RL78 Family

Data Flash Library Type 04
Japanese Release

Installer name: RENESAS_RL78_FDL_T04_xVxx

16-Bit Single-Chip Microcontrollers

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).
Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; underwaterrepeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited by any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)
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.
HOW TO USE THIS MANUAL

Readers
This manual is intended for user engineers who wish to understand the functions of the RL78 microcontrollers Data Flash Library Type 04 and design and develop application systems and programs for these devices. Refer to the following list for the target MCUs.

"Self-Programming Library (Japanese Release) and Supported MCUs" (R20UT2861XJxxxx)
"RL78 Family Self RAM list of Flash Self Programming Library" (R20UT2944)

Purpose
This manual is intended to give users an understanding of the methods (described in the Organization below) for using the Data Flash Library Type 04 to rewrite the data flash memories.

Organization
The RL78 Data Flash Library Type 04 user’s manual is separated into the following parts:

- Overview
- Programming Environment
- Data Flash Library Function

How to Read This Manual
It is assumed that the readers of this manual have general knowledge of electrical engineering, logic circuits, and microcontrollers.

- To gain a general understanding of functions:
  → Read this manual in the order of the CONTENTS.
- To know details of the RL78 Microcontroller instructions:
  → Refer to CHAPTER 3 DATA FLASH LIBRARY FUNCTION.

The mark <R> shows major revised points.

Conventions
Data significance: Higher digits on the left and lower digits on the right
Active low representations: \(^{\text{xxxx}}\) (overscore over pin and signal name)

Note: Footnote for item marked with Note in the text
Caution: Information requiring particular attention
Remark: Supplementary information

Numerical representations:
- Binary \(\cdots\text{xxxx} \text{ or } \cdots\text{xxxxB}\)
- Decimal \(\cdots\text{xxxx}\)
- Hexadecimal \(\cdots\text{xxxxH} \text{ or } '0\text{x}xxxx'\)

All trademarks and registered trademarks are the property of their respective owners.
EEPROM is a trademark of Renesas Electronics Corporation.
CHAPTER 1 OVERVIEW

1. 1 Overview

The data flash library is a software library to perform operations to the data flash memory with the firmware installed on the RL78 microcontroller.

The data flash library performs rewriting and reading of the data flash memory when called from the user program.

Use this data flash library user's manual with the user's manual of the target RL78 microcontroller.

Terms The meanings of the terms used in this manual are described below.

• Data flash library
  Library for data flash memory operations with the functions provided by the RL78 microcontroller.
  It cannot perform operation to the code flash memory.
• Flash self-programming library
  Library for code flash memory operation with the functions provided by the RL78 microcontroller.
  Operation to the data flash memory cannot be done.
• EEPROM emulation library
  Library that provides functions to store data to the built-in flash memory like an EEPROM.
• Block number
  Number that shows a block of the flash memory.  It is the unit of erasure operation in the Data Flash Library Type 04.
• Internal verification
  To check if the signal level of the flash memory cell is appropriate after writing to the flash memory.  If an error occurs in internal verification, the device is determined as failed.  However, if data erasure, data writing, and internal verification are performed and completed normally after the internal verification error, the device is determined as normal.
• FDL
  Abbreviation of "Data Flash Library."
• Sequencer
  The RL78 microcontroller has a dedicated circuit for controlling the flash memory.  In this document, this circuit is called the "sequencer."
• BGO (background operation)
  State in which rewriting of the flash memory can be done while operating the user program by letting the
  sequencer to control the flash memory. For the overview and details, refer to "2.1 Hardware Environment"
  and "3.4 BGO (background operation)."

• Status check
  When the sequencer is used, the processing to check the state of the sequencer (state of control for the flash
  memory) with the program controlling the flash memory is required. In this document, the processing to check
  the state of the sequencer is called "status checking."
1.2 Calling the Data Flash Library Type 04

To perform rewriting of the data flash memory with the Data Flash Library Type 04, the initialization processing for the Data Flash Library Type 04 and the functions corresponding to the functions used need to be executed from the user program by using the C language or assembly language.

Figure 1-1 shows the state transition diagram of the Data Flash Library Type 04. Figure 1-2 shows an example of the code flash memory rewriting flow by using the Data Flash Library Type 04.

Figure 1-1. State Transition Diagram of the Data Flash Library Type 04
[Overview of the state transition diagram]
To operate the data flash memory by using the Data Flash Library Type 04, the provided functions need to be executed sequentially to perform processing. For details of functions, refer to section 3, Data Flash Library Function.

(1) uninitialized/closed
State at Power ON and Reset. To execute the flash self-programming library, EEPROM emulation library, data flash library other than Type 04, STOP command, or HALT command, execute PFDL_Close from the opened state to cause a transition to this state.

(2) opened
State in which the PFDL_Open() function has been executed from the uninitialized / closed state and the data flash library can be executed. In the period from the execution of PFDL_Close to the transition to the uninitialized / closed state, the flash self-programming library, EEPROM emulation library, data flash library other than Type 04, STOP command, or HALT command cannot be executed.
When the PFDL_Open function is executed, the data flash control register (DFLCTL) is set to the state where accessing the data flash memory is permitted (DFLEN = 1), and when the PFDL_Close function is executed, the DFLCTL is set to the access inhibit state (DFLEN = 0).

(3) busy
State in which the specified processing is being executed. The control does not return to the user program until the processing is completed.

(4) sequencer busy
State in which the specified processing is being executed with the sequencer. The PFDL_Execute function specifies the details of control to the data flash memory, and the PFDL_Handler function performs a status check. The executed function returns to the user program without waiting for the completion of sequencer operation. The code flash memory cannot be referred to while the sequencer is being used.
Figure 1-2  Example of Flow of the Data Flash Library Type 04 Operation

1. Start data flash memory control
   - PFDL_Open
   - * DFLEN = 1

2. Status check
   - PFDL_Execute (BLANKCHECK command)
   - Error
   - In control

3. Status check
   - PFDL_Execute (ERASE command)
   - Error
   - In control

4. Status check
   - PFDL_Execute (WRITE command)
   - Error
   - In control

5. Status check
   - PFDL_Execute (VERIFY command)
   - Error
   - In control

6. Status check
   - PFDL_Execute (READ command)
   - Error
   - In control

7. End data flash memory control
   - PFDL_Close
   - * DFLEN = 0

8. Status check
   - PFDL_Handler
     - Normal completion
   - Error
   - Blank check error
RL78 Family
Data Flash Library Type 04

CHAPTER 1  OVERVIEW

PFDL_Open: Initializing and starting the RAM used for the Data Flash Library Type 04
The PFDL_Open function is called to initialize the RAM used for the Data Flash Library Type 04 to enable the Data Flash Library Type 04.
Set the data flash control register (DFLCTL) to the state where accessing the data flash memory is permitted (DFLEN = 1).

PFDL_Execute: Blank checking 1 to 1024 bytes for the specified address
The PFDL_Execute function (with the PFDL_CMD_BLANKCHECK_BYTES command specified) is called to perform blank checking of 1 to 1024 bytes for the specified address (confirm that the area is writable).
The processing cannot be executed across blocks.

PFDL_Execute: Erasing the specified block (1-KB)
The PFDL_Execute function (with the PFDL_CMD_ERASE_BLOCK command specified) is called to erase the specified block (1-KB).

PFDL_Execute: Writing 1 to 1024 bytes data to the specified address
The PFDL_Execute function (with the PFDL_CMD_WRITE_BYTES command specified) is called to write 1 to 1024 bytes to the specified address. The processing cannot be executed across blocks. Writing can be performed only to an area in the blank state or an area that has been erased. It is impossible to rewrite (overwrite) an area that has been written (including areas to which 0xFF has been written).

PFDL_Execute: Internal verification of 1 to 1024 bytes for the specified address
The PFDL_Execute function (with the PFDL_CMD_IVERIFY_BYTES command specified) is called to perform internal verification of 1 to 1024 bytes for the specified address. The processing cannot be executed across blocks.
Note: Internal verification checks if the signal level of the flash memory cell is appropriate. Checking by comparing data is not performed.

PFDL_Execute: Reading 1 to 1024 bytes for the specified address
The PFDL_Execute function (with the PFDL_CMD_READ_BYTES command specified) is called to read 1 to 1024 bytes for the specified address. All the processing of reading is executed within the PFDL_Execute function. The processing cannot be executed across blocks.

PFDL_Close: Ending the Data Flash Library Type 04
The PFDL_Close function is called to end the Data Flash Library Type 04. Also set the data flash control register (DFLCTL) to the state where accessing the data flash memory is inhibited (DFLEN = 0). The PFDL_Close function is executed to end control of the data flash memory.

PFDL_Handler: Status checking
The PFDL_Handler function is called to perform status checking. Status checking must be performed until the control to the data flash memory by the sequencer is finished.
CHAPTER 2 PROGRAMMING ENVIRONMENT

This chapter describes the hardware environment and software environment required to rewrite the data flash memory using the Data Flash Library Type 04.

2. 1 Hardware Environment

The Data Flash Library Type 04 for the RL78 microcontroller uses the sequencer to execute rewrite control of the data flash memory. Because the sequencer controls the data flash memory, the user program can be operated during data flash memory control. This is called BGO (background operation).

During rewriting of the data flash memory, the data flash memory cannot be referred to. However, the code flash memory can be referred to, so interrupt processing, user program, and the Data Flash Library Type 04 can be allocated in the ROM for operation as usual. Note

Figure 2-1 shows the state during a rewrite of the data flash memory. Figure 2-2 shows an example of execution of the flash library functions to perform rewriting of the data flash memory.

FIGURE 2-1  STATE DURING REWRITE OF DATA FLASH MEMORY

The user program can operate as usual with the BGO (background operation) during data flash memory control.

Interrupts can be used as usual.

Note: Interrupts are prohibited for the R5F10266 product while using the Data Flash Library Type 04.
After an execution request of the desired processing is made to the sequencer of the RL78 microcontroller, the control is immediately returned to the user program. For the result of the control of the data flash memory, the status check function (PFDL_Handler function) must be called from the user program to check the control state of the data flash memory.
2. 1. 1 Initialization

When rewriting the data flash memory by using the Data Flash Library Type 04, make the following settings.

(1) Starting high-speed on-chip oscillator

During use of the Data Flash Library Type 04, keep the high-speed on-chip oscillator running. When the oscillator is stopped, start it before using the Data Flash Library Type 04.

(2) Setting CPU operating frequency\(^1\)

In order to calculate the timing in the Data Flash Library Type 04, set the CPU operating frequency at initialization. For the method for setting the frequency, see the description of the PFDL_Open() function.

(3) Setting flash memory programming mode\(^2\)

In order to set the flash memory programming mode for writing, either of the flash memory programming modes shown below should be specified when initializing the Data Flash Library Type 04. See the description of the PFDL_Open function for the settings of the flash memory programming modes.

- Full speed mode
- Wide voltage mode

Notes 1. The CPU operating frequency is used as a parameter for the calculation of internal timing used in the Data Flash Library Type 04. This setting does not affect the CPU operating frequency. This is not the operating frequency for the high-speed on-chip oscillator.
2. For details of the flash memory programming mode, see the target RL78 microcontroller user’s manual.

2. 1. 2 Data flash control register (DFLCTL)

The data flash control register (DFLCTL) enables or disables access to the data flash memory. When the PFDL_Open function is executed, the data flash control register (DFLCTL) is set to the state where access to the data flash memory is enabled (DFLEN = 1), and when the PFDL_Close function is executed, the DFLCTL is set to the access disabled state (DFLEN = 0).

![Figure 2-3 Format of Data Flash Control Register (DFLCTL)]

Address : F0090H  Initial value : 00H  R/W

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFLCTL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DFLEN</td>
</tr>
</tbody>
</table>

DFLEN Controlling access to the data flash memory

0  Access to the data flash memory is disabled
1  Access to the data flash memory is enabled

Note: Refer to the user’s manual of the target RL78 microcontroller for more information on the settings of the data flash control register (DFLCTL).
2. 1. 3 Blocks

The flash memory of the RL78 microcontroller is divided into 1 Kbyte blocks. In the data flash library, erasure processing is performed for the data flash memory in the units of the blocks.

For reading, writing, blank checking, or internal verification, specify the start address and execution size for execution.

Figure 2-4 shows an example of block positions and block numbers of the data flash memory.

Note The address value used when reading and writing data in the flash memory, is a relative address that starts from block 0 of the data flash memory (block 0 is assumed as address 0). Note that this is not an absolute address.

Figure 2-4  Blocks of Data Flash Memory (RL78/G12: When Data Flash Memory is 2 Kbytes)
2. 1. 4  Processing time of the Data Flash Library Type 04

This section describes the time required to process the Data Flash Library Type 04 functions (except for the Read command).

The number of clock cycles required to execute flash functions differs depending on whether the flash functions are allocated to the internal ROM area (flash memory) or they are allocated to the internal RAM area. When the functions are executed in the RAM, the processing time may increase to a maximum of double the time needed when they are executed in the ROM.

This section shows the processing time when the data flash library functions are executed in the ROM. For each segment of data flash library functions, see 3.2 Segments of Data Flash Library Functions.

(1) Data Flash Library Type 04 function processing time

The flash function processing time is the time required from when a user-created program calls a flash function until the processing ends and control returns to the user-created program.

Figure 2-5 shows an overview of the flash function processing time, and Tables 2-1 and 2-2 show the processing time.

![Figure 2-5. Overview of the Data Flash Library Type 04 Function Processing Time](image-url)
Table 2-1. Function Processing Time in Full Speed Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Open</td>
<td>862 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Erase)</td>
<td>536 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(BlankCheck)</td>
<td>484 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Write)</td>
<td>549 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(IVerify)</td>
<td>502 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Read)</td>
<td>53 / fCLK + 17 / fCLK × Bytes</td>
</tr>
<tr>
<td>PFDL_Handler</td>
<td>251 / fCLK + 14</td>
</tr>
<tr>
<td>PFDL_Close</td>
<td>823 / fCLK + 443</td>
</tr>
<tr>
<td>PFDL_GetVersionString</td>
<td>10 / fCLK</td>
</tr>
</tbody>
</table>

Remarks 1. fCLK: CPU operating frequency (For example, when using a 20 MHz clock, fCLK is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)

Table 2-2. Function Processing Time in Wide Voltage Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Open</td>
<td>862 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Erase)</td>
<td>536 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(BlankCheck)</td>
<td>484 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Write)</td>
<td>549 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(IVerify)</td>
<td>502 / fCLK</td>
</tr>
<tr>
<td>PFDL_Execute(Read)</td>
<td>53 / fCLK + 17 / fCLK × Bytes</td>
</tr>
<tr>
<td>PFDL_Handler</td>
<td>251 / fCLK + 14</td>
</tr>
<tr>
<td>PFDL_Close</td>
<td>779 / fCLK + 968</td>
</tr>
<tr>
<td>PFDL_GetVersionString</td>
<td>10 / fCLK</td>
</tr>
</tbody>
</table>

Remarks 1. fCLK: CPU operating frequency (For example, when using a 20 MHz clock, fCLK is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)
(2) Recommended interval of PFDL_Handler (status check)

The PFDL_Handler function is used to check the status except for the reading processing. However, the end of each process cannot be confirmed if the PFDL_Handler function is executed before control by the sequencer finishes. Therefore, spacing each process executed by each flash function by a specific time is useful to enhance the efficiency of status checking.

In addition, because a write process using the Write command must be triggered by status check processing every byte, the status must be checked each time 1-byte is written.

Figures 2-6 and 2-7 show overviews of recommended interval, and Tables 2-3 and 2-4 show the interval time.

When writing 3 bytes using the Data Flash Library Type 04, the sequencer writes data in a 1-byte unit. Therefore, when 1-byte is written, the PFDL_Handler function must trigger the next write. If the PFDL_Handler function is not executed while there are still bytes to be written, the next write does not start, and thus the write process does not end.

Figure 2-6. Overview of Interval for Checking Status When Using Write Command (When Writing 3 Bytes)
When a process is executed by a command other than Write, the sequencer is in the busy state until all processes end. A trigger by the PFDL_Handler function is therefore not required.

**Figure 2-7. Overview of Interval for Checking Status When Using a Command Other Than Write**

(When Erasing Flash Memory)
### Table 2-3. Recommended Interval of PFDL_Handler (Status Check) in Full Speed Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PFDL_Execute(Erase)</strong></td>
<td></td>
</tr>
<tr>
<td>When block is blank</td>
<td>$5860 / f\text{CLK} + 335$</td>
</tr>
<tr>
<td>When block is not blank</td>
<td>$12734 / f\text{CLK} + 6946$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(BlankCheck)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$33 / f\text{CLK} + 18 + (6 / f\text{CLK} + 0.31) \times \text{Bytes}$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(Write)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$61 / f\text{CLK} + 47$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(IVerify)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$13 / f\text{CLK} + 17 + (28 / f\text{CLK} + 4) \times \text{Bytes}$</td>
</tr>
</tbody>
</table>

Remarks:
1. $f\text{CLK}$: CPU operating frequency (For example, when using a 20 MHz clock, $f\text{CLK}$ is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)

### Table 2-4. Recommended Interval of PFDL_Handler (Status Check) in Wide Voltage Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PFDL_Execute(Erase)</strong></td>
<td></td>
</tr>
<tr>
<td>When block is blank</td>
<td>$5065 / f\text{CLK} + 1167$</td>
</tr>
<tr>
<td>When block is not blank</td>
<td>$11144 / f\text{CLK} + 8620$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(BlankCheck)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$30 / f\text{CLK} + 57 + (5 / f\text{CLK} + 1.084) \times \text{Bytes}$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(Write)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$57 / f\text{CLK} + 99$</td>
</tr>
<tr>
<td><strong>PFDL_Execute(IVerify)</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$14 / f\text{CLK} + 44 + (17 / f\text{CLK} + 29) \times \text{Bytes}$</td>
</tr>
</tbody>
</table>

Remarks:
1. $f\text{CLK}$: CPU clock frequency (For example, when using a 20 MHz clock, $f\text{CLK}$ is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)
(3) Total processing time

The total processing time when each command is executed in the PFDL_Execute function is the time until successful termination. This does not include the overhead time (call interval) before the user executes the PFDL_Handler function or the time until abnormal termination due to errors.

Figure 2-8 shows an overview of total processing time when each command is executed in the PFDL_Execute function, and Tables 2-5 and 2-6 show the total processing time.
### Table 2-5. Total Processing Time of the Data Flash Library Type 04 in Full Speed Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Typical (μs)</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Execute(Erase)</td>
<td>$11250 / fCLK + 5800$</td>
<td>$281561 / fCLK + 264790$</td>
</tr>
<tr>
<td>PFDL_Execute(BlankCheck)</td>
<td>$906 / fCLK + 30 + (5 / fCLK +0.26) \times $Bytes</td>
<td>$1088 / fCLK + 36 + (6 / fCLK + 0.31) \times $Bytes</td>
</tr>
<tr>
<td>PFDL_Execute(Write)</td>
<td>$487 / fCLK + 11.67 + (212 / fCLK + 39.17) \times$Bytes</td>
<td>$585 / fCLK + 14 + (714 / fCLK + 430) \times$Bytes</td>
</tr>
<tr>
<td>PFDL_Execute(IVerify)</td>
<td>$621 / fCLK + 25 + (23 / fCLK + 3.33) \times $Bytes</td>
<td>$746 / fCLK + 30 + (28 / fCLK + 4) \times $Bytes</td>
</tr>
</tbody>
</table>

Remarks 1. $fCLK$: CPU operating frequency (For example, when using a 20 MHz clock, $fCLK$ is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)

### Table 2-6. Total Processing Time of the Data Flash Library Type 04 in Full Speed Mode

<table>
<thead>
<tr>
<th>PFDL_Functions</th>
<th>Typical (μs)</th>
<th>Max. (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Execute(Erase)</td>
<td>$9925 / fCLK + 7194.17$</td>
<td>$249000 / fCLK + 299307$</td>
</tr>
<tr>
<td>PFDL_Execute(BlankCheck)</td>
<td>$903 / fCLK + 62.5 + (4 / fCLK +0.9) \times $Bytes</td>
<td>$1084 / fCLK + 75 + (5 / fCLK + 1.084) \times $Bytes</td>
</tr>
<tr>
<td>PFDL_Execute(Write)</td>
<td>$487 / fCLK + 11.67 + (208 / fCLK + 82.5) \times$Bytes</td>
<td>$585 / fCLK + 14 + (669 / fCLK + 954) \times$Bytes</td>
</tr>
<tr>
<td>PFDL_Execute(IVerify)</td>
<td>$622 / fCLK + 48.33 + (14 / fCLK + 24.17) \times$Bytes</td>
<td>$747 / fCLK + 58 + (17 / fCLK + 29) \times$Bytes</td>
</tr>
</tbody>
</table>

Remarks 1. $fCLK$: CPU operating frequency (For example, when using a 20 MHz clock, $fCLK$ is 20.)
2. Bytes: The number of bytes to be written (For example, when specifying 8 bytes, Bytes is 8.)
2.2 Software Environment

Because the Data Flash Library Type 04 for the RL78 Family program needs to be allocated to the user area, the size of the program code will be consumed in the user program area.

To run the data flash library, the CPU, stack, and data buffer are used.

The Data Flash Library Type 04 is provided for the CA78K0R compiler and for the CC-RL compiler. In the tables in the remainder of this document, the versions of the Data Flash Library Type 04 for the CA78K0R and CC-RL compilers are respectively abbreviated as CA78 and CCRL.

Table 2-7 lists the software resources required\(^{Note1,2}\), and Figures 2-9 and 2-10 show examples of arrangement in RAM.

**Table 2-7. Software Resources Used by the Data Flash Library Type 04**

<table>
<thead>
<tr>
<th>Item</th>
<th>Size (byte)</th>
<th>Restrictions on Allocation and Usage(^{Note1,2})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CA78</td>
<td>CCRL</td>
</tr>
<tr>
<td>Self-RAM(^{Note3})</td>
<td>0 to 136(^{Note3})</td>
<td>0 to 136(^{Note3})</td>
</tr>
<tr>
<td>Stack</td>
<td>46 max(^{Note4})</td>
<td>40 max(^{Note4})</td>
</tr>
<tr>
<td>Data Buffer(^{Note5})</td>
<td>1 to 1024</td>
<td>1 to 1024</td>
</tr>
<tr>
<td>Function arguments</td>
<td>0 to 8</td>
<td>0 to 8</td>
</tr>
<tr>
<td>Library size</td>
<td>ROM: 177 max</td>
<td>ROM: 168 max</td>
</tr>
</tbody>
</table>

Notes: 1. For devices not shown in the RL78 Family Self RAM list of Flash Self Programming Library(R20UT2944), contact your Renesas sales agency.

2. This table is not applicable to the R5F10266 product. Refer to 2.2.1 Software resources of the R5F10266 product.

3. An area used as the working area by the Data Flash Library Type 04 is called self-RAM in this manual and the Release Note. The self-RAM requires no user setting because it is an area that is not mapped and automatically used at execution of the data flash library (previous data is discarded). When the data flash library is not used, the self-RAM can be used as a normal RAM space.

4. This is the sum of the amount of stack (maximum of 4 bytes) necessary for calling a function and the stack used by the library (42 bytes in the case of the version for the CA78K0R compiler and 36 bytes in the case of the version for the CC-RL compiler).

5. The data buffer is a RAM area necessary for inputting data for reading and writing. The necessary size changes according to the unit of reading and writing. When reading and writing of 1 byte is performed, the required data buffer size is 1 byte.
Figure 2-9  Example 1 of Arrangement in RAM Including Self-RAM  
(RL78/G13: Product with 4Kbytes RAM and 64Kbytes ROM)

Figure 2-10  Example 2 of Arrangement in RAM without Self-RAM  
(RL78/G13: Product with 2Kbytes RAM and 32Kbytes ROM)
2. 2. 1 Software resources of the R5F10266 product

The R5F10266 product has usage inhibited areas and restrictions on stack allocation. Therefore, the settings of software resources differ from other products. Using an interrupt is also prohibited.

Table 2-8 shows a list of software resources necessary for the R5F10266 product; figure 2-11, an example of RAM area; and figures 2-12 and 2-13, examples of a stack allocation for each language in use.

Table 2-8 The Data Flash Library Type 04 Software Resources of the R5F10266 Product

<table>
<thead>
<tr>
<th>Item</th>
<th>Size(byte)</th>
<th>Restrictions on Allocation and UsageNote2, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CA78</td>
<td>CCRL</td>
</tr>
<tr>
<td>Self-RAM Note3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There is no self-RAM.</td>
</tr>
<tr>
<td>Stack Note1</td>
<td>46 max.</td>
<td>40 max.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allocated in the RAM area of FFEA2H-FFEFFH Note2.</td>
</tr>
<tr>
<td>Data Buffer Note3</td>
<td>1 to 24</td>
<td>1 to 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can be allocated to a RAM area other than the area from FFE20H to FFEEFFH.</td>
</tr>
<tr>
<td>Function arguments</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allocate in the code flash memory area.</td>
</tr>
<tr>
<td>Library size</td>
<td>ROM: 177 max.</td>
<td>ROM: 168 max.</td>
</tr>
</tbody>
</table>

Notes: 1. This is the sum of the amount of stack (maximum of 4 bytes) necessary for calling a function and the stack used by the library (42 bytes in the case of the version for the CA78K0R compiler and 36 bytes in the case of the version for the CC-RL compiler).
2. Be sure to specify the stack for use by the library within this area.
3. The data buffer is a RAM area necessary for inputting data for reading and writing. The necessary size changes according to the unit of reading and writing. When reading and writing of 1 byte is performed, the required data buffer size is 1 byte.

Figure 2-11 Example of RAM Area When Data Flash Library is Used in the R5F10266 Product
Figure 2-12 Example 1 of Stack Allocation
(for the R5F10266 product/ in the case of assembly language)

Note: When the on-chip debugging function is used, the stack space to be used by the on-chip debugging function (4 bytes) is required, so this stack must be allocated in this area. When this stack is allocated, the remaining amounts of space that can be used as the user stack while the Data Flash Library Type 04 is in use are 12 bytes with the version for the CA78K0R compiler and 18 bytes with the version for the CC-RL compiler.
Figure 2-13 Example 2 of Stack Allocation
(for the R5F10266 product/ in the case of C language)

Note: When using C language, the stack space (4 bytes) used when the main function is executed from the
start-up routine is required. When the on-chip debugging function is used, the stack space (4 bytes) used
by the on-chip debugging function is required. Thus, these stacks must be placed in this area. When both
of these parts of the stack space are in use, the remaining amounts of space that can be used as the user
stack while the Data Flash Library Type 04 is in use are 0 bytes with the version for the CA78K0R compiler
and 6 bytes with the version for the CC-RL compiler.
2. 2. 2 Self-RAM

The Data Flash Library Type 04 may use a RAM area as the working area. This area is called the "self-RAM." The data used in the self-RAM is defined within the library, so no user definition of data is required. When a data flash library function is called, the data in the self-RAM area is rewritten. The self-RAM area used for data flash programming varies depending on the microcontroller, and the user RAM may be used in some devices.

In such a device, the user needs to allocate the self-RAM area to the user RAM; be sure to allocate the self-RAM area at linkage (with the CA78K0R compiler, the area can be specified in the link directive file, and with the CC-RL compiler, the area can be specified without allocating a section). For the method of specifying the area in the link directive file, refer to the section "Defining the On-Chip RAM Area" in the release note.

2. 2. 3 Register bank

The Data Flash Library Type 04 uses the general registers, ES/CS register, SP, and PSW of the register bank selected by the user.

2. 2. 4 Stack and data buffer

The Data Flash Library Type 04 uses the sequencer to write to the data flash memory, but it uses the CPU for pre-setting and control. Therefore, to use the Data Flash Library Type 04, the stack specified by the user program is also required.

Remark A link directive is used to allocate the stack and data buffer to user-specified addresses in the case of the CA78K0R compiler. A linker option is used to allocate the section in the case of the CC-RL compiler.

• Stack
  In addition to the stack used by the user program, the stack space required for flash functions must be reserved in advance, and they must be allocated so that the RAM used by the user will not be destroyed in stack processing during the Data Flash Library Type 04 operation. The available range for stack specification is the internal RAM area excluding the self-RAM area and addresses FFE20H to FFEFFH.

• Data buffer
  The uses of the data buffer are as follows.
  - Area in which data to be written is located during writing
  - Area in which data to be obtained is located during reading
  The available range for the start address of the data buffer is the internal RAM area excluding the self-RAM area and RAM addresses FFE20H to FFEFFH, as in the stack.
2. 2. 5 Data flash library

Not all the data flash library functions are linked. Only the data flash library functions to be used are linked.

- Memory allocation of the Data Flash Library Type 04
  Segments are assigned to the functions and variables used in the Data Flash Library Type 04. Areas used in the Data Flash Library Type 04 can be specified to the specific locations.

Note For the assembly language, linking can be done only for the data flash library functions to be used by deleting unnecessary functions from the include file.

2. 2. 6 Program area

This is the area in which the Data Flash Library Type 04 and the user program using the Data Flash Library Type 04 are allocated.

In the Data Flash Library Type 04 for the RL78 microcontroller, the user program can be operated during rewriting of the data flash memory because the data flash memory is rewritten by using the sequencer (background operation).
2.3 Cautions on Programming Environment

(1) Do not execute the flash self-programming library, EEPROM emulation library, or data flash library other than Type 04 during the execution of the Data Flash Library Type 04. When using the flash self-programming library, EEPROM emulation library, or data flash library other than Type 04, be sure to execute PFDL_Close to close the data flash library.

(2) Do not execute the STOP or HALT instruction during the execution of the Data Flash Library Type 04. If the STOP or HALT instruction needs to be executed, be sure to execute the PFDL_Close function to close the data flash library.

(3) The watchdog timer does not stop during the execution of the Data Flash Library Type 04.

(4) The data flash memory cannot be read during data flash memory operation by the Data Flash Library Type 04.

(5) Do not allocate the arguments (data buffer) or stack used in the data flash library function to an address over 0xFFE20 (0xFE20).

(6) When using the data transfer controller (DTC) during the execution of the Data Flash Library Type 04, do not allocate the RAM area used by the DTC to the self-RAM or an address over 0xFFE20 (0xFE20).

(7) Do not use the RAM area (including self-RAM) used by the Data Flash Library Type 04 until data flash library is completed.

(8) Do not execute a data flash library function within interrupt processing because the Data Flash Library Type 04 does not support multiple executions of data flash library functions.

(9) When executing the Data Flash Library Type 04 on the operating system, do not execute data flash library functions from multiple tasks because the Data Flash Library Type 04 does not support multiple executions of data flash library functions.

(10) Before starting the Data Flash Library Type 04, the high-speed on-chip oscillator needs to be started because it is used when the hardware in the RL78 microcontroller rewrites the contents of flash memory.
(11) Note the following regarding the operating frequency of the CPU and the operating frequency value set with the
initialization function (PFDL_Open).
- When a frequency below 4 MHz\(^{Note1}\) is used as the operating frequency of the CPU, 1 MHz, 2 MHz, or 3 MHz
can be used (a frequency such as 1.5 MHz that is not an integer value cannot be used). Also, set an integer
value such as 1, 2, or 3 as the operating frequency value set with the initialization function.
- When 4 MHz\(^{Note1}\) or a higher frequency is used as the operating frequency of the CPU, a frequency with
decimal places can be used. However, set a rounded up integer value as the operating frequency with the
initialization function (PFDL_Open).
  (Example: For 4.5 MHz, set "5" with the initialization function.)
- The operating frequency value of the CPU set with the initialization function (PFDL_Open) is not the operating
frequency of the high-speed on-chip oscillator but the operating frequency used for execution of the user
program. When the high-speed on-chip oscillator is to be used as the clock for execution of the user program,
enter the operating frequency at which the high-speed on-chip oscillator is to run.

Note1 For the range of the operating frequency of the CPU, see the target RL78 microcontroller user's manual.

(12) The data flash control register (DFLCTL) should not be operated during the execution of the Data Flash Library
Type 04. In addition, when the Data Flash Library Type 04 is ended, the DFLCTL is set to access inhibit state by
the PFDL_Close function.
  If accessing the data flash memory is required even after the Data Flash Library Type 04 is ended, confirm the
completion of the PFDL_Close function, set the DFLCTL to the access permit state, and perform the setup
process\(^{Note2}\).

Note2 For the method of the setup, see the target RL78 microcontroller user's manual.

(13) Initialize the arguments (RAM) that are used by the data flash library function. When they are not initialized, a
RAM parity error is detected and the RL78 microcontroller might be reset.
  For a RAM parity error, refer to the user's manual of the target RL78 microcontroller.

(14) Writing to the data flash memory can be performed only to an area in the blank state or the area that has been
erased. It is impossible to rewrite (overwrite) to an area that has been written unless it has been erased. When
rewriting is performed without erasing data, the data flash memory might be damaged.

(15) The Data Flash Library Type 04 does not perform error checking of the parameters set in the arguments of data
flash library functions. Therefore, make sure to set a correct value to the parameter after checking the
specifications of the target RL78 microcontroller. If parameter checking is required to set a correct value,
perform it in the user program.

(16) Interrupts are inhibited for the R5F10266 product while the Data Flash Library Type 04 is in use.
<R> (17) The segment (PFDL_COD) of the Data Flash Library for the CC-RL compiler for the RL78 family cannot be
allocated to extend across the 64 Kbytes boundary. Be sure to allocate segments so that they do not extend
across the 64 Kbytes boundary.

<R> (18) When using an assembler of the CC-RL compiler from Renesas Electronics, the hexadecimal prefix
representation (0x..) cannot be mixed together with the suffix representation (..H). Specify the representation
method by editing the symbol definition in pfdl.inc to match the user environment.

```
pfdl.inc

; __PFDL_INC_BASE_NUMBER_SUFFIX .SET 1

When symbol "__PFDL_INC_BASE_NUMBER_SUFFIX" is not defined (initial state), the prefix representation will
be selected.

pfdl.inc

__PFDL_INC_BASE_NUMBER_SUFFIX .SET 1

When symbol "__PFDL_INC_BASE_NUMBER_SUFFIX" is defined, the suffix representation will be selected.
```
CHAPTER 3 DATA FLASH LIBRARY FUNCTION

This chapter describes the details of the data flash library functions.

3. 1 Type of Data Flash Library Functions

The Data Flash Library Type 04 consists of the following flash functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Open</td>
<td>Initialization and starting of the RAM used for the Data Flash Library Type 04</td>
</tr>
<tr>
<td>PFDL_Close</td>
<td>Ending of the Data Flash Library Type 04</td>
</tr>
<tr>
<td>PFDL_Execute</td>
<td>Execution of control of the data flash memory</td>
</tr>
<tr>
<td>PFDL_Handler</td>
<td>Checking of the control state of the data flash memory and setting of continuous execution (status check processing)</td>
</tr>
<tr>
<td>PFDL_GetVersionString</td>
<td>Acquisition of the version information of the Data Flash Library Type 04</td>
</tr>
</tbody>
</table>

3. 2 Segment (Section) of Data Flash Library Functions

The entity of a data flash library function must be allocated to the specified area.

The entity is used as a segment when memory is allocated in the CA78K0R compiler and as a section in the CC-RL compiler.

The segment (section) is configured as follows.

- **PFDL_COD**: Segment (Section) of the data flash library function.
  - This can be allocated to the ROM or RAM.

When using the data flash library for the CC-RL compiler, read "segment" as "section" in the following descriptions.
3. 3 Commands

The details of operation of the Data Flash Library Type 04 for the data flash memory can be specified with a command in the argument of the PFDL_Execute function. The commands specified in the PFDL_Execute function are shown below. For details on the execution method, refer to the section on the PFDL_Execute function.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Value</th>
<th>Command Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_CMD_READ_BYTES</td>
<td>0x00</td>
<td>Read command</td>
</tr>
<tr>
<td>PFDL_CMD_BLANKCHECK_BYTES</td>
<td>0x08</td>
<td>Blank check command</td>
</tr>
<tr>
<td>PFDL_CMD_ERASE_BLOCK</td>
<td>0x03</td>
<td>Erasure command</td>
</tr>
<tr>
<td>PFDL_CMD_WRITE_BYTES</td>
<td>0x04</td>
<td>Write command</td>
</tr>
<tr>
<td>PFDL_CMD_IVERIFY_BYTES</td>
<td>0x06</td>
<td>Internal verification command</td>
</tr>
</tbody>
</table>
3. 4 BGO (Background Operation)

The data flash library functions can be divided into functions that do not use the sequencer and functions that use the sequencer\(^\text{Note}\). For the functions that use the sequencer\(^\text{Note}\), BGO (background operation) can be performed.

The following shows examples of operation of the Data Flash Library Type 04 during BGO and the presence of sequencer control for each function.

\(^\text{Note}\) Not during the execution of the PFDL_CMD_READ_BYTES command.

---

**Figure 3-1. BGO Operation Example 1 (Write: Writing 3 Bytes)**

- **User**
  - PFDL_Execute function executed
  - PFDL_Execute function closed (Return value: PFDL_BUSY)
  - PFDL_Handler function executed
  - PFDL_Handler function closed (Return value: PFDL_BUSY)
  - PFDL_Handler function executed
  - PFDL_Handler function closed (Return value: PFDL_BUSY)
  - PFDL_Handler function executed
  - PFDL_Handler function closed (Return value: PFDL_BUSY)
  - PFDL_Handler function executed
  - PFDL_Handler function closed (Return value: PFDL_BUSY)
  - PFDL_Handler function executed
  - PFDL_Handler function closed (Return value: PFDL_OK)
  - End of processing

- **Library**
  - Sequencer in operation (Writing 1-byte)
  - Writing of 1 byte completed
  - Writing trigger
  - Sequencer in operation (Writing 1-byte)
  - Writing of 2 bytes completed
  - Writing trigger
  - Sequencer in operation (Writing 1-byte)
  - Writing of all 3 bytes completed

* Executing PFDL_Handler before the sequencer completes writing does not result in trigger processing.

For writing, trigger processing with the PFDL_Handler is required for every byte.
The control returns immediately, so other processing can be executed. The state must be checked until completion.

* Not during the execution of the PFDL_CMD_READ_BYTES command.

Table 3-3. List of Interrupt Reception and BGO of Data Flash Library Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Sequencer Control/BGO Function</th>
<th>Interrupt Reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_Open</td>
<td>No</td>
<td>Allowed</td>
</tr>
<tr>
<td>PFDL_Close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFDL_Execute</td>
<td>Yes(^*)</td>
<td></td>
</tr>
<tr>
<td>PFDL_Handler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFDL_GetVersionString</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Note Not during the execution of the PFDL_CMD_READ_BYTES command.
3.5 List of Data Types, Return Values, and Return Types

The data types are as follows.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pfdl_u08</td>
<td>unsigned char</td>
<td>1-byte (8-bit) unsigned integer</td>
</tr>
<tr>
<td>pfdl_u16</td>
<td>unsigned int</td>
<td>2-byte (16-bit) unsigned integer</td>
</tr>
<tr>
<td>pfdl_u32</td>
<td>unsigned long int</td>
<td>4-byte (32-bit) unsigned integer</td>
</tr>
</tbody>
</table>

The meaning of each return value is as follows.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFDL_OK</td>
<td>0x00</td>
<td>Normal completion</td>
</tr>
<tr>
<td>PFDL_ERR_ERASE</td>
<td>0x1A</td>
<td>Erasure error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Erasure of the target area failed.</td>
</tr>
<tr>
<td>PFDL_ERR_MARG</td>
<td>0x1B</td>
<td>Blank check error or internal verification error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The target area is not in the blank state (not writable).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- An error occurred during internal verification processing of the target area.</td>
</tr>
<tr>
<td>PFDL_ERR_WRITE</td>
<td>0x1C</td>
<td>Writing error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Writing to the target area failed.</td>
</tr>
<tr>
<td>PFDL_IDLE</td>
<td>0x30</td>
<td>Idle state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No command is executed in the PFDL_Execute function.</td>
</tr>
<tr>
<td>PFDL_BUSY</td>
<td>0xFF</td>
<td>Execution start of the PFDL_Execute function command, or in execution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The command specified in the PFDL_Execute function is in execution.</td>
</tr>
<tr>
<td>Other than above</td>
<td>Other than above</td>
<td>Other error. Check the specified command or resource allocation again.</td>
</tr>
</tbody>
</table>

The general register used to pass a return value differs between the RENESAS CA78K0R and CC-RL compilers.

The return value and the general register used in each compiler are as follows.

<table>
<thead>
<tr>
<th>Development tool</th>
<th>Return Value</th>
<th>C Language</th>
<th>Assembly Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>pfdl_status_t</td>
<td>C(General-purpose register)</td>
<td></td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>pfdl_status_t</td>
<td>A(General-purpose register)</td>
<td></td>
</tr>
</tbody>
</table>
3. 6 Description of Data Flash Library Functions

The flash functions are described in the following format.

Data flash library function name

[Overview]
Describes the function overview of this function.

[Format]

<C language>
Describes the format to call this function from a user program written in the C language.

<Assembler>
Describes the format to call this function from a user program written in the assembly language.

[Presetting]
Describes the presetting of this function.

[Function]
Describes the function details and cautions of this function.

[Register State After Call]
Describes the register state after this function is called.

[Argument]
Describes the argument of this function.

[Return Value]
Describes the return values from this function.
PFDL_Open

[Overview]
Initialization and starting of the RAM used for the data flash library

[Format]

<C language>
RENESAS CA78K0R compiler
pfdl_status_t __far PFDL_Open( __near pfdl_descriptor_t* descriptor_pstr )

<Assembler>
CALL !PFDL_Open or CALL !!PFDL_OPEN

Remark Call with "!" when the data flash library is allocated at 00000H to 0FFFFH, or call with "!!" otherwise.

[Presetting]
• The flash self-programming library, program and data flash library to operate the data flash memory, and EEPROM emulation library are not executed or have been ended.
• The high-speed on-chip oscillator is running.

[Function]
• Sets the data flash control register (DFLCTL) to the state where accessing the data flash memory is permitted (DFLEN = 1).
• Reserves, initializes and starts processing of the self-RAM used for the Data Flash Library Type 04. If a self-RAM\(^1\) exists, do not use it until the Data Flash Library Type 04 is finished.
• Defines the flash memory programming mode\(^2\) of the Data Flash Library Type 04 in the wide_voltage_mode_u08, a structure member of the argument pfdl_descriptor_t.
  0x00: Full speed mode
  0x01: Wide voltage mode
• Sets the operating frequency of the CPU in the fx_MHz_u08, a structure member of the argument pfdl_descriptor_t. The setting value is used for the calculation of timing data in the Data Flash Library Type 04\(^3\).

For the value of the operating frequency of the CPU (fx_MHz_u08), note the following.
- When a frequency below 4 MHz\(^4\) is used as the operating frequency of the CPU, 1 MHz, 2 MHz, or 3 MHz can be used (a frequency such as 1.5 MHz that is not an integer value cannot be used). Also, set an integer value such as 1, 2, or 3 as the operating frequency value set with the initialization function.
- When 4 MHz or a higher frequency\(^4\) is used as the operating frequency of the CPU, a frequency with decimal places can be used. However, set a rounded up integer value as the operating frequency set with the initialization function (PFDL_Open).
  (Example: For 4.5 MHz, set "5" with the initialization function.)
- This is not the operating frequency of the high-speed on-chip oscillator.
Notes
1. For more information on the self-RAM, refer to 2.2 Software Environment.
2. For details of the flash memory programming mode, refer to the user's manual of the target RL78 microcontroller.
3. It is a required parameter for timing calculation in the flash self-programming library. This setting does not change the operating frequency of the CPU.
4. For the range of the maximum operating frequency, refer to the user's manual of the target RL78 microcontroller.

Caution
After executing this function, execute the PFDL_Close function. Do not set the data flash control register (DFLCTL) to the state where access to the data flash memory is inhibited (DFLEN = 0) until the Data Flash Library Type 04 ends.

[Register State After Call]

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Return Value</th>
<th>Destructed Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>C(General-purpose register)</td>
<td>AX</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>A(General-purpose register)</td>
<td>X,HLC</td>
</tr>
</tbody>
</table>

[Argument]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__near pfdl_descriptor_t* descriptor_pstr</td>
<td>Initial setting value of the Data Flash Library Type 04 (The flash memory programming mode, CPU frequency)</td>
</tr>
</tbody>
</table>

Definition of Argument

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>C Language (Structure Definition)</th>
<th>Assembly Language (Example of definition)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>typedef struct { pfdl_u08 fx_MHz_u08; pfdl_u08 wide_voltage_mode_u08; } pfdl_descriptor_t;</td>
<td>__descriptor_pstr: DB _fx_MHz_u08 : DB _wide_voltage_mode_u08 :</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>typedef struct { pfdl_u08 fx_MHz_u08; pfdl_u08 wide_voltage_mode_u08; } pfdl_descriptor_t;</td>
<td>__descriptor_pstr: .DB _fx_MHz_u08 : .DB _wide_voltage_mode_u08 :</td>
</tr>
</tbody>
</table>

Contents of __near pfdl_descriptor_t

<table>
<thead>
<tr>
<th>Structure Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fx_MHz_u08</td>
<td>CPU frequency during the execution of the Data Flash Library Type 04</td>
</tr>
<tr>
<td>wide_voltage_mode_u08</td>
<td>Setting of the flash memory programming mode</td>
</tr>
</tbody>
</table>

Contents of Argument Settings

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Argument Type/Register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C Language                              Assembly Language</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>__near pfdl_descriptor_t* descriptor_pstr</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>__near pfdl_descriptor_t* descriptor_pstr</td>
</tr>
</tbody>
</table>

Notes for compilation:
- [NE78K0R compiler] __near pfdlDescriptor_t* descriptor_pstr
- [CC-RL compiler] __near pfdlDescriptor_t* descriptor_pstr

Notes for function:
- [function call] __near pfdlDescriptor_t* descriptor_pstr

Notes for declaration:
- [function declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for assembly:
- [assembly declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for code:
- [code declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for run-time:
- [run-time declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for runtime:
- [runtime declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for kick-start:
- [kick-start declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for boot-up:
- [boot-up declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for start-up:
- [start-up declaration] __near pfdlDescriptor_t* descriptor_pstr

Notes for self-RAM:
- [self-RAM declaration] __near pfdlDescriptor_t* descriptor_pstr
## [Return Value]

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00(PFDL_OK)</td>
<td>Normal completion</td>
</tr>
<tr>
<td></td>
<td>- Initial setting is complete (there is no parameter other than normal completion)</td>
</tr>
</tbody>
</table>


PFDL_Close

[Overview]
Ending of the data flash library

[Format]
<C language>

\begin{verbatim}
// RENESAS CA78K0R compiler
void __far PFDL_Close( void )

// RENESAS CC-RL compiler
void __far PFDL_Close( void )
\end{verbatim}

<Assembler>

\begin{verbatim}
CALL !PFDL_Close  or CALL !!PFDL_Close
\end{verbatim}

Remark Call with "!" when the data flash library is allocated at 00000H to 0FFFFH, or call with "!!" otherwise.

[Presetting]
Before the execution of this function, the PFDL_Open function must be completed normally.

[Function]
- Sets the data flash control register (DFLCTL) to the state where access to the data flash memory is inhibited (DFLEN = 0).
  If accessing the data flash memory is required even after the Data Flash Library Type 04 is ended, confirm the completion of the PFDL_Close function, set the DFLCTL to the access permit state, and perform the setup process \textit{Note}.
- Ends operation of the Data Flash Library Type 04.

\textbf{Note} For the method of the setup, see the target RL78 microcontroller user's manual.

[Register State After Call]

\begin{tabular}{|c|c|c|}
\hline
Development Tool & Return Value & Destructed Register \\
\hline
RENESAS CA78K0R compiler & - & - \\
RENESAS CC-RL compiler & - & C \\
\hline
\end{tabular}

[Argument]
None

[Return Value]
None
[Overview]
Execution of control of the data flash memory

[Format]
<C language>
RENESAS CA78K0R compiler
\[\text{pfdl\_status\_t } \text{__far PFDL\_Execute(__near pfdl\_request\_t* request\_pstr)}\]

<Compiler>
RENESAS CC-RL compiler
\[\text{pfdl\_status\_t } \text{__far PFDL\_Execute(__near pfdl\_request\_t* request\_pstr)}\]

<Assembler>
\[\text{CALL } \text{!PFDL\_Execute or CALL } \text{!!PFDL\_Execute}\]

Remark Call with "!" when the data flash library is allocated at 00000H to 0FFFFH, or call with "!!" otherwise.

[Presetting]
Before the execution of this function, the PFDL\_Open function must be completed normally.

[Function]
Executes the control of the data flash memory according to the specified command.

[Register State After Call]

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Return Value</th>
<th>Destructed Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>C(General-purpose register)</td>
<td>AX</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>A(General-purpose register)</td>
<td>X, BC, DE, HL</td>
</tr>
</tbody>
</table>

[Argument]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__near pfdl_request_t* request_pstr</td>
<td>Specify the details of control of the data flash memory (command and setting value).</td>
</tr>
</tbody>
</table>

Note Initialize the variable area by inputting a value such as "0" before using as an argument of this function for a parameter for which a value is not necessarily set. When a variable including an area that has not been initialized is used, a RAM parity error is detected and the RL78 microcontroller might be reset. For a RAM parity error, refer to the user's manual of the target RL78 microcontroller.
### Definition of __near pfdl_request_t

#### Contents of __near pfdl_request_t

<table>
<thead>
<tr>
<th>Structure Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pfdl_u16  index_u16</td>
<td>Start address of the target area or block number(^{\text{Note}1}) - Erasure: Block number(^{\text{Note}1}) - Other than erasure: Start address(^{\text{Note}1,3}) of the target area</td>
</tr>
<tr>
<td>__near pfdl_u08* data_pu08</td>
<td>Pointer to the data buffer for acquisition of data to be written or read(^{\text{Note}1,2}) Not used for processing other than writing/reading</td>
</tr>
<tr>
<td>pfdl_u16  bytecount_u16</td>
<td>Execution range of the command (byte specification)(^{\text{Note}1,2}) - Erasure: No specification is required. - Other than erasure: Range from the specified start address(^{\text{Note}1}) to the target area</td>
</tr>
<tr>
<td>pfdl_command_t command_enu</td>
<td>Command to execute</td>
</tr>
</tbody>
</table>

#### Notes

1. Initialize the variable area by inputting a value such as "0" before using as an argument of this function for a parameter for which a value is not necessarily set. When a variable including an area that has not been initialized is used, a RAM parity error is detected and the RL78 microcontroller might be reset. For a RAM parity error, refer to the user’s manual of the target RL78 microcontroller...

2. Specify it only for commands requiring the target parameter. Provide the data buffer size for the number of bytes of the data to be written or read.

3. The specified address is the relative address that starts from block 0 of data flash memory (block 0 is assumed as address 0).

### Contents of Argument Settings

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Argument Type/Register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C Language</td>
</tr>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>__near pfdl_request_t* request_pstr</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>__near pfdl_request_t* request_pstr</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## List of Execution Commands (pfdl_command_t)

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| PFDL_CMD_READ_BYTES     | 0x00  | Reads the data of the read size from the specified start address\textsuperscript{Note2} of the data flash memory to the read data input buffer. Since the reading processing is not the background operation (BGO), all processing is executed in the PFDL_Execute function and the PFDL_Handler function does not need to be executed.  
  *The following arguments must be set for execution.  
  • pfdl_request_t.index_u16: Reading start address\textsuperscript{Note2}  
  • pfdl_request_t.bytecount_u16: Read size\textsuperscript{Note1}  
  • pfdl_request_t.data_pu08: Address of the read data input buffer |
| PFDL_CMD_BLANKCHECK_BYTES | 0x08 | Confirms that the execution range area is the erased level (writable area) by referring to the specified start address\textsuperscript{Note2} of the data flash memory.  
  (Erased level cannot be confirmed by reading data.)  
  In the case of an error, writing to the target area cannot be performed.  
  When writing to an area where an error has occurred, the erasure command (PFDL_CMD_ERASE_BLOCK) must be executed.  
  *The following arguments must be set for execution.  
  • pfdl_request_t.index_u16: Start address\textsuperscript{Note2}  
  • pfdl_request_t.bytecount_u16: Execution range from the start address\textsuperscript{Note1} |
| PFDL_CMD_ERASE_BLOCK    | 0x03  | Performs erasure for the block of the specified number in the data flash memory.  
  In the case of an error, writing to the target block cannot be performed. When writing to a block where an error has occurred, re-execute this command and normally terminate the execution.  
  *The following argument must be set for execution.  
  • pfdl_request_t.index_u16: Block number |
| PFDL_CMD_WRITE_BYTES    | 0x04  | Writes the data input in the write data input buffer to the data flash memory from the specified start address\textsuperscript{Note2} for the write size.  
  Writing to the data flash memory can be performed only to an area in the blank state or an area where data has been erased. It is not possible to perform rewriting (overwriting) to an area where data has already been written before the area is erased. Execute the internal verification command (PFDL_CMD_IVERIFY_BYTES) for an area to which data has been written to confirm the state of the data flash memory.  
  *The following arguments must be set for execution.  
  • pfdl_request_t.index_u16: Write start address\textsuperscript{Note2}  
  • pfdl_request_t.bytecount_u16: Write size\textsuperscript{Note1}  
  • pfdl_request_t.data_pu0: Address of the write data input buffer |
| PFDL_CMD_IVERIFY_BYTES  | 0x06  | Performs internal verification from the specified start address\textsuperscript{Note2} of the data flash memory for the area in the execution range.  
  This internal verification is a process to check if the signal level of the flash memory cells in the specified range is appropriate. When an error occurs, the written data is not guaranteed. When performing rewriting to the target area, execute the erasure command (PFDL_CMD_ERASE_BLOCK).  
  *The following arguments must be set for execution.  
  • pfdl_request_t.index_u16: Start address\textsuperscript{Note2}  
  • pfdl_request_t.bytecount_u16: Execution range from the start address\textsuperscript{Note1} |
Notes  1  It cannot be specified across blocks. Specify it within one block.
2  The specified address is the relative address that starts from block 0 of the data flash memory (block 0 is assumed as address 0) when writing and reading the memory. Note that the specified address is not an absolute address.

<table>
<thead>
<tr>
<th>Absolute address</th>
<th>Relative address</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1FFFH</td>
<td>0FFFH</td>
</tr>
<tr>
<td>F1000H</td>
<td>0000H</td>
</tr>
</tbody>
</table>

[Return Value]

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00(PFDL_OK)</td>
<td>Normal completion</td>
</tr>
</tbody>
</table>
| 0x1A(PFDL_ERR_ERASE) | Erasure error  
- An error occurred during erasure processing.                         |
| 0x1B(PFDL_ERR_MARGIN) | Blank check error or internal verification error  
- The target area is not in the blank state (not a writable area).  
- An error occurred during internal verification processing of the target area. |
| 0x1C(PFDL_ERR_WRITE) | Writing error  
- An error occurred during write processing.                                |
| 0xFF(PFDL_BUSY)   | Execution start of the specified command  
- The execution of the specified command has been started.  
(Check the execution state with the PFDL_Handler function.) |
PFDL_Handler

[Overview]
Checking of the control state of the data flash memory and setting of continuous execution (status check processing)

[Format]
<C language>
RENESAS CA78K0R compiler
pfdl_status_t __far PFDL_Handler( void )

<Assembler>
CALL !PFDL_Handler or CALL !!PFDL_Handler

Remark Call with "!" when the data flash library is allocated at 00000H to 0FFFFH, or call with "!!" otherwise.

[Presetting]
Before the execution of this function, the PFDL_Open function must be completed normally.

[Function]
Checks the control state of the command (other than the PFDL_CMD_READ_BYTES command) specified in the PFDL_Execute function executed immediately before this function and performs required settings for continuous execution.

[Register State After Call]

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Return Value</th>
<th>Destructed Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>C(General-purpose register)</td>
<td>-</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>A(General-purpose register)</td>
<td>C</td>
</tr>
</tbody>
</table>

[Argument]
None

[Return Value]

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00(PFDL_OK)</td>
<td>Normal completion</td>
</tr>
<tr>
<td>0x1A(PFDL_ERR_ERASE)</td>
<td>Erasure error</td>
</tr>
<tr>
<td></td>
<td>- An error occurred during erasure processing.</td>
</tr>
<tr>
<td>0x1B(PFDL_ERR_MARGIN)</td>
<td>Blank check error or internal verification error</td>
</tr>
<tr>
<td></td>
<td>- The target area is not in the blank state (not a writable area).</td>
</tr>
<tr>
<td></td>
<td>- An error occurred during internal verification processing of the target area.</td>
</tr>
<tr>
<td>0x1C(PFDL_ERR_WRITE)</td>
<td>Writing error</td>
</tr>
<tr>
<td></td>
<td>- An error occurred during write processing.</td>
</tr>
<tr>
<td>0x30(PFDL_IDLE)</td>
<td>Idle state</td>
</tr>
<tr>
<td></td>
<td>- No command is executed in the PFDL_Execute function.</td>
</tr>
<tr>
<td>0xFF(PFDL_BUSY)</td>
<td>Command in execution</td>
</tr>
<tr>
<td></td>
<td>- The command specified in the PFDL_Execute function is being executed.</td>
</tr>
</tbody>
</table>
**PFDL_GetVersionString**

[Overview]
Acquisition of the version information of the Data Flash Library Type 04

[Format]

**<C language>**
RENESAS CA78K0R compiler
```c
__far pfdl_u08* __far PFDL_GetVersionString( void )
```

RENESAS CC-RL compiler
```c
__far pfdl_u08* __far PFDL_GetVersionString( void )
```

**<Assembler>**
```assembly
CALL !PFDL_GetVersionString or CALL !!PFDL_GetVersionString
```

Remark  Call with "!" when the data flash library is allocated at 00000H to 0FFFFH, or call with "!!" otherwise.

[Presetting]
None

[Function]
Obtains the version information of the Data Flash Library Type 04.

[Register State After Call]

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Return Value</th>
<th>Destructed Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENESAS CA78K0R compiler</td>
<td>BC (0 to 15), DE (16 to 31)</td>
<td>-</td>
</tr>
<tr>
<td>RENESAS CC-RL compiler</td>
<td>DE(0-15), A(16-23)</td>
<td>-</td>
</tr>
</tbody>
</table>

[Argument]
None

[Return Value]

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__far pfdl_u08*</td>
<td>The version information storage start address of the Data Flash Library Type 04. (far area)</td>
</tr>
<tr>
<td></td>
<td>The version information of the flash self-programming library consists of ASCII characters.</td>
</tr>
<tr>
<td></td>
<td>Example: Data Flash Library Type 04</td>
</tr>
<tr>
<td></td>
<td>&quot;DRL78T04LyyyzGVxxx&quot;</td>
</tr>
<tr>
<td></td>
<td>Version information: Example: V105 -&gt; V1,05</td>
</tr>
<tr>
<td></td>
<td>Compiler information(5 or 6 characters): CA78K0R [ex:RyyyG]</td>
</tr>
<tr>
<td></td>
<td>CC-RL [ex:LyyyzG]</td>
</tr>
<tr>
<td></td>
<td>Type No.(3 characters): : T04 -&gt; Type 04</td>
</tr>
<tr>
<td></td>
<td>Supported device(4 characters) : RL78</td>
</tr>
<tr>
<td></td>
<td>Target library(1 character) : FDL</td>
</tr>
</tbody>
</table>
## APPENDIX A REVIEW HISTORY

### A. 1 Major Revisions in This Edition

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughout the document</td>
<td>- The English translation was reviewed and corrected.</td>
<td>(a)</td>
</tr>
<tr>
<td></td>
<td>- The user's manual of the flash self-programming library for CC-RL was integrated into this manual.</td>
<td>(d)</td>
</tr>
<tr>
<td></td>
<td>- A statement regarding reference to the RL78 Family Self RAM list of the target MCU was added (see HOW TO USE THIS MANUAL).</td>
<td>(e)</td>
</tr>
<tr>
<td>Cover</td>
<td>- The ZIP file name was changed to the installer name.</td>
<td>(d)</td>
</tr>
<tr>
<td>Chapter 1 Overview</td>
<td>p.6 A supplementary explanation of the note on areas to which writing has already proceeded was added.</td>
<td>(c)</td>
</tr>
<tr>
<td>Chapter 2 Programming Environment</td>
<td>p.18 A description regarding the supported compilers was added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.18, p.20 In Table 2-7 and 2-8, the software resources for the CC-RL compiler were added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.18 - p.22 In Table 2-8, the stack sizes for the CC-RL compiler were added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.23 The methods for allocating the self-RAM area and specifying desired addresses for the CC-RL compiler were added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.23 The methods for allocating the stack for the CC-RL compiler were added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.25, p.26 A description regarding the start of the high-speed on-chip oscillator was added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.27 The description that each segment must not extend across a 64 Kbytes boundary was added.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.27 The method for representing hexadecimal numbers in the assembler in the RENESAS CC-RL compiler package was added.</td>
<td>(c)</td>
</tr>
<tr>
<td>Chapter 3 Data Flash Library Function</td>
<td>p.28 The heading of section 3.2 was changed (&quot;Section&quot; was added). The description regarding allocation to specified areas was reviewed and corrected.</td>
<td>(c)</td>
</tr>
<tr>
<td></td>
<td>p.32 - p.43 Statements of the return values, formats of the C language call, and definitions and contents of the arguments of each of the library functions were added and changed.</td>
<td>(c)</td>
</tr>
</tbody>
</table>

Remark: "Classification" in the above table classifies revisions as follows.

- (a): Error correction
- (b): Addition/change of specifications
- (c): Addition/change of description or note
- (d): Addition/change of package, part number, or management division
- (e): Addition/change of related documents
### A. 2 Revision History of Preceding Editions

Here is the revision history of the preceding editions. Chapter indicates the chapter of each edition.

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Description</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev.1.05</td>
<td>The English translation was reviewed and corrected.</td>
<td>Throughout the document</td>
</tr>
<tr>
<td></td>
<td>The target device descriptions were deleted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>References to the list of the target MCUs were added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The term &quot;voltage mode&quot; was changed to &quot;flash memory programming mode&quot; for consistency of terminology.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Various types of operating frequency described in the former version were unified to the CPU operating frequency.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A description of the case when flash functions are executed in the RAM was added.</td>
<td>Chapter 2 Programming Environment</td>
</tr>
<tr>
<td></td>
<td>The title of Table 2-2 was corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A description that the reading processing is excluded from the conditions of status check was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The description in (3) Total processing time was reviewed and corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The titles of Tables 2-5 and 2-6 were corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Tables 2-5 and 2-6, the title of the Reference column was changed to &quot;Typical&quot; and its contents (formulas) were changed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The resources used to run the data flash library were corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Table 2-7, the self-RAM size was changed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Table 2-7, the description of the self-RAM area was changed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note 1 on Table 2-7 was changed to a description of the inquiry about device specifications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Figure 2-12, the remaining user stack size was corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The description of the self-RAM was reviewed and corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Table 3-6, the name of the return type in the C Language column was corrected.</td>
<td>Chapter 3 Data Flash Library Function</td>
</tr>
<tr>
<td></td>
<td>The names of the members of the pfdl_descriptor_t structure were corrected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For PFDL_CMD_READ_BYTES in the List of Execution Commands (pfdl_command_t), a description that the PFDL_Handler function does not need to be executed was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A description that execution of the PFDL_Handler function is necessary for the commands other than PFDL_CMD_READ_BYTES was added.</td>
<td></td>
</tr>
</tbody>
</table>
### APPENDIX A  REVISION HISTORY

**Data Flash Library Type 04**

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Description</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rev.1.04</td>
<td>The document on the data flash library, which was classified as the application note (old version of R01AN0608), was changed to the user's manual. The corresponding ZIP file and release version were added to the cover page. The name of the flash data library was changed to the data flash library. Contents of the processing time and software resources were moved from the usage note to this document. Accordingly, the reference destination described in this document was also changed. The notation of high-speed OCO was deleted to unify the notation of high-speed on-chip oscillator. The description of the operating frequency was unified to the operating frequency of CPU since individual descriptions had different notations. Errors in the format of the assembly language for executing functions were modified. Supplemental information was added to the terms of the blank state and blank check. Controls to the data flash control register (DFLCTL) were added. Note on writing was added. Note (description) on internal verification was added. Note describing that an interrupt becomes inhibited when the R5F10266 product is in use was added. Description on the initial setting was added. Description on the data flash control register (DFLCTL) was added. Note on the address specified by the data flash library was added. Items on the processing time was added (description on the processing time was moved from the usage note to this document). The reference (referred to min in older versions) time of Erase and BlankCheck was deleted. Items of RL78/L13 were added to the resources. Items on the resource were added (description of the resource was moved from the usage note to this document). Note on the data buffer was added. The contents of resources when the R5F10266 product is in use were added. Note on the frequency of the high-speed on-chip oscillator was added. Note on the data flash control register (DFLCTL) was added. Note on the RAM parity error was added. Note for writing was added. Note describing that an interrupt becomes inhibited when the R5F10266 product is in use was added. Note on the data flash control register (DFLCTL) was added.</td>
<td>Throughout the document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chapter 1 Overview</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chapter 2 Programming Environment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev.</td>
<td>Description</td>
<td>Chapter</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Rev.1.04</td>
<td>Description on the data flash control register (DFLCTL) was added.</td>
<td>Chapter 3 Data Flash Library Function</td>
</tr>
<tr>
<td></td>
<td>Setting value of the wide voltage mode was changed from &quot;other than 00H&quot; to &quot;01H&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(values from 02H to FFH can normally be set but the description was changed since the defined value in the specification was 01H).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on the frequency of the high-speed on-chip oscillator was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on the data flash control register (DFLCTL) was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Definition of the structure and the table for setting the argument were added and modified.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description on the data flash control register (DFLCTL) was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on the data flash control register (DFLCTL) was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Notes on the RAM parity error were added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Table for the definition of the structure was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Table of the argument and register type was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on the address specified by the data flash library was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Descriptions on individual commands were added</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note for writing was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note (description) on the internal verification was added.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deleted the index.</td>
<td></td>
</tr>
</tbody>
</table>
SALES OFFICES

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

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

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan
Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-9988, Fax: +1-408-434-5351
Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004
Renesas Electronics Europe Limited
Dukas Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FJ, U.K
Tel: +44-1628-651-700
Renesas Electronics Europe GmbH
Arenastrasse 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: +86-10-8235-1105, Fax: +86-10-8235-7679
Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Pudong District, Shanghai, 200333 P. R. China
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: +852-2986-6022
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 805-02 Hynix 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: +60-3-7955-9090, Fax: +60-3-7955-9510
Renesas Electronics India Pvt. Ltd.
No. 777C, 100 Feet Road, 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67239700, Fax: +91-80-67239777
Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06285 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

© 2018 Renesas Electronics Corporation. All rights reserved.