CONFIDENTIAL

Linux Interface Specification Device Driver I2C

User's Manual: Software

R-Car H3/M3/M3N/E3/D3/V3U/V3H Series

Renesas Electronics
www.renesas.com

Rev.3.00   Dec. 2021

Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.

5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
    "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
    "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
    Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall 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. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1   October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas
Electronics Corporation. All trademarks and registered trademarks
are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date
version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademark

- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
- Other company names and product names mentioned herein are registered trademarks or trademarks of their respective owners.
- Registered trademark and trademark symbols (® and ™) are omitted in this document

How to Use This Manual

- **[Readers]**

  This manual is intended for engineers who develop products which use the R-Car H3/M3/M3N/E3/D3/V3U/V3H processor.

- **[Purpose]**

  This manual is intended to give users an understanding of the functions of the R-Car H3/M3/M3N/E3/D3/V3U/V3H processor device driver and to serve as a reference for developing hardware and software for systems that use this driver.

- **[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, and Linux.
    → Read this manual in the order of the CONTENTS.
  — To understand the functions of a multimedia processor for R-Car H3/M3/M3N/E3/D3/V3U/V3H
    → See the R-Car H3/M3/M3N/E3/D3/V3U/V3H User's Manual.
  — To know the electrical specifications of the multimedia processor for R-Car H3/M3/M3N/E3/D3/V3U/V3H
    → See the R-Car H3/M3/M3N/E3/D3/V3U/V3H Data Sheet.

- **[Conventions]**

  The following symbols are used in this manual.

  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

  Numeric representation: Binary ... ××××, 0b××××, or ××××B

  Decimal ... ××××

  Hexadecimal ... 0x×××× or ××××H

  Data type: Double word … 64 bits

  Word … 32 bits

  Half word ... 16 bits

  Byte ... 8 bits

Table of Contents

1. Overview

1.1 Overview

This manual explains the I2C device driver in R-Car H3/M3/M3N/E3/D3/V3U/V3H Linux.

1.2 Function

This module transmits/receives data to/from a device connected to the I2C interface on R-Car H3/M3/M3N/E3/D3/V3U/V3H.

1.2.1 Driver Function

The following table lists the function of this module.

Table 1.2-1 Driver Function

Function	Support status (R-Car H3/M3/M3N/V3U)	Support status (R-Car E3) *1	Support status (R-Car D3)	Support status (R-Car V3H)
Number of channels	7	8	4	6
Channel	Ch0 ~ Ch6	Ch0 ~ Ch6, Ch8	Ch0 ~ Ch3	Ch0 ~ Ch5
Master Mode	Supported			
Slave Mode	Unsupported			
DMA function	Supported (Less than 8 bytes will operate as PIO)			

(*1) Ch7 for R-Car E3 is used in IIC-DVFS. Please refer to "Power Management user's manual" for detail.

1.2.2 Transfer Speed

The following table shows the transfer speed that this module supports.

Table 1.2-2 Transfer speed (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Interface mode	Real transfer speed	Support
Standard mode (100KHz)	100Kbit/s	yes
Fast mode (400KHz)	400Kbit/s	yes
Fast mode plus (1MHz)	1Mbit/s	yes *1

(*1) Only V3U is supported.

1.2.3 Connected Device

This module connects the following device on R-Car H3-SiP/M3-SiP/M3N-SiP/E3/D3/V3U/V3H System Evaluation Board.

Table 1.2-3 Connected device (R-Car H3/M3/M3N)

Channel	Device	Category	Address	Remark
I2C0	EXIO CN B	Connector	-	-
I2C1	EXIO CN A	Connector	-	-
I2C2	AK4613	SSI CODEC	0b0010000x	-
	CS2000	CLK Synthesizer	0b1001111x	Up to 100KHz
	EtherAVB CN	Connector	-	-
	EXIO CN B	Connector	-	-
I2C3	EXIO CN D	Connector	-	
I2C4	PCA9654	I/O Expander	0b0100000x	-
	9FGV0841	CLK Generator	0b1101000x	-
	ADV7482WBBCZ	VIDEO Decoder	0b1110000x	-
	MAX9611	Current-Sense	0b1111111x	-
	MAX9611	Current-Sense	0b1111100x	
	5P49V5923	CLK Generator	0b1101010x	
	EXIO CN C	Connector	-	-
I2C5	Pad	Test point	-	-
I2C6	EXIO CN D	Connector	-	-

Table 1.2-4 Connected device (R-Car E3)

Channel	Device	Category	Address	Remark
I2C0	PCA9654	I/O Expander	0b0100000x	-
	9FGV0841	CLK Generator	0b1101000x	-
	ADV7482WBBCZ	VIDEO Decoder	0b1110000x	-
	ADV7511WBSWZ	HDMI Transmitter	0b0111001x	-
I2C1	-	-	-	-
I2C2	-	-	-	-
I2C3	AK4613	SSI CODEC	0b0010000x	-
	CS2000	CLK Synthesizer	0b1001111x	Up to 100KHz
	EtherAVB CN	Connector	-	-
	BtoB CN	Connector	-	-
	TestIC	Test point	-	-
I2C4	-	-	-	-
I2C5	-	-	-	-
I2C6	-	-	-	-
I2C8	-	-	-	-

Note: Ch7 is used in IIC-DVFS. Please refer to "Power Management user's manual" for detail.

Table 1.2-5   Connected device (R-Car D3)

Channel	Device	Category	Address	Remark
I2C0	BR24T01FVM-W	EEPROM	0b1010000x	-
	ADV7612WBSWZ	HDMI receiver	0b1001100x	-
	ADV7180WBCP32Z	Video processor	0b0100000x	-
	CS2000	Clock synthesizer	0b1001111x	-
	AK4613	SSI Codec	0b0010000x	-
	ADV7511WBSWZ	HDMI transmitter	0b0111001x	-
	MLB CN	Connector	-	-
	EXIO CN A	Connector	-	-
I2C1	EtherAVB CN	Connector	-	-

Table 1.2-6   Connected device (R-Car V3U)

Channel	Device	Category	Address	Remark
I2C0	PCA9654EDTR2G	I/O expander	0b0100000x	
	9FGV0841AKILF	CLK generator	0b1101010x	
	10M04SCM153I7G	MAX 10 FPGA	-	
	BR24G01FVM-3GTTR	EEPROM	0b1010000x	
	5P35023-618NLG2	CLK generator	0b1101001x	
	EXIO CN B	Connector	-	
	5P49V60A554NLG2	CLK generator	0b1101000x	
	GPIO CN	Connector	-	
	BR24G01FVM-3GTTR	EEPROM	0b1010001x	
	Ether	Connector	-	
	BR24G01FVM-3GTTR	EEPROM	0b1010011x	
	CSI_DSI	Connector	-	
	PCA9654EDTR2G	I/O expander	0b0100111x	
	PCA9654EDTR2G	I/O expander	0b0100011x	
	PCA9654EDTR2G	I/O expander	0b0100010x	
	PCA9654EDTR2G	I/O expander	0b0100011x	
I2C1	SN65DSI86ZQER	DSI to eDP	0b0101100x	
	EXIO CN B	Connector	-	
	GPIO CN	Connector	-	
	CSI_DSI	Connector	-	
	MAX20087ATPA/VY+	Power	0b0101000x	
	MAX20087ATPA/VY+	Power	0b0101010x	
	MAX20087ATPA/VY+	Power	0b0101101x	
	MAX96789GTN/V+	Serializer	0b1000000x	
	MAX96712GTB/VY+	DeSerializer	0b1001001x	
	MAX96712GTB/VY+	DeSerializer	0b1001011x	
	MAX96712GTB/VY+	DeSerializer	0b1101011x	

Table 1.2-6   Connected device (R-Car V3U) (Cont.)

I2C2	OcuLink connector	Connector	-	
	EXIO CN B	Connector	-	
	GPIO CN	Connector	-	
I2C3	OcuLink connector	Connector	-	
	EXIO CN B	Connector	-	
	GPIO CN	Connector	-	
I2C4	EXIO CN B	Connector	-	
	GPIO CN	Connector	-	
I2C5	EXIO CN B	Connector	-	
	GPIO CN	Connector		
I2C6	RAA271010	PWM controller	0b1100100x	-
	RAA271001	PMIC	0b1010100x	
			0b1010101x	
	RAA271001	PMIC	0b1011100x	
			0b1011101x	
	R5F10BGGLFB	Micro controller		

Table 1.2-7   Connected device (R-Car V3H)

Channel	Device	Category	Address	Remark
I2C0	PCA9654E	I/O expander	0b0100000x	-
	PCA9654E	I/O expander	0b0100001x	-
	ADV7511W	HDMI Transmitter	0b0111001x	-
	5P35023	Clock generator	0b1101000x	-
	RAA271000	PMIC	0b0011101x	-
			0b0011110x	-
	EthernetAVB PHY connector	Connector	-	-
	EXIO CN A	Connector- CN15	-	-
	EXIO CN C	Connector- CN17	-	-
I2C1	MAX9286	CSI-2 decoder	0b1001000x	-
	MAX9286	CSI-2 decoder	0b1001010x	-
	EXIO CN A	Connector- CN15	-	-
	EXIO CN B	Connector- CN16	-	-
I2C2	EXIO CN A	Connector- CN15	-	-
I2C3	EXIO CN A	Connector- CN15	-	-
I2C4	EXIO CN C	Connector- CN17	-	-
I2C5	EXIO CN A	Connector- CN15	-	-

1.3 Reference

1.3.1 Standards

The following table shows the standard that this module corresponds.

Table 1.3-1   Standard (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Reference Number	Issue	Title	Edition	Date
-	Philips Semiconductors	THE I2C-BUS SPECIFICATION	2.1	Jan. 2000
-	SBS Implementers Forum	System Management Bus (SMBus) Specification	Version 2.0	Aug. 03, 2000

1.3.2 Related Documents

The following table shows the document related to this module.

Table 1.3-2   Related documents (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Number	Issue	Title	Edition	Date
-	Renesas Electronics	R-Car Series, 3rd Generation User's Manual: Hardware	Rev.2.20	Jun. 30, 2020
-	Renesas Electronics	R-CarH3-SiP System Evaluation Board Salvator-X Hardware Manual RTP0RC7795SIPB0011S	Rev.1.09	May. 11, 2017
-	Renesas Electronics	R-CarM3-SiP System Evaluation Board Salvator-X Hardware Manual RTP0RC7796SIPB0011S	Rev.0.04	Oct. 3.2016
-	Renesas Electronics	R-CarH3-SiP/M3-SiP/M3N-SiP System Evaluation Board Salvator-XS Hardware Manual	Rev.2.04	Jul. 17, 2018
-	Renesas Electronics	R-CarE3 System Evaluation Board Ebisu Hardware Manual RTP0RC77990SEB0010S	Rev.0.03	Apr. 11, 2018
-	Renesas Electronics	R-CarE3 System Evaluation Board Ebisu-4D (E3 board 4xDRAM) Hardware Manual	Rev.1.01	Jul. 19, 2018
-	Renesas Electronics	R-CarD3 System Evaluation Board Hardware Manual RTP0RC77995SEB0010S	Rev.1.20	Jul. 25, 2017
-	Renesas Electronics	R-Car V3U Series User's Manual	Rev.0.5	Jul. 31, 2020
-	Renesas Electronics	R-CarV3U System Evaluation Falcon Hardware Manual	Rev.0.01	Sep. 11, 2020
-	Renesas Electronics	R-Car V3H_2, Additional Document for User's Manual: Hardware	Rev.0.50	Jul. 31, 2020
-	Renesas Electronics	R-CarV3H System Evaluation Board Condor-I Hardware Manual	Rev.0.02	Nov. 11,2019

1.4 Restrictions

There is no restriction in this module.

1.5 Notice

- This module supports the ch2 and ch4 of R-Car H3/M3/M3N on Evaluation board.
- This module supports the ch0 and ch3 of R-Car E3 on Evaluation board.
- This module supports the ch0 and ch1 of R-Car D3 on Evaluation board.
- This module supports the ch0 and ch1 of R-Car V3H on Evaluation board.
- Master transfer support. Slave transfer unsupported.
- Usage note for DMA mode of Receive Operation.

  If use DMA mode of Receive Operation more than once with repeated START, issue "STOP condition" and start from beginning of transmission and reception procedure instead of repeated START.

2.   Terminology

The following table shows the terminology related to this module.

Table 1.5-1   Terminology

Terms	Explanation
I2C	Inter-Integrated Circuit

3.   Operating Environment

3.1   Hardware Environment

The following table lists the hardware needed to use this module.

Table 3.1-1   Hardware specification (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Name	Version	Manufacture
R-CarH3-SiP System Evaluation Board Salvator-X	-	Renesas Electronics
R-CarM3-SiP System Evaluation Board Salvator-X	-	Renesas Electronics
R-CarH3-SiP/M3-SiP/M3N-SiP System Evaluation Board Salvator-XS	-	Renesas Electronics
R-CarE3 System Evaluation Board Ebisu	-	Renesas Electronics
R-CarE3 System Evaluation Board Ebisu-4D	-	Renesas Electronics
R-CarD3 System Evaluation Board Draak	-	Renesas Electronics
R-CarV3U System Evaluation Board Falcon	-	Renesas Electronics
R-CarV3H System Evaluation Board Condor-I	-	Renesas Electronics

3.2   Module Configuration

The following figure shows the configuration of this module.

In the case of R-Car H3-Sip/M3-Sip/M3N System Evaluation Board, you can control the connected device using the /dev/i2c-2 or /dev/i2c-4.

Figure 3-1   I2C Driver Module configuration (R-Car H3/M3/M3N)

In the case of R-Car E3 System Evaluation Board, you can control the connected device using the /dev/i2c-0 or /dev/i2c-3.

Figure 3-2   I2C Driver Module configuration (R-Car E3)

In the case of R-Car D3 System Evaluation Board, you can control the connected device using the /dev/i2c-0.
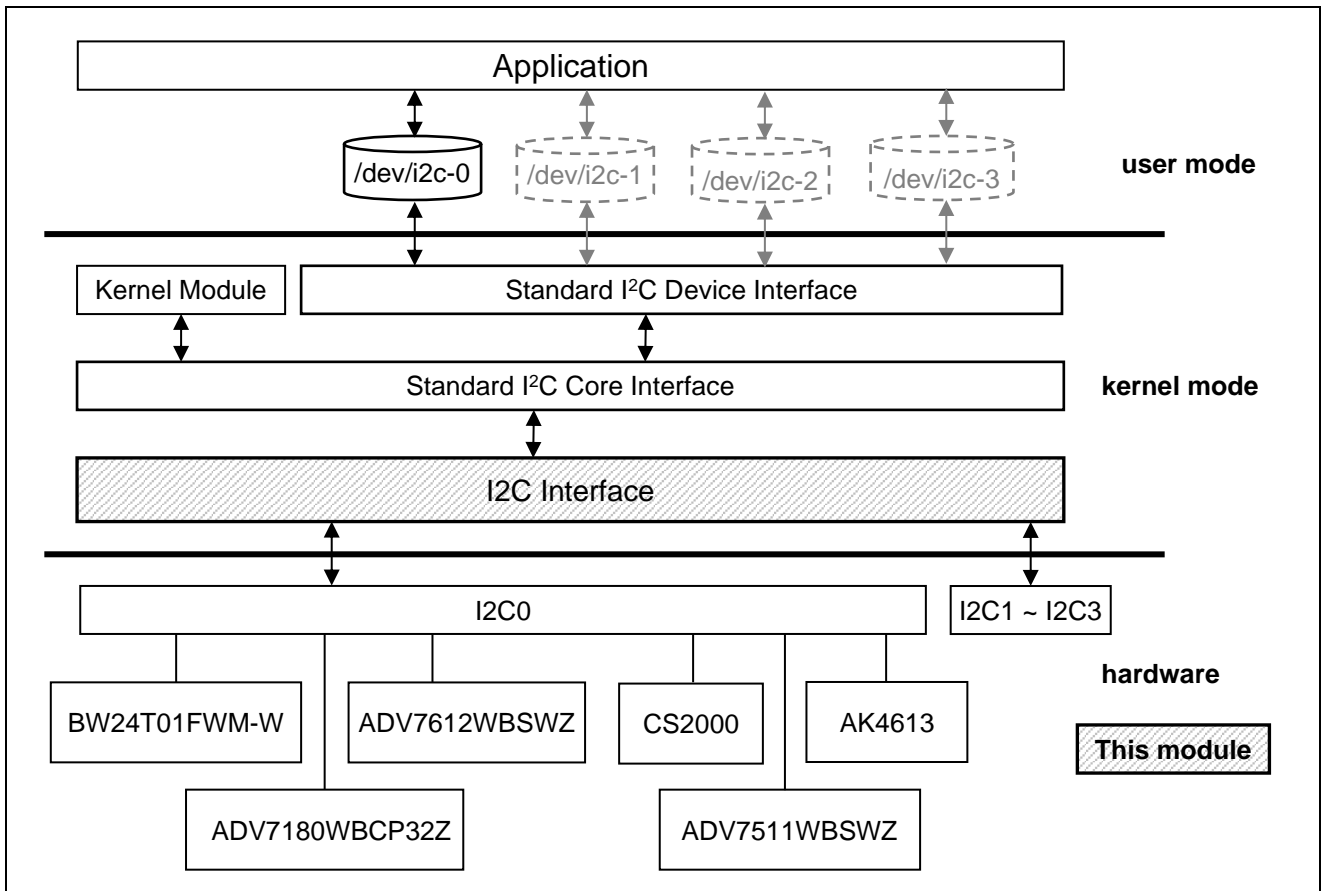
Figure 3-3   I2C Driver Module configuration (R-Car D3)

In the case of R-Car V3U System Evaluation Board, you can control the connected device using the /dev/i2c-0 or /dev/i2c-1

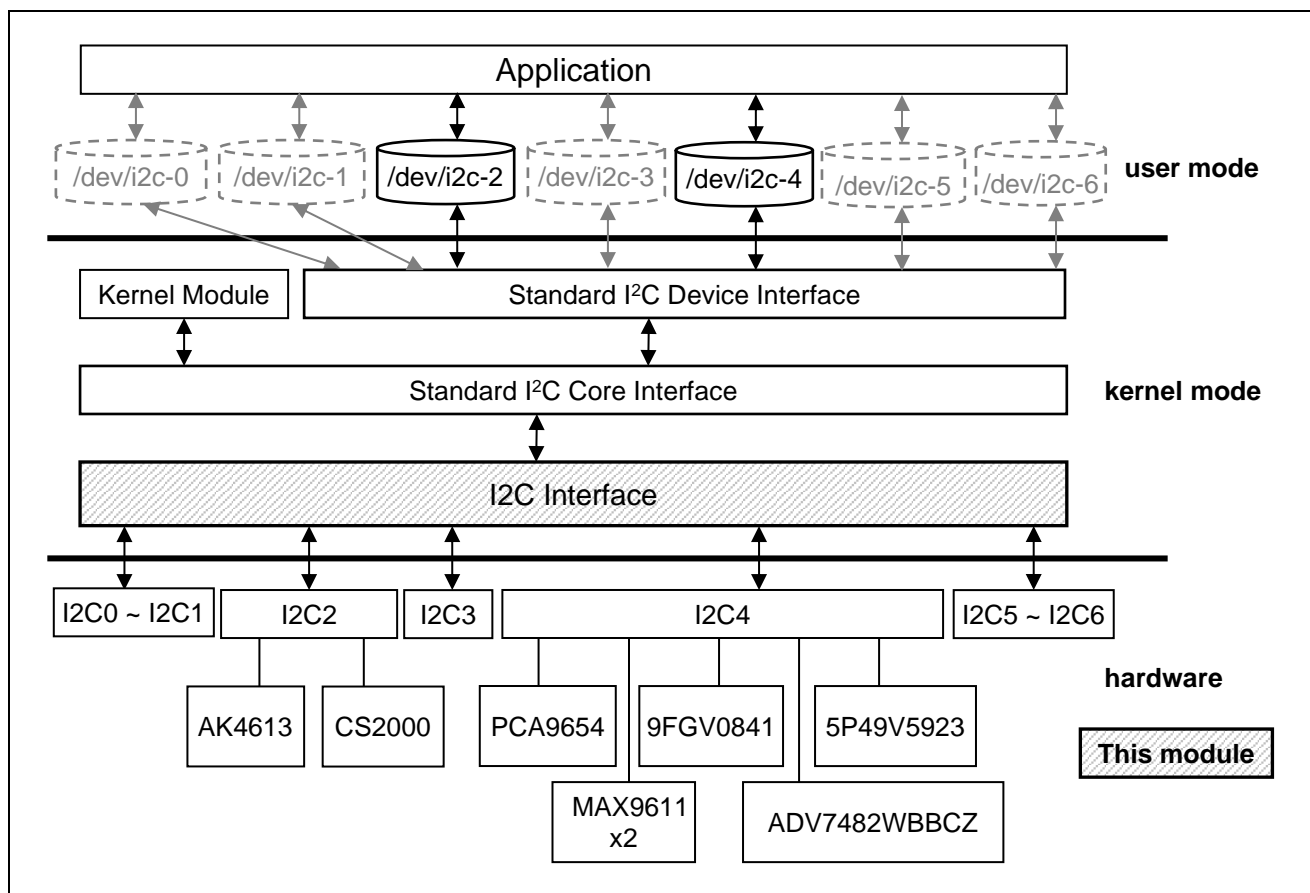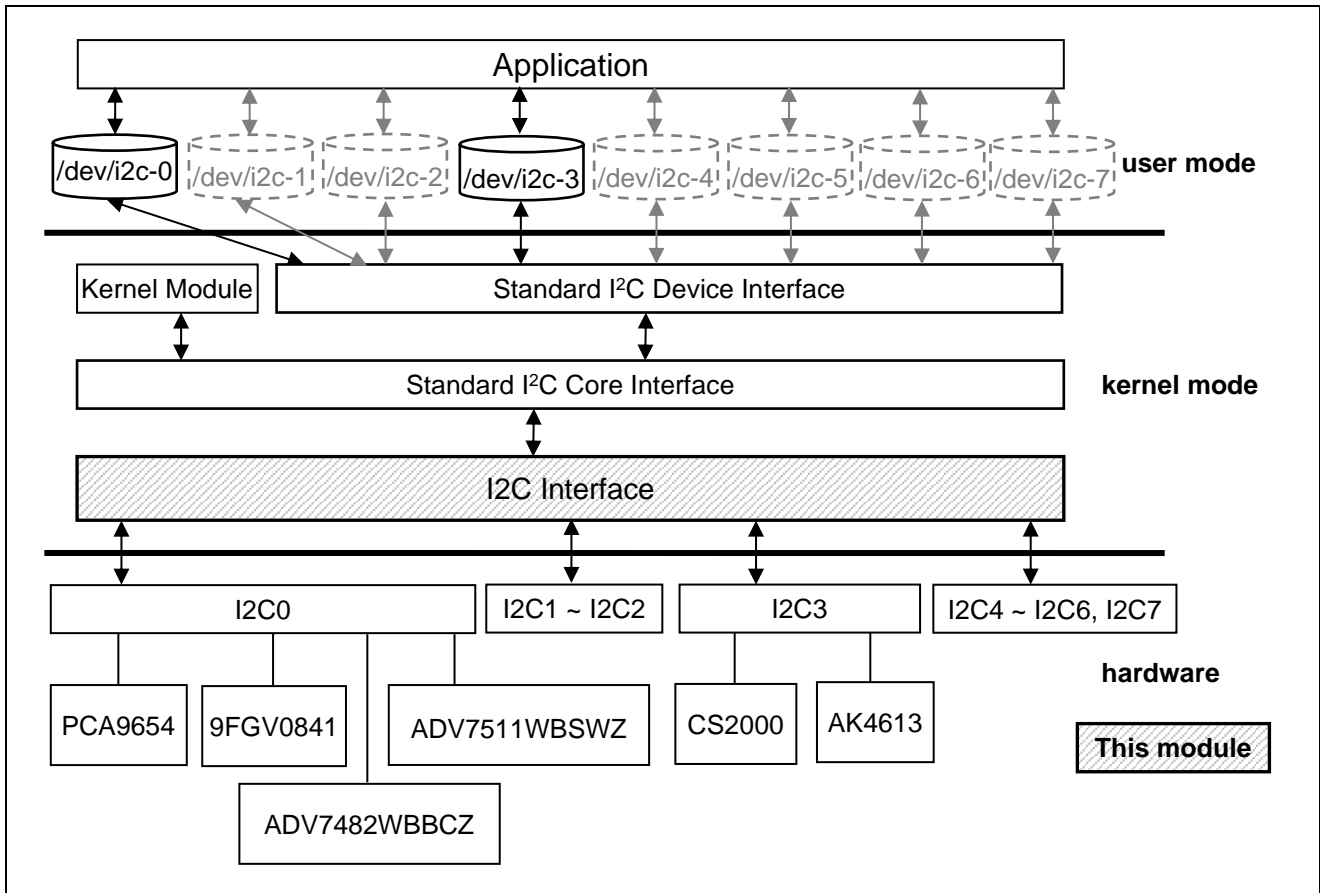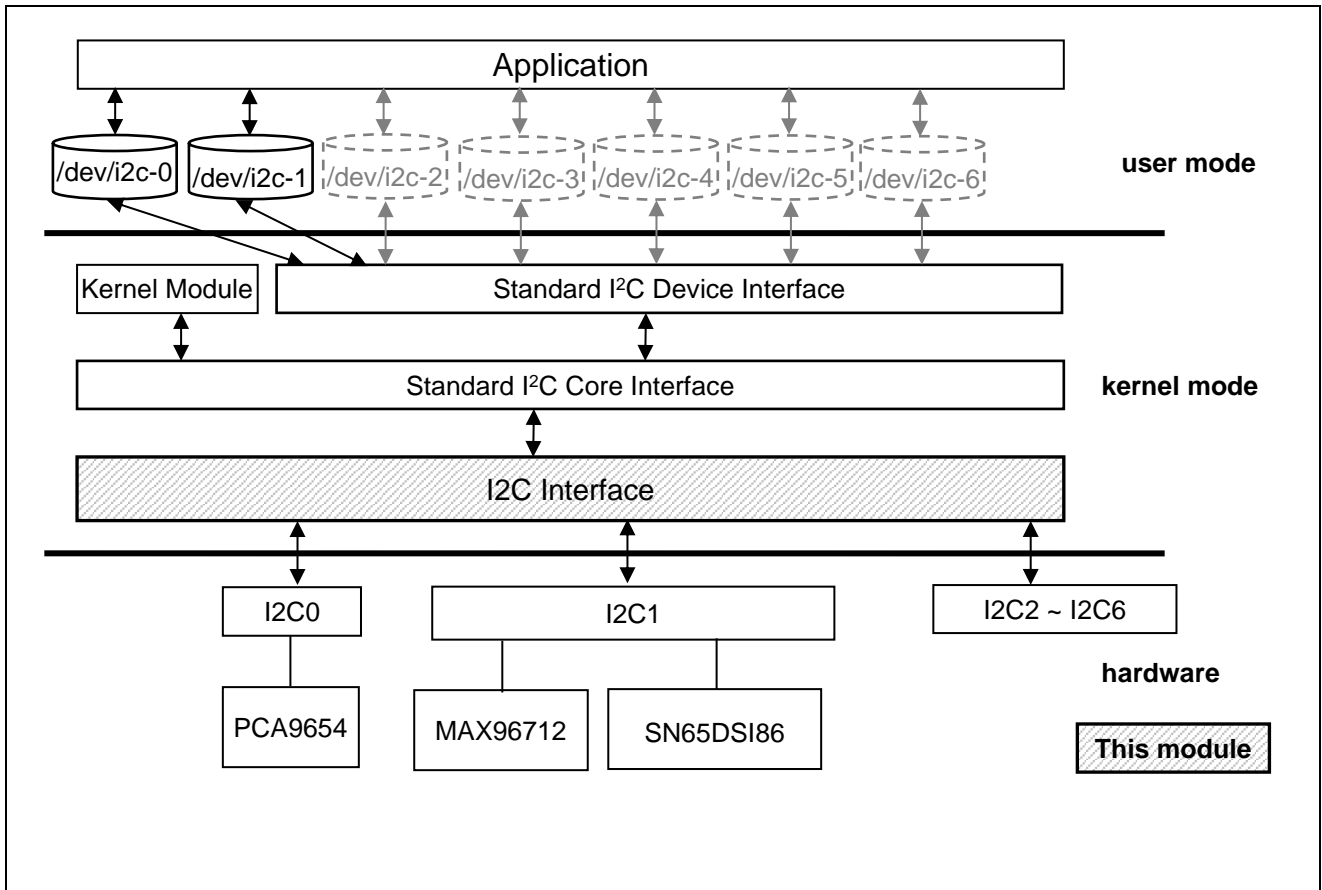**Figure 3-4   I2C Driver Module configuration (R-Car V3U)**

In the case of R-Car V3H System Evaluation Board, you can control the connected device using the /dev/i2c-0 or /dev/i2c-1
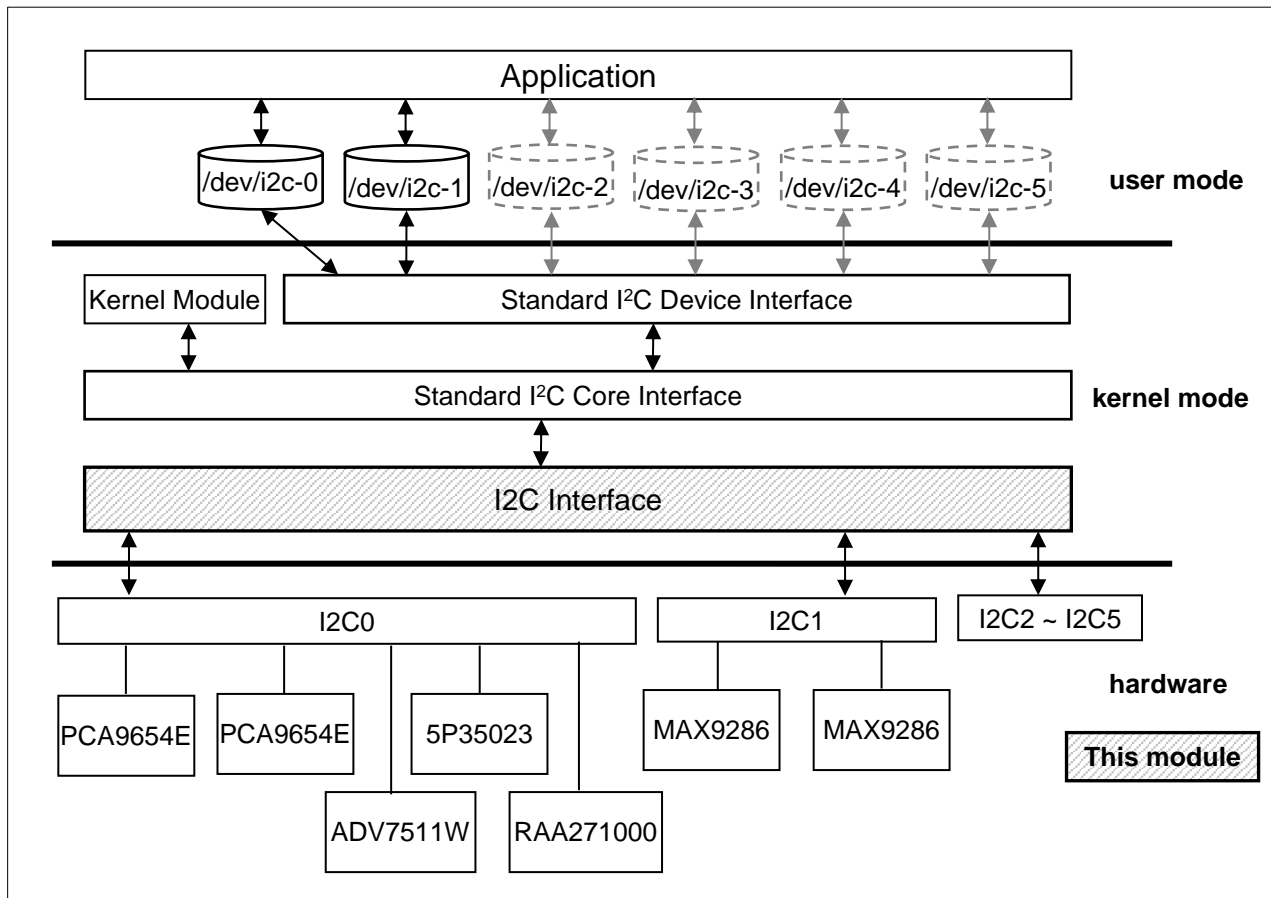
Figure 3.5   I2C Driver Module configuration (R-Car V3H)

3.3   State Transition Diagram

There is no state transition diagram for this module.

4.   External Interface

4.1   Device Node

The following table shows the device node of this module.

Table 4.1-1   I2C device node (R-Car H3/M3/M3N)

Channel	Device node	Major number	Minor number
I2C2	/dev/i2c-2	89	2
I2C4	/dev/i2c-4	89	4

Table 4.1-2   I2C device node (R-Car E3)

Channel	Device node	Major number	Minor number
I2C0	/dev/i2c-0	89	0
I2C3	/dev/i2c-3	89	3

Table 4.1-3   I2C device node (R-Car D3)

Channel	Device node	Major number	Minor number
I2C0	/dev/i2c-0	89	0
I2C1	/dev/i2c-1	89	1

Table 4.1-4   I2C device node (R-Car V3U)

Channel	Device node	Major number	Minor number
I2C0	/dev/i2c-0	89	0
I2C1	/dev/i2c-1	89	1

Table 4.1-5 I2C device node (R-Car V3H)

Channel	Device node	Major number	Minor number
I2C0	/dev/i2c-0	89	0
I2C1	/dev/i2c-1	89	1

4.2   External Function

This section explains in the following format about the functions this module supplies.

[Overview]	Presents an overview of a function.
[Function Name]	Explains the name of the function.
[Calling format]	Explains the format for calling the function.
[Argument]	Explains the argument(s) of the function.
[Return value]	Explains the return value(s) of the function.
[Error number]	Explains the error number(s) of the function.
[Feature]	Explains the features of the function.
[Remark]	Explains points to be noted when using the function.

The following table lists the interface functions in this module, and Standard I2C core Interface.

Table 4.2-1   System calls (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Chapter	Function name	Description
4.2.1	open	Open I2C.
4.2.2	close	Close I2C.
4.2.3	read	Read I2C (8bit data is received).
4.2.4	write	Write I2C (8bit data is sent).
4.2.5	ioctl(I2C_RDWR)	Read/Write processing is performed.
4.2.6	ioctl(I2C_FUNCS)	Return the functionality.
4.2.7	ioctl(I2C_SLAVE)	The slave address is changed.
4.2.8	ioctl(I2C_SMBUS)	It receives and transmits data by SMBus system.

Table 4.2-2   Standard I2C device interface (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Chapter	Function name	Description
4.2.9	i2c_new_probed_device	Probe function.
4.2.10	i2c_register_board_info	Register Board Information.
4.2.11	i2c_add_driver	Addition of I2C client driver.
4.2.12	i2c_del_driver	Deletion of I2C client driver.
4.2.13	i2c_master_send	Single(8bit) data transfer (send).
4.2.14	i2c_master_recv	Single(8bit) data transfer (receive).
4.2.15	i2c_transfer	Execute a single or combined I2C message.
4.2.16	i2c_get_functionality	Return the functionality.
4.2.17	i2c_smbus_read_byte	SMBus "receive byte" protocol.
4.2.18	i2c_smbus_write_byte	SMBus "send byte" protocol.
4.2.19	i2c_smbus_read_byte_data	SMBus "read byte" protocol.
4.2.20	i2c_smbus_write_byte_data	SMBus "write byte" protocol.
4.2.21	i2c_smbus_read_word_data	SMBus "read word(16bit)" protocol.
4.2.22	i2c_smbus_write_word_data	SMBus "write word(16bit)" protocol.
4.2.23	i2c_smbus_read_block_data	SMBus "block read" protocol.
4.2.24	i2c_smbus_write_block_data	SMBus "block write" protocol.
4.2.25	i2c_smbus_read_i2c_block_data	SMBus "read block byte" protocol.
4.2.26	i2c_smbus_write_i2c_block_data	SMBus "write block byte" protocol.

4.2.1 open

[Overview]   Open I2C

[Function Name]   open

[Calling format]   #include <linux/i2c.h>
                    #include <linux/i2c-dev.h>
                    int open(const char *device_name, int flags);

[Arguments]   device_name   Device name (/dev/i2c-X) to open.
                    flags   Open mode.

[Returns]   0   Success
                    -1   Error

[Feature]   Open I2C device.

[Remark]   -

4.2.2 close

[Overview]   Close I2C

[Function Name]   close

[Calling format]   #include <linux/i2c.h>
                    #include <linux/i2c-dev.h>
                    int close(int fd);

[Arguments]   fd   File descriptor

[Returns]   0   Success
                    -1   Error

[Feature]   Close I2C device.

[Remark]   -

4.2.3   read

[Overview]   Read I2C(8bit data)

[Function Name]   read

[Calling format]   #include <linux/i2c.h>
                    #include <linux/i2c-dev.h>
                    ssize_t read(int fd, void *buf, size_t count);

[Arguments]   fd   File descriptor
                    buf   Read data stock area
                    count   Read size

[Returns]   Positive value Success (Read size)
                    -1   Error

[Feature]   Read I2C(8bit data).

[Remark]   -

4.2.4   write

[Overview]   Write I2C(8bit data)

[Function Name]   write

[Calling format]   #include <linux/i2c.h>
                    #include <linux/i2c-dev.h>
                    ssize_t write(int fd, const void *buf, size_t count);

[Arguments]   fd   File descriptor
                    buf   Write data stock area
                    count   Write size

[Returns]   Positive value Success (Write size)
                    -1   Error

[Feature]   Write I2C(8bit data).

[Remark]   -

4.2.5   ioctl(I2C_RDWR)

[Overview]   Read/Write transaction

[Function Name]   ioctl(I2C_RDWR)

[Calling format]   #include <linux/i2c.h>
                    #include <linux/i2c-dev.h>
                    int ioctl(int fd, I2C_RDWR, struct i2c_rdwr_ioctl_data *msgset);

[Arguments]   fd   File descriptor
                    I2C_RDWR   Fixed control
                    msgset   Send/Receive data

[Returns]   0   Success
                    -1   Error

[Error number]   -EFAULT   Bad address
                    -EINVAL   Invalid argument
                    -ENOMEM   There is not enough memory area
                    -EAGAIN   Arbitration lost occurs on I2C bus
                    -EOPNOTSUPP   Operation not supported
                    -ETIMEDOUT   Timeout occurs during I2C device access
                    -ENXIO   NACK occurs on I2C bus
                    -EBUSY   Busy on I2C bus

[Feature]   Do read/write transaction without stop.
                    Only valid if the adapter has I2C_FUNC_I2C.

[Remark]   Refer 4.3.6 about i2c_rdwr_ioctl_data structure.

4.2.6   ioctl(I2C_FUNCS)

[Overview]	Return the functionality	
[Function Name]	ioctl(I2C_FUNCS)	
[Calling format]	#include <linux/i2c.h>	
	#include <linux/i2c-dev.h>	
	int ioctl(int fd, I2C_FUNCS, unsigned long *funcs);	
[Arguments]	fd	File descriptor
	I2C_FUNCS	Fixed control
	funcs	The index of function support
[Returns]	0	Success
	-1	Error
[Error number]	-EFAULT	Bad address
[Feature]	This function returns whether the function is supported (the command).	
	Refer to 4.2.16 in detail.	
[Remark]	-	

4.2.7   ioctl(I2C_SLAVE)

[Overview]	Change the slave address	
[Function Name]	ioctl(I2C_SLAVE)	
[Calling format]	#include <linux/i2c.h>	
	#include <linux/i2c-dev.h>	
	int ioctl(int fd, I2C_SLAVE, long slave);	
[Arguments]	fd	File descriptor
	I2C_SLAVE	Fixed control
	slave	Slave address
[Returns]	0	Success
	-1	Error
[Error number]	-EINVAL	Invalid argument
	-EBUSY	Busy on I2C bus
[Feature]	Change the slave address.	
[Remark]	-	

4.2.8   ioctl(I2C_SMBUS)

[Overview]   Send/Receive data by SMBus system

[Function Name]   ioctl(I2C_SMBUS)

[Calling format] #include <linux/i2c.h>
                  #include <linux/i2c-dev.h>
                  int ioctl(int fd, I2C_SMBUS, struct i2c_smbus_ioctl_data *args);

[Arguments]   fd   File descriptor
                  I2C_SMBUS   Fixed control
                  args   Send/Receive data.

[Returns]   0   Success
                  -1   Error

[Error number]   -EFAULT   Bad address
                  -EINVAL   Invalid argument
                  -EAGAIN   Arbitration lost occurs on I2C bus
                  -ETIMEDOUT   Timeout occurs during I2C device access
                  -ENXIO   NACK occurs on I2C bus
                  -EBUSY   Busy on I2C bus

[Feature]   SMBus transfer

[Remark]   Refer to 4.3.7 about i2c_smbus_ioctl_data structure.

4.2.9   i2c_new_probed_device

[Overview]	Probe function	

[Function Name]   i2c_new_probed_device

[Calling format]   #include <linux/i2c.h>
                         struct i2c_client * i2c_new_probed_device(struct i2c_adapter *adap,
                                       struct i2c_board_info *info, unsigned short const *addr_list)

[Arguments]   adap   Adaptor information.
                       info   Board information.
                       addr_list   Address list to probe.

[Returns]   Positive Number   Success (The pointer to client information)
                       NULL   Error

[Feature]   The probe process for the connected device.

[Remark]   Refer to the following chapter about i2c_client、i2c_adapter、i2c_board_info structure.
                       4.3.1 struct i2c_adapter
                       4.3.2 struct i2c_client
                       4.3.3 struct i2c_board_info

4.2.10   i2c_register_board_info

[Overview]   Register board information

[Function Name]   i2c_register_board_info

[Calling format]   #include <linux/i2c.h>
                       int __init i2c_register_board_info(int busnum,
                                       struct i2c_board_info const *info, unsigned len);

[Arguments]   busnum   A number of the bus connected device.
                       info   The vector of I2C device descriptor.
                       len   The number of descriptor included vector.

[Returns]   Positive Number   Success (A number of the registered board information)
                       Negative number   Error

[Feature]   Register board information.

[Remark]   This function supports the board dependent part call only.
                       Refer to 4.3.3 about i2c_board_info structure.

4.2.11   i2c_add_driver

[Overview]   Addition of I2C client driver

[Function Name]   i2c_add_driver

[Calling format]   #include <linux/i2c.h>
                      int i2c_add_driver(struct i2c_driver *driver);

[Arguments]   driver   The information of client driver.

[Returns]   0   Success
                      Other value   Error

[Feature]   Addition of I2C client driver.

[Remark]   Refer to 4.3.4 about i2c_driver structure.

4.2.12   i2c_del_driver

[Overview]   Deletion of I2C client driver

[Function Name]   i2c_del_driver

[Calling format]   #include <linux/i2c.h>
                      void i2c_del_driver(struct i2c_driver *driver);

[Arguments]   driver   The information of client driver.

[Returns]   -

[Feature]   Deletion of I2C client driver.

[Remark]   Refer to 4.3.4 about i2c_driver structure.

4.2.13   i2c_master_send

[Overview]   Single(8bit) data transfer in master transmit mode

[Function Name]   i2c_master_send

[Calling format]   #include <linux/i2c.h>
                      int i2c_master_send(struct i2c_client *client, const char *buf, int count);

[Arguments]   client   The information of slave device.
                      buf   The data area sent to slave device.
                      count   Sent data size.

[Returns]   Positive Number   Success (Sent data size)
                      Negative Number   Error

[Feature]   Single(8bit) data transfer in master transmit mode.

[Remark]   This function cannot be called from the interrupt handler.
                      4.3.2 struct i2c_client

4.2.14   i2c_master_recv

[Overview]   Single(8bit) data transfer in master receive mode

[Function Name]   i2c_master_recv

[Calling format]   #include <linux/i2c.h>
                      int i2c_master_recv(struct i2c_client *client, char *buf, int count);

[Arguments]   client   The information of slave device.
                      buf   The data area received from slave device.
                      count   Received data size.

[Returns]   Positive Number   Success (Received data size)
                      Negative Number   Error

[Feature]   Single(8bit) data transfer in master receive mode

[Remark]   This function cannot be called from the interrupt handler.
                      Refer to 4.3.2 about i2c_client.

4.2.15   i2c_transfer

[Overview]	Execute a single or combined I2C message	
[Function Name]	i2c_transfer	
[Calling format]	#include <linux/i2c.h> int i2c_transfer(struct i2c_adapter * adap, struct i2c_msg *msgs, int num);	
[Arguments]	adap	The information of I2C bus
	msgs	One or more messages which will be performed before operation stops by STOP. Each message begins with a START.
	num	A number of messages to be executed.
[Returns]	Positive Number	Success (A number of transmitted messages)
	Negative Number	Error
[Feature]	Execute a single or combined I2C message.	
[Remark]	This function cannot be called from the interrupt handler. Refer to the following chapter about i2c_adapter and i2c_msg structure. 4.3.1 struct i2c_adapter 4.3.5 struct i2c_msg	

4.2.16   i2c_get_functionality

[Overview]	Return the functionality	
[Function Name]	i2c_get_functionality	
[Calling format]	#include <linux/i2c.h> u32 i2c_get_functionality(struct i2c_adapter *adap);	
[Arguments]	adap	The information of I2C bus
[Returns]	The support information of function (Refer to Table 4.4 in detail) In the "AND" result of the following bit mask to the return value, "0" is unsupported function, "1" is support function.	
[Feature]	Return the functionality.	
[Remark]	This function cannot be called from the interrupt handler. Refer to 4.3.1 about i2c_adapter structure.	

Table 4.2-3 The support information of function (R-Car H3/M3/M3N/E3/D3/V3U/V3H) (1)

Bit mask	Function name
	Function
0x00000001	I2C_FUNC_I2C
	Plane I2C command (Not execute in SMBus adapter)
0x00000002	I2C_FUNC_10BIT_ADDR
	Extend 10 bits address
0x00000004	I2C_FUNC_PROTOCOL_MANGLING
	Identify the flag of I2C_M_REV_DIR_ADDR, I2C_M_REV_DIR_ADDR andI2C_M_REV_DIR_NOSTART.
0x00000008	I2C_FUNC_SMBUS_PEC
	SMBus pec command
0x00008000	I2C_FUNC_SMBUS_BLOCK_PROC_CALL
	SMBus block_process_call command
0x00010000	I2C_FUNC_SMBUS_QUICK
	SMBus write_quick command
0x00020000	I2C_FUNC_SMBUS_READ_BYTE
	SMBus read_byte command
0x00040000	I2C_FUNC_SMBUS_WRITE_BYTE
	SMBus write_byte command
0x00080000	I2C_FUNC_SMBUS_READ_BYTE_DATA
	SMBus read_byte_data command
0x00100000	I2C_FUNC_SMBUS_WRITE_BYTE_DATA
	SMBus write_byte_data command
0x00200000	I2C_FUNC_SMBUS_READ_WORD_DATA
	SMBus read_word_data command
0x00400000	I2C_FUNC_SMBUS_WRITE_WORD_DATA
	SMBus write_byte_data command
0x00800000	I2C_FUNC_SMBUS_PROC_CALL
	SMBus process_call command
0x01000000	I2C_FUNC_SMBUS_READ_BLOCK_DATA
	SMBus read_block_data command
0x02000000	I2C_FUNC_SMBUS_WRITE_BLOCK_DATA
	SMBus write_block_data command
0x04000000	I2C_FUNC_SMBUS_READ_I2C_BLOCK
	SMBus read_i2c_block_data command
0x08000000	I2C_FUNC_SMBUS_WRITE_I2C_BLOCK
	SMBus write_i2c_block_data command
0x00060000	I2C_FUNC_SMBUS_BYTE
	SMBus read_byte and write_byte command
0x00180000	I2C_FUNC_SMBUS_BYTE_DATA
	SMBus read_byte_data and write_byte_data command

Table 4.2-4   The support information of function (R-Car H3/M3/M3N/E3/D3/V3U/V3H) (2)

Bit mask	Function name
	Function
0x00600000	I2C_FUNC_SMBUS_WORD_DATA
	SMBus read_word_data and write_word_data command
0x03000000	I2C_FUNC_SMBUS_BLOCK_DATA
	SMBus read_block_data and write_block_data command
0x0C000000	I2C_FUNC_SMBUS_I2C_BLOCK
	SMBus read_i2c_block_data and write_i2c_block_data command
0x02FF0008	I2C_FUNC_SMBUS_EMUL
	All SMBus commands that can be emulated by a real I2C adapter (using the transparent emulation layer)

4.2.17   i2c_smbus_read_byte

[Overview]   SMBus "receive byte" protocol

[Function Name]   i2c_smbus_read_byte

[Calling format]   #include <linux/i2c.h>
                       s32 i2c_smbus_read_byte(struct i2c_client *client);

[Arguments]   client   The information of slave device.

[Returns]   Positive Number   Success (Received data size)
                       Negative Number   Error

[Feature]   SMBus "receive byte" protocol.
                       Without specifying a register of a device, this function is received 8 bits.

[Remark]   This function cannot be called from the interrupt handler.
                       Refer to 4.3.2 about i2c_client.

4.2.18   i2c_smbus_write_byte

[Overview]   Single(8bit) "send byte" protocol

[Function Name]   i2c_smbus_write_byte

[Calling format]   #include <linux/i2c.h>
                       s32 i2c_smbus_write_byte(struct i2c_client *client, u8 value);

[Arguments]   client   The information of slave device.
                       value   Sent data size

[Returns]   0   Success
                       Negative Number   Error

[Feature]   SMBus "send byte" protocol.
                       Without specifying a register of a device, this function is sent 8 bits.

[Remark]   This function cannot be called from the interrupt handler.
                       Refer to 4.3.2 about i2c_client.

4.2.19   i2c_smbus_read_byte_data

[Overview]   SMBus "read byte" protocol

[Function Name]   i2c_smbus_read_byte_data

[Calling format]   #include <linux/i2c.h>
                     s32 i2c_smbus_read_byte_data(struct i2c_client *client, u8 command);

[Arguments]   client   The information of slave device.
                     command   The command interpreted by slave.

[Returns]   Positive Number   Success (Read data size)
                     Negative Number   Error

[Feature]   This function is received 8 bits data from the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                     Refer to 4.3.2 about i2c_client.

4.2.20   i2c_smbus_write_byte_data

[Overview]   SMBus "write byte" protocol

[Function Name]   i2c_smbus_write_byte_data

[Calling format]   #include <linux/i2c.h>
                     s32 i2c_smbus_write_byte_data(struct i2c_client *client, u8 command, u8 value);

[Arguments]   client   The information of slave device.
                     command   The command interpreted by slave.
                     value   Write data size

[Returns]   0   Success
                     Negative Number   Error

[Feature]   This function is sent 8 bits data to the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                     Refer to 4.3.2 about i2c_client.

4.2.21   i2c_smbus_read_word_data

[Overview]   SMBus "read word(16bit)" protocol

[Function Name]   i2c_smbus_read_word_data

[Calling format]   #include <linux/i2c.h>
                    s32 i2c_smbus_read_word_data(struct i2c_client *client, u8 command);

[Arguments]   client   The information of slave device.
                    command   The command interpreted by slave.

[Returns]   Positive Number   Success (Read data size(16bit unsigned word))
                    Negative Number   Error

[Feature]   This function is received 16 bits data from the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                    Refer to 4.3.2 about i2c_client.

4.2.22   i2c_smbus_write_word_data

[Overview]   SMBus "write word(16bit)" protocol

[Function Name]   i2c_smbus_write_word_data

[Calling format]   #include <linux/i2c.h>
                    s32 i2c_smbus_write_word_data(struct i2c_client *client, u8 command, u16 value);

[Arguments]   client   The information of slave device.
                    command   The command interpreted by slave.
                    value   Write data size (16bit word)

[Returns]   0   Success
                    Negative Number   Error

[Feature]   This function is sent 16 bits data to the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                    Refer to 4.3.2 about i2c_client.

4.2.23   i2c_smbus_read_block_data

[Overview]   SMBus "block read" protocol

[Function Name]   i2c_smbus_read_block_data

[Calling format]   #include <linux/i2c.h>
                        s32 i2c_smbus_read_block_data(struct i2c_client *client, u8 command, u8 *values);

[Arguments]   client   The information of slave device.
                        command   The command interpreted by slave.
                        values   The pointer to store read data.

[Returns]   Positive Number   Success (Read data block size)
                        Negative Number   Error

[Feature]   This function is received data block from the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                        Refer to 4.3.2 about i2c_client.

4.2.24   i2c_smbus_write_block_data

[Overview]   SMBus "block write" protocol

[Function Name]   i2c_smbus_write_block_data

[Calling format]   #include <linux/i2c.h>
                        s32 i2c_smbus_write_block_data(struct i2c_client *client, u8 command,
                                        u8 length, const u8 *values);

[Arguments]   client   The information of slave device.
                        command   The command interpreted by slave.
                        length   Data block size.
                        value   The pointer to store write data.

[Returns]   0   Success
                        Negative Number   Error

[Feature]   This function is sent data block to the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                        Refer to 4.3.2 about i2c_client.

4.2.25   i2c_smbus_read_i2c_block_data

[Overview]   SMBus "read block byte" protocol

[Function Name]   i2c_smbus_read_i2c_block_data

[Calling format]   #include <linux/i2c.h>
                    s32 i2c_smbus_read_i2c_block_data(struct i2c_client *client, u8 command,
                                u8 length, u8 *values);

[Arguments]   client   The information of slave device.
                    command   The command interpreted by slave.
                    length   Data block size.
                    values   The pointer to store read data.

[Returns]   Positive Number   Success (Read data size)
                    Negative Number   Error

[Feature]   This function is received data block from the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                    Refer to 4.3.2 about i2c_client.

4.2.26   i2c_smbus_write_i2c_block_data

[Overview]   SMBus "write block byte" protocol

[Function Name]   i2c_smbus_write_i2c_block_data

[Calling format]   #include <linux/i2c.h>
                    s32 i2c_smbus_write_i2c_block_data(struct i2c_client *client, u8 command,
                                u8 length, const u8 *values);

[Arguments]   client   The information of slave device.
                    command   The command interpreted by slave.
                    length   Data block size.
                    value   The pointer to store write data.

[Returns]   0   Success (Write data size)
                    Negative Number   Error

[Feature]   This function is sent data block to the specified register of a device.

[Remark]   This function cannot be called from the interrupt handler.
                    Refer to 4.3.2 about i2c_client.

4.3　Structure

Structure of this module is based on Linux v5.10.41

4.3.1　struct i2c_adapter

Table 4.3-1　struct i2c_adapter (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	Member name	Overview
i2c_adapter	struct module *	owner	The owner of this module
	unsigned int	class	The type of I2C device supported by this driver
	const struct i2c_algorithm *	algo	The pointer of the algorithm to access the bus
	void *	algo_data	The algorithm data
	const struct i2c_lock_operations*	lock_ops	Lock operations
	struct rt_mutex	bus_lock	The structure specified rt_mutex
	struct rt_mutex	mux_lock	The structure specified rt_mutex
	int	timeout	Timeout value
	int	retries	The retry number
	struct device	dev	The adapter device
	unsigned long	locked_flags	owned by the I2C core
	int	nr	The adapter ID
	char	name[48]	The name of I2C device driver
	struct completion	dev_released	The structure used to maintain the state of "completion"
	struct mutex	userspace_clients_lock	The mutex of client
	struct list_head	userspace_clients	The list of client
	struct i2c_bus_recovery_info *	bus_recovery_info	The pointer of the information for bus recovery
	const struct i2c_adapter_quirks *	quirks	describe flaws of the i2c adapter
	struct irq_domain *	host_notify_domain	Recovery

4.3.2   struct i2c_client

Table 4.3-2   struct i2c_client (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	**Member name**	**Overview**
i2c_client	unsigned short	flags	The support function flag of client
	unsigned short	addr	The slave address
	char	name[I2C_NAME_SIZE]	The client name
	struct i2c_adapter *	adapter	The adapter information
	struct device	dev	The driver model device node for the slave
	int	init_irq	irq set at initialization
	int	irq	The interrupt number used by the device
	struct list_head	detected	The member of i2c_driver.clients list

4.3.3   struct i2c_board_info

Table 4.3-3   struct i2c_board_info (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	**Member name**	**Overview**
i2c_board_info	char	type[I2C_NAME_SIZE]	The chip type to initialize i2c_client.name
	unsigned short	flags	The flag to initialize i2c_client.flags
	unsigned short	addr	The device address
	const char *	dev_name	The device name
	void *	platform_data	The Platform Data of the device
	struct device_node *	of_node	The information of the device node
	struct fwnode_handle *	fwnode	Device node supplied by the platform firmware
	const struct property_entry *	properties	Additional device properties for the device
	const struct resource *	resources	Resources associated with the device
	unsigned int	num_resources	Number of resources in the @resources array
	int	irq	The interrupt number used by the device

4.3.4   struct i2c_driver

Table 4.3-4   struct i2c_driver (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	**Member name**	**Overview**
i2c_driver	unsigned int	class	The kind of the I2C device created the instance
	int (*probe)(struct i2c_client *, const struct i2c_device_id *)	probe	Callback for device binding
	int (*remove)(struct i2c_client *)	remove	Callback for device unbinding
	int (*probe_new)(struct i2c_client *)	probe_new	Callback for device probing
	void (*shutdown)(struct i2c_client *)	shutdown	Callback for device shutdown

	void (*alert)(struct i2c_client *,
	                enum i2c_alert_protocol protocol,
	                unsigned int data)
		alert	Alert callback

	int (*command)(struct i2c_client *client, unsigned int cmd, void *arg)	command	Callback for bus-wide signaling (optional)
	struct device_driver	driver	The device driver structure
	const struct i2c_device_id *	id_table	List of I2C devices supported by this driver
	int (*detect)(struct i2c_client *, struct i2c_board_info *)	detect	Callback for device detection
	const unsigned short *	address_list	The I2C addresses to probe (for detect)
	struct list_head	clients	List of detected clients we created (for i2c-core use only)

4.3.5   struct i2c_msg

Table 4.3-5   struct i2c_msg (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	Member name	Overview
i2c_msg	__u16	addr	The slave address
	__u16	flags	Specify R/W flag
	__u16	len	The message length
	__u8*	buf	The pointer to the message data

4.3.6   struct i2c_rdwr_ioctl_data

Table 4.3-6   struct i2c_rdwr_ioctl_data (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	Member name	Overview
i2c_rdwr_ioctl_data	struct i2c_msg *	msgs	The pointer to i2c_msg structure
	__u32	nmsgs	A number of i2c_msg

4.3.7   struct i2c_smbus_ioctl_data

Table 4.3-7   struct i2c_smbus_ioctl_data (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Structure name	Member		
	Type	Member name	Overview
i2c_smbus_ioctl_data	__u8	read_write	Specify R/W flag
	__u8	command	The slave address
	__u32	size	Data type (size)
	union i2c_smbus_data *	data	The pointer to the data

4.3.8   union i2c_smbus_data

Table 4.3-8   union i2c_smbus_data (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Union name	Member		
	Type	Member name	Overview
i2c_smbus_data	__u8	byte	8bit data buffer
	__u16	word	16bit data buffer
	__u8	block[I2C_SMBUS_BLOCK_MAX + 2]	Block data buffer

4.4   Global Variables and Constants

4.4.1   Global Variables

There are no global variables for this module.

4.4.2   Global Constants

The following table shows the global constants used by standard I2C core.
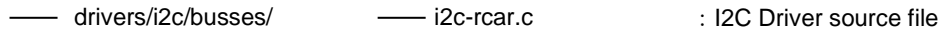
Table 4.4-1   List of Global constants (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Global Constant Name	Value	Remark
I2C_CLIENT_END	0xfffeU	-
I2C_CLIENT_PEC	0x04	-
I2C_CLIENT_TEN	0x10	-
I2C_CLIENT_WAKE	0x80	-
I2C_FUNC_SMBUS_QUICK	0x00010000	-
I2C_FUNC_SMBUS_READ_BYTE	0x00020000	-
I2C_MODULE_PREFIX	"i2c:"	-
I2C_M_RD	0x0001	-
I2C_M_RECV_LEN	0x0400	-
I2C_M_TEN	0x0010	-
I2C_NAME_SIZE	20	-
I2C_SMBUS_BYTE	1	-
I2C_SMBUS_BYTE_DATA	2	-
I2C_SMBUS_BLOCK_DATA	5	-
I2C_SMBUS_BLOCK_MAX	32	-
I2C_SMBUS_BLOCK_PROC_CALL	7	-
I2C_SMBUS_I2C_BLOCK_DATA	8	-
I2C_SMBUS_PROC_CALL	4	-
I2C_SMBUS_QUICK	0	-
I2C_SMBUS_READ	1	-
I2C_SMBUS_WRITE	0	-
I2C_SMBUS_WORD_DATA	3	-
I2C_RDWR	0x0707	-
I2C_FUNCS	0x0705	-
I2C_SLAVE	0x0703	-
I2C_SMBUS	0x0720	-

5. Integration

5.1 Directory Configuration

The directory configuration is shown below.
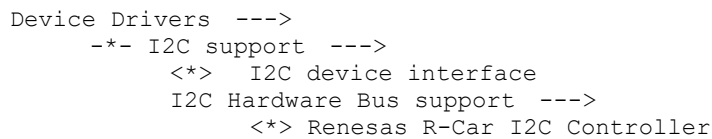
```
——— drivers/i2c/busses/        ——— i2c-rcar.c          : I2C Driver source file
```

Figure 5-1 Directory configuration (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

5.2 Integration Procedure

To enable the function of this module, make the following setting with Kernel Configuration.

```
Device Drivers  --->
    -*- I2C support  --->
          <*>   I2C device interface
          I2C Hardware Bus support  --->
                <*> Renesas R-Car I2C Controller
```

Figure 5-2 Kernel configuration (R-Car H3/M3/M3N/E3/D3/V3U/V3H)

5.3   Option Setting

5.3.1   Module Parameters

5.3.1.1 DT

When operation is in master mode, the SCL clock ratio is generated from an internal clock.
The following equation is used.

SCLfreq = I2Cck/(20 + SCGD × 8 + F[(tICF + tr + IntDelay) × I2Cck])
  I2Cck: I2C internal clock frequency (133.33MHz[R-Car H3/M3/M3N/E3/D3/V3U/V3H])
  SCGD: SCL Clock Generation Divider (Calculated by the driver)
  F[n]:   n rounded down to an integer

Table 5.3-1   Device Tree and Formula Parameter(R-Car H3/M3/M3N/E3/D3/V3U/V3H)

Device Tree property	Formula Parameter	Description	Default value
clock-frequency	SCLfreq	Frequency of bus clock in Hz	100000
i2c-scl-rising-time-ns	tr	I2C SCL rising time (depending on external load)	200
i2c-scl-falling-time-ns	tICF	I2C SCL falling time (depending on external load)	35
i2c-scl-internal-delay-ns	IntDelay	LSI internal delay corresponds to output buffer type.	50

Please set I2C transfer speed 400000 or 100000 to "clock-frequency" of device tree file (salvator-common.dtsi) in
arch/arm64/boot/dts/renesas directory. No setting means 100000 Hz.

```
&i2c2 {
      status = "okay";
      clock-frequency = <100000>;
      ...
}

&i2c4 {
      status = "okay";
      clock-frequency = <400000>;
      ...
}
```

Figure 5-3   Example of setting I2C transfer speed (R-Car H3/M3/M3N)

I2C0, I2C3, I2C4 and I2C5 have Open drain buffer.
Their IntDelay are set 110ns in arch/arm64/boot/dts/renesas/r8a7795.dtsi, r8a7796.dtsi, r8a77965.dtsi.
I2C1, I2C2 and I2C6 have LVTTL (low drive only) buffer.
Their IntDelay are set 6 ns in arch/arm64/boot/dts/renesas/r8a7795.dtsi, r8a7796.dtsi, r8a77965.dtsi.

Please set I2C transfer speed 400000 or 100000 to "clock-frequency" of device tree file (r8a77990-ebisu.dts, r8a77990-es10-ebisu.dts) in arch/arm64/boot/dts/renesas directory. No setting means 100000 Hz.

```
&i2c0 {
    status = "okay";
    clock-frequency = <400000>;
    ...
}

&i2c3 {
    status = "okay";
    clock-frequency = <100000>;
    ...
}
```

Figure 5-4   Example of setting I2C transfer speed (R-Car E3)

I2C0, I2C3 have Open drain buffer.
Their IntDelay are set 110ns in arch/arm64/boot/dts/renesas/r8a77990.dtsi.
I2C1, I2C2, I2C4, I2C5, I2C6 and I2C8 have LVTTL (low drive only) buffer.
Their IntDelay are set 6ns in arch/arm64/boot/dts/renesas/r8a77990.dtsi.

Please set I2C transfer speed 400000 or 100000 to "clock-frequency" of device tree file (r8a77995-draak.dts) in arch/arm64/boot/dts/renesas directory. No setting means 100000 Hz.

```
&i2c0 {
    status = "okay";
    clock-frequency = <400000>;
    ...
}

&i2c1 {
    status = "okay";
    clock-frequency = <100000>;
    ...
}
```

Figure 5-5   Example of setting I2C transfer speed (R-Car D3)

I2C0, I2C1, I2C2 and I2C3 have LVTTL (low drive only) buffer.
Their IntDelay are set 6ns in arch/arm64/boot/dts/renesas/r8a77995.dtsi.

Please set I2C transfer speed 400000 or 100000 to "clock-frequency" of device tree file (r8a779a0-falcon.dts) in arch/arm64/boot/dts/renesas directory. No setting means 100000 Hz.

```
&i2c0 {
    status = "okay";
    clock-frequency = <400000>;
    ...
}

&i2c1 {
    status = "okay";
    clock-frequency = <100000>;
    ...
}
```

Figure 5-6   Example of setting I2C transfer speed (R-Car V3U)

Please set I2C transfer speed 400000 or 100000 to "clock-frequency" of device tree file (r8a77980-condor.dts) in arch/arm64/boot/dts/renesas directory. No setting means 100000 Hz.

```
&i2c0 {
      status = "okay";
      clock-frequency = <400000>;
      ...
}

&i2c1 {
      status = "okay";
      clock-frequency = <100000>;
      ...
}
```

Figure 5.7 Example of setting I2C transfer speed (R-Car V3H)

I2C0, I2C1, I2C2, I2C3, I2C4 and I2C5 have LVTTL (low drive only) buffer.
Their IntDelay are set 6ns in arch/arm64/boot/dts/renesas/r8a77980.dtsi

5.3.1.2 Multi-master

If there is another master on I2C bus, a clock should always be supplied.
Please set "multi-master" of device tree file (salvator-common.dtsi, r8a77990-ebisu.dts, r8a77990-es10-ebisu.dts, r8a77995-draak.dts, r8a779a0-falcon.dts, r8a77980-condor.dts, …) in arch/arm64/boot/dts/renesas directory.

Device Tree property	Description	Remark
multi-master	Support the multi-master	A clock should always be supplied when multi-master to keep arbitration working.

5.3.2   Kernel Parameters

There are no module parameters.

REVISION HISTORY	Linux Interface Specification Device Driver I2C User's Manual: Software		

Rev.	Date	Description		
		Page	**Summary**	
0.1	Nov. 20, 2015	—	New creation.	
0.2	Mar. 18, 2016	2	Table 1.2 Connected device (R-Car H3) Add Max SCL Clock Frequency.	
		26	4.3 Structure Modify the Linux version.(4.2 -> 4.4)	
		31	Add "5.3.1.1 DT" and "5.3.1.2 Multi-muster" section.	
		31	Figure 5.4 Example of setting I2C transfer speed (R-Car H3) Modify the "clock-frequency" value of I2C2. (400000 -> 100000)	
0.3	Apr, 15, 2016	All	Add R-Car M3 support.	
		3	1.3.2 Related Documents Update related documents.	
		31	5.3 Option Setting Add R-Car M3 Device tree file(r8a7796.dtsi).	
0.4	Aug.5.2016	1	Add "1.2-1 Driver Function" section and Table 1.2.1.	
		3	Table 1.3-2 Related Documents - H3 Document Update. - Add M3 Document. 1.5 Notice Delete "DMA Unsupport".	
		31	5.3.1.1 DT The explanation of F[n] is corrected.	
0.5	Dec. 16, 2016	31	5.3.1.1 DT 5.3.1.2 Multi-master Change R-Car H3 device tree file name(r8a7795.dtsi, r8a7795-salvator-x.dts)	
0.6	Mar. 15, 2017	2, 6	1.2.3 Connected Device 3.2 Module Configuration Fix "H3/M3 Solution Board" to "H3-Sip/M3-Sip System Evaluation Board".	
		3	1.3.2 Related Documents Add User's Manual: Hardware Rev0.53.	
		30	Figure 5.1 Directory configuration (R-Car H3/M3) File i2c.h has been abolished by rebasing the kernel.	
		31	5.3.1.1 DT 5.3.1.2 Multi-master Add R-Car H3 device tree file name(r8a7795.dtsi, r8a7795-salvator-x.dts, r8a7795-salvator-xs.dts)	
0.7	Jun. 14, 2017	3	Update User's Manual: Hardware Rev0.53 to Rev0.54.	
1.00	Aug. 8, 2017	All	Update document format.	

Rev.	Date	Description		
		Page	Summary	
1.01	Oct. 24, 2017	All	Add R-Car M3N support.	
		1	Table 1.2 1 Driver Function Add notice for DMA function.	
		3	1.5 Notice Add Usage note for DMA mode of Receive Operation.	
		31	5.3.1.2 Multi-master Add R-Car M3 device tree file name(r8a7796-salvator-xs.dts)	
1.50	Jan. 29, 2018	3	Table 1.5 Related documents (R-Car H3/M3/M3N) - Delete User's Manual: Hardware Rev.0.51. - Delete User's Manual: Hardware Rev.0.55. - Add User's Manual: Hardware Rev.0.80.	
		32	5.3.1 Module Parameters Change the device tree file name.	
1.51	Mar. 28, 2018	All	Add R-Car E3 support.	
		27-30	Update I2C structure based on Linux kernel 4.14	
1.52	Oct. 22, 2018	3	Table 1.5 Related documents (R-Car H3/M3/M3N/E3) - Delete User's Manual: Hardware Rev.0.80. - Add User's Manual: Hardware Rev.1.00.	
1.53	Oct. 29, 2018	29	Table 4.3 4 struct i2c_driver (R-Car H3/M3/M3N/E3) Fix Member Type.	
2.00	Dec. 25, 2018	3	Table 1.3-2 Related documents (R-Car H3/M3/M3N/E3) - Updated board manual.	
		6	Table 3.1-1 Hardware specification (R-Car H3/M3/M3N/E3) - Updated board name.	
		-	Update AddressList.	
2.01	Mar. 11, 2019	3	Table 1.3 2 Related documents (R-Car H3/M3/M3N/E3) - Updated board manual.	
		33, 34	Add device tree of R-Car E3 Ver.1.0	
		-	Update AddressList.	
2.50	Apr. 24, 2020	All	Add R-Car V3U support.	
		All	Update for based on Linux kernel 5.4	
2.51	Dec. 1, 2020	1	Table 1.2-2 Transfer speed Add fast mode plus for R-Car V3U	
		3	Table 1.2-5 Connected device Update connected device	
		5	Table 1.3 2 Related documents Update reference manual	
		11	Figure 3-2 I2C Driver Module configuration (R-Car V3U) Update module configuration (R-Car V3U)	
		12	Table 4.1-3 I2C device node (R-Car V3U) Update device node information	
		38	Add Figure 5-5 Example of setting I2C transfer speed (R-Car V3U)	
2.52	Jan. 29, 2021	All	Add R-Car V3H support	
2.53	Apr. 21, 2021	All	Add R-Car D3 support	

Rev.	Date	Description		
		Page	**Summary**	
3.00	Dec. 10, 2021	-	Add Kernel v5.10.41 support	
		35	Table 4.3-4 struct i2c_driver - Update to v5.10.41	

CONFIDENTIAL

RENESAS

RENESAS

ルネサス エレクトロニクス株式会社

Linux Interface Specification
Device Driver I2C

RENESAS

Renesas Electronics Corporation