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 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-crime 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 them 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

V850ES/Kx2
32-Bit Single-Chip Microcontrollers
Flash Memory Programming (Programmer)

μPD70F3726
μPD70F3728
μPD70F3729
μPD70F3731
μPD70F3732
μPD70F3733
μPD70F3734
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 $V_{IL\ (MAX)}$ and $V_{IH\ (MIN)}$ 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 $V_{IL\ (MAX)}$ and $V_{IH\ (MIN)}$.

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 November, 2006. 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).
INTRODUCTION

Target Readers This application note is intended for users who understand the functions of the V850ES/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 V850ES/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
- Programmer operating environment
- Basic programmer operation
- Command/data frame format
- Description of command processing
- UART communication mode
- 3-wire serial I/O communication mode with handshake supported (CSI + HS)
- 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 V850ES/Kx2’s hardware functions:
→ See the user’s manual of each V850ES/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.

Device-related documents

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>V850ES/KG2 User's Manual</td>
<td>U17703E</td>
</tr>
</tbody>
</table>
CONTENTS

CHAPTER 1  FLASH MEMORY PROGRAMMING ................................................................. 15

  1.1 Overview ............................................................................................................. 15

  1.2 System Configuration ........................................................................................ 16

  1.3 Programming Overview ...................................................................................... 17
     1.3.1 Setting flash memory programming mode ...................................................... 17
     1.3.2 Selecting serial communication mode ........................................................... 17
     1.3.3 Manipulating flash memory via command transmission/reception .............. 18

  1.4 Information Specific to V850ES/Kx2 ................................................................. 18

CHAPTER 2  PROGRAMMER OPERATING ENVIRONMENT ................................................. 20

  2.1 Programmer Control Pins ................................................................................... 20

  2.2 Details of Control Pins ........................................................................................ 21
     2.2.1 Flash memory programming mode setting pins (FLMD0, FLMD1) ............... 21
     2.2.2 Serial interface pins (TxD, RxD, SI, SO, SCK, HS) ....................................... 22
     2.2.3 Reset control pin (RESET) ............................................................................. 23
     2.2.4 Clock control pin (CLK) ................................................................................ 23
     2.2.5 VDD/GND control pins ............................................................................... 24
     2.2.6 Other pins ..................................................................................................... 24

  2.3 Basic Flowchart .................................................................................................. 25

  2.4 Setting Flash Memory Programming Mode ...................................................... 26

  2.5 Selecting Serial Communication Mode ............................................................. 27

  2.6 UART Communication Mode .............................................................................. 27

  2.7 3-Wire Serial I/O Communication Mode with Handshake Supported (CSI + HS) ...... 28

  2.8 3-Wire Serial I/O Communication Mode (CSI) .................................................. 28

  2.9 Shutting Down Target Power Supply ................................................................. 28

  2.10 Manipulation of Flash Memory ........................................................................ 29

  2.11 Command List ................................................................................................... 29

  2.12 Status List ......................................................................................................... 30

CHAPTER 3  BASIC PROGRAMMER OPERATION ............................................................ 31

CHAPTER 4  COMMAND/DATA FRAME FORMAT ......................................................... 32

  4.1 Command Frame Transmission Processing ....................................................... 34

  4.2 Data Frame Transmission Processing ............................................................... 34

  4.3 Data Frame Reception Processing ..................................................................... 34

CHAPTER 5  DESCRIPTION OF COMMAND PROCESSING ............................................ 35

  5.1 Status Command ............................................................................................... 35
     5.1.1 Description .................................................................................................... 35
     5.1.2 Command frame and status frame ................................................................. 35

  5.2 Reset Command .................................................................................................. 36
     5.2.1 Description .................................................................................................... 36
     5.2.2 Command frame and status frame ................................................................. 36
CHAPTER 6 UART COMMUNICATION MODE................................................................. 57

6.1 Command Frame Transmission Processing Flowchart.................................. 57
### 6.12 Silicon Signature Command
- Processing sequence chart ................................................................. 96
- Description of processing sequence ......................................................... 97
- Status at processing completion ............................................................... 97
- Flowchart ............................................................................................... 98
- Sample program .................................................................................... 99

### 6.13 Version Get Command
- Processing sequence chart ................................................................. 100
- Description of processing sequence ......................................................... 101
- Status at processing completion ............................................................... 101
- Flowchart ............................................................................................... 102
- Sample program .................................................................................... 103

### 6.14 Checksum Command
- Processing sequence chart ................................................................. 104
- Description of processing sequence ......................................................... 105
- Status at processing completion ............................................................... 105
- Flowchart ............................................................................................... 106
- Sample program .................................................................................... 107

### 6.15 Security Set Command
- Processing sequence chart ................................................................. 108
- Description of processing sequence ......................................................... 109
- Status at processing completion ............................................................... 109
- Flowchart ............................................................................................... 110
- Sample program .................................................................................... 111

### 6.16 Read Command
- Processing sequence chart ................................................................. 113
- Description of processing sequence ......................................................... 114
- Status at processing completion ............................................................... 114
- Flowchart ............................................................................................... 115
- Sample program .................................................................................... 116

### 7.1 Command Frame Transmission Processing Flowchart
- Processing sequence chart ................................................................. 118

### 7.2 Data Frame Transmission Processing Flowchart
- Processing sequence chart ................................................................. 119

### 7.3 Data Frame Reception Processing Flowchart
- Processing sequence chart ................................................................. 120

### 7.4 Status Command
- Processing sequence chart ................................................................. 121
- Description of processing sequence ......................................................... 122
- Status at processing completion ............................................................... 122
- Flowchart ............................................................................................... 123
- Sample program .................................................................................... 124

### 7.5 Reset Command
- Processing sequence chart ................................................................. 125
- Description of processing sequence ......................................................... 126
- Status at processing completion ............................................................... 126
7.6 Oscillating Frequency Set Command ................................................................. 129
  7.6.1 Processing sequence chart ........................................................................... 129
  7.6.2 Description of processing sequence ............................................................ 130
  7.6.3 Status at processing completion ................................................................. 130
  7.6.4 Flowchart ..................................................................................................... 131
  7.6.5 Sample program ........................................................................................... 132

7.7 Chip Erase Command ...................................................................................... 133
  7.7.1 Processing sequence chart ........................................................................... 133
  7.7.2 Description of processing sequence ............................................................ 134
  7.7.3 Status at processing completion ................................................................. 134
  7.7.4 Flowchart ..................................................................................................... 135
  7.7.5 Sample program ........................................................................................... 136

7.8 Block Erase Command .................................................................................... 137
  7.8.1 Processing sequence chart ........................................................................... 137
  7.8.2 Description of processing sequence ............................................................ 138
  7.8.3 Status at processing completion ................................................................. 138
  7.8.4 Flowchart ..................................................................................................... 139
  7.8.5 Sample program ........................................................................................... 140

7.9 Programming Command ................................................................................ 141
  7.9.1 Processing sequence chart ........................................................................... 141
  7.9.2 Description of processing sequence ............................................................ 142
  7.9.3 Status at processing completion ................................................................. 143
  7.9.4 Flowchart ..................................................................................................... 144
  7.9.5 Sample program ........................................................................................... 145

7.10 Verify Command .......................................................................................... 147
  7.10.1 Processing sequence chart ........................................................................ 147
  7.10.2 Description of processing sequence ......................................................... 148
  7.10.3 Status at processing completion ............................................................... 148
  7.10.4 Flowchart ................................................................................................... 149
  7.10.5 Sample program ........................................................................................ 150

7.11 Block Blank Check Command ..................................................................... 152
  7.11.1 Processing sequence chart ....................................................................... 152
  7.11.2 Description of processing sequence ......................................................... 153
  7.11.3 Status at processing completion ............................................................... 153
  7.11.4 Flowchart .................................................................................................. 154
  7.11.5 Sample program ......................................................................................... 155

7.12 Silicon Signature Command ......................................................................... 156
  7.12.1 Processing sequence chart ....................................................................... 156
  7.12.2 Description of processing sequence ......................................................... 157
  7.12.3 Status at processing completion ............................................................... 157
  7.12.4 Flowchart .................................................................................................. 158
  7.12.5 Sample program ......................................................................................... 159

7.13 Version Get Command .................................................................................. 160
  7.13.1 Processing sequence chart ....................................................................... 160
  7.13.2 Description of processing sequence ......................................................... 161
  7.13.3 Status at processing completion ............................................................... 161

Application Note U17885EJ2V1AN
CHAPTER 8  3-WIRE SERIAL I/O COMMUNICATION MODE (CSI) ............................................. 180

8.1 Command Frame Transmission Processing Flowchart............................................. 180
8.2 Data Frame Transmission Processing Flowchart ..................................................... 181
8.3 Data Frame Reception Processing Flowchart .......................................................... 182
8.4 Status Command ..................................................................................................... 183
  8.4.1 Processing sequence chart ................................................................................... 183
  8.4.2 Description of processing sequence ...................................................................... 184
  8.4.3 Status at processing completion .......................................................................... 184
  8.4.4 Flowchart ........................................................................................................... 185
  8.4.5 Sample program .................................................................................................. 186
8.5 Reset Command ...................................................................................................... 187
  8.5.1 Processing sequence chart ................................................................................... 187
  8.5.2 Description of processing sequence ...................................................................... 188
  8.5.3 Status at processing completion .......................................................................... 188
  8.5.4 Flowchart ........................................................................................................... 189
  8.5.5 Sample program .................................................................................................. 190
8.6 Oscillating Frequency Set Command ...................................................................... 191
  8.6.1 Processing sequence chart ................................................................................... 191
  8.6.2 Description of processing sequence ...................................................................... 192
  8.6.3 Status at processing completion .......................................................................... 192
  8.6.4 Flowchart ........................................................................................................... 193
  8.6.5 Sample program .................................................................................................. 194
8.7 Chip Erase Command .............................................................................................. 195
  8.7.1 Processing sequence chart ................................................................................... 195
  8.7.2 Description of processing sequence ...................................................................... 196
  8.7.3 Status at processing completion .......................................................................... 196
  8.7.4 Flowchart ........................................................................................................... 197
8.8 Block Erase Command ................................................................. 199
  8.8.1 Processing sequence chart ...................................................... 199
  8.8.2 Description of processing sequence ....................................... 200
  8.8.3 Status at processing completion ........................................... 200
  8.8.4 Flowchart .............................................................................. 201
  8.8.5 Sample program ................................................................. 202

8.9 Programming Command ......................................................... 203
  8.9.1 Processing sequence chart ...................................................... 203
  8.9.2 Description of processing sequence ....................................... 204
  8.9.3 Status at processing completion ........................................... 205
  8.9.4 Flowchart .............................................................................. 206
  8.9.5 Sample program ................................................................. 207

8.10 Verify Command ........................................................................ 209
  8.10.1 Processing sequence chart ...................................................... 209
  8.10.2 Description of processing sequence ....................................... 210
  8.10.3 Status at processing completion ........................................... 210
  8.10.4 Flowchart .............................................................................. 211
  8.10.5 Sample program ................................................................. 212

8.11 Block Blank Check Command ................................................ 214
  8.11.1 Processing sequence chart ...................................................... 214
  8.11.2 Description of processing sequence ....................................... 215
  8.11.3 Status at processing completion ........................................... 215
  8.11.4 Flowchart .............................................................................. 216
  8.11.5 Sample program ................................................................. 217

8.12 Silicon Signature Command ...................................................... 218
  8.12.1 Processing sequence chart ...................................................... 218
  8.12.2 Description of processing sequence ....................................... 219
  8.12.3 Status at processing completion ........................................... 219
  8.12.4 Flowchart .............................................................................. 220
  8.12.5 Sample program ................................................................. 221

8.13 Version Get Command .............................................................. 222
  8.13.1 Processing sequence chart ...................................................... 222
  8.13.2 Description of processing sequence ....................................... 223
  8.13.3 Status at processing completion ........................................... 223
  8.13.4 Flowchart .............................................................................. 224
  8.13.5 Sample program ................................................................. 225

8.14 Checksum Command ................................................................. 226
  8.14.1 Processing sequence chart ...................................................... 226
  8.14.2 Description of processing sequence ....................................... 227
  8.14.3 Status at processing completion ........................................... 227
  8.14.4 Flowchart .............................................................................. 228
  8.14.5 Sample program ................................................................. 229

8.15 Security Set Command ............................................................. 230
  8.15.1 Processing sequence chart ...................................................... 230
  8.15.2 Description of processing sequence ....................................... 231
  8.15.3 Status at processing completion ........................................... 231
  8.15.4 Flowchart .............................................................................. 232
CHAPTER 1 FLASH MEMORY PROGRAMMING

To rewrite the contents of the internal flash memory of the V850ES/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 V850ES/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 V850ES/Kx2 via serial communication.

Figure 1-1. System Outline of Flash Memory Programming in V850ES/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-FP4 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 with handshake supported (CSI + HS) (MSB-first transfer)**

(3) **3-wire serial I/O communication mode (CSI) (MSB-first transfer)**
1.3 Programming Overview

To rewrite the contents of the flash memory with the programmer, the V850ES/Kx2 must first be set to the flash memory programming mode. After that, select the mode for communication between the programmer and the V850ES/Kx2, transmit commands from the programmer via serial communication, and then rewrite the flash memory. The flowchart of programming is illustrated in Figure 1-3.

Figure 1-3. Programming Flowchart

1.3.1 Setting flash memory programming mode
Suppose a specific voltage to the flash memory programming mode setting pins (FLMD0 and FLMD1) in the V850ES/Kx2 and release a reset; the flash memory programming mode is then set.

1.3.2 Selecting serial communication mode
To select a serial communication mode, generate pulses by changing the voltage at flash memory programming mode setting pin (FLMD0) between the VDD voltage and GND voltage in the flash memory programming mode, and determine the communication mode according to the pulse count.
1.3.3 Manipulating flash memory via command transmission/reception

The flash memory incorporated in the V850ES/Kx2 has functions to rewrite the flash memory contents. The flash memory manipulating functions shown in Table 1-1 are available.

Table 1-1. Outline of Flash Memory Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase</td>
<td>Erases the flash memory contents.</td>
</tr>
<tr>
<td>Write</td>
<td>Writes data to the flash memory.</td>
</tr>
<tr>
<td>Verify</td>
<td>Compares the flash memory contents with data for verify.</td>
</tr>
<tr>
<td>Acquisition of information</td>
<td>Reads information related to the flash memory.</td>
</tr>
</tbody>
</table>

To control these functions, the programmer transmits commands to the V850ES/Kx2 via serial communication. The V850ES/Kx2 returns the response status for the commands. The programming to the flash memory is performed by repeating these series of serial communications.

1.4 Information Specific to V850ES/Kx2

The programmer must manage product-specific information (such as a device name and memory information). Table 1-2 shows the flash memory size of the V850ES/Kx2 and Figure 1-4 shows the configuration of the flash memory.

Table 1-2. Flash Memory Size of V850ES/Kx2

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Flash Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>V850ES/KE2</td>
<td>( \mu P D70F3726 ) 128 KB</td>
</tr>
<tr>
<td>V850ES/KF2</td>
<td>( \mu P D70F3728 ) 128 KB</td>
</tr>
<tr>
<td></td>
<td>( \mu P D70F3729 ) 256 KB</td>
</tr>
<tr>
<td>V850ES/KG2</td>
<td>( \mu P D70F3731 ) 128 KB</td>
</tr>
<tr>
<td></td>
<td>( \mu P D70F3732 ) 256 KB</td>
</tr>
<tr>
<td>V850ES/KJ2</td>
<td>( \mu P D70F3733 ) 128 KB</td>
</tr>
<tr>
<td></td>
<td>( \mu P D70F3734 ) 256 KB</td>
</tr>
</tbody>
</table>
Figure 1-4. Flash Memory Configuration

Remark  Each block consists of 2 KB (this figure only illustrates some parts of entire blocks in the flash memory).
CHAPTER 2 PROGRAMMER OPERATING ENVIRONMENT

2.1 Programmer Control Pins

Table 2-1 lists the pins that the programmer must control to implement the programmer function in the user system. See the following pages for details on each pin.

### Table 2-1. Pin Description

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>I/O</th>
<th>Pin Function</th>
<th>Pin Name</th>
<th>Mode for Communication with Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLMD0</td>
<td>Output</td>
<td>Output of signal level to set programming mode and output of pulse to select communication mode</td>
<td>FLMD0</td>
<td>CSI</td>
</tr>
<tr>
<td>FLMD1</td>
<td>Output</td>
<td>Output of signal level to set programming mode</td>
<td>FLMD1</td>
<td>CSI</td>
</tr>
<tr>
<td>VDD</td>
<td>Output</td>
<td>VDD voltage generation/monitoring</td>
<td>VDD</td>
<td>CSI</td>
</tr>
<tr>
<td>GND</td>
<td>–</td>
<td>Ground</td>
<td>VSS</td>
<td>CSI</td>
</tr>
<tr>
<td>CLK</td>
<td>Output</td>
<td>Operating clock output to V850ES/Kx2</td>
<td>X1, X2rst</td>
<td>CSI</td>
</tr>
<tr>
<td>RESET</td>
<td>Output</td>
<td>Programming mode switching trigger</td>
<td>RESET</td>
<td>CSI</td>
</tr>
<tr>
<td>SO</td>
<td>Output</td>
<td>Command transmission to V850ES/Kx2</td>
<td>SI00</td>
<td>CSI</td>
</tr>
<tr>
<td>SI</td>
<td>Input</td>
<td>Response status and data reception from V850ES/Kx2</td>
<td>SO00</td>
<td>CSI</td>
</tr>
<tr>
<td>SCK</td>
<td>Output</td>
<td>Serial clock supply to V850ES/Kx2</td>
<td>SCK00</td>
<td>CSI</td>
</tr>
<tr>
<td>HS (handshake)</td>
<td>Input</td>
<td>Handshake signal reception for serial communication with V850ES/Kx2</td>
<td>PCM0</td>
<td>CSI</td>
</tr>
<tr>
<td>TxD</td>
<td>Output</td>
<td>Command transmission to V850ES/Kx2</td>
<td>RXD0</td>
<td>CSI</td>
</tr>
<tr>
<td>RxD</td>
<td>Input</td>
<td>Response status and data reception from V850ES/Kx2</td>
<td>TXD0</td>
<td>CSI</td>
</tr>
</tbody>
</table>

**Note**  
Connect the X1 inverse signal to X2.

**Remark**  
- O: Be sure to connect the pin.  
- ×: The pin does not have to be connected.  
- △: The pin does not have to be connected if the signal is generated on the target board.

For the voltage of the pins controlled by the programmer, refer to the user’s manual of the device that is subject to flash memory programming.
2.2 Details of Control Pins

2.2.1 Flash memory programming mode setting pins (FLMD0, FLMD1)

The FLMD0 and FLMD1 pins are used to control the operating mode of the V850ES/Kx2. The V850ES/Kx2 operates in flash memory programming mode when a specific voltage is supplied to these pins and a reset is released.

The mode for the serial communication between the programmer and the V850ES/Kx2 is determined by controlling the voltage at the FLMD0 pin between VDD and GND and outputting pulses, after reset. Refer to Table 2-3 for the relationship between the FLMD0 pulse counts and communication modes.
2.2.2 Serial interface pins (TxD, RxD, SI, SO, SCK, HS)

The serial interface pins are used to transfer the flash memory writing commands between the programmer and the V850ES/Kx2.

With the V850ES/Kx2, the communication mode can be selected from UART, CSI + HS, and CSI. The following figures illustrate the connection of pins used in each communication mode.

**Figure 2-1. Serial Interface Pins**

1. **UART communication mode**

   ![UART connection diagram]

2. **3-wire serial I/O communication mode with handshake supported (CSI + HS)**

   ![3-wire serial I/O with handshake diagram]

3. **3-wire serial I/O communication mode (CSI)**

   ![3-wire serial I/O without handshake diagram]
2.2.3 Reset control pin (RESET)

The reset control pin is used to control the system reset for the V850ES/Kx2 from the programmer. The flash memory programming mode can be selected when a specific voltage is supplied to the FLMD0 and FLMD1 pins and a reset is released.

![Figure 2-2. Reset Control Pin](image)

2.2.4 Clock control pin (CLK)

The clock control pin is used only when the clock is supplied from the programmer to the V850ES/Kx2. Connection of this pin is not necessary when it is not necessary to supply the operating clock to the V850ES/Kx2 from the programmer.

![Figure 2-3. Clock Control Pin](image)
2.2.5 VDD/GND control pins

The VDD control pin is used to supply power to the V850ES/Kx2 from the programmer. Connection of this pin is not necessary when it is not necessary to supply power to the V850ES/Kx2 from the programmer. However, this pin must be connected regardless of whether the power is supplied from the programmer when the dedicated programmer is used, because the dedicated programmer monitors the power supply status of the V850ES/Kx2.

The GND control pin must be connected to VSS of the V850ES/Kx2 regardless of whether the power is supplied from the programmer.

![Figure 2-4. VDD/GND Control Pin](image)

2.2.6 Other pins

For the connection of the pins that are not connected to the programmer, refer to the chapter describing the flash memory in the user’s manual of each device.
2.3 Basic Flowchart

The following illustrates the basic flowchart for performing flash memory rewriting with the programmer.

Figure 2-5. Basic Flowchart for Flash Memory Rewrite Processing

- **Basic flow**
- Power application to target (See Figure 2-6)
- Mode setting (reset release) (See 2.4)
- Selection of communication mode (pulse input) (See 2.4/2.5)
- Synchronization processing (Reset command) (See 5.2)
- UART communication?
  - Yes
    - Baud rate setting (See 5.3)
  - No
    - Command execution
      - Processing completed?
        - No
        - Yes
          - Target power shutdown processing (See 2.9)
      - Yes
        - End

Baud rate setting processing is not required when a mode other than UART communication mode is set.

Reset input and power shutdown during rewriting is prohibited because security information may be lost.
2.4 Setting Flash Memory Programming Mode

To rewrite the contents of the flash memory with the programmer, the V850ES/Kx2 must first be set to the flash memory programming mode by supplying a specific voltage to the flash memory programming mode setting pins (FLMD0, FLMD1) in the V850ES/Kx2, then releasing a reset.

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

![Figure 2-6. Setting Flash Memory Programming Mode and Selecting Communication Mode](image)

The relationship between the settings of the FLMD0 and FLMD1 pins after reset release and the operating mode is shown below.

<table>
<thead>
<tr>
<th>FLMD0</th>
<th>FLMD1</th>
<th>Operating Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low (GND)</td>
<td>Any</td>
<td>Normal operating mode</td>
</tr>
<tr>
<td>High (VDD)</td>
<td>Low (GND)</td>
<td>Flash memory programming mode</td>
</tr>
<tr>
<td>High (VDD)</td>
<td>High (VDD)</td>
<td>Setting prohibited</td>
</tr>
</tbody>
</table>
### 2.5 Selecting Serial Communication Mode

The communication mode is determined by inputting a pulse to the FLMD0 pin in the V850ES/Kx2 after reset release.

The high- and low-levels of the FLMD0 pulse are VDD and GND, respectively.

The following table shows the relationship between the number of FLMD0 pulses (pulse counts) and communication modes that can be selected with the V850ES/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 (UART0)</td>
<td>0</td>
<td>TxD0 (P30), RxD0 (P31)</td>
</tr>
<tr>
<td>3-wire serial I/O (CSI00)</td>
<td>8</td>
<td>SO00 (P41), SI00 (P40), SCK00 (P42)</td>
</tr>
<tr>
<td>3-wire serial I/O with handshake supported (CSI00 + HS)</td>
<td>11</td>
<td>SO00 (P41), SI00 (P40), SCK00 (P42), PCM0</td>
</tr>
<tr>
<td>Setting prohibited</td>
<td>Others</td>
<td>–</td>
</tr>
</tbody>
</table>

### 2.6 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>Selectable from 9,600, 19,200, 31,250, 38,400, 76,800, and 153,600 bps (default: 9,600 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 V850ES/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 without assigning one pin like CSI + HS communication.

**Caution** Set the same baud rate to the master and slave devices when performing UART communication.
2.7 3-Wire Serial I/O Communication Mode with Handshake Supported (CSI + HS)

In the CSI + HS communication mode, the timing for communication of commands or data is optimized. In addition to the SI, SO and SCK pins, the HS (handshake) pin is used for implementing effective communication.

The level of the HS pin signal falls (low level) when the V850ES/Kx2 is ready for transmitting or receiving data. The programmer must check the falling edge of the HS pin signal (low level) before starting transmission/reception of commands or data to the V850ES/Kx2.

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

![Timing Chart of CSI + HS Communication](image)

2.8 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 V850ES/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.

2.9 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.

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

The flash memory incorporated in the V850ES/Kx2 has functions to manipulate the flash memory, as listed in Table 2-5. The programmer transmits commands to control these functions to the V850ES/Kx2, and checks the response status sent from the V850ES/Kx2, to manipulate the flash memory.

<table>
<thead>
<tr>
<th>Classification</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase</td>
<td>Chip erase</td>
<td>Erases the entire flash memory area. Clears the security flag.</td>
</tr>
<tr>
<td></td>
<td>Block erase</td>
<td>Erases a specified block in the flash memory.</td>
</tr>
<tr>
<td>Write</td>
<td>Write</td>
<td>Writes data to a specified area in the flash memory.</td>
</tr>
<tr>
<td>Verify</td>
<td>Verify</td>
<td>Compares data acquired from a specified address in the flash memory with data transmitted from the programmer, on the V850ES/Kx2 side.</td>
</tr>
<tr>
<td>Blank check</td>
<td>Block blank check</td>
<td>Checks the erase status of a specified area in the flash memory.</td>
</tr>
<tr>
<td>Read</td>
<td>Read</td>
<td>Reads data of a specified area in the flash memory.</td>
</tr>
<tr>
<td>Information</td>
<td>Silicon signature acquisition</td>
<td>Acquires writing protocol information.</td>
</tr>
<tr>
<td></td>
<td>Version acquisition</td>
<td>Acquires version information of the V850ES/Kx2 and firmware.</td>
</tr>
<tr>
<td></td>
<td>Status acquisition</td>
<td>Acquires the current operating status.</td>
</tr>
<tr>
<td></td>
<td>Checksum acquisition</td>
<td>Acquires checksum data of a specified area.</td>
</tr>
<tr>
<td>Security</td>
<td>Security setting</td>
<td>Sets security information.</td>
</tr>
<tr>
<td>Other</td>
<td>Reset</td>
<td>Detects synchronization in communication.</td>
</tr>
</tbody>
</table>

2.11 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</th>
</tr>
</thead>
<tbody>
<tr>
<td>70H</td>
<td>Status</td>
<td>Acquires the current operating status (status data).</td>
</tr>
<tr>
<td>00H</td>
<td>Reset</td>
<td>Detects synchronization in communication.</td>
</tr>
<tr>
<td>90H</td>
<td>Oscillating Frequency Set</td>
<td>Specifies the oscillation frequency of the V850ES/Kx2.</td>
</tr>
<tr>
<td>9AH</td>
<td>Baud Rate Set</td>
<td>Sets baud rate when UART communication mode is selected.</td>
</tr>
<tr>
<td>20H</td>
<td>Chip Erase</td>
<td>Erases the entire flash memory area.</td>
</tr>
<tr>
<td>22H</td>
<td>Block Erase</td>
<td>Erases a specified area in the flash memory.</td>
</tr>
<tr>
<td>40H</td>
<td>Programming</td>
<td>Writes data to a specified area in the flash memory.</td>
</tr>
<tr>
<td>13H</td>
<td>Verify</td>
<td>Compares the contents in a specified area in the flash memory with data transmitted from the programmer.</td>
</tr>
<tr>
<td>32H</td>
<td>Block Blank Check</td>
<td>Checks the erase status of a specified block in the flash memory.</td>
</tr>
<tr>
<td>C0H</td>
<td>Silicon Signature</td>
<td>Acquires V850ES/Kx2 information (part number, flash memory configuration, etc.).</td>
</tr>
<tr>
<td>C5H</td>
<td>Version Get</td>
<td>Acquires version information of the V850ES/Kx2 and firmware.</td>
</tr>
<tr>
<td>B0H</td>
<td>Checksum</td>
<td>Acquires checksum data of a specified area.</td>
</tr>
<tr>
<td>A0H</td>
<td>Security Set</td>
<td>Sets security information.</td>
</tr>
<tr>
<td>50H</td>
<td>Read</td>
<td>Reads data of a specified area in the flash memory.</td>
</tr>
</tbody>
</table>
2.12 Status List

The following table lists the status codes the programmer receives from the V850ES/Kx2.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Status Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04H</td>
<td>Command number error</td>
</tr>
<tr>
<td>05H</td>
<td>Parameter error</td>
</tr>
<tr>
<td>06H</td>
<td>Normal acknowledgment (ACK)</td>
</tr>
<tr>
<td>07H</td>
<td>Checksum error</td>
</tr>
<tr>
<td>0FH</td>
<td>Verify error</td>
</tr>
<tr>
<td>10H</td>
<td>Protect error</td>
</tr>
<tr>
<td>15H</td>
<td>Negative acknowledgment (NACK)</td>
</tr>
<tr>
<td>18H</td>
<td>FLMD error</td>
</tr>
<tr>
<td>1AH</td>
<td>MRG10 error</td>
</tr>
<tr>
<td>1BH</td>
<td>MRG11 error</td>
</tr>
<tr>
<td>1CH</td>
<td>Write error</td>
</tr>
<tr>
<td>FFH</td>
<td>Processing in progress (BUSY)</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 or after BUSY status check via the HS pin. 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, HS pin time-out, or time-out in data frame reception during UART communication) occurs, it is recommended to shutdown the power supply to the V850ES/Kx2 (refer to 2.9 Shutting Down Target Power Supply) and then connect the power supply again.
CHAPTER 3 BASIC PROGRAMMER OPERATION

Figure 3-1 illustrates the general command execution flow when flash memory rewriting is performed with the programmer.

Figure 3-1. General Command Execution Flow at Flash Memory Rewriting

- General command flow
- Flash memory programming mode is set
- Reset command
- Oscillating Frequency Set command
- Baud Rate Set command (in UART communication mode only)
- Block Blank Check command
- Chip Erase command
- Programming command
- Security Set command
- Flash memory programming mode is exited
- End

Note It is recommended to perform security settings to disable read as a default in programmer specification.

Remark The Verify command, Checksum command, and Read command can also be supported.
CHAPTER 4 COMMAND/DATA FRAME FORMAT

The programmer uses the command frame to transmit commands to the V850ES/Kx2. The V850ES/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 4-1. 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). Command frame: LEN + COM + command information length Data frame: LEN + 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 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>
</tbody>
</table>

**Figure 4-2. 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 (variable length) (Max. 256 bytes)</td>
</tr>
<tr>
<td>SUM</td>
<td>-</td>
<td>Checksum data for a frame</td>
</tr>
<tr>
<td>ETX or ETB</td>
<td>17H</td>
<td>Footer of data frame other than the last frame</td>
</tr>
</tbody>
</table>

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.

```
SOH   LEN   COM   SUM   ETX
01H   01H   70H   Checksum   03H
```

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.

```
SOH   LEN   COM   SUM   ETX
01H   01H   70H   8FH   03H
```

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

```
STX   LEN   D1   D2   D3   D4   SUM   ETX
02H   04H   FFH   80H   40H   22H   Checksum   03H
```

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.

```
STX   LEN   D1   D2   D3   D4   SUM   ETX
02H   04H   FFH   80H   40H   22H   1BH   03H
```

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.

```
STX   LEN   D1   D2   D3   D4   SUM   ETX
02H   04H   FFH   80H   40H   22H   1AH   03H
```

↑ Should be 1BH, if normal
4.1 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 6.1 Flowchart of Command Frame Transmission Processing.
• For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.1 Flowchart of Command Frame Transmission Processing.
• For the 3-wire serial I/O communication mode (CSI), read 8.1 Flowchart of Command Frame Transmission Processing.

4.2 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 6.2 Flowchart of Data Frame Transmission Processing.
• For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.2 Flowchart of Data Frame Transmission Processing.
• For the 3-wire serial I/O communication mode (CSI), read 8.2 Flowchart of Data Frame Transmission Processing.

4.3 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 6.3 Flowchart of Data Frame Reception Processing.
• For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.3 Flowchart of Data Frame Reception Processing.
• For the 3-wire serial I/O communication mode (CSI), read 8.3 Flowchart of Data Frame Reception Processing.
5.1 Status Command

5.1.1 Description
This command is used to check the operation status of the V850ES/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 V850ES/Kx2 due to problems based on communication or the like, the status setting will not performed in the V850ES/Kx2. As a result, a busy response (FFH), not the status frame, may be received. In such a case, retry the Status command.

5.1.2 Command frame and status frame
Figure 5-1 shows the format of a command frame for the Status command, and Figure 5-2 shows the status frame for the command.

Figure 5-1. Status Command Frame (from Programmer to V850ES/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 5-2. Status Frame for Status Command (from V850ES/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>...</td>
<td>STn</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 V850ES/Kx2.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the V850ES/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 with handshake supported (CSI + HS), read 7.4 Status Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.4 Status Command.

Caution After each command such as write or erase is transmitted in UART communication, the V850ES/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.
5.2 Reset Command

5.2.1 Description
This command is used to check the establishment of communication between the programmer and the V850ES/Kx2 after the communication mode is set.

When UART is selected as the mode for communication with the V850ES/Kx2, the same baud rate must be set in the programmer and V850ES/Kx2. However, the V850ES/Kx2 cannot detect its own operating frequency so the baud rate cannot be set. It makes detection of the operating frequency in the V850ES/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.

5.2.2 Command frame and status frame

Figure 5-3 shows the format of a command frame for the Reset command, and Figure 5-4 shows the status frame for the command.

**Figure 5-3. Reset Command Frame (from Programmer to V850ES/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 5-4. Status Frame for Reset Command (from V850ES/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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.4 Reset Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.5 Reset Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.5 Reset Command.
5.3 Baud Rate Set Command

5.3.1 Description

This command is used to change the baud rate for UART (default value: 9,600 bps).

After the Baud Rate Set command is executed, the Reset command must be executed to confirm synchronization at the new baud rate.

The Baud Rate Set command is valid only in the UART communication mode. Data for setting the baud rate is represented as a 1-byte numeric value.

The V850ES/Kx2 ignores the Baud Rate Set command if it is transmitted in modes other than the UART communication mode.

5.3.2 Command frame and status frame

Figure 5-5 shows the format of a command frame for the Baud Rate Set command, and Figure 5-6 shows the status frame for the command.

Figure 5-5. Baud Rate Set Command Frame (from Programmer to V850ES/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>02H</td>
<td>9AH</td>
<td>D01</td>
<td></td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark  D01: Baud rate selection value

<table>
<thead>
<tr>
<th>D01 Value</th>
<th>03H</th>
<th>04H</th>
<th>05H</th>
<th>06H</th>
<th>07H</th>
<th>08H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate (bps)</td>
<td>9600</td>
<td>19200</td>
<td>31250</td>
<td>38400</td>
<td>76800</td>
<td>153600</td>
</tr>
</tbody>
</table>

Figure 5-6. Status Frame for Baud Rate Set Command (from V850ES/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></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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.5 Baud Rate Set Command.
- The Baud Rate Set command is not used in the 3-wire serial I/O communication mode with handshake supported (CSI + HS).
- The Baud Rate Set command is not used in 3-wire serial I/O communication mode (CSI).
5.4 Oscillating Frequency Set Command

5.4.1 Description

This command is used to set oscillation frequency data in the V850ES/Kx2. Set the frequency of the clock that is actually input to the X1 pin of the V850ES/Kx2. However, when the execution of this command succeeds while the clock frequency is 2 to 5 MHz, the target device will multiply the CPU operation clock by 4 through the internal PLL. Therefore, set the calculated value of clock count multiplied by 4 to each timing parameter.

5.4.2 Command frame and status frame

Figure 5-7 shows the format of a command frame for the Oscillating Frequency Set command, and Figure 5-8 shows the status frame for the command.

Figure 5-7. Oscillating Frequency Set Command Frame (from Programmer to V850ES/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) × 10⁶ (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 × 10⁶ = 6,000 kHz = 6 MHz

Setting example: To set 10 MHz
D01 = 01H
D02 = 00H
D03 = 00H
D04 = 05H
Oscillation frequency = 1 × 0.1 × 10⁶ = 10,000 kHz = 10 MHz

Figure 5-8. Status Frame for Oscillating Frequency Set Command (from V850ES/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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.6 Oscillating Frequency Set Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.6 Oscillating Frequency Set Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.6 Oscillating Frequency Set Command.
5.5 Chip Erase Command

5.5.1 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 erasure is not prohibited by the security setting (see 5.13 Security Set Command).

5.5.2 Command frame and status frame
Figure 5-9 shows the format of a command frame for the Chip Erase command, and Figure 5-10 shows the status frame for the command.

Figure 5-9. Chip Erase Command Frame (from Programmer to V850ES/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 5-10. Status Frame for Chip Erase Command (from V850ES/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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.7 Chip Erase Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.7 Chip Erase Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.7 Chip Erase Command.
5.6 Block Erase Command

5.6.1 Description
This command is used to erase the contents of blocks with the specified number in the flash memory, as long as erasure is not prohibited by the security setting (see 5.13 Security Set Command).

5.6.2 Command frame and status frame
Figure 5-11 shows the format of a command frame for the Block Erase command, and Figure 5-12 shows the status frame for the command.

Figure 5-11. Block Erase Command Frame (from Programmer to V850ES/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>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark SAH, SAM, SAL: Block erase start address (start address of the block)
EAH, EAM, EAL: Block erase end address (end address of the block)

Figure 5-12. Status Frame for Block Erase Command (from V850ES/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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.8 Block Erase Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.8 Block Erase Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.8 Block Erase Command.
5.7 Programming Command

5.7.1 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 V850ES/Kx2 firmware automatically executes internal verify. Therefore, the Status command for this internal verify must be transmitted.

After executing the Programming command, it is recommended to execute the Security Set command that disables read as a default in programmer specification.

5.7.2 Command frame and status frame
Figure 5-13 shows the format of a command frame for the Programming command, and Figure 5-14 shows the status frame for the command.

Figure 5-13. Programming Command Frame (from Programmer to V850ES/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</td>
<td>(Programming)</td>
<td>SAH</td>
<td>SAM</td>
</tr>
</tbody>
</table>

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

Figure 5-14. Status Frame for Programming Command (from V850ES/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

5.7.3 Data frame and status frame
Figure 5-15 shows the format of a frame that includes data to be written, and Figure 5-16 shows the status frame for the data.

Figure 5-15. Data Frame to Be Written (from Programmer to V850ES/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 to FFH (00H = 256)</td>
<td>Write Data</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
</tbody>
</table>

Remark Write Data: User program to be written

Figure 5-16. Status Frame for Data Frame (from V850ES/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>Data</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
</table>
| 02H | 02H | ST1 (b) | ST2 (b) | ST1 (b): Data reception check result  
ST2 (b): Write result

Checksum | 03H |
5.7.4 Completion of transferring all data and status frame

Figure 5-17 shows the status frame after transfer of all data is completed.

**Figure 5-17. Status Frame After Completion of Transferring All Data (from V850ES/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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.9 Programming Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.9 Programming Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.9 Programming Command.
5.8 Verify Command

5.8.1 Description
This command is used to compare the data transmitted from the programmer with the data read from the V850ES/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.

5.8.2 Command frame and status frame
Figure 5-18 shows the format of a command frame for the Verify command, and Figure 5-19 shows the status frame for the command.

Figure 5-18. Verify Command Frame (from Programmer to V850ES/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>(Verify)</td>
<td>SAH</td>
<td>SAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAL</td>
<td>EAH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EAM</td>
<td>EAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Figure 5-19. Status Frame for Verify Command (from V850ES/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></td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark: ST1 (a): Command reception result

5.8.3 Data frame and status frame
Figure 5-20 shows the format of a frame that includes data to be verified, and Figure 5-21 shows the status frame for the data.

Figure 5-20. Data Frame of Data to Be Verified (from Programmer to V850ES/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 to FFH (00H = 256)</td>
<td>Verify data</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
</tbody>
</table>

Remark: Verify Data: User program to be verified
Figure 5-21. Status Frame for Data Frame (from V850ES/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>Checksum</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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.10 Verify Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.10 Verify Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.10 Verify Command.
5.9 Block Blank Check Command

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

5.9.2 Command frame and status frame
Figure 5-22 shows the format of a command frame for the Block Blank Check command, and Figure 5-23 shows the status frame for the command.

Figure 5-22. Block Blank Check Command Frame (from Programmer to V850ES/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></td>
<td></td>
</tr>
</tbody>
</table>

Remark  SAH, SAM, SAL: Block blank check start address (start address of the block)
        EAH, EAM, EAL: Block blank check end address (end address of the block)

Figure 5-23. Status Frame for Block Blank Check Command (from V850ES/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></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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.11 Block Blank Check Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.11 Block Blank Check Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.11 Block Blank Check Command.
5.10 Silicon Signature Command

5.10.1 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 V850ES/Kx2, for example, execute this command to select an appropriate protocol in accordance with the values of the second and third bytes.

5.10.2 Command frame and status frame
Figure 5-24 shows the format of a command frame for the Silicon Signature command, and Figure 5-25 shows the status frame for the command.

Figure 5-24. Silicon Signature Command Frame (from Programmer to V850ES/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>COH</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

(Silicon Signature)

Figure 5-25. Status Frame for Silicon Signature Command (from V850ES/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

5.10.3 Silicon signature data frame
Figure 5-26 shows the format of a frame that includes silicon signature data.

Figure 5-26. Silicon Signature Data Frame (from V850ES/Kx2 to Programmer)

<table>
<thead>
<tr>
<th>STX</th>
<th>LEN</th>
<th>VEN</th>
<th>EXT</th>
<th>MSC</th>
<th>DEC</th>
<th>Invalid data</th>
<th>SCF</th>
<th>BOT</th>
<th>SUM</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>n</td>
<td>VEN</td>
<td>EXT</td>
<td>MSC</td>
<td>DEC</td>
<td>Invalid data</td>
<td>SCF</td>
<td>BOT</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remarks
1. n (LEN): Data length
VEN: Vendor code (NEC: 10H)
EXT: Extension code
MSC: Macro function code
DEC: Device extension code
INVALID DATA: Invalid data of 13 byte length.
SCF: Security flag setting
BOT: Boot block number
2. For the vendor code (VEN), extension code (EXT) and function information (FNC), the highest bit is used as an odd parity. The following shows an example.
Table 5-1. Example of Silicon Signature Data

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
<th>Length (Byte)</th>
<th>Example of Silicon Signature Data&lt;sup&gt;new&lt;/sup&gt;</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEN</td>
<td>Vendor code (NEC)</td>
<td>1</td>
<td>10H (00010000B)</td>
<td>10H</td>
</tr>
<tr>
<td>EXT</td>
<td>Extension code (fixed)</td>
<td>1</td>
<td>7FH (01111111B)</td>
<td>7FH</td>
</tr>
<tr>
<td>MSC</td>
<td>Macro function code (fixed)</td>
<td>1</td>
<td>01H (00000001B)</td>
<td>01H</td>
</tr>
<tr>
<td>DEC</td>
<td>Device extension code (fixed)</td>
<td>1</td>
<td>FEH (11111110B)</td>
<td>7EH</td>
</tr>
<tr>
<td>INVALID</td>
<td>Invalid data</td>
<td>13</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SCF</td>
<td>Security setting</td>
<td>1</td>
<td>7FH (01111111B)</td>
<td>7FH</td>
</tr>
<tr>
<td>BOT</td>
<td>Boot block number</td>
<td>1</td>
<td>00H (no parity)</td>
<td>00H</td>
</tr>
</tbody>
</table>

Note: The shaded bit is an odd parity (the value to adjust the number of “1” in a byte).

Read the following chapters for details on flowcharts of processing sequences between the programmer and the V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.12 Silicon Signature Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.12 Silicon Signature Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.12 Silicon Signature Command.
5.11 Version Get Command

5.11.1 Description
This command is used to acquire information on the V850ES/Kx2 device version and firmware version.
Use this command when the programming parameters must be changed in accordance with the V850ES/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>

5.11.2 Command frame and status frame
Figure 5-28 shows the format of a command frame for the Version Get command, and Figure 5-29 shows the status frame for the command.

**Figure 5-28. Version Get Command Frame (from Programmer to V850ES/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 5-29. Status Frame for Version Get Command (from V850ES/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
5.11.3 Version data frame

Figure 5-30 shows the data frame of version data.

![Figure 5-30. Version Data Frame (from V850ES/Kx2 to Programmer)](image)

<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>
</tbody>
</table>

**Remark**  
DV1: Integer of device version  
DV2: First decimal place of device version  
DV3: Second decimal place of device version  
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 V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.13 Version Get Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.13 Version Get Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.13 Version Get Command.
5.12 Checksum Command

5.12.1 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 (2 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
(00H) in 1-byte units.

5.12.2 Command frame and status frame
Figure 5-31 shows the format of a command frame for the Checksum command, and Figure 5-32 shows the status
frame for the command.

Figure 5-31. Checksum Command Frame (from Programmer to V850ES/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>B0H</td>
<td>(Checksum)</td>
<td>SAH</td>
<td>SAM</td>
</tr>
</tbody>
</table>

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

Figure 5-32. Status Frame for Checksum Command (from V850ES/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

5.12.3 Checksum data frame
Figure 5-33 shows the format of a frame that includes checksum data.

Figure 5-33. Checksum Data Frame (from V850ES/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>CK1</td>
<td>CK2</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

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
V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.14 Checksum Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.14 Checksum
  Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.14 Checksum Command.
5.13 Security Set Command

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

After executing the Programming command, it is recommended to execute the Security Set command that disables read as a default in programmer specification.

Caution Once the security setting is performed, additional setting to the security flags or changing of the setting from disable to enable will no longer be possible. If such settings are attempted, a Protect error (10H) will occur. To re-set the security flag, all the security flags must be initialized by executing the Chip Erase command (the Block Erase command cannot be used to initialize the security flags). If chip erase has been disabled, however, chip erase itself will be impossible and 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.

5.13.2 Command frame and status frame

Figure 5-34 shows the format of a command frame for the Security Set command, and Figure 5-35 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 5-34. Security Set Command Frame (from Programmer to V850ES/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>03H</td>
<td>A0H</td>
<td>(Security Set)</td>
<td>00H (fixed)</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

Remark  BLK, PAG: Fixed to 00H

Figure 5-35. Status Frame for Security Set Command (from V850ES/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
5.13.3 Data frame and status frame

Figure 5-36 shows the format of a security data frame, and Figure 5-37 shows the status frame for the data.

**Figure 5-36  Security Data Frame (from Programmer to V850ES/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 number
(When bit 4 in FLG = 1, it is fixed to 00H. When the bit is 0, its value is arbitrary.)

**Figure 5-37. Status Frame for Security Data Writing (from V850ES/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

5.13.4 Internal verify check and status frame

Figure 5-38 shows the status frame for internal verify check.

**Figure 5-38. Status Frame for Internal Verify Check (from V850ES/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 5-2. Contents of 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 rewrite, 0: Disables rewrite)</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Read disable flag (1: Enables read, 0: Disables read)</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Programming disable flag (1: Enables programming, 0: Disables programming)</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Block erase disable flag (1: Enables block erase, 0: Disables block erase)</td>
</tr>
<tr>
<td>Bit 0</td>
<td>Chip erase disable flag (1: Enables chip erase, 0: Disables 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 5-3. Security Flag Field and Enable/Disable Status of Each Operation

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Flash Memory Programming Mode</th>
<th>Self-Programming Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Setting Item</td>
<td>Command Operation After Security Setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>√: Execution possible, ×: Execution impossible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>△: Writing and block erase in boot area are impossible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Programming</td>
<td>Chip Erase</td>
</tr>
<tr>
<td>Disable programming</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Disable chip erase</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Disable block erase</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Disable read</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Disable boot block rewrite</td>
<td>△</td>
<td>×</td>
</tr>
</tbody>
</table>

- All commands can be executed regardless of the security setting values
- Retention of security setting values and change from enabled to disabled are possible
- Same condition as that in flash memory programming mode (on-board/off-board programming)

Read the following chapters for details on flowcharts of processing sequences between the programmer and the V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read 6.15 Security Set Command.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read 7.15 Security Set Command.
- For the 3-wire serial I/O communication mode (CSI), read 8.15 Security Set Command.
5.14 Read Command

5.14.1 Description
This command is used to read data from the flash memory of the V850ES/Kx2.
The write start/end address can only be set in the block start/end address units.

5.14.2 Command frame and status frame
Figure 5-39 shows the format of a command frame for the Read command, and Figure 5-40 shows the status frame for the command.

Figure 5-39. Read Command Frame (from Programmer to V850ES/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>50H</td>
<td>SAH, SAM, SAL, EAH, EAM, EAL</td>
<td>Checksum</td>
<td>03H</td>
</tr>
</tbody>
</table>

Remark SAH, SAM, SAL: Read start address (start address of the block)
EAH, EAM, EAL: Read end address (end address of the block)

Figure 5-40. Status Frame for Read Command (from V850ES/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

5.14.3 Data frame and status frame
Figure 5-41 shows the format of a data frame that includes data to be read, and Figure 5-42 shows the status frame for the data.

Figure 5-41. Data Frame of Data to Be Read (from Programmer to V850ES/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 to FFH</td>
<td>Read Data</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
</tbody>
</table>

Remark Read Data: Data read from V850ES/Kx2

Figure 5-42. Status Frame for Read Data (from V850ES/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>00H to FFH</td>
<td>ST1 (b)</td>
<td>Checksum</td>
<td>03H/17H</td>
</tr>
</tbody>
</table>

Remark ST1 (b): ACK (06H) or NACK (15H) sent from the programmer for read data
Read the following chapters for details on flowcharts of processing sequences between the programmer and the V850ES/Kx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.16 Read Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.16 Read Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.16 Read Command**.
6.1 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?
          - No
            - Wait between data transmissions
            - 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
          - Yes
            - Wait between data transmissions
            - tDT (UART)
            - No
              - Wait between data transmissions
              - tDT (UART)
6.2 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

Data frame header (STX = 02H) transmission

Wait between data transmissions

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

Wait between data transmissions

Checksum data (SUM) transmission

No

Wait between data transmissions

Last data frame?
Yes

Transmission of last data frame footer (ETX = 03H)

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

End of data frame transmission
### 6.3 Data Frame Reception Processing Flowchart

- **Data frame reception processing**
  - Yes: Data frame header (STX = 02H) received?
    - No
    - Yes: Timed out? (MAX.) (UART)
      - Yes: Reception time-out error
    - No: Data length (LEN) received?
      - Yes: Data length (LEN) received?
        - No: Timed out? (MAX.) (UART)
          - Yes: Reception time-out error
        - Yes: 1-byte data received?
          - Yes: Timed out? (MAX.) (UART)
            - Yes: Reception time-out error
          - No: Length bytes received?
            - Yes: Checksum data (SUM) received?
              - Yes: Checksum data (SUM) received?
                - No: Timed out? (MAX.) (UART)
                  - Yes: Reception time-out error
                - Yes: Data frame footer received?
                  - Yes: Last data frame footer (ETX = 03H)
                    or footer other than those of last data frame (ETB = 17H)
                  - No: Timed out? (MAX.) (UART)
                    - Yes: Reception time-out error
              - No: Timed out? (MAX.) (UART)
                - Yes: Reception time-out error
            - No: Timed out? (MAX.) (UART)
              - Yes: Reception time-out error
          - No: Timed out? (MAX.) (UART)
            - Yes: Reception time-out error
        - No: Timed out? (MAX.) (UART)
          - Yes: Reception time-out error
      - No: Timed out? (MAX.) (UART)
    - No: Timed out? (MAX.) (UART)
      - Yes: Reception time-out error
- No: End of data frame reception
- Yes: Checksum error?
6.4 Reset Command

6.4.1 Processing sequence chart

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

<1> Waits from the previous frame reception until the next command processing starts (wait time $t_{com}$).
<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 $t_{WT0}(\text{MAX.})$).
<8> The status code is checked.

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

6.4.3 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>06H</td>
<td>The command was executed normally and synchronization between the programmer and the V850ES/Kx2 has been established.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>07H</td>
<td>The checksum of the transmitted command frame does not match.</td>
</tr>
<tr>
<td></td>
<td>15H</td>
<td>• A command other than the Status command was received during processing.</td>
</tr>
<tr>
<td></td>
<td></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>
</tbody>
</table>
6.4.4 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?

No

No

Yes

Status frame received?

Timed out?

Yes

Time-out error [C]

No

Yes

Retry count over?

No

Abnormal termination [B]

Normal completion [A]
6.4.5 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_MAX);
        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;
}
```
6.5 Baud Rate Set Command

6.5.1 Processing sequence chart

Baud Rate Set command processing sequence

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

<1> Waits from the previous frame reception until the next command transmission (wait time \(t_{\text{COM}}\)).
<2> The Baud Rate Set command is transmitted by command frame transmission processing.
<3> Waits from command transmission until Reset command transmission (wait time \(t_{\text{WT10}}\)).
<4> The Reset command is transmitted by command frame transmission processing.
<5> 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 \(t_{\text{WT0}}(\text{MAX.})\)).
<6> Since the status code should be ACK, the processing ends normally [A].

6.5.3 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>Data frame reception was timed out. With the V850ES/Kx2, this command also results in errors in the following cases.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Command information (parameter) is invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The command frame includes the checksum error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The data length of the command frame (LEN) is invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The footer of the command frame (ETX) is missing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The Reset command was not detected after setting the baud rate and receiving command frame data for 16 times</td>
</tr>
</tbody>
</table>
6.5.4 Flowchart

- Baud Rate Set command processing
- Wait from previous frame reception until next command transmission
- Command frame transmission processing (Baud Rate Set)
- Wait from command frame transmission until Reset command transmission
- Command frame transmission processing (Reset)
- Status frame received?
  - No
  - Timed out?
    - No
    - Yes: Normal completion [A]
    - Yes: Time-out error [C]
6.5.5 Sample program

The following shows a sample program for Baud Rate Set command processing.

```c
u16 fl_ua_setbaud(u8 brid)
{
    u16   rc;
    u8    br;
    u32   retry;

    switch(brid){
      default:
        case BR_9600: br = 0x03; break;
        case BR_19200: br = 0x04; break;
        case BR_31250: br = 0x05; break;
        case BR_38400: br = 0x06; break;
        case BR_76800: br = 0x07; break;
        case BR_153600: br = 0x08; break;
    }

    fl_cmd_prm[0] = br; // "D01"

    fl_wait(tCOM); // wait before sending command
    put_cmd_ua(FL_COM_SET_BAUDRATE, 2, fl_cmd_prm); // send "Baudrate Set" command

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

    retry = tRS;
    while(1){
        fl_wait(tWT10);

        put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm); // send RESET command

        rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX); // get status frame
        if (rc){
            if (retry--)
                continue;
            else
                return rc;
        }
    }
```
break;       // got ACK !!

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

return rc;
}  
}  

6.6 Oscillating Frequency Set Command

6.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence

Programmer V850ES/Kx2

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

<2> Oscillating Frequency Set command frame transmission

<3> Time-out check for status frame reception

<4> Status frame reception

Time-out occurs

Status frame received within specified time

Time-out error [C]

Reception status [ACK/other than ACK]

Other than ACK

ACK

Abnormal termination [B]

Normal completion [A]
6.6.2 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> 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 $t_{\text{WT9(MAX.)}}$).

<4> The status code is checked.

   When ST1 = ACK: Normal completion [A]
   When ST1 ≠ ACK: Abnormal termination [B]

6.6.3 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>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>–</td>
</tr>
</tbody>
</table>
6.6.4 Flowchart

- Oscillating Frequency Set command processing
  - Wait from previous frame reception until next command transmission
    - Command frame transmission processing (Oscillating Frequency Set)
      - Status frame received?
        - Yes
          - Status = ACK?
            - Yes
              - Normal completion [A]
            - No
              - Timed out?
                - Yes
                  - Time-out error [C]
                - No
                  - No
                    - Abnormal termination [B]
          - No
            - Yes
              - tWT9 (MAX.)
                - Yes
                  - Time-out error [C]
                - No
                  - No
6.6.5 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);

    rc = get_sfrm_ua(fl ua_sfrm, tWT9_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;
}
```
6.7 Chip Erase Command

6.7.1 Processing sequence chart

Chip Erase command processing sequence
6.7.2 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> 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 \( t_{\text{WT1(MAX.)}} \)).

<4> The status code is checked.

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

6.7.3 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>
</tbody>
</table>
|                                 | Protect error | 10H | • Chip erase is prohibited in the security setting.  
|                                 |              |     | • Boot block rewrite is prohibited in the security setting. |
|                                 | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| FLMD error                      | 18H | An erase error has occurred. |
| Write error                     | 1CH | |
| MRG10 error                     | 1AH | |
| MRG11 error                     | 1BH | |
| Time-out error \([C]\)          | – | The status frame was not received within the specified time. |
6.7.4 Flowchart

Chip Erase command processing

Waits from previous frame reception until next command transmission

Command frame transmission processing (Chip Erase)

Status frame received? Yes

No

Timed out? Yes

twT1 (MAX.)

No

Status = ACK? Yes

No

Normal completion [A] Abnormal termination [B]

Yes

Time-out error [C]
6.7.5 Sample program

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

```c
u16 fl_ua_erase_all(void)
{
    u16 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;
}
```

6.8 Block Erase Command

6.8.1 Processing sequence chart

Block Erase command processing sequence

1. Wait from previous frame reception until next command transmission \( t_{COM} \)
2. Block Erase command frame transmission
3. Time-out check for status frame reception \( t_{WT2} \) \( \text{(MAX.)} \) \( x \) number of blocks
4. Status frame reception

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

Reception status [ACK/other than ACK]
- ACK
- Other than ACK

Erasure of specified blocks completed? [Yes/No]
- Yes
- No

Status frame received within specified time

Time-out occurs
6.8.2 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> 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 \( t_{WT2} \) (MAX.) \( \times \) number of blocks).

<4> The status code is checked.

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

6.8.3 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>
</tbody>
</table>
| Abnormal termination [B]       | Parameter error | 05H | • The specified start/end address is out of the flash memory range.  
• The specified start/end address is not the start/end address of the block. |
|                               | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
|                               | Protect error  | 10H | • Block erase is prohibited in the security setting.  
• Boot area is included in the specified range while boot block rewrite is prohibited in the security setting.  
• Chip erase is prohibited in the security setting.  
• Programming is prohibited in the security setting. |
|                               | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
|                               | MRG10 error | 1AH | An erase error has occurred. |
|                               | Time-out error [C] | – | The status frame was not received within the specified time. |
6.8.4 Flowchart

- Block Erase command processing
- Wait from previous frame reception until next command transmission
- Command frame transmission processing (Block Erase)

Status frame received?
- Yes
  - Status = ACK?
    - Yes
      - Normal completion [A]
    - No
      - Abnormal termination [B]
- No
  - Timed out?
    - Yes
      - Time-out error [C]
    - No
      - $t_{WT2}$ (MAX.) x number of blocks
6.8.5 Sample program

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

```c
ATEGORY 6 UART COMMUNICATION MODE

u16 fl_ua_erase_blk(u16 sblk, u16 eblk)
{
    u16 rc;
    u16 block_num;
    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

    block_num = eblk - sblk + 1;

    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, tWT2_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;
}
```
6.9 Programming Command

6.9.1 Processing sequence chart

Programming command processing sequence
6.9.2 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. 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 $t_{WT3}(\text{MAX.})$).
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 $t_{FDB}$).
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 $t_{WT4}(\text{MAX.})$).
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 $t_{WT5}(\text{MAX.}) \times \text{number of blocks}$).
10. The status code is checked.
    - When ST1 = ACK: Normal completion [A]
    - When ST1 $\neq$ ACK: Abnormal termination [E]
### 6.9.3 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  The command was executed normally and the user data was written normally.</td>
</tr>
</tbody>
</table>
| Abnormal termination[B]         | Parameter error             | 05H  • The specified start/end address is not the start/end address of the block.  
|                                 |                           |     • The data length is not 64-word units.                                    |
| Checksum error                  | 07H  The checksum of the transmitted command frame does not match.          |
| Protect error                   | 10H  • Write is prohibited in the security setting.                         
|                                 |                           |     • Boot block rewrite is prohibited in the security setting.                |
| Negative acknowledgment(NACK)   | 15H  Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error[C]               | –  The status frame was not received within the specified time.            |
| Abnormal termination[D]         | FLMD error             | 18H (ST2)  A write error has occurred.                                    |
| Write error                     | 1CH (ST2)  A write error has occurred.                                      |
| Abnormal termination[E]         | MRG11 error            | 1BH  An internal verify error has occurred.                                  |
6.9.4 Flowchart

```
Programming command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Programming)

Status frame received?
  No
  Timed out?
    No
    Status = ACK?
      Yes
      Abnormal termination [B]
      No
      Abnormal termination [B]
    Yes
    Time-out error [C]
  Yes
  ST1 = ACK?
    Yes
    Abnormal termination [B]
    No
    ST2 = ACK?
      Yes
      Abnormal termination [D]
      No
      All data frames transmitted?
        Yes
        Normal completion [A]
        No
        Status frame received?
          Yes
          Time-out?
            No
            Status = ACK?
              Yes
              Time-out error [C]
              No
              Abnormal termination [E]
          No
          Abnormal termination [E]
          x number of blocks
        No
```
6.9.5 Sample program

The following shows a sample program for Programming command processing.

```c
/* Write command */
#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_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]
    }

    /***********************************************************************/
    /* 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;

f1_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 (f1_st2_ua != FLST_ACK){  // ST2 = ACK ?
    rc = decode_status(f1_st2_ua); // No
    return rc;    // case [D]
}
if (is_end)
    break;

/************************************************/
/*     Check internally verify                        */
/************************************************/
rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX_UA * 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;
}
6.10 Verify Command

6.10.1 Processing sequence chart

Verify command processing sequence

Programmer

V850ES/Kx2

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

<2> Verify command frame transmission<br>

<3> Time-out check for status frame reception tWT6 (MAX.)<br>

<4> Status frame reception<br>

<5> Wait from previous frame reception until next data frame transmission<br>

<6> Data frame (user data for verify) transmission<br>

<7> Time-out check for status frame reception tWT3 (MAX.)<br>

<8> Status frame reception (ST1/ST2)<br>

Reception status [ACK/other than ACK]<br>

Abnormal termination [B] Other than ACK<br>

Abnormal termination [D] All data frames transmitted? [Yes/No]<br>

Time-out occurs Time-out occurs<br>

Normal completion [A] Yes<br>

No Go to <5>
6.10.2 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> 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 \( t_{\text{WT6}}(\text{MAX.}) \)).

<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 \( t_{\text{FD3}} \)).

<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 \( t_{\text{WT7}}(\text{MAX.}) \)).

<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: 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 \( \neq \) ACK: Abnormal termination [D]

6.10.3 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 not the start/end address of the block.</td>
</tr>
<tr>
<td>Parameter error</td>
<td>07H</td>
<td>The checksum of the transmitted command frame or data frame does not match.</td>
</tr>
<tr>
<td>Checksum error</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length [LEN] or no ETX).</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td></td>
<td></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>
### 6.10.4 Flowchart

1. **Verify command processing**

2. **Wait from previous frame reception until next command transmission**
   - **tCOM**
   - **tWT6 (MAX.)**
   - **Command frame transmission processing (Verify)**

3. **Status frame received?**
   - **Yes**
   - **ST1 = ACK?**
     - **Yes**
     - Abnormal termination [B]
     - Abnormal termination [B]
     - Abnormal termination [D]
   - **No**
   - **Timed out?**
     - **Yes**
     - Time-out error [C]
     - Time-out error [C]
     - No

4. **Data frame transmission processing (User program)**

5. **Status frame received?**
   - **Yes**
   - **ST1 = ACK?**
     - **Yes**
     - Normal completion [A]
     - Abnormal termination [B]
   - **No**
   - **timt7 (MAX.)**
     - **Yes**
     - Time-out error [C]
     - Time-out error [C]
     - No

6. **Wait from previous frame reception until next data frame transmission**
   - **tFD3**
   - **User program**
   - **ST2 = ACK?**
     - **Yes**
     - Abnormal termination [B]
     - Abnormal termination [D]
     - No

7. **All data frames transmitted?**
   - **Yes**
   - Normal completion [A]
   - **No**

---

Application Note U17885EJ2V1AN

89
6.10.5 Sample program
The following shows a sample program for Verify command processing.

```c
u16 fl_ua_verify(u32 top, u32 bottom, u8 *buf)
{
    u16 rc;
    u32 send_head, send_size;
    bool is_end;

    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_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_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_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]
    }

    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, 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

rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX);  // get status frame
switch(rc) {
    case FLC_NO_ERR:   break; // continue
    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]
6.11 Block Blank Check Command

6.11.1 Processing sequence chart

Block Blank Check command processing sequence

1. Wait from previous frame reception until next command transmission

2. Block Blank Check command frame transmission

3. Time-out check for status frame reception

4. Status frame reception

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

Time-out occurs

Status frame received within specified time

INBT \( \text{MAX.} \) \( x \) number of blocks
6.11.2 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> 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 $t_{WT8(MAX.)} \times$ number of blocks).

<4> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 ≠ ACK: Abnormal termination [B]

6.11.3 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>
</tbody>
</table>
| Abnormal termination [B]       | Parameter error | 05H | • The specified start/end address is out of the flash memory range.  
   • The specified start/end address is not the start/end address of the block. |
|                                | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
|                                | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
|                                | MRG11 error | 1BH | The specified block in the flash memory is not blank. |
|                                | Time-out error [C] | – | The status frame was not received within the specified time. |
6.11.4 Flowchart

- Block Blank Check command processing
- Wait from previous frame reception until next command transmission $t_{COM}$

- Command frame transmission processing (Block Blank Check)

- Status frame received? No → Timed out? No
  - Time-out error [C]
  - $t_{WT8}$ (MAX.) x number of blocks

- Status frame received? Yes → Status = ACK? No
  - Time-out error [C]

- Status frame received? Yes → Status = ACK? Yes
  - Normal completion [A]

- Status frame received? Yes → Status = ACK? No
  - Abnormal termination [B]
6.11.5 Sample program

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

```c
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;
}
```
6.12 Silicon Signature Command

6.12.1 Processing sequence chart

Silicon Signature command processing sequence

Programmer

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

<2> Silicon Signature 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 (silicon signature) reception processing

V850ES/Kx2

ACK

Reception status [ACK/other than ACK]

Other than ACK

Abnormal termination [B]

Time-out occurs

Status frame received within specified time

No

Data frame error [D]

Time-out error [C]

Normal completion [A]

Normal data frame? [Yes/No]

<6> Data frame error [D]

<5> Time-out occurs

<4> Status frame received within specified time

<3> Time-out occurs

<2> Silicon Signature command frame transmission

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

<6> Data frame error [D]
6.12.2 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> 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 $t_{WT1}(MAX.)$).

<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 (silicon signature data) reception.
   If a time-out occurs, a time-out error [C] is returned (time-out time $t_{FD2}(MAX.)$).

<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]

6.12.3 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>Time-out error [C]</td>
<td>–</td>
<td>The status frame or data 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 silicon signature data does not match.</td>
</tr>
</tbody>
</table>
6.12.4 Flowchart

Silicon Signature command processing

Wait from previous frame reception until next command transmission

Command frame transmission processing (Silicon Signature)

Status frame received? Yes

Status = ACK? Yes

Abnormal termination [B]

Data frame (silicon signature) received? No

Normal data frame? Yes

Normal completion [A]

Data frame error [D]

No

No

Timed out? Yes

tWT11 (MAX.)

Time-out error [C]

No

Timed out? No

tFD2 (MAX.)

Status = ACK? Yes

Abnormal termination [B]

No

No

Data frame error [D]
6.12.5 Sample program

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

```c
ul6 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_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]
    }

    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX);  // 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]
}
```
6.13 Version Get Command

6.13.1 Processing sequence chart

Version Get command processing sequence
6.13.2 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> 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 $t_{WT12} (\text{MAX.})$).

<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 $t_{FD2} (\text{MAX.})$).

<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]

6.13.3 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 or data 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 does not match.</td>
</tr>
</tbody>
</table>
6.13.4 Flowchart

- **Version Get command processing**
- **Wait from previous frame reception until next command transmission**

**Command frame transmission processing (Version Get)**

- **Status frame received?**
  - No
    - **Status = ACK?**
      - No
        - **Abnormal termination [B]**
      - Yes
        - **Data frame (version data) received?**
          - No
            - **Timed out?**
              - No
                - **Normal data frame?**
                  - No
                    - **Normal completion [A]**
                  - Yes
                    - **Data frame error [D]**
          - Yes
            - **Time-out error [C]**

- Yes
  - **Timed out?**
    - No
      - **tCOM**
    - Yes
      - **tWT12 (MAX.)**

**Note:**
- [A] Normal completion
- [B] Abnormal termination
- [C] Time-out error
- [D] Data frame error
6.13.5 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_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]
    }

    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX); // 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]
}
```
6.14 Checksum Command

6.14.1 Processing sequence chart

Checksum command processing sequence

Programmer

V850ES/Kx2

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

<2> Checksum command frame transmission

<3> Time-out check for status frame reception

<4> Status frame reception

<5> Time-out check for status frame reception

<6> Data frame (checksum data) reception

Normal data frame? [Yes/No]

No

Data frame error [D]

Yes

Normal completion [A]

Reception status [ACK/other than ACK]

ACK

Other than ACK

Abnormal termination [B]

Time-out error [C]

tCOM

tWT16 (MAX.)

x number of blocks

tFD1 (MAX.)

Time-out check for status frame reception

Status frame received within specified time

Data frame received within specified time

Time-out occurs

Time-out occurs

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

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

<2> The 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 \( t_{\text{WT1}} \times \text{MAX.} \)).

<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 (checksum data) reception.
    If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD1}} \times \text{MAX.} \) \times \text{number of blocks}).

<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]

6.14.3 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) 06H</td>
<td>The command was executed normally and checksum data was acquired normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>Parameter error 05H</td>
<td>The specified start/end address is not a fixed address in block units (2 KB) starting from the top of the flash memory.</td>
</tr>
<tr>
<td></td>
<td>Checksum error 07H</td>
<td>The checksum of the transmitted command frame does not match.</td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK) 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 or data 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 does not match.</td>
</tr>
</tbody>
</table>
6.14.4 Flowchart

- Checksum command processing

  Wait from previous frame reception until next command transmission

  Command frame transmission processing (Checksum)

  Status = ACK?

    Yes
    - Normal completion [A]
    - Time-out error [C]

    No
    - Status frame received?
      - No
        - Timed out?
          - Yes
            - Time-out error [C]
          - No
            - Status = ACK?
              - Yes
                - Abnormal termination [B]
              - No
                - Data frame (checksum data) received?
                  - Yes
                    - Normal data frame?
                      - Yes
                        - Time-out error [C]
                      - No
                        - Data frame error [D]
                  - No
                    - Normal completion [A]

                  Normal completion [A]

                  Time-out error [C]

                  x number of blocks
6.14.5 Sample program
The following shows a sample program for Checksum command processing.

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

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

        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_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]
        }

        rc = get_dfrm_ua(fl_rxdata_frm, (tFD1_MAX * block_num)); // get Checksum
        if (rc){
            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]
    }
```
6.15 Security Set Command

6.15.1 Processing sequence chart

Security Set command processing sequence
6.15.2 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 \( t_{\text{WT13}} \,(\text{MAX.}) \)).

<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 \( t_{\text{FD3}} \)).

<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 \( t_{\text{WT14}} \,(\text{MAX.}) \)).

<8> The status code is checked.
   - When \( ST1 = \text{ACK} \): Proceeds to <9>.
   - When \( ST1 \neq \text{ACK} \): 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 \( t_{\text{WT15}} \,(\text{MAX.}) \)).

<10> The status code is checked.
   - When \( ST1 = \text{ACK} \): Normal completion \([A]\)
   - When \( ST1 \neq \text{ACK} \): Abnormal termination \([E]\)

6.15.3 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 The command was executed normally and security setting was performed normally.</td>
</tr>
<tr>
<td>Abnormal termination ([B])</td>
<td>Checksum error</td>
<td>07H The checksum of the transmitted command frame or data frame does not match.</td>
</tr>
<tr>
<td></td>
<td>Protect error</td>
<td>10H • An attempt was made to enable a flag that was already prohibited in the setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Boot block rewrite is prohibited in the security setting.</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>Time-out error ([C])</td>
<td>–</td>
<td>The status frame or data frame was not received within the specified time.</td>
</tr>
<tr>
<td>Abnormal termination ([D])</td>
<td>FLMD error</td>
<td>18H A write error has occurred.</td>
</tr>
<tr>
<td></td>
<td>MGR10 error</td>
<td>1AH</td>
</tr>
<tr>
<td></td>
<td>Write error</td>
<td>1CH</td>
</tr>
<tr>
<td>Abnormal termination ([E])</td>
<td>MRG11 error</td>
<td>18H An internal verify error has occurred.</td>
</tr>
</tbody>
</table>
6.15.4 Flowchart

Security Set command processing

Command frame transmission processing (Security Set)

Status frame received? No

Timed out? No

Yes

Time-out error [C]

Status = ACK? No

Yes

Abnormal termination [B]

Status frame received? No

Timed out? No

Yes

Time-out error [C]

Status = ACK? No

Yes

Abnormal termination [D]

Status frame received? No

Timed out? No

Yes

Time-out error [C]

Status = ACK? No

Yes

Abnormal termination [E]

Status = ACK? Yes

Normal completion [A]

Abnormal termination [E]
6.15.5 Sample program

The following shows a sample program for Security Set command processing.

```c
/* *******************************************************************/
/* * */
/* * Set security flag command */
/* * */
/* *******************************************************************/
/* [i] u8 scf ... Security flag data */
/* [i] u8 bot ... Boot Block Number */
/* [r] u16 ... error code */
/* *******************************************************************/

u16 fl_ua_setscf(u8 scf, u8 bot)
{
    u16 rc;

    /*************************************************************************/
    /* set params */
    /*************************************************************************
    fl_cmd_prm[0] = 0x00; // 1st byte must be 0x00
    fl_cmd_prm[1] = 0x00; // 2nd byte must be 0x00

    fl_txdata_frm[0] = (scf |= 0b11100000); // "FLG" (upper 5bits must be '1' (to make sure)
    fl_txdata_frm[1] = bot; // "BOT"

    /*************************************************************************/
    /* send command */
    /*************************************************************************
    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_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]
    }
    /*************************************************************************/
    /* send data frame (security setting data) */
    /*************************************************************************
    fl_wait(tFD3_UA);
    put_dfrm_ua(2, fl_txdata_frm, true); // send security setting(FLAG) & BOT data

    rc = get_sfrm_ua(fl_ua_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]
    }
```
rc = get_sfrm_uia(f1_uia_sfrm, tWT15_MAX);  // get status frame

switch(rc) {
    case FLC_NO_ERR:    return rc;  break;  // case [A]
    case FLC_DPTO_ERR:  return rc;  break;  // case [C]
    default:            return rc;  break;  // case [B]
}
return rc;
}
6.16 Read Command

6.16.1 Processing sequence chart

Read command processing sequence

- Time-out occurs
- Time-out error [C]
- Abnormal termination [B]
- Time-out occurs
- Reception status [ACK/other than ACK]
- ACK
- Data frame error [D]
- Data frame received within specified time
- Time-out error [C]
- Data frame (user data) reception
- Reception error occurred? [Yes/No]
- No
- To <6>
- Yes
- Status (NACK) frame transmission
- All data frames received? [Yes/No]
- No
- To <8>
- Yes
- Status (ACK) frame transmission
- Normal completion [A]
- Time-out occurs
- Time-out error [C]
6.16.2 Description of processing sequence

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

<2> The Read 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 \( t_{\text{WT17}} \text{ (MAX.)} \)).

<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 reception of the data frame reception result (user data).
   If a time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT19}} \text{ (MAX.)} \)).

<6> The received data frame is checked.

   If data frame is normal: Proceeds to <9>.
   If data frame is abnormal: Proceeds to <7>.

<7> Waits from the previous frame reception until the next status (NACK) frame transmission (wait time \( t_{\text{WT19}} \)).

<8> The NACK frame is transmitted by data frame transmission processing.
   \( \rightarrow \) A data frame error [D] is returned.

<9> Waits from the previous frame reception until the next status (ACK) frame transmission (wait time \( t_{\text{WT19}} \)).

<10> The ACK frame is transmitted by data frame transmission processing.
   When reception of all data frames is completed, normal completion [A] is returned.
   If there still remain data frames to be received, the processing re-executes the sequence from <5>.

6.16.3 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>Data frame error [D]</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
6.16.4 Flowchart

- **Read command processing**
- **Wait from previous frame reception until next command transmission**
- **Command frame transmission processing (Read)**
  - **ST1 = ACK?**
    - Yes: **Normal completion [A]**
    - No: **Wait from previous frame reception until next status frame transmission**
      - **TWT19**
      - **Data frame received?**
        - Yes: **Status (NACK) frame transmission**
        - No: **Data frame error [D]**
      - **Time-out error [C]**
- **No**: **Status frame received?**
  - Yes: **Wait from previous frame reception until next status frame transmission**
    - **TWT19**
    - **Data frame received?**
      - Yes: **Status (NACK) frame transmission**
    - No: **Data frame error [D]**
  - **Time-out error [C]**
  - **Abnormal termination [B]**
- **No**: **Data frame (user program) reception processing**
  - **Wait from previous frame reception until next status frame transmission**
    - **TWT19**
    - **Data frame received?**
      - Yes: **Status (NACK) frame transmission**
      - No: **Data frame error [D]**
    - **Time-out error [C]**
  - **Abnormal termination [B]**
  - **No**: **Data frame reception error?**
    - Yes: **Wait from previous frame reception until next status frame transmission**
      - **TWT19**
      - **Status (NACK) frame transmission**
    - No: **Data frame error [D]**
  - **Time-out error [C]**
  - **Abnormal termination [B]**
  - **No**: **All data frames received?**
    - Yes: **Normal completion [A]**
    - No: **Wait from previous frame reception until next status frame transmission**
      - **TWT19**
      - **Status (NACK) frame transmission**
    - **Time-out error [C]**
    - **Abnormal termination [B]**
6.16.5 Sample program

The following shows a sample program for Read command processing.

```c
u16 fl_ua_read(u32 top, u32 bottom)
{
    u16 rc;
    u32 read_head;
    u16 len;
    u8 hooter;

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

    put_cmd_ua(FL_COM_READ, 7, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT17_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:   break;
        case FLC_DFTO_ERR: return rc; break; // case [C]
        default:   return rc; break; // case [B]
    }

    read_head = top;
    while(1){
        rc = get_dfrm_ua(fl_rxdata_frm, tWT18_MAX); // get ROM data from FLASH
        switch(rc) {
            case FLC_NO_ERR:   break; // continue
            case FLC_DFTO_ERR: return rc; break; // case [C]
            case FLC_RX_DFSUM_ERR:
                return rc; break; // case [B]
            default:
                fl_wait(tWT19);
                // fl_wait(5000);
                put_sfrm_ua(FLST_NACK); // send status(NACK) frame
                return rc;
                break;
        }
    }
}
```
}  

\fl\_wait(tWT19);  
// \fl\_wait(5000);  
put_sfrm\_ua(FLST\_ACK);  

/*****************************/  
/*  save ROM data */  
/*****************************/  
if ((len = fl\_rxdata\_frm[OFS\_LEN]) == 0) // get length  
    len = 256;  

memcpy(read\_buf+read\_head, fl\_rxdata\_frm+2, len); // save to external RAM  

read\_head += len;  

/*****************************/  
/*  end check */  
/*****************************/  
hooter = fl\_rxdata\_frm[len + 3];  
if (hooter == FL\_ETB) // end frame ?  
    continue;  // no  
break;  // yes  
}

return FLC\_NO\_ERR;  
}
CHAPTER 7  3-WIRE SERIAL I/O COMMUNICATION MODE WITH HANDSHAKE SUPPORTED (CSI + HS)

7.1 Command Frame Transmission Processing Flowchart
7.2 Data Frame Transmission Processing Flowchart

- **Data frame transmission processing**
- **Data frame header (STX = 02H) transmission**
  - **HS pin = BUSY?**
    - No
    - Data length (LEN) transmission
      - **LEN bytes transmitted?**
        - Yes
        - Transmission time-out error
        - No
        - Transmits 1-byte data
          - **HS pin = BUSY?**
            - Yes
            - **Timed out?**
              - Yes, Transmission time-out error
              - No
            - **b'= CSI(MAX.)**
          - No
          - **Transmit 1-byte data**
            - **HS pin = BUSY?**
              - Yes
              - **Timed out?**
                - Yes, Transmission time-out error
                - No
              - **b'= CSI(MAX.)**
            - No
            - **Transmit 1-byte data**
              - **Checksum data (SUM) transmission**
                - **HS pin = BUSY?**
                  - Yes
                  - **Timed out?**
                    - Yes, Transmission time-out error
                    - No
                  - **b'= CSI(MAX.)**
                - No
                - **Transmit checksum data**
                  - **Last data frame?**
                    - Yes
                    - Transmission of last data frame footer (ETX = 03H)
                    - No
                    - Transmission of footer other than those of last data frame (ETB = 17H)
            - **End of data frame transmission**

Application Note U17885EJ2V1AN
7.3 Data Frame Reception Processing Flowchart

Data frame reception processing

Data frame header (STX = 02H) reception

- HS pin = BUSY?
  - Yes
  - Timed out?
    - Yes
    - Reception time-out error
    - No

  - No
    - Receives 1-byte data

  - LEN bytes received?
    - No
    - Reception time-out error
    - Yes

- HS pin = BUSY?
  - Yes
  - Timed out?
    - Yes
    - Reception time-out error
    - No

- Receives 1-byte data

  - Checksum data (SUM) reception

  - HS pin = BUSY?
    - Yes
    - Timed out?
      - Yes
      - Reception time-out error
      - No

  - Checksum error?
    - Yes
    - Checksum error
    - No

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

    - HS pin = BUSY?
      - Yes
      - Timed out?
        - Yes
        - Reception time-out error
        - No

    - Checksum error?
      - Yes
      - Checksum error
      - No

End of data frame reception
7.4 Status Command

7.4.1 Processing sequence chart

Status command processing sequence

Programmer

<1> Status command frame transmission

<2> BUSY time-out check using HS pin

BUSY release

tSF (MAX.)

<3> Status frame reception

V850ES/Kx2

Time-out occurs

Time-out error [C]

Reception status [ACK/other than ACK]

Other than ACK

ACK

Abnormal termination [B]

Normal completion [A]
7.4.2 Description of processing sequence

<1> The Status command is transmitted by command frame transmission processing.
<2> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{SF} \text{ (MAX.)}$).
<3> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 ≠ ACK: Abnormal termination [B]

7.4.3 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>FLMD error</td>
<td>18H</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>Processing timed out due to the busy status at the HS pin.</td>
</tr>
</tbody>
</table>
### 7.4.4 Flowchart

**Status command processing**

**Command frame transmission processing (Status)**

**HS pin = BUSY?**
- **Yes**
- **No**

**HS timed out?**
- **Yes**
- **No**

**Status = ACK?**
- **Yes**
- **No**

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

**Time-out error [C]**

**tsf (MAX.)**
7.4.5 Sample program

The following shows a sample program for Status command processing.

```c
static u16 fl_hs_getstatus(void)
{
    u16 rc;
    u32 retry = 0;

    rc = put_cmd_hs(FL_COM_GET_STA, 1, fl_cmd_prm); // send "Status" command
    if (rc)
        return rc; // case [C]
    if (hs_busy_to(tSF_MAX)) // HS-Busy t.o. ?
        return FLC_HSTO_ERR; // t.o. detected : case [C]
    if (rc = get_sfrm_hs(fl_rxdata_frm))
        return rc; // case [C] or [B(checksum error)]

    rc = decode_status(fl_st1); // decode return code
    return rc; // case [A] or [B]
}
```
7.5 Reset Command

7.5.1 Processing sequence chart

Reset command processing sequence

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

<1> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{com}(\text{MAX.})$).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT0}(\text{MAX.})$).

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

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

<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>
</tbody>
</table>
|                               | Negative acknowledgment (NACK) | 15H | • A command other than the Status command was received during processing.  
|                               |                          |     | • Command frame data is abnormal (such as invalid data length (LEN) or no ETX).  |
| Time-out error [C]             | –                       | Status check processing timed out. Processing timed out due to the busy status at the HS pin. |

7.5.3 Status at processing completion
7.5.4 Flowchart

- Reset command processing
- Command frame transmission processing (Reset)
  - HS pin = BUSY?
  - No
  - Status check processing
  - Result of status check processing = Abnormal termination?
    - Yes
      - Abnormal termination [B]
    - No (normal completion)
  - Result of status check processing = Time-out error?
    - Yes
      - Time-out error [C]
    - No
      - Retry count over?
        - Yes
          - Abnormal termination [B]
        - No
          - Result of status check processing = Abnormal termination?
            - Yes
              - Time-out error [C]
            - No
              - Normal completion [A]
7.5.5 Sample program

The following shows a sample program for Reset command processing.

```c
ul6  fl_hs_reset(void)
{
    ul6  rc;
    u32  retry;

    for (retry = 0; retry < tRS; retry++) {
        if (hs_busy_to(tCOM_MAX))
            return FLC_HSTO_ERR;  // t.o. detected :case [C]

        rc = put_cmd_hs(FL_COM_RESET, 1, fl_cmd_prm);  // send "Reset" command
        if (rc)
            return rc;  // case [C]

        if (hs_busy_to(tWT0_MAX))
            return FLC_HSTO_ERR;  // t.o. detected :case [C]

        rc = fl_hs_getstatus();  // get status frame
        if (rc == FLC_ACK)  // ST1 = ACK ?
            break;  // case [A]
        //continue;  // case [B] (if exit from loop)
    }

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

    return rc;
}
```

/*********************/
/*
/* Reset command (CSI-HS)
/*
/*
/*********************/
/* [r] ul6 ... error code */
/*********************/
ul6  fl_hs_reset(void)
{
    ul6  rc;
    u32  retry;

    for (retry = 0; retry < tRS; retry++) {
        if (hs_busy_to(tCOM_MAX))
            return FLC_HSTO_ERR;  // t.o. detected :case [C]

        rc = put_cmd_hs(FL_COM_RESET, 1, fl_cmd_prm);  // send "Reset" command
        if (rc)
            return rc;  // case [C]

        if (hs_busy_to(tWT0_MAX))
            return FLC_HSTO_ERR;  // t.o. detected :case [C]

        rc = fl_hs_getstatus();  // get status frame
        if (rc == FLC_ACK)  // ST1 = ACK ?
            break;  // case [A]
        //continue;  // case [B] (if exit from loop)
    }

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

    return rc;
}
7.6 Oscillating Frequency Set Command

7.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence

Programmer

<1> BUSY time-out check using HS pin

<2> Oscillating Frequency Set command frame transmission

<3> BUSY time-out check using HS pin

<4> Status check processing

<5> Result of status check processing

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

Normal completion
Abnormal termination

Time-out error

V850ES/Kx2

BUSY release

BUSY release

BUSY release

BUSY release

Time-out occurs

Time-out occurs

Time-out occurs

Time-out occurs
7.6.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{com}^{\text{MAX.}} \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{WT9}^{\text{MAX.}} \)).

<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.

7.6.3 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>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>Processing timed out due to the busy status at the HS pin.</td>
</tr>
</tbody>
</table>
7.6.4 Flowchart

Oscillating Frequency Set command processing

HS pin = BUSY?
Yes

Command frame transmission processing (Oscillating Frequency Set)

Timed out?
No

Status check processing

Yes

Timed out?
No

Yes

Time-out error [C]

No

Yes

Time-out error [C]

No

Normal completion?
Yes

Normal completion [A]

No

Time-out error [C]

No

Abnormal termination [B]
7.6.5 Sample program

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

```c
u16 fl_hs_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"

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected : case [C]

    if (rc = put_cmd_hs(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm))
        return rc; // case [C]

    if (hs_busy_to(tWT9_MAX))
        return FLC_HSTO_ERR; // t.o. detected : case [C]

    rc = fl_hs_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR: return rc; break; // case [A]
        case FLC_HSTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }
    return rc;
}
```
7.7 Chip Erase Command

7.7.1 Processing sequence chart

Chip Erase command processing sequence
7.7.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time tCOM(MAX.)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time tWT1(MAX.)).

<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.

7.7.3 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></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negative acknowledgment (NACK)</td>
<td>15H</td>
</tr>
<tr>
<td></td>
<td>FLMD error</td>
<td>18H</td>
</tr>
<tr>
<td></td>
<td>Write error</td>
<td>1CH</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>Time-out error [C]</td>
<td>–</td>
<td>Processing timed out due to the busy status at the HS pin.</td>
</tr>
</tbody>
</table>
7.7.4 Flowchart

```
Chip Erase command processing

HS pin = BUSY?  
Yes   No
Timed out?  
Yes   No

tCOM (MAX.)

Command frame transmission processing (Chip Erase)

HS pin = BUSY?  
Yes   No
Timed out?  
Yes   No

bWT1 (MAX.)

Status check processing

Time-out error?  
Yes   No

Time-out error [C]

Normal completion?  
Yes   No

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

Application Note U17885EJ2V1AN 135
### 7.7.5 Sample program

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

```c
/** ******************************************************/
/* Erase all(chip) command (CSI-HS) */
/* ******************************************************/
/** [r] u16     ... error code */
/** ******************************************************/
u16 fl_hs_erase_all(void)
{
    u16 rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected

    if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))
        return rc; // send "Chip Erase" command

    if (hs_busy_to(tWT1_MAX))
        return FLC_HSTO_ERR; // case [C]

    rc = fl_hs_getstatus(); // get status frame
    // switch(rc) {
    //   case FLC_NO_ERR: return rc; break; // case [A]
    //   case FLC_HSTO_ERR: return rc; break; // case [C]
    //   default: return rc; break; // case [B]
    // }
    return rc;
}
```
7.8 Block Erase Command

7.8.1 Processing sequence chart

Block Erase command processing sequence
7.8.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time t_{COM}(MAX.)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time t_{WT2}(MAX.) × 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 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.

7.8.3 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>
</tbody>
</table>
| Abnormal termination [B]       | Parameter error | 05H | - The specified start/end address is out of the flash memory range.  
- The specified start/end address is not the start/end address of the block. |
|                                | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
|                                | Protect error  | 10H | - Block erase is prohibited in the security setting.  
- The boot area is included in the specified range while boot block rewrite is prohibited in the security setting.  
- Chip erase is prohibited in the security setting.  
- Programming is prohibited in the security setting. |
|                                | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
|                                | MRG10 error | 1AH | An erase error has occurred. |
|                                | Time-out error [C] | – | Processing timed out due to the busy status at the HS pin. |
7.8.4 Flowchart

Block Erase command processing

HS pin = BUSY? Yes

Command frame transmission processing (Block Erase)

HS pin = BUSY? Yes

Status check processing

Timed out? Yes

Normal completion? Yes

Erasure of specified blocks completed? No

Abnormal termination [B]

Normal completion [A]

Normal completion [A]

Timed out? No

Time-out error? Yes

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]
7.8.5 Sample program

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

```c
u16 fl_hs_erase_blk(u16 sblk, u16 eblk)
{
    u16 rc;
    u16 block_num;
    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
    block_num = eblk - sblk + 1;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm))
        return rc;  // send "Block Erase" command  // case [C]

    if (hs_busy_to(tWT2_MAX * block_num))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    rc = fl_hs_getstatus();  // get status frame
    //switch(rc) {
    //    case FLC_NO_ERR: return rc; break;  // case [A]
    //    case FLC_HSTO_ERR: return rc; break;  // case [C]
    //    default: return rc; break;  // case [B]
    //}
    return rc;
}
```
7.9 Programming Command

7.9.1 Processing sequence chart

Programming command processing sequence

Application Note  U17885EJ2V1AN  141
7.9.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(\text{MAX.)})$).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{W[3]}(\text{MAX.)})$).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD[3]}(\text{MAX.)})$).

<7> User data is transmitted by data frame transmission processing.

<8> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{W[4]}(\text{MAX.)})$).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{W[5]}(\text{MAX.)} \times \text{number of blocks})$).

<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.
### 7.9.3 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>
</tbody>
</table>
| Abnormal termination [B]        | Parameter error | 05H | - The specified start/end address is not the start/end address of the block.  
- The data length is not 64-word units. |
|                                | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
|                                | Protect error  | 10H | - Write is prohibited in the security setting.  
- Boot block rewrite is prohibited in the security setting. |
|                                | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error [C]              | –            | –          | Processing timed out due to the busy status at the HS pin. |
| Abnormal termination [D]        | FLMD error   | 18H (ST2) | A write error has occurred. |
|                                | Write error  | 1CH (ST2) | A write error has occurred. |
| Abnormal termination [E]        | MRG11 error  | 1BH        | An internal verify error has occurred. |
7.9.4 Flowchart
7.9.5 Sample program

The following shows a sample program for Programming command processing.

```c
/*****************************************************************
/* */
/* Write command (CSI-HS) */
/* */
/*****************************************************************/
/* [i] u32 top ... start address */
/* [i] u32 bottom ... end address */
/* [r] u16 ... error code */
/*****************************************************************/
u16 fl_hs_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 */
  /***************************************************************************/
  if (hs_busy_to(tCOM_MAX))
    return FLC_HSTO_ERR;  // t.o. detected

  if (rc = put_cmd_hs(FL_COM_WRITE, 7, fl_cmd_prm)) // send "Programming" command
    return rc;        // t.o. detected

  if (hs_busy_to(tWT3_MAX))
    return FLC_HSTO_ERR; // t.o. detected

  rc = fl_hs_getstatus();  // get status frame
  switch(rc) {
    case FLC_NO_ERR:    break; // continue
    // case FLC_HSTO_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 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;

if (hs_busy_to(tFD3_MAX))   // t.o. check before sending data frame
    return FLC_HSTO_ERR;   // t.o. detected

if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
    // send user data
    return rc;  // error detected

if (hs_busy_to(tWT4_MAX))
    return FLC_HSTO_ERR;   // t.o. detected

rc = fl_hs_getstatus();   // get status frame
switch(rc) {
    case FLC_NO_ERR:
        break; // continue
    case FLC_HSTO_ERR:
        return rc; // case [C]
    default:
        return rc; // 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
}

/*************************************************/
/* Check internally verify */
/*************************************************/
if (hs_busy_to(tWT5_MAX_HS * block_num))
    return FLC_HSTO_ERR;   // t.o. detected

rc = fl_hs_getstatus();   // get status frame
switch(rc) {
    case FLC_NO_ERR:
        return rc; // case [A]
    case FLC_HSTO_ERR:
        return rc; // case [C]
    default:
        return rc; // case [B]
}
    return rc;
7.10 Verify Command

7.10.1 Processing sequence chart

Verify command processing sequence

Programmer V850ES/Kx2

<1> BUSY time-out check using HS pin
<2> Verify command frame transmission
<3> BUSY time-out check using HS pin
<4> Status check processing
<5> Result of status check processing
<6> BUSY time-out check using HS pin
<7> Data frame (user data for verify) transmission
<8> BUSY time-out check using HS pin
<9> Status check processing
<10> Result of status check processing (ST1/ST2)

<1> BUSY release
<2> Verify command frame transmission
<3> BUSY release
<4> Status check processing
<5> Result of status check processing
<6> BUSY release
<7> Data frame (user data for verify) transmission
<8> BUSY release
<9> Status check processing
<10> Result of status check processing (ST1/ST2)

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

Abnormal termination

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

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

ACK

All data frames transmitted? [Yes/No]

Yes

Normal completion [A]

No

Go to <6>

Time-out error [C]

Time-out occurs

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out error [C]

Time-out occurs

Time-out occurs

Time-out occurs

Time-out occurs

Time-out occurs

Abnormal termination [D]

Normal completion

Other than ACK

Abnormal termination [B]
7.10.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{COM}} (\text{MAX.}) \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT6}} (\text{MAX.}) \)).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD3}} (\text{MAX.}) \)).

<7> User data for verifying is transmitted by data frame transmission processing.

<8> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT7}} (\text{MAX.}) \)).

<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 = 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>.
      • When ST2 \( \neq \) ACK: Abnormal termination [D]

7.10.3 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>Time-out error [C]</td>
<td>–</td>
<td>Processing timed out due to the busy status at the HS pin.</td>
</tr>
<tr>
<td>Abnormal termination [D]</td>
<td>Verify error</td>
<td>0FH (ST2)</td>
</tr>
</tbody>
</table>
7.10.4 Flowchart

Verify command processing

HS pin = BUSY?

Yes

Command frame transmission processing (Verify)

No

Timed out?

Yes

Time-out error [C]

No

Time-out error [C]

Normal completion?

No

Abnormal termination [B]

Yes

Abnormal termination [B]

Normal completion?

No

Abnormal termination [B]

Yes

St2 = ACK?

Yes

Normal completion [A]

No

Abnormal termination [D]

All data frames transmitted?

No

Abnormal termination [B]

Yes

Abnormal termination [D]

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

```c
/* Verify command (CSI-HS) */
/*****************************************************************************************************/
/* [i] u32 top ... start address */
/* [i] u32 bottom ... end address */
/* [i] u8  *buf ... pointer to verify data buffer */
/* [r] u16 ... error code */
/*****************************************************************************************************/

u16 fl_hs_verify(u32 top, u32 bottom, u8 *buf)
{
    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 */
    /**************************************************************************************************/
    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected
    if (rc = put_cmd_hs(FL_COM_VERIFY, 7, fl_cmd_prm)) // send "Verify" command
        return rc; // error detected
    if (hs_busy_to(tWT6_MAX))
        return FLC_HSTO_ERR; // t.o. detected
    rc = fl_hs_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR:
            break; // continue
        case FLC_HSTO_ERR:
            return rc; // error detected
        default:
            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, buf+send_head, send_size);// set data frame payload
send_head += send_size;

if (hs_busy_to(tFD3_MAX))
    return FLC_HSTO_ERR;   // t.o. detected
if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
    // send user data
    return rc;    // error detected
if (hs_busy_to(tWT7_MAX))
    return FLC_HSTO_ERR;   // t.o. detected
rc = fl_hs_getstatus();   // get status frame
switch(rc) {
    case FLC_NO_ERR:    break; // continue
    // case FLC_HSTO_ERR:  return rc; break; // case [C]
    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
return FLC_NO_ERR; // case [A]
7.11 Block Blank Check Command

7.11.1 Processing sequence chart

Block Blank Check command processing sequence

Programmer

<1> BUSY time-out check using HS pin

<2> Block Blank Check command frame transmission

<3> BUSY time-out check using HS pin

V850ES/Kx2

<4> Status check processing

<5> Result of status check processing

Result

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

Abnormal termination

Normal completion

Time-out error

Abnormal termination [B]

Normal completion [A]

Time-out error [C]

Time-out occurs

BUSY release

Time-out occurs

BUSY release

BUSY time-out check using HS pin

BUSY time-out check using HS pin

tWT8 (MAX.) x number of blocks

tCOM (MAX.)
7.11.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{com}(\text{MAX.})$).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{W8}(\text{MAX.})$).

<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.

7.11.3 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>
</tbody>
</table>
| Abnormal termination [B]        | Parameter error          | 05H | • The specified start/end address is out of the flash memory range.  
                                |                          |          | • The specified start/end address is not the start/end address of the block. |
| Checksum error                  | 07H | The checksum of the transmitted command frame does not match. |
| Negative acknowledgment (NACK)  | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| MRG11 error                     | 18H | The specified block in the flash memory is not blank. |
| Time-out error [C]              | –  | Processing timed out due to the busy status at the HS pin. |
7.11.4 Flowchart

Block Blank Check command processing

HS pin = BUSY?

Yes

Command frame transmission processing (Block Blank Check)

HS pin = BUSY?

Yes

Timed out?

Yes
t_wt (MAX.)
x number of blocks

Time-out error [C]

No

Status check processing

Timed out?

Yes
t_wt (MAX.)
x number of blocks

Time-out error [C]

No

Time-out error?

Yes

Time-out error [C]

No

Normal completion?

Yes

Normal completion [A]

No

Abnormal termination [B]
7.11.5 Sample program
The following shows a sample program for Block Blank Check command processing for one block.

```c
/** *************************************************************/
/* */
/* Block blank check command (CSI-HS) */
/* */
/** *************************************************************/
/* [i] u16 block  ... block number */
/* [r] u16  ... error code */
/** *************************************************************/

u16 fl_hs_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

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm))
        return rc; // send "Block Blank Check" command
                       // case [C]

    if (hs_busy_to(tWT8_MAX * block_num))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    rc = fl_hs_getstatus(); // get status frame
                       //switch(rc) {
                       //  case  FLC_NO_ERR:  return rc;  break; // case [A]
                       //  case  FLC_HSTO_ERR:  return rc;  break; // case [C]
                       //  default:  return rc;  break; // case [B]
                       //}
    return rc;
}
```
7.12 Silicon Signature Command

7.12.1 Processing sequence chart

Silicon Signature command processing sequence

- `<1>` BUSY time-out check using HS pin
- `<2>` Silicon Signature command frame transmission
- `<3>` BUSY time-out check using HS pin
- `<4>` Status check processing
- `<5>` Result of status check processing
- `<6>` BUSY time-out check using HS pin
- `<7>` Data frame (silicon signature) reception processing

Time-out error [C]

Abnormal termination [B]

Normal completion [A]

Data frame error [D]
7.12.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{COM}}(\text{MAX.}) \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT11}}(\text{MAX.}) \)).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD2}}(\text{MAX.}) \)).

<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]

7.12.3 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>Time-out error [C]</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>
7.12.4 Flowchart

Silicon Signature command processing

---

 HS pin = BUSY?  

Yes

---

 Command frame transmission processing (Silicon Signature)

---

 HS pin = BUSY?  

Yes

---

 Status check processing

---

 Time-out error?  

Yes

---

 Data frame reception processing

---

 Normal completion?  

Yes

---

 Abnormal termination [B]

---

 HS pin = BUSY?  

Yes

---

 Data frame error [D]

---

 Normal data frame?  

Yes

---

 Normal completion [A]

---

 No

---

 Data frame error [D]

---

 Normal completion [A]

---

 Timed out?  

Yes

---

 tcom (MAX.)

---

 Timed out?  

Yes

---

 tWT11 (MAX.)

---

 Timed out?  

Yes

---

 Normal completion [A]

---

 No

---

 Time-out error [C]

---

 No

---

 Time-out error [C]

---

 No

---

 Time-out error [C]
7.12.5 Sample program

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

```c
/******************************************************************
/* Get silicon signature command (CSI-HS) */
/******************************************************************
/* [i] u8 *sig... pointer to signature save area */
/* [r] u16 ... error code */
/******************************************************************
u16  fl_hs_getsig(u8 *sig)
{
    u16  rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm))
        return rc;   // error detected :case [C]

    if (hs_busy_to(tWT11_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

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

    if (hs_busy_to(tFD2_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxdata_frm); // get signature data
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        case FLC_HSTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [D]
    }

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

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

7.13.1 Processing sequence chart

Version Get command processing sequence

- **Programmer V850ES/Kx2**
  - <1> BUSY time-out check using HS pin
  - <2> Version Get command frame transmission
  - <3> BUSY time-out check using HS pin
  - <4> Status check processing
  - <5> Result of status check processing
  - <6> BUSY time-out check using HS pin
  - <7> Data frame (version data) reception processing

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

- **Time-out error [C]**
  - BUSY release
  - BUSY time-out check using HS pin

- **Normal data frame? [Yes/No]**
  - Yes
  - Normal completion [A]
  - No
  - Data frame error [D]

- **Application Note U17885EJ2V0AN**
7.13.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{COM}}(\text{MAX}.) \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT12}}(\text{MAX}.) \)).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin. If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD2}}(\text{MAX}.) \)).

<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]

7.13.3 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>Processing timed out due to the busy status at the HS pin.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data does not match.</td>
</tr>
</tbody>
</table>
7.13.4 Flowchart

Version Get command processing

 HS pin = BUSY?  
 Yes  
 No  
 Command frame transmission processing (Version Get)

 HS pin = BUSY?  
 Yes  
 No  
 Status check processing

 Time-out error?  
 Yes  
 No  
 Normal completion?  
 Yes  
 Abnormal termination [B]

 HS pin = BUSY?  
 Yes  
 No  
 Data frame reception processing

 Normal data frame?  
 Yes  
 Normal completion [A]  
 No  
 Data frame error [D]

 Time-out error [C]

 tcom (MAX.)

 Timed out?  
 Yes  
 No  
 tWT12 (MAX.)

 Timed out?  
 Yes  
 No  
 tFD2 (MAX.)

 Timed out?  
 Yes  
 No  
 Time-out error [C]
### 7.13.5 Sample program

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

```c
u16  fl_hs_getver(u8 *buf)
{
    u16  rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_VERSION, 1, fl_cmd_prm))
        return rc;   // error detected :case [C]

    if (hs_busy_to(tWT12_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

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

    if (hs_busy_to(tFD2_MAX))
        return FLC_HSTO_ERR;  // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxdata_frm);  // get signature data
    switch(rc) {
        case FLC_NO_ERR:   break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:  return rc; break; // case [D]
    }

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

7.14.1 Processing sequence chart

Checksum command processing sequence

Programmer

<1> BUSY time-out check using HS pin

<2> Checksum command frame transmission

<3> BUSY time-out check using HS pin

<4> Status check processing

<5> Result of status check processing

V850ES/Kx2

<6> BUSY time-out check using HS pin

<7> Data frame (checksum data) reception processing

Result

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

Normal completion

<7> Data frame (checksum data) reception processing

Result of status check processing

Normal data frame? 
[Yes/No]

Normal completion [A]

No

Data frame error [D]

Yes

Normal completion [A]

Abnormal termination [B]

Time-out error [C]

Time-out occurs

Time-out error [C]

Time-out occurs

Time-out error [C]

Time-out occurs

<1> BUSY time-out check using HS pin

<2> Checksum command frame transmission

<3> BUSY time-out check using HS pin

<4> Status check processing

<5> Result of status check processing

<6> BUSY time-out check using HS pin

<7> Data frame (checksum data) reception processing
7.14.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{com}(\text{MAX.})$).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT16}(\text{MAX.})$).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD1}\times\text{number of blocks}$).

<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]

7.14.3 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>06H</td>
<td>The command was executed normally and checksum data was acquired normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>05H</td>
<td>The specified start/end address is not a fixed address in block units (2 KB) starting from the top of the flash memory.</td>
</tr>
<tr>
<td>Checksum error</td>
<td>07H</td>
<td>The checksum of the transmitted command frame does not match.</td>
</tr>
<tr>
<td>Time-out error [C]</td>
<td>15H</td>
<td>Command frame data is abnormal (such as invalid data length (LEN) or no ETX).</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>Processing timed out due to the busy status at the HS pin.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</td>
<td>–</td>
<td>The checksum of the data frame received as version data does not match.</td>
</tr>
</tbody>
</table>
7.14.4 Flowchart

[Diagram of flowchart showing steps for normal completion, checksum command processing, command frame transmission processing, status check processing, data frame reception processing, etc.]
7.14.5 Sample program

The following shows a sample program for Checksum command processing.

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

t16 fl_hs_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16 rc;
    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

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm)) //send "Checksum" command
        return rc; // error detected :case [C]

    if (hs_busy_to(tWT16_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

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

    if (hs_busy_to(tFD1_MAX * block_num))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxdata_frm); // get signature data
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [D]
    }

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

7.15.1 Processing sequence chart

Security Set command processing sequence
7.15.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{COM}}(\text{MAX.}) \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT13}}(\text{MAX.}) \)).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{FD3}}(\text{MAX.}) \)).

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

<8> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT14}}(\text{MAX.}) \)).

<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 [D]
   When a time-out error occurs: A time-out error [C] is returned.

<11> A V850ES/Kx2 BUSY status is checked using the HS pin.
   If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT15}}(\text{MAX.}) \)).

<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 [E]
   When a time-out error occurs: A time-out error [C] is returned.
### 7.15.3 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>06H</td>
<td>Normal acknowledgment (ACK)</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>07H</td>
<td>Checksum error</td>
</tr>
</tbody>
</table>
| Protect error                   | 10H         | • An attempt was made to enable a flag that was already prohibited in the setting.  
                           |             | • Boot block rewrite is prohibited in the security setting. |
| Negative acknowledgment (NACK)  | 15H         | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error [C]              | –           | Processing timed out due to the busy status at the HS pin. |
| Abnormal termination [D]        | 18H         | FLMD error | A write error has occurred. |
|                                | 1AH         | MGR10 error |
|                                | 1CH         | Write error |
| Abnormal termination [E]        | 1BH         | MRG11 error | An internal verify error has occurred. |
7.15.4 Flowchart

[Diagram of the flowchart is shown here, depicting the steps and conditions for normal completion, security set command processing, status check processing, and handling of time-out errors and abnormal terminations.]
7.15.5 Sample program
The following shows a sample program for Security Set command processing.

```c
u16 fl_hs_setscf(u8 scf, u8 bot)
{
    u16 rc;

    fl_cmd_prm[0] = 0x00;   // 1st byte must be 0x00
    fl_cmd_prm[1] = 0x00;   // 2nd byte must be 0x00

    fl_txdata_frm[0] = (scf|= 0b11100000);  // "FLG" (upper 3bits must be '1' (to make sure))
    fl_txdata_frm[1] = bot;   // "BOT"

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;   // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_SET_SECURITY, 3, fl_cmd_prm))  // send "Security Set" command
        return rc;    // error detected :case [C]

    if (hs_busy_to(tWT13_MAX))
        return FLC_HSTO_ERR;   // t.o. detected :case [C]

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

    if (hs_busy_to(tFD3_MAX))
        return FLC_HSTO_ERR;   // t.o. detected :case [C]

    if (rc = put_dfrm_hs(2, fl_txdata_frm, true)) // send security setting data
        return rc;    // error detected :case [C]
}
```
if (hs_busy_to(tWT14_MAX))
    return FLC_HSTO_ERR; // t.o. detected :case [C]

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

/******************************************************************************
/* Check internally verify                                              */
/******************************************************************************
if (hs_busy_to(tWT15_MAX))
    return FLC_HSTO_ERR; // t.o. detected

rc = fl_hs_getstatus(); // get status frame again
//switch(rc) {
    // case FLC_NO_ERR: return rc; break; // case [A]
    // case FLC_HSTO_ERR: return rc; break; // case [C]
    // default:        return rc; break; // case [B]
//}
    return rc;
}
7.16 Read Command

7.16.1 Processing sequence chart
7.16.2 Description of processing sequence

<1> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{COM}}(\text{MAX.}) \)).

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

<3> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT1}}(\text{MAX.}) \)).

<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> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT1}}(\text{MAX.}) \)).

<7> The data frame (user data) in the flash memory is received by data frame reception processing.

    When the processing ends normally: Proceeds to <10>.
    When an error such as checksum error occurs: Proceeds to <8>.
    When a time-out error occurs: A time-out error [C] is returned.

<8> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT1}}(\text{MAX.}) \)).

<9> The NACK frame is transmitted by data frame transmission processing.
    A data frame error [D] is returned.

<10> A V850ES/Kx2 BUSY status is checked using the HS pin.
    If a BUSY time-out occurs, a time-out error [C] is returned (time-out time \( t_{\text{WT1}}(\text{MAX.}) \)).

<11> The ACK frame is transmitted by data frame transmission processing.

    When reception of all data frames is completed, the normal completion status [A] is returned.
    If there still remain data frames to be received, the sequence is re-executed from <6>.
7.16.3 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 read data was set normally.</td>
</tr>
<tr>
<td>Abnormal termination [B]</td>
<td>05H</td>
<td>Parameter error The specified start/end address is not the start/end address of the block.</td>
</tr>
<tr>
<td>Checksum error</td>
<td>07H</td>
<td>Checksum error The checksum of the transmitted command frame or data frame does not match.</td>
</tr>
<tr>
<td>Protect error</td>
<td>10H</td>
<td>Protect error Read is prohibited in the security setting.</td>
</tr>
<tr>
<td>Negative acknowledgment (NACK)</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>Time-out error Processing timed out due to the busy status at the HS pin.</td>
</tr>
<tr>
<td>Data frame error [D]</td>
<td>–</td>
<td>Data frame error The checksum of the data frame received as read data does not match.</td>
</tr>
</tbody>
</table>
7.16.4 Flowchart

Read command processing

- HS pin = BUSY?
  - Yes
  - Time-out?
    - Yes
      - Normal completion?
        - Yes
          - Abnormal termination [B]
        - No
          - Time-out error [C]
    - No
      - Normal completion?
        - Yes
        - Abnormal termination [B]
      - No
        - Status check processing?
          - Yes
            - Time-out error?
              - Yes
                - Normal completion?
                  - Yes
                  - Abnormal termination [B]
                - No
                  - Time-out error [C]
              - No
                - Data frame (user program) reception processing?
                  - Yes
                    - Time-out during data frame reception?
                      - Yes
                        - Normal completion?
                          - Yes
                          - Abnormal termination [B]
                        - No
                          - Time-out error [C]
                      - No
                        - Normal completion?
                          - Yes
                          - Abnormal termination [B]
                        - No
                          - Status (ACK) frame transmission?
                            - Yes
                              - Time-out?
                                - Yes
                                  - Normal completion?
                                    - Yes
                                    - Abnormal termination [B]
                                  - No
                                    - Time-out error [C]
                                - No
                                  - Normal completion [A]
                        No
                          - Status (NACK) frame transmission?
                            - Yes
                              - Time-out?
                                - Yes
                                  - Normal completion?
                                    - Yes
                                    - Abnormal termination [B]
                                  - No
                                    - Time-out error [C]
                              - No
                                - Time-out error [C]
                            No
                              - Normal completion [A]
                        No
                          - Status (NACK) frame transmission?
                            - Yes
                              - Time-out?
                                - Yes
                                  - Normal completion?
                                    - Yes
                                    - Abnormal termination [B]
                                  - No
                                    - Time-out error [C]
                              - No
                                - Time-out error [C]
7.16.5 Sample program

The following shows a sample program for Read command processing.

```c
u16 fl_hs_read(u32 top, u32 bottom)
{
    u16 rc;
    u32 read_head;
    u16 len;
    u8 hooter;

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

    /******************************************************************************/
    /* send command & check status   */
    /******************************************************************************/
    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;   // t.o. detected :case [C]
    if (rc = put_cmd_hs(FL_COM_READ, 7, fl_cmd_prm))
        return rc;
    if (hs_busy_to(tWT17_MAX))
        return FLC_HSTO_ERR;   // t.o. detected :case [C]
    rc = fl_hs_getstatus();   // get status frame
    switch(rc) {
        case FLC_NO_ERR:    break; // continue
        // case FLC_HSTO_ERR:  return rc; break; // case [C]
        default:   return rc; break; // case [B]
    }

    /******************************************************************************/
    /* receive user data                        */
    /******************************************************************************/
    read_head = top;

    while(1){
        if (hs_busy_to(tWT18_MAX))
            return FLC_HSTO_ERR;   // t.o. detected :case [C]
        rc = get_dfrm_hs(fl_rxdata_frm);  // get ROM data from FLASH
        switch(rc) {
            case FLC_NO_ERR: break; // continue
            case FLC_HSTO_ERR: return rc; break; // case [C]
        }
    }
```

// case FLC_RX_DF SUM_ERR:
   default: // case [D]
      if (hs_busy_to(tWT19_MAX))
         return FLC_HSTO_ERR; // t.o. detected
         put_sfrm_hs(FLST_NACK); // send status(NACK) frame
         return rc;
      break;
   }
   if (hs_busy_to(tWT19_MAX))
      return FLC_HSTO_ERR; // t.o. detected
      put_sfrm_hs(FLST_ACK); // send status(ACK) frame
/************************************************/
/* save ROM data                                */
/************************************************/
if ((len = fl_rxdata_frm[OFS_LEN]) == 0) // get length
   len = 256;
memcpy(read_buf+read_head, fl_rxdata_frm+2, len); // save to external RAM
read_head += len;
/************************************************/
/* end check                                     */
/************************************************/
   hooter = fl_rxdata_frm[len + 3];
   if (hooter == FL_ETB) // end frame ?
      continue; // no
   break; // yes
}
return FLC_NO_ERR;
}
8.1 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
8.2 Data Frame Transmission Processing Flowchart

1. Data frame transmission processing
2. Data frame header (STX = 02H) transmission
3. Wait between data transmissions
4. Data length (LEN) transmission
5. LEN bytes transmitted?
   - Yes: Transmission of footer other than those of last data frame (ETB = 17H)
   - No: Transmits 1-byte data
7. Wait between data transmissions
8. Checksum data (SUM) transmission
9. Wait between data transmissions
10. Last data frame footer (ETX = 03H) transmission
11. Last data frame? (Yes)
12. End of data frame transmission
13. Last data frame? (No)
14. Transmission of footer other than those of last data frame (ETB = 17H)
8.3 Data Frame Reception Processing Flowchart

1. 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?
               - No
                 - Wait between data receptions
               - Yes
                 - 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
                           - Checksum error
                         - No
                           - End of data frame reception
8.4 Status Command

8.4.1 Processing sequence chart

Status command processing sequence

Note  Applied specifications differ depending on the command executed.
8.4.2 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 \((t_{WT} \times \text{MAX.})^\text{Note}\).
  
  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** Applied specifications differ depending on the command executed.

8.4.3 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>FLMD error</td>
<td>18H</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>
8.4.4 Flowchart

Note  Applied specifications differ depending on the command executed.
8.4.5 Sample program

The following shows a sample program for Status command processing.

```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 frame
        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
        }

        if (fl_rxdata_frm[OFS_LEN] == 2 && fl_st1 == FLST_ACK && fl_st2 == FLST_BUSY){
            if (check_flto()) // time out ?
                return FLC_DFTO_ERR; // Yes, time-out // case [C]
            continue;
        }

        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;
}
```
8.5 Reset Command

8.5.1 Processing sequence chart

Reset command processing sequence

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

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

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

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

<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.

8.5.3 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>
</tbody>
</table>
|                                | Negative acknowledgment (NACK) | 15H | • A command other than the Status command was received during processing.  
• Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error [C]             | –            | Status check processing timed out. |
8.5.4 Flowchart

Reset command processing

Wait from previous frame reception until next command transmission (Reset)

Command frame transmission processing

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

No (normal completion)

Result of status check processing = Time-out error?

Yes

Normal completion [A]

Time-out error [C]
8.5.5 Sample program

The following shows a sample program for Reset command processing.

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

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_MAX); // 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;
}
```
8.6 Oscillating Frequency Set Command

8.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence

---

1. Wait from previous frame reception until next command transmission
2. Oscillating Frequency Set 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
  - [A]
- Abnormal termination
  - [B]
- Time-out error
  - [C]

---

Application Note U17885EJ2V1AN 191
8.6.2 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> Waits from command transmission until status check processing (wait time \( t_{\text{WT9}}(\text{MAX.}) \)).

<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.

8.6.3 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>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>
8.6.4 Flowchart

- Oscillating Frequency Set command processing
  - Wait from previous frame reception until next command transmission \( t_{\text{com}} \)
  - Command frame transmission processing (Oscillating Frequency Set)
  - Wait from command frame transmission until status check \( t_{\text{WTS}} \)
  - Status check processing
  - Time-out error? Yes
    - Normal completion? No
  - Time-out error [C]
  - Normal completion? Yes
    - Abnormal termination [B]
8.6.5 Sample program

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

```c
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_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;
}
```
8.7 Chip Erase Command

8.7.1 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 [C] → Normal completion [A]
- Abnormal termination [B] → Abnormal termination [B]
- Time-out error [C] → Time-out error [C]
8.7.2 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 $t_{\text{WT1(MAX.)}}$).
<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.

8.7.3 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>
</tbody>
</table>
|                                | Protect error       | 10H | • Chip erase is prohibited in the security setting.  
|                                |                        |       | • Boot block rewrite is prohibited in the security setting.  |
|                                | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
|                                | Erase error       | 1AH | An erase error has occurred. |
|                                | Write error       | 1CH | |
|                                | MRG10 error       | 1AH | |
|                                | MRG11 error       | 1BH | |
| Time-out error [C]             | –                | The status frame was not received within the specified time. |
8.7.4 Flowchart

Chip Erase command processing

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

Command frame transmission processing (Chip Erase)

Wait from command frame transmission until status check \( t_{\text{WT1}} \)

Status check processing

Time-out error? Yes

No

Normal completion? Yes

No

Normal completion [A] Abnormal termination [B]

Time-out error [C]
8.7.5 Sample program

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

```c
/******************************************************************
/*                                                            */
/* Erase all(chip) command (CSI)                            */
/*                                                            */
/**************************************************************************/
/* [r] ul6    ... error code                                 */
/**************************************************************************/
ul6    fl_csi_erase_all(void)
{
    ul6    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;
}
```
8.8 Block Erase Command

8.8.1 Processing sequence chart

Block Erase command processing sequence

Programmer

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

Abnormal termination

Time-out error

Normal completion

V850ES/Kx2

Abnormal termination [B]

Time-out error [C]

Normal completion [A]
8.8.2 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 $t_{WT2 (MAX.)} \times \text{number of blocks}$).

<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 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.

8.8.3 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>Protect error</td>
<td>10H</td>
<td>\begin{itemize} \item Block erase is prohibited in the security setting. \item The boot area is included in the specified range while boot block rewrite is prohibited in the security setting. \item Chip erase is prohibited in the security setting. \item Programming is prohibited in the security setting. \end{itemize}</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>MRG10 error</td>
<td>1AH</td>
<td>An erase error has occurred.</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>
8.8.4 Flowchart

Block Erase command processing

Wait from previous frame reception until next command transmission $t_{COM}$

Command frame transmission processing (Block Erase)

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

Status check processing

Time-out error?

Yes

Time-out error [C]

No

Normal completion?

Yes

Normal completion [A]

No

Abnormal termination [B]
8.8.5 Sample program

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

```c
u16 fl_csi_erase_blk(u16 sblk, u16 eblk)
{
    u16 rc;
    u32 top, bottom;
    u16 block_num;

    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

    block_num = eblk - sblk + 1;

    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(tWT2 * block_num);

    rc = fl_csi_getstatus(tWT2_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;
}
```
8.9 Programming Command

8.9.1 Processing sequence chart

Programming command processing sequence

Programmer

V850ES/Kx2

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.

6. Wait from previous frame reception until next data frame transmission.

7. Data frame (user data) transmission.

8. Wait during status check.


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

11. Result of status check processing.

12. Status check processing.

13. Result of status check processing.

Result

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

Abnormal termination

Normal completion

Time-out error

Abnormal termination

Other than ACK

ACK

All data frames transmitted? [Yes/No]

Time-out error

Abnormal termination

Normal completion

Time-out error
8.9.2 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. Waits from command transmission until status check processing (wait time $t_{\text{WT3 (MAX.)}}$).
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 $t_{\text{FD3 (MAX.)}}$).
7. User data to be written to the V850ES/Kx2 flash memory is transmitted by data frame transmission processing.
8. Waits from data frame (user data) transmission until status check processing (wait time $t_{\text{WT4 (MAX.)}}$).
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 $t_{\text{WT5 (MAX.)}} \times \text{number of blocks}$).
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.
### 8.9.3 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>
</tbody>
</table>
| Abnormal termination [B] | Parameter error | 05H | - The specified start/end address is not the start/end address of the block.  
- The data length is not 64-word units. |
| | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
| | Protect error | 10H | - Write is prohibited in the security setting.  
- Boot block rewrite is prohibited in the security setting. |
| | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error [C] | | - | The status frame was not received within the specified time. |
| Abnormal termination [D] | FLMD error | 18H (ST2) | A write error has occurred. |
| | Write error | 1CH (ST2) | A write error has occurred. |
| Abnormal termination [E] | MRG11 error | 1BH | An internal verify error has occurred. |
8.9.4 Flowchart

[Diagram of flowchart showing the process of 3-Wire Serial I/O Communication Mode (CSI), including programming command processing, command frame transmission processing, status check processing, data frame transmission processing, and status check processing with decision points for time-out error, normal completion, and abnormal termination conditions.]
8.9.5 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

    /*******************************************************************************/
    /* send command & check status */
    /*******************************************************************************/
    fl_wait(tCOM);
    put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command
    fl_wait(tWT3);
    rc = fl_csi_getstatus(tWT3_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]
    }

    /*******************************************************************************/
    /* 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
   // (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]
}
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;
}/* 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;
8.10 Verify Command

8.10.1 Processing sequence chart

Verify command processing sequence

- **Programmer**
  - Wait from previous frame reception until next command transmission
  - Verify command frame transmission
  - Wait from command frame transmission until status check
  - Status check processing
  - Result of status check processing
  - Wait from previous frame reception until next command transmission
  - Data frame (user data for verify) transmission
  - Wait during status check (internal verify)
  - Status check processing
  - Result of status check processing (ST1/ST2)
  - Reception status (ST1) (Normal completion/Abnormal termination/Time-out error)
  - Time-out error
  - Abnormal termination (B)
  - Reception status (ST2) (ACK/other than ACK)
  - Normal completion
  - Time-out error
  - Normal completion (A)
  - Other than ACK
  - Abnormal termination (D)
  - All data frames transmitted? (Yes/No)
  - Go to <6>

- **V850ES/Kx2**
  - Normal completion (A)
  - ACK
  - Abnormal termination (D)
  - No
  - Yes
  - Go to <6>
  - Reception status (ST1) (Normal completion/Abnormal termination/Time-out error)
  - Time-out error
  - Normal completion (A)
8.10.2 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. Waits from command transmission until status check processing (wait time $t_{WT6 \text{ (MAX.)}}$).
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 $t_{FD3}$).
7. User data for verifying is transmitted by data frame transmission processing.
8. Waits from data frame transmission until status check processing (wait time $t_{WT7 \text{ (MAX.)}}$).
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.

8.10.3 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>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>
8.10.4 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: Time-out error [C]
  - No: Normal completion?
    - Yes: Abnormal termination [B]
    - No: Status check processing
- 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: Time-out error [C]
  - No: Normal completion?
    - Yes: ST2 = ACK?
      - Yes: Abnormal termination [D]
      - No: All data frames transmitted?
        - Yes: Normal completion [A]
        - No: Abnormal termination [B]
8.10.5 Sample program

The following shows a sample program for Verify command processing.

```c
u16 fl_csi_verify(u32 top, u32 bottom, u8 *buf)
{
    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(tWT_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:        break; // continue
        case FLC_DPTO_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 =
        }
    }
```

memcpy(fl_txdata_frm, 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
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    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]
8.11 Block Blank Check Command

8.11.1 Processing sequence chart

Block Blank Check command processing sequence
8.11.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time $t_{c0m}$).
<2> The Block Blank Check command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time $t_{wT8}(M A X.) \times$ 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: If the blank check for all of the specified blocks is not yet completed,
processing changes the number of blocks and re-executes the sequence from <1>.
If the blank check for all of the specified blocks is completed, the processing
ends normally [A].

8.11.3 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>
</tbody>
</table>
| Abnormal termination [B]       | Parameter error | 05H | The specified start/end address is out of the flash memory range.  
|                                |              |            | The specified start/end address is not the start/end address of the block. |
|                                | Checksum error | 07H | The checksum of the transmitted command frame does not match. |
|                                | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
|                                | MRG11 error | 1BH | The specified block in the flash memory is not blank. |
| Time-out error [C]             | –           | The status frame was not received within the specified time. |
8.11.4 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

Status check processing

Time-out error?

Yes

Time-out error [C]

No

Normal completion?

Yes

Normal completion [A]

No

Blank check for all of specified blocks completed?

Yes

No

Abnormal termination [B]

Yes

Normal completion [A]
8.11.5 Sample program

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

```c
/**
 * fl_csi_blk_blank_chk(u32 top, u32 bottom)
 */
static 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;
}
```
8.12 Silicon Signature Command

8.12.1 Processing sequence chart

Silicon Signature command processing sequence

Programmer

V850ES/Kx2

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

<2> Silicon Signature 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]

Abnormal termination [B]

Normal completion

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

<7> Data frame (silicon signature) reception processing

Abnormal termination [B]

Normal data frame? [Yes/No]

No

Data frame error [D]

Yes

Normal completion [A]
8.12.2 Description of processing sequence

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

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

<3> Waits from command transmission until status check processing (wait time \( t_{\text{WT11(MAX.)}} \)).

<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 \( t_{\text{FD2}} \)).

<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]

8.12.3 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>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 silicon signature data does not match.</td>
</tr>
</tbody>
</table>
8.12.4 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?
            - 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?
                    - No: Data frame error [D]
                    - Yes: Normal completion [A]
8.12.5 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_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_wait(tFD2_SIG);  // wait before getting data frame

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

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

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

8.13.1 Processing sequence chart

Version Get command processing sequence

[Diagram showing the processing sequence of Version Get Command]
8.13.2 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 $t_{WT1\text{2}(\text{MAX.})}$).
<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]

8.13.3 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>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 does not match.</td>
</tr>
<tr>
<td>Abnormal termination [B]</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>Data frame error [D]</td>
<td>–</td>
<td>The checksum of the data frame received as version data does not match.</td>
</tr>
</tbody>
</table>
8.13.4 Flowchart

- Version Get command processing
- Wait from previous frame reception until next command transmission
- Command frame transmission processing (Version Get)
- 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]
    - Time-out error [C]
- 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]
8.13.5 Sample program
The following shows a sample program for Version Get command processing.

```c
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_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_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]
}
```
8.14 Checksum Command

8.14.1 Processing sequence chart

Checksum command processing sequence
8.14.2 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 $t_{WT16} (MAX.)$).
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]

8.14.3 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] Normal acknowledgment (ACK)</td>
<td>06H</td>
<td>The command was executed normally and checksum data was acquired normally.</td>
</tr>
<tr>
<td>Abnormal termination [B] Parameter error</td>
<td>05H</td>
<td>The specified start/end address is not a fixed address in block units (2 KB) starting from the top of the flash memory.</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 does not match.</td>
</tr>
</tbody>
</table>
8.14.4 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?
  - No: Data frame error [D]
  - Yes: Normal completion [A]
8.14.5 Sample program

The following shows a sample program for Checksum command processing.

```c
/* Get checksum command (CSI) */

u16 fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16 rc;
    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
    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_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]
    }

    // 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]
}
```
8.15 Security Set Command

8.15.1 Processing sequence chart

Security Set command processing sequence
8.15.2 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> Waits from command transmission until status check processing (wait time \( t_{\text{WT13}}(\text{MAX.}) \)).

<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 \( t_{\text{FD3}} \)).

<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 \( t_{\text{WT14}}(\text{MAX.}) \)).

<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 [D]
When a time-out error occurs: A time-out error [C] is returned.

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

<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 [E]
When a time-out error occurs: A time-out error [C] is returned.

8.15.3 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>Checksum error</td>
<td>07H</td>
</tr>
</tbody>
</table>
|                                 | Protect error              | 10H | • An attempt was made to enable a flag that was already prohibited in the setting.  
|                                 |                           |    | • Boot block rewrite is prohibited in the security setting. |
|                                 | Negative acknowledgment (NACK) | 15H | Command frame data is abnormal (such as invalid data length (LEN) or no ETX). |
| Time-out error [C]              | –                        | –  | The status frame was not received within the specified time. |
8.15.4 Flowchart

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

Abnormal termination [B]

No

Normal completion?

Yes

Abnormal termination [B]

No

Time-out error [C]

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

Abnormal termination [B]

No

Normal completion?

Yes

Abnormal termination [B]

No

Time-out error [C]

Wait during status check (internal verify)

Status check processing

Time-out error?

Yes

Abnormal termination [B]

No

Normal completion?

Yes

Normal completion [A]

No

Abnormal termination [B]
8.15.5 Sample program

The following shows a sample program for Security Set command processing.

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

  /* set params */
  fl_cmd_prm[0] = 0x00; // 1st byte must be 0x00
  fl_cmd_prm[1] = 0x00; // 2nd byte must be 0x00
  fl_txdata_frm[0] = (scf | 0b11100000); // "FLG" (upper 3 bits must be '1'
                                       // (to make sure))
  fl_txdata_frm[1] = bot;   // "BOT"

  /* send command */
  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_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]
  }

  /* send data frame (security setting data)*/
  fl_wait(tFD3_CSI);   // wait before getting data frame

  put_dfrm_csi(2, fl_txdata_frm, true); // send data frame(Security data)

  fl_wait(tWT14);
```

```c
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;
}
8.16 Read Command

8.16.1 Processing sequence chart

Read command processing sequence
8.16.2 Description of processing sequence

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

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

<3> Waits from command transmission until status check processing (wait time \( t_{WT17} \)).

<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 reception (wait time \( t_{WT18} \)).

<7> The data frame (user data) is received by data frame reception processing.

The following processing is performed according to the result of reception processing.

When the processing ends normally:  Proceeds to <10>.
When the processing ends abnormally: Proceeds to <8>.

<8> Waits from the previous frame reception until the next status (NACK) frame transmission (wait time \( t_{WT19} \)).

<9> The NACK frame is transmitted by data frame transmission processing.

A data frame error [D] is returned.

<10> Waits from the previous frame reception until the next status (ACK) frame transmission (wait time \( t_{WT19} \)).

<11> The ACK frame is transmitted by data frame transmission processing.

When reception of all data frames is completed, the normal completion status [A] is returned.
If there still remain data frames to be received, the sequence is re-executed from <5>.

8.16.3 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>Protect error</td>
<td>10H</td>
<td>Read is prohibited in 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 or data 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 read data does not match.</td>
</tr>
</tbody>
</table>
8.16.4 Flowchart

- **Read command processing**
- **Wait from previous frame reception until next command transmission** (Time-out error [C])
- **Command frame transmission processing** (Read)
- **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
  - **Normal completion?**
    - Yes
    - **Normal completion [A]**
    - No

- **Wait from previous frame reception until next status frame transmission**
- **Status (NACK) frame transmission**
  - **Data frame error [D]**
- **Wait from previous frame reception until next status frame transmission**
  - **Status (ACK) frame transmission**
- **All data frames received?**
  - No
  - **Data frame error [D]**
  - Yes
    - **Normal completion [A]**
8.16.5 Sample program
The following shows a sample program for Read command processing.

```c
/*******************************/
/* */
/* Read command (CSI) */
/* */
/*******************************/
/* [i] u32 top ... start address */
/* [i] u32 bottom ... end address */
/* [r] u16 ... error code */
/*******************************/
u16 fl_csi_read(u32 top, u32 bottom)
{
    u16 rc;
    u32 read_head;
    u16 len;
    u8 hooter;

    /*************************************************************************/
    /* 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_csi(FL_COM_READ, 7, fl_cmd_prm); // send "Read" command
    fl_wait(tWT17); // wait
    rc = fl_csi_getstatus(tWT17_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]
    }

    /*************************************************************************/
    /* receive user data */
    /*************************************************************************/
    read_head = top;
    while(1){
        fl_wait(tWT18);
        rc = get_dfrm_csi(fl_rxdata_frm); // get ROM data from FLASH
        switch(rc) {
            case FLC_NO_ERR:    break; // continue
            // case FLC_RX_DFSUM_ERR:
            default: continue; // case [D]
        }
        fl_wait(tWT19);
        put_sfrm_csi(FLST_NACK); // send status(NACK) frame
        return rc;
        break;
    }

    fl_wait(tWT19);
    put_sfrm_csi(FLST_ACK); // send status(ACK) frame
```
/************************************************/ /* save ROM data */ ************************************************/ if ((len = fl_rxdata_frm[OFS_LEN]) == 0) // get length len = 256; memcpy(read_buf+read_head, fl_rxdata_frm+2, len); // save to external RAM read_head += len; /* ************************************************/ /* end check */ ************************************************/ hooter = fl_rxdata_frm[len + 3]; if (hooter == FL_ETB) // end frame ? continue; // no break; // yes } return FLC_NO_ERR; }
CHAPTER 9 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS

This chapter describes the parameter characteristics between the programmer and the V850ES/Kx2 in the flash memory programming mode.

Be sure to refer to the user’s manual of the V850ES/Kx2 for electrical specifications when designing a programmer.

<Operating clock (fx)>

The internal clock of the V850ES/Kx2 is changed according to the value of the oscillation frequency (fx) specified with the Oscillation Frequency Set command by the programmer.

When \(2.0 \text{ MHz} \leq f_x \leq 5.0 \text{ MHz}\): \(f_{XX} = f_x \times 4 \) (PLL mode)
When \(5.0 \text{ MHz} < f_x \leq 10.0 \text{ MHz}\): \(f_{XX} = f_x \times 1 \) (Clock through mode)

Therefore, it is obtained by assigning \(f_x\) until \(t_{WT9}\) of the Oscillation Frequency Set command and assigning \(f_{XX}\) after \(t_{WT9}\).

9.1 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>VDD↑ to FLMD0↑</td>
<td>(t_{DP})</td>
<td>1 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLMD0↑ to RESET↑</td>
<td>(t_{PR})</td>
<td>2 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count start time from RESET↑ to FLMD0↑</td>
<td>(t_{RP})</td>
<td>(33,740/f_x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count finish time from RESET↑ to FLMD0↑</td>
<td>(t_{RPE})</td>
<td>(132,276/f_x)</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/CSI + HS)</td>
<td>(t_{RC})</td>
<td>(181,787/f_x)</td>
<td>3 s</td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 1 (UART)</td>
<td>(t_{R1})</td>
<td></td>
<td>3 s</td>
<td></td>
</tr>
<tr>
<td>Wait for low-level data 2 (UART)</td>
<td>(t_{R2})</td>
<td>30,000/f_x</td>
<td>3 s</td>
<td></td>
</tr>
<tr>
<td>Wait for Read command (UART)</td>
<td>(t_{R2})</td>
<td>30,000/f_x</td>
<td>3 s</td>
<td></td>
</tr>
<tr>
<td>Width of low-level data 1/2 (UART)</td>
<td>(t_{L1, L2})</td>
<td></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>
</tbody>
</table>

Notes 1. \((66,612/f_x + 549,591/f_x)/2\) 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.
### 9.2 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></td>
<td>Between data frame transmission/reception</td>
<td>Data frame reception</td>
<td>tDR</td>
<td>CSI</td>
<td>440/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UART</td>
<td>140/fX</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data frame transmission</td>
<td>tDT</td>
<td>CSI</td>
<td>560/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UART</td>
<td>0&lt;sup&gt;Max&lt;/sup&gt;</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td>From Status command frame reception until status frame transmission</td>
<td>Chip Erase</td>
<td>tSR</td>
<td>CSI</td>
<td>2,160/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block Erase</td>
<td></td>
<td></td>
<td>1,800/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Programming</td>
<td></td>
<td></td>
<td>1,320/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Verify</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block Blank Check</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security Set</td>
<td></td>
<td></td>
<td>1,920/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other command</td>
<td></td>
<td></td>
<td>1,320/fX</td>
</tr>
<tr>
<td></td>
<td>From status frame transmission until data frame transmission (1)</td>
<td>–</td>
<td>tD0&lt;sub&gt;1&lt;/sub&gt;&lt;sup&gt;Max&lt;/sup&gt;</td>
<td>CSI</td>
<td>95,520/fX + 24 μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UART</td>
<td>0&lt;sup&gt;Max&lt;/sup&gt;</td>
<td>114,624/fX + 29 μs</td>
</tr>
<tr>
<td></td>
<td>From status frame transmission until data frame transmission (2)</td>
<td>Silicon signature data</td>
<td>tD0&lt;sub&gt;2&lt;/sub&gt;</td>
<td>CSI</td>
<td>2,900/fX + 48 μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Version data</td>
<td></td>
<td></td>
<td>280/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>–</td>
<td></td>
<td></td>
<td>UART</td>
</tr>
<tr>
<td></td>
<td>From status frame transmission until data frame reception</td>
<td>–</td>
<td>tD0&lt;sub&gt;3&lt;/sub&gt;</td>
<td>CSI</td>
<td>1,160/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UART</td>
<td>6,720/fX</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td>From status frame transmission until command frame reception</td>
<td>–</td>
<td>tCOM</td>
<td>CSI</td>
<td>158/fX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UART</td>
<td>154/fX</td>
<td>3 s</td>
</tr>
</tbody>
</table>

**Notes**

1. When successive reception is enabled for the programmer
2. Time for one block transmission

**Remark**

The waits are defined as follows.

- tDR, tD0<sub>3</sub>, tCOM

  The V850ES/Kx2 is ready for the next communication after the MIN. time has elapsed after completion of the previous communication.

  The programmer must transmit the next data between the MIN. and MAX. times after completion of the previous communication.

- tDT, tSR, tD0<sub>1</sub>, tD0<sub>2</sub>

  The V850ES/Kx2 is ready for the next communication after the MIN. time has elapsed after completion of the previous communication.

  The programmer must receive the next data between the MIN. and MAX. times after completion of the previous communication.
<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>840/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Chip Erase</td>
<td>twT1</td>
<td>−</td>
<td>[V850ES/KE2]</td>
<td>[V850ES/KE2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,718,706/fx + 135.4 ms</td>
<td>36,374,804/fx + 43,258.0 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,718,706/fx + 237.8 ms</td>
<td>36,374,804/fx + 76,787.6 ms</td>
</tr>
<tr>
<td>Block Erase</td>
<td>twT2&lt;sup&gt;Note 2&lt;/sup&gt;</td>
<td>−</td>
<td>51,601/fx + 13.7 ms</td>
<td>13,176,706/fx + 3,497.6 ms</td>
</tr>
<tr>
<td>Programming</td>
<td>twT3</td>
<td>CSI</td>
<td>1,500/fx + 24 μs</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>twT4&lt;sup&gt;Note 2&lt;/sup&gt;</td>
<td>−</td>
<td>26,980/fx + 22.7 ms</td>
<td>623,736/fx + 285.9 ms</td>
</tr>
<tr>
<td></td>
<td>twT5&lt;sup&gt;Note 2&lt;/sup&gt;</td>
<td>CSI</td>
<td>129,207/fx + 4.2 ms</td>
<td>176,809/fx + 7.2 ms</td>
</tr>
<tr>
<td>Verify</td>
<td>twT6</td>
<td>CSI</td>
<td>440/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>twT7&lt;sup&gt;Note 3&lt;/sup&gt;</td>
<td>CSI</td>
<td>4,240/fx + 404 μs</td>
<td>10,350/fx + 423 μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Block Blank Check</td>
<td>twT8&lt;sup&gt;Note 2&lt;/sup&gt;</td>
<td>CSI</td>
<td>54,778/fx + 2.0 ms</td>
<td>75,899/fx + 3.5 ms</td>
</tr>
<tr>
<td>Oscillating Frequency</td>
<td>twT9</td>
<td>CSI</td>
<td>154,000/fx</td>
<td>3 s</td>
</tr>
<tr>
<td>Set</td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Baud Rate Set</td>
<td>twT10</td>
<td>UART</td>
<td>1,680/fx</td>
<td>3 s</td>
</tr>
<tr>
<td>Silicon Signature</td>
<td>twT11</td>
<td>CSI</td>
<td>520/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Version Get</td>
<td>twT12</td>
<td>CSI</td>
<td>520/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Security Set</td>
<td>twT13</td>
<td>CSI</td>
<td>460/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>twT14</td>
<td>−</td>
<td>94,000/fx + 4.8 ms</td>
<td>13,219,105/fx + 3488.7 ms</td>
</tr>
<tr>
<td></td>
<td>twT15</td>
<td>CSI</td>
<td>482,000/fx + 16.9 ms</td>
<td>13,607,105/fx + 3500.8 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>twT16</td>
<td>CSI</td>
<td>640/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>twT17</td>
<td>CSI</td>
<td>1,520/fx + 24 μs</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>twT18&lt;sup&gt;Note 3&lt;/sup&gt;</td>
<td>CSI</td>
<td>13,920/fx</td>
<td>3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART</td>
<td>Note 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>twT19</td>
<td>−</td>
<td>116/fx</td>
<td>Note 4</td>
</tr>
</tbody>
</table>
Notes 1. Reception must be enabled for the programmer before command transmission.
2. Time for one block transmission
3. Time for 256-byte data transmission
4. Time until the programmer transmits “ACK”

Remark The waits are defined as follows.
<\text{tWT0 to tWT19}>
The V850ES/Kx2 completes command processing between the MIN. and MAX. times.
The programmer must repeat the status check until the MAX. time is elapsed.

9.3 UART Communication Mode

- Data frame

- Programming mode setting/Reset command

Note FLMD0 counter rise/fall time
• Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command

![Diagram of Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command]

- Baud Rate Set command

![Diagram of Baud Rate Set command]

- Silicon Signature command/Version Get command

![Diagram of Silicon Signature command/Version Get command]

- Checksum command

![Diagram of Checksum command]
• Programming command

• Verify command
• Security Set command

• Read command

• Wait before command frame transmission
9.4 3-Wire Serial I/O Communication Mode

- Data frame

- Programming mode setting/Reset command

Note  FLMD0 counter rise/fall time

- Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command
• Silicon Signature command/Version Get command

• Checksum command

• Programming command
• Verify command

• Security Set command
• Read command

• Wait before command frame transmission
[MEMO]
APPENDIX A CIRCUIT DIAGRAM (REFERENCE)

Figure A-1 shows a circuit diagram of the programmer and the V850ES/Kx2, for reference.
Figure A-1. Reference Circuit Diagram of Programmer and V850ES/Kx2
# APPENDIX B  REVISION HISTORY

## B.1 Major Revisions in This Edition

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U17885EJ2V0AN00 → U17885EJ2V1AN00</td>
<td></td>
</tr>
<tr>
<td>p.240</td>
<td>Modification of MIN. value and MAX. value of ( t_{RP} ), ( t_{PE} ), ( t_{IC} ), and ( t_{RE} ) in 9.1 Flash Memory Programming Mode Setting Time</td>
</tr>
<tr>
<td>p.241</td>
<td>Addition of &quot;Other command&quot; to ( t_{SF} ) in 9.2 Programming Characteristics</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: 01-3067-5800

Succursal en España
Juan Esplanadiu, 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
Steigergweg 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/

NEC Electronics Shanghai Ltd.
Room 2511-2512, Bank of China Tower,
Pudong New Area, Shanghai P.R. China P.C:200120
Tel: 021-5888-5400
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
12/F., Cityplaza 4,
12 Taikoo Wan Road, 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/