Notes on Usage of the RX Family C/C++ Compiler V.1.00 Release 01 and Corrections in the User's Manual

There are notes on usage of the RX Family C/C++ compiler V.1.00 Release 01 and corrections to be made in the bundled user's manual (REJ10J2062-0100) as listed below.

1. Notes on Usage

1.1 Note on a Case of the C1804 Message

[C/C++ Compiler]

When the int_to_short option is specified and a file including a C standard header is compiled as C++ or EC++, the compiler may show the C1804(W) message. In this case, simply ignore the message because there are no problems.

[NOTE] In compilation of C++ or EC++, the int_to_short option will be invalid.

Data that are shared between C and C++ (EC++) program must be declared as the long or short type rather than as the int type.

1.2 Note on using MVTC or POPC instructions

[Assembler]

In the assembly language, the program counter (PC) cannot be specified for MVTC or POPC instructions.

1.3 Note on the delete Option for Linkage

[Optimizing linkage editor]

When a function symbol is removed by the delete option, its following function in the source program is not allowed to have a breakpoint at its function name on the editor in your debugging. If you would like to set a breakpoint at the function entrance, set the breakpoint via the Label window or at the prologue code of the function.

1.4 Notes of Paths Name and File Name

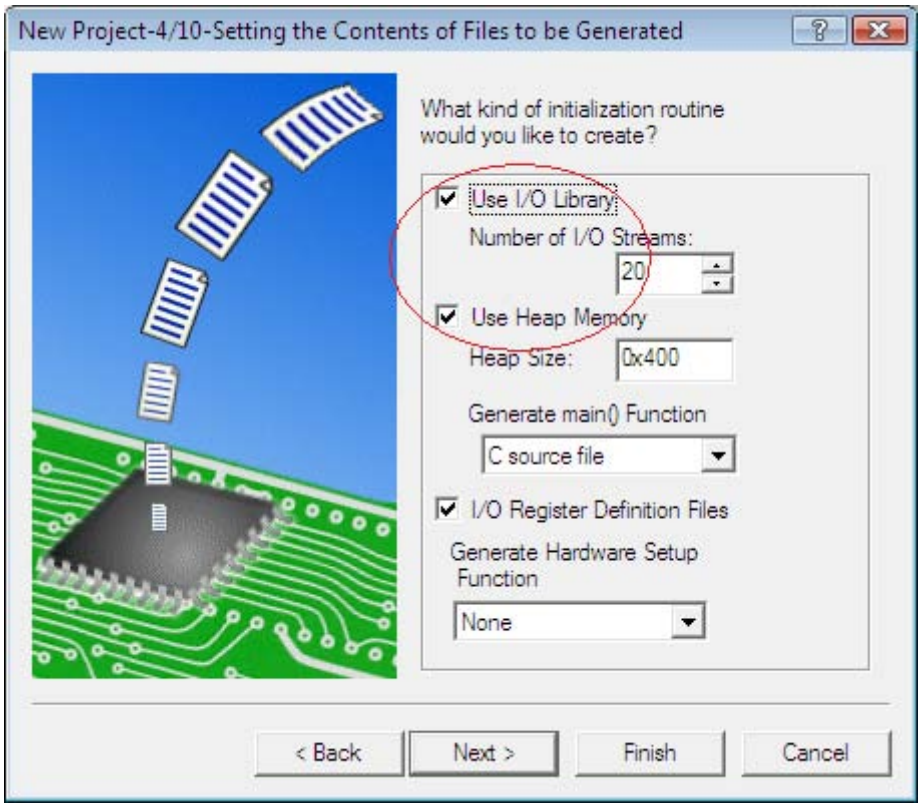
[Optimizing linkage editor]

In the specification of the optimizing linkage editor, the parentheses "(" and ")" signs for the path names and file names cannot be used because these signs are used for the option descriptions.

1.5 Note on Using the I/O Library

[High-Performance Embedded Workshop - Generating Projects]

Tick [Use Heap Memory] if [Use IO Library] has been ticked for generation of a project.



If [Use I/O Library] is ticked and [Use Heap Memory] is not, the following message will be output.

```
L2310 (E) Undefined external symbol "_sbrk" referenced in "xgetmem"
```

When you have encountered this problem, you should add the following C program to your project.

```
#include <stddef.h>
#include <stdio.h>

#define HEAPSIZE 0x400

signed char  *sbrk(size_t size);

union HEAP_TYPE {
    signed long  dummy ;
    signed char  heap[HEAPSIZE];
};

static union HEAP_TYPE heap_area ;

/* End address allocated by sbrk     */
static signed char *brk=(signed char *)&heap_area;

signed char  *sbrk(size_t size)
{
    signed char  *p;

    if(brk+size > heap_area.heap+HEAPSIZE){
        p = (signed char *)-1;
    }
    else {
        p = brk;
        brk += size;
    }
    return p;
}
```

2. Corrections in the User's Manual

2.1 Additions and Corrections

■ (Page 17) 2.1 Source Options / check option / check=nc

[Correction]

Before:

When **check=nc** is specified, a message is output for the following:

Now:

When **check=nc** is specified, the following options and types will be checked:

■ (Page 17) 2.1 Source Options / check option / check=ch38

[Correction]

Before:

When **check=ch38** is specified, a message is output for the following:

Now:

When **check=ch38** is specified, the following options and types will be checked:

■(Page 35, 36) 2.4 Optimize Options / inline,noinline options

[Addition]

Remarks: A value from 0 to 65535 is specifiable as <numerical value>.

■(Page 37) 2.4 Optimize Options / case options

[Correction]

Before:

The jump table is output to a constant area section.

Now:

The jump table is output to a switch statement branch table area section.

■(Page 41) 2.4 Optimize Options / map, smap, nomap options

[Correction]

Before:

[output=obj or output=src is specified]

Now:

[output=obj is specified]

■(Page 42) 2.4 Optimize Options / map,smap,nomap options

[Addition]

Remarks:

This option is only valid for C/C++ files. The optimization is not performed for C/C++ files with output=src option or assembly sources.

■(Page 49) 2.5 Microcontroller Options / round options

[Addition]

Remarks:

The rounding method of floating-point operations in program execution cannot be modified by this option.

■(Page 49) 2.5 Microcontroller Options / denormalize options

[Addition]

Remarks:

The treatment of denormalized numbers in floating-point operations in program execution cannot be modified by this option.

■(Page 65) Table 3.1 Library Generator Options / No.5 lang option

[Correction]

Before:

Creates a reentrant library.

After:

Selects the available functions in the C standard library.

■ (Page 65) Table 3.1 Library Generator Options

[Addition]

No. 6

cpu=rx600 Creates a library for the RX600 series.

■(Page 142) 5.2.7 Other Options / DELete option

[Addition]

Remarks:

When form=library is specified, modules can be deleted.

When form={absolute|relocate|hexadecimal|stype|binary} is specified, external symbols can be deleted.

■(Page 240) 9.1.2 Internal Data Representation / Table 9.15 / No.22 bool

[Correction]

Before:

Size = 4

Alignment = 4

Sign = Used

After:

Size = 1

Alignment = 1

Sign = -- (Unspecified)

■(Page 240) 9.1.2 Internal Data Representation / Table 9.15 / No.22 bool

[Addition]

The _Bool type can be used in C99 comlilation.

The _Bool type is compiled as the bool type.

■(Page 240) 9.1.2 Internal Data Representation / Table 9.15 / No.22 bool / Notes *5

[Correction]

Before:

Notes:

5. This data type is valid for C++ and C99 compilation only.

After:

Notes:

5. This data type is only valid for compilation in C++, and C99 with stdbool.h included.

■(Page 250) 9.1.2 Internal Data Representation / Table 9.17 / No.1 / Notes *1(bool)

[Correction]

Before:

Notes:

1. The bool type is only valid for C++ and C99 programs.

After:

Notes:

1. The bool type is only valid for programs in C++, and C99 with stdbool.h included.

■(Page 259) 9.1.3 Floating-Point Number Specifications / (2)float Type / Note:

[Correction]

Before:

Note: A not-a-number is called a quiet NaN when the MSB of the mantissa is 0, or signaling NaN when the MSB of the mantissa is 1.

After:

Note: A not-a-number is called a quiet NaN when the MSB of the mantissa is 1, or signaling NaN when the MSB of the mantissa is 0.

■(Page 261) 9.1.3 Floating-Point Number Specifications / (3) double Types and long double Types / Note:

Before:

Note: A not-a-number is called a quiet NaN when the MSB of the mantissa is 0, or signaling NaN when the MSB of the mantissa is 1.

After:

Note: A not-a-number is called a quiet NaN when the MSB of the mantissa is 1, or signaling NaN when the MSB of the mantissa is 0.

■(Page 275) 9.2.1 #pragma interrupt / Remarks:

[Addition]

When **vect** is used as an interrupt specification, the address of all empty vectors is 0. You can specify a desired address value or symbol for that address with the optimizing linkage editor. For details, refer to the descriptions on the **VECT** and **VECTN** options in section 5.2.2, Output Options.

■(Page 301 / (Page 290)) 9.2.2 Intrinsic Functions / (Table 9.24 Intrinsic Functions No.14)

[Correction]

Before:

void int_exception(unsigned long num)

Now:

void int_exception(signed long num)

■(Page 303 / (Page 290)) 9.2.2 Intrinsic Functions / (Table 9.24 Intrinsic Functions No.17)

[Correction]

Before:

void set_ipl(unsigned long level)

Now:

void set_ipl(signed long level)

■(Page 666) Table 10.1 Types of Name / Label name, Symbol name

[Addition]

(Each symbol name includes a label name.)

■(Page 667) 10.1.4 Coding of Labels

[Addition]

Remarks:

You cannot define a section name which is the same as an existing symbol name. If a symbol and section of the same name are defined, the first one will be effective, and the others will lead to the

A2118 error.

■(Page 716) 10.3.3 Link Directives / .SECTION / Remarks:

[Addition]

You cannot define a symbol name which is the same as an existing section name. If a section and symbol of the same name are defined, the first one will be effective, and the second one will lead to the A2118 error.

The section name **$iop** is reserved and cannot be defined. If this is attempted, error A2049 is reported.

```
.SECTION  $iop,code    ; Error A2049
```

■(Page 723) 10.3.6 Extended Function Directives / Table 10.35 Extended Function Directives

[Deletion]

Note: .FILE is included in the assembly-language file generated by the compiler. .FILE is valid only in the assembly-language file generated by the compiler; do not use it in a user-created assembly-language file.

■(Page 828) 12.2 List of Messages / A2040 (E) Include nesting over

[Correction]

Before:

Rewrite include so that it is nested within the valid levels.

After:

Rewrite include so the depth of include nest is below or equal to 30.

■(Page 836) 12.2 List of Messages / A3202 (F) Can't find work dir

[Correction]

Before:

The work directory is not found.

After:

The work directory is not found.

Make sure that the correct value is set to environment variable TMP_RX.

■ (Page 842) 13.2 List of Messages / L1200 (W) Backed up file "file 1" into "file 2"

[Correction]

Before:

The file **file 1** was backed up to the file **file 2**.

Now:

The **file 1** has been overwritten. A backup copy of the data in the previous **file 1** was saved in **file 2**.

■ (Page 861) 15.1 Notes on Program Coding / (4) Overflow Operation and Zero Division

[Deletion]

```
fa=3.5e+40f; /* (W) Detects overflow in floating-point operation */
```

Note: The sample code given in the example will lead to error C5030 (E).
C5030 (E) Floating constant is out of range

■ (Page 863) 15.1 Notes on Program Coding / (8) Differences between C89 Operation and C99 Operation

[Correction]

Before:

If the above code is compiled with **-lang=c99** specified, it is interpreted as follows:

```
enum {a,b};
 int g(void)
 {
    if(sizeof(enum{b,a}))
    {
       return a;
    }
    return b;
 }
```

Now:

If the above code is compiled with **-lang=c99** specified, it is interpreted as follows:

```
enum {a,b};
int g(void)
{
   {
      if(sizeof(enum{b,a}))
         return a;
   }
   return b;
```

```
}
```

■ (Page 864) 15.1 Notes on Program Coding

[Addition]

(9) Operations and Type Conversions That Lead to Overflows

The results of operations or type conversions must be within the range of values allowed for the selected type (i.e. values must not overflow). If an overflow occurs, the result of the operation or type conversion may be affected by other conditions such as compiler options.

In the standard C language, the results of operations that lead to overflows are undefined and thus may differ depending on the current compiling conditions. Ensure that overflows will not be caused by any operations in programs.

The following example illustrates this problem.

Example: Type conversion from **float** to **unsigned short**

```
float f = 2147483648.0f;
unsigned short ui2;
void ex1func(void)
{
    ui2 = f;    /* Type conversion from float to unsigned short */
}
```

The value of **ui2**, which is acquired as the result of executing **ex1func**, depends on whether **–fpu** or **–nofpu** has been specified.

-fpu (with the FPU): ui2 = 65535

-nofpu (without the FPU): ui2 = 0

This is because the method of type conversion from **float** to **unsigned short** differs depending on whether **–fpu** or **–nofpu** has been specified.