To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

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

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-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.

RENESAS

Application Note

78K0/Lx2

8-Bit Single-Chip Microcontrollers

Flash Memory Programming (Programmer)

78K0/LE2:	μPD78F0361, 78F0362, 78F0363, 78F0363D
78K0/LF2:	μPD78F0372, 78F0373, 78F0374, 78F0375, 78F0376, 78F0376D, 78F0382, 78F0383, 78F0384, 78F0385, 78F0386, 78F0386D
78K0/LG2:	μPD78F0393, 78F0394, 78F0395, 78F0396, 78F0397, 78F0397D

Document No. U18204EJ2V0AN00 (2nd edition)
Date Published March 2009 NS

[MEMO]

NOTES FOR CMOS DEVICES

① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (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 July, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

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

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

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

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

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

(Note)

(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02.11-1

INTRODUCTION

Target Readers

This application note is intended for users who understand the functions of the 78K0/Lx2 and who will use this product to design application systems.

Purpose

The purpose of this application note is to help users understand how to develop dedicated flash memory programmers for rewriting the internal flash memory of the 78K0/Lx2.

The sample programs and circuit diagrams shown in this document are for reference only and are not intended for use in actual design-ins.

Therefore, these sample programs must be used at the user's own risk. Correct operation is not guaranteed if these sample programs are used.

Organization

This manual consists of the following main sections.

- Flash memory programming
- Command/data frame format
- Description of command processing
- UART communication mode
- 3-wire serial I/O communication mode (CSI)
- Flash memory programming parameter characteristics

How to Read This Manual

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

- To gain a general understanding of functions:
  → Read this manual in the order of the **CONTENTS**. The mark "<R>" shows major revised points. The revised points can be easily searched by copying an "<R>" in the PDF file and specifying it in the "Find what:" field.
- To learn more about the 78K0/Lx2's hardware functions:
  → See the user's manual of each 78K0/Lx2 product.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{xxx}$ (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 xxxx
	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

Document Name	Document Number
78K0/LE2 User's Manual	U17734E
78K0/LF2 User's Manual	U17504E
78K0/LG2 User's Manual	U17473E
78K/0 Series Instructions User's Manual	U12326E

Caution The related documents listed above are subject to change without notice. Be sure to use the latest version of each document when designing.

CONTENTS

CHAPTER 1 FLASH MEMORY PROGRAMMING

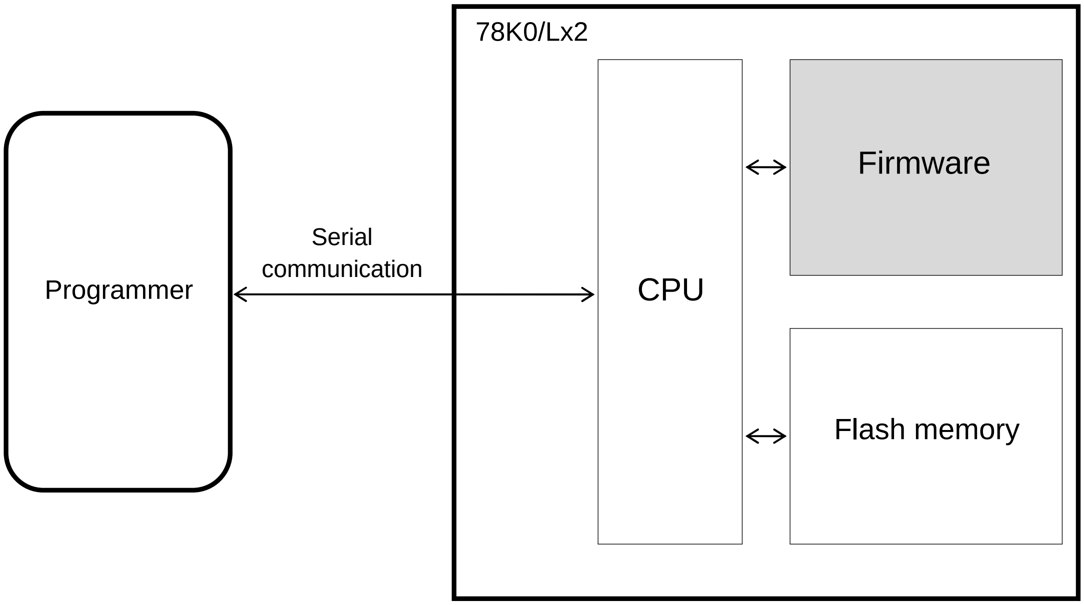
To rewrite the contents of the internal flash memory of the 78K0/Lx2, a dedicated flash memory programmer (hereafter referred to as the “programmer”) is usually used.

This Application Note explains how to develop a dedicated programmer.

1.1 Overview

The 78K0/Lx2 incorporates firmware that controls flash memory programming. The programming to the internal flash memory is performed by transmitting/receiving commands between the programmer and the 78K0/Lx2 via serial communication.

Figure 1-1. System Outline of Flash Memory Programming in 78K0/Lx2



1.2 System Configuration

Examples of the system configuration for programming the flash memory are illustrated in Figure 1-2.

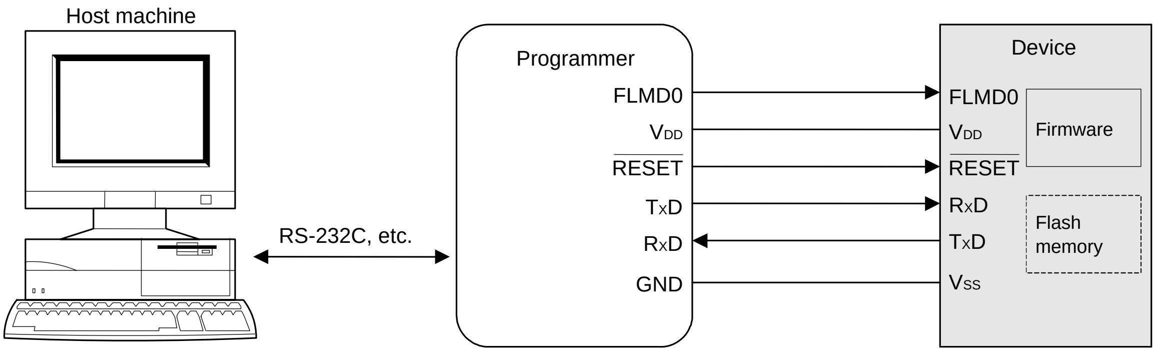
These figures illustrate how to program the flash memory with the programmer, under control of a host machine.

Depending on how the programmer is connected, the programmer can be used in a standalone mode without using the host machine, if a user program has been downloaded to the programmer in advance.

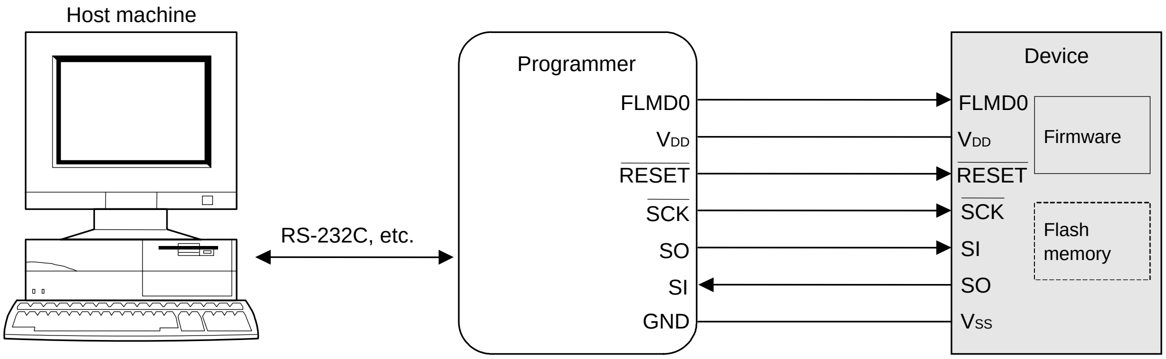
For example, NEC Electronics' flash memory programmer PG-FP5 can execute programming either by using the GUI software with a host machine connected or by itself (standalone).

Figure 1-2. System Configuration Examples

UART communication mode (LSB-first transfer)



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



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

1.3 Flash Memory Configuration

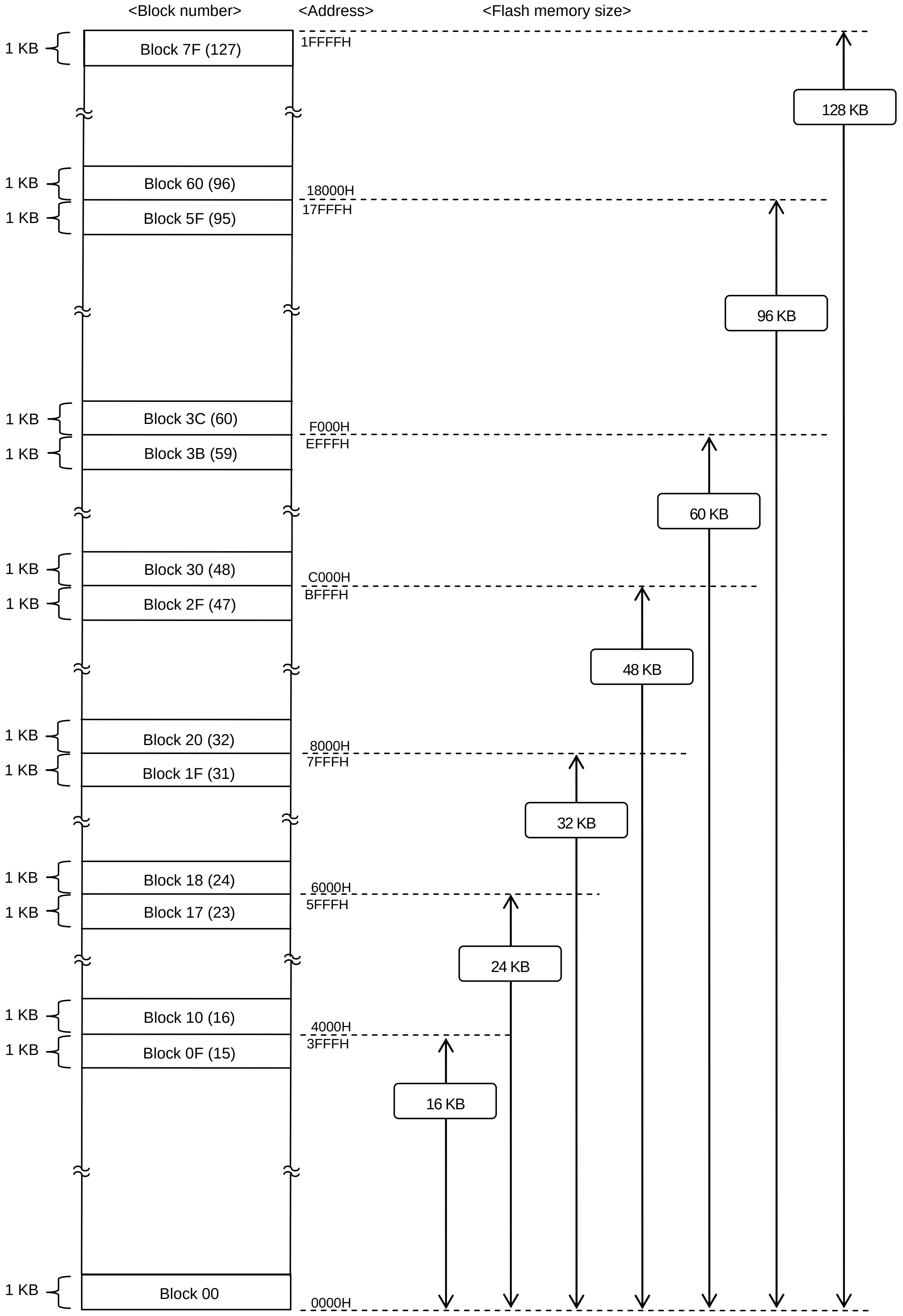
The 78K0/Lx2 must manage product-specific information (such as a device name and memory information).

Table 1-1 shows the flash memory size of the 78K0/Lx2 and Figure 1-3 shows the configuration of the flash memory.

Table 1-1. Flash Memory Size of 78K0/Lx2

Device Name		Flash Memory Size
78K0/LE2	μPD78F0361	16 KB
	μPD78F0362	24 KB
	μPD78F0363, 78F0363D	32 KB
78K0/LF2	μPD78F0372	24 KB
	μPD78F0373	32 KB
	μPD78F0374	48 KB
	μPD78F0375	60 KB
	μPD78F0376, 78F0376D	96 KB
	μPD78F0382	24 KB
	μPD78F0383	32 KB
	μPD78F0384	48 KB
	μPD78F0385	60 KB
	μPD78F0386, 78F0386D	96 KB
78K0/LG2	μPD78F0393	32 KB
	μPD78F0394	48 KB
	μPD78F0395	60 KB
	μPD78F0396	96 KB
	μPD78F0397, 78F0397D	128 KB

Figure 1-3. Flash Memory Configuration



Remark Each block consists of 1 KB (this figure only illustrates some parts of entire blocks in the flash memory).

1.4 Command List and Status List

The flash memory incorporated in the 78K0/Lx2 has functions to manipulate the flash memory, as listed in Table 1-2. The programmer transmits commands to control these functions to the 78K0/Lx2, and manipulates the flash memory with checking the response status from the 78K0/Lx2.

1.4.1 Command List

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

Table 1-2. List of Commands Transmitted from Programmer to 78K0/Lx2

Command Number	Command Name	Function Name	Function
20H	Chip Erase	Erase	Erases the entire flash memory area.
22H	Block Erase		Erases a specified area in the flash memory.
40H	Programming	Write	Writes data to a specified area in the flash memory.
13H	Verify	Verify	Compares the contents in a specified area in the flash memory with data transmitted from the programmer.
32H	Block Blank Check	Blank check	Checks the erase status of a specified block in the flash memory.
70H	Status	Information acquisition	Acquires the current operating status (status data).
C0H	Silicon Signature		Acquires 78K0/Lx2 information (write protocol information).
C5H	Version Get		Acquires version information of the 78K0/Lx2 and firmware.
B0H	Checksum		Acquires checksum data of a specified area.
A0H	Security Set	Security	Sets security information.
00H	Reset	Others	Detects synchronization in communication.
90H	Oscillating Frequency Set		Specifies the oscillation frequency of the 78K0/Lx2.

1.4.2 Status List

The following table lists the status codes the programmer receives from the 78K0/Lx2.

Table 1-3. Status Code List

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

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

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

Although not listed in the above table, if a time-out error (BUSY time-out or time-out in data frame reception during UART communication) occurs, it is recommended to shutdown the power supply to the 78K0/Lx2 (refer to **1.6 Shutting Down Target Power Supply**) and then connect the power supply again.

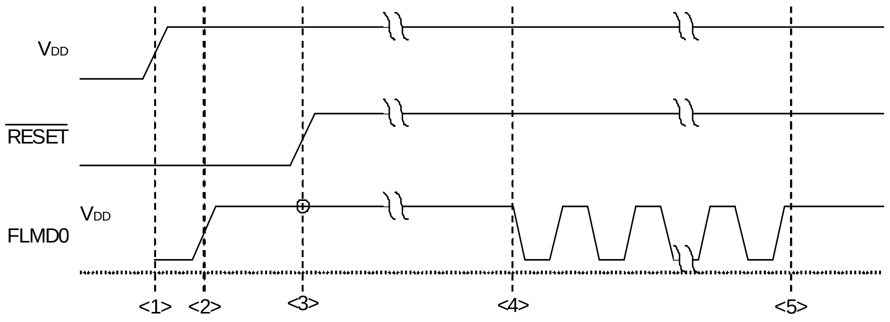
1.5 Power Activation and Setting Flash Memory Programming Mode

To rewrite the contents of the flash memory with the programmer, the 78K0/Lx2 must first be set to the flash memory programming mode by supplying a specific voltage to the flash memory programming mode setting pin (FLMD0) in the 78K0/Lx2, then releasing a reset.

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

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

Figure 1-4. Setting Flash Memory Programming Mode and Selecting Communication Mode



<1>: Power activation (V_{DD})
<2>: FLMD0 = high level
<3>: Reset release (mode setting)
<4>: Pulse output starts
<5>: Pulse output ends

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

Table 1-4. Relationship Between FLMD0 Pin Setting After Reset Release and Operating Mode

FLMD0	Operating Mode
Low (GND)	Normal operating mode
High (V_{DD})	Flash memory programming mode

The following table shows the relationship between the number of FLMD0 pulses (pulse counts) and communication modes that can be selected with the 78K0/Lx2.

Table 1-5. Relationship Between FLMD0 Pluse Counts and Communication Modes

Communication Mode	FLMD0 Pulse Counts	Port Used for Communication
UART (UART6)	0 (when X1 clock (f_X) is used)	TxD6 (P13), RxD6 (P14)
	3 (when external main system clock (f_{EXCLK}) is used)	
3-wire serial I/O (CSI10)	8	SO10 (P12), SI10 (P11), $\overline{\text{SCK10}}$ (P10)

UART Communication Mode

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

Table 1-6. UART Communication Conditions

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

The programmer always operates as the master device during CSI communication, so the programmer must check whether the processing by the 78K0/Lx2, such as writing or erasing, is normally completed. On the other hand, the status of the master and slave is occasionally exchanged during UART communication, so communication at the optimum timing is possible.

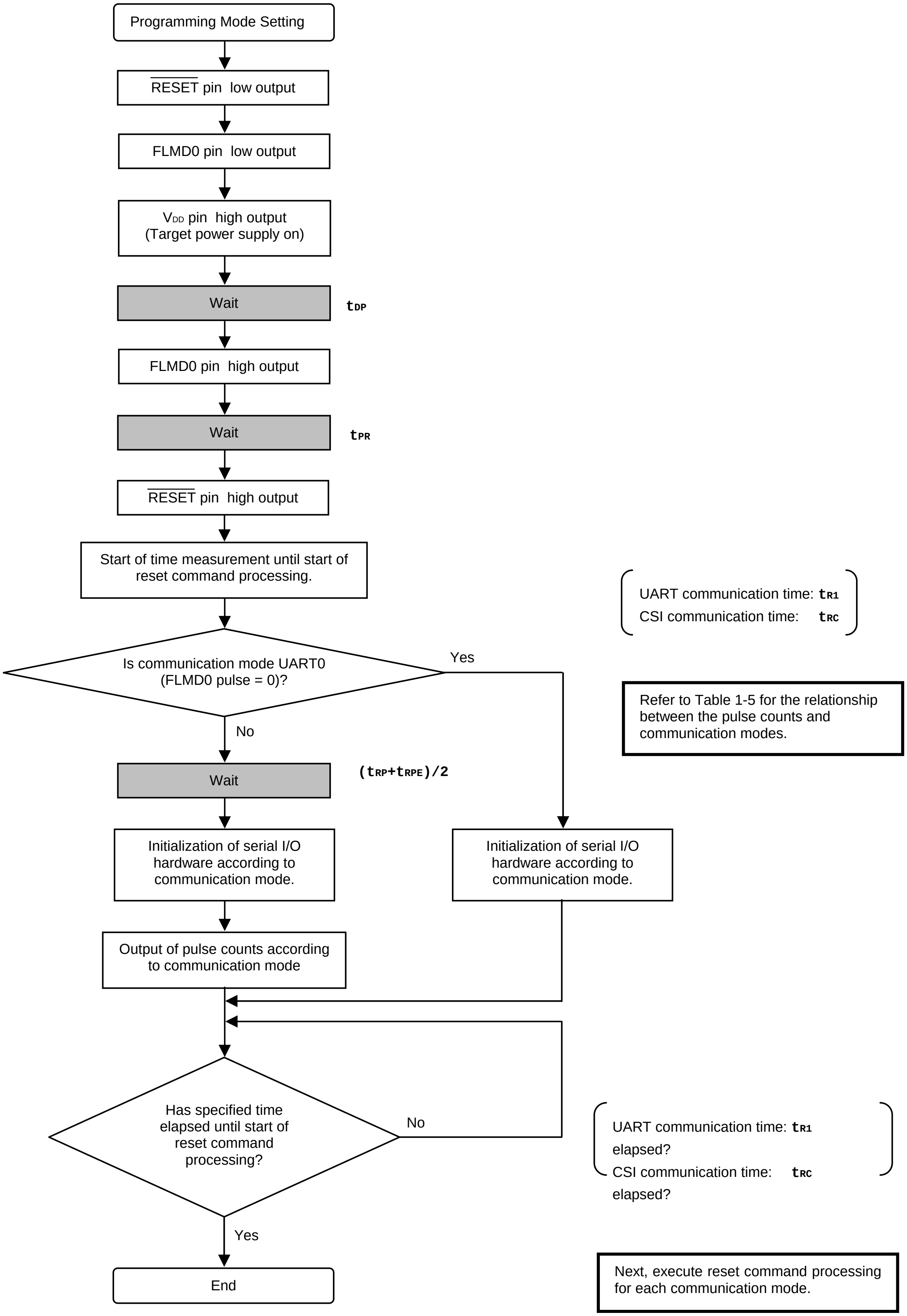
Caution Set the same baud rate to the master and slave devices when performing UART communication.

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

The $\overline{\text{SCK}}$, 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 $\overline{\text{SCK}}$ pin while the 78K0/Lx2 is not ready for transmission/reception.

The communication data format is MSB-first, in 8-bit units.

1.5.1 Mode Setting Flowchart


Programming Mode Setting
RESET pin low output
FLMD0 pin low output
V_{DD} pin high output
(Target power supply on)
Wait
t_{DP}
FLMD0 pin high output
Wait
t_{PR}
RESET pin high output
Start of time measurement until start of reset command processing.
UART communication time: t_{R1}
CSI communication time: t_{RC}
Is communication mode UART0 (FLMD0 pulse = 0)?
Yes
No
Refer to Table 1-5 for the relationship between the pulse counts and communication modes.
Wait
$(t_{RP}+t_{RPE})/2$
Initialization of serial I/O hardware according to communication mode.
Initialization of serial I/O hardware according to communication mode.
Output of pulse counts according to communication mode
Has specified time elapsed until start of reset command processing?
No
UART communication time: t_{R1} elapsed?
CSI communication time: t_{RC} elapsed?
Yes
End
Next, execute reset command processing for each communication mode.

1.5.2 Sample program

The following shows a sample program for mode setting processing.

```
/*****************************************************************/
/*                                                               */
/* connect to Flash device                                       */
/*                                                               */
/*****************************************************************/
void
fl_con_dev(void)
{
extern  void  init_fl_uart(void);
extern  void  init_fl_csi(void);

        int   n;
        int   pulse;

        SRMK0 = true;
        UARTE0 = false;

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

        }

        pFL_RES    = low;           // RESET = low
        pmFL_FLMD0 = PM_OUT;        // FLMD0 = output mode
        pFL_FLMD0  = low;
        FL_VDD_HI();                // VDD = high

        fl_wait(tDP);               // wait

        pFL_FLMD0  = hi;            // FLMD0 = high
        fl_wait(tPR);               // wait

        pFL_RES    = hi;            // RESET = high
        start_flto(fl_if == FLIF_CSI ? tRC : tR1);    // start "tRC" wait timer
        fl_wait((tRP+tRPE)/2);

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

        for (n = 0; n < pulse; n++){// pulse output

          pFL_FLMD0 = low;
          fl_wait(tPW);
```

```
              pFL_FLMD0 = hi;
              fl_wait(tPW);
            }
            while(!check_flto())        // timeout tRC ?
                   ;                    // no

            // start RESET command proc.

    }
```

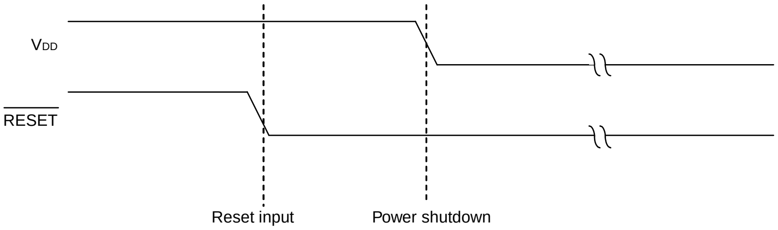
1.6 Shutting Down Target Power Supply

After each command execution is completed, shut down the power supply to the target after setting the $\overline{\text{RESET}}$ pin to low level, as shown below.

Set other pins to Hi-Z when shutting down the power supply to the target.

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

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

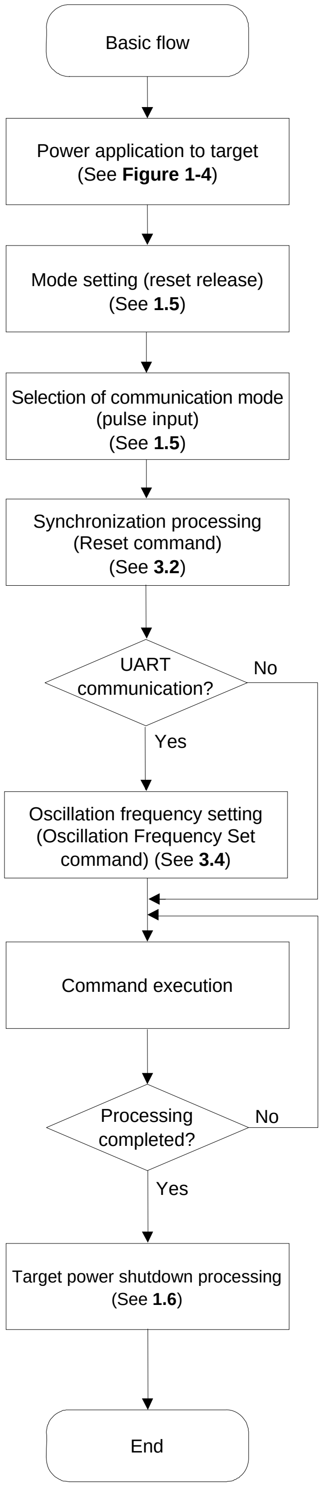


1.7 Command Execution Flow at Flash Memory Rewriting

Figure 1-6 illustrates the basic flowchart when flash memory rewriting is performed with the programmer.

Other than commands shown in the Figure 1-6, the Verify command and Checksum command are also be supported.

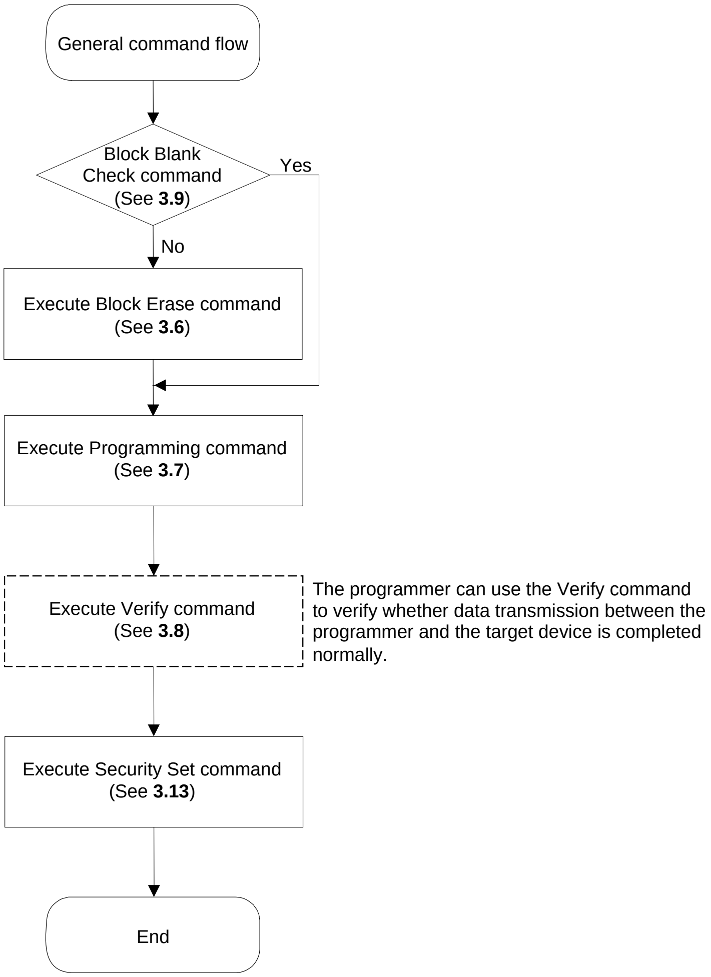
Figure 1-6. Basic Flowchart for Flash Memory Rewrite Processing


Basic flow
Power application to target (See Figure 1-4)
Mode setting (reset release) (See 1.5)
Selection of communication mode (pulse input) (See 1.5)
Synchronization processing (Reset command) (See 3.2)
UART communication?
No
Yes
Oscillation frequency setting (Oscillation Frequency Set command) (See 3.4)
Command execution
Processing completed?
No
Yes
Target power shutdown processing (See 1.6)
End

Reset input and power shutdown during rewriting is prohibited because security information may be lost.

Remark Figure 1-7 shows execution example of each command.

Figure 1-7. General Command Execution Flow at Flash Memory Rewriting


General command flow
Block Blank Check command (See 3.9)
Yes
No
Execute Block Erase command (See 3.6)
Execute Programming command (See 3.7)
Execute Verify command (See 3.8)
Execute Security Set command (See 3.13)
End

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

CHAPTER 2 COMMAND/DATA FRAME FORMAT

The programmer uses the command frame to transmit commands to the 78K0/Lx2. The 78K0/Lx2 uses the data frame to transmit write data or verify data to the programmer. A header, footer, data length information, and checksum are appended to each frame to enhance the reliability of the transferred data.

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

Figure 2-1. Command Frame Format

SOH (1 byte)	LEN (1 byte)	COM (1 byte)	Command information (variable length) (Max. 255 bytes)	SUM (1 byte)	ETX (1 byte)

Figure 2-2. Data Frame Format

STX (1 byte)	LEN (1 byte)	Data (variable length) (Max. 256 bytes)	SUM (1 byte)	ETX or ETB (1 byte)

Table 2-1. Description of Symbols in Each Frame

Symbol	Value	Description
SOH	01H	Command frame header
STX	02H	Data frame header
LEN	–	Data length information (00H indicates 256). Command frame: COM + command information length Data frame: Data field length
COM	–	Command number
SUM	–	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
ETB	17H	Footer of data frame other than the last frame
ETX	03H	Command frame footer, or footer of last data frame

The following shows examples of calculating the checksum (SUM) for a frame.

[Command frame]

No command information is included in the following example of a Status command frame, so LEN and COM are targets of checksum calculation.

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

2.1 Command Frame Transmission Processing

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

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

2.2 Data Frame Transmission Processing

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

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

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

2.3 Data Frame Reception Processing

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

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

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

CHAPTER 3 DESCRIPTION OF COMMAND PROCESSING

3.1 Status Command

3.1.1 Description

This command is used to check the operation status of the 78K0/Lx2 after issuance of each command such as write or erase.

After the Status command is issued, if the Status command frame cannot be received normally in the 78K0/Lx2 due to problems based on communication or the like, the status setting will not performed in the 78K0/Lx2. As a result, a busy response (FFH), not the status frame, may be received. In such a case, retry the Status command.

3.1.2 Command frame and status frame

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

Figure 3-1. Status Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	SUM	ETX
01H	01H	70H (Status)	Checksum	03H

Figure 3-2. Status Frame for Status Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data			SUM	ETX
02H	n	ST1	…	STn	Checksum	03H

Remarks 1. ST1 to STn: Status #1 to Status #n

2. The length of a status frame varies according to each command (such as write or erase) to be transmitted to the 78K0/Lx2.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

Caution After each command such as write or erase is transmitted in UART communication, the 78K0/Lx2 automatically returns the status frame within a specified time. The Status command is therefore not used.

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

3.2 Reset Command

3.2.1 Description

This command is used to check the establishment of communication between the programmer and the 78K0/Lx2 after the communication mode is set.

When UART is selected as the mode for communication with the 78K0/Lx2, the same baud rate must be set in the programmer and 78K0/Lx2. However, the 78K0/Lx2 cannot detect its own baud rate generation clock (f_X or f_{EXCLK}) frequency so the baud rate cannot be set. It makes detection of the baud rate generation clock frequency in the 78K0/Lx2 possible by sending “00H” twice at 9,600 bps from the programmer, measuring the low-level width of “00H”, and then calculating the average of two sent signals. The baud rate can consequently be set, which enables synchronous detection in communication.

3.2.2 Command frame and status frame

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

Figure 3-3. Reset Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	SUM	ETX
01H	01H	00H (Reset)	Checksum	03H

Figure 3-4. Status Frame for Reset Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	1	ST1	Checksum	03H

Remark ST1: Synchronization detection result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **4.4 Reset Command**.
- For the 3-wire serial I/O communication mode (CSI), read **5.5 Reset Command**.

3.3 Baud Rate Set Command

The 78K0/Lx2 does not support the Baud Rate Set command.

With the 78K0/Lx2, UART communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted. After the status frame is received, the communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.

3.4 Oscillating Frequency Set Command

3.4.1 Description

This command is used to specify the frequency of f_X or f_{EXCLK} during UART communication.

The 78K0/Lx2 uses the frequency data in the received packet to realize the baud rate of 115,200 bps.

Caution With the 78K0/Lx2, UART communication is performed at 9,600 bps until the Oscillating Frequency Set command is transmitted.
After the status frame is received, the communication rate is switched to 115,200 bps. After that, the communication rate is fixed to 115,200 bps.

3.4.2 Command frame and status frame

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

Figure 3-5. Oscillating Frequency Set Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information				SUM	ETX
01H	05H	90H (Oscillating Frequency Set)	D01	D02	D03	D04	Checksum	03H

Remark D01 to D04: Oscillation frequency = (D01 × 0.1 + D02 × 0.01 + D03 × 0.001) × 10^{D04} (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 × 10^4 = 6,000 kHz = 6 MHz

Setting example: To set 10 MHz
D01 = 01H
D02 = 00H
D03 = 00H
D04 = 05H
Oscillation frequency = 1 × 0.1 × 10^5 = 10,000 kHz = 10 MHz

Figure 3-6. Status Frame for Oscillating Frequency Set Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Oscillation frequency setting result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.5 Chip Erase Command

3.5.1 Description

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

3.5.2 Command frame and status frame

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

Figure 3-7. Chip Erase Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	SUM	ETX
01H	01H	20H (Chip Erase)	Checksum	03H

Figure 3-8. Status Frame for Chip Erase Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Chip erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.6 Block Erase Command

3.6.1 Description

This command is used to erase the contents of blocks with the specified number in the flash memory.

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

However, if Block Erase command is not impossible by the security setting, the contents is not erased (see **3.13 Security Set Command**).

3.6.2 Command frame and status frame

Figure 3-9 shows the format of a command frame for the Block Erase command, and Figure 3-10 shows the status frame for the command.

Figure 3-9. Block Erase Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	22H (Block Erase)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

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

Figure 3-10. Status Frame for Block Erase Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Block erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.7 Programming Command

3.7.1 Description

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

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

If both of the status frames (ST1 and ST2) after the last data transmission indicate ACK, the 78K0/Lx2 firmware automatically executes internal verify. Therefore, the status code validation for this internal verification is necessary.

3.7.2 Command frame and status frame

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

Figure 3-11. Programming Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	40H (Programming)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

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

Figure 3-12. Status Frame for Programming Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

3.7.3 Data frame and status frame

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

Figure 3-13. Data Frame to Be Written (from Programmer to 78K0/Lx2)

STX	LEN	Data	SUM	ETX/ETB
02H	00H to FFH (00H = 256)	Write Data	Checksum	03H/17H

Remark Write Data: User program to be written

Figure 3-14. Status Frame for Data Frame (from 78K0/Lx2 to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	Checksum	03H

Remark ST1 (b): Data reception check result
ST2 (b): Write result

3.7.4 Completion of transferring all data and status frame

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

Figure 3-15. Status Frame After Completion of Transferring All Data (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (c)	Checksum	03H

Remark ST1 (c): Internal verify result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.8 Verify Command

3.8.1 Description

This command is used to compare the data transmitted from the programmer with the data read from the 78K0/Lx2 (read level) in the specified address range, and check whether they match.

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

3.8.2 Command frame and status frame

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

Figure 3-16. Verify Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	13H (Verify)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

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

Figure 3-17. Status Frame for Verify Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

3.8.3 Data frame and status frame

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

Figure 3-18. Data Frame of Data to Be Verified (from Programmer to 78K0/Lx2)

STX	LEN	Data	SUM	ETX/ETB
02H	00H to FFH (00H = 256)	Verify data	Checksum	03H/17H

Remark Verify Data: User program to be verified

Figure 3-19. Status Frame for Data Frame (from 78K0/Lx2 to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	Checksum	03H

Remark ST1 (b): Data reception check result
ST2 (b): Verify result[Note]

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

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.9 Block Blank Check Command

3.9.1 Description

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

A block can be specified with the start address of the blank check start block and the last address of the blank check end block. Successive multiple blocks can be specified.

3.9.2 Command frame and status frame

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

Figure 3-20. Block Blank Check Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	32H (Block Blank Check)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

Remark SAH, SAM, SAL: Block blank check start address (start address of any block)
SAH: Start address, high (bits 23 to 16)
SAM: Start address, middle (bits 15 to 8)
SAL: Start address, low (bits 7 to 0)
EAH, EAM, EAL: Block blank check end address (last address of any block)
EAH: End address, high (bits 23 to 16)
EAM: End address, middle (bits 15 to 8)
EAL: End address, low (bits 7 to 0)

Figure 3-21. Status Frame for Block Blank Check Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Block blank check result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.10 Silicon Signature Command

3.10.1 Description

This command is used to read the write protocol information (silicon signature) of the device.

If the programmer supports a programming protocol that is not supported in the 78K0/Lx2, for example, execute this command to select an appropriate protocol in accordance with the values of the second and third bytes.

3.10.2 Command frame and status frame

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

Figure 3-22. Silicon Signature Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	SUM	ETX
01H	01H	C0H (Silicon Signature)	Checksum	03H

Figure 3-23. Status Frame for Silicon Signature Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

3.10.3 Silicon signature data frame

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

Figure 3-24. Silicon Signature Data Frame (from 78K0/Lx2 to Programmer)

STX	LEN	Data								SUM	ETX
02H	n	VEN	MET	MSC	DEC	END	INVALID DATA	SCF	BOT	Checksum	03H

Remarks 1.

n (LEN):	Data length
VEN:	Vendor code (NEC: 10H)
MET:	Macro extension code
MSC:	Macro function code
DEC:	Device extension code
END:	Internal flash memory last address
INVALID DATA:	Invalid data of 10 byte length.
SCF:	Security flag information
BOT:	Boot block number (fixed to 03H)

2. For the vendor code (VEN), extension code (MET), function code (MSC), device extension code (DEC), internal flash memory last address (END) and security flag information (SCF), the lower 7 bits are used as data entity, and the highest bit is used as an odd parity. The following shows an example.

Table 3-1. Example of Silicon Signature Data

Field	Contents	Length (Byte)	Example of Silicon Signature Data[Note 1]		Actual Value	Parity
VEN	Vendor code (NEC)	1	10H	(00010000B)	10H	Added
MET	Extension code (fixed in 78K0/Lx2)	1	7FH	(01111111B)	7FH	Added
MSC	Function information (fixed in 78K0/Lx2)	1	04H	(00000100B)	04H	Added
<R> DEC	Device extension code (fixed in 78K0/Lx2)	1	7CH	(01111100B)	7CH	Added
END	Internal flash memory last address (extracted from the lower bytes)	3	7FH	(01111111B)	005FFFH	Added[Note 2]
			BFH	(11011111B)		
			01H	(00000001B)		
INVALID DATA	Invalid data	10	–		–	–
SCF	Security flag information	1	Any		Any	Added[Note 3]
<R> BOT	The last block number of the boot block cluster (fixed)	1	03H	(00000011B)	03H	Not added

Notes 1. 0 and 1 are odd parities (the values to adjust the number of “1” to be the odd number in a byte)

2. The parity calculation for the END field is performed as follows (when the last address is 005FFFH)

<1> The END field is divided in 7-bit units from the lower digit (the higher 3 bits are discarded).

0 0 5 F F F

00000000 01011111 11111111

↓

~~000~~ 0000001 0111111 1111111

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

p0000001 p01111111 p1111111 (p = odd parity bit)

= 0000001 10111111 01111111

= 01 BF 7F

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

7F BF 01

The following shows the procedure to translate the values in the END field that has been sent from the microcontroller to the actual address.

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

7F BF 01
↓
01 BF 7F

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

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

```
01 BF 7F
   ↓
00000001  10111111  01111111
   ↓
0000001   0111111  1111111
   ↓
000  0000001  0111111  1111111
```

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

```
000000001011111111111111
   ↓
00000000  01011111  11111111
   ↓
=   0   0    5   F    F   F
```

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

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

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **4.11 Silicon Signature Command**.
- For the 3-wire serial I/O communication mode (CSI), read **5.12 Silicon Signature Command**.

3.10.4 78K0/Lx2 silicon signature list

Table 3-2. 78K0/Lx2 Silicon Signature Data List

Item	Description	Length (Bytes)	Data (Hex)
Vendor code	NEC	1	10
Extension code	Extension code	1	7F
Function code	Function information	1	04
Device information	Device information	1	7C
Internal flash memory last address	(7-bit data + odd parity bit) × 3	3	**Note**
Invalid data	–	10	–
Security flag information	Security flag information	1	Any
Boot block number	The last block number of the boot cluster that is currently selected	1	03

Note List of internal flash memory last addresses

Item	Description	Length (Bytes)	Data (Hex)
Internal flash memory last address	16 KB (3FFFH)	3	7F7F80
	24 KB (5FFFH)		7FBF01
	32 KB (7FFFH)		7F7F01
	48 KB (BFFFH)		7F7F02
	60 KB (EFFFH)		7FDF83
	96 KB (17FFFH)		7F7F85
	128 KB (1FFFFH)		7F7F07

3.11 Version Get Command

3.11.1 Description

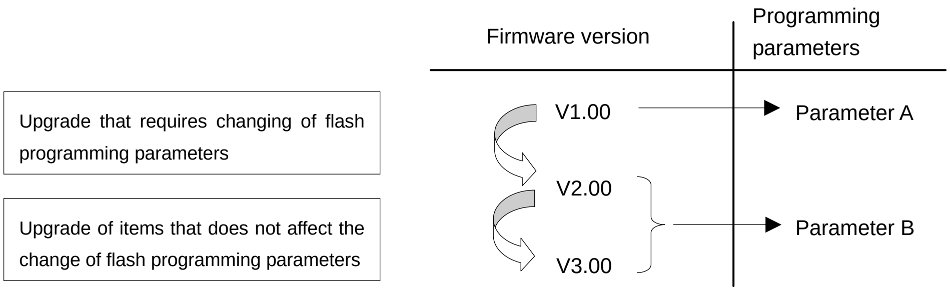
This command is used to acquire information on the 78K0/Lx2 device version and firmware version.

The device version value is fixed to 00H.

Use this command when the programming parameters must be changed in accordance with the 78K0/Lx2 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


Firmware version
Programming parameters
Upgrade that requires changing of flash programming parameters
Upgrade of items that does not affect the change of flash programming parameters
V1.00
V2.00
V3.00
Parameter A
Parameter B

3.11.2 Command frame and status frame

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

Figure 3-25. Version Get Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	SUM	ETX
01H	01H	C5H (Version Get)	Checksum	03H

Figure 3-26. Status Frame for Version Get Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

3.11.3 Version data frame

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

Figure 3-27. Version Data Frame (from 78K0/Lx2 to Programmer)

STX	LEN	Data						SUM	ETX
02H	06H	DV1	DV2	DV3	FV1	FV2	FV3	Checksum	03H

Remark DV1: Integer of device version (fixed to 00H)

DV2: First decimal place of device version (fixed to 00H)

DV3: Second decimal place of device version (fixed to 00H)

FV1: Integer of firmware version

FV2: First decimal place of firmware version

FV3: Second decimal place of firmware version

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.12 Checksum Command

3.12.1 Description

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

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

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

3.12.2 Command frame and status frame

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

Figure 3-28. Checksum Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	B0H (Checksum)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

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

Figure 3-29. Status Frame for Checksum Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

3.12.3 Checksum data frame

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

Figure 3-30. Checksum Data Frame (from 78K0/Lx2 to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	CK1	CK2	Checksum	03H

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

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

3.13 Security Set Command

3.13.1 Description

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

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

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

3.13.2 Command frame and status frame

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

Figure 3-31. Security Set Command Frame (from Programmer to 78K0/Lx2)

SOH	LEN	COM	Command Information		SUM	ETX
01H	03H	A0H (Security Set)	00H (fixed)	00H (fixed)	Checksum	03H

Figure 3-32. Status Frame for Security Set Command (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

3.13.3 Data frame and status frame

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

Figure 3-33. Security Data Frame (from Programmer to 78K0/Lx2)

STX	LEN	Data		SUM	ETX
02H	02H	FLG	BOT	Checksum	03H

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

Figure 3-34. Status Frame for Security Data Writing (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (b)	Checksum	03H

Remark ST1 (b): Security data write result

3.13.4 Internal verify check and status frame

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

Figure 3-35. Status Frame for Internal Verify Check (from 78K0/Lx2 to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (c)	Checksum	03H

Remark ST1 (c): Internal verify result

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

Table 3-3. Contents of Security Flag Field

Item	Contents
Bit 7	Fixed to “1”
Bit 6	
Bit 5	
Bit 4	Boot block cluster rewrite disable flag (1: Enables boot block rewrite, 0: Disable boot block rewrite)
Bit 3	Fixed to “1”
Bit 2	Programming disable flag (1: Enables programming, 0: Disable programming)
Bit 1	Block erase disable flag (1: Enables block erase, 0: Disable block erase)
Bit 0	Chip erase disable flag (1: Enables chip erase, 0: Disable chip erase)

The following table shows the relationship between the security flag field settings and the enable/disable status of each operation.

Table 3-4. Security Flag Field and Enable/Disable Status of Each Operation

Operating Mode	Flash Memory Programming Mode			Self-Programming Mode
Command Security Setting Item	Command Operation After Security Setting √: Execution possible, ×: Execution impossible △: Writing and block erase in boot area are impossible Writing and block erase in area other than boot area are possible			• All commands can be executed regardless of the security setting values • Only retention of security setting values is possible
	Programming	Chip Erase	Block Erase	
Disable programming	×	√	×	
Disable chip erase	√	×	×	
Disable block erase	√	√	×	
Boot block cluster rewrite disable	△	×	△	Same condition as that in flash memory programming mode (on-board/off-board programming)

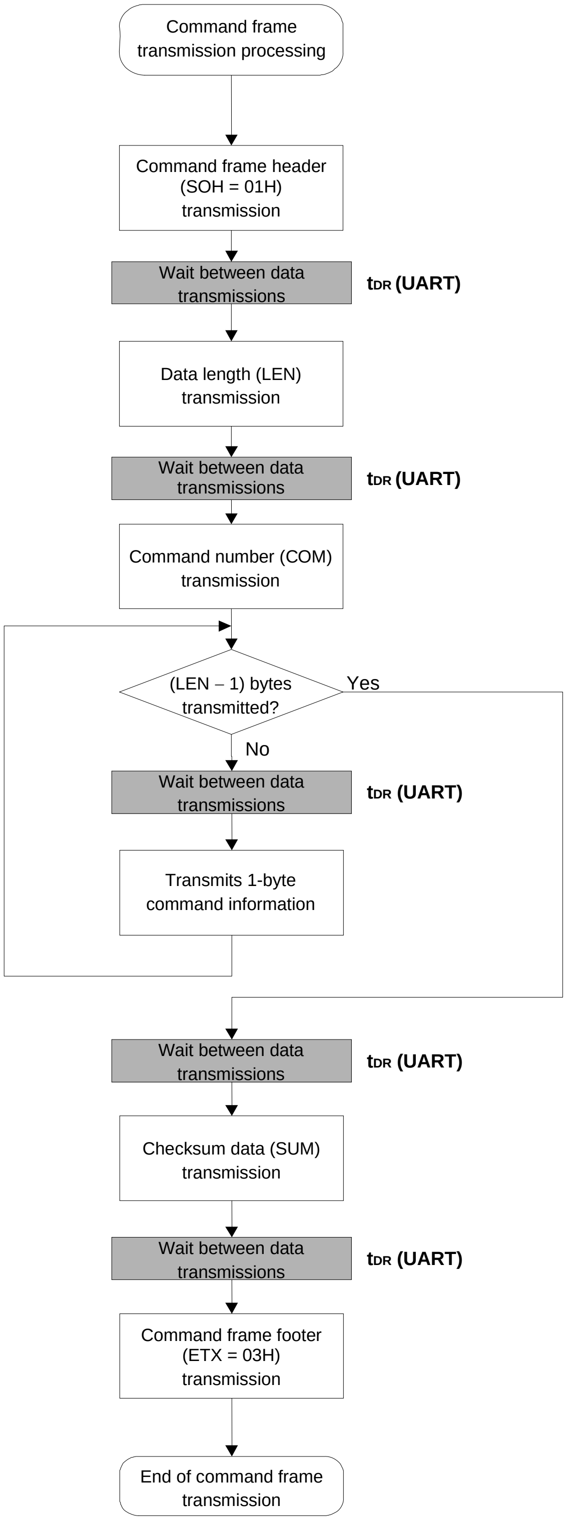
Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Lx2, flowcharts of command processing, and sample programs for each communication mode.

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

CHAPTER 4 UART COMMUNICATION MODE

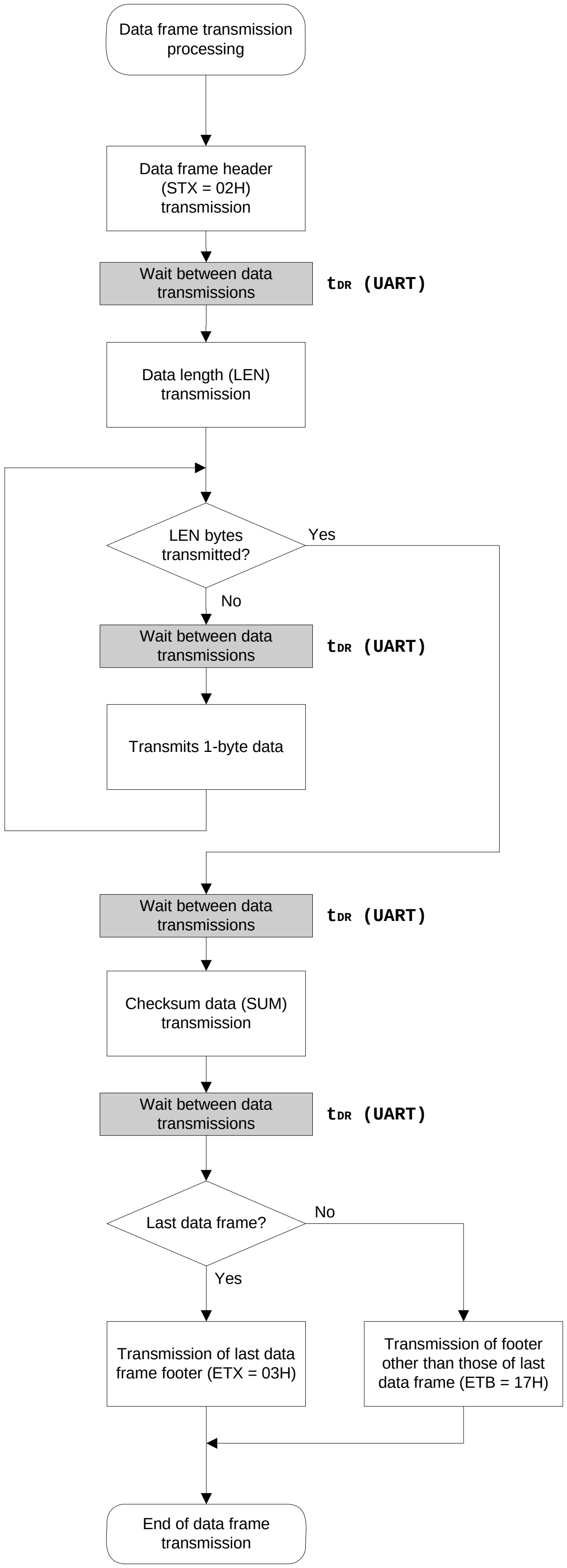
Each of the symbol (t_{XX} and t_{WTXX}) shown in the flowchart in this chapter is the symbol of characteristic item in **CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS**.
For each specified value, refer to **CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS**.

<R> 4.1 Command Frame Transmission Processing Flowchart

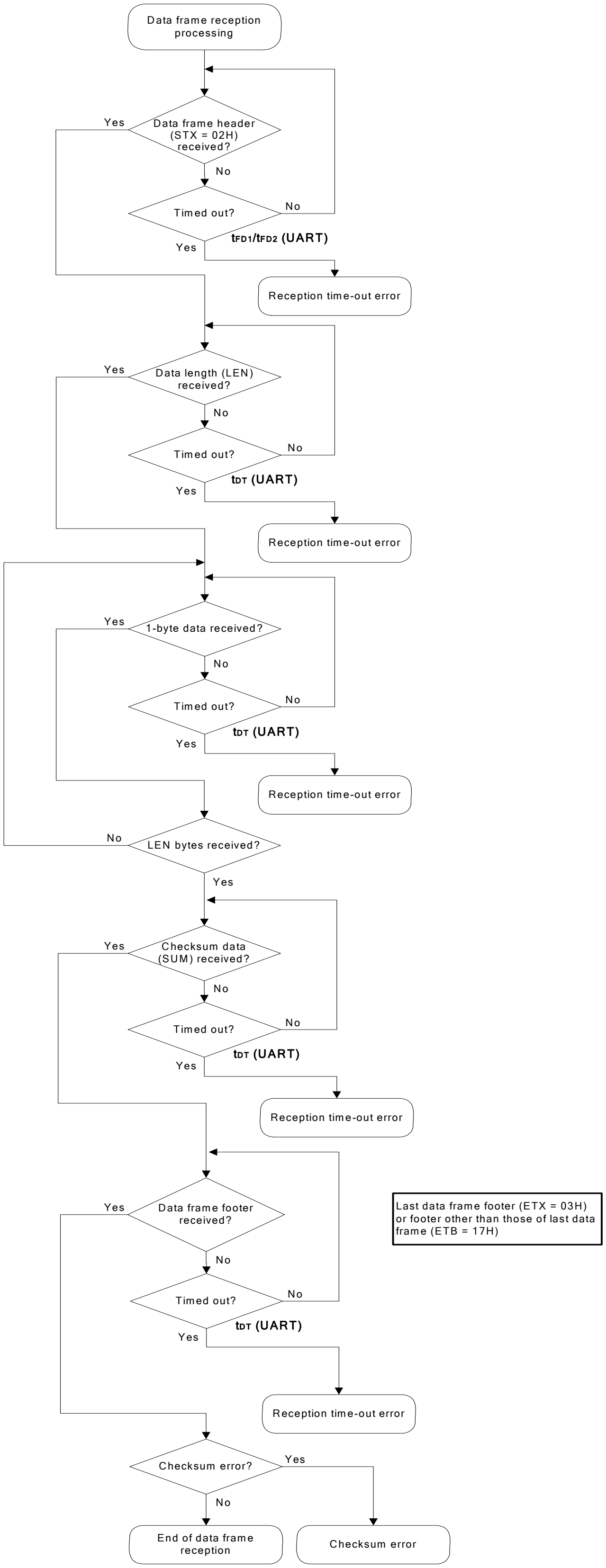

Command frame transmission processing
Command frame header (SOH = 01H) transmission
Wait between data transmissions
tDR (UART)
Data length (LEN) transmission
Wait between data transmissions
tDR (UART)
Command number (COM) transmission
(LEN – 1) bytes transmitted?
Yes
No
Wait between data transmissions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

<R>

4.2 Data Frame Transmission Processing Flowchart


Data frame transmission processing
Data frame header (STX = 02H) transmission
Wait between data transmissions
t_{DR} (UART)
Data length (LEN) transmission
LEN bytes transmitted?
Yes
No
Wait between data transmissions
t_{DR} (UART)
Transmits 1-byte data
Wait between data transmissions
t_{DR} (UART)
Checksum data (SUM) transmission
Wait between data transmissions
t_{DR} (UART)
Last data frame?
No
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

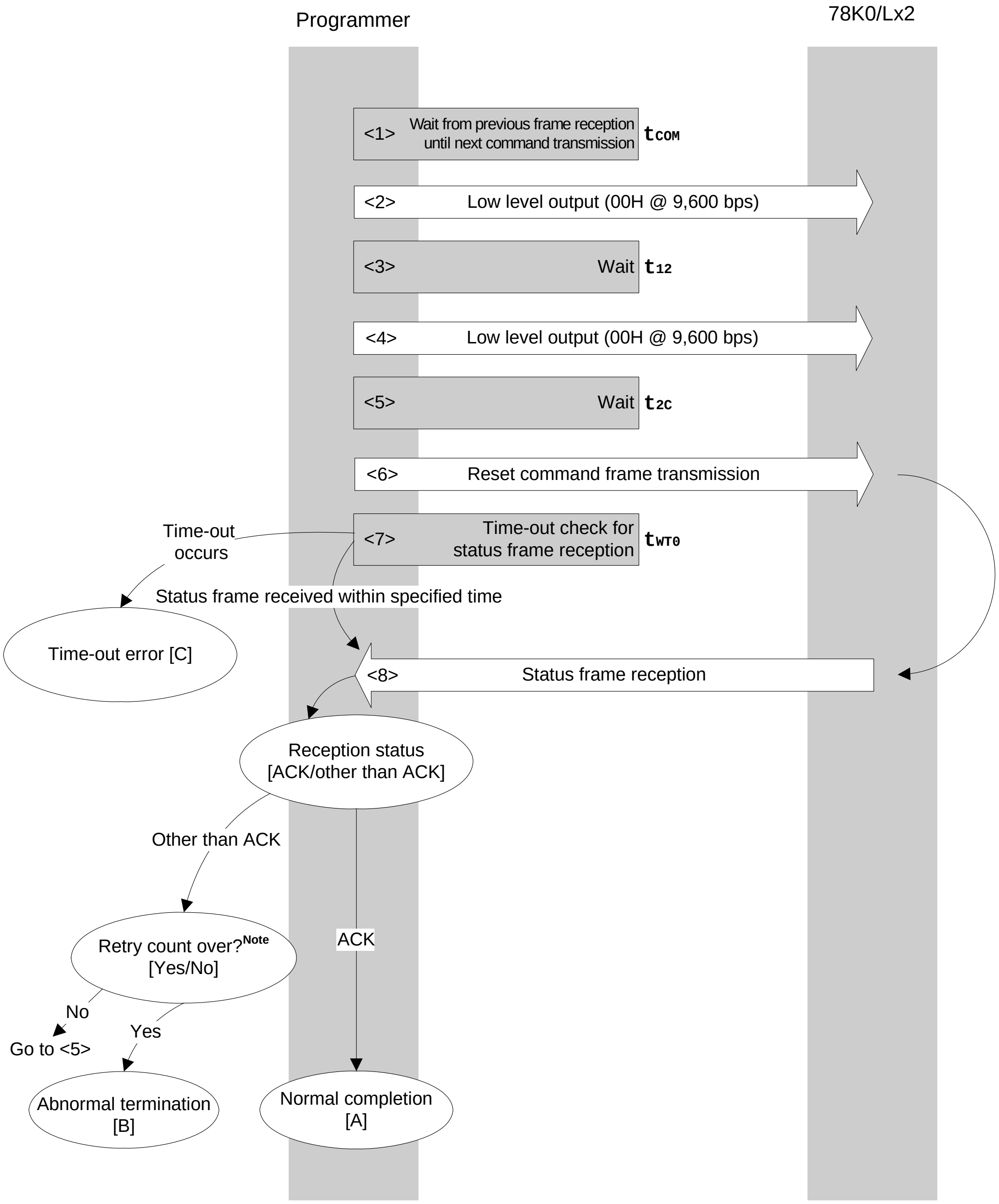
<R> 4.3 Data Frame Reception Processing Flowchart



4.4 Reset Command

4.4.1 Processing sequence chart

Reset command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission tCOM
<2> Low level output (00H @ 9,600 bps)
<3> Wait t12
<4> Low level output (00H @ 9,600 bps)
<5> Wait t2C
<6> Reset command frame transmission
<7> Time-out check for status frame reception tWT0
Time-out occurs
Status frame received within specified time
Time-out error [C]
<8> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Retry count over?Note [Yes/No]
ACK
No
Go to <5>
Yes
Abnormal termination [B]
Normal completion [A]

Note Do not exceed the retry count for the reset command transmission (up to 16 times).

4.4.2 Description of processing sequence

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

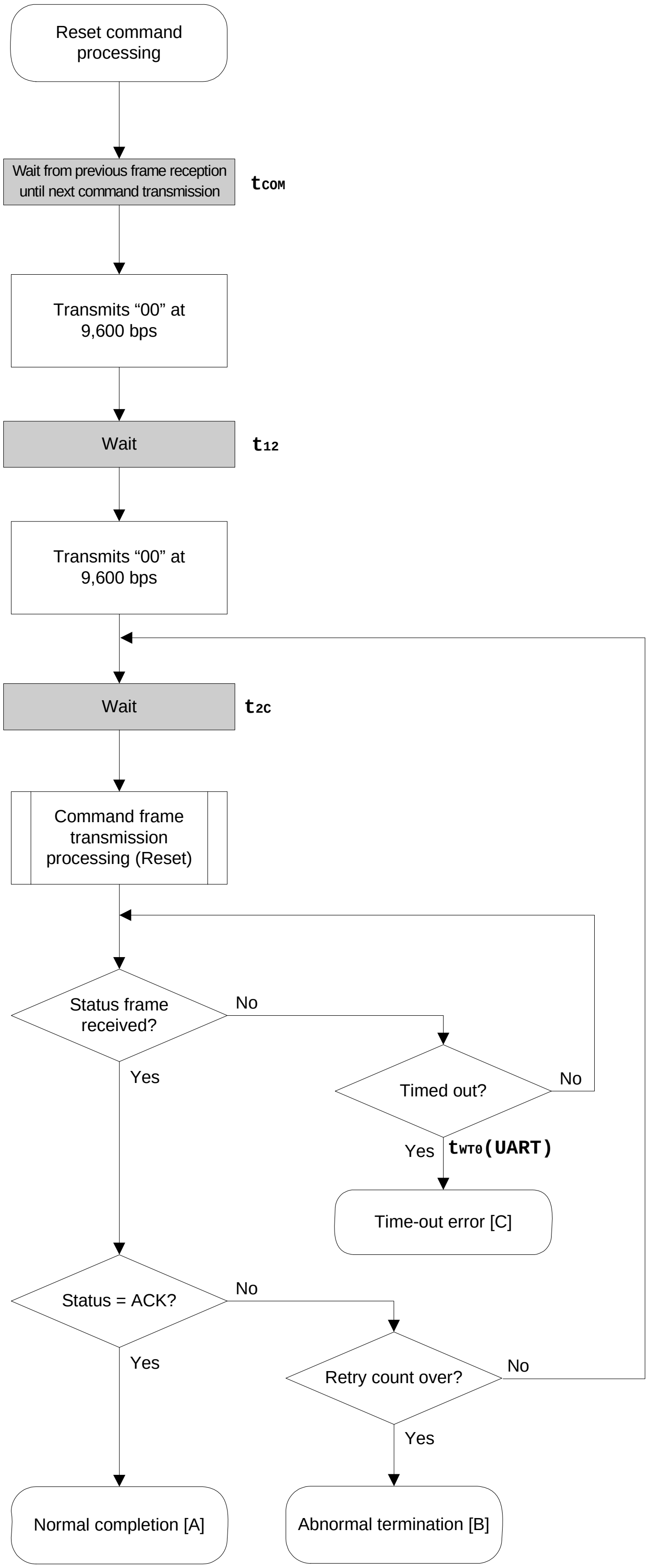
<8> The status code is checked.

When ST1 = ACK: Normal completion [A]
When ST1 ≠ 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].

4.4.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and synchronization between the programmer and the 78K0/Lx2 has been established.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.

4.4.4 Flowchart



4.4.5 Sample program

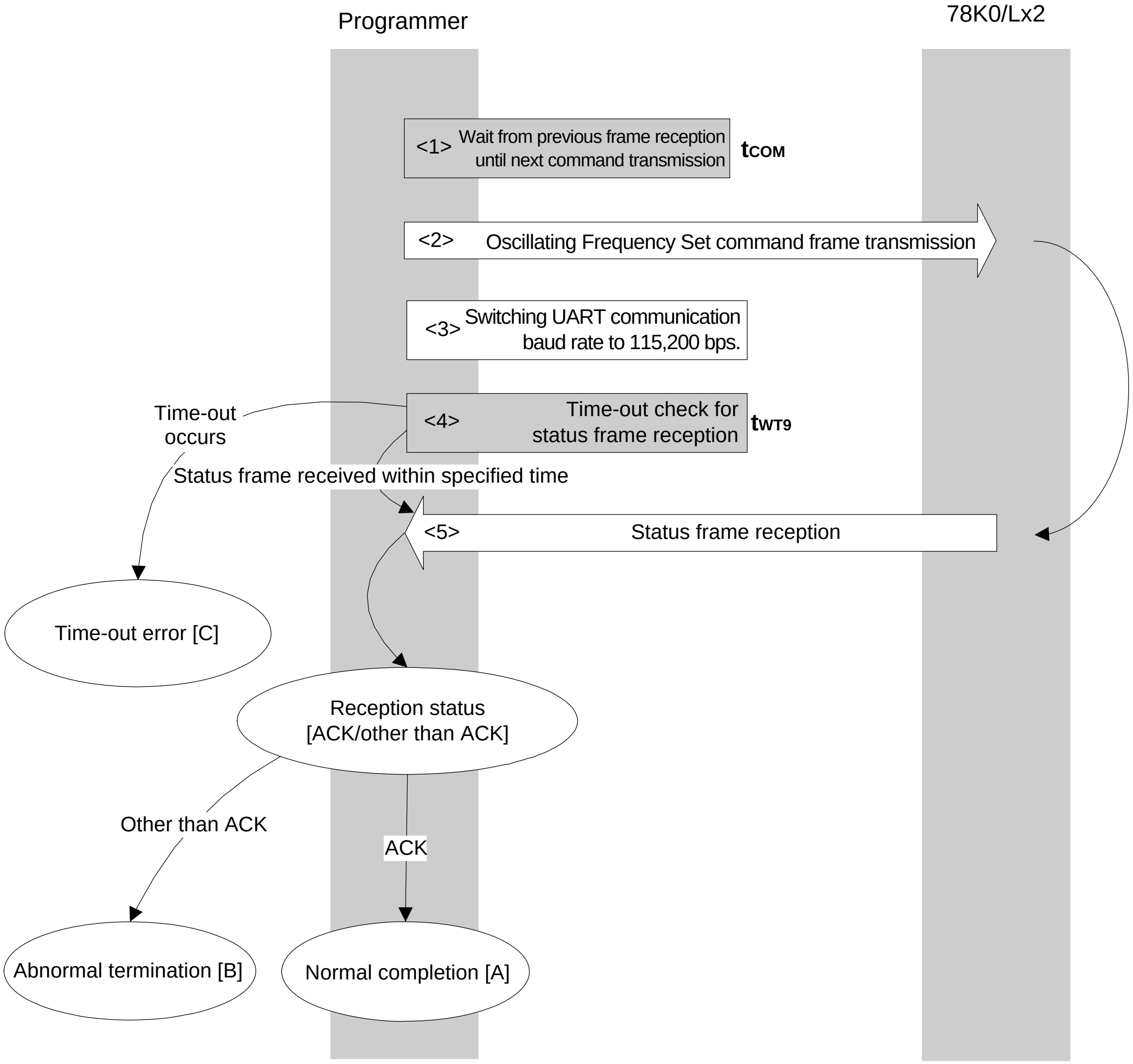
The following shows a sample program for Reset command processing.

```
/*****************************************************************/
/*                                                               */
/* Reset command                                                 */
/*                                                               */
/*****************************************************************/
/* [r] u16        ... error code                                 */
/*****************************************************************/
u16         fl_ua_reset(void)
{
     u16    rc;
     u32    retry;

     set_uart0_br(BR_9600);     // change to 9600bps

     fl_wait(tCOM);             // wait
     putc_ua(0x00);             // send 0x00 @ 9600bps

     fl_wait(t12); // wait
     putc_ua(0x00);             // send 0x00 @ 9600bps

     for (retry = 0; retry < tRS; retry++){

            fl_wait(t2C); // wait

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

            rc = get_sfrm_ua(fl_ua_sfrm, tWT0_TO);
            if (rc == FLC_DFTO_ERR)          // t.o. ?
                   break;                    // yes // case [C]

            if (rc == FLC_ACK){              // ACK ?
                   break;                    // yes // case [A]
            }
            else{
                   NOP();
            }
            //continue;                             // case [B] (if exit from loop)
     }
//   switch(rc) {
//
//          case   FLC_NO_ERR:   return rc;    break; // case [A]
//          case   FLC_DFTO_ERR: return rc;    break; // case [C]
//          default:             return rc;    break; // case [B]
//   }
     return rc;
}
```

4.5 Oscillating Frequency Set Command

4.5.1 Processing sequence chart

Oscillating Frequency Set command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Oscillating Frequency Set command frame transmission
<3> Switching UART communication baud rate to 115,200 bps.
<4> Time-out check for status frame reception
tWT9
Time-out occurs
Status frame received within specified time
<5> Status frame reception
Time-out error [C]
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
Normal completion [A]

4.5.2 Description of processing sequence

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

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

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

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

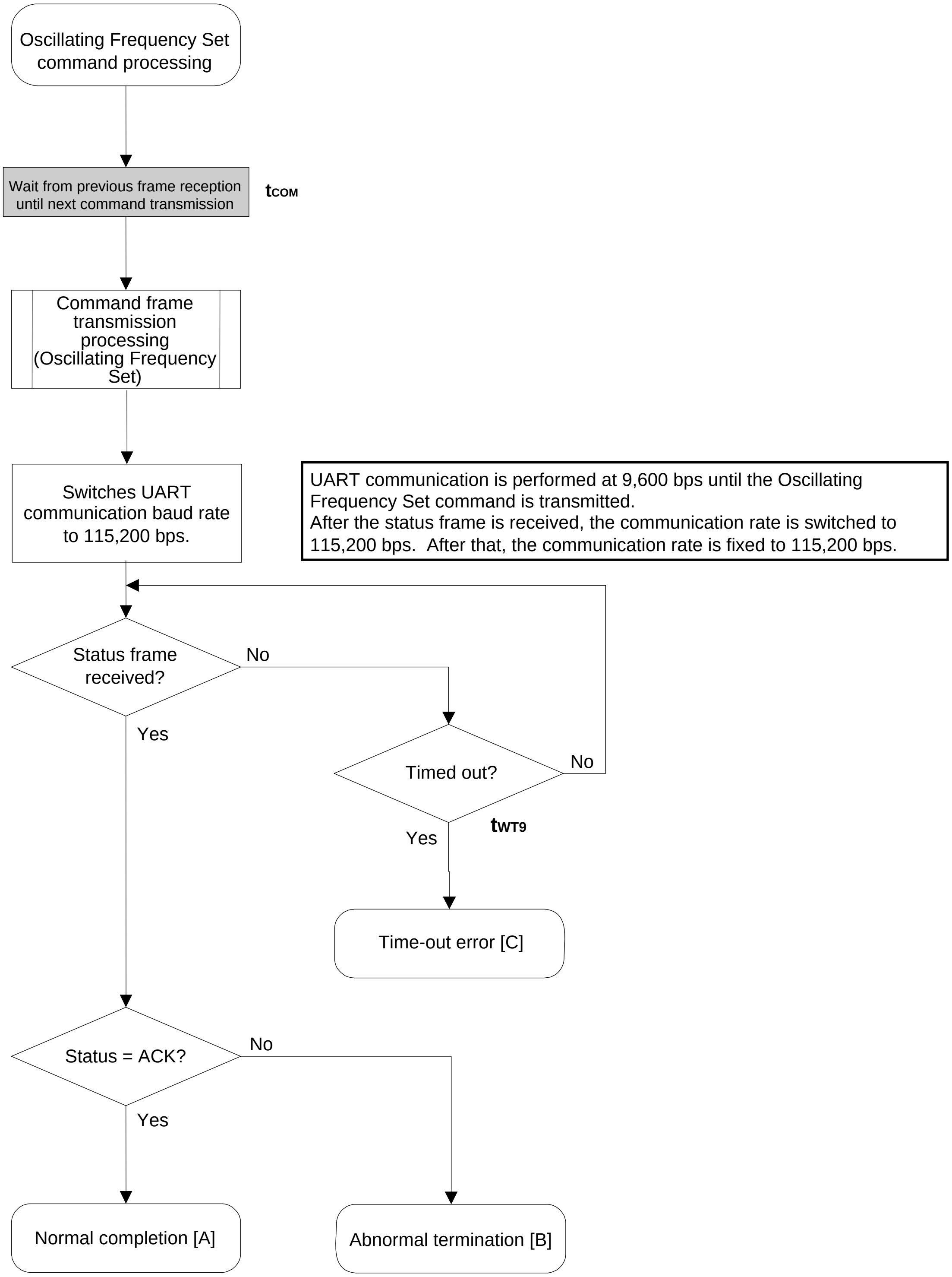
<5> The status code is checked.

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

4.5.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the operating frequency was correctly set to the 78K0/Lx2.
Abnormal termination [B]	Parameter error	05H	The oscillation frequency value is out of range.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.

4.5.4 Flowchart


Oscillating Frequency Set command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Oscillating Frequency Set)
Switches UART communication baud rate to 115,200 bps.
Status frame received?
No
Yes
Timed out?
No
Yes
tWT9
Time-out error [C]
Status = ACK?
No
Yes
Normal completion [A]
Abnormal termination [B]

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

4.5.5 Sample program

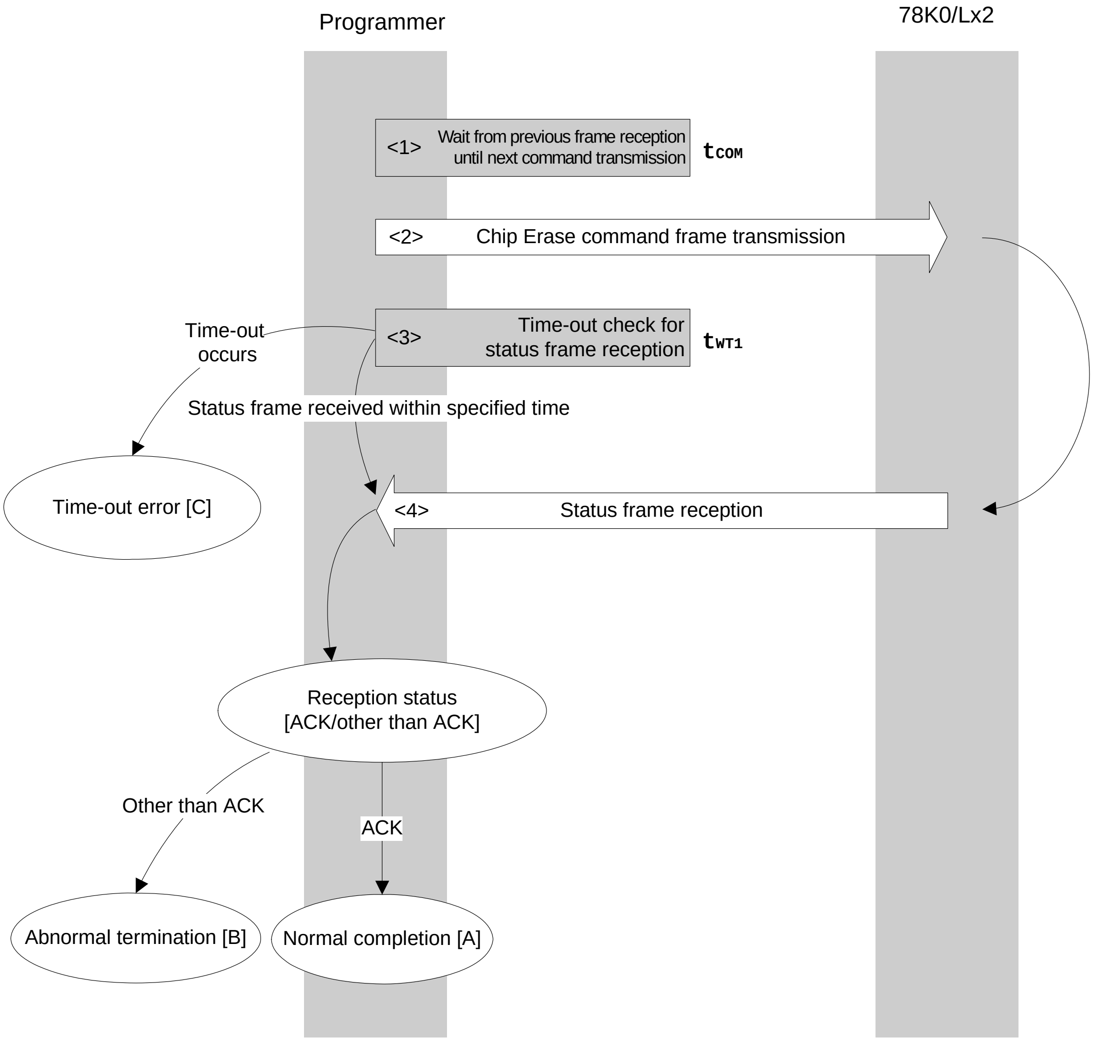
The following shows a sample program for Oscillating Frequency Set command processing.

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

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


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


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


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

    return rc;
}
```

4.6 Chip Erase Command

4.6.1 Processing sequence chart

Chip Erase command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Chip Erase command frame transmission
<3> Time-out check for status frame reception
tWT1
Time-out occurs
Status frame received within specified time
Time-out error [C]
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
Normal completion [A]

4.6.2 Description of processing sequence

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

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

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

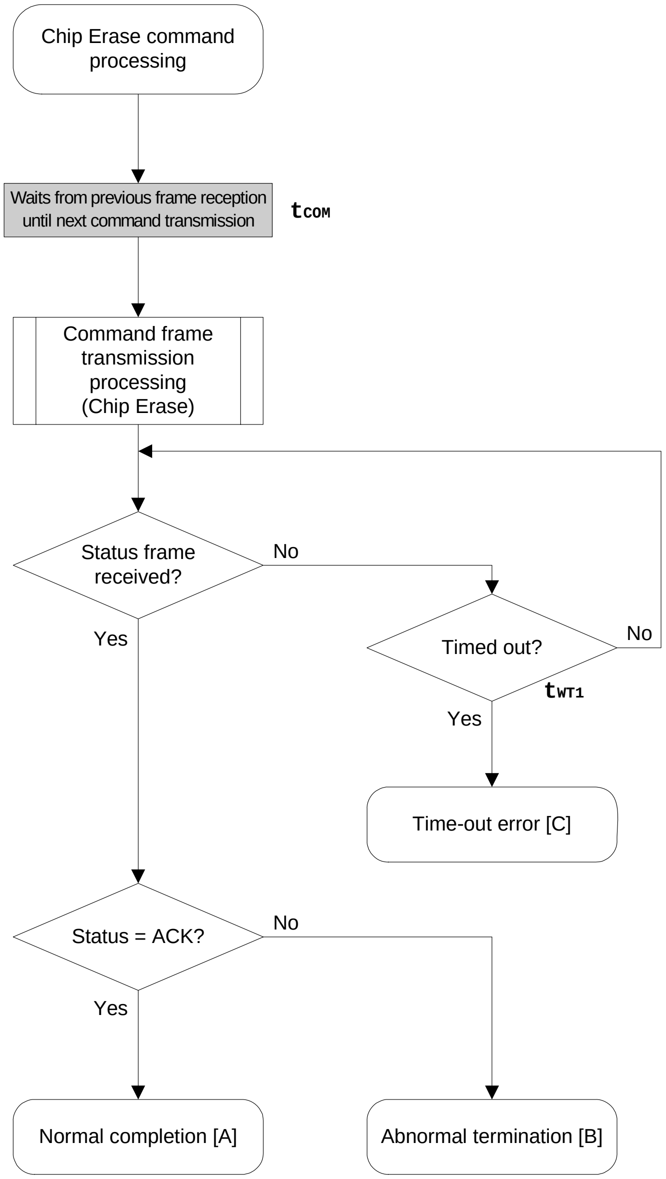
<4> The status code is checked.

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

4.6.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and chip erase was performed normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Protect error	10H	Chip Erase command is prohibited by 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.
Time-out error [C]		–	The status frame was not received within the specified time.

4.6.4 Flowchart


Chip Erase command processing
Waits from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Chip Erase)
Status frame received?
No
Yes
Timed out?
No
tWT1
Yes
Time-out error [C]
Status = ACK?
No
Yes
Normal completion [A]
Abnormal termination [B]

4.6.5 Sample program

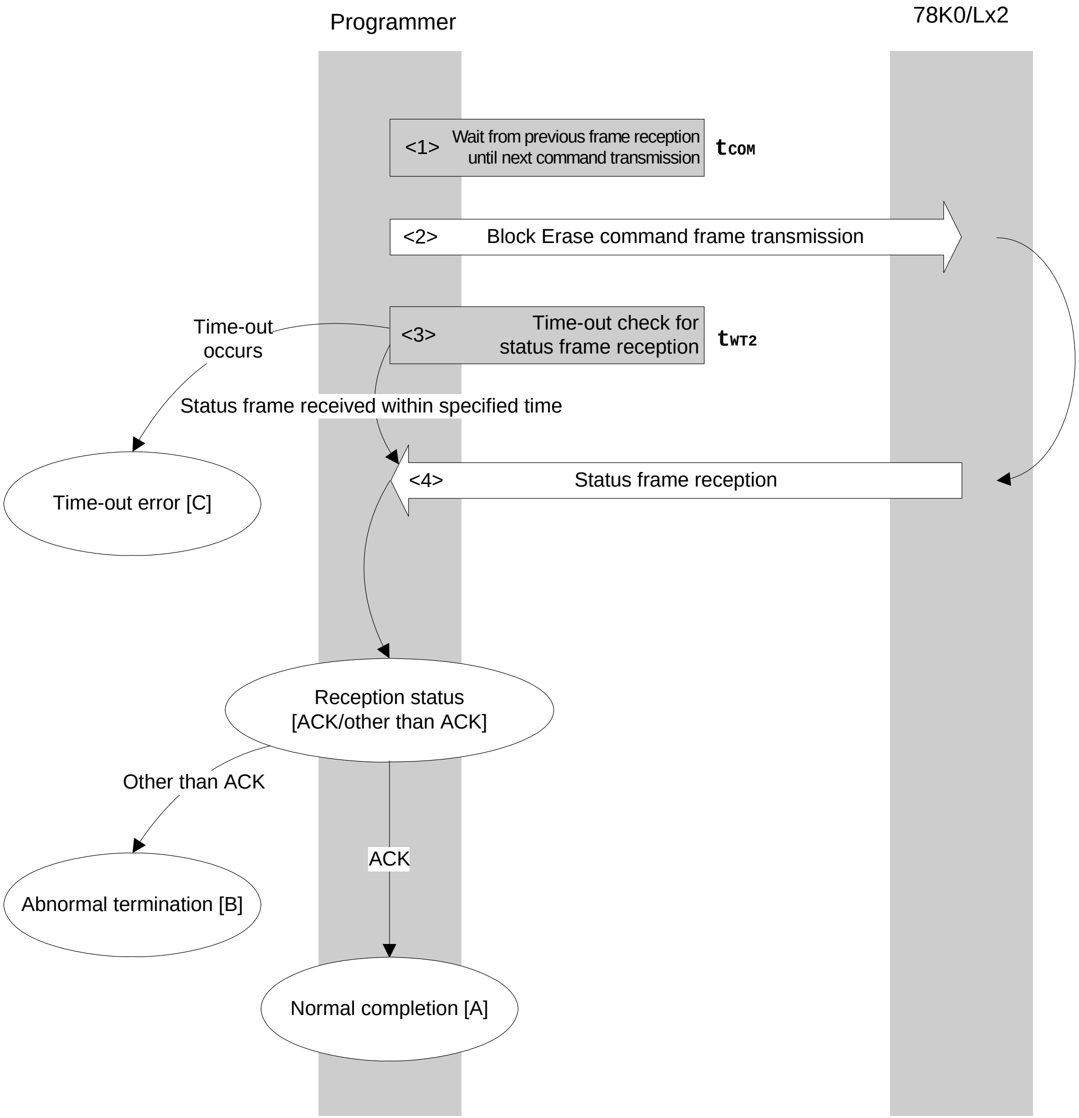
The following shows a sample program for Chip Erase command processing.

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

}
```

4.7 Block Erase Command

4.7.1 Processing sequence chart

Block Erase command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Block Erase command frame transmission
<3> Time-out check for status frame reception
tWT2
Time-out occurs
Status frame received within specified time
Time-out error [C]
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
Normal completion [A]

4.7.2 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}).

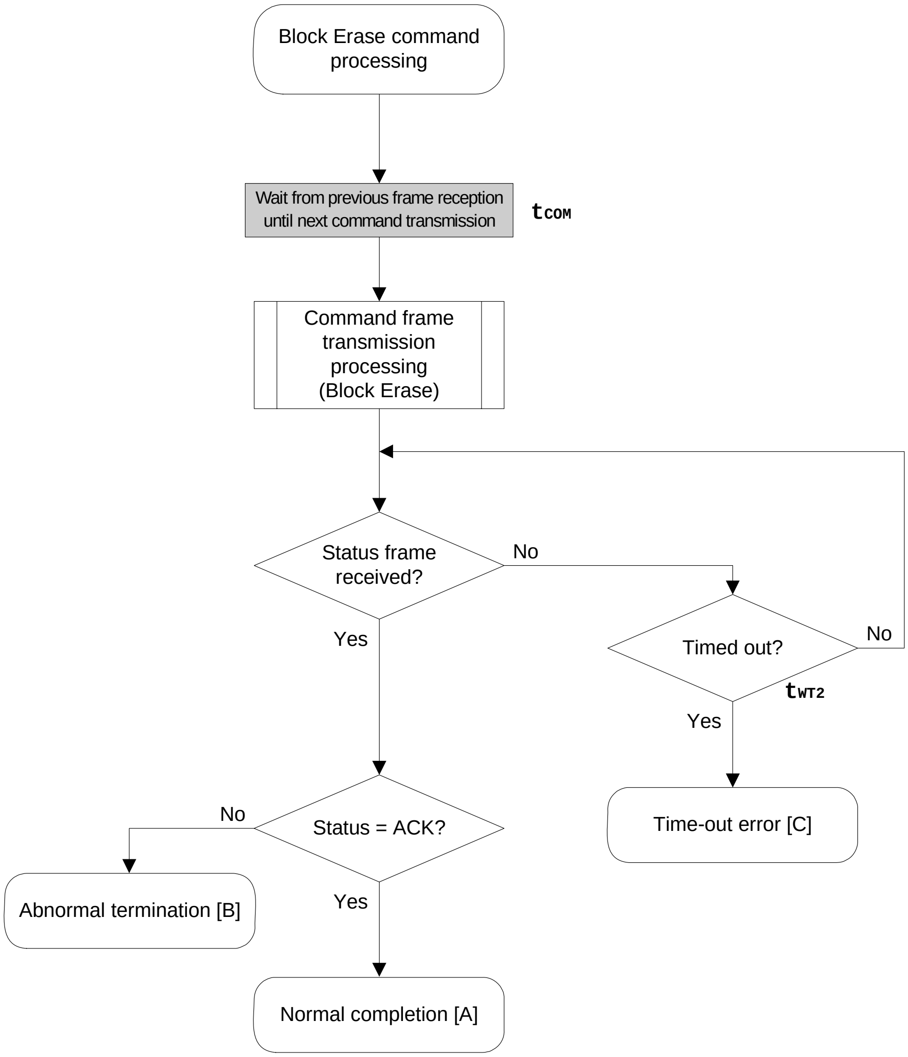
<4> The status code is checked.

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

4.7.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and block erase was performed normally.
Abnormal termination [B]	Parameter error	05H	The number of blocks is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Block Erase command is prohibited by 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.
Time-out error [C]		–	The status frame was not received within the specified time.

4.7.4 Flowchart


Block Erase command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Block Erase)
Status frame received?
No
Yes
Timed out?
No
tWT2
Yes
Time-out error [C]
No
Status = ACK?
Abnormal termination [B]
Yes
Normal completion [A]

4.7.5 Sample program

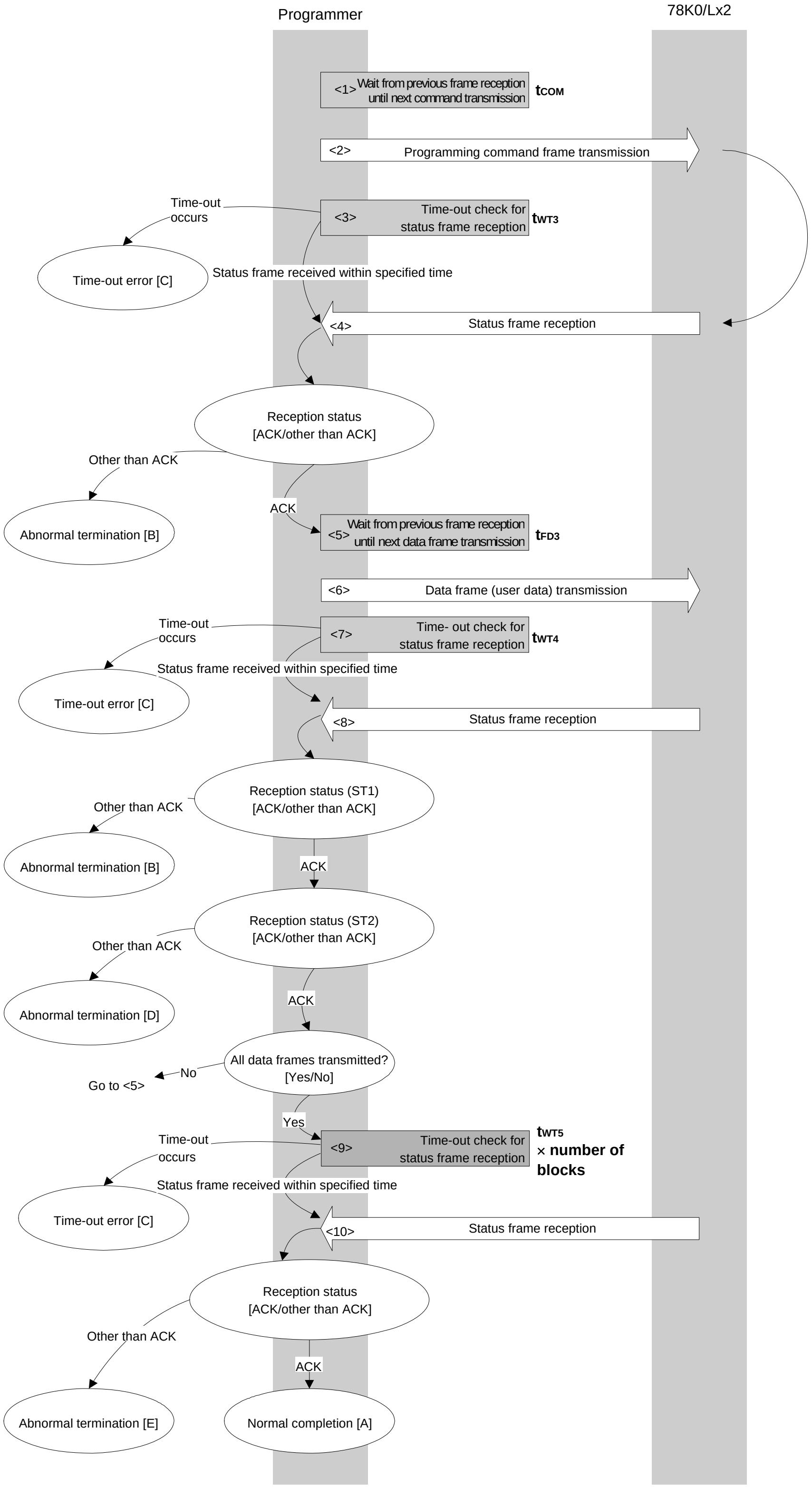
The following shows a sample program for Block Erase command processing for one block.

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

    u16    rc;
    u32    wt2_max;
    u32    top, bottom;

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

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

    wt2_max = make_wt2_max(sblk, eblk);

    fl_wait(tCOM);             // wait before sending command

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

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


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

    return rc;

}
```

4.8 Programming Command

4.8.1 Processing sequence chart

Programming command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Programming command frame transmission
Time-out occurs
<3> Time-out check for status frame reception
tWT3
Time-out error [C]
Status frame received within specified time
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Abnormal termination [B]
ACK
<5> Wait from previous frame reception until next data frame transmission
tFD3
<6> Data frame (user data) transmission
Time-out occurs
<7> Time- out check for status frame reception
tWT4
Time-out error [C]
Status frame received within specified time
<8> Status frame reception
Reception status (ST1) [ACK/other than ACK]
Other than ACK
Abnormal termination [B]
ACK
Reception status (ST2) [ACK/other than ACK]
Other than ACK
Abnormal termination [D]
ACK
All data frames transmitted? [Yes/No]
No
Go to <5>
Yes
Time-out occurs
<9> Time-out check for status frame reception
tWT5 × number of blocks
Time-out error [C]
Status frame received within specified time
<10> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Abnormal termination [E]
ACK
Normal completion [A]

4.8.2 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}).

<4> The status code is checked.

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

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

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

<7> A time-out check is performed from user data transmission until data frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT4}).

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

When ST1 ≠ 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 ≠ 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} × number of blocks).

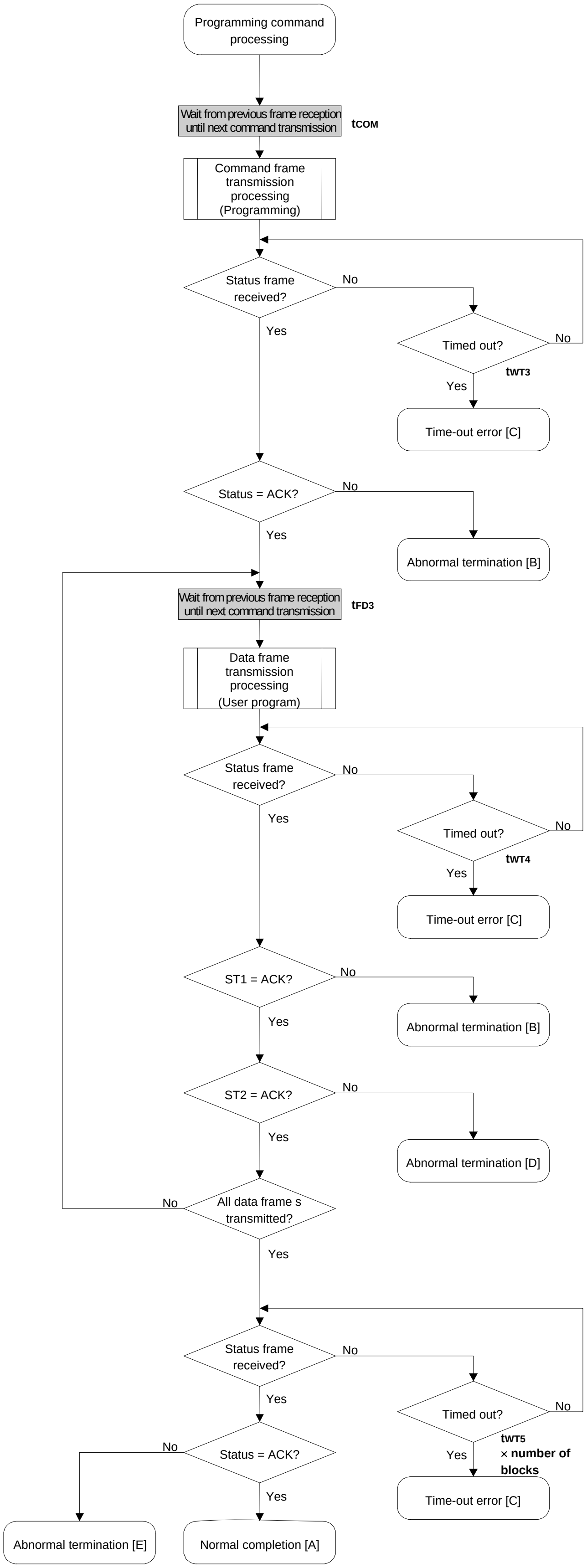
<10> The status code is checked.

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

4.8.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the user data was written normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or is not a multiple of 8.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Protect error	10H	Programming command is prohibited by 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]	Write error	1CH (ST2)	A write error has occurred.
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

4.8.4 Flowchart



4.8.5 Sample program

The following shows a sample program for Programming command processing.

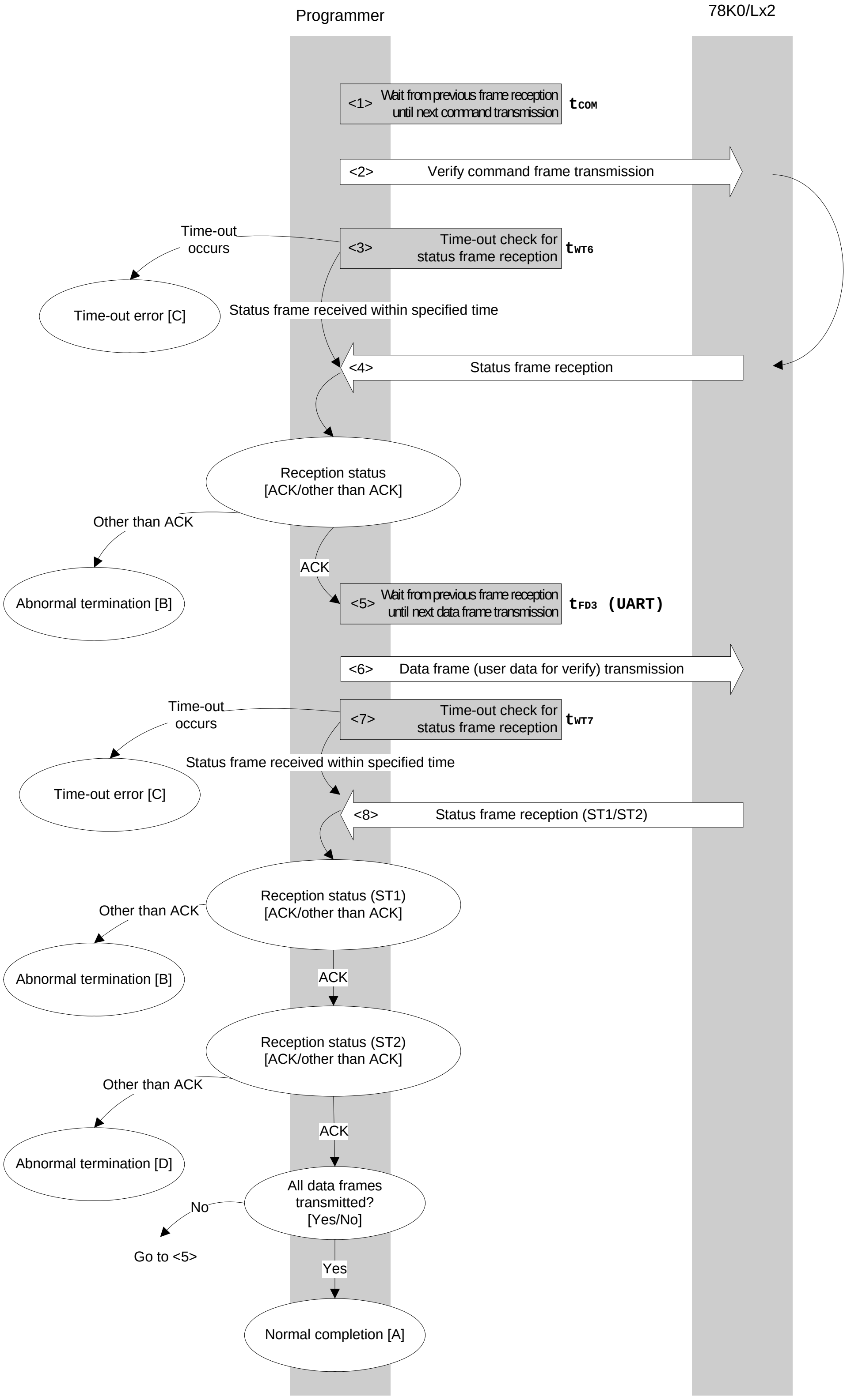
```
/****************************************************************/
/*                                                              */
/* Write command                                                */
/*                                                              */
/****************************************************************/
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/

#define           fl_st2_ua    (fl_ua_sfrm[OFS_STA_PLD+1])

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

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

    block_num = get_block_num(top, bottom); // get block num

    /************************************************/
    /*      send command & check status             */
    /************************************************/
    fl_wait(tCOM);             // wait before sending command

    put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command

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

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

```
     while(1){

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


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

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

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

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

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

4.9 Verify Command

4.9.1 Processing sequence chart

Verify command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Verify command frame transmission
Time-out occurs
<3> Time-out check for status frame reception
tWT6
Time-out error [C]
Status frame received within specified time
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Abnormal termination [B]
ACK
<5> Wait from previous frame reception until next data frame transmission
tFD3 (UART)
<6> Data frame (user data for verify) transmission
Time-out occurs
<7> Time-out check for status frame reception
tWT7
Time-out error [C]
Status frame received within specified time
<8> Status frame reception (ST1/ST2)
Reception status (ST1) [ACK/other than ACK]
Other than ACK
Abnormal termination [B]
ACK
Reception status (ST2) [ACK/other than ACK]
Other than ACK
Abnormal termination [D]
ACK
All data frames transmitted? [Yes/No]
No
Go to <5>
Yes
Normal completion [A]

4.9.2 Description of processing sequence

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

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

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

<4> The status code is checked.

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

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

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

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

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

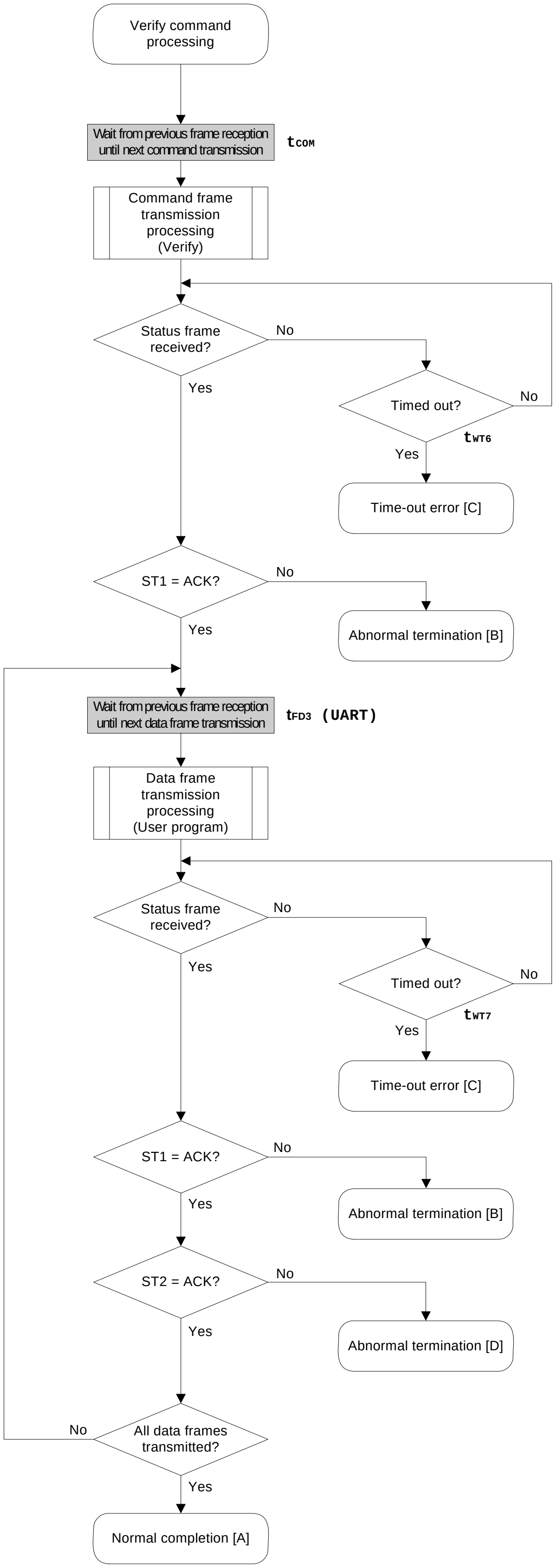
When ST1 ≠ ACK: Abnormal termination [B]
When ST1 = ACK: The following processing is performed according to the ST2 value.

- 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 ≠ ACK: Abnormal termination [D]

4.9.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the verify was completed normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range.
	Checksum error	07H	The checksum of the transmitted command frame or data frame is abnormal.
	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]	Verify error	0FH (ST2)	The verify has failed, or another error has occurred.

4.9.4 Flowchart



4.9.5 Sample program

The following shows a sample program for Verify command processing.

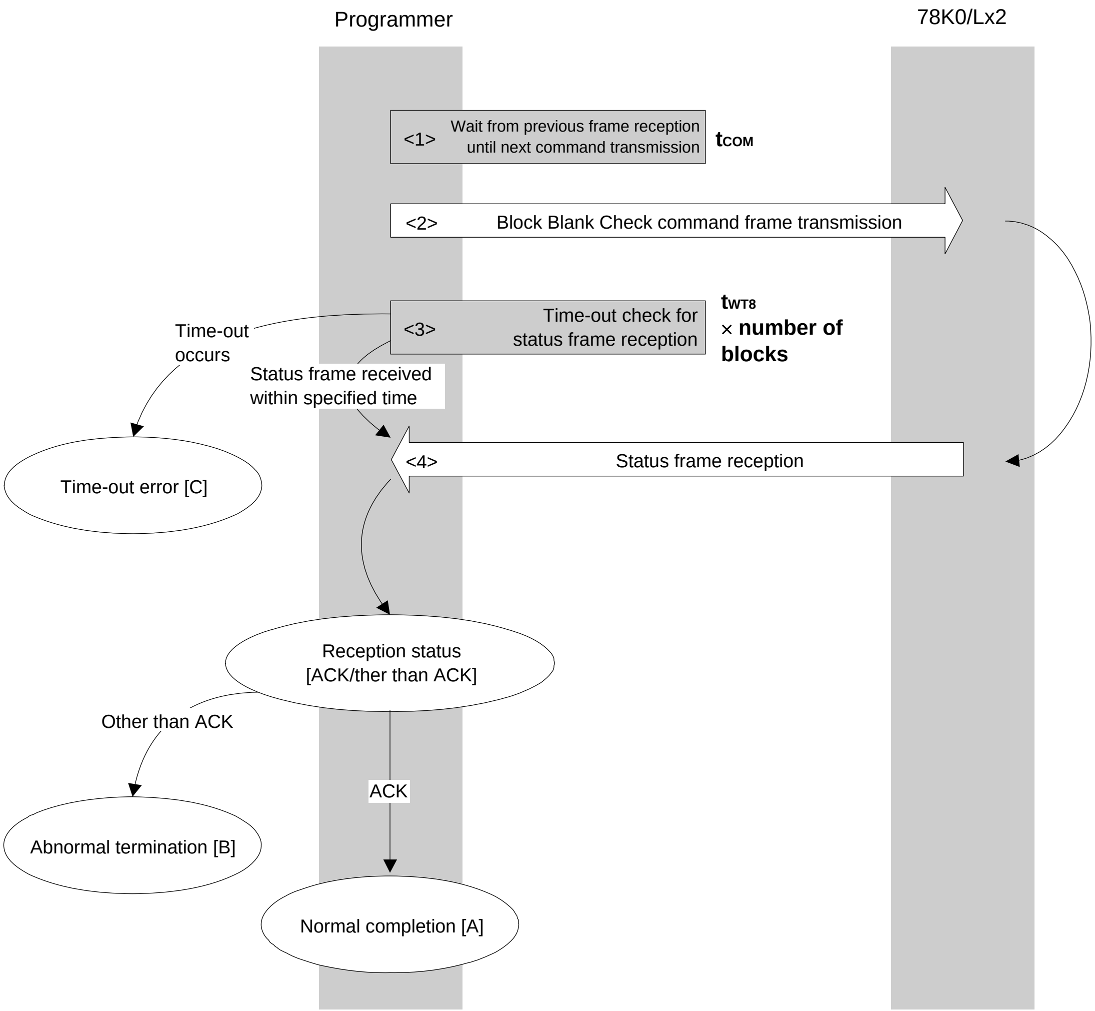
```
/****************************************************************/
/*                                                              */
/* Verify command                                               */
/*                                                              */
/****************************************************************/
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_ua_verify(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;

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


    /************************************************/
    /*      send command & check status             */
    /************************************************/

    fl_wait(tCOM);            // wait before sending command

    put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm);      // send VERIFY command

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

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

    while(1){

           // make send data frame
           if ((bottom - send_head) > 256){        // rest size > 256 ?
```

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

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

           fl_wait(tFD3_UA);
           put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

           rc = get_sfrm_ua(fl_ua_sfrm, tWT7_TO);         // get status frame
           switch(rc) {
                  case  FLC_NO_ERR:                break; // continue
           //     case  FLC_DFTO_ERR:return rc;    break; // case [C]
                  default:            return rc;   break; // case [B]
           }
           if (fl_st2_ua != FLST_ACK){             // ST2 = ACK ?
                  rc = decode_status(fl_st2_ua);   // No
                  return rc;                       // case [D]
           }
           if (is_end)                // send all user data ?
                  break;              // yes
           //continue;
     }
     return FLC_NO_ERR;  // case [A]
}
```

4.10 Block Blank Check Command

4.10.1 Processing sequence chart

Block Blank Check command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Block Blank Check command frame transmission
<3> Time-out check for status frame reception
tWT8 × number of blocks
Time-out occurs
Status frame received within specified time
Time-out error [C]
<4> Status frame reception
Reception status [ACK/ther than ACK]
Other than ACK
ACK
Abnormal termination [B]
Normal completion [A]

4.10.2 Description of processing sequence

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

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

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

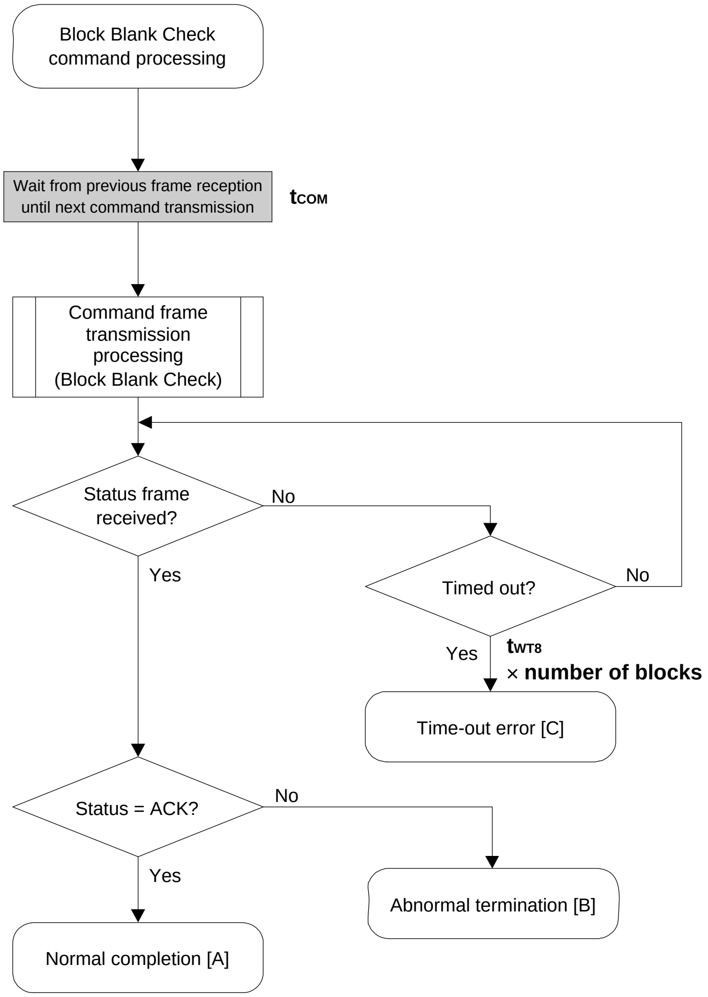
<4> The status code is checked.

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

4.10.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and all of the specified blocks are blank.
Abnormal termination [B]	Parameter error	05H	The number of blocks is out of range.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.

4.10.4 Flowchart


Block Blank Check command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Block Blank Check)
Status frame received?
No
Yes
Timed out?
No
Yes
tWT8 × number of blocks
Time-out error [C]
Status = ACK?
No
Yes
Abnormal termination [B]
Normal completion [A]

4.10.5 Sample program

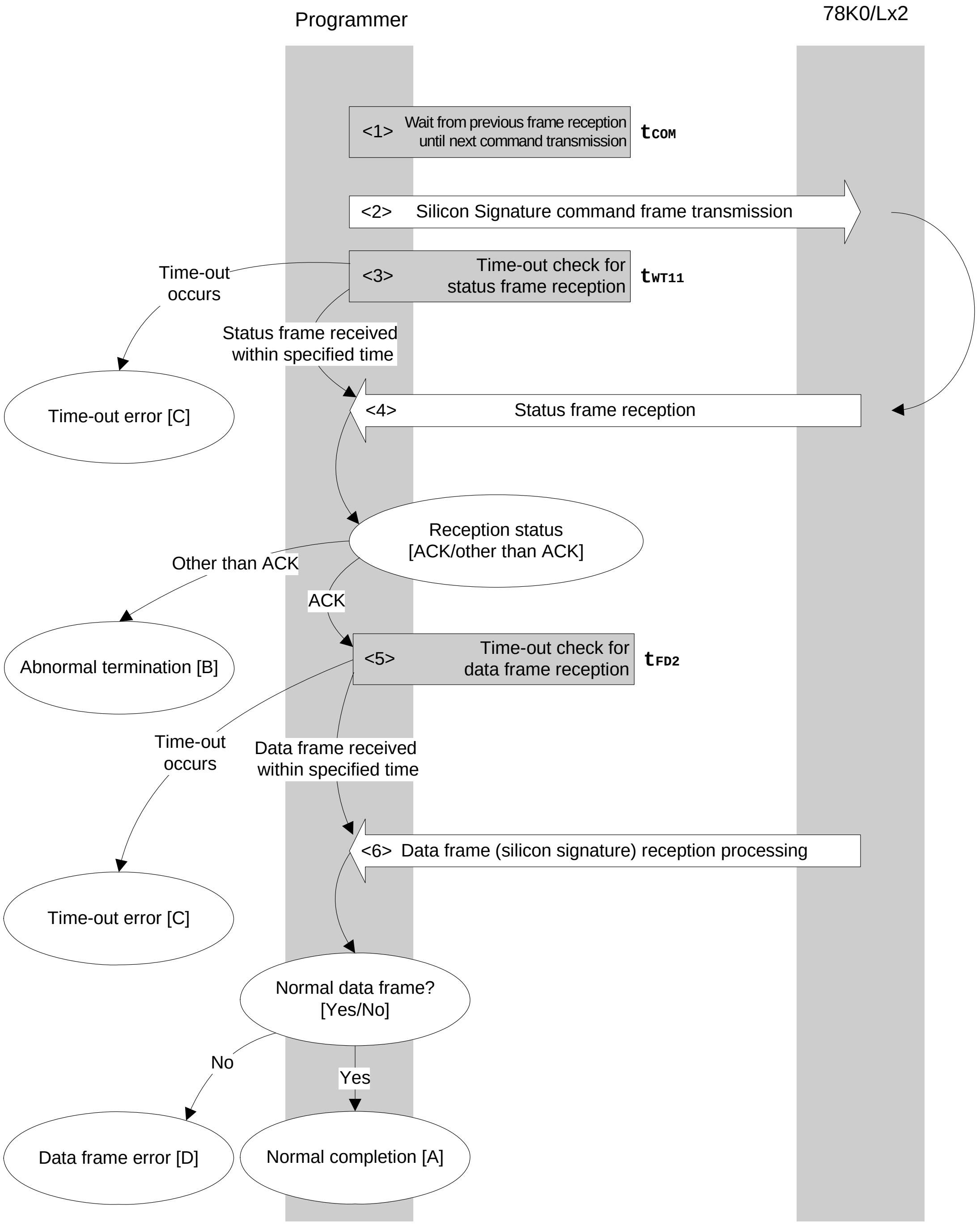
The following shows a sample program for Block Blank Check command processing.

```
/****************************************************************/
/*                                                              */
/* Block blank check command                                    */
/*                                                              */
/****************************************************************/
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_ua_blk_blank_chk(u32 top, u32 bottom)
{
    u16    rc;
    u16    block_num;

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

    fl_wait(tCOM);             // wait before sending command

    put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);


    rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num);  // get status frame
//  switch(rc) {
//
//         case   FLC_NO_ERR:  return rc;   break; // case [A]
//         case   FLC_DFTO_ERR:return rc;   break; // case [C]
//         default:            return rc;   break; // case [B]
//  }
    return rc;

}
```

4.11 Silicon Signature Command

4.11.1 Processing sequence chart

Silicon Signature command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Silicon Signature command frame transmission
<3> Time-out check for status frame reception
tWT11
Time-out occurs
Status frame received within specified time
Time-out error [C]
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
<5> Time-out check for data frame reception
tFD2
Time-out occurs
Data frame received within specified time
<6> Data frame (silicon signature) reception processing
Time-out error [C]
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

4.11.2 Description of processing sequence

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

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

<3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT11}).

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

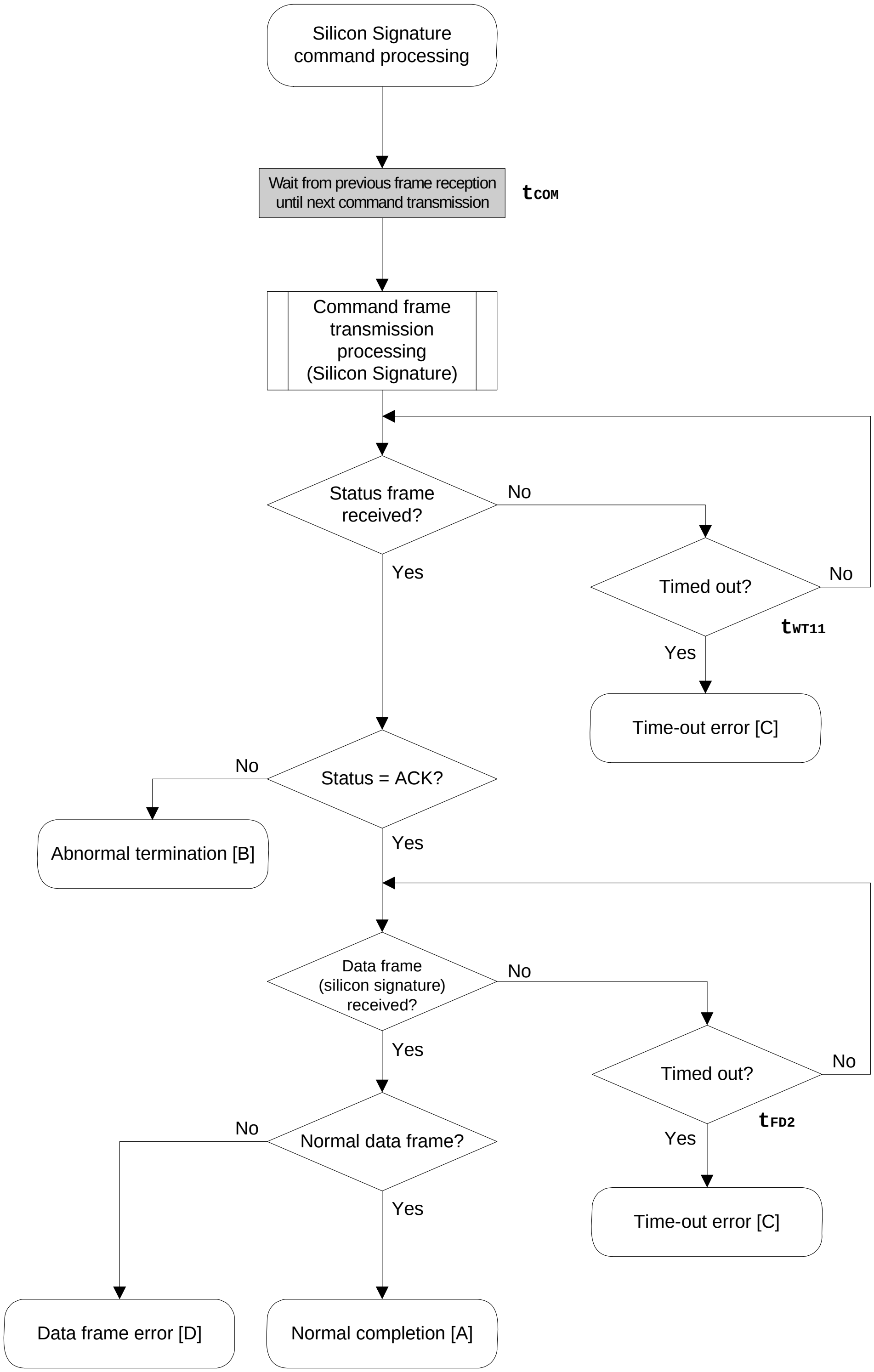
<6> The received data frame (silicon signature data) is checked.

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

4.11.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the silicon signature was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Read error	20H	Reading of security information failed.
Time-out error [C]		–	The status frame or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as silicon signature data is abnormal.

4.11.4 Flowchart



4.11.5 Sample program

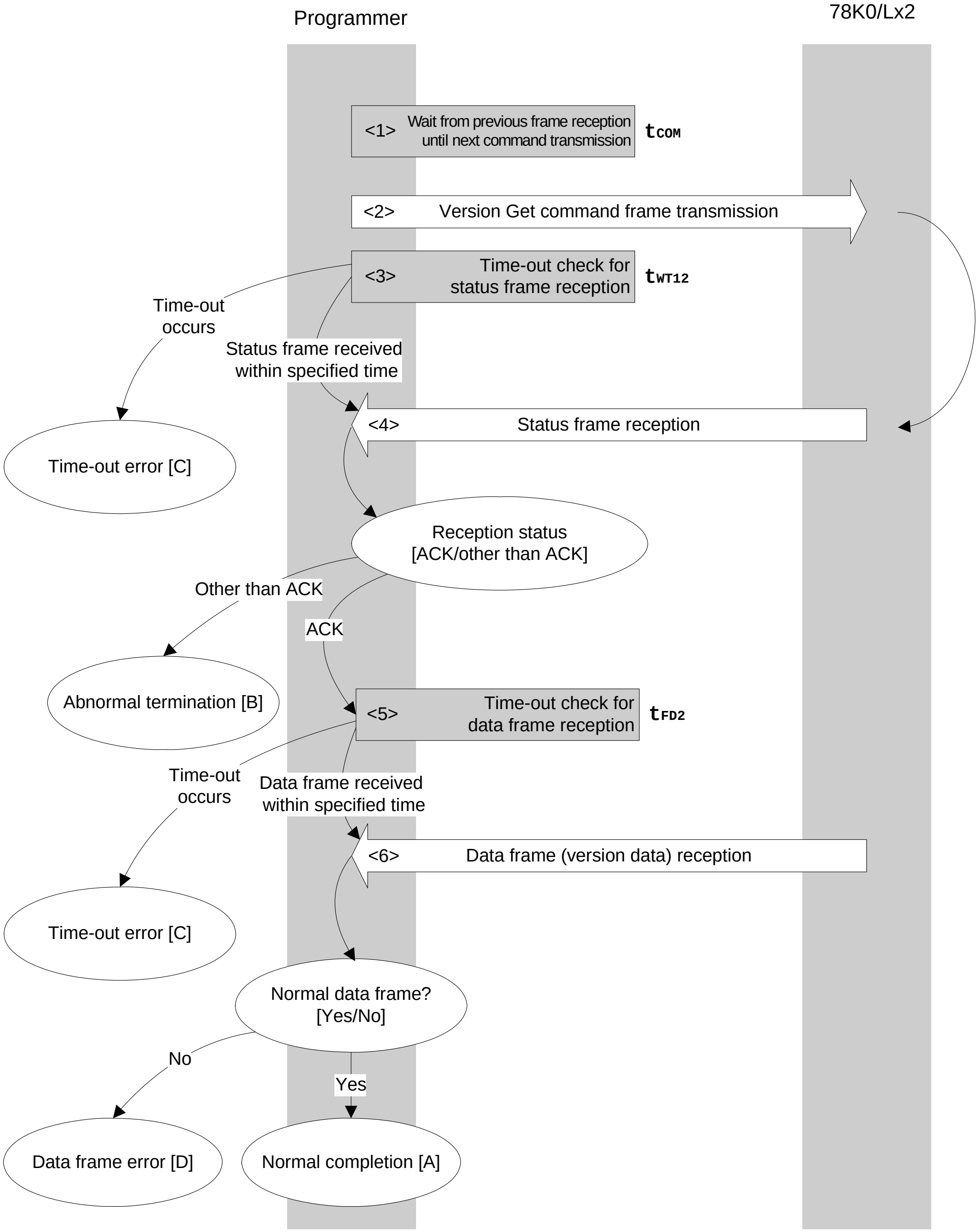
The following shows a sample program for Silicon Signature command processing.

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


    fl_wait(tCOM);             // wait before sending command

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

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


    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO);      // get status frame
    if (rc){                                // if error
           return rc;                       // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                          // copy Signature data
    return rc;                        // case [A]
}
```

4.12 Version Get Command

4.12.1 Processing sequence chart

Version Get command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Version Get command frame transmission
<3> Time-out check for status frame reception
tWT12
Time-out occurs
Status frame received within specified time
<4> Status frame reception
Time-out error [C]
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
<5> Time-out check for data frame reception
tFD2
Time-out occurs
Data frame received within specified time
<6> Data frame (version data) reception
Time-out error [C]
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

4.12.2 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}).

<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}).

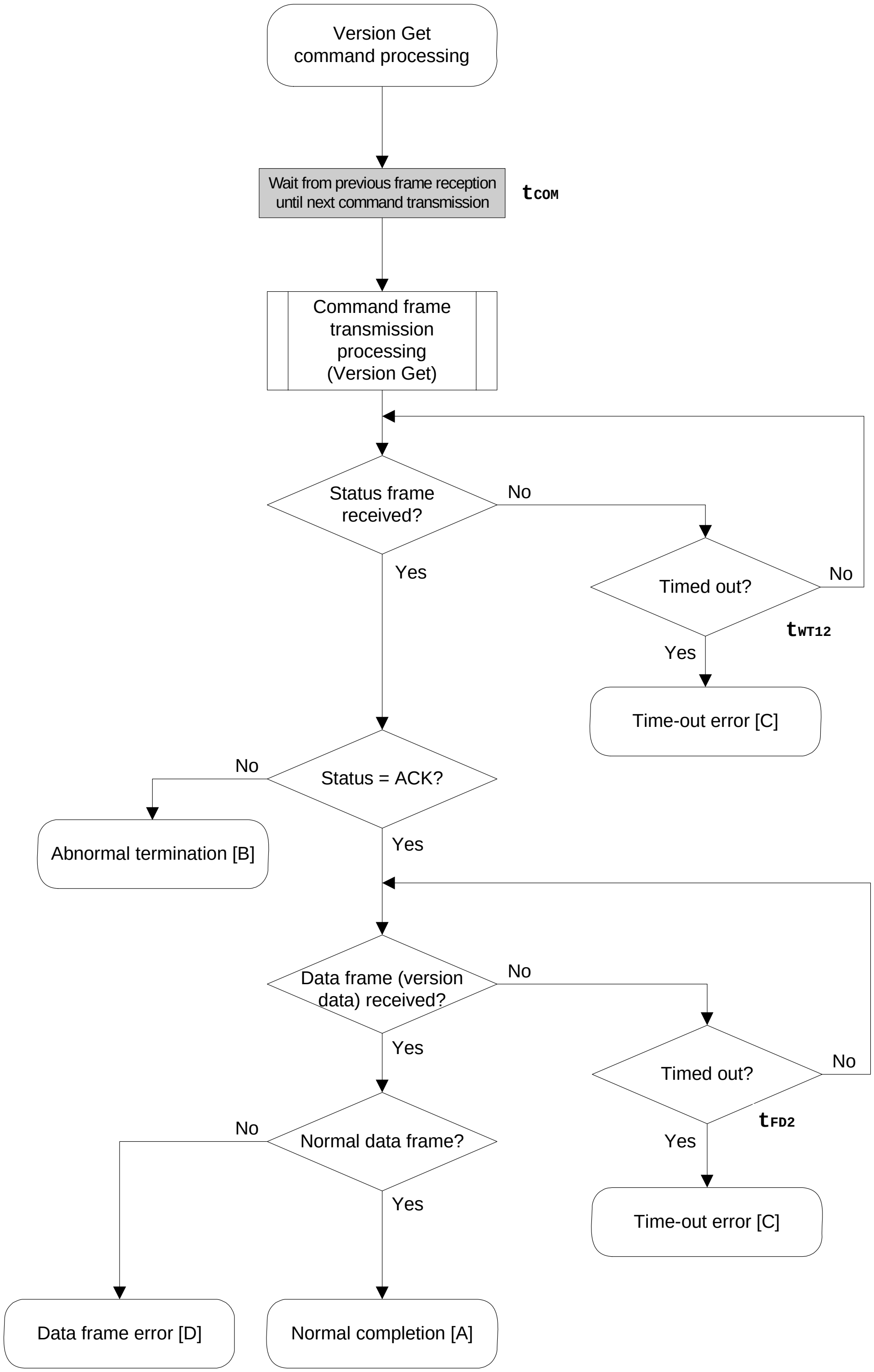
<6> The received data frame (version data) is checked.

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

4.12.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and version data was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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 or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data is abnormal.

4.12.4 Flowchart



4.12.5 Sample program

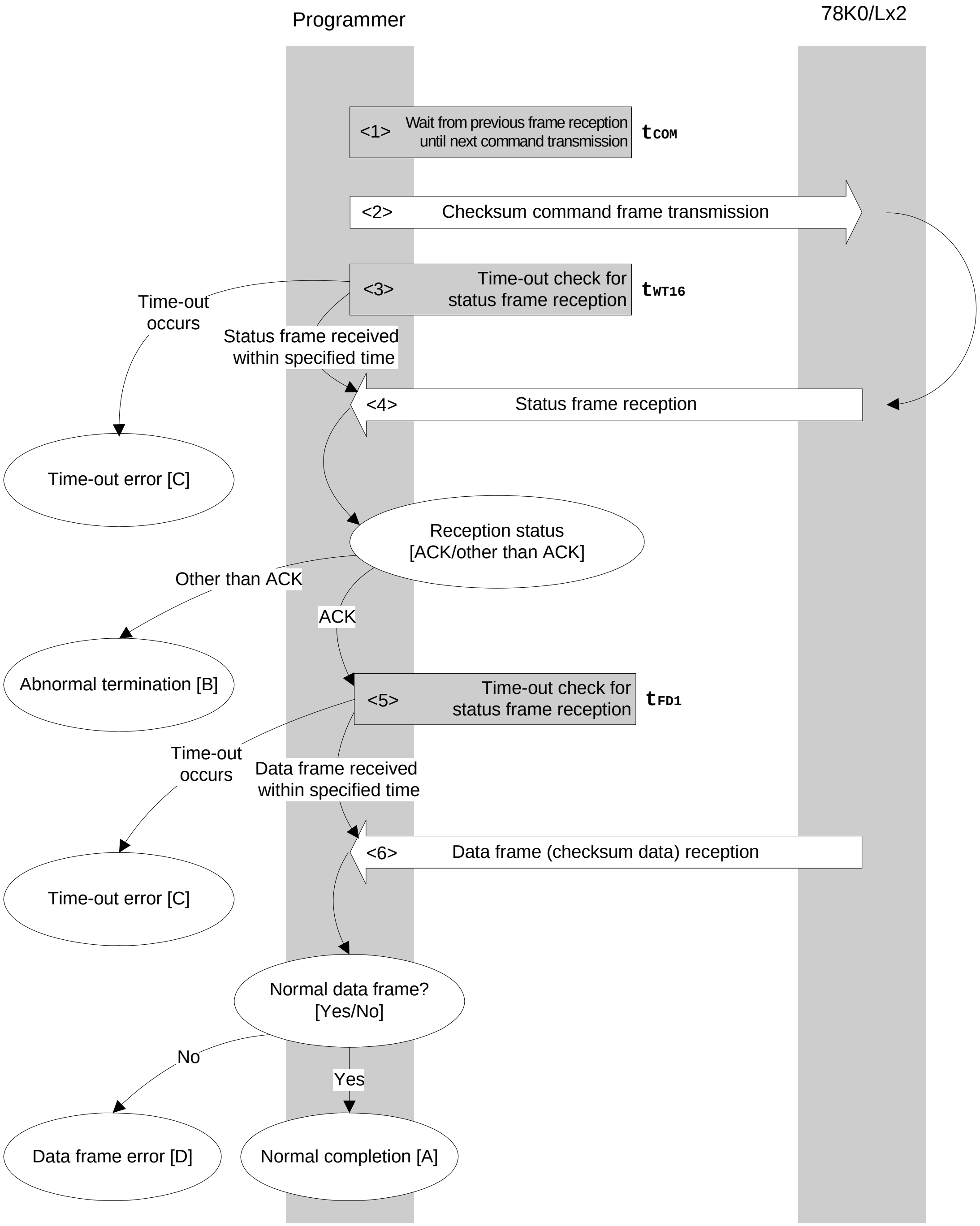
The following shows a sample program for Version Get command processing.

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


    fl_wait(tCOM);             // wait before sending command


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

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

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

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

4.13 Checksum Command

4.13.1 Processing sequence chart

Checksum command processing sequence



Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Checksum command frame transmission
<3> Time-out check for status frame reception
tWT16
Time-out occurs
Status frame received within specified time
<4> Status frame reception
Time-out error [C]
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [B]
<5> Time-out check for status frame reception
tFD1
Time-out occurs
Data frame received within specified time
<6> Data frame (checksum data) reception
Time-out error [C]
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

4.13.2 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> 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_{WT16}).

<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 (checksum data) reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{FD1}).

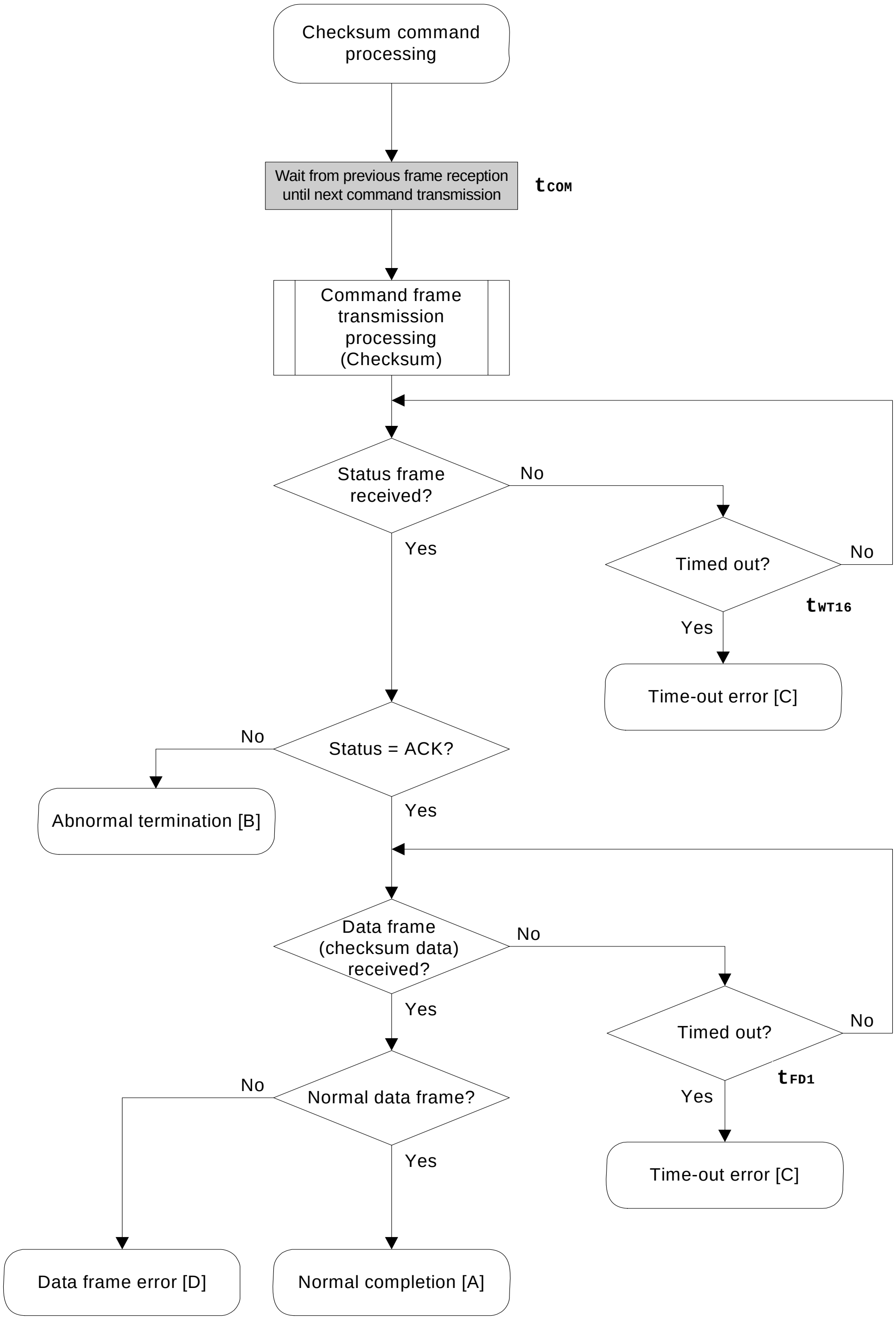
<6> The received data frame (checksum data) is checked.

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

4.13.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and checksum data was acquired normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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 or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data is abnormal.

4.13.4 Flowchart



4.13.5 Sample program

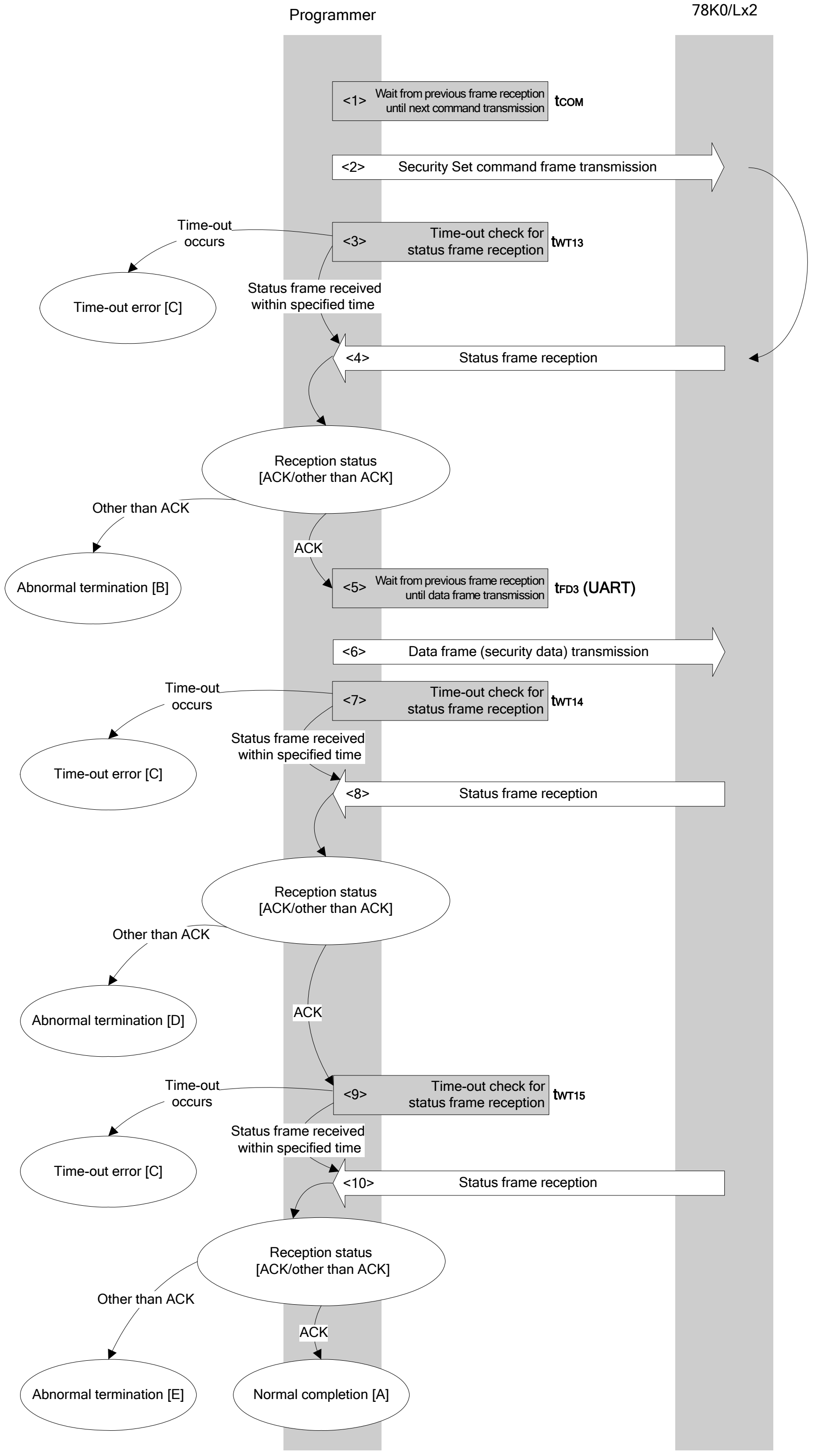
The following shows a sample program for Checksum command processing.

```
/****************************************************************/
/*                                                              */
/* Get checksum command                                         */
/*                                                              */
/****************************************************************/
/* [i] u16 *sum    ... pointer to checksum save area            */
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;

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


    /************************************************/
    /*      send command                            */
    /************************************************/

    fl_wait(tCOM);             // wait before sending command

    put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send GET VERSION command

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

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

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

}
```

4.14 Security Set Command

4.14.1 Processing sequence chart

Security Set command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Security Set command frame transmission
Time-out occurs
<3> Time-out check for status frame reception
tWT13
Time-out error [C]
Status frame received within specified time
<4> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Abnormal termination [B]
ACK
<5> Wait from previous frame reception until data frame transmission
tFD3 (UART)
<6> Data frame (security data) transmission
Time-out occurs
<7> Time-out check for status frame reception
tWT14
Time-out error [C]
Status frame received within specified time
<8> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
Abnormal termination [D]
ACK
Time-out occurs
<9> Time-out check for status frame reception
tWT15
Time-out error [C]
Status frame received within specified time
<10> Status frame reception
Reception status [ACK/other than ACK]
Other than ACK
ACK
Abnormal termination [E]
Normal completion [A]

4.14.2 Description of processing sequence

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

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

<3> 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_{WT13}).

<4> The status code is checked.

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

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

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

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

<8> The status code is checked.

When ST1 = ACK: Proceeds to <9>.
When ST1 ≠ 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_{WT15}).

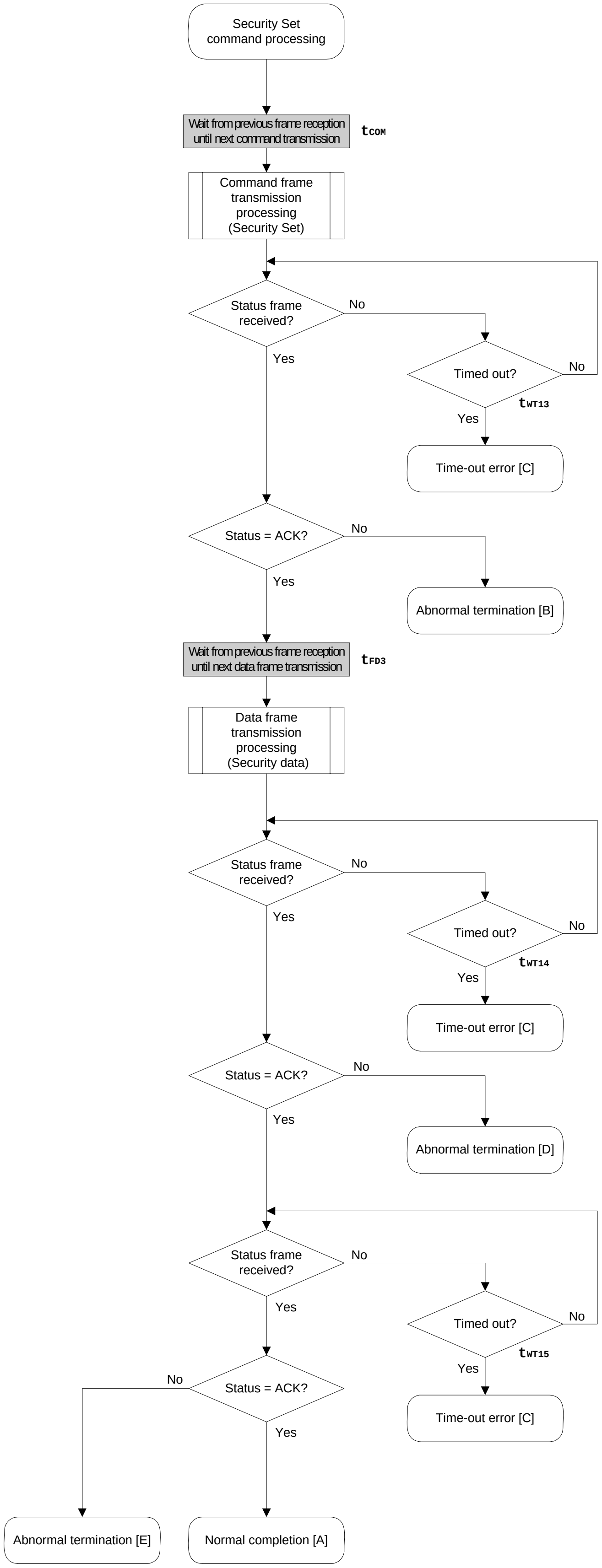
<10> The status code is checked.

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

4.14.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and security setting was performed normally.
Abnormal termination [B]	Parameter error	05H	Command information (parameter) is not 00H.
	Checksum error	07H	The checksum of the transmitted command frame or data frame is abnormal.
	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 or data frame was not received within the specified time.
Abnormal termination [D]	FLMD error	18H	A write error has occurred.
	Write error	1CH	A write error has occurred (including the case of security data already being set).
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

4.14.4 Flowchart



4.14.5 Sample program

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

```
/****************************************************************/
/*                                                              */
/* Set security flag command                                    */
/*                                                              */
/****************************************************************/
/* [i] u8 scf      ... Security flag data                       */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_ua_setscf(u8 scf)
{
    u16    rc;

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

    fl_txdata_frm[1] = 0x03;         // "BOT" (fixed 0x03)


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


    /*************************************************/
    /*      send data frame (security setting data) */
    /*************************************************/

    fl_wait(tFD3_UA);
```

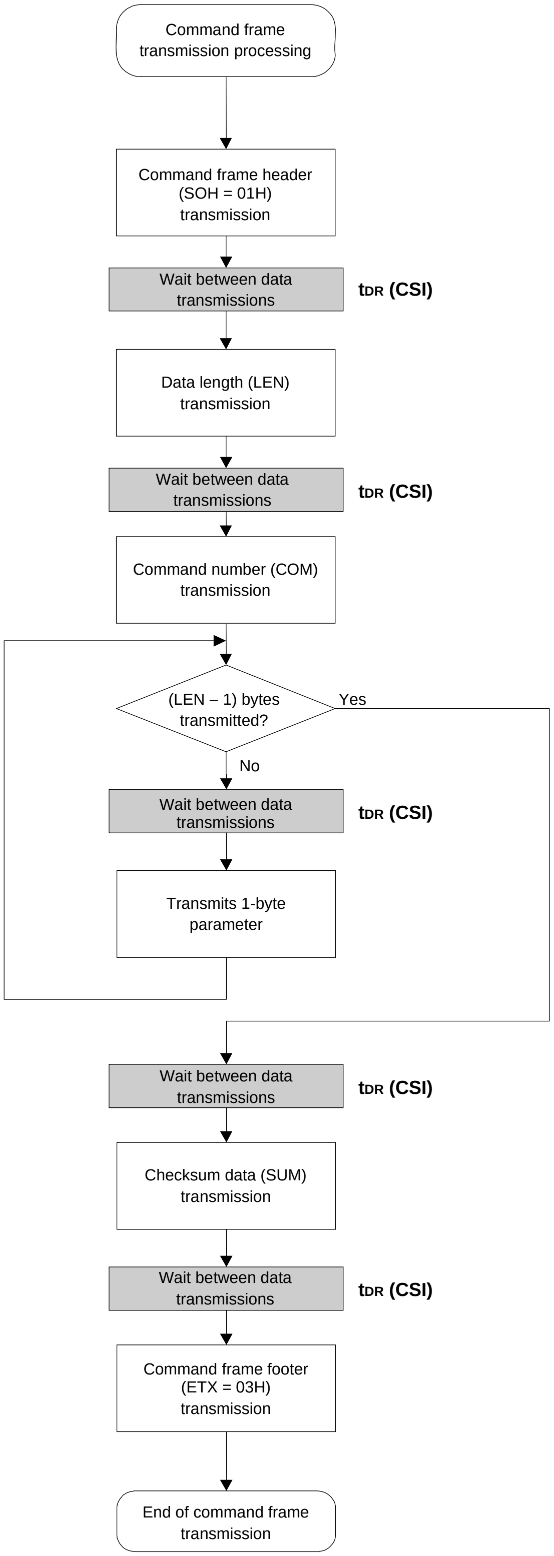
```
     put_dfrm_ua(2, fl_txdata_frm, true);   // send security setting(FLAG) & BOT data


     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]
     }

     /************************************************/
     /*     Check internally verify                  */
     /************************************************/
     rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);      // get status frame
//   switch(rc) {
//
//         case   FLC_NO_ERR:  return rc;     break; // case [A]
//         case   FLC_DFTO_ERR:return rc;     break; // case [C]
//         default:            return rc;     break; // case [B]
//   }
     return rc;
}
```

CHAPTER 5 3-WIRE SERIAL I/O COMMUNICATION MODE (CSI)

Each of the symbol (t_{XX} and t_{WTXX}) shown in the flowchart in this chapter is the symbol of characteristic item in **CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS**.

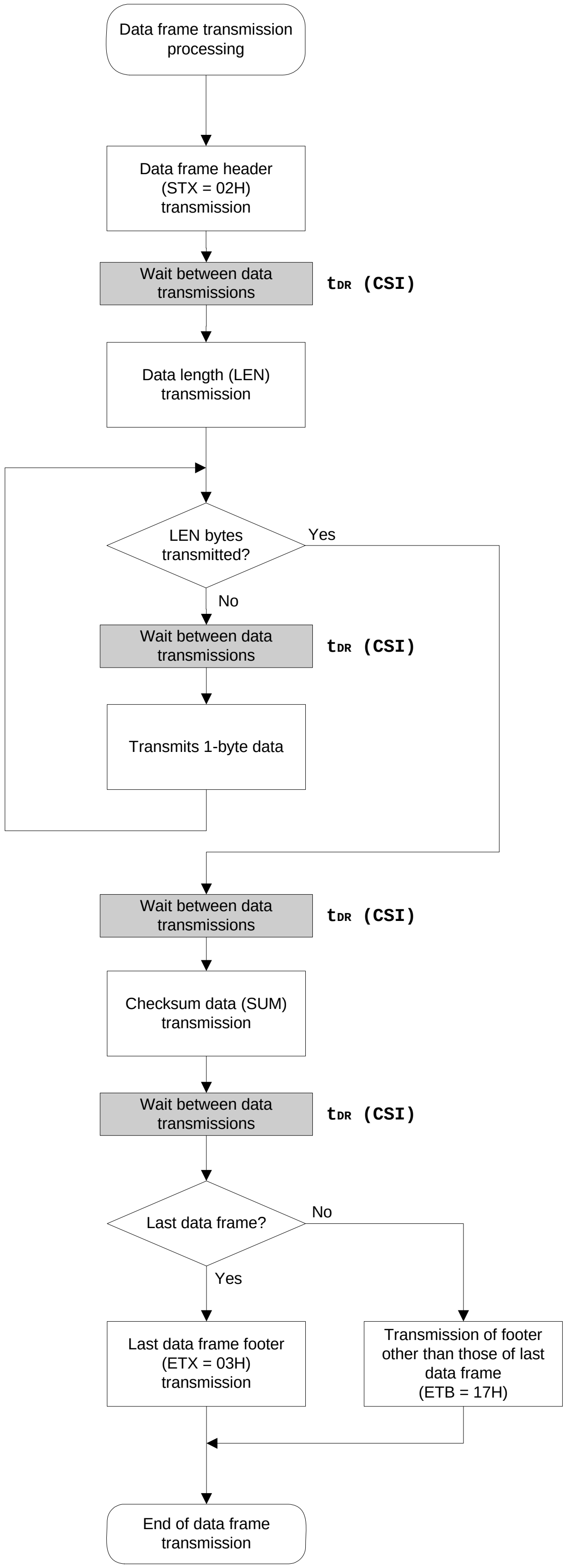
For each specified value, refer to **CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS**.

<R> 5.1 Command Frame Transmission Processing Flowchart

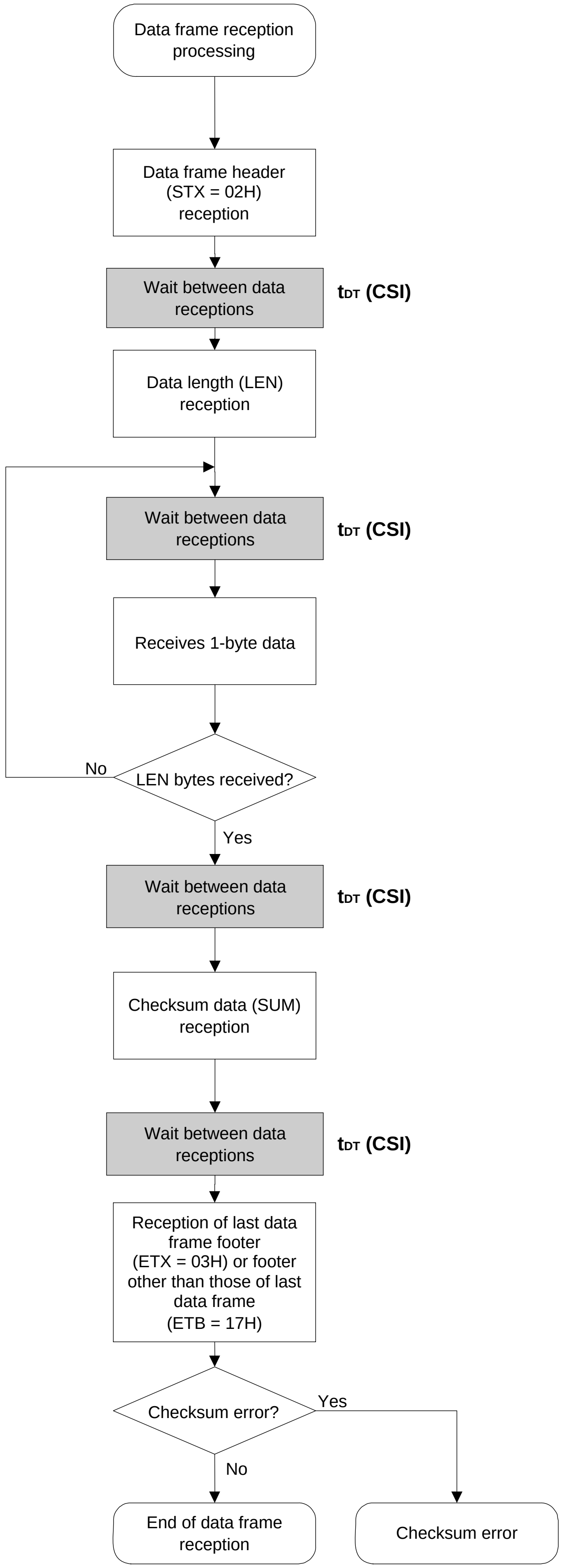

Command frame transmission processing
Command frame header (SOH = 01H) transmission
Wait between data transmissions
tDR (CSI)
Data length (LEN) transmission
Wait between data transmissions
tDR (CSI)
Command number (COM) transmission
(LEN − 1) bytes transmitted?
Yes
No
Wait between data transmissions
tDR (CSI)
Transmits 1-byte parameter
Wait between data transmissions
tDR (CSI)
Checksum data (SUM) transmission
Wait between data transmissions
tDR (CSI)
Command frame footer (ETX = 03H) transmission
End of command frame transmission

<R>

5.2 Data Frame Transmission Processing Flowchart


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

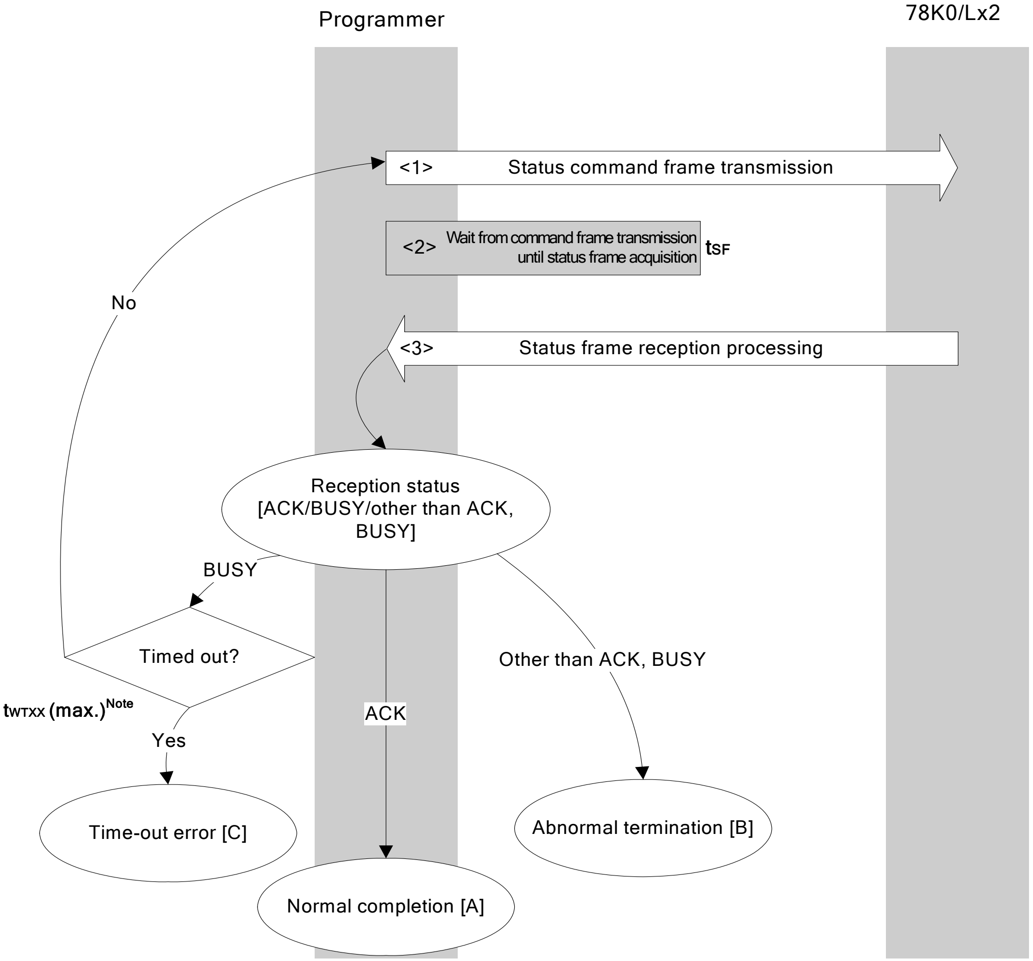
<R> 5.3 Data Frame Reception Processing Flowchart


Data frame reception processing
Data frame header (STX = 02H) reception
Wait between data receptions
tDT (CSI)
Data length (LEN) reception
Wait between data receptions
tDT (CSI)
Receives 1-byte data
No
LEN bytes received?
Yes
Wait between data receptions
tDT (CSI)
Checksum data (SUM) reception
Wait between data receptions
tDT (CSI)
Reception of last data frame footer (ETX = 03H) or footer other than those of last data frame (ETB = 17H)
Checksum error?
Yes
No
End of data frame reception
Checksum error

5.4 Status Command

5.4.1 Processing sequence chart

Status command processing sequence


Programmer
78K0/Lx2
<1> Status command frame transmission
<2> Wait from command frame transmission until status frame acquisition
tSF
<3> Status frame reception processing
Reception status [ACK/BUSY/other than ACK, BUSY]
BUSY
Timed out?
No
tWTXX (max.)Note
Yes
ACK
Other than ACK, BUSY
Time-out error [C]
Normal completion [A]
Abnormal termination [B]

Note Application specifications differ according to execution command.

5.4.2 Description of processing sequence

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

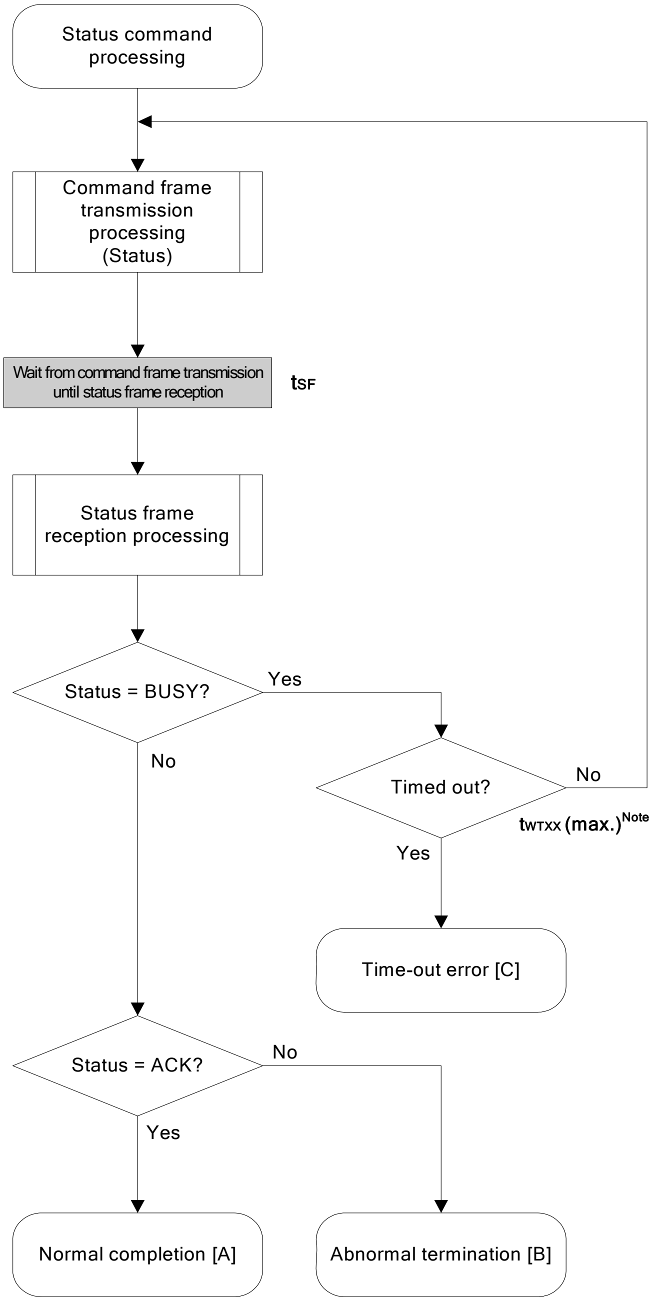
When ST1 = ACK:	Normal completion [A]
When ST1 = BUSY:	A time-out check is performed ($t_{WTXX}(MAX.)$[Note]). If the processing is not timed out, the sequence is re-executed from <1>. If a time-out occurs, a time-out error [C] is returned.
When ST1 ≠ ACK, BUSY:	Abnormal termination [B]

Note Application specifications differ according to execution command.

5.4.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The status frame transmitted from the 78K0/Lx2 has been received normally.
Abnormal termination [B]	Command error	04H	An unsupported command or abnormal frame has been received.
	Parameter error	05H	Command information (parameter) is invalid.
	Checksum error	07H	The data of the frame transmitted from the programmer is abnormal.
	Verify error	0FH	A verify error has occurred for the data of the frame transmitted from the programmer.
	Protect error	10H	An attempt was made to execute processing prohibited by the Security Set command.
	Negative acknowledgment (NACK)	15H	Negative acknowledgment
	Read error	20H	Reading of security information failed.
	MRG10 error	1AH	An erase error has occurred.
	MRG11 error	1BH	An internal verify error has occurred during data write, or a blank check error has occurred.
	Write error	1CH	A write error has occurred.
Time-out error [C]		–	After command transmission, the specified time has elapsed but a BUSY response is still returned.

5.4.4 Flowchart


Status command processing
Command frame transmission processing (Status)
Wait from command frame transmission until status frame reception
tSF
Status frame reception processing
Status = BUSY?
Yes
No
Timed out?
No
twTXX (max.)Note
Yes
Time-out error [C]
Status = ACK?
No
Yes
Normal completion [A]
Abnormal termination [B]

Note Application specifications differ according to execution command.

5.4.5 Sample program

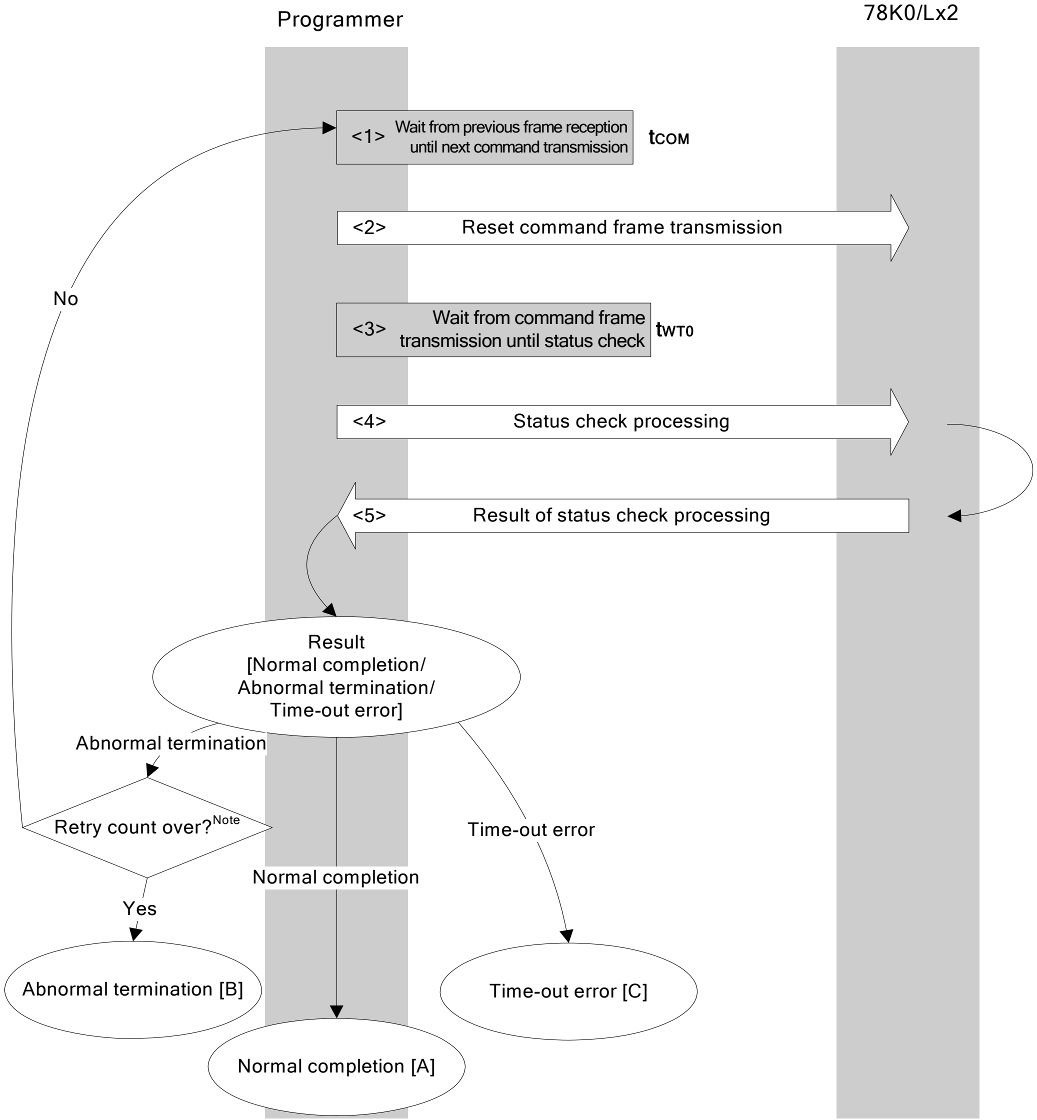
The following shows a sample program for Status command processing.

```
/****************************************************************/
/*                                                              */
/* Get status command (CSI)                                     */
/*                                                              */
/****************************************************************/
/* [r] u16        ... decoded status or error code              */
/*                                                              */
/* (see fl.h/fl-proto.h &                                       */
/*         definition of decode_status() in fl.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
           }
           break;                     // ACK or other error (but BUSY)
    }
    rc = decode_status(fl_st1);      // decode status to return code
//  switch(rc) {
//
//        case  FLC_NO_ERR:  return rc;   break; // case [A]
//        default:           return rc;   break; // case [B]
//  }
    return rc;

}
```

5.5 Reset Command

5.5.1 Processing sequence chart

Reset command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Reset command frame transmission
<3> Wait from command frame transmission until status check
tWT0
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Abnormal termination
Retry count over?Note
No
Yes
Normal completion
Time-out error
Abnormal termination [B]
Normal completion [A]
Time-out error [C]

Note Do not exceed the retry count for the reset command transmission (up to 16 times).

5.5.2 Description of processing sequence

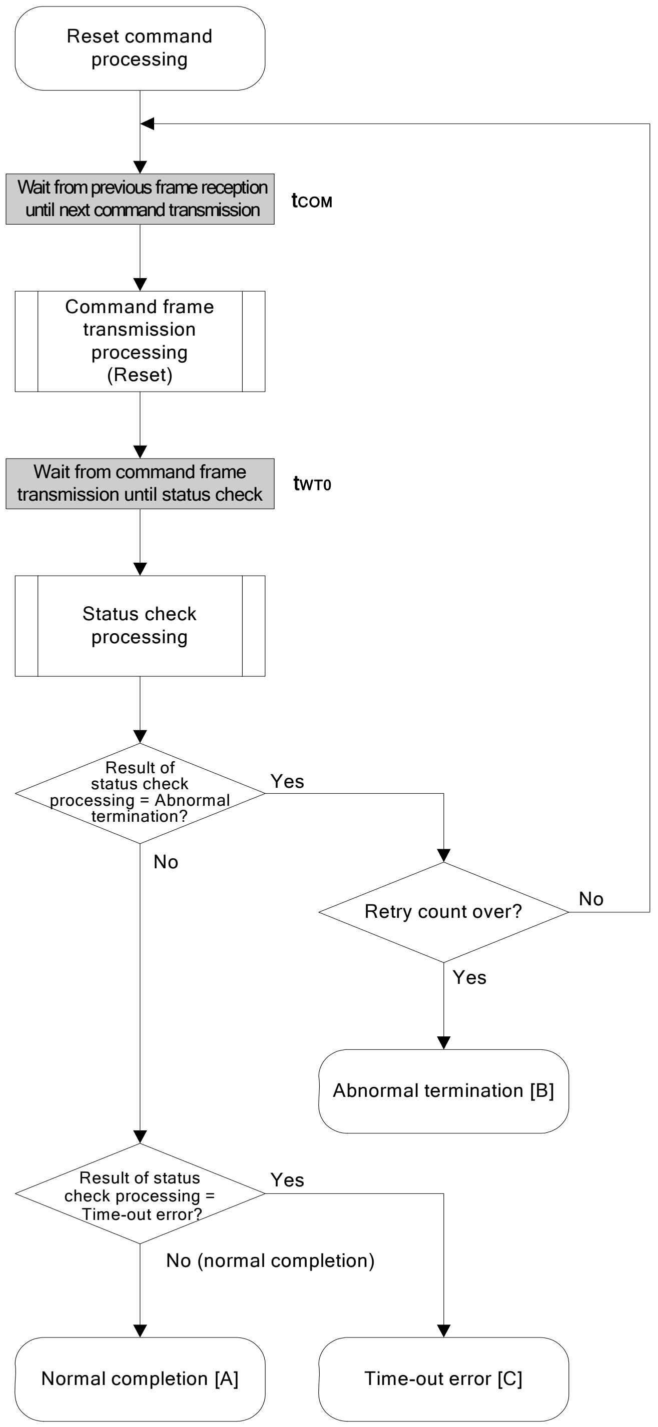
<1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
<2> The Reset command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time t_{WT0}).
<4> The status frame is acquired by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

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

5.5.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and synchronization between the programmer and the 78K0/Lx2 has been established.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Status check processing terminated with time-out.

5.5.4 Flowchart


Reset command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Reset)
Wait from command frame transmission until status check
tWT0
Status check processing
Result of status check processing = Abnormal termination?
Yes
No
Retry count over?
No
Yes
Abnormal termination [B]
Result of status check processing = Time-out error?
Yes
No (normal completion)
Normal completion [A]
Time-out error [C]

5.5.5 Sample program

The following shows a sample program for Reset command processing.

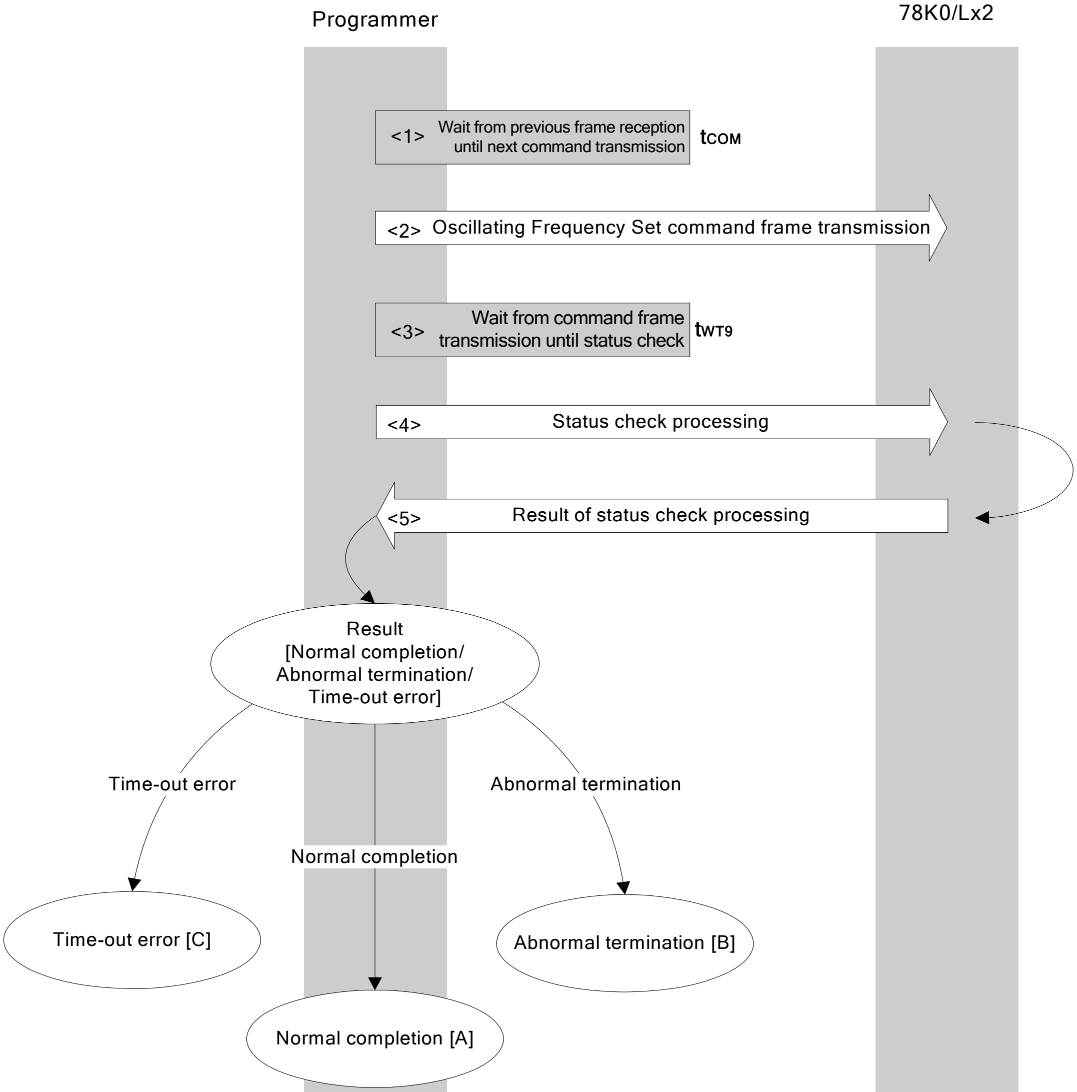
```
/****************************************************************/
/*                                                              */
/* Reset command (CSI)                                          */
/*                                                              */
/****************************************************************/
/* [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_TO);  // get status

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

5.6 Oscillating Frequency Set Command

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

5.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Oscillating Frequency Set command frame transmission
<3> Wait from command frame transmission until status check
tWT9
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Normal completion
Abnormal termination
Time-out error [C]
Abnormal termination [B]
Normal completion [A]

5.6.2 Description of processing sequence

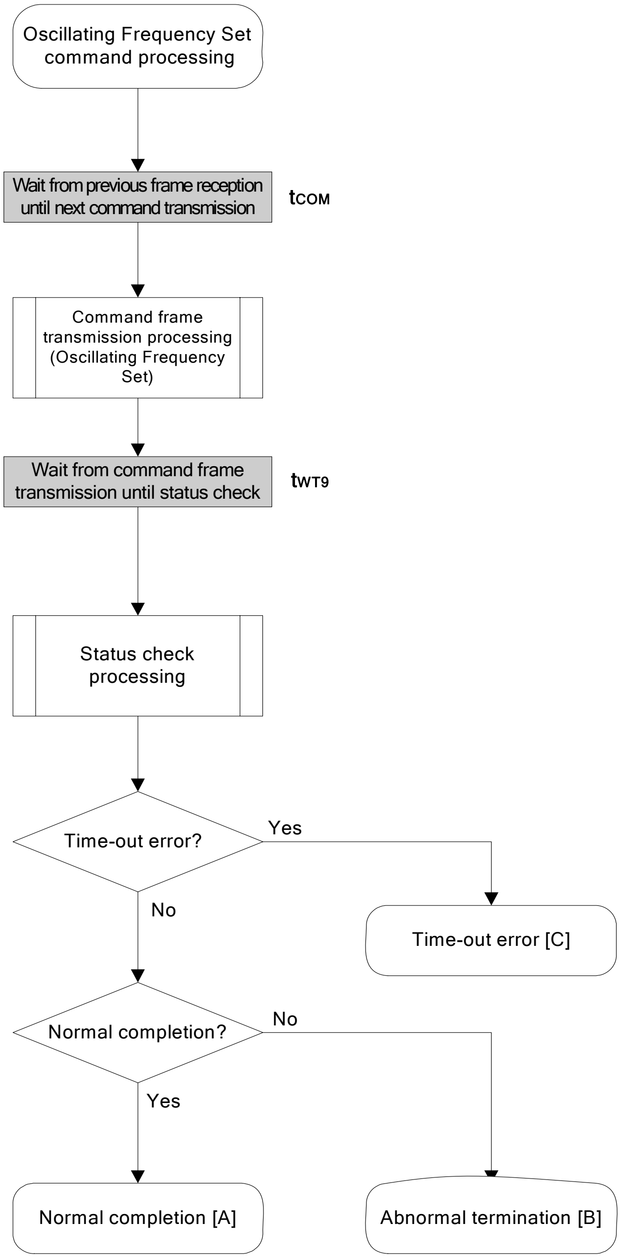
<1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
<2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time t_{WT9}).
<4> The status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

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

5.6.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the operating frequency was correctly set to the 78K0/Lx2.
Abnormal termination [B]	Parameter error	05H	The oscillation frequency value is out of range.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.

5.6.4 Flowchart


Oscillating Frequency Set command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Oscillating Frequency Set)
Wait from command frame transmission until status check
tWT9
Status check processing
Time-out error?
Yes
No
Time-out error [C]
Normal completion?
No
Yes
Normal completion [A]
Abnormal termination [B]

5.6.5 Sample program

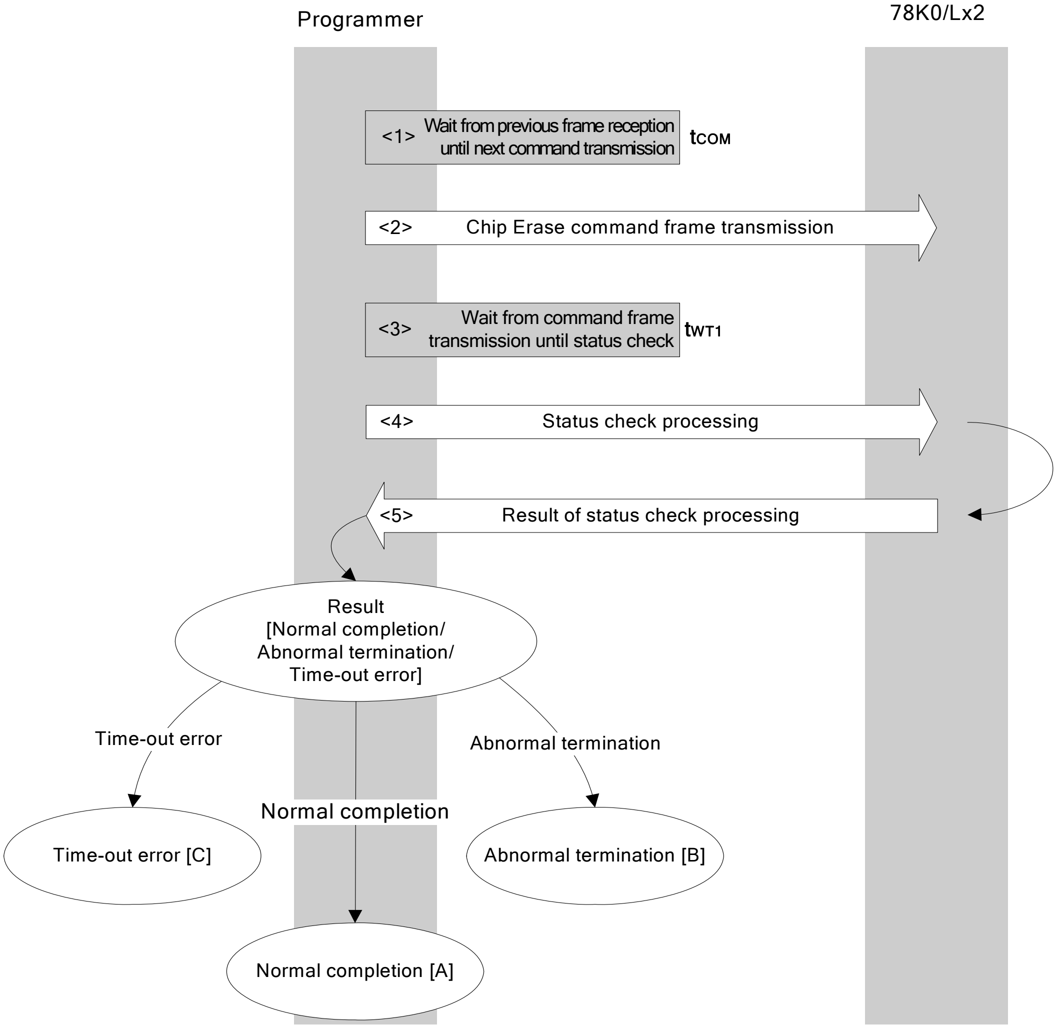
The following shows a sample program for Oscillating Frequency Set command processing.

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

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


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

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

    fl_wait(tWT9);

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

5.7 Chip Erase Command

5.7.1 Processing sequence chart

Chip Erase command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Chip Erase command frame transmission
<3> Wait from command frame transmission until status check
tWT1
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Abnormal termination
Normal completion
Time-out error [C]
Abnormal termination [B]
Normal completion [A]

5.7.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time t_{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_{WT1}).

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

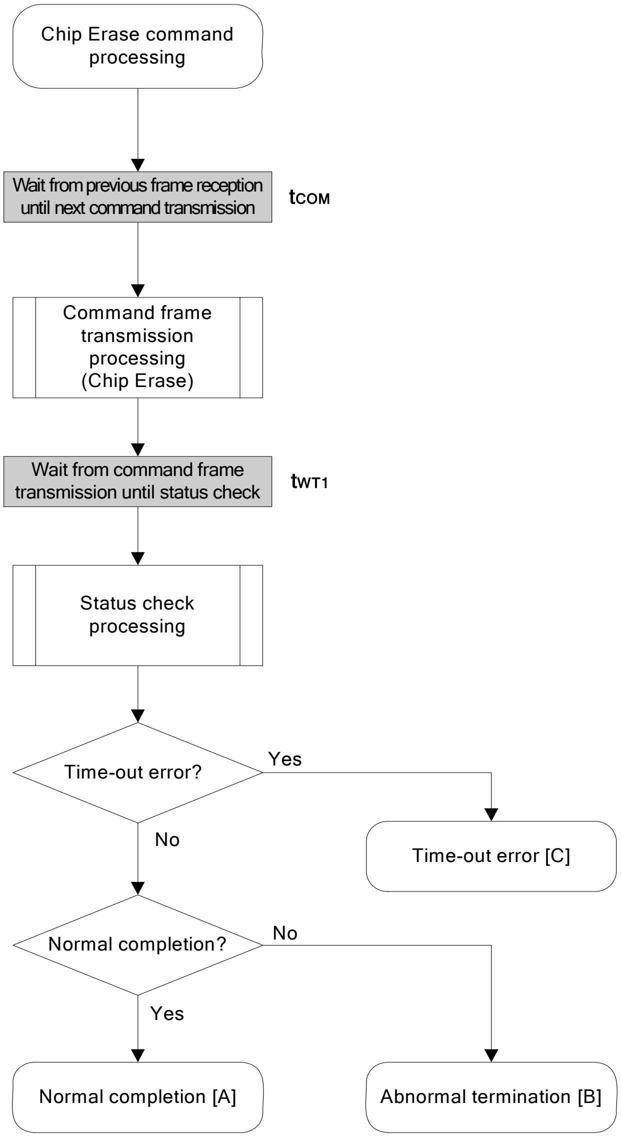
<5> The following processing is performed according to the result of status check processing.

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

5.7.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and chip erase was performed normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Protect error	10H	Chip Erase command is prohibited by 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.
Time-out error [C]		–	The status frame was not received within the specified time.

5.7.4 Flowchart


Chip Erase command processing
Wait from previous frame reception until next command transmission
t_{COM}
Command frame transmission processing (Chip Erase)
Wait from command frame transmission until status check
t_{WT1}
Status check processing
Time-out error?
Yes
No
Time-out error [C]
Normal completion?
No
Yes
Normal completion [A]
Abnormal termination [B]

5.7.5 Sample program

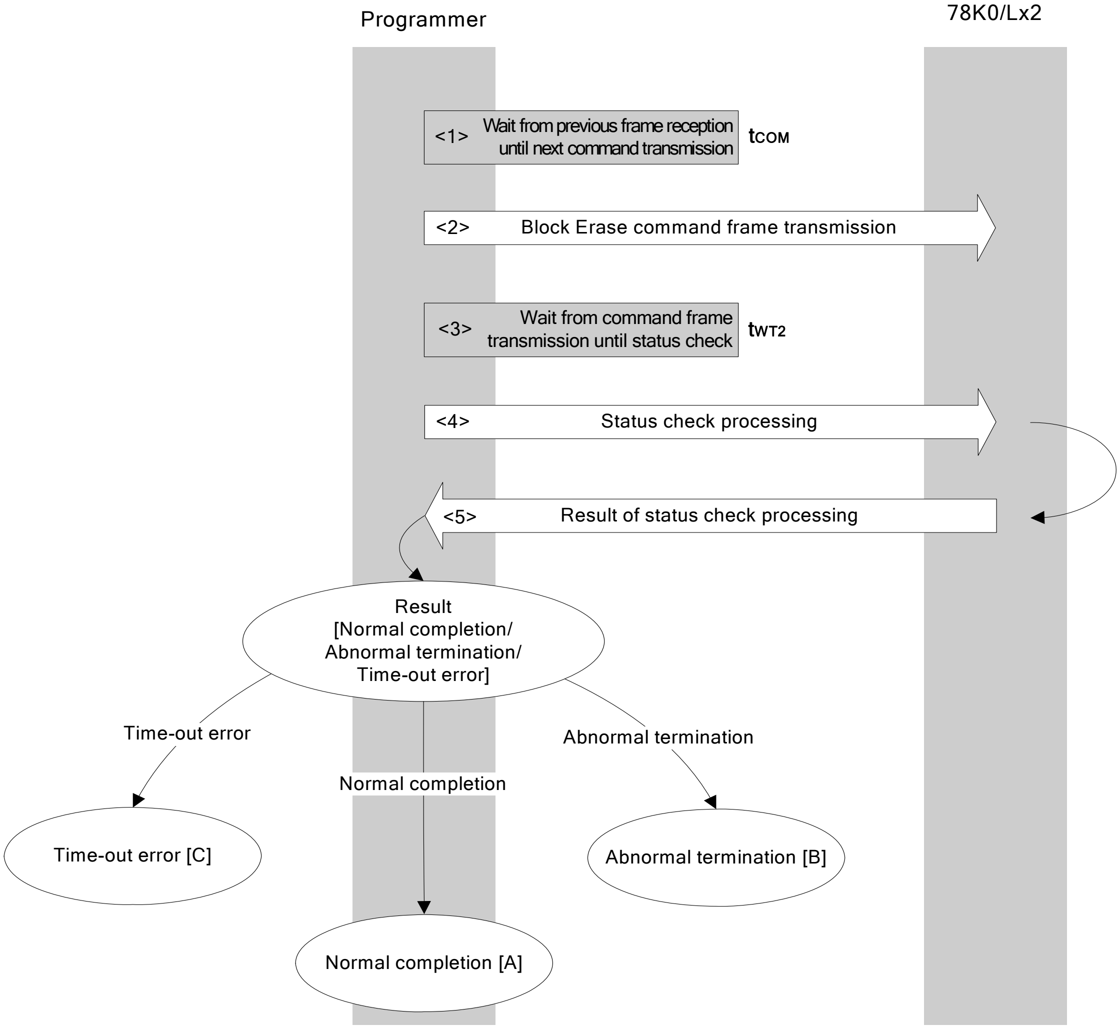
The following shows a sample program for Chip Erase command processing.

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


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

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

    fl_wait(tWT1);

    rc = fl_csi_getstatus(tWT1_MAX);        // get status frame
//  switch(rc) {
//
//         case   FLC_NO_ERR:  return rc;    break; // case [A]
//         case   FLC_DFTO_ERR:return rc;    break; // case [C]
//         default:            return rc;    break; // case [B]
//  }
    return rc;

}
```

5.8 Block Erase Command

5.8.1 Processing sequence chart

Block Erase command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Block Erase command frame transmission
<3> Wait from command frame transmission until status check
tWT2
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Normal completion
Abnormal termination
Time-out error [C]
Normal completion [A]
Abnormal termination [B]

5.8.2 Description of processing sequence

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

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

<3> Waits until status frame acquisition (wait time t_{WT2}).

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

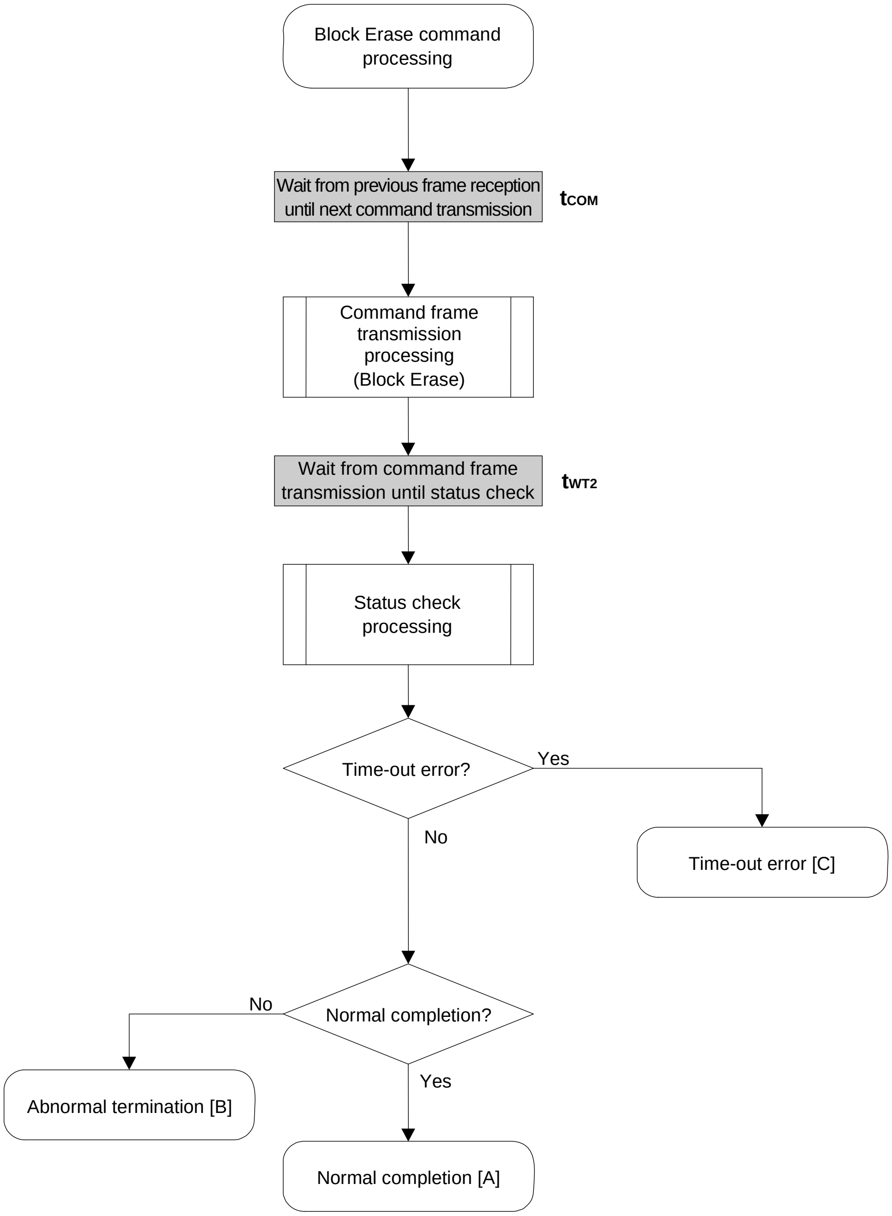
<5> The following processing is performed according to the result of status check processing.

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

5.8.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and block erase was performed normally.
Abnormal termination [B]	Parameter error	05H	The number of blocks is out of range.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Protect error	10H	Block Erase command is prohibited by 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.
Time-out error [C]		–	The status frame was not received within the specified time.

5.8.4 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_{WT2}
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Normal completion [A]

5.8.5 Sample program

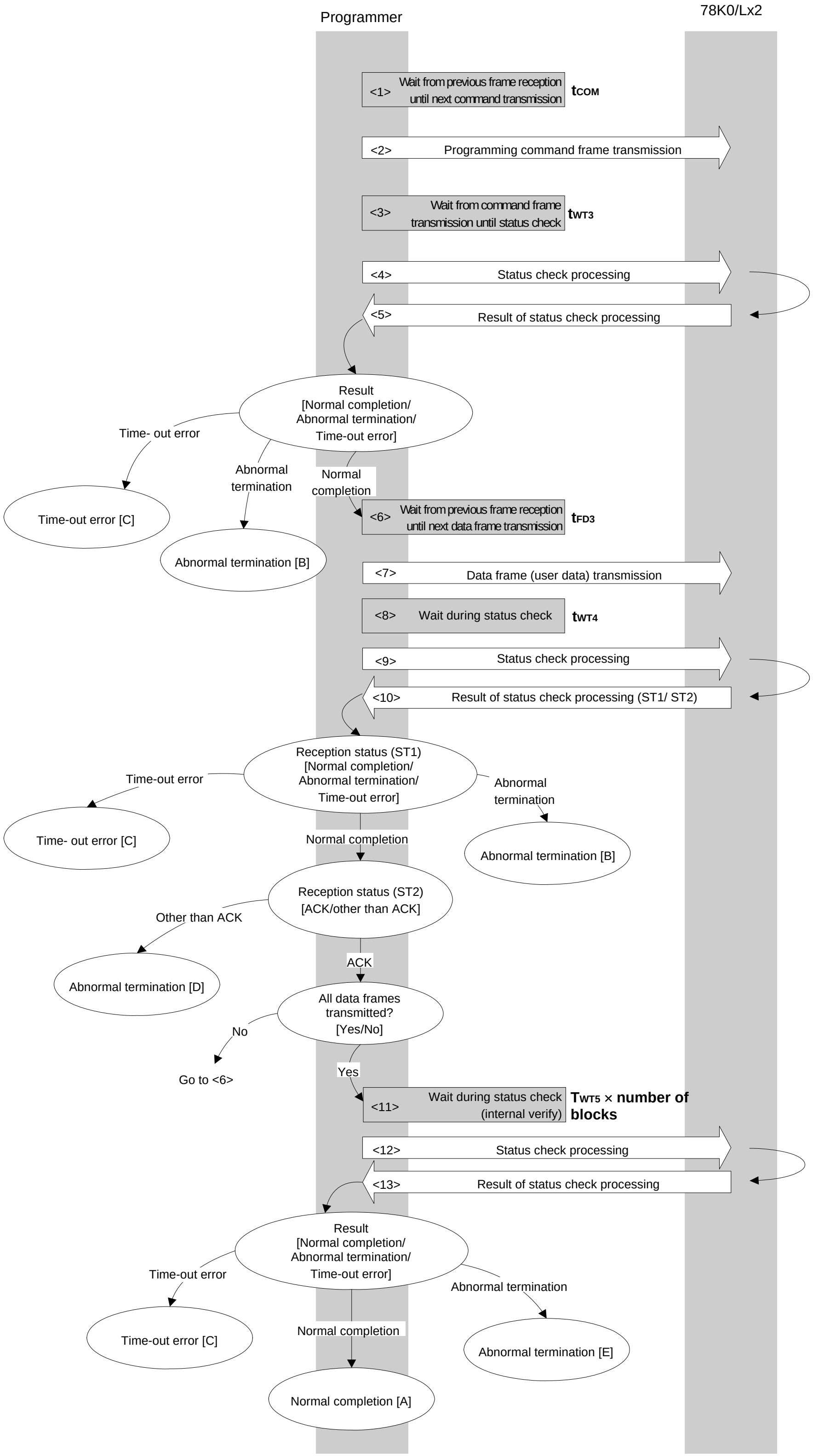
The following shows a sample program for Block Erase command processing.

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

    u16    rc;
    u32    wt2, wt2_max;
    u32    top, bottom;

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

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

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

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

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

    fl_wait(wt2);


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

5.9 Programming Command

5.9.1 Processing sequence chart

Programming command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Programming command frame transmission
<3> Wait from command frame transmission until status check
tWT3
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time- out error
Time-out error [C]
Abnormal termination
Abnormal termination [B]
Normal completion
<6> Wait from previous frame reception until next data frame transmission
tFD3
<7> Data frame (user data) transmission
<8> Wait during status check
tWT4
<9> Status check processing
<10> Result of status check processing (ST1/ ST2)
Reception status (ST1) [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Time- out error [C]
Abnormal termination
Abnormal termination [B]
Normal completion
Reception status (ST2) [ACK/other than ACK]
Other than ACK
Abnormal termination [D]
ACK
All data frames transmitted? [Yes/No]
No
Go to <6>
Yes
<11> Wait during status check (internal verify)
TWT5 × number of blocks
<12> Status check processing
<13> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Time-out error [C]
Abnormal termination
Abnormal termination [E]
Normal completion
Normal completion [A]

5.9.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
<2> The Programming command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time t_{WT3}).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

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

<6> Waits until the next data frame transmission (wait time t_{FD3}).
<7> User data to be written to the 78K0/Lx2 flash memory is transmitted by data frame transmission processing.
<8> Waits from data frame (user data) transmission until status check processing (wait time t_{WT4}).
<9> The status frame is acquired by status check processing.
<10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

When ST1 = abnormal termination: Abnormal termination [B]
When ST1 = time-out error: A time-out error [C] is returned.
When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 ≠ ACK: Abnormal termination [D]
- When ST2 = ACK: Proceeds to <11> when transmission of all of the user data is completed. If there still remain user data to be transmitted, the processing re-executes the sequence from <6>.

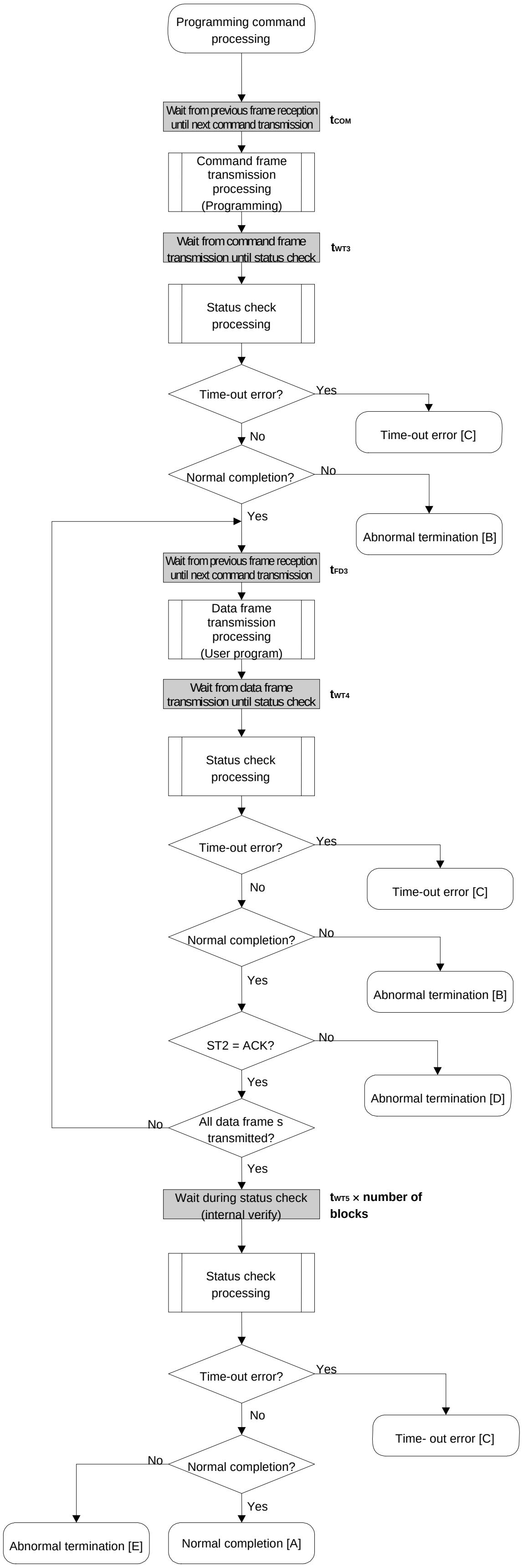
<11> Waits until status check processing (time-out time t_{WT5} × number of blocks).
<12> The status frame is acquired by status check processing.
<13> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A] (Indicating that the internal verify check has performed normally after completion of write)
When the processing ends abnormally: Abnormal termination [E] (Indicating that the internal verify check has not performed normally after completion of write)
When a time-out error occurs: A time-out error [C] is returned.

5.9.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the user data was written normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or is not a multiple of 8.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Protect error	10H	Programming command is prohibited by 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]	Write error	1CH (ST2)	A write error has occurred.
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

5.9.4 Flowchart


Programming command processing
Wait from previous frame reception until next command transmission
t_{COM}
Command frame transmission processing (Programming)
Wait from command frame transmission until status check
t_{WT3}
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Wait from previous frame reception until next command transmission
t_{FD3}
Data frame transmission processing (User program)
Wait from data frame transmission until status check
t_{WT4}
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
ST2 = ACK?
No
Abnormal termination [D]
Yes
All data frame s transmitted?
No
Yes
Wait during status check (internal verify)
t_{WT5} × number of blocks
Status check processing
Time-out error?
Yes
Time- out error [C]
No
Normal completion?
No
Abnormal termination [E]
Yes
Normal completion [A]

5.9.5 Sample program

The following shows a sample program for Programming command processing.

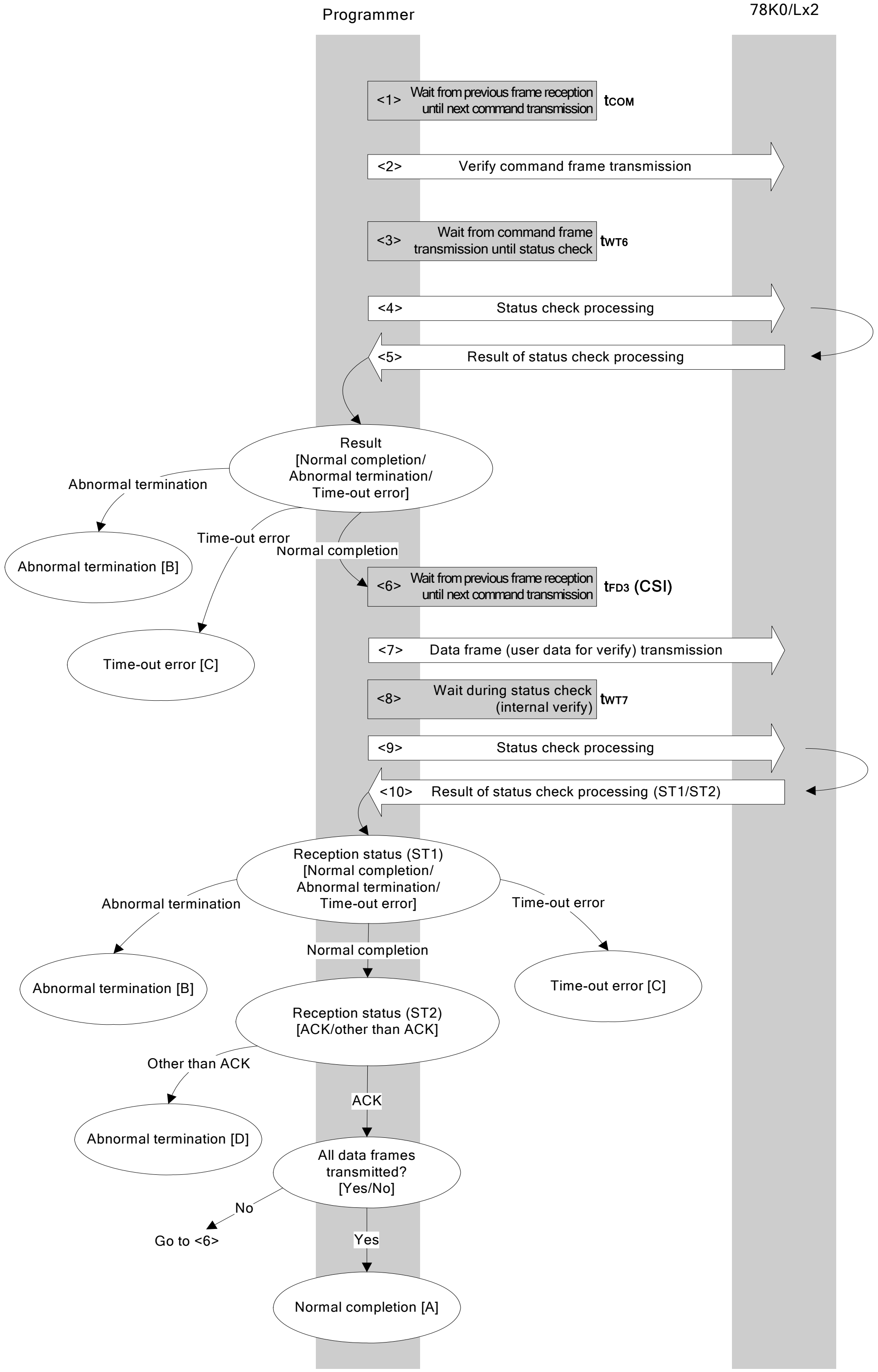
```
/****************************************************************/
/*                                                              */
/* Write command (CSI)                                          */
/*                                                              */
/****************************************************************/
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
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_TO);                // get status frame
    switch(rc) {
           case   FLC_NO_ERR:               break; // continue
    //     case   FLC_DFTO_ERR:return rc;   break; // case [C]
           default:            return rc;   break; // case [B]
    }

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

    while(1){

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

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

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


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

}
```

5.10 Verify Command

5.10.1 Processing sequence chart

Verify command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Verify command frame transmission
<3> Wait from command frame transmission until status check
tWT6
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Abnormal termination
Abnormal termination [B]
Time-out error
Time-out error [C]
Normal completion
<6> Wait from previous frame reception until next command transmission
tFD3 (CSI)
<7> Data frame (user data for verify) transmission
<8> Wait during status check (internal verify)
tWT7
<9> Status check processing
<10> Result of status check processing (ST1/ST2)
Reception status (ST1) [Normal completion/ Abnormal termination/ Time-out error]
Abnormal termination
Abnormal termination [B]
Time-out error
Time-out error [C]
Normal completion
Reception status (ST2) [ACK/other than ACK]
Other than ACK
Abnormal termination [D]
ACK
All data frames transmitted? [Yes/No]
No
Go to <6>
Yes
Normal completion [A]

5.10.2 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}).
<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}).
<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).

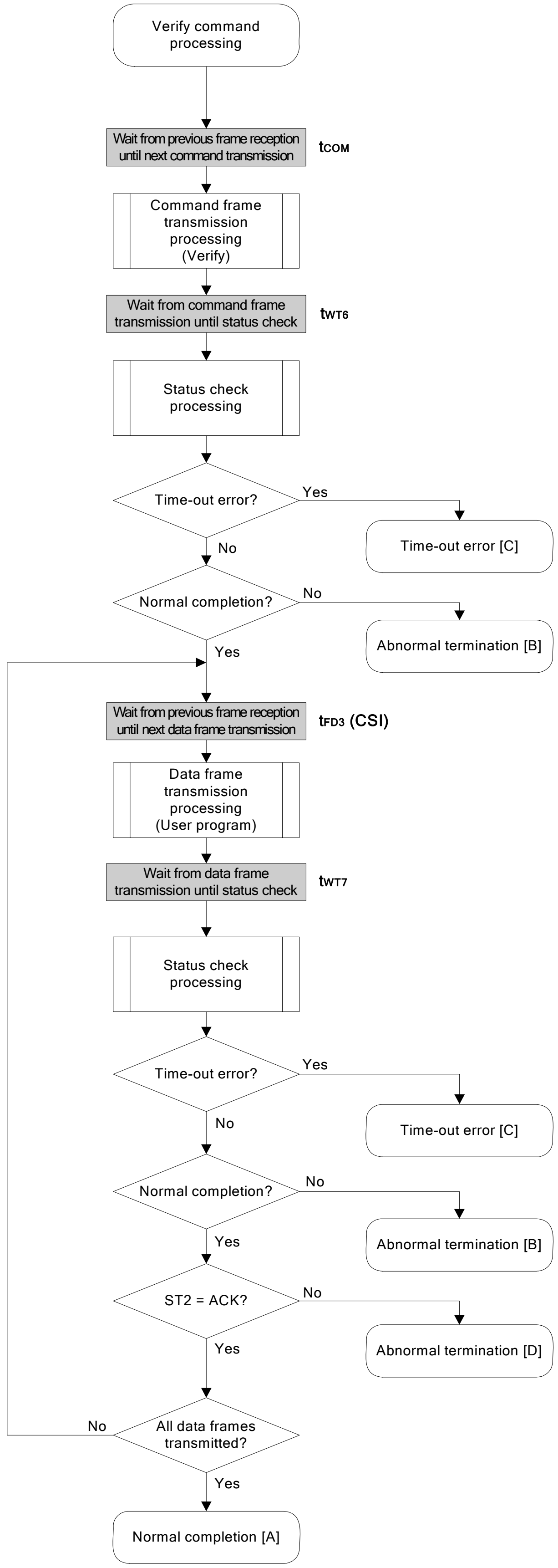
When ST1 = abnormal termination: Abnormal termination [B]
When ST1 = time-out error: A time-out error [C] is returned.
When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 ≠ ACK: Abnormal termination [D]
- When ST2 = ACK: If transmission of all data frames is completed, the processing ends normally [A].
  If there still remain data frames to be transmitted, the processing re-executes the sequence from <6>.

5.10.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the verify was completed normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame or data frame is abnormal.
	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]	Verify error	0FH (ST2)	The verify has failed, or another error has occurred.

5.10.4 Flowchart


Verify command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Verify)
Wait from command frame transmission until status check
tWT6
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Wait from previous frame reception until next data frame transmission
tFD3 (CSI)
Data frame transmission processing (User program)
Wait from data frame transmission until status check
tWT7
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
ST2 = ACK?
No
Abnormal termination [D]
Yes
No
All data frames transmitted?
Yes
Normal completion [A]

5.10.5 Sample program

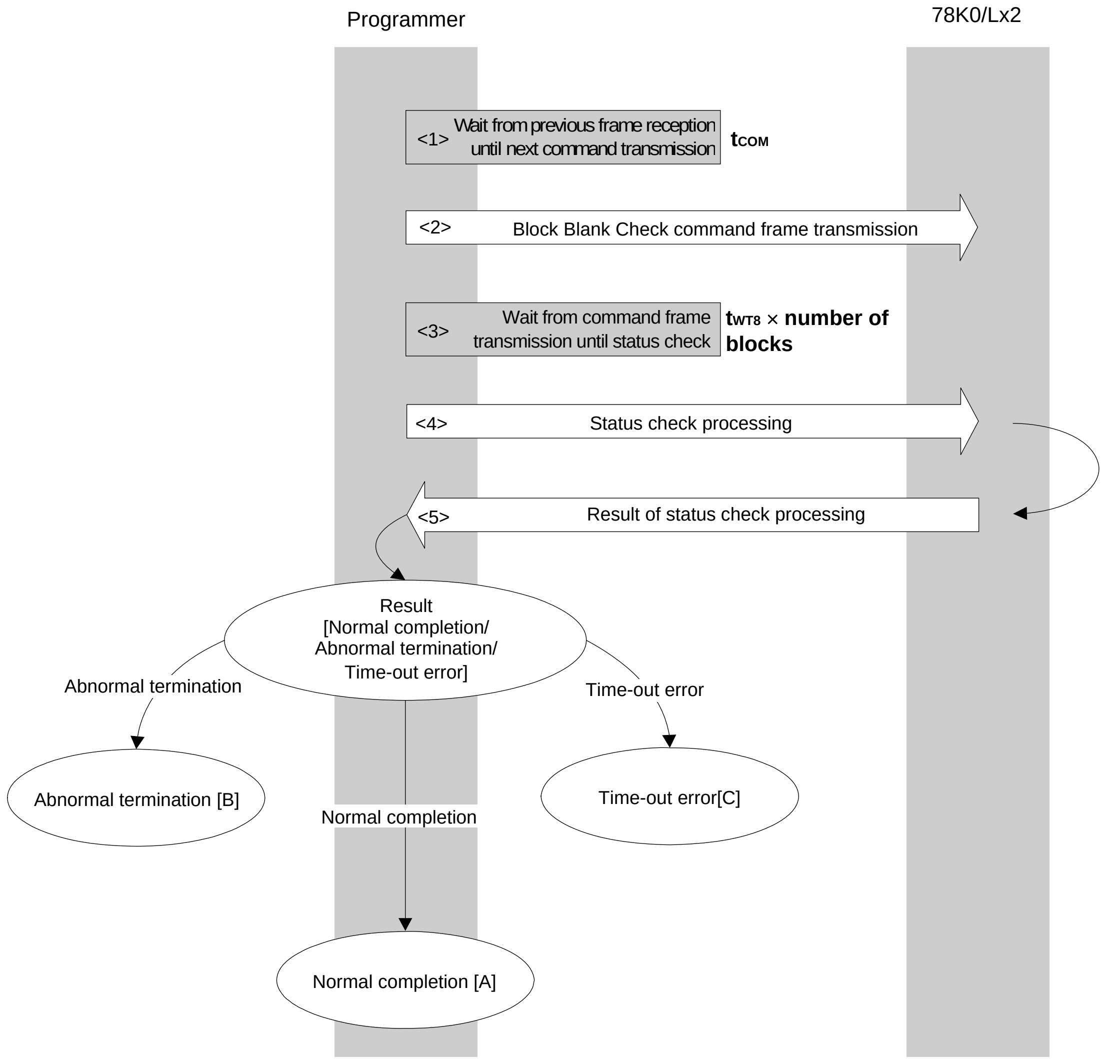
The following shows a sample program for Verify command processing.

```
/***************************************************************/
/*                                                             */
/* Verify command (CSI)                                        */
/*                                                             */
/***************************************************************/
/* [i] u32 top     ... start address                           */
/* [i] u32 bottom  ... end address                             */
/* [r] u16         ... error code                              */
/***************************************************************/
u16        fl_csi_verify(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;

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


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

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


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

    while(1){

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

           }

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

           fl_wait(tFD3_CSI);                      // wait before sending data frame
           put_dfrm_csi(send_size, fl_txdata_frm, is_end);       // send data frame
           fl_wait(tWT7);                          // wait

           rc = fl_csi_getstatus(tWT7_TO);         // get status frame
           switch(rc) {
                  case   FLC_NO_ERR:               break; // continue
           //     case   FLC_DFTO_ERR: return rc;  break; // case [C]
                  default:             return rc;  break; // case [B]
           }
           if (fl_st2 != FLST_ACK){          // ST2 = ACK ?
                  rc = decode_status(fl_st2);      // No
                  return rc;                 // case [D]
           }

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

     }
     return FLC_NO_ERR;  // case [A]

}
```

5.11 Block Blank Check Command

5.11.1 Processing sequence chart

Block Blank Check command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Block Blank Check command frame transmission
<3> Wait from command frame transmission until status check
tWT8 × number of blocks
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Abnormal termination
Time-out error
Abnormal termination [B]
Time-out error[C]
Normal completion
Normal completion [A]

5.11.2 Description of processing sequence

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

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

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

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

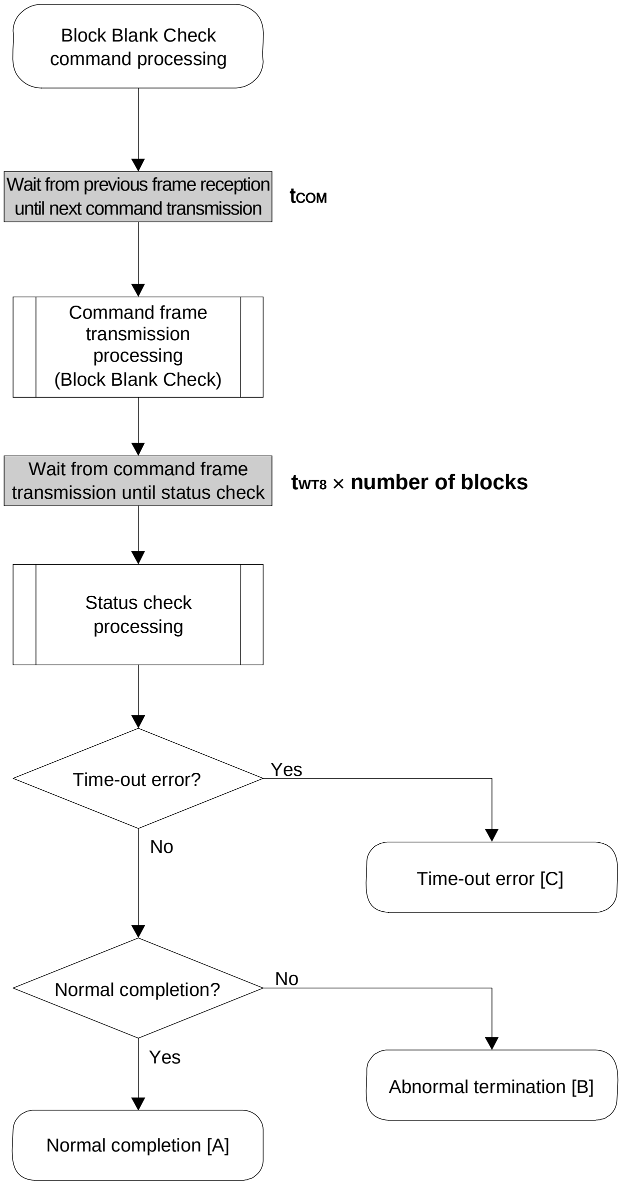
<5> The following processing is performed according to the result of status check processing.

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

5.11.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and all of the specified blocks are blank.
Abnormal termination [B]	Parameter error	05H	The number of blocks is out of range.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.

5.11.4 Flowchart


Block Blank Check command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Block Blank Check)
Wait from command frame transmission until status check
tWT8 × number of blocks
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Normal completion [A]

5.11.5 Sample program

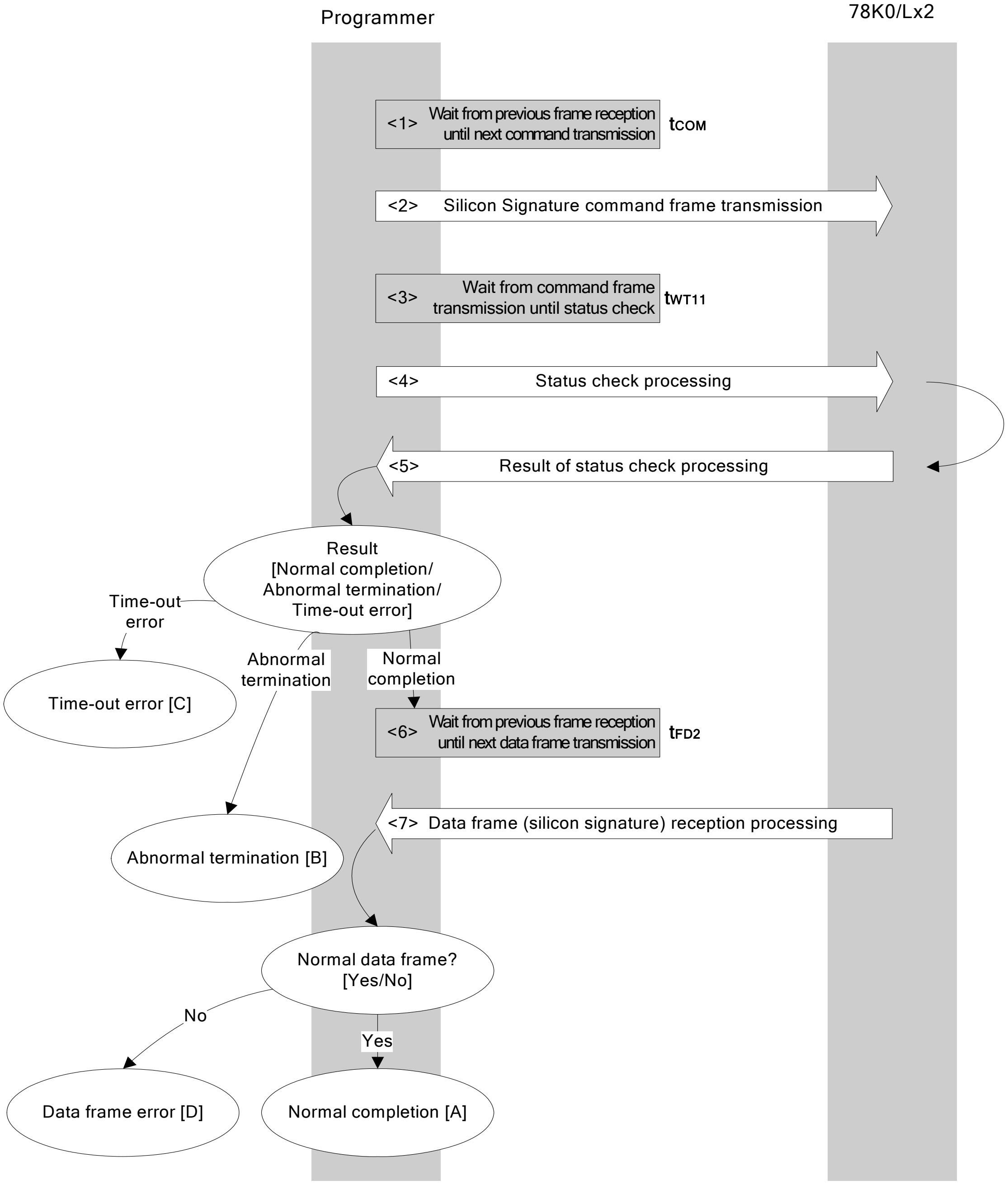
The following shows a sample program for Block Blank Check command processing.

```
/****************************************************************/
/*                                                              */
/* Block blank check command (CSI)                              */
/*                                                              */
/****************************************************************/
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_csi_blk_blank_chk(u32 top, u32 bottom)
{
    u16    rc;
    u16    block_num;

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

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

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

    fl_wait(tWT8 * block_num);

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

5.12 Silicon Signature Command

5.12.1 Processing sequence chart

Silicon Signature command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Silicon Signature command frame transmission
<3> Wait from command frame transmission until status check
tWT11
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Time-out error [C]
Abnormal termination
Normal completion
<6> Wait from previous frame reception until next data frame transmission
tFD2
<7> Data frame (silicon signature) reception processing
Abnormal termination [B]
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

5.12.2 Description of processing sequence

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

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

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

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

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

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

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

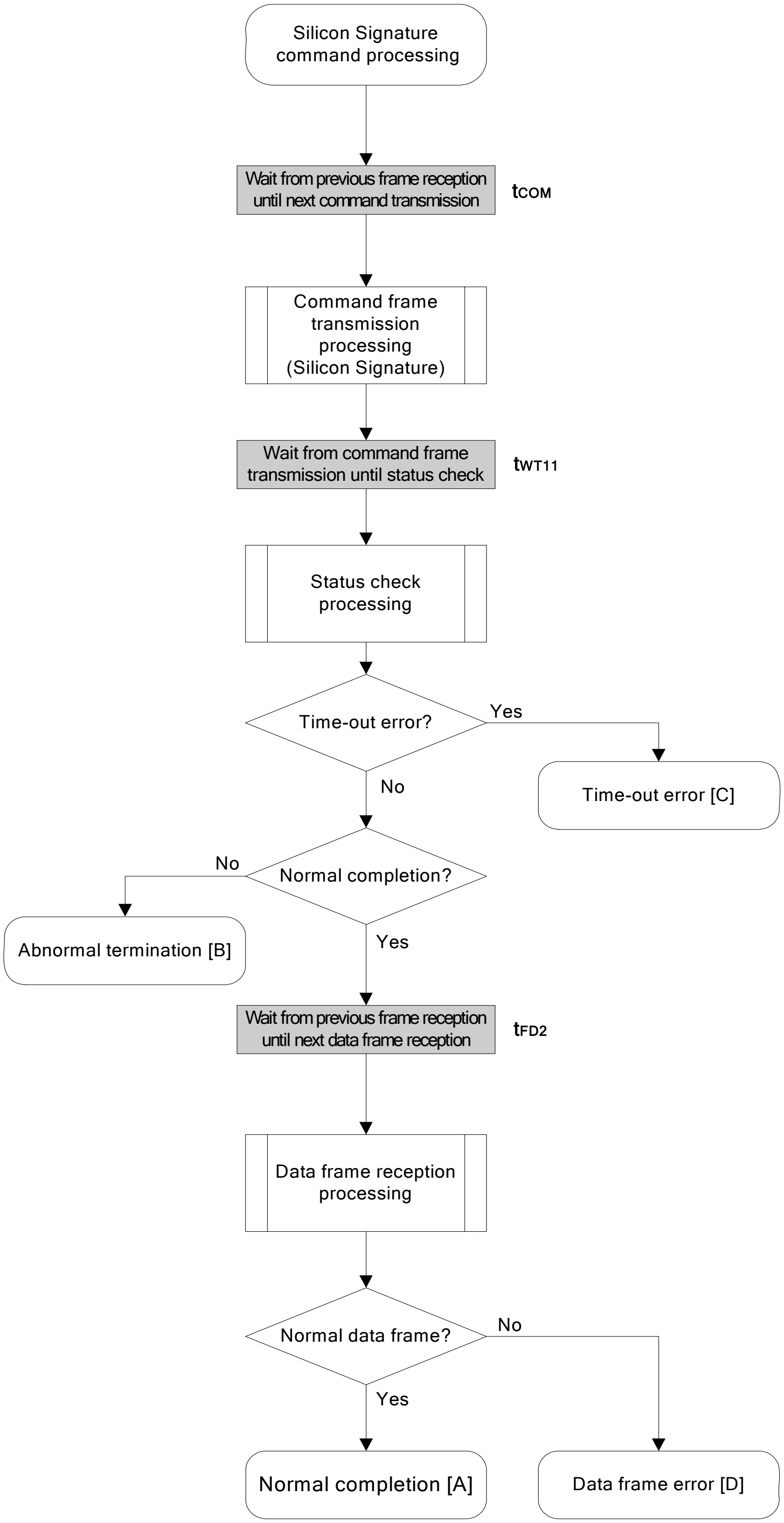
<7> The received data frame (silicon signature data) is checked.

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

5.12.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the silicon signature was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Read error	20H	Reading of security information failed.
Time-out error [C]		–	The status frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as silicon signature data is abnormal.

5.12.4 Flowchart


Silicon Signature command processing
Wait from previous frame reception until next command transmission
t_{COM}
Command frame transmission processing (Silicon Signature)
Wait from command frame transmission until status check
t_{WT11}
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Wait from previous frame reception until next data frame reception
t_{FD2}
Data frame reception processing
Normal data frame?
No
Yes
Normal completion [A]
Data frame error [D]

5.12.5 Sample program

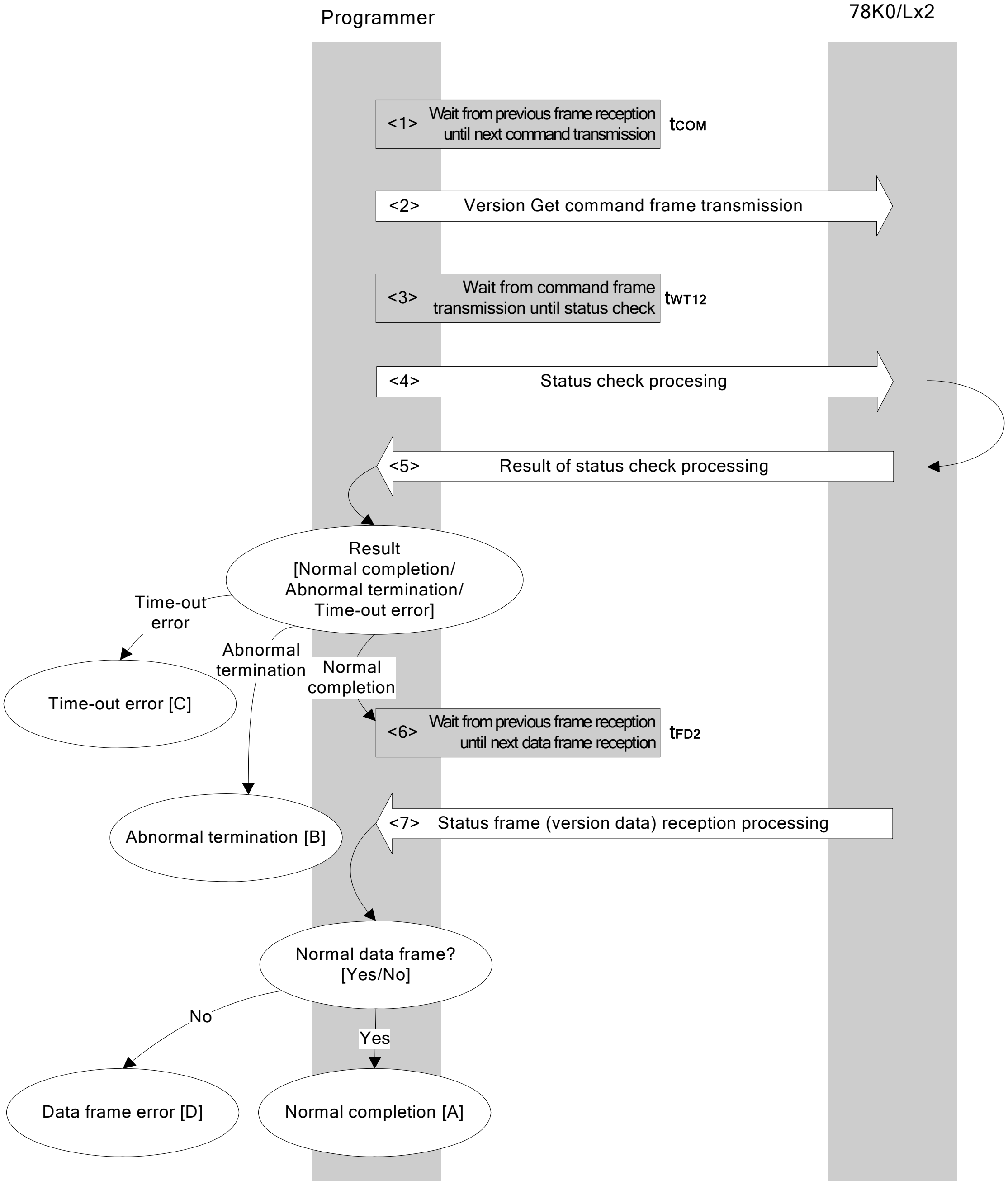
The following shows a sample program for Silicon Signature command processing.

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

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

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

    fl_wait(tWT11);

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

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

    rc = get_dfrm_csi(fl_rxdata_frm); // get data frame (signature data)

    if (rc){                                                  // if no error,
           return rc;                 // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                              // copy Signature data
    return rc;                        // case [A]
}
```

5.13 Version Get Command

5.13.1 Processing sequence chart

Version Get command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Version Get command frame transmission
<3> Wait from command frame transmission until status check
tWT12
<4> Status check procesing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Abnormal termination
Normal completion
Time-out error [C]
<6> Wait from previous frame reception until next data frame reception
tFD2
Abnormal termination [B]
<7> Status frame (version data) reception processing
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

5.13.2 Description of processing sequence

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

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

<3> Waits from command transmission until status check processing (wait time t_{WT12}).

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

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

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

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

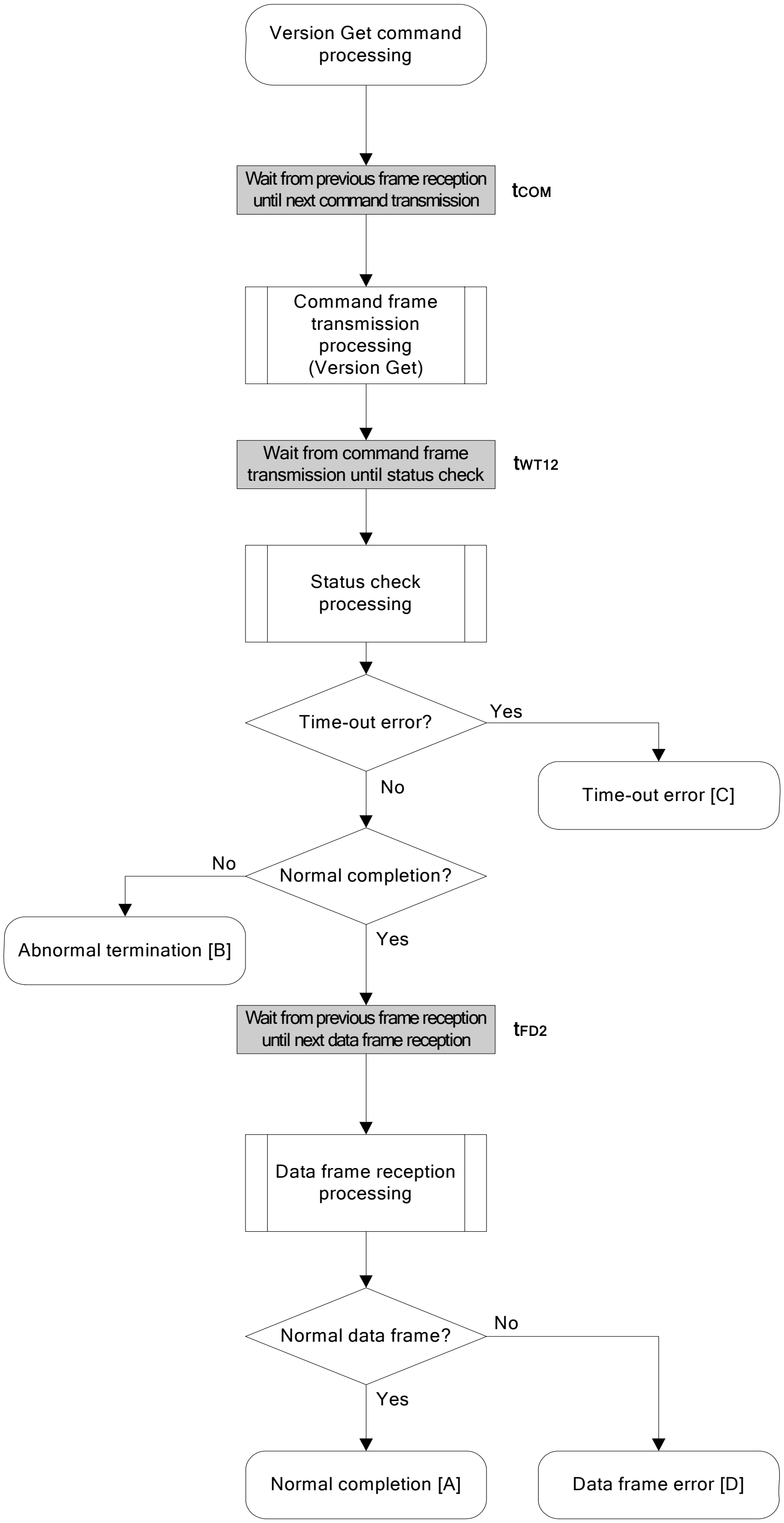
<7> The received data frame (version data) is checked.

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

5.13.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and version data was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.
Data frame error [D]		–	The checksum of the data frame received as version data is abnormal.

5.13.4 Flowchart


Version Get command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Version Get)
Wait from command frame transmission until status check
tWT12
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Wait from previous frame reception until next data frame reception
tFD2
Data frame reception processing
Normal data frame?
No
Yes
Normal completion [A]
Data frame error [D]

5.13.5 Sample program

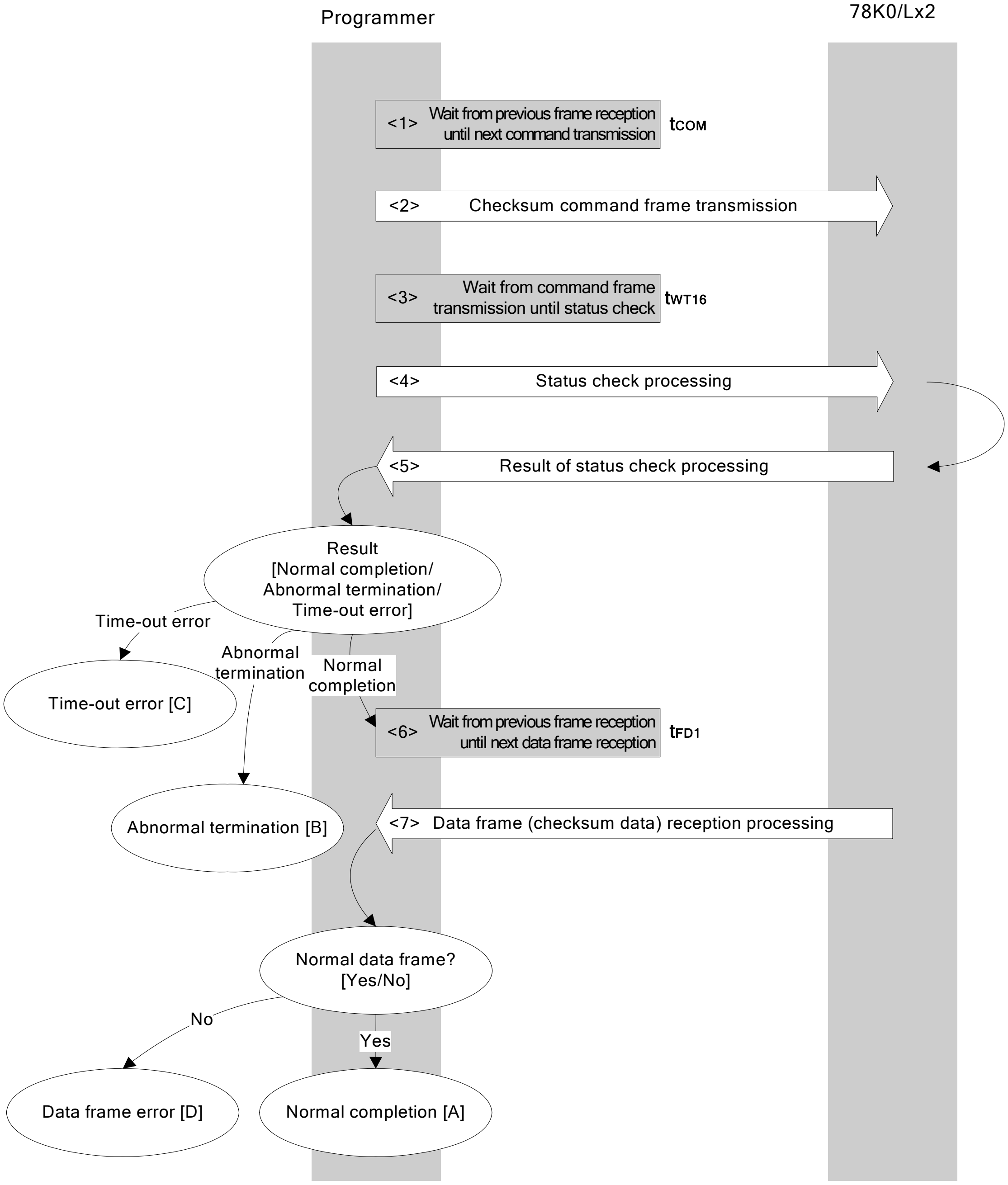
The following shows a sample program for Version Get command processing.

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


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

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

    fl_wait(tWT12);

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

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

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

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

5.14 Checksum Command

5.14.1 Processing sequence chart

Checksum command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Checksum command frame transmission
<3> Wait from command frame transmission until status check
tWT16
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Abnormal termination
Normal completion
Time-out error [C]
<6> Wait from previous frame reception until next data frame reception
tFD1
Abnormal termination [B]
<7> Data frame (checksum data) reception processing
Normal data frame? [Yes/No]
No
Yes
Data frame error [D]
Normal completion [A]

5.14.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
<2> The Checksum command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time t_{WT16}).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

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

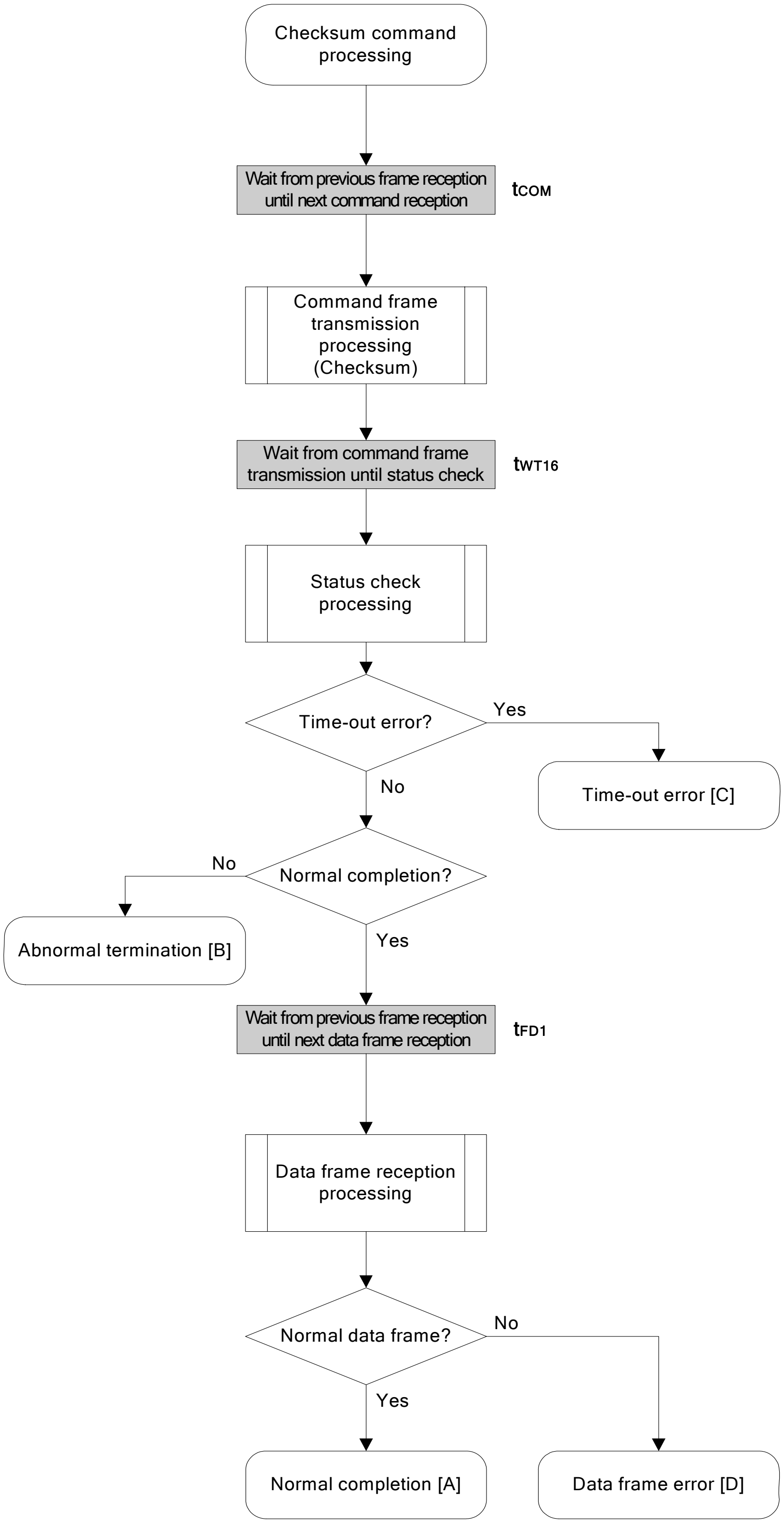
<6> Waits from the previous frame reception until the next command transmission (wait time t_{FD1}).
<7> The received data frame (checksum data) is checked.

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

5.14.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and checksum data was acquired normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame is abnormal.
	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.
Data frame error [D]		–	The checksum of the data frame received as version data is abnormal.

5.14.4 Flowchart


Checksum command processing
Wait from previous frame reception until next command reception
tCOM
Command frame transmission processing (Checksum)
Wait from command frame transmission until status check
tWT16
Status check processing
Time-out error?
Yes
Time-out error [C]
No
Normal completion?
No
Abnormal termination [B]
Yes
Wait from previous frame reception until next data frame reception
tFD1
Data frame reception processing
Normal data frame?
No
Yes
Normal completion [A]
Data frame error [D]

5.14.5 Sample program

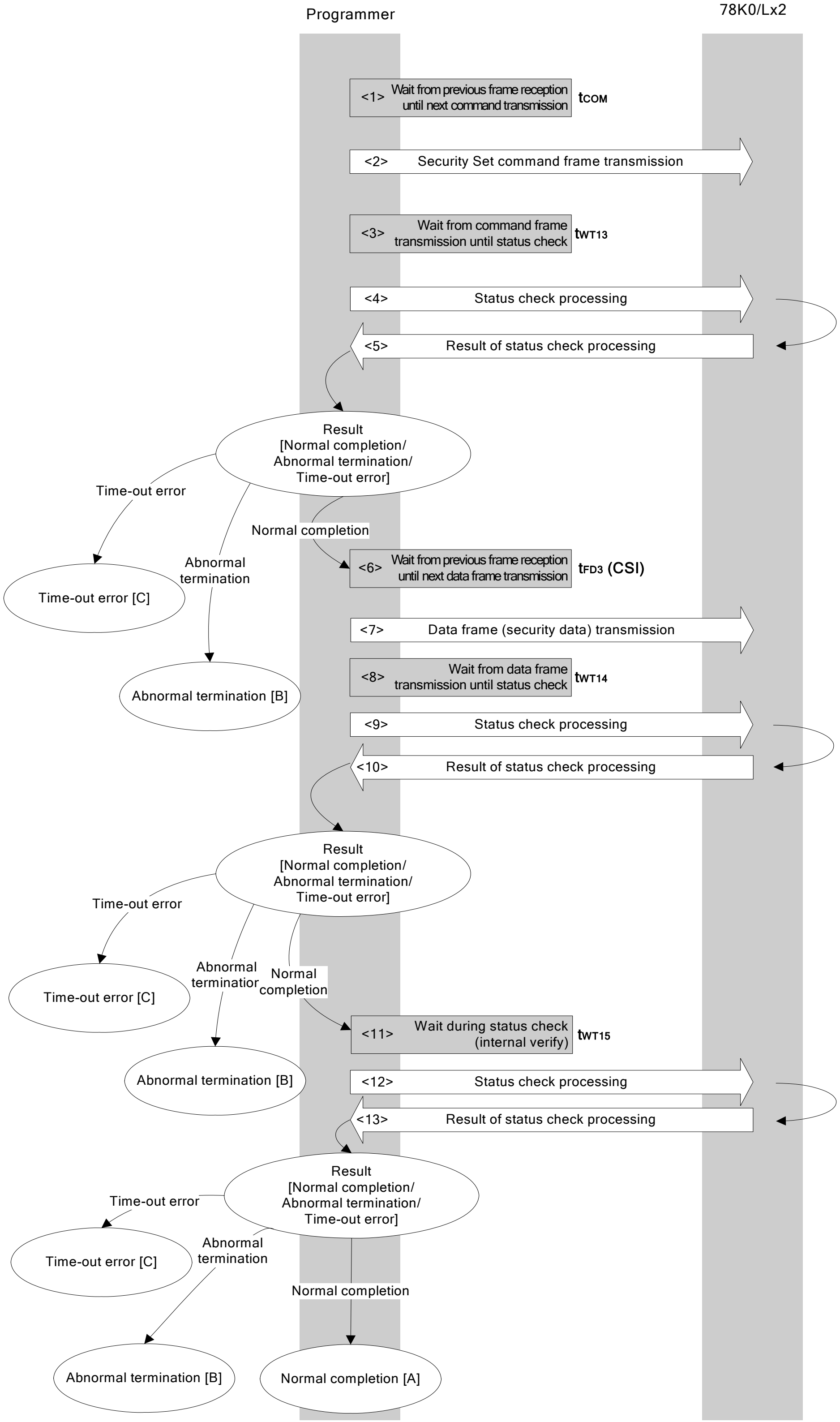
The following shows a sample program for Checksum command processing.

```
/****************************************************************/
/*                                                              */
/* Get checksum command (CSI)                                   */
/*                                                              */
/****************************************************************/
/* [i] u16 *sum    ... pointer to checksum save area            */
/* [i] u32 top     ... start address                            */
/* [i] u32 bottom  ... end address                              */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;
    u16    block_num;

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

    block_num = get_block_num(top, bottom); // get block num

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

    put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm);    // send "Checksum" command

    fl_wait(tWT16);

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


    /************************************************/
    /*      get data frame (Checksum data)          */
    /************************************************/
    fl_wait(tFD1 * block_num);              // wait before getting data frame
    rc = get_dfrm_csi(fl_rxdata_frm);// get data frame(version data)

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

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

5.15 Security Set Command

5.15.1 Processing sequence chart

Security Set command processing sequence


Programmer
78K0/Lx2
<1> Wait from previous frame reception until next command transmission
tCOM
<2> Security Set command frame transmission
<3> Wait from command frame transmission until status check
tWT13
<4> Status check processing
<5> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Normal completion
Abnormal termination
Time-out error [C]
Abnormal termination [B]
<6> Wait from previous frame reception until next data frame transmission
tFD3 (CSI)
<7> Data frame (security data) transmission
<8> Wait from data frame transmission until status check
tWT14
<9> Status check processing
<10> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Abnormal terminatior
Normal completion
Time-out error [C]
Abnormal termination [B]
<11> Wait during status check (internal verify)
tWT15
<12> Status check processing
<13> Result of status check processing
Result [Normal completion/ Abnormal termination/ Time-out error]
Time-out error
Abnormal termination
Normal completion
Time-out error [C]
Abnormal termination [B]
Normal completion [A]

5.15.2 Description of processing sequence

<1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
<2> The Security Set command is transmitted by command frame transmission processing.
<3> Waits from command transmission until status check processing (wait time t_{WT13}).
<4> The status frame is acquired by status check processing.
<5> The following processing is performed according to the result of status check processing.

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

<6> Waits from the previous frame reception until the data frame transmission (wait time t_{FD3}(CSI)).
<7> The data frame (security setting data) is transmitted by data frame transmission processing.
<8> Waits from data frame transmission until status check processing (wait time t_{WT14}).
<9> The status frame is acquired by status check processing.
<10> The following processing is performed according to the result of status check processing.

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

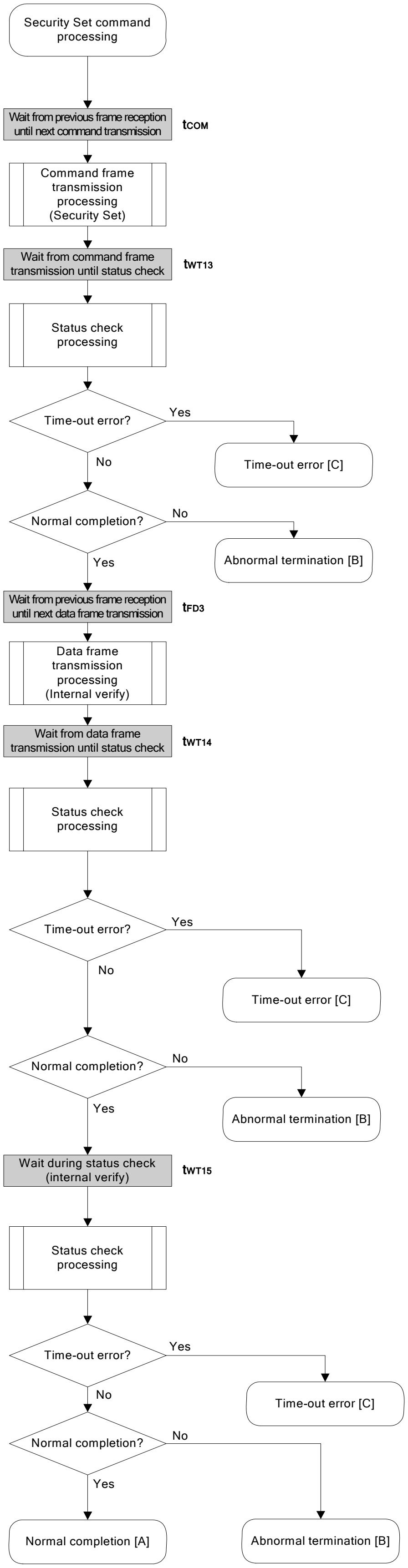
<11> Waits until status acquisition (completion of internal verify) (wait time t_{WT15}).
<12> The status frame is acquired by status check processing.
<13> The following processing is performed according to the result of status check processing.

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

5.15.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and security setting was performed normally.
Abnormal termination [B]	Parameter error	05H	Command information (parameter) is not 00H.
	Checksum error	07H	The checksum of the transmitted command frame or data frame is abnormal.
	Write error	1CH	Security data has already been set, or a write error has occurred.
	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.

5.15.4 Flowchart


Security Set command processing
Wait from previous frame reception until next command transmission
tCOM
Command frame transmission processing (Security Set)
Wait from command frame transmission until status check
tWT13
Status check processing
Time-out error?
Yes
No
Time-out error [C]
Normal completion?
No
Yes
Abnormal termination [B]
Wait from previous frame reception until next data frame transmission
tFD3
Data frame transmission processing (Internal verify)
Wait from data frame transmission until status check
tWT14
Status check processing
Time-out error?
Yes
No
Time-out error [C]
Normal completion?
No
Yes
Abnormal termination [B]
Wait during status check (internal verify)
tWT15
Status check processing
Time-out error?
Yes
No
Time-out error [C]
Normal completion?
No
Yes
Normal completion [A]
Abnormal termination [B]

5.15.5 Sample program

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

```
/****************************************************************/
/*                                                              */
/* Set security flag command (CSI)                              */
/*                                                              */
/****************************************************************/
/* [i] u8 scf     ... Security flag data                        */
/* [r] u16        ... error code                                */
/****************************************************************/
u16        fl_csi_setscf(u8 scf)
{
    u16    rc;

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

    fl_txdata_frm[1] = 0x03;         // "BOT" (fixed 0x03)



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

    /************************************************/
    /*   send 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);

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


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

    rc = fl_csi_getstatus(tWT15_MAX);       // get status frame
//  switch(rc) {
//
//         case  FLC_NO_ERR:   return rc;   break; // case [A]
//         case  FLC_DFTO_ERR: return rc;   break; // case [C]
//         default:            return rc;   break; // case [B]
//  }
    return rc;
}
```

CHAPTER 6 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS

This chapter describes the parameter characteristics between the programmer and the 78K0/Lx2 in the flash memory programming mode.

Be sure to refer to the user's manual of the 78K0/Lx2 for electrical specifications when designing a programmer.

6.1 Basic Characteristics

Parameter	Condition	Symbol	MIN.	TYP.	MAX.	Unit
78K0/Lx2 operating clock in flash memory programming mode	Internal high-speed oscillation clock	f_{RH}	7.6	8	8.4	MHz
X1 clock	During UART communication	f_X	2		20	
External main system clock		f_{EXCLK}	2		20	

<R>

6.2 Flash Memory Programming Mode Setting Time

Parameter		Symbol	MIN.	TYP.	MAX.
V_{DD}↑ to FLMD0↑		t_{DP}	1 ms		
FLMD0↑ to $\overline{\text{RESET}}$↑		t_{PR}	2 ms		
Count start time from $\overline{\text{RESET}}$↑ to FLMD0^Note 1^		t_{RP}	59,327/f_{RH}		
Count finish time from $\overline{\text{RESET}}$↑ to FLMD0^Note 1^		t_{RPE}			238,414/f_{RH}
FLMD0 counter high-level/low-level width		t_{PW}	10 μs		100 μs
Wait for Reset command (CSI)		t_{RC}	444,463/f_{RH}		
Wait for low-level data 1 (UART)	X1 clock	t_{R1}	444,463/f_{RH} + 2^{16}/f_X		
	External main system clock		444,463/f_{RH}		
Wait for low-level data 2 (UART)		t_{12}	15,000/f_{RH}		
Wait for Read command (UART)		t_{2C}	15,000/f_{RH}		
Width of low-level data 1/2^Note 2^		t_{L1}, t_{L2}		**Note 2**	
FLMD0 counter rise/fall time		–			1 μs
Reset low level width ($\overline{\text{RESET}}$↑to $\overline{\text{RESET}}$↓) ^Note 3^		t_{RST}	1,950 ms		

Notes 1. (59,327/f_{RH} + 238,414/f_{RH})/2 is recommended as the standard value for the FLMD0 pulse input timing.

2. The low-level width is the same as the 00H data width at 9,600 bps.

3. When the mode is switched from the normal operating mode to the flash memory programming mode after the microcontroller is powered on (reset is released), be sure to wait for the period of this parameter at minimum before reset for mode switching after power-on (reset release).

Remarks 1. Calculate the parameters assuming that f_{RH} = 8 MHz.

2. The waits are defined as follows.

<t_{R1} (MIN.)>

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

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

<R> 6.3 Programming Characteristics

Wait	Condition	Symbol	Serial I/F	MIN.	MAX.
Between data frame transmission/reception	Data frame reception	t_{DR}	CSI	$64/f_{RH}$	
			UART	$74/f_{RH}$	
	Data frame transmission	t_{DT}	CSI	$88/f_{RH}$	
			UART	0[Note 1]	
From Status command frame reception until status frame transmission	–	t_{SF}	CSI	$166/f_{RH}$	
From status frame transmission until data frame transmission (1)	–	t_{FD1}[Note 2]	CSI	$54,368/f_{RH}$	
			UART	0[Note 1]	
From status frame transmission until data frame transmission (2)	Silicon signature data	t_{FD2}	CSI	$321/f_{RH}$	
	Version data			$136/f_{RH}$	
	–		UART	0[Note 1]	
From status frame transmission until data frame reception	–	t_{FD3}	CSI	$163/f_{RH}$	
			UART	$101/f_{RH}$	
From status frame transmission until command frame reception	–	t_{COM}	CSI	$64/f_{RH}$	
			UART	$71/f_{RH}$	

Notes 1. When successive reception is enabled for the programmer

2. Time for one block transmission

Remarks 1. Calculate the parameters assuming that f_{RH} = 8 MHz.

2. The waits are defined as follows.

<t_{DR}, t_{FD3}, t_{COM}>

The 78K0/Lx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

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

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

<t_{DT}, t_{SF}, t_{FD1}, t_{FD2}>

The 78K0/Lx2 is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

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

The MAX. time is not specified. Continue polling for about 3 seconds until the data is received.

Command	Symbol	Serial I/F	MIN.		MAX.
Reset	t_{WT0}	CSI	172/f_{RH}		
		UART	**Note 1**		
Chip Erase	t_{WT1}	–	857,883/f_{RH} + 44,160 × total number of blocks/f_{RH}		186,444,400/f_{RH} + 11,304,960 × total number of blocks/f_{RH}
Block Erase	t_{WT2} **Note 2**	–	214,714/f_{RH} × execution count of simultaneous selection and erasure + 44,160/f_{RH} × number of blocks to be erased		54,582,372/f_{RH} × execution count of simultaneous selection and erasure + 11,304,960/f_{RH} × number of blocks to be erased
Programming	t_{WT3}	CSI	1,348/f_{RH}		
		UART	**Note 1**		
	t_{WT4} **Note 3**	–	68,118/f_{RH}		397,587/f_{RH}
	t_{WT5} **Note 4**	CSI	Block 0	100,407/f_{RH}	132,144,427/f_{RH}
			Block 1 to 127	100,407/f_{RH}	102,178/f_{RH}
		UART	Block 0	**Note 1**	132,144,427/f_{RH}
			Block 1 to 127	**Note 1**	102,178/f_{RH}
Verify	t_{WT6}	CSI	686/f_{RH}		
		UART	**Note 1**		
	t_{WT7} **Note 3**	CSI	12,827/f_{RH}		
		UART	**Note 1**		
Block Blank Check	t_{WT8} **Note 4**	CSI	45,835/f_{RH}		55,044/f_{RH}
		UART	**Note 1**		55,044/f_{RH}
Oscillating Frequency Set	t_{WT9}	CSI	1,127/f_{RH}		
		UART	**Note 1**		
Silicon Signature	t_{WT11}	CSI	1,233/f_{RH}		
		UART	**Note 1**		
Version Get	t_{WT12}	CSI	242/f_{RH}		
		UART	**Note 1**		
Security Set	t_{WT13}	CSI	923/f_{RH}		
		UART	**Note 1**		
	t_{WT14}	–	275,518/f_{RH}		66,005,812/f_{RH}
	t_{WT15}	CSI	368,277/f_{RH}		66,018,156/f_{RH}
		UART	**Note 1**		66,018,156/f_{RH}
Checksum	t_{WT16}	CSI	583/f_{RH}		
		UART	**Note 1**		

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

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

3. Time for 256-byte data transmission

4. Time for one block transmission

Remark 1. Calculate the parameters assuming that f_{RH} = 8 MHz.

Remark 2. The waits are defined as follows.

<t_{WT0} to t_{WT16}>

The 78K0/Lx2 completes command processing between the MIN. and MAX. times.
The programmer must repeat the status check for the period of the MAX. time (or about 3 seconds, if the MAX. time is not specified).

Supplement Simultaneous selection and erasure performed by Block Erase command

The Block Erase command of the 78K0/Lx2 is executed by repeating “simultaneous selection and erasure”, which erases multiple blocks simultaneously.
The wait time inserted during Block Erase command execution is therefore equal to the total execution time of “simultaneous selection and erasure”.
To calculate the “total execution time of simultaneous selection and erasure”, the execution count (M) of the simultaneous selection and erasure must first be calculated.
“M” is calculated by obtaining the number of blocks to be erased simultaneously (number of blocks to be selected and erased simultaneously).

The following describes the method for calculating the number of blocks to be selected and erased simultaneously and the execution count (M).

(1) Calculation of number of blocks to be selected and erased simultaneously

The number of blocks to be selected and erased simultaneously should be 1, 2, 4, 8, 16, 32, 64, or 128, depending on which satisfies all of the following conditions.

[Condition 1]

(Number of blocks to be erased) ≥ (Number of blocks to be selected and erased simultaneously)

[Condition 2]

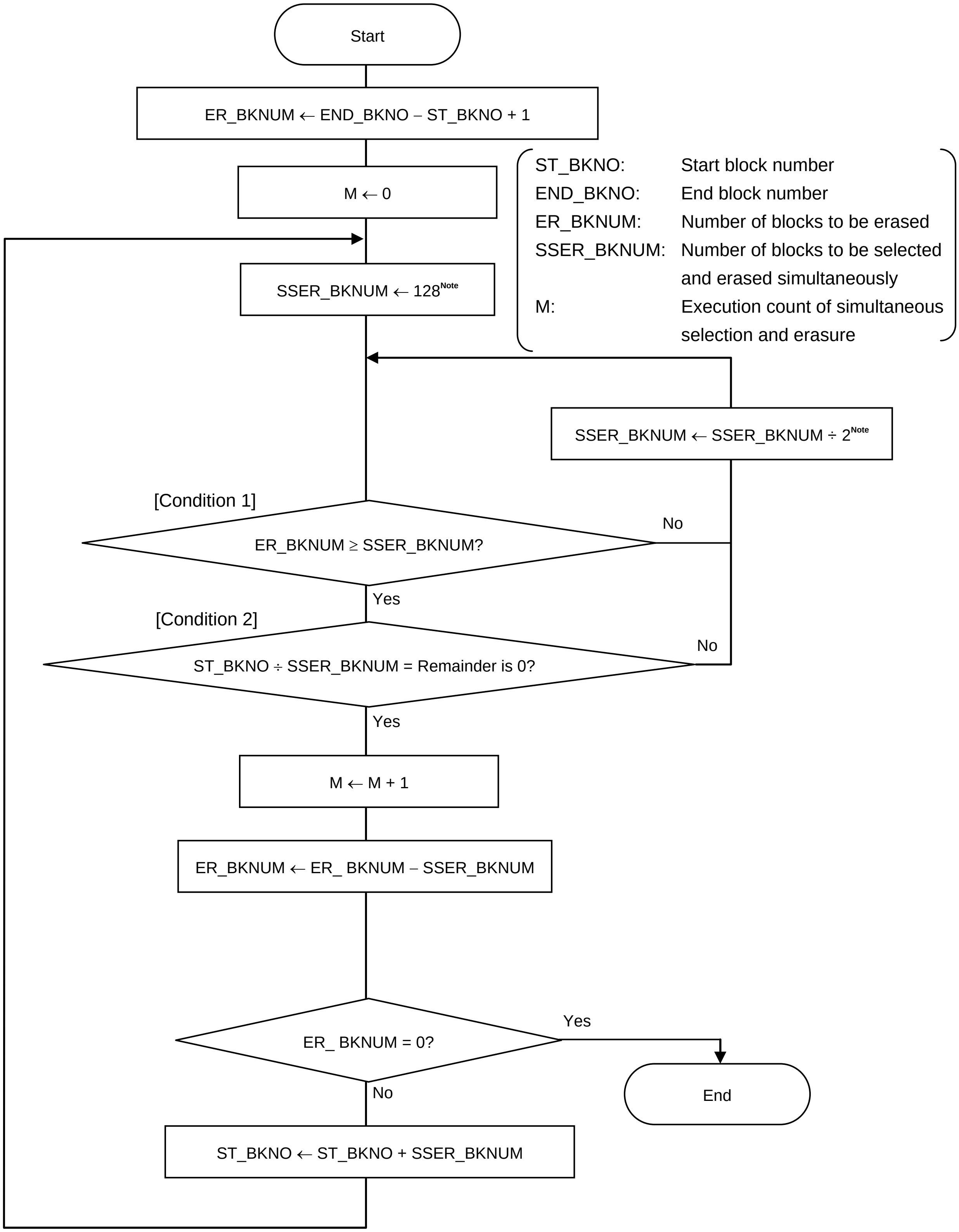
(Start block number) ÷ (Number of blocks to be selected and erased simultaneously) = Remainder is 0

[Condition 3]

The maximum value among the values that satisfy both Conditions 1 and 2

(2) Calculation of the execution count (M) of simultaneous selection and erasure

Calculation of the execution count (M) is illustrated in the following flowchart.


Start
ER_BKNUM ← END_BKNO – ST_BKNO + 1
M ← 0
ST_BKNO: Start block number
END_BKNO: End block number
ER_BKNUM: Number of blocks to be erased
SSER_BKNUM: Number of blocks to be selected and erased simultaneously
M: Execution count of simultaneous selection and erasure
SSER_BKNUM ← 128Note
SSER_BKNUM ← SSER_BKNUM ÷ 2Note
[Condition 1]
ER_BKNUM ≥ SSER_BKNUM?
No
Yes
[Condition 2]
ST_BKNO ÷ SSER_BKNUM = Remainder is 0?
No
Yes
M ← M + 1
ER_BKNUM ← ER_ BKNUM – SSER_BKNUM
ER_ BKNUM = 0?
Yes
End
No
ST_BKNO ← ST_BKNO + SSER_BKNUM

Note Based on the maximum value of SSER_BKNUM (128), obtain the value that satisfies Conditions 1 and 2 by executing SSER_BKNUM ÷ 2; Condition 3 is then satisfied.

Example 1 Erasing blocks 1 to 127 (N (number of blocks to be erased) = 127)

<1> The first start block number is 1 and the number of blocks to be erased is 127; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 1 is then erased.

<2> After block 1 is erased, the next start block number is 2 and the number of blocks to be erased is 126; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 2 and 3 are then erased.

<3> After blocks 2 and 3 are erased, the next start block number is 4 and the number of blocks to be erased is 124; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1, 2, and 4, the value that satisfies Condition 3 is 4, so the number of blocks to be selected and erased simultaneously is 4; blocks 4 to 7 are then erased.

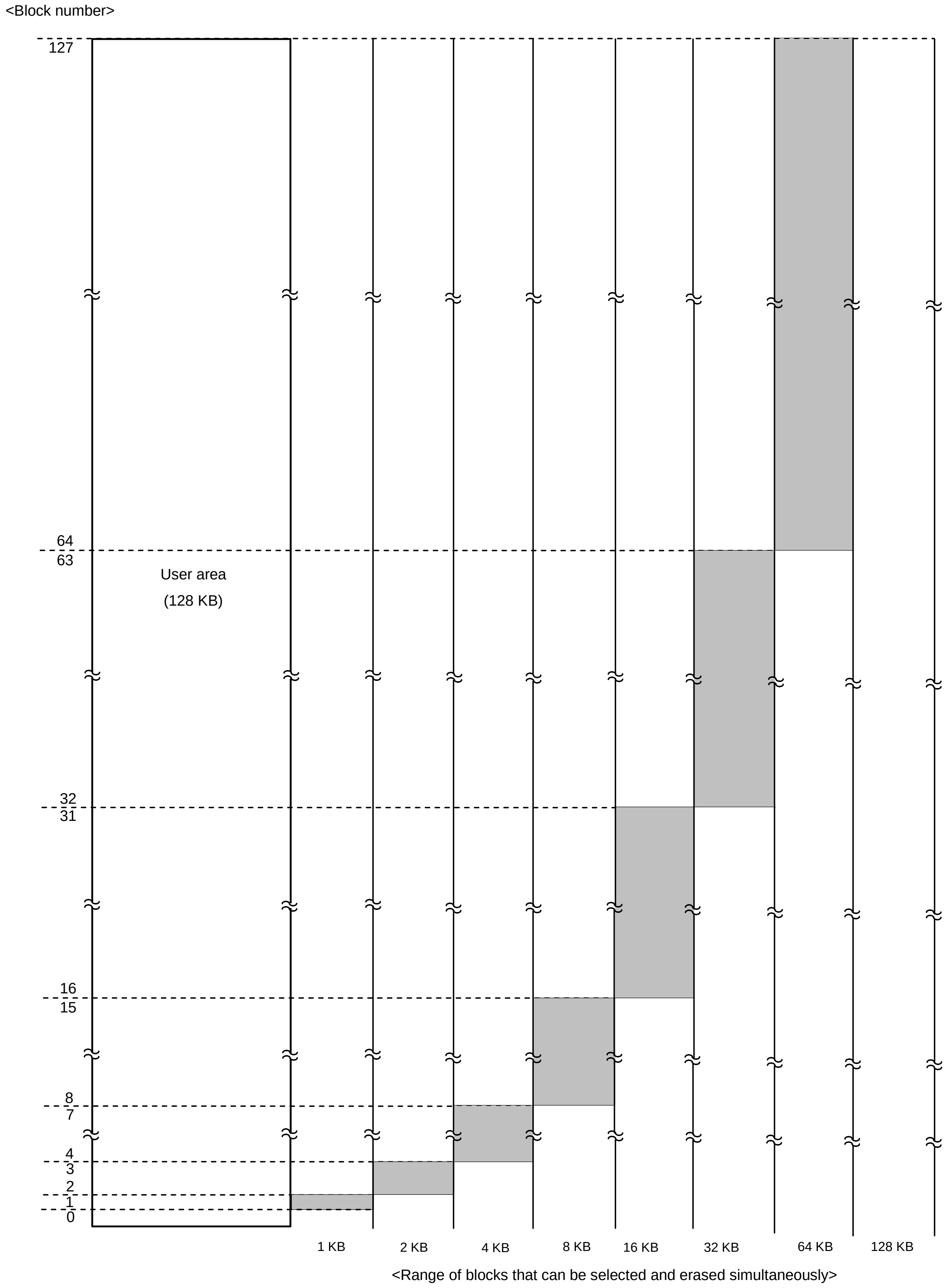
<4> After blocks 4 to 7 are erased, the next start block number is 8 and the number of blocks to be erased is 120; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, and 8, the value that satisfies Condition 3 is 8, so the number of blocks to be selected and erased simultaneously is 8; blocks 8 to 15 are then erased.

<5> After blocks 8 to 15 are erased, the next start block number is 16 and the number of blocks to be erased is 112; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, and 16, the value that satisfies Condition 3 is 16, so the number of blocks to be selected and erased simultaneously is 16; blocks 16 to 31 are then erased.
After blocks 16 to 31 are erased, the next start block number is 32 and the number of blocks to be erased is 96; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, 16, and 32, the value that satisfies Condition 3 is 32, so the number of blocks to be selected and erased simultaneously is 32; blocks 32 to 63 are then erased.

<6> After blocks 32 to 63 are erased, the next start block number is 64 and the number of blocks to be erased is 64; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, 32, and 64.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, 16, 32, and 64, the value that satisfies Condition 3 is 64, so the number of blocks to be selected and erased simultaneously is 64; blocks 64 to 127 are then erased.

Therefore, simultaneous selection and erasure is executed seven times (1, 2 and 3, 4 to 7, 8 to 15, 16 to 31, 32 to 63, and 64 to 127) to erase blocks 1 to 127, so M = 7 is obtained.

Block configuration when executing simultaneous selection and erasure (when erasing blocks 1 to 127)



Example 2 Erasing blocks 5 to 10 (N (number of blocks to be erased) = 6)

<1> The first start block number is 5 and the number of blocks to be erased is 6; the values that satisfy Condition 1 are therefore 1, 2, and 4.
Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 5 is the erased.

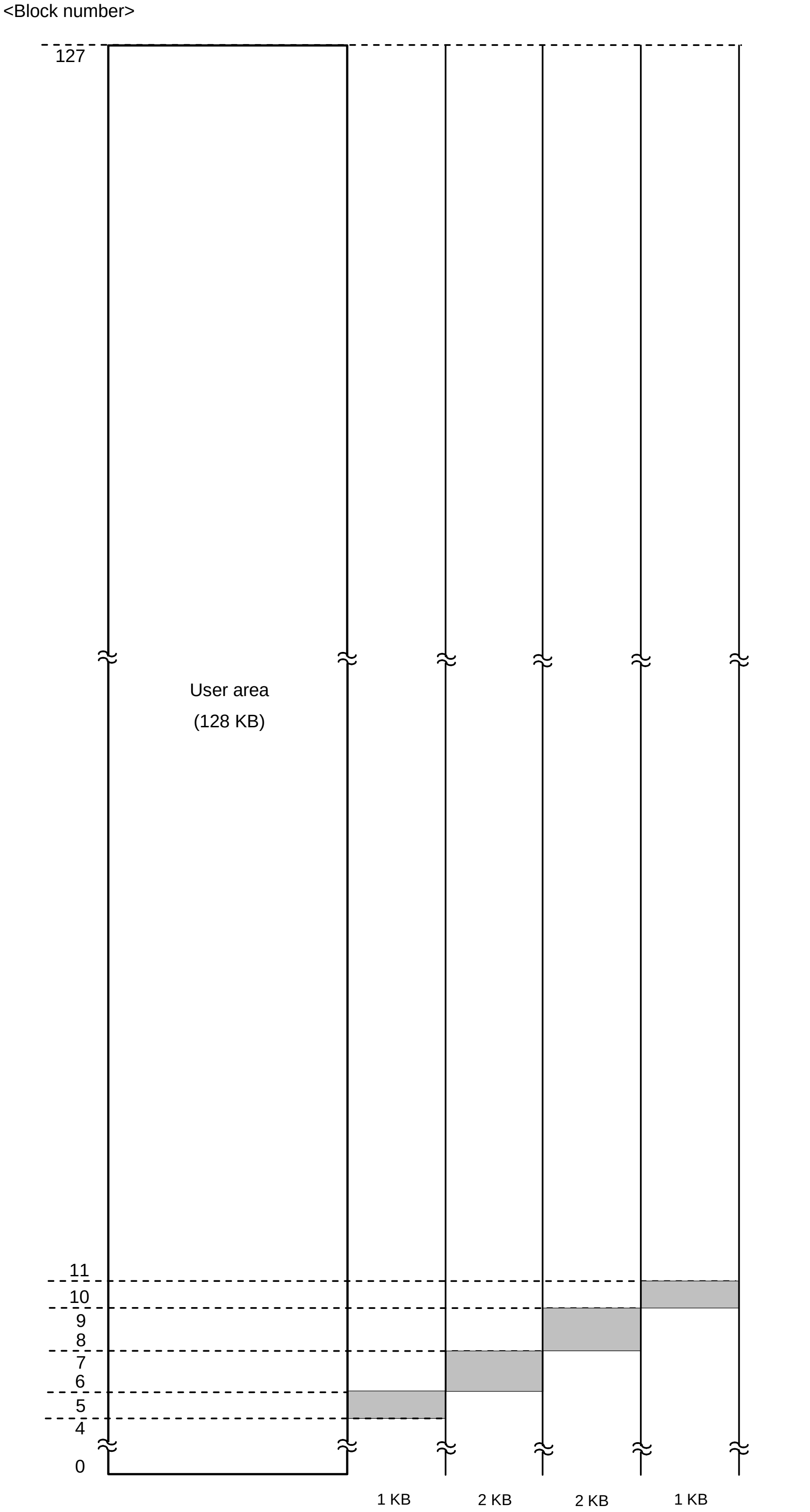
<2> After block 5 is erased, the next start block number is 6 and the number of blocks to be erased is 5; the values that satisfy Condition 1 are therefore 1, 2, and 4.
Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 6 and 7 are then erased.

<3> After blocks 6 and 7 are erased, the next start block number is 8 and the number of blocks to be erased is 3; the values that satisfy Condition 1 are therefore 1 and 2.
Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 8 and 9 are then erased.

<4> After blocks 8 and 9 are erased, the next start block number is 10 and the number of blocks to be erased is 1; the value that satisfies Condition 1 is therefore 1. This also satisfies Conditions 2 and 3, so the number of blocks to be selected and erased simultaneously is 1; block 10 is then erased.

Therefore, simultaneous selection and erasure is executed four times (5, 6 and 7, 8 and 9, and 10) to erase blocks 5 to 10, so M = 4 is obtained.

Block configuration when executing simultaneous selection and erasure (when erasing blocks 5 to 10)



<Range of blocks that can be selected and erased simultaneously>

Example 3 Erasing blocks 25 to 73 (N (number of blocks to be erased) = 49)

<1> The first start block number is 25 and the number of blocks to be erased is 49; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32.
Moreover, the value that satisfies Condition 2 is 1 and the value that satisfies Condition 3 is 1, so the number of blocks to be selected and erased simultaneously is 1; only block 25 is then erased.

<2> After block 25 is erased, the next start block number is 26 and the number of blocks to be erased is 48; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32.
Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 26 and 27 are then erased.

<3> After blocks 26 and 27 are erased, the next start block number is 28 and the number of blocks to be erased is 46; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32.
Moreover, the values that satisfy Condition 2 are 1, 2, and 4, the value that satisfies Condition 3 is 4, so the number of blocks to be selected and erased simultaneously is 4; blocks 28 to 31 are then erased.

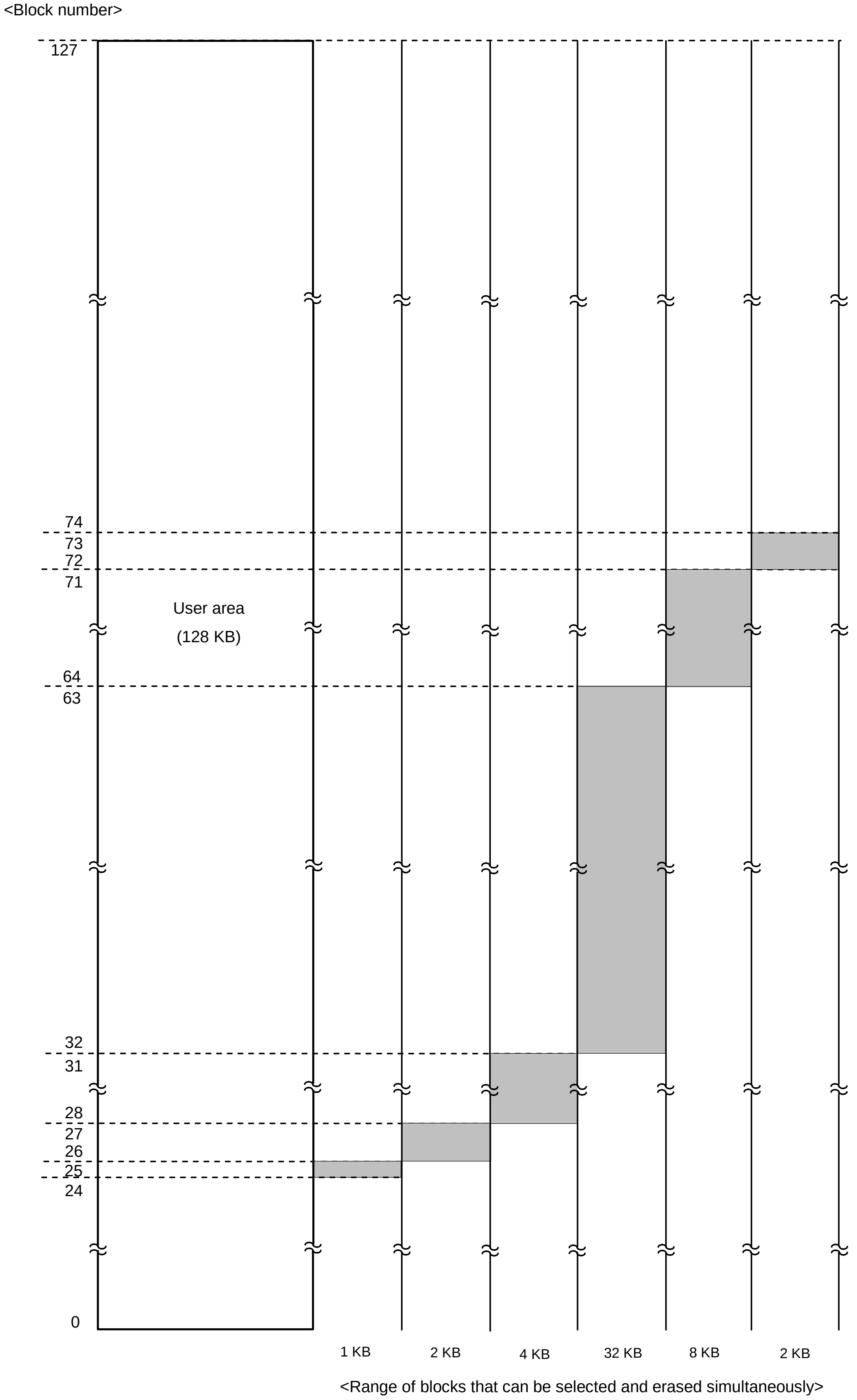
<4> After blocks 28 to 31 are erased, the next start block number is 32 and the number of blocks to be erased is 42; the values that satisfy Condition 1 are therefore 1, 2, 4, 8, 16, and 32.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, 8, and 32, the value that satisfies Condition 3 is 32, so the number of blocks to be selected and erased simultaneously is 32; blocks 32 to 63 are then erased.

<5> After blocks 32 to 63 are erased, the next start block number is 64, and the number of blocks to be erased is 10; the values that satisfy Condition 1 are therefore 1, 2, 4, and 8.
Moreover, the values that satisfy Condition 2 are 1, 2, 4, and 8, the value that satisfies Condition 3 is 8, so the number of blocks to be selected and erased simultaneously is 8; blocks 64 to 71 are then erased.

<6> After blocks 64 to 71 are erased, the next start block number is 72, and the number of blocks to be erased is 2; the values that satisfy Condition 1 are therefore 1 and 2.
Moreover, the values that satisfy Condition 2 are 1 and 2, the value that satisfies Condition 3 is 2, so the number of blocks to be selected and erased simultaneously is 2; blocks 72 and 73 are then erased.

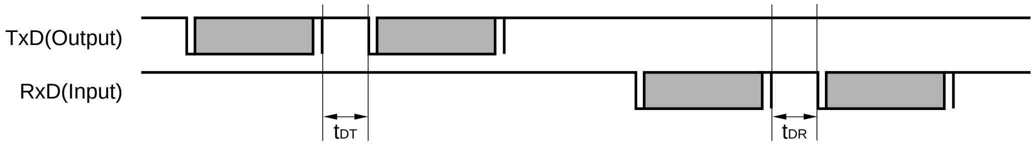
Therefore, simultaneous selection and erasure is executed six times (25, 26 and 27, 28 to 31, 32 to 63, 64 to 71, and 72 and 73) to erase blocks 25 to 73, so M = 6 is obtained.

Block configuration when executing simultaneous selection and erasure (when erasing blocks 25 to 73)

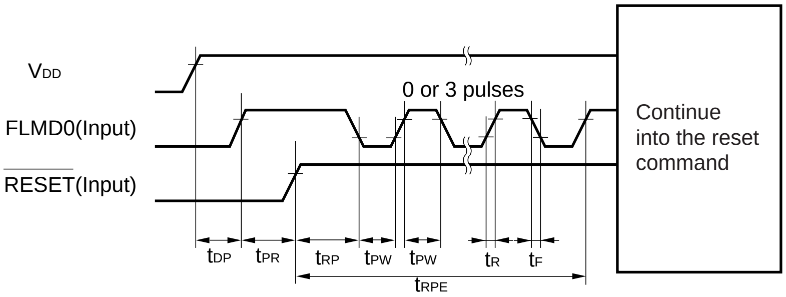


6.4 UART Communication Mode

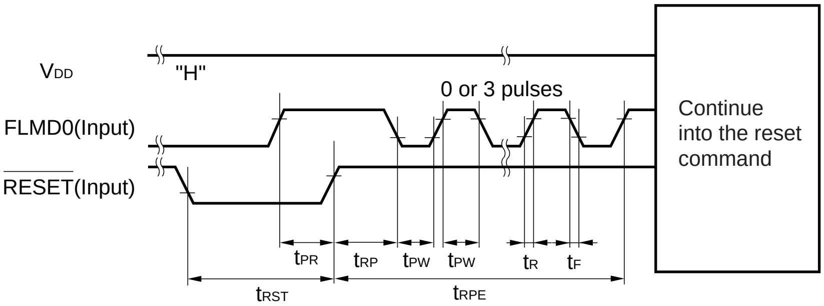
(a) Data frame



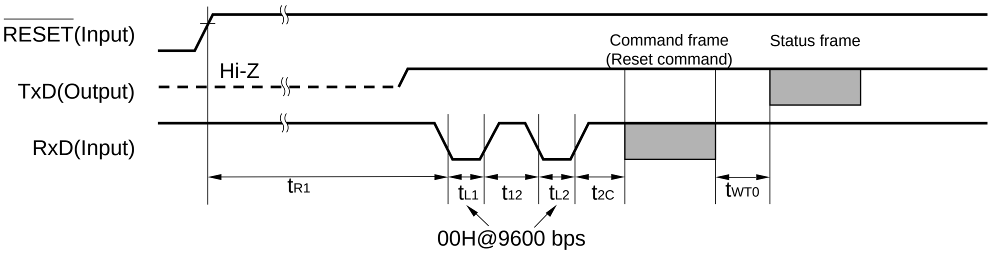
(b) Programming mode setting (At the time of power-on)



<R> (c) Programming mode setting (After power-on)

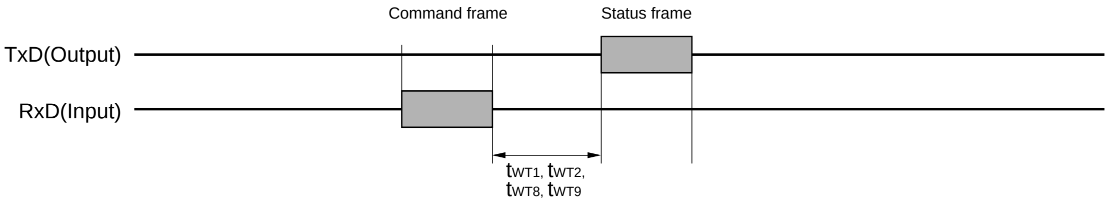


(d) Reset command

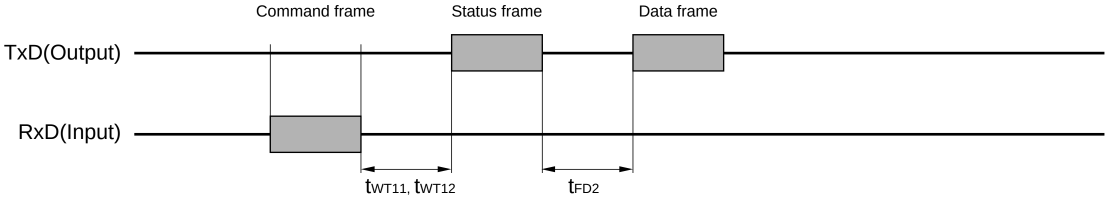


Remark TxD: TxD6
RxD: RxD6

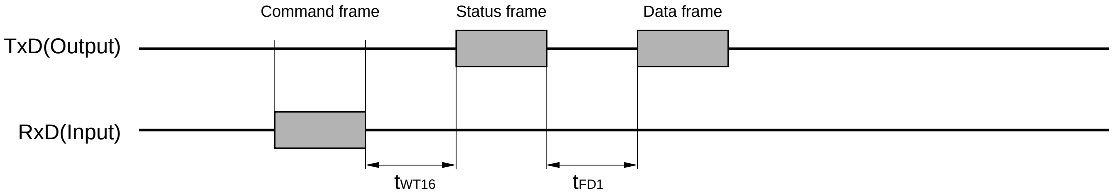
(e) Chip Erase command/Block Erase command/ Block Blank Check command/Oscillating Frequency Set command



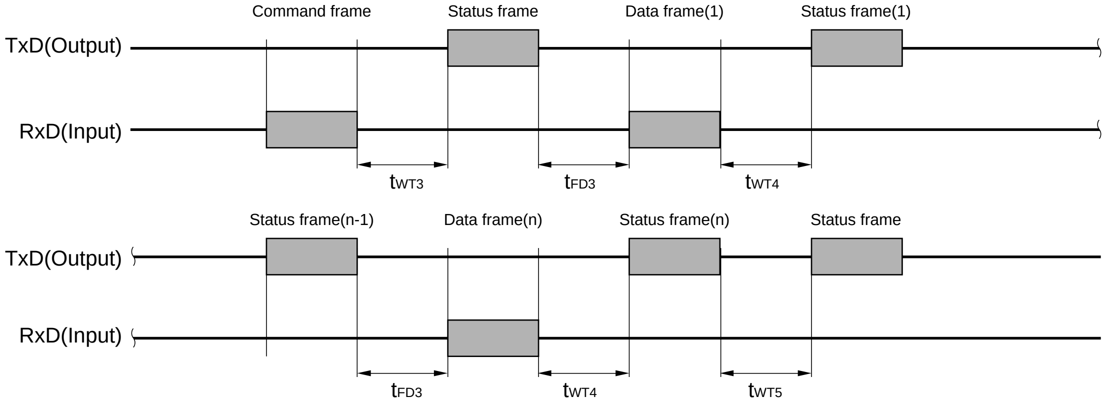
(f) Silicon Signature command/Version Get command



(g) Checksum command

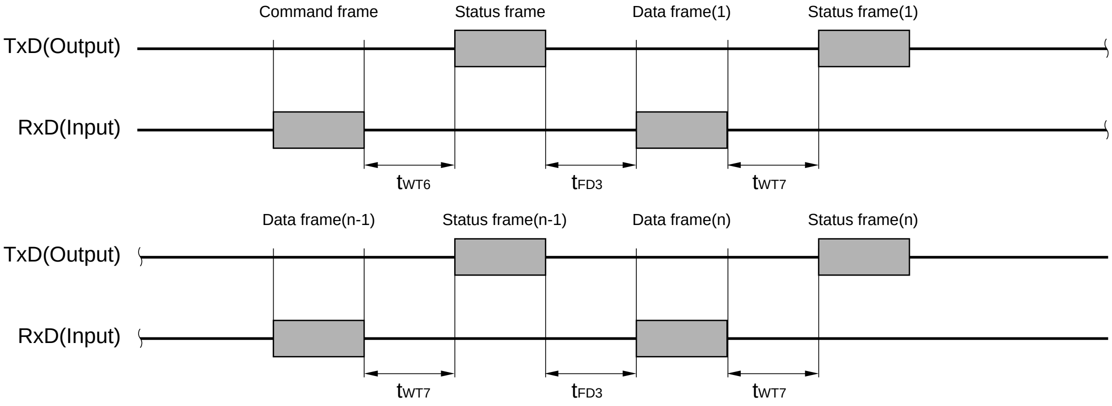


(h) Programming command

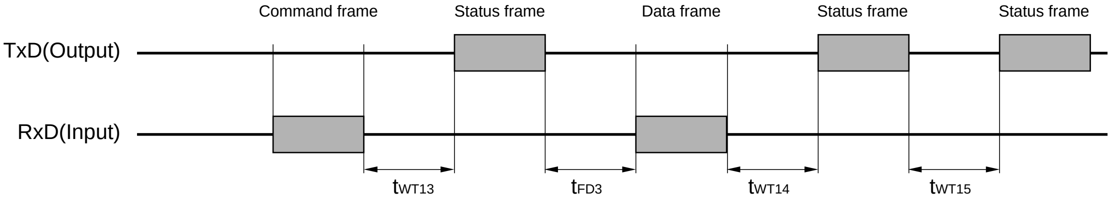


Remark TxD: TxD6
RxD: RxD6

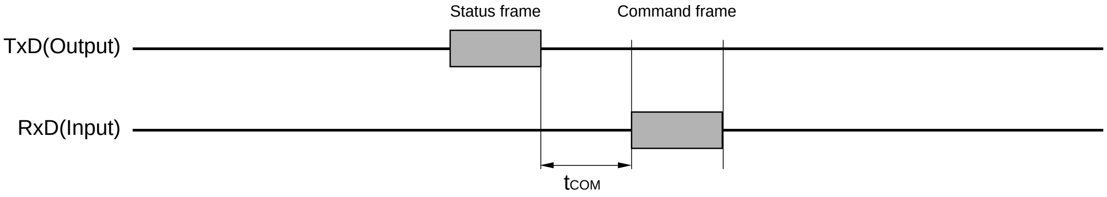
(i) Verify command



(j) Security Set command



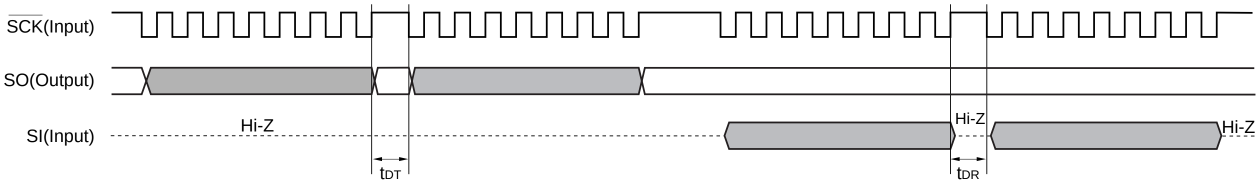
(k) Wait before command frame transmission



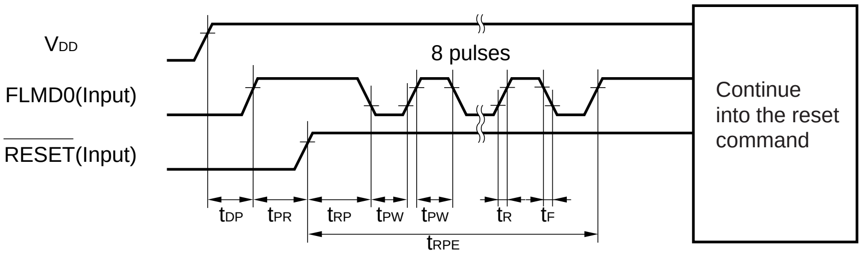
Remark TxD: TxD6
RxD: RxD6

6.5 3-Wire Serial I/O Communication Mode

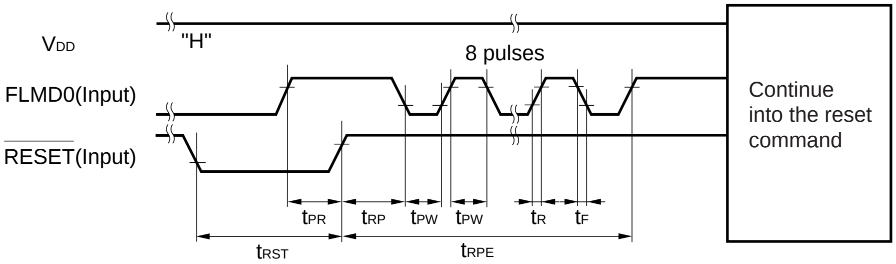
(a) Data frame



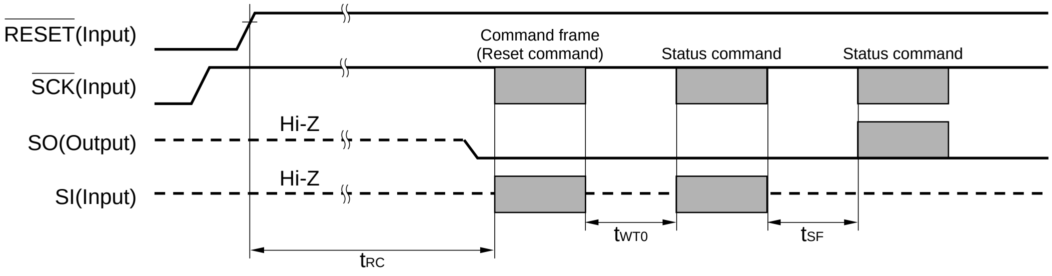
(b) Programming mode setting (At the time of power-on)



<R> (c) Programming mode setting (After power-on)



(d) Reset command

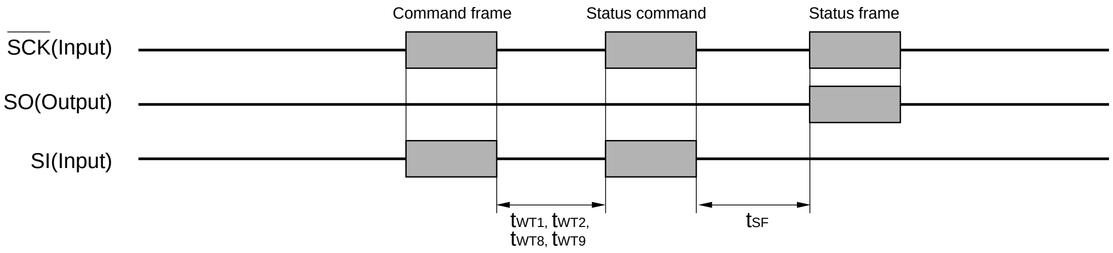


<R>

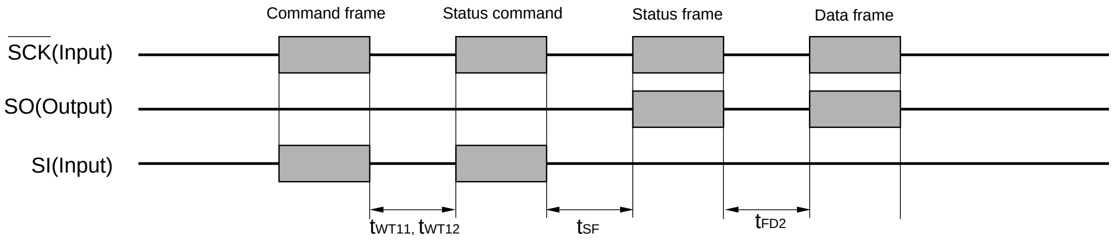
<R>

Remark $\overline{\text{SCK}}$: $\overline{\text{SCK10}}$
SC: SO10
SI: SI10

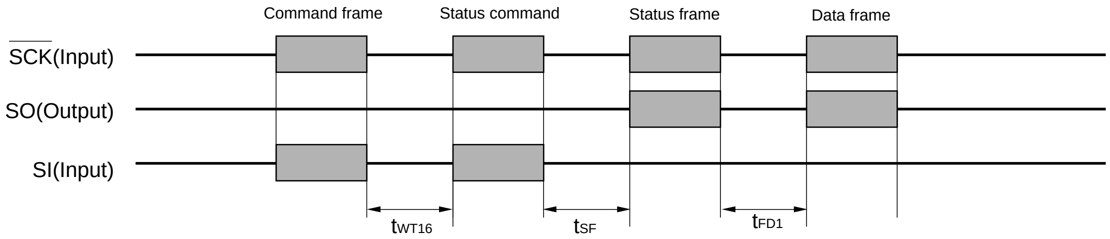
(e) Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command



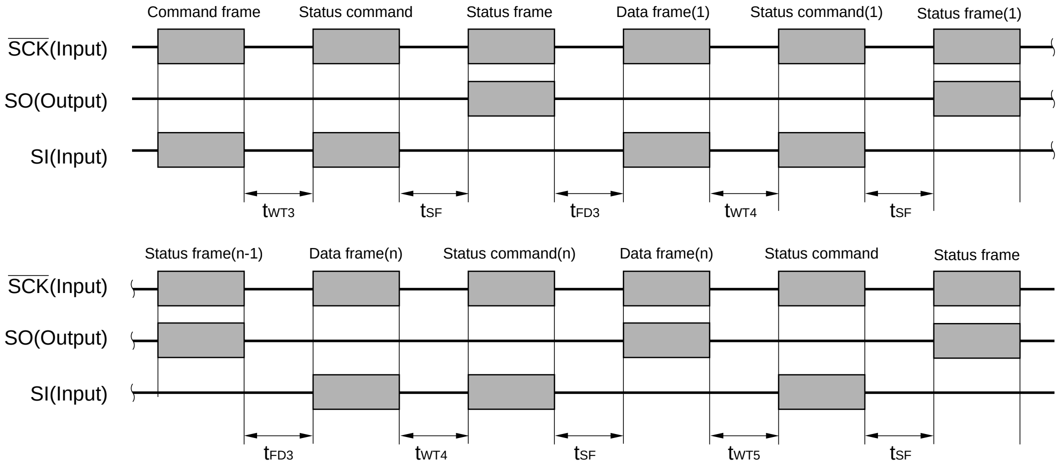
(f) Silicon Signature command/Version Get command



(g) Checksum command

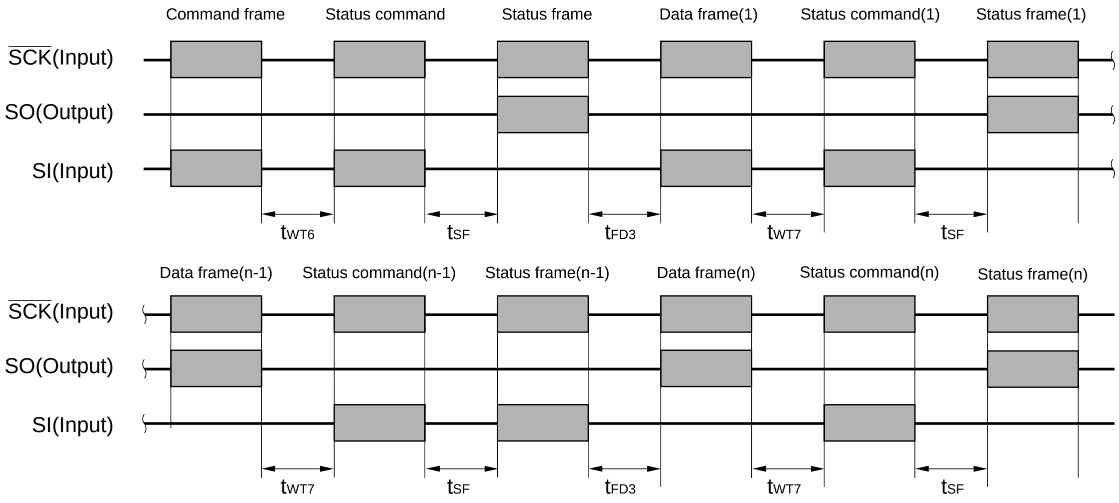


(h) Programming command

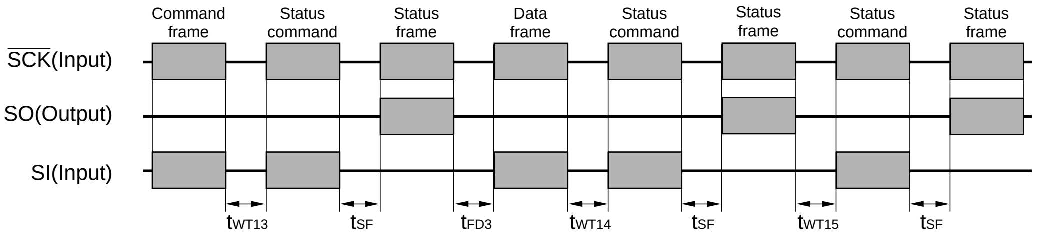


Remark $\overline{\text{SCK}}$: $\overline{\text{SCK10}}$
SC: SO10
SI: SI10

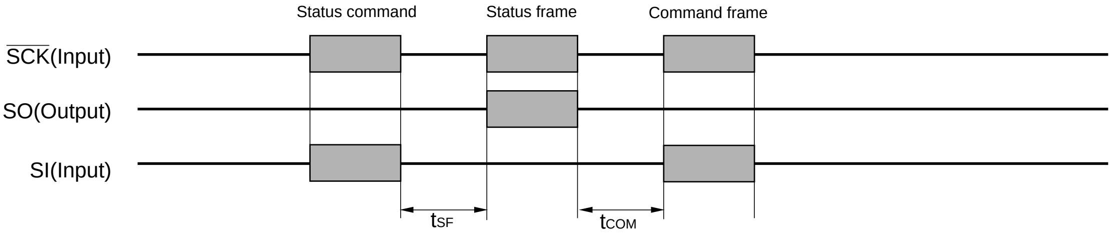
(i) Verify command



(j) Security Set command



(k) Wait before command frame transmission



Remark $\overline{\text{SCK}}$: $\overline{\text{SCK10}}$

SC: SO10

SI: SI10

APPENDIX A CIRCUIT DIAGRAMS (REFERENCE)

Figure A-1 to A-3 show circuit diagrams of the programmer and the 78K0/Lx2, for reference.

As for the pins which are not described in circuit diagrams, refer to the user's manual of each 78K0/Lx2 product when handling the pins.

Figure A-1. Reference Circuit Diagram of Programmer and 78K0/Lx2 (During UART communication: with X1 Clock Used)

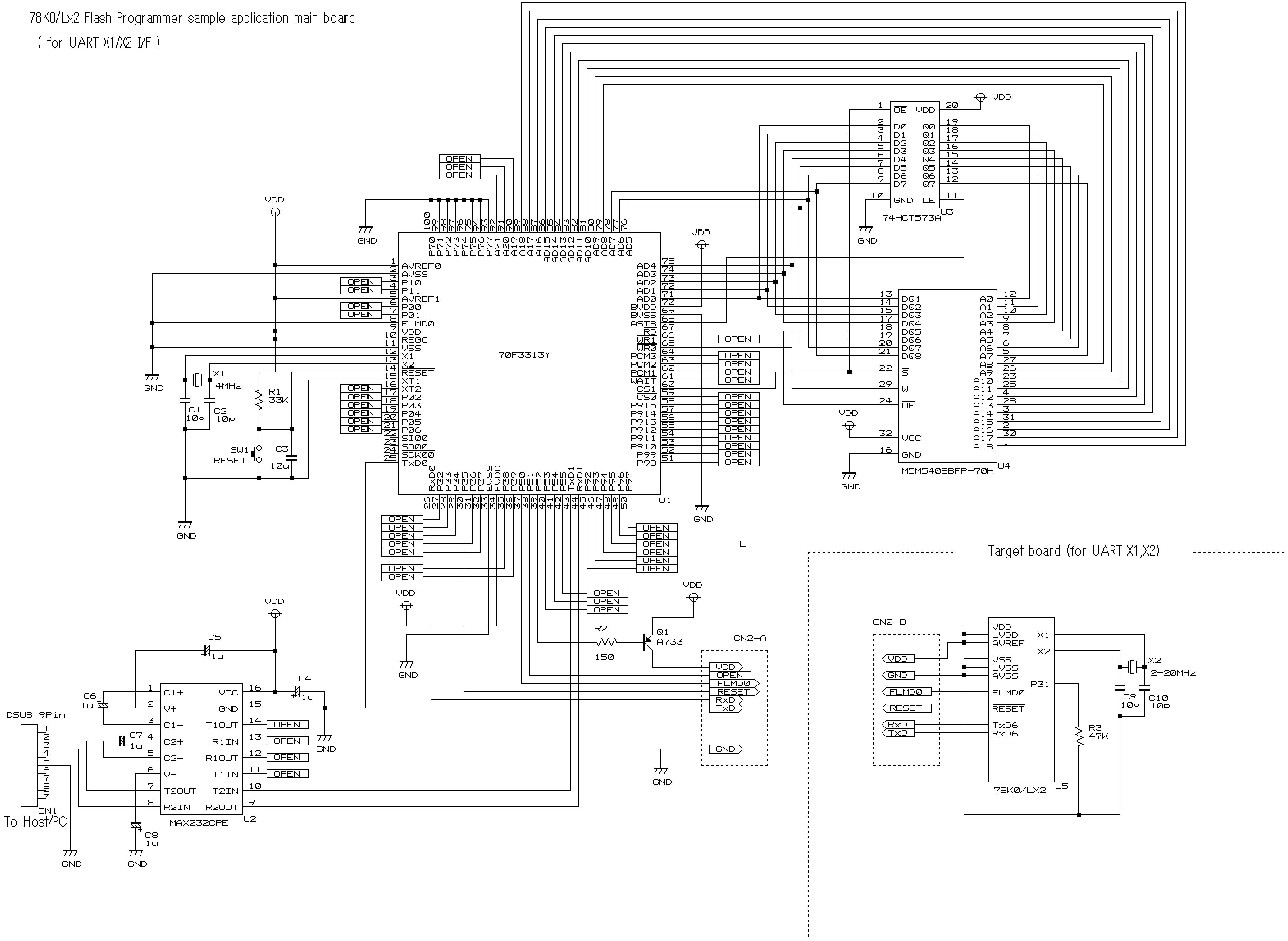


Figure A-2. Reference Circuit Diagram of Programmer and 78K0/Lx2 (During UART communication: with External Clock Used)

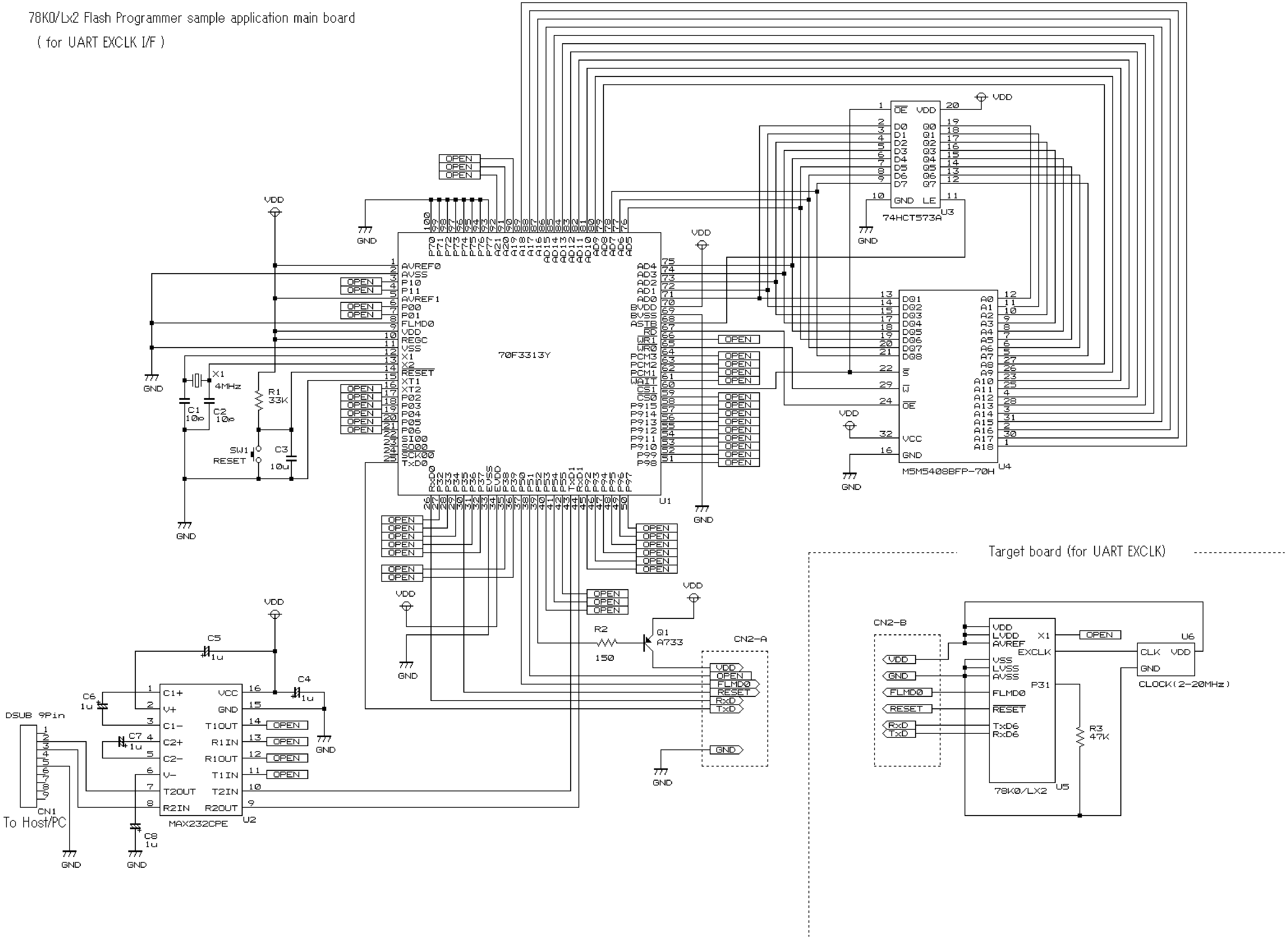
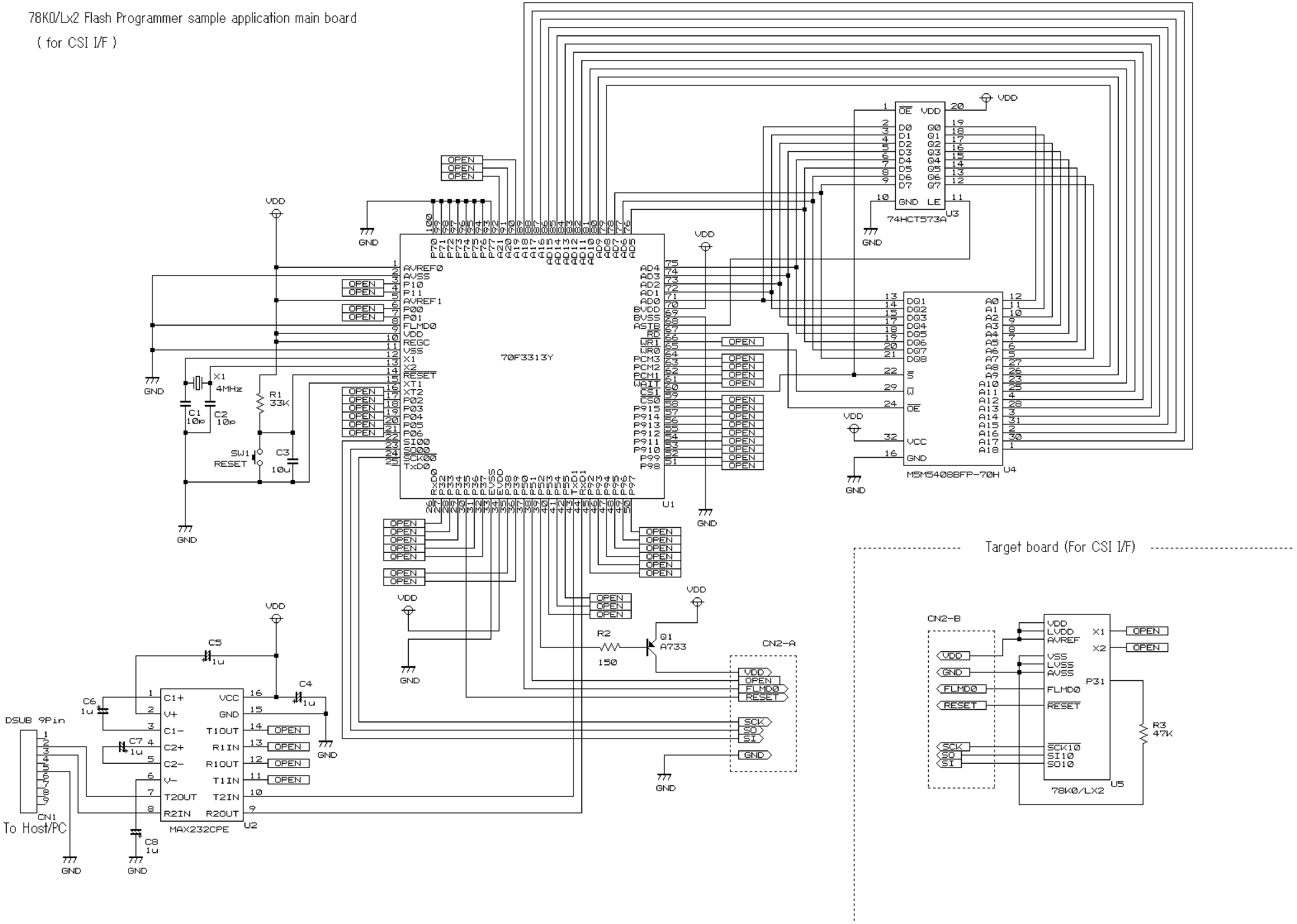


Figure A-3. Reference Circuit Diagram of Programmer and 78K0/Lx2 (During CSI Communication)



<R>

APPENDIX B REVISION HISTORY

B.1 Major Revisions in This Edition

Page	Description
Throughout	Deletion of **2.1 Programmer Control Pins** and **2.2 Details of control pins**
	Move of 9 sections (from **2.3 Basic Flowchart** to **2.11 Status List**) to **Chapter 1 FLASH MEMORY PROGRAMMING**
p.32	Modification of description in **3.5 Chip Erase Command**
p.40	Modification of **Table 3-1. Example of Silicon Signature Data**
pp.50 to 52	From **4.1 Command Frame Transmission Processing Flowchart** to **4.3 Data Frame Reception Processing Flowchart** • Modification of the symbol in the flowchart
pp.102 to 104	From **5.1 Command Frame Transmission Processing Flowchart** to **5.3 Data Frame Reception Processing Flowchart** • Modification of the symbol in the flowchart
p.106	Modification of **5.4.3 Status at processing completion**
p.159	Modification of **6.2 Flash Memory Programming Mode Setting Time**
pp.160 to 162	Modification of **6.3 Programming Characteristics**
p.170	Addition of **6.4 UART Communication Mode**
p.173	Modification of **6.5 3-Wire Serial I/O Communication Mode**
p.183	Addition of **APPENDIX B REVISION HISTORY**

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

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706