DA7280 Sample Software

For US082 PMOD and RA2E1 MCU

This manual is for the sample demo software that is used with the US082 Haptic Touch PMOD board and the RA2E1 MCU (of the RA family); it describes the project structure and how to set up and configure the US082 Haptic Touch demo for evaluation. The US082 Haptic Touch board is a PMOD adapter board with two capacitive touchpads. The haptic feedback sample demo allows the user to write a haptic waveform to the device so that the device plays back from a set of haptic sequences within the waveform as the user presses the touchpad of a board.

Contents

1. Operating Environment . . . 2

2. Haptic Driver Specifications . . . 3

2.1 Overview of DA7280 Haptic Driver Specifications . . . 3

2.2 Overview of US082 Haptic Touch Board Features . . . 3

2.3 Overview of DA7280 Haptic Waveform Memory . . . 3

3. Demo Software Specifications . . . 5

3.1 Overview of Demo Software . . . 5

3.2 List of Haptic API Functions in Demo Software . . . 6

3.3 Guide to Using the API Functions . . . 7

4. Setup Guide . . . 8

4.1 Default Setup . . . 8

4.2 Additional Settings . . . 8

5. Haptic Board Schematic . . . 10

6. Revision History . . . 10

1. Operating Environment

The operation of this software project has been confirmed with the following environment.

Table 1. Operating Environment

Item	Description
Demonstration board	RTK7EKA2E1S00001BE (EK-RA2E1)
Microcontroller	RA2E1 (R7FA2E1A92DFM: 64 pins)
Operating frequency	48 MHz
Operating voltage	+3.3 V
Integrated development environment	e2 Studio 2021-07
C compiler	GCC 10.3.1.20210824
Flexible Software Package (FSP)	V.3.3.0
RTOS	N/A
Emulator	On board (J-LINK)
Interposer	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
Sensor board	Haptic Touch Driver Pmod™ Board (US082-13-DA7280EVZ)

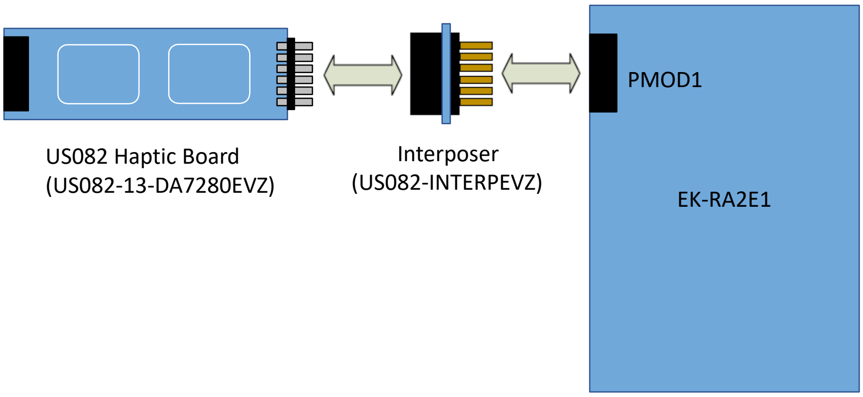


Figure 1. Hardware Connections for the RA Family

2. Haptic Driver Specifications

The Haptic Touch board has two touchpads that can trigger the haptic driver to get a haptic effect, or vibration, with each touch. This section shows the specification of the DA7280 Haptic Driver and the sample demo.

2.1 Overview of DA7280 Haptic Driver Specifications

Table 2 gives an overview of the functionality of the DA7280 Haptic Driver, and the following section describes the operating features of the demo, which capture a subset of the DA7280 functionality. Reference the DA7280 datasheet for more details.

Table 2. Overview of Haptic Driver Specifications

Item	Description
Drive Capability	Linear resonant actuator (LRA)/eccentric rotating mass (ERM)
Driver Input Stream	I^{2}C and PWM
Operation Modes	Direct register override (DRO) Pulse width modulated (PWM) Register triggered waveform memory (RTWM) Edge triggered waveform memory (ETWM)
Input Trigger	3 GPI pins capable of independent haptic sequences and I^{2}C
Average Standby Current	0.8 mA
Supply Voltage	2.8 V to 5.5 V, 3.8 V (typ.)
Operating Temperature	-40 to 125°C

2.2 Overview of US082 Haptic Touch Board Features

The table below gives an overview of the operating features of the sample demo.

Table 3. Overview of Haptic Touch Board Features

Item	Description
Drive Motor	Linear resonant actuator (LRA)
Driver Input Stream	I^{2}C
Operation Modes (Supported)	Register triggered waveform memory (RTWM) Edge triggered waveform memory (ETWM) (default)
Input Trigger	Two touchpads capable of independent haptic sequences
Additional Features	Frequency tracking, sets polarity (rising/falling edge) for two touchpads

2.3 Overview of DA7280 Haptic Waveform Memory

The waveform memory stores multiple haptic sequences. Each sequence is formed by one or more frames, and each frame addresses one or more snippets stored in memory.

In this demo, the waveform memory provides nine haptic sequences for the user to select from to set on each touchpad. These haptic sequences can be found in the rm_da7280.h header file and set in the system.c source file. By changing the sequence ID, the user can access different haptic sequences on each touchpad.

Table 4 provides an overview of the sequences available.

Table 4. Overview of Haptic Sequences

Haptic Sequence Name	Sequence ID	Description
SEQUENCE_SHORT	1	Short, constant buzz (40ms)
SEQUENCE_MEDIUM	2	Medium, constant buzz (140ms)
SEQUENCE_LONG	3	Long, constant buzz (200ms)
SEQUENCE_VERY_LONG	4	Very long, constant buzz (450ms)
SEQUENCE_SHORT_LONG	5	Short buzz, followed by long buzz (600ms)
SEQUENCE_PHONE_RING	6	2 short buzzes, followed by long buzz (800ms)
SEQUENCE_QUIET	7	Low amplitude buzz that reduces in intensity (500ms)
SEQUENCE_RAMP_UP	8	Gradually increasing buzz from silent (500ms)
SEQUENCE_ALARM_CLOCK	9	Sequence of 9 short buzzes (1200ms)

It is possible to create and program a different waveform that provides different haptic sequences constructed by the user. The Waveform Editor in the DA7280 GUI is used to create and combine snippets to form haptic sequences, and the final waveform memory data can be exported. By extracting the waveform memory bytes from this exported script, the waveform memory can be added to the user code as an array (similar to snp_mem on Line 46) in system.c. For details, see the *DA7280 Haptic Driver Datasheet* and *the DA7280 GUI*.

Figure 2 is a representation of each haptic sequence in the demo in terms of amplitude over time. A negative amplitude can be seen as the act of braking on the haptic effect.

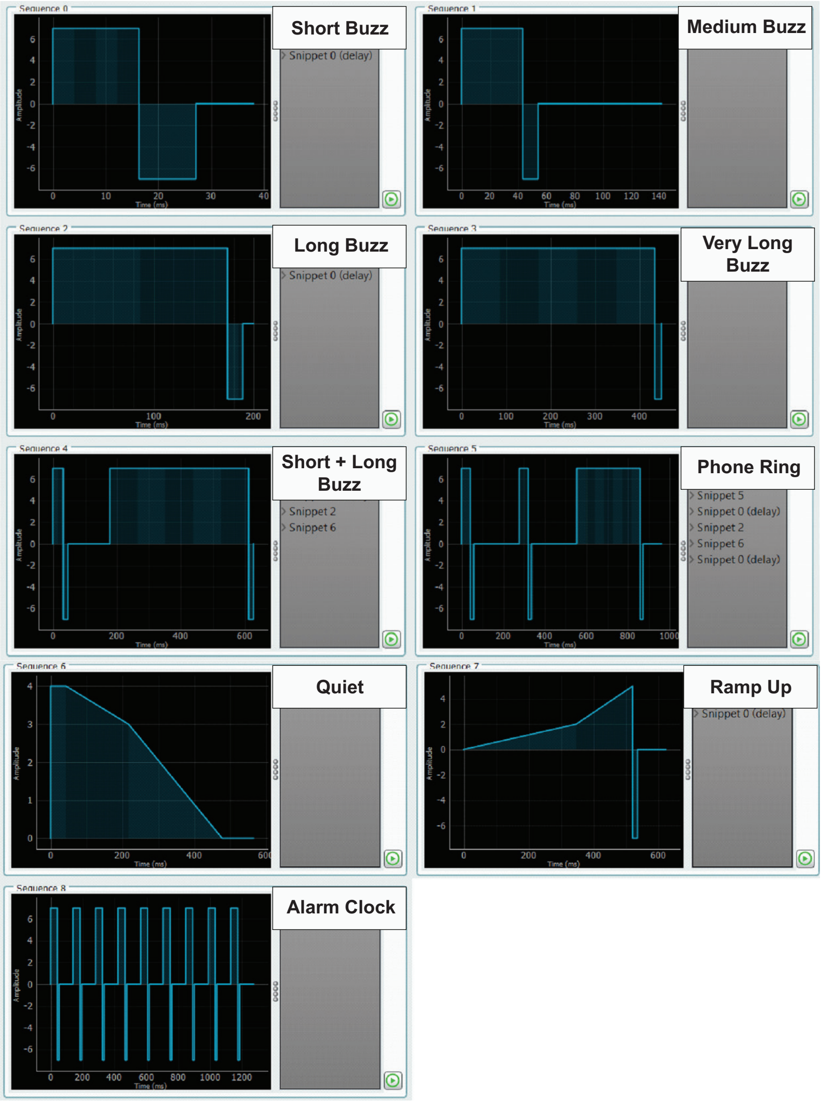


Figure 2. Haptic Sequences in Demo Waveform Memory

3. Demo Software Specifications

3.1 Overview of Demo Software

Figure 3 is a block diagram of the demo software. The user can modify the demo software according to their application, which accesses the DA7280 software library (API) and the I^{2}C middleware and drivers below it. The middleware/drivers are generated by the FSP.

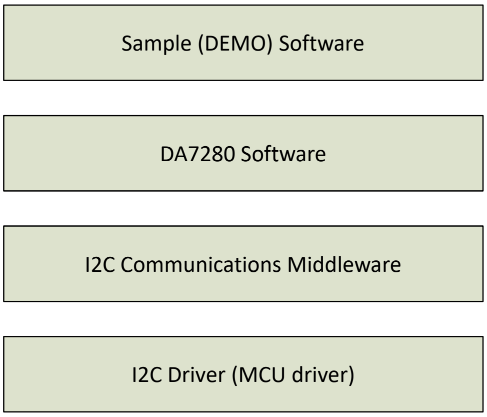


Figure 3. Block Diagram of the Demo Software

3.2 List of Haptic API Functions in Demo Software

Table 4 lists the Haptic Driver API functions found in the rm_da7280.c file. Reference the sys_setup() function in system.c for details on the usage and order of function calls.

Table 5. List of Haptic API Functions

Function	Description
RM_DA7280_Open	Opens and configures the DA7280 module
RM_DA7280_Close	Disables specified DA7280 control block
RM_DA7280_SetFrequencyTracking	Enables or disables DA7280 Frequency Tracking
RM_DA7280_SetMode	Sets the operating mode of the DA7280 (DRO, PWM, RTWM, or ETWM)
RM_DA7280_SetGPIOModePol	Sets DA7280 GPIO mode and polarity
RM_DA7280_WriteWaveform	Writes a new waveform memory pattern to the DA7280
RM_DA7280_PlayFromMemory	Enables the DA7280 to play its waveform from memory

3.3 Guide to Using the API Functions

Figure 4 shows the expected order of calling the API functions. The default configuration for each function is also listed in this section.

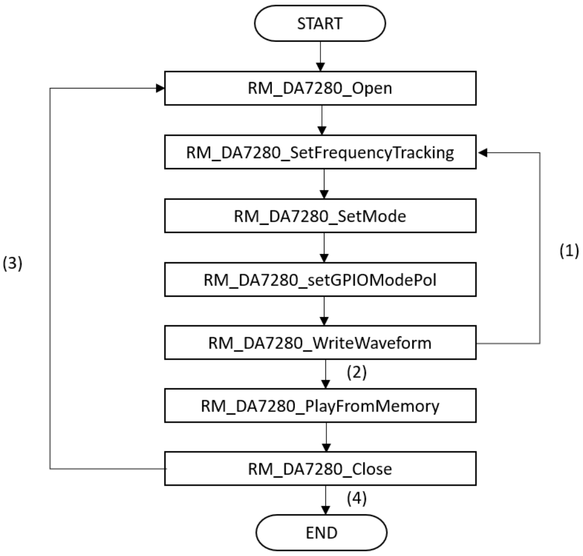


Figure 4. Flowchart of API Functions

1. The functions from RM_DA7280_SetFrequencyTracking to RM_DA7280_WriteWaveform can be repeated at any point in the program execution to reconfigure the device.
2. After RM_DA7280_WriteWaveform, the demo is now functional, and the touchpads triggers a vibration. The RM_DA7280_PlayFromMemory function is optionally available for RTWM mode to play a vibration without touching the touchpads.
3. This module closes when RM_DA7280_Close is called; it is necessary to reopen and reconfigure the device with RM_DA7280_Open.
4. In the sample demo, RM_DA7280_Close is not called

- RM_DA7280_Open: Opens the device, named as g_da7280_sensor0
- RM_DA7280_SetFrequencyTracking: Set to true to enable frequency tracking
- RM_DA7280_SetMode: Set to ETWM mode
- RM_DA7280_SetGPIOModePol:
  Set GPI 0 to SEQUENCE_SHORT, single pin configuration, and rising edge trigger
  Set GPI 1 to SEQUENCE_MEDIUM, single pin configuration, and rising edge trigger
- RM_DA7280_WriteWaveform: Writes the waveform contained in snp_mem array
- RM_DA7280_PlayFromMemory: Not used in this sample (commented out)
- RM_DA7280_Close: Not used in this sample

4. Setup Guide

This section describes the default operation of the sample project for evaluating the Haptic Driver.

4.1 Default Setup

The board can be set up and run using the following steps:

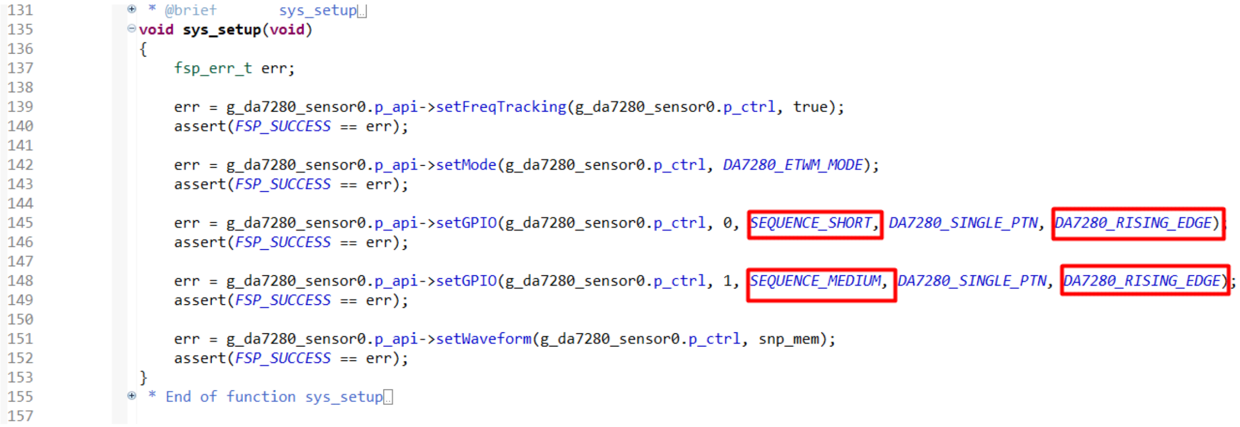
1. Connect the EK-RA2E1 board to the host PC using micro-USB connection to J10 (DEBUG1).
2. Import and build the project in e2 studio in accordance with the *e2 studio User Manual*.
    a. From the main menu, click **File** and select **Import…**
    b. Select **Existing Projects into Workspace** and press **Next**.
    c. Select the **Select archive file** button and browse to the archive file of the project.
    d. Check the box for the project and press **Finish**.
    e. Right-click the project in the Project Explorer and select **Build Project**.
3. After building the project, select **Debug** to flash the code into the board.
4. After the code has been downloaded into the board, the software starts.
5. Press **Resume** twice to run the code. The Haptic Driver board is initialized and is set up.
6. After running the code, touch either of the touchpads on the board and they vibrate.
7. Additionally, the value of the GPIOs (touchpads) is printed in the Renesas Virtual Debug Console (Renesas Views tab > Debug > Renesas Virtual Debug Console).

4.2 Additional Settings

There are three optional changes that can be made to the sample project operation. See Figure 5 and Figure 6 for a visualization of the code.

1. Change the sequence pattern to a different sequence.
    a. Open the system.c file inside the src/System folder.
    b. Navigate to Line 145 or Line 148 of the sys_setup() function and change the SEQUENCE_xx parameter to another sequence (SEQUENCE_SHORT to SEQUENCE_ALARM_CLOCK).
    Note: Different sequences can be set for each GPI (touchpad).
    c. Rebuild the project and Debug as detailed in Default Setup.

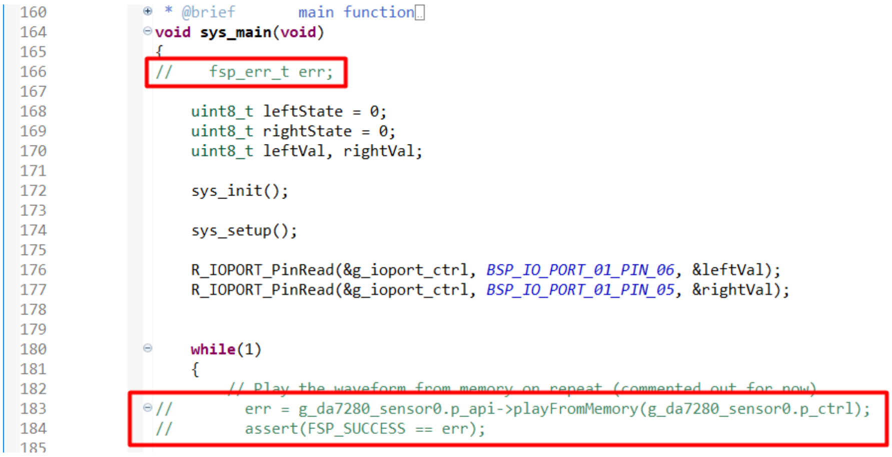
2. Change the edge trigger method.
    a. In the same code as above, change the edge polarity parameter from DA7280_RISING_EDGE (touch the pad) to DA7280_FALLING_EDGE (release the pad) or DA7280_BOTH_EDGE (touch and release).



```
 * @brief       sys_setup
void sys_setup(void)
{
    fsp_err_t err;

    err = g_da7280_sensor0.p_api->setFreqTracking(g_da7280_sensor0.p_ctrl, true);
    assert(FSP_SUCCESS == err);

    err = g_da7280_sensor0.p_api->setMode(g_da7280_sensor0.p_ctrl, DA7280_ETWM_MODE);
    assert(FSP_SUCCESS == err);

    err = g_da7280_sensor0.p_api->setGPIO(g_da7280_sensor0.p_ctrl, 0, SEQUENCE_SHORT, DA7280_SINGLE_PTN, DA7280_RISING_EDGE);
    assert(FSP_SUCCESS == err);

    err = g_da7280_sensor0.p_api->setGPIO(g_da7280_sensor0.p_ctrl, 1, SEQUENCE_MEDIUM, DA7280_SINGLE_PTN, DA7280_RISING_EDGE);
    assert(FSP_SUCCESS == err);

    err = g_da7280_sensor0.p_api->setWaveform(g_da7280_sensor0.p_ctrl, snp_mem);
    assert(FSP_SUCCESS == err);
}
 * End of function sys_setup
```

Figure 5. Changing Sequence ID

3. Play the waveform from Memory without Touching the Touchpad.
    a. Open the system.c file inside the src folder.
    b. Navigate to the sys_main() function and remove the comment marks, //, on Line 166, Line 183, and Line 184.
    c. Rebuild the project and Debug as detailed in Default Setup.



```
160      * @brief       main function
164     void sys_main(void)
165     {
166     //    fsp_err_t err;
167
168         uint8_t leftState = 0;
169         uint8_t rightState = 0;
170         uint8_t leftVal, rightVal;
171
172         sys_init();
173
174         sys_setup();
175
176         R_IOPORT_PinRead(&g_ioport_ctrl, BSP_IO_PORT_01_PIN_06, &leftVal);
177         R_IOPORT_PinRead(&g_ioport_ctrl, BSP_IO_PORT_01_PIN_05, &rightVal);
178
179
180         while(1)
181         {
182             // Play the waveform from memory on repeat (commented out for now)
183     //          err = g_da7280_sensor0.p_api->playFromMemory(g_da7280_sensor0.p_ctrl);
184     //          assert(FSP_SUCCESS == err);
185
```

Figure 6. Playback from Waveform Memory

5. Haptic Board Schematic

The schematic for the US082 Haptic Driver board is shown in Figure 7.

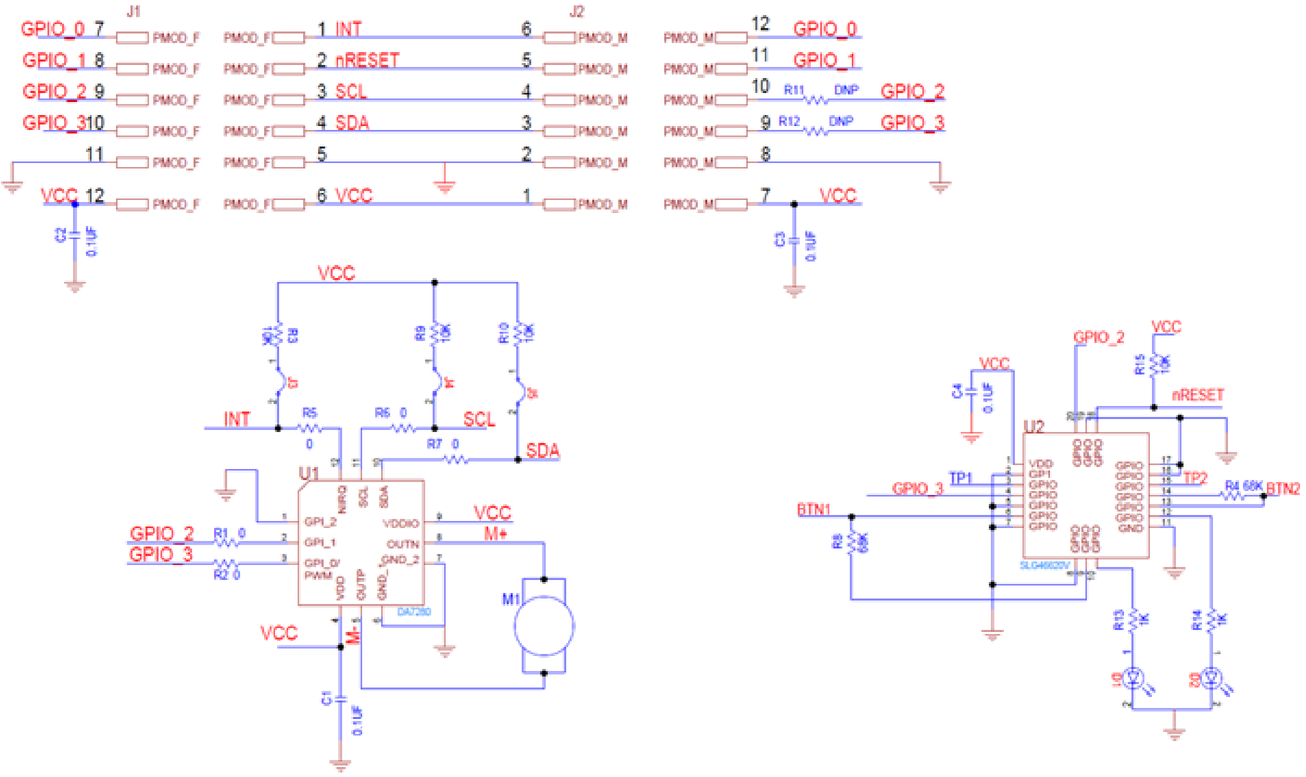


Figure 7. US082 Haptic Driver Board Schematics

6. Revision History

Revision	Date	Description
1.00	Jun 10, 2022	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Rev.1.0 Mar 2020)

Corporate Headquarters

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

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/