RL78 Debugging Functions Using the Serial Port

Introduction

This application note describes how to use the RL78 debugging functions using the serial port.

Target Device

RL78

Contents

1. Overview 4
1.1 Overview of COM Port debugging 4
1.2 Applicable devices 4

2. User system design 5
2.1 Example of recommended connection between USB-to-serial conversion adapter and MCU 5
2.1.1 General RL78 family connectivity 5
2.2 Things to keep in mind when connecting 6
2.2.1 RESET# pin 6
2.2.2 TOOL0/TOOLTxD/TOOLRxD pins 8
2.2.3 GND 9
2.2.4 Power supply voltage 9
2.2.5 Caution about RESET Port and general-purpose port are dual-use pin 9

3. Usage notes 10
3.1 Power on/off 10
3.1.1 Power supply sequence 10
3.2 MCU resources that are used 11
3.2.1 Securing the debug monitor area 12
3.2.2 Securing the stack area for debugging 13
3.2.3 Setting the on-chip debug option byte area 14
3.2.4 Setting the security ID 15
3.3 Reset 16
3.3.1 Behavior after reset 16
3.3.2 Registers after reset 16
3.3.3 Pin reset during break 16
3.4 Flash memory 16
3.4.1 Debugging notes for self-programming 16
3.4.2 Voltage at which flash memory cannot be written and operation in flash operation mode 17
3.5 GDIDIS function 18
3.6 MCUs that are used in debugging 18

3.6.1 Use in mass-production products 18
3.6.2 Standalone operation check 18
3.7 Final evaluation of user program 18
3.8 Debugging functions 19
3.8.1 “Go to Here” function 19
3.8.2 Debugging in standby mode 19
3.8.3 Pseudo RRM/DMM functions 20
3.8.4 Start/Stop functions 20
3.8.5 Emulation of flash memory CRC accumulator function 20
3.8.6 Notes on the break function 21
3.8.7 Setting and deleting events during user program execution 21
3.8.8 Trace function 21
3.8.9 Precautions when using flash read protection 21
3.9 Adjust of Communication baud rate 22

Revision History 23

Definitions of Terms

The terms used in this document are defined and used as follows.

Host machine

A personal computer for controlling the emulator.

User system

The customer's application system using the MCU to be debugged.

User program

The application program to be debugged.

Flash writing software

In this document, this term refers to the Renesas Flash Programmer.

Meaning of “#” at the end of the pin name (signal name)

The # at the end of a pin name (signal name) indicates that it is a “Low” active pin (signal) (example: RESET#).

1. Overview

This guide presents the information you should be aware of when debugging RL78 devices with COM Port debugging using a commercial USB-to-serial conversion adapter.

1.1 Overview of COM Port debugging

With the COM Port debugging system, you perform debugging by connecting to the target board from a USB port of the host machine via a USB-to-serial conversion adapter instead of an emulator such as E2. The adapter is recognized as the COM Port of the host machine when communicating with the target device using the serial port.

It enables debugging of target devices using a commercially available USB-to-serial conversion adapter.

1.2 Applicable devices

Devices in the RL78 family that support COM Port debugging. Check the device's manual to see if it is supported.

Table 1-1 List of debugging functions

Item		Support		Content
		COM Port	E2 Lite	
Reference/change memory during program execution				
Pseudo real-time RAM monitor (RRM)		Y*1	Y	CPU occupied during memory reference
Dynamic Memory Modification (DMM)		Y*1	Y	CPU occupied during memory changes
Event function		Up to 2 points	Up to 2 points	Can be used for hardware breaks (or trace function*2)
Break functions	Software breaks	Y	Y	Up to 2000 points
	Hardware breaks	Y	Y	Execution address or data access
	Forced breaks	Y	Y	-
Trace functions	Acquired information	Y	Y	Branch source PC information
	Start event	Y	Y	User program execution start, event start
	End event	Y	Y	User program stop, event end, trace full
Execution time measurement function		N	Y	-
Hot plugin		N	Y	-
Coverage function		N	N	-

Y: supported, N: not supported

*1 RRM/DMM is possible on a per-variable basis, but it is recommended that the memory panel display should be 1 or 2 lines and the update interval should be 5000 msec or more. If too much is displayed, the debugger may become unresponsive.

*2 Only for devices with a trace function

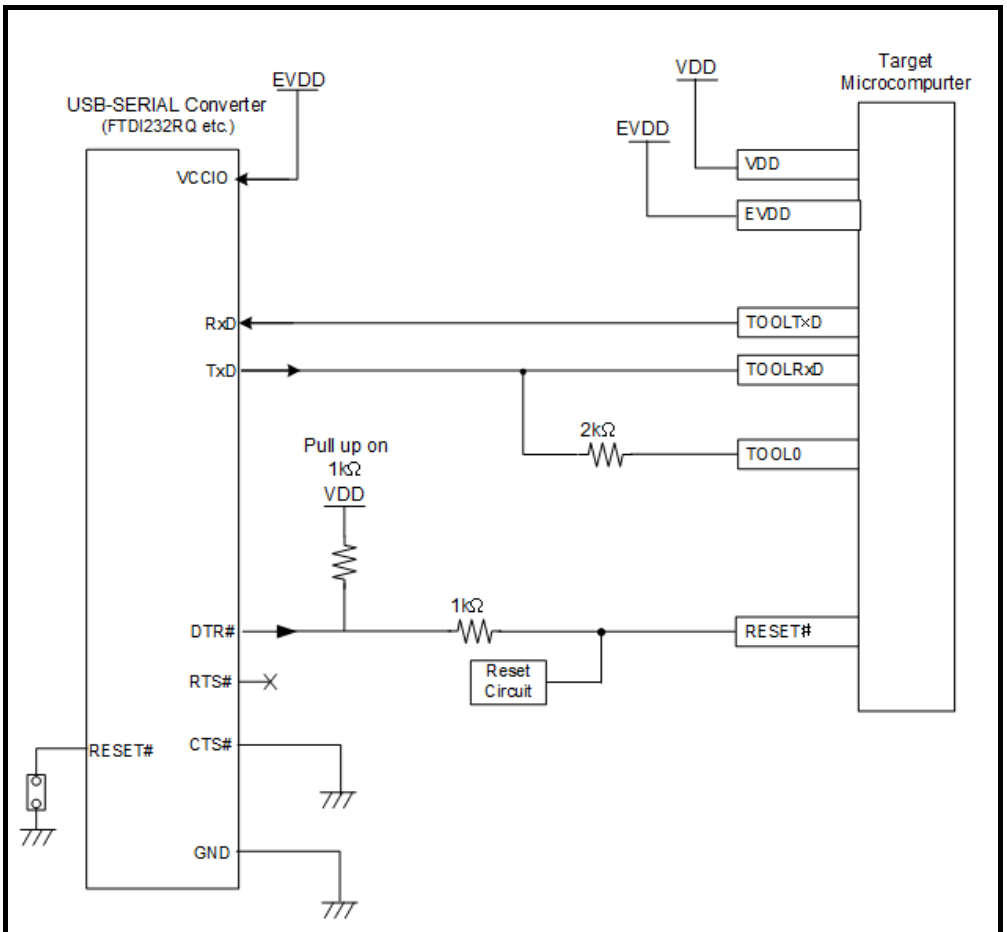
2. User system design

2.1 Example of recommended connection between USB-to-serial conversion adapter and MCU

The recommended connection between the USB-to-serial conversion adapter and the MCU when using COM Port debugging is shown below. For details on the processing of each signal line, refer to 2.2 Things to keep in mind when connecting.

2.1.1 General RL78 family connectivity

An example of connecting the USB-to-serial conversion adapter to the RL78 family is shown in Figure 2.1.



* CTS# is unused. Perform pin processing according to the specifications of the serial conversion adapter you are using.

Figure 2.1 Example of connecting the USB-to-serial conversion adapter to the RL78 family

2.2 Things to keep in mind when connecting

The pattern length between the USB-to-serial conversion adapter and the MCU should be as short as possible (50 mm or less is recommended). Also, do not route signal lines on the board other than between the USB-to-serial conversion adapter and the MCU.

For details on pin processing when COM Port debugging is not used, refer to the user's manual (hardware edition) of the relevant MCU.

2.2.1 RESET# pin

The RESET# pin is used for resetting to the device.

An example of a connection to the RESET# pin area is shown in Figure 2.2.

When performing flash writing from the flash writing software, ensure that the reset signal on the user system does not collide with the reset signal from the COM Port debugging.

Note that COM Port debugging does not support the RESET# pin masking function.

In addition, configure the debugger properties to match the pin (DTR/RTS) that controls the RESET# pin. By default, the debugger/flash writing software uses DTR, so if you want to use RTS, be sure to change the property settings.

The recommended connection is the DTR connection shown in Figure 2.2. If you are using a USB-to-serial conversion adapter without DTR, use the RTS connection in Figure 2.3.

When using Printf debugging in the Arduino development environment, refer to Figure 2.4 when connecting. Since RTS is fixed at LOW and the buffer output is disabled, RESET is pulled up externally.

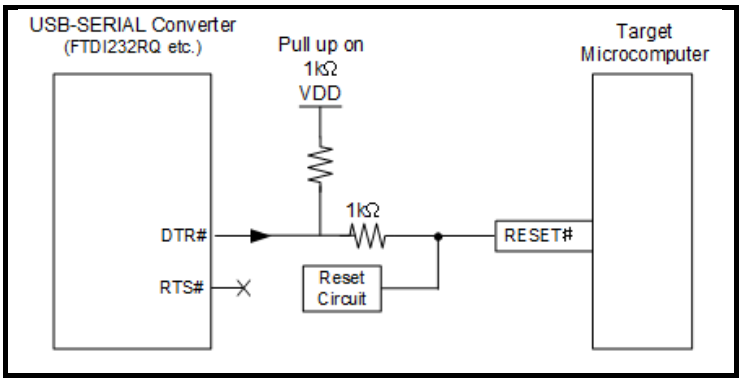


Figure 2.2 RESET# pin connection example (DTR connection)

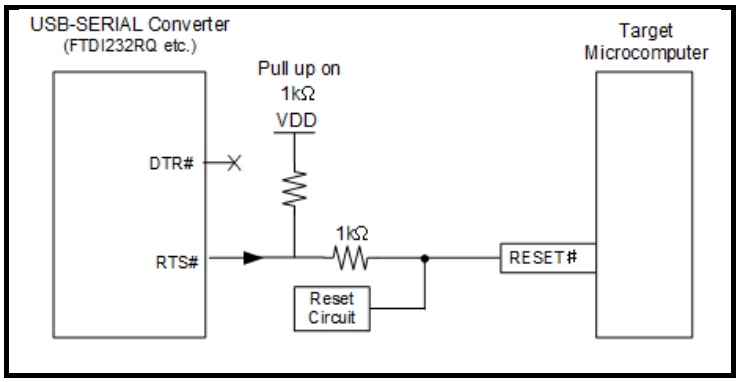
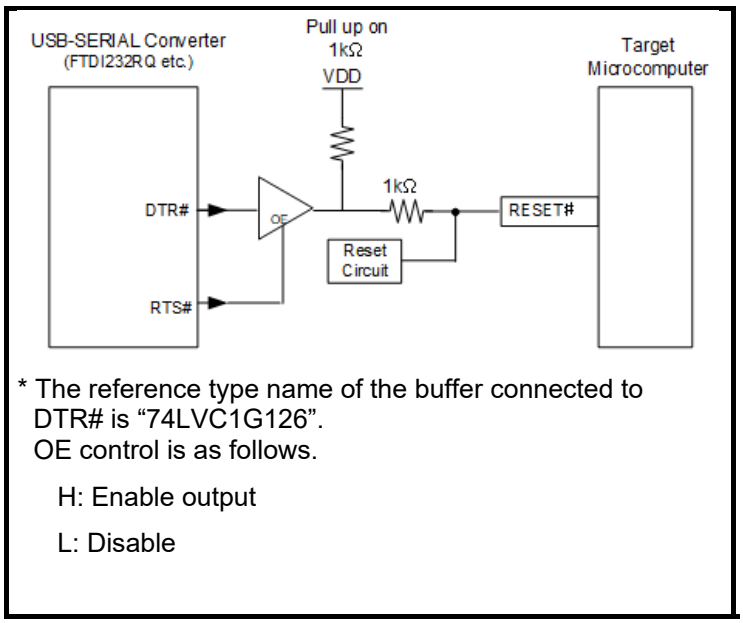


Figure 2.3 RESET# pin connection example (RTS connection)



* The reference type name of the buffer connected to DTR# is “74LVC1G126”.
OE control is as follows.

H: Enable output

L: Disable

Figure 2.4 RESET# pin connection example (with buffer)

The following notes pertain to Figure 2.2, Figure 2.3, and Figure 2.4.

- Do not insert capacitors, additional series resistors, filters, etc. into the signal line as it may prevent normal communication.
- The circuit and resistance values listed are recommended values and are not guaranteed.
  Determine the circuit design and resistance values by taking into account the specifications and noise of the target device.
- The reset circuit should be Nch.O.D output, and the pull-up in the reset circuit should be at least 10 kΩ or no pull-up.

2.2.2 TOOL0/TOOLTxD/TOOLRxD pins

The TOOL0/TOOLTxD/TOOLRxD pins are occupied and used when COM Port debugging is used. Pin functions that are multiplexed to these pins cannot be used.

In addition, COM Port debugging uses the TOOLTxD/TOOLRxD pins to send and receive data to and from the target device. Since mode entry is done using TOOL0, TOOL0 must be connected via the TOOLRxD pin and resistor. No pull-up resistor is necessary.

When using COM Port debugging, do not set the port mode register dualed to TOOLRxD (PM1.1 when TOOLRxD is a dual-use pin for P11) to 0 (output). If you do, the TOOLRxD pin, which is used in COM Port debugging, will be in output mode and you will not be able to debug.

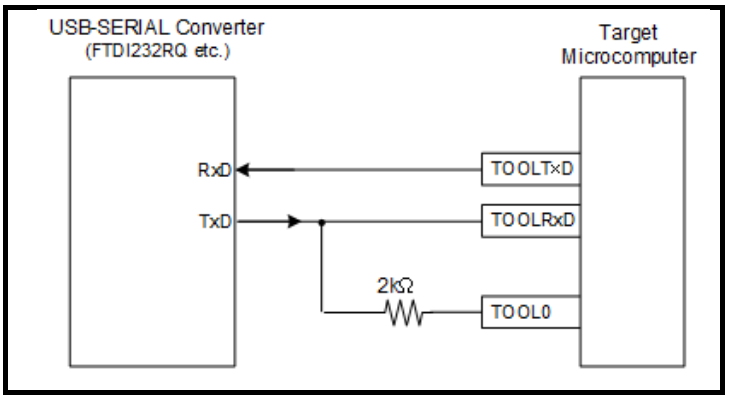


Figure 2.5 TOOL0/TOOLTxD/TOOLRxD pin connection example

Do not insert capacitors, additional series resistors, filters, etc. into the signal line as it may prevent normal communication.

2.2.3 GND

The GND of the USB-to-serial conversion adapter must be the same GND as the VSS pin of the MCU.

2.2.4 Power supply voltage

Use COM Port debugging at 1.8 V or higher, and stay within the operating voltage range of the USB-to-serial conversion adapter. For debugging below 1.8 V, use E2.

2.2.5 Caution about RESET Port and general-purpose port are dual-use pin

If RESET port and general−purpose port are dual-use pin, RESET port must be selected. If select general purpose port, COM Port debugging cannot be used.

3. Usage notes

3.1 Power on/off

Follow the steps below to power the user system on and off.

3.1.1 Power supply sequence

[At the start of use]

(1) Confirm power is off
Make sure the user system is powered off.

(2) Connect user system
Connect the user system to the host machine.

(3) Start debugger or flash writing software
Start the debugger or flash writing software.
*If you are using VBUS supply as the system power supply, go to step (5).

(4) Power up user system
Turn on the power to the user system.

(5) Connect to user system from debugger or flash writing software through COM Port
The connection method varies depending on the software.

[At the end of use]

(1) Disconnect user system from debugger or flash writing software through COM Port
The disconnection method varies depending on the software.
*If you are using VBUS supply as the system power supply, go to step (3).

(2) Power off user system
Turn off the power to the user system.

(3) Exit debugger or flash writing software
Exit the debugger or flash writing software.

(4) Detach user system
Detach the user system from the host machine.

3.2 MCU resources that are used

Figure 3.1 shows the areas occupied during COM Port debugging.

These areas (shaded sections) show the space used for debugging. Do not make changes to these areas, for example by placing user programs or data there. If you make changes there, COM Port debugging will lose control over execution.

However, when “Disable flash rewriting” is selected in the debugger properties, the internal ROM area shown in Figure 3.1 is not used (only the internal RAM area is used).

For details about selecting “Disable flash rewriting” in the debugger properties, refer to 3.2.3

Setting the on-chip debug option byte area and 3.4.2

Voltage at which flash memory cannot be written and operation in flash operation mode.

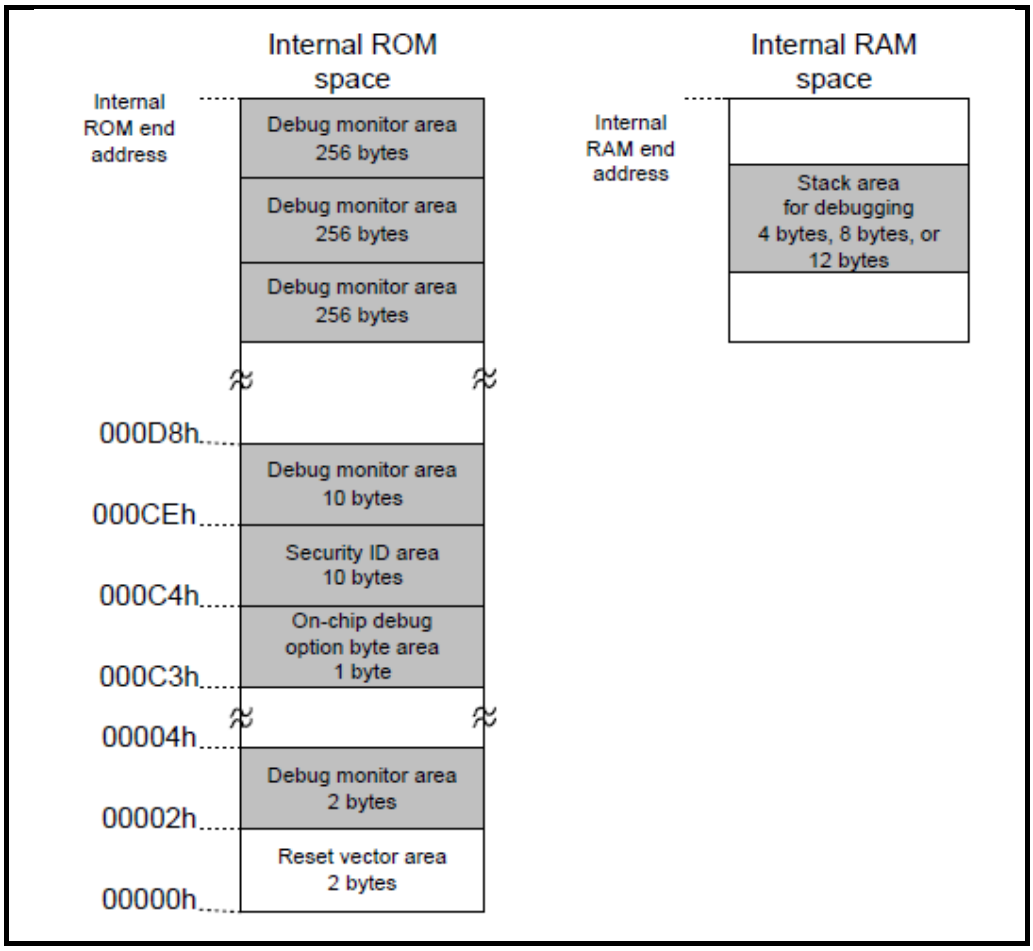


Figure 3.1 MCU resources that are used

*1 The reset vector area is used by the COM Port debugging program during COM Port debugging. If you make changes there, COM Port debugging will lose control over execution.

3.2.1 Securing the debug monitor area

The debug monitor area contains the debug monitor program.

The monitor program performs initialization of the debugging communication interface, run/break processing of the CPU, etc. This area must not be overwritten by the user program.

You must ensure that no user programs or data are placed in the 23-byte area near the on-chip debug option byte area and the area up to 768 bytes before the internal ROM end address. *1
The reset vector is changed to the address where the monitor program is placed.

[How to secure areas]

If you are not using this area in the user program, you do not necessarily need to secure it. However, in order to avoid trouble when starting the debugger, it is recommended to secure this area in advance, for example by using the build tool.

*1 The area will vary depending on whether you are using the pseudo RRM/DMM function and/or the Start/Stop function.

Example (internal ROM 256KB device)

Using neither the pseudo RRM/DMM function nor the Start/Stop function

Monitor program placed at 0x3FF00 to 0x3FFFF (256 bytes)

Using either the pseudo RRM/DMM function or the Start/Stop function

Monitor program placed at 0x3FE00 to 0x3FFFF (512 bytes)

Using both the pseudo RRM/DMM function and the Start/Stop function

Monitor program placed at 0x3FD00 to 0x3FFFF (768 bytes)

3.2.2 Securing the stack area for debugging

Use 4 bytes as the stack area for debugging. Since this area is located directly under the stack area, the address of the debug stack area varies as the stack increases or decreases. In other words, it consumes 4 bytes extra for the stack area consumed by the user program. Be careful not to let the debug stack area extend beyond the internal RAM. *1

When using the Start/Stop function, the debug stack area is 8 bytes.

Figure 3.2 shows an example of the stack area increasing when the internal RAM start address is 0xFCF00.

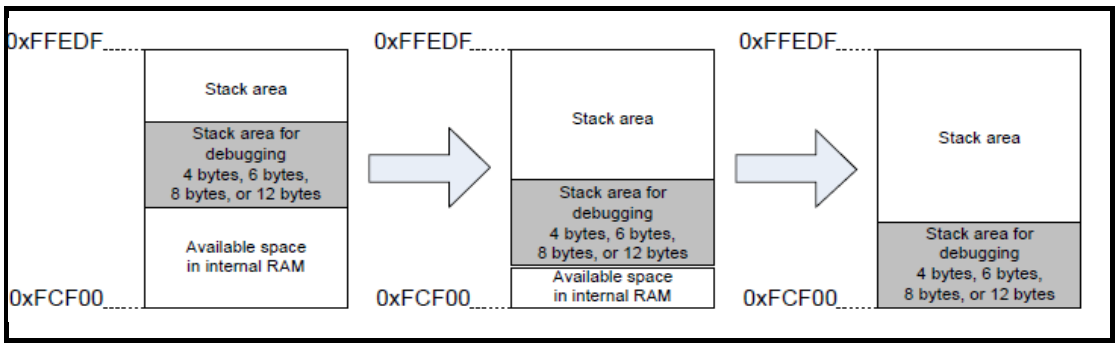


Figure 3.2 Overview of address variation in the debugging stack area

*1 For self-programming, use 12 bytes as the stack area for debugging.
Refer to the self-programming manual for how to secure the stack for self-programming.
The Start/Stop function cannot be used during self-programming.

3.2.3 Setting the on-chip debug option byte area

This is a security setting that prevents a third party from reading the contents of the flash memory.

Refer to the user's manual for your specific device to see the values that can be set.

If connecting the on-chip debugging emulator, it is necessary to set the security ID.

[Setting procedures] *1

There are two ways to set the on-chip debug option byte area.

Make sure that you perform the setting in one of these ways.

(a) Embed the on-chip debug option byte at address 0xC3 of the user program
(b) Set it with the build tool
For details on this setting method, refer to the build tool's user's manual.

*1 When the value of the on-chip debug option byte is set to disable on-chip debugging (OCDENSET = 0) on the device, if you select “Disable flash rewriting” in the debugger properties, you will not be able to start the debugger. If you select “Allow flash rewriting”, you will be able to start the debugger, but the flash memory will be erased when the debugger starts.
Also, note that it may not be possible to start the debugger depending on the LVD setting of option byte 0xC1.

3.2.4 Setting the security ID

This setting prevents the contents of memory from being read by a third party through the debug interface. The security ID is embedded in the internal flash memory at address 0xC4 to 0xCD. *1

The debugger starts only when the security ID set by the debugger matches the memory contents at address 0xC4 to 0xCD. If there is not a match, instead of starting, the behavior of the debugger depends on the settings of the on-chip debugging option byte area (see the user's manual hardware for the specific device).

If you forget the security ID, erase the flash memory and set the security ID again.

There are two ways to set the security ID for the internal flash memory.
If (a) and (b) are set at the same time, the setting of (b) takes precedence.

(a) Embed the security ID at address 0xC4 to 0xCD in the user program
You can embed the security ID at address 0xC4 to 0xCD in the user program.
If you embed the security ID shown in Table 3-1, the security ID set in the debugger would be “0123456789ABCDEF1234” (the letters can be either uppercase or lowercase). *2 *3

(b) Set it with the build tool
For details on this setting method, refer to the build tool's user's manual.

Table 3-1 Security ID setting example

Address	Setting
0x000C4	0x01
0x000C5	0x23
0x000C6	0x45
0x000C7	0x67
0x000C8	0x89
0x000C9	0xAB
0x000CA	0xCD
0x000CB	0xEF
0x000CC	0x12
0x000CD	0x34

*1 It's possible that address and length of security ID is different on every device, so refer to the user's manual for the device.
*2 If you want to connect the debugger to a device that has a security ID, you must enter the security ID in the debugger. For more information about authentication, check the user's manual for the debugger that you are using.
*3 The setting value “0xFFFFFFFFFFFFFFFFFFFF” (all bytes “0xFF”) is prohibited.

3.3 Reset

3.3.1 Behavior after reset

After an external pin reset or an internal reset, the monitor program performs debugging initialization processing. Therefore, the time from the reset occurrence to the execution of the user program differs from the actual device operation. It takes longer to execute the user program when flash rewriting is allowed in the debugging tool than when flash rewriting is disabled (max 100 ms).

3.3.2 Registers after reset

During COM Port debugging, the SP value after reset is as follows.

- If the device has at least 768 bytes of internal RAM: FC00h
- The device has less than 768 bytes of internal RAM: Internal RAM start address+0x20
  Example: RAM start address 0xFEF00: 0xEF20

3.3.3 Pin reset during break

During COM Port debugging, if you perform a pin reset during a break, the debugger becomes unresponsive. Do not perform a pin reset during a break.

3.4 Flash memory

3.4.1 Debugging notes for self-programming

(1) Non-rewritable areas in self-programming
If you rewrite the area that contains the debug monitor by flash self-programming, COM Port debugging will lose control over execution. This also applies when boot swap is performed.

(2) Break during self-programming
If you want to set a break other than forced break during self-programming, enable the debugger property “Self-programming”.
Forced breaks during self-programming can cause COM Port debugging to lose control over execution.

(3) Pseudo RRM/DMM function during self-programming
The pseudo RRM/DMM function is not available while the user program is running the flash self-programming library or data flash library.

(4) Watch panel display during self-programming
While the user program is running the flash self-programming library or data flash library, memory cannot be accessed by the RAM monitor function. Therefore, the value of a variable registered in the watch panel is displayed as “?”.

3.4.2 Voltage at which flash memory cannot be written and operation in flash operation mode

If the following debugger operations, which involve rewriting the flash memory, are executed at a voltage at which flash memory cannot be written or erased, or when flash rewriting is disabled in the debugger properties, the debugger raises an error and the operation is ignored.

- Write to internal flash memory
- Set/cancel software breakpoint
- Start execution from the location where the software breakpoint is set
- Step execution at the location where the software breakpoint is set
- Step-over execution, return-out execution
- Execute to cursor position
- Set/change/cancel hardware break
- Switch internal reset masking
- Switch peripheral break

In addition, the operating frequency range and operating voltage range are set in the flash operation mode. If the operating frequency range or operating voltage range is outside the range, normal operation may fail.

3.5 GDIDIS function

The Global Digital Input Disable Register (GDIDIS) prevents through current in the input buffer when EVDD = 0 V. During COM Port debugging, do not set GDIDIS=1 (input buffer input prohibited) because communication (TOOLTxD/TOOLRxD pins) will not be possible.

3.6 MCUs that are used in debugging

3.6.1 Use in mass-production products

The MCUs that are used in COM Port debugging are placed under stress by repeated programming of flash memory during emulation. MCUs that are used in debugging should not be used in mass-production for end users.

Furthermore, since a program for COM Port debugging is written to the MCU during debugging, do not save the contents of the flash memory of the MCU used in debugging for use as product ROM data.

3.6.2 Standalone operation check

After downloading the load module file to the device for on-chip debugging by COM Port debugging, do not disconnect the USB interface cable to check the operation of the device alone. After debugging, the device contains a processing program for on-chip debugging, which is different from the program used in actual operation.

3.7 Final evaluation of user program

Before entering the mass production phase, be sure to perform a final evaluation of the program written to a flash ROM by the Renesas Flash Programmer flash writing software or other flash programmer on a standalone basis without using COM Port debugging.

3.8 Debugging functions

Stepped execution function

(1) SFR values during stepped execution
The value of some SFRs (special function registers) might remain unchanged when stepping into code. If the value of the SFRs does not change while stepping in, operate the microcontroller by executing the instructions continuously instead of executing them in steps.

Step into (stepped execution): Instructions in the user-created program are executed one by one.

Continuous execution: The user-created program is executed from the current PC.

(2) Division operation step into (for a device equipped with a multiplier and divider/multiply-accumulator)
When the instruction which sets bit 0 (DIVST) of the multiplication/division control register (MDUC) to 1 is stepped, the division operation does not finish. However, there is no problem with the stepped execution through the division operation at the C source level.

(3) Illegal memory access detection function
When the function to detect illegal memory accesses is enabled (IAWCTL.7 = 1), no internal reset will occur if you have done the following:

- Performed stepped execution for an instruction that will generate an illegal memory access.
- Set a software breakpoint at an instruction that will generate an illegal memory access and then executed a program from that point.

(4) RAM guard function
When the RAM guard function is enabled (except when IAWCTL.5,4 = 0,0), the guarded RAM can be overwritten when stepped execution to overwrite the guarded RAM area is performed.

The RAM is also rewritten if the guarded RAM area is rewritten from the memory panel during a break.

3.8.1 “Go to Here” function

If “Go to Here” is selected, the software breakpoints and event breakpoints that have been set so far will be temporarily invalidated.

3.8.2 Debugging in standby mode

A break is a CPU interrupt. Therefore, if a break is generated by the following debugging functions, standby mode is canceled.

- Forced break
- Stepped execution of the standby instruction (stops the user program after execution of the instruction)
- Short break generated by the pseudo RRM function
- Short break generated by the pseudo DMM function
- Short break generated by setting a breakpoint while executing the user program

3.8.3 Pseudo RRM/DMM functions

Note the following points when using the pseudo RRM function or the pseudo DMM function.

- Standby mode (HALT or STOP) may be canceled during monitoring.
- The pseudo RRM/DMM function does not work while the CPU operating clock is stopped.
- When there is large number of monitoring points, the responsiveness of the debugger slows.
- Monitoring by using a watch panel instead of a memory panel reduces the impact on debugger responsiveness.
- The pseudo RRM/DMM function does not work while running on the sub-clock.
- Even when the RAM guard function is enabled, memory contents can be rewritten by the pseudo DMM function.

3.8.4 Start/Stop functions

Note the following points if you intend to use this functionality.

- Even if the start/stop function writes new values to the CPU registers, the states of the registers are restored when the function ends.
- Stepped execution of the start/stop functions is not possible. However, the functions work when RUN is performed internally, such as when the CALL instruction is stepped over.
- Breaks cannot be used in Start/Stop functions.
- When execution of a user program starts from an address where a software breakpoint has been set, the instruction at the breakpoint is executed before the start function is run.
  The order of execution is (a), (b), and (c) below.

  (a) Stepped execution (due to the break) of the instruction where the breakpoint is set

  (b) Running the start function

  (c) Executing instructions of the user program following the address where the breakpoint is set (continuous execution)
- When the start function is enabled and the user program is executed from an address at which an event break condition is set, the condition can only be satisfied after the start function has been executed. Step-execute the instruction to start the user program from the next address after that at which an event break condition is set, or disable the start function if you do not require it.
- If you intend to use a stop function specified by a Start/Stop function, specify a subroutine which returns normally. If the specified subroutine does not return normally, the emulator debugger will lose control over execution. To restore control, issue a reset of processing from the debugger.

3.8.5 Emulation of flash memory CRC accumulator function

(1) High-speed CRC (code flash: all areas)
In on-chip debugging, the CRC calculation result may differ from the actual result because the monitor program is in place and the reset vector is rewritten. Confirm the operation of the high-speed CRC on the device alone without the debugger.

(2) General-purpose CRC (code flash: specified area)
In on-chip debugging, the CRC calculation result may differ from the actual result when one of the following areas is specified as the CRC calculation target area because the monitor program is in place and the reset vector is rewritten.

- Reset vector area
- Debug monitor area
- On-chip debug option byte area
- Areas where software breaks are specified

3.8.6 Notes on the break function

When “Monitor Clock” is set to “System” in the property settings of the debugger, if a break occurs while running on the sub clock, the COM Port debugging program switches the system clock to the main clock. When this happens, note the following:

- If the external clock (EXCLK) is set as the main clock, when a break occurs with the external clock stopped, an error will occur because the clock cannot be switched.
- Even if an SFR related to the clock is changed during a break, this change will be set in the device just before the user program is executed. Accordingly, the status flag does not change the moment the SFR is changed.
- When an SFR register is rewritten to change the system clock to the sub-clock during a break, the COM Port debugging program switches to the sub-clock just before the user program is executed. This sometimes results in an error after user program execution due to the oscillation stabilization wait time. In this case, set “Monitor clock” to “User” before debugging.

3.8.7 Setting and deleting events during user program execution

Events can be set and deleted during user program execution. However, if a pin reset occurs after an event has been set or deleted, the contents of the event that was set or deleted during user program execution will be cleared.

3.8.8 Trace function

When flash rewriting is disabled in the debugger properties, the debug monitor area will not be allocated within 256 bytes of the end address of internal ROM, but no trace data can be acquired for this area.

3.8.9 Precautions when using flash read protection

Flash read protection has been added to the RL78. This feature allows the specified area to be set as read prohibited.

The area set to the flash read protection area can be fetched but not read. If you try to read it, the value 0xFF is read.

In the case of disassembly in the debugger in use with the emulator, the following is displayed:

CS+ ([Disassemble] panel): “?”

e^2 studio ([Disassembly] view): “brk1”

Set flash read protection after debugging the area where you plan to set flash read protection.

Notes on debugging when flash read protection has been set:

- Debugging is not possible in the area set for flash read protection.
- Do not attempt writing to or initializing the flash read protection area from the memory panel. Changed to Disassembly for consistency Doing so will erase all blocks in the area.
- To prevent accidental erasure of the area, we recommend setting “Prohibit inside the range” of the flash read protection area in the flash shield window function, and disabling changes to the flash shield window settings.

The pattern length between the USB-to-serial conversion adapter and the MCU should be as short as possible (50 mm or less is recommended). Also, do not route signal lines on the board other than between the USB-to-serial conversion adapter and the MCU.

For details on pin processing when COM Port debugging is not used, refer to the user's manual (hardware edition) of the relevant MCU.

3.9 Adjust of Communication baud rate

Communication baud rate on COM Port debugging can be set in debugger, communication isn't always performed with the set value.
Communication baud rate which is used by combination of setting-value/device/USB-serial-conversion-adapter is adjusted as shown in the table below:

Table 3-2 Adjust of communication baud rate

<table>
<tr><td rowspan="2">Setting value</td><td colspan="2">RL78/G2x</td><td colspan="2">RL78/G15, G16</td></tr>
<tr><td>FTDI</td><td>Other than FTDI</td><td>FTDI</td><td>Other than FTDI</td></tr>
<tr><td>Auto</td><td>Faster baud rate will be selected by CPU clock frequency.</td><td>115200bps fixed</td><td colspan="2" rowspan="5">115200bps fixed</td></tr>
<tr><td>115200bps</td><td rowspan="4">Faster baud rate will be selected with the set value as the upper limit.</td><td rowspan="4">Faster baud rate will be selected with the set value as the upper limit. But baud rate will be 115200bps depending on CPU clock.</td></tr>
<tr><td>250Kbps</td></tr>
<tr><td>500Kbps</td></tr>
<tr><td>1Mbps</td></tr>
</table>

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.20.2021	-	First Edition issued
2.00	Jan.20.2022	15	Note was added to section 3.2.4.
		22	Section 3.9, deleted “How to improve debug performance” and added “About USB-to-serial conversion adapter”.
2.10	Jan 20.2023	9	2.2.5 Caution about RESET/PORT is dual-use pin was added
		20	3.9 was deleted because of adapting USB serial adapter other than FTDI.
		20	3.9 Adjust of Communication baud rate was added.
2.11	Jul 07.2023	22	3.9 Add RL78/G16 to table.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.

© 2023 Renesas Electronics Corporation. All rights reserved.