CUSTOMER NOTIFICATION

SUD-DT-04-0118

March 8, 2004

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

CP(K), O

78K4 Series C Compiler
CC78K4 Ver. 2.40
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

1. USER'S MANUAL 3
2. SUPPORTED TOOLS 3
3. ITEMS REVISED FROM Ver. 2.30 TO Ver. 2.40 3
3.1 Change of Specifications 3
3.1.1 Supported OS 3
3.1.2 Support of PM plus 3
3.1.3 Enhancement of warning message output function 3
3.2 List of Handling of Restrictions 4
3.2.1 List of corrected bugs 4
3.2.2 Details of corrected bugs 5
4. CAUTIONS 12
4.1 Device File 12
4.2 Cautions on Installation (Windows Version) 12
4.2.1 Product ID 12
4.2.2 Caution on activating installer 12
4.2.3 Caution on restarting computer 12
4.2.4 Caution on authority for installation 12
4.2.5 Caution on installation directory 12
4.2.6 Caution on re-installation 12
4.2.7 Caution on file created at installation 12
4.2.8 Caution on project manager 13
4.2.9 Caution on language used in Windows 13
4.3 Cautions Related to CC78K4 13
5. RESTRICTIONS 13
5.1 Restrictions Related to CC78K4 13
6. CORRECTION OF DOCUMENTS 14
6.1 Correction of Language User's Manual 14

Thank you for purchasing 78K4 Series C compiler CC78K4 Ver. 2.40.
This document explains the modifications from CC78K4 Ver. 2.30 to Ver. 2.40, cautions, usage restrictions, and correction of documents.
Be sure to read this document before using CC78K4 Ver. 2.40.

1. USER'S MANUAL

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

- C compiler CC78K4 Ver. 2.30 or Later - Language (document number: U15556E)
- C compiler CC78K4 Ver. 2.40 or Later - Operation (document number: U16707E)

2. SUPPORTED TOOLS

Use the following tool versions when used in combination with CC78K4 Ver. 2.40.

- PM plus: PM plus Ver. 5.10 or later
- Assembler package: RA78K4 Ver. 1.60 or later
- Integrated debugger: ID78K4-NS V2.52 or later
- System simulator: SM78K4 V2.52 or later

3. ITEMS REVISED FROM Ver. 2.30 TO Ver. 2.40

This section explains the changes from Ver. 2.30 to Ver. 2.40.

3.1 Change of Specifications

The specifications changed in Ver. 2.40 are as follows.

3.1.1 Supported OS

Windows XP is now supported. Windows 95 is no longer supported in Ver. 2.40 and later.

3.1.2 Support of PM plus

PM plus is now supported. PM plus is included in assembler package RA78K4.

3.1.3 Enhancement of warning message output function

A warning message is now output for the following items.

- Unreferenced label
  This enables detection of user-defined unreferenced labels. In addition, this enables detection of spelling errors in the default label in a switch statement.
- Operator that has no operation
  This enables detection of description errors in an assignment statement.

3.2 List of Handling of Restrictions

The following 15 bugs have been corrected in Ver. 2.40.

3.2.1 List of corrected bugs

No.	Bugs
1	If a character string includes a NULL character, the character string following the NULL character is invalid.
2	An illegal code is output under certain conditions such as when using sreg when using a small model or medium model.
3	An illegal code may be output if the address of a function located in the flash memory area is referenced.
4	The constant expression of #if may not be processed correctly.
5	An illegal code may be output if the address of a multidimensional array is referenced and the same array is referenced after that.
6	An illegal code may be output if a variable used by a compound assignment operation of a bit-wise AND/XOR/OR is referenced by the following **switch** statement
7	An illegal code is output if a negative floating-point constant is cast to an unsigned integer type.
8	An illegal code is output if a logical OR or logical AND operation is performed between floating-point constants.
9	An illegal code is output if the function parameter is an array that is declared as a register
10	If variables with the same name are declared in multiple files while using the #pragma section, the variables may not be allocated to the correct section.
11	An illegal code is output if a logical OR or logical AND operation is performed between a floating-point constant and integer-type constant.
12	An illegal code may be output if a function that has a 5-byte or larger structure or union as an argument is called when using a large model
13	An illegal code may be output if a variable is used as the index of a structure array that has a bit field as a member and an assignment operation is performed on the bit field located at the higher byte of an array element.
14	When a conversion specifier of a decimal integer is used in the **scanf** or **sscanf** function and the value after converting a character string is 65536 or larger or –65536 or lower, an illegal result may be returned.
15	An illegal conversion result of a floating-point number may be output when the function **printf**, **sprintf**, **vprintf**, or **vsprintf** is executed in a medium model.

3.2.2 Details of corrected bugs

Bugs corrected in Ver. 2.40 are detailed below. Though items below include [Workaround], it is not necessary to implement the workaround because bugs have already been corrected.

No.1 If a character string includes a NULL character, the character string following the NULL character is invalid.

Example:

```
const char str[] = "test\0TEST";
```

No area is secured for the character string following the NULL character.

[Workaround]

Describe `const char str[] = {'t','e','s','t','\0','T','E','S','T'};`.

No.2 An illegal code is output if any of the following conditions (1) to (3) is satisfied when using a small model or medium model.

(1) An add-sub assignment operation is performed between an offset sreg/__sreg variable and a character string

(2) A conditional expression between **saddr** area data and an array, or an SFR symbol and an array is performed and the offset value of the array is 256 or larger

(3) When the argument _@NRARGx of a **norec** function is passed to an array whose offset value is 256 or larger

Example:

```
#pragma sfr
__leaf void nrfunc(int, int, int, int *);
__sreg struct {
    int a;
    int b;
} st;
int a[200], i;

void func()
{
    st.b -= (int)"abc";            /* (1) */
    if (CR00 == (int)&a[199]) i++; /* (2) */
    nrfunc(1, 2, 3, &a[199]);      /* (3) */
}
```

[Workaround]

Implement the following workarounds for conditions (1) to (3), respectively.

(1) Divide the description into an add-sub operation and an assignment operation.

```
st.b = st.b - (int)"abc";
```

(2) Insert a temporary variable, and execute the operation by assigning the operation result to the temporary variable.

```
int tmp1;
tmp1 = (int)&a[199];
if (CR00 == tmp1) i++;
```

(3) Insert a temporary variable, and execute the operation by assigning the operation result to the temporary variable.

```
int *tmp2;
tmp2 = &a[199];
nrfunc(1, 2, 3, tmp2);
```

No.3 An illegal code may be output if the address of a function located in the flash memory area is referenced.

Example:

```
#pragma ext_table 0x2000
#pragma ext_func func1 1
#pragma ext_func func2 2

int func1(int, int);
int func2(int, int);
int (*func_b1)(int, int) = &func1;
__sreg int (*func_b2)(int, int) = &func2;

void func()
{
    func_b1 = func1;  /* normal code */
    func_b2 = func2;  /* illegal code */
}
```

[Workaround]

There is no workaround.

No.4 The constant expression of #if may not be processed correctly.

Example:

```
#define a
#if a
int i;
#endif
void func()
{
    i++;
}
```

[Workaround]

There is no workaround.

No.5 An illegal code may be output if the address of a multidimensional array is referenced and the same array is referenced after that.

Example:

```
unsigned char a[10][1];
unsigned char *p,c,uc1,uc2;
void func()
{
    p = a[uc1];
    c = a[uc1][uc2]; /* illegal code */
}
```

[Workaround]

There is no workaround.

No.6 An illegal code may be output if a variable used by a compound assignment operation of a bit-wise AND/XOR/OR is referenced by the following **switch** statement.

Example:

```
unsigned int ui, i;
void func(unsigned int p1, unsigned int p2)
{
    ui |= p2;
    switch (p2) {    /* illegal code */
    case 0:
    case 1:
    case 2:
        i++;
    }
}
```

[Workaround]

Divide the function into a bit-wise AND/XOR/OR operation and a simple assignment operation.

```
ui = ui | p2;
```

No.7 An illegal code is output if a negative floating-point constant is cast to an unsigned integer type.

Example:

```
unsigned long ans;
float f1 = -3.8f;
void func()
{
    int a;
    ans = f1;
    a = ((unsigned long)(-3.8f) != ans);
}
```

[Workaround]

Cast a constant to a signed integer type before casting the constant to the unsigned integer type.

```
a = ((unsigned long)(long)(-3.8f) != ans);
```

No.8 An illegal code is output if a logical OR or logical AND operation is performed between floating-point constants.

Example:

```
void func()
{
    int r1, r2;
    float f1 = 17, f2 = 16;
    r1 = f1 || f2;
    r2 = f1 && f2;
}
```

[Workaround]

There is no workaround.

No.9 An illegal code is output if the function parameter is an array that is declared as a register.

Example:

```
void func(register char a[])
{
    register char *p;
    p = (char *)a;
}
```

[Workaround]

Describe a pointer type, instead of describing the array.

```
void func(register char *a)
{
    register char *p;
    p = a;
}
```

No.10 If variables with the same name are declared in multiple files while using the #pragma section, the variable may not be allocated to the correct section.

Example:

```
--- test.c ---
#include "a.h"
#include "b.h"
#include "c.h"

--- a.h ---
#pragma section @@DATA DAT1
int i;
#pragma section @@DATA DAT2

--- b.h ---
#pragma section @@DATA DAT3
int j;
#pragma section @@DATA DAT4
```

```
--- c.h ---
#pragma section @@DATA DAT5
extern int i;        /* same when int i;  */
#pragma section @@DATA DAT6
```

[Workaround]

Do not use variables with the same name in multiple files when using the #pragma section.

No.11 An illegal code is output if a logical OR or logical AND operation is performed between a floating-point constant and integer-type constant.

Example:

```
void func()
{
     int rval;
     int cZero = 0;
     rval = 0.0F || cZero;  /* illegal code in Windows version */
     rval = cZero || 0.0F;  /* illegal code in UNIX version */
}
```

[Workaround]

Do not describe a floating-point constant for the logical OR or logical AND operation.

No.12 An illegal code may be output if a function that has a 5-byte or larger structure or union as an argument is called when using a large model.

Example:

```
struct {
    int a;
    int b;
    int c;
} st;
void (*fp)();

void func()
{
    (*fp)(st);
}
```

[Workaround]

There is no workaround.

No.13 An illegal code may be output if a variable is used as the index of a structure array that has a bit field as a member and an assignment operation is performed on the bit field located at the higher byte of an array element.

Example:

```
struct st {
    unsigned int bit0:1;
    unsigned int bit1:1;
    unsigned int bit2:1;
    unsigned int bit3:1;
    unsigned int bit4:1;
    unsigned int bit5:1;
    unsigned int bit6:1;
    unsigned int bit7:1;
    unsigned int bit8:1;
    unsigned int bit9:1;
    unsigned int bitA:1;
    unsigned int bitB:1;
    unsigned int bitC:1;
    unsigned int bitD:1;
    unsigned int bitE:1;
    unsigned int bitF:1;
} str[5];
unsigned char num;

void func()
{
    str[num].bit8 = 1;      /* illegal code */
}
```

[Workaround]

Insert a temporary variable, and execute the operation by assigning the operation result to the temporary variable.

```
struct st *tmp;
tmp = &str[num];
tmp->bit8 = 1;
```

No.14 When a conversion specifier of a decimal integer is used in the **scanf** or **sscanf** function and the value after converting a character string is 65536 or larger or –65536 or lower, an illegal result may be returned.

Internally, a character string is replaced by numbers sequentially from the lower digit.

(1) One character is acquired and converted into a numeric value.

(2) If all the input characters are converted, the program jumps to (7).

(3) The radix is multiplied (carry operation).

(4) The next character is acquired and converted into a numeric value.

(5) The converted value is added to the multiplied result (processing relevant to this bug)

(6) The program jumps to (2).

(7) End

An illegal value is returned if the result of addition of the lower 2 bytes in step (5) exceeds 65535.

Example:

```
long a;
char buf[16];

void func()
{
    strcpy(buf, "65536");
    sscanf(buf, "%ld", &a);
}
```

[Workaround]

Implement any of the following workarounds.

(1) Describe both the character string and conversion specifer using the octal or hexadecimal expression.

Example:

```
strcpy(buf, "0x10000");
sscanf(buf, "%lx", &a);
```

(2) Use the **atol** function to acquire the value from the character string.

Example:

```
#include<stdlib.h>
a = atol(buf);
```

No.15 An illegal conversion result of a floating-point number may be output when the function **printf**, **sprintf**, **vprintf**, or **vsprintf** is executed in a medium model.

Example:

```
#include <stdio.h>
void func()
{
    char res[20];
    sprintf(res, "%f", 1.2e-38);
}
```

[Workaround]

There is no workaround.

4. CAUTIONS

Cautions on using CC78K4 Ver. 2.40 are described below.

4.1 Device File

A device file is necessary to execute the CC78K4. The device file is not included in the CC78K4. 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 CC78K4 Ver. 2.40. The product ID is shown on the medium or medium case.

4.2.2 Caution on activating installer

Insert the CD-ROM of CC78K4 Ver. 2.40 in the CD-ROM drive. The installer is started automatically. If it does not start automatically, execute “setup.exe” in the directory DISK1 under CC78K4 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 CC78K4 Ver. 2.40 in Windows NT, Windows 2000, or Windows XP.

4.2.5 Caution on installation directory

Do not install CC78K4 Ver. 2.40 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 CC78K4 Ver. 2.40, uninstall the copy of CC78K4 Ver. 2.40 already installed. If this product is installed in a different directory without uninstalling the first copy of CC78K4 Ver. 2.40, the first copy of CC78K4 Ver. 2.40 already installed cannot be uninstalled.

4.2.7 Caution on file created at installation

The following file will be created after CC78K4 Ver. 2.40 has been installed. This file is necessary for uninstalling CC78K4 Ver. 2.40 and must not be deleted (the installation destination is assumed to be C:\NECTools32).

C: \NECTools32\SETUP*.*

4.2.8 Caution on project manager

The project manager is not supported in CC78K4 Ver. 2.40 or later. Use PM plus. PM plus is included in RA78K4 Ver. 1.60 or later, so upgrade to RA78K4 to Ver. 1.60 or later.

4.2.9 Caution on language used in Windows

CC78K4 Ver. 2.40 cannot be installed in the Japanese Windows environment.
If an attempt is made to install CC78K4 Ver. 2.40 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 CC78K4

See **APPENDIX B** in **CC78K4 Ver.2.40 or Later Operation User’s Manual** for cautions related to CC78K4.

5. RESTRICTIONS

Restrictions on using CC78K4 Ver. 2.40 are described below.

5.1 Restrictions Related to CC78K4

See **APPENDIX C** in **CC78K4 Ver.2.40 or Later Operation User’s Manual** for cautions related to CC78K4.

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
CC78K4 C Compiler Ver.2.40 or Later Language	1st	U15556EJ1

6.1.1 Correction of "9.2 Source File Inclusion"

The descriptions in **(1) #include < >** on page 145 have been corrected as follows.

<Incorrect>

FUNCTION

If the directive form is **#include** < >, the C compiler searches the directory specified by the **-i** compiler option, directory specified by the INC78K environment variable,

<Correct>

FUNCTION

If the directive form is **#include** < >, the C compiler searches the directory specified by the **-i** compiler option, directory specified by the INC78K4 environment variable,

<Incorrect>

EXPLANATION

In the above example, the C compiler searches the directory specified by the INC78K environment variable and…

<Correct>

EXPLANATION

In the above example, the C compiler searches the directory specified by the INC78K4 environment variable and…

The descriptions in **(2) #include " "** on page 146 have been corrected as follows.

<Incorrect>

FUNCTION

If it is not found, the directory specified by the **-i** compiler option, directory specified by the INC78K environment variable, …

<Correct>

FUNCTION

If it is not found, the directory specified by the **-i** compiler option, directory specified by the INC78K4 environment variable, …

<Incorrect>

EXPLANATION

In the above example, the C compiler searches the current working directory, the directory specified by the INC78K environment variable, and…

<Correct>

EXPLANATION

In the above example, the C compiler searches the current working directory, the directory specified by the INC78K4 environment variable and…

The descriptions in **(3) #include preprocessing token string** on page 147 have been corrected as follows.

<Incorrect>

EXPLANATION

… the C compiler searches the directory specified by the **-i** compiler option, directory specified by the INC78K environment variable, and directory \NECTools32\INC78K4 registered in the registry for the specified file. If the token string is replaced by "file name", the current working directory is searched. If the specified file is not found, the directory specified by the **-i** compiler option, directory specified by the INC78K environment variable, and directory \NECTools32\INC78K4 registered in the registry is searched.

<Correct>

EXPLANATION

… the C compiler searches the directory specified by the **-i** compiler option, directory specified by the INC78K4 environment variable, and directory \NECTools32\INC78K4 registered in the registry for the specified file. If the token string is replaced by "file name", the current working directory is searched. If the specified file is not found, the directory specified by the **-i** compiler option, directory specified by the INC78K4 environment variable, and directory \NECTools32\INC78K4 registered in the registry is searched.

6.1.2 Correction of "(13) CPU control instruction"

Descriptions have been added to **FUNCTION** on page 352.

<Incorrect>

FUNCTION

- The following codes are output to an object to create an object file.

(1) Instruction for HALT operation[Note 1]
(2) Instruction for STOP operation[Note 2]
(3) BRK instruction
(4) NOP instruction

Notes 1. The setting of STOP mode and selection of the internal system clock is possible using the STBC register. The C compiler reads STBC, checks the CK1/CK0 value of the internal system clock selection, and accordingly outputs the instruction to set the value for HALT to STBC.

2. The C compiler reads STBC, checks the CK1/CK0 value of the internal system clock selection, and accordingly outputs the instruction to set the value for STOP to STBC.

<Correct>

FUNCTION

- The following codes are output to an object to create an object file.

(1) Instruction for HALT operation[Notes 1, 2]
(2) Instruction for STOP operation[Notes 1, 3]
(3) BRK instruction
(4) NOP instruction

Notes 1. The setting of STOP mode and selection of the internal system clock is possible using the STBC register. The codes for the STBC register are output (codes to clear bits 2, 3, 6, and 7 to 0 are output).

	7	6	5	4	3	2	1	0
STBC	0	0	CK1	CK0	0	0	STP	HLT

2. The C compiler reads STBC, checks the CK1/CK0 value of the internal system clock selection, and accordingly outputs the instruction to set the value of CK1/CK2/STP/HLT for HALT to STBC.

3. The C compiler reads STBC, checks the CK1/CK0 value of the internal system clock selection, and accordingly outputs the instruction to set the value of CK1/CK2/STP/HLT for STOP to STBC.

Descriptions have been added to **RESTRICTIONS** on page 353.

<Incorrect>

RESTRICTIONS

- When this feature is used, HALT( ), STOP( ), BRK( ), and NOP( ) cannot be used as function names.
- Describe HALT, STOP, BRK, and NOP in uppercase letters. If they are described in lowercase letters, they are handled as ordinary functions.

<Correct>

RESTRICTIONS

- When this feature is used, HALT( ), STOP( ), BRK( ), and NOP( ) cannot be used as function names.
- Describe HALT, STOP, BRK, and NOP in uppercase letters. If they are described in lowercase letters, they are handled as ordinary functions.
- 0 is written to bits 2, 3, 6, and 7 of the STBC register when HALT( ) or STOP( ) is used. Therefore, do not use these functions where 0 should not be written, such as in a device with a subclock.