To our customers,

---

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.
Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

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

“High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-terrorism systems; safety equipment; and medical equipment not specifically designed for life support.

“Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be 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, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

78K0/Kx2

8-Bit Single-Chip Microcontrollers

Flash Memory Programming (Programmer)


[MEMO]
NOTES FOR CMOS DEVICES

① 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 \(V_{IL} \text{(MAX)}\) and \(V_{IH} \text{(MIN)}\) due to noise, etc., 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 \(V_{IL} \text{(MAX)}\) and \(V_{IH} \text{(MIN)}\).

② HANDLING OF UNUSED INPUT PINS
Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to \(V_{DD}\) or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ PRECAUTION AGAINST ESD
A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that 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 should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ STATUS BEFORE INITIALIZATION
Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

⑤ POWER ON/OFF SEQUENCE
In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

⑥ INPUT OF SIGNAL DURING POWER OFF STATE
Do not input signals or an I/O pull-up power supply while the device is not powered. 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. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.
The information in this document is current as of May, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific". The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02.11-1
INTRODUCTION

Target Readers
This application note is intended for users who understand the functions of the 78K0/Kx2 and who will use this product to design application systems.

Purpose
The purpose of this application note is to help users understand how to develop dedicated flash memory programmers for rewriting the internal flash memory of the 78K0/Kx2.
The sample programs and circuit diagrams shown in this document are for reference only and are not intended for use in actual design-ins.
Therefore, these sample programs must be used at the user’s own risk. Correct operation is not guaranteed if these sample programs are used.

Organization
This manual consists of the following main sections.
• Flash memory programming
• Command/data frame format
• Description of command processing
• UART communication mode
• 3-wire serial I/O communication mode (CSI)
• Flash memory programming parameter characteristics

How to Read This Manual
It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.
• To gain a general understanding of functions:
  → Read this manual in the order of the CONTENTS. The mark “<R>” shows major revised points. The revised points can be easily searched by copying an “<R>” in the PDF file and specifying it in the “Find what:” field.
  • To learn more about the 78K0/Kx2’s hardware functions:
  → See the user’s manual of each 78K0/Kx2 product.

Conventions
Data significance: Higher digits on the left and lower digits on the right
Active low representation: XXX (overscore over pin or signal name)
Note: Footnote for item marked with Note in the text
Caution: Information requiring particular attention
Remark: Supplementary information
Numeral representation: Binary..................xxxx or xxxxB
                         Decimal ..................xxx
                         Hexadecimal ...........xxxxH
Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

<table>
<thead>
<tr>
<th>Device-related documents</th>
<th>Document Name</th>
<th>Document Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/KB2 User's Manual</td>
<td>U17328E</td>
<td></td>
</tr>
<tr>
<td>78K0/KC2 User's Manual</td>
<td>U17336E</td>
<td></td>
</tr>
<tr>
<td>78K0/KD2 User's Manual</td>
<td>U17312E</td>
<td></td>
</tr>
<tr>
<td>78K0/KE2 User's Manual</td>
<td>U17260E</td>
<td></td>
</tr>
<tr>
<td>78K0/KF2 User's Manual</td>
<td>U17397E</td>
<td></td>
</tr>
<tr>
<td>78K/0 Series Instructions User's Manual</td>
<td>U12326E</td>
<td></td>
</tr>
</tbody>
</table>

Caution  The related documents listed above are subject to change without notice. Be sure to use the latest version of each document when designing.
3.9 Block Blank Check Command .............................................................................................................. 37
3.9.1 Description .................................................................................................................................... 37
3.9.2 Command frame and status frame .............................................................................................. 37

3.10 Silicon Signature Command ............................................................................................................... 38
3.10.1 Description ..................................................................................................................................... 38
3.10.2 Command frame and status frame .............................................................................................. 38
3.10.3 Silicon signature data frame ......................................................................................................... 38
3.10.4 78K0/Kx2 silicon signature list ..................................................................................................... 41

3.11 Version Get Command ....................................................................................................................... 46
3.11.1 Description ..................................................................................................................................... 46
3.11.2 Command frame and status frame .............................................................................................. 46
3.11.3 Version data frame ....................................................................................................................... 47

3.12 Checksum Command ........................................................................................................................... 47
3.12.1 Description ..................................................................................................................................... 47
3.12.2 Command frame and status frame .............................................................................................. 47
3.12.3 Checksum data frame .................................................................................................................. 48

3.13 Security Set Command ....................................................................................................................... 48
3.13.1 Description ..................................................................................................................................... 48
3.13.2 Command frame and status frame .............................................................................................. 48
3.13.3 Data frame and status frame ....................................................................................................... 49
3.13.4 Internal verify check and status frame .......................................................................................... 49

CHAPTER 4 UART COMMUNICATION MODE......................................................................................... 51

4.1 Command Frame Transmission Processing Flowchart ......................................................................... 52
4.2 Data Frame Transmission Processing Flowchart ................................................................................. 53
4.3 Data Frame Reception Processing Flowchart .................................................................................... 54
4.4 Reset Command .................................................................................................................................... 55
4.4.1 Processing sequence chart ........................................................................................................... 55
4.4.2 Description of processing sequence .............................................................................................. 56
4.4.3 Status at processing completion .................................................................................................... 56
4.4.4 Flowchart ....................................................................................................................................... 57
4.4.5 Sample program ............................................................................................................................ 58

4.5 Oscillating Frequency Set Command ................................................................................................. 59
4.5.1 Processing sequence chart ........................................................................................................... 59
4.5.2 Description of processing sequence .............................................................................................. 60
4.5.3 Status at processing completion .................................................................................................... 60
4.5.4 Flowchart ....................................................................................................................................... 61
4.5.5 Sample program ............................................................................................................................ 62

4.6 Chip Erase Command ........................................................................................................................... 63
4.6.1 Processing sequence chart ........................................................................................................... 63
4.6.2 Description of processing sequence .............................................................................................. 64
4.6.3 Status at processing completion .................................................................................................... 64
4.6.4 Flowchart ....................................................................................................................................... 65
4.6.5 Sample program ............................................................................................................................ 66

4.7 Block Erase Command ......................................................................................................................... 67
4.7.1 Processing sequence chart ........................................................................................................... 67
4.7.2 Description of processing sequence .............................................................................................. 68
4.8 Programming Command ................................................................. 71
  4.8.1 Processing sequence chart ......................................................... 71
  4.8.2 Description of processing sequence ........................................... 72
  4.8.3 Status at processing completion ................................................ 73
  4.8.4 Flowchart ................................................................................. 74
  4.8.5 Sample program ....................................................................... 75

4.9 Verify Command ........................................................................... 77
  4.9.1 Processing sequence chart ......................................................... 77
  4.9.2 Description of processing sequence ........................................... 78
  4.9.3 Status at processing completion ................................................ 78
  4.9.4 Flowchart ................................................................................. 79
  4.9.5 Sample program ....................................................................... 80

4.10 Block Blank Check Command ......................................................... 82
  4.10.1 Processing sequence chart ......................................................... 82
  4.10.2 Description of processing sequence ........................................... 83
  4.10.3 Status at processing completion ................................................ 83
  4.10.4 Flowchart ................................................................................. 84
  4.10.5 Sample program ....................................................................... 85

4.11 Silicon Signature Command ............................................................ 86
  4.11.1 Processing sequence chart ......................................................... 86
  4.11.2 Description of processing sequence ........................................... 87
  4.11.3 Status at processing completion ................................................ 87
  4.11.4 Flowchart ................................................................................. 88
  4.11.5 Sample program ....................................................................... 89

4.12 Version Get Command ................................................................... 90
  4.12.1 Processing sequence chart ......................................................... 90
  4.12.2 Description of processing sequence ........................................... 91
  4.12.3 Status at processing completion ................................................ 91
  4.12.4 Flowchart ................................................................................. 92
  4.12.5 Sample program ....................................................................... 93

4.13 Checksum Command ..................................................................... 94
  4.13.1 Processing sequence chart ......................................................... 94
  4.13.2 Description of processing sequence ........................................... 95
  4.13.3 Status at processing completion ................................................ 95
  4.13.4 Flowchart ................................................................................. 96
  4.13.5 Sample program ....................................................................... 97

4.14 Security Set Command ................................................................... 98
  4.14.1 Processing sequence chart ......................................................... 98
  4.14.2 Description of processing sequence ........................................... 99
  4.14.3 Status at processing completion ................................................ 99
  4.14.4 Flowchart ................................................................................. 100
  4.14.5 Sample program ....................................................................... 101
CHAPTER  5    3-WIRE SERIAL  I/O COMMUNICATION MODE (CSI) ............................................. 103

5.1 Command Frame Transmission Processing Flowchart............................................... 104
5.2 Data Frame Transmission Processing Flowchart ....................................................... 105
5.3 Data Frame Reception Processing Flowchart............................................................... 106
5.4 Status Command............................................................................................................. 107
  5.4.1 Processing sequence chart........................................................................................ 107
  5.4.2 Description of processing sequence ........................................................................ 108
  5.4.3 Status at processing completion .............................................................................. 108
  5.4.4 Flowchart ................................................................................................................ 109
  5.4.5 Sample program ...................................................................................................... 110
5.5 Reset Command............................................................................................................. 112
  5.5.1 Processing sequence chart........................................................................................ 112
  5.5.2 Description of processing sequence ........................................................................ 113
  5.5.3 Status at processing completion .............................................................................. 113
  5.5.4 Flowchart ................................................................................................................ 114
  5.5.5 Sample program ...................................................................................................... 115
5.6 Oscillating Frequency Set Command ........................................................................ 116
  5.6.1 Processing sequence chart........................................................................................ 116
  5.6.2 Description of processing sequence ........................................................................ 117
  5.6.3 Status at processing completion .............................................................................. 117
  5.6.4 Flowchart ................................................................................................................ 118
  5.6.5 Sample program ...................................................................................................... 119
5.7 Chip Erase Command.................................................................................................. 120
  5.7.1 Processing sequence chart........................................................................................ 120
  5.7.2 Description of processing sequence ........................................................................ 121
  5.7.3 Status at processing completion .............................................................................. 121
  5.7.4 Flowchart ................................................................................................................ 122
  5.7.5 Sample program ...................................................................................................... 123
5.8 Block Erase Command............................................................................................... 124
  5.8.1 Processing sequence chart........................................................................................ 124
  5.8.2 Description of processing sequence ........................................................................ 125
  5.8.3 Status at processing completion .............................................................................. 125
  5.8.4 Flowchart ................................................................................................................ 126
  5.8.5 Sample program ...................................................................................................... 127
5.9 Programming Command.............................................................................................. 128
  5.9.1 Processing sequence chart........................................................................................ 128
  5.9.2 Description of processing sequence ........................................................................ 129
  5.9.3 Status at processing completion .............................................................................. 130
  5.9.4 Flowchart ................................................................................................................ 131
  5.9.5 Sample program ...................................................................................................... 132
5.10 Verify Command........................................................................................................ 134
  5.10.1 Processing sequence chart........................................................................................ 134
  5.10.2 Description of processing sequence ........................................................................ 135
  5.10.3 Status at processing completion .............................................................................. 135
  5.10.4 Flowchart ................................................................................................................ 136
  5.10.5 Sample program ...................................................................................................... 137
5.11 Block Blank Check Command ........................................................................................................ 139
  5.11.1 Processing sequence chart ........................................................................................................ 139
  5.11.2 Description of processing sequence .......................................................................................... 140
  5.11.3 Status at processing completion ............................................................................................... 140
  5.11.4 Flowchart .................................................................................................................................. 141
  5.11.5 Sample program ...................................................................................................................... 142

5.12 Silicon Signature Command ........................................................................................................... 143
  5.12.1 Processing sequence chart ........................................................................................................ 143
  5.12.2 Description of processing sequence .......................................................................................... 144
  5.12.3 Status at processing completion ............................................................................................... 144
  5.12.4 Flowchart .................................................................................................................................. 145
  5.12.5 Sample program ...................................................................................................................... 146

5.13 Version Get Command .................................................................................................................. 147
  5.13.1 Processing sequence chart ........................................................................................................ 147
  5.13.2 Description of processing sequence .......................................................................................... 148
  5.13.3 Status at processing completion ............................................................................................... 148
  5.13.4 Flowchart .................................................................................................................................. 149
  5.13.5 Sample program ...................................................................................................................... 150

5.14 Checksum Command ..................................................................................................................... 151
  5.14.1 Processing sequence chart ........................................................................................................ 151
  5.14.2 Description of processing sequence .......................................................................................... 152
  5.14.3 Status at processing completion ............................................................................................... 152
  5.14.4 Flowchart .................................................................................................................................. 153
  5.14.5 Sample program ...................................................................................................................... 154

5.15 Security Set Command .................................................................................................................. 156
  5.15.1 Processing sequence chart ........................................................................................................ 156
  5.15.2 Description of processing sequence .......................................................................................... 157
  5.15.3 Status at processing completion ............................................................................................... 157
  5.15.4 Flowchart .................................................................................................................................. 158
  5.15.5 Sample program ...................................................................................................................... 159

CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS ................................. 161

6.1 Flash Memory Programming Parameter Characteristics of Expanded Specification Products (μPD78F05xxA) .................................................................................................................... 161
  6.1.1 Basic characteristics .................................................................................................................... 161
  6.1.2 Flash memory programming mode setting time ......................................................................... 161
  6.1.3 Programming characteristics ..................................................................................................... 162

6.2 Flash Memory Programming Parameter Characteristics of Conventional-specification Products (μPD78F05xx) ........................................................................................................................... 164
  6.2.1 Basic characteristics .................................................................................................................... 164
  6.2.2 Flash memory programming mode setting time ......................................................................... 164
  6.2.3 Programming characteristics ..................................................................................................... 165

6.3 Simultaneous Selection and Erasure Performed by Block Erase Command .................................. 167

6.4 UART Communication Mode ....................................................................................................... 175

6.5 3-Wire Serial I/O Communication Mode ....................................................................................... 178

APPENDIX A CIRCUIT DIAGRAMS (REFERENCE) ............................................................................. 181
APPENDIX B  REVISION HISTORY ........................................................................................................ 189

B.1 Major Revisions in This Edition.................................................................................................. 189
B.2 Revision History up to Previous Edition .................................................................................. 190
CHAPTER 1 FLASH MEMORY PROGRAMMING

To rewrite the contents of the internal flash memory of the 78K0/Kx2, a dedicated flash memory programmer (hereafter referred to as the “programmer”) is usually used.

This Application Note explains how to develop a dedicated programmer.

1.1 Overview

The 78K0/Kx2 incorporates firmware that controls flash memory programming. The programming to the internal flash memory is performed by transmitting/receiving commands between the programmer and the 78K0/Kx2 via serial communication.

Figure 1-1. System Outline of Flash Memory Programming in 78K0/Kx2
1.2 System Configuration

Examples of the system configuration for programming the flash memory are illustrated in Figure 1-2. These figures illustrate how to program the flash memory with the programmer, under control of a host machine. Depending on how the programmer is connected, the programmer can be used in a standalone mode without using the host machine, if a user program has been downloaded to the programmer in advance.

For example, NEC Electronics’ flash memory programmer PG-FP5 can execute programming either by using the GUI software with a host machine connected or by itself (standalone).

Figure 1-2. System Configuration Examples

(1) UART communication mode (LSB-first transfer)

(2) 3-wire serial I/O communication mode (CSI) (MSB-first transfer)

Remark  As for the pins used for flash memory programming and the recommended connections of unused pins, see the user's manual of each product.
1.3 Flash Memory Configuration

The 78K0/Kx2 must manage product-specific information (such as a device name and memory information). Table 1-1 shows the flash memory size of the 78K0/Kx2 and Figure 1-3 shows the configuration of the flash memory.

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Flash Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/KB2</td>
<td>μPD78F0500, 78F0500A 8 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0501, 78F0501A 16 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0502, 78F0502A 24 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0503, 78F0503A, 78F0503D, 78F0503DA 32 KB</td>
</tr>
<tr>
<td>78K0/KC2</td>
<td>μPD78F0511, 78F0511A 16 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0512, 78F0512A 24 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0513, 78F0513A, 78F0513D, 78F0513DA 32 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0514, 78F0514A 48 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0515, 78F0515A, 78F0515D, 78F0515DA 60 KB</td>
</tr>
<tr>
<td>78K0/KD2</td>
<td>μPD78F0521, 78F0521A 16 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0522, 78F0522A 24 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0523, 78F0523A 32 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0524, 78F0524A 48 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0525, 78F0525A 60 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0526, 78F0526A 96 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0527, 78F0527A, 78F0527D, 78F0527DA 128 KB</td>
</tr>
<tr>
<td>78K0/KE2</td>
<td>μPD78F0531, 78F0531A 16 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0532, 78F0532A 24 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0533, 78F0533A 32 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0534, 78F0534A 48 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0535, 78F0535A 60 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0536, 78F0536A 96 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0537, 78F0537A, 78F0537D, 78F0537DA 128 KB</td>
</tr>
<tr>
<td>78K0/KF2</td>
<td>μPD78F0544, 78F0544A 48 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0545, 78F0545A 60 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0546, 78F0546A 96 KB</td>
</tr>
<tr>
<td></td>
<td>μPD78F0547, 78F0547A, 78F0547D, 78F0547DA 128 KB</td>
</tr>
</tbody>
</table>
Figure 1-3. Flash Memory Configuration

<table>
<thead>
<tr>
<th>Block number</th>
<th>Address</th>
<th>Flash memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 7F (127)</td>
<td>1FFFH</td>
<td>128 KB</td>
</tr>
<tr>
<td>Block 60 (96)</td>
<td>C000H</td>
<td>96 KB</td>
</tr>
<tr>
<td>Block 5F (95)</td>
<td>BFFFH</td>
<td></td>
</tr>
<tr>
<td>Block 3C (60)</td>
<td>F000H</td>
<td></td>
</tr>
<tr>
<td>Block 3B (59)</td>
<td>EFFFH</td>
<td></td>
</tr>
<tr>
<td>Block 30 (48)</td>
<td>C000H</td>
<td>60 KB</td>
</tr>
<tr>
<td>Block 2F (47)</td>
<td>BFFFH</td>
<td></td>
</tr>
<tr>
<td>Block 20 (32)</td>
<td>8000H</td>
<td>48 KB</td>
</tr>
<tr>
<td>Block 1F (31)</td>
<td>7FFFH</td>
<td></td>
</tr>
<tr>
<td>Block 18 (24)</td>
<td>6000H</td>
<td>32 KB</td>
</tr>
<tr>
<td>Block 17 (23)</td>
<td>5FFFH</td>
<td></td>
</tr>
<tr>
<td>Block 10 (16)</td>
<td>4000H</td>
<td>24 KB</td>
</tr>
<tr>
<td>Block 0F (15)</td>
<td>3FFFH</td>
<td></td>
</tr>
<tr>
<td>Block 08 (08)</td>
<td>2000H</td>
<td>16 KB</td>
</tr>
<tr>
<td>Block 07 (07)</td>
<td>1FFFH</td>
<td></td>
</tr>
<tr>
<td>Block 00</td>
<td>0000H</td>
<td>8 KB</td>
</tr>
</tbody>
</table>

Remark: Each block consists of 1 KB (this figure only illustrates some parts of entire blocks in the flash memory).
1.4 Command List and Status List

The flash memory incorporated in the 78K0/Kx2 has functions to manipulate the flash memory, as listed in Table 1-2. The programmer transmits commands to control these functions to the 78K0/Kx2, and manipulates the flash memory with checking the response status from the 78K0/Kx2.

1.4.1 Command List

The commands used by the programmer and their functions are listed below.

<table>
<thead>
<tr>
<th>Command Number</th>
<th>Command Name</th>
<th>Function Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>20H</td>
<td>Chip Erase</td>
<td>Erase</td>
<td>Erases the entire flash memory area.</td>
</tr>
<tr>
<td>22H</td>
<td>Block Erase</td>
<td>Erase</td>
<td>Erases a specified area in the flash memory.</td>
</tr>
<tr>
<td>40H</td>
<td>Programming</td>
<td>Write</td>
<td>Writes data to a specified area in the flash memory.</td>
</tr>
<tr>
<td>13H</td>
<td>Verify</td>
<td>Verify</td>
<td>Compares the contents in a specified area in the flash memory with data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>transmitted from the programmer.</td>
</tr>
<tr>
<td>32H</td>
<td>Block Blank Check</td>
<td>Blank check</td>
<td>Checks the erase status of a specified block in the flash memory.</td>
</tr>
<tr>
<td>70H</td>
<td>Status</td>
<td>Information acquisition</td>
<td>Acquires the current operating status (status data).</td>
</tr>
<tr>
<td>00H</td>
<td>Reset</td>
<td>Others</td>
<td>Detects synchronization in communication.</td>
</tr>
<tr>
<td>90H</td>
<td>Oscillating Frequency Set</td>
<td></td>
<td>Specifies the oscillation frequency of the 78K0/Kx2.</td>
</tr>
</tbody>
</table>
1.4.2 Status List

The following table lists the status codes the programmer receives from the 78K0/Kx2.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04H</td>
<td>Command number error</td>
<td>Error returned if a command not supported is received</td>
</tr>
<tr>
<td>05H</td>
<td>Parameter error</td>
<td>Error returned if command information (parameter) is invalid</td>
</tr>
<tr>
<td>06H</td>
<td>Normal acknowledgment</td>
<td>Normal acknowledgment</td>
</tr>
<tr>
<td>07H</td>
<td>Checksum error</td>
<td>Error returned if data in a frame transmitted from the programmer is abnormal</td>
</tr>
<tr>
<td>0FH</td>
<td>Verify error</td>
<td>Error returned if a verify error has occurred upon verifying data transmitted from the programmer</td>
</tr>
<tr>
<td>10H</td>
<td>Protect error</td>
<td>Error returned if an attempt is made to execute processing that is prohibited by the Security Set command</td>
</tr>
<tr>
<td>15H</td>
<td>Negative acknowledgment</td>
<td>Negative acknowledgment</td>
</tr>
<tr>
<td>1AH</td>
<td>MRG10 error</td>
<td>Erase verify error</td>
</tr>
<tr>
<td>1BH</td>
<td>MRG11 error</td>
<td>Internal verify error or blank check error during data write</td>
</tr>
<tr>
<td>1CH</td>
<td>Write error</td>
<td>Write error</td>
</tr>
<tr>
<td>20H</td>
<td>Read error</td>
<td>Error returned when reading of security information failed</td>
</tr>
<tr>
<td>FFH</td>
<td>Processing in progress</td>
<td>Busy response</td>
</tr>
</tbody>
</table>

**Note** During CSI communication, 1-byte “FFH” may be transmitted, as well as “FFH” as the data frame format.

Reception of a checksum error or NACK is treated as an immediate abnormal end in this manual. When a dedicated programmer is developed, however, the processing may be retried without problem from the wait immediately before transmission of the command that results a checksum error or NACK. In this event, limiting the retry count is recommended for preventing infinite repetition of the retry operation.

Although not listed in the above table, if a time-out error (BUSY time-out or time-out in data frame reception during UART communication) occurs, it is recommended to shutdown the power supply to the 78K0/Kx2 (refer to 1.6 Shutting Down Target Power Supply) and then connect the power supply again.
1.5 Power Activation and Setting Flash Memory Programming Mode

To rewrite the contents of the flash memory with the programmer, the 78K0/Kx2 must first be set to the flash memory programming mode by supplying a specific voltage to the flash memory programming mode setting pin (FLMD0) in the 78K0/Kx2, then releasing a reset.

The programmer is received pulse input for rewriting flash memory from FLMD0 pin after programming mode transition.

The following illustrates a timing chart for setting the flash memory programming mode and selecting the communication mode.

![Figure 1-4. Setting Flash Memory Programming Mode and Selecting Communication Mode](image)

- **<1>: Power activation (VDD)**
- **<2>: FLMD0 = high level**
- **<3>: Reset release (mode setting)**
- **<4>: Pulse output starts**
- **<5>: Pulse output ends**

The relationship between the setting of the FLMD0 pin after reset release and the operating mode is shown below.

<table>
<thead>
<tr>
<th>FLMD0</th>
<th>Operating Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low (GND)</td>
<td>Normal operating mode</td>
</tr>
<tr>
<td>High (VDD)</td>
<td>Flash memory programming mode</td>
</tr>
</tbody>
</table>

The following table shows the relationship between the number of FLMD0 pulses (pulse counts) and communication modes that can be selected with the 78K0/Kx2.

<table>
<thead>
<tr>
<th>Communication Mode</th>
<th>FLMD0 Pulse Counts</th>
<th>Port Used for Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART (UART6)</td>
<td>0 (when X1 clock (fx) is used)</td>
<td>TxD6 (P13), RxD6 (P14)</td>
</tr>
<tr>
<td></td>
<td>3 (when external main system clock (fEXCLK) is used)</td>
<td></td>
</tr>
<tr>
<td>3-wire serial I/O (CSI10)</td>
<td>8</td>
<td>SO10 (P12), SI10 (P11), SCK10 (P10)</td>
</tr>
<tr>
<td>Setting prohibited</td>
<td>Others</td>
<td>–</td>
</tr>
</tbody>
</table>

Application Note U17739EJ3V0AN
• UART Communication Mode

The RxD and TxD pins are used for UART communication. The communication conditions are as shown below.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate</td>
<td>Communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted.</td>
</tr>
<tr>
<td></td>
<td>After the status frame is received, the communication rate is switched to 115,200 bps.</td>
</tr>
<tr>
<td></td>
<td>After that, the communication rate is fixed to 115,200 bps.</td>
</tr>
<tr>
<td>Parity bit</td>
<td>None</td>
</tr>
<tr>
<td>Data length</td>
<td>8 bits (LSB first)</td>
</tr>
<tr>
<td>Stop bit</td>
<td>1 bit</td>
</tr>
</tbody>
</table>

The programmer always operates as the master device during CSI communication, so the programmer must check whether the processing by the 78K0/Kx2, such as writing or erasing, is normally completed. On the other hand, the status of the master and slave is occasionally exchanged during UART communication, so communication at the optimum timing is possible.

**Caution** Set the same baud rate to the master and slave devices when performing UART communication.

• 3-Wire Serial I/O Communication Mode (CSI)

The SCK, SO and SI pins are used for CSI communication. The programmer always operates as the master device, so communication may not be performed normally if data is transmitted via the SCK pin while the 78K0/Kx2 is not ready for transmission/reception.

The communication data format is MSB-first, in 8-bit units. Keep the clock frequency 2.5 MHz or lower.
1.5.1 Mode Setting Flowchart

Programming Mode Setting

RESET pin low output

FLMD0 pin low output

Vcc pin high output (Target power supply on)

Wait

FLMD0 pin high output

Wait

RESET pin high output

Start of time measurement until start of reset command processing.

Is communication mode UART0 (FLMD0 pulse = 0)?

No

Wait

(tRP+tRPE)/2

Initialization of serial I/O hardware according to communication mode.

Output of pulse counts according to communication mode

Has specified time elapsed until start of reset command processing?

No

UART communication time: tR1

CSI communication time: tRC

Refer to Table 1-5 for the relationship between the pulse counts and communication modes.

Next, execute reset command processing for each communication mode.
## 1.5.2 Sample program

The following shows a sample program for mode setting processing.

```c
void fl_con_dev(void)
{
    extern void init_fl_uart(void);
    extern void init_fl_csi(void);

    int n;
    int pulse;
    SRMK0 = true;
    UARTE0 = false;

    switch (fl_if){
        default:
            pulse = PULSE_UART; break;
        case FLIF_UART:
            pulse = UseEXCLK ? PULSE_UART_EX : PULSE_UART; break;
        case FLIF_CSI:
            pulse = PULSE_CSI; break;
    }

    pFL_RES = low; // RESET = low
    pmFL_FLMD0 = PM_OUT; // FLMD0 = output mode
    pFL_FLMD0 = low;
    FL_VDD_HI(); // VDD = high
    fl_wait(tDP); // wait
    pFL_FLMD0 = hi; // FLMD0 = high
    fl_wait(tPR); // wait
    pFL_RES = hi; // RESET = high
    start_flto(fl_if == FLIF_CSI ? tRC : tR1); // start "tRC" wait timer
    fl_wait((tRP+tRPE)/2);

    if (fl_if == FLIF_UART){
        init_fl_uart(); // Initialize UART h.w.(for Flash device control)
        UARTE0 = true;
        SRIF0 = false;
        SRMK0 = false;
    }
    else{
        init_fl_csi(); // Initialize CSI h.w.
    }

    for (n = 0; n < pulse; n++) { // pulse output
        pFL_FLMD0 = low;
        fl_wait(tPW);
    }
}
```
pFL_FLMD0 = hi;
fl_wait(tPW);
} while(!check_flto()) // timeout tRC ?
; // no
// start RESET command proc.

1.6 Shutting Down Target Power Supply

After each command execution is completed, shut down the power supply to the target after setting the RESET pin to low level, as shown below.
Set other pins to Hi-Z when shutting down the power supply to the target.

Caution Shutting down the power supply and inputting a reset during command processing are prohibited.

Figure 1-5. Timing for Terminating Flash Memory Programming Mode

![Timing for Terminating Flash Memory Programming Mode](image)

1.7 Command Execution Flow at Flash Memory Rewriting

Figure 1-6 illustrates the basic flowchart when flash memory rewriting is performed with the programmer.
Other than commands shown in the Figure 1-6, the Verify command and Checksum command are also be supported.
Figure 1-6. Basic Flowchart for Flash Memory Rewrite Processing

Basic flow

Power application to target
(See Figure 1-4)

Mode setting (reset release)
(See 1.5)

Selection of communication mode
(pulse input)
(See 1.5)

Synchronization processing
(Reset command)
(See 3.2)

UART communication?

Yes

Oscillation frequency setting
(Oscillation Frequency Set command)
(See 3.4)

Command execution

Processing completed?

No

Yes

Target power shutdown processing
(See 1.6)

End

Remark  Figure 1-7 shows execution example of each command.
Figure 1-7. General Command Execution Flow at Flash Memory Rewriting

General command flow

Block Blank Check command
(See 3.9)

Yes

Execute Block Erase command
(See 3.6)

No

Execute Programming command
(See 3.7)

Execute Verify command
(See 3.8)

The programmer can use the Verify command to verify whether data transmission between the programmer and the target device is completed normally.

Execute Security Set command
(See 3.13)

End

The programmer can use the Verify command to verify whether data transmission between the programmer and the target device is completed normally.
CHAPTER 2 COMMAND/DATA FRAME FORMAT

The programmer uses the command frame to transmit commands to the 78K0/Kx2. The 78K0/Kx2 uses the data frame to transmit write data or verify data to the programmer. A header, footer, data length information, and checksum are appended to each frame to enhance the reliability of the transferred data.

The following shows the format of a command frame and data frame.

Figure 2-1. Command Frame Format

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOH</td>
<td>01H</td>
<td>Command frame header</td>
</tr>
<tr>
<td>LEN</td>
<td></td>
<td>Data length information (00H indicates 256).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Command frame: COM + command information length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data frame: Data field length</td>
</tr>
<tr>
<td>COM</td>
<td></td>
<td>Command number</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td>Checksum data for a frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obtained by sequentially subtracting all of calculation target data from the initial value (00H) in 1-byte units (borrow is ignored). The calculation targets are as follows. Command frame: LEN + COM + all of command information Data frame: LEN + all of data</td>
</tr>
<tr>
<td>ETX</td>
<td>03H</td>
<td>Command frame footer, or footer of last data frame</td>
</tr>
<tr>
<td>ETB</td>
<td>17H</td>
<td>Footer of data frame other than the last frame</td>
</tr>
</tbody>
</table>

Figure 2-2. Data Frame Format

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>02H</td>
<td>Data frame header</td>
</tr>
<tr>
<td>LEN</td>
<td></td>
<td>Data length information (00H indicates 256).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Command frame: COM + command information length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data frame: Data field length</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td>Checksum data for a frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obtained by sequentially subtracting all of calculation target data from the initial value (00H) in 1-byte units (borrow is ignored). The calculation targets are as follows. Command frame: LEN + COM + all of command information Data frame: LEN + all of data</td>
</tr>
<tr>
<td>ETX</td>
<td>03H</td>
<td>Command frame footer, or footer of last data frame</td>
</tr>
<tr>
<td>ETB</td>
<td>17H</td>
<td>Footer of data frame other than the last frame</td>
</tr>
</tbody>
</table>

Table 2-1. Description of Symbols in Each Frame

The following shows examples of calculating the checksum (SUM) for a frame.
[Command frame]
No command information is included in the following example of a Status command frame, so LEN and COM are targets of checksum calculation.

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>70H</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Checksum calculation targets

For this command frame, checksum data is obtained as follows.

00H (initial value) – 01H (LEN) – 70H (COM) = 8FH (Borrow ignored. Lower 8 bits only.)

The command frame finally transmitted is as follows.

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>70H</td>
<td>8FH</td>
<td>03H</td>
</tr>
</tbody>
</table>

[Data frame]
To transmit a data frame as shown below, LEN and D1 to D4 are targets of checksum calculation.

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>04H</td>
<td>FFH</td>
<td>80H</td>
<td>40H</td>
<td>22H</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Checksum calculation targets

For this data frame, checksum data is obtained as follows.

00H (initial value) – 04H (LEN) – FFH (D1) – 80H (D2) – 40H (D3) – 22H (D4)
= 1BH (Borrow ignored. Lower 8 bits only.)

The data frame finally transmitted is as follows.

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>04H</td>
<td>FFH</td>
<td>80H</td>
<td>40H</td>
<td>22H</td>
<td>1BH</td>
<td>03H</td>
</tr>
</tbody>
</table>

When a data frame is received, the checksum data is calculated in the same manner, and the obtained value is used to detect a checksum error by judging whether the value is the same as that stored in the SUM field of the receive data. When a data frame as shown below is received, for example, a checksum error is detected.

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>04H</td>
<td>FFH</td>
<td>80H</td>
<td>40H</td>
<td>22H</td>
<td>1AH</td>
<td>03H</td>
</tr>
</tbody>
</table>

↑ Should be 1BH, if normal
2.1 Command Frame Transmission Processing

Read the following chapters for details on flowcharts of command processing to transmit command frames, for each communication mode.

- For the UART communication mode, read 4.1 Flowchart of Command Frame Transmission Processing.
- For the 3-wire serial I/O communication mode (CSI), read 5.1 Flowchart of Command Frame Transmission Processing.

2.2 Data Frame Transmission Processing

The write data frame (user program), verify data frame (user program), and security data frame (security flag) are transmitted as a data frame.

Read the following chapters for details on flowcharts of command processing to transmit data frames, for each communication mode.

- For the UART communication mode, read 4.2 Flowchart of Data Frame Transmission Processing.
- For the 3-wire serial I/O communication mode (CSI), read 5.2 Flowchart of Data Frame Transmission Processing.

2.3 Data Frame Reception Processing

The status frame, silicon signature data frame, version data frame, and checksum data frame are received as a data frame.

Read the following chapters for details on flowcharts of command processing to receive data frames, for each communication mode.

- For the UART communication mode, read 4.3 Flowchart of Data Frame Reception Processing.
- For the 3-wire serial I/O communication mode (CSI), read 5.3 Flowchart of Data Frame Reception Processing.
CHAPTER 3 DESCRIPTION OF COMMAND PROCESSING

3.1 Status Command

3.1.1 Description

This command is used to check the operation status of the 78K0/Kx2 after issuance of each command such as write or erase. After the Status command is issued, if the Status command frame cannot be received normally in the 78K0/Kx2 due to problems based on communication or the like, the status setting will not be performed in the 78K0/Kx2. As a result, a busy response (FFH), not the status frame, may be received. In such a case, retry the Status command.

3.1.2 Command frame and status frame

Figure 3-1 shows the format of a command frame for the Status command, and Figure 3-2 shows the status frame for the command.

### Figure 3-1. Status Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>70H (Status)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

### Figure 3-2. Status Frame for Status Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>n</td>
<td>ST1</td>
<td>... STn</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

Remarks 1. ST1 to STn: Status #1 to Status #n
2. The length of a status frame varies according to each command (such as write or erase) to be transmitted to the 78K0/Kx2.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- The Status command is not used in the UART communication mode.
- For the 3-wire serial I/O communication mode (CSI), read 5.4 Status Command.

**Caution** After each command such as write or erase is transmitted in UART communication, the 78K0/Kx2 automatically returns the status frame within a specified time. The Status command is therefore not used.

If the Status command is transmitted in UART communication, the Command Number Error is returned.
3.2 Reset Command

3.2.1 Description

This command is used to check the establishment of communication between the programmer and the 78K0/Kx2 after the communication mode is set.

When UART is selected as the mode for communication with the 78K0/Kx2, the same baud rate must be set in the programmer and 78K0/Kx2. However, the 78K0/Kx2 cannot detect its own baud rate generation clock (fx or fEXCLK) frequency so the baud rate cannot be set. It makes detection of the baud rate generation clock frequency in the 78K0/Kx2 possible by sending “00H” twice at 9,600 bps from the programmer, measuring the low-level width of “00H”, and then calculating the average of two sent signals. The baud rate can consequently be set, which enables synchronous detection in communication.

3.2.2 Command frame and status frame

Figure 3-3 shows the format of a command frame for the Reset command, and Figure 3-4 shows the status frame for the command.

Figure 3-3. Reset Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>00H (Reset)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Figure 3-4. Status Frame for Reset Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>1</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1: Synchronization detection result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

• For the UART communication mode, read 4.4 Reset Command.
• For the 3-wire serial I/O communication mode (CSI), read 5.5 Reset Command.
3.3 Baud Rate Set Command

The 78K0/Kx2 does not support the Baud Rate Set command. With the 78K0/Kx2, UART communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted. After the status frame is received, the communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.

3.4 Oscillating Frequency Set Command

3.4.1 Description

This command is used to specify the frequency of fX or fEXCLK during UART communication. The 78K0/Kx2 uses the frequency data in the received packet to realize the baud rate of 115,200 bps.

Caution With the 78K0/Kx2, UART communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted. After the status frame is received, the communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.

3.4.2 Command frame and status frame

Figure 3-5 shows the format of a command frame for the Oscillating Frequency Set command, and Figure 3-6 shows the status frame for the command.

Figure 3-5. Oscillating Frequency Set Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>05H</td>
<td>90H</td>
<td>(Oscillating Frequency Set)</td>
<td>D01</td>
<td>D02</td>
</tr>
</tbody>
</table>

Remark D01 to D04: Oscillation frequency = (D01 × 0.1 + D02 × 0.01 + D03 × 0.001) × 10⁵ (Unit: kHz) Settings can be made from 10 kHz to 100 MHz, but set the value according to the specifications of each device when actually transmitting the command. D01 to D03 hold unpacked BCDs, and D04 holds a signed integer.

Setting example: To set 6 MHz

D01 = 06H  
D02 = 00H  
D03 = 00H  
D04 = 04H  
Oscillation frequency = 6 × 0.1 × 10⁵ = 6,000 kHz = 6 MHz

Setting example: To set 10 MHz

D01 = 01H  
D02 = 00H  
D03 = 00H  
D04 = 05H  
Oscillation frequency = 1 × 0.1 × 10⁵ = 10,000 kHz = 10 MHz
CHAPTER 3 DESCRIPTION OF COMMAND PROCESSING

Figure 3-6. Status Frame for Oscillating Frequency Set Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1: Oscillation frequency setting result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.5 Oscillating Frequency Set Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.6 Oscillating Frequency Set Command.

3.5 Chip Erase Command

3.5.1 Description

This command is used to erase the entire contents of the flash memory. In addition, all of the information that is set by security setting processing can be initialized by chip erase processing, as long as Chip Erase command execution is impossible due to the security setting (see 3.13 Security Set Command).

3.5.2 Command frame and status frame

Figure 3-7 shows the format of a command frame for the Chip Erase command, and Figure 3-8 shows the status frame for the command.

Figure 3-7. Chip Erase Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>20H</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

(Chip Erase)

Figure 3-8. Status Frame for Chip Erase Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1: Chip erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.6 Chip Erase Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.7 Chip Erase Command.
3.6 Block Erase Command

3.6.1 Description
   Specify from the start address of erase start block to the end address of erase end block. It can specify multiple
   contiguous blocks.
   However, if Block Erase command is not impossible by the security setting, the contents is not erased (see 3.13
   Security Set Command).
   
3.6.2 Command frame and status frame
   Figure 3-9 shows the format of a command frame for the Block Erase command, and Figure 3-10 shows the status
   frame for the command.

   **Figure 3-9. Block Erase Command Frame (from Programmer to 78K0/Kx2)**

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>07H</td>
<td>22H</td>
<td>SAH SAM SAL EAH EAM EAL</td>
<td>03H</td>
<td></td>
</tr>
</tbody>
</table>

   **Remark**
   SAH, SAM, SAL: Block erase start address (start address of any block)
   SAH: Start address, high (bits 23 to 16) (fixed to 00H)
   SAM: Start address, middle (bits 15 to 8) (fixed to 00H)
   SAL: Start address, low (bits 7 to 0) (fixed to 00H)
   EAH, EAM, EAL: Block erase end address (last address of the internal flash memory)
   EAH: End address, high (bits 23 to 16)
   EAM: End address, middle (bits 15 to 8)
   EAL: End address, low (bits 7 to 0)

   **Figure 3-10. Status Frame for Block Erase Command (from 78K0/Kx2 to Programmer)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

   **Remark**
   ST1: Block erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the
78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.7 Block Erase Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.8 Block Erase Command.
3.7 Programming Command

3.7.1 Description

This command is used to transmit data by the number of written bytes after the write start address and the write end address are transmitted. This command then writes the user program to the flash memory and verifies it internally.

The write start/end address can be set only in the block start/end address units.

If both of the status frames (ST1 and ST2) after the last data transmission indicate ACK, the 78K0/Kx2 firmware automatically executes internal verify. Therefore, the status code validation for this internal verification is necessary.

3.7.2 Command frame and status frame

Figure 3-11 shows the format of a command frame for the Programming command, and Figure 3-12 shows the status frame for the command.

**Figure 3-11. Programming Command Frame (from Programmer to 78K0/Kx2)**

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>07H</td>
<td>40H (Programming)</td>
<td>SAH SAM SAL EAH EAM EAL</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Remark**  
SAH, SAM, SAL: Write start addresses  
EAH, EAM, EAL: Write end addresses

**Figure 3-12. Status Frame for Programming Command (from 78K0/Kx2 to Programmer)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (a)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Remark**  
ST1 (a): Command reception result

3.7.3 Data frame and status frame

Figure 3-13 shows the format of a frame that includes data to be written, and Figure 3-14 shows the status frame for the data.

**Figure 3-13. Data Frame to Be Written (from Programmer to 78K0/Kx2)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX/ETB</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>00H</td>
<td>00H to FFH</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
</tbody>
</table>

**Remark**  
Write Data: User program to be written

**Figure 3-14. Status Frame for Data Frame (from 78K0/Kx2 to Programmer)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>02H</td>
<td>ST1 (b) ST2 (b)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Remark**  
ST1 (b): Data reception check result  
ST2 (b): Write result
3.7.4 Completion of transferring all data and status frame

Figure 3-15 shows the status frame after transfer of all data is completed.

Figure 3-15. Status Frame After Completion of Transferring All Data (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (c)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1 (c): Internal verify result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.8 Programming Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.9 Programming Command.

3.8 Verify Command

3.8.1 Description

This command is used to compare the data transmitted from the programmer with the data read from the 78K0/Kx2 (read level) in the specified address range, and check whether they match.

The verify start/end address can be set only in the block start/end address units.

3.8.2 Command frame and status frame

Figure 3-16 shows the format of a command frame for the Verify command, and Figure 3-17 shows the status frame for the command.

Figure 3-16. Verify Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>07H</td>
<td>13H</td>
<td>SAH, SAM, SAL, EAH, EAM, EAL</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  SAH, SAM, SAL: Verify start addresses

EAH, EAM, EAL: Verify end addresses

Figure 3-17. Status Frame for Verify Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (a)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1 (a): Command reception result
3.8.3 Data frame and status frame

Figure 3-18 shows the format of a frame that includes data to be verified, and Figure 3-19 shows the status frame for the data.

**Figure 3-18. Data Frame of Data to Be Verified (from Programmer to 78K0/Kx2)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX/ETB</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>00H</td>
<td>Verify data</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
<tr>
<td>00H to FFH (00H = 256)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remark** Verify Data: User program to be verified

**Figure 3-19. Status Frame for Data Frame (from 78K0/Kx2 to Programmer)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>02H</td>
<td>ST1 (b)</td>
<td>ST2 (b)</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Remark** ST1 (b): Data reception check result
ST2 (b): Verify result

**Note** Even if a verify error occurs in the specified address range, ACK is always returned as the verify result. The status of all verify errors are reflected in the verify result for the last data. Therefore, the occurrence of verify errors can be checked only when all the verify processing for the specified address range is completed.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.9 Verify Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.10 Verify Command.
3.9 Block Blank Check Command

3.9.1 Description

This command is used to check if a block in the flash memory, with a specified block number, is blank (erased state).

Specify from the start address of blank check start block to the last address of blank check end block. It can specify multiple contiguous blocks.

3.9.2 Command frame and status frame

Figure 3-20 shows the format of a command frame for the Block Blank Check command, and Figure 3-21 shows the status frame for the command.

Figure 3-20. Block Blank Check Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>07H</td>
<td>32H</td>
<td>Block Blank Check</td>
<td>SAH</td>
<td>SAM</td>
</tr>
</tbody>
</table>

Remark  SAH, SAM, SAL: Block blank check start address (start address of any block)

SAH: Start address, high (bits 23 to 16)
SAM: Start address, middle (bits 15 to 8)
SAL: Start address, low (bits 7 to 0)

EAH, EAM, EAL: Block blank check end address (last address of any block)

EAH: End address, high (bits 23 to 16)
EAM: End address, middle (bits 15 to 8)
EAL: End address, low (bits 7 to 0)

Figure 3-21. Status Frame for Block Blank Check Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1: Block blank check result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.10 Block Blank Check Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.11 Block Blank Check Command.
3.10 Silicon Signature Command

3.10.1 Description
This command is used to read the write protocol information (silicon signature) of the device.
If the programmer supports a programming protocol that is not supported in the 78K0/Kx2, for example, execute
this command to select an appropriate protocol in accordance with the values of the second and third bytes.

3.10.2 Command frame and status frame
Figure 3-22 shows the format of a command frame for the Silicon Signature command, and Figure 3-23 shows the
status frame for the command.

Figure 3-22. Silicon Signature Command Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>C0H</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

(Silicon Signature)

Figure 3-23. Status Frame for Silicon Signature Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1: Command reception result

3.10.3 Silicon signature data frame
Figure 3-24 shows the format of a frame that includes silicon signature data.

Figure 3-24. Silicon Signature Data Frame (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>n</td>
<td>VEN</td>
<td>MET</td>
<td>MSC</td>
</tr>
</tbody>
</table>

Remarks 1.  
- n (LEN): Data length
- VEN: Vendor code (NEC: 10H)
- MET: Macro extension code
- MSC: Macro function code
- DEC: Device extension code
- END: Internal flash memory last address
- DEV: Device name ($\mu$PDxx)
- SCF: Security flag information
- BOT: Boot block number (fixed to 03H)

2.  For above fields except boot block number (BOT), the lower 7 bits are used as data entity, and
the highest bit is used as an odd parity. The following shows an example.
### Table 3-1. Example of Silicon Signature Data (In Case of μPD78F0522 (78K0/KD2))

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
<th>Length (Byte)</th>
<th>Example of Silicon Signature Data&lt;sup&gt;Note 1&lt;/sup&gt;</th>
<th>Actual Value</th>
<th>Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEN</td>
<td>Vendor code (NEC)</td>
<td>1</td>
<td>10H (00010000B)</td>
<td>10H</td>
<td>Added</td>
</tr>
<tr>
<td>MET</td>
<td>Extension code (fixed in 78K0/Kx2)</td>
<td>1</td>
<td>7FH (01111111B)</td>
<td>7FH</td>
<td>Added</td>
</tr>
<tr>
<td>MSC</td>
<td>Function information (fixed in 78K0/Kx2)</td>
<td>1</td>
<td>04H (00000100B)</td>
<td>04H</td>
<td>Added</td>
</tr>
<tr>
<td>DEC</td>
<td>Device extension code (fixed in 78K0/Kx2)</td>
<td>1</td>
<td>7CH (01111110B)</td>
<td>07H</td>
<td>Added</td>
</tr>
<tr>
<td>END</td>
<td>Internal flash memory last address (extracted from the lower bytes)</td>
<td>3</td>
<td>7FH (01111111B) B0H (10110000B) B5H (10110101B) 01H (0000001B)</td>
<td>005FFFH</td>
<td>Added&lt;sup&gt;Note 2&lt;/sup&gt;</td>
</tr>
<tr>
<td>DEV</td>
<td>Device name</td>
<td>10</td>
<td>C4H (11000100B) = 'D'</td>
<td>'D'</td>
<td>Added</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>37H (00110111B) = '7'</td>
<td>'7'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>38H (00111000B) = '8'</td>
<td>'8'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>46H (01000110B) = 'F'</td>
<td>'F'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B0H (10110000B) = '0'</td>
<td>'0'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B5H (10110101B) = '5'</td>
<td>'5'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32H (01100101B) = '2'</td>
<td>'2'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32H (00110010B) = '2'</td>
<td>'2'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20H (01000000B = ' ')</td>
<td>' '</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20H (00100000B = ' ')</td>
<td>' '</td>
<td></td>
</tr>
<tr>
<td>SCF</td>
<td>Security flag information</td>
<td>1</td>
<td>Any</td>
<td>Any</td>
<td>Added&lt;sup&gt;Note 3&lt;/sup&gt;</td>
</tr>
<tr>
<td>BOT</td>
<td>The last block number of the boot block cluster (fixed)</td>
<td>1</td>
<td>03H (0000001B)</td>
<td>03H</td>
<td>Not added</td>
</tr>
</tbody>
</table>

**Notes**

1. 0 and 1 are odd parities (the values to adjust the number of “1” to be the odd number in a byte)
2. The parity calculation for the END field is performed as follows (when the last address is 005FFFH)
   
   <1> The END field is divided in 7-bit units from the lower digit (the higher 3 bits are discarded).

   $\begin{array}{c}
   0 \ 0 \ 5 \ F \ F \ F \\
   \hline
   00000000 \ 01011111 \ 11111111
   \end{array}$

   $\downarrow$

   $\begin{array}{c}
   0000001 \ 0111111 \ 1111111
   \end{array}$

   <2> The odd parity bit is appended to the highest bit.

   $\begin{array}{c}
   00000001 \ 011111111 \ 111111111
   \end{array}$  ($p = \text{odd parity bit}$)

   $\begin{array}{c}
   = 0000001 \ 10111111 \ 0111111
   \end{array}$

   $\begin{array}{c}
   = 01 \ BF \ 7F
   \end{array}$

   <3> The order of the higher, middle, and lower bytes is reversed, as follows.

   $\begin{array}{c}
   7F \ BF \ 01
   \end{array}$
The following shows the procedure to translate the values in the END field that has been sent from the microcontroller to the actual address.

<1> The order of the higher, middle, and lower bytes is reversed, as follows.

\[
\begin{align*}
7F &\rightarrow 01 \\
BF &\rightarrow BF \\
01 &\rightarrow 7F
\end{align*}
\]

<2> Checks that the number of “1” is odd in each byte (this can be performed at another timing).

<3> The parity bit is removed and a 3-bit 0 is added to the highest bit.

\[
\begin{align*}
01 &\rightarrow 00000001 10111111 01111111 \\
BF &\rightarrow 0000001 01111111 11111111 \\
7F &\rightarrow 000 0000001 0111111 1111111
\end{align*}
\]

<4> The values are translated into groups in 8-bit units.

\[
\begin{align*}
000000010111111111111111 &\rightarrow 00000000 01011111 11111111 \\
000000010111111111111111 &\rightarrow 00000000 01011111 11111111 \\
000000001011111111111111 &\rightarrow 0 0 5 F F F
\end{align*}
\]

If “7F BF 01” is given to the END field, the actual last address is consequently 005FFFH.

**Note 3.** When security flag information is set using the Security Set command, the highest bit is fixed to “1”. If the security flag information is read using the Silicon Signature command, however, the highest bit is the odd parity.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **4.11 Silicon Signature Command**.
- For the 3-wire serial I/O communication mode (CSI), read **5.12 Silicon Signature Command**.
### 3.10.4 78K0/Kx2 silicon signature list

Table 3-2. 78K0/Kx2 Silicon Signature Data List

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Length (Bytes)</th>
<th>Data (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor code</td>
<td>NEC</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Extension code</td>
<td>Extension code</td>
<td>1</td>
<td>7F</td>
</tr>
<tr>
<td>Function code</td>
<td>Function information</td>
<td>1</td>
<td>04</td>
</tr>
<tr>
<td>Device information</td>
<td>Device information</td>
<td>1</td>
<td>7C</td>
</tr>
<tr>
<td>Internal flash memory last address</td>
<td>(7-bit data + odd parity bit) \times 3</td>
<td>3</td>
<td>Note 1</td>
</tr>
<tr>
<td>Security information</td>
<td>Security information</td>
<td>1</td>
<td>Any</td>
</tr>
<tr>
<td>Boot block number</td>
<td>The last block number of the boot cluster that is currently selected</td>
<td>1</td>
<td>03</td>
</tr>
</tbody>
</table>

**Notes 1.** List of internal flash memory last addresses

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Length (Bytes)</th>
<th>Data (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal flash memory last address</td>
<td>8 KB (1FFFH)</td>
<td>3</td>
<td>7FBF80</td>
</tr>
<tr>
<td></td>
<td>16 KB (3FFFH)</td>
<td></td>
<td>7FF780</td>
</tr>
<tr>
<td></td>
<td>24 KB (5FFFH)</td>
<td></td>
<td>7FBF01</td>
</tr>
<tr>
<td></td>
<td>32 KB (7FFFH)</td>
<td></td>
<td>7FF701</td>
</tr>
<tr>
<td></td>
<td>48 KB (BFFFFH)</td>
<td></td>
<td>7FF702</td>
</tr>
<tr>
<td></td>
<td>60 KB (EFFFFH)</td>
<td></td>
<td>7FFDF83</td>
</tr>
<tr>
<td></td>
<td>96 KB (17FFFFH)</td>
<td></td>
<td>7FF705</td>
</tr>
<tr>
<td></td>
<td>128 KB (1FFFFH)</td>
<td></td>
<td>7FF7F07</td>
</tr>
</tbody>
</table>

(Notes 2 is listed on the next page.)
Notes 2.  The device names are listed below.

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Device name</th>
<th>Length (bytes)</th>
<th>Actual Value</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/KB2</td>
<td>D78F0500</td>
<td>10</td>
<td>C4 37 38 46 B0 B5 B0 B0 20 20</td>
<td>D 7 8 F 0 5 0 0 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0500A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 C1 20</td>
<td>D 7 8 F 0 5 0 0 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0501</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 31 20 20</td>
<td>D 7 8 F 0 5 0 1 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0501A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 31 C1 20</td>
<td>D 7 8 F 0 5 0 1 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0502</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 32 20 20</td>
<td>D 7 8 F 0 5 0 2 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0502A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 32 C1 20</td>
<td>D 7 8 F 0 5 0 2 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0503</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 B3 20 20</td>
<td>D 7 8 F 0 5 0 3 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0503D</td>
<td></td>
<td>C4 37 38 46 B0 B5 B0 B0 B3 C1 20</td>
<td>D 7 8 F 0 5 0 3 A -</td>
<td></td>
</tr>
<tr>
<td>78K0/KC2</td>
<td>D78F0511</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 31 31 C1 20</td>
<td>D 7 8 F 0 5 1 1 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0511A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 31 C1 20</td>
<td>D 7 8 F 0 5 1 1 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0512</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 32 20 20</td>
<td>D 7 8 F 0 5 1 2 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0512A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 32 C1 20</td>
<td>D 7 8 F 0 5 1 2 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0513</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 31 B3 20 20</td>
<td>D 7 8 F 0 5 1 3 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0513D</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 31 B3 C1 20</td>
<td>D 7 8 F 0 5 1 3 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0513A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 34 20 20</td>
<td>D 7 8 F 0 5 1 4 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0514</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 34 C1 20</td>
<td>D 7 8 F 0 5 1 4 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0515</td>
<td></td>
<td>C4 37 38 46 B0 B5 B5 20 20</td>
<td>D 7 8 F 0 5 1 5 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0515D</td>
<td></td>
<td>C4 37 38 46 B0 B5 B5 C1 20</td>
<td>D 7 8 F 0 5 1 5 A -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0515A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B5 31 20 20</td>
<td>D 7 8 F 0 5 1 6 - -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D78F0515DA</td>
<td></td>
<td>C4 37 38 46 B0 B5 B5 31 C1 20</td>
<td>D 7 8 F 0 5 1 6 A -</td>
<td></td>
</tr>
</tbody>
</table>
### Device name list (2/4)

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Device name</th>
<th>Length (bytes)</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/KD2</td>
<td>D78F0521</td>
<td>10</td>
<td>C4 37 38 46 B0 B5 32 31 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 1 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0521A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 31 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 1 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0522</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 2 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0522A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 2 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0523</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B3 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 3 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0523A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B3 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 3 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0524</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 34 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 4 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0524A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 34 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 4 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0525</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B5 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 5 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0525A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B5 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 5 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0526</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B6 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 6 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0526A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 32 B6 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 2 6 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0527</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 37 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0527D</td>
<td></td>
<td>D 7 8 F 0 5 2 7 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0527A</td>
<td></td>
<td>C4 37 38 46 B0 B5 32 37 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0527DA</td>
<td></td>
<td>D 7 8 F 0 5 2 7 A -</td>
</tr>
</tbody>
</table>
## Device name list (3/4)

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Device name</th>
<th>Length (bytes)</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Upper row: Signature code</td>
</tr>
<tr>
<td>78K0/KE2</td>
<td>D78F0531</td>
<td>10</td>
<td>C4 37 38 46 B0 B5 B3 31 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0531A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 31 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0532</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 32 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0532A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 32 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0533</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B3 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0533A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B3 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0534</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B3 34 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0534A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B3 34 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0535</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B5 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0535A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B5 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0536</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B6 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0536A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 B6 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0537</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 37 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0537A</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 37 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0537DA</td>
<td></td>
<td>C4 37 38 46 B0 B5 B3 37 C1 20</td>
</tr>
</tbody>
</table>
## Device name list (4/4)

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Device name</th>
<th>Length (bytes)</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/KF2</td>
<td>D78F0544</td>
<td>10</td>
<td>C4 37 38 46 B0 B5 34 34 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 4 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0544A</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 34 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 4 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0545</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 B5 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 5 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0545A</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 B5 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 5 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0546</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 B6 20 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 6 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0546A</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 B6 C1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D 7 8 F 0 5 4 6 A -</td>
</tr>
<tr>
<td></td>
<td>D78F0547</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 37 20 20</td>
</tr>
<tr>
<td></td>
<td>D78F0547D</td>
<td></td>
<td>D 7 8 F 0 5 4 7 - -</td>
</tr>
<tr>
<td></td>
<td>D78F0547A</td>
<td></td>
<td>C4 37 38 46 B0 B5 34 37 C1 20</td>
</tr>
<tr>
<td></td>
<td>D78F0547DA</td>
<td></td>
<td>D 7 8 F 0 5 4 7 A -</td>
</tr>
</tbody>
</table>
3.11 Version Get Command

3.11.1 Description

This command is used to acquire information on the 78K0/Kx2 device version and firmware version. The device version value is fixed to 00H. Use this command when the programming parameters must be changed in accordance with the 78K0/Kx2 firmware version.

Caution The firmware version may be updated during firmware update that does not affect the change of flash programming parameters (at this time, update of the firmware version is not reported).

Example Firmware version and reprogramming parameters

<table>
<thead>
<tr>
<th>Firmware version</th>
<th>Programming parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.00</td>
<td>Parameter A</td>
</tr>
<tr>
<td>V2.00</td>
<td>Parameter B</td>
</tr>
<tr>
<td>V3.00</td>
<td></td>
</tr>
</tbody>
</table>

3.11.2 Command frame and status frame

Figure 3-25 shows the format of a command frame for the Version Get command, and Figure 3-26 shows the status frame for the command.

**Figure 3-25. Version Get Command Frame (from Programmer to 78K0/Kx2)**

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>01H</td>
<td>C5H</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Figure 3-26. Status Frame for Version Get Command (from 78K0/Kx2 to Programmer)**

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

**Remark** ST1: Command reception result
3.11.3 Version data frame

Figure 3-27 shows the data frame of version data.

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>06H</td>
<td>DV1</td>
<td>DV2</td>
<td>DV3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FV1</td>
<td>FV2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark
DV1: Integer of device version (fixed to 00H)
DV2: First decimal place of device version (fixed to 00H)
DV3: Second decimal place of device version (fixed to 00H)
FV1: Integer of firmware version
FV2: First decimal place of firmware version
FV3: Second decimal place of firmware version

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.12 Version Get Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.13 Version Get Command.

3.12 Checksum Command

3.12.1 Description

This command is used to acquire the checksum data in the specified area.

For the checksum calculation start/end address, specify a fixed address in block units (1 KB) starting from the top of the flash memory.

Checksum data is obtained by sequentially subtracting data in the specified address range from the initial value (0000H) in 1-byte units.

3.12.2 Command frame and status frame

Figure 3-28 shows the format of a command frame for the Checksum command, and Figure 3-29 shows the status frame for the command.

<table>
<thead>
<tr>
<th>SOH</th>
<th>LEN</th>
<th>COM</th>
<th>Command Information</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>07H</td>
<td>00H (Checksum)</td>
<td>SAH</td>
<td>SAM</td>
<td>SAL</td>
</tr>
</tbody>
</table>

Remark
SAH, SAM, SAL: Checksum calculation start addresses
EAH, EAM, EAL: Checksum calculation end addresses

Figure 3-29. Status Frame for Checksum Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark
ST1: Command reception result
3.12.3 Checksum data frame

Figure 3-30 shows the format of a frame that includes checksum data.

![Figure 3-30. Checksum Data Frame (from 78K0/Kx2 to Programmer)](image)

**Remark**  CK1: Higher 8 bits of checksum data
CK2: Lower 8 bits of checksum data

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **4.13 Checksum Command**.
- For the 3-wire serial I/O communication mode (CSI), read **5.14 Checksum Command**.

### 3.13 Security Set Command

#### 3.13.1 Description

This command is used to perform security settings (enable or disable of write, block erase, chip erase, and boot block cluster rewriting). By performing these settings with this command, rewriting of the flash memory by an unauthorized person can be restricted.

**Caution**  Even after the security setting, additional setting of changing from enable to disable can be performed; however, changing from disable to enable is not possible. If an attempt is made to perform such a setting, a protect error (10H) will occur. If such setting is required, all of the security flags must first be initialized by executing the Chip Erase command (the Block Erase command cannot be used to initialize the security flags).

If chip erase or boot block cluster rewrite has been disabled, however, chip erase itself will be impossible, so the settings cannot be erased from the programmer. Re-confirmation of security setting execution is therefore recommended before disabling chip erase, due to this programmer specification.

#### 3.13.2 Command frame and status frame

Figure 3-31 shows the format of a command frame for the Security Set command, and Figure 3-32 shows the status frame for the command.

The Security Set command frame includes the block number field and page number field but these fields do not have any particular usage, so set these fields to 00H.

![Figure 3-31. Security Set Command Frame (from Programmer to 78K0/Kx2)](image)
Figure 3-32. Status Frame for Security Set Command (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (a)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1 (a): Command reception result

3.13.3 Data frame and status frame

Figure 3-33 shows the format of a security data frame, and Figure 3-34 shows the status frame for the data.

Figure 3-33. Security Data Frame (from Programmer to 78K0/Kx2)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>02H</td>
<td>FLG</td>
<td>BOT</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

Remark  FLG: Security flag  
BOT: Boot block cluster last block number (fixed to 03H)

Figure 3-34. Status Frame for Security Data Writing (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (b)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1 (b): Security data write result

3.13.4 Internal verify check and status frame

Figure 3-35 shows the status frame for internal verify check.

Figure 3-35. Status Frame for Internal Verify Check (from 78K0/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>01H</td>
<td>ST1 (c)</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  ST1 (c): Internal verify result

The following table shows the contents in the security flag field.

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
<td>Fixed to “1”</td>
</tr>
<tr>
<td>Bit 6</td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td></td>
</tr>
<tr>
<td>Bit 4</td>
<td>Boot block cluster rewrite disable flag (1: Enables boot block rewrite, 0: Disable boot block rewrite)</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Fixed to “1”</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Programming disable flag (1: Enables programming, 0: Disable programming)</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Block erase disable flag (1: Enables block erase, 0: Disable block erase)</td>
</tr>
<tr>
<td>Bit 0</td>
<td>Chip erase disable flag (1: Enables chip erase, 0: Disable chip erase)</td>
</tr>
</tbody>
</table>
The following table shows the relationship between the security flag field settings and the enable/disable status of each operation.

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Command Setting Item</th>
<th>Flash Memory Programming Mode</th>
<th>Self-Programming Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Command</td>
<td>Programming</td>
<td>Chip Erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>Disable programming</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Disable chip erase</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>Disable block erase</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Boot block rewrite disable flag</td>
<td>△</td>
<td>×</td>
</tr>
</tbody>
</table>

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 4.14 Security Set Command.
- For the 3-wire serial I/O communication mode (CSI), read 5.15 Security Set Command.
CHAPTER 4 UART COMMUNICATION MODE

Each of the symbol (tXX and tWXX) shown in the flowchart in this chapter is the symbol of characteristic item in CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS.

For each specified value, refer to CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS.
4.1 Command Frame Transmission Processing Flowchart

- Command frame transmission processing
  - Command frame header (SOH = 01H) transmission
  - Wait between data transmissions
    - Data length (LEN) transmission
    - Wait between data transmissions
    - Command number (COM) transmission
    - (LEN – 1) bytes transmitted?
      - Yes: End of command frame transmission
      - No: Transmits 1-byte command information
        - Wait between data transmissions
          - Checksum data (SUM) transmission
          - Wait between data transmissions
          - Command frame footer (ETX = 03H) transmission
          - End of command frame transmission
4.2 Data Frame Transmission Processing Flowchart

[Flowchart diagram]

- Data frame transmission processing
  - Data frame header (STX = 02H) transmission
    - Wait between data transmissions
      - LEN bytes transmitted?
        - Yes
          - Data length (LEN) transmission
        - No
          - Wait between data transmissions
            - Transmits 1-byte data
            - Wait between data transmissions
              - Checksum data (SUM) transmission
                - Wait between data transmissions
                  - Last data frame?
                    - No
                      - Transmission of footer other than those of last data frame (ETB = 17H)
                      - End of data frame transmission
                    - Yes
                      - Transmission of last data frame footer (ETX = 03H)
                      - End of data frame transmission
4.3 Data Frame Reception Processing Flowchart
4.4 Reset Command

4.4.1 Processing sequence chart

Reset command processing sequence

**Programmer 78K0/Kx2**

- **<1>** Wait from previous frame reception until next command transmission
- **<2>** Low level output (00H @ 9,600 bps)
- **<3>** Wait \( t_{12} \)
- **<4>** Low level output (00H @ 9,600 bps)
- **<5>** Wait \( t_{2c} \)
- **<6>** Reset command frame transmission
- **<7>** Time-out check for status frame reception
- **<8>** Status frame reception

**Status frame received within specified time**

- Reception status [ACK/other than ACK]
  - Other than ACK
    - Reception status [ACK/other than ACK]
      - ACK
        - Normal completion [A]
      - Other than ACK
        - Retry count over? [Yes/No]
          - Yes
            - Normal completion [A]
          - No
            - Go to <5>
      - Abnormal termination [B]
  - ACK
    - Normal completion [A]
  - Other than ACK
    - Go to <5>

**Time-out error [C]**

- Time-out occurs

**Note**  Do not exceed the retry count for the reset command transmission (up to 16 times).
4.4.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command processing starts (wait time $t_{\text{COM}}$).

<2> The low level is output (data 00H is transmitted at 9,600 bps).

<3> Wait state (wait time $t_{12}$).

<4> The low level is output (data 00H is transmitted at 9,600 bps).

<5> Wait state (wait time $t_{2C}$).

<6> The Reset command is transmitted by command frame transmission processing.

<7> A time-out check is performed from command transmission until status frame reception.

If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT0}}$).

<8> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 ≠ ACK: The retry count ($t_{\text{RS}}$) is checked.

The sequence is re-executed from <5> if the retry count is not over.
If the retry count is over, the processing ends abnormally [B].

4.4.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
4.4.4 Flowchart

- Reset command processing
- Wait from previous frame reception until next command transmission
- Transmits "00" at 9,600 bps
- Wait
- Transmits "00" at 9,600 bps
- Wait
- Command frame transmission processing (Reset)
- Status frame received?
  - Yes
    - Status = ACK?
      - Yes
        - Retry count over?
          - Yes
            - Abnormal termination [B]
          - No
            - Normal completion [A]
      - No
        - Timed out?
          - Yes
            - Error (UART)
          - No
            - Time-out error [C]
- No
4.4.5 Sample program
The following shows a sample program for Reset command processing.

```c
u16  fl_ua_reset(void)
{
    u16  rc;
    u32  retry;

    set_uart0_br(BR_9600); // change to 9600bps

    fl_wait(tCOM);  // wait
    putc ua(0x00);  // send 0x00 @ 9600bps
    fl_wait(t12); // wait
    putc ua(0x00);  // send 0x00 @ 9600bps

    for (retry = 0; retry < tRS; retry++){
        fl_wait(t2C); // wait

        put_cmd ua(FL_COM_RESET, 1, fl_cmd_prm); // send RESET command

        rc = get_sfrm ua(fl ua_sfrm, tWT0_TO);
        if (rc == FLC_DFTO_ERR) // t.o. ?
            break;      // yes // case [C]

        if (rc == FLC_ACK){   // ACK ?
            break;    // yes // case [A]
        }
        else{ NOP();
            //continue;     // case [B] (if exit from loop)
        }
    }

    // switch(rc) {
        //
        //  case  FLC_NO_ERR: return rc; break; // case [A]
        //  case  FLC_DFTO_ERR: return rc; break; // case [C]
        //  default:   return rc; break; // case [B]
        // }
    return rc;
}
```
4.5 Oscillating Frequency Set Command

4.5.1 Processing sequence chart

Oscillating Frequency Set command processing sequence

Programmer

<1> Wait from previous frame reception until next command transmission

<2> Oscillating Frequency Set command frame transmission

<3> Switching UART communication baud rate to 115,200 bps.

<4> Time-out check for status frame reception

<5> Status frame reception

78K0/Kx2

Time-out occurs

Status frame received within specified time

Reception status [ACK/other than ACK]

Other than ACK

ACK

Abnormal termination [B]

Normal completion [A]

<1> <2> <3> <4> <5>
4.5.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{\text{com}} \)).

<2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.

<3> After the status frame is received, the UART communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.

<4> A time-out check is performed from command transmission until status frame reception. If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT9}} \)).

<5> The status code is checked.

When \( ST1 = \text{ACK} \): Normal completion [A]
When \( ST1 \neq \text{ACK} \): Abnormal termination [B]

4.5.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
4.5.4 Flowchart

UART communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted. After the status frame is received, the communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.
4.5.5 Sample program

The following shows a sample program for Oscillating Frequency Set command processing.

```c
u16  fl_ua_setclk(u8 clk[])  
{  
    u16   rc;  

    fl_cmd_prm[0] = clk[0]; // "D01"
    fl_cmd_prm[1] = clk[1]; // "D02"
    fl_cmd_prm[2] = clk[2]; // "D03"
    fl_cmd_prm[3] = clk[3]; // "D04"

    fl_wait(tCOM);        // wait before sending command
    put_cmd_ua(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);

    set_flbaud(BR_115200);  // change baud-rate
    set_uart0_br(BR_115200); // change baud-rate (h.w.)

    rc = get_sfrm_ua(fl_ua_sfrm, tWT9_TO); // get status frame
    // switch(rc) {
    //     //
    //     //    case FLC_NO_ERR: return rc; break; // case [A]
    //     //    case FLC_DFTO_ERR: return rc; break; // case [C]
    //     //    default: return rc; break; // case [B]
    //    //
    //
    return rc;
    }
    
```
4.6 Chip Erase Command

4.6.1 Processing sequence chart

Chip Erase command processing sequence

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1&gt;</td>
<td>Wait from previous frame reception until next command transmission</td>
</tr>
<tr>
<td>&lt;2&gt;</td>
<td>Chip Erase command frame transmission</td>
</tr>
<tr>
<td>&lt;3&gt;</td>
<td>Time-out check for status frame reception</td>
</tr>
<tr>
<td>&lt;4&gt;</td>
<td>Status frame reception</td>
</tr>
</tbody>
</table>

- Normal completion [A]
- Time-out error [C]
- Abnormal termination [B]
- Reception status [ACK/other than ACK]
- Other than ACK
- ACK
4.6.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{com}$).

<2> The Chip Erase command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
    If a time-out occurs, a time-out error [C] is returned (time-out time $t_{wtt}$).

<4> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 $\neq$ ACK: Abnormal termination [B]

4.6.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Erase error</td>
<td>1AH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
4.6.4 Flowchart

- Chip Erase command processing
- Waits from previous frame reception until next command transmission ($t_{com}$)

- Command frame transmission processing (Chip Erase)

- Status frame received? (Yes/No)
- Timed out? ($t_{w1}$) (Yes/No)
- Status = ACK? (Yes/No)

- Normal completion [A]
- Abnormal termination [B]

- Time-out error [C]
4.6.5 Sample program

The following shows a sample program for Chip Erase command processing.

```c
/* Erase all(chip) command */
ul6 fl_ua_erase_all(void)
{
  ul6 rc;

  fl_wait(tCOM); // wait before sending command

  put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send ERASE CHIP command

  rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX); // get status frame
  // switch(rc) {
  //   // case FLC_NO_ERR: return rc; break; // case [A]
  //   // case FLC_DFTO_ERR: return rc; break; // case [C]
  //   default: return rc; break; // case [B]
  // }
  return rc;
}
```
4.7 Block Erase Command

4.7.1 Processing sequence chart

Block Erase command processing sequence

- **<1>** Wait from previous frame reception until next command transmission
- **<2>** Block Erase command frame transmission
- **<3>** Time-out check for status frame reception
- **<4>** Status frame reception

**Reception status** [ACK/other than ACK]

- **Other than ACK**
  - Abnormal termination [B]
- **ACK**
  - Normal completion [A]

**Time-out error** [C]

**Status frame received within specified time**

**Time-out occurs**
4.7.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{\text{com}} \)).

<2> The Block Erase command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT2}} \)).

<4> The status code is checked.

   When \( ST1 = \text{ACK} \): Normal completion [A]
   When \( ST1 \neq \text{ACK} \): Abnormal termination [B]

4.7.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Erase error</td>
<td>1AH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
4.7.4 Flowchart

Block Erase command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Block Erase)

Status frame received?

Status = ACK?

Abnormal termination [B]

Normal completion [A]

Timed out? Time-out error [C]

Yes No

No

Yes

No
4.7.5 Sample program

The following shows a sample program for Block Erase command processing for one block.

```c
/* *******************************************************/
/* Erase block command */
/* *******************************************************/
/* [i] u16 sblk   ... start block to erase (0...255) */
/* [i] u16 eblk   ... end block to erase   (0...255) */
/* [r] u16        ... error code */
/* *******************************************************/

u16 fl_ua_erase_blk(u16 sblk, u16 eblk)
{
    u16 rc;
    u32 wt2_max;
    u32 top, bottom;

    top = get_top_addr(sblk);  // get start address of start block
    bottom = get_bottom_addr(eblk); // get end address of end block

    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    wt2_max = make_wt2_max(sblk, eblk);

    fl_wait(tCOM);  // wait before sending command

    put_cmd_ua(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm); // send ERASE CHIP command

    rc = get_sfrm_ua(fl_ua_sfrm, wt2_max); // get status frame

    // switch(rc) {
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }

    return rc;
}
```
4.8 Programming Command

4.8.1 Processing sequence chart

Programming command processing sequence:

1. Wait from previous frame reception until next command transmission.
2. Programming command frame transmission.
3. Time-out check for status frame reception.
5. Wait from previous frame reception until next data frame transmission.
6. Data frame (user data) transmission.
7. Time-out check for status frame reception.
8. Status frame reception.
10. Status frame reception.

Abnormal termination modes:
- [B] Other than ACK
- [D] All data frames transmitted?
- [E] Normal completion
- [C] Time-out error

Status frame received within specified time:
- ACK
- Other than ACK

Time-out occurs:
- Status frame received within specified time
- Time-out error
- Normal completion

Abnormal termination:
- [B] Other than ACK
- [D] All data frames transmitted?
- [E] Normal completion

All data frames transmitted? [Yes/No]:
- Yes
- No

Time-out occurs:
- Status frame received within specified time
- Time-out error
- Normal completion

Time-out error:
- [C] Time-out

ACK

Other than ACK

Abnormal termination
4.8.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{\text{COM}}$).

<2> The Programming command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT}}$).

<4> The status code is checked.

   When ST1 = ACK: Proceeds to <5>.
   When ST1 $\neq$ ACK: Abnormal termination [B]

<5> Waits from the previous frame reception until the next data frame transmission (wait time $t_{\text{FD}} (UART)$).

<6> User data is transmitted by data frame transmission processing.

<7> A time-out check is performed from user data transmission until data frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT}}$).

<8> The status code (ST1/ST2) is checked (also refer to the processing sequence chart and flowchart).

   When ST1 $\neq$ ACK: Abnormal termination [B]
   When ST1 = ACK: The following processing is performed according to the ST2 value.
      • When ST2 = ACK: Proceeds to <9> when transmission of all data frames is completed.
        If there still remain data frames to be transmitted, the processing re-executes the
        sequence from <5>.
      • When ST2 $\neq$ ACK: Abnormal termination [D]

<9> A time-out check is performed until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT}} \times$ number of blocks).

<10> The status code is checked.

   When ST1 = ACK: Normal completion [A]
   When ST1 $\neq$ ACK: Abnormal termination [E]
### 4.8.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>Write error</td>
<td>1CH (ST2)</td>
</tr>
<tr>
<td>Abnormal termination [E]</td>
<td>MRG11 error</td>
<td>18H</td>
</tr>
</tbody>
</table>
4.8.4 Flowchart

```
        Programming command processing
          ↓
Wait from previous frame reception until next command transmission
          ↓
Command frame transmission processing (Programming)
          ↓
Status frame received?
          ↓
No                     Time out?
          ↓
No                     No                   Time-out error [C]
          ↓
Yes                    Status = ACK?
          ↓
No                     Abnormal termination [B]
          ↓
Yes
          ↓
Status frame received?
          ↓
No                     Status frame received?
          ↓
No                     Timed out?
          ↓
No                     No                   Time-out error [C]
          ↓
Yes                    Status = ACK?
          ↓
No                     Abnormal termination [B]
          ↓
Yes                    ST1 = ACK?
          ↓
No                     Abnormal termination [B]
          ↓
Yes                    ST2 = ACK?
          ↓
No                     Abnormal termination [D]
          ↓
Yes
          ↓
All data frames transmitted?
          ↓
Yes
          ↓
Abnormal termination [E]
          ↓
No                     Status frame received?
          ↓
No                     Status = ACK?
          ↓
No                     Abnormal termination [E]
          ↓
Yes                    Timed out?
          ↓
No                     No                   Time-out error [C]
          ↓
Yes                    Yes                   Abnormal termination [A]
```

```
4.8.5 Sample program
The following shows a sample program for Programming command processing.

```c
#pragma once

#define fl_st2_ua (fl_ua_sfrm[OFS_STA_PLD+1])

u16 fl_ua_write(u32 top, u32 bottom) {
    u16 rc;
    u32 send_head, send_size;
    bool is_end;
    u16 block_num;

    /******************************************************************************
    /* set params                              */
    /******************************************************************************/
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom); // get block num

    /******************************************************************************
    /* send command & check status           */
    /******************************************************************************/
    fl_wait(tCOM);  // wait before sending command
    put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT3_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [B]
    }

    /******************************************************************************
    /* send user data                       */
    ******************************************************************************/
    send_head = top;
```
while(1){
    // make send data frame
    if ((bottom - send_head) > 256){  // rest size > 256?
        is_end = false;  // yes, not is_end frame
        send_size = 256;  // transmit size = 256 byte
    }
    else{
        is_end = true;
        send_size = bottom - send_head + 1;  // transmit size = (bottom -
            // send_head)+1 byte
    }
    memcpy(fl_txdata_frm, rom_buf+send_head, send_size);  // set data frame
                // payload
    send_head += send_size;

    fl_wait(tFD3_UA);  // wait before sending data frame

    put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

    rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX);  // get status frame
    switch(rc) {
        case FLC_NO_ERR: break;  // continue
        case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }
    if (fl_st2_ua != FLST_ACK){  // ST2 = ACK?
        rc = decode_status(fl_st2_ua); // No
        return rc;  // case [D]
    }
    if (is_end)
        break;
}

/*********************************************************
/* Check internally verify                                  */
/***********************************************************/
rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX * block_num));  // get status frame again
// switch(rc) {
    // case FLC_NO_ERR: return rc; break; // case [A]
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    // default: return rc; break; // case [E]
// }
return rc;
}
4.9 Verify Command

4.9.1 Processing sequence chart

Verify command processing sequence
4.9.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{COM} \)).

<2> The Verify command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{WT6} \)).

<4> The status code is checked.
   - When \( ST1 = \text{ACK} \): Proceeds to <5>.
   - When \( ST1 \neq \text{ACK} \): Abnormal termination [B]

<5> Waits from the previous frame reception until the next data frame transmission (wait time \( t_{FD3} \text{(UART)} \)).

<6> User data for verifying is transmitted by data frame transmission processing.

<7> A time-out check is performed from user data transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{WT7} \)).

<8> The status code (ST1/ST2) is checked (also refer to the processing sequence chart and flowchart).
   - When \( ST1 \neq \text{ACK} \): Abnormal termination [B]
   - When \( ST1 = \text{ACK} \): The following processing is performed according to the ST2 value.
     - When \( ST2 = \text{ACK} \): If transmission of all data frames is completed, the processing ends normally [A].
       If there still remain data frames to be transmitted, the processing re-executes the sequence from <5>.
     - When \( ST2 \neq \text{ACK} \): Abnormal termination [D]

4.9.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>Verify error</td>
<td>0FH (ST2)</td>
</tr>
</tbody>
</table>
4.9.4 Flowchart

Verify command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Verify)

Status frame received?

Yes

No

ST1 = ACK?

Yes

ST2 = ACK?

Abnormal termination [B]

Abnormal termination [B]

No

Timed out?

Yes

Time-out error [C]

No

Wait from previous frame reception until next data frame transmission

Data frame transmission processing (User program)

Status frame received?

Yes

No

Timed out?

Yes

Time-out error [C]

No

ST1 = ACK?

No

Yes

Abnormal termination [B]

ST2 = ACK?

No

Yes

Abnormal termination [D]

All data frames transmitted?

No

Yes

Normal completion [A]
4.9.5 Sample program
The following shows a sample program for Verify command processing.

```c
u16 fl_ua_verify(u32 top, u32 bottom)
{
  u16 rc;
  u32 send_head, send_size;
  bool is_end;

  /* set params */
  set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

  /* send command & check status */
  fl_wait(tCOM); // wait before sending command
  put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm); // send VERIFY command
  rc = get_sfrm_ua(fl ua_sfrm, tWT6_TO); // get status frame
  switch(rc) {
    case FLC_NO_ERR: break; // continue
  //    case FLC_DFTO_ERR: return rc; break; // case [C]
    default: return rc; break; // case [B]
  }

  /* send user data */
  send_head = top;
  while(1){
    // make send data frame
    if ((bottom - send_head) > 256){ // rest size > 256 ?
```
is_end = false;  // yes, not is_end frame
send_size = 256;  // transmit size = 256 byte
}
else{
    is_end = true;
    send_size = bottom - send_head + 1;  // transmit size = (bottom - send_head)+1 byte
}
memcpy(fl_txdata_frm, rom_buf+send_head, send_size);  // set data frame
    // payload
send_head += send_size;

fl_wait(tFD3_UA);
put_dfrm_ua(send_size, fl_txdata_frm, is_end);  // send user data
c = get_sfrm_ua(fl_ua_sfrm, tWT7_TO);  // get status frame
switch(rc) {
    case FLC_NO_ERR:  break;  // continue

    case FLC_DFTO_ERR:  return rc;  break;  // case [C]

    default:  return rc;  break;  // case [B]
}
if (fl_st2_ua != FLST_ACK) {  // ST2 = ACK ?
    rc = decode_status(fl_st2_ua);  // No
    return rc;  // case [D]
}
if (is_end)  // send all user data ?
    break;  // yes
    //continue;
}
return FLC_NO_ERR;  // case [A]
4.10 Block Blank Check Command

4.10.1 Processing sequence chart

Block Blank Check command processing sequence
4.10.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{COM}$).

<2> The Block Blank Check command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT8} \times$ number of blocks).

<4> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 $\neq$ ACK: Abnormal termination [B]

4.10.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>MRG11 error</td>
<td>18H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
4.10.4 Flowchart

- Block Blank Check command processing
- Wait from previous frame reception until next command transmission
- Command frame transmission processing (Block Blank Check)
- Status frame received?
  - Yes
    - Status = ACK?
      - Yes: Normal completion [A]
      - No
        - Timed out?
          - Yes: Time-out error [C]
          - No
            - tcom
            - × number of blocks

- Abnormal termination [B]
4.10.5 Sample program

The following shows a sample program for Block Blank Check command processing.

```c
/**
 * Block blank check command
 */
/**
 * [i] u32 top     ... start address
 * [i] u32 bottom ... end address
 * [r] u16         ... error code
 */

u16 fl_ua_blk_blank_chk(u32 top, u32 bottom)
{
    u16 rc;
    u16 block_num;

    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom); // get block num

    fl_wait(tCOM);  // wait before sending command

    put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num); // get status frame

    // switch(rc) {
    //         //
    //         // case FLC_NO_ERR: return rc; break; // case [A]
    //         // case FLC_DFTO_ERR: return rc; break; // case [C]
    //         // default:        return rc; break; // case [B]
    //    // }

    return rc;
}
```
4.11 Silicon Signature Command

4.11.1 Processing sequence chart

Silicon Signature command processing sequence
4.11.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{com} \)).

<2> The Silicon Signature command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
    If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{WT1} \)).

<4> The status code is checked.

    When \( ST1 = \text{ACK} \): Proceeds to <5>.
    When \( ST1 \neq \text{ACK} \): Abnormal termination [B]

<5> A time-out check is performed until data frame (silicon signature data) reception.
    If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{FD2} \)).

<6> The received data frame (silicon signature data) is checked.

    If data frame is normal: Normal completion [A]
    If data frame is abnormal: Data frame error [D]

4.11.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H The command was executed normally and the silicon signature was acquired normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H The checksum of the transmitted command frame is abnormal.</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td></td>
<td>Read error</td>
<td>20H Reading of security information failed.</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>Time-out error of status frame reception or data frame reception has occurred.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as silicon signature data is abnormal.</td>
</tr>
</tbody>
</table>
4.11.4 Flowchart

Silicon Signature command processing

Wait from previous frame reception until next command transmission

t\text{com}

Command frame transmission processing (Silicon Signature)

Status frame received?

Yes

Timed out?

No

Yes

Time-out error [C]

No

Status = ACK?

Yes

Abnormal termination [B]

No

Data frame (silicon signature) received?

Yes

Timed out?

No

Yes

Time-out error [C]

No

Normal data frame?

Yes

Data frame error [D]

Normal completion [A]

No

Timed out?

No

Yes
4.11.5 Sample program

The following shows a sample program for Silicon Signature command processing.

```c
u16  fl_ua_getsig(u8 *sig)
{
    u16  rc;

    fl_wait(tCOM);  // wait before sending command

    put_cmd_ua(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm); // send GET SIGNATURE command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT11_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        default:  return rc; break; // case [B]
    }

    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO);  // get status frame
    if (rc){                             // if error
        return rc;                      // case [D]
    }

    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]); // copy Signature data

    return rc;                           // case [A]
}
```
4.12 Version Get Command

4.12.1 Processing sequence chart

Version Get command processing sequence

Programmer 78K0/Kx2

<1> Wait from previous frame reception until next command transmission

<2> Version Get command frame transmission

<3> Time-out check for status frame reception

<4> Status frame reception

<5> Time-out check for data frame reception

<6> Data frame (version data) reception

Data frame received within specified time

Status frame received within specified time

ACK

Other than ACK

Abnormal termination [B]

Time-out [C]

Data frame error [D]

Normal data frame? [Yes/No]

Yes

Normal completion [A]

No

Time-out occurs

Time-out occurs

Time-out occurs
4.12.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{\text{com}} \)).

<2> The Version Get command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT12}} \)).

<4> The status code is checked.

   When ST1 = ACK: Proceeds to <5>.
   When ST1 ≠ ACK: Abnormal termination [B]

<5> A time-out check is performed until data frame (version data) reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD2}} \)).

<6> The received data frame (version data) is checked.

   If data frame is normal: Normal completion [A]
   If data frame is abnormal: Data frame error [D]

4.12.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A] (ACK)</td>
<td>06H</td>
<td>The command was executed normally and version data was acquired normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>07H</td>
<td>The checksum of the transmitted command frame is abnormal.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>Time-out error of status frame reception or data frame reception has occurred.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data is abnormal.</td>
</tr>
</tbody>
</table>
4.12.4 Flowchart

- Version Get command processing
- Wait from previous frame reception until next command transmission
- Command frame transmission processing (Version Get)
  - Status frame received?
    - Yes
      - Status = ACK?
        - Yes
          - Normal completion [A]
        - No
          - Data frame (version data) received?
            - Yes
              - Normal data frame?
                - Yes
                  - Time-out error [C]
                - No
                  - Data frame error [D]
            - No
              - Time-out error [C]
    - No
      - Timed out?
        - Yes
          - Time-out error [C]
        - No
          - Abnormal termination [B]
- No
  - Timed out?
    - Yes
      - Time-out error [C]
    - No
      - No
4.12.5 Sample program

The following shows a sample program for Version Get command processing.

```c
/**************************************************************************/
/*                                                                      */
/*  Get device/firmware version command                                 */
/*                                                                      */
/**************************************************************************/
/*  [i]  u8  *buf       ... pointer to version date save area          */
/*  [r]  u16           ... error code                                */
/**************************************************************************/

u16  fl_ua_getver(u8 *buf)  
{
  u16  rc;

  fl_wait(tCOM);  // wait before sending command

  put_cmd_ua(FL_COM_GET_VERSION, 1, fl_cmd_prm);  // send GET VERSION command

  rc = get_sfrm_ua(fl_ua_sfrm, tWT12_TO);    // get status frame
  switch(rc) {
    case FLC_NO_ERR:   break;   // continue
    // case FLC_DFTO_ERR: return rc;  break;  // case [C]
    default:  return rc;  break;   // case [B]
  }

  rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO);    // get data frame
  if (rc)
  {
    return rc;    // case [D]
  }

  memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
  return rc;    // case [A]
}
```
4.13 Checksum Command

4.13.1 Processing sequence chart

Checksum command processing sequence
4.13.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{\text{com}} \)).

<2> The Checksum command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception. If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT16}} \)).

<4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.
When ST1 \( \neq \) ACK: Abnormal termination [B]

<5> A time-out check is performed until data frame (checksum data) reception. If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD1}} \)).

<6> The received data frame (checksum data) is checked.

If data frame is normal: Normal completion [A]
If data frame is abnormal: Data frame error [D]

4.13.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td>Checksum error</td>
<td>07H</td>
<td>The checksum of the transmitted command frame is abnormal.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>Time-out error of status frame reception or data frame reception has occurred.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data is abnormal.</td>
</tr>
</tbody>
</table>
4.13.4 Flowchart

- Checksum command processing
  - Wait from previous frame reception until next command transmission
  - Command frame transmission processing (Checksum)
  - Status frame received?
    - Yes, Status = ACK?
      - Yes, Data frame (checksum data) received?
        - Yes, Normal data frame?
          - Yes, Normal completion [A]
          - No, Abnormal termination [B]
        - No, Data frame error [D]
      - No, Abnormal termination [B]
    - No, Data frame error [D]
  - No, Timed out? [tWT16]
  - No, Time-out error [C]
4.13.5 Sample program

The following shows a sample program for Checksum command processing.

```c
/**************************************************************************/
/*                                                                  */
/* Get checksum command                                              */
/*                                                                  */
/**************************************************************************/
/* [i] u16 *sum    ... pointer to checksum save area */
/* [i] u32 top     ... start address */
/* [i] u32 bottom   ... end address */
/* [r] u16         ... error code */
/**************************************************************************/

u16 fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16 rc;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    // send command
    fl_wait(tCOM); // wait before sending command
    put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send GET VERSION command

    rc = get_sfrm_ua(fl_rxdata_frm, tWT16_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

    // get data frame (Checksum data)
    rc = get_dfrm_ua(fl_rxdata_frm, tFD1_TO); // get status frame
    if (rc){
        // if no error,
        return rc; // case [D]
    }

    *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1]; // set SUM data
    return rc; // case [A]
}
```
4.14 Security Set Command

4.14.1 Processing sequence chart

Security Set command processing sequence

- **<1>** Wait from previous frame reception until next command transmission
- **<2>** Security Set command frame transmission
- **<3>** Time-out check for status frame reception
- **<4>** Status frame reception
- **<5>** Wait from previous frame reception until data frame transmission
- **<6>** Data frame (security data) transmission
- **<7>** Time-out check for status frame reception
- **<8>** Status frame reception
- **<9>** Time-out check for status frame reception
- **<10>** Status frame reception
- **<11>** Wait from previous frame reception until next command transmission

- **Abnormal termination [B]**
- **Normal completion [A]**
- **Other than ACK**
- **Time-out [C]**
- **Abnormal termination [D]**
- **Other than ACK**
- **Status frame received within specified time**
- **Time-out error [C]**
- **Status frame received within specified time**
- **Time-out error [D]**

**Programmer**

**78K0/Kx2**
4.14.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{\text{com}}$).

<2> The Security Set command is transmitted by command frame transmission processing.

<3> A time-out check is performed from command transmission until status frame reception. If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT13}}$).

<4> The status code is checked.

   When ST1 = ACK: Proceeds to <5>.
   When ST1 $\neq$ ACK: Abnormal termination [B]

<5> Waits from the previous frame reception until the next data frame transmission (wait time $t_{\text{FD3(UART)}}$).

<6> The data frame (security setting data) is transmitted by data frame transmission processing.

<7> A time-out check is performed until status frame reception. If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT14}}$).

<8> The status code is checked.

   When ST1 = ACK: Proceeds to <9>.
   When ST1 $\neq$ ACK: Abnormal termination [D]

<9> A time-out check is performed until status frame reception. If a time-out occurs, a time-out error [C] is returned (time-out time $t_{\text{WT15}}$).

<10> The status code is checked.

   When ST1 = ACK: Normal completion [A]
   When ST1 $\neq$ ACK: Abnormal termination [E]

4.14.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>Time-out error of status frame reception or data frame reception has occurred.</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>FLMD error</td>
<td>18H</td>
</tr>
<tr>
<td></td>
<td>Write error</td>
<td>1CH</td>
</tr>
<tr>
<td>Abnormal termination [E]</td>
<td>MRG11 error</td>
<td>1BH</td>
</tr>
</tbody>
</table>
4.14.4 Flowchart

Security Set command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Security Set)

Status frame received?

Yes

Timed out?

No

Status = ACK?

No

Abnormal termination [B]

Yes

Wait from previous frame reception until next data frame transmission

Data frame transmission processing (Security data)

Status frame received?

Yes

Timed out?

No

Status = ACK?

No

Abnormal termination [D]

Yes

Abnormal termination [E]

Status = ACK?

No

Time-out error [C]

Yes

Normal completion [A]

Abnormal termination [E]
4.14.5 Sample program
The following shows a sample program for Security Set command processing.

```c
u16 fl_ua_setscf(u8 scf)
{
    u16 rc;

    fl_cmd_prm[0] = 0x00;   // "BLK" (must be 0x00)
    fl_cmd_prm[1] = 0x00;   // "PAG" (must be 0x00)
    fl_txdata_frm[0] = (scf |= 0b11101000);  // "FLG" (bit7, 6, 5, 3 must be '1' (to make sure))
    fl_txdata_frm[1] = 0x03;  // "BOT" (fixed 0x03)

    fl_wait(tCOM);  // wait before sending command
    put_cmd_ua(FL_COM_SET_SECURITY, 3, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT13_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        default:  return rc; break; // case [B]
    }

    fl_wait(tFD3_UA);
```
put_dfrm_ua(2, fl_txdata_frm, true);  // send security setting(FLAG) & BOT data

rc = get_sfrm_ua(fl_uid_sfrm, tWT14_MAX);  // get status frame
switch(rc) {
    case FLC_NO_ERR: break; // continue
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    default: return rc; break; // case [B]
}

/************************************************/
/*     Check internally verify                  */
/************************************************/
rc = get_sfrm_ua(fl_uid_sfrm, tWT15_MAX);  // get status frame
switch(rc) {
    // case FLC_NO_ERR: return rc; break; // case [A]
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    // default: return rc; break; // case [B]
    
    return rc;
}
CHAPTER 5  3-WIRE SERIAL I/O COMMUNICATION MODE (CSI)

Each of the symbol (tXX and tWXX) shown in the flowchart in this chapter is the symbol of characteristic item in CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS.

For each specified value, refer to CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS.
5.1 Command Frame Transmission Processing Flowchart

Command frame transmission processing

Command frame header (SOH = 01H) transmission

Wait between data transmissions

Data length (LEN) transmission

Wait between data transmissions

Command number (COM) transmission

(LEN - 1) bytes transmitted?

Yes

No

Wait between data transmissions

Transmits 1-byte parameter

Wait between data transmissions

Checksum data (SUM) transmission

Wait between data transmissions

Command frame footer (ETX = 03H) transmission

End of command frame transmission
5.2 Data Frame Transmission Processing Flowchart

Data frame transmission processing

Data frame header (STX = 02H) transmission

Wait between data transmissions

Data length (LEN) transmission

LEN bytes transmitted?

Yes

No

Wait between data transmissions

Transmits 1-byte data

Wait between data transmissions

Checksum data (SUM) transmission

Wait between data transmissions

Last data frame?

Yes

No

Last data frame footer (ETX = 03H) transmission

Transmission of footer other than those of last data frame (ETB = 17H)

End of data frame transmission
5.3 Data Frame Reception Processing Flowchart

Data frame reception processing

Data frame header (STX = 02H) reception

Wait between data receptions

Data length (LEN) reception

Wait between data receptions

Receives 1-byte data

LEN bytes received?

Yes

Wait between data receptions

Checksum data (SUM) reception

Wait between data receptions

Reception of last data frame footer (ETX = 03H) or footer other than those of last data frame (ETB = 17H)

Checksum error?

Yes

End of data frame reception

No

Checksum error
5.4 Status Command

5.4.1 Processing sequence chart

Status command processing sequence

Note  Application specifications differ according to execution command.
5.4.2 Description of processing sequence

1. The Status command is transmitted by command frame transmission processing.
2. Waits from command transmission until status frame reception (wait time \( t_{SF} \)).
3. The status code is checked.

   - When ST1 = ACK: Normal completion [A]
   - When ST1 = BUSY: A time-out check is performed (\( t_{WTXX} \) MAX).
     If the processing is not timed out, the sequence is re-executed from 1.
     If a time-out occurs, a time-out error [C] is returned.
   - When ST1 \( \neq \) ACK, BUSY: Abnormal termination [B]

Note Application specifications differ according to execution command.

5.4.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Command error</td>
<td>04H</td>
</tr>
<tr>
<td></td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Verify error</td>
<td>0FH</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Read error</td>
<td>20H</td>
</tr>
<tr>
<td></td>
<td>MRG10 error</td>
<td>1AH</td>
</tr>
<tr>
<td></td>
<td>MRG11 error</td>
<td>1BH</td>
</tr>
<tr>
<td></td>
<td>Write error</td>
<td>1CH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>After command transmission, the specified time has elapsed but a BUSY response is still returned.</td>
</tr>
</tbody>
</table>
5.4.4 Flowchart

Note  Application specifications differ according to execution command.
5.4.5 Sample program

The following shows a sample program for Status command processing.

```c
#ifndef _FL_H
#define _FL_H

#define FL_COM_GET_STA 0x06
#define FL_COM_STS     0x01
#define FL_COM_WRITE   0x02
#define FL_COM_READ    0x03
#define FL_COM_IOCTL   0x04
#define FL_COM_END     0x08
#define FL_COM_LIGHT   0x10
#define FL_COM_CTRL    0x20
#define FL_COM_TEST    0x40
#define FL_COM_TIME    0x80

#define FLC_BUSY       0x01
#define FLC_NO_ERR     0x02
#define FLC_DFTO_ERR   0x04

#define FLST_BUSY     0x01
#define FLST_NO_ERR    0x02
#define FLST_DFTO_ERR  0x04

#endif
```

```c
static u16 fl_csi_getstatus(u32 limit)
{
    u16 rc;

    start_flto(limit);

    while(1){

        put_cmd_csi(FL_COM_GET_STA, 1, fl_cmd_prm); // send "Status" command
        fl_wait(tSF); // wait

        rc = get_sfrm_csi(fl_rxdata_frm);  // get status frame

        switch(rc){
            case FLC_BUSY:
                if (check_flto()) // time out ?
                    return FLC_DFTO_ERR; // Yes, time-out // case [C]
                continue; // No, retry

                default: // checksum error
                    return rc;

            case FLC_NO_ERR: // no error
                break;

        }

        if (fl_st1 == FLST_BUSY){ // ST1 = BUSY
            if (check_flto()) // time out ?
                return FLC_DFTO_ERR; // Yes, time-out // case [C]
            continue; // No, retry
        }

        break; // ACK or other error (but BUSY)
    }
}
```
rc = decode_status(fl_st1);  // decode status to return code
// switch(rc) {
//  
// case FLC_NO_ERR: return rc; break; // case [A]
// default: return rc; break; // case [B]
// }
return rc;
}
5.5 Reset Command

5.5.1 Processing sequence chart

Reset command processing sequence

**Note** Do not exceed the retry count for the reset command transmission (up to 16 times).
5.5.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{COM} \)).
<2> The Reset command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time \( t_{WT0} \)).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
When the processing ends abnormally: The sequence is re-executed from <1> if the retry count is not over.
If the retry count is over, the processing ends abnormally [B].
When a time-out error occurs: A time-out error [C] is returned.

5.5.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>Status check processing terminated with time-out.</td>
</tr>
</tbody>
</table>
5.5.4 Flowchart

Reset command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Reset)

Wait from command frame transmission until status check

Status check processing

Result of status check processing = Abnormal termination?

Yes

Retry count over?

Yes

Abnormal termination [B]

No

Result of status check processing = Time-out error?

Yes

Normal completion [A]

No (normal completion)

Result of status check processing = Abnormal termination?

No

Result of status check processing = Time-out error?

Yes

Time-out error [C]

No (normal completion)
5.5.5 Sample program
The following shows a sample program for Reset command processing.

```
/***********************_RESET COMMAND (CSI)******************************/
/               */
/* Reset command (CSI) */
/*               */
/***********************_RESET COMMAND (CSI)******************************/
/* [r] u16 ... error code */
/***********************_RESET COMMAND (CSI)******************************/

u16  fl_csi_reset(void)
{
    u16  rc;
    u32  retry;

    for (retry = 0; retry < tRS; retry++){
        fl_wait(tCOM);  // wait before sending command frame
        put_cmd_csi(FL_COM_RESET, 1, fl_cmd_prm);  // send "Reset" command frame
        fl_wait(tWT0);
        rc = fl_csi_getstatus(tWT0_TO);  // get status

        if (rc == FLC_DFTO_ERR)  // timeout error ?
            break;  // yes // case [C]
        if (rc == FLC_ACK)  // Ack ?
            break;  // yes // case [A]
            //continue;  // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}
```
5.6 Oscillating Frequency Set Command

Execution of this command is not necessary during CSI communication (if execution of this command is required during CSI communication according to the programmer specifications, set the frequency to 8 MHz).

5.6.1 Processing sequence chart
5.6.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{COM}$).
<2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time $t_{WT9}$).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
When the processing ends abnormally: Abnormal termination [B]
When a time-out error occurs: A time-out error [C] is returned.

5.6.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>06H</td>
<td>Normal acknowledgment (ACK) The command was executed normally and the operating frequency was correctly set to the 78K0/Kx2.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>05H</td>
<td>Parameter error The oscillation frequency value is out of range.</td>
</tr>
<tr>
<td></td>
<td>07H</td>
<td>Checksum error The checksum of the transmitted command frame is abnormal.</td>
</tr>
<tr>
<td></td>
<td>15H</td>
<td>Negative acknowledgment (NACK) Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
5.6.4 Flowchart

Oscillating Frequency Set command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Oscillating Frequency Set)

Wait from command frame transmission until status check

Status check processing

Time-out error?

Yes

Time-out error [C]

No

Normal completion?

Yes

Normal completion [A]

No

Abnormal termination [B]
5.6.5 Sample program

The following shows a sample program for Oscillating Frequency Set command processing.

```c
/* **************************************************************************
/*                                                             */
/* Set Flash device clock value command (CSI)                   */
/*                                                             */
/* **************************************************************************/
/* [i] u8 clk[4]   ... frequency data(D1-D4)                    */
/* [r] u16         ... error code                               */
/* **************************************************************************/

u16  fl_csi_setclk(u8 clk[])
{
    u16  rc;

    fl_cmd_prm[0] = clk[0]; // "D01"
    fl_cmd_prm[1] = clk[1]; // "D02"
    fl_cmd_prm[2] = clk[2]; // "D03"
    fl_cmd_prm[3] = clk[3]; // "D04"

    fl_wait(tCOM);       // wait before sending command frame

    put_cmd_csi(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);
    // send "Oscillation Frequency Set" command

    fl_wait(tWT9);

    rc = fl_csi_getstatus(tWT9_TO); // get status frame
    // switch(rc) {
    //   //
    //   //     case FLC_NO_ERR: return rc; break; // case [A]
    //   //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //   default: return rc; break; // case [B]
    //   //
    // }
    return rc;
}
```
5.7 Chip Erase Command

5.7.1 Processing sequence chart

Chip Erase command processing sequence

<1> Wait from previous frame reception until next command transmission

<2> Chip Erase command frame transmission

<3> Wait from command frame transmission until status check

<4> Status check processing

<5> Result of status check processing

Result [Normal completion/ Abnormal termination/ Time-out error]

Time-out error

Abnormal termination

Normal completion

Time-out error [C]

Abnormal termination [B]

Normal completion [A]
5.7.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{\text{com}} \)).

<2> The Chip Erase command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time \( t_{\text{wt1}} \)).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
When the processing ends abnormally: Abnormal termination [B]
When a time-out error occurs: A time-out error [C] is returned.

5.7.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Erase error</td>
<td>1AH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>

Application Note U17739EJ3V0AN 121
5.7.4 Flowchart

Chip Erase command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Chip Erase)

Wait from command frame transmission until status check

Status check processing

Time-out error?

No

Normal completion?

Yes

Normal completion [A]

No

Time-out error [C]

Yes

Abnormal termination [B]
5.7.5 Sample program

The following shows a sample program for Chip Erase command processing.

```c
/* ******************************************************************
/* Erase all(chip) command (CSI)                                 */
/* ******************************************************************
/* [r] u16          ... error code                              */
/* ******************************************************************

u16 fl_csi_erase_all(void)
{
  u16 rc;

  fl_wait(tCOM); // wait before sending command frame

  put_cmd_csi(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send "Chip Erase" command

  fl_wait(tWT1);

  rc = fl_csi_getstatus(tWT1_MAX); // get status frame

  switch(rc) {
    //
    //    case FLC_NO_ERR: return rc; break; // case [A]
    //    case FLC_DFTO_ERR: return rc; break; // case [C]
    //    default: return rc; break; // case [B]
    //  }
    return rc;

} 
```
5.8 Block Erase Command

5.8.1 Processing sequence chart

Block Erase command processing sequence

- **<1>** Wait from previous frame reception until next command transmission
- **<2>** Block Erase command frame transmission
- **<3>** Wait from command frame transmission until status check
- **<4>** Status check processing
- **<5>** Result of status check processing

Result:
- Normal completion
- Abnormal termination
- Time-out error

Time-out error [C]

Normal completion [A]

Abnormal termination [B]
5.8.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{com} \)).
<2> The Block Erase command is transmitted by command frame transmission processing.
<3> Waits until status frame acquisition (wait time \( t_{WT2} \)).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
When the processing ends abnormally: Abnormal termination [B]
When a time-out error occurs: A time-out error [C] is returned.

5.8.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Erase error</td>
<td>1AH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
5.8.4 Flowchart

Block Erase command processing

Wait from previous frame reception until next command transmission  \( t_{\text{COM}} \)

Command frame transmission processing (Block Erase)

Wait from command frame transmission until status check  \( t_{\text{WT2}} \)

Status check processing

Time-out error? Yes

No

Normal completion? Yes

Abnormal termination [B]

Time-out error [C]

Normal completion [A]
5.8.5 Sample program

The following shows a sample program for Block Erase command processing.

```c
/* *************************************************************************/
/*                                                                    */
/* Erase block command (CSI)                                         */
/*                                                                    */
/* *************************************************************************/
/* [i] u16 sblk   ... start block to erase (0...255)                  */
/* [i] u16 eblk  ... end block to erase   (0...255)                    */
/* [r] u16       ... error code                                        */
/* *************************************************************************/

u16  fl_csi_erase_blk(u16 sblk, u16 eblk)
{
    u16  rc;
    u32  wt2, wt2_max;
    u32  top, bottom;

    top = get_top_addr(sblk); // get start address of start block
    bottom = get_bottom_addr(eblk); // get end address of end block

    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    wt2 = make_wt2(sblk, eblk);
    wt2_max = make_wt2_max(sblk, eblk);

    fl_wait(tCOM);    // wait before sending command frame

    put_cmd_csi(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm);    // send "Block Erase" command

    fl_wait(wt2);

    rc = fl_csi_getstatus(wt2_max); // get status frame
    switch(rc) {
        case FLC_NO_ERR: return rc; break; // case [A]
        case FLC_DFTO_ERR: return rc; break; // case [C]
        default:            return rc; break; // case [B]
    }
    return rc;
}
```
5.9 Programming Command

5.9.1 Processing sequence chart

Programmer 78K0/Kx2

Programming command processing sequence

1. Wait from previous frame reception until next command transmission
2. Programming command frame transmission
3. Wait from command frame transmission until status check
4. Status check processing
5. Result of status check processing

<8> Wait during status check

<9> Status check processing

<10> Result of status check processing (ST1/ST2)

<11> Wait during status check (internal verify) TWRx x number of blocks

<12> Status check processing

<13> Result of status check processing

All data frames transmitted? [Yes/No]

ACK Other than ACK

Abnormal termination

Normal completion

Reception status (ST1) [Normal completion/ Abnormal termination/ Time-out error]

Reception status (ST2) [ACK/other than ACK]

Go to <11>
5.9.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{COM}$).

<2> The Programming command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time $t_{WT3}$).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

- When the processing ends normally: Proceeds to <6>.
- When the processing ends abnormally: Abnormal termination [B]
- When a time-out error occurs: A time-out error [C] is returned.

<6> Waits until the next data frame transmission (wait time $t_{FD3}$).

<7> User data to be written to the 78K0/Kx2 flash memory is transmitted by data frame transmission processing.

<8> Waits from data frame (user data) transmission until status check processing (wait time $t_{WT4}$).

<9> The status frame is acquired by status check processing.

<10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

- When ST1 = abnormal termination: Abnormal termination [B]
- When ST1 = time-out error: A time-out error [C] is returned.
- When ST1 = normal completion: The following processing is performed according to the ST2 value.
  - When ST2 $\neq$ ACK: Abnormal termination [D]
  - When ST2 = ACK: Proceeds to <11> when transmission of all of the user data is completed.

  If there still remain user data to be transmitted, the processing re-executes the sequence from <6>.

<11> Waits until status check processing (time-out time $t_{WT5} \times$ number of blocks).

<12> The status frame is acquired by status check processing.

<13> The following processing is performed according to the result of status check processing.

- When the processing ends normally: Normal completion [A]
  (indicating that the internal verify check has performed normally after completion of write)

- When the processing ends abnormally: Abnormal termination [E]
  (indicating that the internal verify check has not performed normally after completion of write)

- When a time-out error occurs: A time-out error [C] is returned.
5.9.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>06H</td>
<td>The command was executed normally and the user data was written normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>05H</td>
<td>The specified start/end address is out of the flash memory range, or is not a multiple of 8.</td>
</tr>
<tr>
<td>Checksum error</td>
<td>07H</td>
<td>The checksum of the transmitted command frame is abnormal.</td>
</tr>
<tr>
<td>Protect error</td>
<td>10H</td>
<td>Write is prohibited by the security setting.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>1CH (ST2)</td>
<td>A write error has occurred.</td>
</tr>
<tr>
<td>Abnormal termination [E]</td>
<td>1BH</td>
<td>An internal verify error has occurred.</td>
</tr>
</tbody>
</table>
5.9.4 Flowchart
5.9.5 Sample program

The following shows a sample program for Programming command processing.

```c
u16 fl_csi_write(u32 top, u32 bottom)
{
    u16 rc;
    u32 send_head, send_size;
    bool is_end;
    u16 block_num;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom); // get block num

    fl_wait(tCOM);
    put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command
    fl_wait(tWT3);

    rc = fl_csi_getstatus(tWT3_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

    // send user data
    send_head = top;

    while(1){
        if ((bottom - send_head) > 256){ // rest size > 256 ?
            is_end = false; // yes, not end frame
            send_size  = 256; // transmit size = 256 byte
```

```c
```
} } else{
    is_end = true;
    send_size = bottom - send_head + 1;
    // transmit size = (bottom - send_head)+1 byte
}

memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
    // set data frame payload
send_head += send_size;

fl_wait(tFD3_CSI); // wait before sending data frame
put_dfrm_csi(send_size, fl_txdata_frm, is_end);
    // send data frame (user data)
fl_wait(tWT4);    // wait

rc = fl_csi_getstatus(tWT4_MAX);     // get status frame
switch(rc) {
   case FLC_NO_ERR: break; // continue
   // case FLC_DFTO_ERR: return rc; break; // case [C]
   default: return rc; break; // case [B]
}
    // fl_st2 != FLST_ACK){  // ST2 = ACK ?
   if (fl_st2 != FLST_ACK){
      rc = decode_status(fl_st2); // No
      return rc;  // case [D]
   }
    //}

if (is_end)   // send all user data ?
    break;   // yes
    // continue;
}

/******************************************
/* Check internally verify                */
/******************************************/
fl_wait(tWT5 * block_num);   // wait
rc = fl_csi_getstatus(tWT5_MAX * block_num);  // get status frame
switch(rc) {
   // case FLC_NO_ERR: return rc; break; // case [A]
   // case FLC_DFTO_ERR: return rc; break; // case [C]
   // default: return rc; break; // case [E]
   // }
   return rc;
}


5.10 Verify Command

5.10.1 Processing sequence chart

Verify command processing sequence

Programmer

1. Wait from previous frame reception until next command transmission

2. Verify command frame transmission

3. Wait from command frame transmission until status check

4. Status check processing

5. Result of status check processing

78K0/Kx2

6. Wait from previous frame reception until next command transmission

7. Data frame (user data for verify) transmission

8. Wait during status check (internal verify)

9. Status check processing

10. Result of status check processing (ST1/ST2)

Result

[Normal completion/Abnormal termination/Time-out error]

Abnormal termination [B]

Time-out error [C]

Abnormal termination

Normal completion

Time-out error

Reception status (ST1) [Normal completion/Abnormal termination/Time-out error]

Reception status (ST2) [ACK/other than ACK]

All data frames transmitted? [Yes/No]

ACK

Go to 6

No

Normal completion [A]

Other than ACK

Abnormal termination (D)

ACK

Yes

Normal completion [A]

Time-out error [C]
5.10.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{\text{COM}}$).
<2> The Verify command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time $t_{\text{WT6}}$).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

- When the processing ends normally: Proceeds to <6>.
- When the processing ends abnormally: Abnormal termination [B]
- When a time-out error occurs: A time-out error [C] is returned.

<6> Waits from the previous frame reception until the next data frame transmission (wait time $t_{\text{FD3}}$).
<7> User data for verifying is transmitted by data frame transmission processing.
<8> Waits from data frame transmission until status check processing (wait time $t_{\text{WT7}}$).
<9> The status frame is acquired by status check processing.
<10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

- When ST1 = abnormal termination: Abnormal termination [B]
- When ST1 = time-out error: A time-out error [C] is returned.
- When ST1 = normal completion: The following processing is performed according to the ST2 value.
  - When ST2 $\neq$ ACK: Abnormal termination [D]
  - When ST2 = ACK: If transmission of all data frames is completed, the processing ends normally [A].
    If there still remain data frames to be transmitted, the processing re-executes the sequence from <6>.

5.10.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>06H</td>
<td>The command was executed normally and the verify was completed normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>05H</td>
<td>The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.</td>
</tr>
<tr>
<td>Parameter error</td>
<td>07H</td>
<td>The checksum of the transmitted command frame or data frame is abnormal.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>0FH (ST2)</td>
<td>The verify has failed, or another error has occurred.</td>
</tr>
</tbody>
</table>
5.10.4 Flowchart

Verify command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Verify)

Wait from command frame transmission until status check

Status check processing

Time-out error? Yes

Normal completion? No

Time-out error [C]

Normal completion? Yes

Abnormal termination [B]

Wait from previous frame reception until next data frame transmission

Data frame transmission processing (User program)

Wait from data frame transmission until status check

Status check processing

Time-out error? Yes

Normal completion? No

Time-out error [C]

Normal completion? Yes

ST2 = ACK? No

Abnormal termination [B]

Yes

Abnormal termination [D]

No

All data frames transmitted?

Yes

Normal completion [A]

No
5.10.5 Sample program
The following shows a sample program for Verify command processing.

```c
u16 fl_csi_verify(u32 top, u32 bottom)
{
    u16 rc;
    u32 send_head, send_size;
    bool is_end;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom);  // set SAH/SAM/SAL, EAH/EAM/EAL

    /***************************************************************************
     * send command & check status                                    *
     /***************************************************************************/
    fl_wait(tCOM);
    put_cmd_csi(FL_COM_VERIFY, 7, fl_cmd_prm);  // send "Verify" command
    fl_wait(tWT6);

    rc = fl_csi_getstatus(tWT6_TO);  // get status frame
    switch(rc) {
        case FLC_NO_ERR: break;  // continue
        // case FLC_DFTO_ERR: return rc; break;  // case [C]
        default: return rc; break;  // case [B]
    }

    /***************************************************************************
     * send user data                                                   *
     /***************************************************************************/
    send_head = top;

    while(1){
        if ((bottom - send_head) > 256){  // rest size > 256 ?
            is_end = false;  // yes, not end frame
            send_size = 256;  // transmit size = 256 byte
        }
    }
}
```
else{
    is_end = true;
    send_size = bottom - send_head + 1;
    // transmit size = (bottom - send_head)+1 byte
}

memcpy(fl txdata_frm, rom_buf+send_head, send_size); // set data
    // frame payload
send_head += send_size;

fl_wait(tFD3_CSI); // wait before sending data frame
put_dfrm_csi(send_size, fl txdata_frm, is_end); // send data frame
fl_wait(tWT7); // wait
rc = fl_csi_getstatus(tWT7_MAX); // get status frame
switch(rc) {
    case FLC_NO_ERR: break; // continue

    default: return rc; break; // case [B]
}
if (fl_st2 != FLST_ACK){ // ST2 = ACK ?
    rc = decode_status(fl_st2); // No
    return rc; // case [D]
}

if (is_end) // send all user data ?
    break; // yes
    //continue;

} return FLC_NO_ERR; // case [A]
5.11 Block Blank Check Command

5.11.1 Processing sequence chart

Block Blank Check command processing sequence
5.11.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{com}$).

<2> The Block Blank Check command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time $t_{WT8} \times$ number of blocks).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

- When a time-out error occurs: A time-out error [C] is returned.
- When the processing ends abnormally: Abnormal termination [B]
- When the processing ends normally: Normal completion [A]

5.11.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>MRG11 error</td>
<td>1BH</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
5.11.4 Flowchart

Block Blank Check command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Block Blank Check)

Wait from command frame transmission until status check $t_{WTx} \times \text{number of blocks}$

Status check processing

Time-out error?

Yes

Time-out error [C]

No

Normal completion?

Yes

Normal completion [A]

No

Abnormal termination [B]
5.11.5 Sample program

The following shows a sample program for Block Blank Check command processing.

```c
u16  fl_csi_blk_blank_chk(u32 top, u32 bottom)
{
    u16  rc;
    u16  block_num;

    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom); // get block num

    fl_wait(tCOM);                      // wait before sending command frame

    put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);  // send "Block Blank Check" command

    fl_wait(tWT8 * block_num);

    rc = fl_csi_getstatus(tWT8_MAX * block_num);      // get status frame
    // switch(rc) {
    //     // case FLC_NO_ERR: return rc;        break; // case [A]
    //     // case FLC_DFTO_ERR: return rc;       break; // case [C]
    //     default:        return rc;           break; // case [B]
    // }
    return rc;
}
```
5.12 Silicon Signature Command

5.12.1 Processing sequence chart

Silicon Signature command processing sequence

Diagram showing the processing sequence with steps labeled 1 to 7 and outcomes such as normal completion, abnormal termination, time-out error, and data frame reception.
5.12.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{COM} \)).

<2> The Silicon Signature command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time \( t_{WT1} \)).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

   When the processing ends normally: Proceeds to <6>.
   When the processing ends abnormally: Abnormal termination [B]
   When a time-out error occurs: A time-out error [C] is returned.

<6> Waits from the previous frame reception until the next command transmission (wait time \( t_{FD2} \)).

<7> The received data frame (silicon signature data) is checked.

   If data frame is normal: Normal completion [A]
   If data frame is abnormal: Data frame error [D]

5.12.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>Read error</td>
<td>20H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
5.12.4 Flowchart

- Silicon Signature command processing
  - Wait from previous frame reception until next command transmission
    - Command frame transmission processing (Silicon Signature)
      - Wait from command frame transmission until status check
        - Status check processing
          - Time-out error?
            - No
              - Normal completion?
                - Yes
                  - Abnormal termination [B]
                  - Normal completion [A]
                - No
                  - Data frame reception processing
                    - Normal data frame?
                      - No
                        - Data frame error [D]
                      - Yes
                        - Normal completion [A]
  - Time-out error [C]
  - No
  - Normal data frame?
    - No
      - Data frame error [D]
    - Yes
      - Normal completion [A]
5.12.5 Sample program

The following shows a sample program for Silicon Signature command processing.

```c
u16 fl_csi_getsig(u8 *sig) 
{
    u16 rc;

    fl_wait(tCOM);    // wait before sending command frame

    put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);  // send "Silicon Signature" command

    fl_wait(tWT11);

    rc = fl_csi_getstatus(tWT11_TO);  // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [B]
    }

    fl_wait(tFD2_SIG);   // wait before getting data frame

    rc = get_dfrm_csi(fl_rxdata_frm); // get data frame (signature data)
    if (rc){  // if no error,
        return rc;    // case [D]
    }

    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);  // copy Signature data

    return rc;    // case [A]
}
```
5.13 Version Get Command

5.13.1 Processing sequence chart

Version Get command processing sequence

Programmer  78K0/Kx2

<1> Wait from previous frame reception until next command transmission

<2> Version Get command frame transmission

<3> Wait from command frame transmission until status check

<4> Status check processing

<5> Result of status check processing

Result
[Normal completion/ Abnormal termination/ Time-out error]

Time-out error

Abnormal termination

Normal completion

<6> Wait from previous frame reception until next data frame reception

<7> Status frame (version data) reception processing

Normal data frame?
[Yes/No]

No

Data frame error [D]

Yes

Normal completion [A]

Abnormal termination [B]

Time-out error [C]
5.13.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \(t_{COM}\)).

<2> The Version Get command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time \(t_{WT1}\)).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
When the processing ends abnormally: Abnormal termination [B]
When a time-out error occurs: A time-out error [C] is returned.

<6> Waits from the previous frame reception until the next command transmission (wait time \(t_{FD2}\)).

<7> The received data frame (version data) is checked.

If data frame is normal: Normal completion [A]
If data frame is abnormal: Data frame error [D]

5.13.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data is abnormal.</td>
</tr>
</tbody>
</table>
5.13.4 Flowchart

[Diagram of flowchart showing the sequence of operations and decision points for 3-Wire Serial I/O Communication Mode (CSI)]
5.13.5 Sample program

The following shows a sample program for Version Get command processing.

```c
/**
* Get device/firmware version command (CSI)
*/
/**
* [i] u8 *buf     ... pointer to version date save area
* [r] u16         ... error code
*/

u16  fl_csi_getver(u8 *buf)
{
    u16  rc;

    fl_wait(tCOM);    // wait before sending command frame

    put_cmd_csi(FL_COM_GET_VERSION, 1, fl_cmd_prm);    // send "Version Get" command

    fl_wait(tWT12);

    rc = fl_csi_getstatus(tWT12_TO);  // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [B]
    }

    fl_wait(tFD2_VG);   // wait before getting data frame

    rc = get_dfrm_csi(fl_rxdata_frm); // get version data

    if (rc){       // if no error,
        return rc;   // case [D]
    }
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);  // copy version data
    return rc;    // case [A]
}
```
5.14 Checksum Command

5.14.1 Processing sequence chart

Checksum command processing sequence

Programmer 78K0/Kx2

1. Wait from previous frame reception until next command transmission (tcOM)
2. Checksum command frame transmission
3. Wait from command frame transmission until status check (tWT16)
4. Status check processing
5. Result of status check processing
6. Wait from previous frame reception until next data frame reception (tFD1)
7. Data frame (checksum data) reception processing

Result [Normal completion/ Abnormal termination/ Time-out error]

Time-out error [C]
Abnormal termination [B]
Normal completion

Abnormal termination [B]
Normal data frame? [Yes/No]

Yes
Normal completion [A]
No
Data frame error [D]
5.14.2 Description of processing sequence

1. Waits from the previous frame reception until the next command transmission (wait time \( t_{com} \)).
2. The Checksum command is transmitted by command frame transmission processing.
3. Waits from command transmission until status check processing (wait time \( t_{WT1} \)).
4. The status frame is acquired by status check processing.
5. The following processing is performed according to the result of status check processing.

   - When the processing ends normally: Proceeds to <6>.
   - When the processing ends abnormally: Abnormal termination [B]
   - When a time-out error occurs: A time-out error [C] is returned.

6. Waits from the previous frame reception until the next command transmission (wait time \( t_{FD1} \)).
7. The received data frame (checksum data) is checked.

   - If data frame is normal: Normal completion [A]
   - If data frame is abnormal: Data frame error [D]

5.14.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data is abnormal.</td>
</tr>
</tbody>
</table>
5.14.4 Flowchart

- Checksum command processing
- Wait from previous frame reception until next command reception
- Command frame transmission processing (Checksum)
- Wait from command frame transmission until status check
- Status check processing
  - Time-out error?
    - Yes: Time-out error [C]
    - No: Normal completion?
      - Yes: Abnormal termination [B]
      - No: Wait from previous frame reception until next data frame reception
- Data frame reception processing
  - Normal data frame?
    - Yes: Normal completion [A]
    - No: Data frame error [D]
5.14.5 Sample program

The following shows a sample program for Checksum command processing.

```c
FormattedMessage
/* **********************************************************/
/* * Get checksum command (CSI) */
/* **********************************************************/
/* [i] u16 *sum ... pointer to checksum save area */
/* [i] u32 top ... start address */
/* [i] u32 bottom ... end address */
/* [r] u16 ... error code */
/****************************************************************************/

u16  fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16  rc;
    u16  block_num;
    //************************************************/
    //      set params                                
    //************************************************/
    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom); // get block num
    //************************************************/
    //      send command                              
    //************************************************/
    fl_wait(tCOM);    // wait before sending command frame
    put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send "Checksum" command
    fl_wait(tWT16);
    rc = fl_csi_getstatus(tWT16_TO); // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [B]
    }
    //************************************************/
    //      get data frame (Checksum data)            
    //************************************************/
    fl_wait(tFD1 * block_num); // wait before getting data frame
```
rc = get_dfrm_csi(fl_rxdata_frm); // get data frame(version data)

if (rc){
    // if error,
    return rc; // case [D]
}

*sum = (fl_rxdata_frm[OFS_STA_PLD] <<= 8) + fl_rxdata_frm[OFS_STA_PLD+1]; // set SUM data

return rc; // case [A]
5.15 Security Set Command

5.15.1 Processing sequence chart

Security Set command processing sequence
5.15.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time \( t_{COM} \)).

<2> The Security Set command is transmitted by command frame transmission processing.

<3> Waits from command transmission until status check processing (wait time \( t_{WT13} \)).

<4> The status frame is acquired by status check processing.

<5> The following processing is performed according to the result of status check processing.

   When the processing ends normally: Proceeds to <6>.
   When the processing ends abnormally: Abnormal termination [B]
   When a time-out error occurs: A time-out error [C] is returned.

<6> Waits from the previous frame reception until the data frame transmission (wait time \( t_{FD3(CSI)} \)).

<7> The data frame (security setting data) is transmitted by data frame transmission processing.

<8> Waits from data frame transmission until status check processing (wait time \( t_{WT14} \)).

<9> The status frame is acquired by status check processing.

<10> The following processing is performed according to the result of status check processing.

   When the processing ends normally: Proceeds to <11>.
   When the processing ends abnormally: Abnormal termination [B]
   When a time-out error occurs: A time-out error [C] is returned.

<11> Waits until status acquisition (completion of internal verify) (wait time \( t_{WT15} \)).

<12> The status frame is acquired by status check processing.

<13> The following processing is performed according to the result of status check processing.

   When the processing ends normally: Normal completion [A]
   When the processing ends abnormally: Abnormal termination [B]
   When a time-out error occurs: A time-out error [C] is returned.

5.15.3 Status at processing completion

<table>
<thead>
<tr>
<th>Status at Processing Completion</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion [A]</td>
<td>Normal acknowledgment (ACK)</td>
<td>06H</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error</td>
<td>05H</td>
</tr>
<tr>
<td></td>
<td>Checksum error</td>
<td>07H</td>
</tr>
<tr>
<td></td>
<td>Write error</td>
<td>1CH</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>–</td>
<td>The status frame was not received within the specified time.</td>
</tr>
</tbody>
</table>
5.15.4 Flowchart

1. **Security Set command processing**
   - Wait from previous frame reception until next command transmission
   - Command frame transmission processing (Security Set)
   - Wait from command frame transmission until status check
   - Status check processing
     - Time-out error?
       - Yes
       - Time-out error [C]
       - No
       - Normal completion?
         - Yes
         - Abnormal termination [B]
         - No
         - Wait from previous frame reception until next data frame transmission
   - Data frame transmission processing (Internal verify)
   - Wait from data frame transmission until status check
   - Status check processing
     - Time-out error?
       - Yes
       - Time-out error [C]
       - No
       - Normal completion?
         - Yes
         - Abnormal termination [B]
         - No
         - Wait during status check (internal verify)
           - Status check processing
             - Time-out error?
               - Yes
               - Time-out error [C]
               - No
               - Normal completion?
                 - Yes
                 - Normal completion [A]
                 - No
                 - Abnormal termination [B]
5.15.5 Sample program
The following shows a sample program for Security Set command processing.

```c
u16 fl_csi_setscf(u8 scf)
{
    u16 rc;

    fl_cmd_prm[0] = 0x00;  // "BLK" (must be 0x00)
    fl_cmd_prm[1] = 0x00;  // "PAG" (must be 0x00)
    fl_txdata_frm[0] = (scf |= 0b11101000);  // "FLG" (upper 5bits must be '1' (to make sure))
    fl_txdata_frm[1] = 0x03;  // "BOT" (fixed 0x03)

    fl_wait(tCOM);    // wait before sending command frame
    put_cmd_csi(FL_COM_SET_SECURITY, 3, fl_cmd_prm);  // send "Security Set" command

    fl_wait(tWT13);   // wait

    rc = fl_csi_getstatus(tWT13_TO);  // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [B]
    }

    fl_wait(tFD3_CSI);   // wait before getting data frame
}
```
put_dfrm_csi(2, fl_txdatal_frm, true); // send data frame (Security data)

fl_wait(tWT14);

rc = fl_csi_getstatus(tWT14_MAX);  // get status frame
switch(rc) {
    case FLC_NO_ERR:   break; // continue
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    default:  return rc; break; // case [B]
}

/************************************************/
/* Check internally verify                        */
/************************************************/
fl_wait(tWT15);

rc = fl_csi_getstatus(tWT15_MAX);  // get status frame
// switch(rc) {
//    //
//    // case FLC_NO_ERR: return rc; break; // case [A]
//    // case FLC_DFTO_ERR: return rc; break; // case [C]
//    // default: return rc; break; // case [B]
// }
return rc;
}
CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS

This chapter describes the parameter characteristics between the programmer and the 78K0/Kx2 in the flash memory programming mode.

Be sure to refer to the user’s manual of the 78K0/Kx2 for electrical specifications when designing a programmer.

6.1 Flash Memory Programming Parameter Characteristics of Expanded Specification Products (μPD78F05xxA)

6.1.1 Basic characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Condition</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/Kx2 operating clock in flash memory programming mode</td>
<td>Internal high-speed oscillation clock</td>
<td>( f_{RH} )</td>
<td>7.6</td>
<td>8</td>
<td>8.4</td>
<td>MHz</td>
</tr>
<tr>
<td>X1 clock</td>
<td>During UART communication</td>
<td>( f_{X} )</td>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>External main system clock</td>
<td></td>
<td>( f_{oCLK} )</td>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

6.1.2 Flash memory programming mode setting time

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{DD} \uparrow ) to ( FLMD0 \uparrow )</td>
<td>( t_{BP} )</td>
<td>1 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( FLMD0 \uparrow ) to ( \text{RESET} \uparrow )</td>
<td>( t_{BR} )</td>
<td>2 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count start time from ( \text{RESET} \uparrow ) to ( FLMD0 \uparrow \text{Note 1} )</td>
<td>( t_{BP} )</td>
<td>59,327/( f_{RH} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count finish time from ( \text{RESET} \uparrow ) to ( FLMD0 \uparrow \text{Note 1} )</td>
<td>( t_{BFPE} )</td>
<td>238,414/( f_{RH} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLMD0 counter high-level/low-level width</td>
<td>( t_{PW} )</td>
<td>10 ( \mu s )</td>
<td>100 ( \mu s )</td>
<td></td>
</tr>
<tr>
<td>Wait for Reset command (CSI) \text{Note 1}</td>
<td>( t_{BC} )</td>
<td>444,463/( f_{RH} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 1 (UART) \text{Note 1}</td>
<td>( t_{B1} )</td>
<td>444,463/( f_{RH} ) + ( 2^{16}/f_X )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 2 (UART)</td>
<td>( t_{L1}, t_{L2} )</td>
<td>Note 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait for Read command (UART)</td>
<td>( t_{BC} )</td>
<td>15,000/( f_{RH} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Width of low-level data 1/2 \text{Note 2}</td>
<td>( t_{L1}, t_{L2} )</td>
<td>Note 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLMD0 counter rise/fall time</td>
<td>–</td>
<td>1 ( \mu s )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset low level width (( \text{RESET} \uparrow ) to ( \text{RESET} \downarrow )) \text{Note 3}</td>
<td>( t_{BSTR} )</td>
<td>1,950 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes 1. (59,327/\( f_{RH} \) + 238,414/\( f_{RH} \))/2 is recommended as the standard value for the FLMD0 pulse input timing.
2. The low-level width is the same as the 00H data width at 9,600 bps.
3. When the mode is switched from the normal operating mode to the flash memory programming mode after the microcontroller is powered on (reset is released), be sure to wait for the period of this parameter at minimum before reset for mode switching after power-on (reset release).

(Remarks are carried over to the next page.)
Remarks 1. Calculate the parameters assuming that \( f_{RH} = 8 \text{ MHz} \).

2. The waits are defined as follows.

\(<t_{R1} \ (\text{MIN.})>\)

The baud rate for the UART is generated based on the external clock.

Input pulses by making allowances for this specification and the oscillation stabilization time of the external clock used.

### 6.1.3 Programming characteristics

<table>
<thead>
<tr>
<th>Wait Condition</th>
<th>Symbol</th>
<th>Serial I/F</th>
<th>MIN.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between data frame transmission/reception</td>
<td>( t_{DR} )</td>
<td>CSI</td>
<td>64/( f_{RH} )</td>
<td>74/( f_{RH} )</td>
</tr>
<tr>
<td>Data frame transmission</td>
<td>( t_{DT} )</td>
<td>CSI</td>
<td>88/( f_{RH} )</td>
<td></td>
</tr>
<tr>
<td>From Status command frame reception until status frame transmission</td>
<td>( t_{SF} )</td>
<td>CSI</td>
<td>215/( f_{RH} )</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until data frame transmission (1)</td>
<td>( t_{FD1} )</td>
<td>CSI</td>
<td>54,368/( f_{RH} )</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until data frame transmission (2)</td>
<td>( t_{FD2} )</td>
<td>CSI</td>
<td>321/( f_{RH} )</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until data frame reception</td>
<td>( t_{FD3} )</td>
<td>CSI</td>
<td>163/( f_{RH} )</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until command frame reception</td>
<td>( t_{COM} )</td>
<td>CSI</td>
<td>106/( f_{RH} )</td>
<td></td>
</tr>
</tbody>
</table>

Notes 1. When successive reception is enabled for the programmer

2. Time for one block transmission

Remarks 1. Calculate the parameters assuming that \( f_{RH} = 8 \text{ MHz} \).

2. The waits are defined as follows.

\(<t_{DR}, t_{FD2}, t_{COM}>\)

The 78K0/Kx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer can transmit the next data after the MIN. time has elapsed after completion of the previous communication.

The MAX. time is not specified. Transmit the next data within about 3 seconds.

\(<t_{DT}, t_{SF}, t_{FD1}, t_{FD2}>\)

The 78K0/Kx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer must prepare to receive the next data before the MIN. time has elapsed after completion of the previous communication.

The MAX. time is not specified. Continue polling for about 3 seconds until the data is received.
<table>
<thead>
<tr>
<th>Command</th>
<th>Symbol</th>
<th>Serial I/F</th>
<th>MIN.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>tWT0</td>
<td>CSI</td>
<td>172/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip Erase</td>
<td>tWT1</td>
<td>–</td>
<td>857,883/fRH + 44,160 × total number of blocks/fRH</td>
<td>186,444,400/fRH + 11,304,960 × total number of blocks/fRH</td>
</tr>
<tr>
<td>Block Erase</td>
<td>tWT2</td>
<td>Note 2</td>
<td>–</td>
<td>54,582,372/fRH × execution count of simultaneous selection and erasure + 11,304,960/fRH × number of blocks to be erased</td>
</tr>
<tr>
<td>Programming</td>
<td>tWT3</td>
<td>CSI</td>
<td>1,506/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tWT4</td>
<td>Note 3</td>
<td>–</td>
<td>72,412/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>893,355/fRH</td>
</tr>
<tr>
<td></td>
<td>tWT5</td>
<td>Note 4</td>
<td>CSI</td>
<td>Block 0 100,407/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>132,144,427/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Block 0</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>132,144,427/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Block 1 to 127</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>102,178/fRH</td>
</tr>
<tr>
<td>Verify</td>
<td>tWT6</td>
<td>CSI</td>
<td>686/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tWT7</td>
<td>Note 3</td>
<td>CSI</td>
<td>12,827/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Block Blank Check</td>
<td>tWT8</td>
<td>Note 4</td>
<td>CSI</td>
<td>45,870/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>55,044/fRH</td>
</tr>
<tr>
<td>Oscillating</td>
<td>tWT9</td>
<td>CSI</td>
<td>1,238/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td>Frequency Set</td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Silicon Signature</td>
<td>tWT11</td>
<td>CSI</td>
<td>1,233/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version Get</td>
<td>tWT12</td>
<td>CSI</td>
<td>252/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Set</td>
<td>tWT13</td>
<td>CSI</td>
<td>975/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tWT14</td>
<td>–</td>
<td>275,518/fRH</td>
<td>66,005,812/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tWT15</td>
<td>CSI</td>
<td>368,277/fRH</td>
<td>66,018,156/fRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Checksum</td>
<td>tWT16</td>
<td>CSI</td>
<td>583/fRH</td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes 1.** Reception must be enabled for the programmer before command transmission.

2. See 6.3 Simultaneous selection and erasure performed by Block Erase command for the calculation method of the execution count of simultaneous selection and erasure.

3. Time for 256-byte data transmission

4. Time for one block transmission

**Remarks 1.** Calculate the parameters assuming that fRH = 8 MHz.

2. The waits are defined as follows.

<twT0 to tWT16>

The 78K0/Kx2 completes command processing between the MIN. and MAX. times.
The programmer must repeat the status check for the period of the MAX. time (or about 3 seconds, if the MAX. time is not specified).

6.2 Flash Memory Programming Parameter Characteristics of Conventional-specification Products (μPD78F05xx)

6.2.1 Basic characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Condition</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0/Kx2 operating clock in flash memory programming mode</td>
<td>Internal high-speed oscillation clock</td>
<td>( f_{RH} )</td>
<td>7.6</td>
<td>8</td>
<td>8.4</td>
<td>MHz</td>
</tr>
<tr>
<td>X1 clock</td>
<td>During UART communication</td>
<td>( f_x )</td>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>External main system clock</td>
<td></td>
<td>( f_{EXCLK} )</td>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.2.2 Flash memory programming mode setting time

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{CC} \uparrow ) to FLMD0\uparrow</td>
<td>( t_{DP} )</td>
<td>1 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLMD0\uparrow to RESET\uparrow</td>
<td>( t_{FR} )</td>
<td>2 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count start time from RESET\uparrow to FLMD0^\textsuperscript{Note 1}</td>
<td>( t_{RP} )</td>
<td>59,327/f(_{RH})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count finish time from RESET\uparrow to FLMD0^\textsuperscript{Note 1}</td>
<td>( t_{RPE} )</td>
<td>238,414/f(_{RH})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLMD0 counter high-level/low-level width</td>
<td>( t_{PW} )</td>
<td>10 ( \mu )s</td>
<td>100 ( \mu )s</td>
<td></td>
</tr>
<tr>
<td>Wait for Reset command (CSI)</td>
<td>( t_{RC} )</td>
<td>444,463/f(_{RH})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 1 (UART)</td>
<td>( X1 ) clock</td>
<td>( t_{R1} )</td>
<td>444,463/f(_{RH}) + 2(^{nd})/( f_x )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>External main system clock</td>
<td></td>
<td>444,463/f(_{RH})</td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 2 (UART)</td>
<td></td>
<td>( t_{L2} )</td>
<td>15,000/f(_{RH})</td>
<td></td>
</tr>
<tr>
<td>Wait for Read command (UART)</td>
<td></td>
<td>( t_{RC} )</td>
<td>15,000/f(_{RH})</td>
<td></td>
</tr>
<tr>
<td>Width of low-level data 1/2^\textsuperscript{Note 2}</td>
<td></td>
<td>( t_{L1}, t_{L2} )</td>
<td>Note 2</td>
<td></td>
</tr>
<tr>
<td>FLMD0 counter rise/fall time</td>
<td></td>
<td></td>
<td>1 ( \mu )s</td>
<td></td>
</tr>
<tr>
<td>Reset low level width (RESET\uparrow to RESET\downarrow)^\textsuperscript{Note 2}</td>
<td></td>
<td>( t_{RST} )</td>
<td>1,950 ms</td>
<td></td>
</tr>
</tbody>
</table>

Notes 1. \((59,327/f_{RH} + 238,414/f_{RH})/2\) is recommended as the standard value for the FLMD0 pulse input timing.
2. The low-level width is the same as the 00H data width at 9,600 bps, and the value described here is half that data width.
3. When the mode is switched from the normal operating mode to the flash memory programming mode after the microcontroller is powered on (reset is released), be sure to wait for the period of this parameter at minimum before reset for mode switching after power-on (reset release).

Remarks 1. Calculate the parameters assuming that \( f_{RH} = 8 \) MHz.
2. The waits are defined as follows.
   <\( t_{R1} \) (MIN.)> 
   The baud rate for the UART is generated based on the external clock.
   Input pulses by making allowances for this specification and the oscillation stabilization time of the external clock used.
### 6.2.3 Programming characteristics

<table>
<thead>
<tr>
<th>Wait</th>
<th>Condition</th>
<th>Symbol</th>
<th>Serial I/F</th>
<th>MIN.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between data frame transmission/reception</td>
<td>Data frame reception</td>
<td>$t_{DR}$</td>
<td>CSI</td>
<td>64/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data frame transmission</td>
<td>$t_{DT}$</td>
<td>CSI</td>
<td>88/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data frame transmission</td>
<td>$t_{DS}$</td>
<td>UART</td>
<td>0$^{Note 1}$</td>
<td></td>
</tr>
<tr>
<td>From Status command frame reception until status frame transmission</td>
<td>–</td>
<td>$t_{SF}$</td>
<td>CSI</td>
<td>166/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until data frame transmission (1)</td>
<td>–</td>
<td>$t_{FD1}^{Note 2}$</td>
<td>CSI</td>
<td>54,368/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Silicon signature data</td>
<td>$t_{FD2}$</td>
<td>UART</td>
<td>0$^{Note 1}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Version data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until data frame transmission (2)</td>
<td>–</td>
<td>$t_{FD3}$</td>
<td>CSI</td>
<td>163/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>–</td>
<td>$t_{FD3}$</td>
<td>UART</td>
<td>101/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td>From status frame transmission until command frame reception</td>
<td>–</td>
<td>$t_{COM}$</td>
<td>CSI</td>
<td>64/f$_{RH}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>–</td>
<td>$t_{COM}$</td>
<td>UART</td>
<td>71/f$_{RH}$</td>
<td></td>
</tr>
</tbody>
</table>

**Notes**

1. When successive reception is enabled for the programmer
2. Time for one block transmission

**Remarks**

1. Calculate the parameters assuming that f$_{RH}$ = 8 MHz.
2. The waits are defined as follows.

$$<t_{DR}, t_{FD3}, t_{COM}>$$

The 78K0/Kx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer can transmit the next data after the MIN. time has elapsed after completion of the previous communication.

The MAX. time is not specified. Transmit the next data within about 3 seconds.

$$<t_{DT}, t_{SF}, t_{FD1}, t_{FD2}>$$

The 78K0/Kx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer must prepare to receive the next data before the MIN. time has elapsed after completion of the previous communication.

The MAX. time is not specified. Continue polling for about 3 seconds until the data is received.
<table>
<thead>
<tr>
<th>Command</th>
<th>Symbol</th>
<th>Serial I/F</th>
<th>MIN.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>tWT0</td>
<td>CSI</td>
<td>172/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Chip Erase</td>
<td>tWT1</td>
<td>–</td>
<td>857,883/(f_{RH}) + 44,160 \times \text{total number of blocks}/f_{RH}</td>
<td>186,444,400/(f_{RH}) + 11,304,960 \times \text{total number of blocks}/f_{RH}</td>
</tr>
<tr>
<td>Block Erase</td>
<td>tWT2\textsuperscript{Note 2}</td>
<td>–</td>
<td>214,714/(f_{RH}) \times \text{execution count of simultaneous selection and erasure + 44,160/(f_{RH}) \times \text{number of blocks to be erased}}</td>
<td>54,582,372/(f_{RH}) \times \text{execution count of simultaneous selection and erasure + 11,304,960/(f_{RH}) \times \text{number of blocks to be erased}}</td>
</tr>
<tr>
<td>Programming</td>
<td>tWT3</td>
<td>CSI</td>
<td>1,348/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td>tWT4\textsuperscript{Note 3}</td>
<td>–</td>
<td>68,118/(f_{RH})</td>
<td>397,587/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td>tWT5\textsuperscript{Note 4}</td>
<td>CSI</td>
<td>Block 0</td>
<td>100,407/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td>Block 0</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block 1 to 127</td>
<td>100,407/(f_{RH})</td>
<td>102,178/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block 1 to 127</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Verify</td>
<td>tWT6</td>
<td>CSI</td>
<td>686/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td>tWT7\textsuperscript{Note 3}</td>
<td>CSI</td>
<td>12,827/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Block Blank Check</td>
<td>tWT8\textsuperscript{Note 4}</td>
<td>CSI</td>
<td>45,835/(f_{RH})</td>
<td>55,044/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Oscillating Frequency Set</td>
<td>tWT9</td>
<td>CSI</td>
<td>1,127/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Silicon Signature</td>
<td>tWT11</td>
<td>CSI</td>
<td>1,233/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Version Get</td>
<td>tWT12</td>
<td>CSI</td>
<td>242/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Security Set</td>
<td>tWT13</td>
<td>CSI</td>
<td>923/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td></td>
<td>tWT14</td>
<td>–</td>
<td>275,518/(f_{RH})</td>
<td>66,005,812/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td>tWT15</td>
<td>CSI</td>
<td>368,277/(f_{RH})</td>
<td>66,018,156/(f_{RH})</td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
<tr>
<td>Checksum</td>
<td>tWT16</td>
<td>CSI</td>
<td>583/(f_{RH})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART</td>
<td></td>
<td></td>
<td>Note 1</td>
</tr>
</tbody>
</table>

**Notes**

1. Reception must be enabled for the programmer before command transmission.
2. See 6.3 *Simultaneous selection and erasure performed by Block Erase command* for the calculation method of the execution count of simultaneous selection and erasure.
3. Time for 256-byte data transmission
4. Time for one block transmission

**Remarks**

1. Calculate the parameters assuming that \(f_{RH} = 8\) MHz.
2. The waits are defined as follows.
   
   <tWT0 to tWT16>

   The 78K0/Kx2 completes command processing between the MIN. and MAX. times.
The programmer must repeat the status check for the period of the MAX. time (or about 3 seconds, if the MAX. time is not specified).

6.3 Simultaneous Selection and Erasure Performed by Block Erase Command

The Block Erase command of the 78K0/Kx2 is executed by repeating “simultaneous selection and erasure”, which erases multiple blocks simultaneously.

The wait time inserted during Block Erase command execution is therefore equal to the total execution time of “simultaneous selection and erasure”.

To calculate the “total execution time of simultaneous selection and erasure”, the execution count (M) of the simultaneous selection and erasure must first be calculated.

“M” is calculated by obtaining the number of blocks to be erased simultaneously (number of blocks to be selected and erased simultaneously).

The following describes the method for calculating the number of blocks to be selected and erased simultaneously and the execution count (M).

(1) Calculation of number of blocks to be selected and erased simultaneously

The number of blocks to be selected and erased simultaneously should be 1, 2, 4, 8, 16, 32, 64, or 128, depending on which satisfies all of the following conditions.

[Condition 1]
(Number of blocks to be erased) ≥ (Number of blocks to be selected and erased simultaneously)

[Condition 2]
(Start block number) ÷ (Number of blocks to be selected and erased simultaneously) = Remainder is 0

[Condition 3]
The maximum value among the values that satisfy both Conditions 1 and 2
(2) Calculation of the execution count (M) of simultaneous selection and erasure

Calculation of the execution count (M) is illustrated in the following flowchart.

![Flowchart showing the calculation of execution count (M) for simultaneous selection and erasure](chart)

**Variables**:
- **ST_BKNO**: Start block number
- **END_BKNO**: End block number
- **ER_BKNUM**: Number of blocks to be erased
- **SSER_BKNUM**: Number of blocks to be selected and erased simultaneously
- **M**: Execution count of simultaneous selection and erasure

**Legend**:
- **No**: Continue flowchart
- **Yes**: Go to next step

**Notes**:
- **Condition 1**: ER_BKNUM ≥ SSER_BKNUM
- **Condition 2**: ST_BKNO ÷ SSER_BKNUM = Remainder is 0
- **Condition 3**: SSER_BKNUM ÷ 2^n is satisfied

**Based on the maximum value of SSER_BKNUM (128), obtain the value that satisfies Conditions 1 and 2 by executing SSER_BKNUM ÷ 2; Condition 3 is then satisfied.**
Example 1  Erasing blocks 1 to 127 (N (number of blocks to be erased) = 127)

<1> The first start block number is 1 and the number of blocks to be erased is 127; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 1 is then erased.

<2> After block 1 is erased, the next start block number is 2 and the number of blocks to be erased is 126; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 2 and 3 are then erased.

<3> After blocks 2 and 3 are erased, the next start block number is 4 and the number of blocks to be erased is 124; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1, 2, and 4, the value that satisfies Condition 3 is 4, so the number of blocks to be selected and erased simultaneously is 4; blocks 4 to 7 are then erased.

<4> After blocks 4 to 7 are erased, the next start block number is 8 and the number of blocks to be erased is 120; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1, 2, 4, and 8, the value that satisfies Condition 3 is 8, so the number of blocks to be selected and erased simultaneously is 8; blocks 8 to 15 are then erased.

<5> After blocks 8 to 15 are erased, the next start block number is 16 and the number of blocks to be erased is 112; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, and 16, the value that satisfies Condition 3 is 16, so the number of blocks to be selected and erased simultaneously is 16; blocks 16 to 31 are then erased. After blocks 16 to 31 are erased, the next start block number is 32 and the number of blocks to be erased is 96; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, 16, and 32, the value that satisfies Condition 3 is 32, so the number of blocks to be selected and erased simultaneously is 32; blocks 32 to 63 are then erased.

<6> After blocks 32 to 63 are erased, the next start block number is 64 and the number of blocks to be erased is 64; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64. Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, 16, 32, and 64, the value that satisfies Condition 3 is 64, so the number of blocks to be selected and erased simultaneously is 64; blocks 64 to 127 are then erased.

Therefore, simultaneous selection and erasure is executed seven times (1, 2 and 3, 4 to 7, 8 to 15, 16 to 31, 32 to 63, and 64 to 127) to erase blocks 1 to 127, so $M = 7$ is obtained.
Block configuration when executing simultaneous selection and erasure (when erasing blocks 1 to 127)

<Range of blocks that can be selected and erased simultaneously>
Example 2  Erasing blocks 5 to 10 (N (number of blocks to be erased) = 6)

<1> The first start block number is 5 and the number of blocks to be erased is 6; the values that satisfy Condition 1 are therefore 1, 2, and 4. Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 5 is the erased.

<2> After block 5 is erased, the next start block number is 6 and the number of blocks to be erased is 5; the values that satisfy Condition 1 are therefore 1, 2, and 4. Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 6 and 7 are then erased.

<3> After blocks 6 and 7 are erased, the next start block number is 8 and the number of blocks to be erased is 3; the values that satisfy Condition 1 are therefore 1 and 2. Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 8 and 9 are then erased.

<4> After blocks 8 and 9 are erased, the next start block number is 10 and the number of blocks to be erased is 1; the value that satisfies Condition 1 is therefore 1. This also satisfies Conditions 2 and 3, so the number of blocks to be selected and erased simultaneously is 1; block 10 is then erased.

Therefore, simultaneous selection and erasure is executed four times (5, 6 and 7, 8 and 9, and 10) to erase blocks 5 to 10, so $M = 4$ is obtained.
Block configuration when executing simultaneous selection and erasure (when erasing blocks 5 to 10)

User area (128 KB)

<Range of blocks that can be selected and erased simultaneously>
Example 3  Erasing blocks 25 to 73 (N (number of blocks to be erased) = 49)

<1> The first start block number is 25 and the number of blocks to be erased is 49; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32. Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 25 is then erased.

<2> After block 25 is erased, the next start block number is 26 and the number of blocks to be erased is 48; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32. Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 26 and 27 are then erased.

<3> After blocks 26 and 27 are erased, the next start block number is 28 and the number of blocks to be erased is 46; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32. Moreover, the values that satisfy Condition 2 are 1, 2, and 4, the value that satisfies Condition 3 is 4, so the number of blocks to be selected and erased simultaneously is 4; blocks 28 to 31 are then erased.

<4> After blocks 28 to 31 are erased, the next start block number is 32 and the number of blocks to be erased is 42; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32. Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, and 32, the value that satisfies Condition 3 is 32, so the number of blocks to be selected and erased simultaneously is 32; blocks 32 to 63 are then erased.

<5> After blocks 32 to 63 are erased, the next start block number is 64, and the number of blocks to be erased is 10; the values that satisfy Condition 1 are therefore 1, 2, 4, and 8. Moreover, the values that satisfy Condition 2 are 1, 2, 4, and 8, the value that satisfies Condition 3 is 8, so the number of blocks to be selected and erased simultaneously is 8; blocks 64 to 71 are then erased.

<6> After blocks 64 to 71 are erased, the next start block number is 72, and the number of blocks to be erased is 2; the values that satisfy Condition 1 are therefore 1 and 2. Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 72 and 73 are then erased.

Therefore, simultaneous selection and erasure is executed six times (25, 26 and 27, 28 to 31, 32 to 63, 64 to 71, and 72 and 73) to erase blocks 25 to 73, so M = 6 is obtained.
Block configuration when executing simultaneous selection and erasure (when erasing blocks 25 to 73)

User area (128 KB)

Range of blocks that can be selected and erased simultaneously:

- Blocks 24 to 63
- Blocks 64 to 127

Block number:
6.4 UART Communication Mode

(a) Data frame

(b) Programming mode setting (At the time of power-on)

(c) Programming mode setting (After power-on)

(d) Reset command

Remark  TxD: TxD6  
         RxD: RxD6
(e) Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command

(f) Silicon Signature command/Version Get command

(g) Checksum command

(h) Programming command

Remark  TxD: TXD6  
        RxD: RXD6
(i) Verify command

![Diagram of Verify command]

(j) Security Set command

![Diagram of Security Set command]

(k) Wait before command frame transmission

![Diagram of Wait before command frame transmission]

**Remark**  
TxD: TxD6  
RxD: RxD6
6.5 3-Wire Serial I/O Communication Mode

(a) Data frame

(b) Programming mode setting (At the time of power-on)

(c) Programming mode setting (After power-on)

(d) Reset command

Remark
- SCK: SCK10
- SC: SO10
- SI: SI10
(e) Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command

(f) Silicon Signature command/Version Get command

(g) Checksum command

(h) Programming command

Remark
- SCK: SCK10
- SC: SO10
- SI: SI10
(i) Verify command

(j) Security Set command

(k) Wait before command frame transmission

Remark

SCK: SCK10
SC: SO10
SI: SI10
APPENDIX A  CIRCUIT DIAGRAMS (REFERENCE)

Figure A-1 to A-3 show circuit diagrams of the programmer and the 78K0/Kx2, for reference.
[MEMO]
Figure A-1. Reference Circuit Diagram of Programmer and 78K0/Kx2 (During UART communication: with X1 Clock Used)
Figure A-2. Reference Circuit Diagram of Programmer and 78K0/Kx2 (During UART communication: with External Clock Used)
Figure A-3. Reference Circuit Diagram of Programmer and 78K0/Kx2 (During CSI Communication)
## APPENDIX B REVISION HISTORY

### B.1 Major Revisions in This Edition

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
</table>
| Throughout | Addition of Expanded Specification Products  
Deletion of 2.1 Programmer Control Pins and 2.2 Details of control pins  
Move of 9 sections (from 2.3 Basic Flowchart to 2.11 Status List) to Chapter 1 FLASH MEMORY PROGRAMMING  
Addition of 6.1 Flash Memory Programming Parameter Characteristics of Expanded Specification Products ($\mu$PD78F05xxA)  
Modification of 6.2.2 Flash Memory Programming Mode Setting Time  
Modification of 6.2.3 Programming Characteristics  
Modification of 6.4 (c) Programming mode setting (After power-on)  
Modification of 6.5 3-Wire Serial I/O Communication Mode  
Addition of B.2 Revision History up to Previous Edition |
| p.32 | Modification of description in 3.5 Chip Erase Command |
| p.39 | Modification of Table 3-1. Example of Silicon Signature Data (In Case of $\mu$PD78F0522 (78K0/KD2)) |
| pp.41 to 45 | Modification of 3.10.4 78K0/Kx2 silicon signature list |
| pp.52 to 54 | From 4.1 Command Frame Transmission Processing Flowchart to 4.3 Data Frame Reception Processing Flowchart  
• Modification of the symbol in the flowchart |
| pp.104 to 106 | From 5.1 Command Frame Transmission Processing Flowchart to 5.3 Data Frame Reception Processing Flowchart  
• Modification of the symbol in the flowchart |
| p.108 | Modification of 5.4.3 Status at processing completion |
| pp.161 to 164 | Addition of 6.1 Flash Memory Programming Parameter Characteristics of Expanded Specification Products ($\mu$PD78F05xxA) |
| p.164 | Modification of 6.2.2 Flash Memory Programming Mode Setting Time |
| pp.165 to 167 | Modification of 6.2.3 Programming Characteristics |
| p.175 | Addition of 6.4 (c) Programming mode setting (After power-on) |
| p.178 | Modification of 6.5 3-Wire Serial I/O Communication Mode |
| p.190 | Addition of B.2 Revision History up to Previous Edition |
## B.2 Revision History up to Previous Edition

The following table shows the revision history up to the previous editions. The "Applied to:" column indicates the chapters of each edition in which the revision was applied.

<table>
<thead>
<tr>
<th>Edition</th>
<th>Major Revision from Previous Edition</th>
<th>Applied to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd edition</td>
<td>Modification of <strong>Figure 2-5. Basic Flowchart for Flash Memory Rewrite Processing</strong></td>
<td>CHAPTER 2 PROGRAMMER OPERATING ENVIRONMENT</td>
</tr>
<tr>
<td></td>
<td>Addition of <strong>2.4.1 Mode Setting flowchart</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Addition of <strong>2.4.2 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>6.8.3 Status at processing completion</strong></td>
<td>CHAPTER 6 UART COMMUNICATION MODE</td>
</tr>
<tr>
<td></td>
<td>• Deletion of FLMD error in Abnormal termination [D]</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>6.9.3 Status at processing completion</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Deletion of description of Parameter error in Abnormal termination [B]</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>6.11.3 Status at processing completion</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Addition of Read error in Abnormal termination [B]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>6.14.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Addition of <strong>Note in 7.4.1 Processing sequence chart</strong></td>
<td>CHAPTER 7 3-WIRE SERIAL I/O COMMUNICATION MODE (CSI)</td>
</tr>
<tr>
<td></td>
<td>Modification of description and addition of <strong>Note in 7.4.2 Description of processing sequence</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Addition of <strong>Note in 7.4.4 Flowchart</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.4.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.5.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.6.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.7.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.8.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>7.9.3 Status at processing completion</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Deletion of FLMD error in Abnormal termination [D]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.9.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.10.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.11.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>7.12.3 Status at processing completion</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Addition of Read error in Abnormal termination [B]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.12.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.13.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.14.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of <strong>7.15.5 Sample program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modification of Wait for low-level data 1 (UART) in <strong>8.2 Flash Memory Programming Mode Setting Time</strong></td>
<td>CHAPTER 8 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS</td>
</tr>
<tr>
<td></td>
<td>Deletion of <strong>CHAPTER 9 ELECTRICAL SPECIFICATIONS (REFERENCE)</strong></td>
<td>Previous edition CHAPTER 9 ELECTRICAL SPECIFICATIONS (REFERENCE)</td>
</tr>
<tr>
<td>Edition</td>
<td>Major Revision from Previous Edition</td>
<td>Applied to:</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>2nd edition</td>
<td>Modification of Figure A-1. Reference Circuit Diagram of Programmer and 78K0/Kx2 (During UART communication: with X1 Clock Used) to Figure A-3. Reference Circuit Diagram of Programmer and 78K0/Kx2 (During CSI Communication)</td>
<td>APPENDIX A CIRCUIT DIAGRAMS (REFERENCE)</td>
</tr>
<tr>
<td></td>
<td>Addition of APPENDIX B REVISION HISTORY</td>
<td>APPENDIX B REVISION HISTORY</td>
</tr>
</tbody>
</table>
For further information, please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211-8668, Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5600

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steigenweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F ., Excellence Times Square Building,
No. 406 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Honk Kong Ltd.
Unit 1601-1613, 16/F ., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied’or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/