CAN Communication Time Measurement Solution

(E2 Emulator, CS+)

R20AN0458EJ0100
Rev.1.00
Jul. 16, 2017

Introduction

This application note introduces how to measure CAN bus reception processing times by combining the E2 emulator and the integrated development environment (CS+ for CC), and notes on usage.

This solution is used to improve the problems related to the speed performance of the CAN bus communication applications. You can easily measure the time from detecting a specific CAN frame on the bus up to executing a desired code in a program. The timing of a CAN frame and desired program processing can be shown on the same time base and the time taken for debugging to identify the cause can be shortened.

Target Device

RH850 Family

Contents

Configuration of Manuals

The documents related to the CAN communication time measurement solution, which is an extended function of the E2, consist of the following.

- E2 Emulator User’s Manual
- E1/E20 Emulator, E2 Emulator Additional Document for User’s Manual
- User’s manual and help for the emulator debugger
- Application Note for the CAN Communication Time Measurement Solution (E2 Emulator, CS+) (this document)

(1) E2 Emulator User’s Manual (document No.: R20UT3538EJxxxx)
The E2 emulator user’s manual has the following contents:
- Components of the E2 emulator
- Hardware specifications of the E2 emulator
- Connection to the E2 emulator and the host machine and user system

(2) E1/E20 Emulator, E2 Emulator Additional Document for User's Manual
The E1/E20 Emulator, E2 Emulator Additional Document for User's Manual describes the functions of the debugger and also provides information and notes which depend on each microcontroller.

(3) User’s manual and help for the emulator debugger
The user’s manual and help for the emulator debugger describe the functions of the E1/E20/E2 emulator debugger and the operating instructions.
Refer to the following.
- CS+ Integrated Development Environment User's Manual: RH850 Debug Tool

The CAN communication time measurement solution can also be referred to from online help for the CS+.
- Online help for the CS+

(4) Application Note for the CAN Communication Time Measurement Solution (E2 Emulator, CS+) (this document)
The Application Note for the CAN Communication Time Measurement Solution (E2 Emulator, CS+) covers the setup method, usages, and notes on measuring CAN bus communication processing times.

Terminology

Some specific words used in this user's manual are defined below.

Integrated development environment

This tool provides powerful support for the development of embedded applications for Renesas microcomputers. It has an emulator debugger function allowing the emulator to be controlled from the host machine via an interface. Furthermore, it permits a range of operations from editing a project to building and debugging it to be performed within the same application. In addition, it supports version management.

Emulator debugger

This means a software tool that is started up from the integrated development environment, and controls the emulator and enables debugging.

Host machine

This means a personal computer used to control the emulator.

Target device

This means the device to be debugged.

User system

This means a user's application system in which the device to be debugged is used.

User program

This means the application program to be debugged.

User system interface

This means an interface that connects the target device to the E2 emulator.

Extended function of the E2

This means an extended function which is provided with the E2 emulator.

E2 expansion interface

This means the interface required for extended functions of the E2 emulator.

E2 storage

This means the memory which is mounted on the E2 emulator and used for storing data regarding extended functions of the E2 emulator.

1. Overview

The CAN communication time measurement solution, which is provided by the E2 emulator and integrated development environment (CS+ for CC), is used to easily measure the time from data reception to a desired software processing in a system using CAN bus communication, such as that shown in Figure 1-1. Since it is possible to make a setting to halt program execution when the time from data reception to processing end exceeds the design value, viewing trace data or the history of CAN bus communication after program stop will contribute in determining the cause at an early stage.

The characteristics of the CAN communication time measurement solution are given on the following pages.

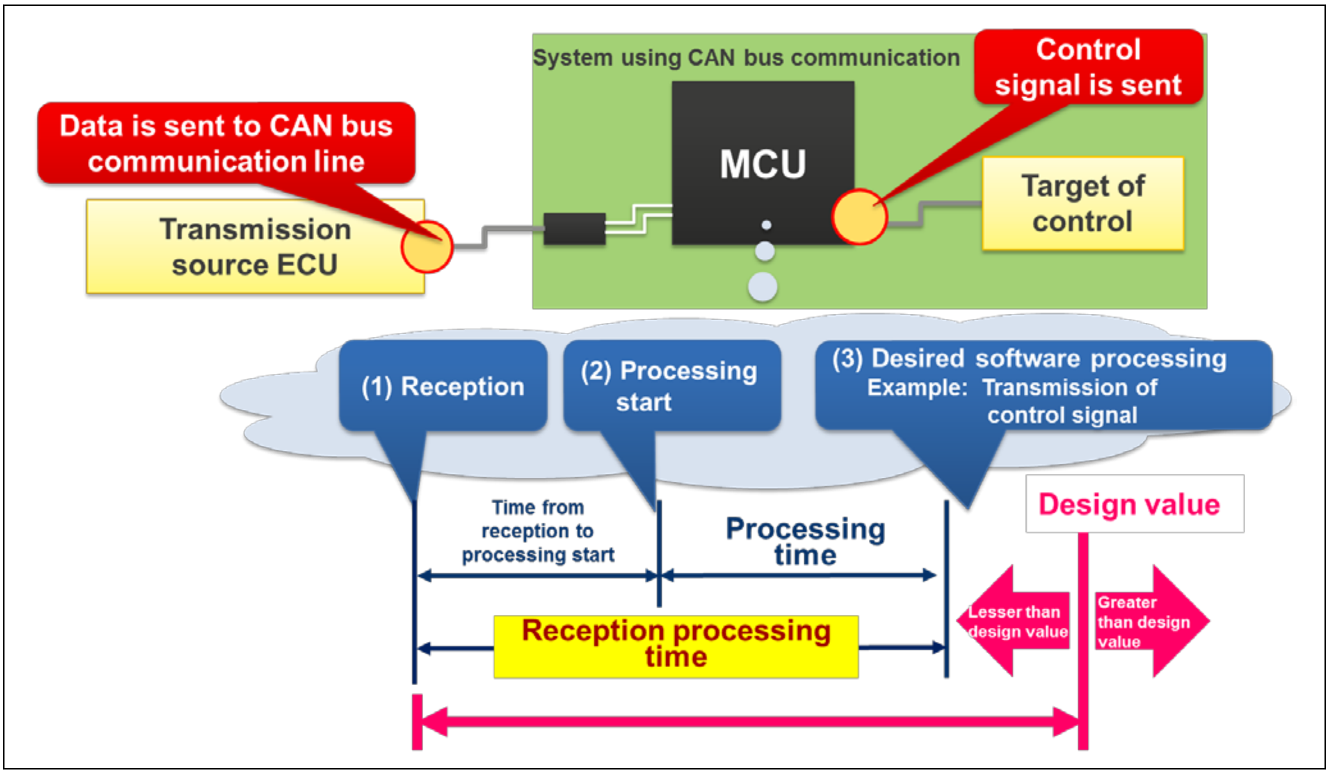


Figure 1-1 Example of a System Using CAN Bus Communication

1.1 Measuring the Time from Receiving a Specific Frame of CAN Bus Communication to Performing a Specific Processing

The first characteristic is automation of measurement as shown in Figure 1-3 which will reduce the working time. Up to now, measurement was repeatedly done by hand as shown in Figure 1-2. The working time can also be reduced by automating the embedding of debugging code which has been performed manually at the measurement points of the program. Furthermore, the timing of running the program can be measured by embedding debugging code even if the user system has no port pins available for debugging.

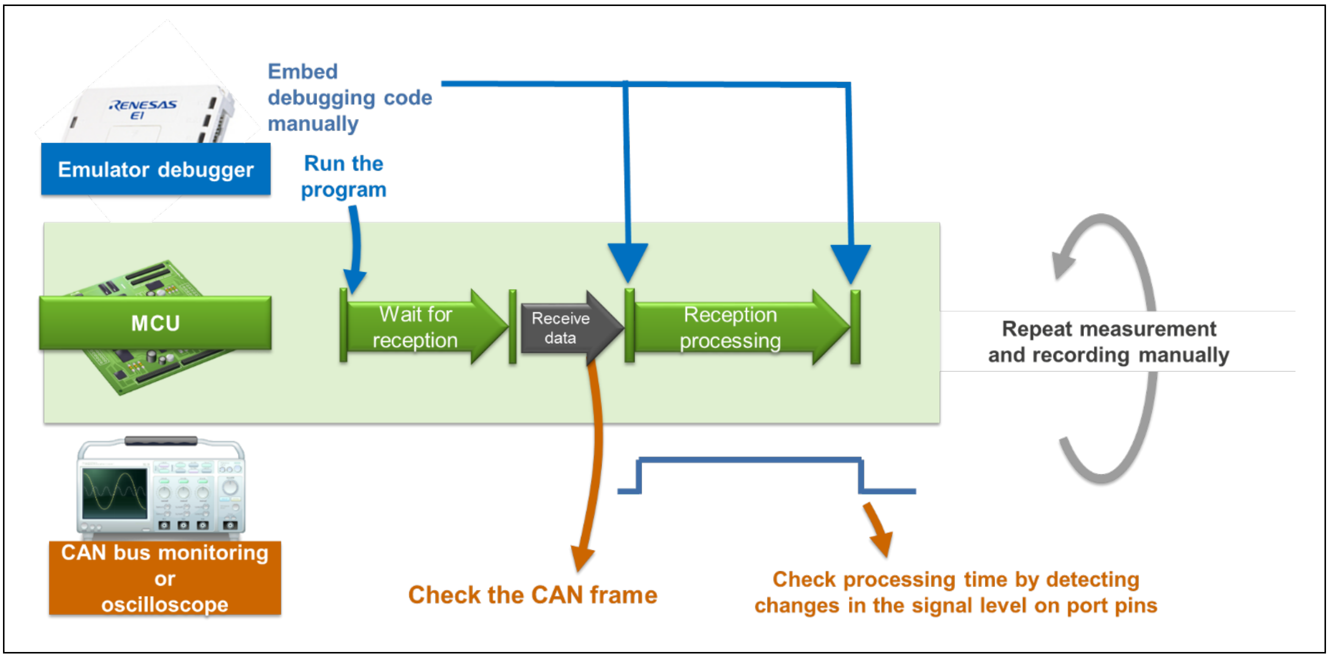


Figure 1-2 Conventional Method

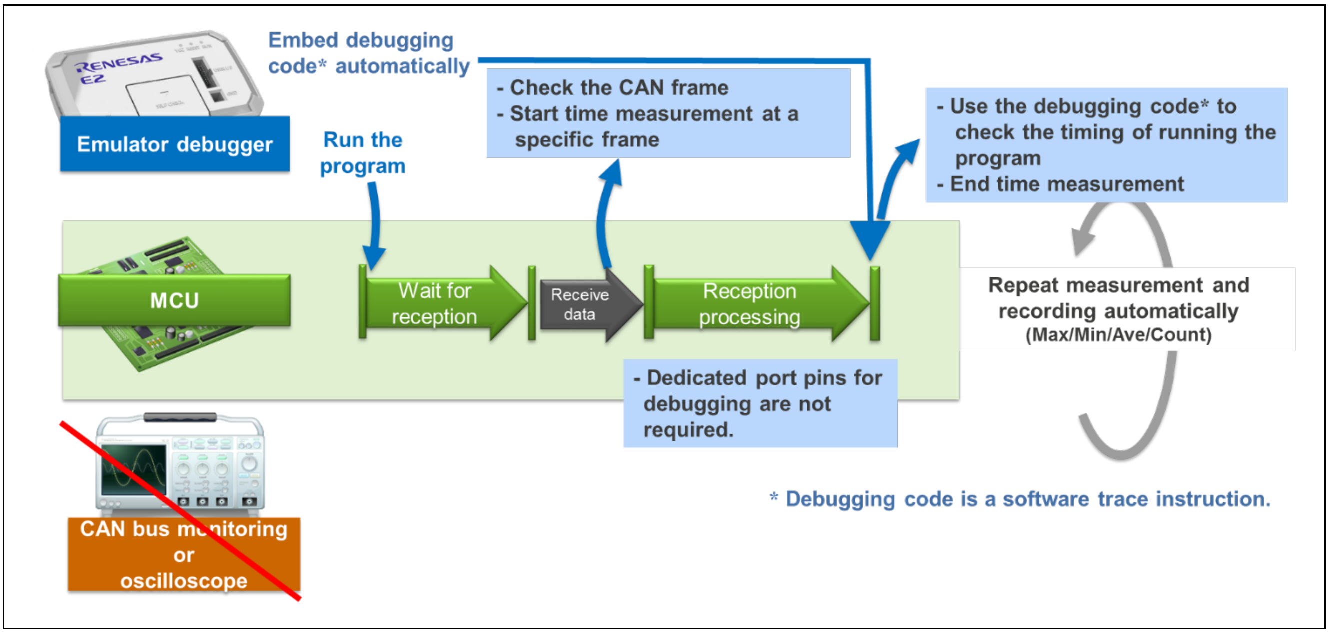
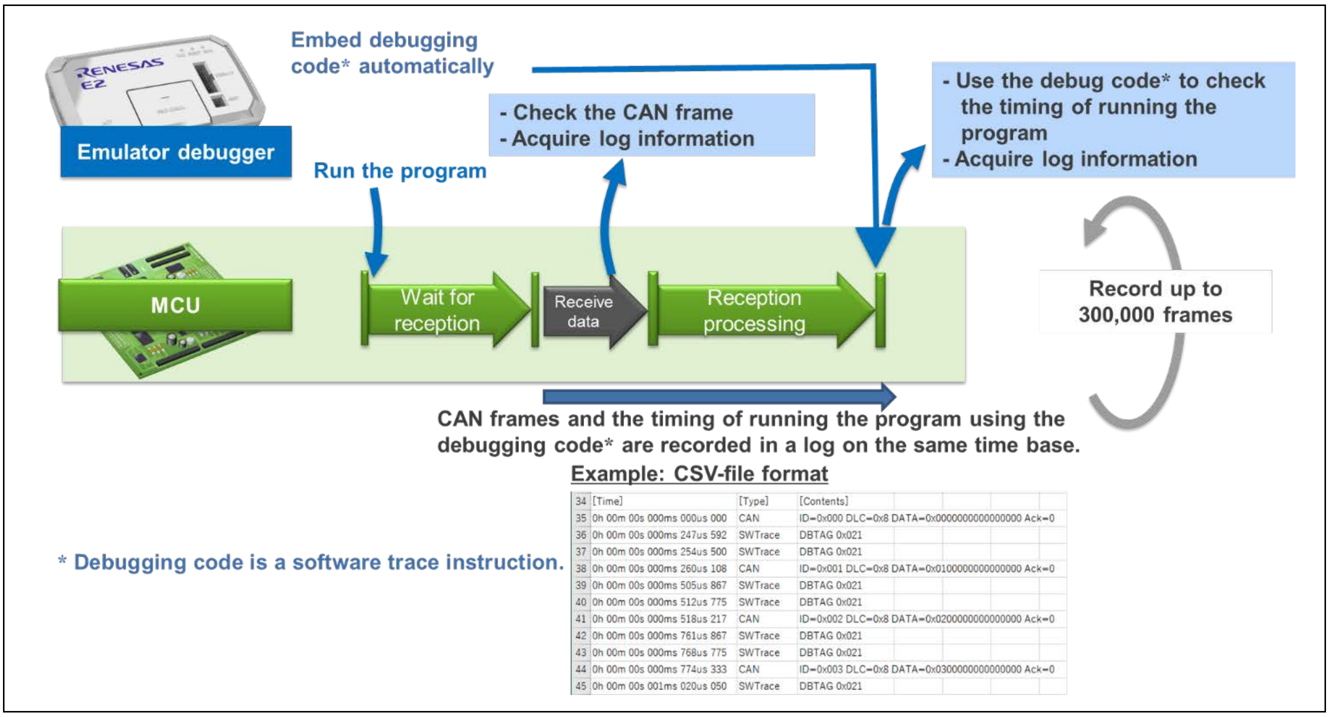


Figure 1-3 Measuring CAN Bus Reception Processing Times

1.2 Recording CAN Frames and Operation Timing of Software in a Log on the Same Time Base

The second characteristic is confirmation of CAN frames and the timing of running the program on the same time base by using only the E2 emulator, as shown in Figure 1-4. The log can be useful in identifying the cause in a case where the time from receiving a CAN frame to the processing being completed exceeds the design value. Figure 1-5 shows an example of the procedure for investigating the cause.



34	[Time]	[Type]	[Contents]
35	0h 00m 00s 000ms 000us 000	CAN	ID=0x000 DLC=0x8 DATA=0x0000000000000000 Ack=0
36	0h 00m 00s 000ms 247us 592	SWTrace	DBTAG 0x021
37	0h 00m 00s 000ms 254us 500	SWTrace	DBTAG 0x021
38	0h 00m 00s 000ms 260us 108	CAN	ID=0x001 DLC=0x8 DATA=0x0100000000000000 Ack=0
39	0h 00m 00s 000ms 505us 867	SWTrace	DBTAG 0x021
40	0h 00m 00s 000ms 512us 775	SWTrace	DBTAG 0x021
41	0h 00m 00s 000ms 518us 217	CAN	ID=0x002 DLC=0x8 DATA=0x0200000000000000 Ack=0
42	0h 00m 00s 000ms 761us 867	SWTrace	DBTAG 0x021
43	0h 00m 00s 000ms 768us 775	SWTrace	DBTAG 0x021
44	0h 00m 00s 000ms 774us 333	CAN	ID=0x003 DLC=0x8 DATA=0x0300000000000000 Ack=0
45	0h 00m 00s 001ms 020us 050	SWTrace	DBTAG 0x021

Figure 1-4 Log Acquisition of CAN Frames and Program Execution Timing

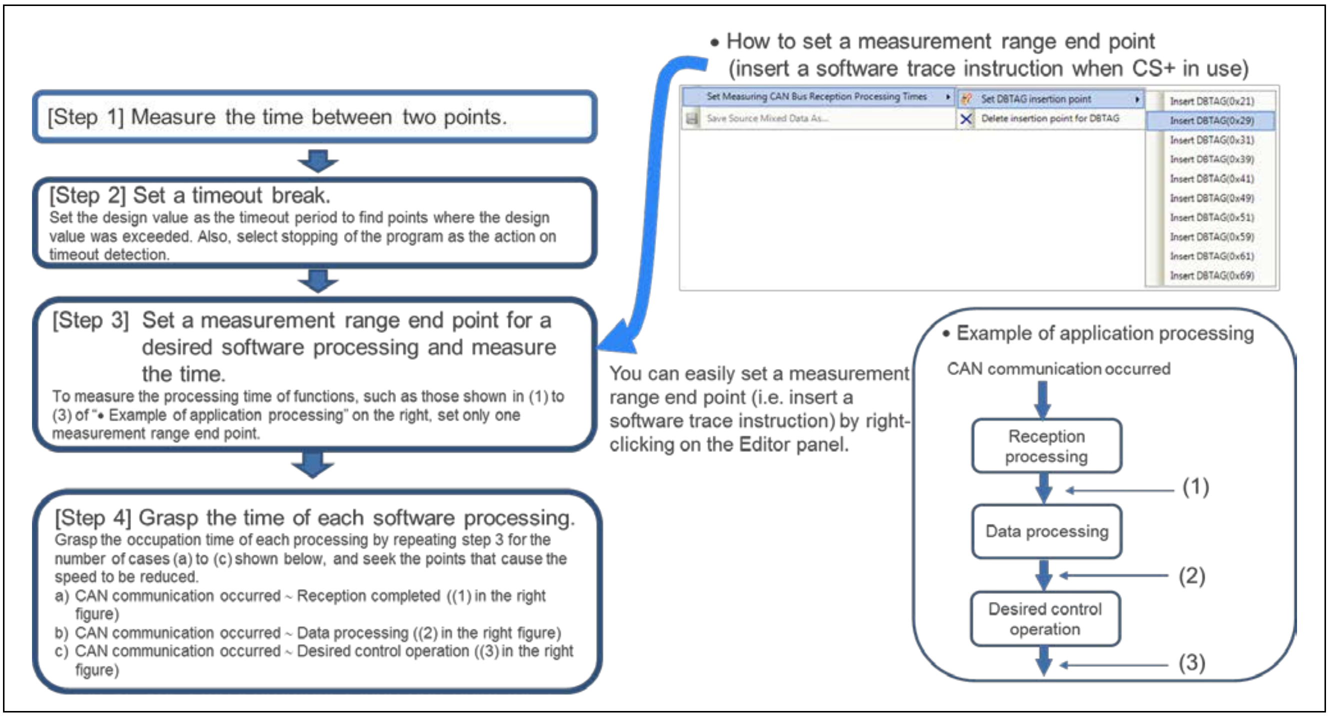


Figure 1-5 Example of Procedure for Investigating the Cause

2. Functional Description

This section introduces the specifications and performance of the CAN communication time measurement solution.

2.1 List of Functional Specifications

The functional specifications are listed in Table 2-1 and Table 2-2.

Table 2-1 List of Functional Specifications (1/2)

<table>
<tr><th>Item</th><th>Specification</th></tr>
<tr><td>Measurement method
Measurement target</td><td>• Measurement method
Connect the signal input to the CAN bus reception data input pin or external trigger input pin of the RH850 device to the E2 expansion interface with a test lead (accessory).
• Measurement target
The time between two points can be measured. One point is when a given CAN signal (of the specified ID and data, for which masking of bits is also possible) reaches the RH850, and the other point is when the code at the specified address (at which a debugging instruction is inserted*) is executed.
* A software trace instruction is used as a debugging instruction.
The following combinations are possible for measuring the time between two points.
<table>
<tr><th>No.</th><th>Start Condition for Time Measurement</th><th>End Condition for Time Measurement</th></tr>
<tr><td>1</td><td>Arrival of a given CAN bus communication signal</td><td>Execution of a given software trace instruction</td></tr>
<tr><td>2</td><td>Arrival of a given CAN bus communication signal</td><td>External trigger input</td></tr>
<tr><td>3</td><td>External trigger input</td><td>Execution of a given software trace instruction</td></tr>
<tr><td>4</td><td>External trigger input</td><td>External trigger input</td></tr>
</table>
</td></tr>
<tr><td>Measurement point</td><td>Measurement is possible through up to two CAN bus reception data input pins simultaneously.</td></tr>
<tr><td>Types of time measurement</td><td>Maximum time, minimum time, average time, pass count</td></tr>
<tr><td>Timeout detection</td><td>A timeout of the specified time can be detected.</td></tr>
<tr><td>Action on timeout detection</td><td>A break, stopping of internal tracing, or external trigger output can be selected as the action on timeout detection.</td></tr>
<tr><td>Recording capacity</td><td>8 Mbytes (E2 storage)
• In full stop mode for detection of only CAN frames
Number of recorded data items: 348,000 frames (dependent on the timing when CAN bus communication takes place)
• In full stop mode for only software tracing
Number of recorded data items: 500,000 frames (when using the DBCP instruction, the DBTAG instruction, or the DBPUSH instruction with only a single register specified)</td></tr>
</table>

Table 2-2 List of Functional Specifications (2/2)

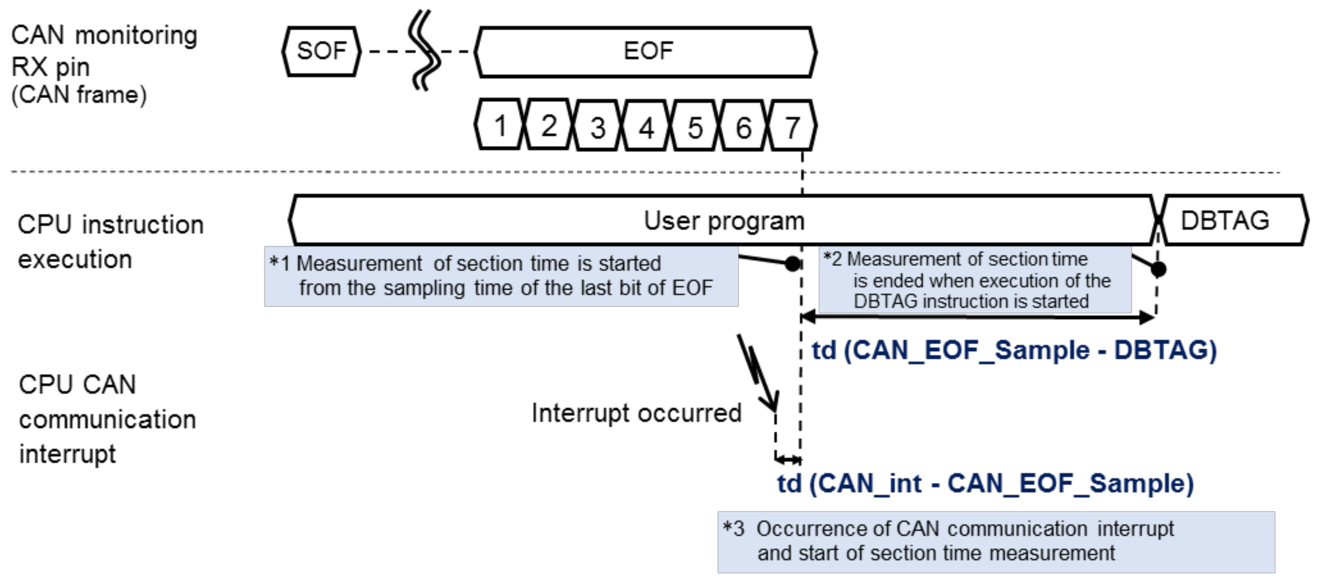
Item	Specification
Recording mode	The following three recording modes are supported. • Overwrite the recording memory and continue after the recording memory is full • Stop recording after the recording memory is full • Stop the program after the recording memory is full
Types of recorded data	Timestamp + software trace data (DBTAG number) Timestamp + CAN frame (ID, DATA, DLC, ACK)
CAN bus communication	Conforms to CAN 2.0B. 1M bps, 500K bps, 250K bps, and 125K bps are supported as the communication baud rate.
Sampling frequency	120 [MHz]
Timestamp	Count source: 8.33 [ns] (120 [MHz])
Storage format of recorded data	The recorded data are saved in the CSV format or Microsoft Office Excel book (*.xls) format.

2.2 Timing for Measuring CAN Bus Reception Processing Times

The timing for measuring the section time (td) is described for each measurement item.

For the measurement error of each measurement item, see "5.6 Notes on Time Measurement Function".

(1) Timing for measuring the section time from receiving CAN frame to software tracing (DBTAG)



[Note]

There is a period in which the section time cannot be measured.

When a software trace (DBTAG) instruction that was set as the end condition for measuring the section time is executed during the period of td (CAN_int – CAN_EOF-Sample)*3, the measured time becomes 0.

Figure 2-1 Timing for Measuring the Time of Section td (CAN_EOF_Sample - DBTAG)

(2) Timing for measuring the section time from receiving CAN frame to trigger input

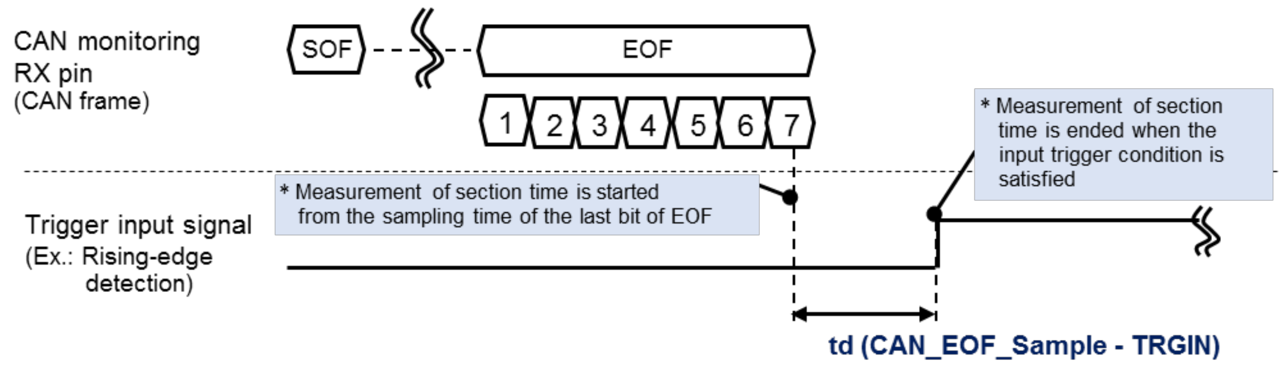


Figure 2-2 Timing for Measuring the Time of Section td (CAN_EOF_Sample - TRGIN)

(3) Timing for measuring the section time from trigger input to software tracing (DBTAG)

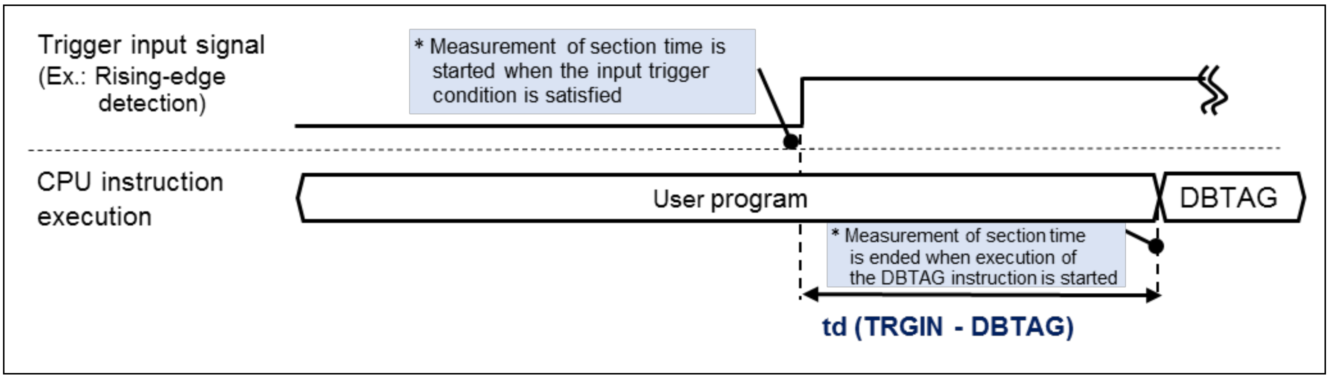


Figure 2-3 Timing for Measuring the Time of Section td (TRGIN – DBTAG)

(4) Timing for measuring the section time from trigger input to trigger input

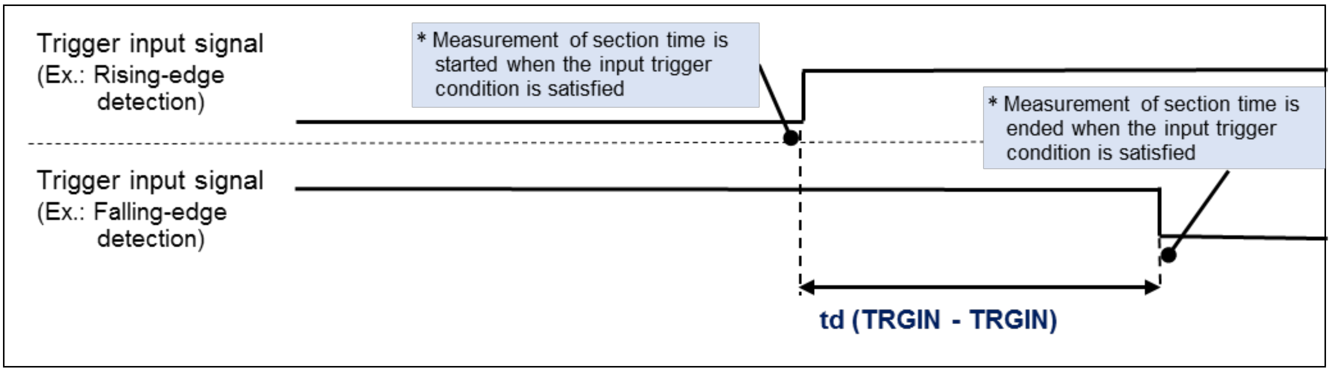


Figure 2-4 Timing for Measuring the Time of Section td (TRGIN – TRGIN)

2.3 Timestamp Specifications

(1) The timestamp starts from the data that was recorded first.

(2) The timing for recording the timestamp is as shown in the figure below.

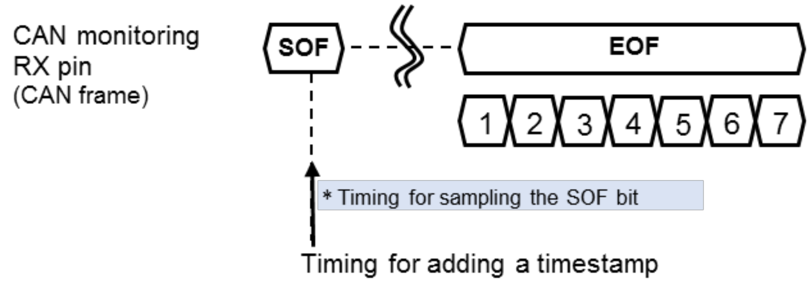


Figure 2-5 Timing for Recording the Timestamp (CAN Frame)

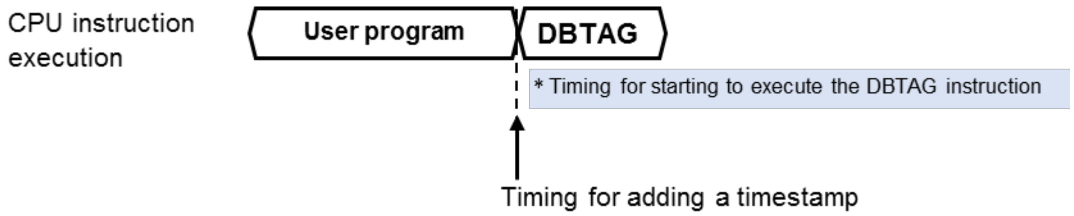


Figure 2-6 Timing for Recording the Timestamp (DBTAG Instruction)

3. Setup

The setup procedure is given in Figure 3-1.

[Step 1] Install the emulator debugger.
See "3.1 Installing the Emulator Debugger".

↓

[Step 2] Set the hardware environment.
See "3.2 Setting the Hardware Environment".

↓

[Step 3] Turn on the emulator and user system.
See "3.3 Turning on the E2 Emulator and User System".

Figure 3-1 Setup Procedure

3.1 Installing the Emulator Debugger

If you intend to use the E2 emulator, download and install the latest integrated development environment from the following Web site.

https://www.renesas.com/e2-download

3.2 Setting the Hardware Environment

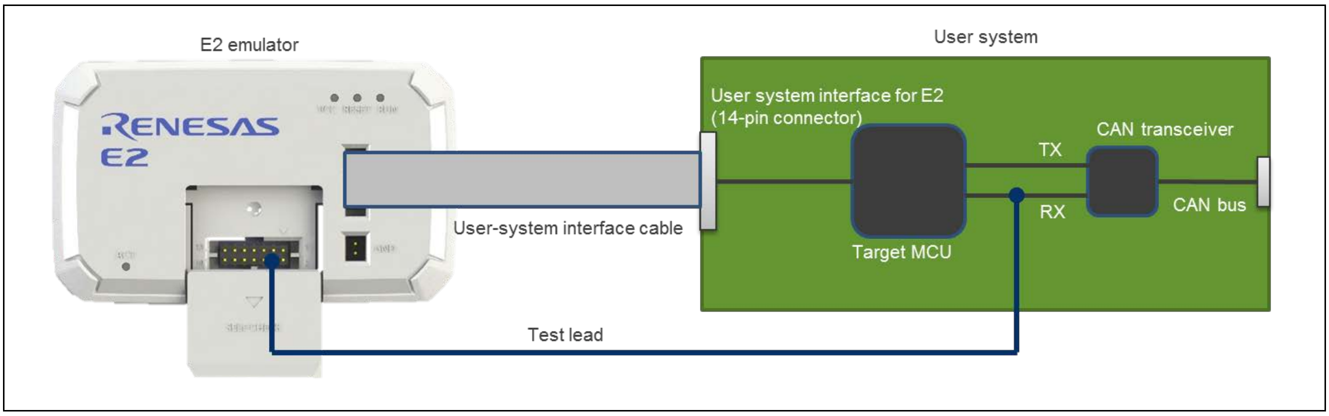


Figure 3-2 System Configuration Example

(1) Connecting the E2 emulator to the user system

Connect the E2 emulator to the user system with the user-system interface cable.

Set the switch on the 20-pin (1.27-mm pin spacing) to 14-pin (2.54-mm pin spacing) connector conversion adapter to position “1”.

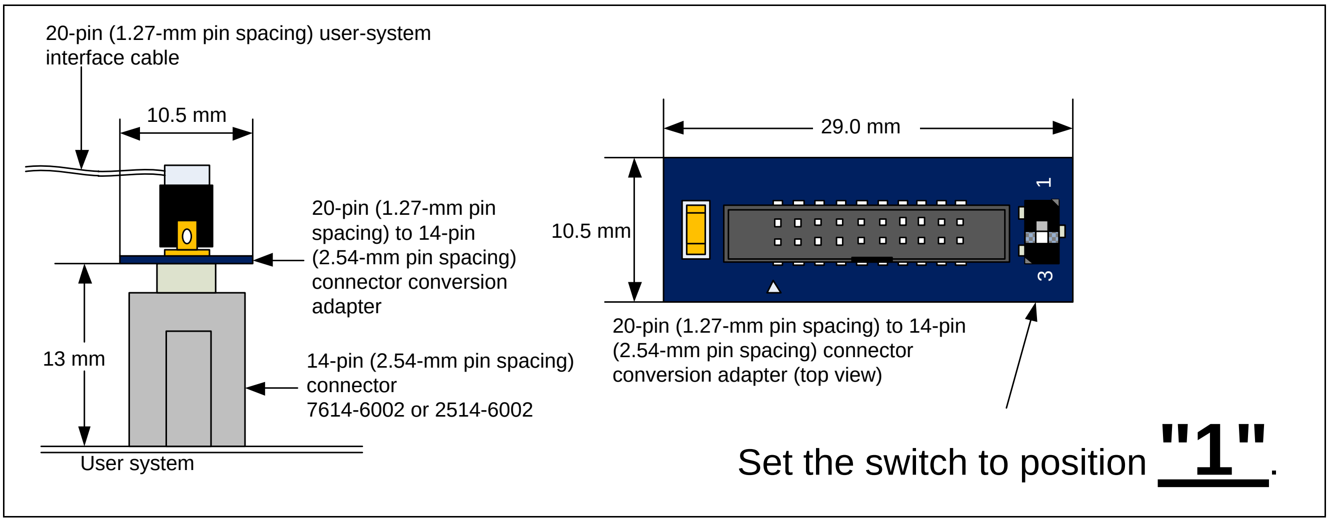


Figure 3-3 Connecting the User-System Interface Cable to the 14-Pin Connector

(2) Connecting to the E2 expansion interface

1. Connect the expansion interface (GND: pin 13) of the E2 emulator to the GND of the user system using the attached test lead.
2. Connect the expansion interface (CAN monitoring RX pins: pin 4 for ch0, pin 8 for ch1) of the E2 emulator to the RH850 CAN receive data input pin of the user system using the attached test lead.
   For pin arrangement of the E2 expansion interface, see Figure 3-4 and Table 3-1.

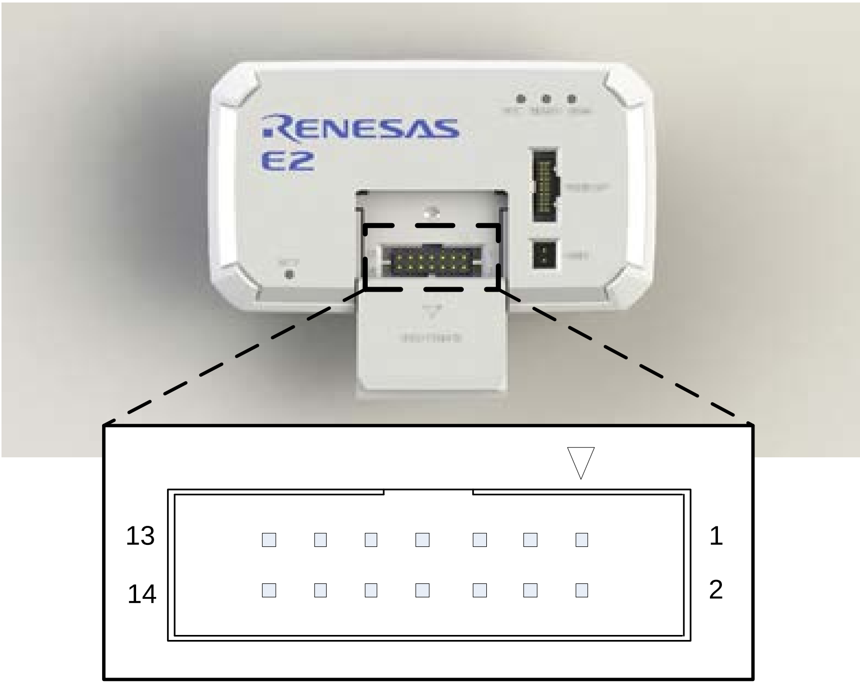


Figure 3-4 Pin Arrangement of E2 Expansion Interface

Table 3-1 Pins of E2 Expansion Interface

Pin No.	Input/Output	Description
1	-	-
2	-	-
3	-	-
4	Input	CAN monitoring RX (ch. 0)
5	-	-
6	-	-
7	-	-
8	Input	CAN monitoring RX (ch. 1)
9	Output	External trigger output (ch. 0)
10	Output	External trigger output (ch. 1)
11	Input	External trigger input (ch. 0)
12	Input	External trigger input (ch. 1)
13	-	GND
14	Output	A pin for output of the power-supply voltage for the E2 expansion interface (1.8 V to 5.0 V)

3.3 Turning on the E2 Emulator and User System

1. Connect the A plug of the USB interface cable to the USB interface connector of the host machine.
2. Connect the mini-B plug of the USB interface cable to the USB interface connector of the E2 emulator. The power of the E2 emulator is turned on by connecting the emulator to the host machine with a USB interface cable.
3. Turn on the user system.

4. Using the Emulator Debugger

4.1 Setup when Starting the Emulator Debugger

(1) Selecting the emulator

Right-click on on the Project Tree panel and select an emulator as shown in the figure below.

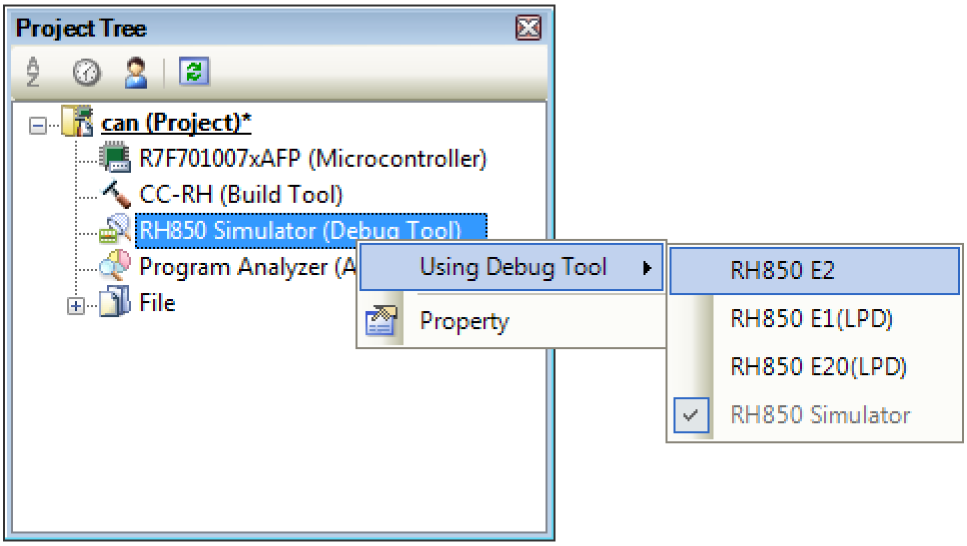


Figure 4-1 Starting the Emulator Debugger (Selection of Emulator)

(2) Making settings for connection

1. Right-click on  on the Project Tree panel and select [Property].

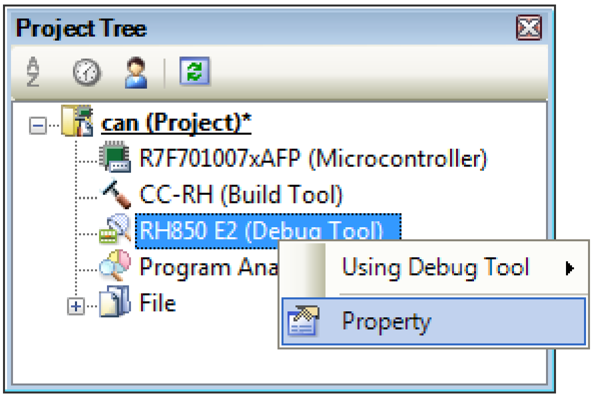


Figure 4-2 Starting the Emulator Debugger (Selection of Property)

2. Make a setting for [E2 Expansion Interface] in the [Connection Settings] tabbed page on the Property panel of [RH850 E2], as shown in the figure below.

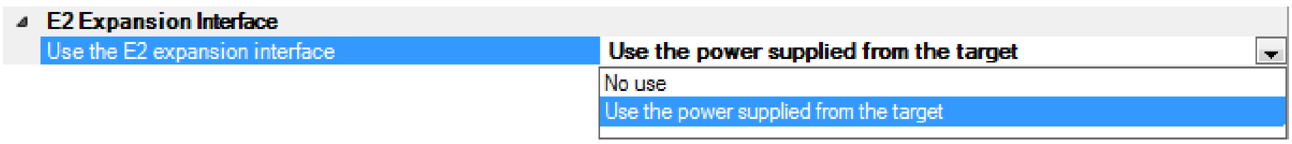


Figure 4-3 Starting the Emulator Debugger (Usage of E2 Expansion Interface)

3. Make a setting for software tracing in the [Connection Settings] tabbed page on the Property panel of [RH850 E2], as shown in the figure below.

   a) Enable software tracing through LPD output.

Figure 4-4 Starting the Emulator Debugger (Software Trace a)

   b) For a multicore MCU, select the core on which you wish to execute the debugging instruction (DBTAG instruction) that was set as a measurement range end condition.
   This setting is unnecessary for single-core MCUs. (The following menu will not appear.)

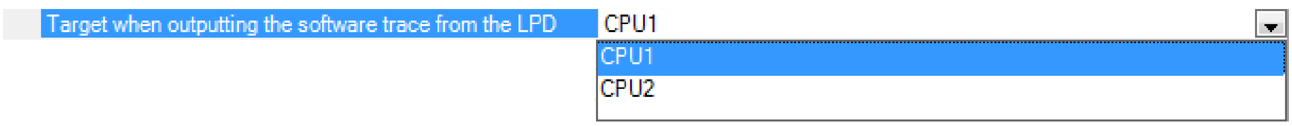


Figure 4-5 Starting the Emulator Debugger (Software Trace b)

   c) Enable output of the DBTAG instruction.

Figure 4-6 Starting the Emulator Debugger (Software Trace c)

(3) Connecting the emulator debugger

1. Select [Build & Download] from the [Debug] menu to start the emulator debugger and download data.

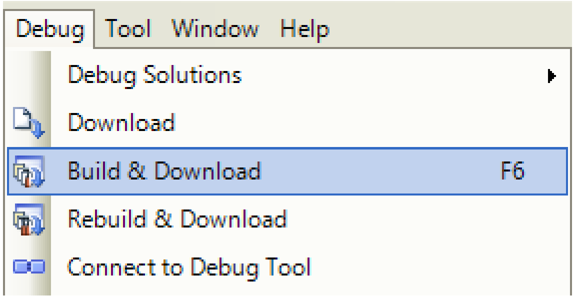


Figure 4-7 Starting the Emulator Debugger (Connection of Emulator Debugger)

4.2 Setting Measurement Conditions and Starting Measurement

(1) Setting a measurement range end point (inserting the DBTAG instruction for software tracing)

Use either one of the two methods shown below.

— Automatically inserting a debugging instruction (DBTAG) from the Editor panel of the emulator debugger
— Writing a debugging instruction (DBTAG) to the source code

- Automatically inserting a debugging instruction (DBTAG) from the Editor panel
  Move the cursor in the Editor panel to a point in the source code at which you want to stop measuring the section time. Right-click on the point and select the DBTAG instruction to be inserted, as shown in Figure 4-8.

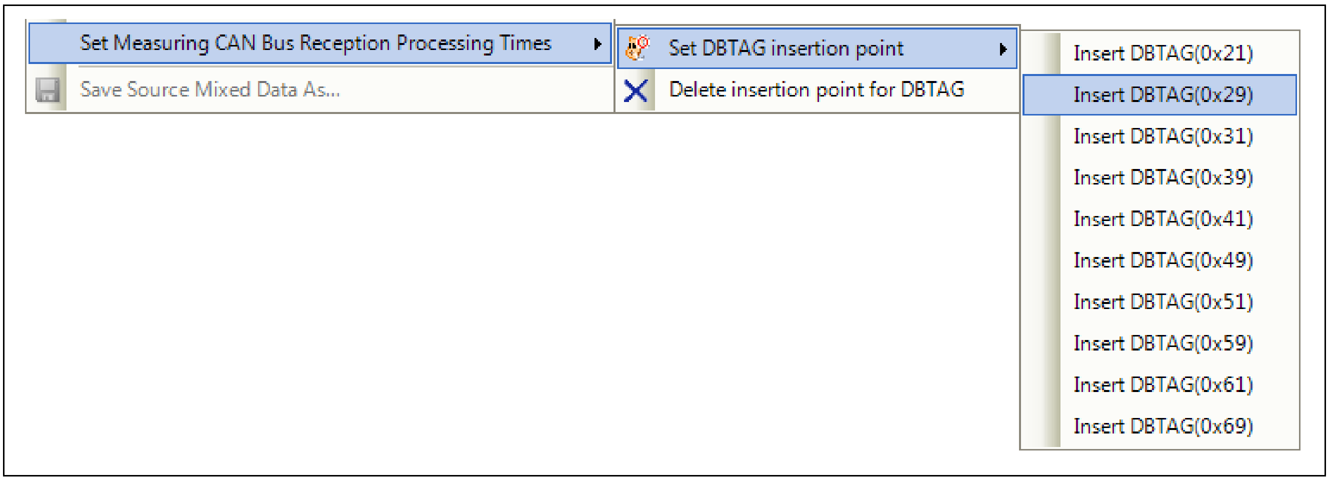


Figure 4-8 Setting Measurement Conditions and Starting Measurement (Insertion of DBTAG Instruction)

[Note] A debugging instruction cannot be automatically inserted when the compiler does not support this function.
Compiler supporting this function: CC-RH V1.06.00 or later

- Writing a debugging instruction (DBTAG) to the source code
  Sample code of the DBTAG instructions supported by the CAN communication time measurement solution is shown in Figure 4-9.

```
__dbgtag(0x21)
__dbgtag(0x29)
__dbgtag(0x31)
__dbgtag(0x39)
__dbgtag(0x41)
__dbgtag(0x49)
__dbgtag(0x51)
__dbgtag(0x59)
__dbgtag(0x61)
__dbgtag(0x69)
```

Figure 4-9 Sample Code of DBTAG Instructions

```
void INTRCAN5REC(void)
{
        unsigned long rcv_data[4];

        PORT.AP0     |= 0x0001;
        while(1){
                err_num = ReceiveCANCF( 15, rcv_data );
                if (err_num == 0) {
                        DataSave_push_back(DATA_SAVE_RECV, rcv_data);
                        RecvCount++;
                } else {
                        break;
                }
        }
        RSCAN0CFSTS15 &= ~0x00000008;
        __dbtag(0x21);
}
```

Figure 4-10 Setting Measurement Conditions and Starting Measurement (Sample Code of DBTAG Instruction)

(2) Selecting a solution

1. Select the [View] menu and click on [Solution List].

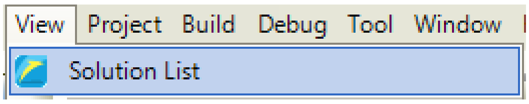


Figure 4-11 Setting Measurement Conditions and Starting Measurement (Display of Solution List)

2. Click on CAN (surrounded by a blue line in Figure 4-12) to select the CAN communication time measurement solution.

Figure 4-12 Setting Measurement Conditions and Starting Measurement (Solution List)

3. After step 2, the [Measuring CAN Bus Reception Processing Times] panel appears.

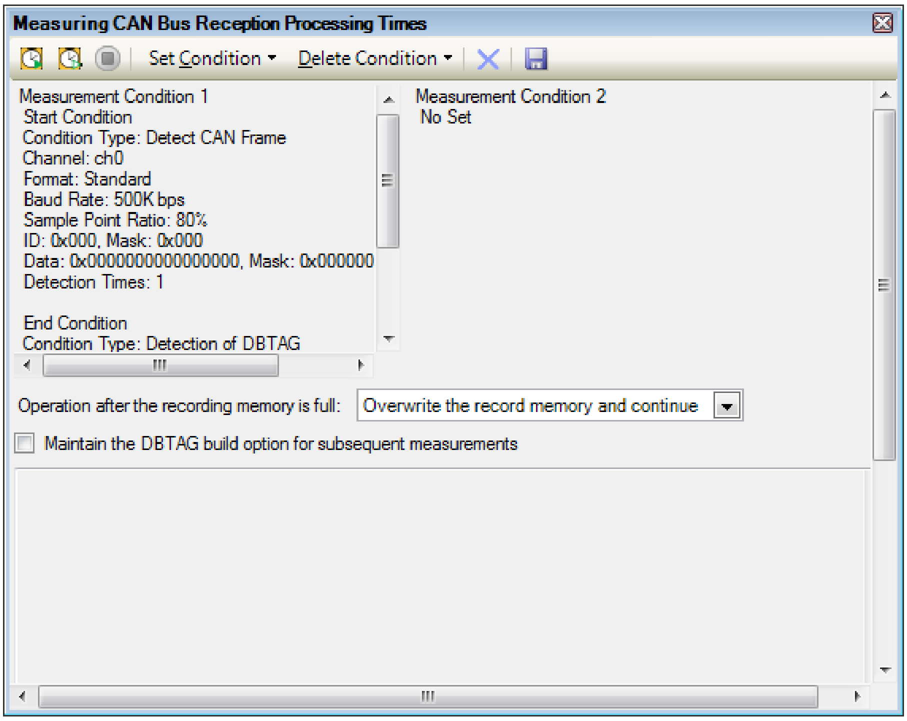


Figure 4-13 Setting Measurement Conditions and Starting Measurement ([Measuring CAN Bus Reception Processing Times] Panel)

(3) Setting measurement conditions

1. Click on the [Set Condition] menu in the [Measuring CAN Bus Reception Processing Times] panel.

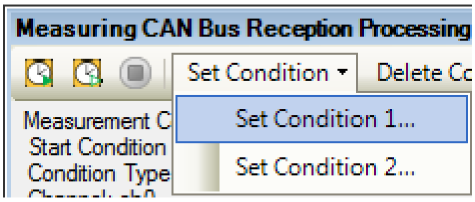
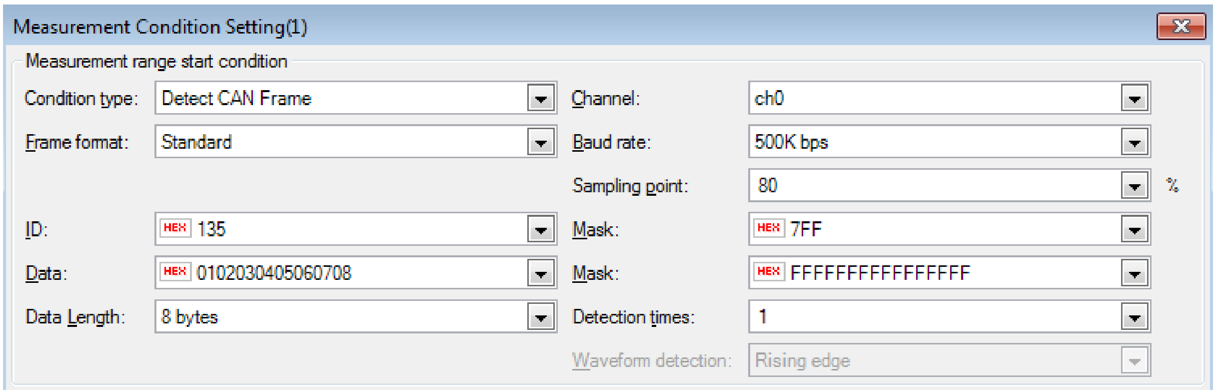


Figure 4-14 Setting Measurement Conditions and Starting Measurement (Setting Measurement Conditions)

2. Set detection of a CAN frame as the measurement range start condition.

   Setting example:

   ch0, standard format, baud rate: 500K bps, sampling point: 80%, ID: 0x135h
   data length: 8 bytes, data: 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08
   no mask, detection times: 1



- [Condition type]
  — Select detection of a CAN frame.
- [Frame format]
  — Select the standard or extended format.
- [ID], [Data], [Data Length]
  — Set the ID, data, and data length to be detected.
  — For data arrangement of the ID, see Figure 4-16.
  — For data arrangement of the data, see Figure 4-17.
- [Channel]
  — Select the channel to be used.
  ch0: Connect pin 4 of the E2 expansion interface.
  ch1: Connect pin 8 of the E2 expansion interface.
- [Baud rate]
  — Select the baud rate of communication.
- [Mask]
  — Set the mask value for the ID and data.
  — The definition of the mask value is as follows:
  0: Masked
  1: Not masked
  (The bit arrangement is the same as that for the ID and data.)
- [Detection times]
  — Set the number of times the CAN bus communication settings (ID, data, and data length) should be detected. When the condition has been satisfied for the number of times set as the detection times, the measurement range start condition is satisfied and time measurement is started.

Figure 4-15 Setting Measurement Conditions and Starting Measurement (Example: CAN Frame Detection is Set as the Start Condition)

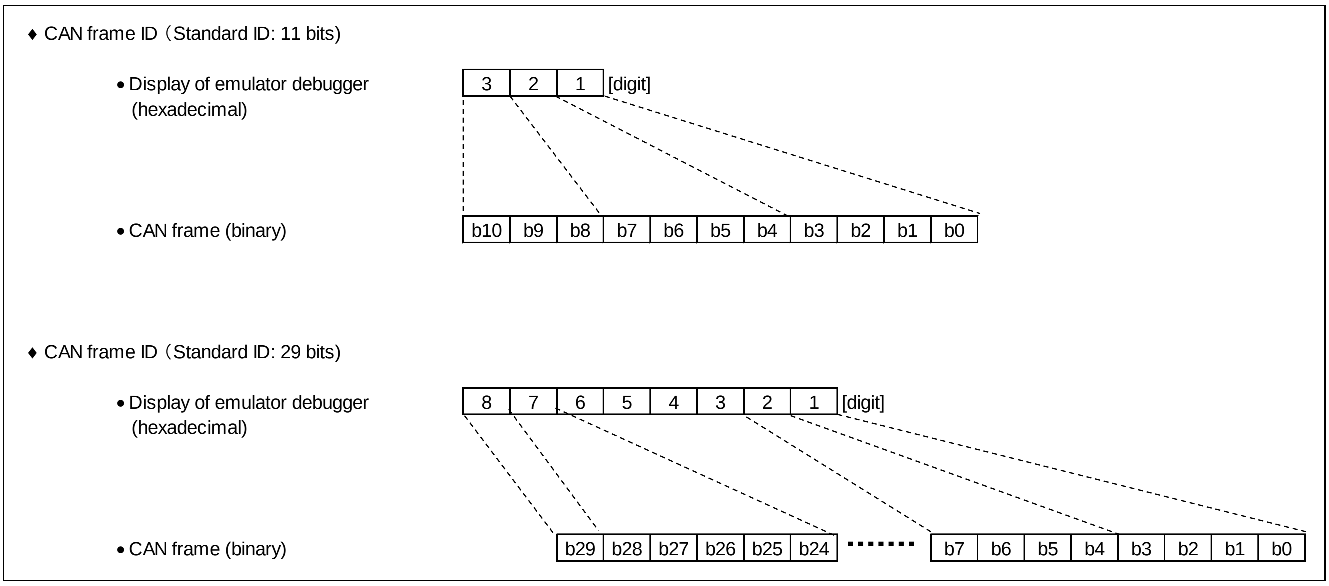


Figure 4-16 Setting Measurement Conditions and Starting Measurement (Data Arrangement of ID/ID Mask of CAN Frame)

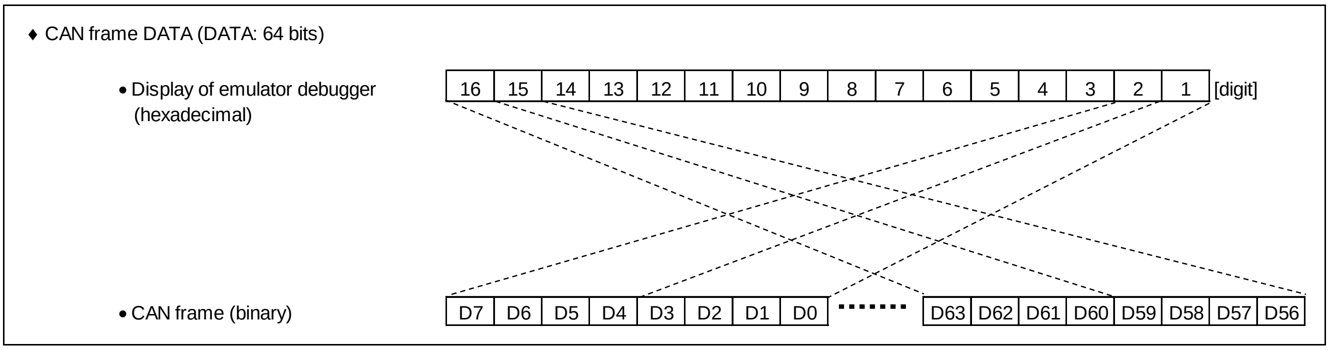
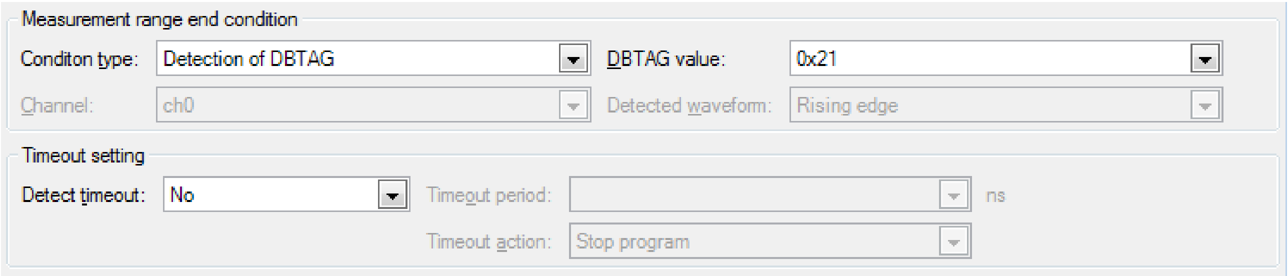


Figure 4-17 Setting Measurement Conditions and Starting Measurement (Data Arrangement of Data/Data Mask of CAN Frame)

3. Set detection of DBTAG as the measurement range end condition.

   Setting example:

   DBTAG value: 0x21



- [Measurement range end condition]
  - Select detection of DBTAG.
- [Timeout setting]
  - The user program or internal tracing can be stopped when the measurement range exceeds the timeout period.
  - The timeout period can be specified as a decimal number from 0 to 2,345,624,805,922,133 (unit: nanosecond).
- [DBTAG value]
  - Set the DBTAG value as the measurement range end condition.
  - For details on setting the DBTAG instruction, see 4.2 (1).

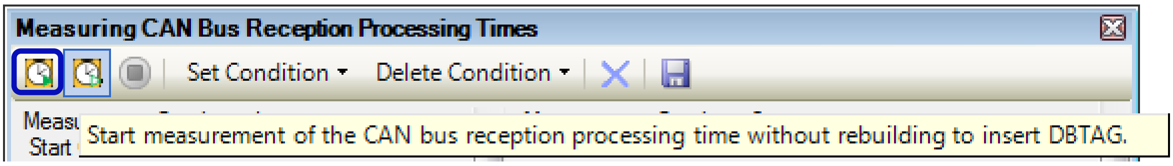
Figure 4-18 Setting Measurement Conditions and Starting Measurement (Example: DBTAG Detection is Set as the End Condition)

4. Start time measurement.

Click on the button located in the [Measuring CAN Bus Reception Processing Times] panel shown in Figure 4-19 to run the user program and start measuring the time.

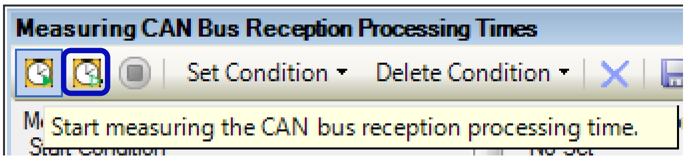
- When automatically inserting a debugging instruction (DBTAG) from the Editor panel
  See a) below.
- When writing a debugging instruction (DBTAG) to the source code
  See b) below.

a) Run the user program and start measuring the time (with build).



Before starting measurement, execute "Rebuild & Download" for inserting the DBTAG instruction and a reset.

b) Run the user program and start measuring the time (without build).



"Rebuild & Download" for inserting the DBTAG instruction is not necessary.

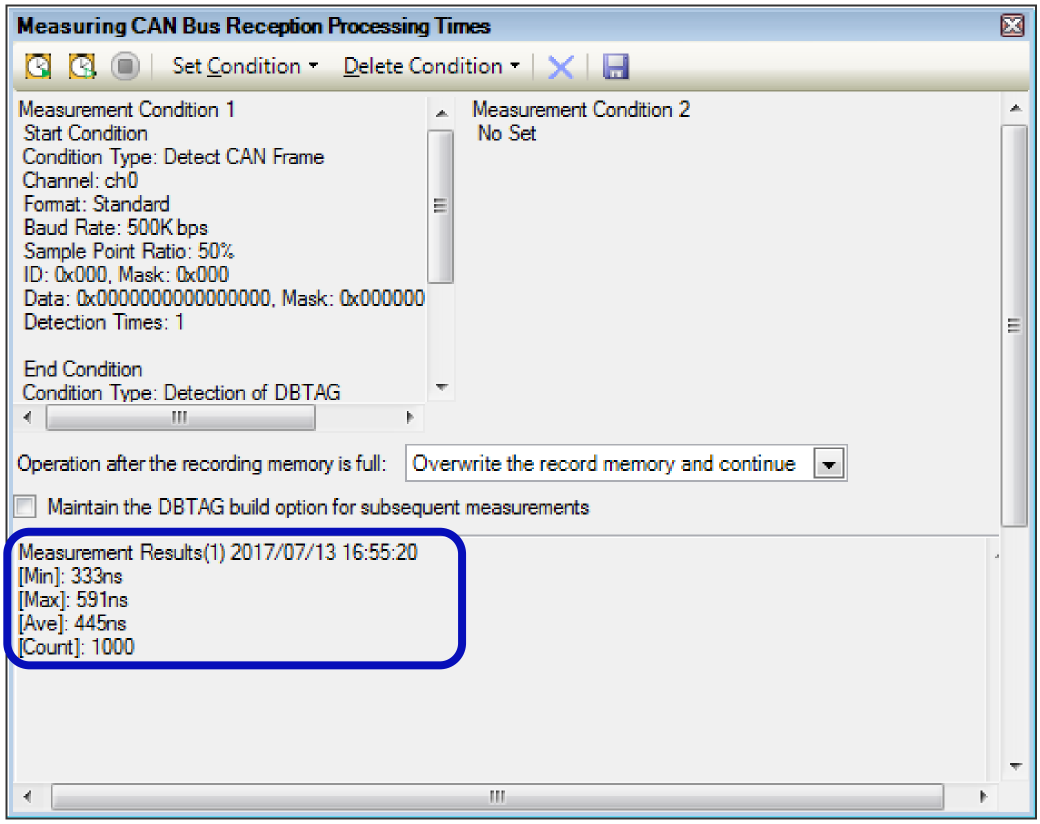
Figure 4-19 Setting Measurement Conditions and Starting Measurement (Start of Measurement)

5. Stop time measurement.

Time measurement stops when the user program is halted.

4.3 Referencing the Time Measurement Results

(1) Display of Min/Max/Ave/Count values of time measurement results



- Min
  The minimum value of the time measurement between two points is shown in nanosecond units.
- Max
  The maximum value of the time measurement between two points is shown in nanosecond units.
- Ave
  The average value of the time measurement between two points is shown in nanosecond units.
- Count
  The number of times measurement was carried out is shown.

For the timing to measure the section time, see Figure 2-1.

Figure 4-20 Displaying the Time Measurement Results (Display of Min/Max/Ave/Count Values)

(2) Referencing the CAN bus communication history and execution history of the DBTAG instruction

1. Click on the button in the [Measuring CAN Bus Reception Processing Times] panel and specify the file format (CSV format or Microsoft Office Excel book (*.xls) format) for recording the results.

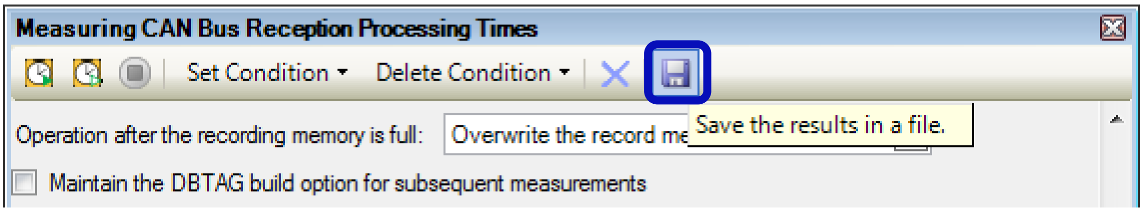


Figure 4-21 Referencing the Execution History (Save in a File)

2. Reference the file that was saved in step 1.

Example: Incrementation from CAN bus communication ID=0, incrementation from DATA=0, DBTAG instruction (0x21)

Example: CSV file format

34	[Time]	[Type]	[Contents]
35	0h 00m 00s 000ms 000us 000	CAN	ID=0x000 DLC=0x8 DATA=0x0000000000000000 Ack=0
36	0h 00m 00s 000ms 247us 592	SWTrace	DBTAG 0x021
37	0h 00m 00s 000ms 254us 500	SWTrace	DBTAG 0x021
38	0h 00m 00s 000ms 260us 108	CAN	ID=0x001 DLC=0x8 DATA=0x0100000000000000 Ack=0
39	0h 00m 00s 000ms 505us 867	SWTrace	DBTAG 0x021
40	0h 00m 00s 000ms 512us 775	SWTrace	DBTAG 0x021
41	0h 00m 00s 000ms 518us 217	CAN	ID=0x002 DLC=0x8 DATA=0x0200000000000000 Ack=0
42	0h 00m 00s 000ms 761us 867	SWTrace	DBTAG 0x021
43	0h 00m 00s 000ms 768us 775	SWTrace	DBTAG 0x021
44	0h 00m 00s 000ms 774us 333	CAN	ID=0x003 DLC=0x8 DATA=0x0300000000000000 Ack=0
45	0h 00m 00s 001ms 020us 050	SWTrace	DBTAG 0x021

- [Time] column
  The elapsed time is shown.
  Counting starts at the first time a CAN frame or software trace data is acquired.
- [Type] column
  CAN: CAN frame
  SWTrace: Software trace
- [Contents] column
  CAN: CAN frame ID, DLC, DATA, and ACK are shown.
  — For data arrangement of the ID, see Figure 4-16.
  — For data arrangement of the data, see Figure 4-17.
  SWTrace: Instructions executed in software tracing are shown.

Figure 4-22 Referencing the Execution History (Reference to CSV File)

5. Usage Notes

5.1 Combined Use with Other Functions

(1) When this solution is used, the following functions cannot be used.

- Hot plug-in connection
- While the user program is running, the only function that can be manipulated from the emulator debugger is a forcible break. The other functions are not usable.

5.2 Notes on Breaks

(1) Immediately after the program starts running, there is a period in which breaks cannot be detected.
- When the CPU is operating in the low-speed OCO mode
  Breaks are not detectable during the 100-μsec period after a program has started to run.
- Other than above
  Breaks are not detectable during the 10-μsec period after a program has started to run.

(2) After a break event occurs, there is a delay before the actual break occurs.
For the delay time, see Table 5-1.

Table 5-1 Relationship between the Break Type Supported by the CAN Communication Time Measurement Solution and the Delay Time

No.	Break Type	Device Type	Delay Time
1	Timeout break during the period from a CAN frame to software tracing	Device that supports deep stop mode	Table 5-2, Table 5-3
2		Device that does not support deep stop mode	Table 5-4, Table 5-5
3	Timeout break during the period from trigger input to software tracing	Device that supports deep stop mode	Table 5-2, Table 5-3
4		Device that does not support deep stop mode	Table 5-4, Table 5-5
5	Timeout break during the period from a CAN frame to trigger input	—	Table 5-6, Table 5-7
6	Timeout break during the period from trigger input to trigger input	—	Table 5-6, Table 5-7

Table 5-2 Delay Time until Break Occurrence in LPD 4-pin Mode (with software tracing, when the device supports deep stop mode)

Debug I/F / Delay Time	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Delay time	5 to 8 [μs]	6 to 10 [μs]	11 to 16 [μs]

Table 5-3 Delay Time until Break Occurrence in LPD 1-pin Mode (with software tracing, when the device supports deep stop mode)

Debug I/F / Delay Time	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Delay time	26 to 37 [μs]	51 to 70 [μs]	99 to 135 [μs]

Table 5-4 Delay Time until Break Occurrence in LPD 4-pin Mode (with software tracing, when the device does not support deep stop mode)

Debug I/F / Delay Time	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Delay time	8 to 11 [μs]	10 to 15 [μs]	13 to 19 [μs]	22 to 31 [μs]

Table 5-5 Delay Time until Break Occurrence in LPD 1-pin Mode (with software tracing, when the device does not support deep stop mode)

Debug I/F / Delay Time	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Delay time	51 to 70 [μs]	99 to 135 [μs]	196 to 266 [μs]

Table 5-6 Delay Time until Break Occurrence in LPD 4-pin Mode (without software tracing)

Debug I/F / Delay Time	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Delay time	3 to 7 [μs]	4 to 12 [μs]	6 to 16 [μs]	10 to 29 [μs]

Table 5-7 Delay Time until Break Occurrence in LPD 1-pin Mode (without software tracing)

Debug I/F / Delay Time	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Delay time	25 to 75 [μs]	50 to 147 [μs]	98 to 290 [μs]

5.3 Notes on Software Tracing

(1) Execution interval of software trace instructions

The execution interval of software trace instructions is restricted.

The time shown in Table 5-8 should be ensured as the execution interval of software trace instructions. If the required time is not satisfied, the following problems will arise.

- Time measurement by this solution cannot be performed correctly.
- A loss occurs in software tracing and data cannot be recorded.

Table 5-8 Required Time of Execution Interval of Software Trace Instructions

Debug I/F / Instruction Type	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
DBTAG (without PC)	0.1 [μs]	0.2 [μs]	0.3 [μs]	0.6 [μs]	1.6 [μs]	3.2 [μs]	6.4 [μs]
DBTAG (with PC)	2.3 [μs]	4.7 [μs]	7 [μs]	13.9 [μs]	38 [μs]	76 [μs]	152 [μs]

(2) Software tracing function when the 1-pin LPD interface is selected

When the 1-pin LPD is selected and a break is generated as a forced break, a trace-full break from the E2 storage, or a break due to the input of an external trigger, software tracing cannot be used when execution of the program is subsequently resumed. To proceed with software tracing again, re-connect the emulator to the debugger.

(3) Cautionary point regarding trace data acquired by software tracing

When a break is generated as a forced break, a trace-full break from the E2 storage, or a break due to the input of an external trigger, information from a debugging instruction that was executed immediately before the break will not be stored in the E2 storage.

When a debugging instruction is executed during single-stepped execution and a software break or hardware break is specified and executed by the debugging instruction, software trace data are not output through the LPD interface.

When trace acquisition is stopped due to a break generated by a software break, hardware break, event break, or full break of internal trace memory, the history of execution from a DBCP instruction executed in the debugging area is stored as the final trace data in the E2 storage and internal trace memory after the break in execution.

(4) Automatic insertion of a debugging instruction (DBTAG) from the emulator debugger

If the compiler does not support this function, a debugging instruction cannot be automatically inserted.

Compiler supporting this function: CC-RH V1.06.00 or later

5.4 Notes on External Trigger Function

(1) Delay time of external trigger output

After an event for external trigger output is detected, there is a delay before the output of the external trigger signal is started.

The delay time is explained in Table 5-9 to Table 5-13.

Table 5-9 Delay Time of Trigger Output at Detection of Software Trace (DBTAG Instruction) in LPD 4-pin Mode

Debug I/F / Delay Time	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Delay time	(CPU clock × 3) + (0.05 to 0.1) [μs]	(CPU clock × 3) + (0.07 to 1.9) [μs]	(CPU clock × 3) + (0.1 to 2.7) [μs]	(CPU clock × 3) + (0.1 to 5.3) [μs]

Table 5-10 Delay Time of Trigger Output at Detection of Software Trace (DBTAG Instruction) in LPD 1-pin Mode

Debug I/F / Delay Time	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Delay time	(CPU clock × 3) + (0.5 to 12) [μs]	(CPU clock × 3) + (0.9 to 23) [μs]	(CPU clock × 3) + (1.8 to 45) [μs]

Table 5-11 Delay Time of Trigger Output at Detection of CAN Frame

Delay time	8 to 17 [ns]

Table 5-12 Delay Time of Trigger Output at Detection of Timeout in LPD 4-pin Mode

Debug I/F / Delay Time	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Delay time	(CPU clock × 3) + (0.05 to 0.1) [μs]	(CPU clock × 3) + (0.07 to 1.9) [μs]	(CPU clock × 3) + (0.1 to 2.7) [μs]	(CPU clock × 3) + (0.2 to 5.3) [μs]

Table 5-13 Delay Time of Trigger Output at Detection of Timeout in LPD 1-pin Mode

Debug I/F / Delay Time	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Delay time	(CPU clock × 3) + (0.5 to 12) [μs]	(CPU clock × 3) + (0.9 to 23) [μs]	(CPU clock × 3) + (1.8 to 45) [μs]

(2) Setting of external trigger output

This solution cannot be used together with the external trigger input/output function that is provided as a basic function of the emulator debugger. While measuring CAN bus reception processing times, only the external trigger setting that was set by the CAN communication time measurement solution will be enabled.

5.5 Note on CAN Bus Monitoring

(1) Setup

When using this solution, connect the CAN receive data input pins of the RH850 to the E2 emulator with a test lead. If the attached test lead cannot be used for connection, separately prepare a connectable cable.

For the setup method, see "3.2 Setting the Hardware Environment".

5.6 Notes on Time Measurement Function

(1) Measurement error in time measurement function

There is a measurement error in the time measurement function. The measurement error is explained for each time measurement item in Table 5-14 to Table 5-16.

Table 5-14 Error When Measuring the Time from Receiving CAN Frame or Trigger Input to Software Tracing in LPD 4-pin Mode

Debug I/F / Measure-ment Error	LPD 4-pin 33 [MHz]	LPD 4-pin 16.5 [MHz]	LPD 4-pin 11 [MHz]	LPD 4-pin 5.5 [MHz]
Measurement error	+(CPU clock × 3) + (0 to 0.06) [μs]	+(CPU clock × 3) + (0 to 0.1) [μs]	+(CPU clock × 3) + (0 to 0.15) [μs]	+(CPU clock × 3) + (0 to 0.25) [μs]

Table 5-15 Error When Measuring the Time from Receiving CAN Frame or Trigger Input to Software Tracing in LPD 1-pin Mode

Debug I/F / Measure-ment Error	LPD 1-pin 2 [MHz]	LPD 1-pin 1 [MHz]	LPD 1-pin 500 [kHz]
Measurement error	+(CPU clock × 3) + (0 to 0.6) [μs]	+(CPU clock × 3) + (0 to 1.1) [μs]	+(CPU clock × 3) + (0 to 2.3) [μs]

Table 5-16 Error When Measuring the Time from Receiving CAN Frame to Trigger Input

Measurement error	+17 to 34 [ns]

(2) Restriction for setting the measurement range start/end condition when measuring CAN bus reception processing times

- The following cannot be set as measurement condition 1 or 2.
  — CAN bus monitoring of the same channel
  — External trigger input of the same channel
- The same operation cannot be set in [Timeout action] of measurement condition 1 or 2.

(3) When a break occurs while measuring the section time

After a start condition for time measurement is satisfied, if a break occurs before the end condition is satisfied, this will be excluded from time measurement.

(4) Stopping internal tracing on timeout detection

When [Output the software trace from the LPD] (see Figure 4-4) is selected, stopping internal tracing is not selectable as the action on timeout detection.

(5) Measurement range of the section time

There is a period in which the section time cannot be measured. When the software trace (DBTAG) instruction that was set as the end condition for measuring the section time is executed at the timing defined with td (CAN_int – CAN_EOF-Sample) in Figure 2-1, the measurement result of the section time becomes 0. Make adjustments so that no software trace instruction will be executed during the period of td (CAN_int – CAN_EOF-Sample). The maximum value of td (CAN_int – CAN_EOF-Sample) is the time for one bit of baud rate of CAN bus communication.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul. 16, 2017	—	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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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.

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, HAL II 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 6.0