CUSTOMER NOTIFICATION

SUD-DT-03-0192-E

June 20, 2003

Koji Nishibayashi, Senior System Integrator
Microcomputer Group
2nd Solutions Division
Solutions Operations Unit
NEC Electronics Corporation

CP(K), O

78K0 Series C Compiler
CC78K0 V3.50
Operating Precautions

PC-9800 Series (Windows) based
IBM PC/AT and Compatibles (Windows) based
HP9000 Series 700 (HP-UX) based
SPARCstation Family (SunOS, Solaris) based

Be sure to read this document before using the product.

1. PRODUCT OUTLINE

Use the 78K0 Series C compiler CC78K0 in combination with the assembler package RA78K0.

1.1 When Using CC78K0 in Windows

PM plus is necessary when using the CC78K0 in Windows. PM plus is included in the assembler package.

1.2 When Using CC78K0 from Command Line

The C compiler can be activated from the DOS prompt (Windows 98 or Windows Me) or command prompt (Windows NT 4.0, 2000, or XP) in Windows, and from the command line of a shell in UNIX.

2. CAUTIONS

See the CC78K0 Ver. 3.50 or Later Operation User's Manual APPENDIX B and APPENDIX C for cautions related to the compiler.

2.1 Device File

A device file is necessary to execute the CC78K0. The device file is not included in the CC78K0 package. Download the device file via ODS (online delivery service).

NEC Electronics Microprocessor home page (URL: http://www.necel.com/micro/index_e.html)
→ [Development Tools Download] → [DeviceFile]

2.2 Supported Tools

When CC78K0 V3.50 is used with other tools, use tools with the following versions.

Tools		Version
Assembler package	RA78K0	V3.60 or later
Integrated debugger	ID78K0-NS	V2.52 or later
System simulator	SM78K0	V2.52 or later

2.3 Installation (Windows Version)

A product ID is required to install CC78K0 V3.50. The product ID is shown on the CD-ROM case.

3. ITEMS REVISED FROM V3.40 TO V3.50

3.1 Change of Specifications

The specifications changed in V3.50 are as follows.

- PM plus is now supported. See the PM plus V5.10 User's Manual for details of PM plus.
- A warning message is now output for an unreferenced the label. This enables detection of spelling errors in the case or default label in a switch statement.
- A warning message is now output to an operator that has no operation. This enables detection of description errors in an assignment statement.

3.2 Correction of Bugs

The bugs corrected in V3.50 are as follows.

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 if the address of a function allocated to the flash area is referenced.
3	The constant expression of #if may not be processed correctly.
4	If values are read in order of floating-point type to integer type in scanf/sscanf, the value of the integer type cannot be input correctly.
5	An illegal code is output if a structure pointer is assigned to a register and data is passed between members pointed to by the pointer.
6	An illegal code is output if a negative floating-point constant is cast to an unsigned integer type.
7	An illegal code is output if a logical OR or logical AND operation is performed between floating-point constants.
8	An illegal code is output if an array includes a function parameter that is declared as register.
9	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.
10	An illegal code is output if a logical OR or logical AND operation is performed between a floating-point constant and integer-type constant.

Details of the corrected bugs are described on the following pages.

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 the address of a function allocated to the flash 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.3 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.4 If values are read in order of floating-point type to integer type in scanf/sscanf, the value of the integer type cannot be input correctly.

(Example)

```
void func()
{
    int i;
    float f;

    sscanf("1.2 10", "%f%d", &f, &i);
}
```

[Workaround]

There is no workaround.

No.5 An illegal code is output if a structure pointer is assigned to a register and data is passed between members pointed to by the pointer.

(Example)

```
struct tag {
    int a;
    int b;
};
void func()
{
    register struct tag *sp;
    sp->b = sp->a;
}
```

[Workaround]

There is no workaround.

No.6 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.

(Example)

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

No.7 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.8 An illegal code is output if an array includes a function parameter that is declared as register.

(Example)

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

[Workaround]

Describe the parameter as a pointer type.

(Example)

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

No.9 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)

```
--- a.c ---
#include "a1.h"
#include "a2.h"
#include "a3.h"

--- a1.h ---
#pragma section @@DATA DAT1
int a;
```

```
#pragma section @@DATA DAT2

--- a2.h ---
#pragma section @@DATA DAT3
int b;
#pragma section @@DATA DAT4

--- a3.h ---
#pragma section @@DATA DAT5
extern int a; /* same when int a;  */
#pragma section @@DATA DAT6
```

[Workaround]

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

No.10 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.

4. CHANGED POINTS IN USER'S MANUAL AND ONLINE HELP

- Correction of "environmental variable INC78K" to "environmental variable INC78K0" described in CC78K0 C Compiler Language User's Manual and online help.

- Modification of document numbers described in the user's manual and online help.

	(Incorrect)	(Correct)
CC78K0 Operation User's Manual	U14297	U16613
CC78K0 Language User's Manual	U16628	U14298
RA78K0 Operation User's Manual	U14445	U16629
RA78K0 Language User's Manual	U16630	U14446

- Cautions and restrictions described in the online help assume CC78K0 Ver. 3.30. See the CC78K0 C Compiler Ver. 3.50 or Later Operation User's Manual when using CC78K0 Ver. 3.50.

- Deletion of (d) Project file from No.23 [Cautions when using PM plus] in APPENDIX B LIST OF USE-RELATED CAUTIONS in the CC78K0 C Compiler Ver. 3.50 or Later Operation User's Manual.

- Addition of the following item to APPENDIX B LIST OF USE-RELATED CAUTIONS in the CC78K0 C Compiler Ver. 3.50 or Later Operation User's Manual.

  No.32 [Caution related to Kanji characters]

  When using a source including an EUC code in Windows, set the environmental variable LANG78K to euc or specify the -ZE option.