RZ/A1H Group

GUI Sample Program

Introduction

This document describes the interface between the TES Guiliani Graphical User Interface library and the sample application within the RZ/A1H Software Package.

Target Device / Target Board

Target Device : RZ/A1H

Target Board : Renesas Starter Kit+ for RZ/A1H (R0K77210C000BE)

Contents

1. Overview 4

2. Developing the GUI Sample Program 4
2.1 Running the GUI Sample Program 4
2.2 The Behaviour of the GUI Sample Program 4
2.3 GUI Development Process (Modifying the GUI Sample Program) 5
2.3.1 Invoking the TES Guiliani Stream Editor 5
2.3.2 Opening and Editing the GUI Sample Project 5
2.3.3 Simulating the Edited GUI 5
2.3.4 Exporting the Edited GUI 5
2.3.5 Downloading GUI Resources 5
2.3.6 Editing the User Application 5
2.3.7 Building and Running Application Program 5

3. How to Connect the GUI and Hardware 6
3.1 Overview of the GUI Sample 6
3.2 Implementing the Switch Function 7
3.2.1 GUI Editor 7
3.2.2 User Application 8
3.3 Implementing a Real Time Clock Function 9
3.3.1 GUI Editor 9
3.3.2 User Application 9

4. Tips for Developing a GUI Application 11
4.1 Adding a New Screen Image (Dialog Box) 11
4.1.1 GUI Editor 11
4.2 Adding a New Screen Image Transition 12
4.2.1 GUI Editor 12
4.3 Sharing Values Between Objects 13
4.3.1 GUI Editor 13
4.3.2 User Application 13
4.4 Changing Images and Adding New Images 14
4.4.1 GUI Editor 14
4.5 How to Stop GUI Log Output 15

List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
GSE	Guiliani Stream Editor
GUI	Graphical User Interface
I/O	Input/Output
LED	Light Emitting Diode
OS	Operating System
PC	Personal Computer
RTC	Real Time Clock
SDK	Software Development Kit
URL	Uniform Resource Locator
WYSIWYG	What You See Is What You Get

1. Table 1-1 List of Abbreviations and Acronyms

1. Overview

This document describes the operation of the GUI Sample Program included in the RZ/A1H Software Package. You can develop your own GUI application by modifying this sample program using information from this document.

2. Developing the GUI Sample Program

Running the GUI sample program and the GUI development process are described in this section.

2.1 Running the GUI Sample Program

By following the sequence below, you can run the GUI sample program:

1. Run the application program.
   Run the application. For details, please refer to Quick Start Guide (R01QS0039).

2. Launch the GUI Sample program.
   Type ‘gui[enter]’ into the command console.

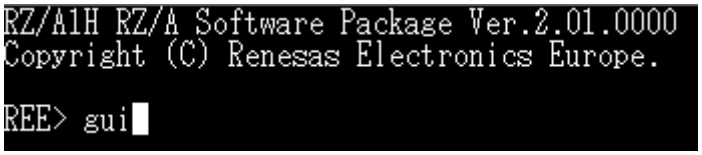

Figure 2-1 Command to Launch the GUI Sample Program

2.2 The Behaviour of the GUI Sample Program

Figure 2-2 shows the image displayed on the Renesas Starter Kit + for RZ/A1H board.

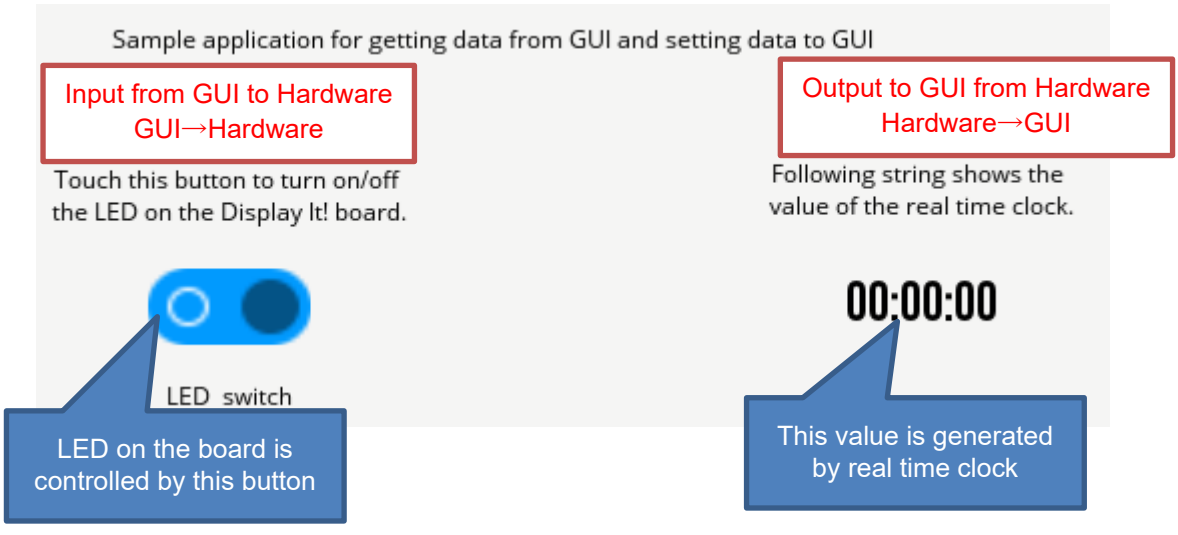

Figure 2-2 The Image Displayed on the Board

2.3 GUI Development Process (Modifying the GUI Sample Program)

The procedure for modifying the GUI Sample Program is described in this section.

2.3.1 Invoking the TES Guiliani Stream Editor

Execute `RZ/A1H_Sample\src\tes\GSE\GSE.exe` on your PC. `GSE.exe` is a WYSIWYG editor.

2.3.2 Opening and Editing the GUI Sample Project

1. Select the menu **File → Open Project**, and select following file:
   `RZ/A1H_Sample\src\tes\GUI_Sample\GuilianiDemo\800x480\GuilianiDemo.gpr`
2. The procedure for editing the GUI and application is described in section 3.

2.3.3 Simulating the Edited GUI

You can simulate the GUI you edited on your PC.

Select menu **File → Run Simulation**, and a simulation dialog box will appear. Press **Run** on the dialog.

2.3.4 Exporting the Edited GUI

To run the edited GUI on the Renesas Starter Kit + for RZ/A1H board, export the GUI Sample Project.

1. Select menu **File → Save Project** to save the project.
2. Select menu **Resource → Export**, and the export dialog will appear.
3. Select following directory and press **OK**.
   — `RZ/A1H_Sample\src\tes\GUI_Sample\Include\GUIConfig`

2.3.5 Downloading GUI Resources

Download the GUI resources to the board. For details, please refer to the Quick Start Guide (R01QS0039).

2.3.6 Editing the User Application

If your system requires interaction between the application and the GUI, such as passing data between the hardware and the GUI, then you will need to modify the sample application. For more details on transferring data between the GUI and sample application, please refer to section 3.

2.3.7 Building and Running Application Program

Build and run the application project. For details, please refer to the Quick Start Guide (R01QS0039).

3. How to Connect the GUI and Hardware

In this section, we describe the interface between the GUI and hardware. In the GUI Sample Program, some Guiliani APIs are used. For details of these and other APIs, please refer to the following URL:

https://www.guiliani.de/mediawiki/downloads/Guiliani_doc_2.2/index.html

The modifications needed for the GUI editor are described in the subsection **GUI Editor**. The modifications needed for the rest of the application are described in the subsection **User Application**.

Please bear in mind that each Guiliani API function has to be called from 'prvGuilianiTask' context.

3.1 Overview of the GUI Sample

A switch object and text field object are used in this GUI sample. Please refer to the figure below for an overview. In the diagram, the black arrows indicate input from GUI, the red arrows indicate output to the GUI.

A DataPool and an Object ID are needed to exchange data between the application and the Guiliani library.

After creating a system image using the GSE editor, the DataPool and Object ID should be exported from the GSE editor and then included in the application. Both the Guiliani library and the application use these DataPool and Object ID variables.

For switch input, the DataPool object provides a callback to the application. This callback is invoked whenever there is any change in state of the switch object.

For time display, the application outputs the current time to the Guiliani library after reading it from the RTC registers. The application calls a Guiliani library set function with Object ID in order to update the object.

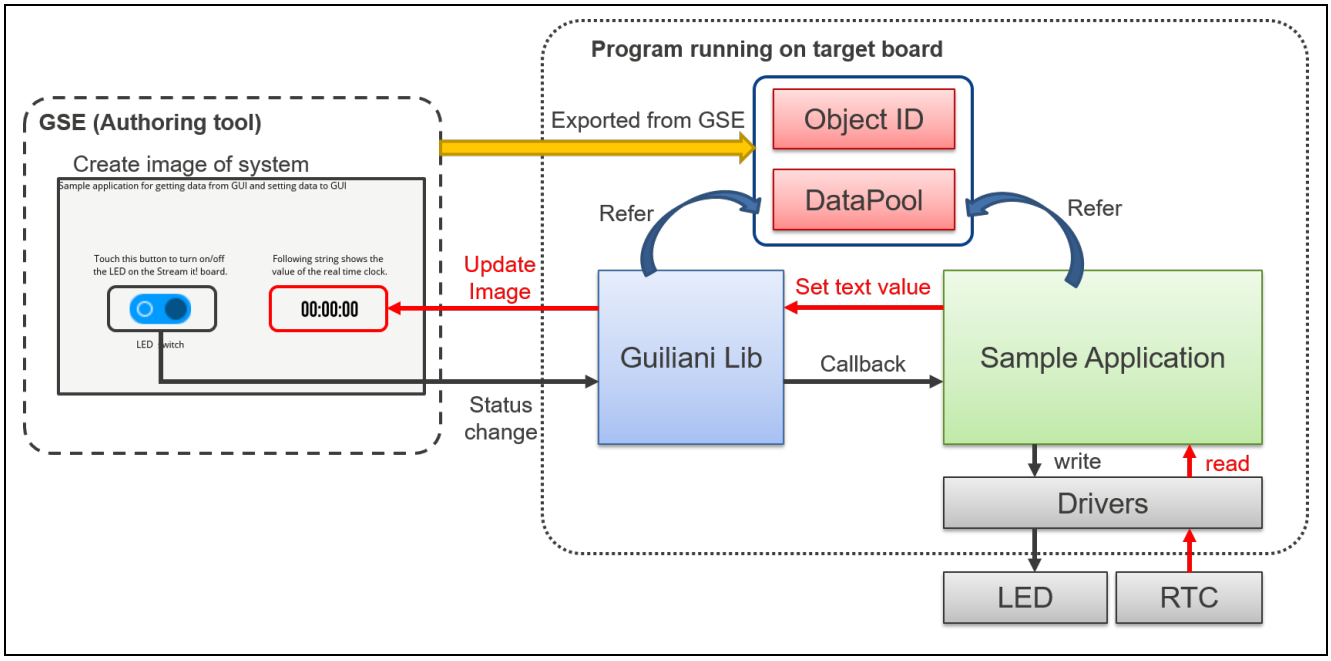

Figure 3-1 Overview of the GUI Sample

3.2 Implementing the Switch Function

By connecting an input object and a DataPool object, then when the input object is modified, the callback function registered by the sample application will be invoked.

The following sequence shows the development of the LED button in the GUI Sample Program.

3.2.1 GUI Editor

1. Add an input object.
   In this example, a CheckBox is added.
2. Give the object a unique object ID name.
   In this example, the CheckBox is named ‘AID_CHECKBOX_1’.

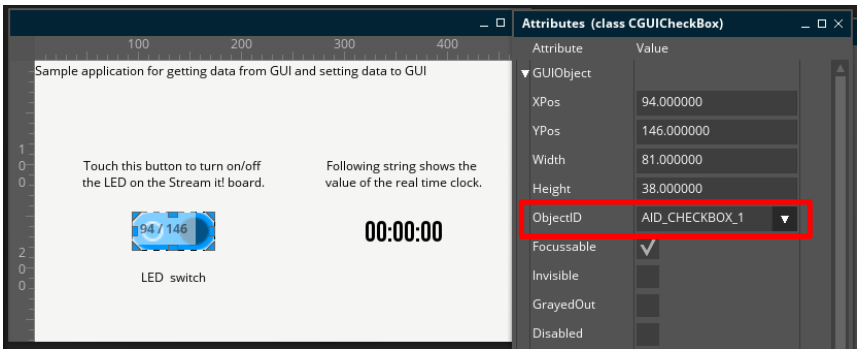

Figure 3-2 Setting ObjectID of CheckBox

3. Add a DataPool object by selecting menu **Resources** → **Manage** → **DataPool**. The **Manage Datapool…** dialog will appear.
   a. Press **Add new Entry**.
   b. Give the DataPool a unique name. In this example, the DataPool is named ‘DATAPOOL_LED’.
   c. Press the ▼ icon and select the object ID you entered in the step 2. In this example, this is ‘AID_CHECKBOX_1’.
   d. Press **Add as Observer**.
   e. Press **Close**.

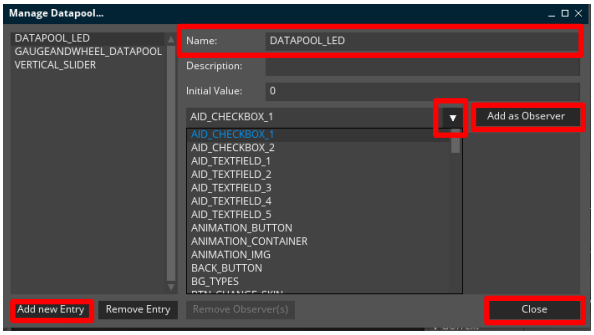

Figure 3-3 **Setting DataPool parameters**

Note that ‘GAUGEANDWHEEL_DATAPOOL’ and ‘VERTICAL_SLIDAR’ were defined as DataPool for the original TES Guiliani demonstration, but the current GUI sample does not use these DataPool definitions.

3.2.2 User Application

1. Register the callback function.
   Please bear in mind that the Guiliani library needs to be running before the callback function can be registered.
   To register the callback function, call the `CGUIDataPool::Register()` function.
   Figure 3-4 shows the sample code registering the callback function with the DataPool object.
   This can be found in the `GUI_Sample()` function in file:
   `RZ/A1H_Sample\src\tes\GUI_Sample\Source\MyGUI_SR.cpp`

```
/* register callback function */
CGUIDataPool::Register(DATAPOOL_LED,&pvLedButtonCallback);
```

Registering callback function

The ID of DataPool.

callback function to register

Figure 3-4 Registering a Callback Function

2. Get the current switch value.
   To get a value from the DataPool, use the `CGUIDataPool::Get()` function.
   Figure 3-5 shows the code from sample program which gets the data from the GUI.
   This can be found in:
   `RZ/A1H_Sample\src\tes\GUI_Sample\Source\MyGUI_SR.cpp`

```
void CMyGUI::pvLedButtonCallback(CDataPoolEntry& data)
{
    CGUIValue value;
    uint16_t led = LED0;
    int_t led_handle = (-1);

    /* get the value of datapool for LED checkbox */
    CGUIDataPool::Get(DATAPOOL_LED,value);
    /* open LED driver */
    led_handle = open( DEVICE_INDENTIFIER "led", O_RDWR);

    /* check the value of datapool for LED checkbox */
    if (value.ToInt() == 0)
    {
        /* LED OFF */
        control(led_handle, CTL_SET_LED_OFF, &led);
    }
    else
    {
        /* LED ON */
        control(led_handle, CTL_SET_LED_ON, &led);
    }
    close(led_handle);
}
```

The value of DataPool and object is copied to the variable “value”.

The ID of DataPool.

Checking the value of DataPool and object. value “0” means “off”, and value “1” means “on”.

Figure 3-5 Sample Code Callback Function

3.3 Implementing a Real Time Clock Function

Next we show how the real time clock in the GUI Sample Program was developed.

Note that the methods for updating the text and numeric values are different.

3.3.1 GUI Editor

1. Add an object for output.
   In this sample, a TextField is added.
2. Give the object a unique ID name.
   In this sample, the TextField is named ‘AID_TEXTFIELD_2’.

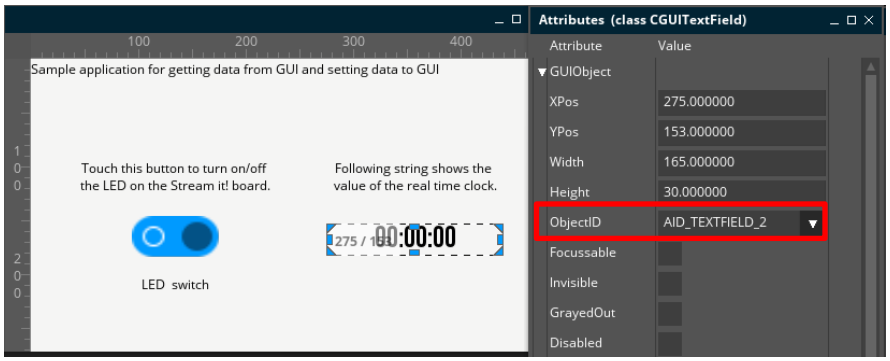

Figure 3-6 Setting CheckBox ObjectID

3.3.2 User Application

1. Add a callback for polling the Real Time Clock.

RZ/A1H_Sample\src\tes\GUI_Sample\Source\MyGUI_SR.cpp

```
// add callback for polling RTC
GETTIMER.AddAnimationCallback(100, this);
```

ms order

add DoAnimate member of this class as callback.

Figure 3-7 Example Callback

2. Update the current time as text by using the `SetLabel()` function in the callback.
   Figure 3-8 shows the sample program code that updates the screen.
   In the package, the text is updated by the `CMyGUI::DoAnimate()` function in:
   `RZ/A1H_Sample\src\tes\GUI_Sample\Source\MyGUI_SR.cpp`

```
DATE last_date;

void CMyGUI::DoAnimate(const eC_Value &vTimes)
{
    /* polling real time clock */
    {
        char date_str[32];
        DATE date;
        /* open real time clock */
        int_t rtc_handle = open(DEVICE_INDENTIFIER "rtc", O_RDWR);
        if (control(rtc_handle, CTL_GET_DATE, &date) == 0)
        {
            if( date.Field.Second != last_date.Field.Second )
            {
                /* create text for time */
                sprintf(date_str,"%.2d:%.2d:%.2d\0",(int_t) date.Field.Hour, (int_t) date.Field.Minute,
(int_t) date.Field.Second);
                /* get the object for AID_TEXTFIELD_2 */
                CGUITextField* pkTextField =
static_cast<CGUITextField*>(GETGUI.GetObjectByID(AID_TEXTFIELD_2));
                /* set the new label for AID_TEXTFIELD_2 */
                pkTextField->SetLabel(date_str);
            }
            last_date = date;
        }
        close(rtc_handle);
    }
}
```

Set current time value. Guiliani will update screen image.

Figure 3-8 Screen Update Sample Code

4. Tips for Developing a GUI Application

This section details some GUI features that are not used in the sample program.

For further details, please refer to the Guiliani SDK from TES solutions:

guiliani.de : https://www.guiliani.de/mediawiki/index.php?title=Downloads:EvalKits
Guiliani 2.2 SDK including GSE and GuilianiDemo for Renesas RZ/A1H (DisplayIt) with eGML (FreeRTOS10 for e² studio 7.2+)

4.1 Adding a New Screen Image (Dialog Box)

You can add a new screen image into your application. In the GCE editor this is achieved through the **Create new dialog** facility. Adding a new dialog box does not require modifying the user sample application.

4.1.1 GUI Editor

1. Press the **+** icon on the lower left hand side of the Dialogs window. The **Create new dialog** window will appear.
2. Enter a unique name for the new dialog.
3. Specify the width and the height of the dialog.
4. Press **OK**.

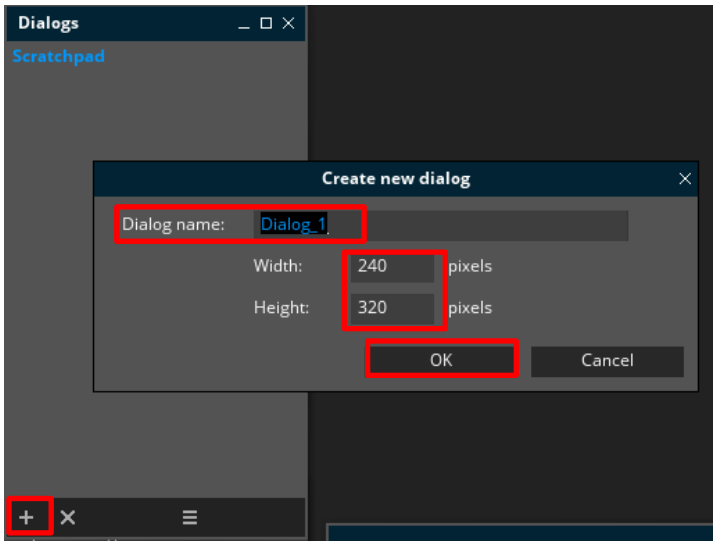

Figure 4-1 Adding a New Dialog Box

4.2 Adding a New Screen Image Transition

After creating a new dialog, you can add the screen image transition function. Again, this function does not require any modification to the sample application.

4.2.1 GUI Editor

1. Add an object such as a switch, button, etc, for launching the dialog box. Here, we're using the switch object from the GUI sample.
2. Change the 'Command Class ID' parameter to CMD_DIALOG_TRANSITION.
3. Press the ▼ icon on the right of the 'DestDialogFileName'. Select the ID of the dialog to transition.
4. Press the ▼ icon on the right of the 'Source Object ID'. Select the ID of the parent dialog.

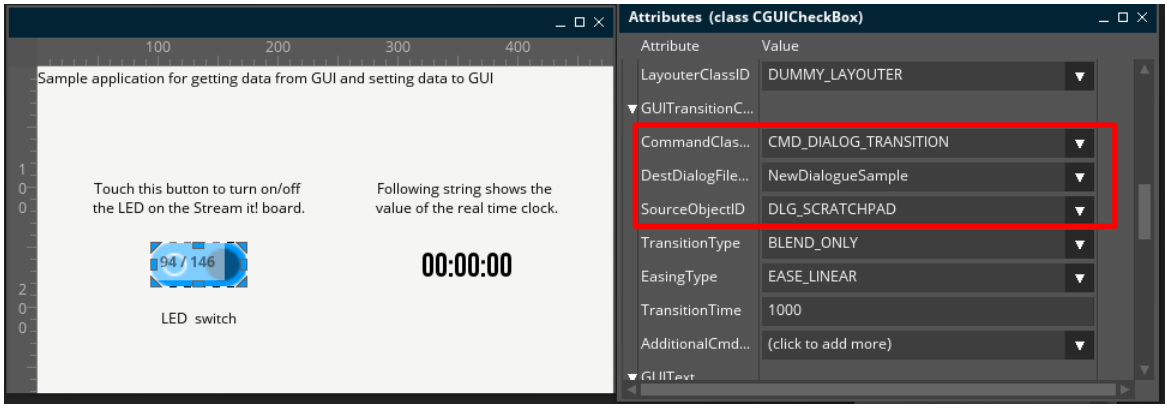

Figure 4-2 Modifying the Dialog Transition Attributes

4.3 Sharing Values Between Objects

You can share values between some objects by using a DataPool object. For example, a system requires that a lamp object on the GUI works with a switch object.

4.3.1 GUI Editor

A DataPool can be added by selecting menu **Resources** → **Manage** → **DataPool**. The **Manage Datapool…** dialog box will appear.

a. Press **Add new Entry**.
b. Give the DataPool a unique name. In this sample, the DataPool is 'DATAPOOL_LED'.
c. Press the ▼ icon and select the object ID you want to share the value with. In this sample, AID_CHECKBOX_1 is chosen.
d. Press **Add as Observer**.
e. Repeat c. and d for any remaining objects that you need to add as an observer.
f. Press **Close**.

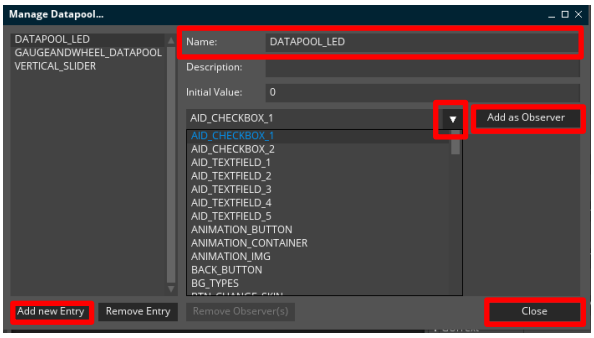

Figure 4-3 Setting DataPool Parameters

4.3.2 User Application

The value shared by the DataPool can be updated by using CGUIDataPool::Set() function:

```
/* set the value of datapool for LED checkbox */
CGUIDataPool::Set(DATAPOOL_LED,((eC_Int)50);
                  The ID of DataPool.
```

Figure 4-4 Setting a DataPool Value

4.4 Changing Images and Adding New Images

4.4.1 GUI Editor

You can change the image of an object by modifying the attribute that is set to ‘IMG_**’. By Clicking ‘IMG_**’, the **Images…** dialog will appear. You can choose the image using this dialog.

You can also add your own image by clicking the **Insert new image** button.

Images bundled in the package are stored in the following directory:

`RZ\A1H_Sample\src\tes\GSE\Resources`

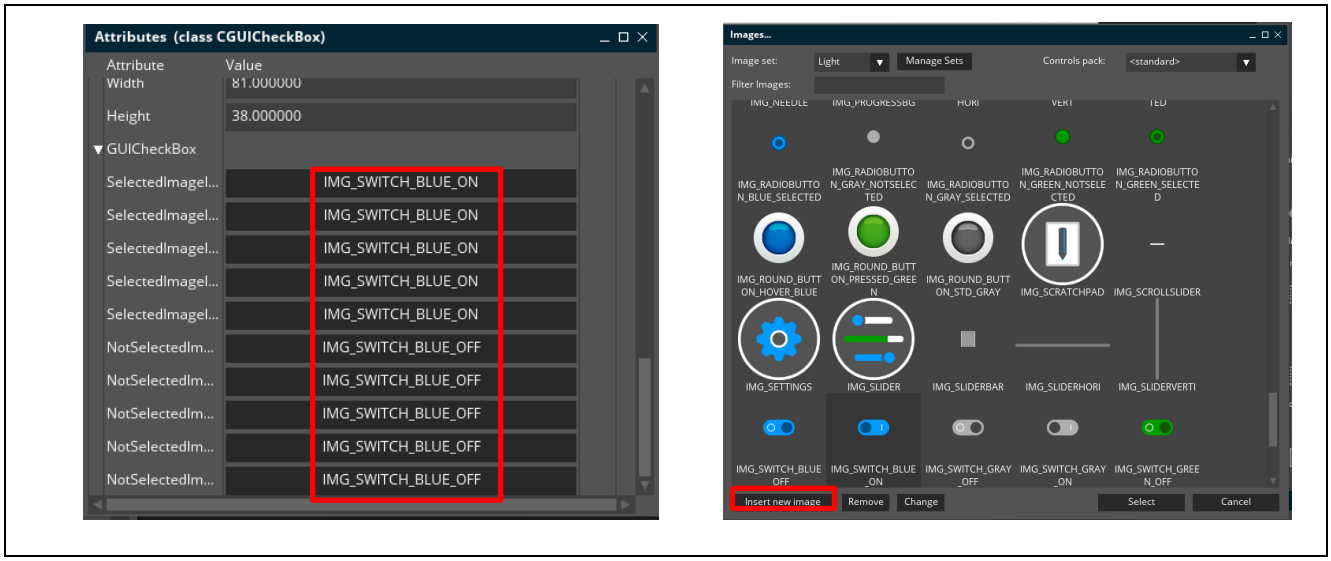

Figure 4-5 The Images Dialog

4.5 How to Stop GUI Log Output

In this package, the debug library of Guiliani has been used. This library outputs messages to the GUI logs.

If using the GCC toolchain, then by switching the debug configuration to ‘Release’, GUI logging output will be stopped.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 29, 2019	-	First Edition issued

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

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/.