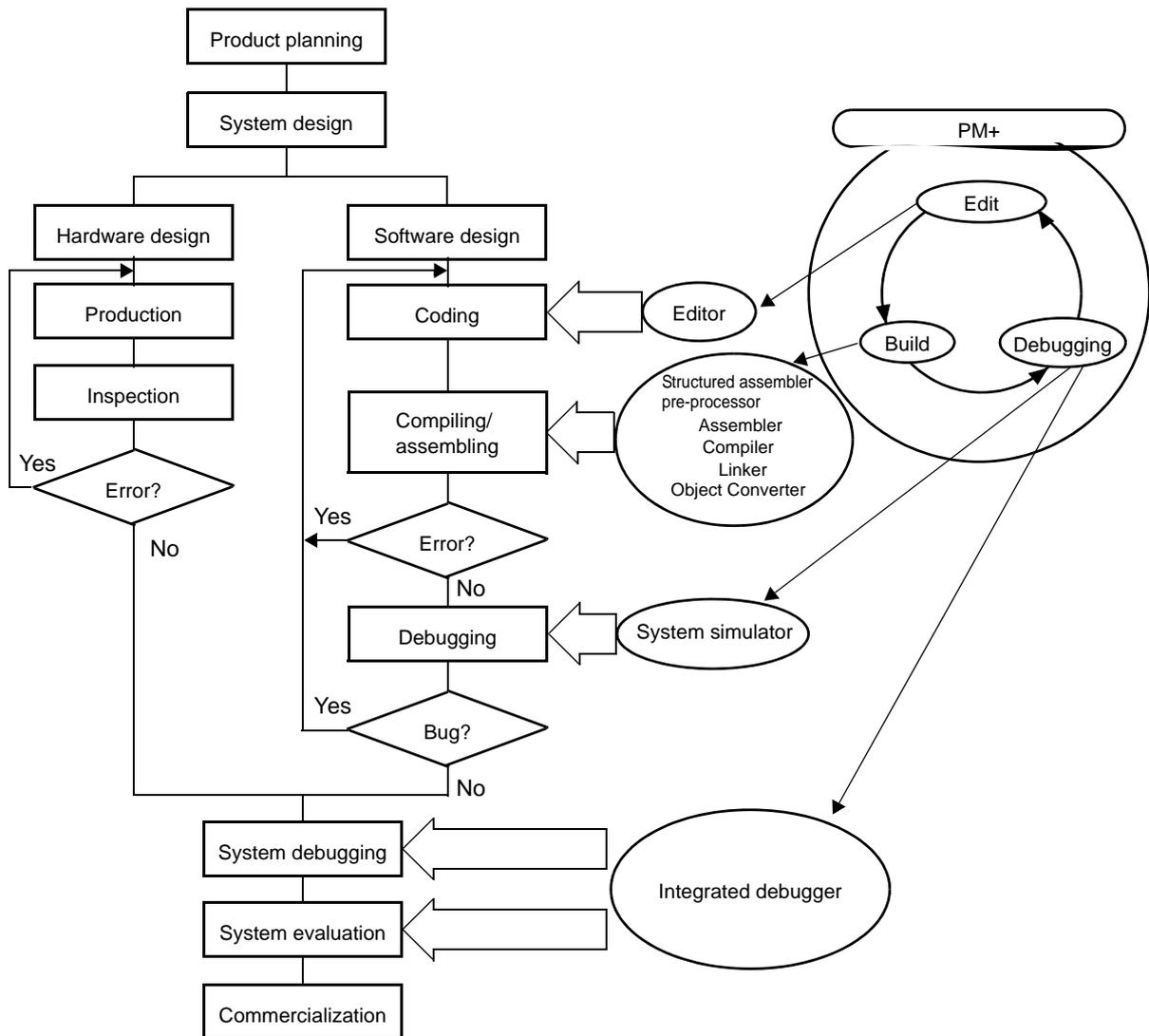


1.2.9 PM+

PM+ is software that uses the DLL files added to CC78K0S and is able to start CC78K0S on Windows. Editing the source, automatically creating the MAKEFILE, and compiling to linking can be performed from the startup screen of PM+. Thus, editing to debugging can be performed using GUI images.

PM+ is included to the RA78K0S Assembler Package (sold separately). The installer for the RA78K0S Assembler Package is used to install and to make the settings. If CC78K0S will be started from PM+, install the RA78K0S Assembler Package before installing the compiler.

Figure1-12 PM+ Function



Remark Build analyzes and executes the make file to create the executable file. The dependency relationships described in the make file basically remove unused assembling, compiling, and linking and can create efficient executable files.

CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION

This chapter explains the procedure to install the files stored in the supply media of the CC78K0S to the user development environment (host machine) and the procedure to uninstall them from the user development environment.

2.1 Host Machines and Supply Media

This C compiler supports the development environments listed in [Table2-1](#).

Table2-1 Supply Media and Recording Formats for C Compiler

Host Machine	OS	Supply Media
IBM PC/AT™ compatibles	Windows (98/Me/2000/XP/NT 4.0) ^{Note}	CD-ROM

Note PM+ is required if the C compiler is used on Windows. The C compiler can be started up from the DOS prompt (Windows 98/Me) or command prompt (Windows 2000/XP/NT 4.0) if PM+ is not used.

2.2 Installation

The procedure for installing to the host machine the files provided in the CC78K0S'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 CC78K0S'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.

Remark If the setup program does not start automatically, execute SETUP.EXE in the CC78K0S\DISK1 folder.

(3) Confirmation of files

Using Windows Explorer, etc., check that the files contained in the CC78K0S'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/tool/DeviceFile/list.html>

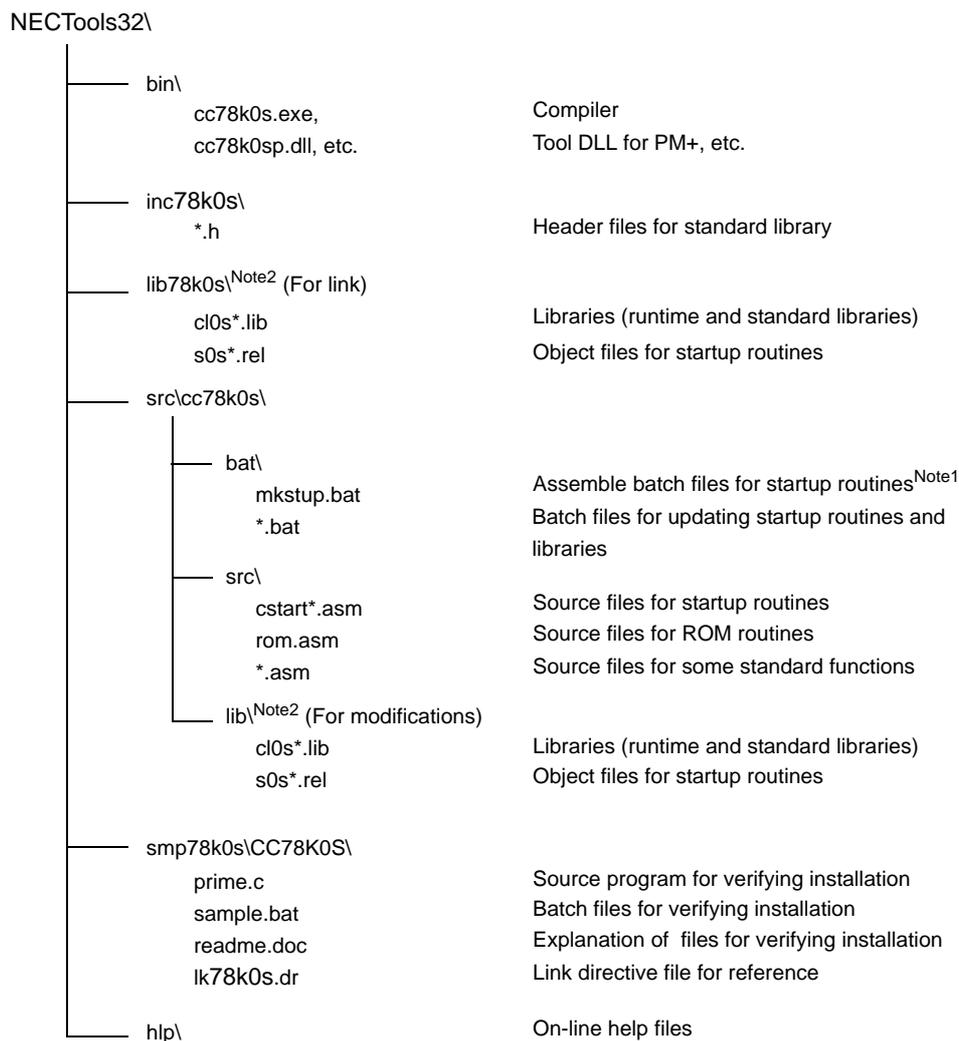
Use the device file installer to install the device files. The device file installer is installed at the same time as the CC78K0S.

2.4 Folder Configuration

The standard folder displayed during installation is "NECTools32" 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 (CC78K0S, RA78K0S) to the same drive and folder.

The descriptions in this manual assume installation to the standard folder with "NECTools32", which is the default program name, according to the setup program default directions.

Figure2-1 Folder Configuration



Notes 1. This batch file cannot be used in PM+. To use the batch file, run it from the DOS prompt (Windows 98/Me) or command prompt (Windows 2000/XP/NT 4.0).

Notes 2. To modify the start-up routine, modify the following source below `src\cc78k0s\lib` folder. A file assembled with the batch file is stored in `src\cc78k0s\lib`, so copy this file to the `lib78k0s` folder, then execute linking.

2.5 Uninstallation Procedure

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) Opening [Control Panel] window

Press the [Start] button and select [Settings]-[Control Panel] to open the [Control Panel] window.

(3) Opening of [Add/Remove Programs] window

Double-click the [Add/Remove Programs] icon in the [Control Panel] window to open the [Add/Remove Programs] window.

Remark On Windows XP, [Add or Remove Programs] is displayed instead of [Add/Remove Programs].

(4) Deletion of CC78K0S

After selecting "NEC CC78K0S 78K/0S C Compiler Vx.xx" from the list of installed software displayed in the [Install/Uninstall] tab in the [Add/Remove Programs] window, click the [Add/Remove] button.

When the [System Setting Change] window is opened, click the [Yes] button.

(5) 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](#)".

2.6 Environment Settings

2.6.1 Host machine

The CC78K0S handles 32 bits and runs on models equipped with the i386™ CPU or later versions.

Since handling 32 bits is implemented by using DOS Extender, it is designed to run on the following operating systems.

- Windows98/Me/2000/XP/NT4.0
- DOS prompt in Windows98/Me
- Command prompt in Windows2000/XP/NT4.0

2.6.2 Environment variables

Set the following environment variables for DOS prompt (Windows 98/Me) or command prompt (Windows 2000/XP/NT 4.0) operation.

Table2-2 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. sjs : Shift JIS (Default) euc : EUC none : No 2-byte codes
INC78K0S	Specifies the folder where the standard header files of the compiler are located.
LIB78K0S	Specifies the folder where the compiler's libraries are located.

Specification Example

```

PATH = %PATH% ; C:\NECTools32\bin
set    TMP = C:\
set    LANG78K = sjs

```

2.6.3 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.

Table2-3 File Organization (* = Alphanumeric Symbols)

Folder Name	File Name	Description
bin\	cc78k0s.exe	Compiler
	cc78k0s.msg	Message file
	*.hlp	Help files
	*.dll	DLL files
inc78k0s\	*.h ^{Note 1}	Header files for standard library
src\cc78k0s\ bat ^{Note 2}	mkstup.bat	Assemble batch files for startup routines
	reprom.bat	For updating rom.asm
	*.bat ^{Note 3}	Batch files for updating standard functions (partial)
src\cc78k0s\ src	cstart*.asm ^{Note 4}	Source files for startup routines
	rom.asm	Source files for ROMization routine
	*.asm ^{Note 5}	Source files for standard functions (partial)
hlp	*.chm	On-line help file

Notes 1. Refer to CC78K0S C Compiler Language User's Manual.

Notes 2. The batch files in this folder cannot be used in PM+. Use these batch files only when the source must be revised.

Notes 3. Refer to the contents in [Table8-1](#).

Notes 4. * = N (N : when the standard libraries are not used)

Notes 5. Refer to the contents in [Table8-2](#).

2.6.4 Library files

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

Table2-4 lists the folder contents.

Table2-4 Library Files

Folder Name	File Name	File Role
lib78k0s\	cl0s.lib cl0sr.lib cl0ss.lib cl0sf.lib cl0sx.lib ^{Note 3} cl0sxr.lib ^{Note 3} cl0sxs.lib ^{Note 3}	Library (runtime and standard libraries) ^{Note 1}
	s0s.rel s0sl.rel s0ss.rel s0ssl.rel	Object files for startup routines ^{Note 2}

Notes 1. The rule for naming libraries is given below.

```
lib78k0s\cl0s< mul >< float >< pascal >< model >.lib
```

< mul >

None Multiplier not used

x Multiplier used

< float >

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

f For floating point library

< pascal >

None When normal function interface is used

r When pascal function interface is used (when compile option -zr is specified)

< model >

None Normal model

s Static model

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

```
lib78k0s\s0s< model >< lib >.rel
```

< model >

None Normal model

s Static model

< lib >

None When standard library functions are not used

l When standard library functions are used

Notes 3. The CC78K0S 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 CC78K0S C Compiler Language User's Manual in regards to library functions and interrupt disable times.

< Special function register >

Function	Reserved words	Addresses	Size
16-bit multiplication result store register 0	MUL0	FF10H/FF12H/ FF14H	16 bit
Multiplication data register A0	MRA0	FFD0H	8 bit
Multiplication data register B0	MRB0	FFD1H	8 bit
Multiplier control register 0	MULC0	FFD2H	8 bit

Use repcmul.bat, a batch file for updating library functions, when using a device in which the 16-bit multiplication result store register (MUL0) is allocated to other than address FF10H. (For details about the batch files, CC78K0S C Compiler Language User's Manual).

In the CC78K0S, only the runtime libraries @@cumul (multiplication of unsigned char types) and @@csmul (multiplication of signed char types) are included as the libraries that support the multiplier. The runtime library for multiplication between int and long types (with multiplier used) is not included because although such multiplication can reduce the number of execution clocks, it increases the code size.

Using batch files repimul.bat and replmul.bat for updating library functions, add runtime libraries that support the multiplier as necessary, referring to the code sizes and the numbers of execution clocks listed below. (For details about the batch files, CC78K0S C Compiler Language User's Manual).

[@@csmul/@@cumul]

Model	Library Not Supporting Multiplier	Library Supporting Multiplier
Normal	28 bytes, 236 to 380 clocks	21 bytes, 60 clocks
Static	28 bytes, 236 to 380 clocks	21 bytes, 60 clocks

[@@ismul/@@iumul]

Model	Library Not Supporting Multiplier	Library Supporting Multiplier
Normal	47 bytes, 874 to 1290 clocks	86 bytes, 236 clocks
Static	51 bytes, 892 to 1308 clocks	85 bytes, 236 clocks

[@@ismul/@@lumul]

Model	Library Not Supporting Multiplier	Library Supporting Multiplier
Normal	96 bytes, 3034 to 4634 clocks	320 bytes, 818 clocks

CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING

This chapter uses the CC78K0S and the RA78K0S 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 CC78K0S is started in PM+ included in the RA78K0S Assembler Package. If the CC78K0S is started from PM+, cc78k0sp.dll included in CC78K0S is referenced.

3.1.1 Position of cc78k0sp.dll (tools DLL)

The tools DLL file, such as the cc78k0sp.dll file, is needed to run the Windows version of the 78K0S Series C compiler (CC78K0S) from PM+.

3.1.2 Execution environment

This environment conforms to PM+.

3.1.3 CC78K0S option setting menu

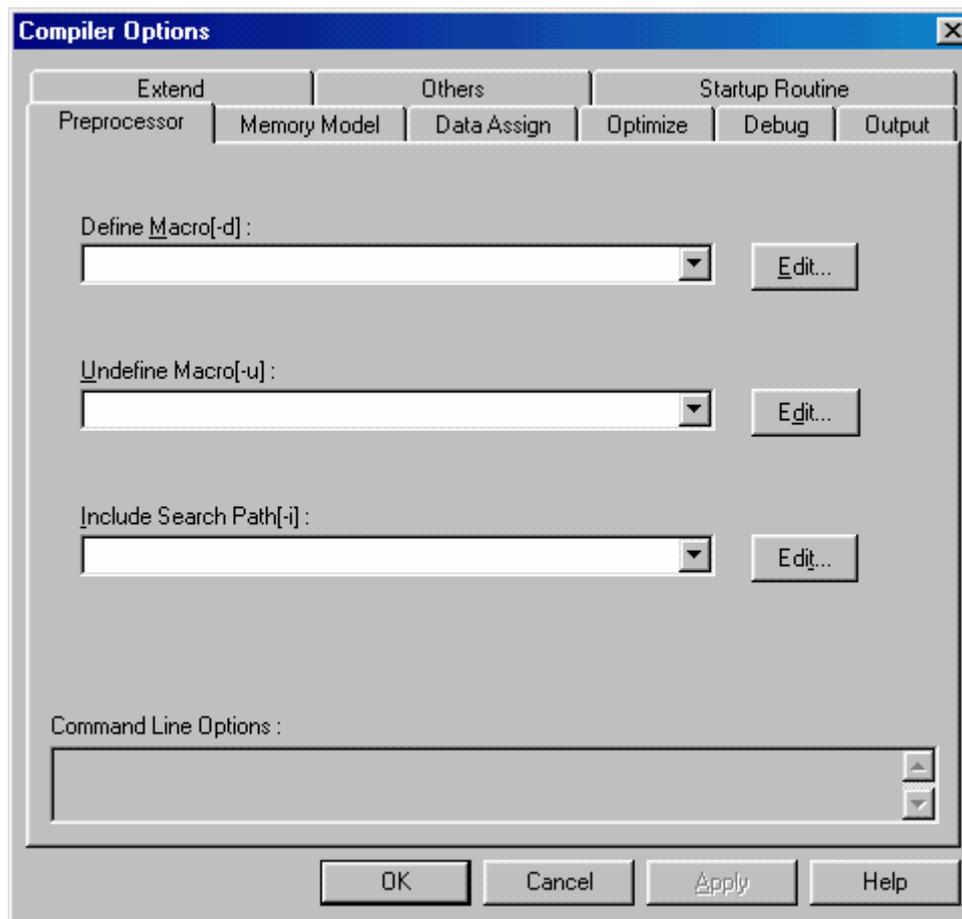
(1) Option menu items

The item [Compiler Options] is added to the [Tools] menu in PM+ by the tools DLL file included in the CC78K0S 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. The < Compiler Options > dialog box is shown below.

Figure3-1 < Compiler Options > Dialog Box



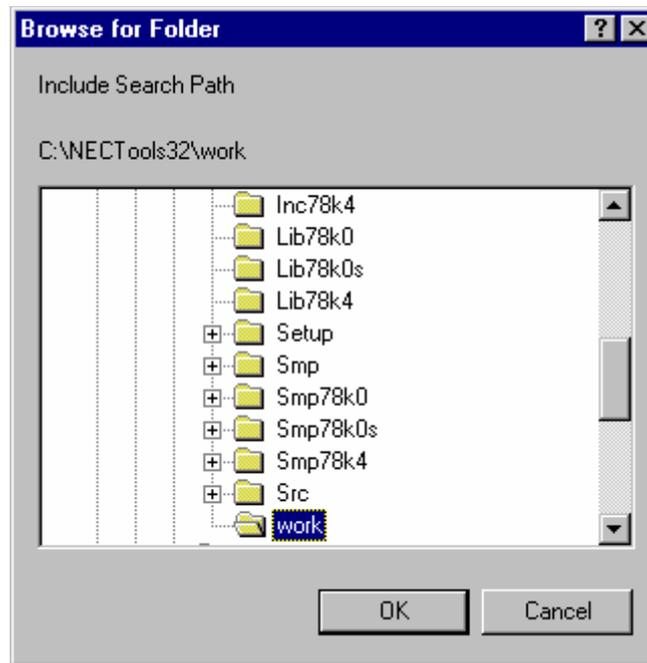
(3) Browse for Folder dialog box

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
- Assembler module file output path
- Error list file output path
- Cross-reference list file output path
- Preprocessor list file output path

- Temporary file path

Figure3-2 < Browse for Folder > Dialog Box



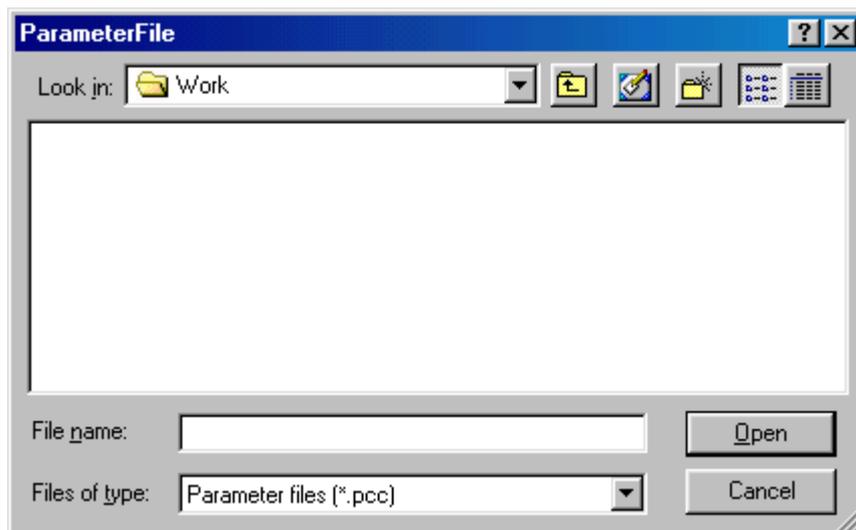
When the [Browse] button is clicked in the parameter file specification, the following dialog box appears.

This dialog box is as follows.

This dialog box displays the following.

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

Figure3-3 < ParameterFile > Dialog Box



(4) < Edit Option > dialog box

Items are edited in list format in the < Edit Option > dialog box.

The < Edit Option > dialog box is described below.

Figure3-4 < Edit Option > Dialog Box

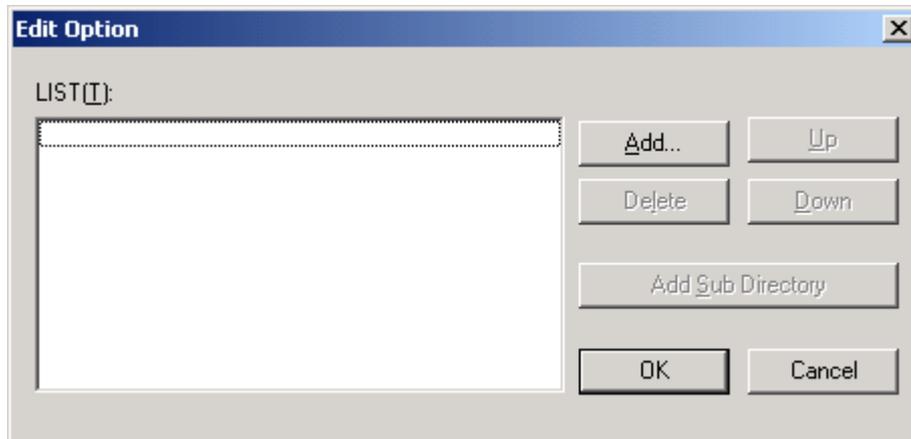
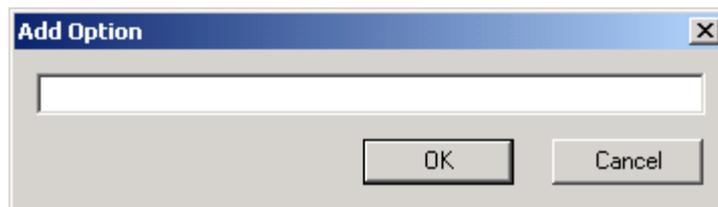


Figure3-5 < Add Option > Dialog Box

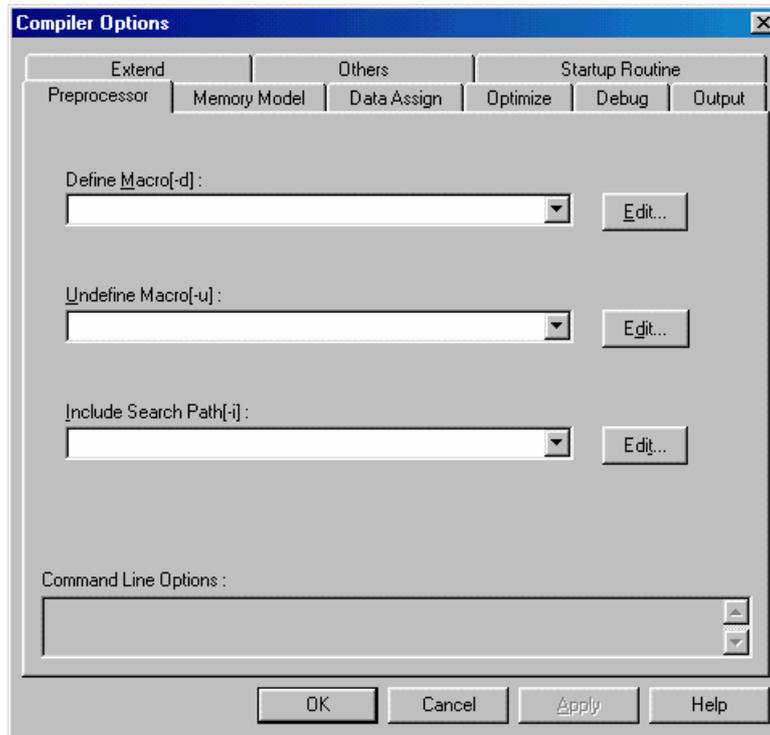


- [Add] button
Adds a list item.
If the item to be added is a file or folder, the corresponding < Browse for Folder > dialog box opens.
In all other cases, the < Add Option > dialog box opens. Specify details of the item to be added in this box.
- [Delete] button
Deletes the selected list item.
- [Up] 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) on the << Others >> Tab.

3.1.4 Description of each part of < Compiler Options > dialog box

Each part of the < Compiler Options > dialog box is described.

Figure3-6 < 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 in [Other Options] of < Others > dialog box is reflected and displayed in real time.

Nothing can be input in this display area. Even though the default option of the CC78K0S is the "specified" state (i.e., a check box is checked, 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 [ScrollBar].

- [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 this 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.

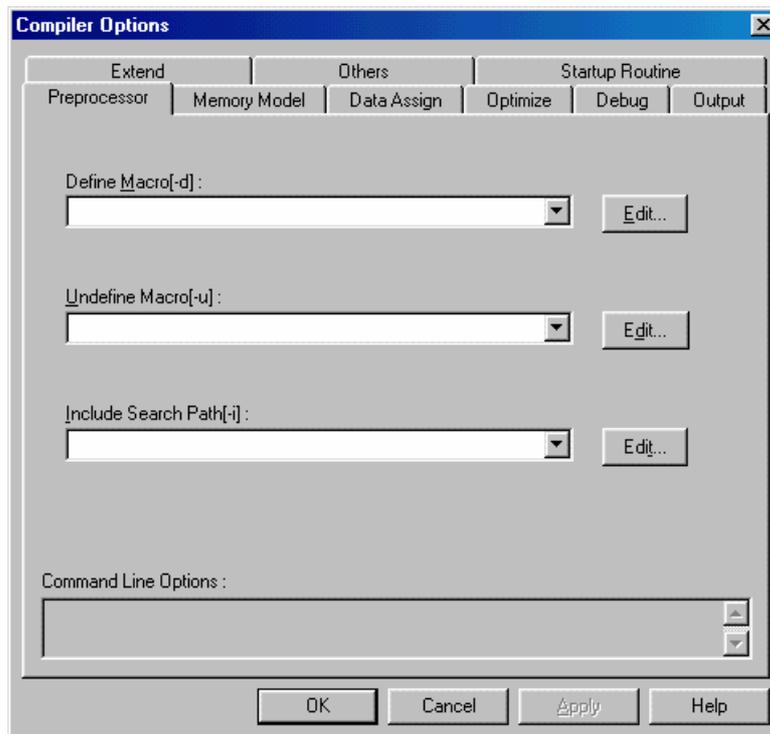
The edited contents in this dialog box are applied and the < Compiler Options > dialog box remains displayed.

- [Help] button

The help file for this dialog box opens.

(1) << Preprocessor >> tab

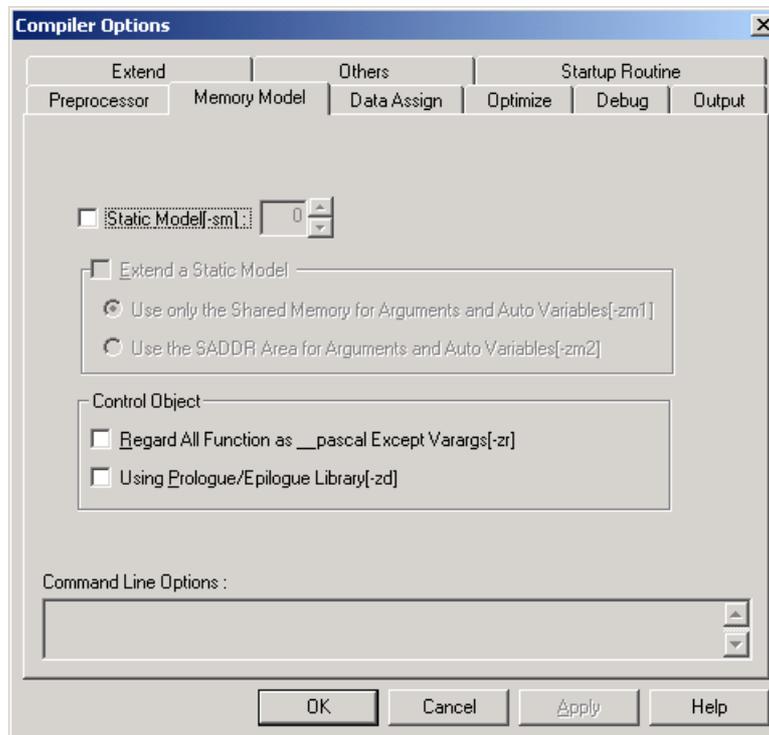
Figure3-7 < Compiler Options > Dialog Box (When << Preprocessor >> Tab Is Selected)



- **Define Macro[-d]**
The macro name and definition name specified by the -d option is input to the combo box.
For the macro name, multiple macro definitions can be performed at once by delimiting with ",".
Can be specified by using the [Edit] button.
- **Undefine Macro[-u]**
The macro name specified by the -u option is input to the combo box.
For the macro name, multiple macro definitions can be invalidated at once by delimiting with ",".
Can be specified by using the [Edit] button.
- **Include Search Path[-i]**
The folder that contains include files specified by the -i option is input to the combo box.
Multiple directories can be specified at once by delimiting with ",".
Can be specified by using the [Edit] button.
Unexisted path cannot be specified.

(2) << Memory Model >> tab

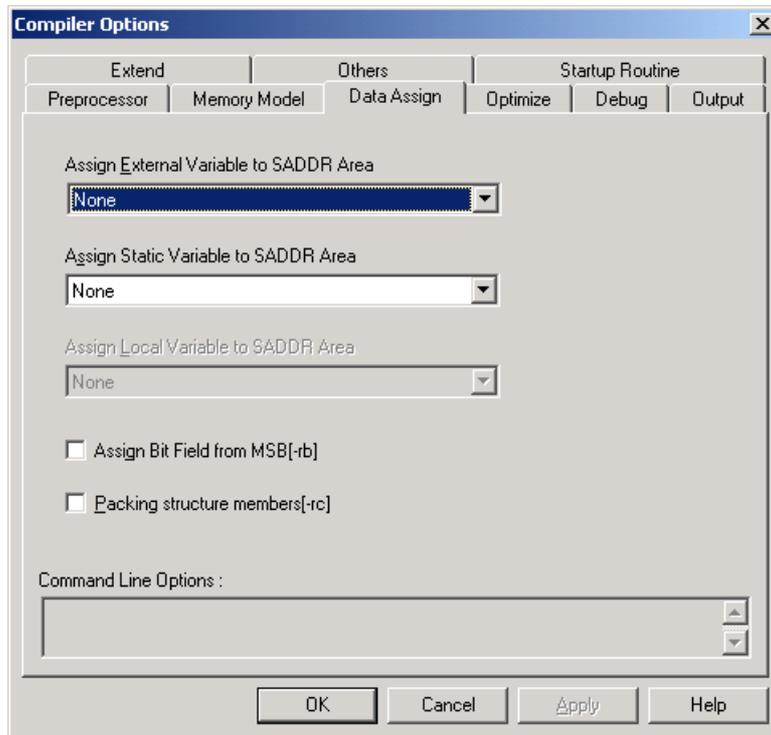
Figure3-8 < Compiler Options > Dialog Box (When << Memory Model >> Tab Is Selected)



- **Static Model[-sm]**
Use a static model by checking the check box and specify a number of bytes of the common area.
- **Extend a Static Model**
If the -sm option is specified and you wish to extend a static model, select this check box.
Select the area to be used for arguments and auto variables by clicking the appropriate radio button.
The information of the selected radio button is saved even if the check box is left unchecked.
- **Control Object**
Regard All Function as __pascal Except Varargs[-zr]
Select this check box to enable the -zr option.
Using Prologue/Epilogue Library[-zd]
Select this check box to enable the -zd option.

(3) << Data Assign >> tab

Figure3-9 < Compiler Options > Dialog Box (When << Data Assign >> Tab Is Selected)

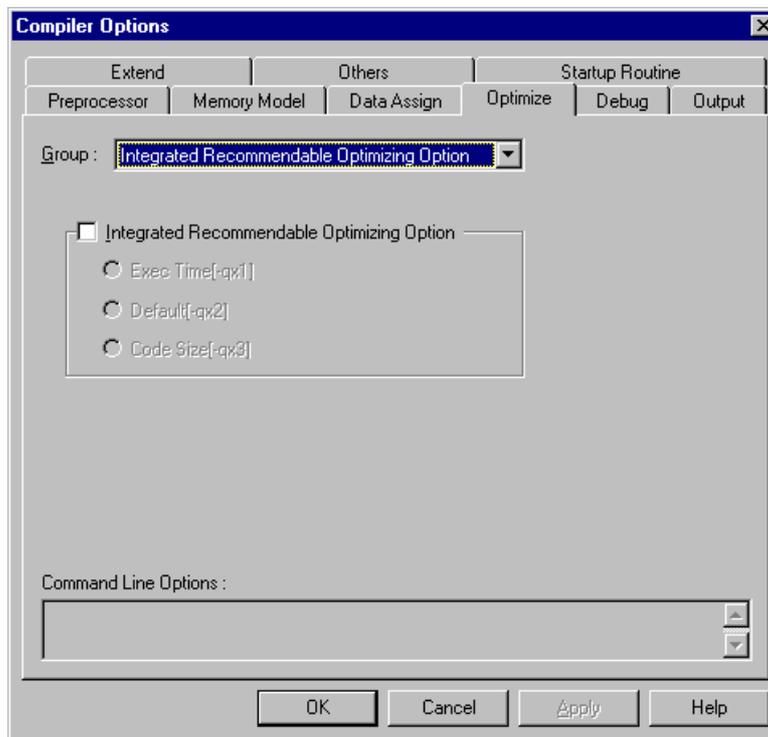


- Assign External Variable to SADDR Area
The type of an external variable to be assigned to the saddr area is selected in the drop-down list box.
- Assign Static Variable to SADDR Area
The type of a static variable to be assigned to the saddr area is selected in the drop-down list box.
- Assign Local Variable to SADDR Area
The type of an automatic variable to be assigned to the saddr area is selected in the drop-down list box.
- Assign Bit Field from MSB[-rb]
Check the check box to validate the -rb option.
- Packing structure members[-rc]
Check the check box to validate the -rc option.

(4) << Optimize >> tab

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

Figure3-10 < Compiler Options > Dialog Box (When << Integrated Recommendable Optimizing Option >> Is Selected)



- 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.

There are 3 settings : "Exec Time [-qx1]", "Default [-qx2]", and "Code Size [-qx3]". Their meanings are as follows.

Exec Time[-qx1]

-qx1 option. Select this option when the efficiency of executing speed is important.

Default[-qx2]

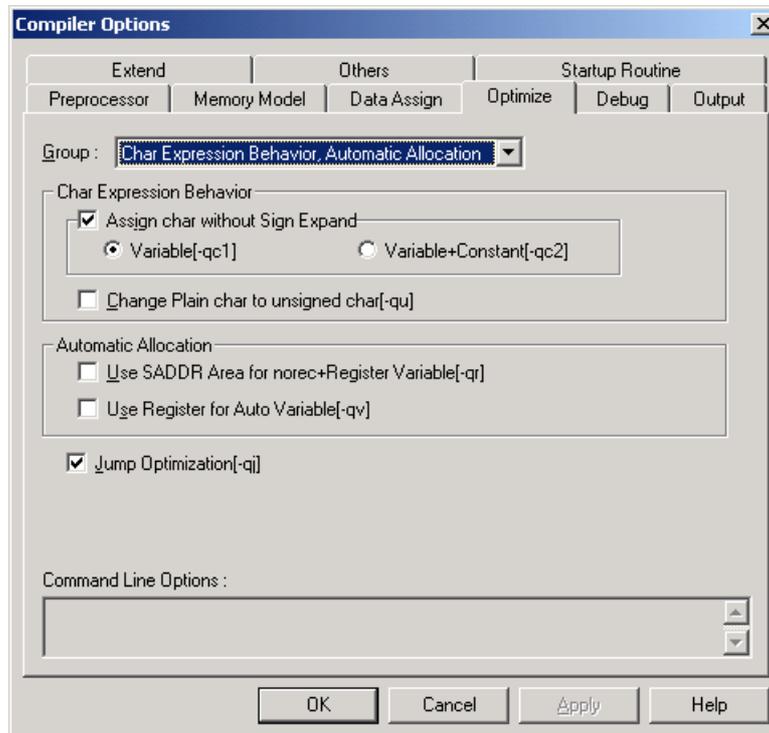
-qx2 option. Select this option when both the efficiency of executing speed and the efficiency of object code size are equally important.

Code Size[-qx3]

-qx3 option. Select this option when the efficiency of object code size is important.

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

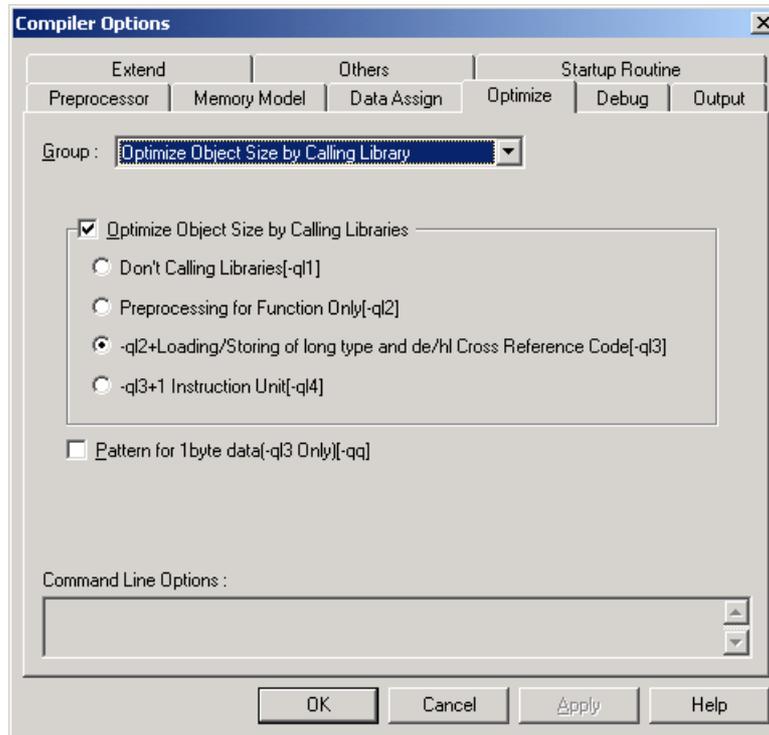
Figure3-11 < Compiler Options > Dialog Box (When << Char Expression Behavior, Automatic Allocation >> Is Selected)



- Char Expression Behavior
 - Assign char without Sign Expand
 - Check this check box to validate the -qc option (do not execute integrate promotion).
 - Select the type of non sign-expanded char operation by checking a radio button.
 - Change Plain char to unsigned char[-qu]
 - Check this check box to validate the -qu option.
- Automatic Allocation
 - Use SADDR Area for norec + Register Variable[-qr]
 - Check this check box to validate the -qr option and select a variable to be assigned by checking a radio button.
 - Use Register for Auto Variable[-qv]
 - Check this check box to validate the -qv option.
- Jump Optimization[-qj]
 - Check this check box to validate the -qj option.

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

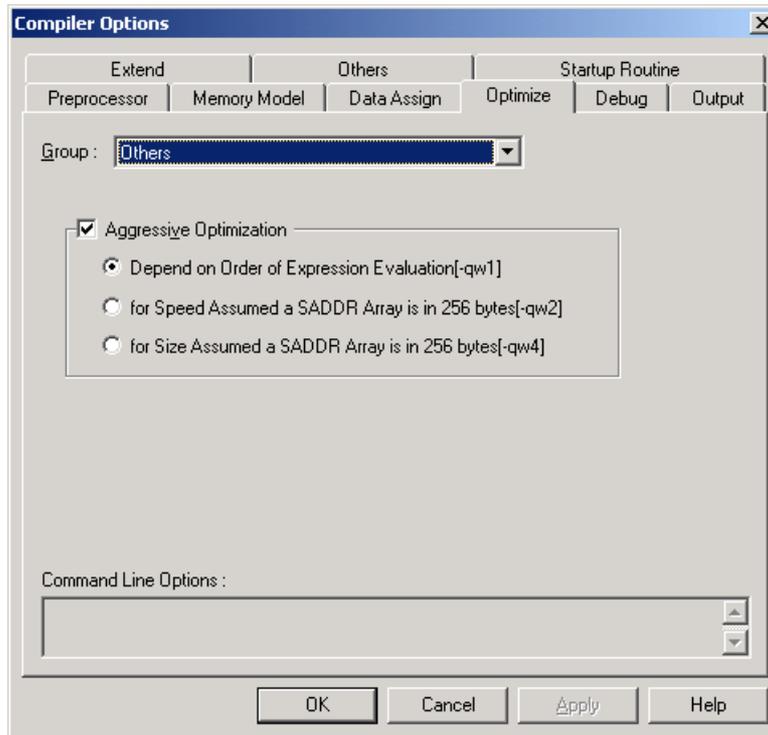
Figure3-12 < Compiler Options > Dialog Box (When << Optimize Object Size by Calling Library >> Is Selected)



- **Optimize Object Size by Calling Libraries**
Check this check box to validate the -ql option and specify the level of the object size priority optimization by checking a radio button. When the number n of -qln becomes greater, the object code size becomes smaller, and accordingly the executing speed becomes slower.
- **Pattern for 1byte data(-q3 Only)[-qq]**
Select this check box to enable the -qq option.

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

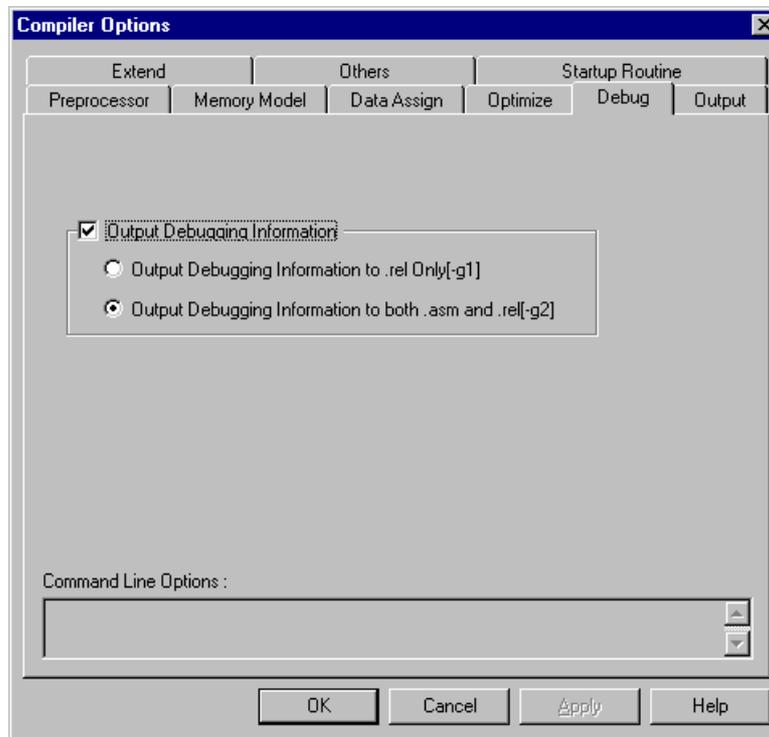
Figure3-13 < Compiler Options > Dialog Box (When << Others >> Is Selected)



- **Aggressive Optimization**
Check this check box to validate the -qw option.

(5) << Debug >> tab

Figure3-14 < Compiler Options > Dialog Box (When << Debug >> Tab Is Selected)



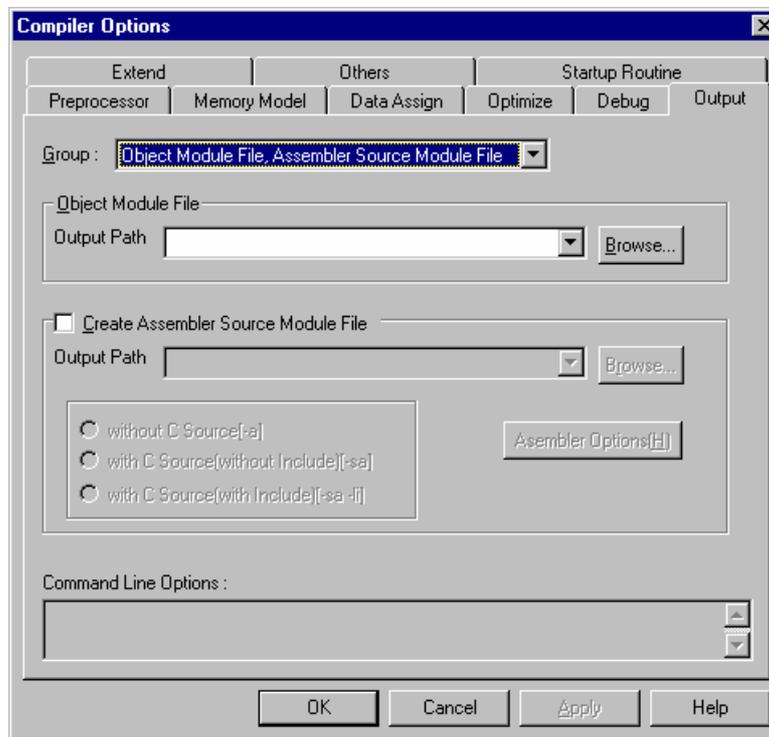
- Output Debugging Information

Check this check box to validate the -g option and select a file that should output debug information by checking a radio button. If [Debug] is disabled by a PM+ option, it is not possible to perform settings in the < Debug > dialog box, and debug information is not output.

(6) << Output >> tab

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

Figure3-15 < 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. Specification is also possible using the [Browse] button.

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**

To enable the -a/-sa/-li options, select this check box, and select with/without C source to attach to the assembler source module file and with/without include file contents by clicking the appropriate radio button.

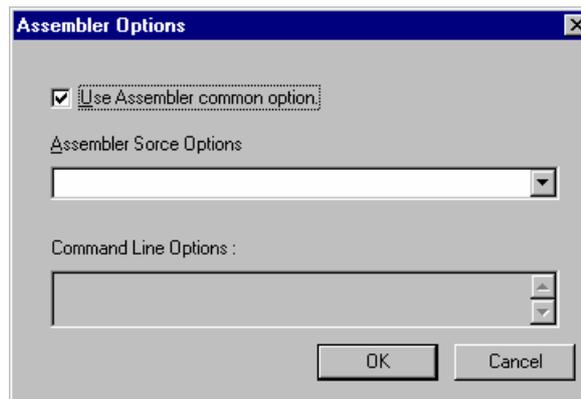
To specify the output path of the assembler source module file, input the path name in the combo box. Specification is also possible using the [Browse] button.

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.

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

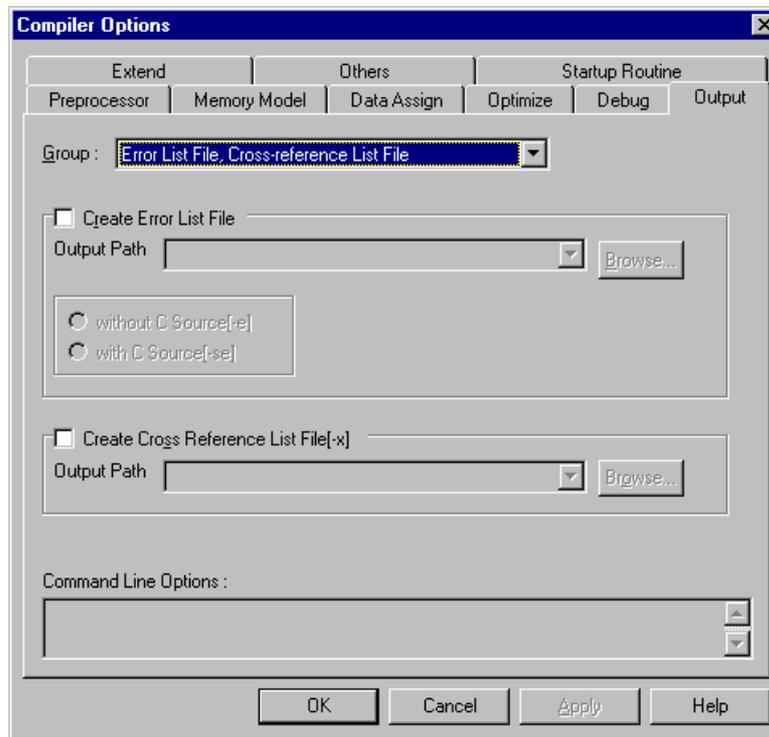
Figure3-16 < Assembler Options > Dialog Box



- Use Assembler common option
Select this check box to enable all the options set in the < Assembler Options > dialog box.
- Assembler Source Options
To enable options for the output assembler source file of the compiler, input a character string including the option name in the combo box.
Past inputs can be selected by clicking the [DropDownList] at the right of the 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.
If the character strings do not all fit in the box, they can be viewed by scrolling with the [ScrollBar].
All the character strings specified by setting a button or inputting in a box are immediately displayed in this edit box.
Assembler common options and output assembler options are displayed as the option character strings.

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

Figure3-17 < Compiler Options > Dialog Box (When << Error List File, Cross-reference List File >> Is Selected)



- **Create Error List File**

Select this check box to enable the `-e/-se` option. Also select whether or not to attach the C source to the error list by selecting the appropriate radio button.

To specify the error list file output path, input the path name in the combo box. Specification is also possible using the **[Browse]** button.

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]**

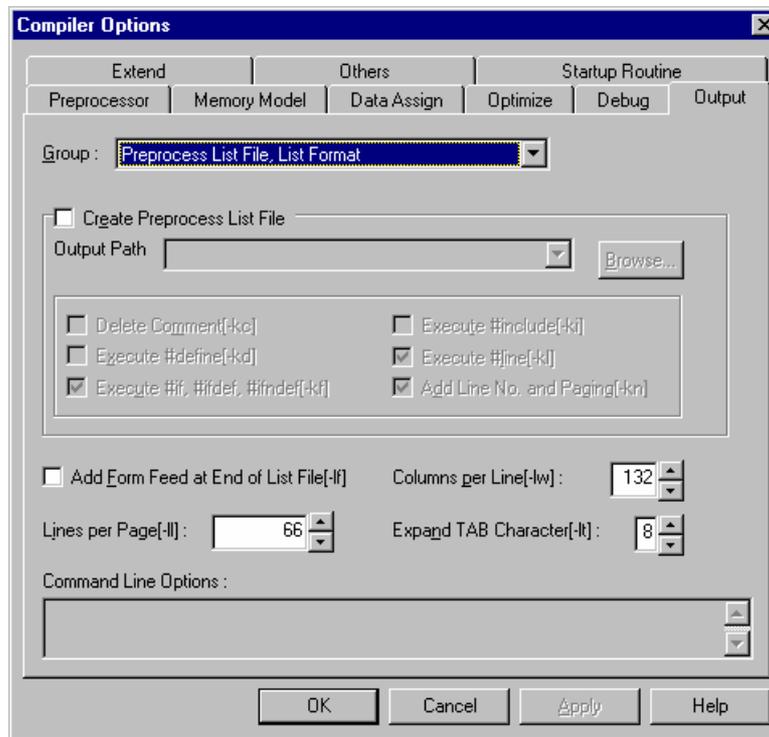
Select this check box to enable the `-x` option. To specify the cross-reference list file output path, input the path name in the combo box. Specification is also possible using the **[Browse]** button.

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.

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

Figure3-18 < Compiler Options > Dialog Box (When << Preprocess List File, List Format >> Is Selected)



- **Create Preprocess List File**

Check this check box to validate the -p option and the specification for the following preprocess list files.

Delete Comment[-kc]

Check this check box to validate the -kc option.

Execute #define[-kd]

Check this check box to validate the -kd option.

Execute #if, #ifdef, #ifndef[-kf]

Check this check box to validate the -kf option.

Execute #include[-ki]

Check this check box to validate the -ki option.

Execute #line[-kl]

Check this check box to validate the -kl option.

Add Line No. and Paging[-kn]

Check this check box to validate the -kn option.

To specify the preprocess list file output path, input the path name in the combo box. Specification is also possible using the [Browse] button.

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]

Check this check box to validate the -lf option.

- Columns per Line[-lw]

Specifies the number of characters in 1 line by using the -lw option. To increase/decrease the number of characters in the box, click [UpDown] button.

- Lines per Page[-ll]

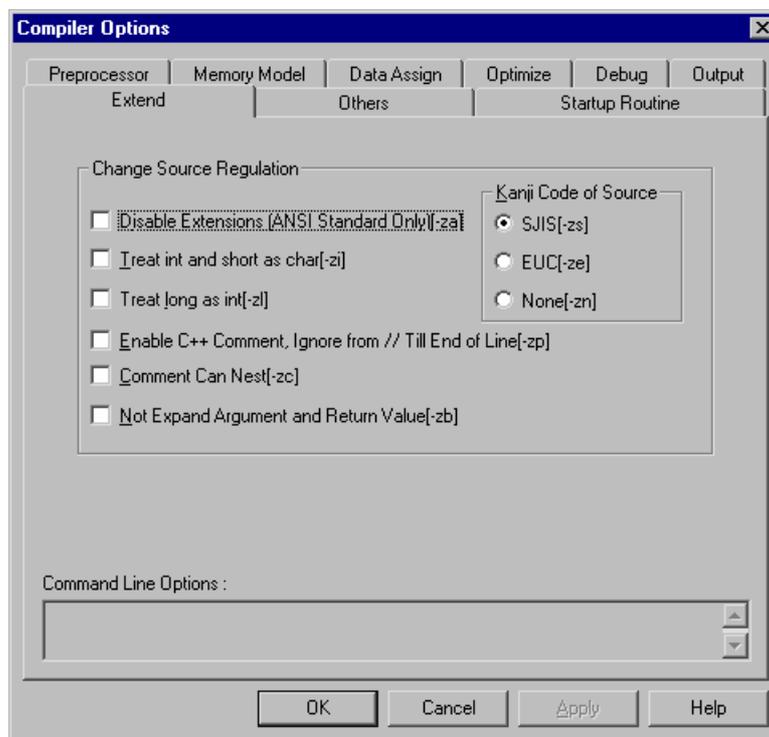
Specifies the number of lines in 1 page by using the -ll option. To increase/decrease the number of characters in the box, click [UpDown] button.

- Expand TAB Character[-lt]

Specifies the length of tab character by using the -lt option. To increase/decrease the number of characters in the box, click [UpDown] button.

(7) << Extend >> tab

Figure3-19 < Compiler Options > Dialog Box (When << Extend >> Tab Is Selected)



- Change Source Regulation

Disable Extensions (ANSI Standard Only)[-za]

Check this check box to validate the -za option.

Treat int and short as char[-zi]

Check this check box to validate the -zi option.

Treat long as int[-zl]

Check this check box to validate the -zl option.

This option is default setting in a static model.

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

Check this check box to validate the -zp option.

Comment Can Nest[-zc]

Check this check box to validate the -zc option.

Not Expand Argument and Return Value[-zb]

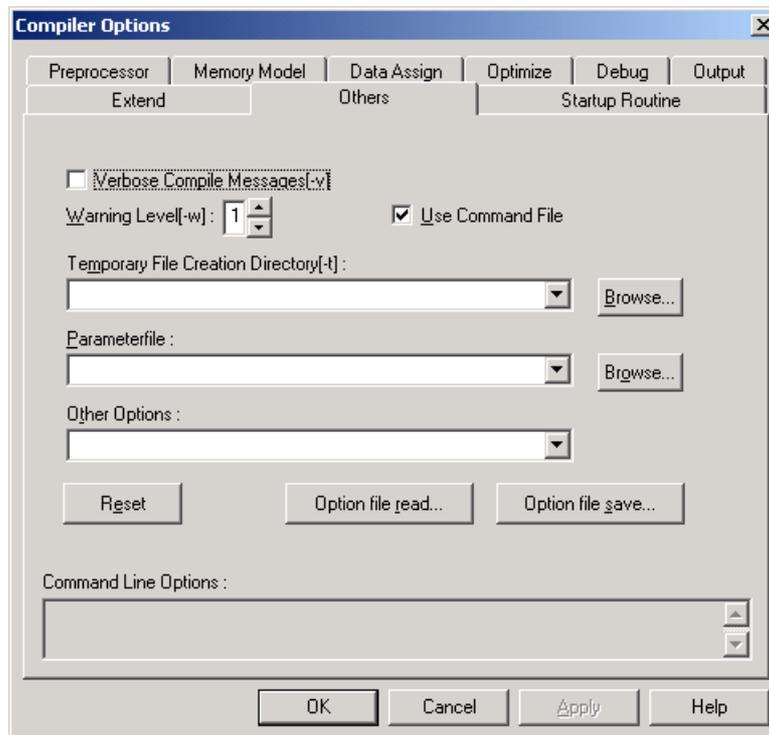
Check this check box to validate the -zb option.

Kanji Code of Source

Select the type (SJIS/EUC/None) of Kanji code used in the comment of the source by selecting the appropriate radio button.

(8) << Others >> tab

Figure3-20 < Compiler Options > Dialog Box (When << Others >> Tab Is Selected)



- **V**erbose Compile Messages[-v]
Select this check box to enable the -v option.
- **W**arning Level[-w]
Use the [UpDown] button to change the -w option level.
- **U**se Command File
By selecting this check box, 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. This check box is selected by default.
- **T**emporary File Creation Directory[-t]
Input the directory in which to store the temporary files specified with the -t option in the combo box.
- **P**arameterfile
Input the parameter file name specified with the -f option in the combo box.
Past inputs can be selected by clicking the [DropDownList] at the right of the combo box.
- **O**ther Options
If a compiler option other than the various option specification items must be specified, input that option in the combo box.
Past inputs can be selected by clicking the [DropDownList] button at the right of the 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.

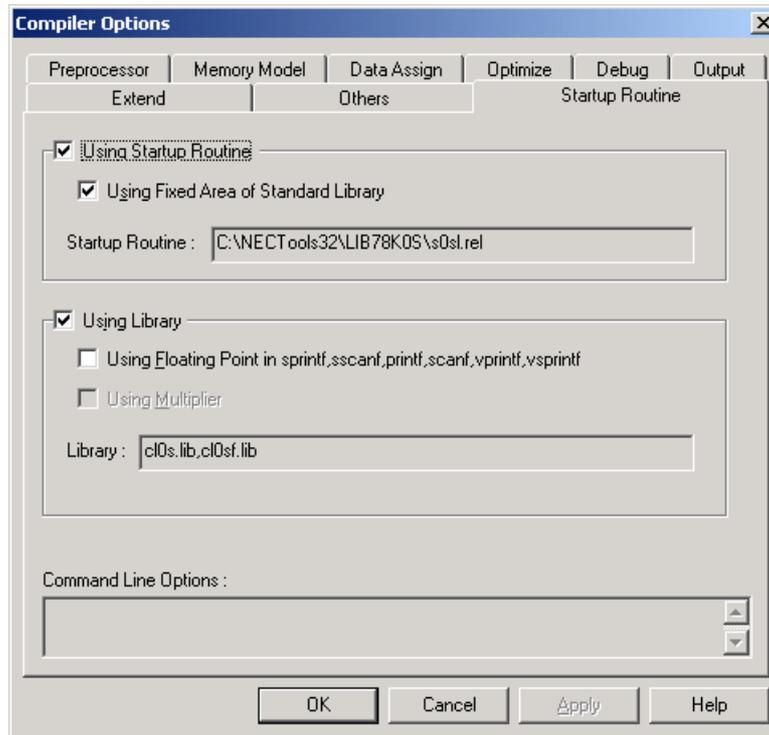
- [Option file save] button

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

Option settings are saved as an option information file.

(9) << Startup Routine >> tab

Figure3-21 < Compiler Options > Dialog Box (When << Startup Routine >> Tab Is Selected)



< Startup Routine > dialog box settings cannot be performed when a source is specified.

- Using Startup Routine

Select this check box to use the standard startup routine provided for this C compiler.

Using Fixed Area of Standard Library

Select this check box to use the fixed area used by the standard library.

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.

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.

If the `[Static Model[-sm]]`, or `[Regard All Function as __pascal Except Varargs[-zr]]` option is specified, the `sprintf`, `sscanf`, `printf`, `scanf`, `vprintf`, and `vsprintf` functions supporting floating points cannot be used.

Using Multiplier

Select this check box when using the multiplier of a product that have multiplier.

Remark 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

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 [NECTools32] menu is created in the Programs folder displayed from the [Start] button, and PM+ and other programs are registered in this menu. Click [PM plus] from the menu to start up PM+.

(2) Creating project

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

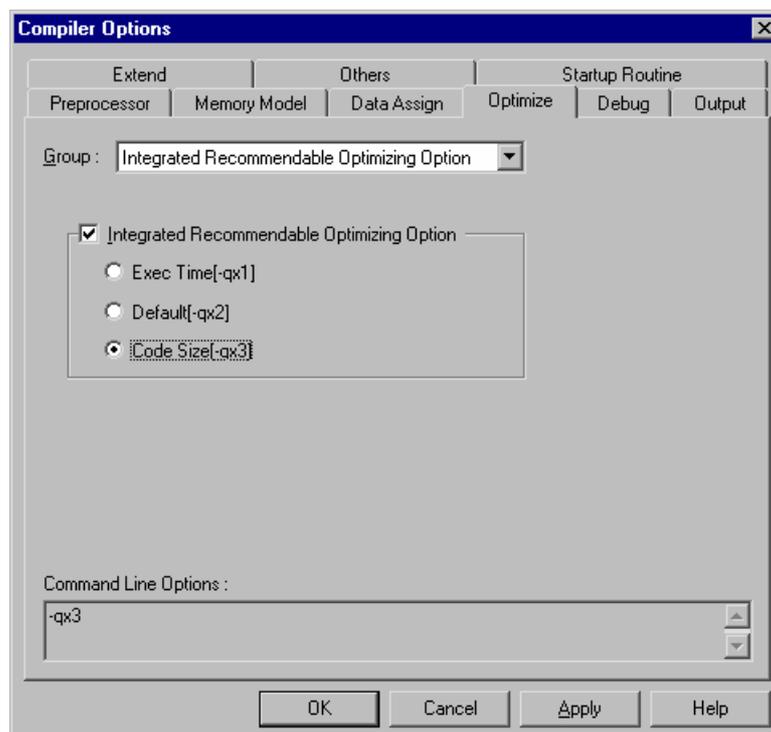
To register a project, first create the workspace in which that project is managed. For the procedure to create a workspace, refer to the PM+ User's Manual.

(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 Code Size[-qx3] is shown below.

Figure3-22 < Compiler Options > Dialog Box (When << Optimize >> Tab Is Selected)



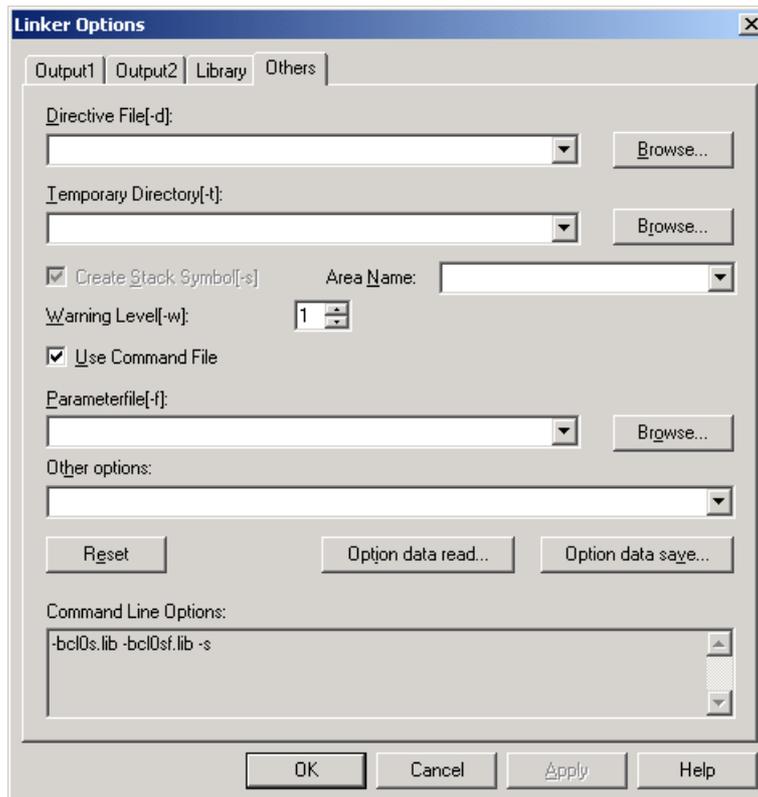
If "Using Startup Routine" is selected in the << Startup Routine >> tab of the < Compiler Options > dialog box, the standard startup routine for this compiler gets linked before all sources (not displayed to the < Linker Options > dialog box).

When "Using Library" is selected, the standard library for this 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.

Figure3-23 < Linker Options > Dialog Box



(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.

Remark 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 DOS prompt)

(1) When parameter file is not used

The command below is used to start the CC78K0S, 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 ] cc78k0s [ Δ option ] Δ C source name [ Δ option ]
>[ path name ] ra78k0s [ Δ option ] Δ assembler source name [ Δ option ]
>[ path name ] lk78k0s [ Δ option ] Δ object module name [ Δ option ]
```

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

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

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

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

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

(Library specification order)

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

- (i) User program library file (specified with `-b` option)
- (ii) Library file attached to C compiler (specified with `-b` option)
- (iii) Floating point library file attached to C compiler (specified with `-b` option)

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

- (i) User program library file (specified with `-b` option)
- (ii) Floating point library file attached to C compiler (specified with `-b` option)
- (iii) Library file attached to C compiler (specified with `-b` option)

(Specification order of other files)

- (i) Object file of startup routine attached to CC78K0S
- (ii) Object module file of user program

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

```
C>cc78k0s -c9024 s1.c -e -a -iC:\NECTools32\inc78k0s -yC:\NECTools32\dev -sm16
C>ra78k0s -c9024 s2.asm -e -yC:\NECTools32\dev
C>lk78k0s s0ss1.rel s1.rel s2.rel -bC:\NECTools32\lib78k0s\cl0ss.lib -s -osample.lmf -yC:\NECTools32\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 RA78K0S Assembler Package Operation User's Manual.

(2) When parameter file is used

When multiple options are input in starting a 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 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 ] cc78k0s Δ -f parameter file name
>[ path name ] ra78k0s Δ -f parameter file name
>[ path name ] lk78k0s Δ -f parameter file name
```

The following shows a usage example.

```
C>cc78k0s -fpara.pcc
C>ra78k0s -fpara.pra
C>lk78k0s -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 >

```
-c9024 s1.c -e -a -iC:\NECTools32\inc78k0s -yC:\NECTools32\dev -sm16
```

< Contents of para.pra >

```
-c9024 s2.asm -e -yC:\NECTools32\dev
```

< Contents of para.plk >

```
s0ss1.rel s1.rel s2.rel -bC:\NECTools32\lib78k0s\cl0ss.lib -s -osample.lmf
-yC:\NECTools32\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 RA78K0S Assembler Package Operation User's Manual.

3.3 I/O Files of C Compiler

The CC78K0S 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 CC78K0S I/O files are shown below.

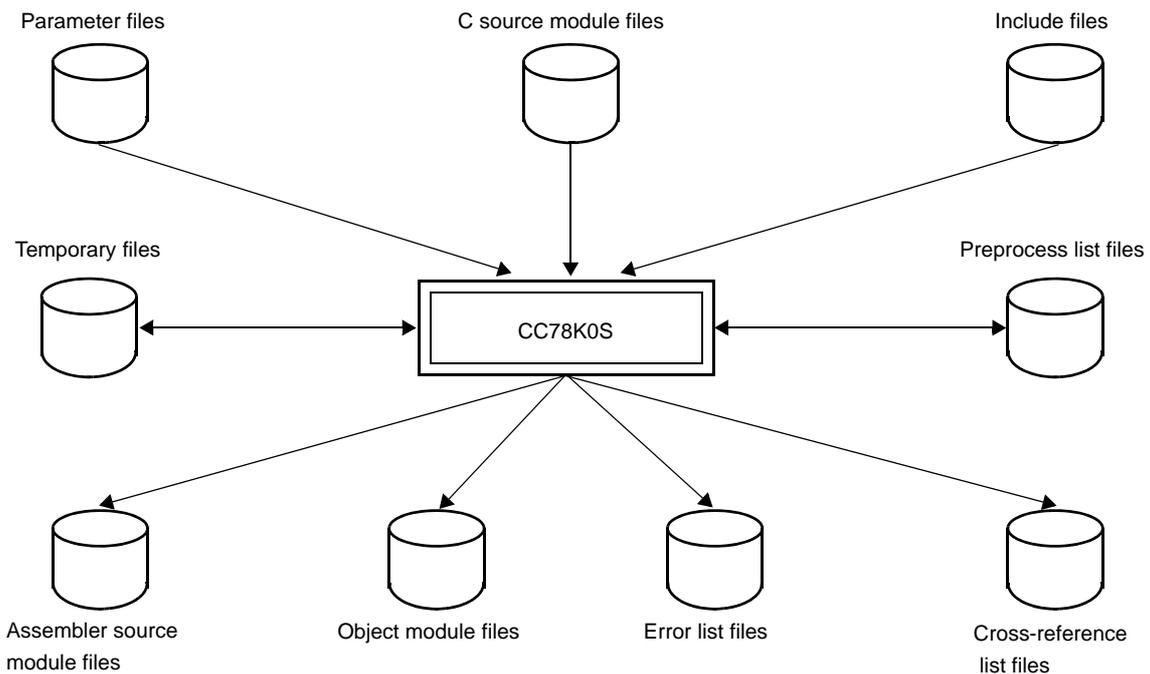
Table3-1 C Compiler I/O Files

Type	File Name	Description	Default File Type
Input Files	C source module file	<ul style="list-style-type: none"> - Source file written in the C language - File created by the user 	c
	Include file	<ul style="list-style-type: none"> - File referenced by a C source module file - File written in the C language - File created by the user 	h
	Parameter file	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - ASCII image file of the object code output by the compiler 	asm
	Preprocess list file	<ul style="list-style-type: none"> - List file output by the preprocess instructions such as #include - ASCII image file 	ppl
	Cross-reference list file	<ul style="list-style-type: none"> - List file containing the function name and variable name information used in the C source module file 	xrf
	Error list file	<ul style="list-style-type: none"> - List file containing the source file and compiler error messages 	ecc cer her er ^{Note}
I/O File	Temporary file	<ul style="list-style-type: none"> - 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.

- 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)

Figure3-24 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.

3.4 Execution Start and End Messages

3.4.1 Execution start message

When the CC78K0S starts, the execution start message is displayed on the console.

```
78K/0S Series C Compiler Vx.xx [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx
```

3.4.2 Execution end message

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

```
Target chip : uPD789xxx
Device file : Vx.xx

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

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

```
PRIME.C(18) : CC78K0S warning W0745 : Expected function prototype
PRIME.C(20) : CC78K0S warning W0745 : Expected function prototype
PRIME.C(26) : CC78K0S warning W0622 : No return value
PRIME.C(37) : CC78K0S warning W0622 : No return value
PRIME.C(44) : CC78K0S warning W0622 : No return value

Target chip : uPD789xxx
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.

```
C>cc78k0s -c9024 -e prime.c -m

78K/0S Series C Compiler Vx.xx [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

CC78K0S error F0018 : Option is not recognized ' -m '
Please enter ' CC78K0S -- ' , if you want help messages.
Program aborted.
:
```

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

If the compiler outputs error messages and stops the compilation, find the sources of these error messages in "[CHAPTER 9 ERROR MESSAGES](#)" and correct.

CHAPTER 4 CC78K0S FUNCTIONS

4.1 Optimization Method

Optimization is performed to create efficient object module files in the CC78K0S. [Table4-1](#) lists the supported optimization methods.

Table4-1 Optimization Methods

Phase	Contents	Example
Syntax		
(1)	Execute during constant computations compilation	<code>a = 3 * 5 ; --> a = 15 ;</code>
(2)	True or false decision based on partial evaluation of a logical expression	<code>0 && (a b) --> 0</code> <code>1 (a && b) --> 1</code>
(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.	<code>a = a + 1 ; --> Use the inc instruction.</code> 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. <code>mov a, #0</code> or <code>xor a, a</code> (differs depending on the device)
(7)	Change long jump instructions to short jump instructions.	The intermediate code that was output is reprocessed.
Optimizer		
(8)	Delete common partial expressions.	<code>a = b + c ; --> a = b + c ;</code> <code>d = b + c + e ; d = a + e ;</code>
(9)	Move outside an instruction loop.	<pre>for (i = 0 ; i < 10 ; i++) { : a = b + c ; : } ↓ a = b + c ; for (i = 0 ; i < 10 ; i++) { : }</pre>
(10)	Delete unused instructions.	<code>a = a ; --> Delete</code> After <code>a = b ;</code> , <code>a</code> is not referenced --> Delete (<code>a</code> is an automatic variable)

Table4-1 Optimization Methods

Phase	Contents	Example
(11)	Delete copies.	<pre>a = b ; c = a + d ; --> c = b + d ; a is not referenced any more (a is an automatic variable).</pre>
(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 <code>a * 1 --> a</code> , <code>a + 0 --> a</code>
(15)	Decrease the strength of the calculation.	Examples <code>a * 2 --> a + a</code> , <code>a << 1</code>
(16)	Memory device allocation (register variables)	Data is allocated to rapidly accessible memory. Examples Registers, <code>saddr</code> (only when <code>-qr</code> is specified)
(17)	Jump optimization (<code>-qj</code> option)	Consecutive jump instructions are combined into 1 instruction.
(18)	Register allocation (<code>-qv</code> , <code>-qr</code> , <code>-rd</code> , <code>-rk</code> , <code>-rs</code> options)	Variables are automatically allocated to registers.

Remark (1) to (7) are performed regardless of the optimization option specifications.

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

Future support is planned for the optimizations in (8) to (13).

(14) and (15) are performed regardless of the optimization option specifications.

(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 CC78K0S 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.

(1) s0s*.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.

(2) cl0s*.lib

Library attached to CC78K0S. The library files of the CC78K0S include the following 2 libraries.

Runtime library

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

"_@" is added to the symbol head.

Standard library

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

(3) *.lib

Library created by a user. "_" is added to the symbol head.

Remark The CC78K0S 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 "[2.6.4 Library files](#)".

CHAPTER 5 COMPILER OPTIONS

When the C compiler is started, the compiler options can be specified. The compiler options provide instructions for compiler 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.

- (1) Specified in the command line when the C compiler starts.
- (2) Specified in the < Compiler Options > dialog box of PM+.
- (3) 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

```
cc78k0s Δ -c9024 Δ prime.c Δ -aprime.asm Δ -qx3
```

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.

Table5-1 Prioritization of Compiler Options

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

[Location marked by NG]

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

[Location marked by Δ]

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

[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 1

```
C>cc78k0s -c9024 -e sample.c -no -rd -g
```

Remark The -rd and -g options become invalid.

Example 2

```
C>cc78k0s -c9024 -e sample.c -p -k
```

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

Example 3

```
C>cc78k0s -c9024 -e sample.c -utest -dtest = 1
```

Remark 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 4

```
C>cc78k0s -c9024 -e sample.c -o -no
```

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

Options not described in [Table5-1](#) 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+, press the [Help] button in each option dialog box of PM+.

5.3 Types

Compiler options are categorized into the following 19 types.

Table5-2 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	
	-rk	Specifies the automatic assignment of a function argument and auto variable (except for the static auto 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.

Table5-2 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 the levels of the warning messages that are output to the console.
Execution state display specification	-v	Specifies the execution status of compilation that 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 processing for function expansion.
	-nz	
Device file search path	-y	Specifies paths that search device files.
Static model specification	-sm	Specifies object static model or normal model.

5.4 Descriptions

This section describes each compiler option in detail.

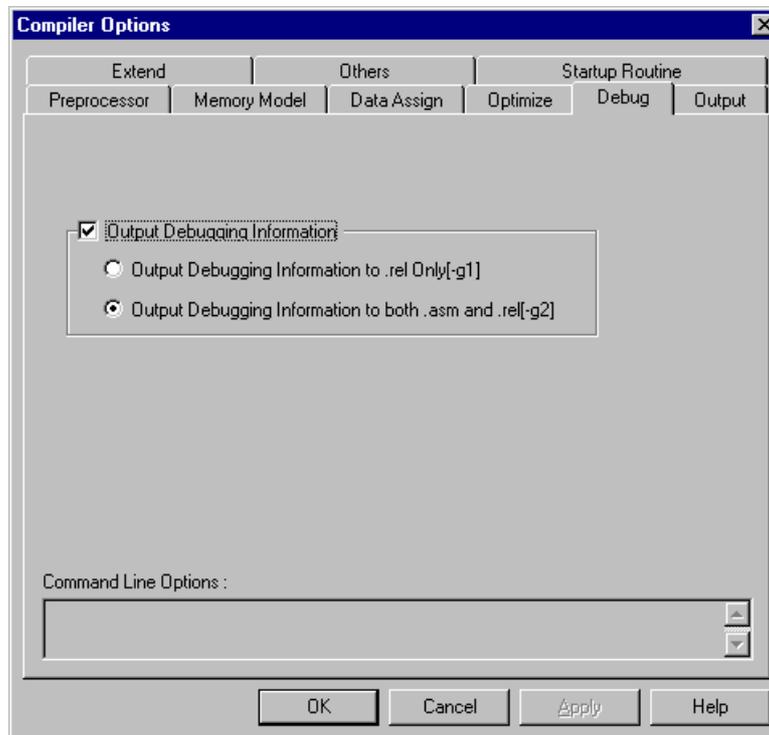
This example illustrates starting the CC78K0S in the command line. To start in PM+, specify the command, device type specification, and options left out of the C source in the < Compiler Options > dialog box.

Example (In command line)

```
C>cc78k0s -c9024 prime.c -g
```

Example (When using PM+)

Figure5-1 < Compiler Options > Dialog Box



(1) Device type specification

Device type specification (-c)

[Description format]

```
-c device-type
```

- Interpretation when omitted
None

[Function]

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

[Application]

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

[Description]

- Refer to the advice about use in the supplemental product materials of the device file for the target devices that can be specified by the -c option and the corresponding device type.
- When CC78K0S 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]

- The specification is made in the command line. The target device is the uPD789024.

```
C>cc78k0s -c9024 prime.c
```

- This specification is made in the C source and the compiler is started.

```
#pragma pc ( 9024 )
#define TRUE 1
#define FALSE 0
#define SIZE 200

char mark [ SIZE + 1 ] ;

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

Therefore, the target device specification can be omitted from the command line.

```
C>cc78k0s prime.c
```

- Different devices are specified in the C source and the command line and the compiler is started.

< C source >

```
#pragma pc ( 9024 )
#define TRUE 1
#define FALSE 0
#define SIZE 200

char mark [ SIZE + 1 ] ;

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

< Command line >

```
C>cc78k0s -c9014 prime.c
```

After the command line is executed, compiling is executed as follows.

```
78K/0S Series C Compiler Vx.xx [ xx xxx xxxx ]
  Copyright ( C ) NEC Electronics Corporation xxxx , xxxx

sample\prime.c ( 1 ) : CC78K0S warning W0832 : Duplicated chip specifier
sample\prime.c ( 18 ) : CC78K0S warning W0745 : Expected function prototype
sample\prime.c ( 20 ) : CC78K0S warning W0745 : Expected function prototype
sample\prime.c ( 26 ) : CC78K0S warning W0622 : No return value
sample\prime.c ( 37 ) : CC78K0S warning W0622 : No return value
sample\prime.c ( 44 ) : CC78K0S warning W0622 : No return value

Target chip : uPD789014
Device file : Vx.xx

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

The target device specification in the command line has priority.

(2) Object module file creation specification

Object module file creation specification (-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 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- Both the -no and -o options are specified (-o has priority) in this example.

```
C>cc78k0s -c9024 prime.c -no -o
```

(3) Memory assignment specification

Memory assignment specification (`-r/-nr`, `-rd/-nr`, `-rk/-nr`, `-rs/-nr`)

(a) `-r/-nr`

[Description formats]

```
-r [ process-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, an Fatal error (F0012) occurs.

Table5-3 Process Types Specifiable by -r Option

Process type	Function
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.
k [n] [m] (n = 1, 2, 4)	In a static model, assigns a function argument and auto variable (except for the static auto variable) automatically to the saddr area, irrespective of whether there is an sreg declaration or not.
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.
c	Does not insert any align data to allocate a 2-byte or more structure member to an even address (i.e., performs packing structure).

Remark Multiple process types can be specified.

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

Table5-4 Interpretation When Specifying NR

Process type	Function
b	Assigns a bit field from the least significant bit (LSB).
d	Does not automatically assign any variable to the saddr area.
k	Does not automatically assign any variable to the saddr area.
s	Does not automatically assign any variable to the saddr area.
c	Does not pack any structure members.

[Use Example]

```
C>cc78k0s -c9024 -rds
```

(b) -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.

Table5-5 Maximum (-rd) Range of Variables

Value of n	Variable types to be assigned
1	char, unsigned char
2	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer
4	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer, long, unsigned long
m	Structure, Union, Array
Omitted	All variables

- The sreg-declared variable is automatically assigned to the saddr area irrespective of -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.

(c) -rk/-nr

[Description formats]

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

- Interpretation when omitted
-nr

[Function]

- The -rk option specifies the automatic assignment of a function argument and auto variable (except for the static auto variable) to the saddr area.
- The -nr option invalidates the -rk option.

[Application]

- With a static model, if you want to automatically assign a function argument and auto variable (except for the static auto variable) to the saddr area irrespective of whether there is an sreg declaration or not, specify the -rk option.

[Description]

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

Table5-6 Maximum (-rk) Range of Variables to be Assigned

Value of n	Variable types to be assigned
1	char, unsigned char
2	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer
4	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer, long, unsigned long
m	Structure, Union, Array
Omitted	All variables

- The register-declared variable is not assigned.
- The sreg-declared variable is automatically assigned to the saddr area irrespective of -rk option specification.
- The function argument and auto variable that are assigned to the saddr area by specifying this option are handled in a similar way to an sreg-declared function argument and sreg-declared auto variable.

[Caution]

- This option is valid only when the -sm option is specified. If the -sm option is not specified, a warning message is output and the -rk option is ignored.

(d) -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.

Table5-7 Maximum (-rs) Range of Variables to be Assigned

Value of n	Variable types to be assigned
1	char, unsigned char
2	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer
4	char, unsigned char, short, unsigned short, int, unsigned int, enum, pointer, long, unsigned long
m	Structure, Union, Array
Omitted	All variables

- The sreg-declared variable is automatically assigned to the saddr area irrespective of -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.

(4) Optimization specification

Optimization specification (-q/-nq)

[Description formats]

```
-q [ optimization-type ] (If multiple types are specified, specify them
consecutively)
-nq
```

- Interpretation when omitted
-qcjlw

[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 [Table5-8](#).

[Description]

- [Table5-8](#) lists the optimization types that can be specified by the -q option.

Table5-8 Optimization Types

Optimization Type	Process Description
No specification	Regarded as the -qcjlw specification.
u	Regards the char with no qualifier as a unsigned char to improve code efficiency
c [n] (n = 1, 2)	By executing char calculations without integral promotion, the code becomes more efficient. Integral promotion indicates an ANSI-C rule that is set so that a calculation for a type smaller than an integer (char, short) is converted to int ^{Note} . The scope changes depending on the value of n as follows. If n is omitted, it is interpreted as n = 1. 1 : Only variables are not integral-promoted 2 : Neither variables nor constants are integral-promoted
q	Switches to the library set specialized for 1-byte data.
r	Adds a register variable to a register and assigns it to the saddr area.
j	Optimize jump instructions.

Table5-8 Optimization Types

Optimization Type	Process Description
<p>x [n] (n = 1-3)</p>	<p>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.</p> <ul style="list-style-type: none"> 1 : Speed precedence. Regarded as the -qcyjw option specification. 2 : Default. Regarded as the -q option specification. 3 : Code size precedence. Regarded as the -qcjl4w option specification.
<p>w [n] (n = 1, 2, 4)</p>	<p>Outputs an efficient code and design for the effective use of the registers by changing the execution order in an expression (i.e., changing the execution order of the right subexpression and the left subexpression in an expression with 2 terms). However, if the option is not included (although within the scope of the standard, since the ANSI-C standard omits some of the operators and does not set the execution order), the result of the execution sometimes differs. According to the ANSI-C standard, this is not a problem in a properly written source. The scope changes depending on the value of n as follows. If n is omitted, it is interpreted as n = 1.</p> <ul style="list-style-type: none"> 1 : Changes the execution order in an expression 2 : Changes the execution order in an expression. Performs address calculation without a carry, giving priority to the speed, when a char, unsigned char, short, unsigned short, int, or unsigned int array that is allocated to the saddr area is referenced with an unsigned char variable, while assuming that the size of the array does not exceed 256 bytes. 3 : Changes the execution order in an expression. Performs address calculation without a carry, giving priority to the code size, when a char, unsigned char, short, unsigned short, int, or unsigned int array that is allocated to the saddr area is referenced with an unsigned char variable, while assuming that the size of the array does not exceed 256 bytes.
<p>v</p>	<p>Assigns an automatic variable automatically to a register or the saddr area</p>
<p>l [n] (n = 1-4)</p>	<p>The constant code pattern is replaced with a library. The scope changes depending on the value of n as follows. If n is omitted, it is interpreted as n = 1.</p> <ul style="list-style-type: none"> 1 : No replacement 2 : Executes the only the processes before/after a function 3 : Executes the processes before/after a function, loads/stores a long-type variable, and DE/HL indirect reference code 4 : Executes the processes before/after a function and 1 instruction

Note When the `-qc` option is specified in the CC78K0S, the types of constants and character constants are handled in the following way.

0 to 127 , 0x00 to 0x7F , 00 to 0177	char type
128 to 255 , 0x80 to 0xFF , 0200 to 0377	unsigned char type
0U to 255U	unsigned char type
' \0 ' to ' \377 '	char type

However, when the `-qu` option is specified, character constants in the range from '\200' to '\377' are handled as unsigned char type constants and have the values from +128 to +255.

The constant added – (minus) is treated as follows.

-0 to 128	char type
From -129	int type

If the result of constant or variable calculation is overflow, cast either the constant or variable to a type capable of representing the calculation result. By casting or specifying the `-ql` option, changing the data type can be avoided. When the `-qc1` option is specified, constant calculation is sign-extended.

- 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 \rightarrow 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.
- If both the `-qr` and the `-sm` are simultaneously specified, a warning message is output and `-qr` is ignored.

[Use Example]

- Optimize so that a char without modifier is regarded as unsigned.

```
C>cc78k0s -c9024 prime.c -qu
```

- If both the `-qc` and `-qr` options are specified as below, the `-qc` option becomes invalid, and the `-qr` option is validated.

```
C>cc78k0s -c9024 prime.c -qc -qr
```

- If you want to validate both the `-qc` and `-qr` options, input the following command.

```
C>cc78k0s -c9024 prime.c -qcr
```

(5) Debugging information output specification

Debugging information output specification(-g/-ng)

[Description formats]

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

- 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.

Table5-9 Operation Differs Depending on Value of n

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]

- The -g option is specified.

```
C>cc78k0s -c9024 prime.c -g
```

(6) Preprocess list file creation specification

Preprocess list file creation specification (-p, -k)

(a) -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 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The preprocess list file sample.ppl is output.

```
C>cc78k0s -c9024 prime.c -psample.ppl
```

(b) -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.

Table5-10 Process Types of -k Option

Process type	Description
Omitted	Same as specifying -fln
c	Delete comments
d	#define expansion
f	Conditional compilations of #if, #ifdef, and #ifndef
i	#include expansion
l	#line processing
n	Same as specifying FLN

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]

- Comments are deleted from the preprocess list prime.ppl, and line number and paging processing are performed.

```
C>cc78k0s -c9024 prime.c -p -kcn
```

prime.ppl is referenced.

```
/*
78K/0S Series C Compiler VX.XX Preprocess List
                                Date : XX XXX XXXX Page : 1

Command       : -c9024 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   : uPD789024
Device file   : Vx.xx
*/
```

(7) Preprocess specification

Preprocess specification (-d, -u, -i)

(a) -d

[Description format]

```
-d macro-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 replace all of the macro names for the specified constants.

[Description]

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

[Use Example]

```
C>cc78k0s -c9024 prime.c -dTEST , TIME = 10
```

(b) -u

[Description format]

```
-u macro-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 ",", multiple 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 compiler 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 same macro name is specified by the -d and -u options. In this example, the TEST macro is disabled.

```
C>cc78k0s -c9024 prime.c -dTEST -uTEST
```

(c) -i

[Description format]

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

- Interpretation when omitted
 - (i) Folder with source file^{Note 1}
 - (ii) Folder specified by environment variable INC78K0S
 - (iii) C:\NECTools32\inc78k0s^{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, multiple directories can be specified at one time.
- Spaces cannot be inserted before and after a comma ",".
- If multiple directories 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 -i option
 - (iii) Folder specified with environment variable INC78K0S
 - (iv) C:\NECTools32\inc78k0s^{Note 2}

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

Notes 2. This is an example of when the CC78K0S is installed to C:\NECTools32.

[Use Example]

- The -i option is specified.

```
C>cc78k0s -c9024 prime.c -ib: , b:\sample
```

(8) Assembler source module file creation specification

Assembler source module file creation specification (-a, -sa)

(a) -a

[Description format]

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

- Interpretation when omitted
No assembler source module file is output.
- Output file
.asm (: alphanumeric symbols)

[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 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The assembler source module file sample.asm is created.

```
C>cc78k0s -c9024 prime.c -asample.asm
```

(b) -sa

[Description format]

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

- Interpretation when omitted
No assembler source module file is output.
- Output file
.asm (: alphanumeric symbols)

[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 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The -sa option is specified.

```
C>cc78k0s -c9024 prime.c -sa
```

prime.asm is referenced.

```

; 78K/0S Series C Compiler Vx.xx Assembler Source
;
; Command      : -c9024 prime.c -sa
; In-file      : prime.c
; Asm-file     : prime.asm
; Para-file    :

$PROCESSOR ( 9024 )
$DEBUG
$NODEBUGA
$KANJI CODE SJIS
$TOL_INF      03FH , 0130H , 02H , 00H

$DGS  FIL_NAM ,      .file , 033H , 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 _mark
PUBLIC _main
PUBLIC _printf
PUBLIC _putchar
:
@@CODE CSEG
_main :
$DGL  1 , 13
    push    hl                ; [ INF ] 1 , 4
    movw   ax , #08H          ; [ INF ] 3 , 6
    callt  [ @_cprep ]        ; [ INF ] 1 , 8
??bf_main :
; line  9 :    int i , prime , k , count ;
; line 10 :
; line 11 :    count = 0 ;
$DGL  0 , 4
    xor     a , a              ; [ INF ] 2 , 4
    mov    [ hl ] , a          ; [ INF ] 1 , 6
    mov    [ hl + 1 ] , a      ; [ INF ] 2 , 6
; line 12 :
; line 13 :    for ( i = 0 ; i <= SIZE ; i++ )
$DGL  0 , 6
    mov    [ hl + 6 ] , a      ; [ INF ] 2 , 6
    mov    [ hl + 7 ] , a      ; [ INF ] 2 , 6
?L0003 :
    mov    a , [ hl + 6 ]      ; [ INF ] 2 , 6
    xch    a , x              ; [ INF ] 1 , 4
    mov    a , [ hl + 7 ]      ; [ INF ] 2 , 6
    xor    a , #080H          ; 128      ; [ INF ] 2 , 4
    cmpw  ax , #080C8H        ; -32568 ; [ INF ] 3 , 6
    bc    $$ + 4              ; [ INF ] 2 , 6
    bnz   $?L0004
    :
    END

```

```
; *** Code Information ***
;
; $FILE H : \um\prime.c
;
; $FUNC main ( 8 )
;     bc = ( void )
;     CODE SIZE = 222 bytes , CLOCK_SIZE = 654 clocks , STACK_SIZE = 14 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 ( 31 )
;     bc = ( pointer s : ax , int i : [ sp + 2 ] )
;     CODE SIZE = 28 bytes , CLOCK_SIZE = 108 clocks , STACK_SIZE = 10 bytes
;
; $FUNC printf ( 41 )
;     bc = ( char c : x )
;     CODE SIZE = 14 bytes , CLOCK_SIZE = 58 clocks , STACK_SIZE = 8 bytes

; Target chip : uPD789024
; Device file : Vx.xx
```

(9) Error list file creation specification

Error list file creation specification (-e, -se)

(a) -e

[Description format]

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

- Interpretation when omitted
No error list file is output.
- Output file
.ecc (: alphanumeric symbols)

[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 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The -e option is specified.

```
C>cc78k0s -c9024 prime.c -e
```

The error list file is referenced.

```
prime.c ( 18 ) : CC78K0S warning W0745 : Expected function prototype
prime.c ( 20 ) : CC78K0S warning W0745 : Expected function prototype
prime.c ( 26 ) : CC78K0S warning W0622 : No return value
prime.c ( 37 ) : CC78K0S warning W0622 : No return value
prime.c ( 44 ) : CC78K0S warning W0622 : No return value
```

```
Target chip : uPD789024
Device file : Vx.xx
```

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

(b) -se

[Description format]

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

- Interpretation when omitted
No error list file is output.
- Output files
 - *.cer : Error list for *.c files (* : alphanumeric symbols)
 - *.her : Error list for *.h files
 - *.er : Error list for files other than *.c and *.h files

[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 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. If 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The -se option is specified.

```
C>cc78k0s -c9024 prime.c -se
```

prime.cer is referenced.

```
/*
78K/0S Series C Compiler VX.XX Error List Date : XX XXX XXXX Time : XX : XX : XX

Command      : -c9024 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 ] ;
main ( )
{
    :
    prime = i + i + 3 ;
    printf ( " %6d " , prime ) ;
*** CC78K0S warning W0745 : Expected function prototype
    count++ ;
    if ( ( count%8 ) == 0 ) putchar ( ' \n ' ) ;
*** CC78K0S warning W0745 : Expected function prototype
    for ( k = i + prime ; k <= SIZE ; k += prime )
        :
}

```

(10) Cross-reference list file creation specification

Cross-reference list file creation specification (-x)

[Description format]

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

- Interpretation when omitted
No cross-reference list file is output.
- Output file
.xrf (: alphanumeric symbols)

[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".
- 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 options are specified, the output file name can also be changed.
- Specify the file name or the output destination in the "Output File" combo box under the << Output >> tab.

[Use Example]

- The -x option is specified.

```
C>cc78k0s -c9024 prime.c -x
```

prime.xrf is referenced.

```
78K/0S Series C Compiler Vx.xx Cross reference List  Date : XX XXX XXXX Page : 1
```

```
Command      : -c9024 prime -x
In-file      : prime.c
Xref-file    : prime.xrf
Para-file    :
```

ATTRIB	MODIFY	TYPE	SYMBOL	DEFIN	REFERENCE				
EXTERN	array		mark	5	14	16	22		
EXTERN	func		main	7					
AUTO1	int		i	9	13	13	13	14	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	func		printf	28	18	25			
EXTERN	func		putchar	39	20				
PARAM	pointer		s	29	36				
PARAM	int		i	30	35				
AUTO1	int		j	32	35				
AUTO1	pointer		ss	33	36				
PARAM	char		c	40	43				
AUTO1	char		d	42	43				
	#define		TRUE	1	14				
	#define		FALSE	2	22				
	#define		SIZE	3	5	13	15	21	

```
Target chip  : uPD789024
Device file  : Vx.xx
```

(11) List format specification

List format specification (-lw, -ll, -lt, -lf, -li)

(a) -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).
 $72 \leq \text{number of characters printed in 1 line} \leq 132$
- 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]

- The cross-reference list file when the -lw option is omitted is output to "file-name.xrf".

```
C>cc78k0s -c9024 prime.c -x
```

(b) -ll

[Description format]

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

- Interpretation when omitted
-ll66 (For console output, this becomes 65,535 lines)

[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.
 $20 \leq \text{number of lines printed on 1 page} \leq 65535$
- If -ll0 is specified, there is no page break.
- If the number of lines is omitted, the number of lines on 1 page becomes 66 lines (If output to the console, this becomes 65,535).
- If the list file specification specifies nothing, the -ll option is invalid.

[Use Example]

- The number of lines on 1 page of the cross-reference list file is set to 20 lines.

```
C>cc78k0s -c9024 prime.c -x -ll20
```

(c) -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.
 $0 \leq \text{number of specifiable characters} \leq 8$
- 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]

- The -lt option is omitted.

```
C>cc78k0s -c9024 prime.c -p
```

- The blanks based on the HT code are set to 1.

```
C>cc78k0s -c9024 prime.c -p -lt1
```

(d) -lf

[Description format]

```
-lf
```

- Interpretation when omitted
None

[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]

- The -lf option is specified.

```
C>cc78k0s -c9024 prime.c -a -lf
```

(e) -li

[Description format]

```
-li
```

- Interpretation when omitted
None

[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]

- The -li option is specified.

```
C>cc78k0s -c9024 prime.c -sa -li
```

(12) Warning output specification

Warning output specification (-w)

[Description format]

```
-w [ level ]
```

- Interpretation when omitted
-w1

[Function]

- The -w option specifies the output of warning messages 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.

Table5-11 Warning Message Levels

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]

- The warning messages when the -w option is omitted are referenced.

```
C>cc78k0s -c9024 prime.c
```

(13) Execution state display specification

Execution state display specification (-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 execute compiling while continuing to output the execution state of the compilation to the console.

[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]

- The -v option is specified.

```
C>cc78k0s -c9024 prime.c -v
```

(14) Parameter file specification

Parameter file specification (-f)

[Description format]

```
-f file-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 a compiler 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 parameter file prime.pcc

```
; parameter file  
prime.c -c9024 -aprime.asm -e -x
```

prime.pcc is used in the compilation.

```
C>cc78k0s -fprime.pcc
```

(15) Temporary file creation folder specification

Temporary file creation folder specification (-t)

[Description format]

```
-t folder
```

- Interpretation when omitted
The files are created in the drive and 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]

- This specifies output of the temporary files to the TMP folder.

```
C>cc78k0s -c9024 prime.c -ttmp
```

(16) Help specification

Help specification (--/?/-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+, press the [Help] button in the < Compiler Options > dialog box.

[Application]

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

[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 return key. To exit the display before the end, press any character other than the return key, and then press the return key.

[Use Example]

- The -h option is specified.

```
C>cc78k0s -h
```

(17) Function expansion specification

Function expansion specification (-z/-nz)

[Description formats]

```
-z [ type ] (If multiple types are specified, specify them consecutively)
-nz
```

- Interpretation when omitted
 - nz

[Function]

- The -z option enables the processing for type specification.
- The -nz option invalidates the -z option.
- Types must not be omitted, otherwise, an Fatal error (F0012) will occur.

[Application]

- The functions for processing by the following type specifications are available for the 78K0S Series expansion functions.

[Description]

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

Table5-12 Type Specifications of -z Option

Type Specification	Description
Omitted	Regarded as -nz specification.
p	The characters after "/" until the line return are interpreted as a comment.
c	Nested comments "/* */" are allowed.
s ^{Note}	Interprets the type of kanji in comments as SJIS code.
e ^{Note}	Interprets the type of kanji in comments as EUC code.
n ^{Note}	Interprets comments as not containing kanji codes.
b	char-/unsigned char-type argument and return value are not int-extended.

Table5-12 Type Specifications of -z Option

Type Specification	Description
a	<p>Functions not in the ANSI standard are illegal. The ANSI-compliant portion of the functions are valid.</p> <p>Specifically, the following tasks are performed.</p> <ul style="list-style-type: none"> - The following are no longer reserved words. callt, noauto, norec, sreg, bit, boolean, #asm, #endasm - The trigraph sequence (3-character representation) becomes valid. - The compiler-defined macro <code>__STDC__</code> is 1. - The following warning is output for a char type bit field. (CC78K0S warning W0787: Bit field type is char) - If -w2 is specified, the following warnings are output for the -qc, -zp, -zc, -zi, and -zl options. (CC78K0S warning W0029: '-QC' option is not portable) (CC78K0S warning W0031: '-ZP' option is not portable) (CC78K0S warning W0032: '-ZC' option is not portable) (CC78K0S warning W0036: '-ZI' option is not portable) (CC78K0S warning W0037: '-ZL' option is not portable) - If -w2 is specified, the following warning is output for each #pragma statement. (CC78K0S 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. (CC78K0S warning W0850: Asm statement is not portable) - If -w2 is specified, the following error is output for an #asm to #endasm block. (CC78K0S error E0801: Undefined control, etc.)
m [n] (n = 1, 2)	<p>Enables use of extend specifications for static model.</p> <p>Up to 6 arguments can be described in int size, and up to 9 arguments can be described in char size.</p> <p>Enables description of structure/union return value for 1-, 2-byte structure/union arguments and function return values.</p> <p>The <code>__@KREGxx</code> utilization method is changed by the value of n. If n is omitted, n is considered to be 1.</p> <p>1 : Use <code>__@KREGxx</code> as the shared area only for leaf function.</p> <p>2 : Perform <code>__@KREGxx</code> save/restore and allocate argument and automatic variable to <code>__@KREGxx</code>.</p>
d	<p>Place the processing routines before and after the function into a library.</p> <p>Outputs warning for -ql4 and processes as -ql3.</p>
r	<p>Automatically adds a pascal function modifier.</p>
i	<p>Regards int and short descriptions as char. The compiler definition macro <code>_FROM_INT_TO_CHAR</code> is regarded as 1.</p>
l	<p>Regards long description as int. The compiler definition macro <code>_FROM_LONG_TO_INT</code> is regarded as 1.</p>

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

[Use Example]

- The -zc and -zp options are specified.

```
C>cc78k0s -c9024 prime.c -zpc
```

(18) Device file search path

Device file search path (-y)

[Description format]

```
-y folder
```

- Interpretation when omitted
Normal search path only

[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.
The normal search paths are as follows.
 - (i) <..\dev> (for the path where cc78k0s.exe started)
 - (ii) Path where CC78K0S started
 - (iii) Current folder
 - (iv) PATH environment variable

[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 an option with this compiler.

[Use Example]

- The -y option is specified.

```
C>cc78k0s -c9024 -ya:\tmp\dev
```

(19) Static model specification

Static model specification (-sm)

[Description format]

```
-sm [ n ] ( n = 1-16 )
```

- Interpretation when omitted
Normal model (n = 0)

[Function]

- Specify the -sm option while compiling. The object when the -sm option is specified is called a static model, and the object when the -sm option is not specified is called a normal model.
- Normally, the instruction accessing a static area is shorter and can be executed faster than the instruction accessing a stack frame. Therefore, an object code can be shortened and execution speed improved.
- Interrupts can be serviced faster if the -sm option is specified. This is because the saving/returning of arguments and variables that use the saddr area (i.e., register variables in the interrupt function, arguments/automatic variables in the norec function, arguments of the run-time library, etc.) is not performed in the static model, whereas it is performed in the normal model.
- Memory capacitance is saved since data is shared with multiple leaf functions.

[Application]

- If you want to improve the object execution speed or want to make interrupt servicing faster, specify the -sm option to change a normal model to a static model.

[Description]

- All function arguments are given via a register, and a function assigns function arguments and automatic variables to a static area.
- The leaf function assigns function arguments and automatic variables from higher addresses to the FEFFH and lower area of the saddr in a description order. This saddr area is called the "common area", since this area is shared by the leaf functions of all modules.
- The value of n indicates the size of the common area.
- When n = 0 or n is omitted, there is no common area.
- The compiler definition macro `__STATIC__MODEL` is assumed to be 1.
- `sreg/__sreg` keyword can be added to function arguments and automatic variables. The function arguments and automatic variables that have an `sreg/__sreg` keyword added are assigned to the saddr area, and can be manipulated in 1-bit units.
- Specifying the -rk option assigns the function argument and automatic variable (except for a static variable in a function) to the saddr area and enables them to be manipulated in 1-bit units.

[Caution]

- Since arguments and automatic variables are secured statically, the contents of arguments and automatic variables of a recursive function may be damaged. When a recursive function calls itself, an error occurs. When a function is called to where another function has been called, however, no error occurs since the compiler cannot detect it.
- If a function that is processed during an interrupt is called by means of interrupt servicing (interrupt function or function called by interrupt function), its argument/automatic variable may be damaged.
- Even if a function that is processed during interrupt servicing uses a common area, saving/returning to/from a common area is not performed.

[Use Example]

```
C>cc78k0s -c9024 test.c -sm16
```

CHAPTER 6 C COMPILER OUTPUT FILES

This chapter describes the files that the CC78K0S outputs.

The CC78K0S 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]

```

; 78K/0S Series C Compiler V(1)x.xx Assembler Source
;                                     Date: (2)xxxxxx Time: (3)xxxxxx

; Command      : (4)-c9024 prime.c -sa
; In-file      : (5)prime.c
; Asm-file     : (6)prime.asm
; Para-file    : (7)

$PROCESSOR ((8) 9024 )
(9) $DEBUG
(10) $NODEBUGA
(11) $KANJICODE SJIS
(12) $TOL_INF 03FH , 0130H , 02H , 00H

(13) $DGS    FIL_NAM , .file , 033H , 0FFFEH , 03FH , 067H , 01H , 00H
;
(14)        EXTRN    @_cprep
;
; 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    1.13
(14)        push    hl                ; (21) [ INF ] 1 , 4
(14)        movw   ax , #08H          ; (21) [ INF ] 3 , 6
(14)        callt  [ @_cprep ]       ; (21) [ INF ] 1 , 8
;
(18) ??bf_main :
;
; (22) *** Code Information ***
;
; (23) $FILE C:\NECTools32\Smp78k0s\CC78K0S\prime.c
; (24) $FUNC main ( 8 )
; (25) bc = ( void )
; (26) CODE SIZE = 222 bytes , CLOCK_SIZE = 654 clocks , STACK_SIZE = 14
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 + 2 ] )
; (26) CODE SIZE = 28 bytes , CLOCK_SIZE = 108 clocks , STACK_SIZE = 10
bytes
;
; (24) $FUNC putchar ( 41 )
; (25) bc = ( char c : x )
; (26) CODE SIZE = 14 bytes , CLOCK_SIZE = 58 clocks , STACK_SIZE = 8 bytes

; Target chip : (19)uPD789024
; Device file : (20)Vx.xx

```

[Description of output items]

Table6-1 Description of Output Items (assembler source module file)

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 "CC78K0S". 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. See the documentation describing device files.
(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.
(11)	Kanji type information	Maximum 15 (variable)	Outputs the Kanji code type. Output is \$KANJI CODE SJIS, \$KANJI CODE EUC, or \$KANJI CODE 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.

Table6-1 Description of Output Items (assembler source module file)

Item Number	Description	Number of Columns	Format
(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]).
(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).
(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).

Table6-1 Description of Output Items (assembler source module file)

Item Number	Description	Number of Columns	Format
(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]

```

/*
78K/0S Series C Compiler V (1) x.xx Error List Date : (2) xxxxx Time : (3)
xxxxx

Command      : (4) -c9024 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) main ( )
(8) {
(8)     int      i , prime , k , count ;
(8)     cont = 0 ;
*** CC78K0S 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 ] ) {
(8)             prime = i + i + 3 ;
(8)             printf ( " %6d " , prime ) ;
*** CC78K0S warning (9) W0745 : (10) Expected function prototype
          :
/*
(11) Target chip : uPD789024
(12) Device file : Vx.xx
Compilation complete , (13) 1 error ( s ) and (14) 5 warning ( s ) found.
*/

```

[Description of output items]

Table6-2 Description of Output Items (error list file with C source)

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 "CC78K0S". 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. "C" is output if "#" is an internal error, "F" if it is a fatal error, "E" if is an error caused by syntax error or restriction of compiler, 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 ) : CC78K0S warning (3) W0745 : (4) Expected function
prototype
(1) prime.c ( (2) 20 ) : CC78K0S warning (3) W0745 : (4) Expected function
prototype
(1) prime.c ( (2) 26 ) : CC78K0S warning (3) W0622 : (4) No return value
(1) prime.c ( (2) 37 ) : CC78K0S warning (3) W0622 : (4) No return value
(1) prime.c ( (2) 44 ) : CC78K0S warning (3) W0622 : (4) No return value

Target chip : (7) uPD789024
Device file : (8) Vx.xx

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

[Description of output items]

Table6-3 Description of Output Items (error list file with error message only)

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. "C" is output if "#" is an internal error, "F" if it is a fatal error, "E" if it is an error caused by syntax error or restriction of compiler, 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 PAGESWIDTH = 80 >

```

/*
78K/0S Series C Compiler V (1) x.xx Preprocess List   Date : (2) xxxxx Page :
(3) xxx

Command :      (4) -c9024 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 : uPD789024
(11) Device file : Vx.xx
*/

```

[Description of output items]

Table6-4 Description of Output Items (preprocess list file)

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 "CC78K0S". 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.

Table6-4 Description of Output Items (preprocess list file)

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 PAGESWIDTH = 80 >

```

78K/0S Series C Compiler V (1) x.xx Cross reference List
                                         Date : (2) xxxxx Page : (3) xxx

Command      : (4) -c9024 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                array      mark      5      14  16  22
  EXTERN                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 : uPD789024
(16) Device file : Vx.xx */

```

[Description of output items]

Table6-5 Description of Output Items (cross-reference list file)

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 "CC78K0S". 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.
(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 AUTO nn , a register variable as REG nn (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, a callf function as CALLF, a noauto function as NOAUTO, a norec function as NOREC, 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, and a bank function as BANK.

Table6-5 Description of Output Items (cross-reference list file)

Item Number	Description	Number of Columns	Format
(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	7 (fixed)	This outputs the line number and file name defined for the symbol, and is displayed as : line number (5-digit) : include file number
(14)	Symbol reference line number	7 (fixed)	This outputs the line number and file name that reference the symbol, and is displayed as : line number (5-digit) : include file number If the line contents exceed the line length, the remaining contents are output beginning at column 47 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 CC78K0S.

7.1 Efficient Operation (EXIT Status Function)

When the compilation ends, the CC78K0S returns the top error level generated during compilation to the operating system as the EXIT status.

The EXIT status is shown below.

Ends normally :	0
WARNING :	0
FATAL ERROR :	1
ABORT :	2

If PM+ is not used and the CC78K0S is started in the command line, efficient operation can be further improved by using the status in a batch file.

[Use Example]

```
cc78k0s -c9024 %1
IF ERRORLEVEL 1 GOTO ERR
cc78k0s -c9024 %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 CC78K0S supports the following environment variables.

PATH :	Search path for executable forms
INC78K0S :	Search path for include files
TMP :	Search path for temporary files
LANG78K :	Type of kanji code (can be specified by -ze, -zs, or -zn option) (euc : EUC code, sjis : shift JIS code, none : no 2-byte codes)
LIB78K0S :	Search path for libraries

[Use Example]

< When using DOS prompt >

```
; AUTOEXEC.BAT
PATH C:\NECTools32\bin ; C:\bat ; C:\cc78k0s ; C:\tool
VERIFY ON
BREAK ON
SET INC78K0S = C:\NECTools32\inc78k0s
SET LIB78K0S = C:\NECTools32\lib78k0s
SET TMP = C:\tmp
SET LANG78K = sjis
```

[Description]

- Executable files are searched in the sequence of C:\NECTools32\bin, C:\bat, C:\cc78k0s, C:\tool by path specification.
- Include files are searched from C:\NECTools32\inc78k0s.
If no setting is made, search is performed from C:\NECTools32\inc78k0s (if the CC78K0S is installed to C:\NECTools32).
- Library files are searched from C:\NECTools32\lib78k0s during linking.
If no setting is made, search is performed from C:\NECTools32\lib78k0s (if the CC78K0S is installed to C:\NECTools32).
- 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+ window, or click the [Stop Build] button in the tool bar. When building in PM+, command key input is not accepted.

CHAPTER 8 STARTUP ROUTINES

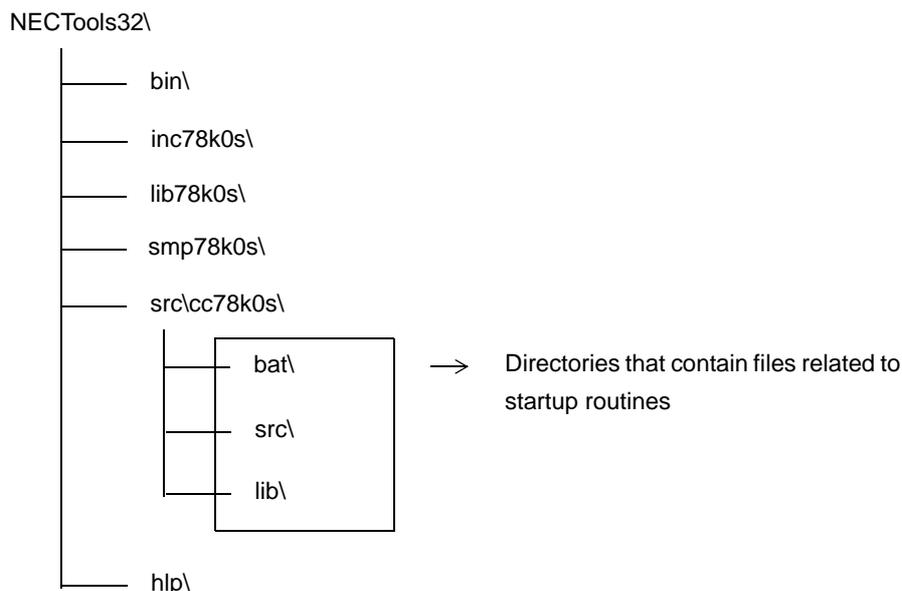
To execute a C language program, a program is needed to activate ROMization for inclusion in the system and the user program (main function). This program is called the startup routine.

To execute a program written by a user, a startup routine must be created for that program. The CC78K0S 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\cc78k0s` of the compiler package.



The contents of the directories under `src\cc78k0s` are shown next.

The `lib` folder contains the object files of the startup routines and the assembled library sources. An object file can be linked to a program for any target device in the 78K0S Series. If no particular revisions are needed, link the unmodified object files that are already available. If `mkstup.bat` offered by the CC78K0S is executed, this object file can be overwritten.

For the file contents, see "[2.6.4 Library files](#)".

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.

The device files (d9026.78k) in the bat folder are not for development, and are used when a batch file for updating library, etc., is started. Therefore, other optional device files are required for development.

Table8-1 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}
repcmul.bat	Batch file for updating runtime libraries for char type multiplication (with multiplier used)
repimul.bat	Batch file for adding or deleting runtime libraries for int type multiplication (with multiplier used)
replmul.bat	Batch file for adding or deleting runtime libraries for long type multiplication (with multiplier used)

Notes 1. Since ROMization routines are in the library, the library is also updated by this batch file.

Notes 2. The setjmp and longjmp that save the compiler reserved area (saddr area secured for KREG¥¥, etc.), and the setjmp and longjmp that do not save the compiler reserved area (only the registers are saved) are created.

8.1.2 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.

Table8-2 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
xcmul.asm	@@csmul, @@cumul function (Multiplier used)
ximul.asm	@@ismul, @@iumul function (Multiplier used)
xlmul.asm	@@lsmul, @@lumul function (Multiplier used)
def.inc	For setting library according to type
macro.inc	Macro definition for each typical pattern

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.

8.2 Batch File Description

8.2.1 Batch files for creating startup routines

The mkstup.bat in the bat directory is used to create the object file of a startup routine.

The assembler in the RA78K0S 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\cc78k0s\bat directory containing mkstup.bat.

```
mkstup device-typeNote
```

Note Refer to the document related to device files.

[Use Example]

- The startup routine to be used is created when the target device is the uPD789024.

```
mkstup 9024
```

The mkstup.bat batch file is stored in the form that overwrites the object file of the startup routine in the lib directory at the same level as the bat directory as shown below.

The startup routine that is required to link the object file is output to each directory.

The names of the object files created in lib are shown below.

```
lib\  _____ s0s.rel
      s0sl.rel
      s0ss.rel
      s0ssl.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

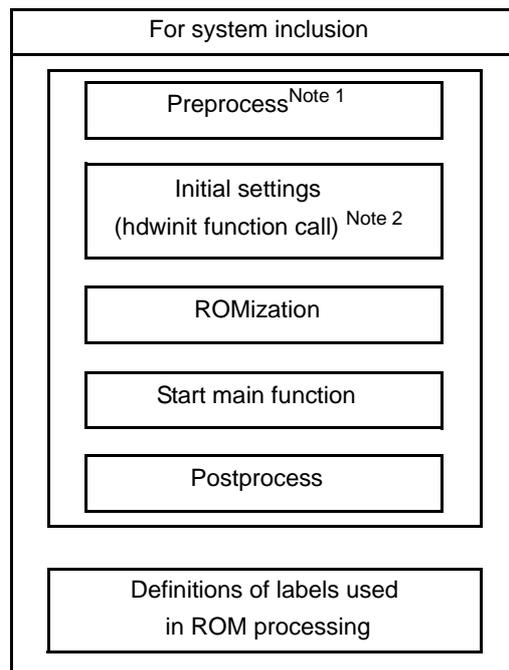
Memory initialization, ROMization for inclusion in the system, and the starting and ending processes for the C source program are performed.

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

Figure8-1 shows the programs related to the startup routines and their configurations.

Figure8-1 Overview of Startup Routines



Notes 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.

Notes 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.

Table 8-3 shows the differences between cstart.asm and cstartn.asm.

Table 8-3 Differences Between Startup Routine Sources

Type of Startup Routine	Uses Library Processing
cstart.asm	Yes
cstartn.asm	No

(3) Uses of startup routines

Table 8-4 lists the names of the object files for the source files provided by the CC78K0S.

Table 8-4 Correspondence Between Source Files and Object Files

File Type	Source File	Object File
Startup routine	cstart*.asm ^{Notes 1}	s0s*.rel ^{Notes 2}
ROM file	rom.asm	Included in library

Notes 1. *: If the standard library is not used, "n" is added. If used, the character is not added.

Notes 2. *: If a fixed area in the standard library is used, "l" is added.

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 in (1) to (6) (see below).

[cstart.asm preprocessing]

```

NAME      @cstart

$INCLUDE ( def.inc )                      ; (1)
$INCLUDE ( macro.inc )                    ; (2)
                                           ; (3)

BRKSW     EQU 1 ; brk , sbrk , calloc , free , malloc , realloc function use
EXITSW    EQU 1 ; exit , atexitfunction use
$_IF ( _STATIC )
RANDSW    EQU 0 ; rand , srandfunction use
DIVSW     EQU 0 ; divfunction use
LDIVSW    EQU 0 ; ldivfunction use
FLOATSW   EQU 0 ; floating point variables use
$ELSE
RANDSW    EQU 1 ; rand , srandfunction use
DIVSW     EQU 1 ; divfunction use
LDIVSW    EQU 1 ; ldivfunction use
FLOATSW   EQU 1 ; floating point variables use
$ENDIF
STRTOKSW  EQU 1 ; strtokfunction use

PUBLIC   _@cstart , _@cend

$_IF ( BRKSW )
PUBLIC  _@BRKADR , _@MEMTOP , _@MEMBTM
      :
$ENDIF
EXTRN  _main , _@STBEG , _hdwinit          ; (4)
$_IF ( EXITSW )
EXTRN  _exit
$ENDIF
                                           ; (5)
EXTRN  _?R_INIT , _?R_INIS , _?DATA , _?DATS
                                           ; (6)

@@DATA  DSEG      UNITP
$_IF ( EXITSW )
_@FNCTBL :        DS      2 * 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 RA78K0S Assembler Package Operation User's Manual).

[Example when `-sSTACK` is specified in linking]

Create `lk78k0s.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 `lk78k0s.dr` in the `smp78k0s\cc78k0s` folder.

	First address	Size	
	↓	↓	
memory SDR	:	(0FE20h , 00098h)	
memory STACK	:	(xxxxh , xxxh)	<-- Specify the first address and size here, then specify lk78k0s.dr by the -d linker option. (Example: -dlk78k0s.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 in (1) to (4).

[Initial settings in cstart.asm]

```

@@VECT00          CSEG    AT      0                                ; (1)
                  DW      @_cstart

@LCODE           CSEG
_cstart :

                  MOVW   AX , #_@STBEG    ; SP <-stack begin address      ; (2)
                  MOVW   SP , AX
                  CALL   !_hdwinit        ; (3)
$ENDIF
                  :
$_IF ( BRKSW OR EXITSW OR RANDSW OR FLOATSW )
                  MOVW   AX , #0
$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) 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.

(3) 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]

```

; *****
; ROM DATA COPY
; *****
; copy external variables having initial value
    MOVW    HL , #_@R_INIT
    MOVW    DE , #_@INIT
LINIT1 :
    MOVW    AX , HL
    CMPW    AX , #_?R_INIT
    BZ      $LINIT2
    MOV     A , [ HL ]
    MOV     [ DE ] , A
    INCW    HL
    INCW    DE
    BR      $LINIT1
LINIT2 :
    MOVW    HL , #_@DATA
; copy external variables which doesn't have initial value
LDATA1 :
    :
```

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

main ( )
{
    :
}
```

Note The external variables without initial value and sreg variables without initial value are not copied, and zeros are written directly to RAM.

- [Figure8-2](#) 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 compiler. The ROMization processing copies this value to the @@INIT segment in RAM (the same processes are performed for the variable (c)).
- The first and last labels in the @@R_INIT segment are defined by @_R_INIT and _?R_INIT. The first and last 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 (see [Table8-6](#)). [Table8-5](#) and [Table8-6](#) show the segment names of the ROM and RAM areas where the variables (a) to (d) are placed, and the first and last labels of the initial values in each segment.

Figure8-2 ROMization Processing

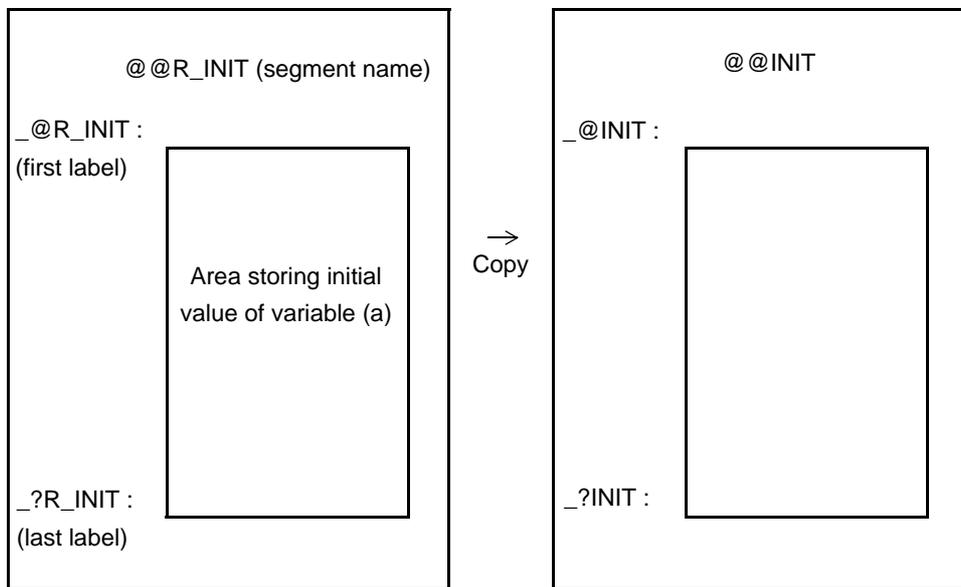


Table8-5 ROM Area for Initial Values

Variable Type	Segment	First Label	Last Label
External variable with initial value (a)	@@R_INIT	@_R_INIT	_?R_INIT
sreg variable with initial value (c)	@@R_INIS	@_R_INIS	_?R_INIS

Table8-6 RAM Area for Initial Values (Copy Destination)

Variable Type	Segment	First Label	Last Label
External variable with initial value (a)	@@INIT	@_INIT	_?INIT
External variable without initial value (b)	@@DATA	@_DATA	_?DATA
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 in (1) to (3).

[Starting main function and postprocessing]

```

CALL    !_main          ; main ( ) ;           ; (1)
$_IF ( EXITSW )
  MOVW   AX , #0
  CALL   !_exit         ; exit ( 0 ) ;        ; (2)
$ENDIF
  BR     $$
;
_@cend :
                                           ; (3)
@@R_INIT      CSEG    UNITP
_@R_INIT :
@@R_INIS      CSEG
_@R_INIS :
@@INIT       DSEG
_@INIT :
@@DATA       DSEG
_@DATA :
@@INIS       DSEG    SADDRP
_@INIS :
@@DATS       DSEG    SADDRP
_@DATS :
@@CALT       CSEG    CALLT0
@@CNST       CSEG
@@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 , _?R_INIS
PUBLIC    _?INIT , _?DATA , _?INIS , _?DATS
;
@@R_INIT      CSEG                      ; (1)
_?R_INIT :
@@R_INIS      CSEG      UNITP
_?R_INIS :
@@INIT        DSEG
_?INIT :
@@DATA        DSEG
_?DATA :
@@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 CC78K0S 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\cc78k0s\bat` folder to assemble the revised source file (`cstart*.asm`) (* : alphanumeric symbols).

- Symbols used in standard library functions

If the library functions listed in [Table8-7](#) 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.

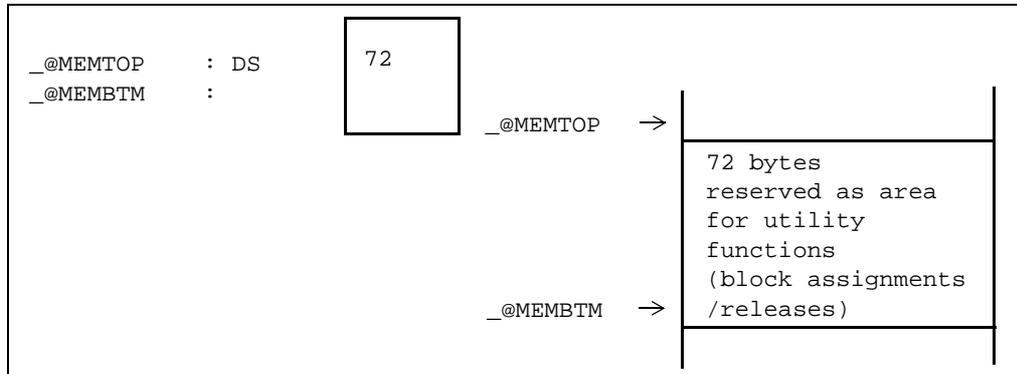
Table8-7 Symbols Used in Library Functions

Library Function Name	Symbols Used
brk sbrk strtol strtoul 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.

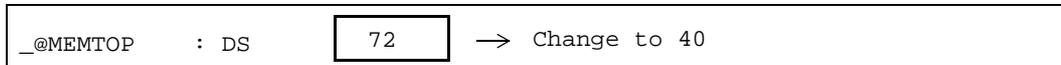
Example If you want to reserve 72 bytes for use by utility functions (block assignments/releases), make the following changes to the initial settings of the startup routine.



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



(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 RA78K0S Assembler Package Operation User's Manual).

- The CC78K0S sometimes uses a portion of the short direct address area (saddr area) in the following compiler-specific objectives. Specifically, this is the 40-byte area of FED8H to FEFFH for a normal model. When a static model is specified with the -sm[n] option, the part of saddr area [FEF0H to FEFFH] is used as the common area.

(Normal model)

- (a) Arguments of runtime library [FED8H to FEFFH]
 - (b) Arguments or automatic variables of norec function [FEF8H to FEF7H]
 - (c) register variable when the -qr option is specified [FED8H to FEF7H]
 - (d) Standard library task (part of the area (b) and (c)).
- If the user does not use the standard library, the area (d) is not used.

(Static model)

- (a) Common area [FEF0H to FEFFH]

The following shows an example of changing RAM size with a link directive file (lk78k0s.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.

< lk78k0s.dr >

	<u>First address</u>	<u>Size</u>	
memory RAM :	(0FB00h ,	00320h)	-> Make this size larger.
memory SDR :	(0FE20h ,	00098h)	(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 CC78K0S 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.

CHAPTER 9 ERROR MESSAGES

This chapter explains the causes of error messages output by the CC78K0S.

9.1 Error Message Format

The error message format is as follows.

```
Source-file-name (line-number) : Error-message
```

Examples

```
prime.c ( 8 ) :      CC78K0S error E0712 : Declaration syntax
prime.c ( 8 ) :      CC78K0S error E0301 : Syntax error
prime.c ( 8 ) :      CC78K0S error E0701 : External definition syntax
prime.c ( 19 ) :     CC78K0S warning W0745 : Expected function prototype
```

However, the C0101, C0103, and C0104 internal errors are output in the following format.

```
[ xxx.c < yyy > zzz ] CC78K0S error C0101 : Internal error
[ xxx.c < yyy > zzz ] CC78K0S error C0103 : Intermediate file error
[ xxx.c < yyy > zzz ] CC78K0S error C0104 : Illegal use of register
```

Remark xxx.c : source file name, yyy : line number, zzz : message

9.2 Types of Error Messages

The following 10 types of error messages are output by the compiler.

- (1) Error message for a command line
- (2) Error message for an internal error and memory
- (3) Error message for a character
- (4) Error message for configuration element
- (5) Error message for conversion
- (6) Error message for an expression
- (7) Error message for a statement
- (8) Error message for a declaration and function definition
- (9) Error message for a preprocessing directive
- (10) Error message for fatal file I/O and running on an illegal operating system

9.3 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 compiler processing for the error.

The error number format is as follows.

C/F/E/Wnnnn

C : Internal error

Compilation is always stopped if this error occurs. The object module file and assembler source file are not output.

F : Fatal error

Compilation is always stopped if this error occurs. The object module file and assembler source file are not output.

E : Error caused by syntax error or restriction of compiler

If more than a specific number of this error occurs, compilation is stopped. The object module file and assembler source file are not output.

W : Warning

Compilation continues.

nnnn (4-digit number)

From 0001 Error message for a command line

From 0101 Error message for an internal error or memory

From 0201 Error message for a character

From 0301 Error message for a configuration element

From 0401 Error message for conversion

From 0501 Error message for an expression

From 0601 Error message for a statement

From 0701 Error message for a declaration or a function definition

From 0801 Error message for a preprocessing directive

From 0901 Error message for fatal file I/O or 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.

9.3.1 Error message for a command line

Table9-1 Error Message for a Command Line < from 0001 >

Error Number	Error Message	
F0001	Message	Missing input file
	Cause	The input source file name was not specified.
	Response	"Please enter 'cc78k0s--' 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.
	Response	"Please enter 'cc78k0s--' 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.
	Response	"Please enter 'cc78k0s--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and input the parameter correctly.

Table9-1 Error Message for a Command Line < from 0001 >

Error Number	Error Message	
F0013	Message	Parameter not needed 'option'
	Cause	An unnecessary option parameter was specified.
	Response	"Please enter 'cc78k0s--' 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.
	Response	"Please enter 'cc78k0s--' 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.
	Response	"Please enter 'cc78k0s--' 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.
	Response	"Please enter 'cc78k0s--' 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.
	Response	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.
	Compiler	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.
	Compiler	The device type in the command line has priority.
F0024	Message	Illegal chip specifier on command line
	Cause	The device type in the command line is incorrect.

Table9-1 Error Message for a Command Line < from 0001 >

Error Number	Error Message	
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.
	Response	"Please enter 'cc78k0s--' if you want help message" is output. Use the --, -?, or -h option to reference the help file and correct the input.
W0036	Message	'-ZI' option is not portable
	Cause	The -zi option does not conform to the ANSI standard (For details about -zi, see " CHAPTER 5 COMPILER OPTIONS ").
W0037	Message	'-ZL' option is not portable
	Cause	The -zl option does not conform to the ANSI standard (For details about -zl, see " CHAPTER 5 COMPILER OPTIONS ").
W0038	Message	'-ZI' option specified - regarded as '-QC'
	Cause	Since the -zi that regards int and short as char is specified, the int extension control optimization option -qc becomes valid.
W0039	Message	'-SM' option specified - regarded as '-ZL'
	Cause	Since the static model specification option -sm is specified, the option -zl that regards long as int becomes valid.
W0040	Message	'-RK' option required '-SM' - ignored '-RK'
	Cause	The local variable optimization option -rk is valid only when the static model specification option -sm is specified. The option -rk is ignored.
W0041	Message	'-SM' option specified - ignored '-QR'
	Cause	Since the static model specification option -sm is specified, the register optimization option -qr is ignored.
W0045	Message	'-SM' option specified - ignored '-ZR'
	Cause	Since the static model specification option -sm is specified, the pascal function interface specification option -zr is ignored.
W0052	Message	'-ZD' option specified - regarded as '-QL3'
	Cause	Option (-zd) using library supporting prologue/epilogue is specified, so that -ql4 is treated as -ql3 for standard code pattern library conversion option (-ql).

Table9-1 Error Message for a Command Line < from 0001 >

Error Number	Error Message	
W0055	Message	'-ZM' option required '-SM' - ignored '-ZM'
	Cause	Static model extension specification option (-zm) is enabled only when static model specification option (-sm) is specified. -zm option is ignored.
W0065	Message	'-QW3' option deleted - regarded as '-QW2'
	Cause	Since the optimization specification option -qw3 is deleted, -qw2 becomes valid.
W0066	Message	'-QW5' option deleted - regarded as '-QW4'
	Cause	Since the optimization specification option -qw5 is deleted, -qw4 becomes valid.

9.3.2 Error message for an internal error and memory

Table9-2 Error Message for an Internal Error and Memory < from 0101 >

Error Number	Error Message	
C0101	Message	Internal error
	Cause	An internal error occurred.
	Response	Contact support.
E0102	Message	Too many errors
	Cause	The total number of FATAL errors exceeded 30.
	Compiler	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.
	Response	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.
	Response	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.
	Response	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.
	Response	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.
	Response	Decrease the parameters so that 64 KB is not exceeded.
E0110	Message	Compiler limit : too much code defined in file
	Cause	The area allocated for the code in the file exceeded the limit of 64 KB.
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.
E0113	Message	Compiler limit: too many local labels
	Cause	Number of local labels in 1 function exceeds the process limit.
	Response	The function itself is too large. Divide it.

9.3.3 Error message for a character

Table9-3 Error Message for a 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.
	Response	If the -za option was specified, since trigraph sequences are valid, this warning is not output.

9.3.4 Error message for configuration element

Table9-4 Error Message for Configuration Element < from 0301 >

Error Number	Error Message	
E0301	Message	Syntax error
	Cause	A syntax error occurred.
	Response	Make sure there are no description errors in the source.
E0303	Message	Expected identifier
	Cause	An identifier is required for the goto statement.
	Response	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.
	Response	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.

Table9-4 Error Message 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 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.
W0330	Message	'-ZI' option specified-int & short are treated as char in this file
	Cause	The -zi option is specified. int and short in this file are treated as char.
W0331	Message	'-ZL' option specified-long is treated as int in this file
	Cause	The -zl option is specified. long in this file is treated as int.
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.

Table9-4 Error Message for Configuration Element < from 0301 >

Error Number	Error Message	
W0333	Message	'-SM' option specified-ignored 'function attributes' keyword in this file
	Cause	The static model specification option -sm is specified. Function attributes in this file are ignored.
E0334	Message	'-SM' option specified-float & double keywords are not allowed
	Cause	The static model specification option -sm is specified. float and double keywords are not allowed.
W0335	Message	'-SM' option specified-long constant is treated as int constant
	Cause	The static model specification option -sm is specified. long constant is treated as int constant.
W0339	Message	'__temp' required '-SM -ZM' -ignored '__temp' in this file
	Cause	Temporary variable specification keyword __temp is enabled only when static model specification option (-sm) and static model extension specification option (-zm) are specified. The __ temp keyword is ignored in this file.
W0340	Message	Unreferenced label 'label name'
	Cause	The specified label has been defined, but has not been referenced even once.

9.3.5 Error message for conversion

Table9-5 Error Message 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.

9.3.6 Error message for an expression

Table9-6 Error Message for an 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.
W0508	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.
W0509	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.
E0512	Message	Cannot call function in norec function
	Cause	A function is called in the norec function. A function cannot be called in a norec function.
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.
	Response	Make the expression before the '->' operator a pointer to a structure or a union.

Table9-6 Error Message for an 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.
	Response	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.
E0530	Message	Illegal sizeof operand
	Cause	The operand of the sizeof expression must be an identifier or a type name.

Table9-6 Error Message for an Expression < from 0501 >

Error Number	Error Message	
E0531	Message	Disallowed conversion
	Cause	Illegal casting occurred.
	Response	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.
W0543	Message	Undeclared parameter in noauto or norec function prototype
	Cause	The parameter declarations are not in the prototype declarations of the noauto or norec function.
E0544	Message	Illegal type for parameter in noauto or norec function prototype
	Cause	Parameters with illegal types are declared in the prototype declarations of the noauto or norec function.
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.

Table9-6 Error Message for an Expression < from 0501 >

Error Number	Error Message	
E0549	Message	'-SM' option specified-recursive function is not allowed
	Cause	The static model specification option -sm is specified. Recursive call is not allowed.

9.3.7 Error message for a statement

Table9-7 Error Message for a 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 2048.
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.

Table9-7 Error Message for a 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.
	Response	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.

9.3.8 Error message for a declaration and function definition

Table9-8 Error Message for a 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.
	Response	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.
E0705	Message	Too many parameters in noauto or norec function
	Cause	A noauto or norec function has too many parameters.
	Response	Decrease the number of parameters.
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.
W0708	Message	Already declared symbol 'variable name'
	Cause	The same variable has already been declared.
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.
	Response	Set the variable definition once.
W0714	Message	Too many register variables
	Cause	There are too many declarations of register variables.
	Response	Decrease the number of register variables. For the number that can be used, refer to CC78K0S C Compiler Language User's Manual.

Table9-8 Error Message for a Declaration and Function Definition < from 0701 >

Error Number	Error Message	
E0715	Message	Too many sreg variables
	Cause	There are too many declarations of sreg variables.
E0716	Message	Not allowed automatic data in noauto function
	Cause	Automatic variables cannot be used in the noauto function.
E0717	Message	Too many automatic data in noauto or norec function
	Cause	There are too many automatic variables in a noauto or norec function.
	Response	Decrease the number of automatic variables in a noauto or norec function. For the number that can be used, refer to CC78K0S C Compiler Language User's Manual.
E0718	Message	Too many bit, boolean type variables
	Cause	There are too many bit and boolean type variables.
	Response	Decrease the number of bit, boolean, and __boolean type variables. For the number that can be used, refer to CC78K0S 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.
	Compiler	The register declaration is ignored and processing continues.
E0723	Message	Illegal type for parameter in noauto or norec function
	Cause	The type of a parameter in a noauto or norec function is too big.
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.
	Response	When an array is used in the member of a structure and the index is given by a constant computation, sometimes there is overflow by the -qc2 action and the area is not secured. In this case, specify -qc1 as in -qc. -qc is included in the default options.
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.

Table9-8 Error Message for a Declaration and Function Definition < from 0701 >

Error Number	Error Message	
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.
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.
	Response	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.

Table9-8 Error Message for a Declaration and Function Definition < from 0701 >

Error Number	Error Message	
W0745	Message	Expected function prototype
	Cause	The function prototype is not declared.
E0747	Message	Function prototype mismatch
	Cause	The function prototype declaration contains errors.
	Response	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.
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.
	Compiler	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.

Table9-8 Error Message for a Declaration and Function Definition < from 0701 >

Error Number	Error Message	
W0763	Message	Cannot declare const with bit, boolean
	Cause	bit and boolean type variables cannot have const declarations.
	Compiler	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.
	Compiler	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.
E0766	Message	Illegal type for automatic data in noauto or norec function
	Cause	The type of the automatic variable in a noauto or norec function is large.
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 / callf / noauto / norec / __banked / __pascal are not allowed for interrupt function
	Cause	An interrupt function cannot be declared callt, callf, noauto, norec, __banked, or __pascal.
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.
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.
E0781	Message	'-SM' option specified-variable parameters are not allowed
	Cause	The static model specification option -sm is specified. Variable parameters are not allowed.
E0782	Message	'-SM' option specified-structure & union parameter is not allowed
	Cause	The static model specification option -sm is specified. Structure and union parameters are not allowed.
E0783	Message	'-SM' option specified-structure & union return value is not allowed
	Cause	The static model specification option -sm is specified. Structure and union return values are not allowed.
E0784	Message	'-SM' option specified-too many parameters of function
	Cause	The static model specification option -sm is specified. Function arguments exceed the limit of 3 arguments/6 bytes.

Table9-8 Error Message for a Declaration and Function Definition < from 0701 >

Error Number	Error Message	
E0785	Message	'-SM' option specified-expected function prototype
	Cause	The static model specification option -sm is specified. Function prototype declaration is absent.
W0786	Message	'-SM' option specified-undeclared parameter in function prototype
	Cause	The static model specification option -sm is specified. Parameters are not declared in function prototype declaration.
W0787	Message	Bit field type is char
	Cause	char type is specified for bit field type.
W0792	Message	Undeclared parameter in __pascal function definition or prototype
	Cause	Parameters are not declared in __pascal function definition or prototype declaration. void must be described if there is no parameter.
W0793	Message	Variable parameters are not allowed for __pascal function - ignored __pascal
	Cause	Variable parameters cannot be specified for __pascal function. __pascal keyword is ignored.
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.

9.3.9 Error message for a preprocessing directive

Table9-9 Error Message for a 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.
	Response	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.
	Response	Specify the path in which an include file exists or specify a path using -i option for the environmental variable INC78K0S.
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 8.
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.
W0816	Message	Redefined macro name 'macro name'
	Cause	The macro name is redefined.

Table9-9 Error Message for a Preprocessing Directive < from 0801 >

Error Number	Error Message	
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.
	Compiler	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.
	Response	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.
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.

Table9-9 Error Message for a Preprocessing Directive < from 0801 >

Error Number	Error Message	
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.
	Response	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 'character string'
	Cause	The #pragma VECT 'character string', or INTERRUPT 'character string' is duplicated.
E0839	Message	Unrecognized pragma VECT or INTERRUPT 'character string'
	Cause	There is an unrecognized #pragma VECT 'character string', or INTERRUPT 'character string'.
W0840	Message	Undefined interrupt function 'function name'- ignored NOBANK or LEAFWORK specified
	Cause	The save destination is specified for an undefined interrupt function.
	Compiler	NOBANK specifications, or LEAFWORK 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.
W0846	Message	Rechanged section name 'section name'
	Cause	The same section name is duplicated and its specification is changed.
	Compiler	The section name specified last is valid and processing continues.
E0847	Message	Different NOBANK or LEAFWORK specified on same interrupt function 'function name'
	Cause	A different NOBANK specification or EAFWORK 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.

Table9-9 Error Message for a Preprocessing Directive < from 0801 >

Error Number	Error Message	
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.
	Compiler	It is shortened to the displayed 'module name'.
E0853	Message	Unrecognized pragma NAME 'module name'
	Cause	Unrecognizable characters are in the 'module name'.
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.
	Compiler	It is shortened to the displayed 'section name'. Make the section name 8 or fewer characters.
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.

9.3.10 Error message for fatal file I/O and running on an illegal operating system

Table9-10 Error Message 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.
	Response	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.
	Response	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 (18) 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.
	Response	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. <ol style="list-style-type: none"> 1. A file with the same name as a temporary file already exists on the drive and it has the read-only attribute. 2. The output temporary file is opened with the read-only attribute because of internal conflicts.
F0911	Message	Reading unwritten data, no available directory space
	Cause	An I/O error was generated for the following reasons. <ol style="list-style-type: none"> 1. EOF was passed and input proceeded. 2. The temporary file cannot be created because of insufficient folder area on the drive.

Table9-10 Error Message for Fatal File I/O and Running on an Illegal Operating System < from 0901 >

Error Number	Error Message	
F0912	Message	Write error on temp
	Cause	A write error was generated to the output temporary file.
	Response	A complex source expression (such as too deep nesting) may be the cause. Contact support.
F0913	Message	Requires MS-DOS V2.11 or greater
	Cause	The operating system is not MS-DOS (Ver. 2.11 or later).
F0914	Message	Insufficient memory in hostmachine
	Cause	The compiler cannot start because of insufficient memory.
	Response	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.

APPENDIX A SAMPLE PROGRAMS

This chapter introduces sample programs for the CC78K0S.

A.1 C Source Module File

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

char    mark [ SIZE + 1 ] ;

main ( )
{
    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 ) ;
}

printf ( s , i )
char    *s ;
int     i ;
{
    int     j ;
    char    *ss ;
    j = i ;
    ss = s ;
}

putchar ( c )
char    c ;
{
    char    d ;
    d = c ;
}
```

A.2 Execution Example

```
C>cc78K0S -c9024 prime.c -a -p -x -e -ng
```

```
78K/0S Series C Compiler Vx.xx [ xx xxx xxxx ]  
Copyright ( C ) NEC Electronics Corporation xxxx , xxxx  
  
sample\prime.c ( 18 ) : CC78K0S warning W0745 : Expected function prototype  
sample\prime.c ( 20 ) : CC78K0S warning W0745 : Expected function prototype  
sample\prime.c ( 26 ) : CC78K0S warning W0622 : No return value  
sample\prime.c ( 37 ) : CC78K0S warning W0622 : No return value  
sample\prime.c ( 44 ) : CC78K0S warning W0622 : No return value  
  
Target chip : uPD789024  
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

```

; 78K/0S Series C Compiler Vx.xx Assembler Source
;
;                               Date : xx xxx xxxx Time : xx : xx : xx
; Command      : -c9024 prime.c -a -p -x -e -ng
; In-file      : prime.c
; Asm-file     : prime.asm
; Para-file    :

$PROCESSOR      ( 9024 )
$NODEBUG
$NODEBUGA
$KANJI CODE SJIS
$TOL_INF       03FH , 0130H , 02H , 00H

        EXTRN    @_cprep
        EXTRN    @_RTARG0
        EXTRN    @@isrem
        PUBLIC   _mark
        PUBLIC   _main
        PUBLIC   _printf
        PUBLIC   _putchar

@@CNST CSEG
L0011 : DB      ' %6d '
        DB      00H
L0017 : DB      0AH
        DB      ' %d primes found. '
        DB      00H

@@DATA DSEG
_mark : DS      ( 201 )

; line 5
; line 8

@@CODE CSEG
_main :
        push    hl                ; [ INF ] 1 , 4
        movw   ax , #08H          ; [ INF ] 3 , 6
        callt  [ @_cprep ]       ; [ INF ] 1 , 8
; line 11
        mov    a , a              ; [ INF ] 2 , 4
        mov   [ hl ] , a          ; count ; [ INF ] 1 , 6
        mov   [ hl + 1 ] , a      ; count ; [ INF ] 2 , 6
; line 13
        mov   [ hl + 6 ] , a      ; i ; [ INF ] 2 , 8
        mov   [ hl + 7 ] , a      ; i ; [ INF ] 2 , 8
L0003 :
        mov   a , [ hl + 6 ]      ; i ; [ INF ] 2 , 6
        xch  a , x               ; [ INF ] 1 , 4
        mov   a , [ hl + 7 ]      ; i ; [ INF ] 2 , 6
        xor  a , #080H           ; 128 ; [ INF ] 2 , 4
        cmpw ax , #080C8H        ; -32568 ; [ INF ] 3 , 6
        bc   $$ + 4              ; [ INF ] 2 , 6
        bnz  $L0004              ; [ INF ] 2 , 6

```

```

; line 14
xor    a , #080H           ; 128           ; [ INF ] 2 , 4
addw  ax , #_mark         ; [ INF ] 3 , 6
movw  de , ax              ; [ INF ] 1 , 4
mov   a , #01H            ; 1           ; [ INF ] 3 , 6
mov   [ de ] , a           ; [ INF ] 1 , 6
mov   a , [ hl + 6 ]      ; i           ; [ INF ] 2 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   a , [ hl + 7 ] , a  ; i           ; [ INF ] 2 , 6
incw  ax                   ; [ INF ] 1 , 4
mov   [ hl + 7 ] , a      ; i           ; [ INF ] 2 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   [ hl + 6 ] , a      ; i           ; [ INF ] 2 , 6
br    $L0003               ; [ INF ] 2 , 6

L0004 :
; line 15
xor    a , a               ; [ INF ] 2 , 4
mov   [ hl + 6 ] , a      ; i           ; [ INF ] 2 , 6
mov   [ hl + 7 ] , a      ; i           ; [ INF ] 2 , 6

L0006 :
mov   a , [ hl + 6 ]      ; i           ; [ INF ] 2 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   a , [ hl + 7 ]      ; i           ; [ INF ] 2 , 6
xor   a , #080H           ; 128         ; [ INF ] 2 , 4
cmpw  ax , #080C8H        ; 200         ; [ INF ] 3 , 6
bc    $$ + 7              ; [ INF ] 2 , 6
bz    $$ + 5              ; [ INF ] 2 , 6
br    !L0007              ; [ INF ] 3 , 6

; line 16
xor    a , #080H           ; 128         ; [ INF ] 2 , 4
addw  ax , #_mark         ; [ INF ] 3 , 6
movw  de , ax              ; [ INF ] 1 , 4
mov   a , [ de ]          ; [ INF ] 1 , 6
cmp   a , #00H            ; 0           ; [ INF ] 2 , 4
bz    $L0015v             ; [ INF ] 2 , 6

; line 17
mov   a , [ hl + 6 ]      ; i           ; [ INF ] 2 , 8
rolc  a , 1                ; [ INF ] 1 , 2
xch  a , x                 ; [ INF ] 1 , 4
mov   a , [ hl + 7 ]      ; i           ; [ INF ] 2 , 6
rolc  a , 1                ; [ INF ] 1 , 2
addw  ax , #03H           ; 3           ; [ INF ] 3 , 6
mov   [ hl + 5 ] , a      ; prime       ; [ INF ] 2 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   [ hl + 4 ] , a      ; prime       ; [ INF ] 2 , 6

; line 18
xch  a , x                 ; [ INF ] 1 , 4
push  ax                   ; [ INF ] 1 , 4
movw  ax , #L0011         ; [ INF ] 3 , 6
cal   !_printf            ; [ INF ] 3 , 6
pop   ax                   ; [ INF ] 1 , 6

; line 19
mov   a , [ hl ]          ; count       ; [ INF ] 1 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   a , [ hl + 1 ]      ; count       ; [ INF ] 2 , 6
incw  ax                   ; [ INF ] 1 , 4
mov   [ hl + 1 ] , a      ; count       ; [ INF ] 2 , 6
xch  a , x                 ; [ INF ] 1 , 4
mov   [ hl ] , a          ; count       ; [ INF ] 1 , 6

```

```

; line 19
mov     a , [ hl ]           ; count           ; [ INF ] 1 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 1 ]       ; count           ; [ INF ] 2 , 6
incw    ax                   ; [ INF ] 1 , 4
mov     [ hl + 1 ] , a       ; count           ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     [ hl ] , a           ; count           ; [ INF ] 1 , 6
; line 20
xch     a , x                ; [ INF ] 1 , 4
movw    @_RTARG0 , ax        ; [ INF ] 2 , 8
movw    ax , #08H            ; 8              ; [ INF ] 3 , 6
call    !@@isrem            ; [ INF ] 3 , 6
or      a , x                ; [ INF ] 2 , 4
bnz     $L0012              ; [ INF ] 2 , 6
movw    ax , #0AH           ; 10             ; [ INF ] 3 , 6
call    !_putchar          ; [ INF ] 3 , 6
L0012 :
; line 21
mov     a , [ hl + 6 ]       ; i              ; [ INF ] 2 , 6
add     a , [ hl + 4 ]       ; prime          ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 7 ]       ; i              ; [ INF ] 2 , 6
addc    a , [ hl + 5 ]       ; prime          ; [ INF ] 2 , 6
mov     [ hl + 3 ] , a       ; k              ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     [ hl + 2 ] , a       ; k              ; [ INF ] 2 , 6
L0014 :
mov     a , [ hl + 2 ]       ; k              ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 3 ]       ; k              ; [ INF ] 2 , 6
xor     a , #080H           ; 128            ; [ INF ] 2 , 4
cmpw    ax , #080C8H        ; 200            ; [ INF ] 3 , 6
bc      $$ + 4              ; [ INF ] 2 , 6
bnz     $L0015              ; [ INF ] 2 , 6
; line 22
xor     a , #080H           ; 128            ; [ INF ] 2 , 4
addw    ax , #_mark         ; [ INF ] 3 , 6
movw    de , ax             ; [ INF ] 1 , 4
xor     a , a               ; 128            ; [ INF ] 2 , 4
mov     [ de ] , a          ; [ INF ] 1 , 6
mov     a , [ hl + 2 ]       ; k              ; [ INF ] 2 , 6
add     a , [ hl + 4 ]       ; prime          ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 3 ]       ; k              ; [ INF ] 2 , 6
addc    a , [ hl + 5 ]       ; prime          ; [ INF ] 2 , 6
mov     [ hl + 3 ] , a       ; k              ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     [ hl + 2 ] , a       ; k              ; [ INF ] 2 , 6
br      $L0014              ; [ INF ] 2 , 6
L0015 :
; line 24
mov     a , [ hl + 6 ]       ; i              ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 7 ]       ; i              ; [ INF ] 2 , 6
incw    ax                   ; [ INF ] 1 , 4
mov     [ hl + 7 ] , a       ; i              ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     [ hl + 6 ] , a       ; i              ; [ INF ] 2 , 6
br      !L0006              ; [ INF ] 3 , 6

```

```

L0007 :
; line 25
mov     a , [ hl ]           ; count           ; [ INF ] 1 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 1 ]       ; count           ; [ INF ] 2 , 6
push    ax                   ; [ INF ] 1 , 4
movw    ax , #L0017          ; [ INF ] 3 , 6
call    !_printf             ; [ INF ] 3 , 6
pop     ax                   ; [ INF ] 1 , 6
; line 26
movw    ax , #08H            ; [ INF ] 3 , 6
callt   [ @_cdisp ]         ; [ INF ] 1 , 8
pop     hl                   ; [ INF ] 1 , 6
ret     ; [ INF ] 1 , 6
; line 31
_printf :
push    hl                   ; [ INF ] 1 , 4
push    ax                   ; [ INF ] 1 , 4
movw    ax , #04H           ; [ INF ] 3 , 6
callt   [ @_cprep ]        ; [ INF ] 1 , 8
; line 35
mov     a , [ hl + 10 ]      ; i               ; [ INF ] 2 , 6
mov     [ hl + 2 ] , a       ; j               ; [ INF ] 2 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 11 ]     ; i               ; [ INF ] 2 , 6
mov     [ hl + 3 ] , a       ; j               ; [ INF ] 2 , 6
; line 36
mov     a , [ hl + 4 ]      ; s               ; [ INF ] 2 , 6
mov     [ hl ] , a          ; ss              ; [ INF ] 1 , 6
xch     a , x                ; [ INF ] 1 , 4
mov     a , [ hl + 5 ]     ; s               ; [ INF ] 2 , 6
mov     [ hl + 1 ] , a      ; ss              ; [ INF ] 2 , 6
; line 37
pop     ax                   ; [ INF ] 1 , 6
pop     ax                   ; [ INF ] 1 , 6
pop     ax                   ; [ INF ] 1 , 6
pop     hl                   ; [ INF ] 1 , 6
ret     ; [ INF ] 1 , 6
; line 41
_putchar :
push    hl                   ; [ INF ] 1 , 4
push    ax                   ; [ INF ] 1 , 4
movw    ax , #02H           ; [ INF ] 3 , 6
callt   [ @_cprep ]        ; [ INF ] 1 , 8
; line 43
mov     a , [ hl + 2 ]      ; c               ; [ INF ] 2 , 6
mov     [ hl + 1 ] , a      ; d               ; [ INF ] 2 , 6
; line 44
pop     ax                   ; [ INF ] 1 , 6
pop     ax                   ; [ INF ] 1 , 6
pop     hl                   ; [ INF ] 1 , 6
ret     ; [ INF ] 1 , 6
END

```

```
; ***Code Information***  
;  
; $FILE C:\NECTools32\sample\prime.c  
;  
; $FUNC main ( 8 )  
;   bc = ( void )  
;   CODE SIZE = 218 bytes , CLOCK_SIZE = 654 clocks , STACK_SIZE = 14 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 ( 31 )  
;   bc = ( pointer s : ax , int i : [ sp + 2 ] )  
;   CODE SIZE = 28 bytes , CLOCK_SIZE = 108 clocks , STACK_SIZE = 10 bytes  
;  
; $FUNC putchar ( 41 )  
;   bc = ( char c : x )  
;   CODE SIZE = 14 bytes , CLOCK_SIZE = 58 clocks , STACK_SIZE = 8 bytes  
  
; Target chip : uPD789024  
; Device file : Vx.xx
```

A.3.2 Preprocess list file

```

/*
78K/0S Series C Compiler Vx.xx Preprocess List      Date : xx xxx xxxx Page : 1
Command      : -c9024 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 : char    mark [ SIZE + 1 ] ;
6 :
7 : main ( )
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 ( s , i )
29 : char    *s ;
30 : int     i ;
31 : {
32 :     int     j ;
33 :     char    *ss ;
34 :
35 :     j = i ;
36 :     ss = s ;
37 : }
38 :
39 : putchar ( c )
40 : char    c ;
41 : {
42 :     char    d ;
43 :     d = c ;
44 : }

/*
Target chip : uPD789024
Device file : Vx.xx
*/

```

A.3.3 Cross-reference list file

78K/0S Series C Compiler Vx.xx Cross reference List Date : XX XXX XXXX Page : 1

Command : -c9024 prime.c -x
 In-file : prime.c
 Xref-file : prime.xrf
 Para-file :

ATTRIB	MODIFY	TYPE	SYMBOL	DEFINE	REFERENCE
EXTERN		array	mark	5	14 16 22
EXTERN		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		func	printf	28	18 25
EXTERN		func	putchar	39	20
PARAM		pointer	s	29	36
PARAM		int	i	30	35
AUTO1		int	j	32	35
AUTO1		pointer	ss	33	36
PARAM		char	c	40	43
AUTO1		char	d	42	43
		#define	TRUE	1	14
		#define	FALSE	2	22
		#define	SIZE	3	5 13 15 21

Target chip : uPD789024
 Device file : Vx.xx

A.3.4 Error list file

```
PRIME.C ( 18 ) : CC78K0S warning W0745 : Expected function prototype
PRIME.C ( 20 ) : CC78K0S warning W0745 : Expected function prototype
PRIME.C ( 26 ) : CC78K0S warning W0622 : No return value
PRIME.C ( 37 ) : CC78K0S warning W0622 : No return value
PRIME.C ( 44 ) : CC78K0S warning W0622 : No return value
```

```
Target chip : uPD789024
```

```
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 CC78K0S.

TableB-1 List of Use-related Cautions

Number	Cautions
1	<p>[Cautions related to specification of options]</p> <ul style="list-style-type: none"> - 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 occurs. 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	<p>[Cautions related to file output destinations]</p> <p>Only disk-type files can be specified as the output destination for object module files.</p>
3	<p>[Cautions related to error messages]</p> <p>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.</p>
4	<p>[Cautions related to source file names]</p> <p>In the CC78K0S, 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.</p> <ul style="list-style-type: none"> - 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 (.). When using PM+, separate the primary name and extension with a dot (.), and use ".c", ".C" as the C source extension. - 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. When PM+ is used, do not specify path names that include a space, or path names that include 2-byte characters, such as Chinese characters. - Sharp symbol (#) cannot be used for file names and path names in parameter files. - An error is output during linking for files that have the same name as the first 8 characters of the primary name. - If using the ID78K0S-NS or SM78K0S, the characters that can be used for the file name are lowercase letters (a to z), uppercase letters (A to Z), numbers (0 to 9), underscores (_), and dots (.)
5	<p>[Cautions related to include files]</p> <p>It is not possible to define functions (excluding declarations) in an include file and then expand the file within the C source.</p> <p>When definitions are made within an include file, problems such as incorrect display of definition lines during source debugging may occur.</p>

TableB-1 List of Use-related Cautions

Number	Cautions
6	<p>[Cautions related to use of output assembler source]</p> <p>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, such as when descriptions using assembly language are used.</p> <ul style="list-style-type: none"> - 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 compiler 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 RA78K0S 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.
7	<p>[Usable assembler package]</p> <p>Because long file names are supported, an error may occur if an RA78K0S earlier than Ver.1.30 is used.</p>
8	<p>[Cautions when using network]</p> <p>When the folder where the temporary files are created is placed in a file system shared on a network, file contention may arise, depending on the type of network software being used, and abnormal operation may result. Avoid such contention by setting the options and the environment variables. Do not use the CC78K0S in the network environment when using PM+.</p>

TableB-1 List of Use-related Cautions

Number	Cautions
9	<p>[Creating link directive file]</p> <p>When an area outside of the ROM or RAM area of the target device is used when linking the objects created by the compiler, 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.</p> <p>For information about creating link directive files, see RA78K0S Assembler Package Operation User's Manual and lk78k0s.dr (in the smp78k0s folder) equipped with the compiler.</p> <p>Example When you want to place external variables without initial values (except sreg variables) from a certain C source file to the external memory.</p> <p>(1) Change the section name for the external variables without initial value at the beginning of the C source.</p> <pre>#pragma section @@DATA EXTDATA :</pre> <p>Caution Initialization of the changed segment and ROMization should be performed by changing the startup routine.</p> <p>(2) Create a link directive file.</p> <p>< lk78k0s.dr ></p> <pre>memory EXTRAM : (0F000h , 00200h) merge EXTDATA : = EXTRAM</pre> <p>Heed the following points when creating a link directive file.</p> <p>(1) 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).</p> <p>Example When added to the link directive file lk78k0s.dr</p> <pre>memory EXTRAM : (0F000h , 00200h) memory STK : (0FB00H , 20H) merge EXTDATA : = EXTRAM</pre> <p>(Command line)</p> <pre>> lk78k0s s0sl.rel prime.rel -bcl0s.lib -sstk -dlk78k0s.dr</pre> <p>(2) The following error may be output when linking in the defined memory area.</p> <pre>" *** RA78K0S error E3206: Segment 'xxx' can't allocate to memory- ignored. "</pre> <p>[Cause]</p> <p>Because of insufficient space in the defined memory area, the indicated segment cannot be located.</p> <p>[Response]</p> <p>The response action is roughly divided into the following 3 steps.</p> <p>(i) Examine the size of a segment that cannot be located (refer to the .map file).</p> <p>(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.</p> <p>(iii) Specify the directive file specification option -d and link.</p> <p>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.</p> <ul style="list-style-type: none"> - 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).

TableB-1 List of Use-related Cautions

Number	Cautions
10	<p>[Cautions 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).</p> <ul style="list-style-type: none"> - When the first argument is specified, use the va_starttop - When the second and subsequent arguments are specified, use the va_start macro.
11	<p>[Cautions when referencing special function register (SFR) constant address] If the 16-bit SFR is referenced by a constant address reference, use the SFR name to reference it since an illegal code is generated to access in 8-bit units.</p>
12	<p>[Startup routines and libraries]</p> <ul style="list-style-type: none"> - Use the provided startup routines and libraries with the same versions as the files in the executable form (cc78k0s.exe or cc78k0s). - (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 "<u>±</u>", <u>±</u>0 is regarded as the conversion result. <p>[Prevention method] None</p>
13	<p>[Cautions when source debugging with ID78K0S-NS] When calling a pascal function, the Next command operates as the same as the Step command. Return to the calling side of the function with the Return command, etc. When the compile option -zr is specified, all functions become pascal functions. Therefore, never execute the Next command.</p>
14	<p>[Cautions when source debugging with SM78K0S] Do not execute the Next command when calling a pascal function. Otherwise, a runaway will occur. When the compile option -zr is specified, all functions become pascal functions. Therefore, be sure not to execute the Next command when the -zr is specified.</p>

TableB-1 List of Use-related Cautions

Number	Cautions
15	<p>[When performing ROMization]</p> <p>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 CC78K0S, a code is generated by default for ROMization. Therefore, it is necessary to perform linking with the startup routine including ROMization during linking.</p> <p>The following startup routines, all including ROMization processing, are provided by the C compiler. If the flash memory self rewrite mode for is used, refer to Table 8-4.</p> <p>Startup routines :</p> <p>(1) When not using C standard library area : s0s.rel (2) When using C standard library area : s0sl.rel</p> <p>[Usage example]</p> <pre>C:>lk78k0s s0s.rel sample.rel -s -bc10s.lib -osample.lmf</pre> <pre>sample.rel : Object module file of user program s0s.rel : Startup routine c10s.lib : Runtime library, standard library</pre> <p>The -s option is a stack symbol (_@STBEG, _@STEND) automatic generation option.</p> <p>Cautions</p> <ul style="list-style-type: none"> - Be sure to link the startup routine at the beginning. - When creating a library, create it separately from the library provided by the CC78K0S, and specify it prior to the compiler library during linking. - Do not add user functions to the CC78K0S library. - When using a floating point library (c10s*f.lib), it is necessary to link the startup routine including the ROMization processing to both the standard library and the floating point library. <p>When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points</p> <p>Example</p> <pre>-bmylib.lib -bc10sf.lib -bc10s.lib</pre> <p>When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points</p> <p>Example</p> <pre>-bmylib.lib -bc10s.lib -bc10sf.lib</pre>
16	<p>[Stack area symbol generation (-s)]</p> <p>In CC78K0S, the user cannot secure a stack area.</p> <p>To secure a stack area, specify the -s option during linking.</p> <p>When using PM+, the -s option is automatically attached when the source file specification includes the C source.</p>
17	<p>[ROM code]</p> <p>When ordering ROM code, specify the -r or -u object converter options, such as -r, -u0FFH (do not cancel the specification).</p> <p>Example</p> <pre>-r -u0FFH</pre> <pre>-r : Sort HEX file contents by order of addresses. -u fill value : Fill empty space in ROM code with the specified fill value.</pre>
18	<p>[Help specification option]</p> <p>In PM+, compiler options --, -?, and -h, which display option descriptions, are ignored.</p> <p>For help, click the [Help] button in the < Option Setup > dialog box of each tool.</p>

TableB-1 List of Use-related Cautions

Number	Cautions
19	<p>[-ll option specification] When using PM+, the maximum number that can be specified for the -ll option is 32767. If a number that exceeds 32767 is specified, specify -ll with another option.</p>
20	<p>[Cautions regarding symbol name length] When using ID78K0S-NS Ver.1.01 and SM78K0S Ver.1.42 or earlier versions, do not use symbol names with more than 127 characters.</p>
21	<p>[Cautions when using PM+]</p> <p>(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.</p> <ul style="list-style-type: none"> - 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. <p>(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.</p> <p>(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.</p> <p>Example</p> <pre>#if 0 #include " header1.h " /* Dependence relationship judged */ /* to exist */ #else / * ! zero */ #include " header2.h " #endif /* #include " header3.h " */</pre> <p>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+.</p> <p>[Prevention method] None. However, this has no effect on the build processing.</p> <p>(d) Project-related file settings Compiler 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 compiler attribute startup routine and standard library settings from the << Startup Routine >> tab in the < Compiler Options > dialog box.</p>

TableB-1 List of Use-related Cautions

Number	Cautions
22	<p>[Cautions related to prototype declaration] If a function prototype declaration does not contain a function type specification, an error (E0301, E0701) results.</p> <p>Example</p> <pre>f (void) ; /* E0301 : Syntax error */ /* E0701 : External definition syntax */</pre> <p>[Prevention method] Describe the function type.</p> <p>Example</p> <pre>int f (void) ;</pre>
23	<p>[Cautions related to 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.</p> <p>Example</p> <pre>extren int i ; /* extern spelling error. No error results here. */ /* comment */ void f (void) ; [EOF] /* Error such as E0712 */</pre> <p>[Prevention method] None</p>
24	<p>[Cautions related to description of comments in preprocessing directive] In the description of preprocessing directives, when a comment is described at the same line as a function type macro either before or within a preprocessing directive, an error (E0803, E0814, E0821, etc.) results.</p> <p>Example</p> <pre>/* 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 */</pre> <p>[Prevention method] Describe the comment after the preprocessing directive.</p> <p>Example</p> <pre>#pragma sfr /* com1 */ #define ONE 1 /* com2 */ #define TWO 2 /* com3 */ #ifdef ANSI_C /* com4 */ #endif /* com5 */ #define SUB (p1 , p2) p2 = p1 /* com6 */</pre>

TableB-1 List of Use-related Cautions

Number	Cautions
25	<p>[Cautions related to 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 results if condition (a) below is fulfilled, and an error results if condition (b) below is fulfilled.</p> <p>(a) If the tag is used in an argument declaration and a pointer to a structure or union is defined, warning (W0510) results when a function is called.</p> <p>Example</p> <pre>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) ; }</pre> <p>(b) If the tag is used in a return value type declaration of an argument declaration, and a structure, union, or enum type is specified, error (E0737) results.</p> <p>Example</p> <pre>/* 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 ; } ;</pre> <p>[Prevention method] Define the tag of the structure, union, or enum beforehand.</p>
26	<p>[Cautions related to 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.</p> <p>Example</p> <pre>void f (void) ; void f (void) { char *p , *p1 , *p2 ; char *ca [3] = { p , p1 , p2 } ; /* Error(E0750) */ }</pre> <p>[Prevention method] Describe an assignment statement and use it instead of initialization.</p> <p>Example</p> <pre>void f (void) ; void f (void) { char *ca [3] ; char *p , *p1 , *p2 ; ca [0] = p ; ca [1] = p1 ; ca [2] = p2 ; }</pre>

TableB-1 List of Use-related Cautions

Number	Cautions
27	<p>[Cautions related to extern callt function]</p> <p>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 results during assembly.</p> <p>Example</p> <pre> callt extern void funca (void) ; callt extern void funcb (void) ; callt extern void funcc (void) ; static void (* const func []) () = { funca , funcb , funcc } ; callf void func2 (void) { funcc () ; funcb () ; funca () ; } </pre> <p>[Prevention method] Separate the function table and function call module.</p>
28	<p>[Cautions related to functions returning a structure]</p> <p>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.</p> <p>Example</p> <pre> struct str { char c ; int i ; long l ; } st ; struct str func () { /* Interrupt occurrence */ : } void main () { st = func () ; /* Interrupt occurrence */ } </pre> <p>If the func function is called at the interrupt destination during the above servicing, st may be corrupted.</p> <p>[Prevention method] None</p>

TableB-1 List of Use-related Cautions

Number	Cautions
29	<p>[Cautions related to union initialization] When, during initialization of unions having structures, unions, or arrays as members, the initializer syntax is specified with nesting, error (E0750) results.</p> <p>Example</p> <pre> struct Ss { int d1 , d2 ; } ; union Au { struct Ss t1 ; } u = { { 1 , 2 } } ; /* E0750 Initializer syntax */ </pre> <p>[Prevention method] Do not specify the initializer of a union with nesting.</p> <p>Example</p> <pre> struct Ss { int d1 , d2 ; } ; union Au { struct Ss t1 ; } u = { 1 , 2 } ; </pre>
30	<p>[Cautions related to kanji code classification] To use a source containing EUC code, set the environmental variable LANG78K to euc, or specify the -ze option.</p>

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.

TableC-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
Device type specification	-c device-type	Specifies the type of target device.	Independent	None
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	-r [process-type] (Multiple specifications are possible)	Specifies the method of memory assignment.	If -r and -nr, -rd and -nr, -rk 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.		
	-rk [n] [m] (n = 1, 2, 4)	Assigns a function argument and auto variable (except for the static auto variable) automatically 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] (If multiple types are specified, specify them consecutively)	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
	-ng	Invalidates the -g option.		

TableC-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
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.	Nothing (no file is output)
	-k [process-type] (Multiple specifications are possible)	Specifies processing for the preprocess lists.		-kfln
Preprocess specification	-d macro-name [= definition-name] [, macro-name [=definition-name]]... (Multiple specifications are possible)	Specifies processing which is compatible for text that is defined in C source.	Independent	Only the macro definitions in a C source module file are valid.
	-u macro-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.
	-i folder [, 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 INC78K0S 3. C:\NECTools 32\inc78k0s
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.

TableC-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	-ll66 (For console output, this becomes 65,535 lines)
	-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	None
	-li	Adds the C source of the include file to the assembler source module file with C source comments.	Independent	None
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.
Temporary file creation folder specification	-t folder	Creates temporary files in specified drives and directories.	Independent	The drive and folder specified by the environment variable TMP
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

TableC-1 Compiler Options

Types	Description Format	Functions	Relationship with Other Options	Interpretation when omitted
Function expansion specification	-z [type] (If multiple types are specified, specify them consecutively)	Specifies the processing for type specification.	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
Static model specification	-sm [n] (n = 1-16)	Specifies the static models or normal models of objects.	Independent	Normal model (n = 0)

APPENDIX D INDEX

Symbols

#pragma pc ... 75
*.asm ... 31
*.bat ... 31
*.chm ... 31
*.dll ... 31
*.h ... 31
*.hlp ... 31
-/?/-h option ... 113
_@BRKADR ... 149
_@DIVR ... 149
_@FNCENT ... 149
_@FNCTBL ... 149
_@LDIVR ... 149
_@MEMBTM ... 149
_@MEMTOP ... 149
_@SEED ... 149
_@STBEG ... 143, 144
_@TOKPTR ... 149

A

-a option ... 94
ABORT ... 133
ANSI-C ... 13
Assembler ... 18
Assembler source ... 194
Assembler source module file ... 120

B

Build ... 24

C

C compiler ... 17
-c option ... 75
cc78k0s.exe ... 31
cc78k0s.msg ... 31
cc78k0sp.dll ... 35
cer ... 63
Constant address reference ... 196
Cross-reference list file ... 130
cstart*.asm ... 31, 141
cstart.asm ... 138, 141, 142
cstartn.asm ... 138, 141

D

-d option ... 91
Debugger ... 22

E

-e option ... 98
ecc ... 63
Environment variable ... 30

er ... 63
_errno ... 149
Error level ... 133
Error list file ... 125

F

-f option ... 111

G

-g option ... 87

H

Hardware initialization function ... 144
Hdwinit function ... 140, 144
her ... 63

I

-i option ... 93
INC78K0S ... 30, 93, 134
Include file ... 193, 198

K

-k option ... 89

L

LANG78K ... 30, 134
-lf option ... 107
-li option ... 108
LIB78K0S ... 30, 134
Librarian ... 21
Library ... 32, 196
Library file ... 32
Library naming rule ... 32
Link directive file ... 143, 151, 195
Linker ... 19
-ll option ... 105
-lt option ... 106
-lw option ... 104

M

mkstup.bat ... 31, 137, 139

N

-ng option ... 87
-no option ... 78
-nq option ... 84
-nr option ... 79, 81, 82, 83
-nv option ... 110
-nz option ... 114

O

-o option ... 78
Object converter ... 20
Object module file ... 119
On-line help file ... 31
Optimization ... 66
Option setting dialog box ... 39

P

-p option ... 88
Parameterfile ... 55
PATH ... 30, 134
Preprocess list file ... 128

Q

-q option ... 84

R

-r option ... 79
-rd option ... 81
repcmul.bat ... 137
repgetc.bat ... 137
repimul.bat ... 137
replmul.bat ... 137
repputc.bat ... 137
repputcs.bat ... 137
reprom.bat ... 31, 137
repselo.bat ... 137
repselon.bat ... 137
Reset vector ... 144
-rk option ... 82
rom.asm ... 31, 141
ROMization ... 68, 136
ROMization processing ... 145, 146
ROMization routine ... 137
-rs option ... 83
Runtime library ... 32, 68

S

s0s*.rel ... 141
-sa option ... 95
-se option ... 100
SETUP.EXE ... 26
sjis ... 30
-sm option ... 117
Source debug ... 196
Source file name ... 193
Stack pointer ... 144
Standard library ... 32, 68
Startup routine ... 32, 68, 136, 139, 140, 195
Startup routine naming rule ... 33
System simulator ... 23

T

-t option ... 112
TMP ... 30, 134

U

-u option ... 92

V

-v option ... 110

W

-w option ... 109
WARNING ... 133

X

-x option ... 102

Y

-y option ... 116

Z

-z option ... 114

*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/>