AN-1182 Fitness Game

This application note shows how to build an interactive fitness game that combines a mobile phone app with real movement. The game is a stage divided into four sections: up, down, left, and right, which is connected to an Android app. In the Android app, black arrows drop down sequentially and the player must stand on the correct stage section, similar to the game Dance Dance Revolution (DDR). This makes the fitness exercises interactive and fun. We implemented the game with a GreenPAK SLG46620V chip, an HC-06 Bluetooth module, and metal foil. We'll use the GreenPAK chip as a controller for the circuit for many reasons:

- GreenPAK is a tiny chip that contains all the components that our project requires, so the PCB will be small and can be sewn into the mat (stage), which is made from fabric. This means the board will be foldable and portable, and will also keep costs down.
- GreenPAK has efficient power consumption. Since our mat is wireless and portable, the battery life will be longer.

Common DDR games need a computer or a screen to show the game, so they are typically indoor games. However, our game board connects wirelessly with a mobile phone app, so you can take it and play wherever you go.

The project contains two stages:

1. Fitness board, with four touch sensors to detect player steps

2. Mobile game, realized via an Android app developed to build the game platform

Fitness Board

The board that we built is divided into four sections. Every section is covered with metal paper used as a touch sensor. By monitoring the GreenPAK analog pins that are connected to each section, the player's steps are registered (up, down, left, or right) and a signal is sent to the mobile game via Bluetooth. Since we're using the SLG46620V chip, which has six Analog Comparators (ACMPs), we can build up to six touch sensors. The chip converts the signals collected from the touch sensors to a UART frame to send to the Bluetooth module.

GreenPAK design

The GreenPAK chip receives signals from the touch sensors, reformats them into a UART frame, and writes the data to the Bluetooth module.

The HC-06 module that we use for Bluetooth communication uses UART as its communication protocol. A UART (Universal Asynchronous Receiver/Transmitter) is a piece of computer hardware that translates data between parallel and serial forms. It is an integrated circuit used for serial communication that contains a receiver (serial to parallel converter) and transmitter (parallel to serial converter), each clocked separately.

Data Framing

In the UART protocol, the idle, no-data state is logic HIGH. Each packet begins with a START bit (logic LOW), followed by 8 data bits, and a STOP bit (logic HIGH).

Bit #	1	2	3	4	5	6	7	8	9	10	11
Function	Start Bit	D0	D1	D2	D3	D4	D5	D6	D7	D8	Stop Bit

Table 1. Data frame of UART

In our project, three bits are enough to represent the touch sensors states, but we'll use four bits (one for each sensor) so that the game can detect multiple steps at the same time. The other data bits are sent as logic low bits. Table 2 shows the frame that will be sent in our design.

Function	Start Bit	D0	D1	D2	D3	D4	D5	D6	D7	D8	Stop Bit
	0	D0	D1	D2	D3	0	0	0	0	0	1

Table 2. Our project data frame that will be sent

UART packets are sent LSB-first. According to the bit sequence above, we will send the following packets when the player steps on each pad:

Action	8-Bit Represent	Decimal Represent
No touching	00000000	0
Left	10000000	1
Right	01000000	2
Up	00100000	4
Down	00010000	8

Table 3. Bit Representation of control

Touch Sensors

The simplest touch sensor that can be built is a resistive touch sensor. In this sensor, foot skin touching two contacts makes a closed circuit as a low current passes through the foot. By taking this current as input to an analog comparator, a logic high is moved to the next block (see Figure 1).

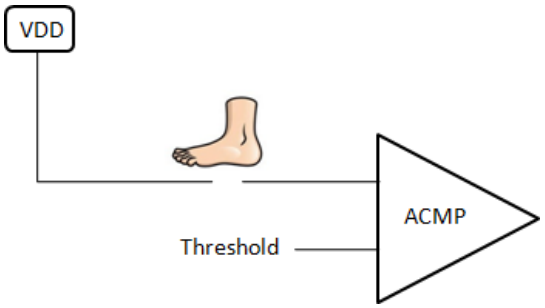

Figure 1. Resistive touch sensor

To build the touch sensors, we'll use metal foil to make rows that will give the circuit contact with the foot.

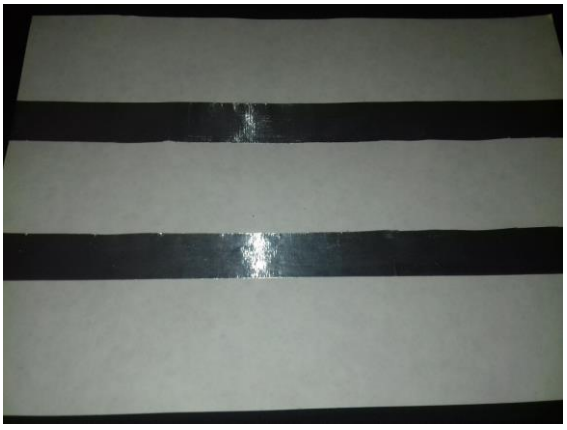

Figure 2. Resistive touch sensor built with metal foil

GreenPAK design

The GreenPAK design consists of 4 stages:

1. Read analog value from touch sensors
2. Enable writing at periodic intervals
3. A 4-Bit Mod-10 counter
4. Write start bit, data bits, and stop bit

1. Read analog value from touch sensors

The four touch sensors are connected to GreenPAK pins 3, 6, 12, and 15, which in turn are connected to analog comparators ACMP0 and ACMP4 in Matrix 0 and ACMP1 and ACMP3 in Matrix 1. The ACMPs are configured to improve the noise tolerance of the sensors and wires. When the user stands on one of the sides, the analog pin connected with this sensor detects a small voltage. This causes the ACMP output to be high, and sends this signal to the order frame that will be sent to the Bluetooth module. Counters are used for an 8 ms delay to prevent short pulses that may happen in analog inputs (see Figure 3).

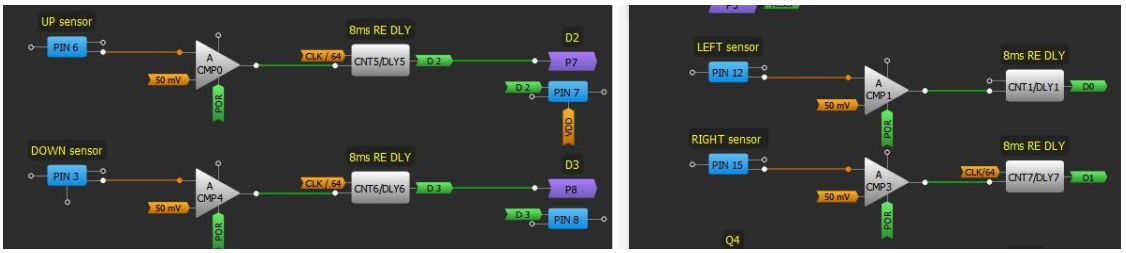

Figure 3. Receive touch sensors signals

2. Enable writing at periodic intervals

In this stage, PIN 20 is connected to an AND gate (2-bit LUT6), which enables the writing from the GreenPAK to the Bluetooth module. Since packets written continuously are buffered in the Bluetooth module, there is a delay in the synchronization between the game and the board. To prevent that, CNT3 and DFF6 are used to write the frame once every 50ms, making the game more stable. DFF8 is used to start writing with the rising CLK edge, so that the full start bit width is given and no bit shifting happens in the receiver. When the stop bit is sent, the two DFFs reset from the 4-Bit Mod-10 counter that is described in the next section.

3. 4-Bit Mod-10 counter

To send a bit sequence, a 4-Bit synchronous counter circuit is required to switch control between the eight data bits, the start bit, and the stop bit. This counter is built using four DFFs. The circuit counts up to 16 states by taking the DFF outputs Q1, Q2, Q3, and Q4, with Q4 being MSB and Q1 being LSB. Since our frame consists of ten bits, the counter will count to 10 and then be reset by the signal coming from LUT3. Therefore, after reaching the tenth state (1001), we need to reset the counter to its initial state (0000) for the next cycle.

The first block of the counter is enabled by P10 (*Enable writing*). The XOR gate (2-L0) then sets its output high, which sets the 'DFF 0' output high.

UART protocol has a 9600 baud rate by default in the Bluetooth module, which means it is able to transfer 9600 bits per second. To find the period of one cycle, we have to divide 1 second by 9600. CLK is set to have a period of 104µs through the OSC and CNT0 block settings to provide the required baud rate.

Every 104µs the clock input checks the DFF 0 block input (coming from XOR gate 2-L0) and sets it HIGH or LOW. The high Q1 output is then passed through the AND gate (3-L0) for the next block. During the next cycle of CLK, the HIGH Q1 at the input of XOR gate (2-L0) toggles its output to zero and is passed to the next block. In this way, the output of the first block toggles between 0 and 1 for every clock cycle of the CLK.

The second block of the counter is similar to the first, except the enable input is coming from the first block (3-L0 output) as shown in Figure 4. This repeats for the following two blocks.

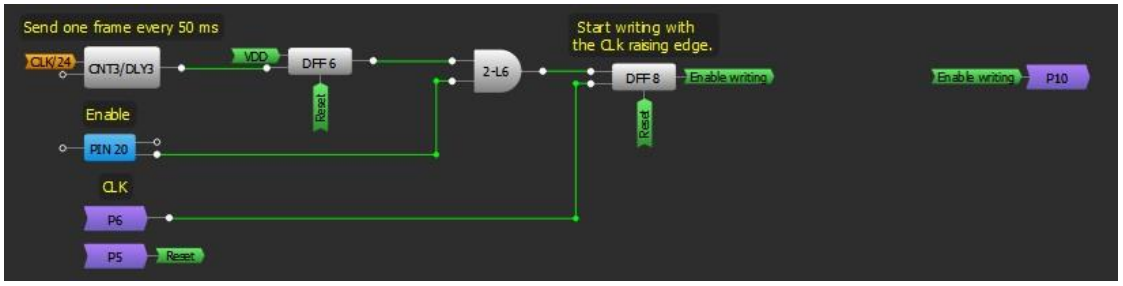

Figure 4. Enable writing and make delay between frames

1. The first block is enabled during every cycle of CLK, hence its output is toggled every cycle.

2. The second, third, and fourth blocks are enabled during every second, third, and fourth cycles of CLK, respectively; hence, their output is toggled every second, third, and fourth cycle, respectively. The four DFFs reset after reaching 10 (1001), meaning that Q2 and Q4 are high (Figure 5). This reset signal is piped to Matrix 1 through P5 to reset DFF6 and DFF8, so the writing stops and waits 50 ms to start a new writing cycle.

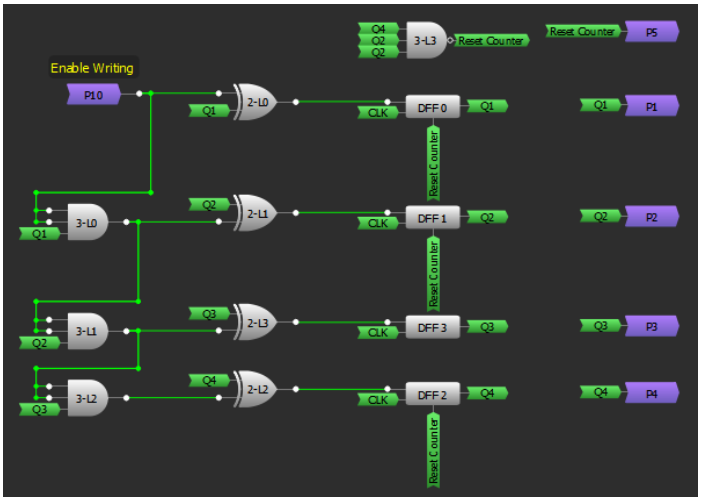

Figure 5. 4-Bit Mod-10 Counter

4. Write start bit, data bits, and stop bit

In this stage, the 10-bit counter's outputs trigger one of the 10 bits (one start bit, eight digital inputs, and one stop bit) to be routed to PIN 19, one by one. In the idle state, the output on PIN 19 should be HIGH. When data starts to be written, a start bit (0) is sent, then eight data bits, and then a stop bit (1) at the end of the packet. Since we use just four data bits, with the other bits LOW, the 4-bit LUT1 is used to output HIGH in the turns D0, D1, D3, and stop bit, and output LOW in the other cases. The four counter outputs Q1, Q2, Q3, and Q4 are the inputs of LUT1 (Table 4).

Q4 Q3 Q2 Q1	Decimal	Data	LUT1 Output
0000	0	Start bit(0)	0
0001	1	D0	1
0010	2	D1	1
0011	3	D2	0
0100	4	D3	1
0101	5	0	0
0110	6	0	0
0111	7	0	0
1000	8	0	0
1001	9	Stop bit (1)	1

Table 4. Counter outputs and 4-bit LUT1 outputs

When *Enable* writing is LOW, a continuous HIGH is present at PIN 19 (see LUT 4 truth table in properties). After *Enable writing* is HIGH and all four outputs of the 4-bit counter are LOW, the LOW signal (start bit) of LUT1 appears on PIN 19. In the second count, the output of LUT1 is HIGH and Q1 is HIGH , which enables the 3-bit LUT8 AND gate, and D0 appears on PIN 19 as the first data bit. In the third count, LUT1's output is HIGH and Q2 is HIGH, which enables the 3-bit LUT9 AND gate, and D1 appears as the second data bit. This repeats in the same way for the D2 and D3 turns. After that, a series of logic zeros is written one by one every 104 µs, which represents data bits D4 to D7. When the counter reaches ten, LUT1's output is high and Q4 is HIGH , which enables the 2-bit LUT5 AND gate and stop bit (1) to appear at the end of the packet before the writing stops (see Figure 6).

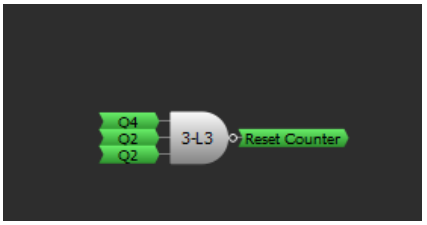

Figure 6. Reset counter blocks

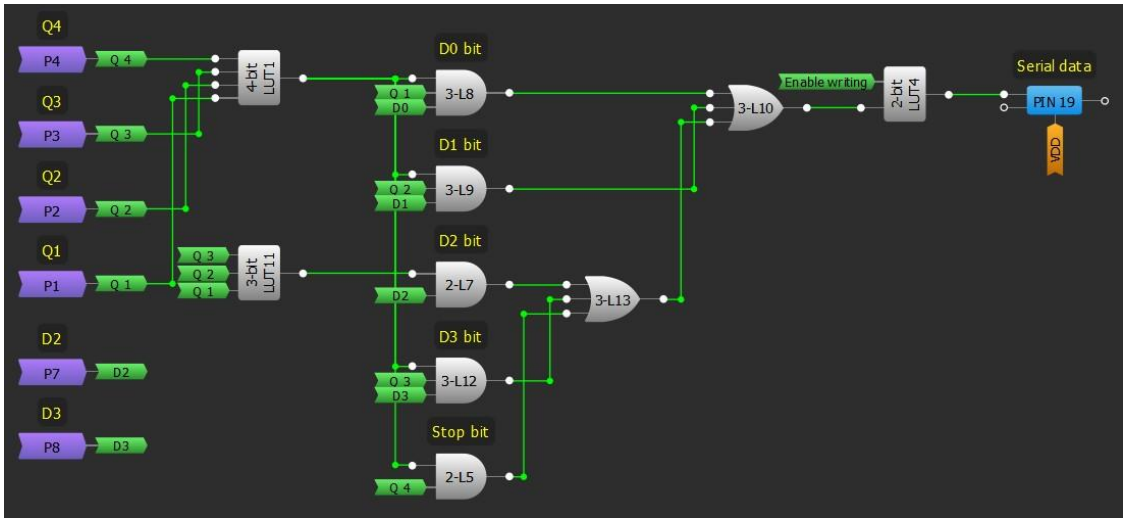

Figure 7. Writing serial data frame

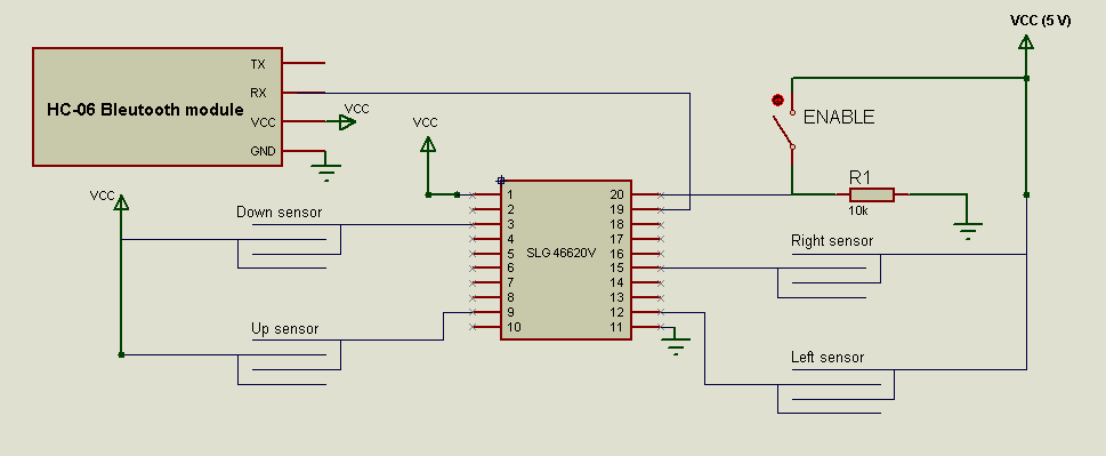

Figure 8. Circuit schematic

Android Game

For this stage, a game for Android devices was built to receive data from the Bluetooth module connected to the GreenPAK chip. The game has a graphical interface with three buttons: Connect, Reset, and Speed. The four arrows in the game drop down continuously, and the score increases by one point with every correct move. If the arrow isn't caught, the "Miss" label increases by one. The Reset button clears the score.

MIT app inventor

Game programming platforms are a bit complex and require prior experience. In this application note, we'll employ a simple method to create the game using the MIT App Inventor. The App Inventor lets you develop applications for Android phones using a web browser with programming blocks.

To create the Android game, a new project is started and the visible components must be removed from the designer screen.

The following components then need to be created: three buttons (Connect, Reset, and Speed), a Canvas component (the background that we will be putting our image sprites on), and four images sprites (one for each arrow). Other components that need to be added are the Bluetooth client, two timers, and the sound player. Figure 9 displays a screen capture of our Android Game's user interface.

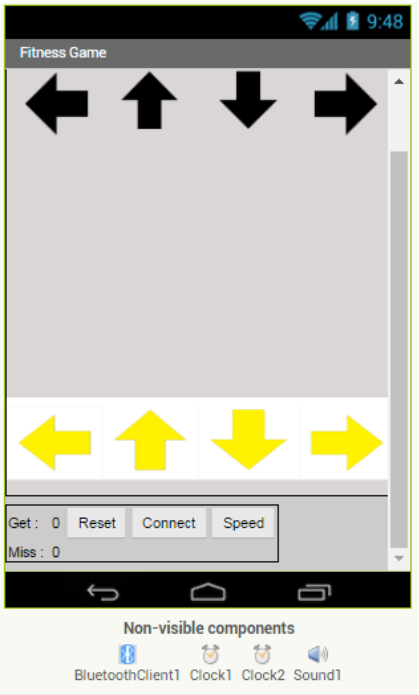

Figure 9. Android App GUI

To start programming, the "Block" button needs to be clicked. By dragging and dropping, we can add components from the bar on the left side. In this step, the Arrows_dropping procedure is created to make the animation for the arrows. 250 pixels are set as the space between arrows, as shown in Figure 10. This procedure is executed with every Clock2 timer pulse, which is set as 50ms timer.

Figure 10. Programming block of left, right, and up arrows animation

Now we need to receive the signal from the GreenPAK to synchronize the game with the player's steps. The game checks new received bytes every 4ms using the Clock1 timer. When a new byte is received, the arrows' positions are tested. If the black arrow is crossed with the yellow and the player stands on the correct sensor, the score increases by one point and a tone is played (Figure 11).

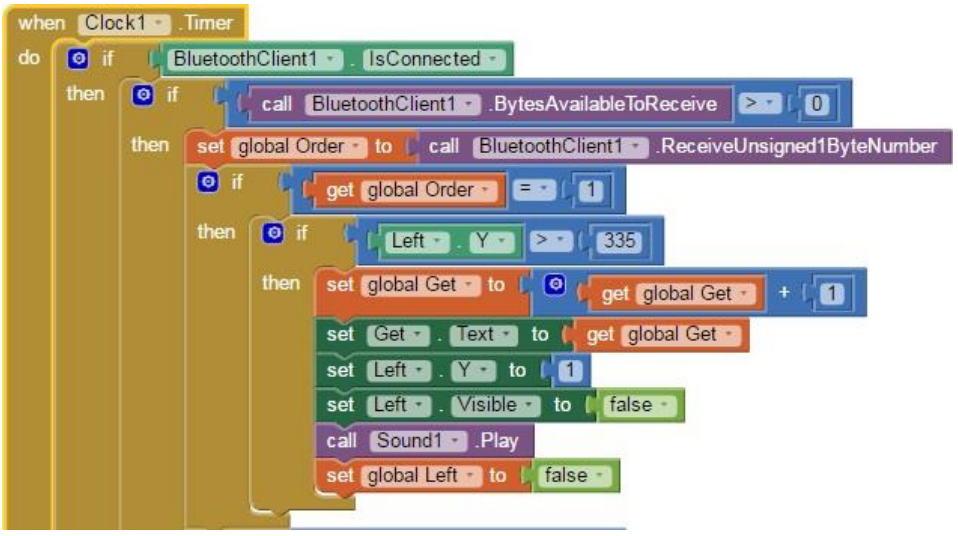

Figure 11. Receive one byte every Clock1 pulse and check left arrow position

If the arrow isn't caught and reaches the edge of the screen, the "Miss" label increases by one and the sprite moves up, becomes invisible, and waits for the other sprites to finish. After that, a new cycle starts. Figure 12 shows the function that's executed when the left sprite reaches the edge.

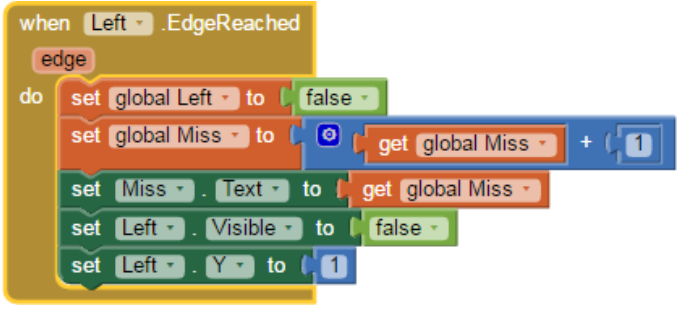

Figure 12. Programming block of left arrow when edge is reached

Other blocks are included that are used to make the buttons function, including clearing the score when the Reset button is clicked, making the connection with the Bluetooth Module when the Connect button is clicked, and choosing the game speed from the speed list. You can find the completed project of this Android game in the files that you have downloaded for this project.

Figure 13. Final board

Conclusion

In this application note, we built a board with resistive touch sensors to detect player steps and send control data from the GreenPAK SLG46620V wirelessly via Bluetooth to control a mobile game. The GreenPAK IC was successfully configured to support the UART protocol in data sequence and synchronization. It allowed us to easily integrate several discrete components from the touch sensor to the UART transmitter into a single small chip. Additionally, GreenPAK has a variety of digital and analog components that facilitate the creation of moderately complex designs.

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 who are 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 to develop 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 from 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.

(Disclaimer Rev.1.01)

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-us/.

Trademarks

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