CUSTOMER NOTIFICATION

SUD-DT-04-0137

March 24, 2004

Yoshiro Harada, Senior System Integrator
Microcomputer Group
2nd Solutions Division
Solutions Operations Unit
NEC Electronics Corporation

CP(K), O

78K0 Series C Compiler
CC78K0 Ver. 3.60
Operating Precautions

Windows Based
HP9000 Series 700 (HP-UX) Based
SPARCstation Family (SunOS/Solaris) Based

Be sure to read this document before using the product.

CONTENTS

Thank you for purchasing 78K0 Series C compiler CC78K0 Ver. 3.60.
This document explains the modifications from CC78K0 Ver. 3.50 to Ver. 3.60, cautions, usage restrictions, and correction of documents.
Be sure to read this document before using CC78K0 Ver. 3.60.

1. USER'S MANUAL

The following user's manuals are available for this version as a PDF file only.

- C compiler CC78K0 Language (document number: U17016E)
- C compiler CC78K0 Operation (document number: U17017E)

2. SUPPORTED TOOLS

Use the following tool versions when used in combination with CC78K0 Ver. 3.60.

• PM plus:	PM plus Ver. 5.20 or later
• Assembler package:	RA78K0 Ver. 3.70 or later
• Integrated debugger:	ID78K0-NS Ver. 2.52 or later
	ID78K0-QB Ver. 2.81 or later
• System simulator:	SM78K0 Ver. 2.52 or later

3. ITEMS REVISED FROM Ver. 3.50 TO Ver. 3.60

This section explains the changes from Ver. 3.50 to Ver. 3.60 concerning usage restrictions and the C compiler.

3.1 Changes in Usage Restrictions

This section explains changes in usage restrictions from Ver. 3.60 are as follows. The item number corresponds to the number that has been reported in the usage restriction documents.

3.1.1 Removal of usage restrictions

The following three restrictions have been removed (reported in SBG-DT-04-0110).

No.17 Information of the startup routine and standard library of the old version of the CC78K0 may not be passed on to the new file.

[Description]

When a project file created in the old version of the CC78K0 is loaded, information of the startup routine and standard library may not be passed on to the new file.

No.18 An illegal code is output as a result of a simple assignment operation in which the left-side and right-side operands are cast to the pointer for a structure, union, or array.

[Description]

An illegal code is output when the following four conditions are satisfied.

(1) Simple assignment operation

(2) The left-side and right-side operands are both indirection reference data that uses a structure, union, array, or pointer

(3) The left-side and right-side operands both reference the address of a symbol.

(4) 4-byte data (long, float, double, or long double type) is the target of the operation.

No.19 An illegal code may be output as a result of the **memcpy** function when #pragma inline is specified.

[Description]

An illegal code may be output when the following three conditions are satisfied.

(1) #pragma inline is specified.

(2) A **memcpy** function is used.

(3) The third argument of the **memcpy** function in (2) is not a constant (when the third argument uses the HL register)

3.2 Changes in C Compiler

This section explains the changes in CC78K0 Ver. 3.50 to Ver. 3.60.

3.2.1 Support of space and 2-byte character use for folder name

The space and 2-byte characters can now be used for the folder name. However, they still cannot be used for the file path specification. (See **4.2.5 Caution on installation directory**.)

3.2.2 Change of messages

The output format and number of error and warning messages have been changed. As a result, the error message highlighted display function and help jump function of PM plus are now supported.

• Change of symbols

The symbols have been modified as shown below.

Old Symbol	New Symbol	Meaning
A	F	Fatal error
F	E	Error
F	C	Internal error
W		Warning

• Addition of digit to message number

The number of message digits has been increased from three to four (xxx → 0xxx). There is no change in second to fourth digits.

- Change of format

  The format of the message has been changed and unified as shown below.

  ```
  [file-number(line-number) :] tool-name  key-word  symbol-and-number:  message
  ```

  Output example:

  ```
  test.c(26) : CC78K0 warning W0622: No return value
  ```

3.2.3 Expansion of functions in option dialog box

The edit functions used when specifying the following options have been added to the option specification dialog box in PM plus.

- Include file search specification
- Define Macro specification
- Undefine Macro specification

4. CAUTIONS

Cautions on using CC78K0 Ver. 3.60 are described below.

4.1 Device File

A device file is necessary to execute the CC78K0. The device file is not included in the CC78K0. Download the device file via ODS (online delivery service).
NEC Electronics Microprocessor website (URL: http://www.necel.com/micro/index_e.html)
→ [Development Tools Download] → [DeviceFile]

4.2 Cautions on Installation (Windows Version)

4.2.1 Product ID

A product ID is required to install CC78K0 Ver. 3.60. The product ID is shown on the medium or medium case.

4.2.2 Caution on activating installer

Insert the CD-ROM of CC78K0 Ver. 3.60 in the CD-ROM drive. The installer is started automatically. If it does not start automatically, execute “setup.exe” in the directory DISK1 under CC78K0 from Windows Explorer.

4.2.3 Caution on restarting computer

Because it may be necessary to restart the computer after installation, terminate all other applications.

4.2.4 Caution on authority for installation

Administrator right is required for installation of CC78K0 Ver. 3.60 in Windows NT, Windows 2000, or Windows XP.

4.2.5 Caution on installation directory

Do not install CC78K0 Ver. 3.60 in a directory with a name containing a space or a multi-byte character; otherwise the development tools may not be correctly executed.

4.2.6 Caution on re-installation

To re-install CC78K0 Ver. 3.60, uninstall the copy of CC78K0 Ver. 3.60 already installed. If this product is installed in a different directory without uninstalling the first copy of CC78K0 Ver. 3.60, the first copy of CC78K0 Ver. 3.60 already installed cannot be uninstalled.

4.2.7 Caution on file created at installation

The following file will be created after CC78K0 Ver. 3.60 has been installed. This file is necessary for uninstalling CC78K0 Ver. 3.60 and must not be deleted (the installation destination is assumed to be C:\NECTools32).

```
C: \NECTools32\SETUP\*.*
```

4.2.8 Caution on language used in Windows

CC78K0 Ver. 3.60 cannot be installed in the Japanese Windows environment.
If an attempt is made to install CC78K0 Ver. 3.60 in the Japanese Windows environment, a message indicating a file transfer error is displayed and the installation is aborted.
This situation also occurs if Japanese is specified as the system language in the “Regional Settings Properties” tab.

4.3 Cautions Related to CC78K0

See **APPENDIX B** in **CC78K0 Ver.3.60 or Later Operation User’s Manual** for cautions related to the CC78K0.

4.4 Cautions Related to Library Source

CC78K0-L Ver. 3.50 is sold separately as a library source for CC78K0 Ver. 3.60. Note the version is not Ver. 3.60.

5. RESTRICTIONS

This section explains restrictions only apply to CC78K0 Ver. 3.60.
Since the numbers assigned to the restrictions have not been changed, the restriction numbers shown below are not consecutive.

5.1 List of Restrictions

No.	Restrictions
11	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.
12	Binding a variable with the same name to a variable declared extern in the block is sometimes illegal.
13	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.
14	Sometimes a multidimensional array with an undefined size does not operate properly.
15	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.
16	The signed type bit field is handled as an unsigned bit field.

5.2 Details of Restrictions

No.11 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.

Example:

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

[Workaround]

There is no workaround.

No.12 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.

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.

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.

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.

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

```
  }
```

[Workaround]

There is no workaround.

No.13 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.

Example 1:

```
typedef int  FTYPE();

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

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' */
```

[Workaround]

There is no workaround.

No.14 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.

Example 1:

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

Example 2:

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

[Workaround]

Define the size of the multidimensional array.

No.15 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.

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.

[Workaround]

There is no workaround.

No.16 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.

[Workaround]

There is no workaround.

6. CORRECTION OF DOCUMENTS

6.1 Correction of Language User's Manual

The affected user's manual is as follows.

User's Manual	Version	Document Number
CC78K0 C Compiler Operation	1st	U17017E

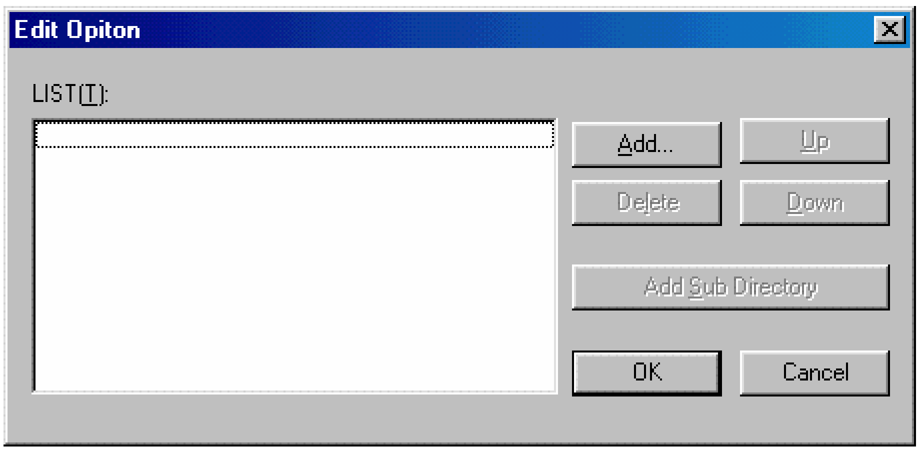
6.1.1 Path edit dialog box

Descriptions for when [Edit…] button of the following options is clicked have been added.

- Include Search Path specification in the Compiler Options dialog box on page 40.

<Addition>

- When the [Edit…] button is clicked, the <Edit Option> dialog box appears. The specified paths can be edited in the list in this dialog box.
- If multiple paths are specified, paths are listed from the path searched first.
- When the [Add…] button is clicked, the <Browse for Folder> dialog box appears and the selected path is added to the list. If there was already a path in the list, the new path is added at the top of the list.
- When the [Delete] button is clicked, the selected path is deleted from the list.
- When the [Up] or [Down] button is clicked, the selected path is moved up or down in the list.
- When the [Add Sub Directory] button is clicked, the sub-directory under the selected directory is added below the selected position.
- When the [OK] button is clicked, the list editing contents are reflected to the combo box.
- When the [Cancel] button is clicked, the list editing contents are not reflected to the combo box.



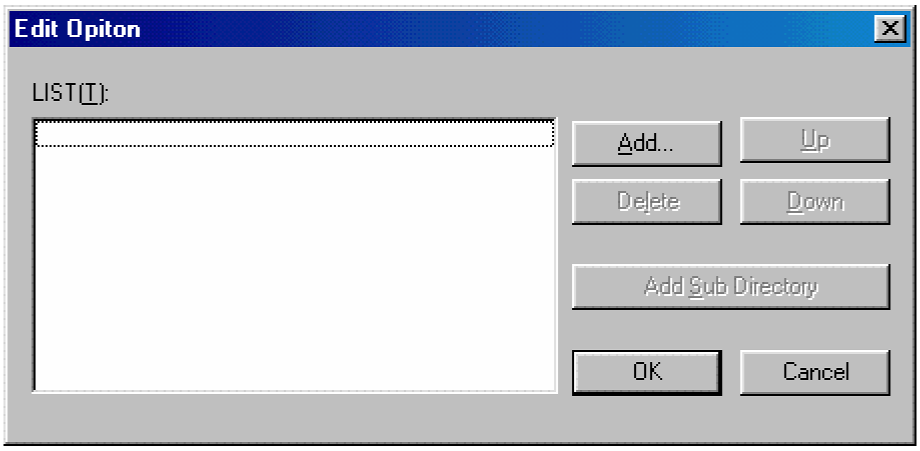
6.1.2 Symbol edit dialog box

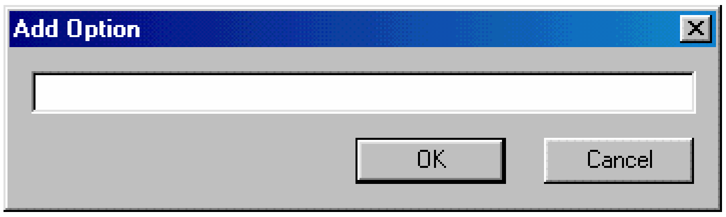
Descriptions for when [Edit…] button of the following options is clicked have been added.

- Define Macro specification in the Compiler Options dialog box on page 40
- Undefine Macro specification in the Compiler Options dialog box on page 04

<Addition>

- When the [Edit…] button is clicked, the <Edit Option> dialog box appears. The specified symbols can be edited in the list in this dialog box.
- When the [Add…] button is clicked, the <Add Option> dialog box appears and symbols to be added can be edited. If there was already a defined symbol in the list, the new symbol is added at the top of the list.
- When the [Delete] button is clicked, the selected symbol is deleted from the list.
- When the [Up] or [Down] button is clicked, the selected symbol is moved up or down in the list.
- When the [OK] button is clicked, the list editing contents are reflected to the combo box.
- When the [Cancel] button is clicked, the list editing contents are not reflected to the combo box.





6.1.3 Changes in cautions

The message numbers described in (a), (b), and (f) in item 6 [Cautions related to use of output assembler source] on page 209 have been modified. **<<Japanese version only>>**

<Incorrect>

(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 Fatal error (F0114) 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 (W0717) is output.

(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, error (C0106) is output during assembly.

<Correct>

(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 Fatal error (F2114) 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 (W2717) is output.

(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, error (F2106) is output during assembly.

The description in item 19 [ROM code] on page 213 has been modified.

<Incorrect>

When ordering ROM code, specify the -R or -U object converter options, such as –r, -u0FFH (do not cancel the specification).

-R: Sort HEX file contents by order of addresses.

-U fill value: Fill empty space in ROM code with the specified fill value.

<Correct>

When ordering ROM code, specify the -R or -U object converter options, such as –r, -u0FFH (this option is specified by default; do not cancel the specification).

-R: Sort HEX file contents by order of addresses.

-U fill value: Fill empty space in ROM code with the specified fill value.