# Compiler Package

Stack Estimation Using Call Walker for CS+

## Introduction

The CS+ compiler package includes the Call Walker, a stack calculation utility. The Call Walker can be used to analyze the stack information obtained during the build process and to check the sizes of stack usage by individual function trees and symbols in the trees. This document describes the usage of the Call Walker.

Target Software: Call Walker V.2.05 and later versions (contained in the CS+ package)

## Overview

The Call Walker reads a stack usage information file (*.sni) output by the optimizing linkage editor or a profile information file (*.pro) output by the simulator debugger, displays the relationships between the calling and called functions in tree form, and lists the stack information (symbol name, attribute, address, size, static stack size, and file name) for each function symbol.

The user can edit the stack information to estimate dynamic stack sizes. The edited information can be saved in a call information file (*.cal), which can be read for later processing.

## Contents

# 1. Overview of the Call Walker

## 1.1 Configuration of Files

The Call Walker will usually read a stack usage information file (*.sni) output by the optimizing linkage editor and display the static stack sizes for each of the symbols.

The following shows the relationships between this tool (the Call Walker) and the files to be used.

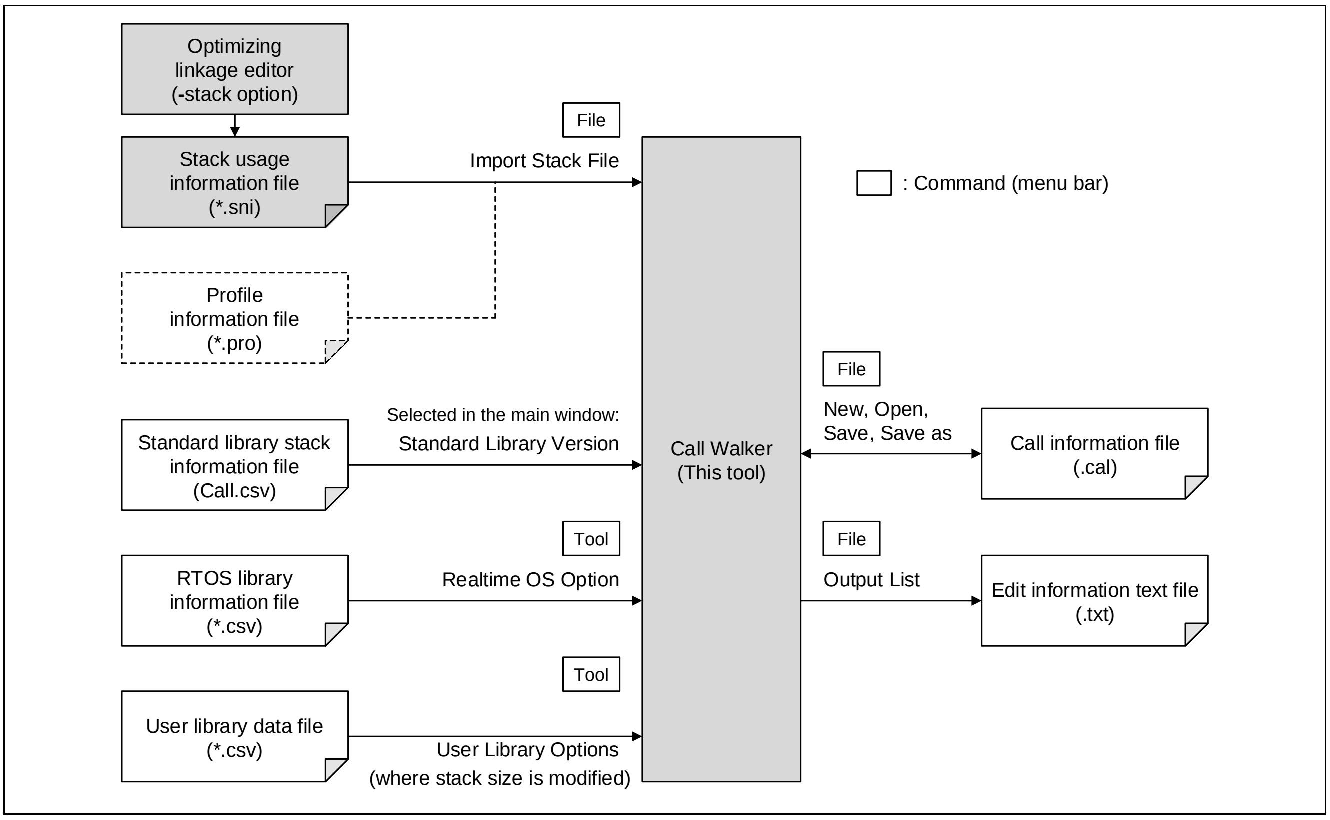

Optimizing linkage editor (-stack option)
Stack usage information file (*.sni)
File
Import Stack File
Profile information file (*.pro)
: Command (menu bar)
Selected in the main window:
Standard Library Version
Standard library stack information file (Call.csv)
Call Walker (This tool)
File
New, Open, Save, Save as
Call information file (.cal)
Tool
Realtime OS Option
RTOS library information file (*.csv)
File
Output List
Edit information text file (.txt)
Tool
User library data file (*.csv)
User Library Options (where stack size is modified)

**Figure 1-1 Call Walker and Related Files**

## 1.2 Stack Usage

The stacks in memory are generally used as working areas for storing the following items.

- Program counter values (addresses) to which execution is to return from functions
- Parameters for functions
- Values returned by functions
- Local variables
- Temporary variables (variables for temporary use in processing by the compiler)
- Exception handling context

Note that the actual stack handling (such as the locations for storing the parameters and return values for function calls or the relationship between the optimization settings and the stack usage) depends on the architecture of the CPU and the specifications of the compiler.

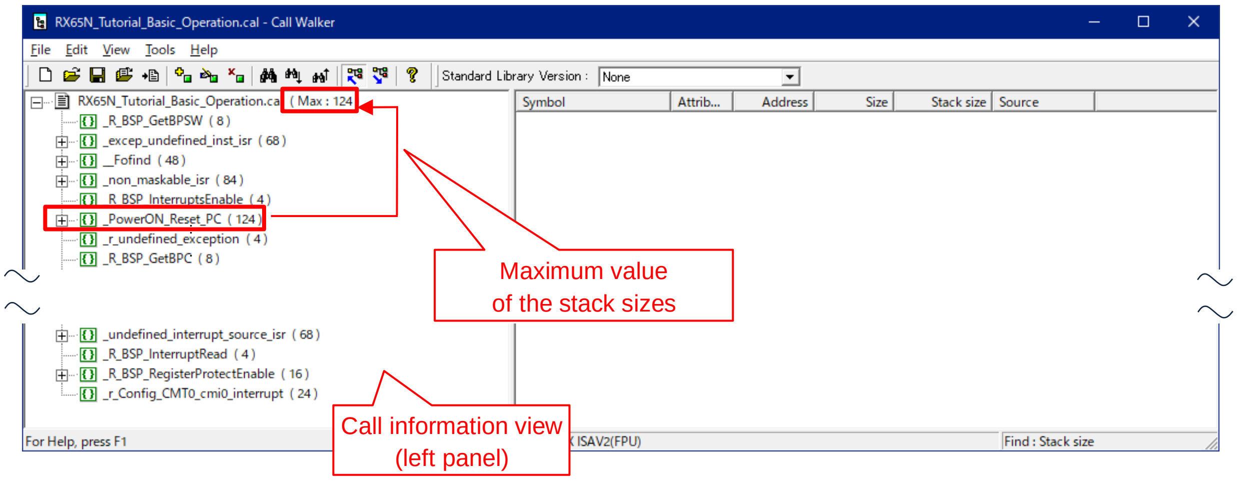
Regarding stack handling in the environment you are using, refer to the user's manuals of the compiler and real-time OS (if you will be using one).

## 1.3 Display of Stack Sizes by the Call Walker

The value displayed to the right of "Max:" at the top of the call information view (the panel in the left side of the Call Walker window) is the maximum value among the static stack sizes calculated based on the output from the optimizing linkage editor. Note that some restrictions apply to the use of the Call Walker; be sure to read section 1.4, Notes on Call Walker Usage.

This value is the maximum among the stack sizes used by functions in the function tree (next to the file name at the top of the hierarchy).

How the stack usage at the second and lower levels of the tree hierarchy is displayed depends on whether [Show Required Stack] or [Show Used Stack] is selected from the [View] menu. See Figure 5-3, Dependence of Display on the Selection of [Show Required Stack] or [Show Used Stack].



```
RX65N_Tutorial_Basic_Operation.cal ( Max : 124 )
    _R_BSP_GetBPSW ( 8 )
  + _excep_undefined_inst_isr ( 68 )
  + __Fofind ( 48 )
  + _non_maskable_isr ( 84 )
    _R_BSP_InterruptsEnable ( 4 )
  + _PowerON_Reset_PC ( 124 )
    _r_undefined_exception ( 4 )
    _R_BSP_GetBPC ( 8 )

  + _undefined_interrupt_source_isr ( 68 )
    _R_BSP_InterruptRead ( 4 )
  + _R_BSP_RegisterProtectEnable ( 16 )
    _r_Config_CMT0_cmi0_interrupt ( 24 )
```

RX65N_Tutorial_Basic_Operation.cal - Call Walker
File Edit View Tools Help
Standard Library Version : None
Symbol
Attrib...
Address
Size
Stack size
Source
Maximum value of the stack sizes
Call information view (left panel)
For Help, press F1
ISAV2(FPU)
Find : Stack size

**Figure 1-2 Displaying the Maximum Stack Size by the Call Walker**

## 1.4 Notes on Call Walker Usage

Note the following points when using the Call Walker.

- The stack sizes will depend on the build settings (such as optimization options), even for the same program.
- The dynamic stack sizes such as those used for recursive functions, multiple interrupts, etc. cannot be displayed.
- When a function is called through a function pointer, the stack size for the function cannot be calculated.
- In the generation of a stack usage information file (*.sni), the optimizing linkage editor handles inline assembly-language functions as if they have no stack usage.

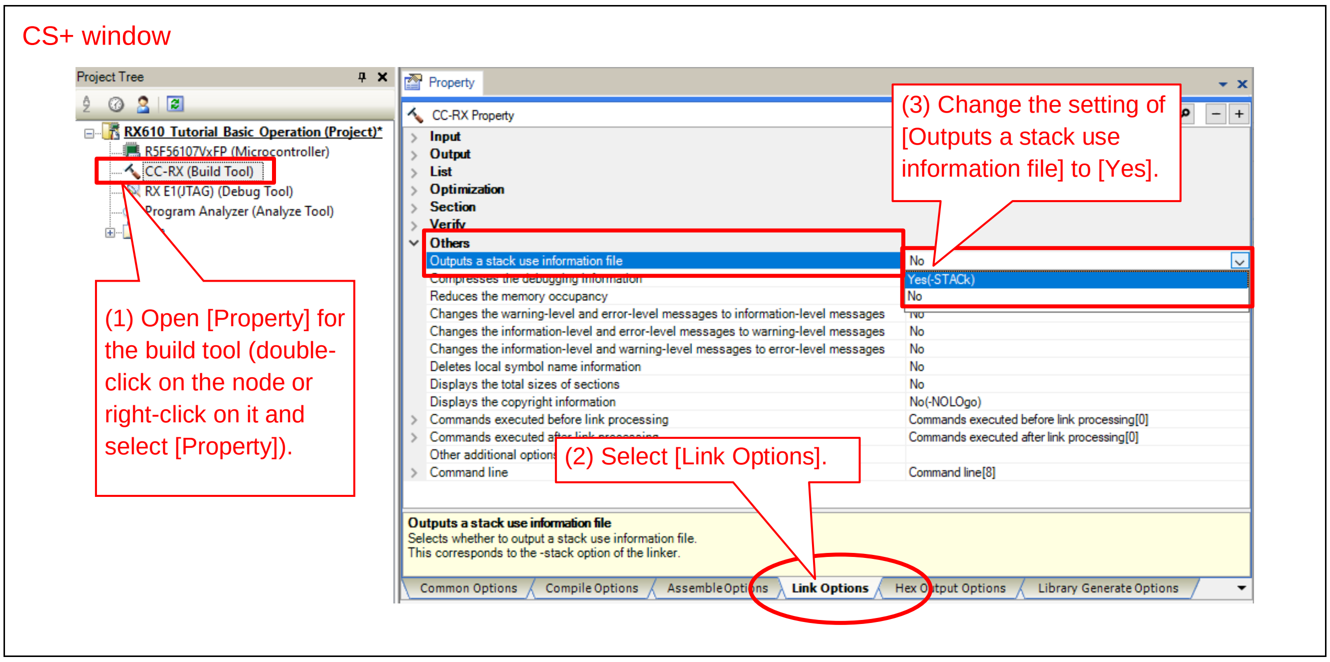
If the Call Walker is not capable of checking the stack size for a function, calculate it manually or edit the route of calls from that function to its subordinate symbols so that the Call Walker can recognize the maximum value of the stack sizes along that route and obtain the maximum stack size.

## 2. Creating Stack Information Files

### 2.1 Stack Usage Information Files (*.sni)

In the [Project Tree] panel of CS+, open the [Property] panel from the [Build Tool] node. Select the [Link Options] tabbed page and select [Yes] for the [Outputs a stack use information file] property. This enables the "-stack" option for the optimizing linkage editor.

Building a program after having made these settings will generate a stack usage information file (*.sni). The default location for storing this file is the "DefaultBuild" folder immediately under the project folder.


CS+ window
Project Tree
RX610 Tutorial Basic Operation (Project)*
R5F56107VxFP (Microcontroller)
CC-RX (Build Tool)
RX E1(JTAG) (Debug Tool)
Program Analyzer (Analyze Tool)
(1) Open [Property] for the build tool (double-click on the node or right-click on it and select [Property]).
Property
CC-RX Property
Input
Output
List
Optimization
Section
Verify
Others
(3) Change the setting of [Outputs a stack use information file] to [Yes].
(2) Select [Link Options].
Common Options
Compile Options
Assemble Options
Link Options
Hex Output Options
Library Generate Options

| | |
|---|---|
| Outputs a stack use information file | No (Yes(-STACk) / No) |
| Compresses the debugging information | |
| Reduces the memory occupancy | |
| Changes the warning-level and error-level messages to information-level messages | No |
| Changes the information-level and error-level messages to warning-level messages | No |
| Changes the information-level and warning-level messages to error-level messages | No |
| Deletes local symbol name information | No |
| Displays the total sizes of sections | No |
| Displays the copyright information | No(-NOLOgo) |
| Commands executed before link processing | Commands executed before link processing[0] |
| Commands executed after link processing | Commands executed after link processing[0] |
| Other additional options | |
| Command line | Command line[8] |

**Outputs a stack use information file**
Selects whether to output a stack use information file.
This corresponds to the -stack option of the linker.

**Figure 2-1 Setting of the Linkage Option for CC-RX**

### 2.2 Profile Information Files (*.pro)

The Call Walker also has a facility for importing a profile information file (*.pro). This file type is supported by the High-performance Embedded Workshop (HEW).

## 3. Starting the Call Walker

The Call Walker is built-in to CS+ and can be started from the toolbar of CS+. Alternatively, the Call Walker can be started as a stand-alone program from the Start menu of Windows; in this case, the information output from the optimizing linkage editor will require manual importing.

### (1) Starting from the Toolbar of CS+

Build a program according to the descriptions in section 2, Creating Stack Information Files, and then start the Call Walker as follows.

Toolbar of CS+ → [Tool] → Click on [Startup Stack Usage Tracer] Note.

Note: If this option does not appear in the [Tool] menu of the toolbar, select [Plug-in Setting] from the [Tool] menu of CS+ and check that "Stack Usage Tracer" is selected in the [Additional Function] tabbed page.

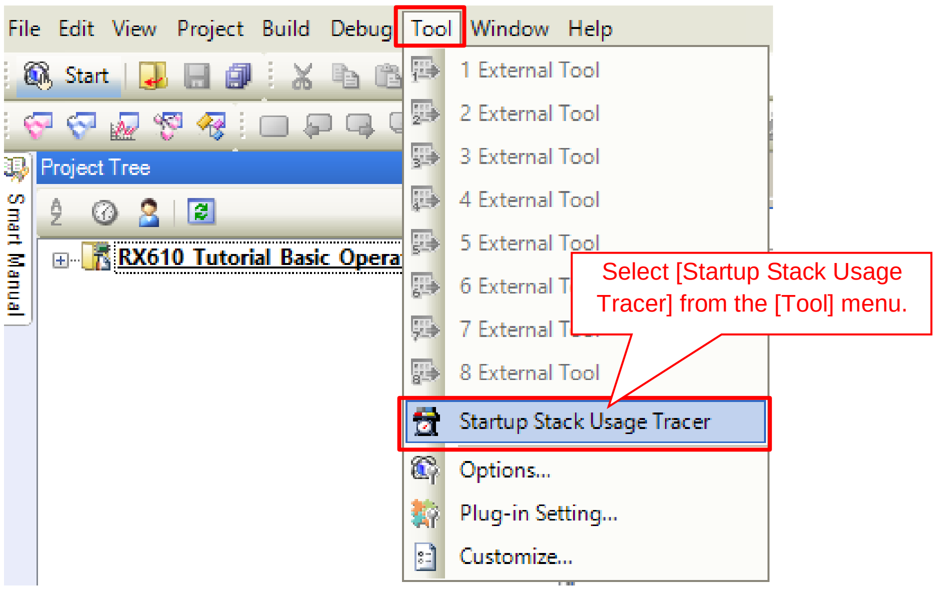


**Figure 3-1 Starting the Call Walker from the Toolbar of CS+**

### (2) Starting from the Start Menu of Windows

The Call Walker can also be started as a stand-alone item of CS+.

Start menu → List of applications → Renesas Electronics CS+ → Select "Call Walker"

After the Call Walker has started, import a stack usage information file (.sni) or a profile information file (.pro).

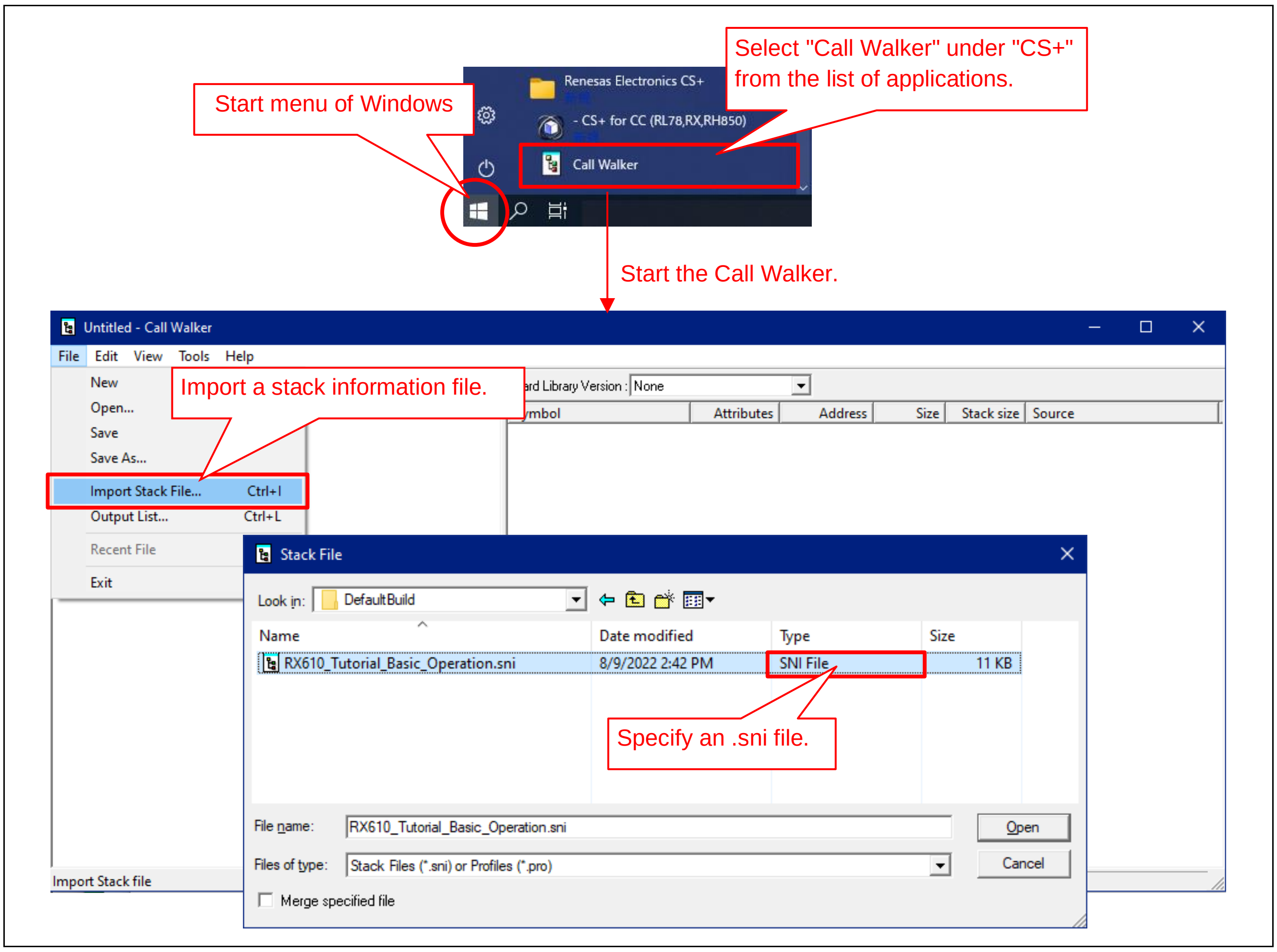

Select "Call Walker" under "CS+" from the list of applications.
Start menu of Windows
Renesas Electronics CS+
- CS+ for CC (RL78,RX,RH850)
Call Walker
Start the Call Walker.
Untitled - Call Walker
File Edit View Tools Help
New
Open...
Save
Save As...
Import Stack File... Ctrl+I
Output List... Ctrl+L
Recent File
Exit
Import a stack information file.
rd Library Version : None
ymbol Attributes Address Size Stack size Source
Import Stack file
Stack File
Look in: DefaultBuild
Name Date modified Type Size
RX610_Tutorial_Basic_Operation.sni 8/9/2022 2:42 PM SNI File 11 KB
Specify an .sni file.
File name: RX610_Tutorial_Basic_Operation.sni
Files of type: Stack Files (*.sni) or Profiles (*.pro)
Open
Cancel
Merge specified file

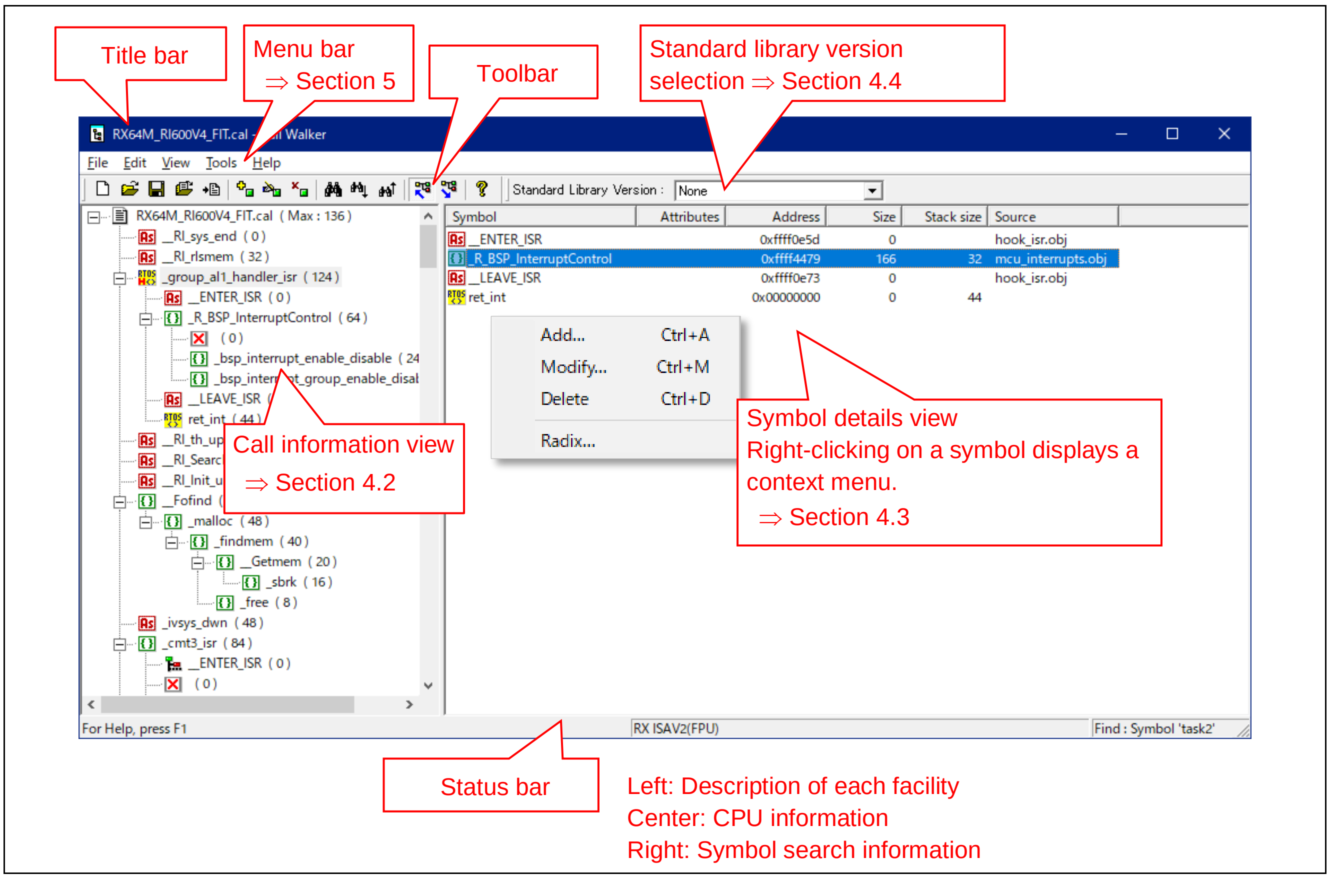
**Figure 3-2 Starting the Call Walker from the Start Menu and Importing Stack Information**

# 4. Main Window

## 4.1 Structure of the Main Window

After the Call Walker has started, the main window will be open. Only one stack information file at a time can be opened with the Call Walker.

The structure of the main window is shown below.


Title bar
Menu bar ⇒ Section 5
Toolbar
Standard library version selection ⇒ Section 4.4
RX64M_RI600V4_FIT.cal - Call Walker
File Edit View Tools Help
Standard Library Version : None
Call information view ⇒ Section 4.2
Add... Ctrl+A
Modify... Ctrl+M
Delete Ctrl+D
Radix...
Symbol details view
Right-clicking on a symbol displays a context menu.
⇒ Section 4.3
For Help, press F1
RX ISAV2(FPU)
Find : Symbol 'task2'
Status bar
Left: Description of each facility
Center: CPU information
Right: Symbol search information

```
RX64M_RI600V4_FIT.cal ( Max : 136 )
  __RI_sys_end ( 0 )
  __RI_rlsmem ( 32 )
  _group_al1_handler_isr ( 124 )
    __ENTER_ISR ( 0 )
    _R_BSP_InterruptControl ( 64 )
       ( 0 )
      _bsp_interrupt_enable_disable ( 24
      _bsp_interrupt_group_enable_disal
    __LEAVE_ISR (
    ret_int ( 44 )
  __RI_th_up
  __RI_Searc
  __RI_Init_u
  __Fofind (
    _malloc ( 48 )
      _findmem ( 40 )
        __Getmem ( 20 )
          _sbrk ( 16 )
        _free ( 8 )
  _ivsys_dwn ( 48 )
  _cmt3_isr ( 84 )
    __ENTER_ISR ( 0 )
     ( 0 )
```

| Symbol | Attributes | Address | Size | Stack size | Source |
|---|---|---|---|---|---|
| __ENTER_ISR | | 0xffff0e5d | 0 | | hook_isr.obj |
| _R_BSP_InterruptControl | | 0xffff4479 | 166 | 32 | mcu_interrupts.obj |
| __LEAVE_ISR | | 0xffff0e73 | 0 | | hook_isr.obj |
| ret_int | | 0x00000000 | 0 | 44 | |

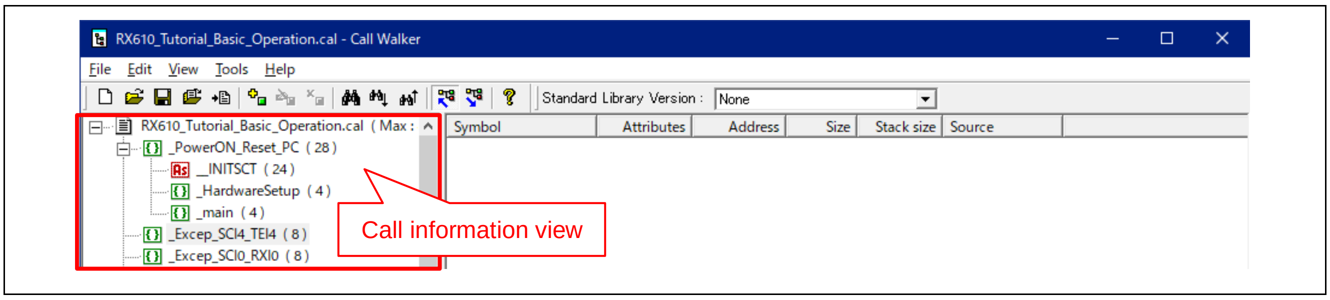
**Figure 4-1 Names of the Components of the Main Call Walker Window**

**Table 4-1 Icons for Symbol Types**

| Icon | Description |
|---|---|
| 📄 | File being edited |
| As | Assembly-language label |
| {} | C/C++ function |
| 🔄 | Direct or indirect recursive call function |
| RTOS <> | RTOS function |
| ? | Function with reference to unknown symbol |
| ❌ | Function with unresolved reference address |
| ⋯ | Abbreviation icon |
| RTOS H<> | RTOS handler function |
| RTOS T<> | RTOS task function |
| (...) | Variadic function |

## 4.2 Call Information View

The call information view (panel in the left of the main window) shows the link hierarchy of function symbols. The number shown in the parentheses to the right of each symbol indicates the stack size used for the symbol.


RX610_Tutorial_Basic_Operation.cal - Call Walker
File Edit View Tools Help
Standard Library Version : None
Symbol Attributes Address Size Stack size Source
Call information view

```
RX610_Tutorial_Basic_Operation.cal ( Max :
  _PowerON_Reset_PC ( 28 )
    __INITSCT ( 24 )
    _HardwareSetup ( 4 )
    _main ( 4 )
  _Excep_SCI4_TEI4 ( 8 )
  _Excep_SCI0_RXI0 ( 8 )
```

**Figure 4-2 Call Information View**

### (1) Icons for Symbols

The following lists the icons for symbol types.

| | |
|---|---|
| 🗎 | File being edited |
| As | Assembly-language label |
| {} | C/C++ function |
| 🔄 | Direct or indirect recursive call function |

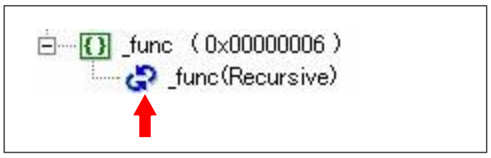
(a) Direct recursive call function

The icon indicates that the function directly calls itself.

Example:

```
void func(int x)
{
    x++;
    if(x != OFF)
        func(x);

    if(x == MAX)
        return;
}
```



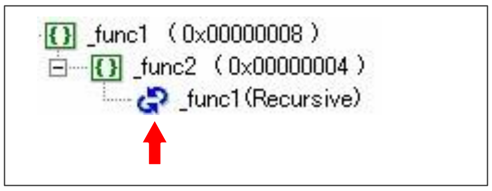
```
_func ( 0x00000006 )
  _func(Recursive)
```

(b) Indirect recursive call function

The icon indicates that the function indirectly calls itself.

Example:

```
void func1(int a)
{
    func2(10);
}
void func2(int b)
{
    func1(9);
}
```



```
_func1 ( 0x00000008 )
  _func2 ( 0x00000004 )
    _func1(Recursive)
```

| | |
|---|---|
| RTOS | RTOS function (such as a symbol in ITRON) |

| | |
|---|---|
| ? | Function with reference to unknown symbol |

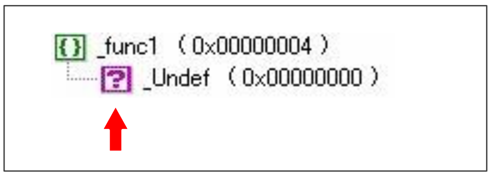
In the following example, the func1() function calls the Undef() function. If no entity is defined for the Undef() function, this icon is displayed for the Undef() function.

Although calling a function without its entity actually results in a linkage error, the error message can be changed to a warning message by using the change_message link option.

When the error message is changed to a warning message, a load module can be generated and therefore a stack information file can also be generated.

Example:

```
void func1(void)
{
    Undef();
}
```



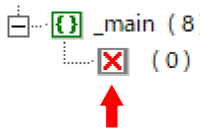
```
{} _func1 (0x00000004)
   ? _Undef (0x00000000)
```

| | |
|---|---|
| ☒ | Function with unresolved reference address |

This icon indicates that the function is indirectly called through a pointer.

Example:

```
void main(void)
{
    void (*ptr)() = subfunc;
    ptr();
    while(1);
}
```



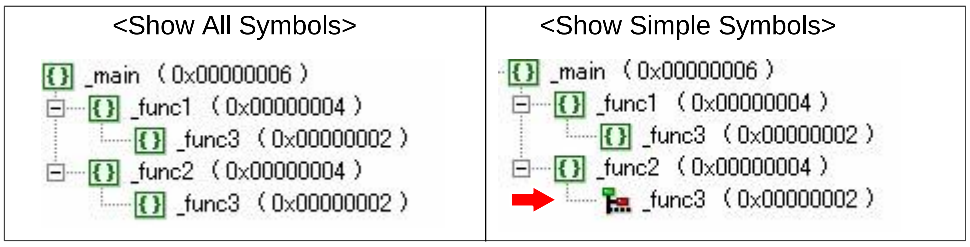
```
⊟ {} _main (8)
     ☒ (0)
```

| | |
|---|---|
| ⚑ | Abbreviation icon |

The Call Walker displays the call structures of all symbols. If the user application is large, the number of symbols to be displayed is enormous.

The number of displayed symbols can be reduced by showing each symbol and its subordinate structure of calls only at the first appearance of the symbol in the view and omitting the subordinate structure at the second and later appearances of the symbol by changing the icon for the symbol as shown below.

Example:



<Show All Symbols>

```
{} _main (0x00000006)
⊟ {} _func1 (0x00000004)
      {} _func3 (0x00000002)
⊟ {} _func2 (0x00000004)
      {} _func3 (0x00000002)
```

<Show Simple Symbols>

```
{} _main (0x00000006)
⊟ {} _func1 (0x00000004)
      {} _func3 (0x00000002)
⊟ {} _func2 (0x00000004)
      ⚑ _func3 (0x00000002)
```

The display mode can be switched by selecting [View] → [Show All Symbols] or [Show Simple Symbols].

| | |
|---|---|
| RTOS H<> | RTOS handler function (#pragma almhandler, #pragma cychandler, #pragma interrupt) |

| | |
|---|---|
| RTOS T<> | RTOS task function (#pragma task, #pragma taskexception) |

| | |
|---|---|
| (·) | Variadic function |

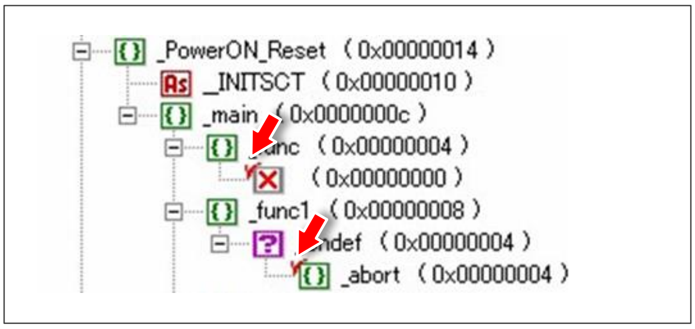
### (2) Displaying Stack Sizes

The number shown in parentheses to the right of each symbol in the call information view indicates the stack size used for the symbol. The display mode of the stack sizes can be switched by selecting [Show Required Stack] or [Show Used Stack] from the [View] menu. Refer to section 5.3, View Menu, for the difference.

If the functions called from a symbol are collapsed in the tree view, the size shown for the symbol is the maximum value among the stack sizes used when the collapsed functions are called.

### (3) Moving Symbols in the Call Information View

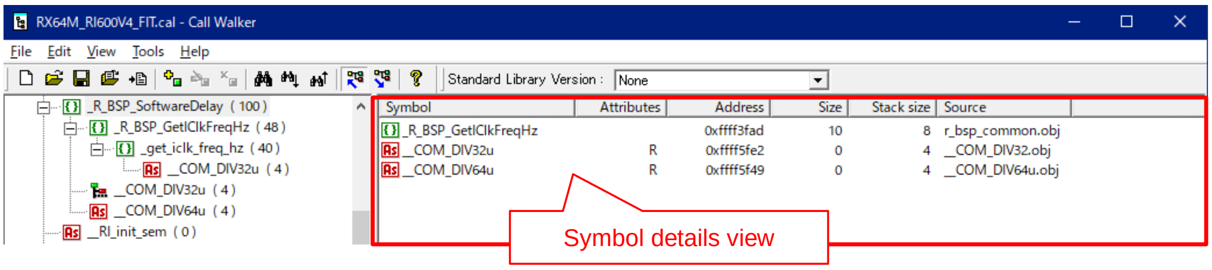
The user can change the positions of symbols by dragging and dropping symbols in the call information view on the panel in the left of the window. Check marks will appear next to the moved or edited symbols in the call information view (indicated by arrows in the figure below).



```
_PowerON_Reset (0x00000014)
  _INITSCT (0x00000010)
  _main (0x0000000c)
    _func (0x00000004)
      (0x00000000)
    _func1 (0x00000008)
      _undef (0x00000004)
        _abort (0x00000004)
```

## 4.3 Symbol Details View

The symbol details view (the panel in the right of the main window) shows a list of information of symbols to which the symbol selected in the call information view refers.


RX64M_RI600V4_FIT.cal - Call Walker
File Edit View Tools Help
Standard Library Version : None
Symbol details view

```
_R_BSP_SoftwareDelay ( 100 )
  _R_BSP_GetIClkFreqHz ( 48 )
    _get_iclk_freq_hz ( 40 )
      __COM_DIV32u ( 4 )
  __COM_DIV32u ( 4 )
  __COM_DIV64u ( 4 )
_RI_init_sem ( 0 )
```

| Symbol | Attributes | Address | Size | Stack size | Source |
|---|---|---|---|---|---|
| _R_BSP_GetIClkFreqHz | | 0xffff3fad | 10 | 8 | r_bsp_common.obj |
| __COM_DIV32u | R | 0xffff5fe2 | 0 | 4 | __COM_DIV32.obj |
| __COM_DIV64u | R | 0xffff5f49 | 0 | 4 | __COM_DIV64u.obj |

**Figure 4-3 Symbol Details View**

The list can be sorted in the ascending or descending order of each information item by clicking on the header for the column of the symbol details view.

The following shows the items of information to be displayed.

**Table 4-2 Items of Information in the Symbol Details View**

| Item | Description |
|---|---|
| Symbol | The icons have the same meanings as those in the call information view. |
| Attributes | Attributes of the symbol<br>R: Runtime library<br>O: Function created with optimization<br>I: Interrupt function<br>S: Static function<br>V: Virtual function<br>L: Function using a local stack |
| Address | Address to which the symbol is allocated |
| Size | Size of the symbol |
| Stack size | Stack size used by the symbol<br>(When the stack size is 0xffffffff, a white space is displayed.) |
| Source | Name of the object file containing the symbol that is referenced |

## 4.4 Selection of the Standard Library Version

Select the standard library version that was used to create the currently open stack information file in the [Standard Library Version] list box. This enables display of the stack sizes used by the assembly-language functions of the standard library.

Notes: 1. Selection of the standard library version is not needed when the RX-family CC-RX compiler is used (leave the selection as "None").

2. For CC-RL V1.03.00 or an earlier version, select "Standard_library_RL78_V1".
For CC-RL V1.04.00 or a later version, select "Standard_library_RL78_V2".

# 5. Commands on the Menu Bar

## 5.1 File Menu

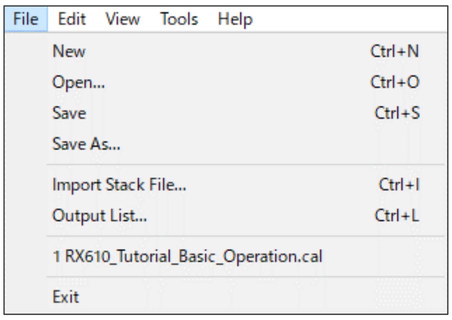


**Table 5-1 Commands in the [File] Menu**

| Command | Description | Icon and Shortcut |
|---|---|---|
| New | The current edit information is discarded and new information is created. If the information currently produced by editing has not been saved, a dialog box for saving the current information appears before the information is discarded. If [Yes] is selected in the dialog box for saving the information, a file is saved in the same way as with the [Save As] command. | Ctrl + N |
| Open | The file selected from the list or specified by entering its name is opened.<br>Selecting the [Merge specified file] checkbox merges the information currently produced by editing with the contents of the selected file with the method specified by the [Merge Option] command in the [Tools] menu (see Figure 5-1). | Ctrl + O |
| Save | The information currently produced by editing is saved by overwriting the current file (*.cal). When this command is executed for a newly created file (no name), the operation is the same as with the [Save As] command. | Ctrl + S |
| Save As | The information currently produced by editing is saved with a specified new file name (*.cal). | — |
| Import Stack File | A stack usage information file (*.sni) output from the optimizing linkage editor is read and data for editing are created.<br>Selecting the [Merge specified file] checkbox merges the information currently produced by editing with the contents of the selected file with the method specified by the [Merge Option] command in the [Tools] menu (see Figure 5-2). | Ctrl + I |
| Output List | The information currently produced by editing is output as a text file (*.txt). | Ctrl + L |
| Recent files | Up to the four most recently used files are listed in the [File] menu. This list is convenient for selecting and opening a recently used file in the next session of the Call Walker. | — |
| Exit | The Call Walker session is terminated. If the information currently produced by editing has not been saved, a dialog box appears to confirm whether to save the information before the Call Walker session is terminated. | — |

### (1) Dialog Box for the [Open] Command

Selecting the [Open] command opens the following dialog box.

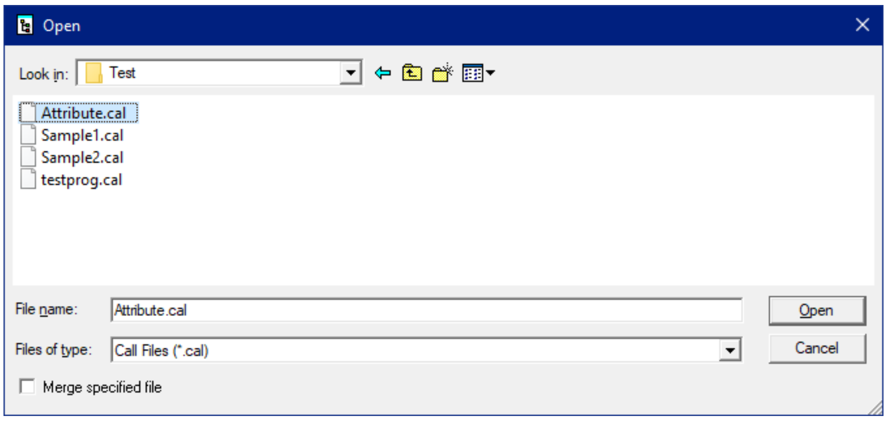


**Figure 5-1 Dialog Box for the [Open] Command**

Select a file from the list or enter a file name and click on the [Open] button to open the file.

Selecting the [Merge specified file] checkbox merges the information currently produced by editing with the contents of the selected file with the method specified by the [Merge Option] command. For the methods of merging, refer to the description of the [Merge Option] command in the [Tools] menu. When this checkbox is not selected, the information currently produced by editing is discarded and the selected file is opened.

If the information currently produced by editing has not been saved before the [Open] command is selected, a dialog box appears to confirm whether to save the information before the file is opened. Click on the [No] button to discard the current information. Click on the [Cancel] button to abort the [Open] command.

### (2) Dialog Box for the [Import Stack File] Command

Selecting the [Import Stack File] command opens the following dialog box.

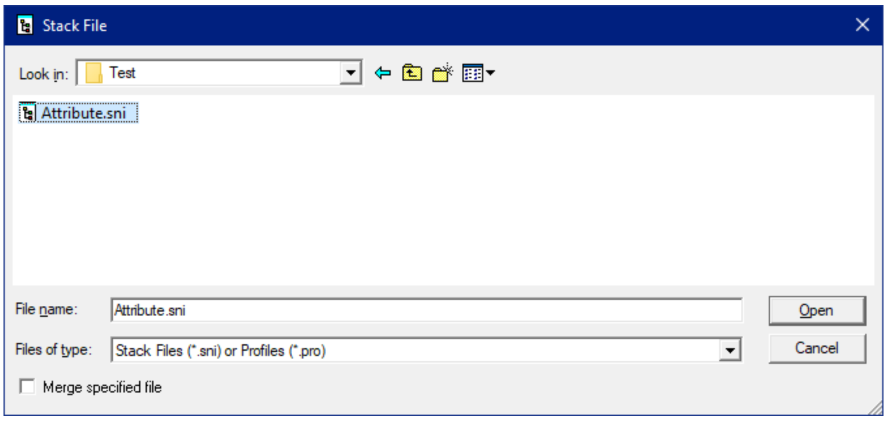


**Figure 5-2 Dialog Box for the [Import Stack File] Command**

Select a file from the list or enter a file name and click on the [Open] button to open the file, create data for editing, and display the data.

Selecting the [Merge specified file] checkbox produces merging of the information currently produced by editing with the contents of the selected file with the method specified by the [Merge Option] command. For the methods of merging, refer to the description of the [Merge Option] command in the [Tools] menu. When this checkbox is not selected, the information currently produced by editing is discarded and the selected file is opened.

## 5.2 Edit Menu

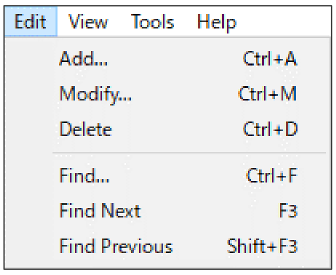
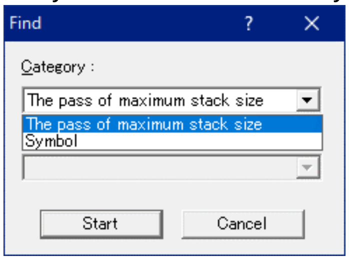


**Table 5-2 Commands in the [Edit] Menu**

| Command | Description | Icon and Shortcut |
|---|---|---|
| Add | New symbol information is added to the level of the hierarchy immediately below the currently selected symbol through a dialog box.<br>(Refer to section 7, Editing Stack Information.) | Ctrl + A |
| Modify | The currently selected symbol information is modified through a dialog box. Note that the [Modify] command cannot be used when an RTOS symbol is selected.<br>(Refer to section 7, Editing Stack Information.) | Ctrl + M |
| Delete | The currently selected symbol information and the symbols under the current symbol are deleted. No message for confirmation will appear before deletion. | Ctrl + D |
| Find | A backward search for symbol information in the call information proceeds.<br>• The pass of maximum stack size: Searches for the path along which the maximum stack size is found.<br>• Symbol: Searches for a symbol name.<br> Find ? ✕<br>Category :<br>The pass of maximum stack size<br>The pass of maximum stack size<br>Symbol<br>Start<br>Cancel | Ctrl + F |
| Find Next | A forward search for symbol information proceeds with the condition specified by the previous [Find] command (the condition shown to the right of "Find:" on the status bar). | F3 |
| Find Previous | A backward search for symbol information proceeds with the condition specified by the previous [Find] command (the condition shown to the right of "Find:" on the status bar). | Shift + F3 |

## 5.3 View Menu

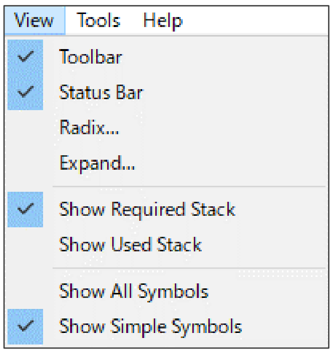
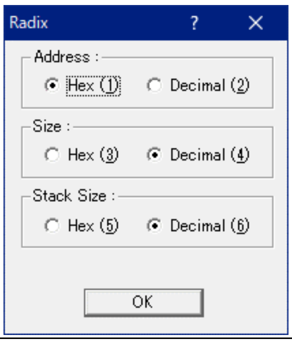
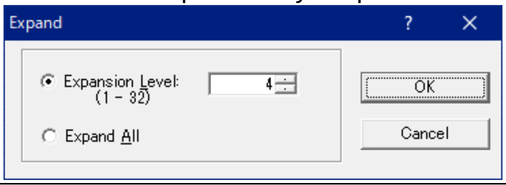


**Table 5-3 Commands in the [View] Menu**

| Command | Description | Icon and Shortcut |
|---|---|---|
| Toolbar | The toolbar is displayed or hidden. A check mark is displayed next to this command name while the toolbar is being displayed. | — |
| Status Bar | The status bar is displayed or hidden. A check mark is displayed next to this command name while the status bar is being displayed. | — |
| Radix | The radices of the numerical values (the "Address", "Size", and "Stack Size" columns) in the symbol information are changed through a dialog box. The radix of the values (total stack sizes) in parentheses in the call information view is also changed to that specified for "Stack Size".<br><br>Radix<br>Address : [✓ Hex (1)] Decimal (2)<br>Size : Hex (3) [✓ Decimal (4)]<br>Stack Size : Hex (5) [✓ Decimal (6)]<br>OK | — |
| Expand | The list of the symbols called from the selected symbol is expanded. The number of levels of expansion can be selected or expansion of all levels can be specified by "Expand All:" through the dialog box.<br><br>Expand<br>[✓ Expansion Level: (1 - 32)] 4<br>Expand All<br>OK<br>Cancel | — |
| Show Required Stack<br>or<br>Show Used Stack | The mode for displaying the stack sizes in the call information view is selected.<br>• Show Required Stack: The displayed stack sizes are cumulative from the lower-level symbols to the higher-level symbols.<br>• Show Used Stack: The displayed stack sizes are cumulative from the higher-level symbols to the lower-level symbols. (See Figure 5-3.) | Show Required Stack<br>Show Used Stack |
| Show All Symbols<br>or<br>Show Simple Symbols | Abbreviation of symbol display in the call information view is selected.<br>• Show All Symbols: All symbols are displayed without omission. | — |

| | | |
|---|---|---|
| | • Show Simple Symbols: The subordinate structure of calls under each symbol is only displayed at the first appearance of the symbol in the call information view and the second and later appearances are displayed in the abbreviated form. | |

### (1) Dependence of Display on the Selection of [Show Required Stack] or [Show Used Stack]

The settings and display of the stack sizes in the call information view are as shown below.

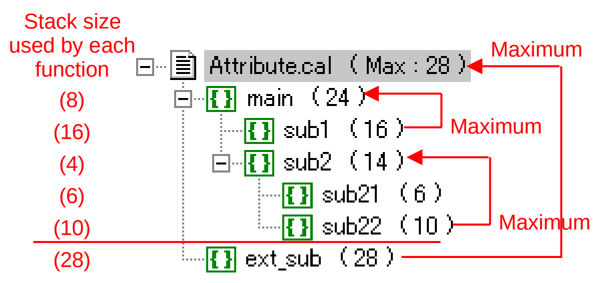
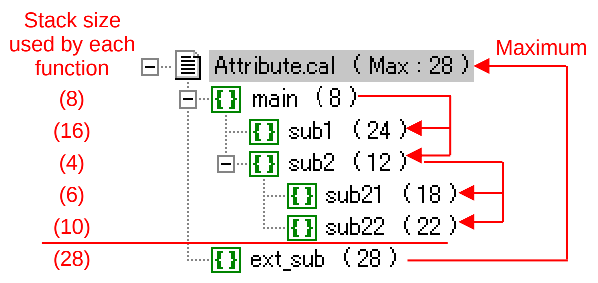
| Show Required Stack | Show Used Stack |
|---|---|
|  Stack size used by each function<br>Attribute.cal (Max : 28)<br>Maximum<br>(8) main (24)<br>(16) sub1 (16)<br>Maximum<br>(4) sub2 (14)<br>(6) sub21 (6)<br>(10) sub22 (10)<br>Maximum<br>(28) ext_sub (28) |  Stack size used by each function<br>Attribute.cal (Max : 28)<br>Maximum<br>(8) main (8)<br>(16) sub1 (24)<br>(4) sub2 (12)<br>(6) sub21 (18)<br>(10) sub22 (22)<br>(28) ext_sub (28) |
| Starting from the lowest-level function, the maximum value of the stack sizes among the lower-level symbols is added to the stack size used by the higher-level symbol. | The stack size of the higher-level symbol is added to the stack sizes of the lower-level symbols. |

**Figure 5-3 Dependence of Display on the Selection of [Show Required Stack] or [Show Used Stack]**

## 5.4 Tools Menu

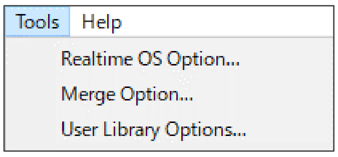
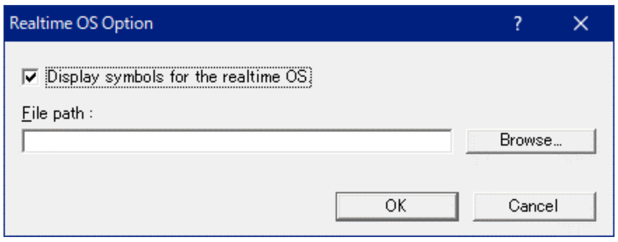


Table 5-4 Commands in the [Tools] Menu

| Command | Description | Icon and Shortcut |
|---|---|---|
| Realtime OS Option | The display of standard RTOS functions is specified. Selecting the [Display symbols for the real-time OS] checkbox in the dialog box enables the specification of an RTOS library data file.<br><br> Realtime OS Option ? ✕ Display symbols for the realtime OS File path : Browse... OK Cancel<br>An RTOS library data file is a file with the extension ".csv". Note that the Call.csv file (standard library stack information file) stored in the same directory as the Call Walker is not to be used here; do not specify this file.<br>For the RI600V4 RX-family real-time OS, the stack sizes are specified with .STACK directives in the kernel source code (assembly-language code) and the sizes will already have been reflected in the information of the Call Walker. | — |
| Merge Option | The method for merging the edit information is specified through a dialog box.<br>For details, refer to section 8, Merging Stack Information. | — |
| User Library Options | A user library data file is read, the symbols displayed by the Call Walker are replaced with the stack sizes written in the user library data file, and the display of stack information is updated.<br>Create a user library data file (*.csv), specify it in the dialog box opened by this command, and click on the [Add] button to update the stack information.<br>For details, refer to section 9, Changing the Displayed Stack Sizes by Reading a User Library Data File. | — |

## 5.5 Help Menu

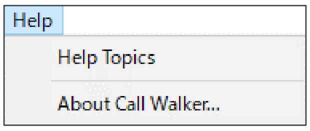


**Table 5-5 Commands in the [Help] Menu**

| Command | Description | Icon and Shortcut |
|---|---|---|
| Help Topics | The help information on the Call Walker is displayed. For details of this command, refer to the help information itself. | — |
| About Call Walker | Information of the Call Walker such as the version and copyright notice are displayed. | ? |

## 6. Stack Sizes Used by an Assembly-Language Program

Unlike a C/C++ program, automatic calculation of the stack sizes used by an assembly-language program in assembling is not possible. Therefore, an assembly-language extended function directive, .STACK, is provided to enable the Call Walker to display the stack sizes used by an assembly-language program.

Remark: The .STACK directive only enables the display of stack sizes in the Call Walker. It does not affect the operation of the program.

### 6.1 .STACK Assembly-Language Extended Function Directive

The .STACK directive defines the stack size of a specified symbol so that the Call Walker can refer to the size.

The syntax of the .STACK directive is shown below.

```
.STACK△<symbol>=<value>          △: At least one single-byte space
```

- The stack value can be defined only once per symbol; any second and later definitions for the same symbol are ignored.
- In CC-RX or CC-RH, multiples of 4 in the range from 0x0 to 0xFFFFFFFC can be specified for the stack values and definitions with any other values are ignored.
- In CC-RL, multiples of 2 in the range from 0x0 to 0xFFFE can be specified for the stack values and definitions with any other values are ignored.
- <value> must be a constant specified without using a forward reference symbol, an external reference symbol, or a relative address symbol.

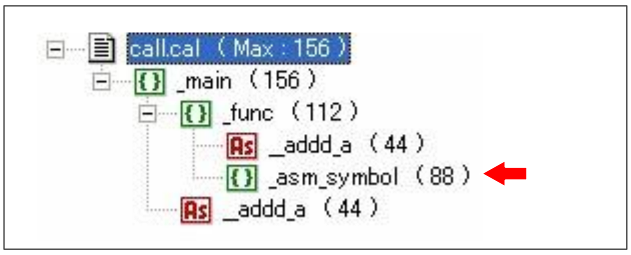
### 6.2 Example of Assembly-Language Program

The stack size for the _asm_symbol function is set to "88".

```
   .GLB _asm_symbol
   .SECTION P,CODE
_asm_symbol:
   .STACK  _asm_symbol=88
         ;
   RTS
   .END
```

Example of Display by the Call Walker

The Call Walker displays "88" as the stack size for use by the _asm_symbol function as shown below.



```
call.cal ( Max : 156 )
  _main ( 156 )
    _func ( 112 )
      _addd_a ( 44 )
      _asm_symbol ( 88 )
    _addd_a ( 44 )
```

## 7. Editing Stack Information

The Call Walker can calculate the maximum static stack size to be used but the user should edit the information file to include the maximum dynamic size to be for multiple interrupts, etc. in the calculation.

Select a symbol in the call information view on the panel in the left of the main window and use a desired command (Add, Modify, or Delete) in the [Edit] menu to edit the information (add, modify, or delete the symbol). These commands can also be selected from the menu opened by right-clicking on the symbol details view on the panel in the right of the window (for the [Radix] command in the menu, refer to section 5.3, View Menu).

The user can change the positions of symbols by dragging and dropping desired symbols in the call information view on the left panel.

Check marks appear next to the symbols moved or edited by the [Add] or [Modify] commands in the call information view (refer to (3) in section 4.2).

### 7.1 Using the [Add] Command to Add a Symbol

Symbol information can be added by either adding a new symbol or selecting from among the existing symbols.

To add a new symbol, select the [New Symbol] checkbox and enter or select the other information items such as the symbol name. The [Address] value is used as the key for managing the symbol information.

To select from among the existing symbols, deselect the [New Symbol] checkbox and select the target symbol from the [Symbol list] list box. The contents of the symbol information items cannot be modified; the existing information is reflected without change. The symbols in the [Symbol list] list box can be sorted by clicking on the header for each column (information item name) in the same way as sorting in the symbol details view of the main window.

No symbols can be added under a direct or indirect recursive call function, or a function indicated by the abbreviation icon.

An RTOS function can be added by selecting a symbol that exists in the edit information, but a new symbol cannot be added as an RTOS function. Note that when the name of a newly added symbol matches the condition for conversion to an RTOS symbol, the new symbol is displayed as an RTOS symbol.

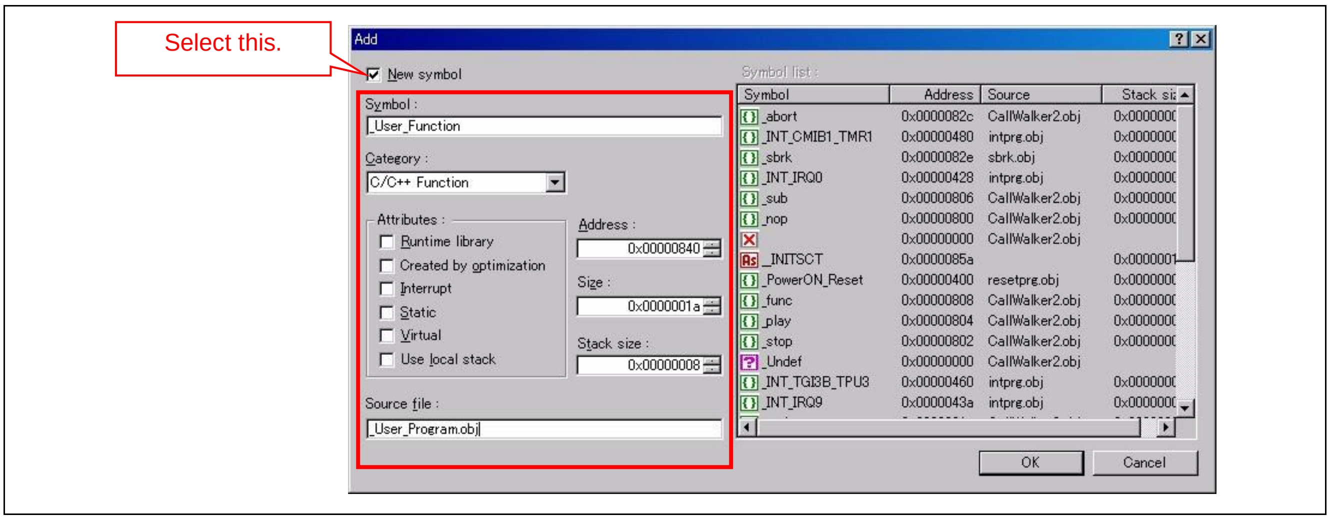
Enter or select symbol information and click on the [OK] button. The symbol information will be added immediately below the selected symbol.

If another symbol having the same name as the added symbol exists and the existing symbol has subordinate symbols, the subordinate symbols are also added. If a symbol having the same name as the selected symbol exists in a different level of the hierarchy, the same hierarchical structure as for the added symbol and its subordinate symbols is also added below the existing symbol.

Note that the initial radix for the values in the "Address", "Size", and "Stack Size" columns depends on the [Radix] command setting. A value manually entered with the prefix 0x is handled as hexadecimal and a value manually entered without a prefix is handled as decimal, regardless of the [Radix] command setting.

### (1) Adding a New Symbol

Select the [New symbol] checkbox on the left side of the dialog box opened by the [Add] command to create a new symbol in the level of the hierarchy immediately below the selected symbol. The information on the new symbol, such as the name, category, attributes, address, stack size, object file, etc. can be specified in the corresponding boxes in the dialog box.



[✓] New symbol

Symbol : _User_Function

Category : C/C++ Function

Attributes :
- [ ] Runtime library
- [ ] Created by optimization
- [ ] Interrupt
- [ ] Static
- [ ] Virtual
- [ ] Use local stack

Address : 0x00000840

Size : 0x0000001a

Stack size : 0x00000008

Source file : _User_Program.obj

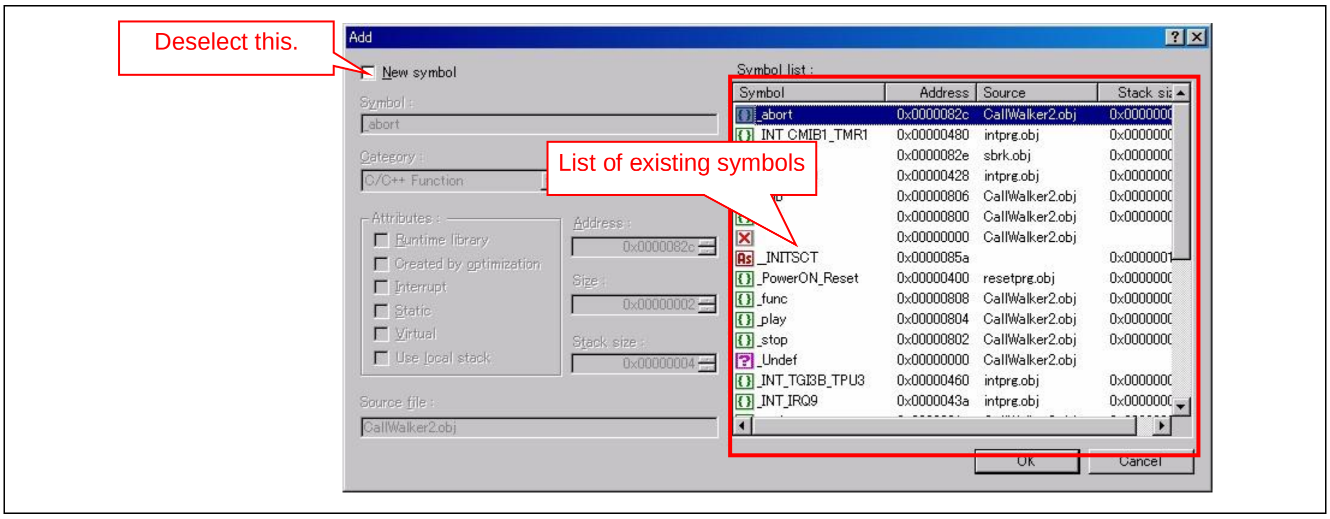
Symbol list :

| Symbol | Address | Source | Stack si |
|---|---|---|---|
| _abort | 0x0000082c | CallWalker2.obj | 0x0000000 |
| _INT_CMIB1_TMR1 | 0x00000480 | intprg.obj | 0x0000000 |
| _sbrk | 0x0000082e | sbrk.obj | 0x0000000 |
| _INT_IRQ0 | 0x00000428 | intprg.obj | 0x0000000 |
| _sub | 0x00000806 | CallWalker2.obj | 0x0000000 |
| _nop | 0x00000800 | CallWalker2.obj | 0x0000000 |
|  | 0x00000000 | CallWalker2.obj |  |
| __INITSCT | 0x0000085a |  | 0x0000001 |
| _PowerON_Reset | 0x00000400 | resetprg.obj | 0x0000000 |
| _func | 0x00000808 | CallWalker2.obj | 0x0000000 |
| _play | 0x00000804 | CallWalker2.obj | 0x0000000 |
| _stop | 0x00000802 | CallWalker2.obj | 0x0000000 |
| _Undef | 0x00000000 | CallWalker2.obj |  |
| _INT_TGI3B_TPU3 | 0x00000460 | intprg.obj | 0x0000000 |
| _INT_IRQ9 | 0x0000043a | intprg.obj | 0x0000000 |

**Figure 7-1 Dialog Box Opened by the [Add] Command (Adding a New Symbol)**

### (2) Adding an Existing Symbol

Select a symbol and click on the [Add] command in the [Edit] menu to open the following dialog box. The list on the right shows the symbols in the current file. To add an existing symbol in the level of the hierarchy immediately below the selected symbol, select the desired symbol from the list and click on the [OK] button.



[ ] New symbol

Symbol : _abort

Category : C/C++ Function

Attributes :
- [ ] Runtime library
- [ ] Created by optimization
- [ ] Interrupt
- [ ] Static
- [ ] Virtual
- [ ] Use local stack

Address : 0x0000082c

Size : 0x00000002

Stack size : 0x00000004

Source file : CallWalker2.obj

Symbol list :

| Symbol | Address | Source | Stack si |
|---|---|---|---|
| _abort | 0x0000082c | CallWalker2.obj | 0x0000000 |
| _INT_CMIB1_TMR1 | 0x00000480 | intprg.obj | 0x0000000 |
|  | 0x0000082e | sbrk.obj | 0x0000000 |
|  | 0x00000428 | intprg.obj | 0x0000000 |
|  | 0x00000806 | CallWalker2.obj | 0x0000000 |
|  | 0x00000800 | CallWalker2.obj | 0x0000000 |
|  | 0x00000000 | CallWalker2.obj |  |
| __INITSCT | 0x0000085a |  | 0x0000001 |
| _PowerON_Reset | 0x00000400 | resetprg.obj | 0x0000000 |
| _func | 0x00000808 | CallWalker2.obj | 0x0000000 |
| _play | 0x00000804 | CallWalker2.obj | 0x0000000 |
| _stop | 0x00000802 | CallWalker2.obj | 0x0000000 |
| _Undef | 0x00000000 | CallWalker2.obj |  |
| _INT_TGI3B_TPU3 | 0x00000460 | intprg.obj | 0x0000000 |
| _INT_IRQ9 | 0x0000043a | intprg.obj | 0x0000000 |

**Figure 7-2 Dialog Box Opened by the [Add] Command (Adding an Existing Symbol)**

## (3) Examples of Adding Symbols

Symbols are added as shown in the example given as Figure 7-3.

(1) sub3 is added as a symbol to be called from sub1.
(2) As sub4 is present as a symbol to be called from sub3, sub4 is also added under the new sub3.
(3) As sub1 is present as a symbol to be called from sub2, sub3 and sub4 are also added under sub1.

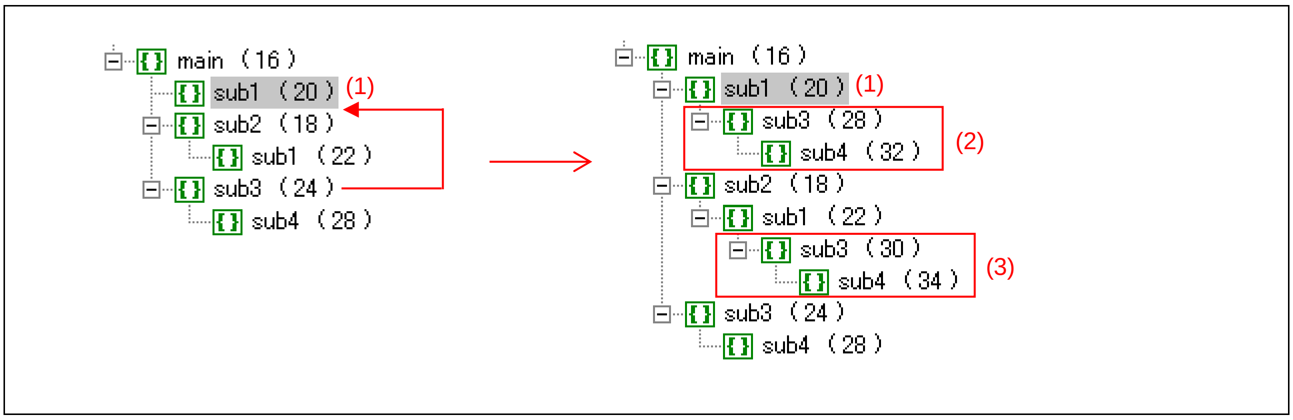
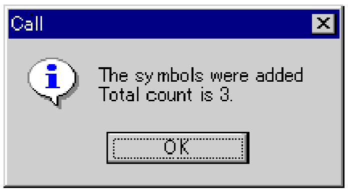


**Figure 7-3 Example of Adding Symbols**

If the specified symbol is added to multiple locations in addition to the specified location as in the example of Figure 7-3, the following information message will be displayed to indicate that the symbol has been added to multiple locations.


Call

The symbols were added
Total count is 3.

**Figure 7-4 Message Indicating Symbols Added to Multiple Locations**

[Total count] only indicates the total number of locations of the symbol specified by the [Add] command. The subordinate added symbols are not counted. When the call information view is displayed in the [Show Simple Symbols] mode, the symbols omitted from the view are not counted.

Even a new symbol having the same address as that of an existing symbol can be added. If a new symbol is the same as an existing symbol in terms of address, name, and category but different in stack size, a message will appear to confirm whether to change the icon to that for a variadic function. Click on [OK] to change it to a variadic symbol, after which the new symbol inherits all information from the existing symbol except for the stack size.

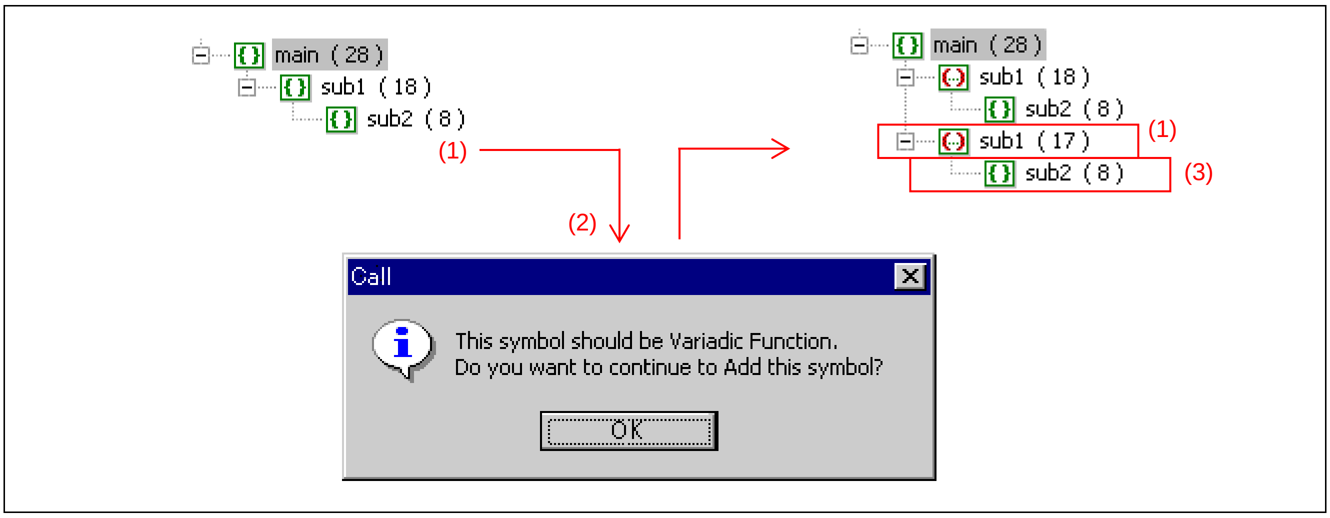
Such symbols are added in the way shown in the example given as Figure 7-5.

(1) sub1 is added as a symbol to be called from the main function.

(2) The new symbol is the same as the existing sub1 symbol in terms of address, name, and category but has a different stack size. A message will appear asking you to confirm whether to change it to a variadic function symbol.

(3) As sub2 (existing symbol) is a symbol to be called from sub1, sub2 is also added as a symbol to be called from the new symbol.

This only applies when the conditions for changing to a variadic function symbol (same address, name, and category but different stack size) are satisfied.



This symbol should be Variadic Function.
Do you want to continue to Add this symbol?

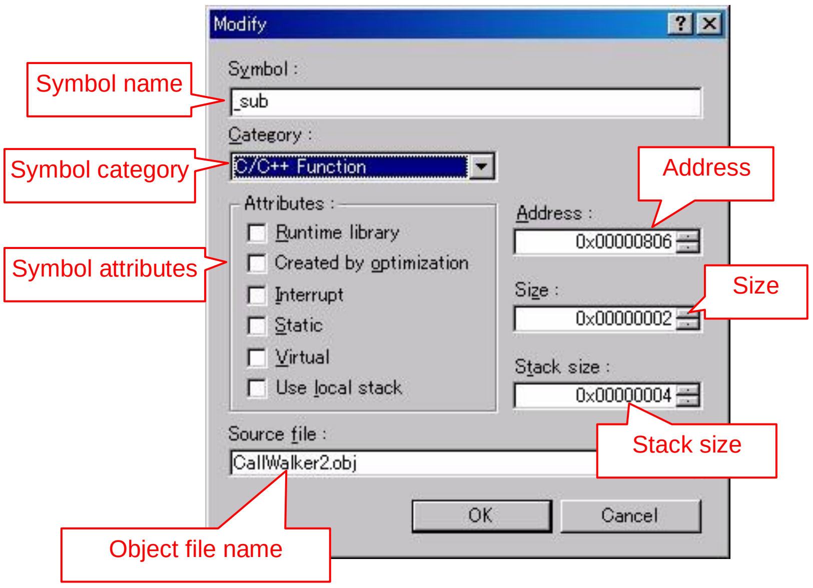
**Figure 7-5 Example of Adding a Symbol Having the Same Address as that of an Existing Symbol**

## 7.2 Using the [Modify] Command to Modify Symbol Information

### (1) Modifying the Information on a Selected Symbol

Select a desired symbol for modification and click on the [Modify] command in the [Edit] menu. The following dialog box appears. The user can modify the information items in the dialog box.

Note that the [Modify] command cannot be used on an RTOS symbol.



Symbol : _sub
Category : C/C++ Function
Attributes :
- ☐ Runtime library
- ☐ Created by optimization
- ☐ Interrupt
- ☐ Static
- ☐ Virtual
- ☐ Use local stack

Address : 0x00000806
Size : 0x00000002
Stack size : 0x00000004
Source file : CallWalker2.obj

**Figure 7-6 Dialog Box Opened by the [Modify] Command**

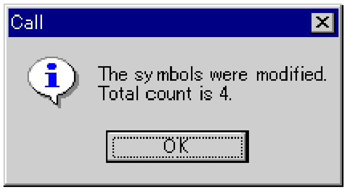
Enter or select information items in the dialog box and click on the [OK] button. The symbol information will be modified to the entered or selected information.

The [Address] value is used as the key for managing the symbol information.

Note that the initial radix for the values in the "Address", "Size", and "Stack Size" columns depends on the [Radix] command setting. A value manually entered with the prefix 0x is handled as hexadecimal and a value manually entered without a prefix is handled as decimal, regardless of the [Radix] command setting.

This command cannot be used to change the symbol category to an RTOS function. However, if the symbol name after modification matches that of an RTOS symbol, the modified symbol is displayed as an RTOS symbol.

If the modified symbol is present at multiple locations in addition to the specified location, the modifications to the symbol will apply at all locations. In this case, the following information message will be displayed to indicate that the modification has been applied to the symbol at multiple locations.



The symbols were modified.
Total count is 4.

**Figure 7-7 Message Indicating the Symbol Modification Applied to Multiple Locations**

[Total count] only indicates the total number of locations of the symbol specified by the [Modify] command. The subordinate modified symbols are not counted. When the call information view is displayed in the [Show Simple Symbols] mode, the symbols omitted from the view are not counted.

## (2) Rules for Modifying Symbols

The following rules apply to the modification of symbols.

### (a) When a variadic function symbol is selected

Only modifications of both the category and address of a variadic function symbol apply to the symbol in the same way as in modification of a non-variadic function symbol.

When modification of information other than the symbol category is specified, the following rules apply to modification of the symbol.

- If the new address after modification is the same as that of another symbol, an error will occur.
- If the address, name, source file, size, or attribute is modified, the modification applies to all symbols in the group of variadic function symbols (the variadic function symbols having the same name and address).
- If the new stack size after modification is the same as that for another symbol in the group of variadic function symbols, an error will occur.

The following shows an example of modifying the information of variadic function symbols.

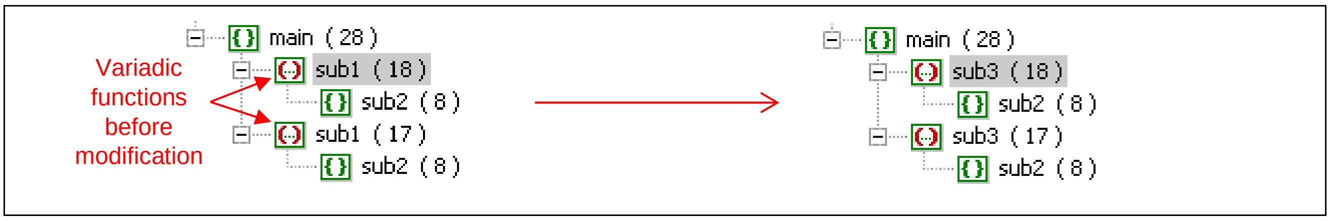


**Figure 7-8 Example of Modifying the Information of Variadic Function Symbols**

When the symbol name sub1 (first symbol) is changed to sub3, the second sub1 symbol is also changed to sub3 because both symbols belong to the same group of variadic function symbols.

### (b) When a non-variadic function symbol is selected

When the new address after modification differs from that of the other symbols, the information of the selected symbol is modified as specified.

If the new address matches that of another symbol, the following rules apply to modification of the symbol.

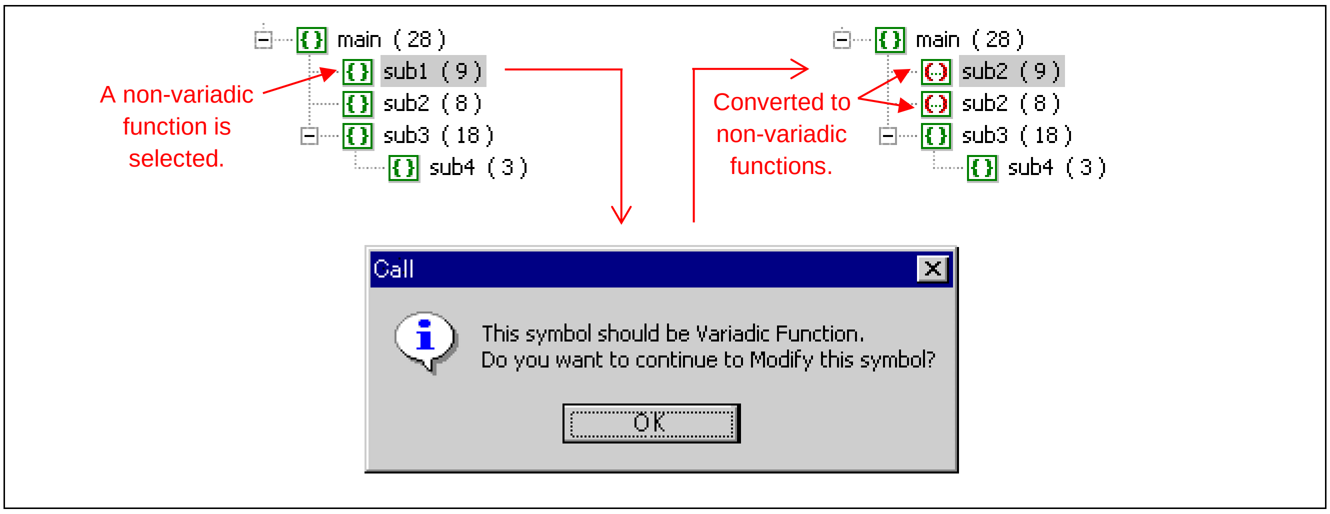
- If the selected symbol or an existing symbol having the same address and name as those specified in the [Modify] dialog box calls another symbol, an error will occur.
- If the modified symbol is the same as an existing symbol in terms of the [Symbol], [Address], and [Category] setting in the [Modify] dialog box but different in the [Stack size] setting, a message will appear to confirm whether the symbols should be changed to variadic function symbols. Click on [OK] to change them to variadic function symbols, and the modified symbols inherit all information entered in the [Modify] dialog box except for the [Stack size] setting.

The following shows an example of modifying the information of non-variadic function symbols.

Symbols are modified as follows in the example shown in Figure 7-9.

(1) sub1 is modified to sub2. The settings in the [Modify] dialog box are the same as those for an existing symbol (sub2) in terms of [Symbol], [Address], and [Category] but different for [Stack size]. Here, we assume that neither sub1 nor sub2 call other symbols.
(2) A message will appear to ask for confirmation of whether to change the symbols to variadic function symbols.
(3) Clicking on [OK] modifies the information of sub2 that was changed from sub1 to the settings in the [Modify] dialog box except for [Stack size], which is kept at the value for the symbol before modification.

This only applies when the conditions for changing to a variadic function symbol (same address, name, and category but different stack size) are satisfied.



This symbol should be Variadic Function.
Do you want to continue to Modify this symbol?

**Figure 7-9 Example of Modifying the Information of Non-Variadic Function Symbols**

## 8. Merging Stack Information

After the information from editing with the use of the Call Walker has been saved, the stack information following editing can be merged with another stack information file.

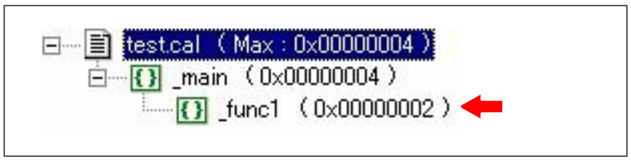
This facility can be used to prevent overwriting of the edited stack information by the new stack information created by re-building.

### 8.1 Sample Procedure for Merging

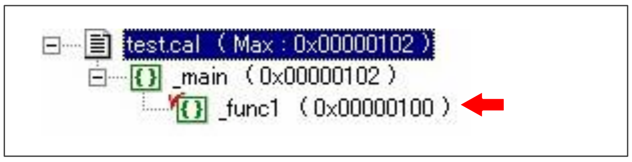
(1) Contents of test.c

```
void  main(void)
{
    func1();
}
```

(2) Build test.c and open the stack information in the Call Walker.


test.cal ( Max : 0x00000004 )
_main ( 0x00000004 )
_func1 ( 0x00000002 )

(3) Modify the contents (change the stack size for func1 to 0x00000100).

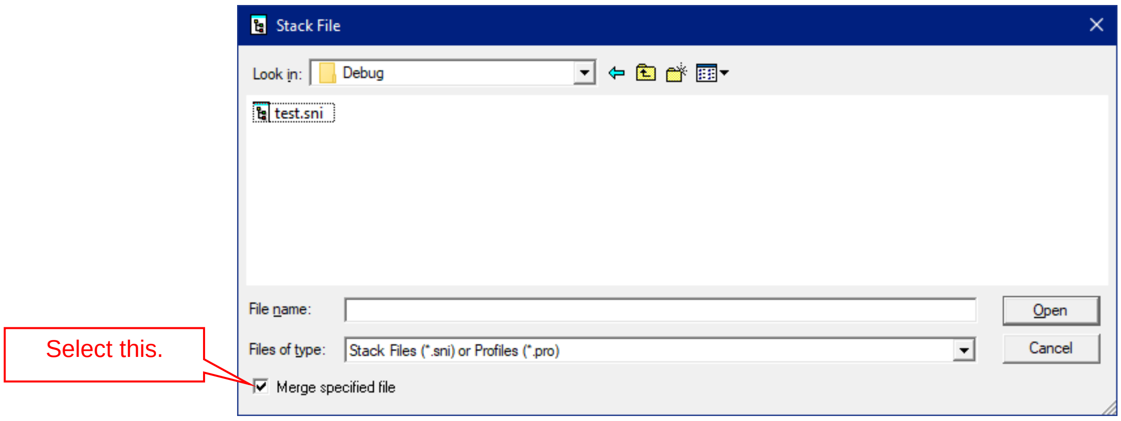

test.cal ( Max : 0x00000102 )
_main ( 0x00000102 )
_func1 ( 0x00000100 )

(4) Modify the contents of test.c (add a call of func2).

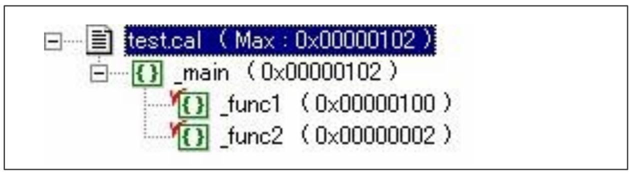
```
void  main(void)
{
    func1();
    func2();
}
```

(5) Re-build test.c with the Call Walker remaining open.

(6) Import test.sni through the Call Walker session that is still open. Here, select the [Merge specified file] checkbox and click on the [Open] button.

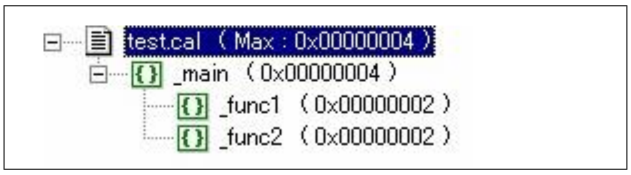

Stack File
Look in: Debug
test.sni
File name:
Files of type: Stack Files (*.sni) or Profiles (*.pro)
[✓] Merge specified file

(7) As a result of importing of the file in step (6), the information of func2 is added but the stack size of func1 is kept at the value to which it was changed in step (3). Stack information is merged in this way.



```
test.cal  ( Max : 0x00000102 )
  _main  ( 0x00000102 )
    _func1  ( 0x00000100 )
    _func2  ( 0x00000002 )
```

If [Merge specified file] is not selected in step (6), the stack size of func1, which has been changed in step (3), will be restored to the value before modification as shown below.

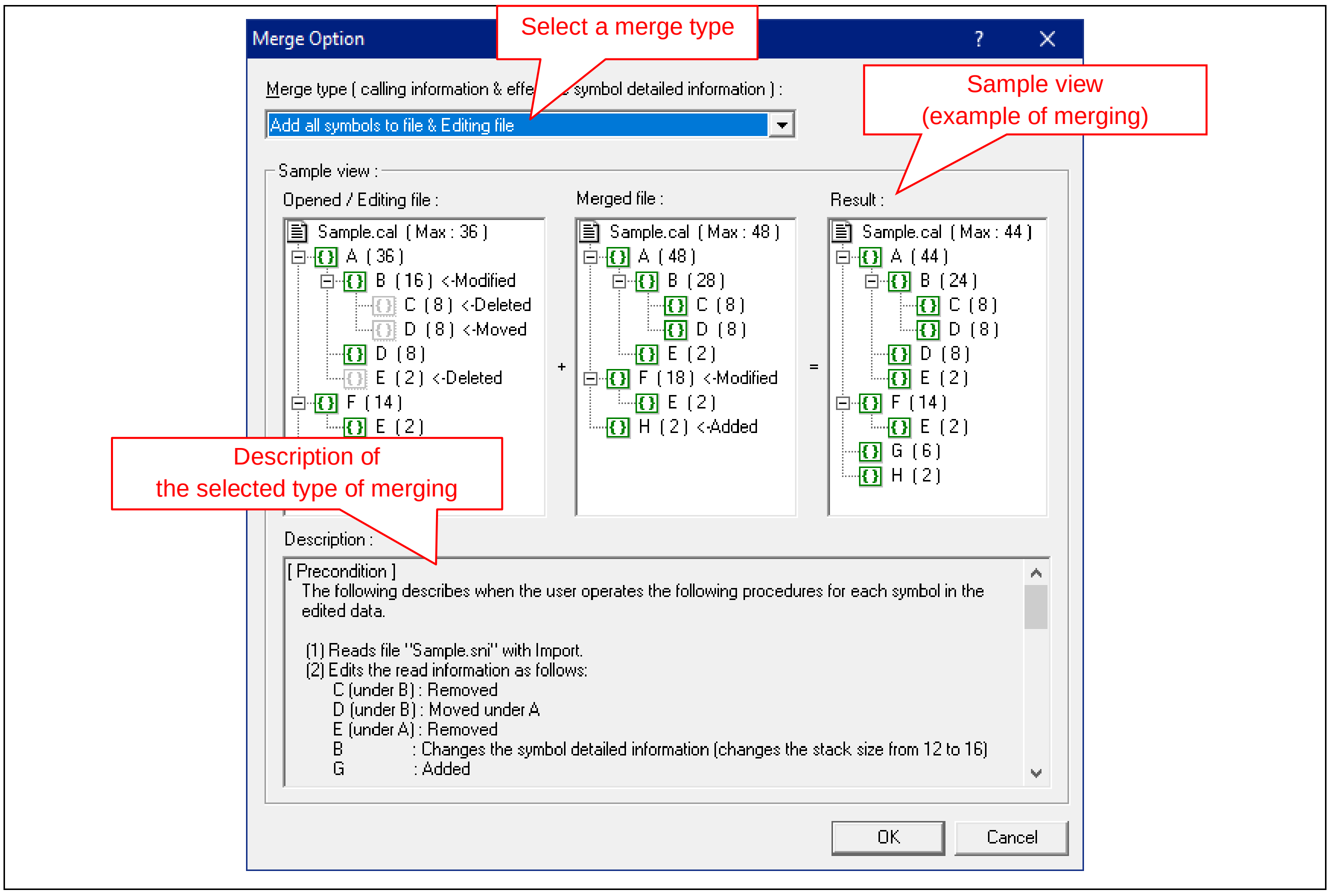


```
test.cal  ( Max : 0x00000004 )
  _main  ( 0x00000004 )
    _func1  ( 0x00000002 )
    _func2  ( 0x00000002 )
```

## 8.2 Selecting the Type of Merging

There are five types (methods) of merging, which can be changed through the [Merge Option] dialog box. Select [Tools] → [Merge Option] to open this dialog box.

When a merge type is selected in the [Merge type] list box of the dialog box, an example of merging with the selected merge type is displayed in the [Sample view] and the details of the method of merging and a description of the example of merging are displayed under [Description].



Merge type ( calling information & effe... symbol detailed information ) :
Add all symbols to file & Editing file

Sample view :

```
Opened / Editing file :          Merged file :                    Result :
Sample.cal ( Max : 36 )          Sample.cal ( Max : 48 )          Sample.cal ( Max : 44 )
  A ( 36 )                         A ( 48 )                         A ( 44 )
    B ( 16 ) <-Modified              B ( 28 )                         B ( 24 )
      C ( 8 ) <-Deleted                C ( 8 )                          C ( 8 )
      D ( 8 ) <-Moved                  D ( 8 )                          D ( 8 )
    D ( 8 )                   +      E ( 2 )                   =      D ( 8 )
    E ( 2 ) <-Deleted              F ( 18 ) <-Modified                E ( 2 )
  F ( 14 )                           E ( 2 )                        F ( 14 )
    E ( 2 )                        H ( 2 ) <-Added                    E ( 2 )
                                                                    G ( 6 )
                                                                    H ( 2 )
```

Description :

```
[ Precondition ]
  The following describes when the user operates the following procedures for each symbol in the
  edited data.

  (1) Reads file "Sample.sni" with Import.
  (2) Edits the read information as follows:
       C (under B) : Removed
       D (under B) : Moved under A
       E (under A) : Removed
       B           : Changes the symbol detailed information (changes the stack size from 12 to 16)
       G           : Added
```

**Figure 8-1 [Merge Option] Dialog Box**

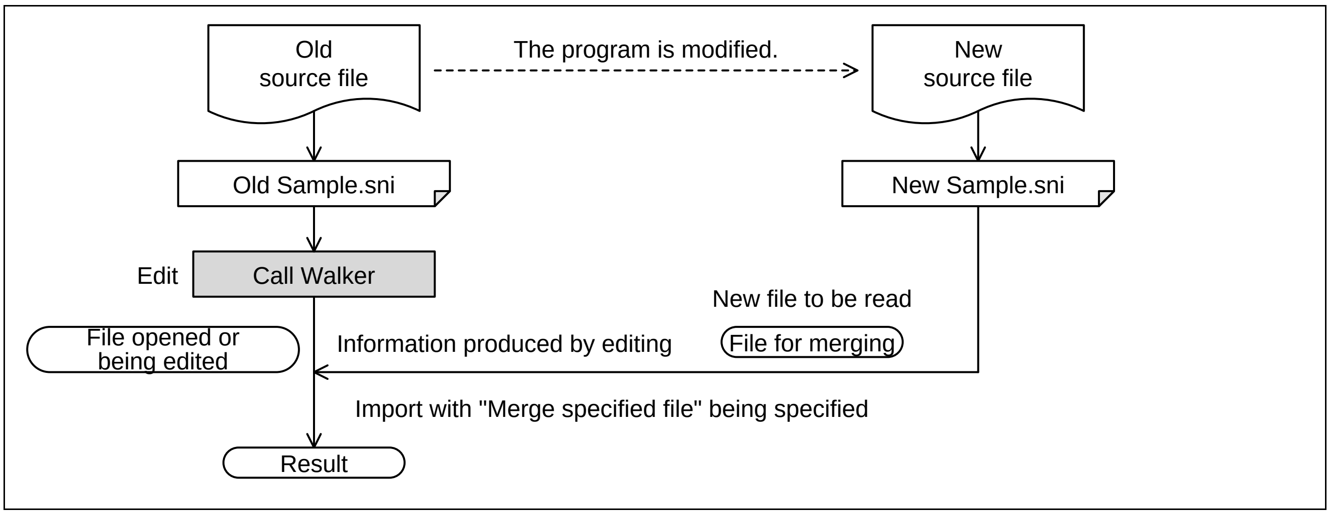


**Figure 8-2 Supplementary Figure: Processing of the Sample View (Preconditions)**

Merge types:

(1) Not merge calling information & Merged file
(2) Add all symbols to file & Editing file
(3) Add all symbols to file & Merged file
(4) Add only new symbols to file & Editing file
(5) Add only new symbols to file & Merged file

**Table 8-1 Overview of Processing for Merging with Each Merge Type**

| Merge Type | Call Information | Symbol Details |
|---|---|---|
| (1) | Not merge calling information:<br>• The current information produced by editing is retained (the call information in the newly read file is not merged). | Merged file:<br>• The detailed information for the symbols in the newly read file becomes valid (the detailed information that was modified for the symbols in the current information produced by editing is lost). |
| (2) | Add all symbols to file:<br>• The symbols that exist in the newly read file but not in the current information produced by editing are added.<br>• Any deleted symbols are added again.<br>• The symbols that were moved to different locations are added again at their original locations. The symbols at the locations to which they were moved also remain. | Editing file:<br>• The current information produced by editing remains valid and the detailed information of the symbols in the newly read file is ignored.<br>• Note that only for the new symbols added from the new file, the contents of the new file are reflected to the current information. |
| (3) | Add all symbols to file:<br>Same as in (2) | Merged file:<br>Same as in (1) |
| (4) | Add only new symbols to file:<br>• The symbols that exist in the newly read file but not in the current information produced by editing are added.<br>• Any deleted symbols are not added.<br>• The symbols that were moved to different locations are not added again at their original locations. | Editing file:<br>Same as in (2) |
| (5) | Add only new symbols to file:<br>Same as in (4) | Merged file:<br>Same as in (1) |

## 9. Changing the Displayed Stack Sizes by Reading a User Library Data File

The [User Library Options] command in the [Tools] menu can be used to read a user library data file and replace the display of the stack sizes for the symbols shown on the Call Walker window with the stack sizes written in the user library data file.

The format of a user library data file is shown below.

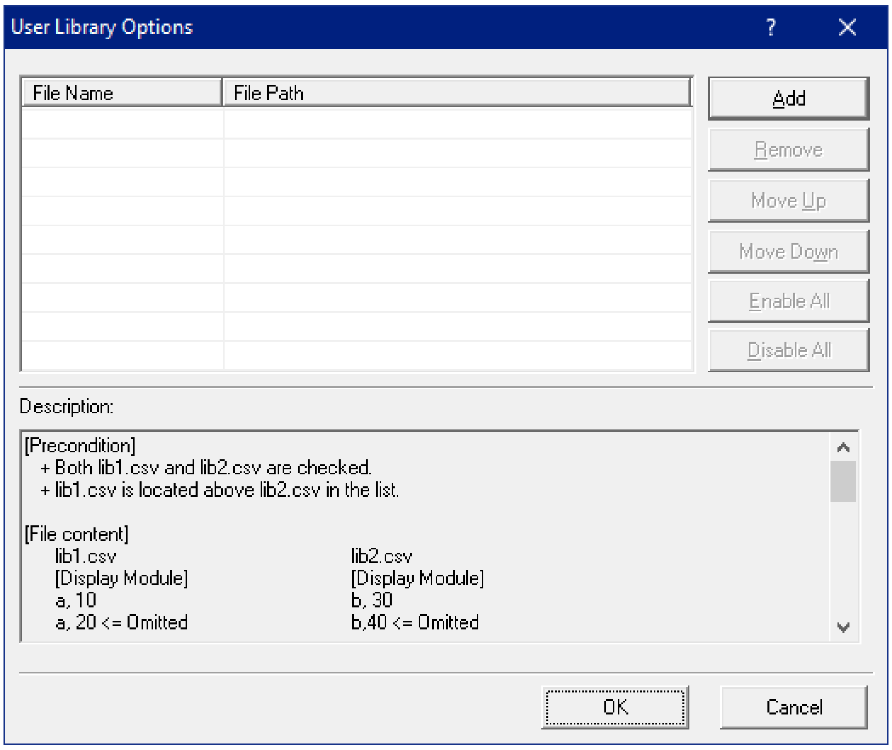
```
[Display Module]
Symbol name for display, stack size
[Non Display Module]
Symbol name to be hidden
```

The following shows an example of specifications in a file (use a text editor to create the file and store it with a file name of the form *.csv).

```
[Display Module]
_sub1, 500
[Non Display Module]
_sub2
```

The created user library data file (*.csv) should be selected in the dialog box opened by the [User Library Options] command for updating the stack information.

Specifically, select the [User Library Options] command from the [Tools] menu to open the following dialog box.



```
[Precondition]
  + Both lib1.csv and lib2.csv are checked.
  + lib1.csv is located above lib2.csv in the list.

[File content]
    lib1.csv                     lib2.csv
    [Display Module]             [Display Module]
    a, 10                        b, 30
    a, 20 <= Omitted             b,40 <= Omitted
```

**Figure 9-1 Dialog Box Opened by the [User Library Options] Command**

Click on the [Add] button to specify a user library data file.

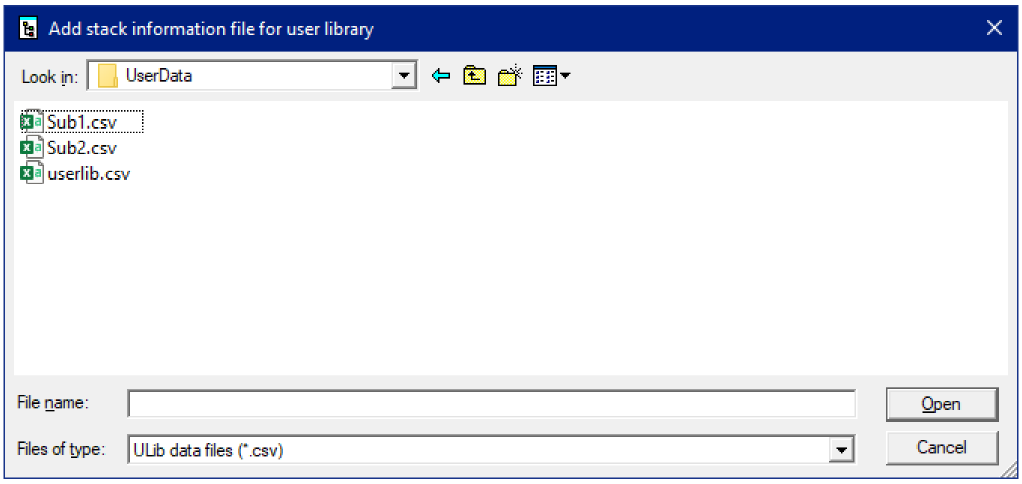
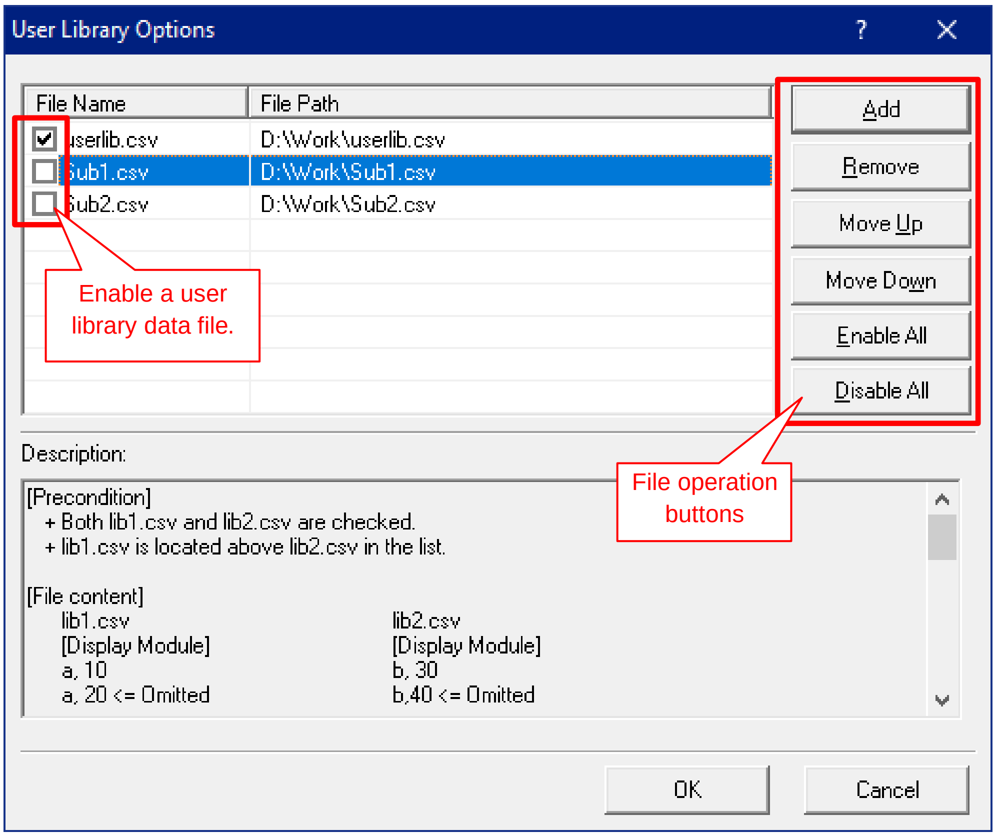

Add stack information file for user library
Look in: UserData
Sub1.csv
Sub2.csv
userlib.csv
File name:
Files of type: ULib data files (*.csv)
Open
Cancel

**Figure 9-2 Adding a User Library Data File**

To enable a user library data file, select the checkbox for the file. Multiple user library data files can be selected and enabled at the same time.



| | File Name | File Path |
|---|---|---|
| ✓ | userlib.csv | D:\Work\userlib.csv |
| ☐ | Sub1.csv | D:\Work\Sub1.csv |
| ☐ | Sub2.csv | D:\Work\Sub2.csv |

```
[Precondition]
  + Both lib1.csv and lib2.csv are checked.
  + lib1.csv is located above lib2.csv in the list.

[File content]
    lib1.csv                    lib2.csv
    [Display Module]            [Display Module]
    a, 10                       b, 30
    a, 20 <= Omitted            b,40 <= Omitted
```

**Figure 9-3 Operation of the [User Library Options] Dialog Box**

**Table 9-1 File Operation Buttons in the [User Library Options] Dialog Box**

| Button | Operation |
|---|---|
| Remove | The selected user library data file is removed from the list. |
| Move Up | The selected file is moved upward in the list. |
| Move Down | The selected file is moved downward in the list. |
| Enable All | All user library files are enabled. |
| Disable All | All user library files are disabled. |

Click on the [OK] button to replace and update the current display of the stack sizes for the symbols with those specified in each file that has been read.

The following shows an example of replacing and hiding stack sizes.

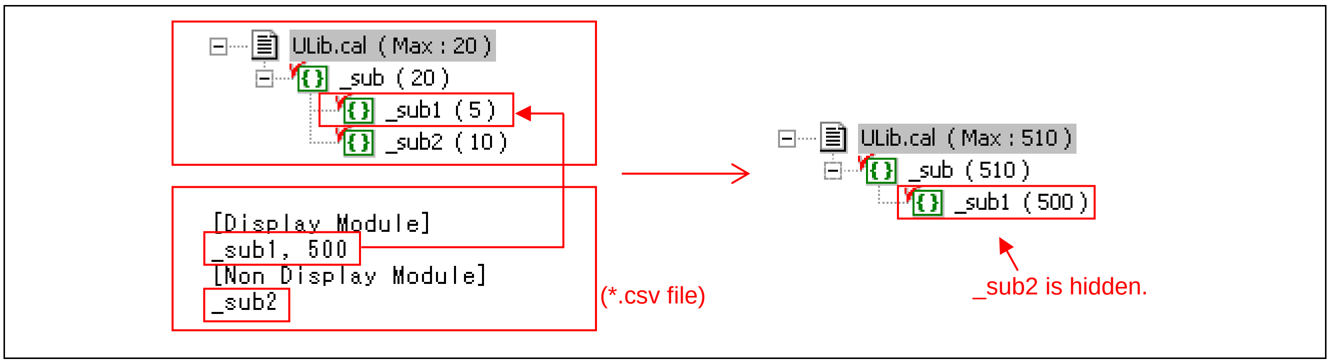

```
ULib.cal ( Max : 20 )
  _sub ( 20 )
    _sub1 ( 5 )
    _sub2 ( 10 )
```
```
[Display Module]
_sub1, 500
[Non Display Module]
_sub2
```
```
ULib.cal ( Max : 510 )
  _sub ( 510 )
    _sub1 ( 500 )
```

**Figure 9-4 Example of Replacing and Hiding Stack Sizes**

## Appendix 1 Information Messages

(1/2)

| | | |
|---|---|---|
| 1 | Message | The symbols were added.<br>Total count is <number of added symbols>. |
| | Meaning | Symbols were also added to locations other than the location specified for the [Add] command. |
| | Description | When a symbol is added by using the [Add] command, if the symbol selected as the location for adding the new symbol is also found at another location, the new symbol is also added at the other location where it was found. This message is output when the symbol was added to locations other than the specified location. |
| | Behavior of Tool | The tool continues operating. |
| | Action by User | None. |
| 2 | Message | The symbols were modified.<br>Total count is <number of modified symbols>. |
| | Meaning | Symbols were also modified at locations other than that specified for the [Modify] command. |
| | Description | When a symbol is modified by using the [Modify] command, if the specified symbol is also found at another location, the symbol is also modified at the other location where it was found. This message is output when the symbol was modified at locations other than the specified location. |
| | Behavior of Tool | The tool continues operating. |
| | Action by User | None. |
| 3 | Message | The symbols were removed.<br>Total count is <number of deleted symbols>. |
| | Meaning | Symbols were also deleted at locations other than that specified for the [Delete] command. |
| | Description | When a symbol is deleted by using the [Delete] command, if a symbol that refers to the symbol to be deleted at the specified location is also found at another location and the symbol found at the other location also refers to the symbol to be deleted, the symbol is also deleted at the other location where it was found. This message is output when the symbol was deleted at locations other than the specified location. |
| | Behavior of Tool | The tool continues operating. |
| | Action by User | None. |
| 4 | Message | Call Walker has finished searching a symbol |
| | Meaning | The search processing has been completed. (Find Option: [The pass of maximum stack size]) |
| | Description | This message is output when searching through all symbols has been completed. The symbol with the minimum stack size (Find Next) or the maximum stack size (Find Previous) is selected in the view. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | None. |
| 5 | Message | Cannot find the symbol <Symbol name>. |
| | Meaning | The symbol was not found. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check the target symbol name and correct it. |
| 6 | Message | The number of items is too large. Cannot display all. |
| | Meaning | All items cannot be displayed because the number of items is too large. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check the display option. If [Show All Symbols] is selected, switch to [Show Simple Symbol]. If [Show All Symbols] is not selected, use the [Output List] command to see all symbol information. |

(2/2)

| | | |
|---|---|---|
| 7 | Message | This symbol should be Variadic Function.<br>Do you want to continue to [Add/Modify] this symbol? |
| | Meaning | This message is output when the specified symbol can be changed to a variadic function. |
| | Behavior of Tool | The behavior depends on the action by the user. |
| | Action by User | Select [OK] to change the symbol to a variadic function.<br>Select [Cancel] to terminate the processing with a warning message. |

## Appendix 2 Error Messages

(1/3)

| | | |
|---|---|---|
| 1 | Message | Cannot open a file.<br>'<Error file>' |
| | Meaning | The file indicated as <Error file> cannot be opened. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Specify a correct file name.<br>Check for any problems with the target storage device. |
| 2 | Message | Cannot read a file.<br>'<Error file>' |
| | Meaning | An error occurred during the processing for reading of the file indicated as <Error file>. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check if the file is damaged.<br>Check for any problems with the target storage device. |
| 3 | Message | Cannot write a file.<br>'<Error file>' |
| | Meaning | An error occurred during the processing for writing the file indicated as <Error file>. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check the location and privilege of writing.<br>Check if the specified location is write-protected. |
| 4 | Message | Cannot close a file.<br>'<Error file>' |
| | Meaning | An error occurred during the processing for closing the file indicated as <Error file>. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check for any problems with the memory or other storage. |
| 5 | Message | The version of file which is not supported was specified.<br>'<Error file>' |
| | Meaning | The version of the file indicated as <Error file> is not supported. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Specify the name of a file from a supported version. |
| 6 | Message | The contents of the specified file are wrong.<br>'<Error file>' |
| | Meaning | The contents of the file indicated as <Error file> are incorrect. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Specify the name of a file with correct contents. |
| 7 | Message | Not enough memory. |
| | Meaning | The memory does not have enough available space. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Terminate other applications to increase the available space in the memory. |

(2/3)

| | | |
|---|---|---|
| 8 | Message | Cannot move here ... <Detail Message>. |
| | | <Detail Message><br>• Same symbol exists on destination symbol:<br>A symbol with the same name already exists at the destination.<br>• Move to omitted symbol:<br>A symbol with an abbreviation icon was specified as the destination.<br>• Move to recursive symbol:<br>A direct or indirect recursive call was specified as the destination.<br>• Move to lower symbol:<br>A level below the source symbol was specified as the destination.<br>• Move to own symbol:<br>A symbol that directly refers to the target symbol (and has a fixed location) was specified as the destination.<br>• Move to symbol without name:<br>A symbol without a name and with an unresolved address for reference was specified as the destination. |
| | Meaning | Dragging and dropping a symbol to a location to which it cannot be moved was attempted. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | The symbol cannot be moved to the location shown in <Detail Message>. Check the location and specify a correct location to which the symbol can be moved. |
| 9 | Message | Failed to input ... <Detail Message>. |
| | | <Detail Message><br>• Illegal value input on Address/Size/Stack Size/Expansion Level:<br>An illegal value was specified for "Address", "Size", "Stack size", or "Expansion Level".<br>• Symbol name was not selected: No symbol name was selected.<br>• Symbol name was not input: No symbol name was entered.<br>• A symbol of same address already exists:<br>A symbol has already been registered at the specified address.<br>• Specified symbol already exists:<br>The specified symbol already exists at the same level of the hierarchy.<br>• A filename cannot contain any of the following characters (¥ / : ; * ? " < > \|):<br>A character that cannot be used for a file name was specified as part of "Source File".<br>• Symbol name is not input or selected:<br>No symbol name was entered or selected.<br>• A symbol name cannot contain character (\|):<br>A character that cannot be used for a symbol name was used. |
| | Meaning | The [Add], [Modify], or [Expand] command cannot be executed because an illegal value was specified in the dialog box of the command. |
| | Behavior of Tool | The tool aborts the processing and re-displays the dialog box so that the user can enter values again. |
| | Action by User | Check the specified values and correct them.<br>For "Address", "Size", and "Stack size", a value without a prefix is handled as a decimal value and a value with the prefix 0x is handled as a hexadecimal value. The allowable values are 0 (0x00000000) to 4294967295 (0xffffffff).<br>For "Expansion Level", only a decimal value in the range from 1 to 32 can be specified and no other characters are specifiable.<br>An error will occur if any of the characters that cannot be used in a file name (¥ / : ; * ? < > \| ) is specified as part of "Source File". |

(3/3)

| | | |
|---|---|---|
| 10 | Message | Failed to input ... Symbol name is not input or selected |
| | Meaning | No symbol name was selected or entered in the [Find] dialog box. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | To search for a symbol name, select the symbol name from the [Symbol] combo box or enter the whole or part of the symbol name. |
| 11 | Message | Shell version is too old. |
| | Meaning | This tool cannot be executed because the version of the shell is old. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check the environment for executing this tool. |
| 12 | Message | Illegal file format<br>'<Error File>' |
| | Meaning | The file specified for opening is not in the correct format of a call information file for use by this tool. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check that a correct file is specified.<br>This message is also output when a call information file created in Version 1.0 or Version 1.1 Release 1 of the Call Walker was specified. |
| 13 | Message | Illegal input file name specified.<br>Illegal output file name specified. |
| | Meaning | No input or output file was specified. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | When the executable file of the Call Walker was directly specified and started from the Command Prompt of Windows, check if only the –list option was specified or a null string was specified as the file name for the –import option. |
| 14 | Message | Internal error has occurred.<br>'<Detail Message>' |
| | Meaning | An internal error occurred in the tool. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Restart the tool. If the error occurs again, install the tool again. |
| 15 | Message | Syntax error. |
| | Meaning | A syntax error was found in the specification at the Command Prompt. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | When the executable file of the Call Walker was directly specified and started from the Command Prompt of Windows, check if multiple input file names or output file names were specified.<br>Also check if any option other than –import or –list was specified. |
| 16 | Message | File already exists<br>'<Path of file>' |
| | Meaning | The specified file cannot be added because it already exists in the file list. |
| | Behavior of Tool | The tool terminates the processing. |
| | Action by User | Check the name of the file to be added. If the file path is wrong, select a correct file name. |
| 17 | Message | File does not exist<br>or File has illegal format in User Library Options... |
| | Meaning | The specified file is not shown in the [User Library Options] dialog box or the format of the file is wrong. |
| | Behavior of Tool | The tool does not select this file. |
| | Action by User | None. This file is automatically deselected by the tool. |

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Sep. 30, 2022 | — | First edition issued (Guide for CS+). |

## General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.