User's Manual

NEC

CC78K0

C Compiler Ver. 3.50 or Later

Operation

Target Devices

78K0 Series

Document No. U16613EJ1V0UM00 (1st edition)
Date Published June 2003 N CP(K)

© NEC Electronics Corporation 2003
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.
UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.
SPARCstation is a trademark of SPARC International, Inc.
SunOS and Solaris are trademarks of Sun Microsystems, Inc.
HP9000 Series 700 and HP-UX are trademarks of Hewlett-Packard Company.

- **The information in this document is current as of February, 2003. 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).

M8E 02. 11-1

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, please contact the NEC Electronics office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

[GLOBAL SUPPORT]
http://www.necel.com/en/support/support.html

NEC Electronics America, Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 80 820
- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-558-3737

NEC Electronics Shanghai, Ltd.
Shanghai, P.R. China
Tel: 021-6841-1138

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 6253-8311

J03.4

INTRODUCTION

The purpose of this manual is to enable complete understanding of the functions and operation of the CC78K0 (78K0 Series C Compiler).

This manual does not explain how to write CC78K0 source programs. Therefore, before reading this manual, please read **"CC78K0 C Compiler Language User's Manual (U16628E)"** (hereafter called the "Language manual").

[Target Devices]

Software for 78K0 Series microcontrollers can be developed by using the CC78K0. To use this software, the RA78K0 (78K0 Series 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 CC78K0 in microcontroller development.

CHAPTER 2 PRODUCT OVERVIEW AND INSTALLATION

This chapter describes how to install the CC78K0, 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 CC78K0 and presents examples showing the processes from compiling to linking.

CHAPTER 4 CC78K0 FUNCTIONS

This chapter describes optimization methods and ROMization functions in the CC78K0.

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 CC78K0.

CHAPTER 7 USING C COMPILER

This chapter introduces techniques to aid in the skillful use of the CC78K0.

CHAPTER 8 STARTUP ROUTINES

The CC78K0 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 CC78K0.

APPENDICES

The appendices provide and a sample program, a list of the cautions encountered during use, lists of the restrictions related to the CC78K0 and an index.

[How to Read This Manual]

First, those who want to see how to actually use CC78K0, 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.
CC78K0 C Compiler Ver.3.50 or Later	Operation	This document
	Language	U16628E
RA78K0 Assembler Package Ver.3.60 or Later	Operation	U16629E
	Assembly language	U16630E
	Structured assembly language	U11789E
SM78K0 System Simulator	Operation	To be prepared
ID78K Series Integrated Debugger Ver.2.30 or Later	Operation	U15185E
ID78K0-NS Integrated Debugger Ver.2.51 or Later	Operation	U16488E
RX78K0 Real-Time OS	Basics	U11537E
	Installation	U11536E
PM plus Ver.5.10		To be prepared

[Conventions]

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

RTOS:	Real-time OS for 78K0 Series 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).
_ :	Underlining at important locations or in examples is the input character sequence.
Δ :	At least one space
⋮ :	Indicates an omission in a program description
( ) :	Characters between parentheses are as shown (character string).
/ :	Delimiter
\\:	Backslash

[File Name Conventions]

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

(1) Specifying disk file names

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

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

<2> Specifies the name of the root directory.

<3> Specify the subdirectory name.

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

Characters that can be used:

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

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

<4> Primary name

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

Characters that can be used:

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

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

<5> File type

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

Characters that can be used:

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

Example: `C:\nectools32\smp78k0\CC78k0\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.)

CONTENTS

CHAPTER 1 OVERVIEW

The CC78K0 C compiler program translates C source programs written in ANSI-C[Note] or the C language for the 78K0 Series into the machine language for the 78K0 Series.

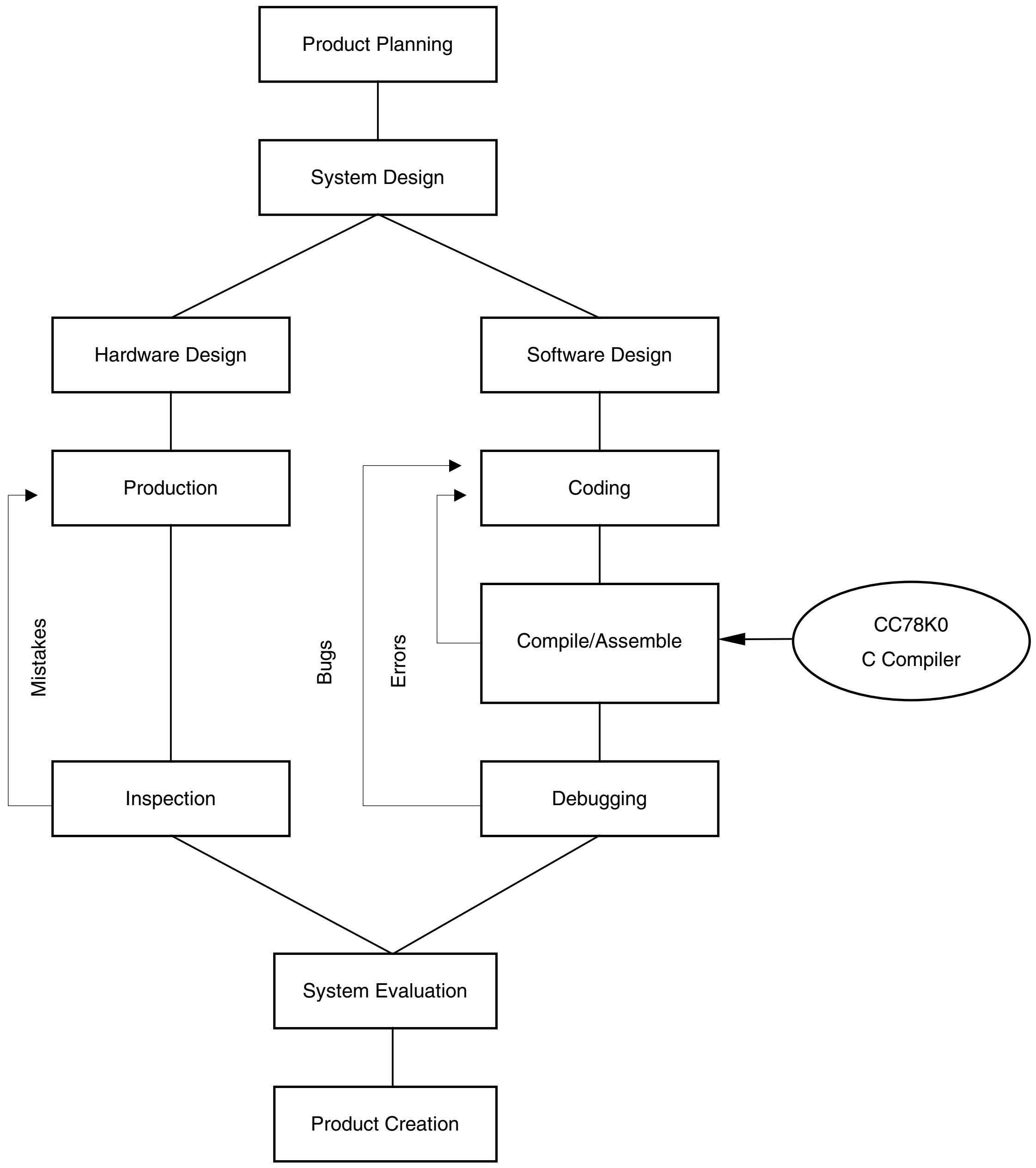
The CC78K0 can be run on Windows™ 98/Me/2000/XP or Windows NT™ 4.0 when using PM plus included in the assembler package for the 78K0 Series. If PM plus is not used, the compiler can be run from the DOS prompt (Windows 98/Me) or command prompt (Windows NT 4.0/2000/XP) (for the Windows version).

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

1.1 Microcontroller Application Product Development and Role of CC78K0

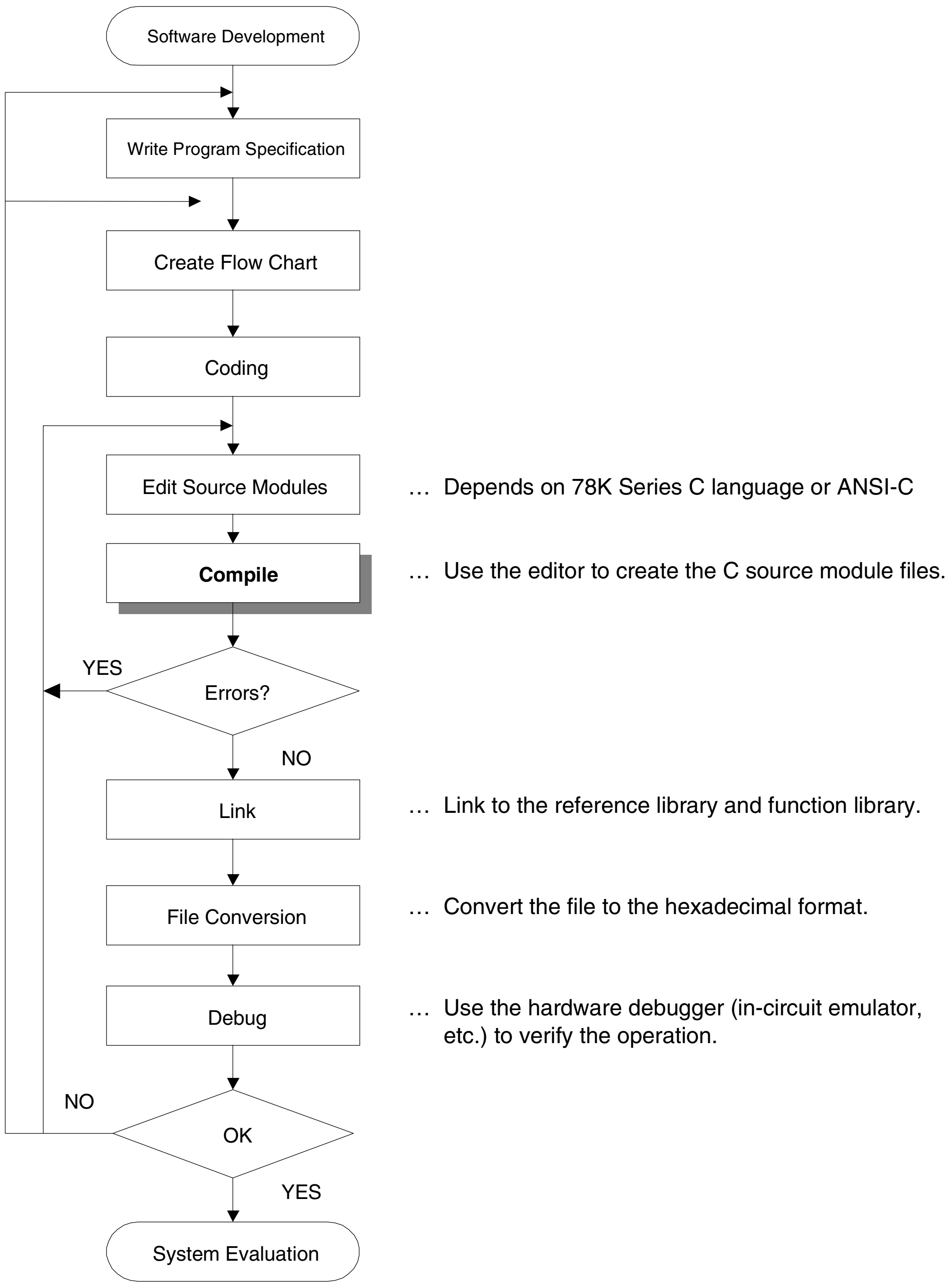
The position of CC78K0 in product development is shown below.

Figure 1-1. Development Process for Microcontroller Application Products



The software development process is shown below.

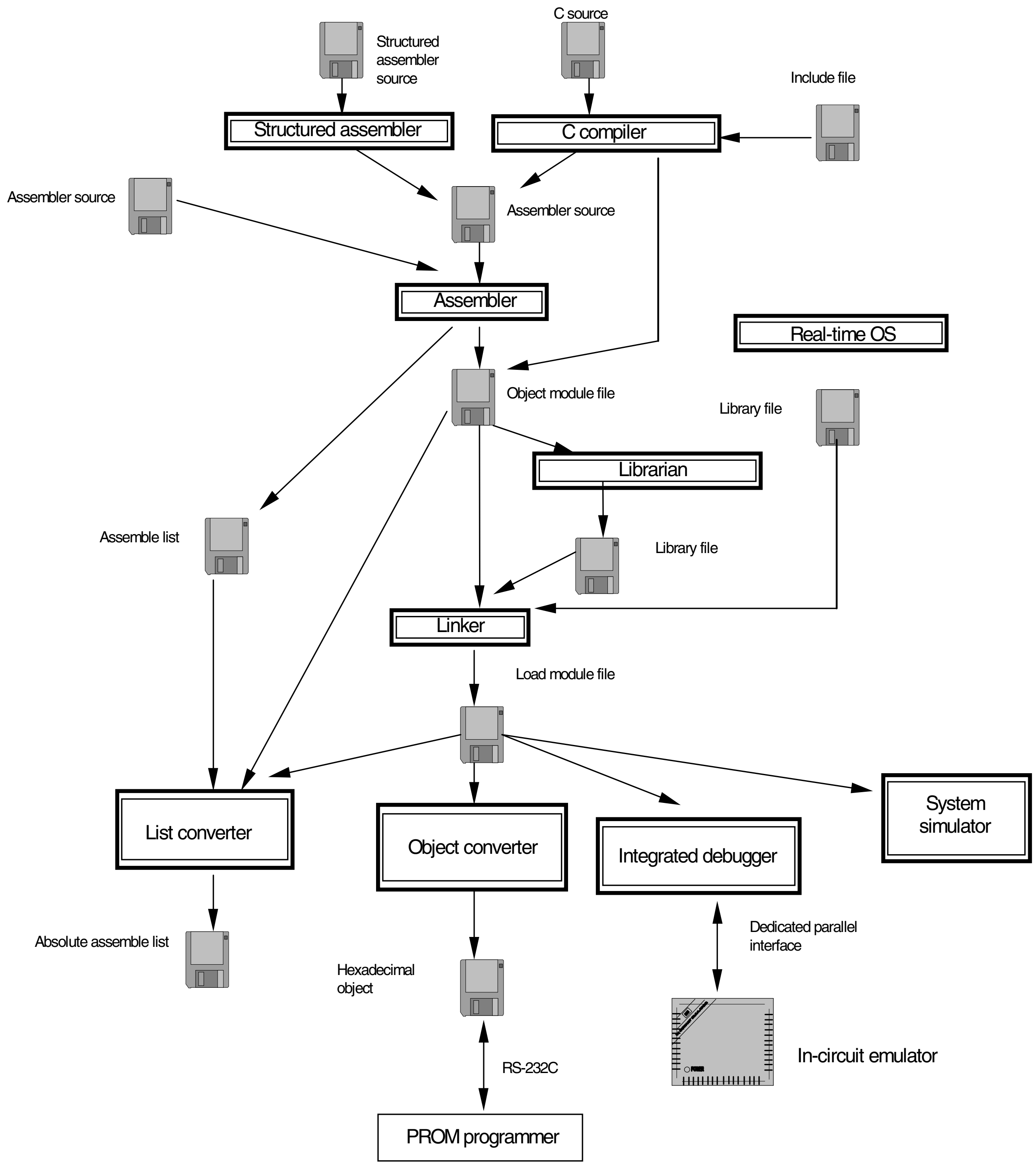
Figure 1-2. Software Development Process


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

1.2 Development Procedure Using CC78K0

The development procedure using CC78K0 is shown below.

Figure 1-3. Program Development Procedure Using CC78K0



1.2.1 Using editor to create source module files

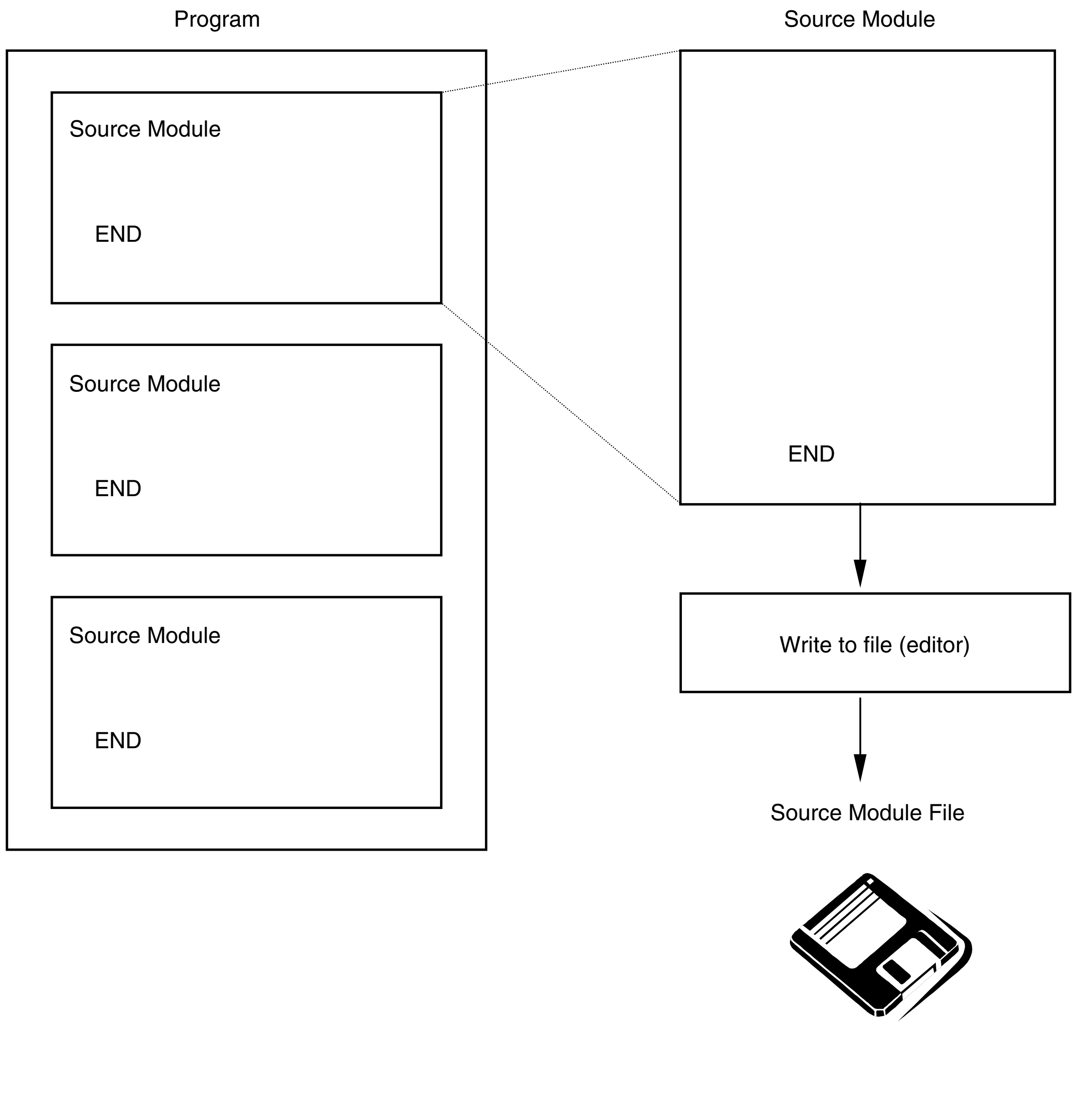
One program is divided into several functional modules.

One 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 CC78K0 input files.

Figure 1-4. Creating Source Module Files

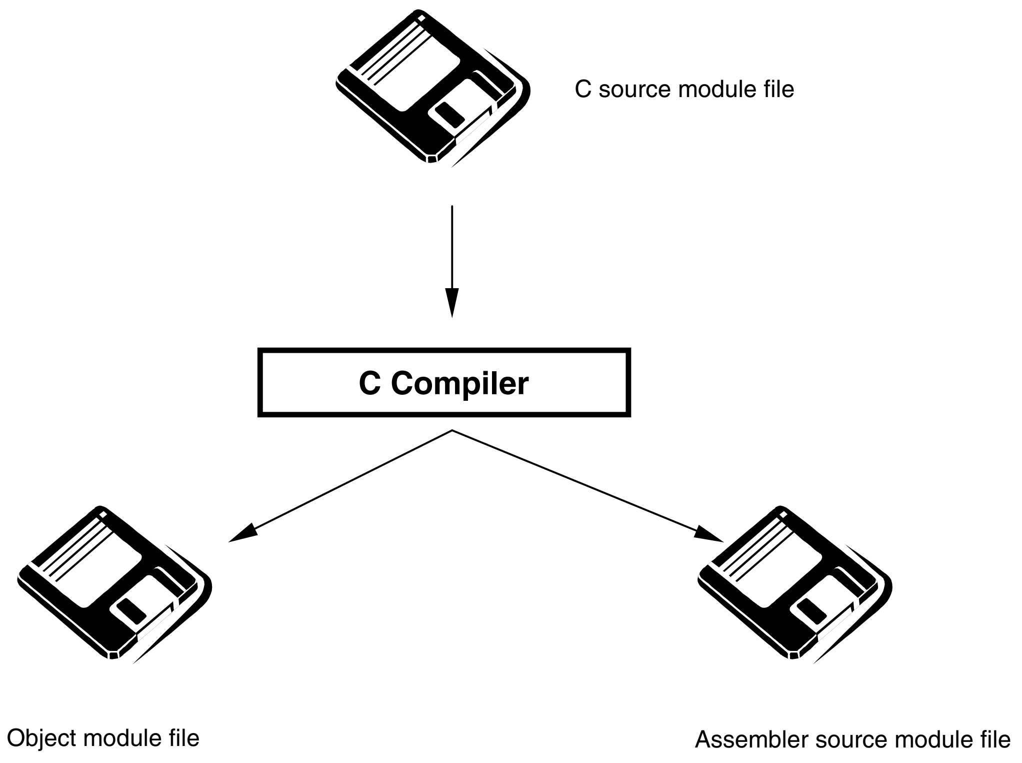


1.2.2 C compiler

The C compiler inputs the C source modules and converts the C language into machine language. If description errors are detected in the C source module, compiling errors are output.

If there are no compiling errors, the object module files are output. To correct and check the programs at the assembly language level, assembler source module files can be output. If you want to output assembler source module files, specify the -A or -SA option in the specification for creating assembler module files when compiling (for information about the options, see **CHAPTER 5 COMPILER OPTIONS**).

Figure 1-5. C Compiler Function



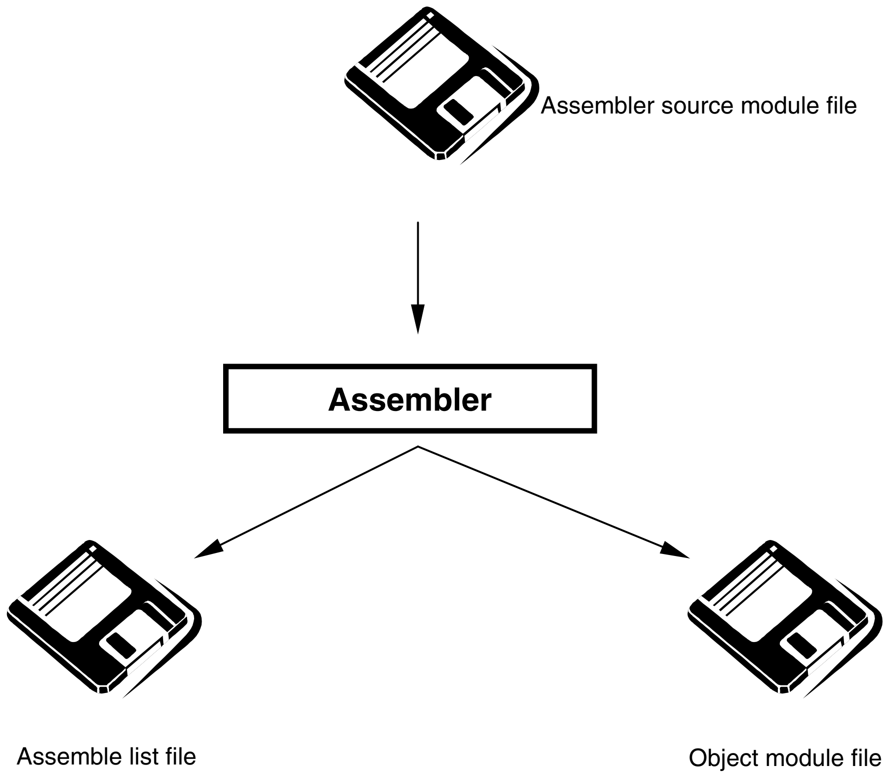
1.2.3 Assembler

Assembling is performed by using the assembler included in the RA78K0 Assembler Package (sold separately).

The assembler is the program that inputs an assembler source module file and translates assembly language into machine language. If description errors were discovered in the source module, the assemble errors are output.

If there are no assemble errors, the output is the object module file that includes machine language information and location information such as at which address each machine language code should be placed in memory. In addition, information during assembly is output as an assemble list file.

Figure 1-6. Assembler Function



1.2.4 Linker

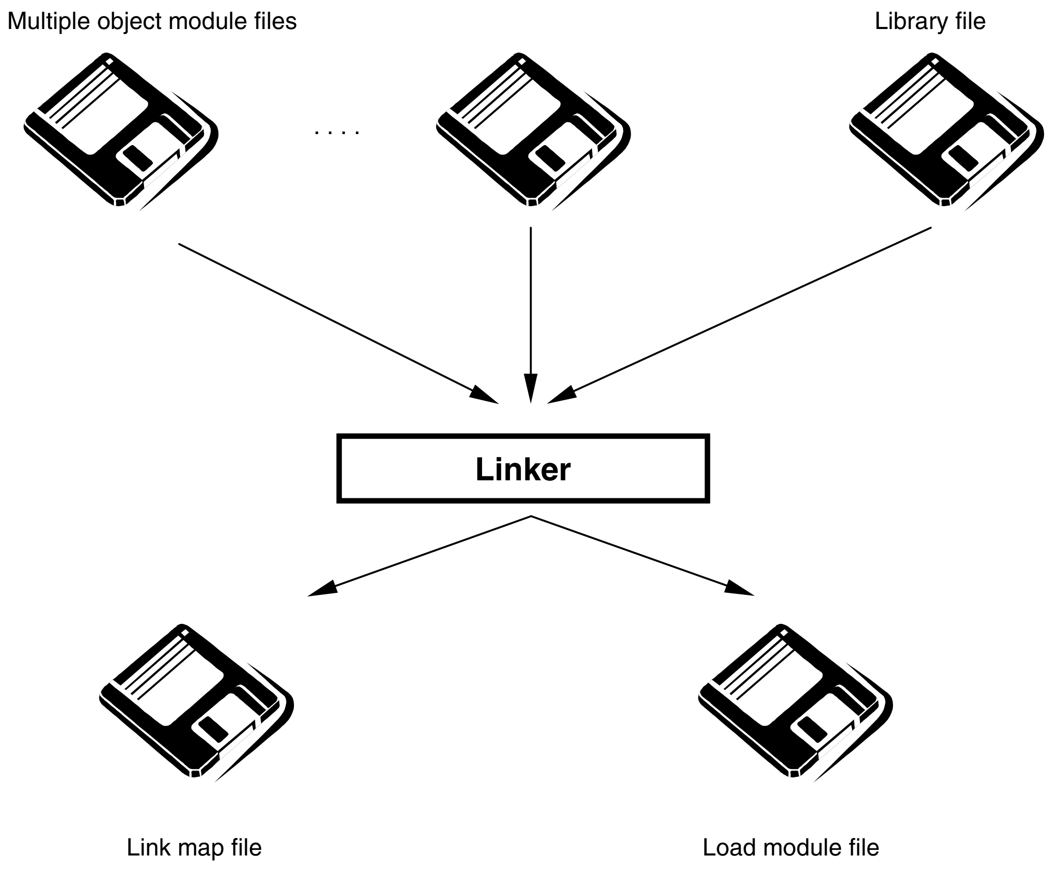
Linking is performed by using the linker included in the RA78K0 Assembler Package (sold separately).

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 one object module, linking must be performed). One load module file is output.

In this case, the linker determines the location addresses of relocatable segments in the input module. This determines the values of relocatable symbols and external reference symbols, and embeds the correct values in the load module file.

The linker outputs the linking information as a link map file.

Figure 1-7. Linker Function



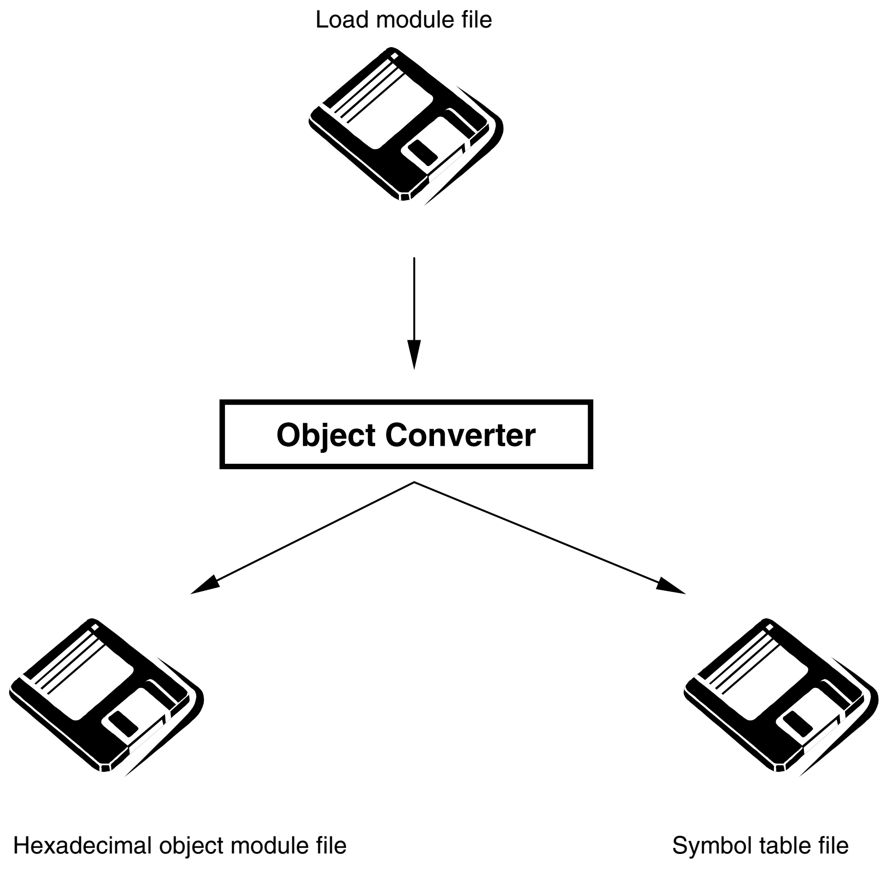
1.2.5 Object converter

The object converter uses the converter included in the RA78K0 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.

Figure 1-8. Object Converter Function



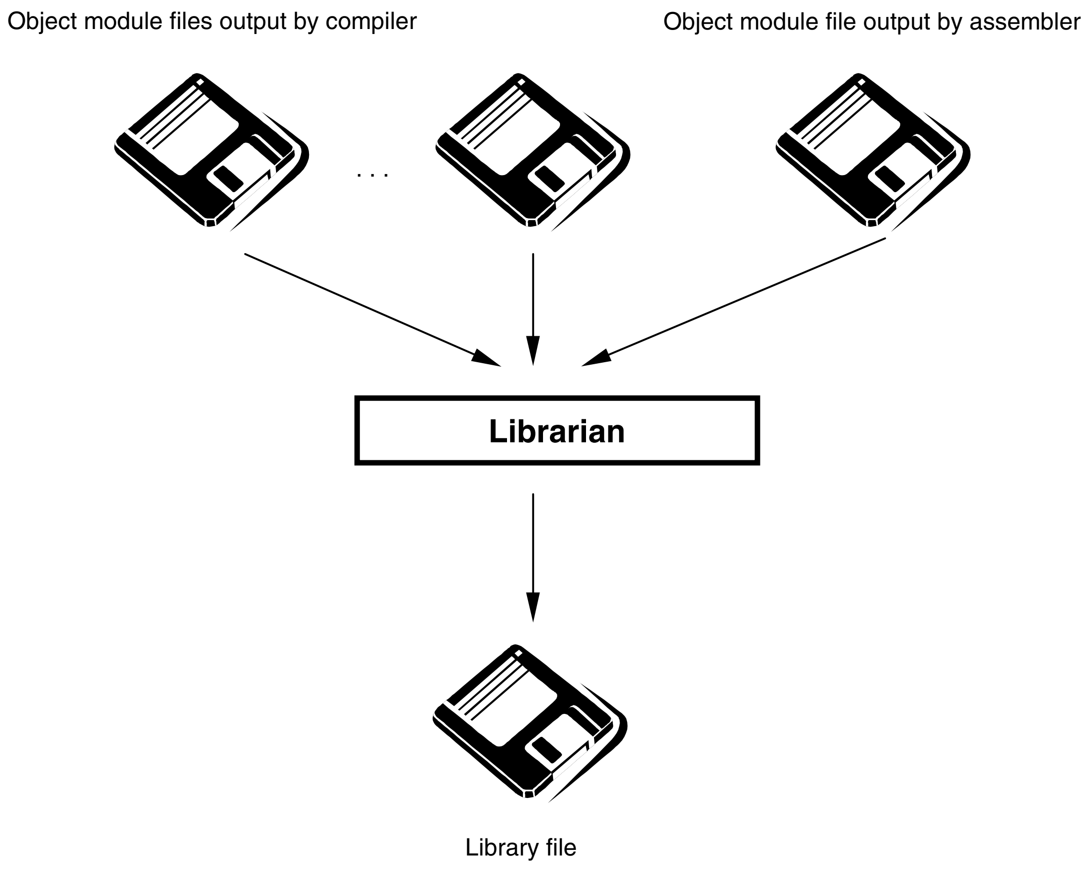
1.2.6 Librarian

Clearly defined modules having a general interface are formed into a library for convenience. By creating the library, many object modules form one 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 one 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 RA78K0 Assembler Package (sold separately).

Figure 1-9. Librarian Function

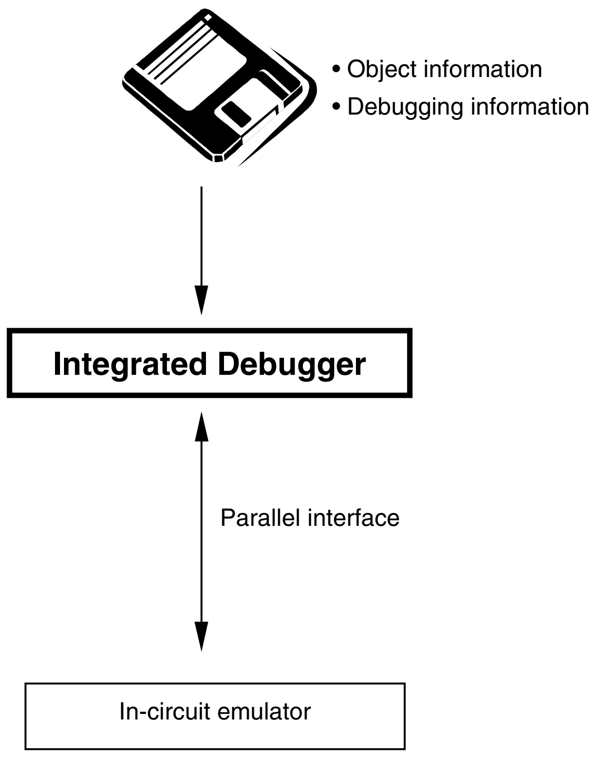

Object module files output by compiler
Object module file output by assembler
. . .
Librarian
Library file

1.2.7 Debugger

Source debugging using a graphical user interface becomes possible by loading the load module files output by the linker into the IE (in-circuit emulator) by using the ID78K0-NS (78K0 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**.

Figure 1-10. Debugger Function

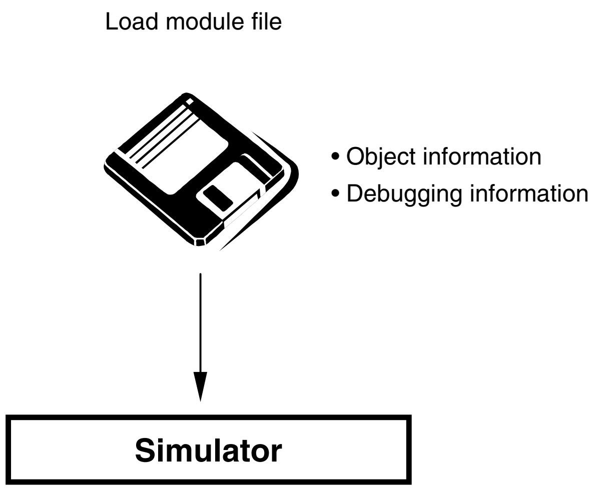


1.2.8 System simulator

Source debugging using a graphical user interface becomes possible by downloading the load module files output from the linker by using the SM78K0 (78K0 Series system simulator).

SM78K0 is software that has the same operating image as the ID78K0-NS and performs simulations on the host machine. In addition to simulating machine instructions in the SM78K0, the on-chip peripherals for the devices and the interrupts can be simulated. Since external parts and procedures are provided to construct dummy target systems, the programs including the operation of the target system are debugged at an early stage independent of hardware development.

Figure 1-11. Simulator Function

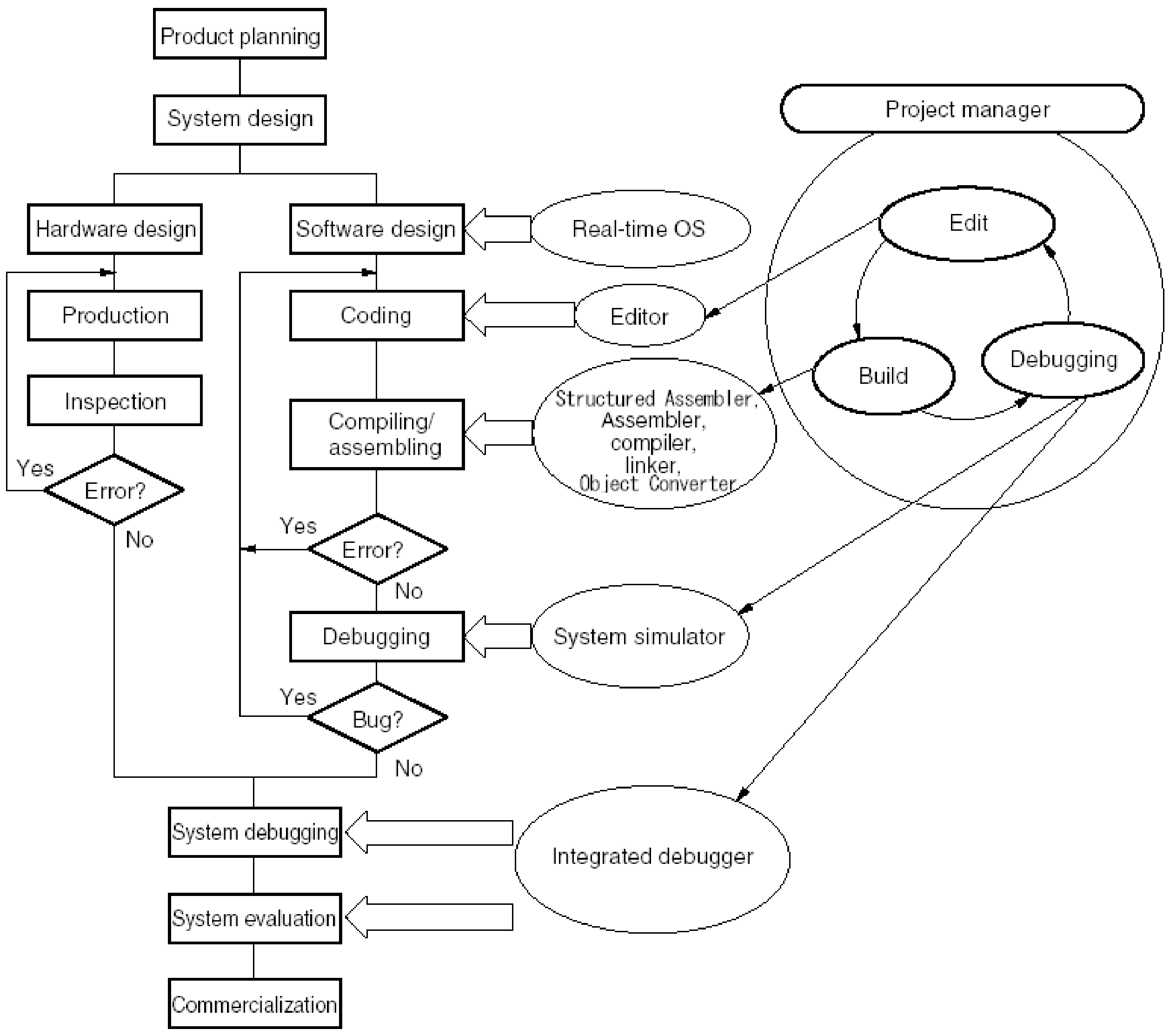


1.2.9 PM plus

PM plus is software that uses the DLL files added to CC78K0 and is able to start CC78K0 on Windows 98/Me/2000/XP or Windows NT 4.0. Editing the source, automatically creating the MAKEFILE, and compiling to linking can be performed from the startup screen of PM plus. Thus, editing to debugging can be performed using GUI images.

PM plus is included to the RA78K0 Assembler Package. The installer for the RA78K0 Assembler Package is used to install and to make the settings. If CC78K0 will be started from PM plus, install the RA78K0 Assembler Package before installing the compiler.

Figure 1-12. PM plus 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 CC78K0 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 Table 2-1.

Table 2-1. Supply Media and Recording Formats for C Compiler

Host Machine	OS	Supply Media	Recording Format
PC-9800 Series	Japanese Windows (98/Me/2000/XP/NT 4.0)Note	CD-ROM	Standard Windows installer supported
IBM PC/ATTM and compatibles	Japanese Windows (98/Me/2000/XP/NT 4.0)Note English Windows (98/Me/2000/XP/NT 4.0)Note		
HP9000 Series 700TM	HP-UXTM (Rel. 10.10 and later)	CD-ROM	cp command
SPARCstationTM Family	SunOSTM (Rel. 4.1.4 and later) SolarisTM (Rel. 2.5.1 and later)		

Note PM plus 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 NT 4.0/2000/XP) if PM plus is not used.

2.2 Installation

2.2.1 Installation of Windows version

The procedure for installing to the host machine the files provided in the CC78K0'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 CC78K0's supply media in the appropriate drive (CD-ROM drive) of the host machine. The setup programs will start automatically. Perform the installation by following the messages displayed in the monitor screen.

Caution If the setup program does not start automatically, execute SETUP.EXE in the CC78K0\DISK1 folder.

(3) Confirmation of files

Using Windows Explorer, etc., check that the files contained in the CC78K0's supply media have been installed to the host machine.
For the details of each folder, refer to **2.4.1 Windows version directory configuration**.

2.2.2 Installation of UNIX version

Install the UNIX version with the following procedure. Installation to /nectools is assumed here.

(1) Login

Log in to the host machine.

(2) Directory selection

Go to the install directory.

```
%cd /nectools
```

(3) Setting of supply media

Set the CD-ROM in the CD-ROM drive and close the tray.

(4) Execute the cp command for the files to copy the files from the CD-ROM (copy the files after checking that the CD-ROM has been set in the CD-ROM drive).

(5) Add /nectools/bin to the environmental variable PATH.

2.3 Installation of Device Files

2.3.1 Installation of Windows version

Use the device file installer to install the device files. The device file installer is installed at the same time as the CC78K0.

2.3.2 Installation of UNIX version

Either specify the directory for device files with the -y option (example: -y/nectools/dev), or copy the device files to a directory with the compiler execution format (example: /nectools/bin).

2.4 Directory Configuration

2.4.1 Windows version directory configuration

The standard directory displayed during installation is “NECTools32” of the Windows system. The configuration under the install directory is as follows. Note that the drive and install directory can be changed during installation. When performing MAKE operation with PM plus, perform installation of tools (CC78K0, RA78K0) to the same drive and directory.

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

Figure 2-1. Directory Configuration

```
NECTools32\
  |
  |------- bin\
  |          cc78k0.exe,                Executable form of compiler
  |          cc78k0p.dll,etc.           PM plus Tools DLL
  |
  |------- inc78k0\
  |          *.h                        Header files for standard library
  |
  |------- lib78k0\Note 2 (For link)
  |          cl0*.lib                   Libraries (runtime and standard libraries)
  |          s0*.rel                    Object files for startup routines
  |
  |------- src\cc78k0\
  |         |
  |         |------ bat\
  |         |         mkstup.bat        Assemble batch files for startup routinesNote 1
  |         |         *.bat
  |         |
  |         |------ src\
  |         |         cstart*.asm       Source files for startup routines
  |         |         rom.asm           Source files for ROMization routines
  |         |         *.asm             Source files for some standard functions
  |         |
  |         |------ lib\Note 2 (For modifications)
  |                   cl0*.lib          Libraries (runtime and standard libraries)
  |                   s0*.rel           Object files for startup routines
  |
  |------- smp78k0\CC78K0\
  |          prime.c                    Source program for verifying installation
  |          sample.bat                 Batch files for verifying installation
  |          readme.doc
  |          lk78k0.dr                  Link directive file for reference
  |
  |------- hlp\
             cc78k0*.hlp                On-line help files
```

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

2. The startup routines and libraries in the lib78k0 directory are identical to those in the src\cc78k0 directory. If a startup routine is modified, change the source in the src\cc78k0 directory. Since assembled files by the batch file are stored in src\cc78k0\lib, copy lib78k0 directory and link.

2.4.2 UNIX version directory configuration

The file organization when the cp command was used for installation to /nectools is shown below.

```
nectools/
  |
  |---- bin/
  |       cc78k0, etc.           Executable form of compiler
  |
  |---- inc78k0/
  |       *.h                    Header files for standard library
  |
  |---- lib78k0/Note (For link)
  |       cl0*.lib               Libraries (runtime and standard libraries)
  |       s0*.lib                Object files for startup routines
  |
  |---- src/cc78k0/
  |       |
  |       |---- bat/
  |       |       mkstup.sh      Assemble batch files for startup routines
  |       |       *.sh
  |       |
  |       |---- src/
  |       |       cstart*.asm    Source files for startup routines
  |       |       rom.asm        Source files for ROMization routines
  |       |       *.asm          Source files for some standard functions
  |       |
  |       |---- lib/Note (For modifications)
  |               cl0*.lib       Libraries (runtime and standard libraries)
  |               s0*.rel        Object files for startup routines
  |
  |---- smp78k0/cc78k0/
          prime.c                Source program for verifying installation
          sample.sh              Batch files for verifying installation
          readme.doc
          lk78k0.dr              Link directive file for reference
```

Note The startup routines and libraries in the lib78k0 directory are identical to those in the src/cc78k0 directory. If a startup routine is modified, change the source in the src/cc78k0 directory. Since assembled files by the batch file are stored in src/cc78k0/lib, copy lib78k0 directory and link.

2.5 Uninstallation Procedure

2.5.1 Uninstallation of Windows version

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 Properties> window

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

(4) Deletion of CC78K0

After selecting "NEC CC78K0 78K/0 C Compiler Vx.xx" from the list of installed software displayed in the <<Install/Uninstall>> tab in the <Add/Remove Programs Properties> 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 **2.4.1 Windows version directory configuration**.

2.5.2 Uninstallation of UNIX version

Delete the files copied in **2.3.2 Installation of UNIX version** with the rm command.

2.6 Environment Settings

2.6.1 Host machine (for PC-9800 Series and IBM PC/AT compatibles)

The CC78K0 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.

DOS prompt in Windows 98/Me
Command prompt in Windows 2000/XP/NT 4.0

2.6.2 Environment variables

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

Table 2-2. Environment Variables

Environment Variable	Description
PATH	Specifies the directory where the executable form of the compiler is located.
TMP	Specifies the directory where temporary files are created (only valid for PC-9800 Series and IBM PC/AT compatibles).
LANG78K	Specifies the kanji code (2-byte code) in the source files. sjis Shift JIS (Default for PC-9800 Series and HP9000 Series 700) euc EUC (Default for SPARCstation) none No 2-byte codes (Default for IBM PC/AT compatibles)
INC78K0	Specifies the directory where the standard header files of the compiler are located. (required only for EWS)
LIB78K0	Specifies the directory where the compiler's libraries are located. (required only for EWS)

Specification Example

For PC-9800 Series and IBM PC/AT compatibles

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

For HP9000 Series 700 and SPARCstation

Example using csh
```
     set path = ($path /nectools/bin)
     setenv LANG78K  euc
     setenv INC78K0  /nectools/inc78k0
     setenv LIB78K0  /nectools/lib78k0/lib
```

Example using sh
```
     PATH = $PATH:/nectools/bin
     LANG78K = euc
     INC78K0 = /nectools/inc78k0
     LIB78K0 = /nectools/lib78k0
     export PATH LANG78K INC78K0 LIB78K0
```

2.6.3 File organization

The table below lists the contents of each directory. The files for PC-9800 Series and IBM PC/AT compatibles are described. The directory structure and file organization are the ones obtained when the installer was used.

Remark Some of the file extensions differ in UNIX.

Table 2-3. File Organization (* = Alphanumeric Symbols)

Directory Name	File Name	Description
BIN\	cc78k0.exe	Compiler
	cc78k0.msg	Message file
	*.hlp	Help files
	*.dll	DLL files
INC78K0\	*.h[Note 1]	Header files for standard library
SRC\CC78K0\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\CC78K0\SRC	cstart*.asm[Note 4] rom.asm *.asm[Note 5]	Source files for startup routines Source files for ROMization routine Source files for standard functions (partial)
HLP	*.hlp	On-line help file

Notes 1. See **10.2 Header Files** in the **Language** manual **(U16628E)**.

2. The batch files in this directory cannot be used in PM plus. Use these batch files only when the source must be revised.

3. Refer to the contents in **Table 8-1 BAT Directory Contents**.

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

5. Refer to the contents in **Table 8-2 SRC Directory Contents**.

Remark A command file (EXE file type) is the file first read into memory when the program starts.

2.6.4 Library files

- These files consist of standard libraries, runtime libraries, and startup routines.

Table 2-4 lists the directory contents.

Table 2-4. Library Files

Directory Name	File Name			File Role
	Normal	Boot Area	Flash Area	
LIB78K0\	cl00.lib cl00r.lib cl00o.lib cl00sm.lib cl00f.lib	cl00.lib cl00r.lib cl00o.lib cl00sm.lib cl00f.lib	cl00e.lib cl00re.lib cl00oe.lib cl00sme.lib cl00fe.lib	Library (runtime and standard libraries)Note 1 (without multiply/divide instruction)
	cl0.lib cl0r.lib cl0o.lib cl0sm.lib cl0f.lib	cl0.lib cl0r.lib cl0o.lib cl0sm.lib cl0f.lib	cl0e.lib cl0re.lib cl0oe.lib cl0sme.lib cl0fe.lib	Library (runtime and standard libraries)Note 1 (with multiply/divide instruction)
	s0.rel s0l.rel s0sm.rel s0sml.rel	s0b.rel s0lb.rel s0smb.rel s0smlb.rel	s0e.rel s0le.rel s0sme.rel s0smle.rel	Object files for startup routines Note 2

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

```
    lib78k0\cl0<i/f><float><pascal><model><flash>.lib
```

<i/f>

None	Use the function interface that conforms to the CC78K0 V3.50 specification (when compile option -ZO is not specified)
o	Use the function interface that conforms to the CC78K0 V2.11 specification (when compile option -ZO is specified)

<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
sm	Static model

<flash>

None	For normal/boot area
e	For flash memory area

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

```
lib78k0\s0<lib><model><flash>.rel
```

<model>

None	Normal model
sm	Static model

<lib>

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

<flash>

None	Normal
b	For boot area
e	For flash memory area

CHAPTER 3 PROCEDURE FROM COMPILING TO LINKING

This chapter uses the CC78K0 and the RA78K0 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 PM plus is described for the PC-9800 Series and IBM PC/AT compatible machines. For other machines, how to execute from the command line is described (for information on installation, see **2.2 Installation**).

3.1 PM plus

This section describes the user interface when the CC78K0 is started in PM plus included in the RA78K0 Assembler Package. If the CC78K0 is started from PM plus, CC78K0P.DLL included in CC78K0 is referenced.

3.1.1 Position of CC78K0P.DLL (tools DLL)

The tools DLL file, such as the CC78K0P.DLL file, is needed to run the Windows version of the 78K0 Series C compiler (CC78K0) from PM plus in Windows 98/Me/2000/XP or Windows NT 4.0.

3.1.2 Execution environment

This environment conforms to PM plus.

The display mode switches between Japanese and English according to the operating system (Windows English version/Japanese version).

3.1.3 CC78K0 option setting menu

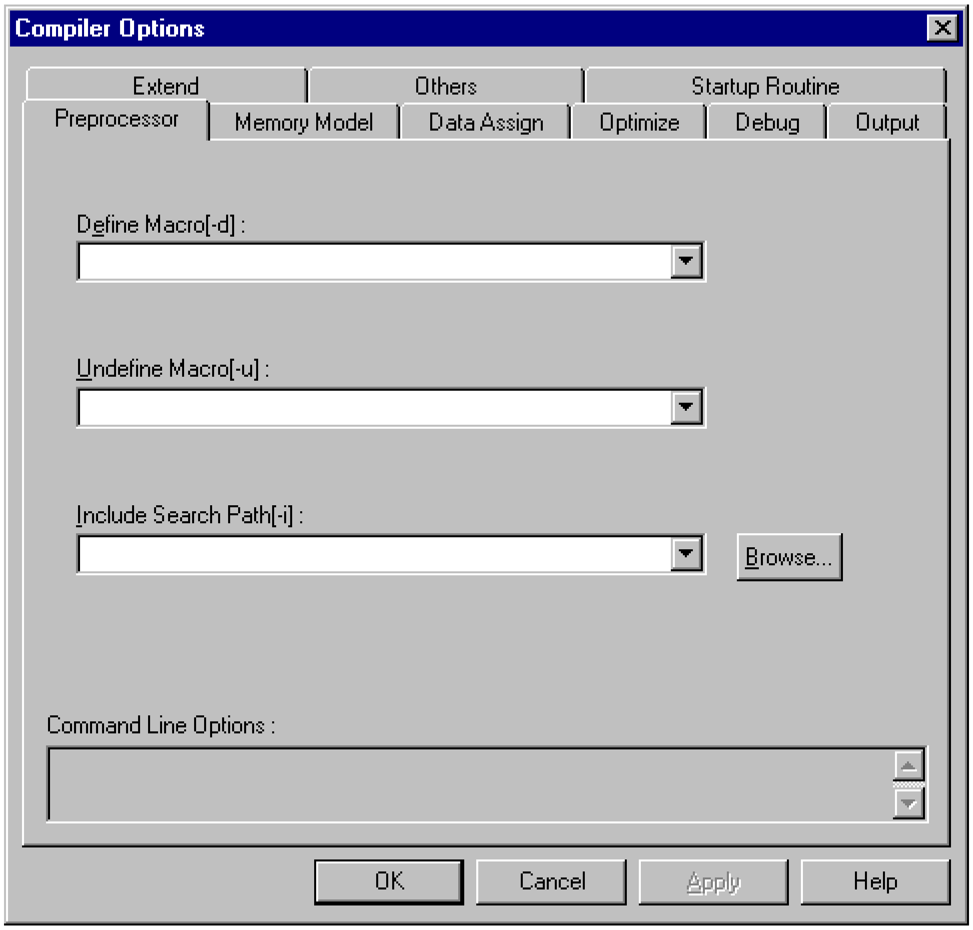
(1) Option menu items

The item “Compiler Options…” is added to the [Tools] menu in PM plus by the tools DLL file included in the CC78K0 C Compiler Package.

(2) Compiler Options dialog box

Select the [Compiler Options…] menu under [Tools] in PM plus to call the option setting function for the tools DLL.

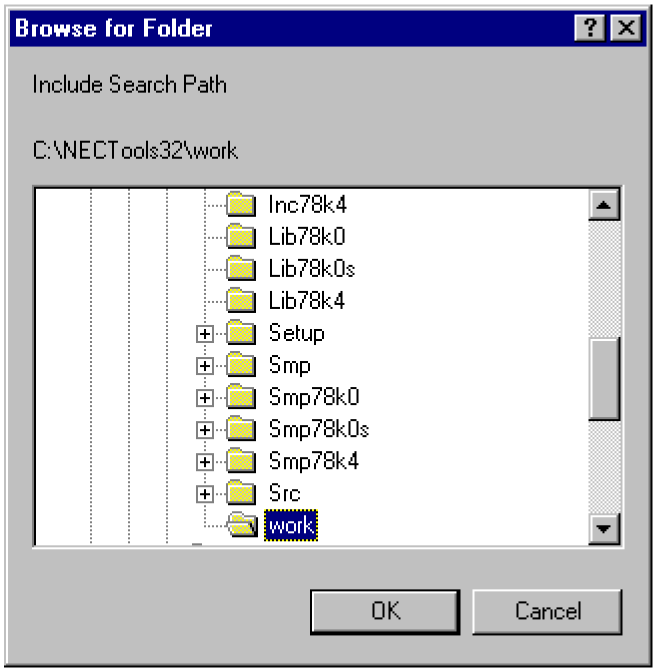
The <Compiler Options> dialog box is shown below.



(3) Browse for Folder dialog box

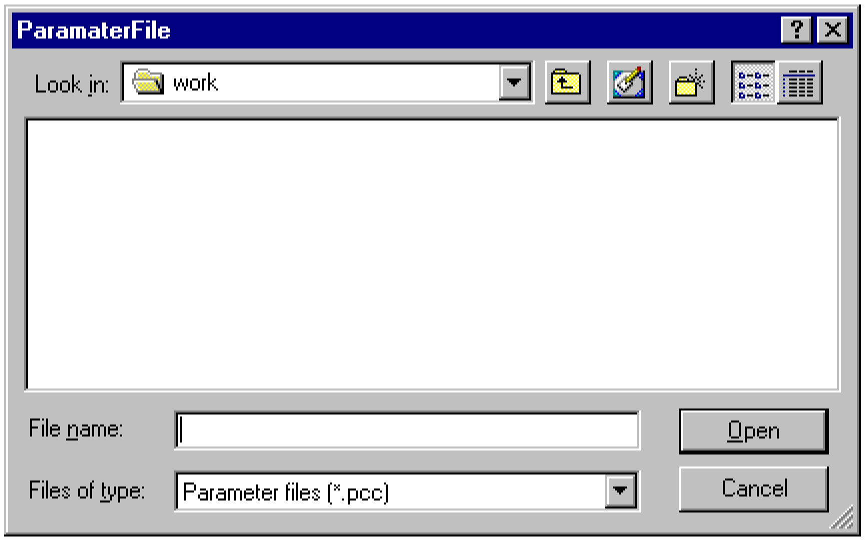
In the <Compiler Options> dialog box, when the Browse... button is clicked for the following path settings, the following dialog box appears. Only the folders can be specified in this dialog box.

- Include file path
- 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



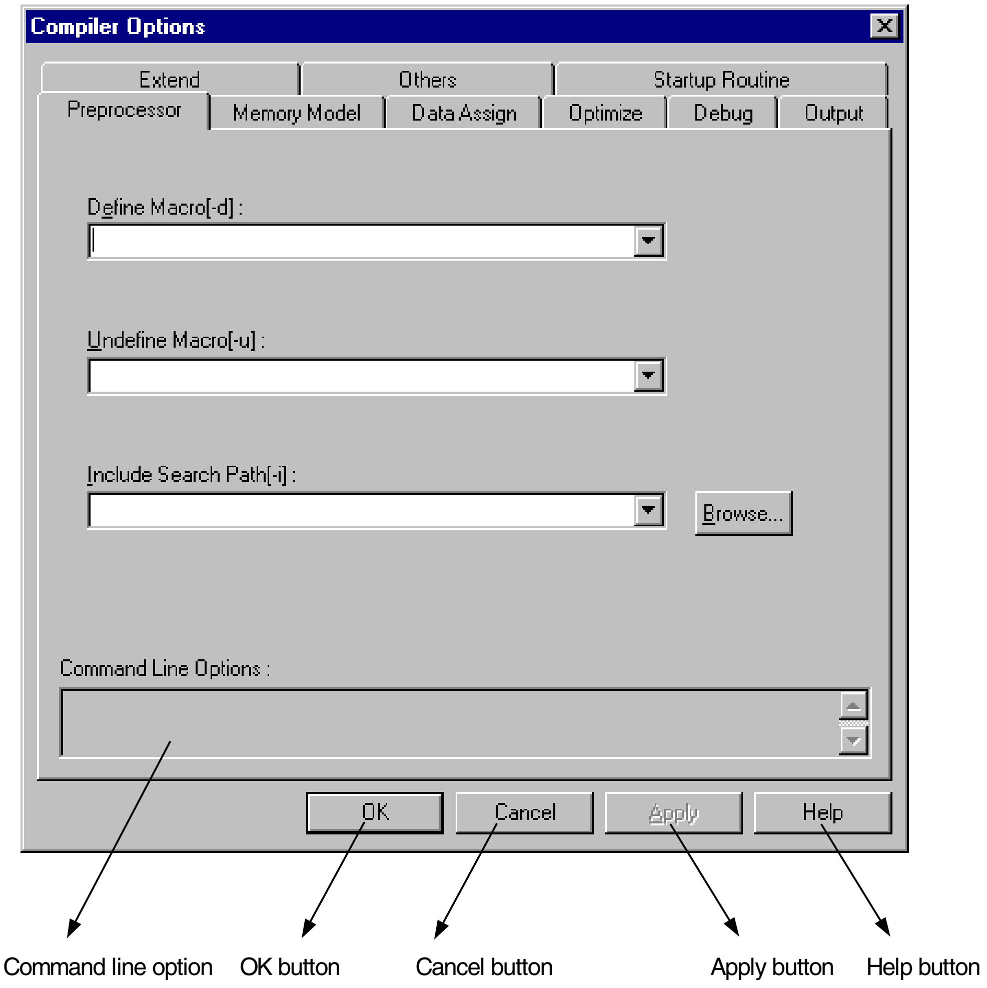
When the Browse... button is clicked in the parameter file specification, the following dialog box appears. This dialog box is as follows.

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



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

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



- **[OK] button**

  The settings edited in this dialog box are set, and the <Compiler Options> dialog box closes. If a source file is selected in this source list, the options are set for this file. If nothing is selected, 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**

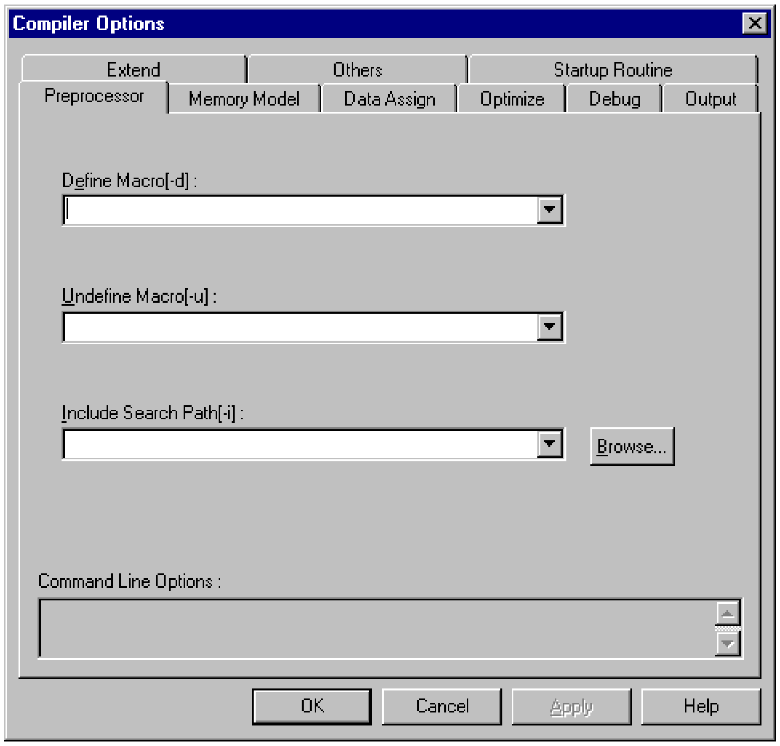
  The help file for this dialog box opens.

- 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 CC78K0 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 ▲▼ button.

- Setting of compiler options
  The compiler options are divided into the following nine options and set respectively. Each setting screen is displayed by clicking the corresponding tab at the top of the dialog box.

  Preprocessor (default)
  Memory Model
  Data Assign
  Optimize
  Debug
  Output
  Extend
  Others
  Startup Routine

(1) Screen when “Preprocessor” is selected

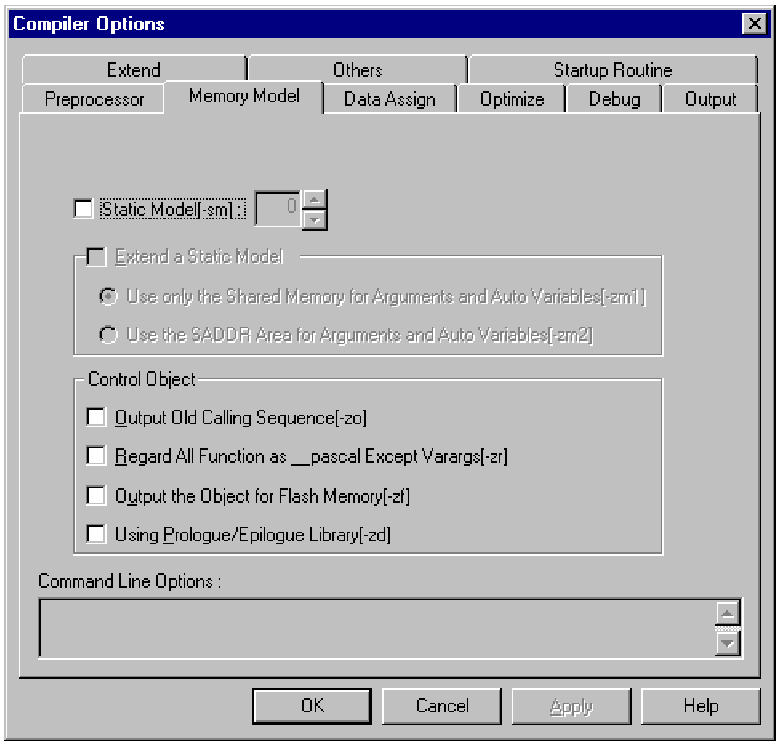


- Define Macro[-d]:
  The macro name and definition name specified by the -D option is input to the combo box.
  For the macro name, multiple macro definitions can be performed at once by delimiting with ‘,’.

- 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 ‘,’.

- Include Search Path[-i]:
  The directory 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 ‘,’.
  The [Browse..] button can also be used for specification.
  Unexisted path cannot be specified.

(2) Screen when “Memory Model” is selected



Static Model[-sm] : 0

Extend a Static Model

Use only the Shared Memory for Arguments and Auto Variables[-zm1]

Use the SADDR Area for Arguments and Auto Variables[-zm2]

Control Object

Output Old Calling Sequence[-zo]

Regard All Function as __pascal Except Varargs[-zr]

Output the Object for Flash Memory[-zf]

Using Prologue/Epilogue Library[-zd]

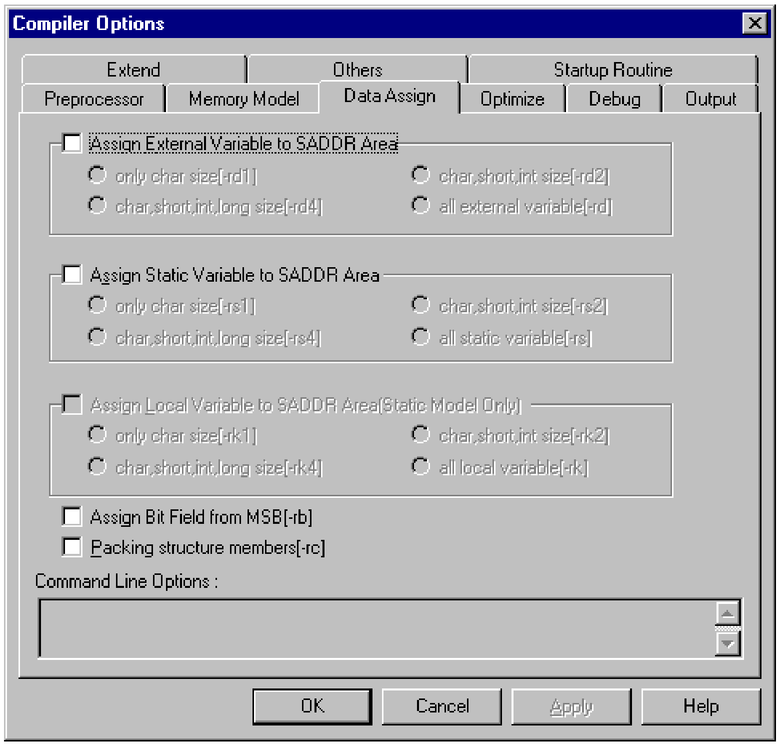
Command Line Options :

- 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
  Output Old Calling Sequence[-zo]
    Select this check box to enable the -ZO option.
  Regard All Function as _ _pascal Except Varargs[-zr]
    Select this check box to enable the -ZR option.
  Output the Object for Flash Memory[-zf]
    Select this check box to enable the -ZF option.
  Using Prologue/Epilogue Library[-zd]
    Select this check box to enable the -ZD option.

(3) Setting screen when “Data Assign” is selected



☐ Assign External Variable to SADDR Area
- only char size[-rd1]
- char,short,int size[-rd2]
- char,short,int,long size[-rd4]
- all external variable[-rd]

☐ Assign Static Variable to SADDR Area
- only char size[-rs1]
- char,short,int size[-rs2]
- char,short,int,long size[-rs4]
- all static variable[-rs]

☐ Assign Local Variable to SADDR Area(Static Model Only)
- only char size[-rk1]
- char,short,int size[-rk2]
- char,short,int,long size[-rk4]
- all local variable[-rk]

☐ Assign Bit Field from MSB[-rb]

☐ Packing structure members[-rc]

Command Line Options :

- Assign External Variable to SADDR Area
  Check the check box to validate the -RD option.
  The type of an external variable to be assigned to the saddr area is selected by checking the radio button.

- Assign Static Variable to SADDR Area
  Check the check box to validate the -RS option.
  The type of a static variable to be assigned to the saddr area is selected by checking the radio button.

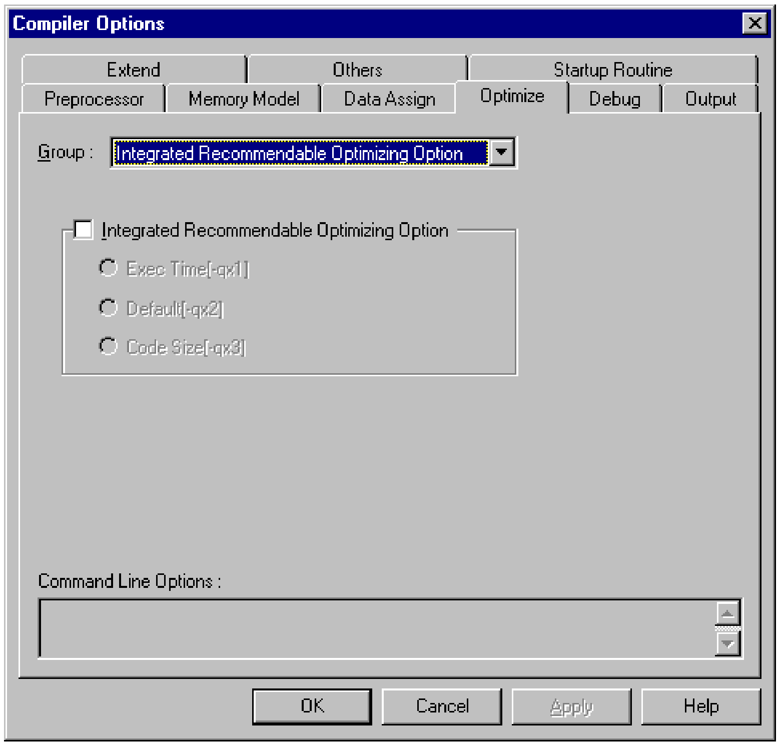
- Assign Local Variable to SADDR Area[Static Model Only]
  Check the check box to validate the -RK option.
  The type of an automatic variable to be assigned to the saddr area is selected by checking the radio button.

- 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) Screen when “Optimize” is selected

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



Group : Integrated Recommendable Optimizing Option

☐ Integrated Recommendable Optimizing Option

○ Exec Time[-qx1]

○ Default[-qx2]

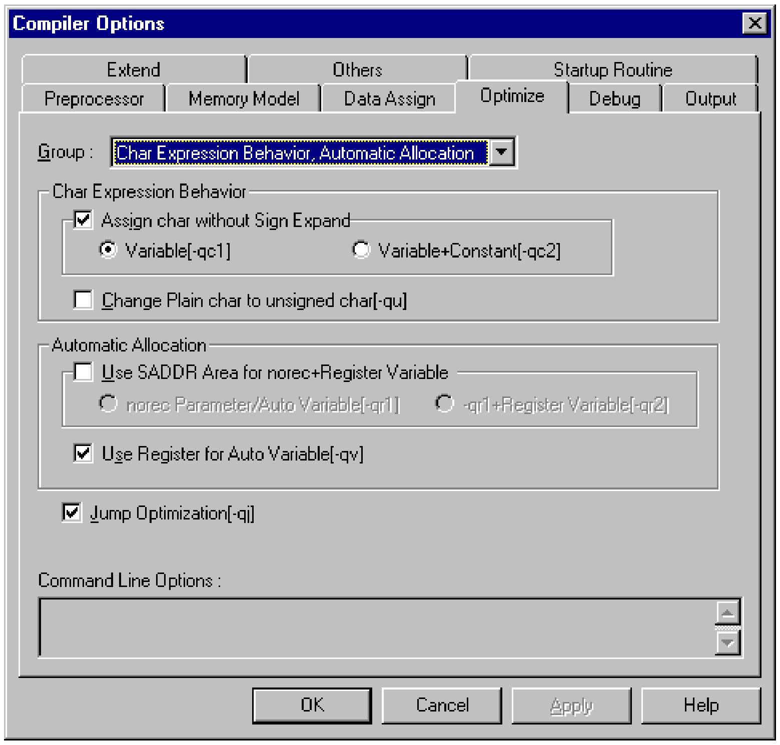
○ Code Size[-qx3]

Command Line Options :

- 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 three 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



Group : Char Expression Behavior, Automatic Allocation

Char Expression Behavior
- [✓] Assign char without Sign Expand
  - (●) Variable[-qc1] ( ) Variable+Constant[-qc2]
- [ ] Change Plain char to unsigned char[-qu]

Automatic Allocation
- [ ] Use SADDR Area for norec+Register Variable
  - ( ) norec Parameter/Auto Variable[-qr1] ( ) -qr1+Register Variable[-qr2]
- [✓] Use Register for Auto Variable[-qv]

[✓] Jump Optimization[-qj]

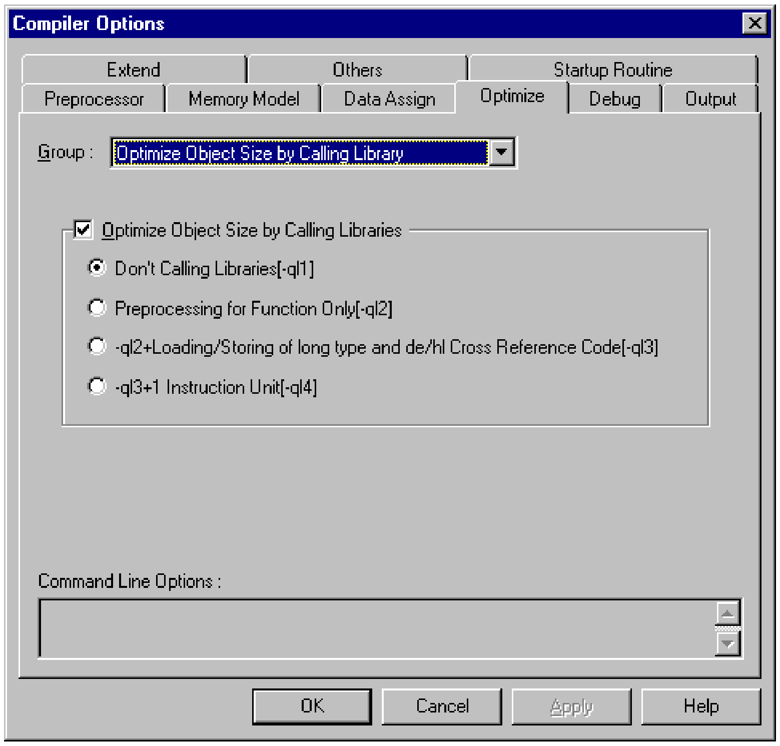
Command Line Options :

- 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
    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



Group : Optimize Object Size by Calling Library

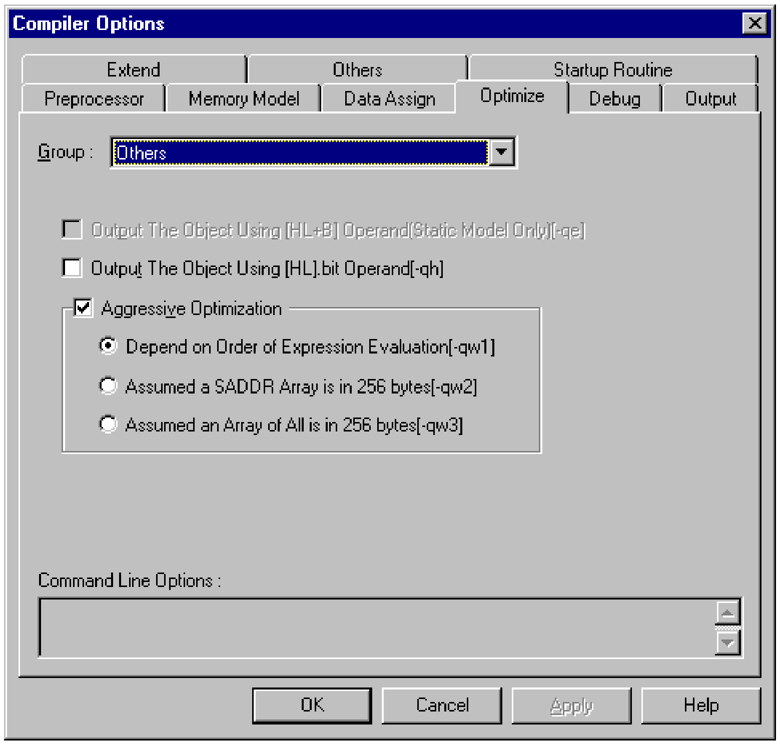
[✓] Optimize Object Size by Calling Libraries

- ● Don't Calling Libraries[-ql1]
- ○ Preprocessing for Function Only[-ql2]
- ○ -ql2+Loading/Storing of long type and de/hl Cross Reference Code[-ql3]
- ○ -ql3+1 Instruction Unit[-ql4]

Command Line Options :

- 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. A device may have some limitations in operation if -QL4 (max.) is specified.

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



Group : Others

[ ] Output The Object Using [HL+B] Operand(Static Model Only)[-qe]

[ ] Output The Object Using [HL].bit Operand[-qh]

[✓] Aggressive Optimization

[✓ Depend on Order of Expression Evaluation[-qw1]]

Assumed a SADDR Array is in 256 bytes[-qw2]

Assumed an Array of All is in 256 bytes[-qw3]

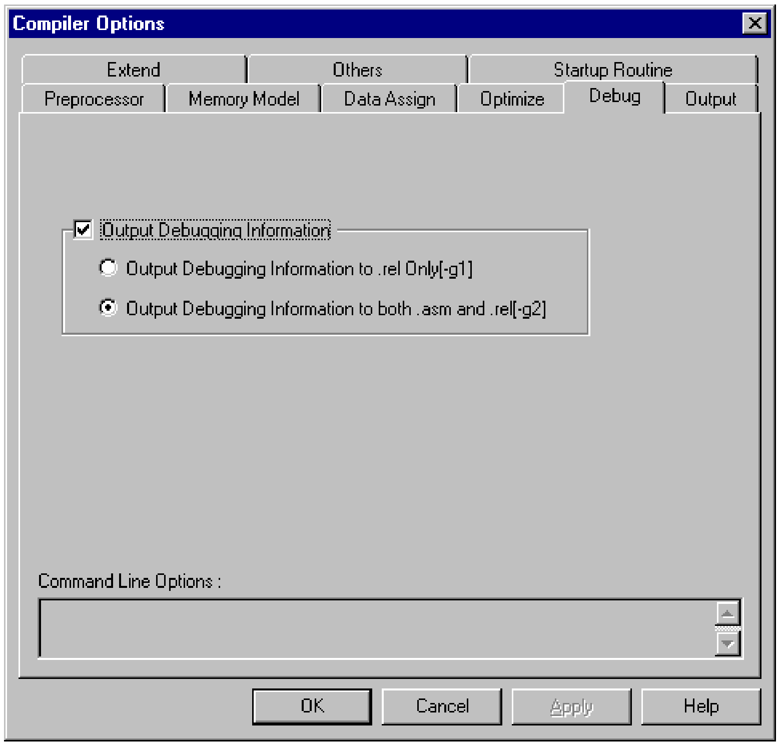
Command Line Options :

- Output The Object Using [HL+B] Operand (Static Model Only)[-qe]
  Check this check box to perform code output that [HL+B] is used for operand.

- Output The Object Using [HL], bit Operand[-qh]
  Check this check box to perform code output that [HL].bit is used for operand.

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

(5) Screen when “Debug” is selected



[✓] Output Debugging Information

○ Output Debugging Information to .rel Only[-g1]

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

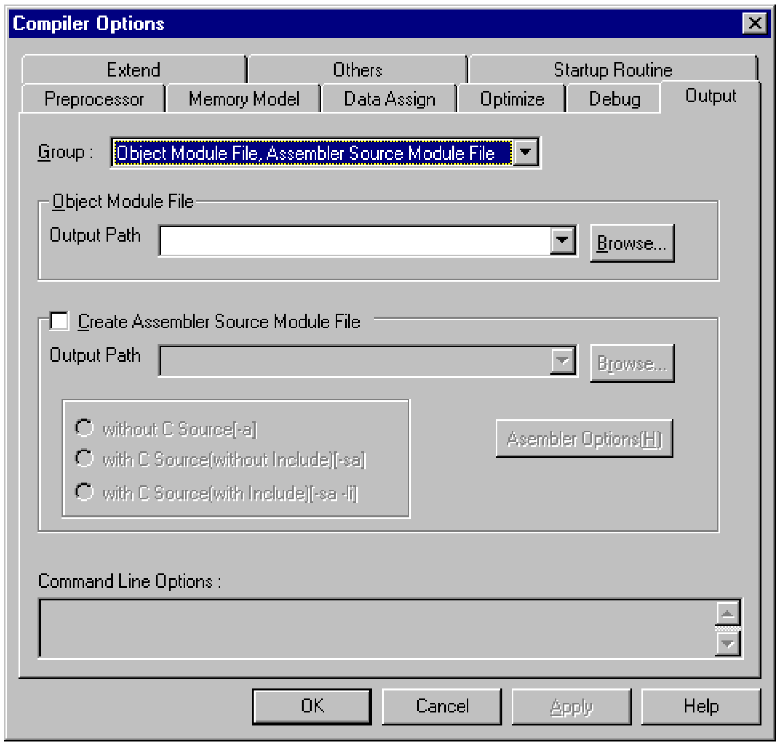
Command Line Options :

- Output Debugging Information

  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 plus option, it is not possible to perform settings in the <Debug> dialog box, and debug information is not output.

(6) Screen when “Output” is selected

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



Group : Object Module File, Assembler Source Module File

☐ Create Assembler Source Module File

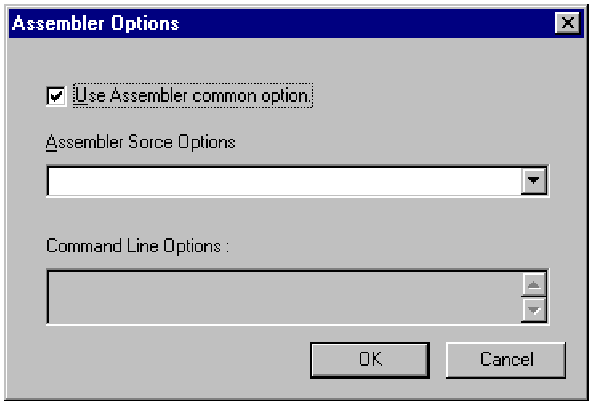
- Object Module File
  To specify an object module file output path, input the path name in the combo box. Specification is also possible using the [Browse…] button.
  When universal options are specified in PM plus, processing is always performed assuming that the path name is specified.
  When the 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 plus, 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.



- 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 of the compiler, input a character string including the option name in the combo box.

  Past inputs can be selected by clicking the ▼ button 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.

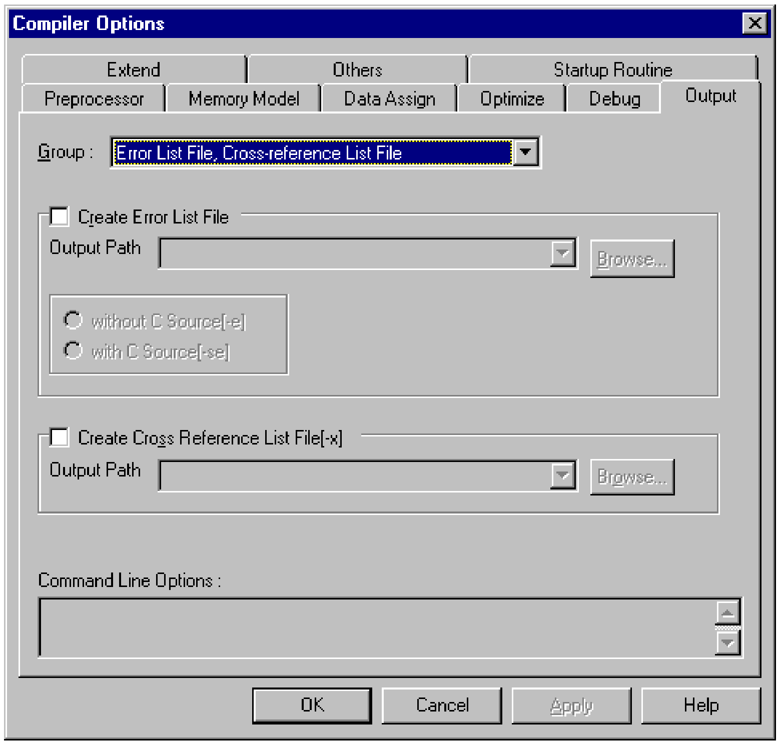
  The option character strings that are currently set are displayed.

  If the character strings do not all fit in the box, they can be viewed by scrolling with the ▲▼ button.

  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



Group : Error List File, Cross-reference List File

☐ Create Error List File
Output Path
○ without C Source[-e]
○ with C Source[-se]

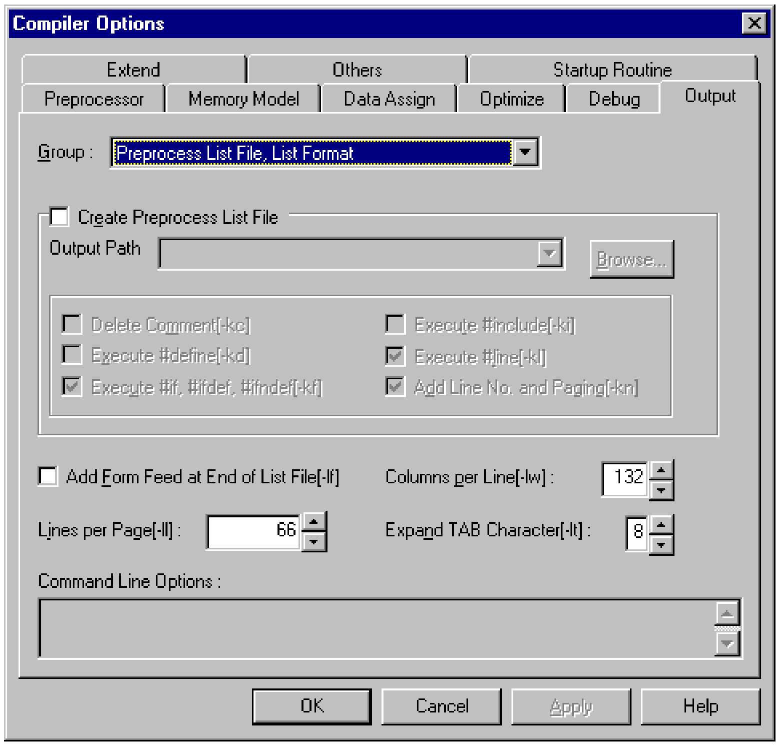
☐ Create Cross Reference List File[-x]
Output Path

Command Line Options :

- Create Error List File
  Select this check box to 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



Compiler Options

Extend | Others | Startup Routine

Preprocessor | Memory Model | Data Assign | Optimize | Debug | Output

Group : Preprocess List File, List Format

☐ Create Preprocess List File

Output Path Browse...

☐ Delete Comment[-kc] ☐ Execute #include[-ki]

☐ Execute #define[-kd] ☑ Execute #line[-kl]

☑ Execute #if, #ifdef, #ifndef[-kf] ☑ Add Line No. and Paging[-kn]

☐ Add Form Feed at End of List File[-lf] Columns per Line[-lw] : 132

Lines per Page[-ll] : 66 Expand TAB Character[-lt] : 8

Command Line Options :

OK Cancel Apply Help

- 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.

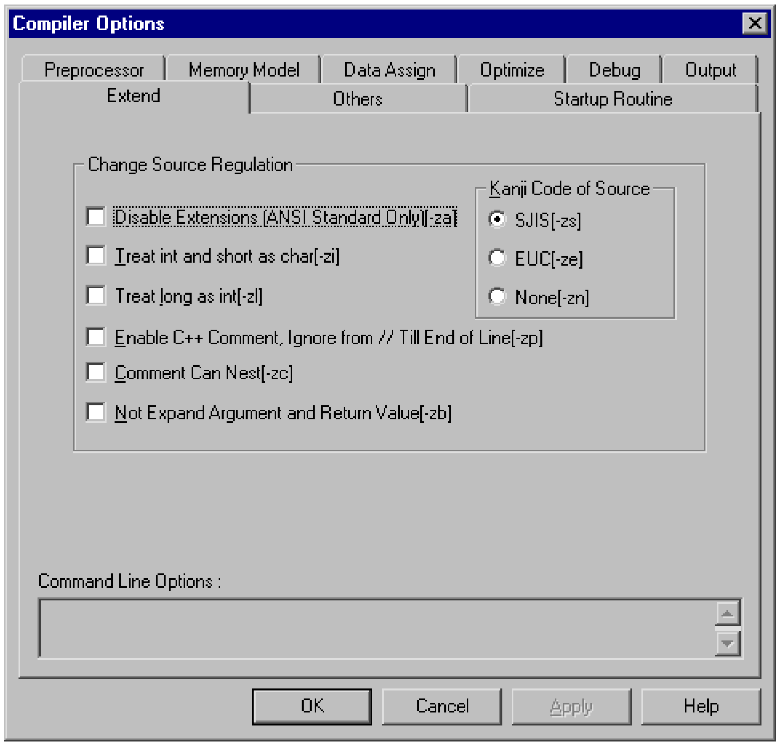
- List setting
  The list is output in the following format specified when the output option of each list is set.

  Columns per Line[-lw]:
  Specifies the number of characters in one line by using the -LW option. To increase/decrease the number of characters in the box, click ▲▼ button.

  Lines per Page[-ll]:
  Specifies the number of lines in one page by using the -LL option. To increase/decrease the number of characters in the box, click ▲▼ 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 ▲▼ button.

(7) Screen when “Extend” is selected



Change Source Regulation

- [ ] Disable Extensions [ANSI Standard Only][-za]
- [ ] Treat int and short as char[-zi]
- [ ] Treat long as int[-zl]
- [ ] Enable C++ Comment, Ignore from // Till End of Line[-zp]
- [ ] Comment Can Nest[-zc]
- [ ] Not Expand Argument and Return Value[-zb]

Kanji Code of Source

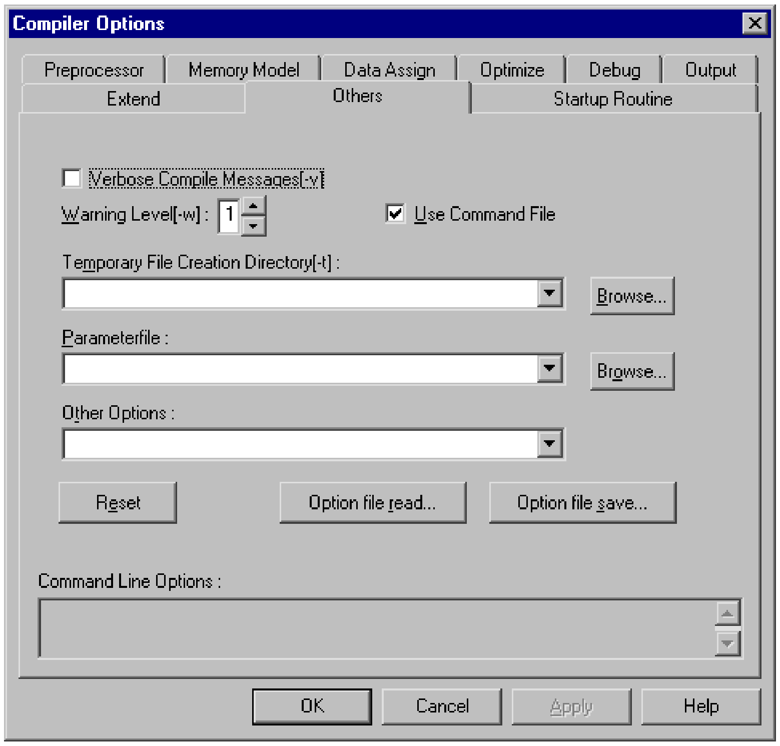
[✓ SJIS[-zs]] EUC[-ze] None[-zn]

Command Line Options :

- Change Source Regulation

  Disable Extensions (ANSI Standard Only)[-za]
  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) Screen when “Others” is selected



☐ Verbose Compile Messages[-v]
Warning Level[-w] : 1
☑ Use Command File

- Verbose Compile Messages[-v]
  Select this check box to enable the -V option.

- Warning Level[-w]:
  Use the ▲▼ button to change the -W option level.

- Temporary File Creation Directory[-t]:
  Input the directory in which to store the temporary files specified with the -T option in the combo box.

- Parameterfile:
  Input the parameter file name specified with the -F option in the combo box.
  Past inputs can be selected by clicking the ▼ button at the right of the combo box.

- Other 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 ▼ button at the right of the combo box.

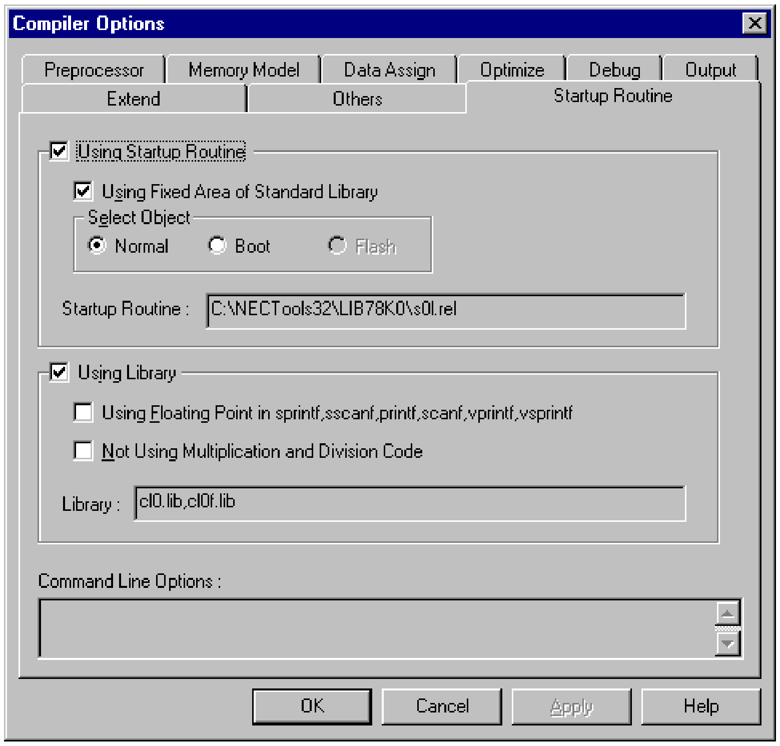
- [Reset] button
  Clicking this button sets the default option settings.

- [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.

- Use 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.

(9) Screen when “Startup Routine” is selected



[✓] Using Startup Routine

[✓] Using Fixed Area of Standard Library

Select Object: [● Normal] ○ Boot ○ Flash

Startup Routine : C:\NECTools32\LIB78K0\s0l.rel

[✓] Using Library

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

[ ] Not Using Multiplication and Division Code

Library : cl0.lib,cl0f.lib

Command Line Options :

<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.

  - Select Object
    Select the desired startup routine for the normal, boot, or flash area by selecting the corresponding radio button.

    If the [Output the Object for Flash Memory[-zf]] check box under the <Memory Model> tab is not selected, the startup routine for the normal or boot areas can be selected, and if the check box is selected, only the startup routine for the flash area can be selected.

  - Startup Routine:
    Indicates the file name of the startup routine to be used.

- Using Library
  Select this check box to use the standard library provided for this C compiler.

  - 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]:], [Output Old Calling Sequence[-zo]], 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.

  - Not Using Multiplication and Division Code
    Select this check box to use products that do not have multiply and divide instructions.

  - Library:
    Displays the file name of the library to be used.

3.2 Procedure from Compiling to Linking (When Not Using Flash Memory Self Rewrite Mode)

3.2.1 MAKE from PM plus

The MAKE method using PM plus with a PC-9800 Series or IBM PC/AT compatible is described below.

PM plus is a software program used for the integrated management of tools as the core of the development environment. Using PM plus 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.

3.2.2 Starting up PM plus

When a development tool packages are correctly installed, the [NECTools32] menu is created in the Programs folder displayed from the $\boxed{\text{Start}}$ button, and PM plus and other programs are registered in this menu.

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

3.2.3 Creating project

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

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 plus Ver. 5.10 User's Manual (U16569E)**.

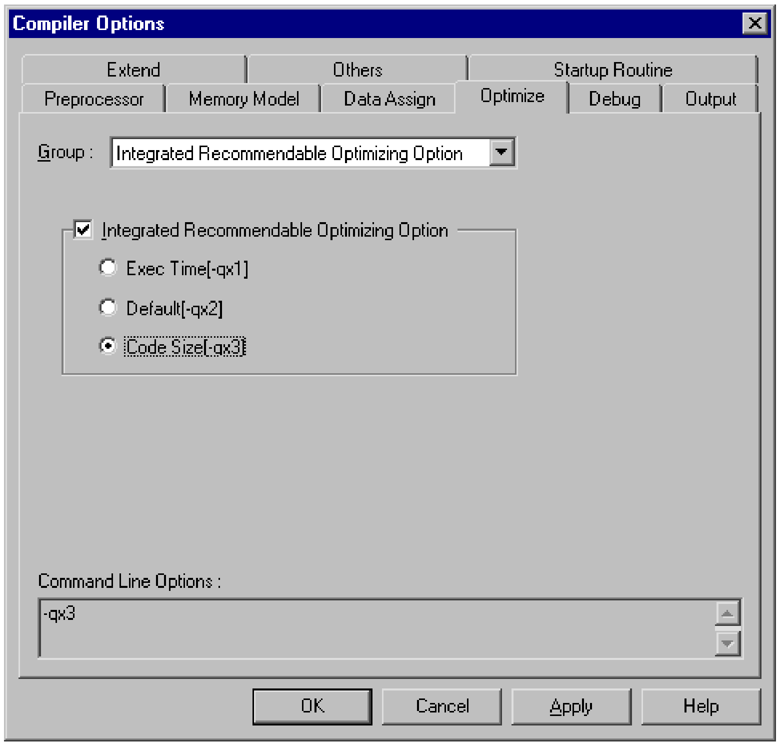
3.2.4 Setting compiler and linker options

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.

Figure 3-1. Selection of Optimize Options



Group : Integrated Recommendable Optimizing Option

[✓] Integrated Recommendable Optimizing Option
- ○ Exec Time[-qx1]
- ○ Default[-qx2]
- ● Code Size[-qx3]

Command Line Options :

```
-qx3
```

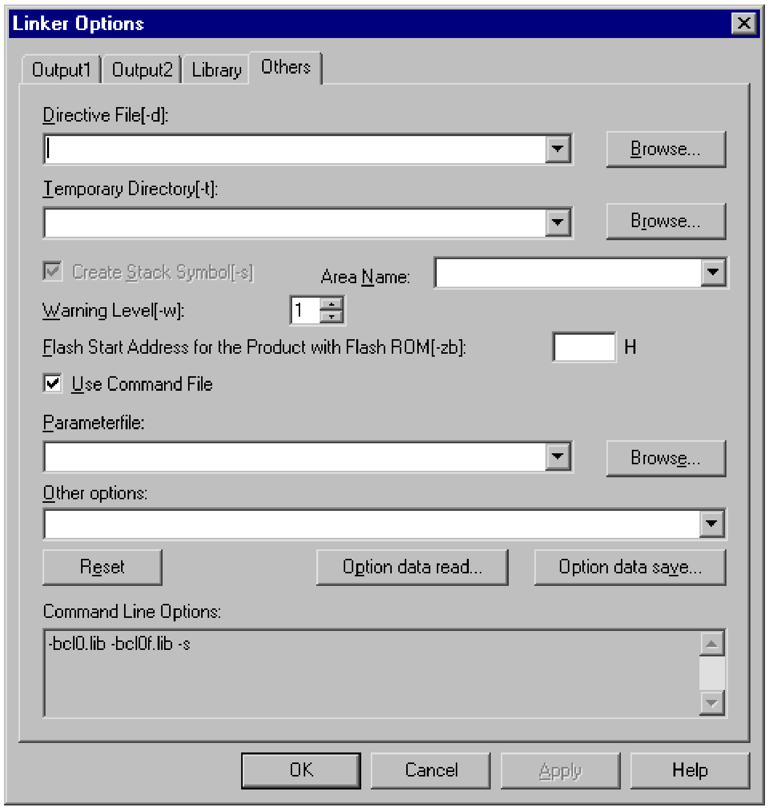
If "Using Startup Routine" is selected in the <<Startup Routine>> tab of the <Compiler Options> dialog box, the standard startup routine for this compiler gets linked before all sources (not displayed to the <Linker Options> dialog box).

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.

Figure 3-2. Linker Options Dialog Box



Linker Options

Output1 | Output2 | Library | Others

Directive File[-d]:

Temporary Directory[-t]:

☑ Create Stack Symbol[-s]   Area Name:

Warning Level[-w]: 1

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

☑ Use Command File

Parameterfile:

Other options:

Command Line Options:

-bcl0.lib -bcl0f.lib -s

3.2.5 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 button on the tool bar. PM plus MAKE is started up by the automatically created MAKE file.

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

Caution The contents displayed in the <Output> window during build are saved as the "Project file name + .plg" file name to the project directory.

3.2.6 Compiling to linking in command line (for DOS prompt and EWS)

(1) When parameter file is not used

The command below is used to start the CC78K0, 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] CC78K0 [Δ option] Δ C  source name [Δ option]
> [path name] RA78K0 [Δ option] Δ assembler source name [Δ option]
> [path name] LK78K0  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, sscanf, printf, scanf, 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, sscanf, printf, scanf, 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, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points

1. **User program library file (specified with -B option)**
2. **Library file attached to C compiler (specified with -B option)**
3. **Floating point library file attached to C compiler (specified with -B option)**

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

1. **User program library file (specified with -B option)**
2. **Floating point library file attached to C compiler (specified with -B option)**
3. **Library file attached to C compiler (specified with -B option)**

(Specification order of other files)

1. **Object file of startup routine attached to CC78K0**
2. **Object module file of user program**

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

```
C>cc78k0 -c054 s1.c -e -a -iC:\nectools32\inc78k0 –yC:\nectools32\dev
C>ra78k0 -c054 s2.asm -e -yC:\nectools32\dev
C>lk78k0 s01.rel s1.rel s2.rel -bC:\nectools32\lib78k0\cl0.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 **RA78K0 Assembler Package Operation User's Manual (U16629E).**

(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.

Caution Parameter files cannot be specified by means of the option setting of PM plus.

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

```
>[path name]CC78K0 Δ -F parameter file name
>[path name]RA78K0 Δ -F parameter file name
>[path name]LK78K0 Δ -F parameter file name
```

The following shows a usage example.

Example

```
C>cc78k0  -Fpara.pcc
C>ra78k0  -Fpara.pra
C>lk78k0  -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)

```
-c054 s1.c -e -a -iC:\nectools32\inc78k0 -yC:\nectools32\dev
```

(Contents of para.pra)

```
-c054 s2.asm -e -yC:\nectools32\dev
```

(Contents of para.plk)

```
s01.rel s1.rel s2.rel -bC:\nectools32\lib78k0\cl0.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 **RA78K0 Assembler Package Operation User's Manual (U16629E).**

3.3 Compiling to Linking (When Using Flash Memory Self Rewrite Mode)

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

3.3.1 Compiling to linking via PM plus

PM plus is used on the PC-9800 Series and IBM PC/AT compatibles to illustrate the MAKE technique.
Be sure to execute compiling to linking in the following order.

(1) Compiling to linking program for boot area

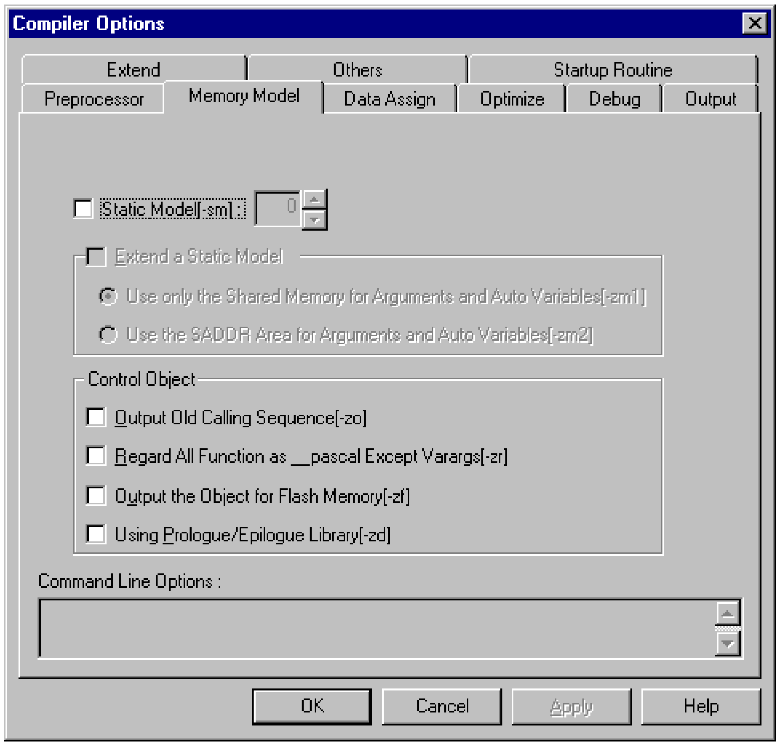
(a) Creating a project
Create a project for the boot area and register the source file.

(b) Compiler, linker, and object converter options settings
Only the minimum options required for build are set in MAKE file automatically created when project creation is ended. Project-specific options are set with the [Tools] menu.
Selecting [Compiler Options] in the [Tools] menu displays the <Compiler Options> dialog box.

<1> Setting compiler option

Do not specify the -ZF option in the <<Memory Model>> tab.

Figure 3-3. Compiler Options Dialog Box



[ ] Static Model[-sm] : 0

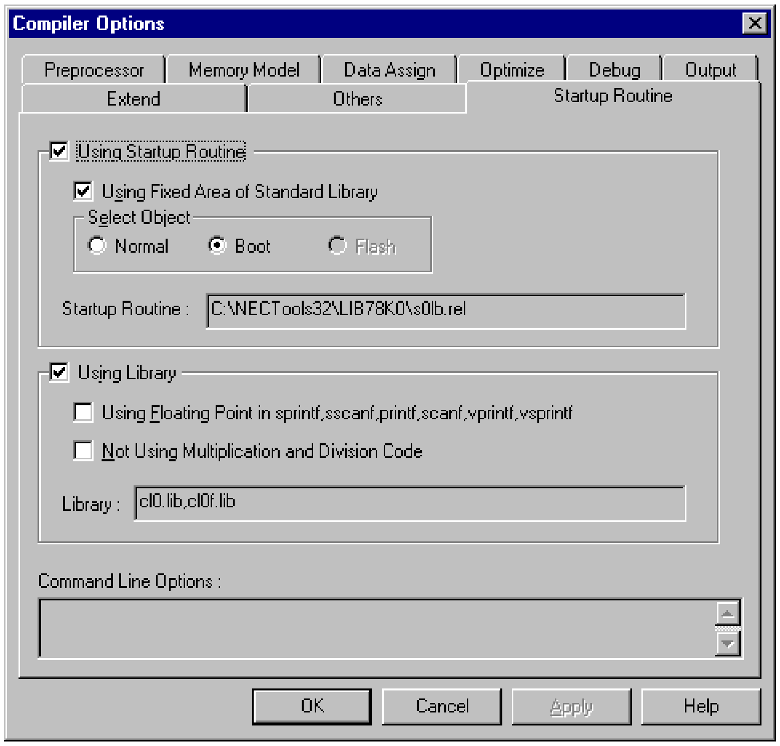
[ ] Extend a Static Model
(●) Use only the Shared Memory for Arguments and Auto Variables[-zm1]
( ) Use the SADDR Area for Arguments and Auto Variables[-zm2]

Control Object
[ ] Output Old Calling Sequence[-zo]
[ ] Regard All Function as __pascal Except Varargs[-zr]
[ ] Output the Object for Flash Memory[-zf]
[ ] Using Prologue/Epilogue Library[-zd]

Command Line Options :

Select "Boot" in the [Select Object] box under the <<Startup Routine>> tab.

Figure 3-4. Selection of Boot Area Object



[✓] Using Startup Routine
[✓] Using Fixed Area of Standard Library
Select Object
( ) Normal (●) Boot ( ) Flash
Startup Routine : C:\NECTools32\LIB78K0\s0lb.rel

[✓] Using Library
[ ] Using Floating Point in sprintf,sscanf,printf,scanf,vprintf,vsprintf
[ ] Not Using Multiplication and Division Code
Library : cl0.lib,cl0f.lib

Command Line Options :

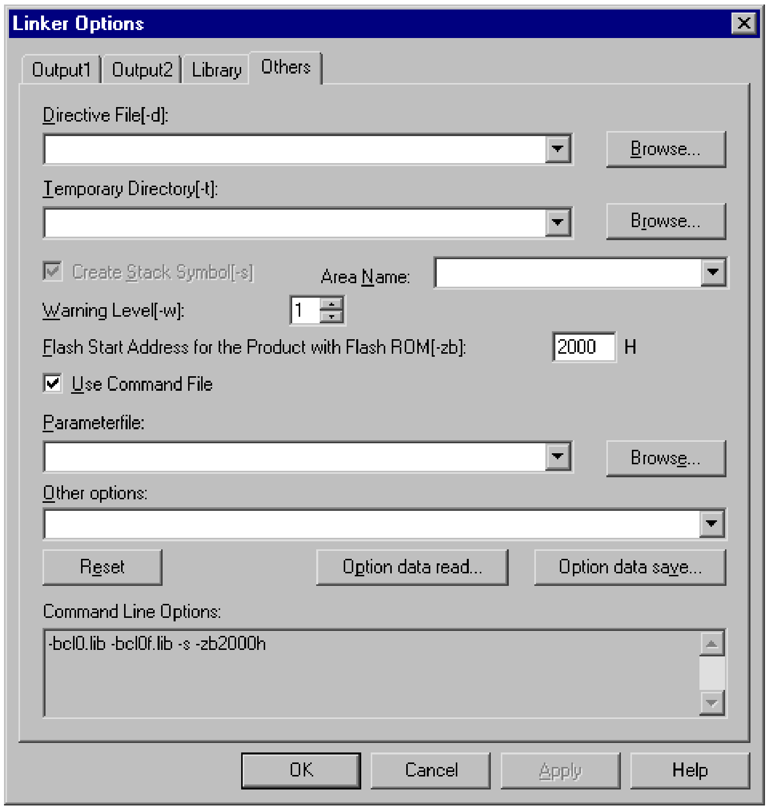
<2> Setting linker option

Specify flash start address specification option -ZB and then click the [OK] button.

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

Also, since C source (boot.c) is included in the source file specification, stack symbol automatic generation option -S is automatically set.

Remark For information about the linker options, refer to **RA78K0 Assembler Package Operation User's Manual (U16629E)**.



✓ Create Stack Symbol[-s]

Warning Level[-w]: 1

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

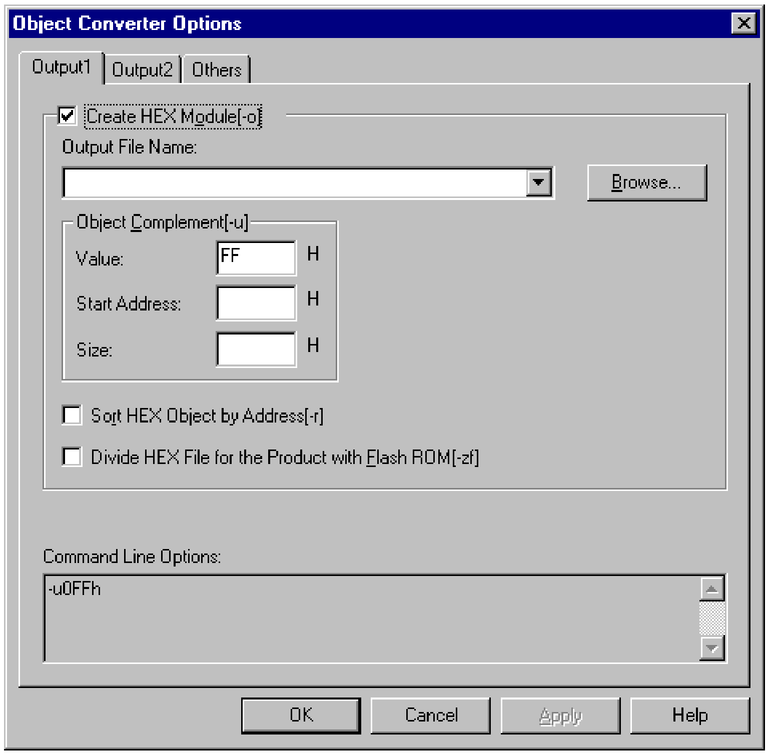
✓ Use Command File

Command Line Options:

```
-bcl0.lib -bcl0f.lib -s -zb2000h
```

<3> Setting object converter option

Do not specify the object converter option -ZF.


Object Converter Options

Output1 | Output2 | Others

[✓] Create HEX Module[-o]

Output File Name:

Browse...

Object Complement[-u]

Value: FF H

Start Address: H

Size: H

[ ] Sort HEX Object by Address[-r]

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

Command Line Options:

```
-u0FFh
```

OK | Cancel | Apply | Help

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

(c) Building project

Projects are built with the set options.

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

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

Caution The contents displayed in the <Output> window during build are saved as the "Project file name + .plg" file name to the project directory.

(2) Compiling to linking program for flash area

(a) Creating a project

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

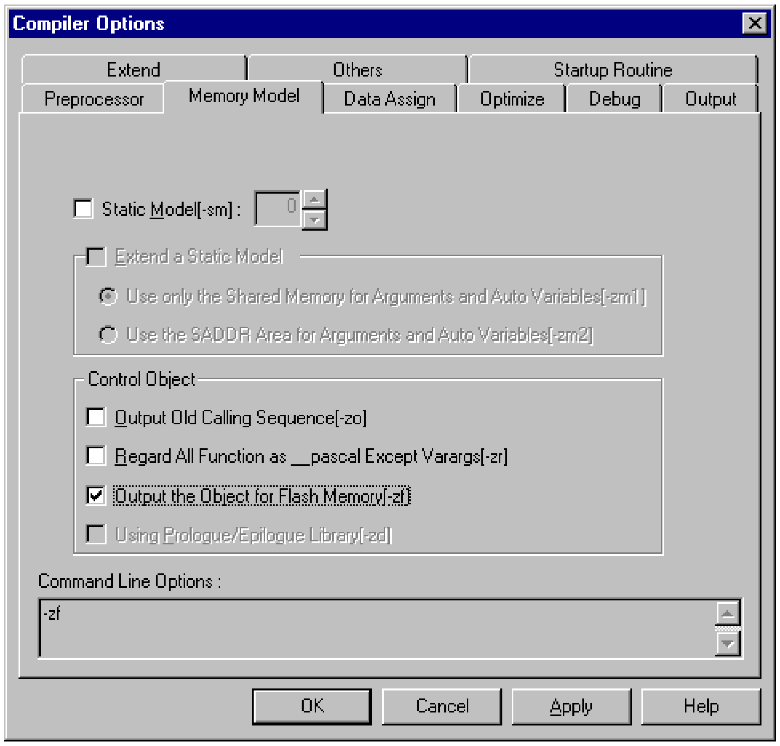
(b) Compiler, linker, and object converter option settings

Only the minimum options required for build are set in MAKE file automatically created when project creation is ended. Project-specific options are set with the [Tools] menu.
Selecting [Compiler Options] in the [Tools] menu displays the <Compiler Options> dialog box.

<1> Setting compiler option

Specify the -ZF option in the <<Memory Model>> tab.

Figure 3-5. Compiler Options Dialog Box



☐ Static Model[-sm] : 0

☐ Extend a Static Model

● Use only the Shared Memory for Arguments and Auto Variables[-zm1]

○ Use the SADDR Area for Arguments and Auto Variables[-zm2]

Control Object

☐ Output Old Calling Sequence[-zo]

☐ Regard All Function as __pascal Except Varargs[-zr]

☑ Output the Object for Flash Memory[-zf]

☐ Using Prologue/Epilogue Library[-zd]

Command Line Options :

-zf

Flash is automatically selected in the [Select Object] box under the <<Startup Routine>> tab.

<2> **Setting linker option**

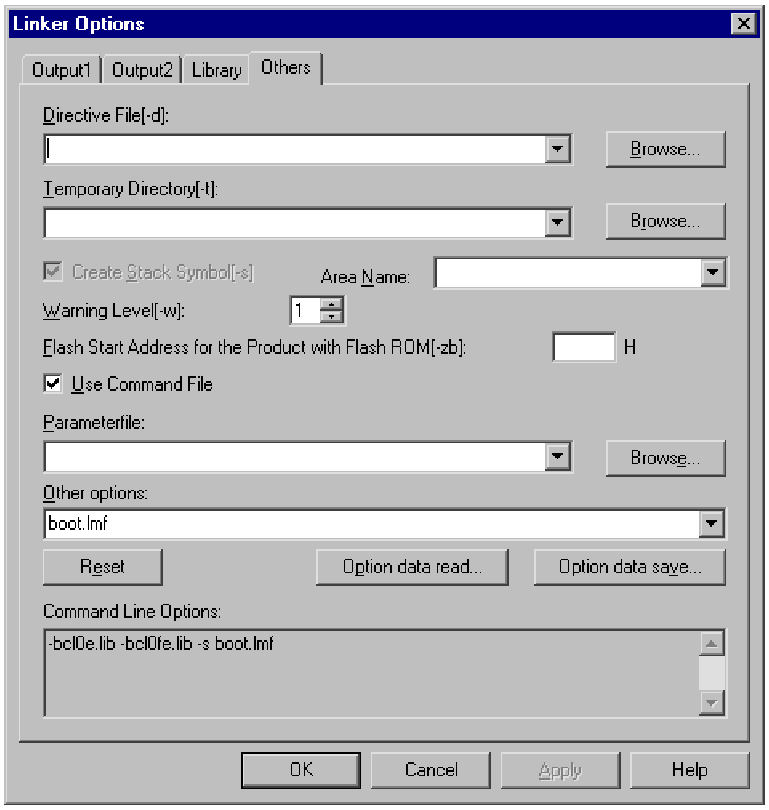
Specify the load module file for the boot area to be linked and then click the [OK] button.

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

Also, since C source (flash.c) is included in the source file specification, stack symbol automatic generation option -S is automatically set.

Remark For information about the linker options, refer to **RA78K0 Assembler Package Operation User's Manual (U16629E).**

Figure 3-6. Linker Options Dialog Box



Directive File[-d]:

Temporary Directory[-t]:

[✓] Create Stack Symbol[-s] Area Name:

Warning Level[-w]: 1

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

[✓] Use Command File

Parameterfile:

Other options: boot.lmf

Command Line Options:

```
-bcl0e.lib -bcl0fe.lib -s boot.lmf
```

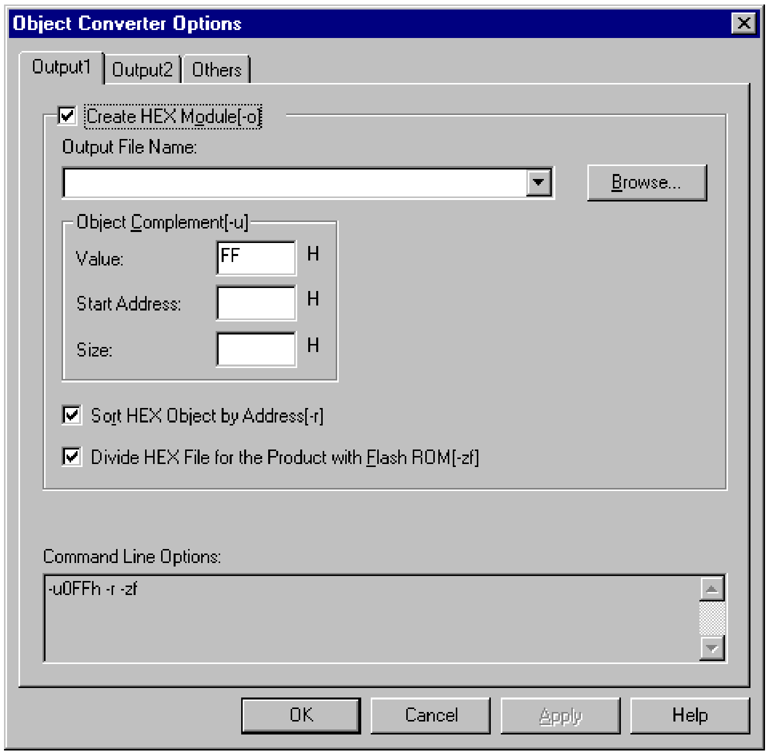
<3> Setting object converter option (for flash area)

Be sure to specify the object converter option -ZF.

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

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

Figure 3-7. Object Converter Options Dialog Box



☑ Create HEX Module[-o]

Output File Name:

Object Complement[-u]

Value: FF H

Start Address: H

Size: H

☑ Sort HEX Object by Address[-r]

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

Command Line Options:

```
-u0FFh -r -zf
```

(c) Building project

Projects are built with the set options.

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

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

Caution The contents displayed in the <Output> window during build are saved as the "Project file name + .plg" file name to the project directory.

3.3.2 Compiling to linking in command line (for DOS prompt and EWS)

(1) When parameter file is not used

The command below is used to start the CC78K0, 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]CC78K0[Δ option] ΔC source name[Δ option]
>[path name]RA78K0[Δ option] Δassembler source name[Δ option]
>[path name]LK78K0[Δ option] object module name, etc.[Δ option]
```

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

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

Examples <1> Compiling program for boot area

```
C> cc78k0 -cf0078 boot.c -iC:\nectools32\inc78k0 -yC:\nectools32\dev
```

<2> Linking program for boot area

```
C> lk78k0 s01b.rel boot.rel -bC:\nectools32\lib78k0\cl0.lib -s
   -oboot.lmf -zb2000h -yC:\nectools32\dev
```

<3> Object-converting program for boot area

```
C> oc78k0 boot.lmf -u0FFh -oboot.lmf -yC:\nectools32\dev
```

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

(b) Compiling to linking program for flash area

Examples <4> Compiling program for flash area

```
C> cc78k0 -cf0078 flash.c -zf -iC:\nectools32\inc78k0
   -yC:\nectools32\dev
```

<5> Linking program for flash area

```
C> lk78k0 boot.lmf s0le.rel flash.rel -bC:\nectools32\lib780\
   cl0e.lib -s -oflash.lmf -yC:\nectools32\dev
```

<6> Object-converting program for flash area

```
C> oc78k0 flash.lmf -u0FFh -r -oflash.lmf -yC:\nectools32\dev
```

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

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

The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see **CHAPTER 5 COMPILER OPTIONS** and **RA78K0 Assembler Package Operation User's Manual (U16629E)**.

Caution When linking a library created by a user or a floating-point library, be sure to specify the library attached to the CC78K0 at the end of the library line. When linking a program for flash area and a program for boot area, specify the load module file for boot area in the beginning, and specify the startup routine for flash area before the user program.
The following shows the library and object module file specification orders when linking.

(Library specification order)
When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points

1. User program library file (specified with -B option)
2. Library file attached to C compiler (specified with -B option)
3. Floating point library file attached to C compiler (specified with -B option)

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

1. User program library file (specified with -B option)
2. Floating point library file attached to C compiler (specified with -B option)
3. Library file attached to C compiler (specified with -B option)

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

(Specification order of other files)

1. Load module file for boot area of user program
2. Startup routine object module file for flash area attached to CC78K0
3. Object module file for flash area of user program

(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.

Caution Parameter files cannot be specified by means of the option setting of PM plus.

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

```
>[path name]CC78K0 Δ-F parameter file name
>[path name]RA78K0 Δ-F parameter file name
>[path name]LK78K0 Δ-F parameter file name
```

The following shows a usage example.

Example

```
C>cc78k0  -Fpara.pcc
C>lk78k0  -Fpara.plk
```

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

The following shows examples of creating parameters by Editor.

(Contents of para.pcc)

```
-cf0078 boot.c -iC:\nectools32\inc78k0 -yC:\nectools32\dev
```

(Contents of para.pra)

```
s0lb.rel boot.rel -bC:\nectools32\lib78k0\cl0.lib -s -oboot.lmf -zb2000h
-yC:\nectools32\dev
```

Remark The -i option specification, -b option path specification, and -y option specification can be omitted depending on the condition. For details, see **CHAPTER 5 COMPILER OPTIONS** and **RA78K0 Assembler Package Operation User's Manual (U16629E)**.

3.4 I/O Files of C Compiler

The CC78K0 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 CC78K0 I/O files are shown below.

Table 3-1. C Compiler I/O Files

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

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

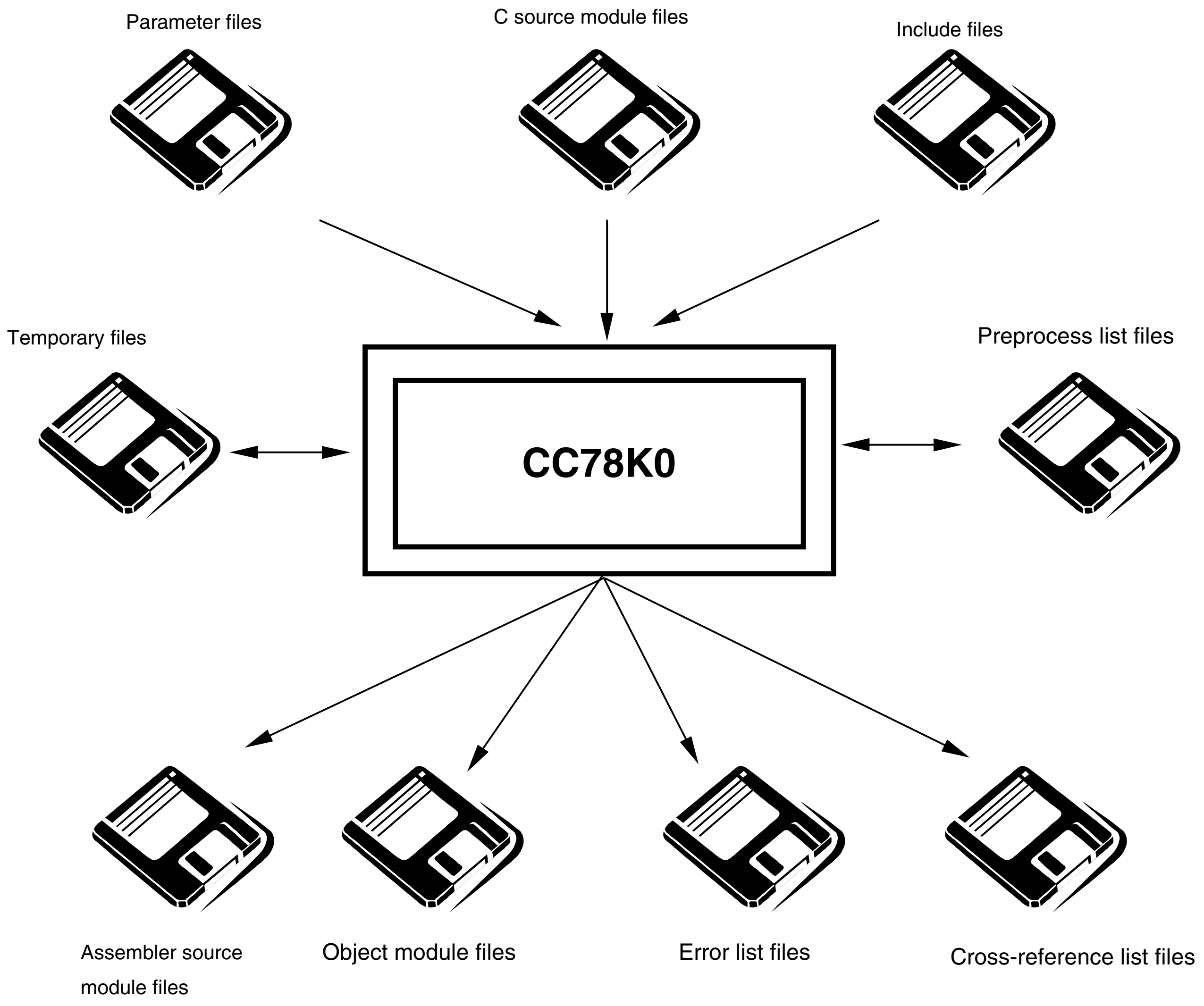
CER: Error list files with C source corresponding to *.C' files (output by specifying the -SE option)

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)

Figure 3-8. C Compiler I/O Files



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

3.5 Execution Start and End Messages

(1) Execution start message

When the CC78K0 starts, the execution start message is displayed on the console.

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

(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 : uPD780xx
    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) : W745 Expected function prototype
    PRIME.C(20) : W745 Expected function prototype
    PRIME.C(26) : W622 No return value
    PRIME.C(37) : W622 No return value
    PRIME.C(44) : W622 No return value

    Target chip : uPD780xx
    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>cc78k0 –c054 -e prime.c -m

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

  A018 Option is not recognized '-m'
  Please enter ‘ CC78K0  -- ‘ , 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 CC78K0 FUNCTIONS

4.1 Optimization Method

Optimization is performed to create efficient object module files in the CC78K0. **Table 4-1 Optimization Methods** lists the supported optimization methods.

Table 4-1. Optimization Methods (1/2)

Phase		Contents	Example
Syntax Analyzer	<1>	Execute during constant computations compilation.	a=3*5; → a=15;
	<2>	True or false decision based on partial evaluation of a logical expression	0 && (a || b) → 0 1 || (a && b) → 1
	<3>	Offset calculations of pointers, arrays, etc.	Calculate the offsets during compilation.
Code Generator	<4>	Register management	Effectively use unused registers.
	<5>	Use the special instructions of the target CPU.	a=a+1; → Use the inc instruction. Use the move instruction to substitute array elements.
	<6>	Use short instructions.	If there is an instruction with the same operation, use the instruction with fewer bytes. mov a, #0 or xor a, a (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.	a=b+c; → a=b+c;
			d=b+c+e; d=a+e;

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

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

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

	<12>	Change the calculation order in an expression.	The calculation whose result remains in the register as valid before other calculations is executed.

Table 4-1. Optimization Methods (2/2)

Phase		Contents	Example
Optimizer	<13>	Memory device allocation (temporary variables)	Variables used locally are allocated to registers.
	<14>	Peephole optimization	Replacement of special patterns Examples a*1 → a, a+0 → a
	<15>	Decrease the strength of the calculation.	Examples a*2 → a+a, a<< 1
	<16>	Memory device allocation (register variables)	Data is allocated to rapidly accessible memory. Examples Registers, saddr (only when -QR is specified)
	<17>	Jump optimization (-QJ option)	Consecutive jump instructions are combined into one instruction.
	<18>	Register allocation (-QV, -QR 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 CC78K0 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.

The startup routine is described using the example of s0l.rel (which is used when ROMization processing is required and the standard libraries are used).

4.2.1 Linking

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

(1) s0l.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) cl0*.lib: Library attached to CC78K0. The library files of the CC78K0 include the following two libraries.

<1> 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.

<2> 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.

Caution The CC78K0 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 2.3.4 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 plus.
(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.

Example (Δ: blanks such as spaces)

```
        CC78K0Δ-c054Δprime.cΔ-aprime.asmΔ-qx3
```

5.2 Prioritization of Compiler Options

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

Table 5-1. Prioritization of Compiler Options

	-NO	-G	-P	-NP	-D	-U	-A	-E	-X	--	-SA
-R	×									×	
-Q	×									×	
-G	×									×	
-K			Δ	×						×	
-D						O				×	
-U					O					×	
-SA							×			×	
-LW			Δ				Δ	Δ	Δ	×	
-LL			Δ				Δ	Δ	Δ	×	
-LT			Δ				Δ	Δ	Δ	×	
-LF			Δ				Δ	Δ	Δ	×	
-LI										×	Δ

← Horizontal axis

↑
Vertical axis

[Location marked by ×]
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 O]
The option specified last out of an option in the horizontal axis and an option in the vertical axis has priority.

Example 1
```
C>cc78k0 –c054 -e sample.c -no -r -g
```

The -RD and -G options become invalid.

Example 2
```
C>cc78k0 –c054 -e sample.c -p -k
```

Since the -P option is specified, the -K option is valid.

Example 3
```
C>cc78k0 –c054 -e sample.c -utest -dtest=1
```

Since the -D option is specified last, the -U option becomes invalid, and the -D option has priority.

As with the -O and -NO options, the option specified last has priority even if N can be added before the option name.

Example 4
```
C>cc78k0 –c054 -e sample.c -o -no
```

Since the -NO option is specified last, the -O option becomes invalid, and the -NO option has priority.

Options not described in **Table 5-1 Prioritization of Compiler Options** are not particularly affected by other options. However, if the help specification option "- -" was specified, all of the option specifications become invalid. The help specification option cannot be specified in PM plus. To reference help in PM plus, press the help button in each option dialog box of PM plus.

5.3 Descriptions of Compiler Options

This section describes each compiler option in detail.

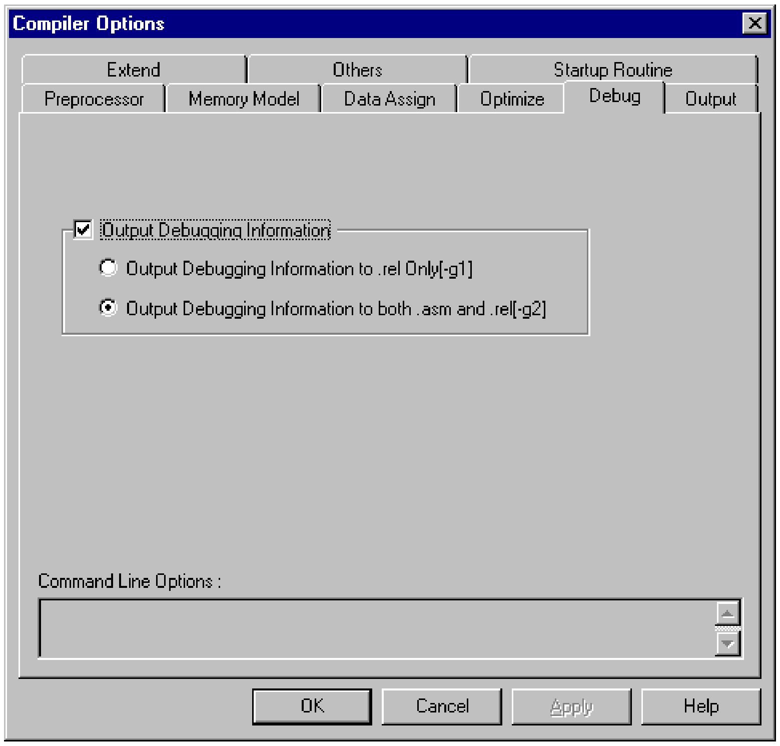
This example illustrates starting the CC78K0 in the command line. To start in PM plus, specify the command, device type specification, and options left out of the C source in the <Compiler Options> dialog box.

Example In command line

```
        C>cc78k0  -c054  prime.c  -g
```

Example When using PM plus

Figure 5-1. Compiler Options Dialog Box



[✓] Output Debugging Information

○ Output Debugging Information to .rel Only[-g1]

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

Command Line Options :

(1) Device type specification (-C)

-C	Device type specification

Description format	-C device-type
Default interpretation	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 CC78K0 is used, device files are required. Use the device file by copying it to the BIN directory or to the DEV directory.

[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 plus is used, because the setting of this option is determined by the project setting.

-C Device type specification

[Use Example]

The specification is made in the command line. The target device is the μPD78054.

```
C>cc78k0  -c054  prime.c
```

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

```
#pragma          pc(054)
#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>cc78k0  prime.c
```

-C Device type specification

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

C source

```
#pragma         pc(054)
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

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

Command line

```
C>cc78k0  -c014  prime.c
```

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

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

 SAMPLE\PRIME.C(1) : W832 Duplicated chip specifier
 sample\prime.c(18): W745 Expected function prototype
 sample\prime.c(20): W745 Expected function prototype
 sample\prime.c(26): W622 No return value
 sample\prime.c(37): W622 No return value
 sample\prime.c(44): W622 No return value

 Target chip : uPD78014
 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 (-O/-NO)

-O/-NO	Object module file creation specification

Description formats	-O [output-file-name]
	-NO
Default interpretation	-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 plus, 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>cc78k0  -c054  prime.c  -no  -o
```

(3) Memory assignment specification (-R/-NR, -RD/-NR, -RK/-NR, -RS/-NR, -RC/-NR)

-R/-NR	Memory assignment specification

Description formats	-R [process-type] (Multiple specifications are possible)
	-NR
Default interpretation	-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 abort error (A012) occurs.

Process type	Function
B	Assigns a bit field from the most significant bit (MSB).
D[n] (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] (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] (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.

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>cc78k0  -c054 -rds
```

-RD/-NR — Memory assignment specification

Description formats	-RD [n] (n = 1, 2, 4)
	-NR
Default interpretation	-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.

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
Omitted	All variables (including structure and union)

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.

-RK/-NR Memory assignment specification

Description formats	-RK [n] (n = 1, 2, 4)
	-NR
Default interpretation	-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.

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
Omitted	All variables (including structure and union)

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.

-RS/-NR Memory assignment specification

Description formats	-RS [n] (n = 1, 2, 4)
	-NR
Default interpretation	-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.

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
Omitted	All variables (including structure and union)

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 (-Q/-NQ)

-Q/-NQ	Optimization specification

Description formats	-Q [optimization-type] (If multiple types are specified, specify them consecutively)
	-NQ
Default interpretation	-QCJLVW

[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 **Table 5-2 Optimization Types**.

[Description]

Table 5-2 lists the optimization types that can be specified by the -Q option.

Table 5-2. Optimization Types (1/2)

Optimization Type	Process Description
No specification	Regarded as the -QCJLVW specification.
U (-QU option)	Regards the char with no qualifier as a unsigned char to improve code efficiency
C[n] (n = 1, 2) (-QC option)	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
R[n] (n = 1, 2) (-QR option)	Adds a register variable to a register and assigns it to the saddr area. The scope of assigning register variable changes depending on the value of n as follows. If n is omitted, it is interpreted as n = 2. 1: Assigns norec argument and auto variable to the saddr area 2: Assigns norec argument, auto variable, and register variable to the saddr area
J (-QJ option)	Optimize jump instructions.
X[n] (n = 1 to 3) (-QX option)	Assigns the optimization options automatically according to the priority of speed/code size. The assigned option differs depending on the value of n as follows. If n is omitted, it is interpreted as n = 2. 1: Speed precedence. Regarded as the -QCJVW option specification. 2: Default. Regarded as the -Q option specification. 3: Code size precedence. Regarded as the -QCJL4VW option specification.
E (-QE option)	Outputs the object using [HL+B]. This option is valid only when the -SM option is specified.
H (-QH option)	Outputs the object using [HL].bit.

-Q/-NQ Optimization specification

Table 5-2. Optimization Types (2/2)

Optimization Type	Process Description
W[n] (n = 1 to 3) (-QW option)	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 two 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. 1: Changes the execution order in an expression 2: Changes the execution order in an expression. Assumed no carry occurs when the saddr-aligned char/unsigned char/short/unsigned short/int/unsigned int array is referenced with an unsigned char variable, performs address calculation of lower 1 byte. 3: Applies the scope 2 to areas other than the saddr area
V (-QV option)	Assigns an automatic variable automatically to a register or the saddr area
L[n] (n = 1 to 4) (-QL option)	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. 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 one instruction

Note When the -QC option is specified in the CC78K0, 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

-Q/-NQ Optimization specification

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 -QI option, changing the data type can be avoided. When the -QC1 option is specified, constant calculation is sign-extended.

```
(Example)  When -QC2 option is specified

           int  i;

           i = (int)20 * 20;        /* 400 */
```

Multiple optimization types can be specified.
If the -Q option or optimization types are omitted, the optimization is identical to when the -QCJLVW option is specified.
To delete a portion of the default options specify the options other than the options you want to delete (Example -QR is specified → Deletes -QCJLVW).
If both the object module file and the assembler source module file are not output, the -Q option other than -QU becomes invalid.
If both the -Q and -NQ options are simultaneously specified, the last specified one is valid.
If several -Q options are simultaneously specified, the last specified one is valid.
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>cc78k0  -c054  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>cc78k0  -c054  prime.c  -qc  -qr
```

If you want to validate both the -QC and -QR options, input the following command.

```
C>cc78k0  -c054  prime.c  -qcr
```

(5) Debugging information output specification (-G/-NG)

-G/-NG	Debugging information output specification

Description formats	-G [n] (n = 1, 2)
	-NG
Default interpretation	-G2

[Function]

The -G option specifies the addition of debugging information to the object module file.

The -NG option invalidates the -G option.

[Application]

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

[Description]

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

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

If both -G and -NG are simultaneously specified, the last specified one is valid.

If both the object module file and the assembler source module file are not output, the -G option becomes invalid.

[Use Example]

The -G option is specified.

```
C>cc78k0 -c054 prime.c -g
```

(6) Preprocess list file creation specification (-P, -K)

-P	Preprocess list file creation specification

Description formats	-P [output-file-name]
Default interpretation	Nothing (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 plus, 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>cc78k0  -c054  prime.c  -psample.ppl
```

-K Preprocess list file creation specification

Description formats	-K [process-type] (Multiple specifications are possible)
Default interpretation	-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.

Table 5-3. 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	Line number and paging processing

Remark Multiple process types can be specified.

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

-K Preprocess list file creation specification

[Use Example]

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

```
      C>cc78k0  -c054  prime.c  -p  -kcn
```

prime.ppl is referenced.

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

 Command   : -c054 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 : uPD78054
 Device file : VX.XX
*/
```

(7) Preprocess specification (-D, -U, -I)

-D	Preprocess specification

Description formats	-D macro-name [=definition-name] [, macro-name [=definition-name]]... (Multiple specifications are possible)
Default interpretation	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>cc78k0  -c054  prime.c  -dTEST,TIME=10
```

-U	Preprocess specification

Description formats	-U macro-name [, macro-name]... (Multiple specifications are possible)
Default interpretation	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>cc78k0  -c054  prime.c  -dTEST  -uTEST
```

-I Preprocess specification

Description format	-I directory [, directory]...	(Multiple specifications are possible)
Default interpretation	Directory with source file[Note 1]	
	Directory specified by environment variable INC78K0	
	C:\NECTools32\INC78K0[Note 2]	

[Function]

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

[Application]

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

[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.

- Directory with source file[Note 1]
- Directory specified with -I option
- Directory specified with environment variable INC78K0
- C:\NECTools32\INC78K0[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.

2. This is an example of when the CC78K0 is installed to C:\NECTools32 (Windows version).

[Use Example]

The -I option is specified.

```
C>cc78k0  -c054  prime.c  -id:, D:\sample
```

(8) Assembler source module file creation specification (-A, -SA)

-A	Assembler source module file creation specification

Description formats	-A [output-file-name]	
Default interpretation	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 plus, 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>cc78k0  -c054  prime.c  -asample.asm
```

The assembler source module file is output to the printer.

```
C>cc78k0  -c054  prime.c -aprn
```

-SA — Assembler source module file creation specification

Description formats	-SA [output-file-name]
Default interpretation	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 plus, 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.

-SA Assembler source module file creation specification

[Use Example]

The -SA option is specified.

```
     C>cc78k0  -c054  prime.c  -sa
```

prime.asm is referenced.

```
; 78K/0 Series C Compiler Vx.xx Assembler Source
;                                                 Date:xx xxx xxxx Time:xx:xx:xx

; Command    : -c054 prime.c -sa
; In-file    : prime.c
; Asm-file   : prime.asm
; Para-file  :

$PROCESSOR (054)
$DEBUG
$NODEBUGA
$KANJICODE SJIS
$TOL_INF       03FH, 0330H, 02H, 020H, 00H

$DGS   FIL_NAM, .file,                 034H,    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, 14
       push   hl                                               ;[INF]1, 4
       push   ax                                               ;[INF]1, 4
       push   ax                                               ;[INF]1, 4
       push   ax                                               ;[INF]1, 4
       push   ax                                               ;[INF]1, 4
       movw   ax, sp                                           ;[INF]2, 8
       movw   hl, ax                                           ;[INF]1, 4
??bf_main:
; line    9 : int i, prime, k, count;
; line   10 :
; line   11 :        count = 0;
$DGL  0, 4
       mov    a, #00H ; 0                                      ;[INF]2, 4
       mov    [hl],a   ; count                                 ;[INF]1, 4
       mov    [hl+1],a       ; count                           ;[INF]2, 8
```

-SA Assembler source module file creation specification

```
; line   12 :
; line   13 :        for ( i = 0 ; i <= SIZE ; i++)
$DGL  0,6
      mov     [hl+6],a       ; i                              ;[INF]2, 8
      mov     [hl+7],a       ; i                              ;[INF]2, 8
?L0003:
      mov     a,[hl+6]       ; i                              ;[INF]2, 8
      xch     a,x                                             ;[INF]1, 2
      mov     a,[hl+7]       ; i                              ;[INF]2, 8
      cmpw    ax,#0C8H       ; 200                            ;[INF]3, 6
      or1     CY,a.7                                          ;[INF]2, 4
      bc      $$+4                                            ;[INF]2, 6
      bnz     $?L0004                                         ;[INF]2, 6
              :
      END

; *** Code Information ***
;
; $FILE H:\um\prime.c
;
; $FUNC main(8)
;       bc = (void)
;       CODE SIZE = 218 bytes, CLOCK_SIZE = 678 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 = 30 bytes, CLOCK_SIZE = 116 clocks, STACK_SIZE = 8 bytes
;
; $FUNC printf(41)
;       bc = (char c:x)
;       CODE SIZE= 14 bytes, CLOCK_SIZE = 58 clocks, STACK_SIZE = 6 bytes

; Target chip : uPD78054
; Device file : Vx.xx
```

(9) Error list file creation specification (-E, -SE)

-E	Error list file creation specification

Description formats	-E [output-file-name]	
Default interpretation	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 plus, 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.

-E Error list file creation specification

[Use Example]

The -E option is specified.

```
     C>cc78k0  -c054  prime.c  -e
```

The error list file is referenced.

```
prime.c(   18) : W745 Expected function prototype
prime.c(   20) : W745 Expected function prototype
prime.c(   26) : W622 No return value
prime.c(   37) : W622 No return value
prime.c(   44) : W622 No return value

Target chip : uPD78054
Device file : Vx.xx

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

-SE Error list file creation specification

Description formats	-SE [output-file-name]	
Default interpretation	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 directory and the file name cannot be specified for include files. If the file type of the include file is ‘H,’ the error list file with the file type of ‘her’ is output to the current drive. It the file type of the include file is ‘C,’ the error list file with the file type of ‘cer’ is output. In all other cases, the error list file with the ‘er’ file type is output.

If there weren’t any errors, the C source is not added. In this case, the error list file is not created for the include file.

If the -W0 option is specified, warning messages are not output.

[Cautions]

To change the output destination when using PM plus, 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.

-SE Error list file creation specification

[Use Example]

The -SE option is specified.

```
      C>cc78k0  -c054  prime.c  -se
```

prime.cer is referenced.

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

Command   : -c054  prime.c  -se
In-file   : prime.c
Err-file  : prime.cer
Para-file :
*/

#defineTRUE    1
#defineFALSE   0
#defineSIZE    200

char   mark[SIZE+1];
main()
{
               ⋮
                        prime = i + i + 3;
                        printf("%6d",prime);
*** WARNING W745 Expected function prototype
                        count++;
                                if((count%8) == 0) putchar('\n');
*** WARNING W745 Expected function prototype
                        for ( k = i + prime ; k <= SIZE ; k += prime)
               ⋮
```

(10) Cross-reference list file creation specification (-X)

-X	Cross-reference list file creation specification

Description formats	-X [output-file-name]	
Default interpretation	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'.

The cross-reference file is created even if a compile error except for fatal errors (F101, abort errors other than A024) occurs. In such a case, however, the file contents are not guaranteed.

[Cautions]

To change the output destination when using PM plus, 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> cc78k0  -c054  prime.c  -x
```

-X Cross-reference list file creation specification

prime.xrf is referenced.

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

Command   : -c054  prime -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
REG1            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
REG1            pointer s             29      36
PARAM
REG1            int     i             30      35
PARAM
AUTO1           int     j             32      35
AUTO1           pointer ss            33      36
REG1            char    c             40      43
PARAM
AUTO1           char    d             42      43
                #define TRUE           1      14
                #define FALSE          2      22
                #define SIZE           3        5      13      15      21

 Target chip : uPD78054
 Device file : VX.XX
```

(11) List format specification (-LW, -LL, -LT, -LF, -LI)

-LW	List format specification

Description format	-LW [number-of-characters]	
Default interpretation	-LW132	(For console output, this becomes 80 characters)

[Function]

The -LW option specifies the number of characters in one line of each type of list file.

[Application]

If you want to change the number of characters in one 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).

$72 \leq$ number of characters printed in one line ≤ 132

If the number of characters is omitted, the number of characters in one 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> cc78k0  -c054  prime.c  -x
```

-LL List format specification

Description format	-LL [number-of-lines]
Default interpretation	-LL66 (For console output, this becomes 65,535 lines)

[Function]

The -LL option specifies the number of lines on one page of each type of list file.

[Application]

If you want to change the number of lines in one 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 ≤ number of lines printed on one page ≤ 65535

If -LL0 is specified, there is no page break.

If the number of lines is omitted, the number of lines on one 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 one page of the cross-reference list file is set to 20 lines.

```
C> cc78k0  -c054  prime.c  -x  -ll20
```

-LT	List format specification

Description format	-LT [number-of-characters]
Default interpretation	-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 one 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.

$0 \leq$ number of specifiable characters ≤ 8

If the -LT0 is specified, the tabulation processing is not performed, and the tab codes are output.

If the number of characters is omitted, the number of expansion characters of a tab becomes 8 characters.

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

[Use Example]

The -LT option is omitted.

```
C> cc78k0  -c054  prime.c  -p
```

The blanks based on the HT code are set to one (1).

```
C> cc78k0  -c054  prime.c  -p  -lt1
```

-LF — List format specification

Description format	-LF
Default interpretation	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> cc78k0  -c054  prime.c  -a  -lf
```

-LI List format specification

Description format	-LI
Default interpretation	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> cc78k0  -c054  prime.c  -sa  -li
```

(12) Warning output specification (-W)

-W	Warning output specification

Description format	-W [level]
Default interpretation	-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.

Table 5-4. 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> cc78k0  -c054  prime.c
```

(13) Execution state display specification (-V/-NV)

-V/-NV	Execution state display specification

Description formats	-V -NV
Default interpretation	-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> cc78k0  -c054  prime.c  -v
```

(14) Parameter file specification (-F)

-F	Parameter file specification

Description format	-F file-name
Default interpretation	The options and the input file name can be input only from the 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.

[Caution]

This option cannot be used when using PM plus (an error occurs).

[Use Example]

Contents of parameter file prime.pcc

```
; parameter file
prime.c  -c054  -aprime.asm  -e  -x
```

prime.pcc is used in the compilation.

```
C> cc78k0  -fprime.pcc
```

(15) Temporary file creation directory specification (-T)

-T	Temporary file creation directory specification

Description format	-T directory
Default interpretation	The files are created in the drive and directory specified by the environment variable TMP. If not specified in a Windows-based system, the files are created in the current drive and current directory. If UNIX-based, they are created in /tmp.

[Function]

The -T option specifies the drive and directory 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 directory 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 directory.

```
C> cc78k0  -c054  prime.c  -ttmp
```

(16) Help specification (--/-?/-H)

--/-?/-H	Help specification

Description formats	--, -?, -H
Default interpretation	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 plus. To reference help in PM plus, 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> cc78k0  -H
```

(17) Function expansion specification (-Z/-NZ)

-Z/-NZ Function expansion specification

Description format	-Z [type] (If multiple types are specified, specify them consecutively)
	-NZ
Default interpretation	-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 abort error (A012) will occur.

[Application]

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

[Description]

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

Table 5-5. Type Specifications of -Z Option (1/2)

Type Specification	Description
Omitted	Regarded as -NZ specification.
O	Outputs code of interface between functions of old specification (CC78K0 V2.11 or before).
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.

Note S, E, and N cannot be specified simultaneously.

-Z/-NZ Function expansion specification

Table 5-5. Type Specifications of -Z Option (2/2)

Type Specification	Description
A	Functions not in the ANSI standard are illegal. The ANSI-compliant portion of the functions are valid. Specifically, the following tasks are performed. The following are no longer reserved words. callt, callf noauto, norec, sreg, bit, boolean, #asm, #endasm The trigraph sequence (3-character representation) becomes valid. The compiler-defined macro _ _STDC_ _ is 1. The following warning is output for a char type bit field. (W787 Bit field type is char) If -W2 is specified, the following warnings are output for the -QC, -ZP, -ZC, -ZI, and -ZL options. (W029 '-QC' option is not portable) (W031 '-ZP' option is not portable) (W032 '-ZC' option is not portable) (W036 '-ZI' option is not portable) (W037 '-ZL' option is not portable) If –W2 is specified, the following warning is output for each #pragma statement. (W849 #pragma statement is not portable) If –W2 is specified, the following warning is output for an _ _asm statement and the assemble output is performed. (W850 Asm statement is not portable) If -W2 is specified, the following error is output for an #asm to #endasm block. (F801 Undefined control, etc.)
M[n] (n = 1, 2)	Enables use of extend specifications for static model. Up to 6 arguments can be described in int size, and up to 9 arguments can be described in char size. Enables description of structure/union return value for 1-, 2-byte structure/union arguments and function return values. The _@KREGxx utilization method is changed by the value of n. If n is omitted, n is considered to be 1. 1: Use _@KREGxx as the shared area only for leaf function. 2: Perform _@KREGxx save/restore and allocate argument and automatic variable to _@KREGxx.
D	Place the processing routines before and after the function into a library. Outputs warning for -QL4 and processes as -QL3.
R	Automatically adds a pascal function modifier.
F	Outputs object from flash.
I	Regards int and short descriptions as char. The compiler definition macro _FROM_INT_TO_CHAR is regarded as 1.
L	Regards long description as int. The compiler definition macro _FROM_LONG_TO_INT is regarded as 1.

[Use Example]

The -ZC and -ZP options are specified.

```
    C> cc78k0  -c054  prime.c  -zpc
```

(18) Device file search path (-Y)

-Y	Device file search path

Description format	-Y
Default interpretation	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.

(1) <..\dev> (for the path where cc78k0.exe started)

(2) Path where CC78K0 started

(3) Current directory

(4) PATH environment variable

[Application]

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

[Caution]

When using PM plus, a directory 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> cc78k0  -c054  -ya:\tmp\dev
```

(19) Static model specification (-SM)

-SM	Static model specification

Description formats	-SM [n] (n = 1 to 16)
Default interpretation	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 FEDFH 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.

-SM Static model specification

[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> cc78k0   -c054  test.c  -sm16
```

CHAPTER 6 C COMPILER OUTPUT FILES

The CC78K0 outputs the following files.

- Object module file
- Assembler source module file
- Preprocess list file
- Cross-reference list file
- Error 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/0 Series C Compiler V(1)x.xx Assembler Source
    ;                                                Date:(2)xxxxx Time:(3)xxxxx

    ; Command     :(4)-c054 prime.c -sa
    ; In-file     :(5)prime.c
    ; Asm-file    :(6)prime.asm
    ; Para-file   :(7)

     $PROCESSOR((8)054)
(9) $DEBUG
(10)$NODEBUGA
(11)$KANJICODE SJIS
(12)$TOL_INF        03FH,  0330H,  02H,  020H,  00H

(13)$DGS    FIL_NAM, .file,          034H,   0FFFEH,   03FH,   067H,   01H,   00H
          :
(14)        EXTRN  _@RTARG0
          :
    ; line (15)1 : (16)#define TRUE   1
    ; line (15)2 : (16)#define FALSE  0
    ; line (15)3 : (16)#define SIZE   200
          :
(14)_main:
(17)$DGL    1.14
(14)        push  hl                                  ;(21)[INF]1, 4
(14)        push  ax                                  ;(21)[INF]1, 4
(14)        push  ax                                  ;(21)[INF]1, 4
(14)        push  ax                                  ;(21)[INF]1, 4
          :
(18)??bf_main:
          :

    ;(22) *** Code Information ***
    ;
    ;(23) $FILE C:\NECTools32\Smp78k0\CC78K0\prime.c
    ;(24) $FUNC main(8)
    ;(25)   bc = (void)
    ;(26)   CODE SIZE= 218 bytes, CLOCK_SIZE = 678 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 = 30 bytes, CLOCK_SIZE = 116 clocks, STACK_SIZE = 8 bytes
    ;
    ;(24) $FUNC putchar(41)
    ;(25)   bc = (char c:x)
    ;(26)   CODE SIZE = 14 bytes, CLOCK_SIZE = 58 clocks, STACK_SIZE = 6 bytes

    ; Target chip : (19)uPD78054
    ; Device file : (20)Vx.xx
```

[Description of output items] (1/2)

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 “CC78K0”. 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 $KANJICODE SJIS, $KANJICODE EUC, or $KANJICODE NONE.
(12)	Tool information	37 (fixed)	Outputs tool information, version number, error information, specified options, etc. (information starts with $TOL_INF).
(13)	Symbol information	—	Outputs symbol information (information starts with $DGS). This information is output only when the debug data output option has been specified. Even then, it is not output if the –G1 option has been specified.
(14)	Assembler source	—	Outputs an assembler source file containing the compilation results.
(15)	Line number	4 (fixed)	Outputs the C source module file’s line numbers as right-aligned decimal value with zeros suppressed.
(16)	C source	—	This is the input C source image. Contents after column 80 are output beginning at column 16 on the next line. A semicolon (;) is output to column 1.
(17)	Line number information	—	Line number for line number entry (information starts with $DGL) This information is output only when the debug data output option has been specified. Even then, it is not output if the –G1 option has been specified.

[Description of output items] (2/2)

Item Number	Description	Number of Columns	Format
(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.
(27)	Function information (call function)	—	Outputs the function name and function call line number as decimal code. (Information starting with ;$CALL).
(28)	Function information (Call function's return value, argument)	—	Outputs return value register and argument information during function call (register or stack position).

6.3 Error List File

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

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

6.3.1 Error list file with C source

[Output format]

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

 Command     : (4) -c054 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;
 ***ERROR (9) F711 (10) Undeclared ‘cont’ ; function ‘main’
(8)     for (i = 0 ; i <= SIZE ; i++)
(8)            mark[i] = TRUE;
(8)     for (i = 0 ; i<= SIZE ; i++) {
(8)            if (mark[i]) {
                          prime = i + i + 3;
                          printf ("%6d", prime);
 ***WARNING (9)W745 (10)Expected function prototype
                  ⋮
 /*
 (11) Target chip: uPD78054
 (12) Device file: Vx.xx
 Compilation complete, (13) 1 error(s) and  (14) 5 warning(s) found.
 */
```

[Description of output items]

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 “CC78K0”. 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	4 (fixed)	Outputs error numbers in the “#nnn” format. “F” is output if "#" is an error and “W” is output if it is a warning. "nnn" (the error number) is displayed as a three-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) :  (3) W745  (4) Expected function prototype
(1) prime.c((2) 20) :  (3) W745  (4) Expected function prototype
(1) prime.c((2) 26) :  (3) W622  (4) No return value
(1) prime.c((2) 37) :  (3) W622  (4) No return value
(1) prime.c((2) 44) :  (3) W622  (4) No return value

Target chip :(7) uPD78054
Device file :(8) Vx.xx

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

[Description of output items]

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	4 (fixed)	Outputs the error message number in "#nnn" format. "F" is output if "#" is an error and "W" is output if it is a warning. "nnn" is the error number.
(4)	Error message	—	See **CHAPTER 9 ERROR MESSAGES**.
(5)	Number of errors	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.
(6)	Number of warnings	4 (fixed)	Outputs a right-aligned decimal value with zeroes suppressed.
(7)	Target device for this compiler	Maximum 15 (variable)	Displays the target device as specified via command line option -C or the source file.
(8)	Device file version	6 (fixed)	Displays the version number of the input device file.

6.4 Preprocess List File

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

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

[Output format]

When PAGEWIDTH = 80

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

 Command      : (4) -c054 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: uPD78054
(11) Device file: Vx.xx
*/
```

[Description of output items]

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 “CC78K0”. 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.
(7)	Parameter file contents	—	Outputs the parameter file contents. Contents that exceed the line length are output beginning at column 13 on the next line. A semicolon (;) is output to column 1. One or more white-space characters or tabs are replaced by a single white-space character.
(8)	Line number	5 (fixed)	Outputs a right-aligned decimal value with zeros suppressed.
(9)	C source	—	This is the input C source. Contents that exceed the line length are output beginning at column 9 on the next line.
(10)	Target device for this compiler	Maximum 15 (variable)	Indicates the target device that is specified by a command line option or in a source file
(11)	Device file version	6 (fixed)	Displays the version number of the input device file.

6.5 Cross-Reference List File

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

[Output format]

When PAGEWIDTH = 80

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


 Command      : (4) -c054 prime.c -x -lw80
 In-file      : (5) prime.c
 Xref-file    : (6) prime.xrf
 Para-file    : (7)
 Inc-file     : [n] (8)

 ATTRIB     MODIFY  TYPE        SYMBOL    DEFINE         REFERENCE

(9) EXTERN  (10)    (11) array  (12) mark  (13) 5   (14) 14   (14)16  (14) 22
(9) EXTERN  (10)    (11) func   (12) main  (13) 7
(9) AUTO1   (10)    (11) int    (12) i     (13) 9   (14) 13  (14) 13  (14) 13  (14) 14
                                                    (14) 15  (14) 15  (14) 15  (14) 16
                                                    (14) 17  (14) 17  (14) 21
(9) AUTO1   (10)    (11) int    (12) prime (13) 9   (14) 17  (14) 18  (14) 21  (14) 21
(9) AUTO1   (10)    (11) int    (12) k     (13) 9   (14) 21  (14) 21  (14) 21  (14) 22
(9) AUTO1   (10)    (11) int    (12) count (13) 9   (14) 11  (14) 19  (14) 20  (14) 25
                                      :
/*
(15) Target chip: uPD78054
(16) Device file: Vx.xx
*/
```

[Description of output items] (1/2)

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 “CC78K0”. 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 AUTOnn, a register variable as REGnn (where nn is the scope value, a numerical value that begins with "1"), an external typedef declaration as EXTYP, an internal typedef declaration as INTYP, a label as LABEL, a structure or union tag as TAG, a member as MEMBER, and a function parameter as PARAM.
(10)	Symbol qualifier attributes	6 (fixed)	Displays the symbol qualifier attributes (left-aligned). A const variable is displayed as CONST, a volatile variable as VLT, a callt function as CALLT, 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, and an interrupt function as VECT.
(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 (five-digit): include file number

[Description of output items] (2/2)

Item Number	Description	Number of Columns	Format
(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 (five-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

7.1 Efficient Operation (EXIT Status Function)

When the compilation ends, the CC78K0 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 plus is not used and the CC78K0 is started in the command line, efficient operation can be further improved by using the status in a batch file.

[Use Example]

```
cc78k0 -c054 %1
IF ERRORLEVEL 1 GOTO ERR
cc78k0 -c054 %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 CC78K0 supports the following environment variables.

• PATH:	Search path for executable forms
• INC78K0:	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)
• LIB78K0:	Search path for libraries

[Use Example] (When using DOS prompt)

```
;AUTOEXEC.BAT
PATH C:\nectools32\bin;c:\bat;c:\cc78k0;c:\tool
VERIFY ON
BREAK ON
SET INC78K0=c:\nectools32\inc78k0
SET LIB78K0=c:\nectools32\lib78k0
SET TMP=c:\tmp
SET LANG78K=sjis
```

[Description]

- Executable files are searched in the sequence of c:\nectools32\bin, c:\bat, c:\cc78k0, c:\tool by path specification.
- Include files are searched from c:\nectools32\inc78k0.
  In the Windows version, if no setting is made, search is performed from C:\NECTools32\INC78K0 (if the CC78K0 is installed to C:\NECTools32).
- Library files are searched from c:\nectools32\lib78k0 during linking.
  In the Windows version, if no setting is made, search is performed from C:\NECTools32\LIB78K0 (if the CC78K0 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 plus.

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 plus, select “Stop build” in the [Run] menu in the PM plus window, or click the button in the tool bar. When building in PM plus, command key input is not accepted.

CHAPTER 8 STARTUP ROUTINES

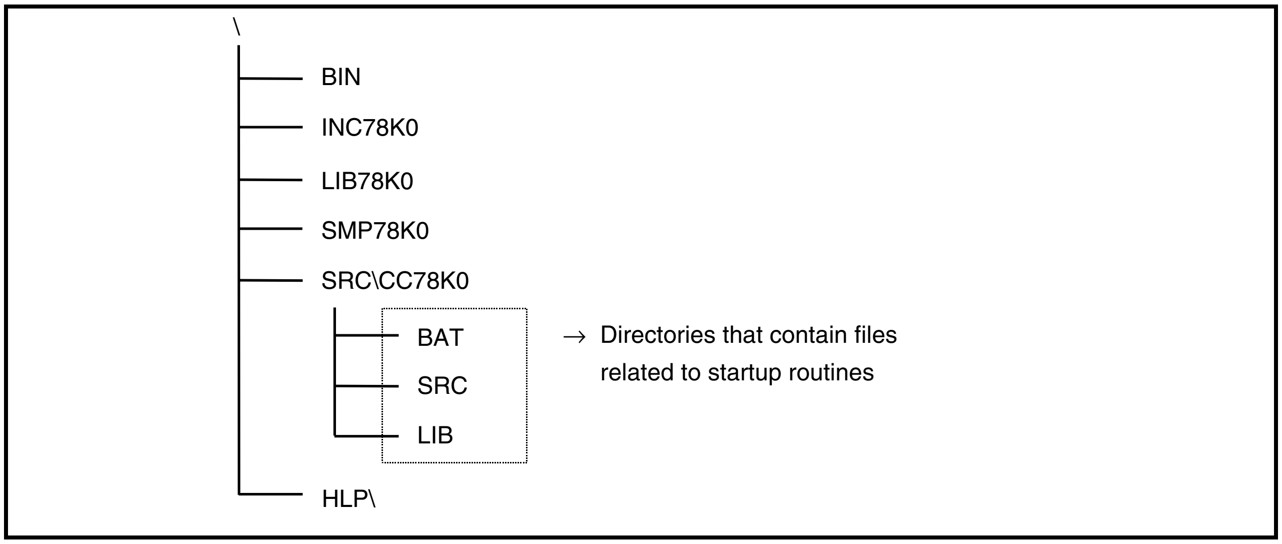
To execute a C language program, a program is needed to activate ROMization for inclusion in the system and the user program (main function). This program is called the startup routine.

To execute a program written by a user, a startup routine must be created for that program. The CC78K0 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 directory SRC\CC78K0 of the compiler package.


\
BIN
INC78K0
LIB78K0
SMP78K0
SRC\CC78K0
BAT
SRC
LIB
→ Directories that contain files related to startup routines
HLP\

The contents of the directories under SRC\CC78K0 are shown next.

The LIB directory 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 78K0 Series. If no particular revisions are needed, link the unmodified object files that are already available. If mkstup.bat (mkstup.sh) offered by the CC78K0 is executed, this object file can be overwritten.

For the file contents, see **2.6.4 Library files**.

8.1.1 BAT directory contents

A batch file in this directory cannot be used in PM plus.

Use these batch files only when the source, such as for a startup routine, must be modified.

The device files (d002.78k and d014.78k) in the BAT directory 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.

Table 8-1. BAT Directory Contents

Batch File Name	Description
mkstup.bat	Assemble batch file for startup routine
reprom.bat	Batch file for updating rom.asm[Note 1]
repbank.bat	Batch file for updating bankcall.asm
repgetc.bat	Batch file for updating getchar.asm
repputc.bat	Batch file for updating putchar.asm
repputcs.bat	Batch file for updating _putchar.asm
repselo.bat	Batch file for updating setjmp.asm and longjmp.asm (the compiler reserved area is saved)[Note 2]
repselon.bat	Batch file for updating setjmp.asm and longjmp.asm (the compiler reserved area is not saved)[Note 2]
repvect.bat	Batch file for updating vect*.asm

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

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 directory contents

The SRC directory 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 directory to assemble.

Table 8-2. SRC Directory 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
bankcall.asm	Source file for bank function call processing
_putchar.asm	_putchar function
putchar.asm	putchar function
getchar.asm	getchar function
longjmp.asm	longjmp function
setjmp.asm	setjmp function
vectxx.asm	Vector source for each interrupt (xx: vector address)
def.inc	For setting library according to type
macro.inc	Macro definition for each typical pattern
vect.inc	Start address of flash memory area branch table
stdio.inc	Character code setting for EOF and LF
library.inc	Selection of library assigned to boot area explicitly

Note A file name with n added is a startup routine that does not have standard library processing. Use only if the standard library will not be used. cstartb*.asm is a startup routine for boot area and cstarte*.asm is a startup routine for flash area.

8.2 Batch File Description

8.2.1 Batch files for creating startup routines

The mkstup.bat (mkstup.sh in UNIX) in the BAT directory is used to create the object file of a startup routine.

The assembler in the RA78K0 Assembler Package is required for mkstup.bat (mkstup.sh). 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\cc78k0\bat directory containing mkstup.bat (mkstup.sh).

```
mkstup device-type Note
```

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 μPD78054.

```
mkstup  054
```

The mkstup.bat (mkstup.sh) 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 ─────────  s0.rel
                    s0b.rel
                    s0e.rel
                    s0l.rel
                    s0lb.rel
                    s0le.rel
                    s0sm.rel
                    s0smb.rel
                    s0sme.rel
                    s0sml.rel
                    s0smlb.rel
                    s0smle.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

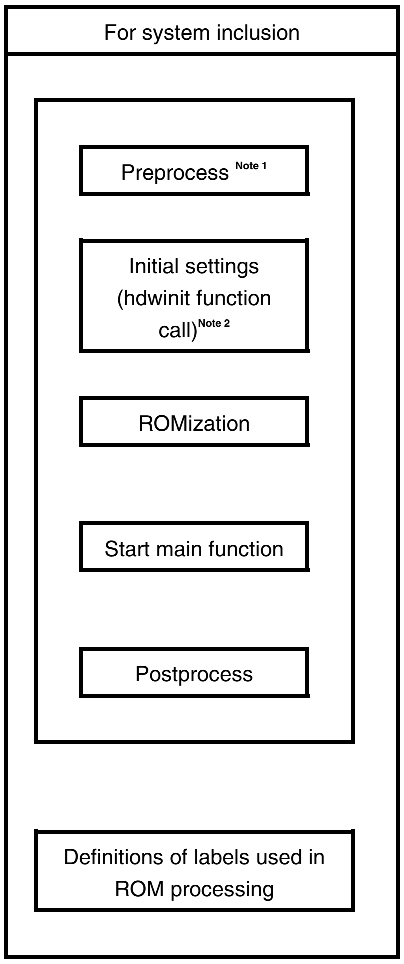
Memory initialization, ROMization for inclusion in the system, and the starting and ending processes for the C source program are performed.

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

Table 8-3 shows the programs related to the startup routines and their configurations.

Table 8-3. Startup Routine Overview


For system inclusion
Preprocess Note 1
Initial settings (hdwinit function call)Note 2
ROMization
Start main function
Postprocess
Definitions of labels used in ROM processing

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.

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 8-4 shows the differences between cstart.asm and cstartn.asm.

Table 8-4. Differences Between Startup Routine Sources

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

(3) Uses of startup routines

Table 8-5 lists the names of the object files for the source files provided by the CC78K0.

Table 8-5. Correspondence Between Source Files and Object Files

File Type	Source File	Object File
Startup routine	cstart*.asm[Notes 1, 2]	s0*.rel[Notes 2, 3]
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.

2. ‘b’ is startup routine for boot area, and ‘e’ is that for flash area.

3. *: 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> Including include files
$INCLUDE (macro.inc)
                              <2> Library switch
                              <3> Symbol definitions

BRKSW    EQU    1      ;brk,sbrk,calloc,free,malloc,realloc function use
EXITSW   EQU    1      ;exit,atexit function use
$_IF (_STATIC)
RANDSW   EQU    0      ;rand,srand  function use
DIVSW    EQU    0      ;div         function use
LDIVSW   EQU    0      ;ldiv        function use
FLOATSW  EQU    0      ;floating point  variable use
$ELSE
RANDSW   EQU    1      ;rand,srand  function use
DIVSW    EQU    1      ;div         function use
LDIVSW   EQU    1      ;ldiv        function use
FLOATSW  EQU    1      ;floating point  variable use
$ENDIF
STRTOKSW EQU    1      ;strtok      function use

       PUBLIC   _@cstart,_@cend

$_IF(BRKSW)
       PUBLIC   _@BRKADR,_@MEMTOP,_@MEMBTM
                :
$ENDIF
                              <4> External reference declaration of symbol for stack resolution
       EXTRN    _main,_@STBEG,_hdwinit
$_IF(EXITSW)
       EXTRN    _exit
$ENDIF

                              <5> External reference declaration of label for ROMization processing
       EXTRN     _?R_INIT,_?R_INIS,_?DATA,_?DATS
                                 <6> Securing area for standard library
@@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 **RA78K0 Assembler Package Operation User's Manual (U16629E)**).

[Example when -sSTACK is specified in linking]

Create lk78k0.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 lk78k0.dr in the smp78k0\cc78k0 directory.

```
              First address     Size
                      ↓           ↓
memory  SDR:   (0FE20h, 00098h)
memory  STACK:   (xxxxh, xxxh) <------  Specify the first address and size here,
                                        then specify lk78k0.dr by the -d linker
                                        option.
                                        (Example  -dlk78k0.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 <7> to <10>.

[Initial settings in cstart.asm]

```
                                             <7> Reset vector setting
@@VECT00 CSEG  AT  0
         DW    _@cstart

@LCODE   CSEG
_@cstart:
         SEL   RB0                           <8> Register bank setting
                                             <9> SP (stack pointer) setting
         MOVW  SP,#_@STBEG                   ;SP <- stack begin address
         CALL  !_hdwinit                     <10> Hardware initialization function call
$ENDIF
                :
$_IF(BRKSW OR EXITSW OR RANDSW OR FLOATSW)
        MOVW    AX,#0
$ENDIF
                :
```

<7> 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
```

<8> Register bank setting

Register bank RB0 is set as the work register.

<9> 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.

<10> 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 do not 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 four 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 value Note
_ _sreg int    si = 0;          (c) sreg variable with initial value
_ _sreg char   sc;              (d) sreg variable without initial value Note
main ()
{
              :
}
```

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

- Figure 8-1 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 **Table 8-7 RAM Area for Initial Values (Copy Destination)**). Tables 8-6 and 8-7 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.

Figure 8-1. ROMization Processing

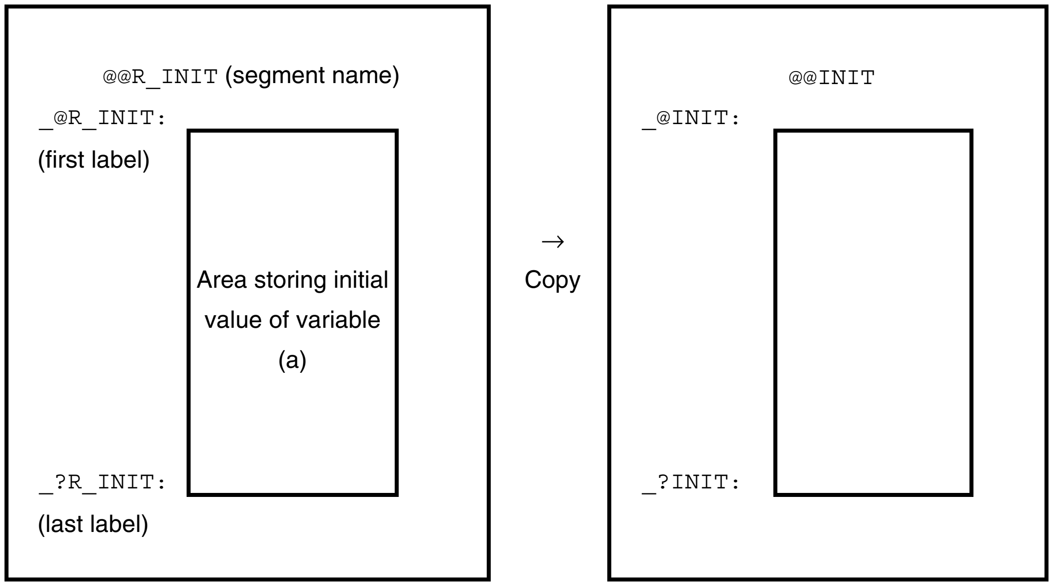


Table 8-6. ROM Area for Initial Values

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

Table 8-7. 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 <11> to <13>.

[Starting main function and postprocessing]

```
        CALL    !_main          ;main();        ;<11> Starting main function
$_IF(EXITSW)
        MOVW    AX,#0
        CALL    !_exit          ;exit(0);       ;<12> Starting exit function
$ENDIF
        BR      $$
;
_@cend:
                                                ;<13> Definitions of segments and labels
@@R_INIT CSEG   UNITP                           ;used in ROMization processing
_@R_INIT:
@@R_INIS CSEG   UNITP
_@R_INIS:
@@INIT   DSEG   UNITP
_@INIT:
@@DATA   DSEG   UNITP
_@DATA:
@@INIS   DSEG   SADDRP
_@INIS:
@@DATS   DSEG   SADDRP
_@DATS:
@@CALT   CSEG   CALLT0
@@CALF   CSEG   FIXED
@@CNST   CSEG   UNITP
@@BITS   BSEG
;
         END
```

<11> Starting main function

The main function is called.

<12> Starting exit function

The exit function is called if needed.

<13> Definitions of segments and labels used in ROMization processing

The segments and labels used in each variable (a) to (d) (see **8.3.2 (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    UNITP      ;<1> Definition of labels used in ROMization processing
_?R_INIT:
@@R_INIS CSEG    UNITP
_?R_INIS:
@@INIT   DSEG    UNITP
_?INIT:
@@DATA   DSEG    UNITP
_?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 **8.3.2 (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 CC78K0 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 (mkstup.sh) in the src\cc78k0\bat directory to assemble the revised source file (cstart*.asm) (*: alphanumeric symbols).

• Symbols used in standard library functions

If the library functions listed in Table 8-8 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.

Table 8-8. Symbols Used in Library Functions

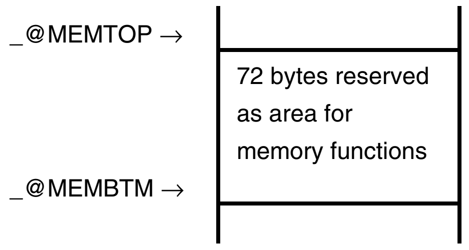
Library Function Name	Symbols Used
brk sbrk strtol strtoul malloc calloc realloc free	_errno _@MEMTOP _@MEMBTM _@BRKADR
exit	_@FNCTBL _@FNCENT
rand srand	_@SEED
div	_@DIVR
ldiv	_@LDIVR
strtok	_@TOKPTR
atof strtod Mathmatical function Floating-point runtime library	_errno

• Area used in memory functions

If the size of the area used by a memory function is defined by the user, this is explained in the following example.

Example) If you want to reserve 72 bytes for use by memory functions, make the following changes to the initial settings of the startup routine.

_@MEMTOP: DS $\boxed{72}$
_@MEMBTM:



If the specified size is too big to be stored in the RAM area, errors may occur when linking.
In this case, decrease the size specified as shown below, or avoid by correcting the link directive file. For correction of the link directive file, see **(2) Link directive file**.

Example) To decrease the specified size

_@MEMTOP: DS $\boxed{72}$ → Change to 40

(2) Link directive file

How to create a link directive file is explained. Specify a file created using the -D option when linking to match the actual target system. Heed the following cautions when creating the file (for the detailed description method for a link directive, see **RA78K0 Assembler Package Operation User's Manual (U16629E)**).

- The CC78K0 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 FEB8H to FEDFH for a normal model. When a static model is specified with the -SM[n] option, the part of saddr area [FED0H to FEDFH] is used as the common area.

  (Normal model)
  (a) Arguments of runtime library [FEB8H to FEBFH]
  (b) Arguments or automatic variables of norec function [FEC0H to FECFH]
  (c) register variable when the -qr option is specified [FED0H to FEDFH]
  (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 [FED0H to FEDFH]

The following shows an example of changing RAM size with a link directive file (lk78k0.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.

```
<lk78k0.dr>
                          First address  Size
  memory RAM:      (0FB00h,    00320h) → Make this size larger.
                                          (also change the first address if necessary)
  memory SDR:      (0FE20h,    00098h)
  merge @@INIS: =SDR  ↑
  merge @@DATS: =SDR  ├→ Specifies the location of the segment.
  merge @@BITS: =SDR  ↓
```

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 **CHAPTER 11** in **CC78K0 Language User's Manual (U16628E)**).
If the result of changing the location of a segment does not provide enough memory for the location, change the corresponding memory statement.

(3) When using RTOS

Initialization routines are respectively provided for RX78K0 and CC78K0 as samples (assembler format). Therefore, when using RX78K0 and CC78K0 in combination, changes must be performed so as to include the processing actions required for each in a single initialization routine.

Here, an example of the editing method is described by adding processing described in startup.asm (initialization routine provided for RX78K0) to cstart.asm (initialization routine provided for CC78K0). Ver. 3.50 is assumed for CC78K0.

Remark cstart.asm is a version that uses a standard library without ROMization.

<1> The following EXTRN declaration required for RX78K0 is added.

[After change]

```
EXTRN     sys_inf,?sysrt
```

<2> The EXTRN declarations of the main and exit functions described in cstart.asm are deleted. If the stack area is secured by the user (when using task stack other than initial task), the EXTRN declaration of _@STBEG is also deleted. (The _@STBEG area is automatically secured by specifying the -s option during linking.)

[Before change]

```
      EXTRN   _main,_@STBEG,_hdwinit
$_IF(EXITSW)
      EXTRN   _exit
$_ENDIF
```

[After change]

```
EXTRN _@STBEG,_hdwinit
```

The EXITSW setting locations are also changed.

[Before change]

```
EXITSW     EQU    1
```

[After change]

```
EXITSW     EQU    0
```

<3> The next location is edited (or vcttbl.asm is edited) to avoid redundancy with vector 0 of vcttbl.asm provided for RX78K0. If _@cstart is not used, change it to the symbol to be used.

[Before change]

```
@@VECT00    CSEG    AT     0
      DW    _@cstart
```

<4> Prior to selecting the register banks, select the interrupt disabled state.

[Before change]

```
SEL    RB0
```

[After change]

```
DI
SEL    RB0
```

<5> If _@STBEG of the stack area is not used, change the following location.

[Before change]

```
MOVW   SP,#_@STBEG       ;SP <- stack begin address
```

<6> Describe the hardware initialization processing required for the user system to the hardware initialization function (hdwinit).

<7> When using RX78K0, delete the following location because the main and exit functions are not needed. Delete also processing that is not required for RX78K0 control, and add processing for transferring control to the RX78K0 system initialization routine.

[Before change]

```
      CALL    !_main      ;main();
$_IF(EXITSW)
      MOVW    AX,#0
      CALL    !_exit      ;exit(0);
$_ENDIF
      BR      $$
```

[After change]

```
      MOVW    HL,#sys_inf
      MOV     A,[HL]
      MOV     X,A
      MOV     A,[HL+1]
      BR      AX
```

<Example of initialization routine after editing>

```
; Copyright (C) NEC Electronics Corporation 20xx
; NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
; All rights reserved by NEC Electronics Corporation.
; This program must be used solely for the purpose for which
; it was furnished by NEC Electronics Corporation. No part of this
; program may be reproduced or disclosed to others, in any
; form, without the prior written permission of NEC Electronics
; Corporation. Use of copyright notice does not evidence
; publication of the program.
;====================================
;      W-1    cstart
;
;      Version x.xx  xx Xxx 20xx
;====================================
       NAME     @cstart

$INCLUDE (def.inc)
$INCLUDE (macro.inc)

;------------------------------------------------------------------------------
; declaration of symbol
;
; attention):  change to EQU value 1 -> 0  if necessary
;------------------------------------------------------------------------------
;
BRKSW    EQU 1        ;brk, sbrk, calloc, free, malloc, realloc function use
EXITSW   EQU 0        ;exit, atexit function use               ;Change location
$_IF (_STATIC)
RANDSW   EQU 0        ;rand, srand  function use
DIVSW    EQU 0        ;div          function use
LDIVSW   EQU 0        ;ldiv         function use
FLOATSW  EQU 0        ;floating point variables use
$ELSE
RANDSW   EQU 1        ;rand,srand   function use
DIVSW    EQU 1        ;div          function use
LDIVSW   EQU 1        ;ldiv         function use
FLOATSW  EQU 1        ;floating point variables use
$ENDIF
STRTOKSW EQU 1        ;strtok       function use

   PUBLIC _@cstart, _@cend

$_IF(BRKSW)
   PUBLIC _@BRKADR, _@MEMTOP, _@MEMBTM
$ENDIF
$_IF(EXITSW)
   PUBLIC _@FNCTBL, _@FNCENT
```

```
$ENDIF
$_IF(RANDSW)
   PUBLIC     _@SEED
$ENDIF
$_IF(DIVSW)
   PUBLIC _@DIVR
$ENDIF
$_IF(LDIVSW)
   PUBLIC _@LDIVR
$ENDIF
$_IF(STRTOKSW)
   PUBLIC _@TOKPTR
$ENDIF
$_IF(BRKSW OR FLOATSW)
   PUBLIC      _errno
$ENDIF

;-----------------------------------------------------------------------------
; external declaration of symbol for stack area
;
; _@STBEG has value of the end address +1 of compiler's stack area.
; _@STBEG is created by linker with -S option.
; Accordingly, specify the -S option when linking.
;-----------------------------------------------------------------------------
  EXTRN  sys_inf, ?sysrt                                 ;Addition location
  EXTRN  _@STBEG, _hdwinit                               ;Change location

;-----------------------------------------------------------------------------
; external declaration of label for ROMable
;-----------------------------------------------------------------------------
  EXTRN  _?R_INIT,_?R_INIS,_?DATA,_?DATS

;-----------------------------------------------------------------------------
; allocation area which library uses
;
; _@FNCTBL    top address of area used in atexit function
; _@FNCENT    total number of functions registered by the atexit function
; _@SEED      sequence of pseudo-random numbers
; _@DIVR      a result of div library
; _@LDIVR     a result of ldiv library
; _@TOKPTR    pointer which strtok function uses
; _errno      errno number code
; _@MEMTOP    top address of area which memory management functions use
; _@MEMBTM    bottom address of area which memory management functions use
; _@BRKADR    break value
;-----------------------------------------------------------------------------
@@DATA DSEG   UNITP

$ _IF(EXITSW)
_@FNCTBL:     DS      2*32
_@FNCENT:     DS      2
$ENDIF
$_IF(RANDSW)
_@SEED;       DS      4
$ENDIF
$_IF(DIVSW)
_@DIVR:       DS      4
$ENDIF
$_IF(LDIVSW)
```

```
_@LDIVR:      DS     8
$ENDIF
$_IF(STRTOKSW)
_@TOKPTR:     DS     2
$ENDIF
$_IF(BRKSW OR FLOATSW)
_errno:       DS     2
$ENDIF
$_IF(BRKSW)
_@BRKADR:     DS     2
_@MEMTOP:     DS     32
_@MEMBTM:
$ENDIF

@@VECT00   CSEG       AT          0                            ;Change if required
      DW   _@cstart                                            ;

@LCODE     CSEG

_@cstart:
;------------------------------------------------------------------------------
; setting the register bank RB0 as work register set
;------------------------------------------------------------------------------
  DI                                                           ;Addition location
  SEL    RB0
;------------------------------------------------------------------------------
; setting the stack pointer
;
; _@STBEG is created by linker with -S option.
;------------------------------------------------------------------------------
  MOVW   sp, #_@STBEG ;SP <- stack begin address               ;Change if required
  CALL   !_hdwinit
;------------------------------------------------------------------------------
; errno and _@FNCENT are initialized to 0
;
; The positive error number will be set by several libraries at called them.
;------------------------------------------------------------------------------
$_IF(BRKSW OR EXITSW OR RANDSW OR FLOATSW)
   MOVM    AX, #0
$ENDIF
$_IF(BRKSW OR FLOATSW)
   MOVW    !_errno, AX    ;errno <- 0
$ENDIF
$_IF(EXITSW)
   MOVW    !_@FNCENT, AX  ;FNCENT <- 0
$ENDIF
;------------------------------------------------------------------------------
; initializing _@SEED as 1
;
; Pseudo-random sequence is decided by _@SEED value. This value can be set by
; srand function. If rand is called before srand, the random sequence will
; be the same as when srand is called with a _@SEED value as 1 at first.
;------------------------------------------------------------------------------
$_IF(RANDSW)
   MOVM  !_@SEED+2, AX
   INC   X
   MOVW  !_@SEED, AX      ;SEED <- 1
$ENDIF
;------------------------------------------------------------------------------
```

```
; setting _@MEMTOP address to _@BRKADR
;-----------------------------------------------------------------------------
$_IF(BRKSW)
   MOVW  AX, #_@MEMTOP
   MOVW  !_@BRKADR, AX    ;BRKADR <- #MEMTOP
$ENDIF

;-----------------------------------------------------------------------------
;  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:
   MOVW    AX, HL
   CMPW    AX, #_?DATA
   BZ      $LDATA2
   MOV     A, #0
   MOV     [HL],A
   INCW    HL
   BR      $LDATA1
LDATA2:
; copy sreg variables having initial value
   MOVW    HL, #_@R_INIS
   MOVW    DE, #_@INIS
LINIS1:
   MOVW    AX, HL
   CMPW    AX, #_?R_INIS
   BZ      $LINIS2
   MOV     A, [HL]
   MOV     [DE], A
   INCW    HL
   INCW    DE
   BR      $LINIS1
LINIS2:
   MOVW    HL, #_@DATS
; copy sreg variables which doesn't have initial value
LDATS1:
   MOVW    AX, HL
   CMPW    AX, #_?DATS
   BZ      $LDATS2
   MOV     A, #0
   MOV     [HL],A
   INCW    HL
   BR      $LDATS1
LDATS2:
```

```
;-----------------------------------------------------------------------------
; branches to the reset routine for system initialization of RX78K0
;-----------------------------------------------------------------------------
   MOVW    HL, #sys_inf                                        ;
   MOV     A, [HL]                                             ;
   MOV     X, A                                                ;Change location
   MOW     A, [HL]                                             ;
   BR      AX                                                  ;
;

_@cend:
;-----------------------------------------------------------------------------
; define segment and label used by ROMable processing
;-----------------------------------------------------------------------------
@@R_INIT      CSEG    UNITP
_@R_INIT:
@@R_INIS      CSEG    UNITP
_@R_INIS:
@@INIT DSEG   UNITP
_@INIT:
@@DATA DSEG   UNITP
_@DATA:
@@INIS DSEG   SADDRP
_@INIS:
@@DATS DSEG   SADDRP
_@DATS:
@@CALT CSEG   CALLT0
@@CALF CSEG   FIXED
@@CNST CSEG   UNITP
@@BITS BSEG
;
END
```

8.4 ROMization Processing in Startup Module for Flash Area

The startup modules for flash differ with the ordinary startup modules in the following points.

Table 8-9. ROM Area Section for Initialization Data

Variable Type	Segment	First Label	Terminal Label
External variable with initial value (a)	@ER_INIT CSEG UNITP	E@R_INIT	E?R_INIT
sreg variable with initial value (c)	@ER_INIS CSEG UNITP	E@R_INIS	E?R_INIS

Table 8-10. RAM Area Section for Copy Destination

Variable Type	Segment	First Label	Terminal Label
External variable with initial value (a)	@EINIT DSEG UNITP	E@INIT	E?INIT
External variable without initial value (b)	@EDATA DSEG UNITP	E@DATA	E?DATA
sreg variable with initial value (c)	@EINIS DSEG SADDRP	E@INIS	E?INIS
sreg variable without initial value (d)	@EDATS DSEG SADDRP	E@DATS	E?DATS

- In the startup module, the following labels are added at the head of each segment in ROM area and RAM area.

  E@R_INIT, E@R_INIS, E@INIT, E@DATA, E@INIS, E@DATS

- In the terminal module, the following labels are added at the terminal of each segment in ROM area and RAM area.

  E?R_INIT, E?R_INIS, E?INIT, E?DATA, E?INIS, E?DATS

- The startup module copies the contents from the first label address of each segment in ROM area to the terminal label address –1, to the area from the first label address of each segment in RAM area.

- Zeros are embedded from E@DATA to E?DATA, and from E@DATS to E?DATS.

CHAPTER 9 ERROR MESSAGES

9.1 Error Message Format

The error message format is as follows.

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

Examples

```
prime.c(8) : F712 Declaration syntax
prime.c(8) : F301 Syntax error
prime.c(8) : F701 External definition syntax
prime.c(19) : W745 Expected function prototype
```

However, the following output format is used only for the internal errors F101, F103, and F104.

```
[xxx.c <yyy> zzz] F101 Internal error
[xxx.c <yyy> zzz] F103 Intermediate file error
[xxx.c <yyy> zzz] F104 Illegal use of register
```

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

9.2 Types of Error Messages

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

(1) Error message for a command line
(2) Error message for an internal error or memory
(3) Error message for a character
(4) Error message for a 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 or 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.

```
A/F/Wnnn
```

A : ABORT

After the error message is output, the compile processing ends immediately. The object module file and the assembler source module file are not output.

F : FATAL

After the error message is output, the error portion is ignored and processing continues. The object module file and the assembler source module file are not output.

W : WARNING

After the warning message is output, processing continues. The file specified by the option is output.

nnn (3-digit number)

From 001	Error message for a command line
From 101	Error message for an internal error or memory
From 201	Error message for a character
From 301	Error message for a configuration element
From 401	Error message for conversion
From 501	Error message for an expression
From 601	Error message for a statement
From 701	Error message for a declaration or a function definition
From 801	Error message for a preprocessing directive
From 901	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.

(1) Error message for a command line <from 001> (1/3)

A001	Message	Missing input file
	Cause	The input source file name was not specified.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the file name correctly.
A002	Message	Too many input files
	Cause	Multiple input source file names are specified.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the file name correctly.
A003	Message	Unrecognized string
	Cause	An item other than an option was specified on the interactive command line.
A004	Message	Illegal file name file name
	Cause	Either the format, characters, or number of characters in the specified file name are incorrect.
A005	Message	Illegal file specification
	Cause	An illegal file name was specified.
A006	Message	File not found
	Cause	The specified input file does not exist.
A007	Message	Input file specification overlapped file name
	Cause	Duplicate input file names were specified.
A008	Message	File specification conflicted file name
	Cause	Duplicate I/O file names were specified.
A009	Message	Unable to make file file name
	Cause	Since the specified output file already exists as a read-only file, it cannot be created.
A010	Message	Directory not found
	Cause	A drive or directory not existed is included in the output file name.
A011	Message	Illegal path
	Cause	An illegal path name was specified in the option setting the path name in the parameter.
A012	Message	Missing parameter ‘option’
	Cause	A required parameter is not specified.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the parameter correctly.
A013	Message	Parameter not needed ‘option’
	Cause	An unnecessary option parameter was specified.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the parameter correctly.
A014	Message	Out of range ‘option’
	Cause	The specified value of the option parameter is out of range.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the value correctly.
A015	Message	Parameter is too long
	Cause	The number of characters in the option parameter exceeded the limit.

(1) Error message for a command line <from 001> (2/3)

A016	Message	Illegal parameter ‘option’
	Cause	There is a syntax error in the option parameter.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the option correctly.
A017	Message	Too many parameters
	Cause	The total number of option parameters exceeds the limit.
A018	Message	Option is not recognized ‘option’
	Cause	An incorrect option was specified.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and input the option correctly.
A019	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.
A020	Message	Parameter file read error
	Cause	The parameter file read failed.
A021	Message	Memory allocation failed
	Cause	Memory allocation failed.
W022	Message	Same category option specified – ignored ‘option’
	Cause	Conflicting options had duplicate specifications.
	Compiler	The option specified later is validated and processing continues.
W023	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.
A024	Message	Illegal chip specifier on command line
	Cause	The device type in the command line is incorrect.
W029	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**).
W031	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**).
W032	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**).
W033	Message	Same category option specified ‘option’
	Cause	Conflicting options had duplicate specifications.
	Response	“Please enter ‘cc78k0--’ if you want help message” is output. Use the --, -?, or -H option to reference the help file and correct the input.

(1) Error message for a command line <from 001> (3/3)

W036	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**).
W037	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**).
W038	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.
W039	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.
W040	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.
W041	Message	'-SM' option specified - ignored '-QR'
	Cause	Since the static model specification option -SM is specified, the register optimization option -QR is ignored.
W043	Message	'-ZO' option specified - ignored '-ZR'
	Cause	Since the old specification function interface specification option -ZO is specified, the pascal function interface specification option -ZR is ignored.
W044	Message	'-SM' option specified - ignored '-ZO'
	Cause	Since the static model specification option -SM is specified, the old specification function interface specification option - ZO is ignored.
W045	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.
W046	Message	'-ZF' option specified - regarded as '-QL1'
	Cause	Since the flash area object creation option -ZF is specified, after -QL2 in the library replace option of constant code pattern -QL is regarded as - QL1.
W052	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).
W053	Message	'-ZO' option specified - ignored '-ZD'
	Cause	Old function specification interface specification option (-ZO) is specified, so that option (-ZO) using library supporting prologue/epilogue is ignored.
W054	Message	'-ZF' option specified - ignored '-ZD'
	Cause	Flash area object generation option (-ZF) is specified, so that option (-ZD) using library supporting prologue/epilogue is ignored.
W055	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.

(2) Error message for an internal error and memory <from 101>

F101	Message	Internal error
	Cause	An internal error occurred.
	Response	Contact support.
F102	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.
F103	Message	Intermediate file error
	Cause	The intermediate file contains errors.
	Response	Contact support.
F104	Message	Illegal use of register
	Cause	The register is incorrectly used.
F105	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.
A106	Message	Stack overflow 'overflow cause'
	Cause	The stack overflowed. The cause of the overflow is the stack or heap.
	Response	Contact support.
F108	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.
F109	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.
F110	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.
F111	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.
F113	Message	Compiler limit: too many local labels
	Cause	Number of local labels in one function exceeds the process limit.
	Response	The function itself is too large. Divide it.

(3) Error message for a character <from 201>

F201	Message	Unknown character 'hexadecimal number'
	Cause	Characters having the specified internal code cannot be recognized.
F202	Message	Unexpected EOF
	Cause	The file ended while the function was operating.
W203	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.

(4) Error message for configuration element <from 301> (1/3)

F301	Message	Syntax error
	Cause	A syntax error occurred.
	Response	Make sure there are no description errors in the source.
F303	Message	Expected identifier
	Cause	An identifier is required for the goto statement.
	Response	Correctly describe the identifier to be specified for the goto statement.
W304	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.
F305	Message	Compiler limit : too many identifiers with block scope
	Cause	There are too many symbols having block scope in one block.
F306	Message	Illegal index , indirection not allowed
	Cause	An index is used in an expression that does not take a pointer value.
F307	Message	Call of non-function 'variable name'
	Cause	The variable name is used as a function name.
F308	Message	Improper use of a typedef name
	Cause	The typedef name is improperly used.
F309	Message	Unused 'variable name'
	Cause	The specified variable is declared in the source, but is never used.
F310	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.
F311	Message	Number syntax
	Cause	The constant expression is illegal.
F312	Message	Illegal octal digit
	Cause	This is illegal as an octal digit.
F313	Message	Illegal hexadecimal digit
	Cause	This is illegal as a hexadecimal digit.
F314	Message	Too big constant
	Cause	The constant is too large and cannot be represented.

(4) Error message for configuration element <from 301> (2/3)

W315	Message	Too small constant
	Cause	The constant is too small and cannot be represented.
F316	Message	Too many character constants
	Cause	The character constant exceeds two characters.
F317	Message	Empty character constant
	Cause	The character constant ‘ ’ is empty.
F318	Message	No terminated string literal
	Cause	There is no double quote “ ” at the end of the string.
F319	Message	Changing string literal
	Cause	A character string literal is rewritten.
W320	Message	No null terminator in string literal
	Cause	The null character is not added to the character string literal.
F321	Message	Compiler limit : too many characters in string literal
	Cause	The number of characters in the character string literal exceeded 509.
F322	Message	Ellipsis requires three periods
	Cause	The compiler detected “..”, but “...” is required.
F323	Message	Missing ‘delimiter’
	Cause	The delimiter is incorrect.
F324	Message	Too many }’s
	Cause	The ‘{’ and ‘}’ are incorrectly paired.
F325	Message	No terminated comment
	Cause	The comment is not terminated by “*/”.
F326	Message	Illegal binary digit
	Cause	This is illegal as a binary digit.
F327	Message	Hex constants must have at least one hex digit
	Cause	At least one hexadecimal digit is required in a hexadecimal constant representation.
W328	Message	Unrecognized character escape sequence ‘character’
	Cause	The escape sequence cannot be correctly recognized.
F329	Message	Compiler limit : too many comment nesting
	Cause	The number of nesting levels of comments exceeded the limit of 255.
W330	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.
W331	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.
W333	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.
F334	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.

(4) Error message for configuration element <from 301> (3/3)

W335	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.
W337	Message	'-ZO' option specified - ignored '__pascal' in this file
	Cause	Since the old specification function interface specification option -ZO is specified, __pascal keyword is ignored in this file.
W339	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.
W340	Message	Unreferenced label 'label name'
	Cause	The specified label has been defined, but has not been referenced even once.

(5) Error message for conversion <from 401> (1/2)

W401	Message	Conversion may lose significant digits
	Cause	A long was converted into int. Be careful the value may be lost.
F402	Message	Incompatible type conversion
	Cause	An illegal type conversion took place in the assignment statement.
F403	Message	Illegal indirection
	Cause	The * operator is used in an integer type expression.
F404	Message	Incompatible structure type conversion
	Cause	The types on both sides of an assignment statement to a structure or structure pair differ.
F405	Message	Illegal lvalue
	Cause	This is an illegal left value.
F406	Message	Cannot modify a const object 'variable name'
	Cause	A variable with the const attribute is rewritten.
F407	Message	Cannot write for read / only sfr 'SFR name'
	Cause	Tried to write to a read-only sfr.
F408	Message	Cannot read for write/only sfr 'SFR name'
	Cause	Tried to read a write-only sfr.
F409	Message	Illegal SFR access 'sfr name'
	Cause	Illegal data was read from or written to an sfr.
W410	Message	Illegal pointer conversion
	Cause	A pointer and an object other than a pointer are converted.
W411	Message	Illegal pointer combination
	Cause	Different types are mixed in the same pointer combination.
W412	Message	Illegal pointer combination in conditional expression
	Cause	Different types in a pointer combination are used in a conditional expression.

(5) Error message for conversion <from 401> (2/2)

F413	Message	Illegal structure pointer combination
	Cause	Pointers to structures with different types are mixed.
F414	Message	Expected pointer
	Cause	A pointer is required.

(6) Error message for an expression <from 501> (1/3)

F501	Message	Expression syntax
	Cause	The expression contained a syntax error.
F502	Message	Compiler limit : too many parentheses
	Cause	The nesting of parentheses in the expression exceeded 32.
W503	Message	Possible use of ‘variable name’ before definition
	Cause	The variable is used before a value is assigned to it.
W504	Message	Possibly incorrect assignment
	Cause	The main operators in conditional expressions, such as if, while, and do statements, are assignment operators.
W505	Message	Operator ‘operator’ has no effect
	Cause	The operator has no effect in the program. This is probably due to a description error.
F507	Message	Expected integral index
	Cause	Only an integer type expression is allowed in the index of an array.
W508	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.
W509	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.
W510	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.
W511	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.
F512	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.
F513	Message	Illegal structure / union member ‘member name’
	Cause	A member that is referenced in the structure and not defined is indicated.
F514	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.

(6) Error message for an expression <from 501> (2/3)

W515	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.
F516	Message	Zero sized structure 'structure name'
	Cause	The size of the structure is zero.
F517	Message	Illegal structure operation
	Cause	An operator that cannot be used in a structure is used.
F518	Message	Illegal structure / union comparison
	Cause	Two structures or unions cannot be compared.
F519	Message	Illegal bit field operation
	Cause	There is an illegal description for a bit field.
F520	Message	Illegal use of pointer
	Cause	The only operators that can be used on pointers are addition, subtraction, assignment, relational, indirection (*), and member reference (->).
F521	Message	Illegal use of floating
	Cause	An operator that cannot be used on floating-point variables is used.
W522	Message	Ambiguous operators need parentheses
	Cause	Two shift, relational, and bit logical operators appear continuously without parentheses.
F523	Message	Illegal bit, boolean type operation
	Cause	An illegal operation is performed on bit or boolean type variables.
F524	Message	'&' on constant
	Cause	A constant address is not obtained.
F525	Message	'&' requires lvalue
	Cause	The '&' operator can only be used in an expression assigned to the left value.
F526	Message	'&' on register variable
	Cause	The address of a register variable is not obtained.
F527	Message	'&' on bit, boolean ignored
	Cause	The address of a bit field, or bit or boolean type variable is not obtained.
W528	Message	'&' is not allowed array / function, ignored
	Cause	The & operator does not have to be applied to an array name or function name.
F529	Message	Sizeof returns zero
	Cause	The value of the sizeof expression becomes zero.
F530	Message	Illegal sizeof operand
	Cause	The operand of the sizeof expression must be an identifier or a type name.

(6) Error message for an expression <from 501> (3/3)

F531	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.
F532	Message	Pointer on left, needs integral right : ‘operator’
	Cause	Since the left operand is a pointer, the right operand must be an integral value.
F533	Message	Invalid left-or-right operand : ‘operator’
	Cause	The left or right operand is illegal for the operator.
F534	Message	Divide check
	Cause	The divisor of the / operation or % operation is zero.
F535	Message	Invalid pointer addition
	Cause	Two pointers are not added.
F536	Message	Must be integral value addition
	Cause	Only integral values can be added to a pointer.
F537	Message	Illegal pointer subtraction
	Cause	The subtraction between pointers must be for pointers having the same type.
F538	Message	Illegal conditional operator
	Cause	The conditional operator is not correctly described.
F539	Message	Expected constant expression
	Cause	A constant expression is required.
W540	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.
F541	Message	Function argument has void type
	Cause	The argument of the function has the void type.
W543	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.
F544	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.
F546	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.
F549	Message	‘-SM’ option specified - recursive function is not allowed
	Cause	The static model specification option -SM is specified. Recursive call is not allowed.

(7) Error message for a statement <from 601>

F602	Message	Compiler limit : too many characters in logical source line
	Cause	The number of characters in a logical source line exceeded 2048.
F603	Message	Compiler limit : too many labels
	Cause	The number of labels exceeded 33.
F604	Message	Case not in switch
	Cause	The case statement is not described in the correct position.
F605	Message	Duplicate case ‘label name’
	Cause	The same case label is described two or more times in a switch statement.
F606	Message	Non constant case expression
	Cause	Something other than an integral constant is specified in a case statement.
F607	Message	Compiler limit : too many case labels
	Cause	The number of case labels in the switch statement exceeded 257.
F608	Message	Default not in switch
	Cause	The default statement is not described in the correct position.
F609	Message	More than one ‘default’
	Cause	The default statement is described multiple times in the switch statement.
F610	Message	Compiler limit : block nest level too depth
	Cause	The block nesting exceeded 45.
F611	Message	Inappropriate ‘else’
	Cause	There is no correspondence between if and else.
W613	Message	Loop entered at top of switch
	Cause	A while, do, or for is specified immediately after the switch statement.
W615	Message	Statement not reached
	Cause	The statement is never reached.
F617	Message	Do statement must have ‘while’
	Cause	A while is required at the end of a do.
F620	Message	Break / continue error
	Cause	The positions of the break and continue statements are incorrect.
F621	Message	Void function ‘function name’ cannot return value
	Cause	A function declared as void returns a value.
W622	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.
F623	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.

(8) Error message for a declaration and function definition <from 701> (1/5)

F701	Message	External definition syntax
	Cause	The function is not correctly defined.
F702	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.
F703	Message	Function has illegal storage class
	Cause	The function is specified with an illegal storage class.
F704	Message	Function returns illegal type
	Cause	The return value of the function is an illegal type.
F705	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.
F706	Message	Parameter list error
	Cause	The function parameter list contains errors.
F707	Message	Not parameter ‘character string’
	Cause	Something other than a parameter is declared in a function definition.
W708	Message	Already declared symbol ‘variable name’
	Cause	The same variable has already been declared.
F709	Message	Different bank direction specified same file
	Cause	A different bank was specified for the same file.
F710	Message	Illegal storage class
	Cause	The auto and register declarations are outside the function or the boolean variable is defined inside the function.
F711	Message	Undeclared ‘variable name’; function ‘function name’
	Cause	An undeclared variable is used.
F712	Message	Declaration syntax
	Cause	The declaration statement does not match the syntax.
F713	Message	Redefined ‘variable name’
	Cause	Two or more of the same variables are defined.
	Response	Set the variable definition once.
W714	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, see **CHAPTER 11** in the **Language (U16628E)** manual.
F715	Message	Too many sreg variables
	Cause	There are too many declarations of sreg variables.
F716	Message	Not allowed automatic data in noauto function
	Cause	Automatic variables cannot be used in the noauto function.

(8) Error message for a declaration and function definition <from 701> (2/5)

F717	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, see **CHAPTER 11** in the **Language (U16628E)** manual.
F718	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, see **CHAPTER 11** in the **Language (U16628E)** manual.
F719	Message	Illegal use of type
	Cause	An illegal type name is used.
F720	Message	Illegal void type for ‘identifier’
	Cause	The identifier is declared by void.
W721	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.
F723	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.
F724	Message	Structure redefinition
	Cause	The same structure is redefined.
W725	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.
F726	Message	Function cannot be structure / union member
	Cause	A function cannot be a member of a structure or a union.
F727	Message	Unknown size structure / union ‘name’
	Cause	Structures or unions have undefined sizes.
F728	Message	Compiler limit : too many structure / union members
	Cause	The members in a structure or union exceeded 256.
F729	Message	Compiler limit : structure / union nesting
	Cause	The nesting of structures or unions exceeded 15.
F730	Message	Bit field outside of structure
	Cause	A bit field is declared outside of the structure.
F731	Message	Illegal bit field type
	Cause	A type other than an integral type is specified in a bit field type.
F732	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.

(8) Error message for a declaration and function definition <from 701> (3/5)

F733	Message	Negative bit field size
	Cause	The number of bit specifications in a bit field declaration is negative.
F734	Message	Illegal enumeration
	Cause	The enumeration type declaration does not match the syntax.
F735	Message	Illegal enumeration constant
	Cause	The enumeration constant is illegal.
F736	Message	Compiler limit : too many enumeration constants
	Cause	The number of enumeration constants exceeded 255.
F737	Message	Undeclared structure / union / enum tag
	Cause	A tag is not declared.
F738	Message	Compiler limit : too many pointer modifying
	Cause	The number of indirection operators (*) exceeded 12 in a pointer definition.
F739	Message	Expected constant
	Cause	A variable is used in the index in an array declaration.
F740	Message	Negative subscript
	Cause	The specification of the size of an array is negative.
F741	Message	Unknown size array ‘array name’
	Cause	The size of an array is undefined.
	Response	Specify the size of the array.
F742	Message	Compiler limit : too many array modifying
	Cause	The array declaration exceeds 12 dimensions.
F743	Message	Array element type cannot be function
	Cause	An array of functions is not allowed.
W744	Message	Zero sized array ‘array name’
	Cause	The number of elements of the defined array is zero.
W745	Message	Expected function prototype
	Cause	The function prototype is not declared.
F747	Message	Function prototype mismatch
	Cause	The function prototype declaration contains errors.
	Response	Check whether the parameter and return value types match the function.
W748	Message	A function is declared as a parameter
	Cause	A function is declared as an argument.
W749	Message	Unused parameter ‘parameter name’
	Cause	The parameter is not used.
F750	Message	Initializer syntax
	Cause	The initialization does not match the syntax.
F751	Message	Illegal initialization
	Cause	The constant of an initial value setting does not match the type of the variable.

(8) Error message for a declaration and function definition <from 701> (4/5)

W752	Message	Undeclared initializer name ‘name’
	Cause	The initializer name is not declared.
F753	Message	Cannot initialize static with automatic
	Cause	The static variable cannot be initialized using an automatic variable.
F756	Message	Too many initializers ‘array name’
	Cause	There are more initial values than elements in the declared array.
F757	Message	Too many structure initializers
	Cause	There are more initial values than members in the declared structure.
F758	Message	Cannot initialize a function ‘function name’
	Cause	The function cannot be initialized.
F759	Message	Compiler limit : initializers too deeply nested
	Cause	The depth of the nesting of initialized elements exceeded the limit.
W760	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.
W761	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.
W762	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.
W763	Message	Cannot declare const with bit, boolean
	Cause	bit and boolean type variables cannot have const declarations.
	Compiler	A const declaration is ignored.
W764	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.
F765	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.
F766	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.
F770	Message	Parameters are not allowed for interrupt function
	Cause	An interrupt function cannot have arguments.
F771	Message	Interrupt function must be void type
	Cause	An interrupt function must have the void type.

(8) Error message for a declaration and function definition <from 701> (5/5)

F772	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.
F773	Message	Cannot call interrupt function
	Cause	An interrupt function cannot be called.
F774	Message	Interrupt function can't use with the other kind interrupts
	Cause	An interrupt function cannot be used in other types of interrupts.
F775	Message	Cannot call rtos_task function
	Cause	RTOS task cannot be called.
F776	Message	Cannot call ret_int / ret_wup except in rtos_interrupt_handler
	Cause	ret_int / ret_wup system call cannot be called except in the RTOS_INTERRUPT handler.
F777	Message	Not call ret_int / ret_wup in rtos_interrupt_handler
	Cause	ret_int / ret_wup system call is not called in the RTOS_INTERRUPT handler.
F778	Message	Cannot call ext_tsk in interrupt function
	Cause	ext_tsk system call cannot be called in the interrupt function/interrupt handler.
F779	Message	Not call ext_tsk in rtos_task
	Cause	ext_tsk system call is not called in the RTOS task.
F780	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.
F781	Message	'-SM' option specified - variable parameters are not allowed
	Cause	The static model specification option -SM is specified. Variable parameters are not allowed.
F782	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.
F783	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.
F784	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.
F785	Message	'-SM' option specified - expected function prototype
	Cause	The static model specification option -SM is specified. Function prototype declaration is absent.
F786	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.
F787	Message	Bit field type is char
	Cause	char type is specified for bit field type.
W792	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.
W793	Message	Variable parameters are not allowed for __pascal function - ignored __pascal
	Cause	Variable parameters cannot be specified for __pascal function. __pascal keyword is ignored.
W799	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) Error message for a preprocessing directive <from 801> (1/4)

F801	Message	Undefined control
	Cause	A symbol starting with # cannot be recognized as a keyword.
F802	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.
F803	Message	Unexpected non-whitespace before preprocess directive
	Cause	A character other than a whitespace character precedes the preprocess directive.
W804	Message	Unexpected characters following ‘preprocess directive’ directive - newline expected
	Cause	Extra characters follow the preprocess directive.
F805	Message	Misplaced else or elif
	Cause	The #if, #ifdef, and #ifndef do not correspond to #else and #elif.
F806	Message	Misplaced endif
	Cause	The #if, #ifdef, and #ifndef do not correspond to #endif.
F807	Message	Compiler limit : too many conditional inclusion nesting
	Cause	The nesting of conditional compiling exceeded 255.
F810	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 INC78K0.
F811	Message	Too long file name ‘file name’
	Cause	The file name is too long.
F812	Message	Include directive syntax
	Cause	The file name in the definition of the #include statement is not correctly enclosed by “ ” or < >.
F813	Message	Compiler limit : too many include nesting
	Cause	The nesting of the include files exceeded 8.
F814	Message	Illegal macro name
	Cause	The macro name is illegal.
F815	Message	Compiler limit: too many macro nesting
	Cause	The number of nesting macros exceeds 200.
W816	Message	Redefined macro name ‘macro name’
	Cause	The macro name is redefined.
W817	Message	Redefined system macro name ‘macro name’
	Cause	The system macro name is redefined.
F818	Message	Redeclared parameter in macro ‘macro name’
	Cause	The same identifier appears in the parameter list in the macro definition.
W819	Message	Mismatch number of parameter ‘macro name’
	Cause	The number of parameters when referencing differs from the number of parameters defined by #define.

(9) Error message for a preprocessing directive <from 801> (2/4)

F821	Message	Illegal macro parameter ‘macro name’
	Cause	The description enclosed by parentheses ( ) in the function format macro is illegal.
F822	Message	Missing) ‘macro name’
	Cause	The right parenthesis ‘)’ was not found in the same line as the #define definition in the function format macro.
F823	Message	Too long macro expansion ‘macro name’
	Cause	The actual argument during macro expansion is too long.
W824	Message	Identifier truncate to ‘macro name’
	Cause	The macro name is too long.
	Compiler	It is shortened to the displayed ‘macro name’.
W825	Message	Macro recursion ‘macro name’
	Cause	The #define definition becomes recursive.
F826	Message	Compiler limit : too many macro defines
	Cause	The number of macro definitions exceeded 10,000.
F827	Message	Compiler limit : too many macro parameters
	Cause	One macro definition had over 31 calling parameters.
F828	Message	Not allowed #undef for system macro name
	Cause	The system macro name is specified by #undef.
W829	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.
F830	Message	No chip specifier : #pragma pc ( )
	Cause	There is no device specifier.
F831	Message	Illegal chip specifier : #pragma pc (device type)
	Cause	The device specifier is illegal.
W832	Message	Duplicated chip specifier
	Cause	The device specifier is duplicated.
F833	Message	Expected #asm
	Cause	There is no #asm.
F834	Message	Expected #endasm
	Cause	There is no #endasm.
W835	Message	Too many characters in assembler source line
	Cause	A line in the assembler source is too long.

(9) Error message for a preprocessing directive <from 801> (3/4)

W836	Message	Expected assembler source
	Cause	There is no assembler source between #asm and #endasm.
W837	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.
F838	Message	Duplicated pragma VECT or INTERRUPT or RTOS_INTERRUPT 'character string'
	Cause	The #pragma VECT 'character string', INTERRUPT 'character string', or RTOS_INTERRUPT 'character string' is duplicated.
F839	Message	Unrecognized pragma VECT or INTERRUPT or RTOS_INTERRUPT 'character string'
	Cause	There is an unrecognized #pragma VECT 'character string', INTERRUPT 'character string', or RTOS_INTERRUPT 'character string'.
W840	Message	Undefined interrupt function 'function name'- ignored BANK or SP_SWITCH or LEAFWORK specified
	Cause	The save destination is specified for an undefined interrupt function.
	Compiler	Register bank specifications, stack switching specifications, or LEAFWORK specifications are ignored.
F842	Message	Unrecognized pragma SECTION 'character string'
	Cause	There is an unrecognized #pragma SECTION 'character string'.
F843	Message	Unspecified start address of 'section name'
	Cause	The correct starting address is not specified after AT in the #pragma section.
F845	Message	Cannot allocate 'section name' out of 'address range'
	Cause	The specified section cannot be placed at the specified starting address.
W846	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.
F847	Message	Different BANK or SP_SWITCH specified on same interrupt function 'function name'
	Cause	A different register bank specification or stack switching specification is specified for an interrupt function with the same name.
W849	Message	#pragma statement is not portable
	Cause	The #pragma statement does not conform to ANSI.
W850	Message	Asm statement is not portable
	Cause	The ASM statement does not conform to ANSI.
W851	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.

(9) Error message for a preprocessing directive <from 801> (4/4)

W852	Message	Module name truncate to ‘module name’
	Cause	The specified module name is too long.
	Compiler	It is shortened to the displayed ‘module name’.
F863	Message	Unrecognized pragma NAME ‘module name’
	Cause	Unrecognizable characters are in the ‘module name’.
W854	Message	Undefined rtos_task ‘character string’
	Cause	The body of RTOS task is not defined.
W855	Message	Cannot assign rtos_interrupt_handler to non-maskable and software interrupt
	Cause	The non-maskable interrupt and software interrupt cannot be specified in the RTOS_INTERRUPT handler.
W856	Message	Rechanged module name ‘module name’
	Cause	Duplicate module names are specified.
W857	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.
W861	Message	No EXIT_TABLE specifier
	Cause	Flash area branch table start address is not specified.
	Compiler	Specify the -zf option only when the self-rewriting function is used in flash memory products with a self-rewriting function.
F866	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.
F867	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.
F868	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.
W869	Message	‘section name’ section cannot change after C body
	Cause	Specified section cannot be changed after C body description.
W870	Message	Data aligned before ‘variable name’ in ‘section name’
	Cause	Data alignment is done before ‘variable name’ is allocated in ‘section name’.
W871	Message	Data aligned after ‘variable name’ in ‘section name’
	Cause	Data alignment is done after ‘variable name’ is allocated in ‘section name’.
F899	Message	Character string specified by #error is output
	Cause	An #error character string was specified.

(10) Error message for fatal file I/O and running on an illegal operating system <from 901> (1/2)

A901	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.
A902	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 **5.3 (18) Device file**.
A903	Message	Cannot open overlay file 'file name'
	Cause	The overlay file cannot be opened.
A904	Message	Cannot open temp
	Cause	The input temporary file cannot be opened.
A905	Message	Cannot create 'file name'
	Cause	A file create error was generated.
A906	Message	Cannot create temp
	Cause	A create error was generated in an output temporary file.
	Response	Check if the environmental variable TMP is specified.
A907	Message	No available data block
	Cause	A temporary file cannot be created because the drive file does not have sufficient capacity.
A908	Message	No available directory space
	Cause	A temporary file cannot be created because of insufficient directory area on the drive.
A909	Message	R/O : read / only disk
	Cause	A temporary file cannot be created because the drive is read only.
A910	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. 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.
A911	Message	Reading unwritten data, no available directory space
	Cause	An I/O error was generated for the following reasons. 1. EOF was passed and input proceeded. 2. The temporary file cannot be created because of insufficient directory area on the drive.
A912	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.
A913	Message	Requires MS-DOS V2.11 or greater
	Cause	The operating system is not MS-DOS (Ver. 2.11 or later).

(10) Error message for fatal file I/O and running on an illegal operating system <from 901> (2/2)

A914	Message	Insufficient memory in hostmachine
	Cause	The compiler cannot start because of insufficient memory.
	Response	Increase the free area in the conventional memory.
W915	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 W837 response.
F922	Message	Heap overflow : please retry compile without -QJ
	Cause	A memory overflow was generated in jump optimization. Recompile without specifying -QJ.
A923	Message	Illegal device file format
	Cause	A device file in an old format was referenced.

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>cc78K0 –c054 prime.c -a -p -x -e -ng
```

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



 sample\prime.c(18) : W745 Expected function prototype
 sample\prime.c(20) : W745 Expected function prototype
 sample\prime.c(26) : W622 No return value
 sample\prime.c(37) : W622 No return value
 sample\prime.c(44) : W622 No return value

 Target chip : uPD78054
 Device file : Vx.xx

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

A.3 Output List

(1) Assembler source module file

```
;78K/0 Series C Compiler Vx.xx Assembler Source
;                                          Date:xx xxx xxxx Time:xx:xx:xx
;Command   : -c054 prime.c -a -p -x -e -ng
;In-file   : prime.c
;Asm-file  : prime.asm
;Para-file :

$PROCESSOR(054)
$NODEBUG
$NODEBUGA
$KANJICODE SJIS
$TOL_INF        03FH, 0330H, 02H, 020H, 00H

       EXTRN     _@RTARG0
       EXTRN     @@isrem
       PUBLIC    _mark
       PUBLIC    _main
       PUBLIC    _printf
       PUBLIC    _putchar

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

@@DATA DSEG      UNITP
_mark: DS        (201)
       DS        (1)

;line      5
;line      8

@@CODE CSEG
_main:
       push     hl                                             ;[INF]1, 4
       push     ax                                             ;[INF]1, 4
       push     ax                                             ;[INF]1, 4
       push     ax                                             ;[INF]1, 4
       push     ax                                             ;[INF]1, 4
       movw     ax, sp                                         ;[INF]2, 8
       movw     hl, ax                                         ;[INF]1, 4
```

```
;line     11
       mov      a, #00H         ;0                                   ;[INF]2, 4
       mov      [hl], a         ;count                               ;[INF]1, 4
       mov      [hl+1], a       ;count                               ;[INF]2, 8
;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, 8
       xch      a, x                                                 ;[INF]1, 2
       mov      a,[hl+7]        ;i                                   ;[INF]2, 8
       cmpw     ax, #0C8H       ; 200                                ;[INF]3, 6
       orl      CY, a.7                                              ;[INF]2, 4
       bc       $$+4                                                 ;[INF]2, 6
       bnz      $L0004                                               ;[INF]2, 6
;line     14
       addw     ax, #_mark                                           ;[INF]3, 6
       movw     de, ax                                               ;[INF]1, 4
       mov      a, #01H;1                                            ;[INF]2, 4
       mov      [de], a                                              ;[INF]1, 4
       mov      a,[hl+6]        ;i                                   ;[INF]2, 8
       xch      a, x                                                 ;[INF]1, 2
       mov      a, [hl+7], a    ;i                                   ;[INF]2, 8
       incw     ax                                                   ;[INF]1, 4
       mov      [hl+7], a       ;i                                   ;[INF]2, 8
       xch      a, x                                                 ;[INF]1, 2
       mov      [hl+6], a       ;i                                   ;[INF]2, 8
       br       $L0003                                               ;[INF]2, 6
L0004:
;line     15
       mov      a, #00H ; 0                                          ;[INF]2, 4
       mov      [hl+6], a       ;i                                   ;[INF]2, 8
       mov      [hl+7], a       ;i                                   ;[INF]2, 8
L0006:
       mov      a,[hl+6]        ;i                                   ;[INF]2, 8
       xch      a,x                                                  ;[INF]1, 2
       mov      a,[hl+7]        ;i                                   ;[INF]2, 8
       cmpw     ax, #0C8H       ;200                                 ;[INF]3, 6
       orl      CY, a.7                                              ;[INF]2, 4
       bc       $$+7                                                 ;[INF]2, 6
       bz       $$+5                                                 ;[INF]2, 6
       br       !L0007                                               ;[INF]3, 6
;line     16
       addw     ax, #_mark                                           ;[INF]3, 6
       movw     de, ax                                               ;[INF]1, 4
       mov      a,[de]                                               ;[INF]1, 4
       cmp      a, #00H;0                                            ;[INF]2, 4
```

```
        bz        $L0015                                                 ;[INF]2, 6
;line      17
        mov       a,[hl+6]         ;i                                    ;[INF]2, 8
        rolc      a, 1                                                   ;[INF]1, 2
        xch       a, x                                                   ;[INF]1, 2
        mov       a,[hl+7]         ;i                                    ;[INF]2, 8
        rolc      a, 1                                                   ;[INF]1, 2
        addw      ax, #03H;3                                             ;[INF]3, 6
        mov       [hl+5],a         ;prime                                ;[INF]2, 8
        xch       a, x                                                   ;[INF]1, 2
        mov       [hl+4], a        ;prime                                ;[INF]2, 8
;line      18
        xch       a,x                                                    ;[INF]1, 2
        push      ax                                                     ;[INF]1, 4
        movw      ax, #L0011                                             ;[INF]3, 6
        call      !_printf                                               ;[INF]3, 7
        pop       ax                                                     ;[INF]1, 4
;line      19
        mov       a,[hl]           ;count                                ;[INF]1, 4
        xch       a, x                                                   ;[INF]1, 2
        mov       a,[hl+1]         ;count                                ;[INF]2, 8
        incw      ax                                                     ;[INF]1, 4
        mov       [hl+1], a        ;count                                ;[INF]2, 8
        xch       a, x                                                   ;[INF]1, 2
        mov       [hl], a          ;count                                ;[INF]1, 4
;line      20
        xch       a, x                                                   ;[INF]1, 2
        movw      _@RTARG0, ax                                           ;[INF]2, 6
        movw      ax, #08H ; 8                                           ;[INF]3, 6
        call      !@@isrem                                               ;[INF]3, 7
        or        a, x                                                   ;[INF]2, 4
        bnz       $L0012                                                 ;[INF]2, 6
        movw      ax, #0AH ; 10                                          ;[INF]3, 6
        call      !_putchar                                              ;[INF]3, 7
L0012:
;line      21
        mov       a,[hl+4]         ;prime                                ;[INF]2, 8
        add       a,[hl+6]         ;i                                    ;[INF]2, 8
        xch       a, x                                                   ;[INF]1, 2
        mov       a,[hl+5]         ;prime                                ;[INF]2, 8
        addc      a,[hl+7]         ;i                                    ;[INF]2, 8
        mov       [hl+3], a        ;k                                    ;[INF]2, 8
        xch       a, x                                                   ;[INF]1, 2
        mov       [hl+2], a        ;k                                    ;[INF]2, 8
L0014:
        mov       a ,[hl+2]        ;k                                    ;[INF]2, 8
        xch       a, x                                                   ;[INF]1, 2
```

```
       mov      a ,[hl+3]       ;k                                  ;[INF]2, 8
       cmpw     ax, #0C8H       ;200                                ;[INF]3, 6
       orl      CY, a.7                                             ;[INF]2, 4
       bc       $$+4                                                ;[INF]2, 6
       bnz      $L0015                                              ;[INF]2, 6
;line      22
       addw     ax, #_mark                                          ;[INF]3, 6
       movw     de, ax                                              ;[INF]1, 4
       mov      a, #00H;0                                           ;[INF]2, 4
       mov      [de], a                                             ;[INF]1, 4
       mov      a,[hl+4]        ;prime                              ;[INF]2, 8
       add      a,[hl+2]        ;k                                  ;[INF]2, 8
       xch      a, x                                                ;[INF]1, 2
       mov      a,[hl+5]        ;prime                              ;[INF]2, 8
       addc     a ,[hl+3]       ;k                                  ;[INF]2, 8
       mov      [hl+3], a       ;k                                  ;[INF]2, 8
       xch      a, x                                                ;[INF]1, 2
       mov      [hl+2], a       ;k                                  ;[INF]2, 8
       br       $L0014                                              ;[INF]2, 6
L0015:
;line      24
       mov      a,[hl+6]        ;i                                  ;[INF]2, 8
       xch      a, x                                                ;[INF]1, 2
       mov      a,[hl+7]        ;i                                  ;[INF]2, 8
       incw     ax                                                  ;[INF]1, 4
       mov      [hl+7],a        ;i                                  ;[INF]2, 8
       xch      a, x                                                ;[INF]1, 2
       mov      [hl+6], a       ;i                                  ;[INF]2, 8
       br       !L0006                                              ;[INF]3, 6
L0007:
;line      25
       mov      a,[hl]          ;count                              ;[INF]1, 4
       xch      a, x                                                ;[INF]1, 2
       mov      a,[hl+1]        ;count                              ;[INF]2, 8
       push     ax                                                  ;[INF]1, 4
       movw     ax, #L0017                                          ;[INF]3, 6
       call     !_printf                                            ;[INF]3, 7
       pop      ax                                                  ;[INF]1, 4
;line      26
       pop      ax                                                  ;[INF]1, 4
       pop      ax                                                  ;[INF]1, 4
       pop      ax                                                  ;[INF]1, 4
       pop      ax                                                  ;[INF]1, 4
       pop      hl                                                  ;[INF]1, 4
       ret                                                          ;[INF]1, 6
;line      31
_printf:
       push     hl                                                     ;[INF]1, 4
       push     ax                                                     ;[INF]1, 4
       push     ax                                                     ;[INF]1, 4
       push     ax                                                     ;[INF]1, 4
       movw     ax, sp                                                 ;[INF]2, 8
       movw     hl, ax                                                 ;[INF]1, 4
;line      35
       mov      a,[hl+10]       ;i                                     ;[INF]2, 8
       mov      [hl+2], a       ;j                                     ;[INF]2, 8
       xch      a,x                                                    ;[INF]1, 2
       mov      a,[hl+11]       ;i                                     ;[INF]2, 8
       mov      [hl+3], a       ;j                                     ;[INF]2, 8
;line      36
       mov      a,[hl+4]        ;s                                     ;[INF]2, 8
       xch      a, x                                                   ;[INF]1, 2
       mov      a,[hl+5]        ;s                                     ;[INF]2, 8
       mov      [hl+1], a       ;ss                                    ;[INF]2, 8
       xch      a, x                                                   ;[INF]1, 2
       mov      [hl], a         ;ss                                    ;[INF]1, 4
;line      37
       pop      ax                                                     ;[INF]1, 4
       pop      ax                                                     ;[INF]1, 4
       pop      ax                                                     ;[INF]1, 4
       pop      hl                                                     ;[INF]1, 4
       ret                                                             ;[INF]1, 6
;line      41
_putchar:
       push     hl                                                     ;[INF]1, 4
       push     ax                                                     ;[INF]1, 4
       push     ax                                                     ;[INF]1, 4
       movw     ax, sp                                                 ;[INF]2, 8
       movw     hl, ax                                                 ;[INF]1, 4
;line      43
       mov      a,[hl+2]        ;c                                     ;[INF]2, 8
       mov      [hl+1], a       ;d                                     ;[INF]2, 8
;line      44
       pop      ax                                                     ;[INF]1, 4
       pop      ax                                                     ;[INF]1, 4
       pop      hl                                                     ;[INF]1, 4
       ret                                                             ;[INF]1, 6
       END

;***Code Information***
;
;$FILE C:\NECTools32\sample\prime.c
;
; $FUNC main(8)
;      bc = (void)
;      CODE SIZE = 218 bytes, CLOCK_SIZE = 678 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 = 30 bytes, CLOCK_SIZE = 116 clocks, STACK_SIZE = 8 bytes
;
; $FUNC putchar(41)
;      bc = (char c:x)
;      CODE SIZE = 14 bytes, CLOCK_SIZE = 58 clocks, STACK_SIZE = 6 bytes

; Target chip:uPD78054
; Device file:Vx.xx
```

(2) Preprocess list file

```
/*
78K/0 Series C Compiler Vx.xx Preprocess List            Date:xx xxx xxxx Page:   1
Command    : -c054 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 : uPD78054
 Device file : Vx.xx
*/
```

(3) Cross-reference list file

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

Command   : -c054 prime.c -a -p -x -e -ng
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
REG1           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
REG1           pointer s           29      36
PARAM
REG1           int     i           30      35
PARAM
AUTO1          int     j           32      35
AUTO1          pointer ss          33      36
REG1           char    c           40      43
PARAM
AUTO1          char    d           42      43
               #define TRUE         1      14
               #define FALSE        2      22
               #define SIZE         3       5      13      15      21


 Target chip : uPD78054
 Device file : VX.XX
```

(4) Error list file

```
PRIME.C(   18) : W745 Expected function prototype
PRIME.C(   20) : W745 Expected function prototype
PRIME.C(   26) : W622 No return value
PRIME.C(   37) : W622 No return value
PRIME.C(   44) : W622 No return value


 Target chip : uPD78054
 Device file : VX.XX
Compilation complete,    0 error(s) and    5 warning(s) found.
```

APPENDIX B LIST OF USE-RELATED CAUTIONS

Number	Cautions
1	**[Cautions related to specification of options]** (a) When several specifications have been made for an option that does not allow multiple specifications, the last specification takes priority (is valid). (b) 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. (c) If the help option has been specified, all other options are ignored.
2	**[Cautions related to file output destinations]** Only disk-type files can be specified as the output destination for object module files.
3	**[Cautions related to error messages]** When a syntax error has been found in a file, an error message is attached to the file name. If a device file has been specified at a prohibited location, the specified character string is output by itself. In all other cases, the drive, path, and file extension must be attached.
4	**[Cautions related to source file names]** In the CC78K0, 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. (a) 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 plus, separate the primary name and extension with a dot (.), and use ".c", ".C" as the C source extension. (b) 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 path name. When PM plus is used, do not specify file names or path names that include a space or square brackets ([]), or path names that include 2-byte characters, such as Chinese characters. (c) Sharp symbol (#) cannot be used for file names and path names in parameter files. (d) An error is output during linking for files that have the same name as the first 8 characters of the primary name. (e) If using the ID78K0/ID78K0-NS or SM78K0, 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	**[Cautions related to include files]** It is not possible to define functions (excluding declarations) in an include file and then expand the file within the C source. When definitions are made within an include file, problems such as incorrect display of definition lines during source debugging may occur.

Number	Cautions
6	**[Cautions related to use of output assembler source]** When a C source program contains descriptions that use assembly language such as #asm blocks or _ _ asm statements, the load module file creation procedure sequence is compile, assemble, and then link. Be careful about the following points when using the assembler by outputting the assembler source to perform assembly without outputting direct objects, such as when descriptions using assembly language are used. (a) 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 of 8 or less characters beginning with the strings ?L@ is not used, the abort message A114 is output during assembly. (b) Describe the definitions of “callf functions” and “functions other than callf function” by combining these into two groups. If definitions are not described in a combination the warning message W717 is output. (c) 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. (d) 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 plus, 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. (e) When using PM plus, the RA78K0 is started regardless of compiler options -O/-NO when compiler option -A or -SA is specified. (f) 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, abort error A106 is output during assembly.
7	**[Cautions when specifying compiler option -QC2]** If the -QC2 option is specified in the CC78K0, the ranges of the types of constants and character constants that can be represented are handled as follows. `-128 to +127` char type `128 to 255` unsigned char type `0U to 255U` unsigned char type `From 256` int type `To -129` int type `'\0' to '\377'` char type When specifying the -QC2 option, the calculation results of a pair of char type constants and a pair of unsigned char type constants are handled as char types and unsigned char types, respectively. The calculation result of a char type constant and an unsigned char type constant is handled as unsigned char type. If the calculation result overflows, cast either of the constants to a type that can represent it or specify the -QC1 or -QC (default) option simultaneously. Casting prevents the data type from changing.

Number	Cautions
7	**Example)** When -QC2 option is specified `int i;` `i = 20*20 /*Negative value*/` `i = (int)20*20 /*400*/` **Remark** However, when specifying the -QU option, all char type data are handled as unsigned char type. Character constants in the range from '\200' to '\377' are handled as unsigned char type and have values from +128 to +255.
8	**[Usable assembler package]** Use the CC78K0 Ver. 3.50 together with assembler package Ver. 3.50 or later. Since long file names are supported, use of an RA78K0 earlier than Ver. 3.50 may result in errors.
9	**[Cautions when using network]** When the directory 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 CC78K0 in the network environment when using PM plus.
10	**[Creating link directive file]** When an area outside of the ROM or RAM area of the target device is used when linking the objects created by the 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. For information about creating link directive files, see **RA78K0 Assembler Package Operation User's Manual (U16629E)** and lk78k0.dr (in the SMP78K0 directory) equipped with the compiler. **Example)** When you want to place external variables without initial values (except sreg variables) from a certain C source file to the external memory. **1.** Change the section name for the external variables without initial value at the beginning of the C source. `#pragma section @@DATA EXTDATA` `⋮` **Caution Initialization of the changed segment and ROMization should be performed by changing the startup routine.** **2.** Create a link directive file. <lk78k0.dr> `memory EXTRAM : (0F000h, 00200h)` `merge EXTDATA := EXTRAM` Heed the following points when creating a link directive file.

Number	Cautions
10	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). **Example)** When added to the link directive file lk78k0.dr `memory EXTRAM : (0F000h, 00200h)` `memory STK : (0FB00H, 20H)` `merge EXTDATA := EXTRAM` (Command line) `> lk78k0 s01.rel prime.rel -bcl0.lib -SSTK -Dlk78k0.dr` 2. The following error may be output when linking in the defined memory area. `"*** ERROR F206 Segment 'xxx' can't allocate to memory-ignored."` **[Cause]** Because of insufficient space in the defined memory area, the indicated segment cannot be located. **[Response]** The response action is roughly divided into the following three steps. 1. Examine the size of a segment that cannot be located (refer to the .map file). 2. Based on the size of the segment examined in step 1, increase the size of the area where the segment is located in the directive file. 3. Specify the directive file specification option -D and link. However, based on the type of the segment marked by an error in step 1, the method used to examine the segment size differs in the following way. (1) When the segment is automatically generated during compilation Examine the size of the segment by the map file that is linked and created. (2) When the segment is created by the user Examine the size of the segment that is not located by the assemble list file (.prn).
11	**[Cautions when using va_start macro]** When -ZO is not specified, 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). • When the first argument is specified, use the va_starttop • When the second and subsequent arguments are specified, use the va_start macro.

Number	Cautions
12	**[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.
13	**[Startup routines and libraries]** (a) Use the provided startup routines and libraries with the same versions as the files in the executable form (cc78k0.exe or cc78k0). (b) For the floating point support functions sprintf, vprintf, and vsprintf, if the result value of a conversion that is specified with the conversion specifiers “%f”, “%e”, “%E”, “%g” or “%G” is smaller than the precision, the value is rounded down. “%f” conversion is executed even if the result value of conversion that is specified with “%g”/“%G” is greater than the precision. For functions sscanf and scanf, if no effective character is read during conversion that is specified with the conversion specifiers “%f”, “%e”, “%E”, “%g”, or “%G”, +0 is regarded as the conversion result. If the conversion result is “±”, ±0 is regarded as the conversion result. **[Prevention method]** None
14	**[-ZO option]** When the source is developed using CC78K0 Ver. 2.11 and earlier or when used with the assembler, changes must be made unless the -ZO option is specified. However, if the -ZO option is specified, the code efficiency drops and the functions in CC78K0 Ver. 3.00 and later versions are not available.
15	**[Cautions when source debugging with ID78K0-NS, ID78K0]** 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.
16	**[Cautions when source debugging with SM78K0]** 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.
17	**[When performing ROMization]** ROMization consists in placing initial values such as those of external variables that have an initial value in ROM, and then copying these values to RAM during system operation. In CC78K0 Ver. 3.50, a code is generated by default for ROMization. Therefore, it is necessary to perform linking with the startup routine including ROMization during linking. 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 8-5**. Startup routines: (1) When not using C standard library area: `S0.REL` (2) When using C standard library area: `S0L.REL`

Number	Cautions
17	**[Usage example]** `C:> LK78K0  s0.REL  SAMPLE.REL  -S  -BCL0.LIB  -OSAMPLE.LMF` SAMPLE.REL: Object module file of user program S0.REL: Startup routine CL0.LIB: Runtime library, standard library The -S option is a stack symbol (_@STBEG, _@STEND) automatic generation option. **Cautions** • **Be sure to link the startup routine at the beginning.** • **When creating a library, create it separately from the library provided by the CC78K0, and specify it prior to the compiler library during linking.** • **Do not add user functions to the CC78K0 library.** • **When using a floating point library (CL0*F.LIB), it is necessary to link the startup routine including the ROMization processing to both the standard library and the floating point library.** When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions supporting floating points **Example)** `-BMYLIB.LIB  -BCL0F.LIB  -BCL0.LIB` When using sprintf, sscanf, printf, scanf, vprintf, and vsprintf functions not supporting floating points **Example)** `-BMYLIB.LIB  -BCL0.LIB  -BCL0F.LIB`
18	**[Stack area symbol generation (-S)]** In CC78K0, the user cannot secure a stack area. To secure a stack area, specify the -S option during linking. When using PM plus, the -S option is automatically attached when the source file specification includes the C source.
19	**[ROM code]** When ordering ROM code, specify the -R or -U object converter options , such as `-r`, `-u0FFH`. -R: Sort HEX file contents by order of addresses. -U fill value: Fill empty space in ROM code with the specified fill value.
20	**[Help specification option]** In PM plus, compiler options --, -?, and -H, which display option descriptions, are ignored. For help, click the help button in the <Option Setup> dialog box of each tool.
21	**[-LL option specification]** When using PM plus, 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.
22	**[Cautions regarding symbol name length]** When using ID78K0-NS V2.01 and SM78K0 V.2.10 or earlier versions, do not use symbol names with more than 127 characters.

Number	Cautions

23	**[Cautions when using PM plus]**
	
	(a) Parameter file created by user
	When PM plus is specified for the parameter file created by the user, those contents are loaded to the parameter file created by PM plus. When creating a parameter file, be careful about the following points. Otherwise, an error will occur during build execution.
	• Specify a file with the same name as the parameter file created by PM plus.
	• Do not describe the device type specification option (-c), device file search path specification option (-y), and source file.
	• No validity check is performed for the options described in the parameter file created by the user.
	(b) <Assembler Options> dialog box
	Do not specify the -C, -F, and -Y options and the source file, or an error will occur during build execution.
	No validity check is performed for the options specified in the <Assembler Options> dialog box, so an error will occur during build execution in case of description errors.
	(c) Include file dependence relationship
	During checking of dependence relationships of include files during MAKE file creation with PM plus, condition statements such as #if are ignored. Therefore, include files not required for build are mistaken as required files. If described as comments or character strings, they are correctly judged as without dependence relationship.
	Example)
	`#if  0`
	`#include  "header1.h"  /* Dependence relationship judged to exist */`
	`#else / * ! zero */`
	`#include  "header2.h"`
	`#endif`
	`/*`
	`#include  "header3.h"`
	`*/`
	header1.h is judged as required for build during the dependence relationship check. If the header1.h file exists, header1.h gets registered to "ProjectWindow" of PM plus.
	[Prevention method] None. However, this has no effect on the build processing.
	(d) Project file
	When the [OK] or [Apply] button in the <Compiler Options> dialog box is not pressed, the [Using Startup Routine] and [Using Fixed Area of Standard Library] check boxes in the <<Startup Routine>> tab in the <Compiler Options> dialog box are selected, but they are not actually enabled.
	[Prevention method] Reflect the settings on PM plus with the following procedure.
	<1> Open the <Compiler Options> dialog box.
	<2> Select the <<Startup Routine>> tab.
	<3> Check the [Using Startup Routine] and [Using Fixed Area of Standard Library] settings.
	<4> Press the [OK] or [Apply] button (the correct settings are reflected when these buttons are pressed).
	(e) Project-related file settings
	Compiler attribute startup routines and standard libraries can be added and deleted from the [Project] menu of PM plus 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.
	(f) File names and path names enclosed in square brackets ([ ]) cannot be handled.

Number | Cautions

24 **[Cautions related to prototype declaration]**

If a function prototype declaration does not contain a function type specification, an error (F301, F701) results.

Example)

```
f ( void ) ;        /* F301 : Syntax error */
                    /* F701 : External definition syntax */
```

[Prevention method] Describe the function type.

Example)

```
int  f ( void ) ;
```

25 **[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.

Example)

```
    extren int i ; /* extern spelling error. No error results here. */
    /* comment */
    void f (void) ;
    [EOF]   /* Error such as F712 */
```

[Prevention method] None

26 **[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 (F803, F814, F821, etc.) results.

Example)

```
/* com1 */ #pragma  sfr                   /* F803 */
/* com2 */ #define ONE 1                  /* F803 */
#define /* com3 */ TWO 2                  /* F814 */
#ifdef /* com4 */ ANSI_C                  /* F814 */

/* com5 */ #endif
#define  SUB( p1,  /* com6 */ p2 )  p2 = p1       /* F821 */
```

[Prevention method] Describe the comment after the preprocessing directive.

Example)

```
#pragma  sfr              /* com1 */
#define ONE 1             /* com2 */
#define TWO 2             /* com3 */
#ifdef  ANSI_C            /* com4 */

#endif                            /* com5 */
#define SUB( p1,  p2 )  p2 = p1   /* com6 */
```

Number	Cautions
27	**[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 (1) below is fulfilled, and an error results if condition (2) below is fulfilled.

(1) If the tag is used in an argument declaration and a pointer to a structure or union is defined, warning W510 results when a function is called.

Example)

```
void func ( int , struct st ) ;

struct st {
    char memb1;
    char memb2;
} st [ ] = {
    { 1, 'a' } , { 2, 'b' }
} ;

void caller ( void ) {
    func ( sizeof ( st ) / sizeof ( st[0] ) , st );  /* W510 Pointer mismatch */
}
```

(2) 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 F737 results.

Example)

```
void func1( int , struct st ) ;    /* F737 Undeclared structure/union/enum tag */
struct st func2 ( int ) ;          /* F737 Undeclared structure/union/enum tag */
struct st {
     char memb1;
     char memb2;
} ;
```

[Prevention method] Define the tag of the structure, union, or enum beforehand.

Number	Cautions

28	**[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.
	Example)
	`void f ( void ) ;`
	`void f ( void ) {`
	`    char *p, *p1, *p2 ;`
	`    char *ca[3] = { p , p1 , p2 } ;  /* Error(F750) */`
	`}`
	[Prevention method] Describe an assignment statement and use it instead of initialization.
	Example)
	`void f ( void ) ;`
	`void f ( void ) {`
	`    char *ca[3] ;`
	`    char *p, *p1, *p2 ;`
	`    ca[0] = p ; ca[1] = p1 ; ca[2] = p2 ;`
	`}`

29	**[Cautions related to extern callt function]**
	
	If the address of an extern callt function is referenced by initializing the function table, etc., and the callt function is called by the same module, the assemble list is illegal and an error results during assembly.
	Example)
	`callt extern void funca ( void ) ;`
	`callt extern void funcb ( void ) ;`
	`callt extern void funcc ( void ) ;`
	
	`static void ( * const func [ ] ) ( ) = {`
	`    funca , funcb , funcc`
	`} ;`
	`callf void func2 ( void ) {`
	`    funcc ( ) ;`
	`    funcb ( ) ;`
	`    funca ( ) ;`
	`}`
	[Prevention method] Separate the function table and function call module.

Number	Cautions

30	**[Cautions related to functions returning a structure]**
	
	When a function returns a structure, an interrupt is generated in the process of returning a return value. If there is a call of the same function during interrupt servicing, the return value is illegal after the interrupt servicing ends.
	Example)
	`struct str {`
	`     char c ;`
	`     int i ;`
	`     long l ;`
	`} st ;`
	
	`struct str func ( ) {`
	`     /* Interrupt occurrence */`
	`      :`
	`}`
	
	`void main ( ) {`
	`     st = func ( ) ;        /* Interrupt occurrence */`
	`}`
	If the func function is called at the interrupt destination during the above servicing, st may be corrupted.
	[Prevention method] None

31	**[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 F750 results.
	Example)
	`struct Ss {`
	`     int  d1, d2 ;`
	`} ;`
	`union Au {`
	`     struct Ss t1;`
	`} u = { { 1, 2 } } ;        /* F750  Initializer syntax */`
	[Prevention method] Do not specify the initializer of a union with nesting.
	Example)
	`struct Ss {`
	`     int  d1, d2 ;`
	`} ;`
	`union Au {`
	`     struct Ss t1;`
	`} u = { 1, 2 } ;`

APPENDIX C LIST OF RESTRICTIONS RELATED TO CC78K0

This chapter describes in detail the restrictions on the CC78K0 and how to avoid them.

Number	Overview of Restrictions
1	The initialization of an external variable declared extern within a block does not become an error. In addition, the debugging information in the assembler source is incorrect.
2	Binding a variable with the same name to a variable declared extern in the block is sometimes illegal.
3	If a type defined by typedef (typedef name) is used in a function prototype declaration or a declaration using a const or volatile type modifier, the typedef expansion is illegal, and an error results.
4	Sometimes a multidimensional array with an undefined size does not operate properly.
5	In a function returning the address of a function with arguments, those arguments cannot be referenced. There is no error when referenced, but illegal code is output.
6	The signed type bit field is handled as an unsigned bit field.

C.1 Details About Restrictions and Prevention Methods

Restriction 1

The initialization of an external variable declared extern within a block does not become an error. In addition, the debugging information in the assembler source is incorrect.

[Description]

Since it is not compliant with the ANSI C language specifications, the initialization of an external variable declared extern within a block should produce an error, but the description does not become an error. The object defined as an external variable with initial value is interpreted and the code is output by the compiler.

The debugging information in the object output by the compiler is correct, but the debugging information in the assembler source is incorrect.

[Reproduced example]

```
int  i;
void  f(void) {
        extern int  i = 2;
}
```

[Prevention method] None

[Generation] All versions from Ver. 1.00 to Ver. 3.50

Restriction 2

Binding a variable with the same name to a variable declared extern in the block is sometimes illegal.

[Description]

Binding a variable with the same name to a variable declared extern in the block is illegal in either of the following cases.

(1) When a variable declared with extern in a block and a variable declared with static after outside the block have the same name

Since no error occurs and there is no binding, illegal code is output when this variable is referenced.

[Reproduced example]

```
void  f(void) {
        extern int  i;
        i = 1;                /* Illegal code output */
}
static int  i;
```

(2) When a variable declared with extern in a block and a variable not declared with static outside the block after a variable declared with extern have the same name
There is no binding, and illegal code is output.

[Reproduced example]

```
void  f(void) {
        extern int  i;
        i = 1;                /* Illegal code output */
}
int  i;
```

(3) When a variable declared with extern in a block and a variable not declared with extern outside the block before a variable declared with extern have the same name, and an automatic variable declared in a block containing the block with the variable declared with extern has the same name
The variable outside the block and the variable declared with extern in the block are not bound, and illegal code is output.

[Reproduced example]

```
int i = 1;
void  f(void) {
        int  i;
        {
               extern int  i;
               i = 1;         /* Illegal code output */
        }
}
```

(4) A variable declared with extern in a block and a variable declared with extern in another block have the same name
There is no binding, and illegal code is output.

[Reproduced example]

```
void  f1(void) {
         extern int  i;
         i = 2;
}
void  f2(void){
         extern int  i;
         i = 3;
}
```

[Prevention method] None

[Generation] All versions from Ver. 1.00 to Ver. 3.50

Restriction 3

If a type defined by typedef (typedef name) is used in a function prototype declaration or a declaration using a const or volatile type modifier, the typedef expansion is illegal, and an error results.

[Description]

If a type defined by typedef (typedef name) is used in a function prototype declaration or a declaration using a const or volatile type modifier, the typedef expansion is illegal, and an error may result.

[Reproduced example 1]

```
typedef int  FTYPE();

FTYPE  func;
int  func(void);                   /* F713 Redefined 'func' */
```

[Reproduced example 2]

```
typedef int  VTYPE[2];
typedef int  *VPTYPE[3];

const VTYPE  *a;
const int  (*a)[2];                /* F713 Redefined 'a' */
volatile VPTYPE  b[2];
volatile int *volatile  b[2][3];   /* F713 Redefined 'b' */
```

[Prevention method] None

[Generation] All versions from Ver. 1.00 to Ver. 3.50

Restriction 4

Sometimes a multidimensional array with an undefined size does not operate properly.

[Description]

Sometimes a multidimensional array with an undefined size does not operate properly.

[Reproduced example 1]

```
char  c[][3]={{1},2,3,4,5};         /* Illegal code */
```

[Reproduced example 2]

```
char  c[][2][3]={"ab","cd","ef"};   /* Error (F756) */
```

[Prevention method]

Define the size of the multidimensional array.

[Generation] All versions from Ver. 1.00 to Ver. 3.50

Restriction 5

In a function returning the address of a function with arguments, those arguments cannot be referenced. There is no error when referenced, but illegal code is output.

[Description]

In a function returning the address of a function with arguments, those arguments cannot be referenced. There is no error when referenced, but an illegal code is output.

[Reproduced example]

```
char *c;
int  *i;
void (*f1(int *))(char *);
void (*f2(void))(char *);
void (*f3(int *))(void);

void main() {
        (*f1(i))(c);          /* Correct description (W510) */
        (*f1(i))(i);          /* Incorrect description */
        (*f2())(c);           /* Correct description (W509) */
        (*f2())();            /* Incorrect description (W509) */
        (*f3(i))();           /* Correct description (W509) */
        (*f3(i))(i);          /* Incorrect description */
}
```

W509 or W510 is output for a correct description. Nothing is output for a description that should produce a warning. However, the output code is normal.

```
void (*f4())(int p) {
       p++;                   /* Incorrect description */
}
```

An error is not output for a description that should cause an error. An illegal code is generated.

[Prevention method] None

[Generation] All versions from Ver. 1.00 to Ver. 3.50

Restriction 6

The signed type bit field is handled as an unsigned bit field.

[Description]

The signed type bit field is handled as an unsigned bit field.

[Prevention method] None

[Generation] All versions from Ver. 1.00 to Ver. 3.50

APPENDIX D INDEX