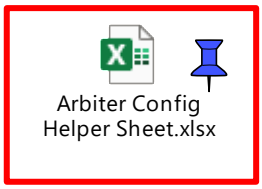
RH850/D1M2

D1M2(H) SDRB DDR2 SDRAM Arbiter Configuration

Introduction

This document provides a sample workflow, on how to create an arbiter configuration for the DDR2-SDRAM Memory Controller (SDRB) based on a use case definition.

Together with this document, you will get a helper-sheet that provides configurable parameters for most known use cases.



The official reference document for the SDRB is the D1x User's Manual: Hardware. (Current version: Rev. 2.20 / January 2018). The content of this document is sample data. Use on your own risk.

For further support, please contact device_support.d1x-eu@lm.renesas.com.

Target Device

All D1x-family devices with support for DDR2-SDRAM memory, which are RH850/D1M2 and RH850/D1M2H.

Contents

1. Introduction: What is arbitration?

In general, arbitration is a procedure for multiple masters to get access to multiple slaves (N:M scheme). Such procedure is needed when masters want to do parallel access, but the slaves are limited to only support a single access at a time. The parallel requests must be serialized and handled one-after-another. There are many different forms and algorithms of arbitration, that can be found on computer networks, CAN buses, memory controllers or even in real live traffic either by Traffic Lights or by “Left yields to right.”

In this case, we will be talking about memory access arbitration, which is not a N:M scheme, but a N:1 scheme.

1.1 Multiple masters share one memory resource

On the usual MCU and SoC devices, there is usually one big “unified” memory that is shared among all functional blocks of the chip. All the functional blocks of the chips can work in parallel without interference. But as it comes to memory access, only one master can be served at a time.

As soon as more than one request hits the memory at the same time, some entity must decide whom to serve first. This entity is called arbiter.

To make sure that all the requesting masters can continue their tasks even while waiting, they implement data queues on their connection to the arbiter. This way, data can be requested in advance and processed data can be written to memory without the masters waiting for any data access to be granted by the arbiter.

Also, the arbiter is aware of these data queues and, depending on the connected master, the target of the arbiter is to prevent build-up of these queues. For a video input for example, the queue should never run full, otherwise you’d lose data to be written to memory.

1.2 Real time and best effort masters

The masters in the system can be roughly categorized into “Real Time” and “Best Effort” masters.

A real time master needs a constant stream of data with specific bandwidth and latency. If the stream is interrupted, this may cause either data corruption or visual effects that can be noticed by the user. For example, if the data stream to the VO is interrupted, the user will see black artifacts on the display.

A best effort master will work as fast as it can given the current limitations of internal speed and external bandwidth. The quality of the result does not rely on speed. In case there is an interruption, it will simply wait or work slower. For example, if the GPU is interrupted, it will simply wait for next data.
Note: In case the GPU cannot finish in-time, this will not cause corruption of display output, as it draws into an invisible back buffer. The application SW in turn may not be able to show this buffer on screen, but this may only cause lagging (delayed display) rather than screen corruption.

1.3 Limited Bandwidth

The memory bandwidth of a system is limited as it is restricted by a physical connection. A 16-bit DDR SDRAM for example can transmit a theoretical peak of 2 Byte * 480 MHz = 960 MB/s. The practically available bandwidth is lower and is usually calculated with a margin of 75%, which results in 720 MB/s.
For example, if the real time master VO needs a fixed bandwidth of 500 MB/s, the remaining masters only have 220 MB/s left. If the GPU would now also need an average bandwidth of 400 MB/s to finish the work, this would not work out. It would either cause screen corruption or lagging as one or both of the masters may not get their full bandwidth.

For the system design this means, that all bandwidth requirements should be estimated during device selection:

- What resolution will my screen have?
- How many video-input and -output layers will be in use?
- Can I use sprites to reduce both memory space and bandwidth?
- What will the GPU do? Refresh every buffer every time?
- Will the GPU just copy images together or will it do more complex operations like rotation and blending?

The more information is available, the better will be the requirements needed for MCU selection.

1.4 Arbitration principle

The default arbitration principle of an unconfigured arbiter is a fair distribution. But even “fair” can be expressed from different point of views. In this system, the arbiter will serve just one request from each individual before serving the next one. This principle is also known as “round-robin arbitration”.

The apparent advantage of round-robin is, that theoretically any of N requesting masters may get 1/Nth of the total bandwidth. But in case a single master needs more than 1/Nth of the total bandwidth, this principle won’t work out.

Coming back to the example of Chapter 1.3, let’s assume the VO needs 500 MB/s of the 720 MB/s that are practically available. This is more than 50% of 720 MB/s. If another master is requesting data at the same time, the round robin arbitration scheme will not give more than 360 MB/s to the VO: The VO will have underruns, which will be visible to the user as screen corruption.

1.5 Arbitration: Distribution of Limited bandwidth

The SDRAM controller has an integrated priority generator. This generator can adjust the seemingly fair scheme of the round robin arbitration. You can use it to give and take priority from certain masters but even make sure that a low-priority master gains priority in case it is waiting too long.

The generator sits within the queues of the arbiter. By manipulating the priorities you can change the order how the different queues are served.

2. Bus structure

The DDR2 memory controller is in the lower half of Figure 1 below. The controller has four ports (connections) each split into read- and write-path. The arbiter sits below the four ports connecting it to the actual DDR2-SDRAM PHY.

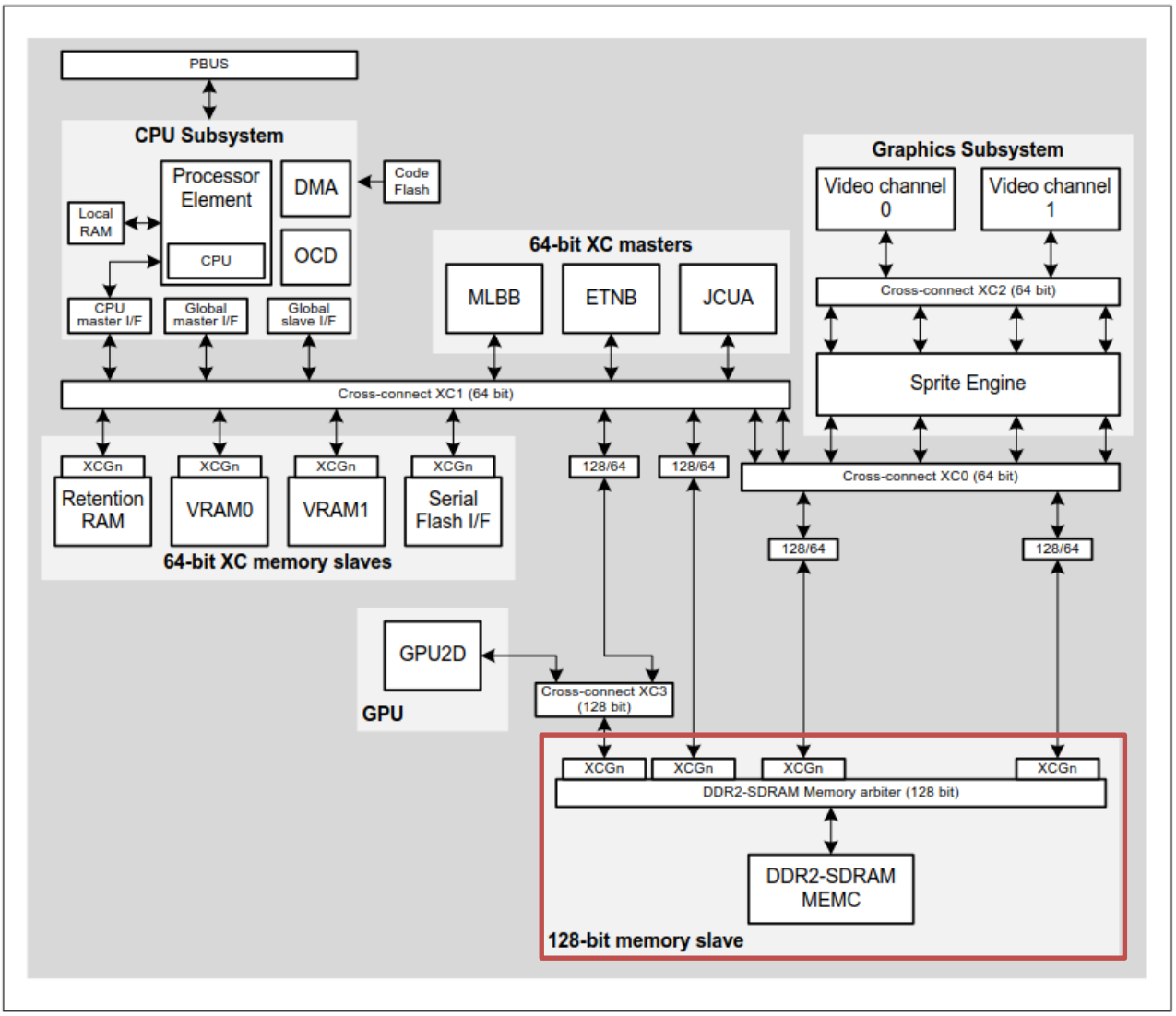


Figure 1: D1M2(H) bus architecture from UM chapter 14.2.3.6 (See UM for latest information)

2.1 Individual priority generators

From perspective of the arbiter, it must handle the following eight paths, as listed in Table 2.1 below.

Table 2.1: Ports of the DDR2-SDRAM arbiter

Register Index	Port	Bus Interface
0	0 R – VO	XC0_0
1	1 R – VO	XC0_1
2	2 R – Other	XC1
3	3 R – GPU	XC3
4	0 W – VI	XC0_0
5	1 W – VI	XC0_1
6	2 W – Other	XC1
7	3 W – GPU	XC3

Each of the paths shown above has its own priority generator, therefore, these are the entities that can be configured in their priority.

The generator of each path can generate four priorities ranging from PRIO0 to PRIO3. If a generator is disabled, it will always output PRIO0. The arbiter will serve higher priorities first. In case a single master currently has PRIO3, it will be served exclusively as long as his requests stay with PRIO3.

The generator itself is coupled to timers, that will measure the time when each individual request was placed into on of the eight queues. The generator knows any time the age of each request that is currently waiting. Instead of directly evaluating this counter, there are configurable thresholds for each port, that will translate the age into the already known priorities PRIO0 to PRIO3.

The thresholds of the timers are the parameters that can be used for arbiter configuration.

The timers are counting with a base clock of 240 MHz. These cycles will be used as time base for later calculations with this document.

2.2 Threshold configuration

There are many different approaches to configure the counter values and thresholds, this document will do it as follows.

2.2.1 Timer values

The timers count downwards and have a range from +8192 to -8192. There are specific mechanisms to evaluate if a timer crossed zero while waiting for a request to be served. This information can be used to calculate penalties or awards for the following requests.

Therefore, the start value of a timer will always be set in the range of [+8129 .. 0].

2.2.2 Priority levels

The initial start values and thresholds of all ports are chosen such that every memory access starts at lowest priority PRIO0. So, in low load case, the arbiter will run in natural round-robin behavior.

In addition, this approach will not use the level PRIO1. The threshold0-1 will be kept next to the threshold1-2, so that is has no effect, but PRIO1 is skipped directly for PRIO2.

The intention is, to have PRIO1 unused in case further tweaking by customer is required. Keeping PRIO1 and PRIO2 close together makes additional tweaking easier as there are fewer side effects if the value was previously unused.

The threshold1-2 for PRIO2 is fixed to 0. As explained above, this means that threshold0-1 is set to +1. To control the interval between PRIO0 and PRIO2, the start value of the counter can be modified. To control the interval between PRIO2 and PRIO3, the threshold2-3 can be modified.

2.2.3 Time interval

Now, the time interval between the thresholds is calculated.

By knowing required bandwidths of each master as well as the bus architecture, it is possible to calculate the number of requests each master needs in a certain timeframe.

For example:

The video output connected to SDRAM read port (index 0) runs with RGBA8888 at 800 x 480 @ 60 fps (thus, the screen has a total height of 480 lines). Therefore, each line must complete within 1 / 60 Hz / 480 lines = 34,7 µs. The bus architecture can deliver 128 Bytes per request. A line of 800 pixels consists of 800px x 32-bit = 3200 Bytes. This makes up a total of 3200 Bytes / 128 Bytes/request = 25 requests.

Finally, this means that one request must be served in 34.7 µs/line / 25 requests/line = 1.39 µs on average.

This will be the basic unit to configure the thresholds for that specific port. If a request cannot be served within the calculated average time, the priority should increase.

Therefore, for this example, we want the interval to be 1.39µs. This must be converted into cycles of the threshold time as mentioned in Chapter 2.1. 1.39 µs * 240 MHz = 334 cycles.

The start value of the counter for SDRAM read port (index 0) must be +334.

If a request waits longer than 334 cycles, the priority will increase. In a multi-master system with several requests waiting, it may even be possible to have several requests waiting on PRIO2. Therefore, as final escalation stage, we will also configure PRIO3. For PRIO3, we will select the same number of cycles. This means, if a request waits two times the time that is allowed to wait on average, it will receive highest priority.

If this delay would happen to every request of a specific master, we would violate the calculated 1.39µs per request. When the delays accumulate, the increasing backlog of unfinished request may finally cause underruns. Therefore, we will activate the “carry-over” feature of the arbiter.

2.2.4 Carry-Over (“Priority Inheritance”)

The carry-over feature of the arbiter makes it possible to remember the delay of the previous transaction on the same port by analyzing the timer value. If the timer is still above zero, the last request was served early, if the timer is below zero, the last request was served late. The function can adjust the following request to either catch-up the delay of the previous request or to decelerate due to the advance of the previous request.

The carry-over does it by adding the counter value to the start value of the next request.

For example:

If the previous request had a start value of +334 cycles, but finished after 300 cycles, it is still +34 cycles ahead of its limit. Therefore, these +34 cycles will be added temporarily to the start value of the next request: 334 +34 = 368 cycles. This request now has 34 additional cycles before transition to PRIO2.

For more details, please also check the UM Chapter 16.5.6.

2.2.5 Custom level PRIO1

As explained above, that level PRIO1 is not used by this Application Note but is free for the user to configure. The intention is to provide a tweaking mechanism that does not have complex side effects.

For Example:

You have the CPU writing data. It uses Port 2 of the SDRAM. You want to make sure, that the CPU runs quickly and is not blocked unnecessarily by masters like the GPU, but you don’t consider the CPU to be a “real time” master. It is clear that the VO is more important and should not be disturbed.

In this case, you could modify threshold0-1 to a higher value, so that CPU request can transition from PRIO0 to PRIO1. The effect would be, that the CPU can be served earlier without directly escalating to PRIO2. It would even be served before requests of the VO, while these are on PRIO0. But as soon as any VO request escalates to PRIO2, the CPU would be overruled again, and the VO is able to catch up.

This way, the custom level PRIO1 can be used to promote selected masters without risking system stability and causing underruns of real-time masters.

3. Creating an arbiter configuration

Attached to this Application Note you will find an Excel Helper sheet. (Please check the cover page for an embedded excel file.) This sheet is prepared to receive various parameters and information about the target application. In general, it will do the same calculations as shown above, but in many places including more parameters.

The sheet will also generate a worst-case traffic estimation. The result cannot be used to estimate average GPU performance, this value is way below the average bandwidth, as it accumulates all peak bandwidths to a single number.

The number is only to assess, if the system is still stable, if all peak loads accumulate at a single point of time: The sum of the individual worst-case traffics should not exceed the total available worst-case bandwidth.

3.1 Filling in the sheet

Any field that is intended to be modified is highlighted in yellow. For the upper part of the sheet there are only numbers to be updated. For the lower part of the sheet starting on row 53: “Priority Generator Configuration”, some cells may contain formulas or references. These are the default recommended values calculated by the information above. Feel free to adjust these values if required.

Please also check out the “USAGE” tab inside the excel sheet as well as the comments next the each cell that is to be filled in.

3.2 General System information

Please input data bus width and frequency of external SDRAM. Possible values are 16 or 32-bit for the data bus. The frequency is usually 240 MHz, as it is bound to the CPU frequency.

Then input general information about connected screens and cameras: Resolution and frame rate. In case you don’t use any VI or VO, leave the values untouched, they will be ignored by later settings.

3.3 Video Output information (Port 0 R and Port 1 R)

Please set a 1 into “active” for each layer that is in use AND is using SDRAM as memory. If the layer would be using VRAM or Serial Flash or is not active at all, set it to 0. No need to fill further cells of a row that is inactive.

Now, input the width and color format of the layers and in case of a sprite layer the maximum number of sprites horizontally next to each other. As the video output works line-by-line, vertical dependencies are not important.

3.4 Video Input information (Port 0 W and Port 1 W)

Please configure active, width and color format for your video inputs that capture into SDRAM. In case you use the VOWE and write the intermediate data to SDRAM, also provide its details here. This will automatically enable the VOWE read path in Port 0 R.

3.5 Other master’s information (Port 2 R and Port 2 W)

In case there are masters other than the Video masters or the GPU to use the SDRAM, they will be configured here. (E.g. CPU, DMA, Ethernet, JPEG-Decoder) Please count all of them that will access the SDRAM and input into “active” column. If any of the masters has a reaction time, chose the one with the smallest allowed latency and input it to column “reaction time”. In the same way, chose the master that must transfer most information from/to SDRAM and input it into “bandwidth”.

3.6 GPU information (Port 3 R and Port 3 W)

This sheet just needs information about those masters that have a “real time” requirement. Those masters considered to be “best effort” will take the remaining resources. This also applies to the GPU, which of course should work as fast as possible, but should not interrupt data streams of “real time” masters.

3.7 Priority generator information

With the information above, the Excel Sheet should already generate a good and stable arbiter configuration. You can find it starting with row 55. By default, all priority generators are activated except those of the GPU. By this measure, requests of the GPU are always bound to PRIO0 and will never interfere with another request, as so as that request escalates to higher priority.

4. Transfer of the arbiter configuration into the system.

In case you already have your own configuration function for the SDRAM, you may take the output rows 55 and below to write the registers on your own.

If you use the Renesas Graphics Library, you also have the possibility to hand this task to the BSP. Please copy the variable starting on row 71 into your source code. During initialization, the BSP automatically calls the function `loc_InitSDRBMemArbiter(void)`. If you put it into this function, the new values will be set during device start-up.

Revision History

Rev.	Date	Description	
		Page	Summary
1.0	2019-01-11	-	First draft of Excel Helper sheet (MNI)
1.1	2019-01-30	-	First preview of Excel Helper sheet (MNI)
1.2	2019-08-08	-	Creation of Application Note (MNI)

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

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, 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.

(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.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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/.