To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

User's Manual

RX850 Pro Ver. 3.20

Real-Time Operating System

Installation

Target Tool
RX850 Pro Ver.3.20

Document No. U17421EJ1V0UM00 (1st edition)
Date Published April 2005 CP(K)

[MEMO]

MS-DOS, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Green Hills Software and MULTI are trademarks of Green Hills Software, Inc.

UNIX is a trademark of X/Open Company, Ltd. licensed in the USA and other countries.

PC/AT is a trademark of IBM Corporation.

TRON is an abbreviation for The Real-time Operating system Nucleus.

ITRON is an abbreviation for Industrial TRON.

μITRON is an abbreviation for Micro Industrial TRON.

- **The information in this document is current as of April, 2005. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, pIease contact the NEC Electronics office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

[GLOBAL SUPPORT]
http://www.necel.com/en/support/support.html

NEC Electronics America, Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65030

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 80 820
- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-558-3737

NEC Electronics Shanghai Ltd.
Shanghai, P.R. China
Tel: 021-5888-5400

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 6253-8311

J04.1

[MEMO]

INTRODUCTION

Readers

This manual is intended for users who design and develop application systems using V850 Series.

Purpose

This manual explains the functions of the RX850 Pro.

Organization

This manual consists of the following major sections.

- Overview
- Installation
- System construction
- Interface libraries
- Memory and memory capacity estimation
- System configuration file
- Configurator (CF850 Pro)

How to read this manual

It is assumed that the readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, microcontrollers, C language, and assemblers.

To understand the hardware functions of the V850 Series
→ Refer to the **User's Manual Hardware** of each product.

To understand the instruction functions of the V850 Series
→ Refer to the **V850ES Architecture User's Manual (U15943E) or V850E1 Architecture User's Manual (U14559E)**.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Note:	Footnote for item marked with **Note** in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	Binary...XXXX or XXXXB
	Decimal...XXXX
	Hexadecimal...0xXXXX

Prefixes indicating power of 2 (address space and memory capacity):

K (kilo)	$2^{10} = 1024$
M (mega)	$2^{20} = 1024^2$

Related Documents Read this manual together with the following documents.
The related documents indicated in this publication may include preliminary versions.
However, preliminary versions are not marked as such.

Documents related to development tools (user's manuals)

Document Name		Document Number
CA850 Ver. 3.00 C Compiler Package	Operation	U17293E
	C Language	U17291E
	Assembly Language	U17292E
	Link Directives	U17294E
ID850 Ver. 3.00 Integrated Debugger	Operation	U17358E
ID850NW Ver. 3.00, 3.10 Integrated Debugger	Operation	U17369E
ID850NWC Ver. 2.51 Integrated Debugger	Operation	U16525E
ID850QB Ver. 3.10 Integrated Debugger	Operation	U17435E
SM+ System Simulator	Operation	U17246E
	User Open Interface	U17247E
SM850 Ver. 2.50 System Simulator	Operation	U16218E
SM850 Ver. 2.00 or later System Simulator	External Part User Open Interface Specifications	U14873E
RX850 Pro Ver. 3.20 Real-Time OS	Basics	U13773E
	Installation	This manual
	Technical	U13772E
	Task Debugger	U17422E
AZ850 Ver. 3.30 System Performance Analyzer		U17423E
PG-FP4 Flash Memory Programmer		U15260E
TW850 Ver. 2.00 Performance Analysis Tuning Tool		U17421E
PM+ Ver. 6.00 Project Manager		U17178E

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 OVERVIEW

1.1 Outline

Rapid advances in semiconductor technologies have led to the explosive spread of microprocessors such that they are now to be found in more fields than many would have imagined only a few years ago. In line with this spread, the number of processing programs that must be created for microprocessors is also increasing. This rule of growth makes it difficult to create processing programs specific to given hardware.

For this reason, there is a need for operating systems (OSs) that can fully exploit the capabilities of the latest generation of ever-newer high-performance, multi-function microprocessors.

Operating systems are broadly classified into 2 types : program-development OSs and control OSs. Program-development OSs are to be found in those environments in which standard OSs (e.g., MS-DOS™, Windows, and UNIX™ OS) predominate because the hardware configuration to be used for development can be limited to some extent (e.g., personal computers).

Conversely, control OSs are incorporated into control units. That is, these OSs are found in those environments where standard OSs cannot easily be applied because the hardware configuration varies from system to system and because efficient operation matching the application is required.

To satisfy these demands, NEC Electronics has developed and released not only the V850 Series of microcontrollers but also the RX850 Pro operating system, which allows users to fully exploit the functions of these microcontrollers and support systematic software creation.

The RX850 Pro is a control OS for real-time, multitasking processing; it has been developed to increase the application range of high-performance, multi-function microprocessors and further improve their versatility.

The RX850 Pro is a built-in real-time, multitasking control OS that provides a highly efficient real-time, multitasking environment to increase the application range of processor control units.

The RX850 Pro is a high-speed, compact OS capable of being stored in and run from the ROM of a target system.

1.2 Features

The RX850 Pro has the following features.

1) Conformity with µITRON3.0 specification

The RX850 Pro is designed as a typical built-in control OS architecture that conform to the µITRON3.0 specification. The RX850 Pro implements µITRON3.0 functions up to level E.

The µITRON3.0 specification applies to built-in, real-time control OS.

2) High versatility

In addition to the system calls specified by the µITRON3.0 specification, the RX850 Pro provides original system calls specific to the RX850 Pro, so that it can run on more generalized application systems.

The RX850 Pro can be used to create a real-time, multitasking OS that is compact and ideal for the user's needs because the functions (system calls) to be used by the application system can be selected while configuring the system.

3) Realization of real-time processing and multitasking

The RX850 Pro supports the following functions to realize complete real-time processing and multitasking.

- Task management function
- Task-associated synchronization function
- Synchronous communication management function
- Interrupt processing management function
- Memory pool management function
- Time management function
- System management function
- Scheduling function

4) Scheduling lock function
The RX850 Pro supports functions for disabling and resuming dispatching (task scheduling).
This allows the user to disable and resume a dispatching process from the processing program level.

5) Compact design
The RX850 Pro is a real-time, multitasking OS that has been designed on the assumption that it will be incorporated into the target system; it has been made as compact as possible to enable it to be loaded into a system's ROM.

6) Utilization of original instructions
The high-speed execution speed of the V850 Series microcontrollers, combined with original instructions, enables high-speed processing.

7) Utilization of internal ROM/RAM
By making use of the internal ROM/RAM built into the V850 Series, rapid instruction execution and data access are possible.

8) Application utility support
The RX850 Pro supports the following utilities to aid in application system construction.

- Configurator CF850 Pro
- Task debugger RD850 Pro
- System performance analyzer AZ850
- High-level language interface library

Note Task debugger for RX850 Pro is referred to as RD850 pro in this user's manual.

9) C compiler package
The RX850 Pro supports the following V850 Series C compiler packages.

- CA850 (NEC Electronics Corporation)
- CCV850/CCV850E (Green Hills Software^{TM} , Inc.)

1.3 Execution Environment

The RX850 Pro has been developed as an OS for embedded control and runs on a target system equipped with the following hardware.

1) Target CPU
V850 core, V850E1 core, V850E2 core, V850ES core

2) Peripheral controller
The RX850 Pro eliminates the hardware-dependent portions from the nucleus and supplies them as sample source files, in order to support a range of execution environments. Therefore, the sample source files that are written for each target system enable support of the execution environments without special peripheral controllers.

1.4 Development Enviornment

This section explains the hardware and software environments required to develop application systems.

1.4.1 Hardware environment

1) Host machine

- IBM PC/AT™-compatible machine

2) In-circuit emulators

- IE-703002-MC (V851, V852, V853, V854, V850/SA1, V850/SBx, V850/SV1)
- IE-703102-MC (V850E/MS1)
- IE-V850E-MC-A (V850E/MA1, V850E/MA2, NB85E core)
- IE-V850E-MC (V850E/IA1, V850E/IA2)

1) I/O board for in-circuit emulator

- IE-703003-MC-EMI (V853)
- IE-703008-MC-EMI (V854)
- IE-703017-MC-EMI (V850/SA1)
- IE-703037-MC-EM1 (V850/SBx)
- IE-703040-MC-EM1 (V850/SV1)
- IE-703102-MC-EM1 (V850E/MS1 5V)
- IE-703102-MC-EM1-A (V850E/MS1 3.3V)
- IE-703107-MC-EM1 (V850E/MA1, V850E/MA2)
- IE-703116-MC-EM1 (V850E/IA1)
- IE-703114-MC-EM1 (V850E/IA2)
- IE-V850E-MC-EM1-A (NB85E core 5V)
- IE-V850E-MC-EM1-B (NB85E core 3.3V)

Note These I/O boards must be used in combination with the in-circuit emulator.

2) PC interface boards

- IE-70000-98-IF-C (for PC-9800 series, C bus)
- IE-70000-PC-IF-C (for IBM PC/AT-compatible machines, ISA bus)
- IE-70000-CD-IF-A (for PCMCIA socket)
- IE-70000-PCI-IF (for PCI bus)

1.4.2 Software environment

1) OS (host machine in parentheses)

- Windows 98, Me, NT 4.0, 2000, XP (IBM PC/AT-compatible machines)

2) Cross tools

- CA850 (NEC Electronics Corporation)
- CCV850/CCV850E (Green Hills Software, Inc.)

3) Debuggers

- ID850 (NEC Electronics Corporation)
- SM850 (NEC Electronics Corporation)
- MULTI™, MULTI2000 (Green Hills Software, Inc.)
- PARTNER™ (Kyoto Microcomputer)

4) Task debugger

- RD850 Pro (NEC Electronics Corporation)

5) System performance analyzer

- AZ850 (NEC Electronics Corporation)

CHAPTER 2 INSTALLATION

This chapter explains how to install or uninstall the RX850 Pro.

2.1 Installing

This section explains how to install the RX850 Pro. To install the RX850 Pro again, uninstall it first.

The RX850 Pro is supplied on a single CD-ROM, regardless of whether it is an object release version or source release version. The package of the RX850 Pro includes the RD850 Pro (task debugger), AZ850 (system performance analyzer) and Online Help. Those program can also be installed at the same time.

Install RX850 Pro by following the procedure below.

1) Start Windows.

2) Insert the CD-ROM into the CD drive. The setup program will start automatically. If the setup program does not start, start Explorer, and then double-click "INSTALL.EXE" on the CD drive. After that, install the RX850 Pro in accordance with the messages displayed on the monitor screen.

3) Check to see if the files stored in the supply media of the RX850 Pro have been installed in the host machine, by using a standard application of Windows such as Explorer. For details of each holder, refer to "2.3 Holder Configuration".

2.2 Uninstalling

This section explains how to uninstall the RX850 Pro.

1) Start Windows.

2) Start "Add/Remove Programs" on the control panel, and select the item to be uninstalled such as "NEC EL RX850PRO NEC EL (Object release)", etc.

2.3 Holder Configuration

This section explains the holder configuration of the files read from the supply medium when RX850 Pro has been installed. The RX850 Pro is supplied in the form of an object release version or a source release version. Each version is available as an CA850 version and a GHS compiler version.

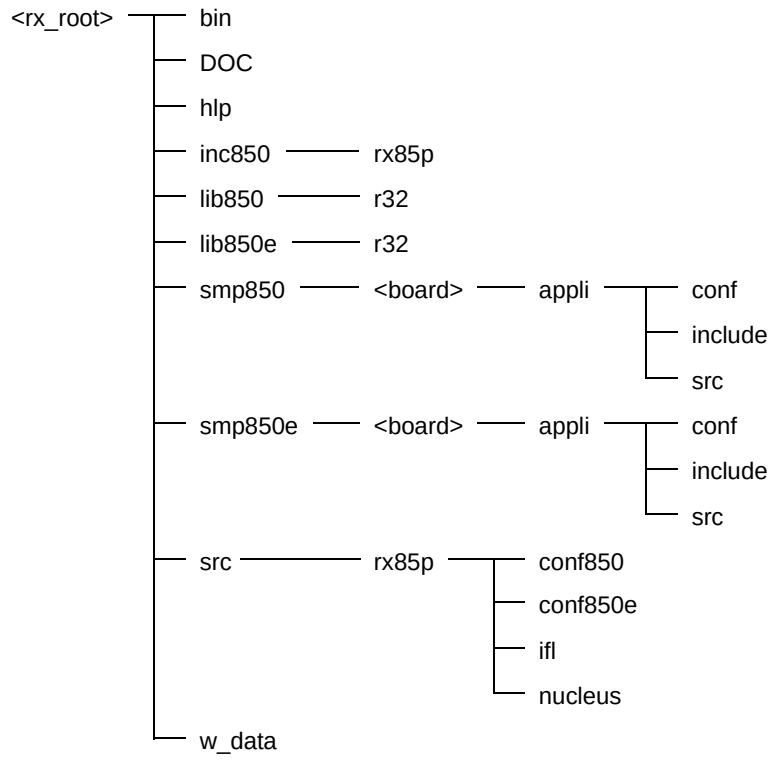
The holder configurations are shown in the following sections.

- Object release version / CA850 version
- Object release version / GHS compiler version
- Source release version / CA850 version
- Source release version / GHS compiler version

2.3.1 Object release version / CA850 version

Figure 2-1 shows the folder configuration when the files (object release version/CA850 version) stored in the RX850 Pro distribution media have been installed.

Figure 2-1 Holder Configuration (Object Release Version/CA850 Version)



The details of each folder are shown below.

1) <rx_root>
This folder is the “installation folder of the RX850 Pro” specified at the time of installation.

2) <rx_root>\bin
This folder the stores the application utility tools for the RX850 Pro.

cf850pro.exe	:	Configurator (CF850 Pro)
rx703100p.dll	:	DLL file for CF850 Pro

3) <rx_root>\DOC
This folder the stores the document files for the RX850 Pro.

4) <rx_root>\hlp
This folder the stores the online help file for the RX850 Pro.

5) <rx_root>\inc850
This folder stores the standard header files for the RX850 Pro.

stdrx85p.h	:	Standard header file (for C language)
stdrx85p.inc	:	Standard header file (for assembly language)

6) <rx_root>\inc850\rx85p
This folder stores the header files for the RX850 Pro.

7) <rx_root>\lib850\r32
This folder stores the library files (for V850 core, 32-register mode) for the RX850 Pro.

librxp.a	:	Nucleus library (Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.)
librxpm.a	:	Nucleus library (Immediately before rel_blk is not issued, the first 4 bytes of the target memory block need to be cleared to 0.)
libchp.a	:	Interface library (With parameter check)
libncp.a	:	Interface library (Without parameter check)
libdbp.a	:	Interface library (With parameter check, including system calls for debugging)
rxcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled.)
rxtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled.)
rxdbcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled. , including system calls for debugging)
rxdbtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled. , including system calls for debugging)

8) <rx_root>\lib850e\r32
This folder stores the library files (for V850E1/V850E2/V850ES core, 32-register mode) for the RX850 Pro.

librxp.a	:	Nucleus library (Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.)
librxpm.a	:	Nucleus library (Immediately before rel_blk is not issued, the first 4 bytes of the target memory block need to be cleared to 0.)
libchp.a	:	Interface library (With parameter check)
libncp.a	:	Interface library (Without parameter check)
libdbp.a	:	Interface library (With parameter check, including system calls for debugging)
rxcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled.)
rxtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled.)
rxdbcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled. , including system calls for debugging)
rxdbtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled. , including system calls for debugging)

9) <rx_root>\smp850\<board>
This folder stores the sample program (for V850 core) for the RX850 Pro.

10) <rx_root>\smp850\<board>\appli\conf
This folder stores the command file that generates a load module of the RX850 Pro.
The load module “sample.out” can be generated into this folder by using the command file in this folder.

sample.prj	:	Project file for load module
sample.prw	:	Work space file for load module

11) <rx_root>\smp850\<board>\appli\include
This folder stores the header files for sample program.

12) <rx_root>\smp850\<board>\appli\src
This folder stores the source files and the link directive file for sample program.

13) <rx_root>\smp850e\<board>
This folder stores the sample program (for V850E1/V850E2/V850ES core) for the RX850 Pro.

14) <rx_root>\smp850e\<board>\appli\conf
This folder stores the command file that generates a load module of the RX850 Pro.
The load module “sample.out” can be generated into this folder by using the command file in this folder.

sample.prj	:	Project file for load module
sample.prw	:	Work space file for load module

15) <rx_root>\smp850e\<board>\appli\include
This folder stores the header files for sample program.

16) <rx_root>\smp850e\<board>\appli\src
This folder stores the source files and the link directive file for sample program.

17) <rx_root>\src\rx85p\conf850
This folder stores the command file that generates the interface library (for V850 core, 32-register mode) of the RX850 Pro.
The interface library “libchp.a, libncp.a” can be generated into this folder by using the command file in <rx_root>\lib850\r32 folder.

libchp.prj	:	Projec file for interface library (libchp.a)
libncp.prj	:	Project file for interface library (libncp.a)
library.prw	:	Work space file for interface library (libchp.a，libncp.a)

18) <rx_root>\src\rx85p\conf850e
This folder stores the command file that generates the interface library (for V850E1/V850E2/V850ES core, 32-register mode) of the RX850 Pro.
The interface library “libchp.a, libncp.a” can be generated into this folder by using the command file in <rx_root>\lib850\r32 folder.

libchp.prj	:	Project file for Interface library (libchp.a)
libncp.prj	:	Project file for Interface library (libncp.a)
library.prw	:	Work space file for Interface library (libchp.a，libncp.a)

19) <rx_root>\src\rx85p\ifl
This folder stores the source files for interface library (libchp.a，libncp.a，libdbp.a).

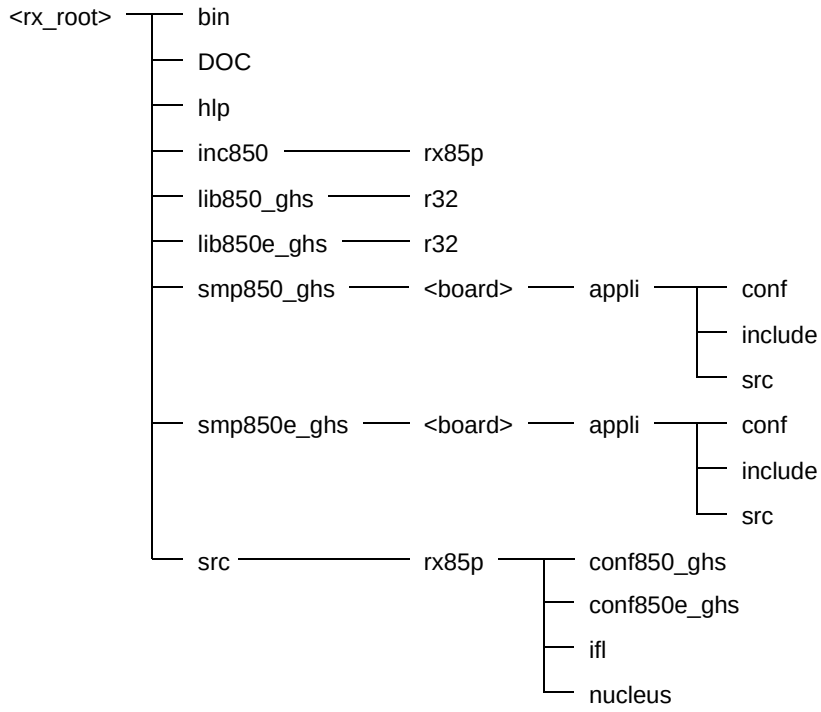
20) <rx_root>\src\rx85p\nucleus
This folder stores the source files for enabling the AZ850 trace function.

21) <rx_root>\w_data
This folder stores the sample link directive file for when the RX850 Pro for PM+ is used.

2.3.2 Object release version / GHS compiler version

Figure 2-2 shows the folder configuration when the files (object release version/GHS compiler version) stored in the RX850 Pro distribution media have been installed.

Figure 2-2 Holder Configuration (Object Release Version/GHS Compiler Version)



The details of each folder are shown below.

1) <rx_root>
This folder is the “installation folder of the RX850 Pro” specified at the time of installation.

2) <rx_root>\bin
This folder the stores the application utility tools for the RX850 Pro.

cf850pro_ghs.exe : Configurator (CF850 Pro)

3) <rx_root>\DOC
This folder the stores the document files for the RX850 Pro.

4) <rx_root>\hlp
This folder the stores the online help file for the RX850 Pro.

5) <rx_root>\inc850
This folder stores the standard header file for the RX850 Pro.

stdrx85p.h : Standard header file

6) <rx_root>\inc850\rx85p
This folder stores the header files for the RX850 Pro.

7) <rx_root>\lib850_ghs\r32
This folder stores the library files (for V850 core, 32-register mode) for the RX850 Pro.

librxp.a : Nucleus library (Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.)
librxpm.a : Nucleus library (Immediately before rel_blk is not issued, the first 4 bytes of the target memory block need to be cleared to 0.)
libchp.a : Interface library (With parameter check)

libncp.a	:	Interface library (Without parameter check)
libdbp.a	:	Interface library (With parameter check, including system calls for debugging)
rxcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled.)
rxtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled.)
rxdbcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled. , including system calls for debugging)
rxdbtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled. , including system calls for debugging)

8) <rx_root>\lib850e_ghs\r32
This folder stores the library files (for V850E1/V850E2/V850ES core, 32-register mode) for the RX850 Pro.

librxp.a	:	Nucleus library (Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.)
librxpm.a	:	Nucleus library (Immediately before rel_blk is not issued, the first 4 bytes of the target memory block need to be cleared to 0.)
libchp.a	:	Interface library (With parameter check)
libncp.a	:	Interface library (Without parameter check)
libdbp.a	:	Interface library (With parameter check, including system calls for debugging)
rxcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled.)
rxtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled.)
rxdbcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of all maskable interrupts is enabled. , including system calls for debugging)
rxdbtmcore.o	:	Nucleus common object (In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled. , including system calls for debugging)

9) <rx_root>\smp850_ghs\<board>
This folder stores the sample program (for V850 core) for the RX850 Pro.

10) <rx_root>\smp850_ghs\<board>\appli\conf
This folder stores the command file that generates a load module of the RX850 Pro.
The load module “sample.out” can be generated into this folder by using the command file in this folder.

sample.bld	:	Bild file for load module

11) <rx_root>\smp850_ghs\<board>\appli\include
This folder stores the header files for sample program.

12) <rx_root>\smp850_ghs\<board>\appli\src
This folder stores the source files and the link directive file for sample program.

13) <rx_root>\smp850e_ghs\<board>
This folder stores the sample program (for V850E1/V850E2/V850ES core) for the RX850 Pro.

14) <rx_root>\smp850e_ghs\<board>\appli\conf
This folder stores the command file that generates a load module of the RX850 Pro.
The load module “sample.out” can be generated into this folder by using the command file in this folder.

sample.bld	:	Bild file for load module

15) <rx_root>\smp850e_ghs\<board>\appli\include
This folder stores the header files for sample program.

16) <rx_root>\smp850e_ghs\<board>\appli\src
This folder stores the source files and the link directive file for sample program.

17) <rx_root>\src\rx85p\conf850_ghs
This folder stores the command file that generates the interface library (for V850 core, 32-register mode) of the RX850 Pro.
The interface library “libchp.a, libncp.a” can be generated into this folder by using the command file in <rx_root>\lib850_ghs\r32 folder.

library.bld	:	Bild file for Interface library (libchp.a，libncp.a)

18) <rx_root>\src\rx85p\conf850e_ghs
This folder stores the command file that generates the interface library (for V850E1/V850E2/V850ES core, 32-register mode) of the RX850 Pro.
The interface library “libchp.a, libncp.a” can be generated into this folder by using the command file in <rx_root>\lib850_ghs\r32 folder.

library.bld	:	Bild file for Interface library (libchp.a，libncp.a)

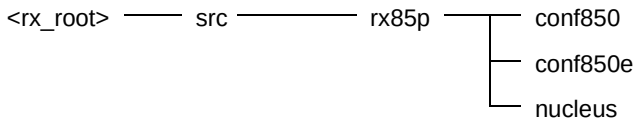
19) <rx_root>\src\rx85p\ifl
This folder stores the source files for interface library (libchp.a，libncp.a，libdbp.a).

20) <rx_root>\src\rx85p\nucleus
This folder stores the source files for enabling the AZ850 trace function.

2.3.3 Source release version / CA850 version

Figure 2-3 shows the folder configuration when the files (source release version/CA850 version) stored in the RX850 Pro distribution media have been installed.

Figure 2-3 Holder Configuration (Source Release Version/CA850 Version)



The details of each folder are shown below.

1) <rx_root>
This folder is the “installation folder of the RX850 Pro” specified at the time of installation.

2) <rx_root>\src\rx85p\conf850
This folder stores the command file that generates the library files (for V850 core, 32-register mode) of the RX850 Pro.
The library files “librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o” can be generated into this folder by using the command file in <rx_root>\lib850\r32 folder.

makefile	:	Make file for library file (librxp.a,librxpm.a,libchp.a,libncp.a,libdbp.a,rxcore.o,rxtmcore.o, rxdbcore.o, rxdbtmcore.o)

3) <rx_root>\src\rx85p\conf850e
This folder stores the command file that generates the library files (for V850E1/V850E2/V850ES core, 32-register mode) of the RX850 Pro.
The library files “librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o” can be generated into this folder by using the command file in <rx_root>\lib850e\r32 folder.

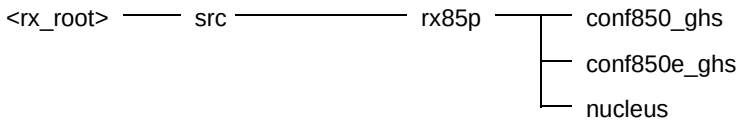
makefile	:	Make file for library file (librxp.a,librxpm.a,libchp.a,libncp.a,libdbp.a,rxcore.o,rxtmcore.o, rxdbcore.o, rxdbtmcore.o)

4) <rx_root>\src\rx85p\nucleus
This folder stores the source files for library files (llibrxp.a, librxpm.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o).

2.3.4 Source release version / GHS compiler version

Figure 2-4 shows the folder configuration when the files (source release version/GHS compiler version) stored in the RX850 Pro distribution media have been installed.

Figure 2-4 Holder Configuration (Source Release Version/GHS Compiler Version)



The details of each folder are shown below.

1) <rx_root>
This folder is the “installation folder of the RX850 Pro” specified at the time of installation.

2) <rx_root>\src\rx85p\conf850_ghs
This folder stores the command file that generates the library files (for V850 core, 32-register mode) of the RX850 Pro.
The library files “librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o” can be generated into this folder by using the command file in <rx_root>\lib850_ghs\r32 folder.

nucleus.bld	:	Bild file for library file (librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o)

3) <rx_root>\src\rx85p\conf850e_ghs
This folder stores the command file that generates the library files (or V850E1/V850E2/V850ES core, 32-register mode) of the RX850 Pro.
The library files “librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o” can be generated into this folder by using the command file in <rx_root>\lib850e_ghs\r32 folder.

nucleus.bld	:	Bild file for library file (librxp.a, librxpm.a, libchp.a, libncp.a, libdbp.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o)

4) <rx_root>\src\rx85p\nucleus
This folder stores the source files for library files (llibrxp.a, librxpm.a, rxcore.o, rxtmcore.o, rxdbcore.o, rxdbtmcore.o).

CHAPTER 3 SYSTEM CONSTRUCTION

This chapter explains how to construct an application system using the RX850 Pro.

3.1 Outline

System construction involves incorporating created load modules into a target system, using the file group copied from the RX850 Pro distribution media to the user development environment (host machine).
The system construction procedure is outlined below.

1) Creating System Configuration File

2) Creating information file

- System information table file
- System call table file
- System information header file

Note The information tables are created by using the configurat.

3) Creating System Initialization Processing

- Boot processing
- Hardware initialization block
- Software initialization block
- Interrupt entry

4) Creating Processing Programs

- Task
- Interrupt handler
- Cyclic handler
- Extended SVC handler
- Extended SVC handler interface library

Note The programs are created by using C language or assembly language.

5) Creating Initialization Data Save Area (only when CA850 is used)

6) Creating Llink Directive File

7) Creating Load Module

8) Embedding System

Figure 3-1 shows the procedure for organizing the system when CA850 version is used. Figure 3-2 shows the procedure for organizing the system when GHS compiler version is used.

Figure 3-1 Example of System Construction (CA850 Version)

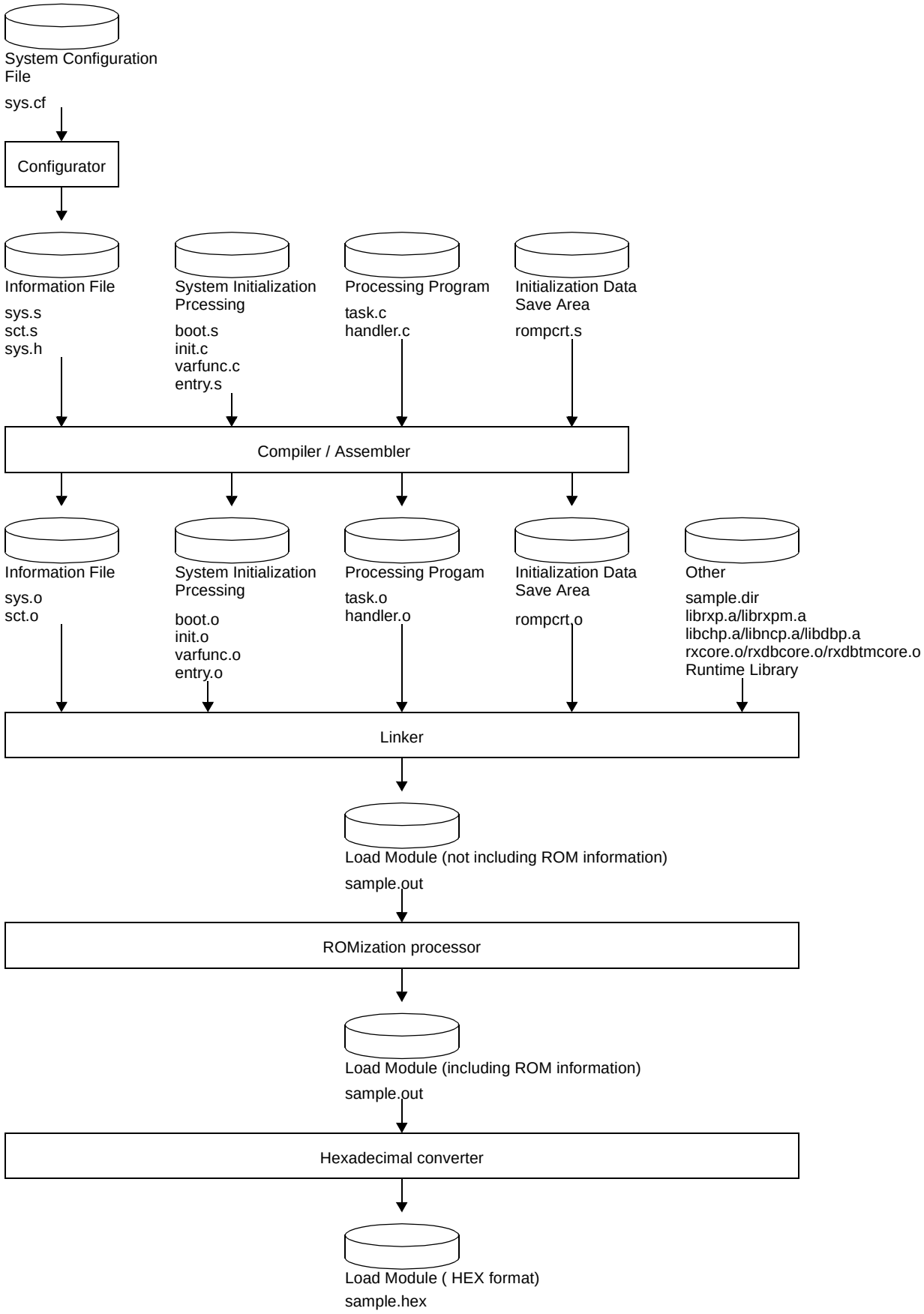
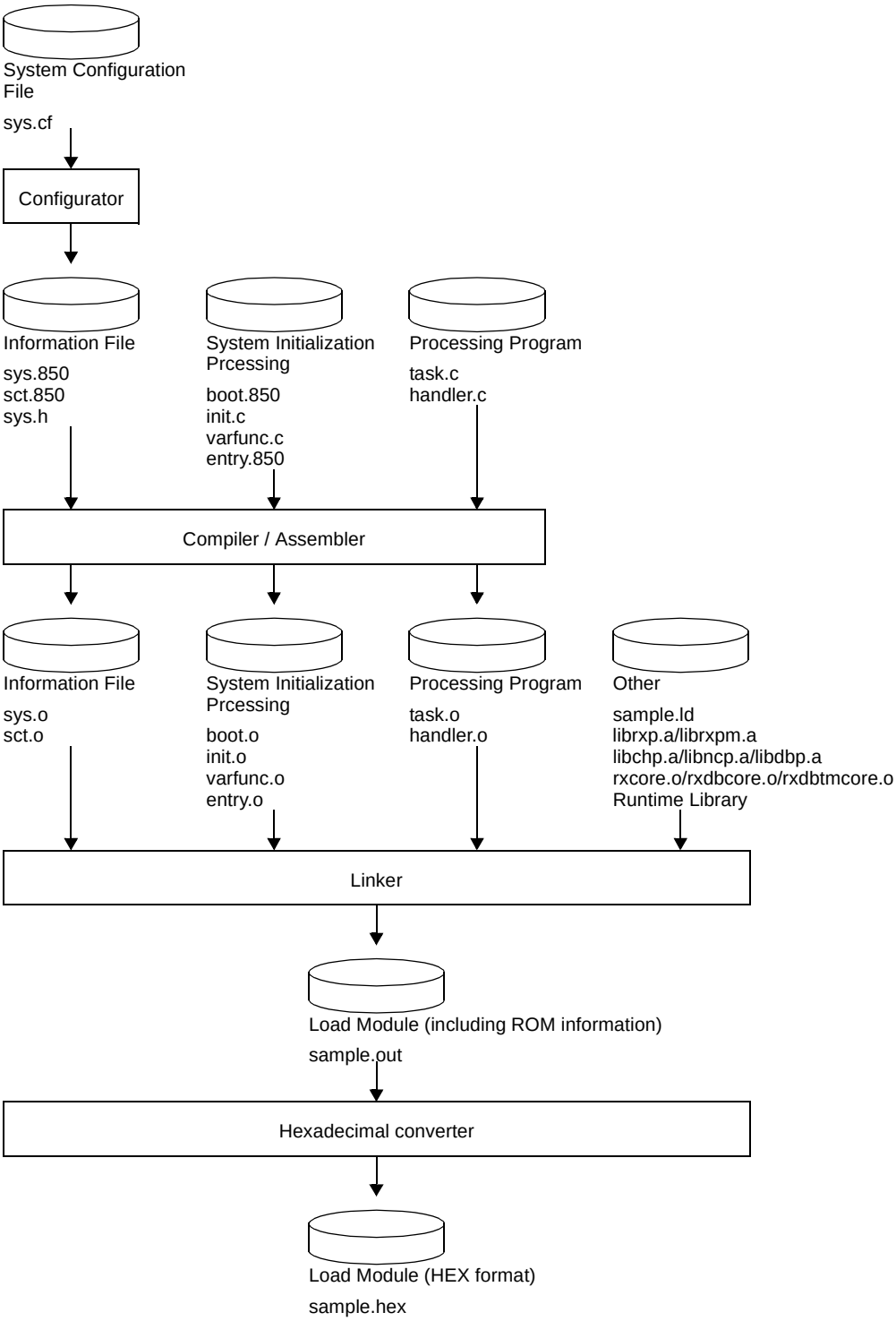


Figure 3-2 Example of System Construction (GHS Compiler Version)



The flow of organizing the system is explained based on the sample program supplied with the package.
The program is stored in the following holder if the RX850 Pro has been installed in the holder <rx_root>.

Table 3-1 Sample Program Storage Holdery

Compiler/CPU	Storage Holder
CA850 version / V850 core	<rx_root>\smp850\rx85p\src
CA850 version / V850E1/V850E2/V850ES core	<rx_root>\smp850e\rx85p\src
GHS compiler version / V850 core	<rx_root>\smp850_ghs\rx85p\src
GHS compiler version / V850E1/V850E2/V850ES core	<rx_root>\smp850e_ghs\rx85p\src

Reference either of the above directories in accordance with the compiler or CPU being used.

3.2 Creating System Configuration File

Create an information table, called a system configuration file, which holds the various data used with the RX850 Pro.
This file is necessary for creating the following using the configurator.

- System information table file
- System call table file
- System information header file

For details on how to create the system configuration file, see "3.2.1 Creating information file".
The "system information table file" contains information on the resources of the RX850 Pro, such as tasks, semaphores, and memory pools. The "system call table file" contains a list of system calls used for applications. The "system information header file" has a description that makes the symbol names specified as resource IDs, such as those of tasks and semaphores created with the system information table file, correspond to the actual symbol ID numbers, by using the #define instruction.
The sample system configuration file is :

- sys.cf

For the contents and syntax of the system configuration file, see "CHAPTER 6 SYSTEM CONFIGURATION FILE".

3.2.1 Creating information file

Create the "system information table file", "system call table file", and "system information header file" from the system configuration file created as described in "3.2 Creating System Configuration File" above, by using the configurator.
The following file names are recommended.

< System information table file >

- CA850 version : sys.s
- GHS compiler version : sys.850

< Systemcall table file >

- CA850 version : sct.s
- GHS compiler version : sct.850

< System information header file >

- sys.h

To organize an application using the RX850 Pro, assemble sys.s (sys.850) and sct.s (sct.850), and link the created object. The C source must include sys.h.
For details on how to use the configurator that is used to create these files, see "CHAPTER 7 CONFIGURATOR (CF850 Pro)".

3.3 Creating System Initialization Processing

The system initialization block is a function consisting of program segments that are dependent upon the user's target system. This function is used to facilitate porting and customization.

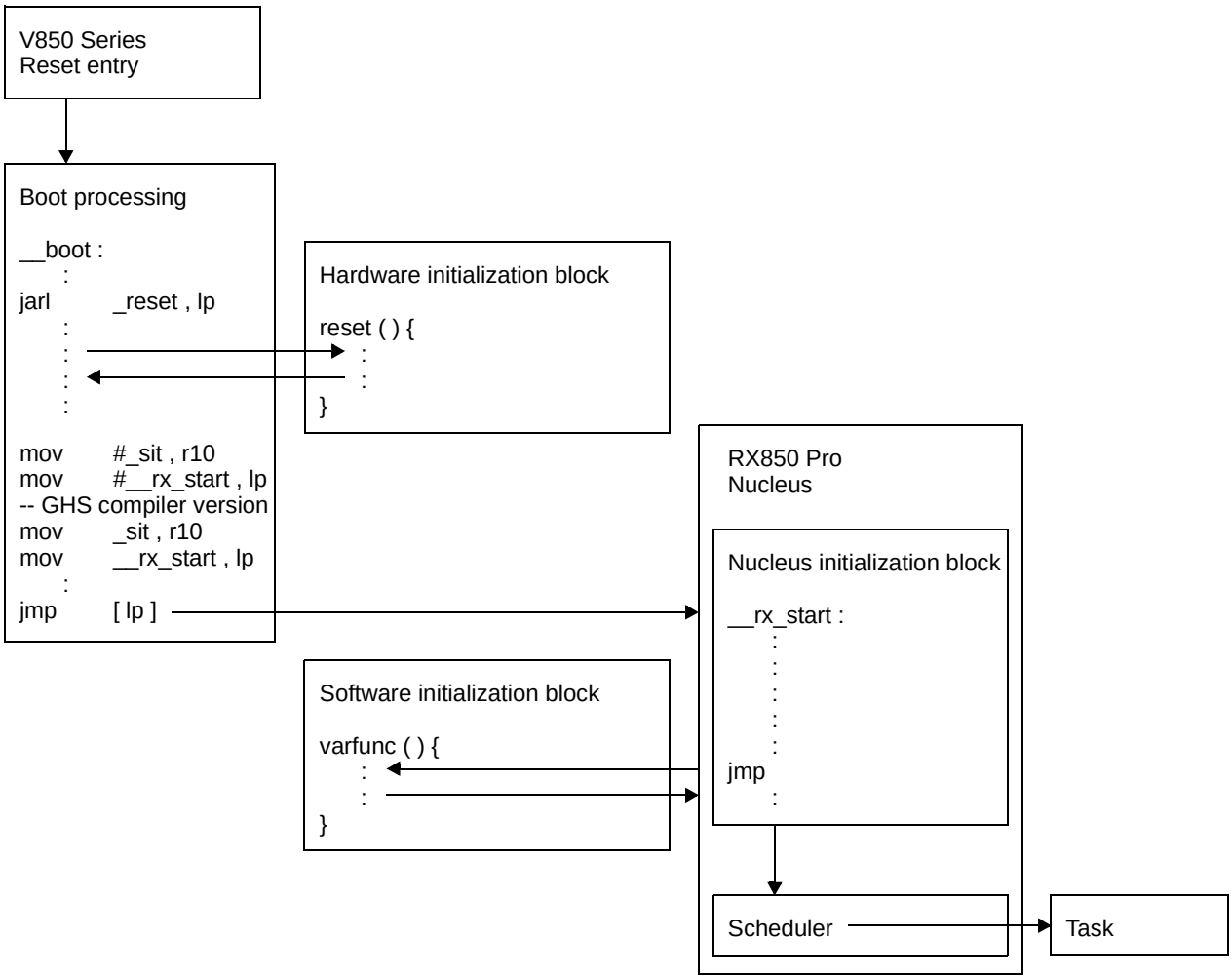
The sample file is as follows.

Table 3-2 Configuration of System Initialization Block

Sample File Name	Type	Function Name	Role
boot.s (CA850 version) boot.850 (GHS compiler version)	Boot processing	_boot	Boot processing of system
init.c	Hardware initialization block	reset	Initialization processing of hardware
varfunc.c	Software initialization block	varfunc	Initialization processing of software
entry.s (CA850 version) entry.850 (GHS compiler version)	Interrupt entry	None	Processing to branch to interrupt servicing

The rough flow of the system initialization block is illustrated below. Each processing is explained next.

Figure 3-3 Flow of System Initialization Block



3.3.1 Boot processing

The boot processing is assigned to the reset entry (handler address : 0x0) of the V850 Series and is the system initialization processing that is executed first.

The description following the label "_boot" in the sample file boot.s (boot.850) is the entity of the boot processing. The instructions that cause execution to jump from the reset entry to this label are as follows. These instructions are in the same entry.s (entry.850) file.

< CA850 version >

```
.section    "RESET"
.extern     __boot

mov         #__boot , lp
jmp         [ lp ]
```

< GHS compiler version >

```
.org        0x00000000
.extern     __boot

mov         __boot , lp
jmp         [ lp ]
```

The lower 2 lines of the instructions are assigned to the handler address [0x0]. When reset is executed, therefore, these instructions are executed, execution jumps to _boot, and the boot processing is executed.

The following must be performed as part of the boot processing.

1) Setting of tp (text pointer), gp (global pointer), and ep (element pointer)
2) Setting of sp (stack pointer) used for boot processing
3) Setting of address of _sit symbol to r10
4) Transferring control to the RX850 Pro nucleus initialization block by jumping to the __rx_start symbol using the jmp instruction

In addition to the above, processing (jarl_reset, lp) that causes execution to jump to the reset function, which is a "hardware initialization block", is executed between 2) and 3) in the sample.

The stack pointer to be set in 2) above is independent of the stack for tasks and interrupt handlers. After the RX850 Pro has been started, the stack used by tasks and interrupt handlers is managed by the RX850 Pro itself, by using the system information table file, and the stack pointer is automatically switched by means of task switching or interrupts. Therefore, the stack pointer specified in the boot processing is used before the RX850 Pro is started.

This stack pointer is used, for example, when execution jumps to a function and that function has data to be saved to the stack. This stack pointer is used if it is necessary to use the stack with the reset function of the sample.

Although the sample program uses a stack of 0x28000 bytes, such a high-capacity stack is not usually necessary. This stack area is of the size defined by the system memory area used for the RX850 Pro ("System memory information" in the system configuration file), and is used as the system memory area after the RX850 Pro has been started.

In the sample of the CA850 version, the bss area on RAM is initialized (cleared to 0) in boot processing. In the sample of the GHS compiler version, this function initializes the bss area and copies the default value data in the software initialization processing (varfunc.c).

With the CA850 version, the default value data is copied by creating an area of the default value data (rompcrt.s) and by using the function _rcopy. For details, refer to "CA850 Operation User's Manual".

At the end of the boot processing, processing for 3) and 4) is necessary. Perform the following processing.

< CA850 version >

```
.extern     _sit
mov         #_sit , r10

.extern     __rx_start
mov         #__rx_start , lp
jmp         [ lp ]
```

< GHS compiler version >

```
.extern     _sit
mov         _sit , r10

.extern     __rx_start
mov         __rx_start , lp
jmp         [ lp ]
```

The description in the RX850 Pro following the symbol "__rx_start" is the nucleus initialization processing of the RX850 Pro. After the boot processing has been completed, transfer control to the nucleus initialization processing by using the jmp instruction. At this time, substitute the address of _sit symbol into the r10 register. This is because resources are created and initialized based on this address substituted into the r10 register and the "system information table file" created from the system configuration file.

NEC Electronics recommends changing the description of the boot processing to an environment suitable for the user, based on the boot processing of the sample.

3.3.2 Hardware initialization block

The hardware initialization block consists of functions called from the boot processing in the sample program. These functions are provided to initialize the hardware on the target system, as a series of boot processing.

In the sample program, the reset function initializes the hardware. This function is called from the boot processing. There is no problem even if this function is not used, if initializing the hardware is not necessary or if the hardware is initialized by other processing.

The hardware initialization block of the sample program performs the following processing.

1) Initialization of interrupt controller/clock controller
2) Initialization of peripheral I/O register/controller
3) Returning control to boot processing

3.3.3 Nucleus initialization block

The nucleus initialization block is an internal routine of the RX850 Pro that is executed after completion of the boot processing. This block creates the RX850 Pro system management block, and creates and initializes information on items such as tasks, semaphores, and memory pools, based on the "system information table file" created from the system configuration file.

The RX850 Pro places the CPU in the HALT status if initialization was not correctly performed by the nucleus initialization block. If the RX850 Pro is not started and the CPU enters the HALT status after execution has jumped from the boot processing to the initialization processing, the system memory area ("System memory information" in the system configuration file) used to create the management block of the RX850 Pro is probably insufficient. Check that a sufficient memory capacity has been reserved.

Once initialization has been completed in the nucleus initialization block, an initialization handler is called. This software initialization block is specified by "Initialization handler information" of the system configuration file and is the varfunc function in the sample program. For details of this function, see "3.3.4 Software initialization block".

When control has returned from the software initialization block, the scheduler is started, and then the RX850 Pro is started.

3.3.4 Software initialization block

The software initialization block is a function that is called from the nucleus initialization block. Describe the processing to be performed before starting the RX850 Pro.

The software initialization block calls a function specified by "Initialization handler information" of the system configuration file. In the sample program, this function is varfunc. Even if the processing is not necessary, create the function as a function that performs no processing. At the end of the handler, return to the nucleus initialization processing by using the return instruction.

In the sample program of the CA850 version, this function is defined as a function that executes nothing. In the sample program of the GHS compiler version, it initializes the bss area on RAM (clears to 0) and copies the initial value data. Because this routine is used in the way recommended by GHS, refer to the manuals of GHS for details. With the CA850 version, the default data value can be copied into the software initialization block. For details of how to copy the default value data, refer to "CA850 Operation User's Manual".

3.3.5 Interrupt entry

An interrupt entry is an instruction that is executed if an interrupt occurs, and is assigned to the "interrupt handler address" of the V850 Series. The interrupt entry must be defined for all the interrupts used by the user, and must be described in assembly language. The interrupt handler of the sample is described in "entry.s" for the CA850 version, and in "entry.850" for the GHS compiler version.

The interrupts of the RX850 Pro are handled by 2 types of handlers : a "directly activated interrupt handler" and an "indirectly activated interrupt handler", each of which differs from the other in entry description.

In the directly activated interrupt handler, describe a branch instruction in the same manner as an ordinary interrupt entry. In the sample, the interrupt "INTP110 (handler address : 0x180)" is an example of a directly activated interrupt handler.

With the CA850 version, the .section quasi directive is used. With the GHS compiler version, the .org instruction is used. For details of each instruction, refer to "CA850 Package Assembly Language User's Manual" for the CA850 version. For the GHS compiler version, refer to the GHS manual related to language.

The entry of the directly activated interrupt handler is as follows.

< CA850 version >

```
        .section    "INTP110"
        jr          _intp110_entry
```

< GHS compiler version >

```
        .org        0x00000180
        jr          _intp110_entry
```

The destination label "_intp110_entry" is defined in the same file, and execution jumps to the entity of the handler (intp130) after the preprocessing and post-processing of the directly activated interrupt handler are described (macro description). For details of how to describe a directly activated interrupt handler, refer to "RX850 Pro Basics User's Manual".

The indirectly activated interrupt handler uses the macro provided for the RX850 Pro. This means that the contents of the macro must be assigned to the handler address. The macro name is "RTOS_IntEntry_Indirect". In the sample program, the interrupt "INTP120 (handler address : 0x1c0)" is used as an example of an indirectly activated interrupt handler.

The CA850 version uses the .section quasi directive and the GHS compiler version uses the .org instruction. For details of each instruction, refer to "CA850 Assembly Language User's Manual" for the CA850 version. For the GHS compiler version, refer to the GHS manual related to language.

The entry of the indirectly activated interrupt handler is as follows.

< CA850 version >

```
        .section    "INTP120"
        RTOS_IntEntry_Indirect
```

< GHS compiler version >

```
        .org            0x000001c0
        RTOS_IntEntry_Indirect
```

An interrupt entry must also be registered for the timer interrupt used with the RX850 Pro in the same manner as the indirectly activated interrupt handler. This interrupt entry is described in the sample program (for V850E) as follows because "INTCMD0 (handler address : 0x240)" is used as a timer interrupt.

< CA850 version >

```
        .section        "INTCMD0"
        RTOS_IntEntry_Indirect
```

< GHS compiler version >

```
        .org            0x00000240
        RTOS_IntEntry_Indirect
```

Note For interrupt handlers defined in the system configuration file, the configurator automatically outputs the interrupt entry to the system information table. Therefore, the user does not have to describe the interrupt entry.

3.4 Creating Processing Programs

Create a processing program (application).
The application processing units required for the RX850 Pro are broadly classified into the following.

- Task
- Directly activated interrupt handler
- Indirectly activated interrupt handler
- Cyclic handler
- Extended SVC handler

The contents of the sample, except the extended SVC handler, are shown below.

Table 3-3 Configuration of Processing Program

Sample File Name	Type	Function Name	Role
task.c	Task	task1 task2	Task entity
handler.c	Indirectly activated interrupt handler Directly activated interrupt handler Cyclic handler	intp120 intp130 cychdr0 cychdr1	Handler processing

If a processing program described in C issues a system call, include the header file "stdrx85p.h" supplied by the RX850 Pro. This file contains the definition necessary for using the system call. The header file "usr.h" in the sample program defines constants used by functions as necessary and is included in the program. In the sample program of the CA850 version, only a macro of constants is defined. In the sample program of the GHS compiler version, in contrast, a macro of the names and addresses of the peripheral I/O ports of the V850 Series are also defined.

For how to describe an extended SVC handler, refer to "RX850 Pro Basics User's Manual".

3.5 Creating Initialization Data Save Area

When the CA850 version is used, it is necessary to create an area for saving the initialization data. This is because it is necessary to store the initialization data in ROM and to copy the default values of the data to RAM before executing a program. Creating a saving area for the initialization data involves reserving a ROM area in which the initialization data is to be stored.

For details of how to create this area, see the description of "ROMization processor" in "CA850 Operation User's Manual".

With the GHS compiler version, this processing is performed in the software initialization block (varfunc function) of the sample.

3.6 Creating Llink Directive File

Create a link directive file (section map file) containing the "section information" and "address information" referenced by the linker when it links modules. The following sample file is a link directive file.

The following file of the sample is the link directive file.

- sample.dir (CA850 version)
- sample.ld (GHS compiler version)

The sections listed in the table below are essential for the RX850 Pro.

Table 3-4 Essential Sections for RX850 Pro

Section Name	Type of Area
.sit	System information area
.system	RX850 Pro common area
.system_cmn	RX850 Pro scheduler-related area
.system_int	RX850 Pro interrupt servicing-related area
.text	RX850 Pro interface library/system call location area

".sit" has a const attribute and the other sections have a text attribute. Information on these sections must be defined in the link directive file (section map file). As described in the file of the sample, these sections define position information.

The file of the sample that corresponds to these sections is as follows.

< CA850 version >

```
const :     !LOAD       ?R              V0x00001000 {
            .sit        = $PROGBITS     ?A          .sit ;
            .const      = $PROGBITS     ?A          .const ;
} ;
TEXT :      !LOAD       ?RX {
            .system     = $PROGBITS     ?AX         .system ;
            .system_cmn = $PROGBITS     ?AX         .system_cmn ;
            .system_int = $PROGBITS     ?AX         .system_int ;
            .text       = $PROGBITS     ?AX         .text ;
} ;
            :
            :
```

< GHS compiler version >

```
-sec {
            .sit            0x00001000      :
            .system                         :
            .system_int                     :
            .system_cmn                     :
            .text                           :
            :
            :
```

In addition, define sections related to the RAM area, such as .data/.bss section, and those related to the ROM area, such as the const section, as necessary. NEC Electronics recommends changing the description of the link directive file (section map file) to an environment suitable for the user.

For details on how to describe the link directive file, refer to "CA850 Operation User's Manual". For details on how to describe the section map file, refer to the GHS manual related to language.

3.7 Creating Load Module

Next, create a load module (executable module).

Link the objects (.o file) created by compiling and assembling the C source file and assemble source file, based on the link directive file (section map file) created in "3.6 Creating Llink Directive File".

It is necessary to reference the following libraries and link the following objects when linking applications using the RX850 Pro.

- Nucleus library

librxp.a	:	Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.
librxpm.a	:	Immediately before rel_blk is not issued, the first 4 bytes of the target memory block need to be cleared to 0.

- Interface library

libchp.a	:	With parameter check
libncp.a	:	Without parameter check
libdbp.a	:	With parameter check, including system calls for debugging

- Nucleus common object

rxcore.o	:	Immediately before rel_blk is issued, the first 4 bytes of the target memory block need to be cleared to 0.
rxtmcore.o	:	In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled.
rxdbcore.o	:	In the cyclic handler, the acceptance of all maskable interrupts is enabled. , including system calls for debugging
rxdbtmcore.o	:	In the cyclic handler, the acceptance of maskable interrupts that have higher priority than timer interrupts is enabled. , including system calls for debugging

Two types of interface libraries and 2 types of nucleus common objects are available. Which type is to be used should be determined according to the application. For details of the nucleus library, see "CHAPTER 4 INTERFACE LIBRARIES". For details of the nucleus common object, refer to the description of the "time management function" in "RX850 Pro Basics User's Manual".

When linking is successful, an executable module (.out file) is created. At this stage, the executable module can be read to the debugger to execute the application.

The load module file created by the linker correctly locates the initialization data in RAM. If initialization data exists in the application of the CA850 version, a module that reserves an initialization data saving area and that incorporates a copy routine must be created. In this case, a load module that passes through a ROMization processor must be created for the load module created by the linker.

For details on how to use the ROMization processor and for details of the copy routine, refer to "ROMization processor" in "CA850 Operation User's Manual".

3.8 Embedding System

Embed the completed load module file in the system.

To do so, the load module file created in Section "3.7 Creating Load Module" must be converted into a hex file.

By using the hex converter provided with each of the CA850 version and GHS compiler versions, create a hex file of the necessary format. Then, embed this file into the system by using a ROM writer, etc.

CHAPTER 4 INTERFACE LIBRARIES

This chapter describes the interface libraries provided by the RX850 Pro.

4.1 Outline

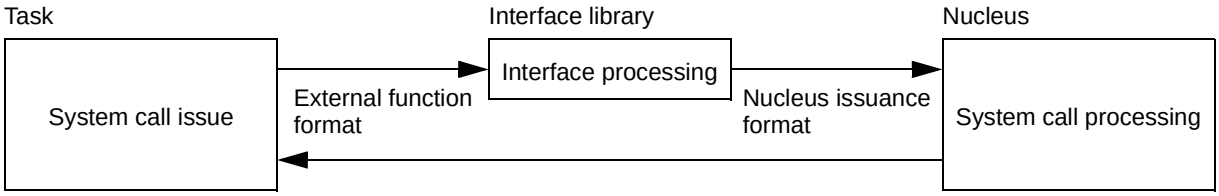
The RX850 Pro provides an interface library that is positioned between the user's processing program and the nucleus of the RX850 Pro. The interface library performs the data setting and other processing required for the nucleus to carry out its functions, before passing control to the nucleus.

Processing programs (tasks, non-tasks) coded in C are generated in the external function format which is used to issue system calls and call extended SVC handlers. The issuance format that can be recognized by the nucleus (nucleus issuance format), however, differs from the external function format.

This necessitates a procedure (interface) for converting the external function format, used to issue system calls and to call extended SVC handlers, into a format that can be issued to the nucleus. This type of interface between a processing program and the nucleus is provided for each system call. The interface library contains a collection of such interfaces.

Figure 4-1 shows the interface library position.

Figure 4-1 Position of Interface Library



4.2 Processing in Interface Library

The interface library performs the following processing.

- Sets necessary information to table managed by nucleus
- Sets necessary data to registers
- Sets error values of system calls (except errors set in nucleus) and returns control to processing program

By preparing the interface library, the nucleus and the processing program of the user can be easily separated. The interface library is linked to the user application. Even if it is necessary to change the processing program of the user after the entity of the nucleus has been stored in ROM, therefore, the ROM storing the nucleus entity does not have to be changed. Also, the user can create load modules while separating them into different sections. The syntax of the interface library for calling system calls is described in “4.5 System Call Interface Library”, and the syntax of the interface library of extended SVC handler is described in “4.6 Extended SVC Handler Interface Library”. Refer to these sections for details.

4.3 Type of Interface Library

The RX850 Pro supplies 2 types of interface libraries : one that has a function to check the parameters of system calls and another that does not. Specify which of the interfaces is to be embedded in the system during linkage.

If the library with the parameter check function is used, and if an illegal parameter is specified when a system call is issued, a return value is always returned. If the library without the parameter check function is used, no return value is returned even if an illegal parameter is specified when a system call is issued.

These 2 types of libraries are used for different applications. For example, the library with the parameter check function may be used for debugging, while the library without the parameter check function is actually embedded in the system, in order to improve the cost effectiveness of the program and save the memory capacity.

An error that returns a return value with the library without the parameter check function is indicated by "*" in the field of the return value of each system call in CHAPTER 11 of “RX850 Pro Basics User's Manual”. If an error that does not return a return value occurs when the library without the parameter check function is used, the operation of the application system is not guaranteed.

4.4 Interface Libraries Supplied

The RX850 Pro supplies the following 2 types of interface libraries.

- For NEC Electronics V850 Series Compiler "CA850"
- For Green Hills Software, Inc. C Cross V800 Compiler "CCV850/CCV850E”

To use any other compiler or to change the interface library as necessary, it is necessary to rewrite the interface library. Once an interface library has been modified, it must be assembled and then defined again as a library.

4.5 System Call Interface Library

The main operation of the system call interface library is detailed below.
The method of system call parameter passing, however, complies with the C compiler used.

- Saves the function code setting of the system call into the r10 register.
- Saves the address setting for the return from the system call into the lp register.
- Checks the system call parameters.
- Acquires the address of the system call entry (the value of the hp register + address 0x100).
- Jumps to the system call entry.

If an error is detected as the result of the system call parameter check, the interface library executes the following.

- Saves the error code associated with the detected error into the r10 register.

Figure 4-2 shows an example of the coding of a system call interface library.

Figure 4-2 Example of System Call Interface Library Coding

```
                .text
                .globl          _syscall_name
                .align          2
_syscall_name
                -- Set the function code to be saved.
                mov             func_code , r10

                -- Parameter check.
                :
                :

                jz              _syscall_err

                -- Acquire the address of the system call entry.
                ld.w            0x100 [ hp ] , r12

                -- Jump to the system call entry.
                jmp             [ r12 ]
```

4.6 Extended SVC Handler Interface Library

The main operation of the extended SVC handler interface library is detailed below.
The method of extended SVC handler parameter passing, however, complies with the C compiler being used.

- Saves the function code setting of the extended SVC handler into the r10 register.
- Saves the address setting for the return from the extended SVC handler into the lp register.
- Checks the extended SVC handler parameters.
- Saves the setting of the extended SVC handler parameter area size into the r11 register.

  Exp. In the case of 4 parameters of int type, 0x10 is saved into the r11 register.
- Acquires the address of the extended SVC handler entry (the value of the hp register + address 0x108).
- Jumps to the extended SVC handler entry.

If an error is detected as a result of the extended SVC handler parameter check, the interface library executes the following.

- Saves the error code associated with the detected error into the r10 register.

Figure 4-3 shows an example of coding an extended SVC handler interface library.

Figure 4-3 Example of Extended SVC Handler Interface Library Coding

```
                .text
                .globl          _svchdr_name
                .align          2
_svchdr_name
                -- Set the function code to be saved.
                mov             func_code , r10

                -- Parameter check.
                :
                :

                jz              _svchdr_err

                -- Set the parameter area size to be saved.
                mov             prm_siz , r11

                -- Acquire the address of the extended SVC handler entry.
                ld.w            0x108 [ hp ] , r12

                -- Jump to the extended SVC handler entry.
                jmp             [ r12 ]
```

CHAPTER 5 MEMORY AND MEMORY CAPACITY ESTIMATION

This chapter explains how the RX850 Pro manages the memory (RAM), and the memory capacity used.

5.1 SPOL and UPOL

The RX850 Pro uses the following 4 RAM areas :

SPOL0 : System Memory Pool 0
SPOL1 : System Memory Pool 1
UPOL0 : User Memory Pool 0
UPOL1 : User Memory Pool 1

The location information of these memory areas is determined by specifying their "first address" and "size" in the system configuration file. In other words, the addresses and size of the usable RAM areas must be specified.

The usage of these memory pools are predetermined as indicated in the table below.

Table 5-1 Types of Memory Pools and Assigned Items

Memory Pool Name	Assigned Items
SPOL0	System base table (SBT) Ready queue Management blocks Stack for task Stack for interrupt handler
SPOL1	Stack for task Stack for interrupt handler Memory pool
UPOL0	Memory pool
UPOL1	Memory pool

SPOL0 must always be generated because information on the system of the RX850 Pro is located in this memory pool. SPOL1 does not have to be generated if SPOL0 suffices. It is possible to improve the performance of the system by locating SPOL0, in which management blocks are located, in the internal RAM, and SPOL1, which requires a relatively large size, in the external RAM.

UPOL1 is necessary for using the memory management function of the RX850 Pro. In this case, also create UPOL0. It is not possible to create just UPOL1.

5.2 Memory Capacity in Management Area

This section explains the size used by the system base table and management blocks of the RX850 Pro. The system base table and management blocks are reserved from SPOL0. Table 5-2 shows the size of a management area used per object and how to calculate the size.

Table 5-2 Size of Object Management Area

Object Name	Management Area Size Per Object (in bytes)	Size Calculation Method (in bytes)
Operating system management table	520 to 1048	504 + Align32 (Task priority range + 4) / 8 + Align4 ((Task priority range + 4) * 2) "Task priority range" is the value of *pri_lvl* of the System maximum value information specified during configuration.
System memory area management table	8	8 * 4 = 32 8 bytes for SPOL0, SPOL1, UPOL0, and UPOL1 each. Even when all the 4 memory pools are not created, 32 bytes are always reserved because 4 tables are always reserved.
Task management table	56	56 * Maximum number of tasks "Maximum number of tasks" is the value of *tsk_cnt* of the System maximum value information specified during configuration.
Semaphore management table	20	20 * Maximum number of semaphores "Maximum number of semaphores" is the value of *sem_cnt* of the System maximum value information specified during configuration.
Event flag management table	20	20 * Maximum number of event flags "Maximum number of event flags" is the value of *flg_cnt* of the System maximum value information specified during configuration.
Mailbox management table	20	20 * Maximum number of mailboxes "Maximum number of mailboxes" is the value of *mbx_cnt* of the System maximum value information specified during configuration.
Interrupt handler management table	16	16 * Maximum number of interrupt handlers + Align4 (Maximum interrupt source number) "Maximum number of interrupt handlers" is the value of *ith_cnt* of the System maximum value information specified during configuration. "Maximum interrupt source number" is the value of *itf_cnt* of the System maximum value information specified during configuration.
Cyclic handler management table	40	40 * Maximum number of cyclic handlers "Maximum number of cyclic handlers" is the value of *cyc_cnt* of the System maximum value information specified during configuration.
Memory pool management table	24	24 * Maximum number of memory pools "Maximum number of memory pools" is the value of *mpl_cnt* of the System maximum value information specified during configuration.

Object Name	Management Area Size Per Object (in bytes)	Size Calculation Method (in bytes)
Extended SVC handler management table	16	16 * Maximum number of extended SVC handlers "Maximum number of extended SVC handlers" is the value of *svc_cnt* of the System maximum value information specified during configuration.

5.3 Capacity of Task Stack

The task stack area is classified into the following 4 areas :

- Task stack management table
- Context area
- Interrupt stack frame
- Task area

When a task is created (configured), the size of the task stack is specified. This value is the sum of the "interrupt stack frame" and "task area". The size actually reserved on the memory is the size to which the size of the "task stack management table" and "context area" is added (the value of this size is aligned to 4 bytes).

The size of the "task area" varies depending on the user application. However, the size of the "stack management table", "context area", and "interrupt stack frame" is predetermined, as follows.

Table 5-3 Size Used for Task Stack

Task Stack Area	Size (in bytes)	
	for V850E	for V850
Task stack management table	28	
Context area (Interrupt stack frame : 72 bytes)	148	140
Task area	Depends on application	

If the size of the task stack is 100 bytes when a task is created (configured), the stack size actually reserved on the memory is :

< for V850E >
100 + 28 + 148 = 276 bytes

< for V850 >
100 + 28 + 140 = 268 bytes

If the extended SVC handler is started from a task, an area in which registers are saved for handler execution and the stack area consumed by the SVC handler are necessary. The size of these areas is as follows.

Table 5-4 Size of Task Stack Used for Extended SVC Handler

Task Stack Area	Size (in bytes)	
	for V850E	for V850
Register saving area for extended SVC handler	28	20
Extended SVC handler area	Depends on application	

The task stack area is reserved from SPOL0 or SPOL1 when a task is created, and is released when the task is deleted (by del_tsk, exd_tsk, or ter_tsk). If there is a possibility that all tasks could be simultaneously created, therefore, the size of each task must be calculated and the size of SPOL0 and SPOL1 must be determined so that the total size of all the tasks can be reserved. If all tasks are not created at the same time, calculate the maximum size of the combination of tasks that are created at the sometime, and determine the size of SPOL0 and SPOL1 based on this size.

The following table summarizes calculation of the task stack size. The total of all the sizes is the size that must be reserved as a memory area, and the total of the shaded sizes is the task stack size specified when a task is created (configured). Note that the value reserved in the memory area is aligned to 4 bytes.

Table 5-5 Summary of Size Used for Task Stack

Task Stack Area	Size (in bytes)		Remark
	for V850E	for V850	
Task stack management table	28		-
Context area (Interrupt stack frame : 72 bytes)	148	140	-
Task area	Depends on application		Calculate and specify size of stack where tasks are pushed and popped. Take the number of variables used into consideration. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from a task.
Register saving area extended SVC handler	28	20	Unnecessary if the extended SVC handler is not used
Extended SVC handler area	Depends on application		Unnecessary if the extended SVC handler is not used. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from the extended SVC handler.

5.4 Capacity of Stack for Interrupt Handler

The stack area for the interrupt handler is used by the following 4 handlers and task.

- Initialization handler
- Idle task
- Interrupt handler
- Cyclic handler

The size used by each handler or task is explained below.

1) Initialization handler
Reserve an area of the size consumed by functions (for pushing and popping) described as the initialization handler. While the initialization handler is being executed, no task or interrupt handler is started. If the consumed size is less than the area size explained in 2 below, therefore, the stack size consumed by the initialization handler does not have to taken into account.

2) Idle task
A stack area is consumed by an interrupt that occurs before the next task is executed while an idle task is being executed or after a task has been terminated (by issuance of del_tsk, exd_tsk, or ter_tsk). The size of this area is 72 bytes. In some cases, the more stack area may be necessary. In those cases, refer to “3) Interrupt handler” below and the sections that follow.
Because this 72-byte stack area is added during initialization processing, it does not have to be taken into account during configuration. This means that the memory size actually reserved is the stack area for interrupt handlers specified during configuration plus 72 bytes.

3) Interrupt handler
If interrupts occur, a stack area is created in the task stack area or the area for the idle task explained in “2) Idle task“ above when the first interrupt occurs (when a task is interrupted). If there is a possibility that multiple interrupts could later occur, the size of as many interrupt stack frames as the number of the maximum nesting levels must be added.
If an interrupt handler is started, 28 bytes are necessary for the V850E and 20 bytes are necessary for the V850 as a register saving area, in addition to the interrupt stack frame. For example, after the information of the interrupt stack frame has been saved to the task stack, the SP (stack pointer) points to the stack for the interrupt handler, to which further information is saved. In a system that enables multiple interrupts, the size of this stack must be taken into consideration. Therefore, add the value of multiplying "maximum interrupt nesting levels + 1 (for the first interrupt)" by 28 to the size of the interrupt handler stack in the case of the V850E. With the V850, add the value of multiplying the above value by 20.
In addition, take into consideration the case where interrupts are nested to the maximum level and the functions described as an interrupt handler consumes the stack by pushing and popping. Therefore, add 4 bytes to issue a system call in the interrupt handler. To issue the extended SVC handler, add 28 bytes in the case of the V850E and 20 bytes in the case of the V850. Add 4 bytes if a system call is issued in the extended SVC handler.
The clock interrupt handler is treated as an interrupt handler that does not consume the stack by pushing and popping. Calculate the stack size taking this into consideration.

4) Cyclic handler
The cyclic handler is provided as a subroutine that is called by the clock interrupt handler.
If another cyclic handler is started because a new clock interrupt occurs while 1 cyclic handler is being executed, the processing of the cyclic handler already under execution takes precedence. Therefore, add the maximum stack size consumed by the function of all the functions described as a cyclic handler to the size of the handler stack.

The table below summarizes the methods of calculating the size of the stack for interrupt handlers. The total size must be reserved as a memory area, and the total of the shaded sizes is the size for the interrupt handler specified during configuration. Note that the value reserved on the memory area is aligned to 4 bytes.

Table 5-6 Stack Size for Interrupt Handler in System Not Enabling Multiple Interrupts

Interrupt Handler Stack Area	Size (in bytes)		Remarks
	for V850E	for V850	
Idle task area	72		-
Register saving area for interrupt handler	28	20	-
Interupt handler area	Depends on application		Calculate and specify the size of the stack where the interrupt handler pushes and pops. Take the number of variables used into consideration. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from an interrupt handler. Specify the stack size used by the handler (including a cyclic handler) that uses the stack most of all the interrupt handlers used.
Register saving area for extended SVC handler	28	20	Unnecessary if an interrupt handler does not call the extended SVC handler.
Extended SVC handler area	Depends on application		Unnecessary if an interrupt handler does not call the extended SVC handler. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from the extended SVC handler.

This table indicates the stack size used when multiple interrupts are not enabled. If the cyclic handler is interrupted and if that interrupt is acknowledged, this is equivalent to multiple interrupts. In other words, the stack size in this table applies to an application that is executed when rxtmcore.o (version that can acknowledge an interrupt with a higher priority than the timer interrupt in the cyclic handler) is used as the nucleus common object, when an interrupt with a priority higher than that of the timer interrupt is not used, and when all the interrupt handlers are disabled.

Table 5-7 Stack Size for Interrupt Handler in System Enabling Multiple Interrupts

Interrupt Handler Stack Area	Size (in bytes)		Remarks
	for V850E	for V850	
Idle task area	72		-
Interrupt stack frame	72 * n		-
Register saving area for interrupt handler	28 * (n+1)	20 * (n+1)	-
Interupt handler area	Depends on application		Calculate and specify the size of the stack where the interrupt handler pushes and pops. Take the number of variables used into consideration. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from an interrupt handler. Specify the stack size used by the handler (including a cyclic handler) that uses the stack most of all the interrupt handlers used.

Interrupt Handler Stack Area	Size (in bytes)		Remarks
	for V850E	for V850	
Register saving area for extended SVC handler	28 * m	20 * m	Unnecessary if an interrupt handler does not call the extended SVC handler.
Extended SVC handler area	Depends on application		Unnecessary if an interrupt handler does not call the extended SVC handler. Add 4 bytes because the RX850 Pro pushes lp (r31) when a system call is issued from the extended SVC handler.

In the above table, n indicates the maximum number of times interrupts are nested, and m indicates the number of interrupt handlers using the extended SVC handler.

5.5 Memory Pool Capacity

This section explains the capacity of a memory pool. The capacity of a memory pool is specified when the memory pool is created (during configuration). The memory size actually allocated, however, is the value of the specified size with 8 added and aligned to 4 bytes.

Eight bytes are added to the beginning of the memory pool and serve as the management area of the pool. Note, therefore, that the size actually reserved as a memory pool is greater than the specified size.

Table 5-8 Size of Memory Pool

Object Name	Method of Calculating Size (in bytes)
Memory pool	Align4 (Size of memory pool + 8) The value of the memory pool size is the same as the value at the time of the issuance of system call cre_mpl or the value of Memory pool information *mpl_siz*.

5.6 Examples of Estimating Memory Capacity

This section shows examples of estimating the capacity of the memory area used as the management area (SPOL and UPOL) of the RX850 Pro. In these examples, it is assumed that the V850E is used as the CPU, and that the system calls "cre_tsk" and "cre_mpl" are not issued.

< Application information >

Information	Value (in bytes)
Stack area for interrupt handler *intstk_siz*	256 bytes from SPOL0
Task priority range pri_lvl	15
Maximum number of tasks *maxtsk*	2
Maximum number of semaphores *maxsem*	1
Maximum number of event flags *maxflg*	2
Maximum number of mailboxes *maxmbx*	3
Maximum number of interrupt handlers *maxint*	4
Maximum number of memory pools *maxmpl*	2
Maximum number of cyclic handlers *maxcyc*	1
Maximum number of extended SVC handlers *maxsvc*	1
Maximum interrupt source number *maxintfactor*	56
Task stack information	256 bytes from SPOL0 256 bytes from SPOL1
Temory pool information	4096 bytes from UPOL0 8192 bytes from UPOL1

< Estimation method] >

Object Information	Calculation Expression Size (in bytes)
Operating system management table	[from SPOL0] 504 + Align32 (15 + 4) / 8 + Align4 ((15 + 4) * 2) = 548
System memory management table	[from SPOL0] 8 * 4 = 32
Task management table	[from SPOL0] 56 * 2 = 112
Semaphore management table	[from SPOL0] 20 * 1 = 20

Object Information	Calculation Expression Size (in bytes)
EVENT flag management table	[from SPOL0] 20 * 2 = 40
Mailbox management table	[from SPOL0] 20 * 3 = 60
Interrupt handler management table	[from SPOL0] 16 * 4 + Align4 (56) = 120
Memory pool management table	[from SPOL0] 24 * 2 = 48
Cyclic handler management table	[from SPOL0] 40 * 1 = 40
Extended SVC handler management table	[from SPOL0] 16 * 1 = 16
Task stack	[from SPOL0] Align4 (100 + 256) + Align4 (100 + 256) = 712
Interrupt handler stack	[from SPOL0] Align4 (256 + 80) = 336
Memory pool	[from UPOL0] 4096 + 8 = 4104 [from UPOL1] 8192 + 8 = 8200

From the above calculation result, the following capacity is necessary.

SPOL0 : 548 + 32 + 112 + 20 + 40 + 60 + 120 + 48 + 40 + 16 + 712 + 336 = 2084 bytes
SPOL1 : 0 bytes
UPOL0 : 4104 bytes
UPOL1 : 8200 bytes

CHAPTER 6 SYSTEM CONFIGURATION FILE

This chapter explains the system configuration file and how to describe it.

6.1 Outline

To organize a system using the RX850 Pro, various data (such as system information and resource information) and information containing the types of system calls to be used is necessary. The former is called the "system information table file" and the latter is called the "system call table file".

The system information table file and system call table file are described in assembly language and include data enumerated in a specified format. These tables can also be described using various editors, but it is time-consuming to describe them because changing and adding information is very difficult.

Therefore, an application called a configurator (CF850 Pro) is supplied.

This application converts a file in an original format, in which the system and resources of the RX850 Pro and information on the system calls to be used are described, into the system information table file and system call table file.

Therefore, the user can obtain the system information table file and system call table file by using the configurator and by creating a system configuration file.

The configurator outputs 3 files from the system configuration file : "system information table file", "system call table file", and "system information header file". The "system information header file" is a file in which symbol names specified as resource IDs, such as created tasks and semaphores, are made to correspond to the actual ID numbers by using the #define instruction.

For how to start the configurator, see "CHAPTER 7 CONFIGURATOR (CF850 Pro)".

How to describe the system configuration file is explained next.

6.2 Declaration

The following shows how to make the necessary declarations when describing a system configuration file.

1) Elements and character codes

a) Character codes
The system configuration file is described in ASCII code. Kanji character codes (SJIS and EUC only) can be used only in comments.

b) Words
In the system configuration file, any series of characters that does not contain any blank characters (space code, tab code, or line feed code) is called a word. In the following explanations, values, symbol names, and keywords are all words.
The configurator distinguishes between uppercase and lowercase characters. For example, "ABC," "Abc," and "abc" are handled as 3 different words.

c) Statements
In the system configuration file, a series of words delimited by one or more spaces is called a statement. One statement is delimited from another by a line feed.
In the system configuration file, a "\" appearing at the end of a line means that the line is continued on the next line. Note that a "\" must be preceded by a space or tab character.

2) Values
Any word beginning with a numerical code is treated as a value. Values are classified as shown in Table 6-1, according to the numerical code that appears at the beginning.
Unless otherwise specified, any 32-bit width (0x0 to 0xFFFFFFFF) can be specified.

Table 6-1 Types of Values

Type	Numerical Code at Beginning	Characters Used	Example
Decimal	0	0 to 7	0123, 0, 056712
Numerical	other than 0	0 to 9	123, 0, 689525

Type	Numerical Code at Beginning	Characters Used	Example
Octal	0x	0 to 9，a to f (, A to F), x, X	0x12C, 0X0, 0xabcdef

3) Symbol names
Symbol names are distinguished from names according to context. A symbol name indicates the name of a function or variable in a user program.
Note, however, that the first character of a symbol name must be an alphabetic character or an underline.

4) Comments
Within a system configuration file, all text between "--" and the end of the line is handled as a comment.

5) Keywords
The character strings shown below are keywords reserved for use with the configurator.
The use of these character strings for any other purpose is prohibited.

clkhdr	clktim	cyc	defstk	flg	flgsvc
ini	inthdr	intstk	intsvc	maxcyc	maxflg
maxint	maxintfactor	maxmbx	maxmpl	maxpri	maxsem
maxsvc	maxtsk	mbx	mbxsvc	mem	mpl
mplsvc	no_use	prtflg	prtmbx	prtmpl	prtsem
prttsk	RX850PRO	rxsers	sct_def	sem	semsvc
ser_def	sit_def	SPOL0	SPOL1	svc	syssvc
TA_ASM	TA_DISINT	TA_ENAINT	TA_HLNG	TA_MFIFO	TA_MPRI
TA_TFIFO	TA_TPRI	TA_WMUL	TA_WSGL	TCY_OFF	TCY_ON
timsvc	tsk	tsksvc	TTS_DMT	TTS_RDY	UPOL0
UPOL1	V850	V850E1	V850E2	V850ES	

6.3 Configuration Information

The system configuration information that is described in the system configuration file is divided into the following 3 main types.

- Real-time OS information
  Data related to the real-time OS being used.

- SIT information
  Data that is necessary for the operation of the RX850 Pro.

- SCT information
  Data that is used to select whether a system function (system call) is to be used.

6.3.1 Real-time OS information

The real-time OS information that is described in the system configuration file consists of the following 1 item.

1) RX series information
The following data is described as RX series information.

- Real-time OS name
- Version number

6.3.2 SIT information

The SIT information that is described in the system configuration file consists of the following 12 items.

1) System information
The following data is defined as system information.

- Processor type
- Basic clock cycle
- Timer interrupt source number
- Default stack size
- Stack information for interrupt handler
  Stack size
  System memoy type
- Range of protected ID numbers
  Range of protected task ID numbers
  Range of protected semaphore ID numbers
  Range of protected event flag ID numbers
  Range of protected mailbox ID numbers
  Range of protected memory pool ID numbers

2) System maximum value information
The following data is defined as system maximum value information.

- Task priority range
- Maximum number of management objects that can be created or registered
  Maximum number of tasks
  Maximum number of semaphores
  Maximum number of event flags
  Maximum number of mailboxes
  Maximum number of memory pools
  Maximum number of cyclic handlers
  Maximum number of extended SVC handlers
  Maximum number of interrupt handlers
  Maximum interrupt source number

3) System memory information
The following data is defined for each system memory (System Memory Pool 0, System Memory Pool 1, User Memory Pool 0, User Memory Pool 1).

- Type
- Section name
- Size

4) Task information
The following data is defined for each task.

- ID number
- Initial status
- Activation code
- Extended information
- Description language
- Activation address
- Initial priority
- Interrupt mask status
- Stack information for task
  Stack size
  System memory type
- GP register-specific value
- TP register-specific value

- Key ID number

5) Semaphore information
The following data is defined for each semaphore.

- ID number
- Extended information
- Task queuing method
- Initial resource count
- Maximum resource count
- Key ID number

6) Event flag information
The following data is defined for each event flag.

- ID number
- Extended information
- Whether waiting for multiple tasks
- Initial bit pattern
- Key ID number

7) Mailbox information
The following data can be defined for each mailbox.

- ID number
- Extended information
- Task queuing method
- Message queuing method
- Key ID number

8) Interrupt handler information
The following data is defined for each interrupt handler.

- Interrupt source number
- Description language
- Activation address
- GP register-specific value
- TP register-specific value

9) Memory pool information
The following data is defined for each memory pool.

- ID number
- Extended information
- Task queuing method
- Memory pool information
  memory pool size
  type of the system memory to be allocated
- Key ID number

10) Cyclic handler information
The following data is defined for each cyclic handler.

- Specification number
- Extended information
- Description language
- Activation address
- Initial activation status

- Activation interval
- GP register-specific value
- TP register-specific value

11) Extended SVC handler information
The following data is defined for each extended SVC handler.

- Extended function code
- Description language
- Activation address
- GP register-specific value
- TP register-specific value

12) Initialization handler information
The following data is defined for each initialization handler.

- Description language
- Activation address
- GP register-specific value
- TP register-specific value

6.3.3 SCT information

The SCT information that is described in a system configuration file consists of the following 8 items.

1) Task management/task-associated synchronization management function system call information
Defines the task management/task-associated synchronization management function system calls used by a user program as the task management/task-associated synchronization management function system call information.
The task management/task-associated synchronization management function system calls supported by the RX850 Pro are listed below.

cre_tsk	del_tsk	sta_tsk	ext_tsk	exd_tsk	ter_tsk
dis_dsp	ena_dsp	chg_pri	rot_rdq	rel_wai	get_tid
ref_tsk	vget_tid	sus_tsk	rsm_tsk	frsm_tsk	slp_tsk
tslp_tsk	wup_tsk	can_wup			

2) Synchronous communication (semaphore) management function system call information
Defines the semaphore management function system calls used by a user program as the semaphore management function system call information.
The semaphore management function system calls supported by the RX850 Pro are listed below.

cre_sem	del_sem	sig_sem	preq_sem	wai_sem	twai_sem
ref_sem	vget_sid				

3) Synchronous communication (event flag) management function system call information
Defines the event flag management function system calls used by a user program as the event flag management function system call information.
The event flag management function system calls supported by the RX850 Pro are listed below.

cre_flg	del_flg	set_flg	clr_flg	wai_flg	pol_flg
twai_flg	ref_flg	vget_fid			

4) Synchronous communication (mailbox) management function system call information
Defines the mailbox management function system calls used by a user program as the mailbox management function system call information.
The mailbox management function system calls supported by the RX850 Pro are listed below.

cre_mbx	del_mbx	snd_msg	rcv_msg	prcv_msg	trcv_msg
ref_mbx	vget_mid				

5) Interrupt servicing management function system call information
Defines the interrupt processing management function system calls used by a user program as the interrupt

processing management function system call information.
The interrupt processing management function system calls supported by the RX850 Pro are listed below.

def_int	ret_int	ret_wup	ena_int	dis_int	loc_cpu
unl_cpu	chg_icr	ref_icr			

6) Memory pool management function system call information
Defines the memory pool management function system calls used by a user program as memory pool management function system call information.
The memory pool management function system calls supported by the RX850 Pro are listed below.

cre_mpl	del_mpl	get_blk	pget_blk	tget_blk	rel_blk
ref_mpl	vget_pid				

7) Time management function system call information
Defines the time management function system calls used by a user program as the time management function system call information.
The time management function system calls supported by the RX850 Pro are listed below.

set_tim	get_tim	dly_tsk	def_cyc	act_cyc	ref_cyc

8) System management function system call information
Defines the system management function system calls used by a user program as the system management function system call information.
The system management function system calls supported by the RX850 Pro are listed below.

get_ver	ref_sys	def_svc	viss_svc

6.4 Specification Format for Real-Time OS Information

The following describes the format that must be observed when describing the real-time OS information in the system configuration file.

In the following explanation, bold text indicates a reserved word, while italics indicate a value, symbol name, or keyword to be supplied by the user.

6.4.1 RX series information

The RX series information defines values for the real-time OS name, version number.
For the system configuration file, the specification of RX series information is required.
The following shows the RX series information format.

rxsers	*rtos_nam*	*rtos_ver*

The items constituting the RX series information are as follows.

- *rtos_nam*
  Specifies the name of the real-time OS.
  The only keyword that can be specified for *rtos_nam* is “RX850PRO”.

- *rtos_ver*
  Specifies the version number of the real-time OS.
  The only keyword that can be specified for *rtos_ver* is “V32x (x is any numer)”.

6.5 Specification Format for SIT Information

The following describes the format that must be observed when describing the SIT information in the system configuration file.

In the following explanation, bold text indicates a reserved word, while italics indicate a value, symbol name, or keyword to be supplied by the user.

6.5.1 System information

The system information defines values for the processor type, basic clock cycle, timer interrupt source number, default stack size, stack information for interrupt handler, range of protected ID numbers.

For the system configuration file, the specification of the system information is required.

The following shows the system information format.

[**cputype**	*chip_type*]
clktim	*time*
clkhdr	*clk_intno*
defstk	*stk_siz*
intstk	*intstk_siz* : *mem_nam*
prttsk	*tsk_idlmt*
prtsem	*sem_idlmt*
prtflg	*flg_idlmt*
prtmbx	*mbx_idlmt*
prtmpl	*mpl_idlmt*

The items constituting the system information are as follows.

- *chip_type*

  Specifies the processor type of the target device.

  The keywords that can be specified for the processor type are V850, V850E1, V850E2, and V850ES.

V850	:	V850 core
V850E1	:	V850E1 core
V850E2	:	V850E2 core
V850ES	:	V850ES core

  If omitted : The processor type of target device is “V850E1”.

- *time*

  Specifies the basic clock cycle of the timer to be used (in ms).

  A value between 0x1 and 0x7fff can be specified for *time*.

  Note The basic clock cycle is the cycle at which the RX850 Pro generates the clock interrupts necessary to realize the time management function (cycle rise, delay rise, timeout).

- *clk_intno*

  Specifies a timer interrupt source number.

  The values that can be specified as *clk_intno* vary depending on the C compiler package to be used.

  < CA850 version >

  The interrupt source number specified with a device file, or a value calculated using "(exception code - 0x80) / 0x10", can be specified for *clk_int*.

  < GHS compiler version >

  A value calculated using "(exception code - 0x80) / 0x10", can be specified for *clk_intno*.

- *stk_siz*

  Specifies the default stack size (in bytes).

  A value between 0x0 and 0x7ffffffc, aligned to a 4-byte boundary, can be specified for *stk_siz*.

  Note The default stack size is the smallest task stack size that can exist within the system. If, therefore, at system activation, a static task is generated or if an active task is generated as a result of a cre_tsk system call, and a stack size smaller than the default is specified, that specification is ignored and the default size is used.

- *intstk_siz* : *mem_nam*
  Specifies the stack size to be used by a interrupt handler, and the type of the system memory to be allocated to that stack (in bytes).
  A value between 0x0 and 0x7ffffffc, aligned to a 4-byte boundary, can be specified for *intstk_siz*.
  The keywords that can be specified for *mem_nam* are SPOL0 and SPOL1.

  SPOL0 : Allocates the interrupt handler stack to System Memory Pool 0.
  SPOL1 : Allocates the interrupt handler stack to System Memory Pool 1.

- *tsk_idlmt*
  Specifies the range of protected ID numbers when a task is generated with no ID number specified.
  A value between 0x0 and *tsk_cnt* can be specified for *tsk_idlmt*.

  Note When 0x0 is specified for *tsk_idlmt*, no protection is imposed on the ID number. The value defined for the maximum number of tasks that can be created (maxtsk in the System maximum value information) is used for *tsk_cnt*.

- *sem_idlmt*
  Specifies the range of protected ID numbers when a semaphore is generated with no ID number specified.
  A value between 0x0 and *sem_cnt* can be specified for *sem_idlmt*.

  Note When 0x0 is specified for *sem_idlmt*, no protection is imposed on the ID number. The value defined for the maximum number of semaphores that can be created (maxsem in the System maximum value information), is used for *sem_cnt*.

- *flg_idlmt*
  Specifies the range of protected ID numbers when an event flag is generated with no ID number specified.
  A value between 0x0 and *flg_cnt* can be specified for *flg_idlmt*.

  Note When 0x0 is specified for *flg_idlmt*, no protection is imposed on the ID number. The value defined for the maximum number of event flags that can be created (maxflg in the System maximum value information), is used for *flg_cnt*.

- *mbx_idlmt*
  Specifies the range of protected ID numbers when a mailbox is generated with no ID number specified.
  A value between 0x0 and *mbx_cnt* can be specified for *mbx_idlmt*.

  Note When 0x0 is specified for *mbx_idlmt*, no protection is imposed on the ID number. The value defined for the maximum number of mailboxes that can be created (maxmbx in the System maximum value information), is used for *mbx_cnt*.

- *mpl_idlmt*
  Specifies the range of protected ID numbers when a memory pool is generated with no ID number specified.
  A value between 0x0 and *mpl_cnt* can be specified for *mpl_idlmt*.

  Note When 0x0 is specified for *mpl_idlmt*, no protection is imposed on the ID number. The value defined for the maximum number of memory pools that can be created (maxmpl in the System maximum value information), is used for *mpl_cnt*.

6.5.2 System maximum value information

The system maximum value information defines values for the task priority range, maximum number of management objects that can be created or registered.

For the system configuration file, the specification of the system maximum value information is required.

The following shows the system maximum value information format.

maxpri	*pri_lvl*
maxtsk	*tsk_cnt*
maxsem	*sem_cnt*
maxflg	*flg_cnt*
maxmbx	*mbx_cnt*
maxmpl	*mpl_cnt*
maxcyc	*cyc_cnt*
maxsvc	*svc_cnt*
maxint	*ith_cnt*
maxintfactor	*itf_cnt*

The items constituting the system maximum value information are as follows.

- *pri_lvl*
  Specifies the priority range (priority values) for the task.
  A value between 0x1 and 0xfc can be specified for *pri_lvl*.

- *tsk_cnt*
  Specifies the maximum number of tasks that can be created.
  A value between 0x1 and 0x7fff can be specified for *tsk_cnt*.

- *sem_cnt*
  Specifies the maximum number of semaphores that can be created.
  A value between 0x0 and 0x7fff can be specified for *sem_cnt*.

- *flg_cnt*
  Specifies the maximum number of event flags that can be created.
  A value between 0x0 and 0x7fff can be specified for *flg_cnt*.

- *mbx_cnt*
  Specifies the maximum number of mailboxes that can be created.。
  A value between 0x0 and 0x7fff can be specified for *mbx_cnt*.

- *mpl_cnt*
  Specifies the maximum number of memory pools that can be created.
  A value between 0x0 and 0x7fff can be specified for *mpl_cnt*.

- *cyc_cnt*
  Specifies the maximum number of cyclic handlers that can be registered.
  A value between 0x0 and 0x7fff can be specified for *cyc_cnt*.

- *svc_cnt*
  Specifies the maximum number of extended SVC handlers that can be registered.
  A value between 0x0 and 0x7fff can be specified for *svc_cnt*.

- *ith_cnt*
  Specifies the maximum number of interrupt handlers that can be registered.
  A value between 0x0 and (*itf_cnt* + 1) can be specified for *ith_cnt*.

- *itf_cnt*
  Specifies a maximum interrupt source number.
  The values that can be specified as *itf_cnt* are only those calculated by the following formula: (The maximum value of the exception code that is determined by the target processor - 0x80)/0x10.

6.5.3 System memory information

The system memory information defines “the type, section name, and size of the system memory” for each of the following memory blocks: System Memory Pool 0, System Memory Pool 1, User Memory Pool 0, and User Memory Pool 1.

For the system configuration file, the specification of the data for System Memory Pool 0 is required.

Also, for the system configuration file, when the data for User Memory Pool 1 is specified, the data for User Memory Pool 0 is required.

The following shows the system memory information format.

mem	*mem_id*	*sec_nam*	*mem_siz*

The items constituting the system memory information are as follows.

- *mem_id*
  Specifies the type of the system memory.
  The keywords that can be specified for the system memory type are SPOL0, SPOL1, UPOL0, and UPOL1.

  SPOL0 : System Memory Pool 0 is set as the system memory type.
  SPOL1 : System Memory Pool 1 is set as the system memory type.
  UPOL0 : User Memory Pool 0 is set as the system memory type.
  UPOL1 : User Memory Pool 1 is set as the system memory type.

- *sec_nam*
  Specifies the section name of the memory area to which the system memory is allocated.
  The values that can be specified as *sec_nam* are only the “section names (.*sec_nam*) defined in the link directive file, from which ‘.’ is deleted.”

- *mem_siz*
  Specifies the size of the system memory (in bytes).
  A value between 0x100 and 0x7ffffffc, aligned to a 4-byte boundary, can be specified for *mem_siz*.

6.5.4 Task information

The task information defines the ID number, initial status, activation code, extended information, description language, activation address, initial priority, interrupt mask status, GP register-specific value, and TP register-specific value, key ID number for the task.

For the system configuration file, the specification of at least 1 item of task information is required.

The number of task information items that can be specified is defined as being within the range of 1 to the maximum number of tasks that can be registered, *tsk_cnt*, as set in the System maximum value information.

The following shows the task information format.

tsk	*tsk_id*	*sts*	*sta_code*	*ext_inf*	*lang*	\
	sta_adr	*pri*	*intr*	*stk_siz* : *mem_nam*		\
	data	*text*	*key_id*			

The items constituting the task information are as follows.

- *tsk_id*
  Specifies the ID number of the task.
  A value between 0x0 and *tsk_cnt*, or a symbol name, can be specified for *tsk_id*.
  When 0x0 or a symbol name is specified, the configurator automatically assigns an unused ID number between *tsk_idlmt* and *tsk_cnt*.
  The value defined for the task ID number protected range (prttsk in the System information) is set as *tsk_idlmt*.
  The value defined fot the maximum number of tasks that can be registered (maxtsk in the System maximum value information) is set as *tsk_cnt*.

- *sts*
  Specifies the initial status of the task.
  The keywords that can be specified for *sts* are TTS_DMT and TTS_RDY.

  TTS_DMT : The system enters the dormant status upon being activated.
  TTS_RDY : The system enters the ready status upon being activated.

  Note If the initial status of every static task is set to TTS_DMT, it is assumed that there are no active tasks when the system is activated. In this case, an appropriate task must be activated using the initialization handler.

- *sta_code*
  Specifies the activation code of the task.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *sta_code*.

  Note *sta_code* is valid only when TTS_RDY is specified for sts. It is invalid when TTS_DMT is specified for *sts*.

- *ext_inf*
  Specifies the extended information of the task.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant task. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_tsk system call by a processing program (task/non-task).

- *lang*
  Specifies the language used to describe the task.
  The keywords that can be specified for *lang* are TA_HLNG and TA_ASM.

  TA_HLNG : A task is described in C language.
  TA_ASM : A task is described in assembly language.

- *sta_adr*
  Specifies the activation address of the task.
  A value between 0x0 and 0xfffffffe, aligned to a 2-byte boundary, or a symbol name, can be specified for *sta_adr*.

- *pri*
  Specifies the initial priority of the task.
  A value between 0x1 and *pri_lvl* can be specified for *pri*.
  The value defined for the task priority range (maxpri in the System maximum value information) is set as *pri_lvl*.

- *intr*
  Specifies the interrupt status at task activation.
  The keywords that can be specified for *intr* are TA_ENAINT and TA_DISINT.

TA_ENAINT	:	All interrupts are enabled at task activation.
TA_DISINT	:	All interrupts are disabled at task activation.

- *stk_siz* : *mem_nam*
  Specifies the stack size to be used by a task, and the type of the system memory to be allocated to that stack (in bytes).
  A value between 0x0 and 0x7ffffffc, aligned to a 4-byte boundary, can be specified for *stk_siz*.
  The keywords that can be specified for *mem_nam* are SPOL0 and SPOL1.

SPOL0	:	Allocates the task stack to System Memory Pool 0.
SPOL1	:	Allocates the task stack to System Memory Pool 1.

- *data*
  Specifies the GP register-specific value of the task.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *data* and no_use can be specified as a keyword.

no_use	:	A GP register-specific value is not set.

- *text*
  Specifies the TP register-specific value of the task.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *text* and no_use can be specified as a keyword.

no_use	:	A TP register-specific value is not set.

- *key_id*
  Specifies the key ID number of the task.
  A value between 0x0 and 0x7fff can be specified for *key_id*.

  Note When 0x0 is specified for *key_id*, the configurator does not assign a key ID number.

6.5.5 Semaphore information

The semaphore information defines the ID number, extended information, task queuing method, initial resource count, maximum resource count, key ID number for the semaphore.

The number of semaphore information items that can be specified is defined as being within the range of 0 to the maximum number of semaphores that can be registered, *sem_cnt*, as set in the System maximum value information.

The following shows the semaphore information format.

sem	*sem_id* *key_id*	*ext_inf*	*twai_opt*	*init_cnt*	*max_cnt*	\

The items constituting the semaphore information are as follows.

- *sem_id*
  Specifies the ID number of the semaphore.
  A value between 0x0 and *sem_cnt*, or a symbol name, can be specified for *sem_id*.
  When 0x0 or a symbol name is specified, the configurator automatically assigns an unused ID number between *sem_idlmt* and *sem_cnt*.
  The value defined for the semaphore ID number protected range (prtsem in the System information) is set as *sem_idlmt*.
  The value defined fot the maximum number of semaphores that can be registered (maxsem in the System maximum value information) is set as *sem_cnt*.

- *ext_inf*
  Specifies the extended information of the semaphore.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant semaphore. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_sem system call by a processing program (task/non-task).

- *twai_opt*
  Specifies the task queuing method.
  The keywords that can be specified for *twai_opt* are TA_TFIFO and TA_TPRI.

  TA_TFIFO : Tasks are queued in the same order as that in which resource requests are issued.
  TA_TPRI : Tasks are queued according to their priority.

- *init_cnt*
  Specifies the initial resource count of the semaphore.
  A value between 0x0 and *max_cnt* can be specified for *init_cnt*.

- *max_cnt*
  Specifies the maximum resource count of the semaphore.
  A value between 0x1 and 0x7fffffff can be specified for *max_cnt*.

- *key_id*
  Specifies the key ID number of the semaphore.
  A value between 0x0 and 0x7fff can be specified for *key_id*.

  Note When 0x0 is specified for *key_id*, the configurator does not assign a key ID number.

6.5.6 Event flag information

The event flag information defines the ID number, extended information, whether waiting for multiple tasks, initial bit pattern, key ID number for the event flag.

The number of cevent flag information items that can be specified is defined as being within the range of 0 to the maximum number of event flags that can be registered, *flg_cnt*, as set in the System maximum value information.

The following shows the event flag information format.

flg	*flg_id*	*ext_inf*	*twai_opt*	*init_ptn*	*key_id*

The items constituting the event flag information are as follows.

- *fig_id*
  Specifies the ID number of the event flag.
  A value between 0x0 and *flg_cnt*, or a symbol name, can be specified for *flg_id*.
  When 0x0 or a symbol name is specified, the configurator automatically assigns an unused ID number between *flg_idlmt* and *flg_cnt*.
  The value defined for the event flag ID number protected range (prtflg in the System information) is set as *flg_idlmt*.
  The value defined fot the maximum number of event flags that can be registered (maxflg in the System maximum value information) is set as *flg_cnt*.

- *ext_inf*
  Specifies the extended information of the event flag.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant event flag. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_flg system call by a processing program (task/non-task).

- *twai_opt*
  Specifies whether wait for multiple tasks is disabled/enabled.
  The keywords that can be specified for *twai_opt* are TA_WSGL and TA_WMUL.

  TA_WSGL : Wait for multiple tasks is disabled.
  TA_WMUL : Wait for multiple tasks is enabled.

- *init_ptn*
  Specifies the initial bit pattern (32-bit width) of the event flag.
  A value between 0x0 and 0xffffffff can be specified for *init_ptn*.

- *key_id*
  Specifies the key ID number. of the event flag
  A value between 0x0 and 0x7fff can be specified for *key_id*.

  Note When 0x0 is specified for *key_id*, the configurator does not assign a key ID number.

6.5.7 Mailbox information

The mailbox information defines the ID number, extended information, task queuing method, message queuing method, key ID number for the mailbox.

The number of mailbox information items that can be specified is defined as being within the range of 0 to the maximum number of maiboxes that can be registered, *mbx_cnt*, as set in the System maximum value information.

The following shows the mailbox information format.

mbx	*mbx_id*	*ext_inf*	*twai_opt*	*mwai_opt*	*key_id*

The items constituting the mailbox information are as follows.

- *mbx_id*
  Specifies the ID number of the mailbox.
  A value between 0x0 and *mbx_cnt*, or a symbol name, can be specified for *mbx_id*.
  When 0x0 or a symbol name is specified, the configurator automatically assigns an unused ID number between *mbx_idlmt* and *mbx_cnt*.
  The value defined for the mailbox ID number protected range (prtmbx in the System information) is set as *mbx_idlmt*.
  The value defined fot the maximum number of mailboxes that can be registered (maxmbx in the System maximum value information) is set as *mbx_cnt*.

- *ext_inf*
  Specifies the extended information of the mailbox.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant mailbox. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_mbx system call by a processing program (task/non-task).

- *twai_opt*
  Specifies the task queuing method.
  The keywords that can be specified for *twai_opt* are TA_TFIFO and TA_TPRI.

  TA_TFIFO : Tasks are queued in the same order as that in which message receive requests are issued.
  TA_TPRI : Tasks are queued according to their priority.

- *mwai_opt*
  Specifies the message queuing method.
  The keywords that can be specified for *mwai_opt* are TA_MFIFO and TA_MPRI.

  TA_MFIFO : Messages are queued in the same order as that in which messages are issued.
  TA_MPRI : Messages are queued according to their priority.

- *key_id*
  Specifies the key ID number of the mailbox.
  A value between 0x0 and 0x7fff can be specified for *key_id*.

  Note When 0x0 is specified for *key_id*, the configurator does not assign a key ID number.

6.5.8 Interrupt handler information

The interrupt handler information defines the interrupt source number, description language, activation address, GP register-specific value, and TP register-specific value for the interrupt handler.

The number of interrupt handler information items that can be specified is defined as being within the range of 0 to the maximum number of interrupt handlers that can be registered, *ith_cnt*, as set in the System maximum value information.

The following shows the interrupt handler information format.

inthdr	*int_no*	*lang*	*hdr_adr*	*data*	*text*

The items constituting the interrupt handler information are as follows.

- *int_no*
  Specifies the interrupt source number of the interrupt handler.
  The values that can be specified as *int_no* vary depending on the C compiler package to be used.

  < CA850 version >
  The interrupt source number specified with a device file, or a value calculated using "(exception code - 0x80) / 0x10", can be specified for *int_no*.

  < GHS compiler version >
  A value calculated using "(exception code - 0x80) / 0x10", can be specified for *int_no*.

  Note The same interrupt source number cannot be specified for *int_no* and *clk_intno*.
  clk_intno is a value defined in a timer interrupt source number clkhdr of System information.

- *lang*
  Specifies the language used to describe the interrupt handler.
  The keywords that can be specified for *lang* are TA_HLNG and TA_ASM.

  TA_HLNG : A interrupt handler is described in C language.
  TA_ASM : A interrupt handler is described in assembly language.

- *hdr_adr*
  Specifies the activation address of the interrupt handler.
  A value between 0x0 and 0xfffffffe, aligned to a 2-byte boundary, or a symbol name, can be specified for *hdr_adr*.

- *data*
  Specifies the GP register-specific value of the interrupt handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *data* and no_use can be specified as a keyword.

  no_use : A GP register-specific value is not set.

- *text*
  Specifies the TP register-specific value of the interrupt handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *text* and no_use can be specified as a keyword.

  no_use : A TP register-specific value is not set.

6.5.9 Memory pool information

The memory pool information defines the ID number, extended information, task queuing method, memory pool information, key ID number for the memory pool.

The number of memory pool information items that can be specified is defined as being within the range of 0 to the maximum number of memory pools that can be registered, *mpl_cnt*, as set in the System maximum value information.

The following shows the memory pool information format.

mpl	*mpl_id* *key_id*	*ext_inf*	*twai_opt*	*mpl_siz* : *mem_nam*	\

The items constituting the memory pool information are as follows.

- *mpl_id*
  Specifies the ID number of the memory pool.
  A value between 0x0 and *mpl_cnt*, or a symbol name, can be specified for *mpl_id*.
  When 0x0 or a symbol name is specified, the configurator automatically assigns an unused ID number between *mpl_idlmt* and *mpl_cnt*.
  The value defined for the memory pool ID number protected range (prtmpl in the System information) is set as *mpl_idlmt*.
  The value defined fot the maximum number of memory pools that can be registered (maxmpl in the System maximum value information) is set as *mpl_cnt*.

- *ext_inf*
  Specifies the extended information of the memory pool.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant memory pool. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_mpl system call by a processing program (task/non-task).

- *twai_opt*
  Specifies the task queuing method.
  The keywords that can be specified for *twai_opt* are TA_TFIFO and TA_TPRI.

  TA_TFIFO : Tasks are queued in the same order as that in which memory block requests are issued.
  TA_TPRI : Tasks are queued according to their priority.

- *mpl_siz* : *mem_nam*
  Specifies the memory pool size, and the type of the system memory to be allocated to that memory pool (in bytes).
  A value between 0x4 and 0x7ffffffc, aligned to a 4-byte boundary, can be specified for *mpl_siz*.
  The keywords that can be specified for *mem_nam* are UPOL0 and UPOL1.

  UPOL0 : Allocates the memory pool to User Memory Pool 0.
  UPOL1 : Allocates the memory pool to User Memory Pool 1.

- *key_id*
  Specifies the key ID number of the memory pool.
  A value between 0x0 and 0x7fff can be specified for *key_id*.

  Note When 0x0 is specified for *key_id*, the configurator does not assign a key ID number.

6.5.10 Cyclic handler information

The cyclic handler information defines the specification number, extended information, description language, activation address, initial activation status, activation interval, GP register-specific value, and TP register-specific value for the cyclic handler.

The number of cyclic handler information items that can be specified is defined as being within the range of 0 to the maximum number of cyclic handlers that can be registered, *cyc_cnt*, as set in the System maximum value information.

The following shows the cyclic handler information format.

cyc	*cyc_no* *intvl*	*ext_inf* *data*	*lang* *text*	*hdr_adr*	*act*	\

The items constituting the cyclic handler information are as follows.

- *cyc_no*
  Specifies the specification number of the cyclic handler.
  A value between 0x1 and *cyc_cnt*, or a symbol name, can be specified for *cyc_no*.
  When a symbol name is specified, the configurator automatically assigns an unused ID number between 0x1 and *cyc_cnt*.
  The value defined fot the maximum number of cyclic handlers that can be registered (maxcyc in the System maximum value information) is set as *cyc_cnt*.

- *ext_inf*
  Specifies the extended information of the cyclic handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *ext_inf*.

  Note *ext_inf* is provided to enable the specification of user own information for the relevant cyclic handler. The user can specify it as necessary.
  The value specified for *ext_inf* can be dynamically allocated upon the issuance of a ref_cyc system call by a processing program (task/non-task).

- *lang*
  Specifies the language used to describe the cyclic handler.
  The keywords that can be specified for *lang* are TA_HLNG and TA_ASM.

  TA_HLNG : A cyclic handler is described in C language.
  TA_ASM : A cyclic handler is described in assembly language.

- *hdr_adr*
  Specifies the activation address of the cyclic handler.
  A value between 0x0 and 0xfffffffe, aligned to a 2-byte boundary, or a symbol name, can be specified for *hdr_adr*.

- *act*
  Specifies the initial activation status of the cyclic handler.
  The keywords that can be specified for *act* are TCY_ON and TCY_OFF.

  TCY_ON : The system enters the ON status upon being activated.
  TCY_OFF : The system enters the OFF status upon being activated.

- *intvl*
  Specifies the activation interval of the cyclic handler (in basic clock cycle).
  A value between 0x1 and 0xffffffff can be specified for *intvl*.

- *data*
  Specifies the GP register-specific value of the cyclic handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *data* and no_use can be specified as a keyword.

  no_use : A GP register-specific value is not set.

- *text*
  Specifies the TP register-specific value of the cyclic handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *text* and no_use can be specified as a keyword.

no_use : A TP register-specific value is not set.

6.5.11 Extended SVC handler information

The extended SVC handler information defines the extended function code, description language, activation address, GP register-specific value, and TP register-specific value for the extended SVC handler.

The number of extended SVC handler information items that can be specified is defined as being within the range of 0 to the maximum number of extended SVC handlers that can be registered, *svc_cnt*, as set in the System maximum value information.

The following shows the extended SVC handler information format.

svc	*svc_no*	*lang*	*hdr_adr*	*data*	*text*

The items constituting the extended SVC handler information are as follows.

- *svc_no*
  Specifies the extended function code of the extended SVC handler.
  A value between 0x1 and *svc_cnt*, or a symbol name, can be specified for *svc_no*.
  When a symbol name is specified, the configurator automatically assigns an unused ID number between 0x1 and *svc_cnt*.
  The value defined fot the maximum number of extended SVC handlers that can be registered (maxsvc in the System maximum value information) is set as svc_cnt.

- *lang*
  Specifies the language used to describe the extended SVC handler.
  The keywords that can be specified for *lang* are TA_HLNG and TA_ASM.

  TA_HLNG : A extended SVC handler is described in C language.
  TA_ASM : A extended SVC handler is described in assembly language.

- *hdr_adr*
  Specifies the activation address of the extended SVC handler.
  A value between 0x0 and 0xfffffffe, aligned to a 2-byte boundary, or a symbol name, can be specified for *hdr_adr*.

- *data*
  Specifies the GP register-specific value of the extended SVC handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *data* and no_use can be specified as a keyword.

  no_use : A GP register-specific value is not set.

- *text*
  Specifies the TP register-specific value of the extended SVC handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *text* and no_use can be specified as a keyword.

  no_use : A TP register-specific value is not set.

6.5.12 Initialization handler information

The initialization handler information defines the description language, activation address, GP register-specific value, and TP register-specific value for the initialization handler.

Information of the initialization handler can be omitted in the system configuration file. If it is omitted, the RX850 Pro assumes that there is no initialization handler, and continues processing.

The following shows the initialization handler information format.

ini	*lang*	*hdr_adr*	*data*	*text*

The items constituting the initialization handler information are as follows.

- *lang*
  Specifies the language used to describe the initialization handler.
  The keywords that can be specified for *lang* are TA_HLNG and TA_ASM.

  TA_HLNG : A initialization handler is described in C language.
  TA_ASM : A initialization handler is described in assembly language.

- *hdr_adr*
  Specifies the activation address of the initialization handler.
  A value between 0x0 and 0xfffffffe, aligned to a 2-byte boundary, or a symbol name, can be specified for *hdr_adr*.

- *data*
  Specifies the GP register-specific value of the initialization handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *data* and no_use can be specified as a keyword.

  no_use : A GP register-specific value is not set.

- *text*
  Specifies the TP register-specific value of the initialization handler.
  A value between 0x0 and 0xffffffff, or a symbol name, can be specified for *text* and no_use can be specified as a keyword.

  no_use : A TP register-specific value is not set.

6.6 Specification Format for SCT Information

The following describes the format that must be observed when describing the SCT information in the system configuration file.

In the following explanation, bold text indicates a reserved word, while italics indicate a value, symbol name, or keyword to be supplied by the user.

6.6.1 Task management/task-associated synchronization management function system call information

The task management/task-associated synchronization management function system call information defines data that indicates the task management/task-associated synchronization management function system calls used by a user processing program for each system call.

If the task management/task-associated synchronization management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the task management/task-associated synchronization management function system call information format.

tsksvc	*svc_nam*

The items constituting the task management/task-associated synchronization management function system call information are as follows.

- *svc_nam*
  Specifies a task management/task-associated synchronization management function system call name.
  The following keywords can be specified for *svc_nam*.

cre_tsk	del_tsk	sta_tsk	ext_tsk	exd_tsk	ter_tsk
dis_dsp	ena_dsp	chg_pri	rot_rdq	rel_wai	get_tid
ref_tsk	vget_tid	sus_tsk	rsm_tsk	frsm_tsk	slp_tsk
tslp_tsk	wup_tsk	can_wup			

6.6.2 Synchronous communication (semaphore) management function system call information

The synchronous communication (semaphore) management function system call information defines data that indicates the synchronous communication (semaphore) management function system calls used by a user processing program for each system call.

If the synchronous communication (semaphore) management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the synchronous communication (semaphore) management function system call information format.

semsvc	*svc_nam*

The items constituting the synchronous communication (semaphore) management function system call information are as follows.

- *svc_nam*
  Specifies a synchronous communication (semaphore) management function system call name.
  The following keywords can be specified for *svc_nam*.

cre_sem	del_sem	sig_sem	preq_sem	wai_sem	twai_sem
ref_sem	vget_sid				

6.6.3 Synchronous communication (event flag) management function system call information

The synchronous communication (event flag) management function system call information defines data that indicates the synchronous communication (event flag) management function system calls used by a user processing program for each system call.

If the synchronous communication (event flag) management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the synchronous communication (evet flag) management function system call information format.

flgsvc	*svc_nam*

The items constituting the synchronous communication (event flag) management function system call information are as follows.

- *svc_nam*
  Specifies a synchronous communication (event flag) management function system call name.
  The following keywords can be specified for *svc_nam*.

cre_flg	del_flg	set_flg	clr_flg	wai_flg	pol_flg
twai_flg	ref_flg	vget_fid			

6.6.4 Synchronous communication (mailbox) management function system call information

The synchronous communication (mailbox) management function system call information defines data that indicates the synchronous communication (mailbox) management function system calls used by a user processing program for each system call.

If the synchronous communication (mailbox) management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the synchronous communication (mailbox) management function system call information format.

mbxsvc	*svc_nam*

The items constituting the synchronous communication (mailbox) management function system call information are as follows.

- *svc_nam*
  Specifies a synchronous communication (mailbox) management function system call name.
  The following keywords can be specified for *svc_nam*.

  cre_mbx del_mbx snd_msg rcv_msg prcv_msg trcv_msg
  ref_mbx vget_mid

6.6.5 Interrupt servicing management function system call information

The interrupt servicing management function system call information defines data that indicates the interrupt servicing management function system calls used by a user processing program for each system call.

If the interrupt servicing management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the interrupt servicing management function system call information format.

intsvc	*svc_nam*

The items constituting the interrupt servicing management function system call information are as follows.

- *svc_nam*
  Specifies a interrupt servicing management function system call name.
  The following keywords can be specified for *svc_nam*.

def_int	ret_int	ret_wup	ena_int	dis_int	loc_cpu
unl_cpu	chg_icr	ref_icr			

6.6.6 Memory pool management function system call information

The memory pool management function system call information defines data that indicates the memory pool management function system calls used by a user processing program for each system call.

If the memory pool management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the memory pool management function system call information format.

mplsvc	*svc_nam*

The items constituting the memory pool management function system call information are as follows.

- *svc_nam*
  Specifies a memory pool management function system call name.
  The following keywords can be specified for *svc_nam*.

cre_mpl	del_mpl	get_blk	pget_blk	tget_blk	rel_blk
ref_mpl	vget_pid				

6.6.7 Time management function system call information

The time management function system call information defines data that indicates the time management function system calls used by a user processing program for each system call.

If the time management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the time management function system call information format.

timsvc	*svc_nam*

The items constituting the time management function system call information are as follows.

- *svc_nam*
  Specifies a time management function system call name.
  The following keywords can be specified for *svc_nam*.

  set_tim get_tim dly_tsk def_cyc act_cyc ref_cyc

6.6.8 System management function system call information

The system management function system call information defines data that indicates the system management function system calls used by a user processing program for each system call.

If the system management function system call information is not defined and system call is used in an application, E_NOSPT (-17) is returned as the return value of the system call.

The following shows the system management function system call information format.

syssvc	*svc_nam*

The items constituting the system management function system call information are as follows.

- *svc_nam*
  Specifies a system management function system call name.
  The following keywords can be specified for *svc_nam*.

  get_ver ref_sys def_svc viss_svc

6.7 Cautions

In the system configuration file, describe the system configuration information (real-time OS information, SIT information, SCT information) in the following order.

1) Declaration of the start of the Real-time OS information description

2) Real-time OS information description

3) Declaration of the start of the SIT information description

4) SIT information description

5) Declaration of the start of the SCT information description

6) SCT information description

Figure 6-1 illustrates how the system configuration file is described.

Figure 6-1 Describing System Configuration File

```
-- Declaration of the start of the Real-time OS information description
ser_def

-- Real-time OS information description
:
:
:

-- Declaration of the start of the SIT information description
sit_def

-- SIT information description
:
:
:

-- Declaration of the start of the SCT information description
sct_def

-- SCT information description
:
:
:
```

6.8 Description Example

Figure 6-2 show examples of system configuration file description when the V851 is used.
Figure 6-2 is an example of the system configuration file when the CA850 version is used.
In the examples shown in Figure 6-2, the following data is coded :

< Real-time OS information >

1) RX series information
Real-time OS name : RX850PRO
Version number : V320

< SIT information >

1) System information
Processor type : V850E1
Basic clock cycle : 0x1 ms
Timer interrupt source number : 0x20 (INTCM4)
Default stack size : 0x100 bytes
Stack information for interrupt handler : Allocates a memory area for 0x100 bytes, starting from System Memory Pool 0
Range of protected task ID numbers : 0x1
Range of protected semaphore ID numbers : 0x1
Range of protected event flag ID numbers : 0x1
Range of protected mailbox ID numbers : 0x1
Range of protected memory pool ID numbers : 0x1

2) System maximum value information
Task priority range : 0xf
Maximum number of tasks : 0x2
Maximum number of semaphores : 0x1
Maximum number of event flags : 0x2
Maximum number of mailboxes : 0x3
Maximum number of memory pools : 0x2
Maximum number of cyclic handlers : 0x1
Maximum number of extended SVC handlers : 0x1
Maximum number of interrupt handlers : 0x5
Maximum interrupt source number : 0x48

3) System memory information
System Memory Pool 0 : Allocates a memory area for 0x2000 bytes, starting from .syspol0 section
System Memory Pool 1 : Allocates a memory area for 0x1000 bytes, starting from .syspol1 section
User Memory Pool 0 : Allocates a memory area for 0x7000 bytes, starting from .usrpol0_0 section
User Memory Pool 0 : Allocates a memory area for 0x2500 bytes, starting from .usrpol0_1 section
User Memory Pool 1 : Allocates a memory area for 0x1500 bytes, starting from .usrpol1 section

4) Task information
ID number : 0x1
Initial status : ready
Activation code : 0x0
Extended information : 0x0
Description language : Assembly language
Activation address : _task01
Initial priority : 0x8
Interrupt mask status : All interrupts enabled
Stack information for task : Allocates a memory area for 0x100 bytes, starting from System Memory Pool 0
GP register-specific value : Not set
TP register-specific value : Not set
Key ID number : 0x1

ID number : TASK02
Initial status : dormant
Activation code : 0x0
Extended information : 0x0
Description language : C language

Activation address : _task02
Initial priority : 0xf
Interrupt mask status : All interrupts disabled
Stack information for task : Allocates a memory area for 0x100 bytes, starting from System Memory Pool 0
GP register-specific value : Not set
TP register-specific value : Not set
Key ID number : 0x2

5) Semaphore information
ID number : 0x1
Extended information : 0x0
Task queuing method : Same order as that in which resource requests are issued (FIFO)
Initial resource count : 0xff
Maximum resource count : 0xff
Key ID number : 0x1

6) Event flag information
ID number : 0x1
Extended information : 0x0
Whether waiting for multiple tasks : Disable
Initial bit pattern : 0x0
Key ID number : 0x1

7) Mailbox information
ID number : 0x1
Extended information : 0x0
Task queuing method : Same order as that in which message receive requests are issued (FIFO)
Message queuing method : Same order as that in which messages are issued (FIFO)
Key ID number : 0x1

8) Interrupt handler information
Interrupt source number : 0x0 (INTP0)
Description language : Assembly language
Activation address : _inthdr01
GP register-specific value : Not set
TP register-specific value : Not set

9) Memory pool information
ID number : 0x1
Extended information : 0x0
Task queuing method : According to task priority
Memory pool information : Allocates a memory area for 0x2000 bytes, starting from User Memory Pool 0
Key ID number : 0x1

10) Cyclic handler information
Specification number : 0x1
Extended information : 0x0
Description language : C language
Activation address : _cychdr01
Initial activation status : OFF status
Activation interval : 0x100 basic clock cycle
GP register-specific value : Not set
TP register-specific value : Not set

11) Extended SVC handler information
Extended function code : 0x1
Description language : C language
Activation address : _svchdr01
GP register-specific value : Not set
TP register-specific value : Not set

12) Initialization handler information
Description language : C language
Activation address : _varfunc
GP register-specific value : Not set
TP register-specific value : Not set

< SCT information >

13) Task management/task-associated synchronization management function system call information

Define the following as the task management/task-associated synchronization management function system call information used by a user processing program :

sta_tsk exd_tsk

14) Synchronous communication (semaphore) management function system call information

Define the following as the synchronous communication (semaphore) management function system call information used by a user processing program :

sig_sem wai_sem

15) Synchronous communication (event flag) management function system call information

Define the following as the synchronous communication (event flag) management function system call information used by a user processing program :

cre_flg del_flg set_flg wai_flg

16) Synchronous communication (mailbox) management function system call information

Define the following as the synchronous communication (mailbox) management function system call information used by a user processing program :

cre_mbx del_mbx snd_msg rcv_msg

17) Interrupt servicing management function system call information

Define the following as the interrupt servicing management function system call information used by a user processing program :

18) Memory pool management function system call information

Define the following as the memory pool management function system call information used by a user processing program :

cre_mpl del_mpl get_blk rel_blk

19) Time management function system call information

Define the following as the time management function system call information used by a user processing program :

act_cyc ref_cyc

20) System management function system call information

Define the following as the system management function system call information used by a user processing program :

viss_svc

Figure 6-2 Example System Configuration File Description (CA850 version)

```
------------------------------
-- Declaration of the start of the Real-time OS information description
------------------------------
ser_def

------------------------------
-- Real-time OS information description
------------------------------

-- RX series information
rxsers          RX850PRO        V320

------------------------------
--Declaration of the start of the SIT information description
------------------------------
sit_def

------------------------------
-- SIT information description
------------------------------

-- System information
cputype         V850E1
clktim          0x1
clkhdr          INTCM4
defstk          0x100
intstk          0x100 : SPOL0
prtstk          0x1
prtsem          0x1
prtflg          0x1
prtmbx          0x1
prtmpl          0x1

-- System maximum value information
maxpri          0xf
maxtsk          0x2
maxsem          0x1
maxflg          0x2
maxmbx          0x3
maxmpl          0x2
maxcyc          0x1
maxsvc          0x1
maxint          0x5
maxintfactor    0x48

-- System memory information
mem             SPOL0           syspol0         0x2000
mem             SPOL1           syspol1         0x1000
mem             UPOL0           usrpol0_0       0x7000
mem             UPOL0           usrpol0_1       0x2500
mem             UPOL1           usrpol1         0x1500

-- Task information
tsk             0x1             TTS_RDY         0x0             0x0             TA_ASM          \
                _task01         0x8             TA_ENAINT       0x100 : SPOL0   no_use          \
                no_use          0x1
tsk             TASK02          TTS_DMT         0x0             0x0             TA_HLNG         \
                _task02         0xf             TA_DISINT       0x100 : SPOL0   no_use          \
                no_use          0x2

-- Semaphore information
sem             0x1             0x0             TA_TFIFO        0xff            0xff            \
                0x1
```

```
-- Event flag information
flg             0x1             0x0             TA_WSGL         0x0             0x1

-- Mailbox information
mbx             0x1             0x0             TA_TFIFO        TA_MFIFO        0x1

-- Interrupt handler information
inthdr          INTP0           TA_ASM          _inthdr01       no_use          no_use

-- Memory pool information
mpl             0x1             0x0             TA_TPRI         0x2000 : UPOL0          \
                0x1

-- Cyclic handler information
cyc             0x1             0x0             TA_HLNG         _cychdr01       TCY_OFF         \
                0x100           no_use          no_use

-- Extended SVC handler information
svc             0x1             TA_HLNG         _svchdr01       no_use          no_use

-- Initialization handler information
ini             TA_HLNG         _varfunc        no_use          no_use

------------------------------
-- Declaration of the start of the SCT information description
------------------------------
sct_def

------------------------------
-- SCT information description
------------------------------

-- Task management/task-associated synchronization management function system call information
tsksvc          sta_tsk
tsksvc          exd_tsk

-- Synchronous communication (semaphore) management function system call information
semsvc          sig_sem
semsvc          wai_sem

-- Synchronous communication (event flag) management function system call information
flgsvc          cre_flg
flgsvc          del_flg
flgsvc          set_flg
flgsvc          wai_flg

-- Synchronous communication (mailbox) management function system call information
mbxsvc          cre_mbx
mbxsvc          del_mbx
mbxsvc          snd_msg
mbxsvc          rcv_msg

-- Interrupt servicing management function system call information
intsvc          ret_int

-- Memory pool management function system call information
mplsvc          cre_mpl
mplsvc          del_mpl
mplsvc          get_blk
mplsvc          rel_blk

-- Time management function system call information
timsvc          act_cyc
timsvc          ref_cyc
```

```
-- System management function system call information
syssvc          viss_svc
```

CHAPTER 7 CONFIGURATOR (CF850 Pro)

This chapter explains how to activate the configurator (CF850 Pro) and how information files (system information table file, system call table file, system information header file) are created.

7.1 Outline

To build systems (load modules) that use functions provided by the RX850 Pro, the information storing data to be provided for the RX850 Pro is required.

Since information files are basically enumerations of data, it is possible to describe them with various editors.

Information files, however, do not excel in descriptiveness and readability ; therefore substantial time and effort are required when they are described.

To solve this problem, the RX850 Pro provides a utility tool (configurator CF850 Pro) that converts system configuration files which excel in descriptiveness and readability into information files.

The CF850 Pro reads system configuration files as input files, and then outputs information files.

The information files output from the CF850 Pro are explained below.

- System information table file
  An information file that stores data (resource information on the RX850 Pro such as the tasks, semaphores, and event flags) required for the operation of the RX850 Pro.

- System call table file
  The system call table file stores data on types of system calls used in the processing program of the user.

- System information header file
  An information file that stores matching between ID numbers and object names (e.g. task, semaphore, and event flag names) described in system configuration files

Table 7-1 shows the operating environment for the CF850 Pro.

Table 7-1 Operating Environment for CF850 Pro

Host Machine	Operating System
Windows based - IBM PC/AT-compatible machine	Windows 98, Me, NT 4.0, 2000, XP

7.2 Activation Method

7.2.1 Activating from command line

The following is how to activate the CF850 Pro from the command line.
Note that, in the examples below, "C>" indicates the command prompt, and "Δ" indicates pressing of the space key.
The activation options enclosed in "[ ]" can be omitted.

< CA850 version >

```
C> cf850proΔ[ @cmd_file ]Δ[ -cpuΔname ]Δ[ -devpath=path ]Δ[ -iΔsit_file ] [ -cΔsct_file ] [ -dΔh_file ] [ -ni ] [ -nc ]
   [ -nd ] [ -V ] [ -help ] cf_file
```

< GHS compiler version >

```
C> cf850proΔ[ @cmd_file ]Δ[ -iΔsit_file ] [ -cΔsct_file ] [ -dΔh_file ] [ -ni ] [ -nc ] [ -nd ] [ -V ] [ -help ] cf_file
```

The details of each activation option are explained below :

- *@cmd_file*
  Specifies the command file name.

  If omitted : The activation options specified on the command line is valid.

  Notes 1 Specify the command file name "*cmd_file*" within 255 characters including the path name.

  Notes 2 For the details on the command file, see "7.2.3 Command file".

- -cpuΔ*name*
  Specifies type specification names of target devices.

  If omitted : The CF850 Pro does not read device files. Therefore, in system configuration files, definitions using “interrupt factor names specified in device files” cannot be performed.

  Note This activation option can be specified only for the CA850 version.

- -devpath=*path*
  Retrieves the device file corresponding to the target device specified with -cpuΔ*name* from the *path* folder.

  If omitted : The device file is retrieved in the order of the current folder, ..\..\..\dev.

  Note This activation option can be specified only for the CA850 version.

- -iΔ*sit_file*
  Specifies the system information table file name to be output.

  If omitted : The CF850 Pro assumes that the following activation option is specified, and performs processing.

CA850 version	:	-iΔsit.s
GHS compiler version	:	-iΔsit.850

  Notes 1 Specify the output file name "system information table file name : *sit_file*" within 255 characters including the path name.

  Notes 2 When both this activation option and the -ni option are specified at the same time, only that which was input last is effective.

- -cΔ*sct_file*
  Specifies the system call table file name to be output.

  If omitted : The CF850 Pro assumes that the following activation option is specified, and performs processing.

CA850 version	:	-cΔsct.s
GHS compiler version	:	-cΔsct.850

Notes 1 Specify the output file name "system call table file name : *sct_file*" within 255 characters including the path name.

Notes 2 When both this activation option and the -nc option are specified at the same time, only that which was input last is effective.

- -dΔ*h_file*

Specifies the system information header file name to be output.

If omitted : The system changes the extension of the system configuration file name, specified with *cf_file*, to ".h", and outputs the file as the system information header file.

Notes 1 Specify the output file name "system information header file name : *h_file*" within 255 characters including the path name.

Notes 2 When both this activation option and the -nd option are specified at the same time, only that which was input last is effective.

- -ni

Disables output of the system information table file.

If omitted : The CF850 Pro assumes that the following activation option is specified, and performs processing.

CA850 version	:	-iΔsit.s
GHS compiler version	:	-iΔsit.850

Note When both this activation option and the -iΔs*it_file* option are specified at the same time, only that which was input last is effective.

- -nc

Disables output of the system call table file.

If omitted : The CF850 Pro assumes that the following activation option is specified, and performs processing.

CA850 version	:	-cΔsct.s
GHS compiler version	:	-cΔsct.850

Note When both this activation option and the -cΔ*sct_file* option are specified at the same time, only that which was input last is effective.

- -nd

Disables output of the system information header file.

If omitted : The system changes the extension of the system configuration file name, specified with *cf_file*, to ".h", and outputs the file as the system information header file.

Note When both this activation option and the -dΔ*h_file* option are specified at the same time, only that which was input last is effective.

- -V

Outputs version information for the CF850 Pro to the standard output.

If omitted : Version information for the CF850 Pro is not output.

Note Specifying this activation option nullifies all other activation options.

- -help

Outputs the usage of the activation options for the CF850 Pro to the standard output.

If omitted : The usage of the activation options for the CF850 Pro is not output.

Note Specifying this activation option nullifies all other activation options.

- *cf_file*

Specifies the input file name "system configuration file name : *cf_file*" that input to the CF850 Pro.

If omitted : This activation option cannot be omitted.

Note Specify the input file name "*cf_file*" within 255 characters including the path name.

7.2.2 Activating from PM+

In addition, this example below is the setting method for an existing project file.
Below are the methods for setting activation options for the CF850 Pro, with the integrated development environment platform PM+ provided by the CA850.
In addition, this example below is the setting method for an existing project file.

1) Starting PM+
Start the PM+ by clicking the shortcut (default : Windows start menu -> [Program] -> [NEC Electronics Tools] -> [PM+] -> [V*x.xx*] -> [PM+ V*x.xx*]), or double-clicking the executable format file (default : C:\Program Files\NEC Electronics Tools\PM+\V*x.xx*\bin\PMplus.exe).

2) Opening the [Open Workspace]dialog box
Open the [Open Workspace]dialog box after selecting [File] -> [Open Workspace...].

Note For details on the [Open Workspace]dialog box, see "PM+ User's Manual".

3) Specifying a work space file
Select or specify the [Look in], [File name], and [Files of type] areas. Then, click the <Open> button, and specify the workspace file (or the project file) to set the activation option for the CF850 Pro.

4) Opening the [Setting OS] dialog box
Open the [Setting OS] dialog box after selecting [Tool] -> [Select OS...].

Note For details on the [Setting OS] dialog box, see "APPENDIX A WINDOW REFERENCE".

5) Opening the [RX850 Pro Settings] dialog box
Open the [RX850 Pro Settings] dialog box by clicking <OK> button after selecting "RX850 V3.xx" from the list box in [Select OS] area

Note For details on the [RX850 Pro Settings] dialog box, see "APPENDIX A WINDOW REFERENCE".

6) Specifying a system configuration file
Specify the input file name (system configuration file name) with [System Configuration file] area.

7) Specifying a system information table file
After checking [Generate a System Information Table File [-i/-ni]] check box, specify the output file name (system information table file name) with [File name] area.

8) Specifying a system call table file
After checking [Generate a System Call Table File [-c/-nc]] check box, specify the output file name (system call table file name) with [File name)] area.

9) Specifying a system information header file
After checking [Generate System Information Header File [-d/-nd]] check box, specify the output file name (system information header file name) with [File name] area.

10) Specifying a nucleus library
After checking the [Link a Nucleus Library] check box, specify “Nucleus library (librxp.a, librxpm.a, etc.) that is linked when a load module is created (when a system is build)” in the [File name] combo box.

11) Specifying a interface library
After checking the [Link a Interface Library] check box, specify “Interface library (libchp.a, libncp.a, etc.) that is linked when a load module is created (when a system is build)” in the [File name] combo box.

12) Specifying a nucleus common object
After checking the [Link a Nucleus Common Object] check box, specify “Nucleus common object (rxcore.o, rxtm-core.o, etc.) that is linked when a load module is created (when a system is build)” in the [File name] combo box.

13) Checking the activation option
The [Command Line Options] area displays the CF850 Pro activation option format that is specified in processes 6) to 12), which enables explicit checking for whether the results specified in the above processes correspond with the results that are intended.

14) Reflecting the operation results in the project file
Click the < OK > button to cause the above operation results to be reflected in the project file.

7.2.3 Command file

The CF850 Pro performs command file support from the objectives that eliminate specified probable activation option character count restrictions in the command lines.

Description formats of command files are described below.

1) Comment lines
Lines that start with # are treated as comment lines.

2) Activation options
For descriptions of different activation options, insert a space or line-feed code between them.
For descriptions of activation options composed of a parameter and a *-xxx* part such as -cpu, -i, -c, and -d, insert a space or line-feed code between the parameter and the *-xxx* part.

Note When a parameter (whose path name is path) of -devpath has a folder name that contains a space code, the parameter must be enclosed in double quotes.

-devpath="Program Files\DEV"

3) Maximum number of characters
The maximum number of characters that can be written to one single line in command files is 4096.

Figure 7-1 shows a example of command file description for the CA850 version.
In the example of Figure 7-1 below, it is assumed that the following activation options are described.

Target device	:	μPD703000
Reference folder for the device file	:	C:\NECTools32\DEV
System information table file	:	sys.s
System call table file	:	sct.s
System information header file	:	sys.h
System configuration file	:	sys.cf

Figure 7-1 Example of Command File Description (For CA850 Version)

```
# Command File
-cpu
3000
-devpath=C:\NECTools32\DEV
-i
sys.s
-c
sct.s
-d
sys.h
sys.cf
```

7.3 Command Input Examples

Examples of command input for the CF850 for the CA850 version are given below.
In this example, the µPD703000 is used as the target device.

- cf850pro -cpu 3000 -devpath=C:\NECTools32\DEV -i sitfile.s -c sctfile.s -d hfile.h cffile.cf
  This command reads system configuration file “cffile.cf”, which is to be read as an input file, from current folders. It also reads device files support type specification name 3000, which are to be read as input files, from the C:\NECTools32\DEV folder. Then, it outputs system information table file “sitfile.tbl”, system call table file “sctfile.tbl“ and system information header file “hfile.h” .

- cf850pro -cpu 3000 -devpath=C:\NECTools32\DEV -i sitfile.s cffile.cf
  This command reads system configuration file “cffile.cf”, which is to be read as an input file, from current folders. It also reads device files support type specification name 3000, which are to be read as input files, from the C:\NECTools32\DEV folder. Then, it outputs system information table file “sitfile.s”, system call table file “sct.s“ and system information header file “cffile.h” .

- cf850pro -cpu 3000 -devpath=C:\NECTools32\DEV -c sctfile.s cffile.cf
  This command reads system configuration file “cffile.cf”, which is to be read as an input file, from current folders. It also reads device files support type specification name 3000, which are to be read as input files, from the C:\NECTools32\DEV folder. Then, it outputs system information table file “sit.s”, system call table file “sctfile.s“ and system information header file “cffile.h” .

- cf850pro -cpu 3000 -devpath=C:\NECTools32\DEV -d hfile.h cffile.cf
  This command reads system configuration file “cffile.cf”, which is to be read as an input file, from current folders. It also reads device files support type specification name 3000, which are to be read as input files, from the C:\NECTools32\DEV folder. Then, it outputs system information table file “sit.s”, system call table file “sct.s“ and system information header file “hfile.h” .

- cf850pro -cpu 3000 -devpath=C:\NECTools32\DEV cffile.cf
  This command reads system configuration file “cffile.cf”, which is to be read as an input file, from current folders. It also reads device files support type specification name 3000, which are to be read as input files, from the C:\NECTools32\DEV folder. Then, it outputs system information table file “sit.s”, system call table file “sct.s“ and system information header file “cffile.h” .

- cf850pro -V
  This command outputs version information for the CF850 Pro to the standard output.

- cf850pro -help
  This command outputs the usage of the activation options for the CF850 Pro to the standard output.

7.4 Message

Upon the detection of description errors such as "incorrect definition in the system configuration file" during processing, the CF850 Pro generates messages and outputs them to the standard output.

Messages are classified into 3 levels: fatal errors, non-fatal errors, and warning errors. The CF850 Pro adds an alphabetic character representing the error level to the beginning of the message to be output.

F : Fatal errors
If a fatal error occurs, the CF850 Pro outputs a message then stops configuration processing.

Exp. Insufficient memory area.

E : Non-fatal errors
If a non-fatal error occurs, the CF850 Pro outputs a message then stops configuration processing.

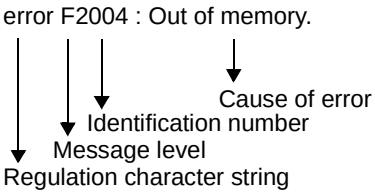
Exp. Duplicated definition.

W : Warnings
If a warning error occurs, the CF850 Pro outputs a message and continues configuration processing.

Exp. The description of a parameter has been omitted.

Figure 7-2 shows the message format.

Figure 7-2 Message Format



The CF850 Pro outputs the following termination message when the processing has been completed.
Note that %d indicates the number of the detected errors and warnings.

total error(s) : %d total warning(s) : %d

7.4.1 Fatal errors

Listed below are the messages that are output in the event of a fatal error.

In the messages, the text in italics (e.g., *file_name*) represents a value that changes depending on the circumstances under which the corresponding fatal error occurs.

F2001 : Usage: cf850pro [@<cfile>] [-cpu <name>] [-devpath=<path>] [-i <SITfile>] [-c <SCTfile>] [-d <includefile>] [-ni] [-nc] [-nd] [-V] [-help] <file>

The start option (CA850 version) specification is invalid.

F2001 : Usage: cf850pro_ghs [@<cfile>] [-i <SITfile>] [-c <SCTfile>] [-d <includefile>] [-ni] [-nc] [-nd] [-V] [-help] <file>

The start option (GHS compiler version) specification is invalid.

F2002 : Can't allocate memory.

There is not enough memory.

F2003 : Can't open file *file_name*.

The file "*file_name*" cannot be opened.

F2004 : Out of memory.

There is not enough memory.

F2005 : Can't open device file.

The device file cannot be opened.

F2006 : Can't read device file.

The device file cannot be read.

F2007 : Unknown device file format.

A device file that is not supported has been specified.

F2008 : Output file "*file_name*" names are the same.

The same name "*file_name*" has been specified for the output files (system information table file and system information header file).

7.4.2 Non-fatal errors

Listed below are the messages that are output in the event of a non-fatal error.

In the messages, the text in italics (e.g., *symbol_name*) represents a value that changes depending on the circumstances under which the corresponding non-fatal error occurs.

E2001 : ser_def not defined.
The real-time OS information start declaration ser_def was not made at the beginning.

E2002 : Illegal ser_def.
The real-time OS information start declaration ser_def is made at an invalid location.

E2003 : ser_def already defined.
The real-time OS information start declaration ser_def is made more than once.

E2004 : sit_def not defined.
The SIT information start declaration sit_def is not defined.

E2005 : Illegal sit_def.
The SIT information start declaration sit_def is made at an invalid location.

E2006 : sit_def already defined.
The SIT information start declaration sit_def is defined more than once.

E2007 : Out of sit_def division.
Data to be contained in SIT information is defined before the SIT information start declaration sit_def.

E2008 : sct_def not defined.
The SCT information start declaration sct_def is not defined.

E2009 : Illegal sct_def.
The SCT information start declaration sct_def is made at an invalid location.

E2010 : sct_def already defined.
The SCT information start declaration sct_def is defined more than once.

E2011 : Out of sct_def division.
Data to be contained in SCT information is defined before SCT information start declaration sct_def.

E2012 : rxsers not defined.
RX series information (rxsers) is not defined.

E2013 : Illegal rxsers.
RX series information (rxsers) is defined at an invalid location.

E2014 : rxsers already defined.
RX series information (rxsers) is defined more than once.

E2101 : Integer overflow.
There is a numeric value that falls outside the 32-bit data range.

E2102 : Syntax error.
The description format of the system configuration file is incorrect.

E2103 : Word too long.
A symbol name is longer than the maximum allowable number of characters.

E2104 : Address out of range.
A value that falls outside the specifiable range is specified as an address.

E2105 : Address must be aligned by 2.
A 2-byte boundary value must be specified as an address.

E2106 : Symbol *symbol_name* already defined.
The symbol name “*symbol_name*“ is defined more than once.

E2107 : Illegal system memorypool for stack.
The type of memory specified for a stack area is invalid.

E2108 : Memory block size out of range.
A system memory size that falls outside the specifiable range is specified.

E2109 : Memory block size must be aligned by 4.
A system memory size other than a 4-byte boundary value is specified.

E2201 : System clock time not defined.
No system clock cycle is defined.

E2202 : System clock time out of range.
A system clock cycle that falls outside the specifiable range is specified.

E2203 : System clock time already defined.
A system clock cycle is defined more than once.

E2204 : Task default stack size not defined.
No default stack size is defined.

E2205 : Task default stack size out of range.
A default stack size that falls outside the specified range is specified.

E2206 : Task default stack size already defined.
A default stack size is defined more than once.

E2207 : System stack size not defined.
The size of a stack (system stack) for interrupt handlers is not defined.

E2208 : System stack size out of range.
The specified size of a stack (system stack) for interrupt handlers falls outside the specifiable range.

E2209 : System stack size already defined.
The size of a stack (system stack) for interrupt handlers is defined more than once.

E2210 : Protect task id not defined.
No task ID number protection range is defined.

E2211 : Protect task id out of range.
A task ID number protection range that falls outside the specifiable range is specified.

E2212 : Protect task id already defined.
A task ID number protection range is defined more than once.

E2213 : Protect task id greater than max task.
The specified task ID number protection range is greater than the maximum number of creatable tasks.

E2214 : Protect semaphore id not defined.
No semaphore ID number protection range is specified.

E2215 : Protect semaphore id out of range.
A semaphore ID number protection range that falls outside the specifiable range is specified.

E2216 : Protect semaphore id already defined.
A semaphore ID number protection range is defined more than once.

E2217 : Protect semaphore id greater than max semaphore.
The specified semaphore ID number protection range is greater than the maximum number of creatable semaphores.

E2218 : Protect eventflag id not defined.
No event flag ID number protection range is defined.

E2219 : Protect eventflag id out of range.
An event flag ID number protection range that falls outside the specifiable range is specified.

E2220 : Protect eventflag id already defined.
An event flag ID number protection range is defined more than once.

E2221 : Protect eventflag id greater than max eventflag.
The specified event flag ID number protection range is greater than the maximum number of creatable event flags.

E2222 : Protect mailbox id not defined.
No mailbox ID number protection range is defined.

E2223 : Protect mailbox id out of range.
A mailbox ID number protection range that falls outside the specifiable range is specified.

E2224 : Protect mailbox id already defined.
A mailbox ID number protection range is defined more than once.

E2225 : Protect mailbox id greater than max mailbox.
The specified mailbox ID number protection range is greater than the maximum number of creatable mailboxes.

E2226 : Protect memorypool id not defined.
No memory pool ID number protection range is defined.

E2227 : Protect memorypool id out of range.
A memory pool ID number protection range that falls outside the specifiable range is specified.

E2228 : Protect memorypool id already defined.
A memory pool ID number protection range is defined more than once.

E2229 : Protect memorypool id greater than max memorypool.
The specified memory pool ID number protection range is greater than the maximum number of creatable memory pools.

E2230 : Max priority level not defined.
No task priority range is specified.

E2231 : Max priority level out of range.
A task priority range that falls outside the specifiable range is specified.

E2232 : Max priority level already defined.
A task priority range is defined more than once.

E2233 : Max task not defined.
The maximum number of creatable tasks is not defined.

E2234 : Max task out of range.
The specified maximum number of creatable tasks falls outside the specifiable range.

E2235 : Max task already defined.
The maximum number of creatable tasks is defined more than once.

E2236 : Max semaphore not defined.
The maximum number of creatable semaphores is not defined.

E2237 : Max semaphore out of range.
The specified maximum number of creatable semaphores is falls outside the specifiable range.

E2238 : Max semaphore already defined.
The maximum number of creatable semaphores is defined more than once.

E2239 : Max eventflag not defined.
The maximum number of creatable event flags is not defined.

E2240 : Max eventflag out of range.
The specified maximum number of creatable event flags falls outside the specifiable range.

E2241 : Max eventflag already defined.
The maximum number of creatable event flags is defined more than once.

E2242 : Max mailbox not defined.
The maximum number of creatable mailboxes is not defined.

E2243 : Max mailbox out of range.
The specified maximum number of creatable mailboxes falls outside the specifiable range.

E2244 : Max mailbox already defined.
The maximum number of creatable mailboxes is defined more than once.

E2245 : Max memorypool not defined.
The maximum number of creatable memory pools is not defined.

E2246 : Max memorypool out of range.
The specified maximum number of creatable memory pools falls outside the specifiable range.

E2247 : Max memorypool already defined.
The maximum number of creatable memory pools is defined more than once.

E2248 : Max cyclic handler not defined.
The maximum number of registerable cyclic handlers is not defined.

E2249 : Max cyclic handler out of range.
The specified maximum number of registerable cyclic handlers falls outside the specifiable range.

E2250 : Max cyclic handler already defined.
The maximum number of registerable cyclic handlers is defined more than once.

E2251 : Max svc handler not defined.
The maximum number of registerable extended SVC handlers is not defined.

E2252 : Max svc handler out of range.
The specified maximum number of registerable extended SVC handlers falls outside the specifiable range.

E2253 : Max svc handler already defined.
The maximum number of registerable extended SVC handlers is defined more than once.

E2254 : System memorypool "*mem_id*" not defined.
The system memory pool name "*mem_id*" is not defined.

E2255 : System memorypool id out of range.
A system memory type that falls outside the specifiable range is specified.

E2256 : Illegal system memorypool id.
A specified system memory type is invalid.

E2257 : Memory section "*sec_nam*" already defined.
The section name “*sec_nam*” of the memory area to which the system memory is allocated is already defined.

E2258 : Memory block address must be symbol.
The section name of the memory area to which the system memory is allocated is illegal.

E2259 : Not enough system memorypool "*mem_id*" block size.
A size that is not sufficient for allocating the management objects, stacks, or memory pools is specified for system memory pool name “*mem_id*“. Alternatively, system memory pool name “*mem_id*” is divided into small noncontiguous areas, so that a size required for management objects, stacks, or memory pools cannot be allocated.

E2260 : System memorypool size exceeds 4Gbytes.
The total system memory size exceeds 4 GBs.

E2261 : Memory block overlap.
System memory areas overlap one another.

E2262 : Task not defined.
Task information is not defined.

E2263 : Task id "*tsk_id*" already defined.
The task ID number "*tsk_id*" is defined more than once.

E2264 : Non-protect task id all assigned.
All task ID numbers that can be automatically assigned are already being used.

E2265 : Too many tasks.
The Task information count exceeds the maximum number of creatable tasks.

E2266 : Task id out of range.
A task ID number that falls outside the specifiable range is specified.

E2267 : Task id greater than max task.
The specified task ID number is greater than the maximum number of creatable tasks.

E2268 : Task priority greater than max priority.
The specified initial task priority level is greater than the specifiable task priority range.

E2269 : Task priority out of range.
An initial task priority level that falls outside the specifiable range is specified.

E2270 : Task stack size out of range.
The specified size of a stack for tasks falls outside the specifiable range.

E2271 : Task key-id out of range.
A task key ID that falls outside the specifiable range is specified.

E2272 : Task key-id "*key_id*" already defined.
The task key ID "*key_id*" is defined more than once.

E2273 : Semaphore id "*sem_id*" already defined.
The semaphore ID number "*sem_id*" is defined more than once.

E2274 : Non-protect semaphore id all assigned.
All semaphore ID numbers that can be automatically assigned are already being used.

E2275 : Too many semaphores.
The Semaphore information count exceeds the maximum number of creatable semaphores.

E2276 : Semaphore id out of range.
A semaphore ID number that falls outside the specifiable range is specified.

E2277 : Semaphore id greater than max semaphore.
The specified semaphore ID number is greater than the maximum number of creatable semaphores.

E2278 : Initial resource count out of range.
The specified number of initial semaphore resources falls outside the specifiable range.

E2279 : Max resource count out of range.
The specified maximum number of semaphore resources falls outside the specifiable range.

E2280 : Initial resource count greater than max resource count.
The specified maximum number of initial semaphore resources is greater than the maximum number of semaphore resources.

E2281 : Semaphore key-id out of range.
A semaphore key ID that falls outside the specifiable range is specified.

E2282 : Semaphore id is 0, but semaphore key-id not specified.
Neither a symbol name nor a key ID is specified for an automatic ID generation semaphore.

E2283 : Semaphore key-id "*key_id*" already defined.
The semaphore key ID number "*key_id*" is defined more than once.

E2284 : Eventflag id "*flg_id*" already defined.
The event flag ID number "*flg_id*" is defined more than once.

E2285 : Non-protect eventflag id all assigned.
All event flag ID numbers that can be automatically assigned are already being used.

E2286 : Too many eventflags.
The Event flag information count exceeds the maximum number of creatable event flags.

E2287 : Eventflag id out of range.
An event flag ID number that falls outside the specifiable range is specified.

E2288 : Eventflag id greater than max eventflag.
The specified event flag ID number is greater than the maximum number of creatable event flags.

E2289 : Initial pattern out of range.
An initial event flag bit pattern that falls outside the specifiable range is specified.

E2290 : Eventflag key-id out of range.
An event flag key ID that falls outside the specifiable range is specified.

E2291 : Eventflag id is 0, but eventflag key-id not specified.
Neither a symbol name nor a key ID is specified for an automatic ID generation event flag.

E2292 : Eventflag key-id "*key_id*" already defined.
The event flag key ID number "*key_id*" is defined more than once.

E2293 : Mailbox id "*mbx_id*" already defined.
The mailbox key ID number "*mbx_id*" is defined more than once.

E2294 : Non-protect mailbox id all assigned.
All mailbox ID numbers that can be automatically assigned are already being used.

E2295 : Too many mailboxes.
The Mailbox information count exceeds the maximum number of creatable mailboxes.

E2296 : Mailbox id out of range.
A mailbox ID number that falls outside the specifiable range is specified.

E2297 : Mailbox id greater than max mailbox.
The specified mailbox ID number is greater than the maximum number of creatable mailboxes.

E2298 : Mailbox key-id out of range.
A mailbox key ID that falls outside the specifiable range is specified.

E2299 : Mailbox id is 0, but mailbox key-id not specified.
Neither a symbol name nor a key ID is specified for an automatic ID generation mailbox.

E2300 : Mailbox key-id "*key_id*" already defined.
The mailbox key ID number "*key_id*" is defined more than once.

E2301 : Memorypool id "*mpl_id*" already defined.
The memory pool ID number "*mpl_id*" is defined more than once.

E2302 : Non-protect memorypool id all assigned.
All memory pool ID numbers that can be automatically assigned are already being used.

E2303 : Too many memorypools.
The Memory pool information count exceeds the maximum number of creatable memory pools.

E2304 : Memorypool id out of range.
A memory pool ID number that falls outside the specifiable range is specified.

E2305 : Memorypool id greater than max memorypool.
The specified memory pool ID number is greater than the maximum number of creatable memory pools.

E2306 : Illegal system memorypool for memorypool.
The type of system memory is invalid.

E2307 : Memorypool key-id out of range.
A memory pool key ID that falls outside the specifiable range is specified.

E2308 : Memorypool id is 0, but memorypool key-id not specified.
Neither a symbol name nor a key ID is specified for an automatic ID generation memory pool.

E2309 : Memorypool key-id "*key_id*" already defined.
The memory pool key ID number "*key_id*" is defined more than once.

E2310 : Interrupt handler number "*int_no*" already defined.
The interrupt handler interrupt source "*int_no*" is defined more than once.

E2311 : Interrupt handler number out of range.
An interrupt handler interrupt source that falls outside the specifiable range is specified.

E2312 : Cyclic handler number "*cyc_no*" already defined.
The cyclic handler specification number "*cyc_no*" is defined more than once.

E2313 : Too many cyclic handlers.
The Cyclic handler information count exceeds the maximum number of registerable cyclic handlers.

E2314 : Cyclic handler number out of range.
A cyclic handler specification number that falls outside the specifiable range is specified.

E2315 : Cyclic handler number greater than max cyclic handler.
The specified cyclic handler specification number is greater than the maximum number of registerable cyclic handlers.

E2316 : Interval time out of range.
A cyclic handler start time interval that falls outside the specifiable range is specified.

E2317 : Svc handler number "*svc_no*" already defined.
Extended SVC handler specification number "*svc_no*" is defined more than once.

E2318 : Too many svc handlers.
The Extended SVC handler information count exceeds the maximum number of registerable extended SVC handlers.

E2319 : Svc handler number out of range.
The specified extended SVC handler extension function code falls outside the specifiable range.

E2320 : Svc handler number greater than max svc handler.
The specified extended SVC handler extension function code is greater than the maximum number of registerable extended SVC handlers.

E2321 : Initial handler not defined.
Initialization handler information is not defined.

E2322 : Initial handler already defined.
An initial handler is defined more than once.

E2323 : Illegal system call name.
A system call name is invalid, or the use of a system call of another group is declared.

E2324 : Max interrupt handler not defined.
The maximum number of registerable interrupt handlers is not defined.

E2325 : Max interrupt handler out of range.
The specified maximum number of registerable interrupt handlers falls outside the specifiable range.

E2326 : Max interrupt handler already defined.
The maximum number of registerable interrupt handlers is defined more than once.

E2327 : Max interrupt handler greater than (max interrupt factor + 1).
The specified maximum number of registerable interrupt handlers is greater than the maximum interrupt source number plus 1.

E2328 : Too many interrupt handlers.
The Interrupt handler information count exceeds the maximum number of registerable interrupt handlers.

E2329 : Interrupt factor is already assigned by clkhdr.
The interrupt source number assigned to an interrupt handler is already specified as a timer interrupt source number.

E2330 : Max interrupt factor not defined.
No maximum interrupt source number is defined.

E2331 : Max interrupt factor out of range.
The maximum number of registerable interrupt handlers falls outside the specifiable range.

E2332 : Max interrupt factor already defined.
A maximum interrupt source number is defined more than once.

E2333 : Clock handler number not defined.
No clock interrupt source is defined.

E2334 : Clock handler number out of range.
A clock interrupt source that falls outside the specifiable range is specified.

E2335 : Clock handler number already defined.
A clock interrupt source is defined more than once.

E2336 : "*chip_type*" cannot define.
The processor type specified in System information is illegal.

E2337 : CPU type already defined.
The processor type of target device is already defined.

7.4.3 Warnings

Listed below are the messages that are output as warnings.

In the messages, the text in italics (e.g., *name*) represents a value that changes depending on the circumstances under which the warning is issued.

W2201 : Task id is 0, but task key-id not specified.
A value 0x0 is specified as the ID number and key ID number of the task.
The CF850 Pro automatically assigns unused ID numbers to a range from *tsk_idlmt* (task ID number protection range) to *tsk_cnt* (the maximum number of creatable tasks).

W2202 : System call "*svc_nam*" already defined.
svc_nam is already defined in the system call declaration used in the processing program for the user that exists in the SCT information.
The CF850 Pro keeps on performing processing, ignoring the fact that *svc_nam* is already defined.

W2203 : Cannot open command file "*cmd_file*".
The command file "*cmd_file*" cannot be opened.
The CF850 Pro assumes the specified @*cmd_file* to be undefined, and continues processing.

W2204 : Nested command file "*file_name*".
An incorrect activation option?@*cmd_file* is specified with the command file "*file_name*".
The CF850 Pro assumes the specified @*cmd_file* to be undefined, and continues processing.

W2205 : cputype in CF file is different device file. (device file assumed)
The type specification name specified in an activation option -cpu∆*name* is not matched to the processor type defined in System information.
The CF850 Pro keeps on performing processing, assuming that the activation option -cpu∆*name* is valid information.

APPENDIX A WINDOW REFERENCE

This appendix explains the dialog boxes that are used when the activation option for the CF850 Pro is specified from the integrated development environment platform PM+ provided by the CA850.

A.1 Outline

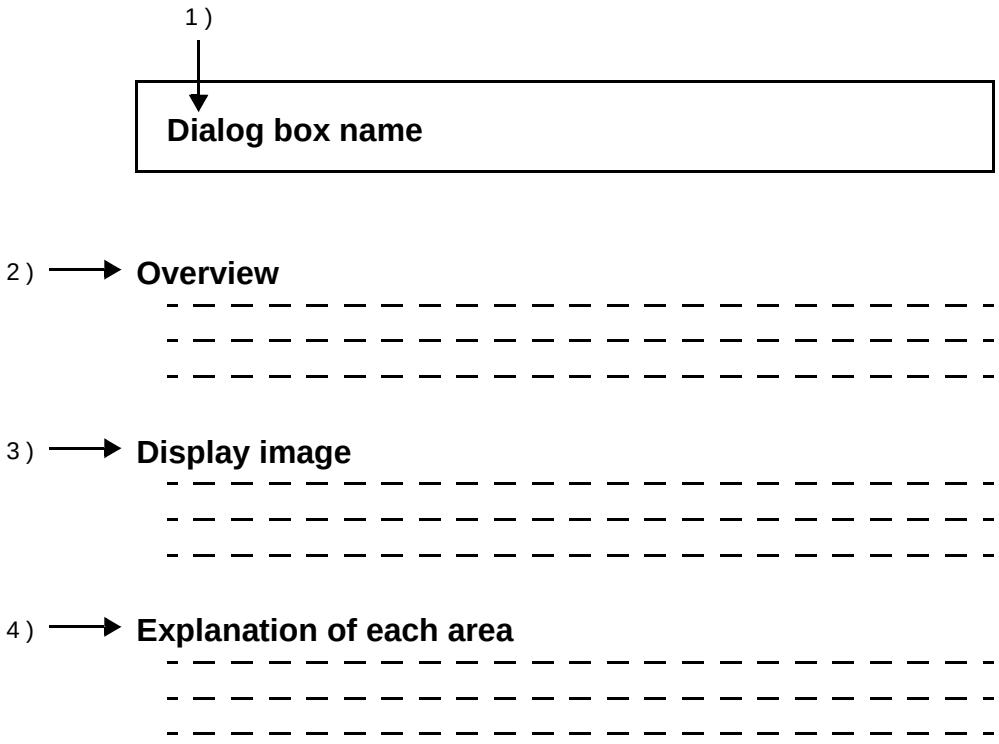
Table A-1 shows the list of dialog boxes.

Table A-1 List of Dialog Boxes

Dialog Box Name	Functional Outline
[Setting OS] dialog box	This dialog box is used to specify the following information. - Real-time OS name
[RX850 Pro Settings] dialog box	This dialog box is used to set the information items below as activation options for the CF850 Pro and to notify the integrated development environment platform PM+ about the settings. - System configuration file name - System information table file name - System call table file name - System information header file name - Nucleus library file name - Interface library file name - Nucleus common object file name
[Select System Configuration File] dialog box	This dialog box is used to load a existing system configuration file.
[RX850 Pro ERROR] dialog box	This dialog box is used to display error information.

A.2 Explanation of Dialog boxes

This section explains each dialog box according to the following description format.



1) Dialog box name
Shown in the frame are the dialog box name.

2) Overview
The functional outline and how to open the dialog box are also explained.

3) Display image
The display image is also explained.

4) Explanation of each area
Describes the detailed functions according to a structural element.

[Setting OS] dialog box

Overview

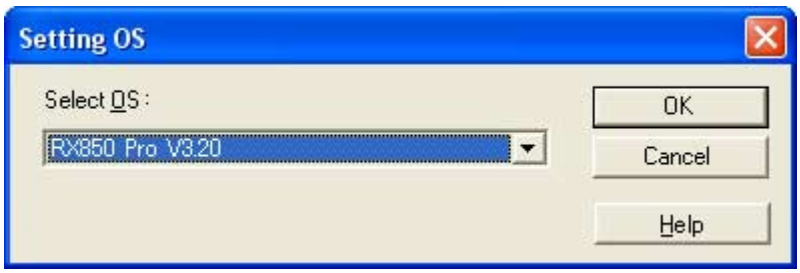
This dialog box is used to specify the following information :

- Real-time OS name

This dialog box can be opened as follows :

- Select [Tool] menu -> [Select OS...] on the main window of PM+.

Display image



Explanation of each area

1) [Select OS] area

- List box
  Select the name of the real-time OS.
  The only menu that can be specified for the name is “RX850 Pro V3.2x (x is any numer)”.

2) Function buttons

- < OK > button
  Opens the [RX850 Pro Settings] dialog box.
- < Cancel > button
  Closes this dialog box.
- < Help > button
  Opens the help for this dialog box.

[RX850 Pro Settings] dialog box

Overview

This dialog box is used to set the information items below as activation options for the CF850 Pro and to notify the integrated development environment platform PM+ about the settings.

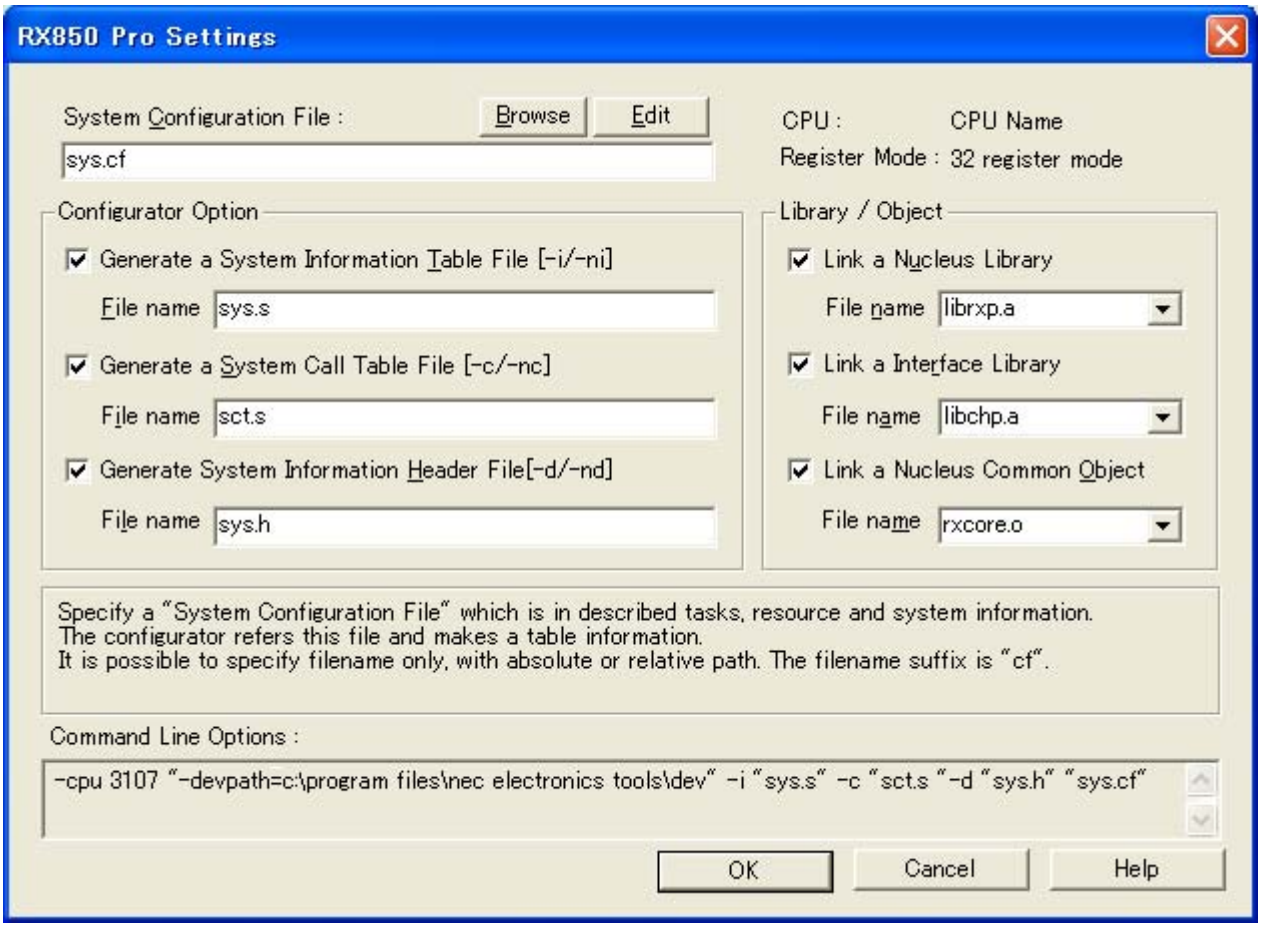
- System configuration file name
- System information table file name
- System call table file name
- System information header file name
- Nucleus library name
- Interface library name
- Nucleus common object name

This dialog box can be opened as follows :

- After selecting "RX850 Pro V3.*xx*" in the [Select OS] area of the [Setting OS] dialog box, click the < OK > button.

Note For type specification names and folder names to be searched from the device file, information specified in the [Project Settings] dialog box of the integrated development environment platform PM+ provided by the CA850 is reflected. Therefore, the aforementioned names need not be specified in this dialog box.
For details of the [Project Settings] dialog box, refer to "PM+ user's manual".

Display image



Explanation of each area

1) [System Configuration file] area

- Text box
  Specifies the file (system configuration file name) input to the configurator.

  Note Specify the input file name within 255 characters including the path name.

- < Browse > button
  Opens the [Select System Configuration File] dialog box.

- < Edit > button
  Opens the [Edit] window.

  Notes 1 For details of the [Edit] window, refer to "PM+ user's manual".

  Notes 2 [Setting OS] dialog boxThis dialog box and the [Setting OS] dialog box must be closed when the contents of system configuration file which is displayed on the [Edit] window is edited by using the editor "ideal-L" provided with the PM+.

2) [Configurator Option] area

- [Generate a System Information Table File [-i/-ni]] check box
  Specify whether the system information table file is output or not while the CF850 Pro is activated.

  Checked : Outputs the system information table file with the file name specified in [File name] area.
  Not checked : Disables output of the system information table file.

- [File name] area
  Specify the output file name (system information table file name) while the CF850 Pro is activated.

  Note Specify the output file name within 255 characters including the path name.

- [Generate a System Call Table File [-c/-nc]] check box
  Specify whether the system call table file is output or not while the CF850 Pro is activated.

  Checked : Outputs the system call table file with the file name specified in [File name] area.
  Not checked : Disables output of the system call table file.

- [File name)] area
  Specify the output file name (system call table file name) while the CF850 Pro is activated.

  Note Specify the output file name within 255 characters including the path name.

- [Generate System Information Header File [-d/-nd]] check box
  Specify whether the system information header file is output or not while the CF850 Pro is activated.

  Checked : Outputs the system information header file with the file name specified in [File name] area.
  Not checked : Disables output of the system information header file.

- [File name] area
  Specify the output file name (system information header file name) while the CF850 Pro is activated.

  Note Specify the output file name within 255 characters including the path name.

3) [Library / Object] area

- [Link a Nucleus Library] check box
  Specify whether the nucleus library specified in the [Fine name] combo box, as "a link option for linker Id850", is notified to the PM+ while the CF850 Pro is activated.

- [Fine name] combo box
  Specify the output file name (nucleus library) while the CF850 Pro is activated.

- [Link a Interface Library] check box
  Specify whether the interface library specified in the [File name] combo box, as "a link option for linker Id850", is notified to the PM+ while the CF850 Pro is activated.

- [File name] combo box
  Specify the output file name (interface library) while the CF850 Pro is activated.

- [Link a Nucleus Common Object] check box
  Specify whether the nucleus common object specified in the [File name] combo box, as “a link option for linker Id850”, is notified to the PM+ while the CF850 Pro is activated.

- [File name] combo box
  Specify the output file name (nucleus common object) while the CF850 Pro is activated.

4) [Command Line Options] area
Displays information specified in the [System Configuration File] and [Configurator Option] areas (including information specified in the [Project Settings] dialog box of the integrated development environment platform PM+), as a command input format for the CF850 Pro.

5) Function buttons

- < OK > button
  Notify the PM+ of the information specified in the [System Configuration File] and [Configurator Option] areas (including information specified in the [Project Settings] dialog box of the integrated development environment platform PM+), as “an activation option for the CF850 Pro”. Also, notify the PM+ of the information specified in the [Library / Object] area, as a “link option for linker Id850”. Then, close this dialog box.

- < Cancel > button
  Closes this dialog box.

- < Help > button
  Opens the help for this dialog box.

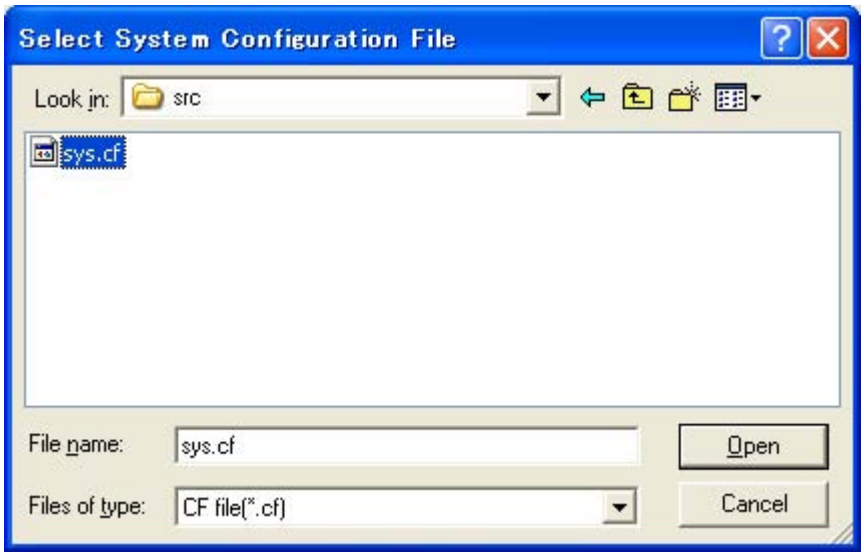
[Select System Configuration File] dialog box

Overview

This dialog box is used to load a existing system configuration file.
This dialog box can be opened as follows :

- Click the < Browse > button on the [RX850 Pro Settings] dialog box.

Display image



Explanation of each area

1) [Look in] area

- Combo box
  Select the folder in which the system configuration file is stored.

2) File name display area
This area displays the list of display files.

3) [File name] area

- Text box
  Specify the name of the file to be opened.

4) [Files of type] area

- Combo box
  Select the type of the file to be opened.

5) Function buttons

- < Open > button
  Loads the system configuration file specified with [Look in] area and [File name] area.
- < Cancel > button
  Closes this dialog box.

[RX850 Pro ERROR] dialog box

Overview

This dialog box is used to display error information.
This dialog box is automatically opened when the incorrect information has been set in the [RX850 Pro Settings] dialog box, and so on.

Display image

Explanation of each area

1) Error message display area
Displays a message (error number, cause of error) corresponding to the detected error.
The messages displayed in this area are listed below.

Error Number	Cause of Error
E1006 :	File name too long.
E1009 :	This file name already exists. Specify another file name.
E1010 :	System information table file name and system call table file name is same. Specify another file name.
E1012 :	Nucleus library file name and interface library file name is same. Specify another file name.
E1022 :	System configuration file name or path name is wrong.
E1023 :	System information table file name or path name is wrong.
E1024 :	System information header file name or path name is wrong.
E1025 :	System call table file name or path name is wrong.
E1027 :	Nucleus library file name is wrong.
E1028 :	Interface library file name is wrong.
E1029 :	Nucleus common object file name is wrong.
E2011 :	System configuration file name is not exist.
E2012 :	System information table file name is not exist.
E2013 :	System call table file name is not exist.
E2010 :	System information header file name is not exist.
E2016 :	Nucleus library file name is not exist.
E2017 :	Interface library file name is not exist.
E2018 :	Nucleus common object file name is not exist.

Error Number	Cause of Error
E2030 :	32 register mode is not specified. Changes to 32 register mode in case of using RX850 Pro.
E2040 :	Online help file is not exist.

2) Function buttons

- < OK > button
  Closes this dialog box.

INDEX

N

P

R

S

T

U

W