Application Note

Renesas Synergy™ Platform

CAN HAL Driver Module Guide

R11AN0065EU0101
Rev.1.01
Aug 22, 2017

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and an efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are included in this document and should be valuable resources for creating more complex designs.

The Controller Area Network (CAN) HAL module is a generic API for a CAN network. The CAN HAL module is implemented on r_can and supports the CAN peripherals available on Renesas Synergy microcontroller hardware. A user-callback function must be defined, which the driver invokes when transmit, receive or error interrupts are received. The callback returns a parameter that indicates the channel, mailbox and event.

Contents

1. CAN HAL Module Features

- Supports both standard (11-bit) and extended (29-bit) messaging format.
- Support for bit-timing configuration as defined in the CAN specification.
- Support for up to 32 transmit or receive mailboxes with standard or extended ID frames.
- Receive mailboxes can be configured to capture either data or remote CAN frames.
- Supports a user-callback function when transmit, receive or error interrupts are received.

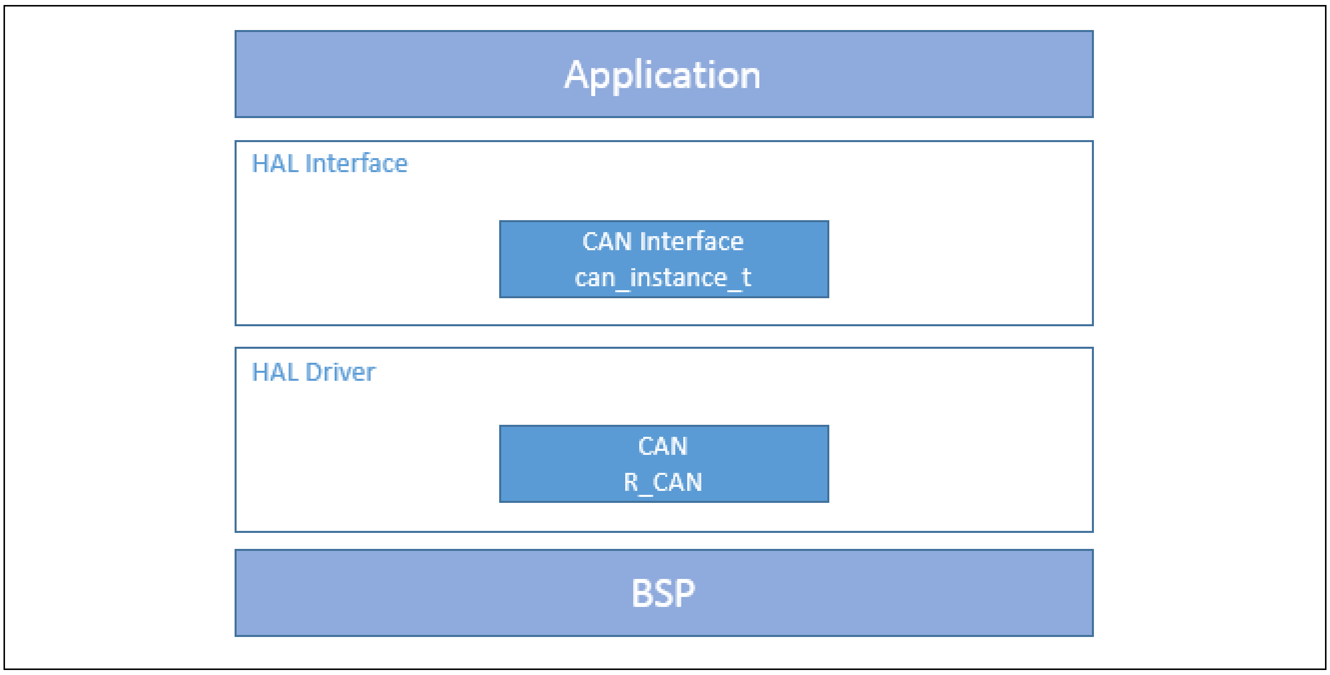


Figure 1 CAN HAL Module Organization, Options, and Stack Implementations

2. CAN HAL Module APIs Overview

The CAN HAL defines APIs for opening, closing, writing, (transmitting) and reading (receiving) CAN data; it also provides some additional functions, such as `control`, `Infoget` and `VersionGet` to assist in processing more complex commands. A complete list of the available APIs, an example API call and a short description of each can be found in the following table. A table of status return values follows the API summary table.

Table 1 CAN HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_can0.p_api->open(g_can0.p_ctrl, g_can0.p_cfg)` The open API configures CAN Channel 0. This function must be called before any other CAN functions. Note: This call is made automatically during system initialization, prior to entering the users thread. Unless the user closes the module, open will not need to be called.
.close	`g_can0.p_api->close(g_can0.p_ctrl)` The close API handles the clean-up of internal driver data.
.read	`g_can0.p_api->read (g_can0.p_ctrl, p_args->mailbox, &receiveFrame)` The `read` API reads received CAN data.
.write	`g_can0.p_api->write (g_can0.p_ctrl, 0, &transmitFrame)` The `write` API write data into the CAN transmit frame buffer and send it out.

Function Name	Example API Call and Description
.control	`g_can0.p_api->control(g_can0.p_ctrl, CAN_COMMAND_MODE_SWITCH, &mode);` with `can_mode_t mode = CAN_MODE_LOOPBACK_INTERNAL;` The control API can change the CAN mode of operation.
.infoGet	`g_can0.p_api->infoGet(g_can0.p_ctrl, p_info)` The `infoGet` API retrieves the CAN mode of operation.
.versionGet	`g_can0.p_api->versionGet(version)` The `versionGet` API retrieves the module version information.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables, review the *SSP User's Manual* API References for the associated module.

Table 2 Status Return Values

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_HW_LOCKED	Lock already owned by another user.
SSP_ERR_CAN_MODE_SWITCH_FAILED	Channel failed to switch modes.
SSP_ERR_CAN_INIT_FAILED	Channel failed to initialize.
SSP_ERR_ASSERTION	Null pointer presented.
SSP_ERR_NOT_OPEN	Port is not open.
SSP_ERR_CAN_DATA_UNAVAILABLE	No data available.
SSP_ERR_CAN_TRANSMIT_MAILBOX	Mailbox is not setup for receive.
SSP_ERR_CAN_TRANSMIT_NOT_READY	Transmit in progress, cannot write data at this time.
SSP_ERR_CAN_RECEIVE_MAILBOX	Mailbox is setup for receive and cannot send.

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.

3. CAN HAL Module Operational Overview

The CAN HAL module controls CAN peripherals on Synergy microcontrollers according to the user configuration. The API provides open, close, read, write, control and information functions. The driver allows for bit-timing as defined in the CAN specification. The driver can be configured for up to 32 transmit or receive mailboxes with standard or extended ID frames. Receive mailboxes can be configured to capture either data or remote CAN frames. The user callback is invoked with channel, mailbox and event information when transmit, receive, or error interrupt occurs.

Using the CAN IDs and Masks

Each CAN Mailbox configured to receive messages has an ID and Mask set. Incoming messages are placed in the lowest mailbox where the following is true:

Incoming ID & Mailbox Mask == Mailbox ID & Mailbox Mask

Example 1: A mailbox with an ID of 0x25 and a mask of 0x1FFFFFFF can receive messages with IDs of 0x25.

Example 2: A mailbox with an ID of 0x25 and a mask of 0x1FFFFFF0 can receive messages with IDs of 0x20 through 0x2F.

Using the CAN HAL Module Test Modes

The CAN Module has three test modes: listen only, external loopback and internal loopback.

- In listen-only mode, valid data frames and remote frames can be received; however, only recessive bits can be sent on the CAN bus. The ACK bit, overload flag, and active error flag cannot be sent. Listen-only mode can be used for baud rate detection
- In external-loopback mode, the protocol module treats its own transmitted messages as those received by the CAN transceiver and stores them into the receive mailbox. To be independent from external stimulation, the protocol module generates the ACK bit. Connect the CTX and CRX pins to the transceiver.
- Internal-loopback mode is similar to external-loopback mode, excepting that the protocol controller performs internal loopback from the internal CTX pin to the internal CRX pin. The input value of the external CRX pin is ignored. The external CTX pin outputs only recessive bits. The CTX and CRX pins are not required to be connected to the CAN bus or any external device.

Changing the CAN HAL Module Operating Modes

The CAN module can be switched between modes using the `control` API. Pass the CAN_COMMAND_MODE_SWITCH and a pointer to a can_mode_t variable to set to the desired mode in the `control` API.

Table 3 CAN modes and can_mode_t values

Mode	can_mode_t value	Reason for use
Normal	CAN_MODE_NORMAL	Normal operation mode
Internal Loopback	CAN_MODE_LOOPBACK_INTERNAL	Internal loopback testing
External Loopback	CAN_MODE_LOOPBACK_EXTERNAL	External loopback testing
Listen Only	CAN_MODE_LISTEN	Baud rate detection
Halt	CAN_MODE_HALT	Mailbox configuration and test mode setting
Sleep	CAN_MODE_SLEEP	Stops the clock supply to the CAN module reducing current consumption
Exit Sleep	CAN_MODE_EXIT_SLEEP	Internal use only
Reset	CAN_MODE_RESET	Communication configuration

Note: See the CAN HAL Module API Summary table example of a `control` API for setting loopback mode.

3.1 CAN HAL Module Important Operational Notes and Limitations

3.1.1 CAN HAL Module Operational Notes

- The user application must start the main-clock oscillator (CANMCLK or XTAL) at run-time using the CGC Interface, if it has not already started (for example, if it is not used as the MCU clock source.)
- For S7, S5, S3 and S1 MCUs: the following clock restriction must be satisfied for the CAN HAL module when the clock source is the main-clock oscillator (CANMCLK): fPCLKB >= fCANCLK (XTAL / Baud Rate Prescaler)
- For S7, S5 and S3 MCUs: the source of the peripheral module clocks must be PLL for the CAN HAL module when the clock source is PCLKB.
- For S3 MCUs: the clock frequency ratio of PCLKA and PCLKB must be 2:1 when using the CAN HAL module. Operation is not guaranteed for other settings.
- For S1 MCUs: the clock frequency ratio of ICLK and PCLKB must be 2:1 when using the CAN HAL module. Operation is not guaranteed for other settings.
- SJW (Synchronization Jump Width) is often given by the bus administrator. Select 1 <= SJW <= 4.
- Time segment and SJW settings must adhere to the following constraints: TS1 > TS2 >= SJW.

3.1.2 CAN HAL Module Limitations

Refer to the latest SSP Release Notes for any additional operational limitations for this module.

4. Including the CAN HAL Module in an Application

This section describes how to include the CAN HAL module in an application using the SSP configurator.

Note: It is assumed you are familiar with creating a project, adding threads, adding a stack to a thread and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

To add the CAN Driver to an application, simply add it to a HAL/ Common Stack or any thread using the stacks selection sequence given in the following table. (The default name for the CAN HAL module is g_can0. This name can be changed in the associated Properties window.)

Table 4 CAN HAL Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_can0 CAN Driver on r_can	Threads	New Stack> Driver> Connectivity> CAN Driver on r_can

When the CAN HAL module on r_can is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information are highlighted in red. Modules with a Gray band are individual modules that stand alone.

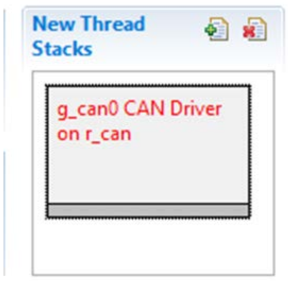


Figure 2 CAN HAL Module Stack

5. Configuring the CAN HAL Module

The CAN HAL module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and not available for changes, and are identified with a lock icon for the 'locked' property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less-error prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the properties tab within the SSP Configurator, and are shown in the following tables for easy reference.

Note: You may want to open your ISDE, create the CAN HAL module and explore the property settings in parallel with looking over the following configuration table settings; this helps to orient you and can be a useful 'hands-on' approach to learning the ins and outs of developing with SSP.

Table 5 Configuration Settings for the CAN HAL Module on r_can

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	Selects if code for parameter checking is to be included in the build.
Error Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	Specify error interrupt priority 0-15 (required).
Receive Mailbox Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	Specify error interrupt priority 0-15 (required).
Transmit Mailbox Interrupt Priority	Disabled, Priority 0-15 (Default: Disabled)	Specify error interrupt priority 0-15 (required).
Name	Default: g_can0	CAN driver module name.
Channel	0, 1	Specify if CAN channel to use 0 or 1 (S7G2 only).
Baud Rate Prescaler	Default 5	Specify baud rate prescaler(0-1023).
Time Segment 1	15 Time Quanta	Specify time segment 1 value. (4-16).
Time Segment 2	8 Time Quanta	Specify time segment 2 value (2-8).

ISDE Property	Value	Description
Synchronization Jump Width	2 Time Quanta	Specify Synchronization Jump Width value (1-4).
Clock Source	PCLKB (S7G2 and S3A7), CANMCLK (Default: CANMCLK)	CAN clock source, CANMCLK or PCLKB (S7G2 and S3A7 only).
Callback	NULL	A user callback function can be registered in can_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time any interrupt occurs.
Overwrite/Overrun mode	Overwrite Mode, Overrun mode (Default: Overwrite mode)	Select whether receive mailbox will be overwritten or overrun if data is not read in time.
Standard or Extended ID Mode	Standard ID Mode, Extended ID Mode (Default: Standard ID Mode)	Select whether the driver will use the CAN standard or extended IDs.
Number of Mailboxes	32 Mailboxes	Select 4, 8, 16 or 32 mailboxes.
Mailbox [0] ID … Mailbox [31] ID	Default n	Select the receive ID for mailbox 0, between 0 and 0x7ff when using standard IDs, between 0 and 0x1FFFFFFF when using extended IDs. Value is not used when the mailbox is set as transmit type.
Mailbox [0] Type … Mailbox [31] Type	N = 0: Transmit Mailbox; N = 1 …31: Receive Mailbox	Select whether the mailbox is used for receive or transmit.
Mailbox [0] Frame Type … Mailbox [31] Frame Type	N = 0: Remote Mailbox; N = 1 …31: Data Mailbox	Select whether the mailbox is used to capture data frames or remote frames (receive only).
Mailbox 0-3 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 0-3.
Mailbox 4 - 7 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 4-7.
Mailbox 8 - 11 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 8 -11.
Mailbox 12 - 15 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 12 -15.
Mailbox 16 - 19 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 16 -19.
Mailbox 20 - 23 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 20 -23.
Mailbox 24 - 27 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 24 -27.
Mailbox 28 - 31 Group Mask	Default: 0x1FFF FFFF	Select the Mask for mailboxes 28 -31.

Note: The example values and defaults are for a project using the Synergy S7G2 MCU. Other MCUs may have different default values and available configuration settings. Most of the property settings for lower-level modules are fairly intuitive and usually can be determined by inspection of the associated Properties window from the SSP configurator.

5.1 CAN HAL Module Clock Configuration

The CAN peripheral uses the CANMCLK (main-clock oscillator) or PCLKB (S7G2 or S3A7 only) as its clock source (fCAN, CAN System Clock.) Using the PCLKB with the default of 60 MHz and the Synergy default, (S7G2 DK) CAN configuration provides a CAN bit rate of 500 Kbit.

To set the PCLKB frequency, use the clock configurator in e^2 studio (Clocks tab in the configurator.)

To change the clock frequency at run-time, use the CGC Interface. Refer to the CGC module guide for details on configuring clocks.

5.2 CAN HAL Module Pin Configuration

The CAN peripheral module uses the pins on the MCU to communicate to external devices connected on the CAN bus. Under Peripherals, select CAN and then CAN0 for channel 0 or CAN1 (S7G2 and S3A7 only) for channel 1. The operation mode for the channel must be enabled and the CRXn and CTXn pins must be selected to match your PC board layout. The pin configurator sets appropriate CAN pin configuration in the `pin_cfg` field for the associated pin. The following table lists the method for selecting the pins within the SSP configuration window and the subsequent table demonstrates an example selection for the CAN pins.

Note: The operation mode selected determines what peripheral signals are available and the MCU pins required.

Table 6 Pin Selection Sequence for CAN Driver on r_can

Resource	ISDE Tab	Pin selection Sequence
CAN	Pins	Select **Peripherals > CAN>CAN0**

Note: The selection sequence assumes CAN0 is the desired hardware target for the driver.

Table 6: Pin Configuration Settings for CAN0

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Enabled	Enable the mode to use CAN0
CRX	None, P202, P402 (Default: P402)	CAN0_CRX0
CTX	None, P203 P401 (Default: P401)	CAN0_CTX0

Note: The example values are for a project using the Synergy S7G2 MCU and the SK-S7G2 Kit. Other Synergy MCUs and other Synergy Kits may have different available pin configuration settings.

5.3 CAN HAL Module Bit Rate Calculation

A time quanta (Tq) is one bit-time of the CAN communication clock, fCANCLK. This is not the CAN bit-time but the internal clock period of the CAN peripheral. The frequency is determined by the baud-rate prescaler value and the CAN source clock, fCAN (CANMCLK or PCLKB). One bit-time is divided into a number of time quanta, Tqtot. One time quantum is equal to the period of the fCANCLK. Each bitrate register is then given a certain number of Tq of the total of Tq that make up one CAN-bit period. The default ISDE bitrate setting (S7G2 DK template) is 500 Kbps for a fCAN at 60 MHz (using PCLKB.)

The formulas to calculate the bitrate register settings are as follows:

```
fCAN = (fPCLKB or fCANMCLK)
```

The baud-rate prescaler scales the CAN peripheral clock down.

```
fCANCLK = fCAN/ Baud Rate Prescaler = 60 MHz (default)/ 5(default) = 12 MHz
```

One time quantum is one clock period of the CAN clock.

```
Tqtot =1/fCANCLK
```

Tqtot is the total number of CAN peripheral clock cycles during one CAN bit time and is by the peripheral built by the sum of the “time segments” and “SS” which is always 1.

```
Tqtot = TSEG1 + TSEG2 + SS (TSEG1 must be > TSEG2) = 15 + 8 + 1 = 24 (default)
```

The bitrate is then:

```
Bitrate = fCANCLK/Tqtot = 12 MHz / 24 = 500Kbps
```

Important notes:

- The user application must start the main-clock oscillator (CANMCLK or XTAL) at run-time using the CGC Interface if it is not already started (for example, if it is not used as the MCU clock source.)
- For S7G2, S3A7 and S124 MCUs: the following clock restriction must be satisfied for the CAN module when the clock source is the main-clock oscillator (CANMCLK): fPCLKB >= fCANCLK (fCAN / baud-rate prescaler)
- For S7G2 and S3A7 MCUs: the source of the peripheral module clocks must be PLL for the CAN HAL module when the clock source is PCLKB.
- For S3A7 MCUs: the clock frequency ratio, PCLKA and PCLKB must be 2:1 when using the CAN HAL module. Operation is not guaranteed for other settings.
- For S124 MCUs: the clock frequency ratio of ICLK and PCLKB must be 2:1 when using the CAN HAL module. Operation is not guaranteed for other settings.
- SJW (Synchronization Jump Width) is often given by the bus administrator. Select 1 <= SJW <= 4.

Table 7 Configurator sample values for different CAN bit-rate

fCAN (either PCLKB or CAN MCLK)	Baud Rate Prescalar	fCANCLK = fCAN/Baude Rate Prescalar	Time Segment 1 (TSEG1)	Time Segment 2 (TSEG2)	Synchronization Jump Width (SS)	Tqtot = TSEG1 + TSEG2 +SS	Bitrate = fCANCLK/Tqtot
240	10	24	15	8	1	24	1 Mbps
60	5	12	15	8	1	24	500 kbps
240	48	5	16	2	2	20	250 kbps
240	96	2.5	16	2	2	20	125 kbps

5.4 Setting the CAN HAL Module Mailbox Group Masks

There are eight mailbox group-masks, one for each group of four mailboxes. These masks allow the mailboxes to be configured to receive more than one ID. If the mask is all ones (0x7ff for standard IDs or 0x1FFFFFFF for extended ID) the mailboxes within the group do not mask any bits of the ID, requiring all bits of the mailbox ID to match the mailbox ID before a message is captured. If any bits of the mask are set to zero, those bits are not necessary to match the same bits of the mailbox ID. For example, if Mailbox ID 1 is set to 0x7ff and Mailbox 0-3 Group Mask is set to 0x7ff, mailbox 1 only captures messages with the ID of 0x7ff. If the mailbox 0-3 group mask is set to 0x7fe, mailbox 1 still captures messages with IDs of 0x7f, but also captures messages with IDs of 0x7fe.

6. Using the CAN HAL Module in an Application

A CAN application requires a minimum of two nodes to demonstrate CAN communication. One node can be a transmitter, while the other can be a receiver (or both can behave as a transmitter and receiver).

The typical steps in using the CAN module in an application are:

1. Initialize the CAN HAL module using the `open` API.
2. (Optional) Enter internal loopback or external loopback test modes using `control` API.
3. (Optional) Information about the module status, including bit rate, can be retrieved using the `infoGet` API.
4. To transmit a message:
   A. Create and configure the CAN frame ensuring correct ID and frame type.
   B. Write the CAN frame to a mailbox configured in transmit mode using the `write` API.
5. To receive a message: read from a mailbox that has received a frame using the `read` API.
6. Close the CAN HAL module using the `close` API (if needed.)

These common steps are illustrated in a typical operational flow diagram in the following figure:

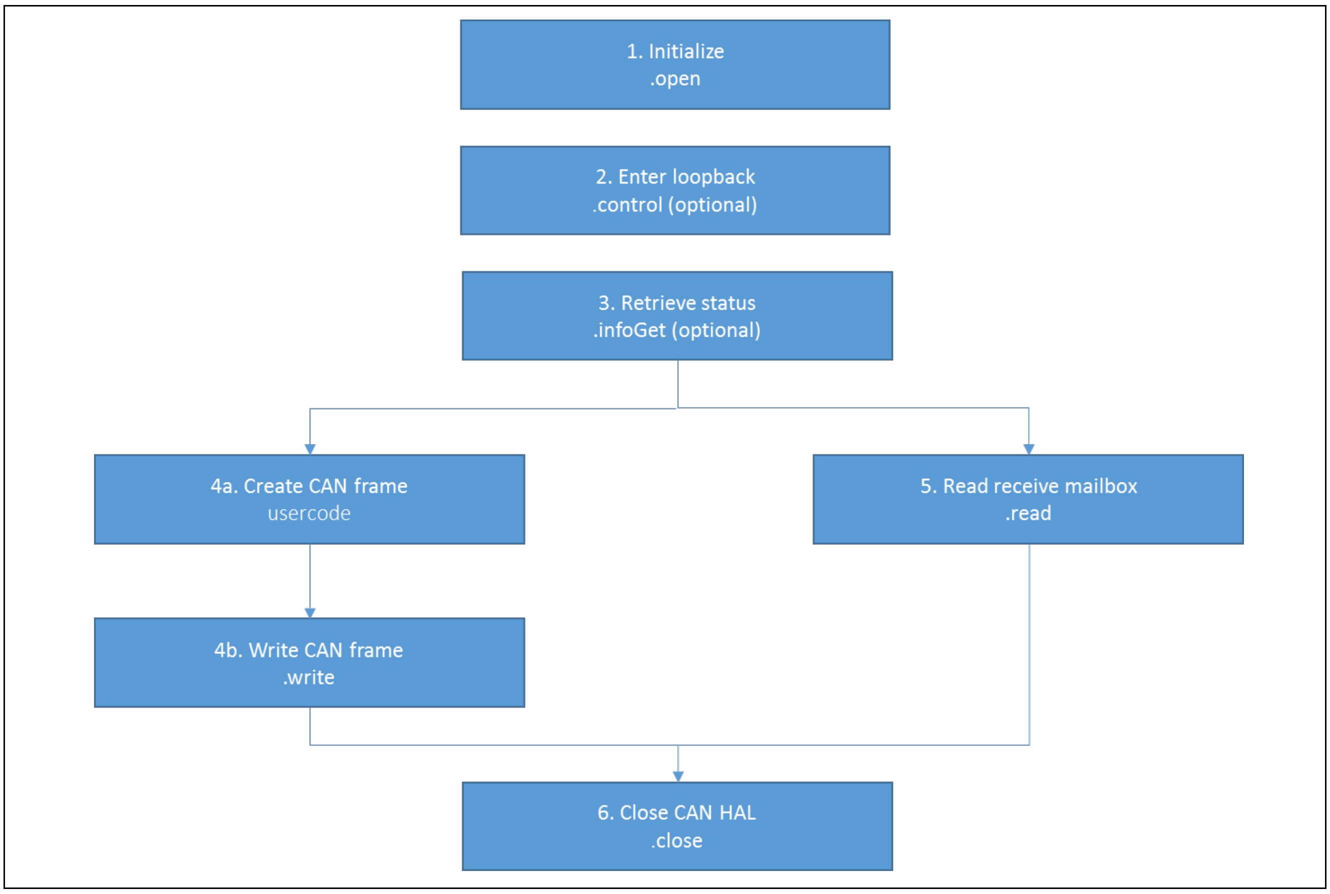


Figure 3 Flow Diagram of a Typical CAN HAL Module Application

7. The CAN HAL Module Application Project

The application project associated with this module guide illustrates the steps in a full design. You may want to import and open the application project within the ISDE and view the configuration settings for the CAN HAL module. You can also read the code (`can_tx_rx.c`) used to demonstrate the CAN APIs in a complete design.

The application project demonstrates the typical use of the CAN APIs. The application project initializes the CAN HAL module and transmits a message when the user presses the switch S4 on the SK-S7G2 board. If the switch is not pressed, the program is in continuous-reception mode. The results are printed on the Debug Console using the common semi-hosting function. The following table identifies the target versions for the associated software and hardware used by the application project:

Table 8 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit (2)

A simple flow diagram of the Application project is shown in the following figure:

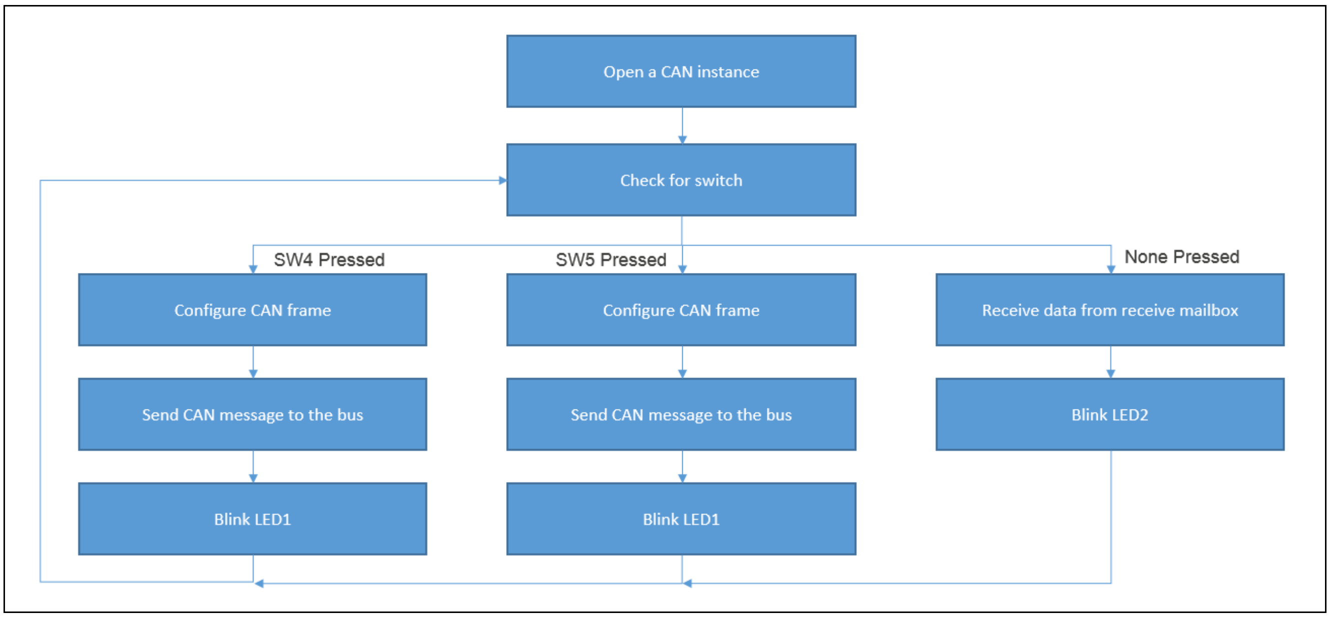


Figure 4 CAN HAL Module Application Project Flow Diagram

The `hal_entry.c` file is in the project once it has been imported into the ISDE. You can open this file within the ISDE and read it along with the following description to help identify key uses of the APIs. The function `can_tx_rx()` is called from HAL entry. The `can_tx_rx.c` file contains the code section to allow semi-hosting to display results using `printf().`

The `can_tx_rx.h` file has the header files which reference the CAN instance structure, definitions of flags for CAN communication and CAN frame declaration. The `can_tx_rx.c` file is the entry function for the main program-control section. The CAN HAL module is initialized using the `open` API. Inside the 'forever' while loop the status of switch S4 and switch S5 is monitored. If a switch is pressed, then the associated CAN transmit function is called. If no switch is pressed, the module waits for the CAN receive flag to be set for CAN reception. Each switch has a different CAN message configuration. The CAN transmit function calls the `write` API to write the frame on the CAN bus. The CAN receive function calls the `read` API to read the frame from the CAN bus. The transmitted and received data are displayed on the console using the `printf()` function.

The last section is the callback function; this function sets the respective flag as per the CAN event executed.

Note: It is assumed that you are familiar with using `printf()`with the Debug Console in the Synergy Software Package. If you are unfamiliar, refer to the "*How do I Use Printf() with the Debug Console in the Synergy Software Package*" Knowledge Base article, available as described in the References section at the end of this document. Alternatively, the user can see results via the watch variables in the debug mode.

A few key properties are configured in this application project to support operations and physical properties required by the target board and MCU. The following table lists the properties with the values set for this specific project. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 9 CAN HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Parameter Checking	Disabled
Error Interrupt Priority	Priority 4
Receive Mailbox Interrupt Priority	Priority 4
Transmit Mailbox Interrupt Priority	Priority 4
Name	g_can0
Channel	0
Callback	can_callback
Mailbox 1 ID	0x7FF
Mailbox 0-3 Group Mask	0x7FF

Message filtering is applied so the CAN node does not receive any message other than the one set in the configurator.

8. Customizing the CAN HAL Module for a Target Application

Some configuration settings are normally changed by the developer from those shown in the application project. For example, you can configure different formats of messages by updating the ID and data length in the code. Note that the maximum data-size that can be handled over one CAN message is 8 bytes. The CAN bitrate can be changed by updating the clock settings using the Clock tab in the ISDE.

9. Running the CAN HAL Module Application Project

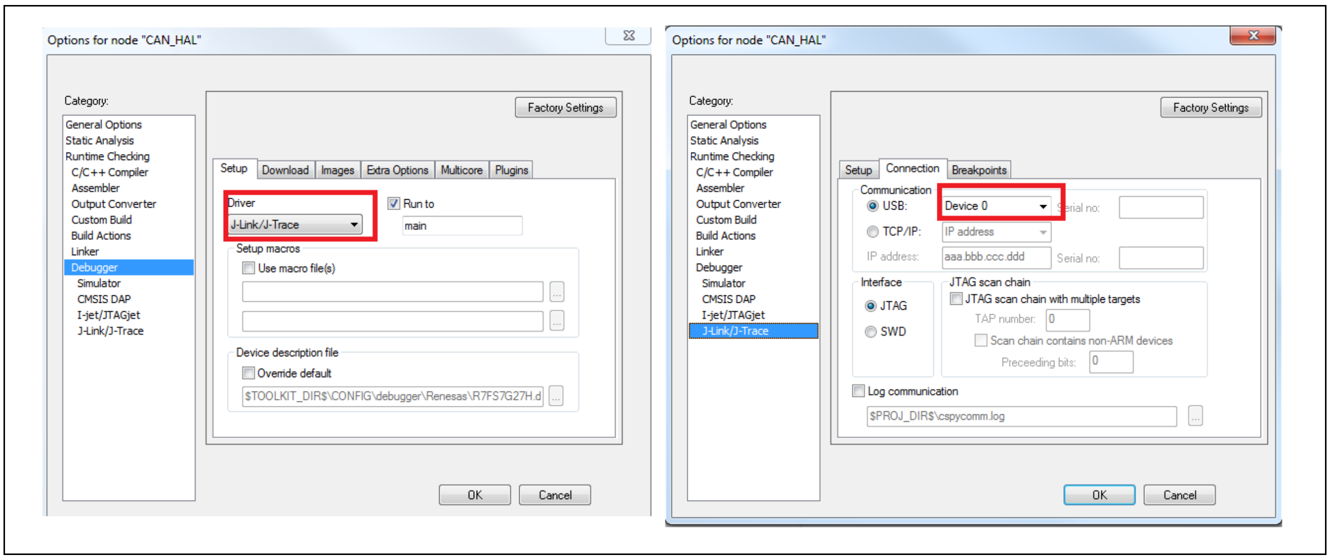
To run the CAN HAL module application project and to see it execute on a target kit, you can simply import it into your ISDE, compile, and run debug. For this demonstration, you need two (2) boards and two instances of e^2 studio with the same project imported in both. CAN pins are present on the J7 port. Connect CANH from one board to another; then connect CANL to the GND. The two instances have different workspaces. Use different port numbers of GDB and ADM in the debug configurations on both instances of the e^2 studio. Use the following table for the pin connections.

Table 10 Pin connectivity for CAN communication between two boards

Board 1	Board 2
CANH	CANH
CANL	CANL
GND	GND

If using the IAR EW for Synergy, you must update the debugger settings by following these steps:

1. Open project properties.
2. Select debugger and change driver to J-Link/ J-Trace.
3. Now select J-Link/ J-Trace and click Connection tab.
4. In USB, select another USB device.



Options for node "CAN_HAL"

Category: General Options, Static Analysis, Runtime Checking, C/C++ Compiler, Assembler, Output Converter, Custom Build, Build Actions, Linker, Debugger, Simulator, CMSIS DAP, I-jet/JTAGjet, J-Link/J-Trace

Debugger – Setup: Driver: J-Link/J-Trace; Run to: main

Setup macros: Use macro file(s)

Device description file: Override default; $TOOLKIT_DIR$\CONFIG\debugger\Renesas\R7FS7G27H.d

Options for node "CAN_HAL"

J-Link/J-Trace – Connection: Communication: USB: Device 0; TCP/IP: IP address; IP address: aaa.bbb.ccc.ddd

Interface: JTAG

JTAG scan chain: JTAG scan chain with multiple targets; TAP number: 0; Scan chain contains non-ARM devices; Preceeding bits: 0

Log communication: $PROJ_DIR$\cspycomm.log

Figure 5 Configuring Debugger in IAR EW for Synergy

Note: These steps are to be followed only when using one PC for both boards. If you are running the application on two different systems, there is no need to change the debugger configurations.

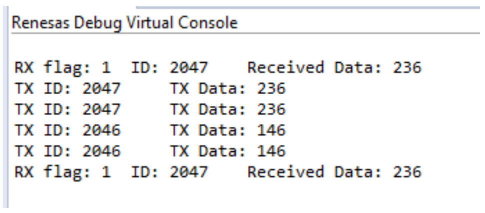
To implement the CAN HAL module application in a new project, use the following steps for defining, configuring, auto-generating files, adding code, compiling and debugging on the target kit. Using these steps is a hands-on approach that can help make the development process with SSP more practical, while just reading over this guide tends to be more theoretical.

Refer to *Importing a Renesas Synergy Project* (r11an0023eu0116-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e^2 studio or IAR EW for Synergy and building/running the application.

Note: The following steps are in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the *SSP User's Manual* for how to accomplish these steps.

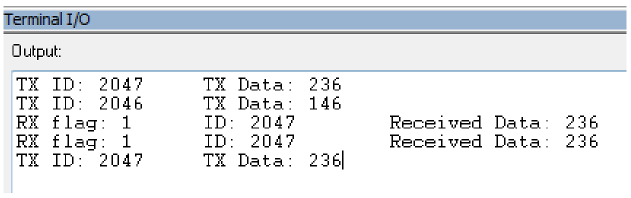
1. Connect both SK-S7G2 board to the host PC via a micro USB cable to J19.
2. Start to debug the application.

The output can be viewed in the ISDE Debug Console (e^2 studio or terminal I/O (IAR EW for Synergy).



```
RX flag: 1  ID: 2047    Received Data: 236
TX ID: 2047     TX Data: 236
TX ID: 2047     TX Data: 236
TX ID: 2046     TX Data: 146
TX ID: 2046     TX Data: 146
RX flag: 1  ID: 2047    Received Data: 236
```

Figure 6 Example Output from CAN Application Project (Board1)



```
TX ID: 2047      TX Data: 236
TX ID: 2046      TX Data: 146
RX flag: 1       ID: 2047        Received Data: 236
RX flag: 1       ID: 2047        Received Data: 236
TX ID: 2047      TX Data: 236
```

Figure 7 Example Output from CAN Application Project (Board 2)

When switch S4 on board 2 was pressed, the ID 2047 message is received on board 2. The message with ID 2046 message was not received on board 1; the configurator was set to filter out that message. Whenever the board receives a message of interest, the RX flag is set.

10. CAN HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy Platform makes these steps much less time consuming and removes the common errors, like conflicting configuration settings or the incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrate additional development time savings by allowing work to begin at a high level, avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

11. CAN HAL Module Next Steps

After you have mastered a simple CAN HAL module project, you may want to merge this with other projects where the data from other modules can be communicated over the CAN bus. For example, if your application has a sensor interface on the ADC and connects with another board using the CAN bus, you may want to merge the CAN module and the ADC module together in one project.

12. CAN HAL Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date r_can module reference materials and resources are available on the Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/r_rtc_Module_Guide_Resources/r_can_HAL_Module_Guide_Resources.

Website and Support

Support:	https://synergygallery.renesas.com/support

Technical Contact Details:

•	America:	https://renesas.zendesk.com/anonymous_requests/new
•	Europe:	https://www.renesas.com/en-eu/support/contact.html
•	Japan:	https://www.renesas.com/ja-jp/support/contact.html

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	May 15, 2017		Initial Release
1.01	Aug 22, 2017		Application update

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0