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.

R8C Family

Debugging with MR8C/4

Introduction

High-performance Embedded Workshop is equipped with RTOS Graphical User Interface; MR Window intended to ease the development of applications incorporating MR8C/4.

Testing and debugging of program constitutes a significant amount of the development time. Debugging time itself can account for up to 50% of the total time required for software development.

This document explains the powerful yet intuitive way of debugging application with MR8C/4 using MR Window. This document illustrates with Renesas Starter Kit for R8C/21256 and "MR8C_4_RTOS" sample program.

Target Device

Applicable MCU: R8C/2x

Contents

1. Guide in using this Document 2

2. Overview of Debugging Techniques 3

3. Preparation for Debugging 4

4. Understanding System Down Routine 5

5. Understanding MR Window through Practical Hands-On 7

6. Exercise: Debugging "Watch_RSKR8C_DemoOS" Program 10

7. Troubleshooting FAQ 12

8. Reference Documents 14

9. Appendix I: MR Window Overview 15

10. Appendix II: Solution to Exercise: Debugging "Watch_RSKR8C_DemoOS" Program 24

1. Guide in using this Document

This document aims to provide a startup for users with basic knowledge of debugging MR8C/4 by familiarizing them with MR Window feature in High-performance Embedded Workshop with practical hands-on.

With sample program “MR8C_4_RTOS”; users will have a complete understanding the MR Window. Couple with an exercise program “Watch_RSKR8C_DemoOS”, it will help users to strengthen their skills in debugging MR8C/4.

Table 1 Explanation of Document Topics

Topic	Objective	Pre-requisite
Overview of Debugging Techniques	Present an overview of the debugging procedure	None
Preparation for Debugging	Describes the setup procedures necessary for performing the debugging	Knowledge in High-performance Embedded Workshop and E8a Emulator
Understanding System Down Routine	Discuss on the method of implementing System Down Routine used for debugging.	C Compiler Package for M16C Series and R8C Family
Understanding MR Window through Practical Hands-On	Guides to having a practical feel of using MR Window.	Knowledge in High-performance Embedded Workshop and MR8C/4
Exercise: Debugging “Watch_RSKR8C_DemoOS” Program	A practical exercise for users to access and enhance their understanding in debugging	Knowledge in MR Window functionalities and MR8C/4
Troubleshooting FAQ	Listing of FAQs that users might have	Knowledge in MR8C/4
MR Window Overview	A theoretical explanation of MR Window feature	Knowledge in High-performance Embedded Workshop
Reference Documents	Listing of documents that equip users with knowledge in the pre-requisite requirements	None

2. Overview of Debugging Techniques

A general debugging procedure can be as followed.

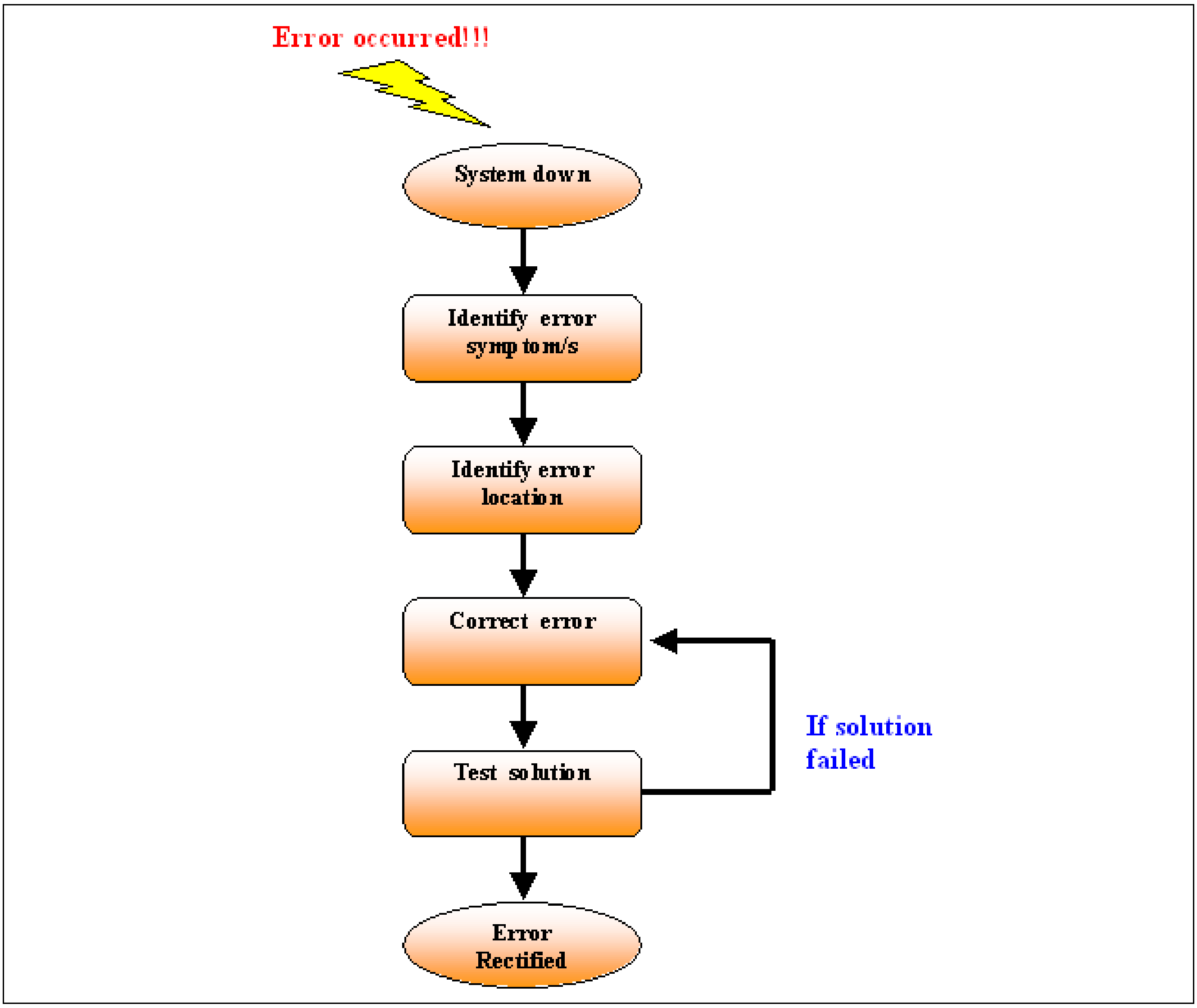


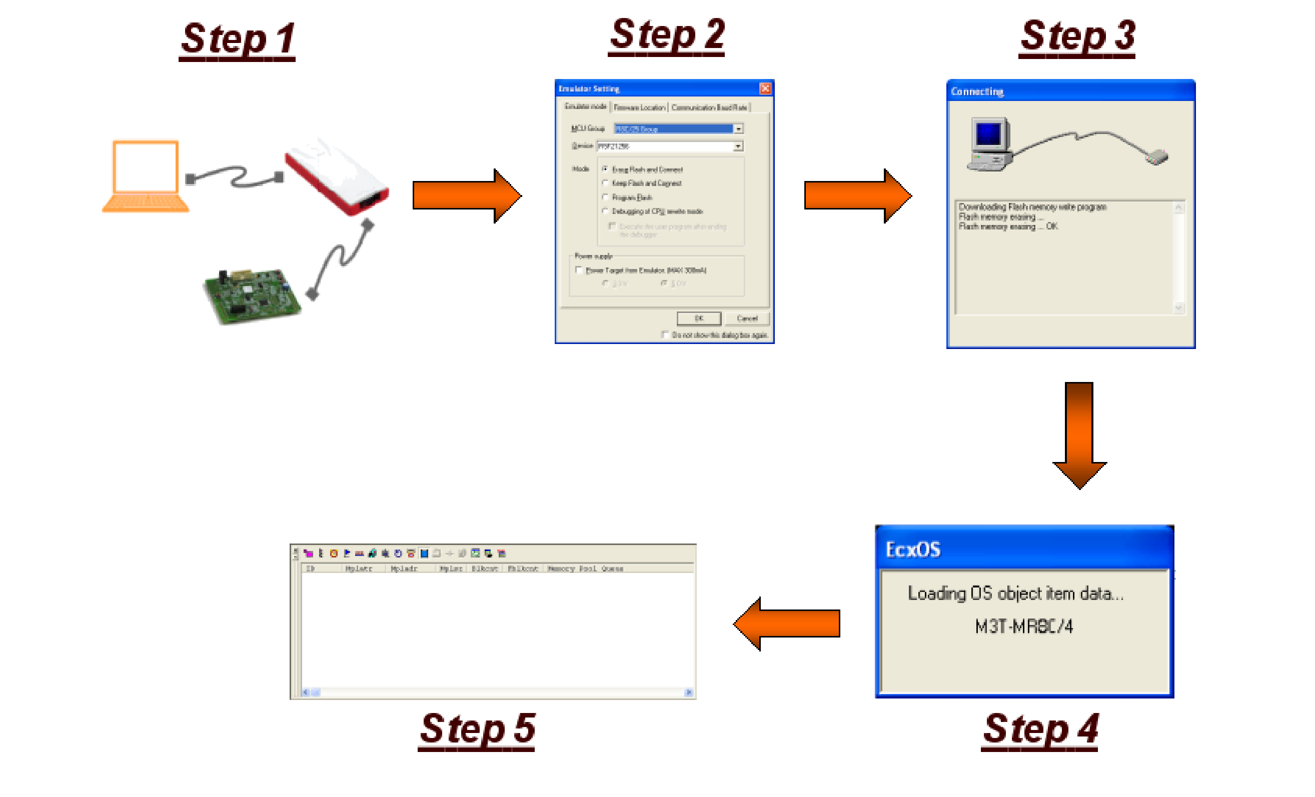
Figure 1 Debugging Procedure

In applications incorporating MR8C/4, users will be able to obtain the status of the RTOS resources created using the MR Window. In addition, error code of most service calls will be returned when they are being executed. Thus, a user may perform a comprehensive debugging by knowing how to interpret MR Window to identify the symptoms and creating a system down routine to identify the error location.

3. Preparation for Debugging

The setup procedures are as followed:

Step 1. Connect up your target board to a terminal using E8a emulator.
Step 2. Open, compile and link your program (with debugging code).
Step 3. Setup emulator and connect to High-performance Embedded Workshop.
Step 4. Download program.
Step 5. Open MR Window and run program.



MCU Group: R8C/25 Group
Device: R5F21256
Mode:
● Erase Flash and Connect
○ Keep Flash and Connect
○ Program Flash
○ Debugging of CPU rewrite mode
□ Execute the user program after ending the debugger
Power supply
□ Power Target from Emulator. (MAX 300mA)
○ 3.3V ● 5.0V
□ Do not show this dialog box again.

Downloading Flash memory write program
Flash memory erasing ...
Flash memory erasing ... OK

Loading OS object item data...
M3T-MR8C/4

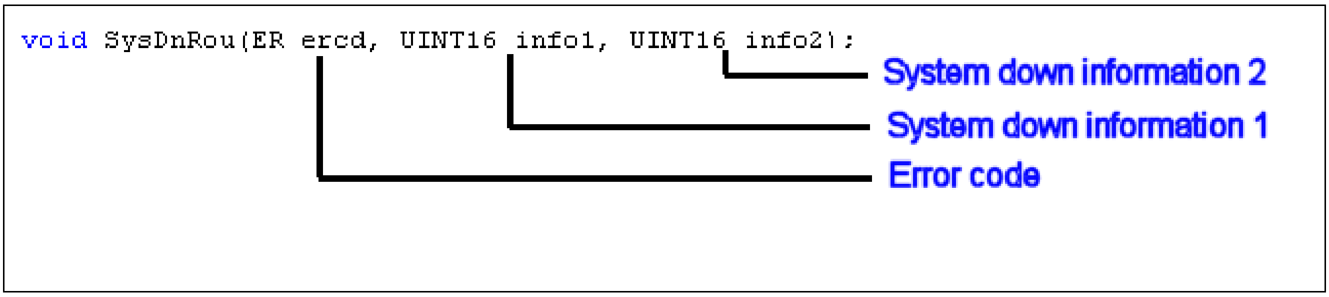
ID	Mplatr	Mpladr	Mplsz	Blkcnt	FBlkcnt	Memory Pool Queue

Figure 2 Setup Procedures

4. Understanding System Down Routine

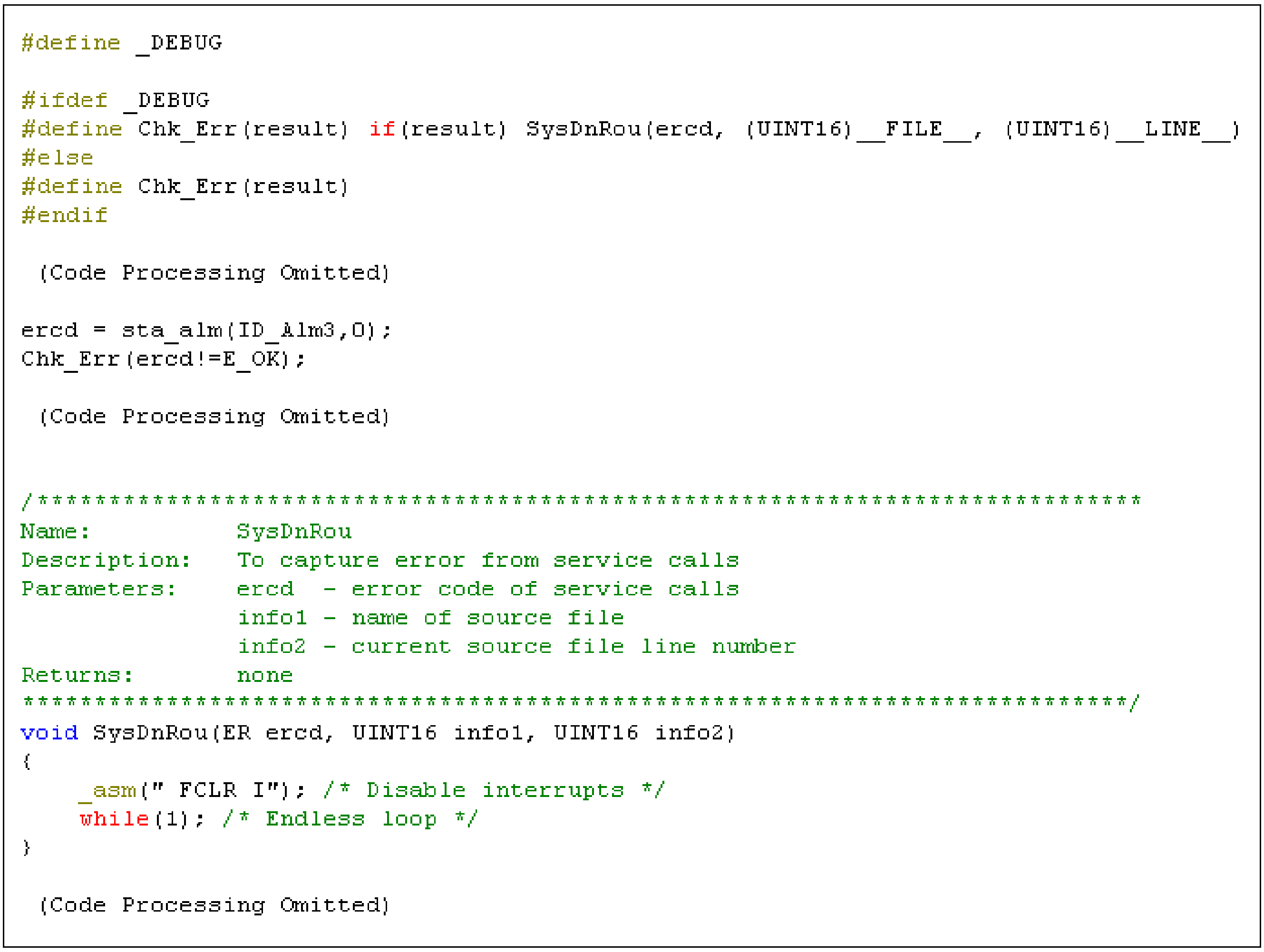
Implementation of a system down routine is not mandatory if user is verse in using MR Window for the debugging. Nevertheless, it might serve to be useful if there is more than one error in an application.

A system down routine serves to identify an error and stop the processing of an application when a service call returns a fatal error code. The calling interface for the system down routine is critical to ensue its effectiveness.



```
void SysDnRou(ER ercd, UINT16 info1, UINT16 info2);
```

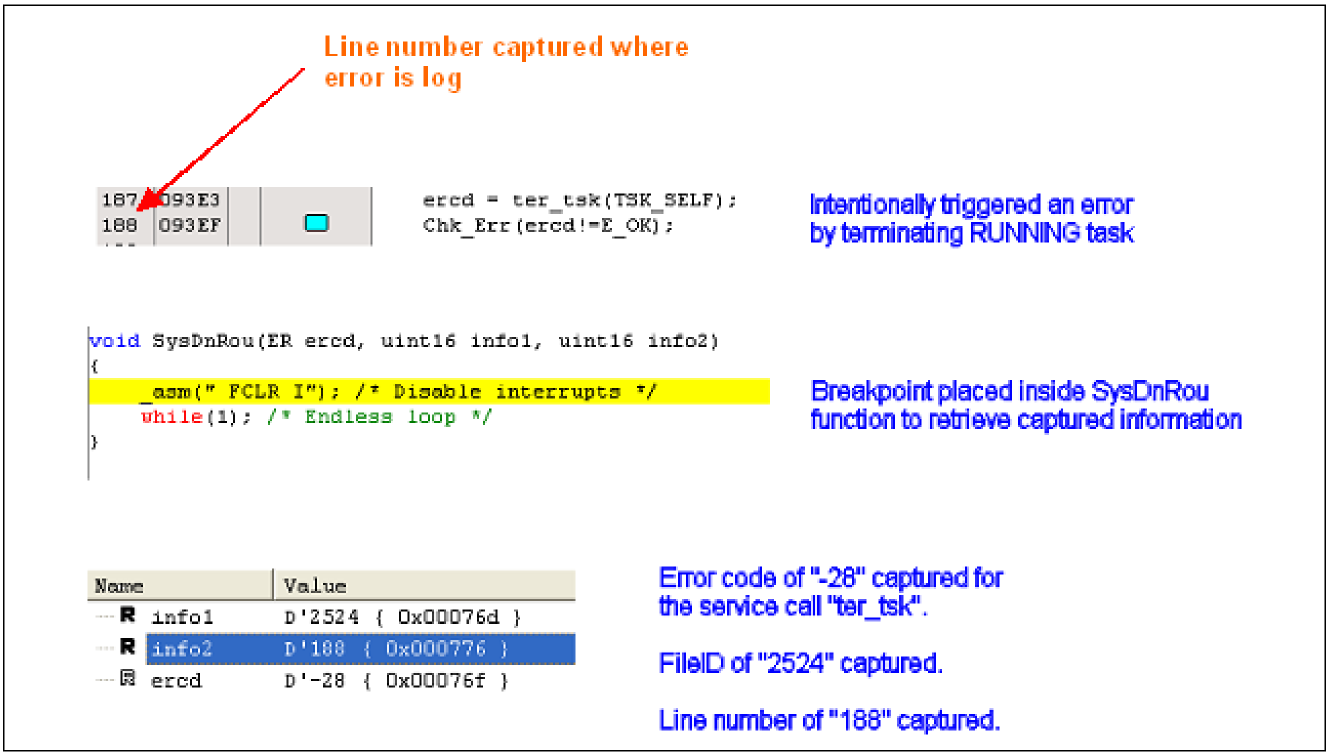
Figure 3 Example of Calling Interface of System Down Routine



```
#define _DEBUG

#ifdef _DEBUG
#define Chk_Err(result) if(result) SysDnRou(ercd, (UINT16)__FILE__, (UINT16)__LINE__)
#else
#define Chk_Err(result)
#endif

 (Code Processing Omitted)

ercd = sta_alm(ID_Alm3,0);
Chk_Err(ercd!=E_OK);

 (Code Processing Omitted)


/*****************************************************************************
Name:          SysDnRou
Description:   To capture error from service calls
Parameters:    ercd  - error code of service calls
               info1 - name of source file
               info2 - current source file line number
Returns:       none
*****************************************************************************/
void SysDnRou(ER ercd, UINT16 info1, UINT16 info2)
{
    _asm(" FCLR I"); /* Disable interrupts */
    while(1); /* Endless loop */
}

 (Code Processing Omitted)
```

Figure 4 An Example of Debugging Code

Note: User is required to put a breakpoint in the function “SysDnRou” to capture the errors.



Line number captured where error is log

```
187  093E3        ercd = ter_tsk(TSK_SELF);
188  093EF        Chk_Err(ercd!=E_OK);
```

Intentionally triggered an error by terminating RUNNING task

```
void SysDnRou(ER ercd, uint16 info1, uint16 info2)
{
    _asm(" FCLR I"); /* Disable interrupts */
    while(1); /* Endless loop */
}
```

Breakpoint placed inside SysDnRou function to retrieve captured information

Name	Value
info1	D'2524 { 0x00076d }
info2	D'188 { 0x000776 }
ercd	D'-28 { 0x00076f }

Error code of "-28" captured for the service call "ter_tsk".

FileID of "2524" captured.

Line number of "188" captured.

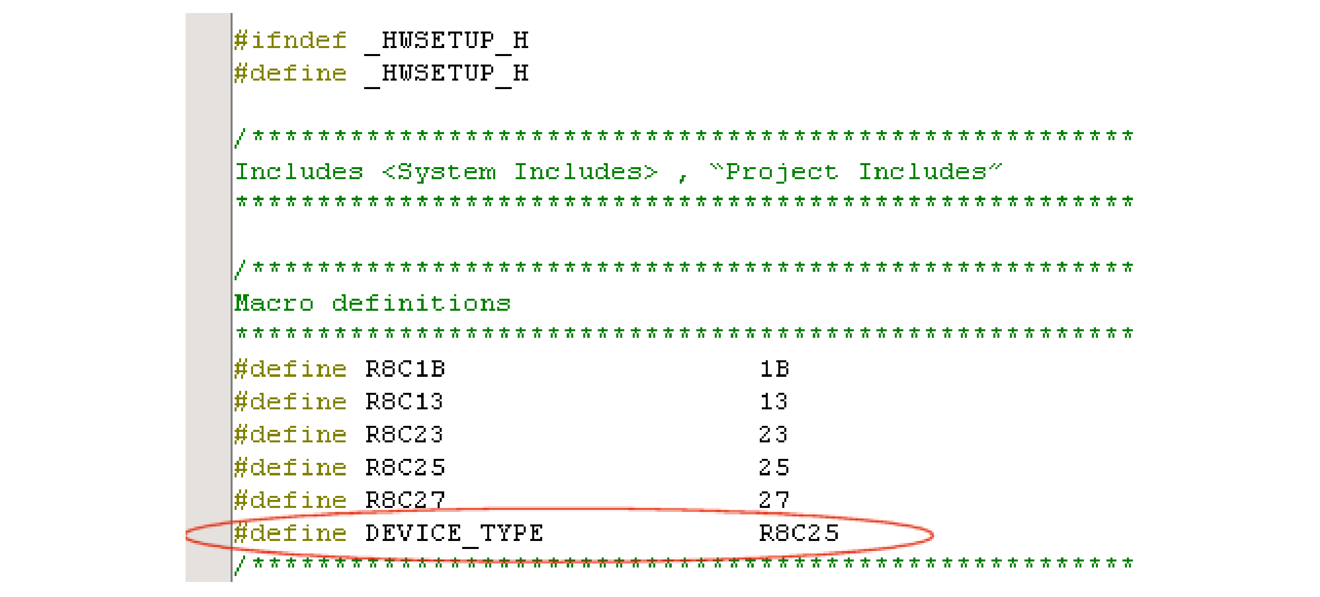
Figure 5 An Example of Captured Information

5. Understanding MR Window through Practical Hands-On

5.1 Settings of MR8C_4_RTOS

Sample program “MR8C_4_RTOS” will be used in the illustration of this topic. “MR8C_4_RTOS” works with Renesas Starter Kit for R8C/13, R8C/1B, R8C/23, R8C/25 and R8C/27.

To make the device choice, open “hwsetup.h” and make the selection.

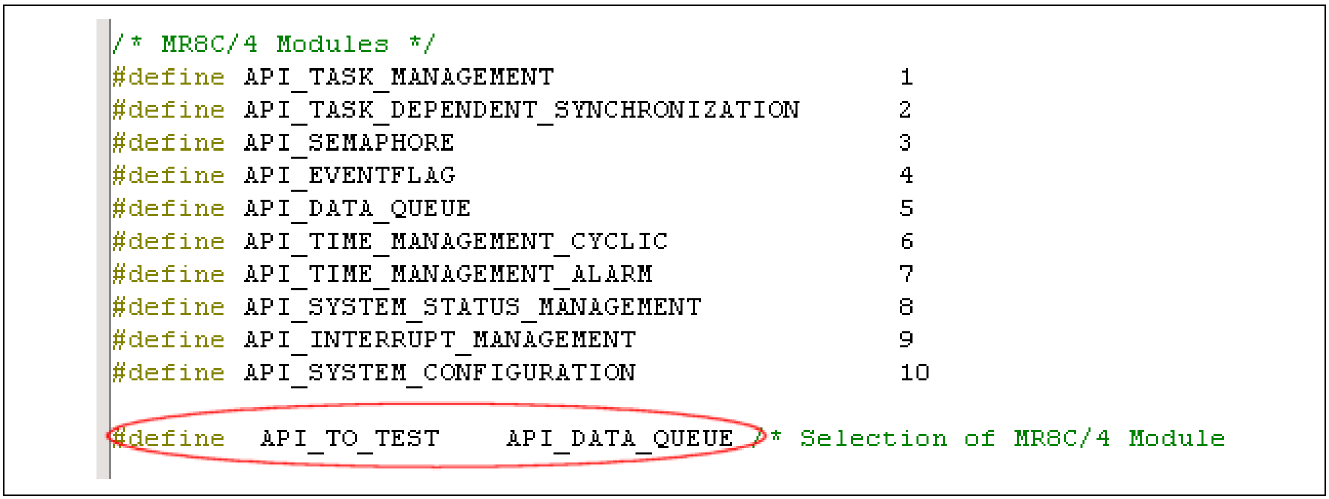


```
#ifndef _HWSETUP_H
#define _HWSETUP_H

/*******************************************************
Includes <System Includes> , "Project Includes"
********************************************************

/*******************************************************
Macro definitions
********************************************************
#define R8C1B                   1B
#define R8C13                   13
#define R8C23                   23
#define R8C25                   25
#define R8C27                   27
#define DEVICE_TYPE             R8C25
/*******************************************************
```

Figure 6 Selecting the Device

“MR8C_4_RTOS” illustrates the functionality of APIs in all of the ten MR8C/4 kernel modules. Users may perform a step through of all the tasks in individual module whilst monitoring the status updates of each display in the MR Window. The comments in the coding serves to guide users in the step through process.

Users may select the module to step through in the file “MR8C_4_RTOS.h”.

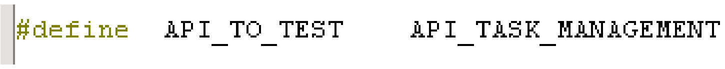


```
/* MR8C/4 Modules */
#define API_TASK_MANAGEMENT                     1
#define API_TASK_DEPENDENT_SYNCHRONIZATION      2
#define API_SEMAPHORE                           3
#define API_EVENTFLAG                           4
#define API_DATA_QUEUE                          5
#define API_TIME_MANAGEMENT_CYCLIC              6
#define API_TIME_MANAGEMENT_ALARM               7
#define API_SYSTEM_STATUS_MANAGEMENT            8
#define API_INTERRUPT_MANAGEMENT                9
#define API_SYSTEM_CONFIGURATION                10

#define  API_TO_TEST    API_DATA_QUEUE /* Selection of MR8C/4 Module
```

Figure 7 Selecting the Kernel Module

5.2 Demonstration of Task Management Module

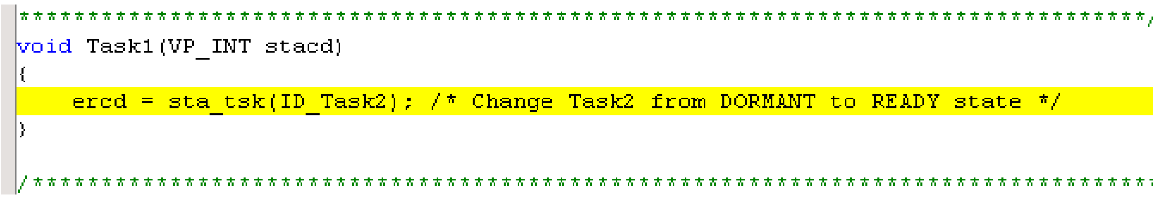
Open workspace of “MR8C_4_RTOS”, select the Renesas Starter Kit you are working on and define “API_TO_TEST” to be “API_TASK_MANAGEMENT” in “MR8C_4_RTOS.h”.



```
#define   API_TO_TEST     API_TASK_MANAGEMENT
```

Figure 8 Selecting API_TASK_MANAGEMENT Modules

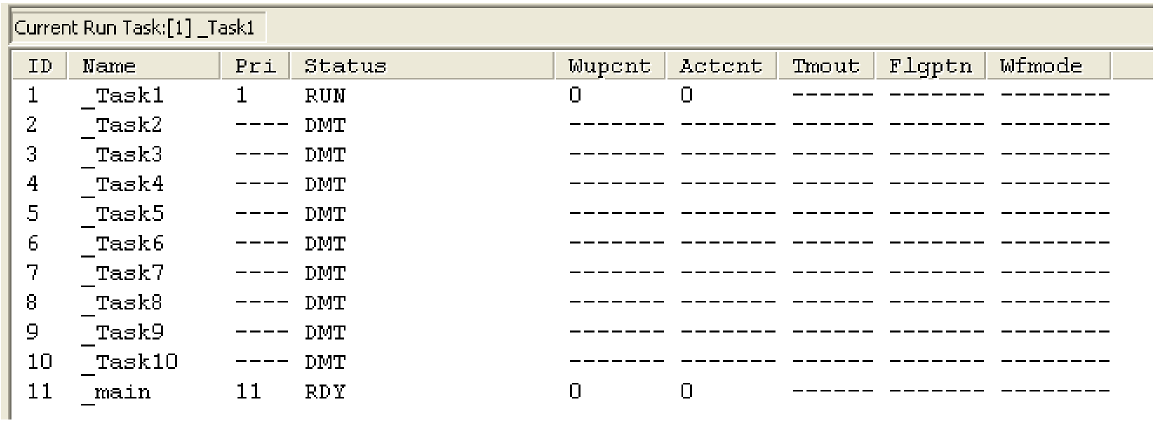
Follow the setup procedures described in section 3.1, put a breakpoint in Task1 and run the sample program “MR8C_4_RTOS”.



```
*********************************************************************************/
void Task1(VP_INT stacd)
{
    ercd = sta_tsk(ID_Task2); /* Change Task2 from DORMANT to READY state */
}

/*********************************************************************************
```

Figure 9 Setting Breakpoint in Task1

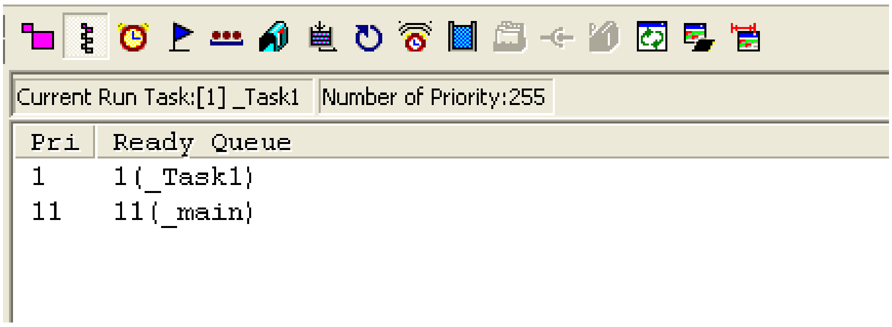
Observe the MR Window and you will be able to identify “Task1” is in a RUNNING state, main in READY state and rest of tasks in DORMANT state. It is also clearly indicated task1 has a priority of ‘1’ and main has a priority of ‘11’. The current running task is also indicated to be “_Task1” in the top right hand corner of MR Window.



```
Current Run Task:[1] _Task1

ID  Name      Pri   Status            Wupcnt  Actcnt  Tmout  Flgptn  Wfmode
1   _Task1    1     RUN               0       0       ------ ------- --------
2   _Task2    ----  DMT               ------- ------- ------ ------- --------
3   _Task3    ----  DMT               ------- ------- ------ ------- --------
4   _Task4    ----  DMT               ------- ------- ------ ------- --------
5   _Task5    ----  DMT               ------- ------- ------ ------- --------
6   _Task6    ----  DMT               ------- ------- ------ ------- --------
7   _Task7    ----  DMT               ------- ------- ------ ------- --------
8   _Task8    ----  DMT               ------- ------- ------ ------- --------
9   _Task9    ----  DMT               ------- ------- ------ ------- --------
10  _Task10   ----  DMT               ------- ------- ------ ------- --------
11  _main     11    RDY               0       0       ------ ------- --------
```

Figure 10 Task Status Display in MR Window (Breakpoint in Task1)

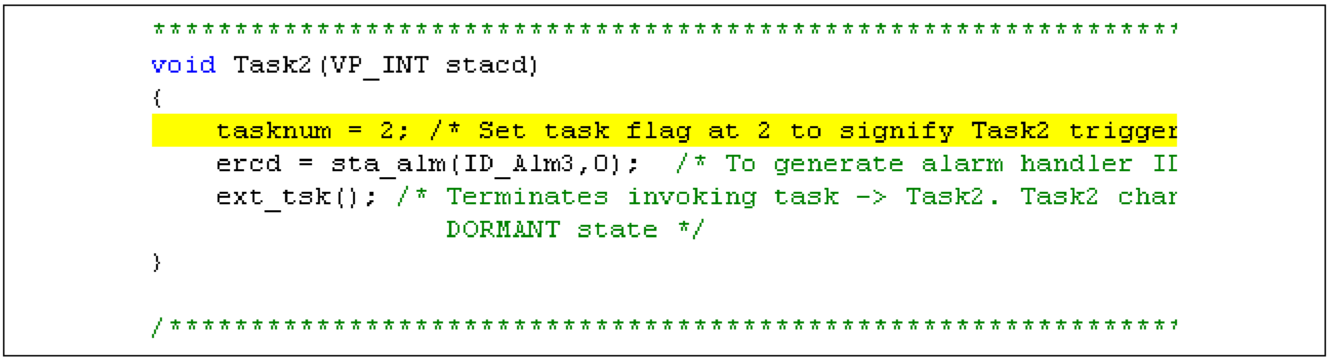
Switch to Ready Queue display window and you may observe Task1 “is at the top of the queue followed by” main.



```
Current Run Task:[1] _Task1   Number of Priority:255

Pri   Ready Queue
1     1(_Task1)
11    11(_main)
```

Figure 11 Ready Queue Display in MR Window (Breakpoint in Task1)

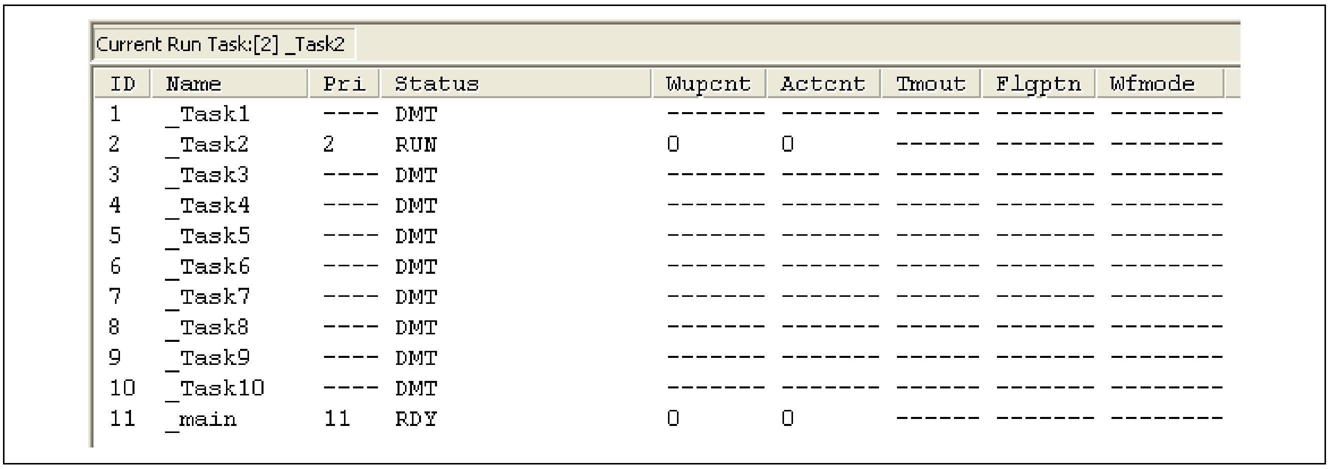
Next put a breakpoint in “Task2” and proceed on with ‘F5’.



```
****************************************************************
void Task2(VP_INT stacd)
{
    tasknum = 2; /* Set task flag at 2 to signify Task2 trigger
    ercd = sta_alm(ID_Alm3,0);  /* To generate alarm handler II
    ext_tsk(); /* Terminates invoking task -> Task2. Task2 char
                  DORMANT state */
}

/***************************************************************
```

Figure 12 Setting Breakpoint in Task2

You may observe that “Task1” switch to DORMANT state, whilst “Task2” changed from DORMANT to RUNNING state. Task “main” remains in READY state as it has a “while” loop.

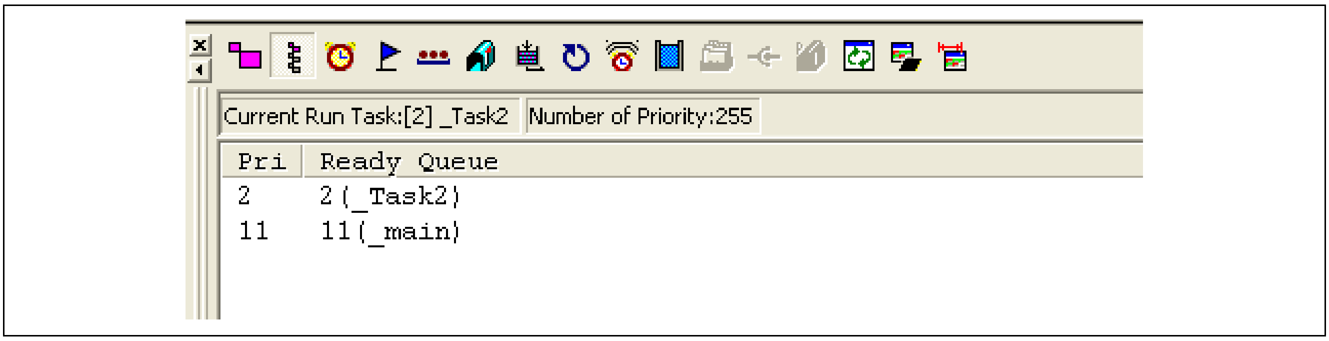


Current Run Task:[2] _Task2

ID	Name	Pri	Status	Wupcnt	Actcnt	Tmout	Flgptn	Wfmode
1	_Task1	----	DMT	--------	--------	-------	--------	---------
2	_Task2	2	RUN	0	0	-------	--------	---------
3	_Task3	----	DMT	--------	--------	-------	--------	---------
4	_Task4	----	DMT	--------	--------	-------	--------	---------
5	_Task5	----	DMT	--------	--------	-------	--------	---------
6	_Task6	----	DMT	--------	--------	-------	--------	---------
7	_Task7	----	DMT	--------	--------	-------	--------	---------
8	_Task8	----	DMT	--------	--------	-------	--------	---------
9	_Task9	----	DMT	--------	--------	-------	--------	---------
10	_Task10	----	DMT	--------	--------	-------	--------	---------
11	_main	11	RDY	0	0	-------	--------	---------

Figure 13 Task Status Display in MR Window (Breakpoint in Task2)

Switch to Ready Queue display window and you may observe Task2 “is at the top of the queue followed by” main.



Current Run Task:[2] _Task2   Number of Priority:255

Pri	Ready Queue
2	2(_Task2)
11	11(_main)

Figure 14 Ready Queue Display in MR Window (Breakpoint in Task2)

Follow this step through process for the rest of the tasks and repeat the whole process for the rest of the kernel modules.

6. Exercise: Debugging “Watch_RSKR8C_DemoOS” Program

Users may proceed on with this exercise upon gaining familiarity and ease of using MR Window.

In this exercise, users are required to identify three bugs in the “Watch_RSKR8C_DemoOS_Debug” program. Users may refer to the solutions given in the APPENDIX section to verify your findings. Users may also refer to the working version: “Watch_RSKR8C_DemoOS”.

“Watch_RSKR8C_DemoOS” is a sample program that serves to function as a digital watch with the following features:

1. Clock Timing Display
2. Alarm Setting
3. Stop Watch
4. Illumination

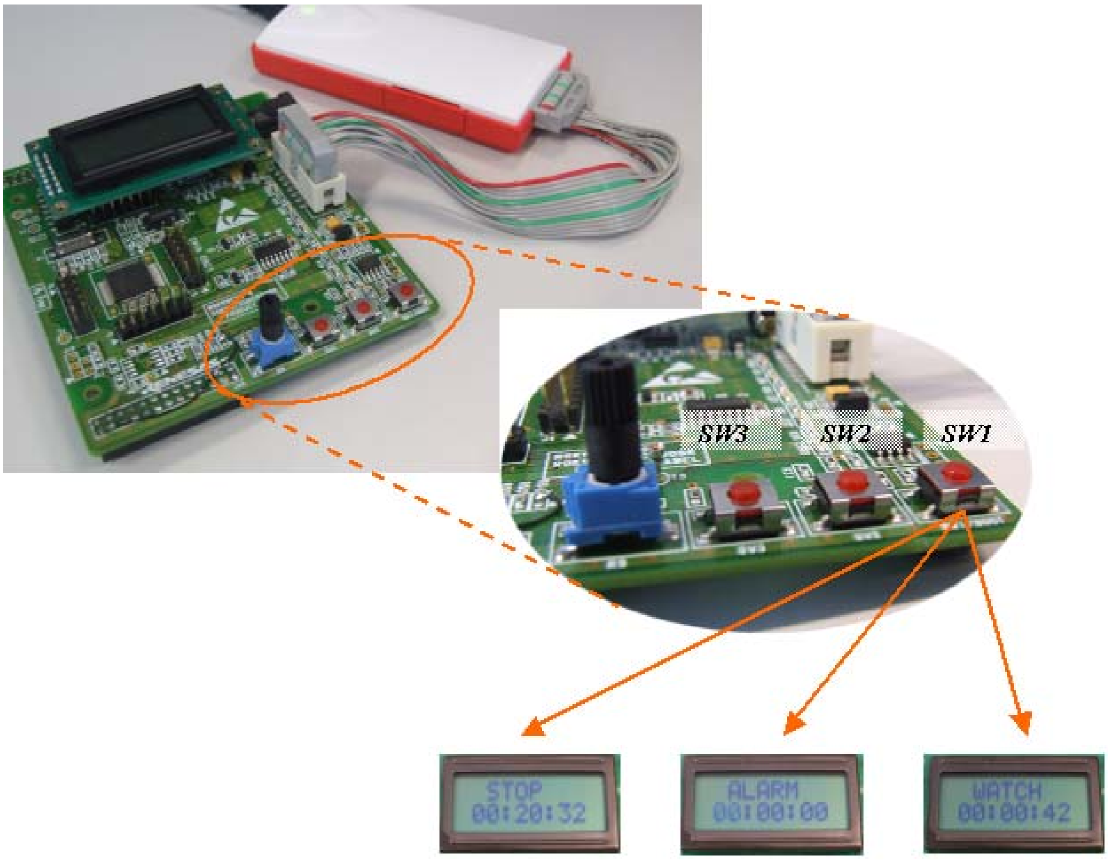


Figure 15 RSKMR8C/25 Running with “Watch_RSKR8C_DemoOS” Program

The respective switches functionalities are as follows.

Table 2 “Watch_RSKR8C_DemoOS” Program Controls

Switch	Function	Procedures
SW1	Select mode "Watch", "Alarm" or "Stop Watch"	Depress SW1 to switch between the modes
SW2	For adjusting of clock timing, alarm setting and stop watch counting	For Clock setting, press and hold SW2 for a few seconds in "Watch" mode. Display will flicker and user may set the timing.
		For Alarm setting, depress SW2 once to increment the count by one in "Alarm" mode
		For Stop watch counting, depress SW2 to start/stop the counting in "Stop Watch" mode
SW3	For illuminating the LEDs, enabling and disabling of alarm setting.	Depress SW3 in "Watch" mode to turn on the LEDs
		Depress SW3 once in "Stop Watch" mode to enable/disable alarm activation. A symbol '^' will be displayed if alarm is enabled

Follow the setup procedures described in section 3.1, download and run the program “Watch_RSKR8C_DemoOS”.

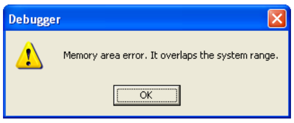
*Hint: The 3 bugs are related to the following symptoms; 1) Cannot switch to “Stop Watch” mode, 2) Inaccurate stop watch counting and 3) Unable to adjust watch clock timing.

7. Troubleshooting FAQ

Few of the common problems one may encounter in using MR8C/4 and MR Window can be the followings:

7.1 Memory Area Error

This error occurred during the downloading of the module file.

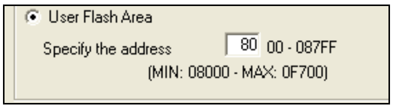


Memory area error. It overlaps the system range.

Figure 16 Memory Error Message

7.1.1 Possible Solutions

Ensure both the declaration of data area in “c_sec.inc” file and specification of address for User Flash Area for the emulator setting is correct.



● User Flash Area

Specify the address 80 00 - 087FF

(MIN: 08000 - MAX: 0F700)

Figure 17 Emulator Setting

7.2 No MR Window

MR Window option not available when program downloaded.

7.2.1 Possible Solutions

Do ensure an emulator that support the MR Window option is used. (E.g. E8a emulator)

Note: E8 emulator do not support MR Window option

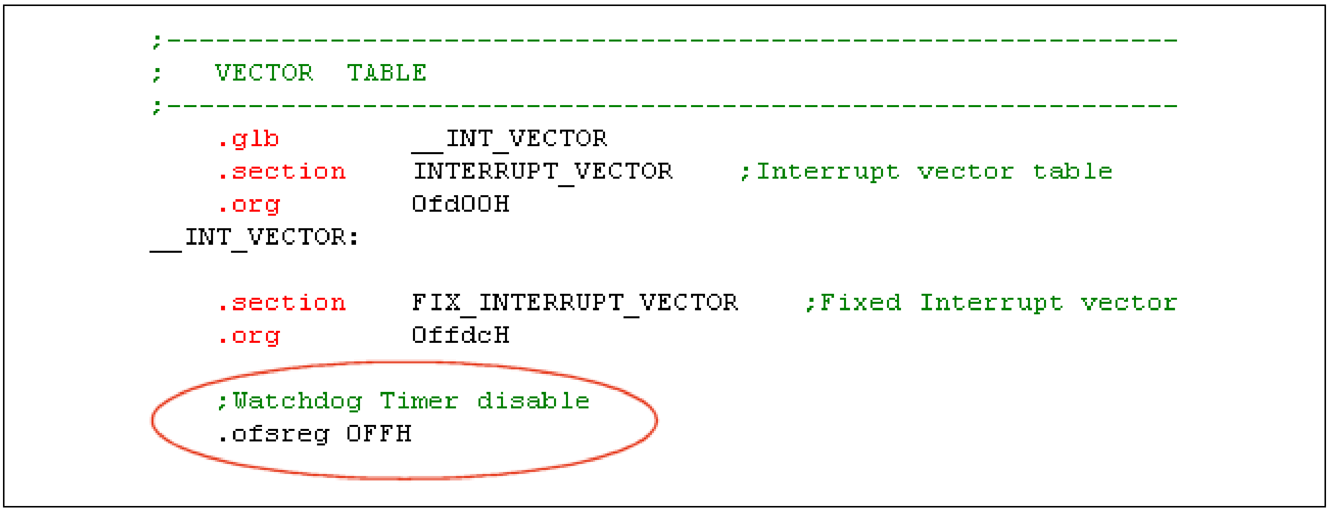
7.3 Program failed to startup

Program failed to transit to the “Task” that is set to run on initial startup.

7.3.1 Possible Solutions

If any of the service calls belonging to the kernel module “Time Management Function” is used (e.g. sta_cyc), it is require assigning one timer for use as the system clock.

To perform the assignment, define the timer handler in the configuration file and initialize the timer correctly whilst ensuring the watchdog timer is disabled in Figure 18.



```
;-----------------------------------------------------------------
;    VECTOR  TABLE
;-----------------------------------------------------------------
    .glb          __INT_VECTOR
    .section      INTERRUPT_VECTOR    ;Interrupt vector table
    .org          0fd00H
__INT_VECTOR:

    .section      FIX_INTERRUPT_VECTOR    ;Fixed Interrupt vector
    .org          0ffdcH

    ;Watchdog Timer disable
    .ofsreg 0FFH
```

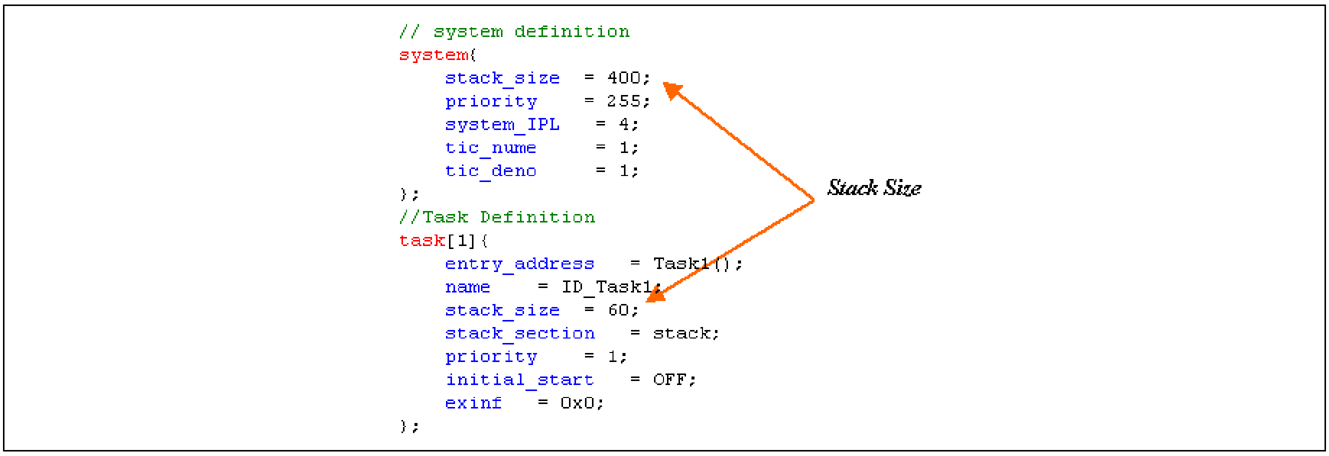
Figure 18 Watchdog Timer Disabled in c_sec.inc

7.4 Abnormality occurred in program; Task did not perform in accordance to expectations

Abnormality occurs with program failing to perform and no errors observed.

7.4.1 Possible Solutions

Do ensure adequate amount of stack sizes are specified for system and individual task in the configurator file.



```
// system definition
system{
    stack_size   = 400;
    priority     = 255;
    system_IPL    = 4;
    tic_nume      = 1;
    tic_deno      = 1;
};
//Task Definition
task[1]{
    entry_address    = Task1();
    name      = ID_Task1;
    stack_size  = 60;
    stack_section    = stack;
    priority     = 1;
    initial_start    = OFF;
    exinf     = 0x0;
};
```

Figure 19 Stack Size Definitions in Configuration File

8. Reference Documents

User’s Manual

- MR8C/4 V1.00 User’s Manual
- E8a Emulator User’s Manual
- High-performance Embedded Workshop V4.05 User’s Manual
- C Compiler Package for M16C Series and R8C Family V5.45 User’s Manual
- R8C Family Software Manual

The latest version can be downloaded from the Renesas Technology website

9. Appendix I: MR Window Overview

MR Window displays the kernel resources and status of MR8C/4. To open up the window, select [View->RTOS->MR Window] or click on (icon).

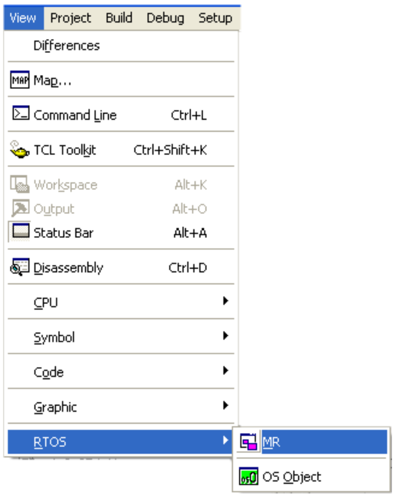
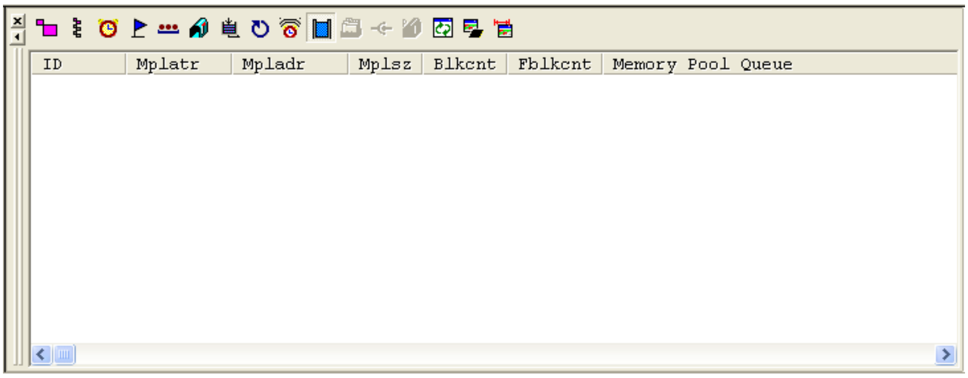


Figure 22 Activation of MR Window



ID	Mplatr	Mpladr	Mplsz	Blkcnt	Fblkcnt	Memory Pool Queue

Figure 23 MR Window

MR Window supports the following kernel resources display for MR8C:

- Task status
- Ready queue status
- Event flag status
- Semaphore status
- Data queue status
- Cyclic handler status
- Alarm handler status

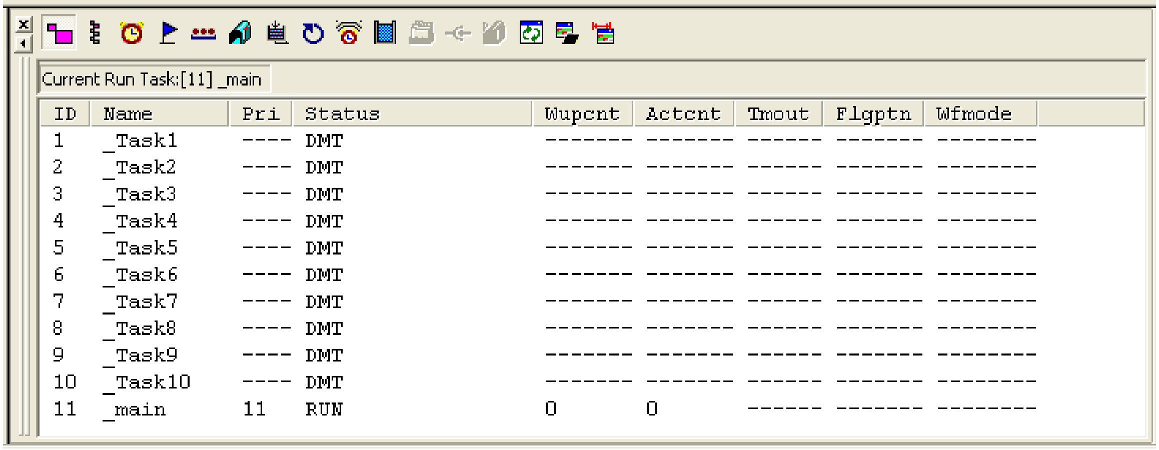
9.1 Task Status Display

9.1.1 Purpose

Task status window displays all the tasks that have been created. The display is done in the order of the tasks' creation. This window serves to provide users the knowledge of all the tasks created and its status.

9.1.2 Activation

In the MR window, depress .

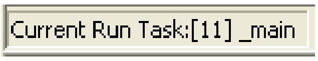


Current Run Task:[11] _main

ID	Name	Pri	Status	Wupcnt	Actcnt	Tmout	Flgptn	Wfmode
1	_Task1	----	DMT	-------	-------	------	-------	--------
2	_Task2	----	DMT	-------	-------	------	-------	--------
3	_Task3	----	DMT	-------	-------	------	-------	--------
4	_Task4	----	DMT	-------	-------	------	-------	--------
5	_Task5	----	DMT	-------	-------	------	-------	--------
6	_Task6	----	DMT	-------	-------	------	-------	--------
7	_Task7	----	DMT	-------	-------	------	-------	--------
8	_Task8	----	DMT	-------	-------	------	-------	--------
9	_Task9	----	DMT	-------	-------	------	-------	--------
10	_Task10	----	DMT	-------	-------	------	-------	--------
11	_main	11	RUN	0	0	------	-------	--------

Figure 24 Task Status Display

For an instant identification of running task, refer to the display in the status bar.



Current Run Task:[11] _main

(Task ID: 11, Task Name: main)

Figure 25 Task Status Bar

9.1.3 Fields of Display

The fields of each individual task displayed in the MR Task window are described below.

Table 3 Fields of Task Status Display

Fields	Descriptions
ID	ID number of Task (1 to 255)
Name	Entry_address of Task
Pri	Current Priority of Task (1 to 255)
Status	Current Status of Task
Wupcnt	Number of queued wakeup request counts
Flgptn	Wait bit pattern of event flag
Wfmode	Wait cancellation condition of event flag

Table 4 Listing of Task Status

Status	Descriptions
RUN	Running state
RDY	Ready State
SUS	Suspended state
DMT	Dormant state
WAI(SLP)	Wait state kept by sleep request
WAI(SLP)-SUS	Wait state kept by sleep request in suspended
WAI(DLY)	Wait state kept by delay request
WAI(DLY)-SUS	Wait state kept by delay request in suspended
WAI(FLG)	Wait state kept by event flag request
WAI(FLG)-SUS	Wait state kept by event flag request in suspended
WAI(SEM)	Wait state kept by semaphore request
WAI(SEM)-SUS	Wait state kept by semaphore request in suspended
WAI(SDTQ)	Wait state kept by send data queue request
WAI(SDTQ)-SUS	Wait state kept by send data queue request in suspended
WAI(RDTQ)	Wait state kept by request data queue request
WAI(RDTQ)-SUS	Wait state kept by request data queue request in suspended

Table 5 Listing of Wait Cancellation Condition of Event Flag

Wfmode (wait mode)	Descriptions
TWF_ANDW	Wait until all bits specified by waiptn are set (wait for the bits AND'ed)
TWF_ORW	Wait until one of the bits specified by waiptn is set (wait for the bits OR'ed)

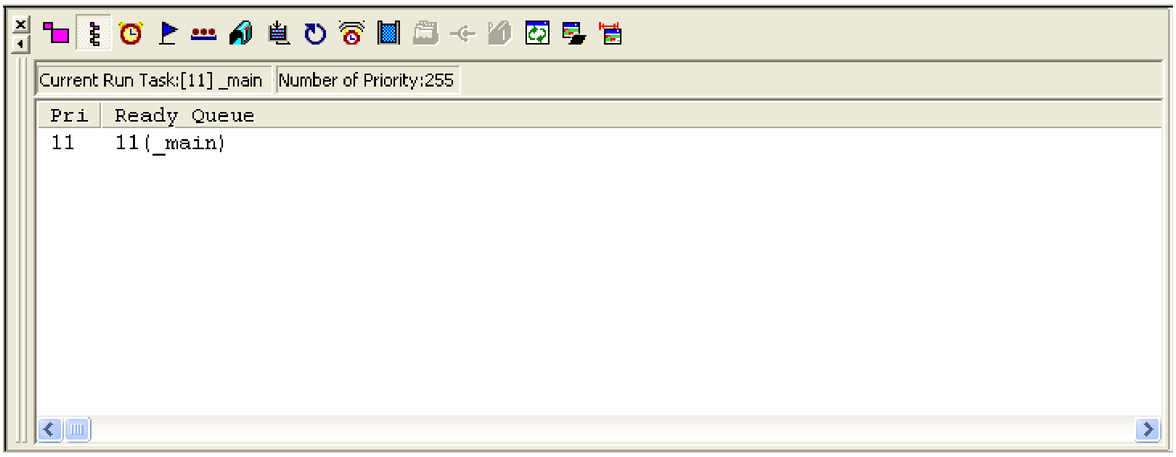
9.2 Ready Queue Status Display

9.2.1 Purpose

In this window, tasks that are either in RUN or RDY state will be displayed in the order of their priority. Its corresponding priority level, ID number and entry_address will be defined. This window serves to allow users to identify all the tasks in the ready queue.

9.2.2 Activation

In the MR window, depress .



Current Run Task:[11] _main | Number of Priority:255

```
Pri  Ready Queue
11   11(_main)
```

Figure 26 Ready Queue Status Display

For an instant identification of total priority level, refer to the display in the status bar.

Number of Priority:255

(Total number of priority level at 255)

Figure 27 Priority Status Bar

9.2.3 Fields of Display

The fields of each individual task displayed in the MR Ready Queue window are described below.

Table 6 Fields of Priority Status Display

Fields	Descriptions
Pri	Priority level of task (1 to 255)
RdyQ	ID number and entry address of task

Up to 9 characters of the task entry_address may be displayed in the RdyQ field. Extra characters will be omitted and displayed in the form of "..".

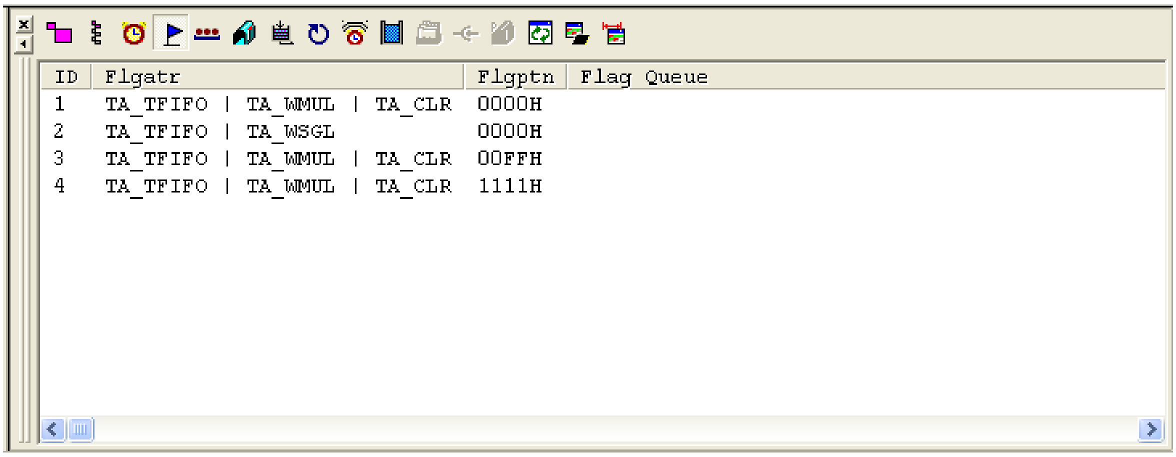
9.3 Ready Event Flag Status Display

9.3.1 Purpose

All the event flags defined in configurator file and created will be listed in the Ready Event Flag window. The window carries a real-time illustration of the flag status including ID number, attributes, bit pattern and task kept in flag ready queue by the particular event flag. This window facilitates user in the debugging of errors occurred due to synchronization of multiple tasks using event flag. .

9.3.2 Activation

In the MR window, depress ▶.



```
ID  Flgatr                          Flgptn  Flag Queue
1   TA_TFIFO | TA_WMUL | TA_CLR    0000H
2   TA_TFIFO | TA_WSGL             0000H
3   TA_TFIFO | TA_WMUL | TA_CLR    00FFH
4   TA_TFIFO | TA_WMUL | TA_CLR    1111H
```

Figure 28 Ready Event Flag Status Display

9.3.3 Fields of Display

The fields of each individual event flag displayed in the MR Event Flag window are described below.

Table 7 Fields of Ready Event Flag Status Display

Fields	Descriptions
ID	ID number of event flag (1 to 255)
Flgatr	Attribute of event flag
Flgptn	Bit pattern of each event flag (16 bits)
Flag Queue	ID number and entry_address of tasks in the event flag queue

Up to 9 characters of the task entry_address may be displayed in the Flag Queue field. Extra characters will be omitted and displayed in the form of "..".

Table 8 Listing of Event Flag attributes

Flgatr	Descriptions
TA_TFIFO	Event flag waiting queue in FIFO order
TA_WSGL	Event flag waiting queue that supports single task to be enqueued
TA_WMUL	Event flag waiting queue that supports multiple tasks to be enqueued
TA_CLR	Indicates clear attribute of event flag enabled and signify event flag bit pattern will be set to 0x0H when waiting task is released

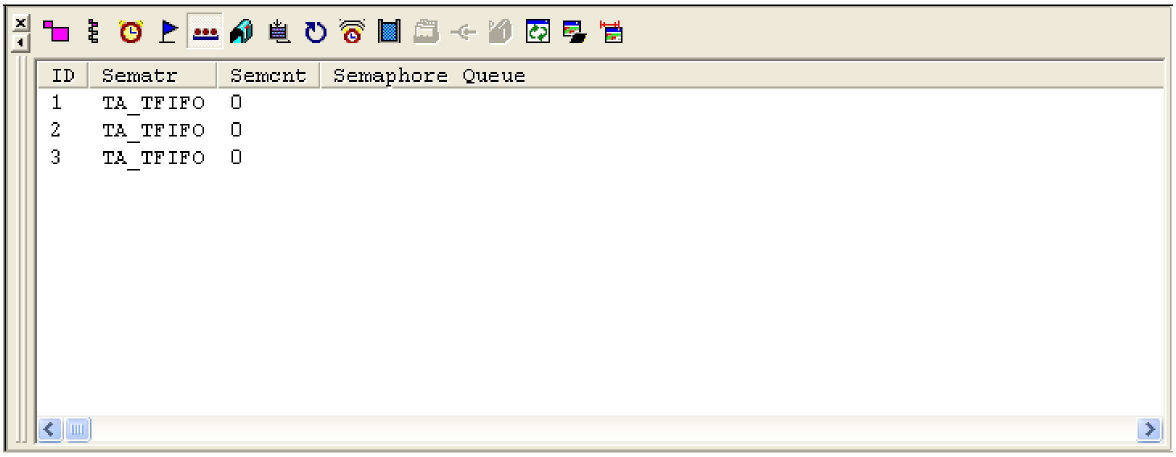
9.4 Semaphore(SEM) Status Display

9.4.1 Purpose

Semaphore window displays all the SEMs created. The SEMs are listed in the order of ID number stating the attribute, resource count and tasks enqueued in the semaphore queue. Likewise, semaphore window assists users in the debugging of errors occurred due to synchronization of multiple tasks using semaphore resources.

9.4.2 Activation

In the MR window, depress •••.



ID	Sematr	Semcnt	Semaphore Queue
1	TA_TFIFO	0	
2	TA_TFIFO	0	
3	TA_TFIFO	0	

Figure 29 Semaphore(SEM) Status Display

9.4.3 Fields of Display

The fields of each individual semaphore displayed in the MR Semaphore window are described below.

Table 9 Fields of Semaphore Status Display

Fields	Descriptions
ID	ID number of semaphore (1 to 255)
Sematr	Attribute of semaphore
Semcnt	Semaphore count (0 to 65535)
Semaphore Queue	ID number and entry_address of tasks in the semaphore queue

Up to 9 characters of the task entry_address may be displayed in the Semaphore Queue field. Extra characters will be omitted and displayed in the form of "..".

Table 10 Listing of Semaphore attributes

Sematr	Descriptions
TA_TFIFO	Semaphore waiting queue in FIFO order

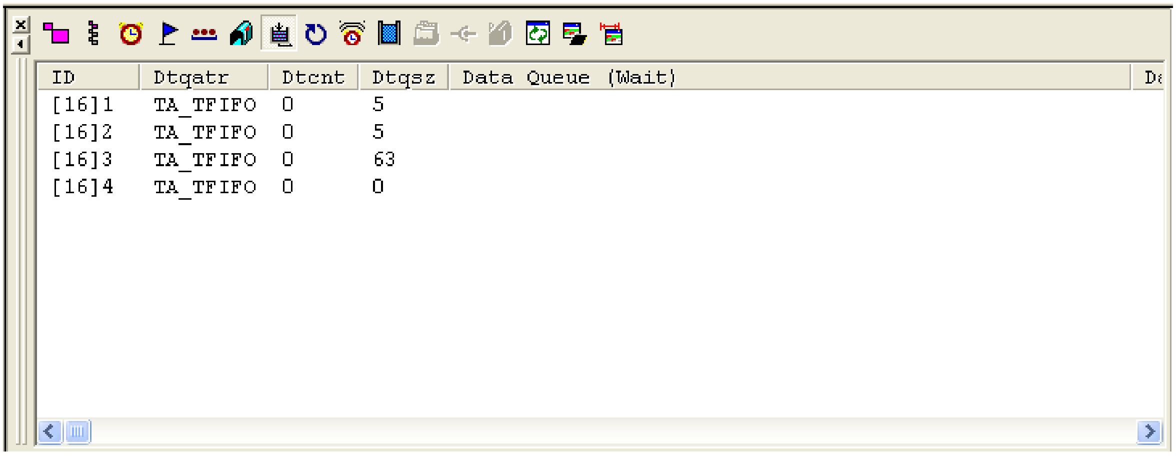
9.5 Data Queue Status Display

9.5.1 Purpose

Data Queue window displays the entire data queue objects defined in the configuration file (template.cfg). The window carries the attributes of the data queue objects and enable users to trace the synchronous transmission and reception of message (term "*data element*") between the tasks.

9.5.2 Activation

In the MR window, depress .



```
ID      Dtqatr    Dtcnt  Dtqsz  Data Queue (Wait)                          D
[16]1   TA_TFIFO  0      5
[16]2   TA_TFIFO  0      5
[16]3   TA_TFIFO  0      63
[16]4   TA_TFIFO  0      0
```

Figure 30 Data Queue Status Display

9.5.3 Fields of Display

The fields of each individual data queue object displayed in the MR Data Queue window are described below.

Table 11 Fields of Data Queue Status Display

Fields	Descriptions
ID	ID number of data queue (1 to 255)
Dtqatr	Attribute of data queue
Dtcnt	Data queue count (0 to 65535)
Dtqsz	Data queue size (16 bits)
Data Queue (Wait)	ID number and entry_address of tasks in the data queue's send-wait queue

Up to 9 characters of the task entry_address may be displayed in the Data Queue (Wait) field. Extra characters will be omitted and displayed in the form of "..".

Table 12 Listing of Data Queue attributes

Sematr	Descriptions
TA_TFIFO	Data queue's waiting queue in FIFO order

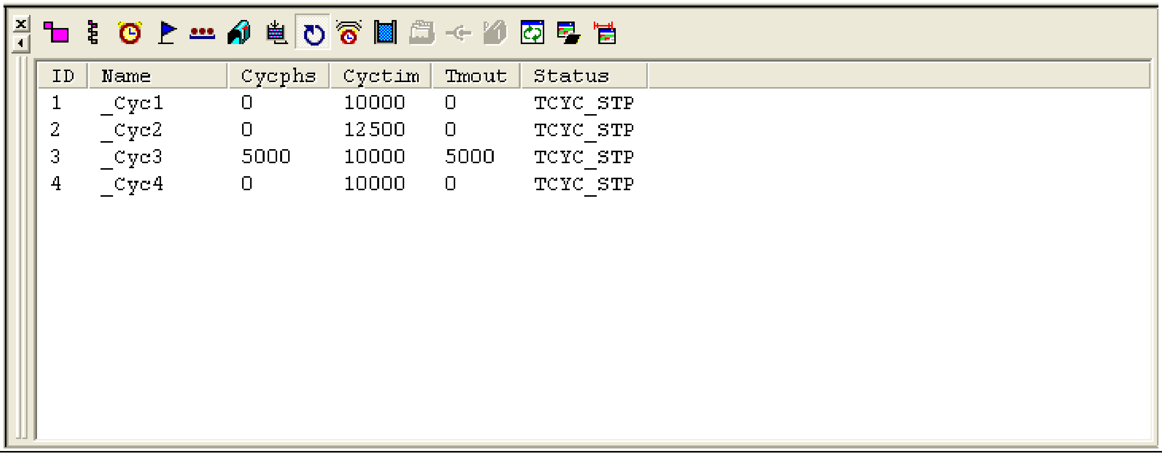
9.6 Cyclic Handler Status Display

9.6.1 Purpose

Cyclic handler window displays all cyclic handlers created. Accurate triggering of cyclic handler is dependent on a number of factors including resolution of system clock time tick, specification of cyclic handlers' attributes and integrity or program. Cyclic handler window provide users a mean of identifying bugs in the inaccurate triggering of cyclic handlers.

9.6.2 Activation

In the MR window, depress ↻.



ID	Name	Cycphs	Cyctim	Tmout	Status
1	_Cyc1	0	10000	0	TCYC_STP
2	_Cyc2	0	12500	0	TCYC_STP
3	_Cyc3	5000	10000	5000	TCYC_STP
4	_Cyc4	0	10000	0	TCYC_STP

Figure 31 Cyclic Handler Status Display

9.6.3 Fields of Display

The fields of each individual cyclic handler displayed in the MR Cyclic Handler window are described below.

Table 13 Fields of Cyclic Handler Status Display

Fields	Descriptions
ID	ID number of cyclic handler (1 to 255)
Name	Entry_address of cyclic handler
Cycphs	Activation phase interval (0 to 7fffffff[ms])
Cyctim	Activation cycle interval (0 to 7fffffff[ms])
Timout	Remaining time for the activation of next cycle in millisecond
Status	State of cyclic handler

9.6.4 Listing of Cyclic Handler Attributes

Table 14 Listing of Cyclic Handler Attributes

Status	Descriptions
TCYC_STA	Cyclic handler is in an operational state
TCYC_STP	Cyclic handler is in a non-operational state

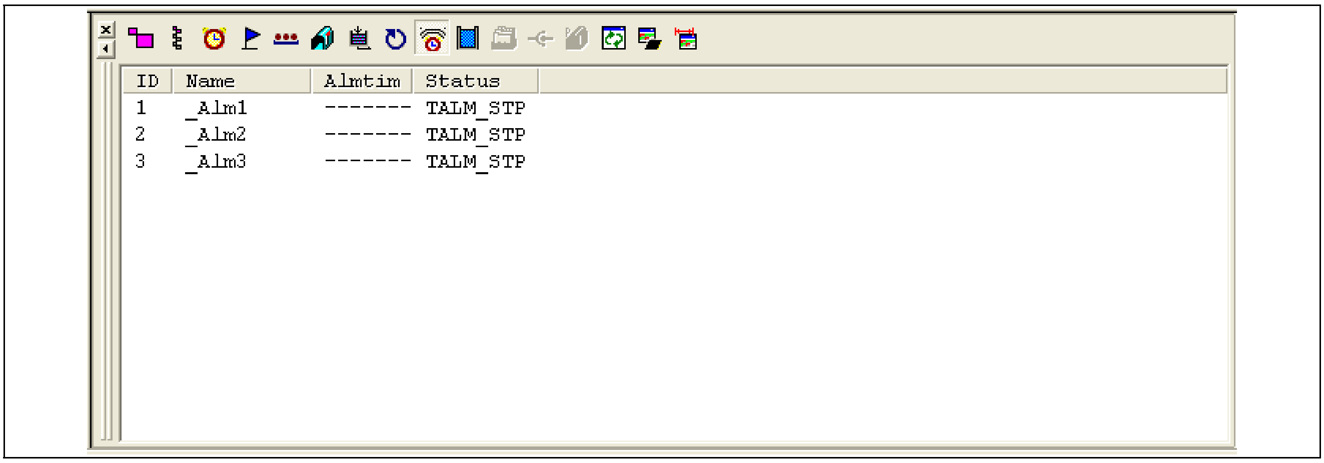
9.7 Alarm Handler Status Display

9.7.1 Purpose

Alarm handler window displays all alarm handlers created. Alarm handler window allows users to verify the integrity of the alarm handlers.

9.7.2 Activation

In the MR window, depress ⏰.



```
ID  Name      Almtim   Status
1   _Alm1     -------  TALM_STP
2   _Alm2     -------  TALM_STP
3   _Alm3     -------  TALM_STP
```

Figure 32 Alarm Handler Status Display

9.7.3 Fields of Display

The fields of each individual alarm handler displayed in the MR alarm handler window are described below.

Table 15 Fields of Alarm Handler Status Display

Fields	Descriptions
ID	ID number of alarm handler (1 to 255)
Name	Entry_address of alarm handler
Almtim	Remaining time for the activation of next cycle in millisecond
Status	State of alarm handler

9.7.4 Listing of Alarm Handler attributes

Table 16 Listing of Alarm Handler Attributes

Status	Descriptions
TALM_STA	Alarm handler is in an operational state
TALM_STP	Alarm handler is in a non-operational state

10. Appendix II: Solution to Exercise: Debugging “Watch_RSKR8C_DemoOS” Program

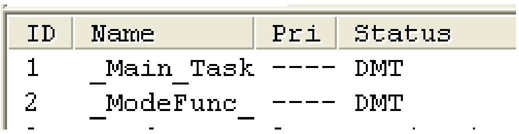
10.1 Bug 1: Failed to Switch to Stop Watch Mode

10.1.1 Description

The first symptom occurs when user depresses SW1 and it failed to switch from “Alarm” to “Stop Watch” mode.

10.1.2 Observations

“ModeFunc_task” is the task that switches the mode. It goes from “WAITING” to “DORMANT” state when SW1 is depressed whilst in “Alarm” mode. As the task is in “DORMANT” state, thus it is impossible to wakeup the task with “ercd = iwup_tsk(ID_Task2_ModeFunc)” service call.

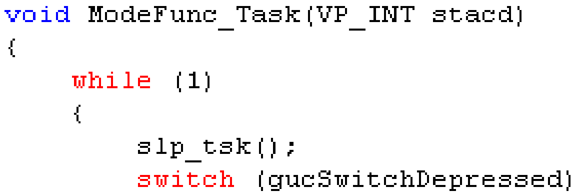


ID	Name	Pri	Status
1	_Main_Task	----	DMT
2	_ModeFunc_	----	DMT

Figure 33 ModeFunc_task in Dormant State

10.1.3 Solution

Place a “while” loop in the task to keep the task in “WAITING” state with “slp_tsk” service call.



```
void ModeFunc_Task(VP_INT stacd)
{
    while (1)
    {
        slp_tsk();
        switch (gucSwitchDepressed)
```

Figure 34 “while” Loop Placed in ModeFunc_task

10.2 Bug 2: Inaccurate Stop Watch Counting

10.2.1 Description

The second symptom occurs when user depresses SW2 in “Stop Watch” mode and the counting is not accurate.

10.2.2 Observations

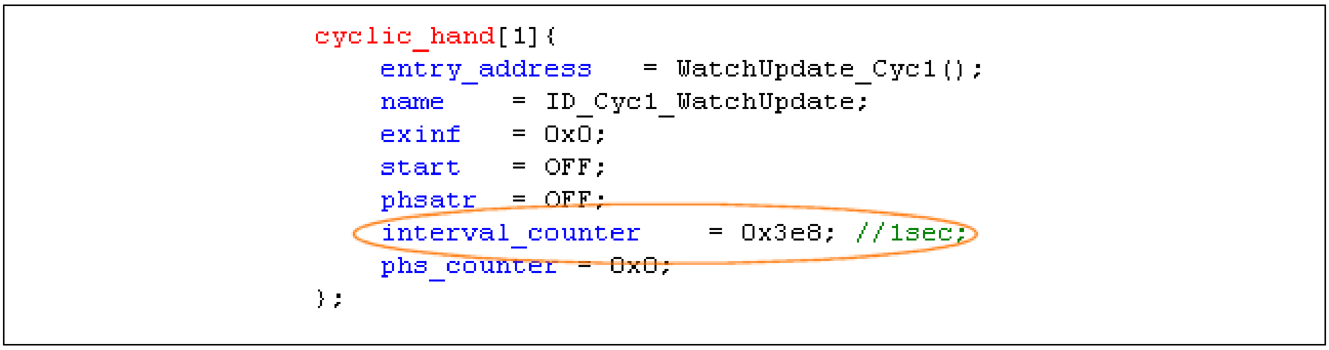
The stop watch counting can be found to be slower by 0.5sec for each count. ID_Cyc3_StopWatchCount is the cyclic assigned for the stopwatch counting.

At a setting of 1msec for timer RA, to generate a 1 sec counting, interval counter of ID_Cyc3_StopWatchCount will need to be 1000 (1 sec divide by 1msec).

The present setting for ID_Cyc3_StopWatchCount is 0x1F4H (=2000). Which will generate a cyclic at an interval of 2 sec.

10.2.3 Solution

Modify the interval counter setting for ID_Cyc3_StopWatchCount to be 0x3E8H.



```
cyclic_hand[1]{
    entry_address    = WatchUpdate_Cyc1();
    name     = ID_Cyc1_WatchUpdate;
    exinf    = 0x0;
    start    = OFF;
    phsatr   = OFF;
    interval_counter     = 0x3e8; //1sec;
    phs_counter = 0x0;
};
```

Figure 35 Setting for ID_Cyc3_StopWatchCount

10.3 Bug 3: Unable to adjust watch clock timing

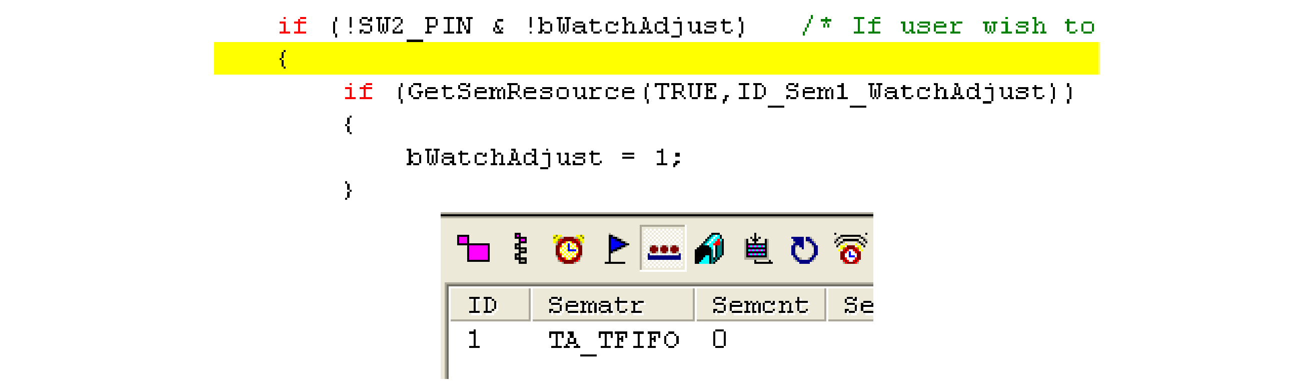
10.3.1 Description

The third symptom occur when user press and hold SW2 in “Watch” mode and display failed to flicker to signal for adjustment.

10.3.2 Observations

“bWatchAdjust” is a variable to enable the adjustment of clock timing in “Watch” mode. To allow user to adjust the timing, “bWatchAdjust” must be set and this require the semaphore resource of “ID_Sem1_WatchAdjust”.

A semcnt of ‘0’ can be observed from the MR Window when the program breaks in the function to acquire the semaphore resource. This zero semcnt inhibit “bWatchAdjust” to be set.



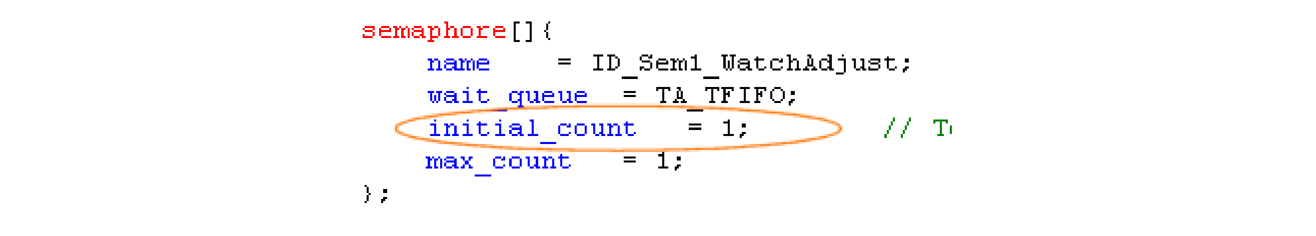
```
if (!SW2_PIN & !bWatchAdjust)    /* If user wish to
{
    if (GetSemResource(TRUE,ID_Sem1_WatchAdjust))
    {
        bWatchAdjust = 1;
    }
```

ID	Sematr	Semcnt	Se
1	TA_TFIFO	0	

Figure 36 Status of Semaphore “ID_Sem1_WatchAdjust” Resource

10.3.3 Solution

Modify the initial_count setting for “ID_Sem1_WatchAdjust” to be 1.



```
semaphore[]{
    name        = ID_Sem1_WatchAdjust;
    wait_queue    = TA_TFIFO;
    initial_count     = 1;        // T
    max_count      = 1;
};
```

Figure 37 Semaphore “ID_Sem1_WatchAdjust” Initial Count

Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.01.10	—	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.