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.

应用笔记

V850ES/Jx2

32 位单片微控制器

Flash 存储器编程（编程器）

μPD70F3715
μPD70F3716
μPD70F3717
μPD70F3718
μPD70F3719
μPD70F3720
μPD70F3721
μPD70F3722
μPD70F3723
μPD70F3724

文件编号．U17965CA1V0AN00 （第一版）
出版日期 2008 年 2 月 N

日本印刷

［备忘录］

CMOS设备的注释

① 输入引脚处的电压波形

输入噪音或一个反射波引起的波形失真可能导致错误发生。如果由于噪音等的影响使CMOS设备的输入电压范围保持在V_{IL}（MAX）和V_{IH}（MIN）之间，设备可能发生错误。在输入电平固定时以及输入电平从V_{IL}（MAX）过渡到V_{IH}（MIN）时的传输期间，要防止散射噪声影响设备。

② 未使用的输入引脚的处理

CMOS设备的输入端保持开路可能导致误操作。如果一个输入引脚未被连接，则由于噪音等原因可能会产生内部输入电平，从而导致误操作。CMOS设备的操作特性与Bipolar或NMOS设备不同。CMOS设备的输入电平必须借助上拉或下拉电路固定在高电平或低电平。每一个未使用引脚都应该通过附加电阻连接到V_{DD}或GND。如果有可能尽量定义为输出引脚。对未使用引脚的处理因设备而异，必须遵循与设备相关的规定和说明。

③ ESD防护措施

如果MOS设备周围有强电场，将会击穿氧化栅极，从而影响设备的运行。因此必须采取措施，尽可能防止静电产生。一旦有静电，必须立即释放。对于环境必须有适当的控制。如果空气干燥，应当使用增湿器。建议避免使用容易产生静电的绝缘体。半导体设备的存放和运输必须使用抗静电容器、抗静电屏蔽袋或导电材料容器。所有的测试和测量工具包括工作台和工作面必须良好接地。操作员应当佩戴静电消除手带以保证良好接地。不能用手直接接触半导体设备。对于装配有半导体设备的PW板也应采取类似的静电防范措施。

④ 初始化之前的状态

在上电时MOS设备的初始状态是不确定的。在刚刚上电之后，具有复位功能的MOS设备并没有被初始化。因此上电不能保证输出引脚的电平，I/O设置和寄存器的内容。设备在收到复位信号后才进行初始化。具有复位功能的设备在上电后必须立即进行复位操作。

⑤ 电源开关顺序

在一个设备的内部操作和外部接口使用不同的电源的情况下，按照规定，应先在接通内部电源之后再接通外部电源。当关闭电源时，按照规定，先关闭外部电源再关闭内部电源。如果电源开关顺序颠倒，可能会导致设备的内部组件过电压，产生异常电流，从而引起内部组件的误操作和性能的退化。

对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

⑥ 电源关闭状态下的输入信号

不要向没有加电的设备输入信号或提供I/O上拉电源。因为输入信号或提供I/O上拉电源将引起电流注入，从而引起设备的误操作，并产生异常电流，从而使内部组件退化。

每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断。

- **本文档所登载的内容有效期截止至 2008 年 2 月，信息先于产品的生产周期发布。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表或数据手册等相关资料以获取本公司产品的最新规格。**
- 并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。否则因本文档所登载内容引发的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权做出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：

  “标准等级”、“专业等级”以及“特殊等级”三种质量等级。

  “特殊等级”仅适用于为特定用途而根据用户指定的质量保证程序所开发的日电电子产品。另外，各种日电电子产品的推荐用途取决于其质量等级，详见如下。用户在选用本公司的产品时，请事先确认产品的质量等级。

  “标准等级”： 计算机，办公自动化设备，通信设备，测试和测量设备，音频·视频设备，家电，加工机械以及产业用机器人。

  “专业等级”： 运输设备（汽车、火车、船舶等），交通用信号控制设备，防灾装置，防止犯罪装置，各种安全装置以及医疗设备（不包括专门为维持生命而设计的设备）。

  “特殊等级： 航空器械，宇航设备，海底中继设备，原子能控制系统，为了维持生命的医疗设备、用于维持生命的装置或系统等。

  除在本公司半导体产品的数据表或数据手册等资料中另有特别规定以外，本公司半导体产品的质量等级均为“标准等级”。如果用户希望在本公司设计意图以外使用本公司半导体产品，务必事先与本公司销售代表联系以确认本公司是否同意为该项应用提供支持。

（注）

（1） 本声明中的“本公司”是指日本电气电子株式会社（NEC Electronics Corporation）及其控股公司

（2） 本声明中的“本公司产品”是指所有由日本电气电子株式会社开发或制造的产品或为日本电气电子株式会社（定义如上）开发或制造的产品。

M5 02.11-1

前言

目标读者

这个应用笔记是写给那些希望理解 V850ES/Jx2 的功能以及使用这个产品设计应用系统的用户。

目的

这个应用笔记的目的是帮助用户理解如何开发专用的 flash 编程器来重新写入 V850ES/Jx2 的内部 flash 存储器。
这个文档中的例程和电路图仅供参考，并且不是用于实际设计中的使用。
因此，这些例程必须在用户自己的风险下使用。如果这些例程被使用，不保证一定能正确工作。

结构

这个应用笔记由以下章节组成。

- Flash 存储器编程
- 编程器工作环境
- 基本编程器操作
- 命令/数据帧格式
- 命令处理的说明
- UART 通信模式
- 正常握手的 3 线串行输入 / 输出通信模式（CSI + HS）
- 3 线串行输入 / 输出通信模式（CSI）
- Flash 存储器编程参数特性
- 电气规范（参考）

如何阅读这个应用笔记

我们认为这个应用笔记的读者具有电气工程、逻辑电路和微控制器的基本知识。

☐ 要学习更多的 V850ES/Jx2 微控制器硬件功能：
→ 参见每个 V850ES/Jx2 产品的用户用户手册。

约定

数据重要性：	高位数字在左边，低位数字在右边
低有效表示法：	$\overline{xxx}$ （管脚和信号名上的线）
注：	文中用**注**标记的项目的脚注
注意事项：	需要特别注意的信息
备注：	增补的信息
数字表示法：	二进制 .....xxxx 或 xxxxB
	十进制 xxxx
	十六进制.....xxxxH

相关文档　　这个文档中表示的相关文档可能包含初稿版本。然而，初稿版本没有被标志出来。

器件相关文档

文档名	文档号
V850ES/JG2 用户手册	U17715E
V850ES/JJ2 用户手册	U17714E
V850ES 体系结构用户手册	U15943E

目录

第 1 章 FLASH存储器编程

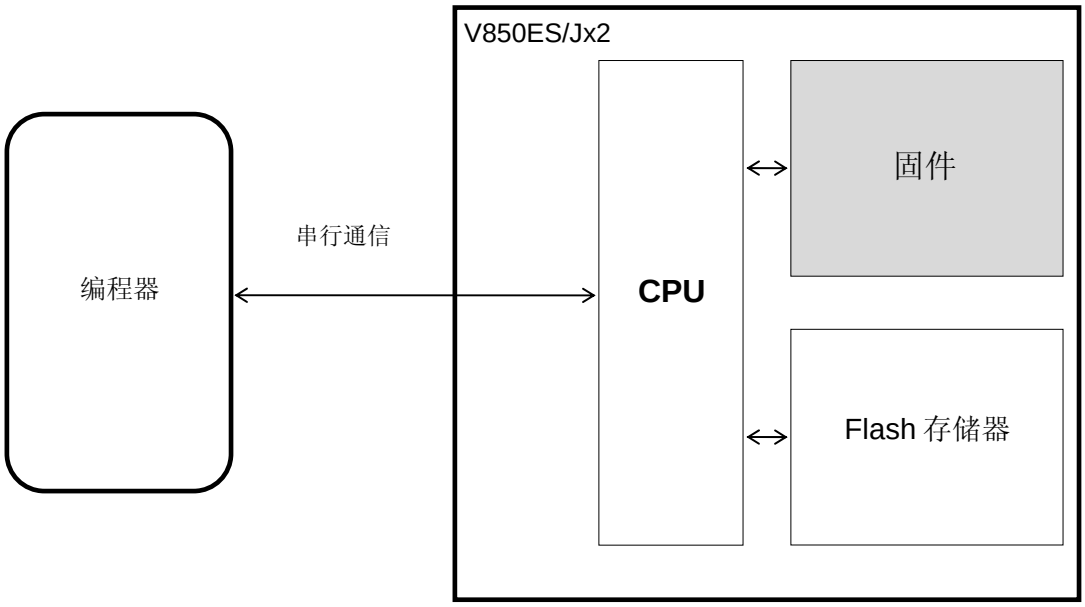
要改写 V850ES/Jx2 的内部 flash 存储器的内容，一般使用一个专用的 flash 存储器编程器（文中简称“编程器”）。

这个应用笔记解释如何开发一个专用的编程器。

1.1 综述

V850ES/Jx2 集成控制 flash 存储器编程的固件。通过串行通信在编程器和 V850ES/Jx2 之间发送 / 接收命令，内部 flash 存储器的编程被执行。

图 1-1. V850ES/Jx2 中 Flash 存储器编程的系统结构

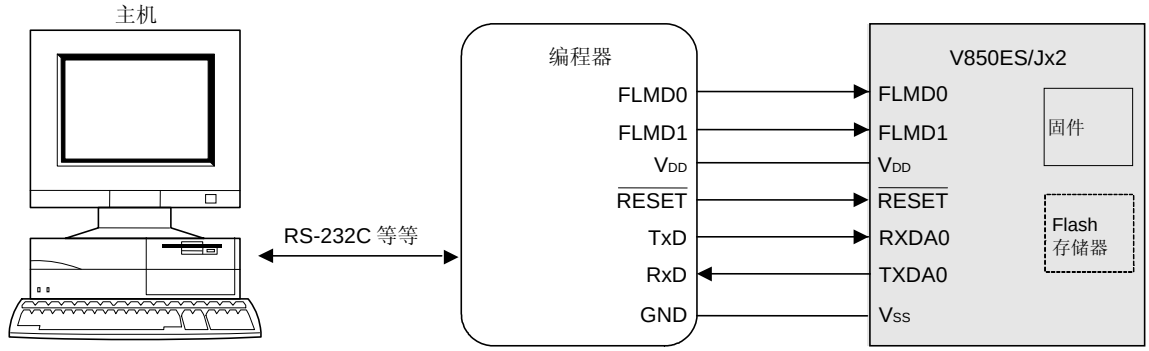


1.2 系统配置

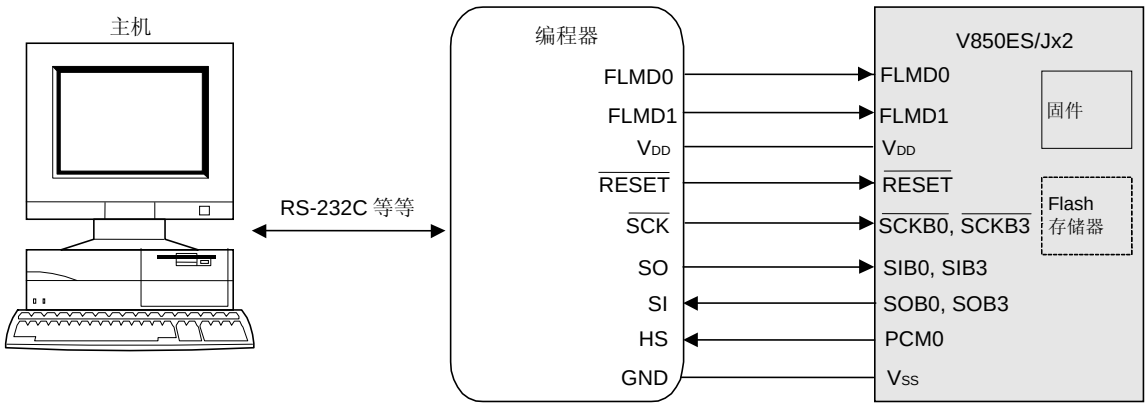
编程 flash 存储器的系统配置的例子参见图 1-2 中。
这些图表示在一台主机的控制下使用编程器如何编程 flash 存储器。
根据编程器如何被连接，如果用户程序已经事先被下载到编程器中，编程器可以在单独模式下使用，而不用主机。
例如，日电电子的 flash 存储器编程器 PG-FP4 可以通过连接的主机的 GUI 软件或者通过它自己（单独）进行编程。

图 1-2. 系统配置

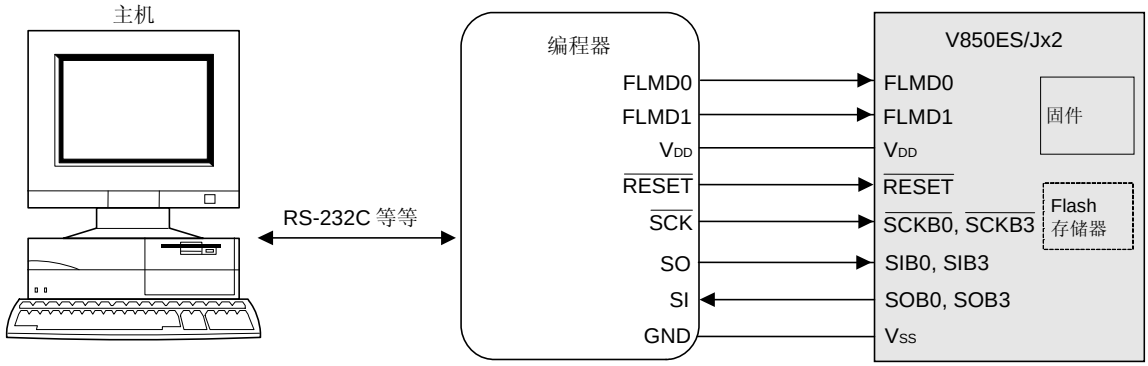
（1）单线 UART 通信模式（LSB 首先发送）



（2）支持握手的 3 线串行输入 / 输出通信模式（CSI + HS）（MSB 首先发送）



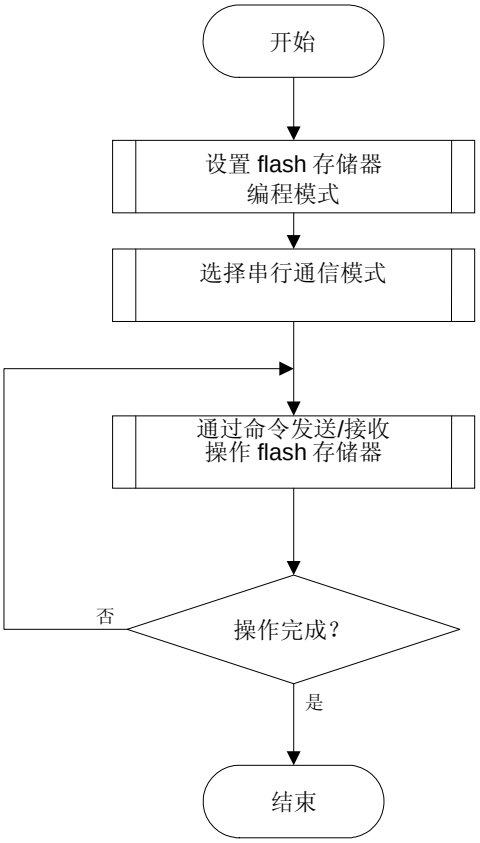
（3）支持握手的 3 线串行输入 / 输出通信模式（CSI）（MSB 首先发送）



1.3 编程综述

要使用编程器重新写入 flash 存储器的内容，V850ES/Jx2 必须首先被设置为 flash 存储器编程模式。然后，选择编程器和 V850ES/Jx2 之间的通信模式，通过串行通信从编程器发送命令并且重新写 flash 存储器。编程的流程图如图 1-3 所示。

图 1-3. 编程流程图



1.3.1 设置flash存储编程模式

提供一个特定的电压到 V850ES/Jx2 中的 flash 存储器编程模式设定管脚（FLMD0 和 FLMD1）并释放复位，然后，flash 存储器编程模式被设置。

1.3.2 选择串行通信模式

要选择一个串行通信模式，在 flash 存储器编程模式下通过在 V_{DD} 电压和 GND 电压之间更改 flash 存储器编程模式设置管脚（FLMD0）上的电压来产生脉冲，并且根据脉冲个数确定通信模式。

1.3.3 通过命令发送 / 接收操作flash存储器

集成在 V850ES/Jx2 中的 flash 存储器有重新写入 flash 存储器内容的功能。表 1-1 中显示的 flash 存储器操作功能可以使用。

表 1-1. Flash 存储器功能要点

功能	要点
擦除	擦除 flash 存储器内容。
写入	写入数据到 flash 存储器。
校验	比较 flash 存储器内容和校验的数据。
信息获取	读取 flash 存储器相关的信息。

要控制这些功能，编程器通过串行通信发送命令到 V850ES/Jx2。V850ES/Jx2 返回命令的响应状态。flash 存储器编程通过重复串行通信的一系列命令被执行。

1.4 V850ES/Jx2 的特有信息

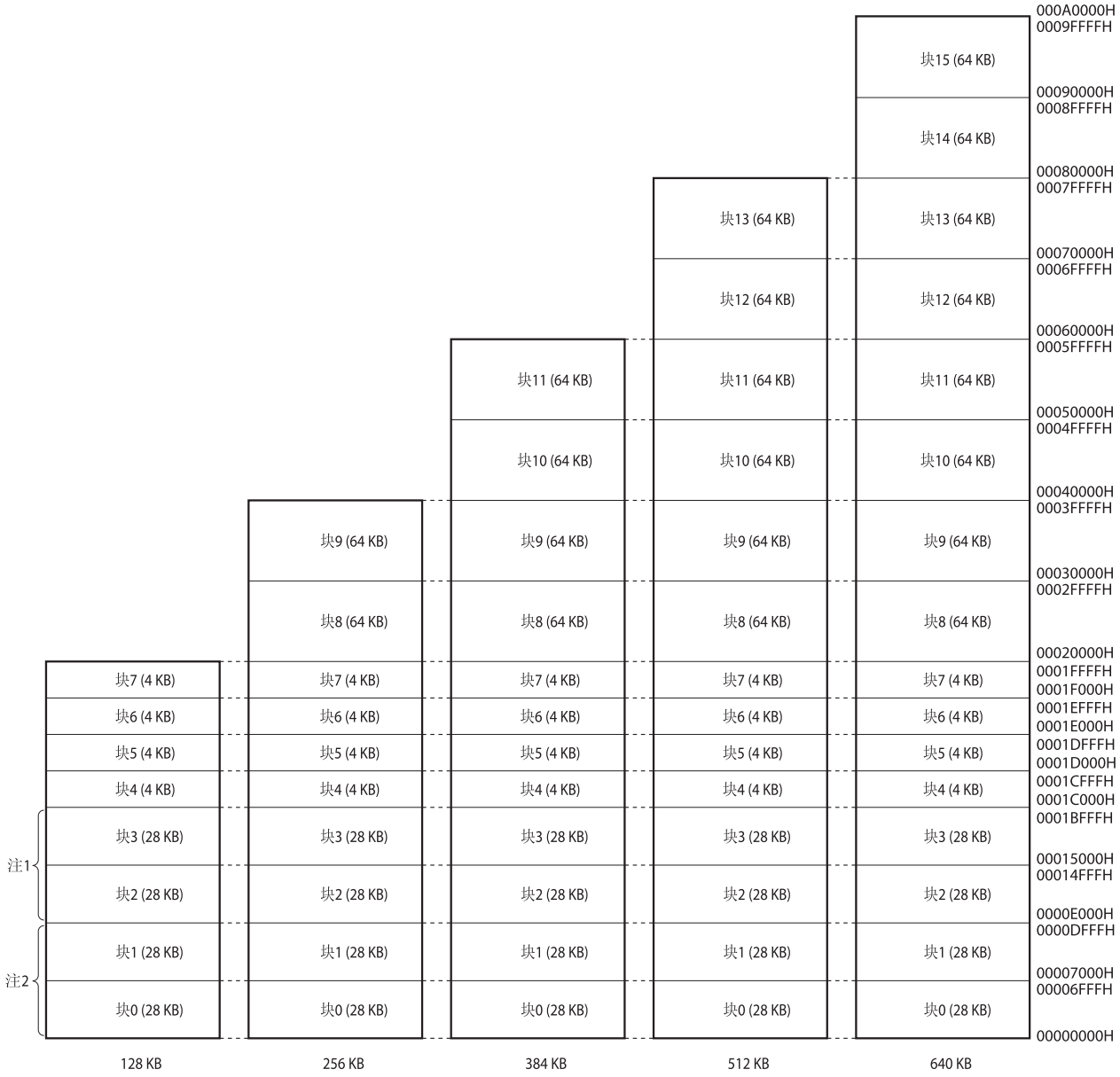
编程器必须管理产品特有信息（例如设备名和存储器信息）。

表 1-2 表示 V850ES/Jx2 存储器大小，图 1-4 表示 flash 存储器的结构。

表 1-2. V850ES/Jx2 的 Flash 存储器大小

设备名		Flash 存储器大小
V850ES/JG2	μPD70F3715	128 KB
	μPD70F3716	256 KB
	μPD70F3717	384 KB
	μPD70F3718	512 KB
	μPD70F3719	640 KB
V850ES/JJ2	μPD70F3720	128 KB
	μPD70F3721	256 KB
	μPD70F3722	384 KB
	μPD70F3723	512 KB
	μPD70F3724	640 KB

图 1-4. Flash 存储器结构



注 **1.** 块 2 和 3：通过引导交换功能和引导区域进行交换的区域 。

2. 块 0 和 1：引导区域

第 2 章 编程器工作环境

2.1 编程器控制管脚

表 2-1 列表在用户系统中要实现编程器功能时，编程器必须控制的管脚。关于每个管脚的细节，见下一页。

表 2-1. 管脚说明

编程器			V850ES/Jx2	与目标系统的通信模式		
信号名	输入 / 输出	管脚功能	管脚名	CSI	CSI + HS	UART
FLMD0	输出	设置编程模式的信号的输出以及选择通信模式的脉冲的输出	FLMD0	○	○	○
FLMD1	输出	设置编程模式的信号的输出	FLMD1	○	○	○
V_{DD}	输出	V_{DD} 电压产生/监视	V_{DD}	△	△	△
GND	–	地	V_{SS}	○	○	○
CLK	输出	输出到 V850ES/Jx2 的工作时钟	–	×[注]	×[注]	×[注]
$\overline{\text{RESET}}$	输出	编程模式切换触发	$\overline{\text{RESET}}$	○	○	○
SO	输出	到 V850ES/Jx2 的命令发送	SIB0，SIB3	○	○	×
SI	输入	从 V850ES/Jx2 的响应状态和数据接收	SOB0，SOB3	○	○	×
$\overline{\text{SCK}}$	输出	提供给 V850ES/Jx2 的串行时钟	$\overline{\text{SCKB0}}$，$\overline{\text{SCKB3}}$	○	○	×
HS （握手）	输入	与 V850ES/Jx2 串行通信的握手信号接收	PCM0	×	○	×
TxD	输出	到 V850ES/Jx2 的命令发送	RXDA0	×	×	○
RxD	输入	从 V850ES/Jx2 的响应状态和数据接收	TXDA0	×	×	○

注 编程器的 CLK 管脚不提供时钟。安装一个振荡器电路到目标系统来提供时钟。

备注 ○：确认连接这个管脚。
×： 这个管脚不连接。
△：如果信号在用户系统被产生，这个管脚不必被连接。

关于编程器控制的管脚的电压，参阅使用 flash 存储器编程的设备的用户手册。

2.2 控制管脚的细节

2.2.1 Flash存储器编程模式设置管脚（FLMD0，FLMD1）

FLMD0 和 FLMD1 管脚被用来控制 V850ES/Jx2 的工作模式。当特定的电压被提供到这些管脚并且复位被释放时，V850ES/Jx2 工作于 flash 存储器编程模式。

复位后，编程器和 V850ES/Jx2 之间的串行通信模式通过在 V_{DD} 和 GND 之间控制 FLMD0 管脚的电压并输出脉冲来确定。关于 FLMD0 脉冲个数和通信模式之间的关系，参阅 **2.5 选择串行通信模式**中的**表 2-3**。

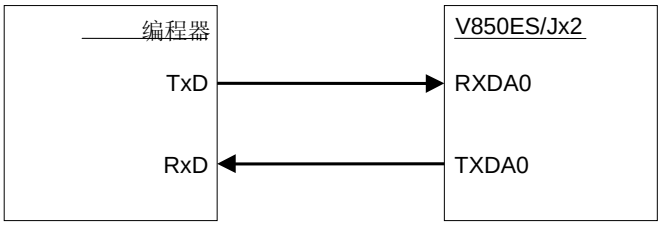
2.2.2 串行接口管脚（TxD，RxD，SI，SO，$\overline{SCK}$，HS）

串行接口管脚被用于在编程器和 V850ES/Jx2 之间发送 flash 存储器写入命令。

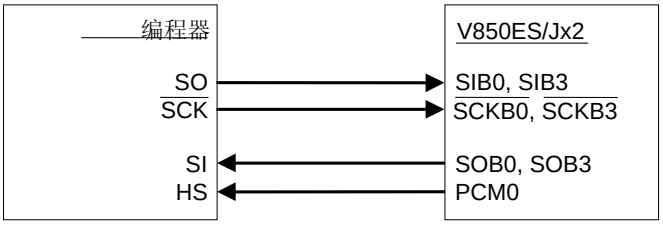
对于 V850ES/Jx2，通信模式可以从 UART、CSI + HS 和 CSI 中选择。以下图表明每种通信模式中的管脚的连接。

图 2-1. 串行接口管脚

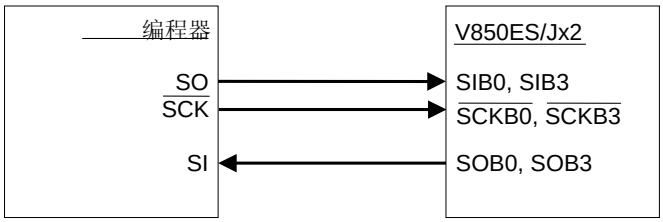
（1）UART 通信模式



（2）支持握手的 3 线串行输入 / 输出通信模式（CSI + HS）



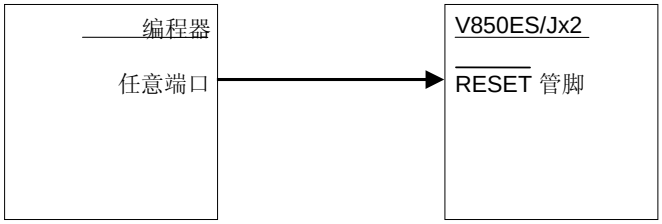
（3）3 线串行输入 / 输出通信模式（CSI）



2.2.3 复位控制管脚 （$\overline{\text{RESET}}$）

复位控制管脚被用于从编程器控制 V850ES/Jx2 的系统复位。当特定的电压被提供到 FLMD0 和 FLMD1 管脚并且复位被释放时，flash 存储器编程模式可以被选择。

图 2-2. RESET 控制管脚



2.2.4 时钟控制管脚（CLK）

时钟控制管脚不被使用。

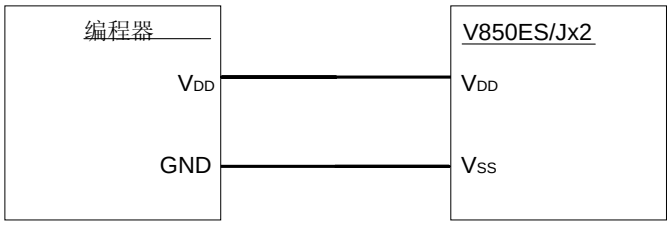
没有时钟可以从编程器的 CLK 管脚被提供。安装一个振荡器电路到目标系统来提供时钟。

2.2.5 V_{DD}/GND 控制管脚

V_{DD} 控制管脚被用于从编程器向 V850ES/Jx2 提供电源。当不需要从编程器向 V850ES/Jx2 提供电源时，这个管脚不需要连接。然而，当专用的编程器被使用时，不论是否从编程器提供电源，这个管脚必须被连接，因为专用的编程器监视 V850ES/Jx2 的电源状态。

不论是否从编程器提供电源，GND 控制管脚必须被连接到 V850ES/Jx2 的 V_{SS}。

图 2-3. V_{DD}/地 控制管脚



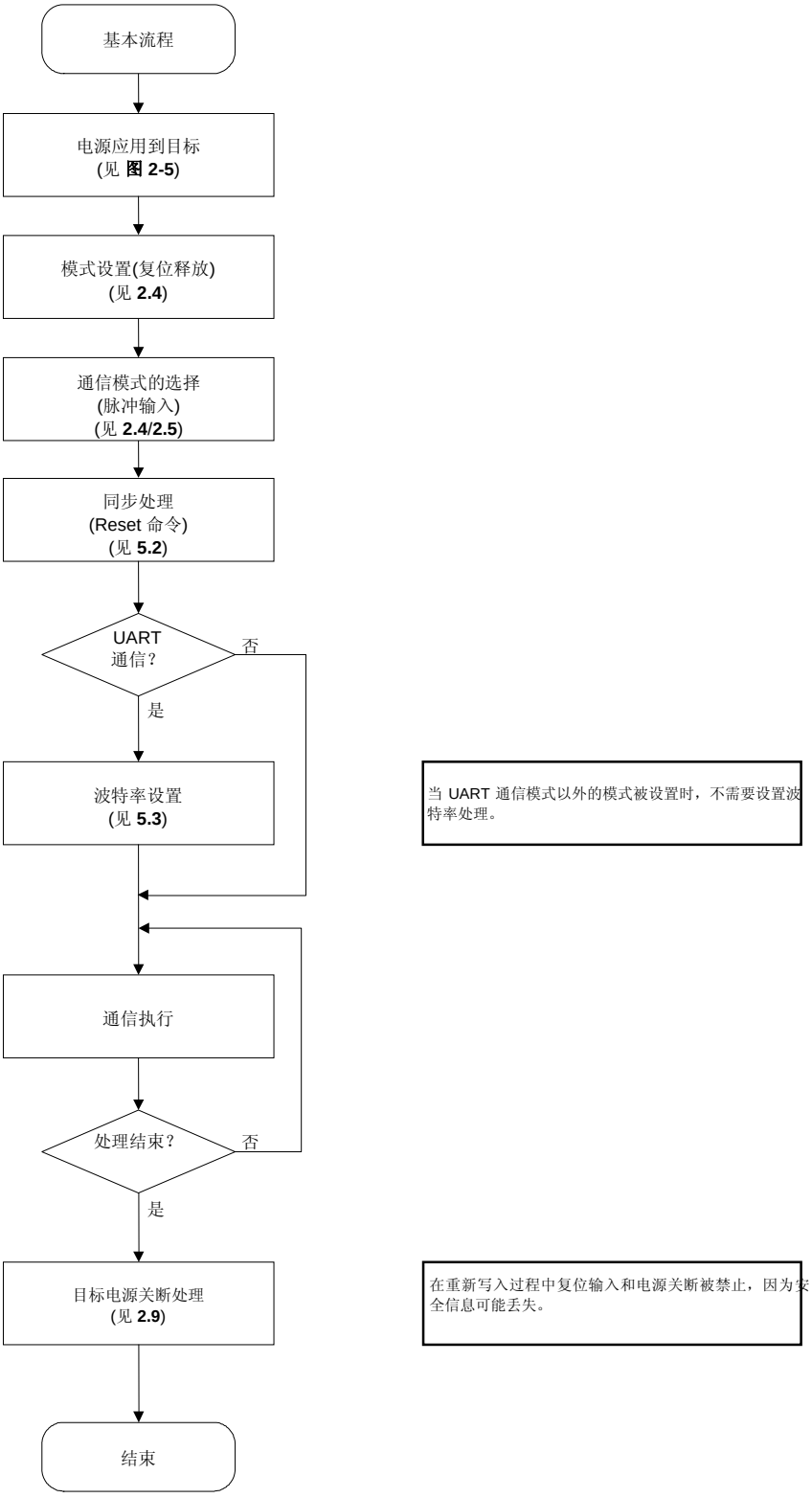
2.2.6 其它管脚

有关没有连接到编程器的管脚的连接，参阅每个设备用户手册中描述 flash 存储器的章节。

2.3 基本流程图

以下显示使用编程器执行 flash 存储器重新写入的基本流程图。

图 2-4. Flash 存储器重新写入处理的基本流程图

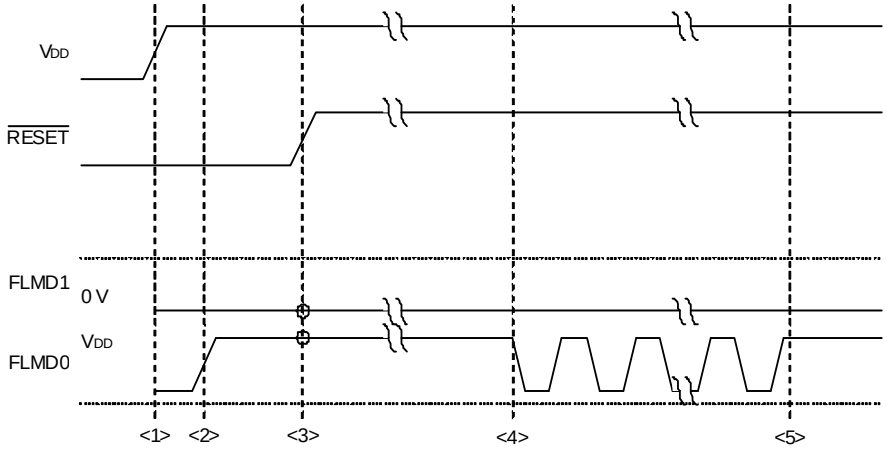


2.4 设置Flash存储器编程模式

要使用编程器重新写入 flash 存储器内容，必须首先向 V850ES/Jx2 中的 flash 存储器编程模式设置管脚（FLMD0，FLMD1）提供一个特定的电压并释放复位，将 V850ES/Jx2 设置为 flash 存储器编程模式。

以下表示设置 flash 存储器编程模式和选择通信模式的时序图。

图 2-5. 设置 Flash 存储器编程模式和选择通信模式



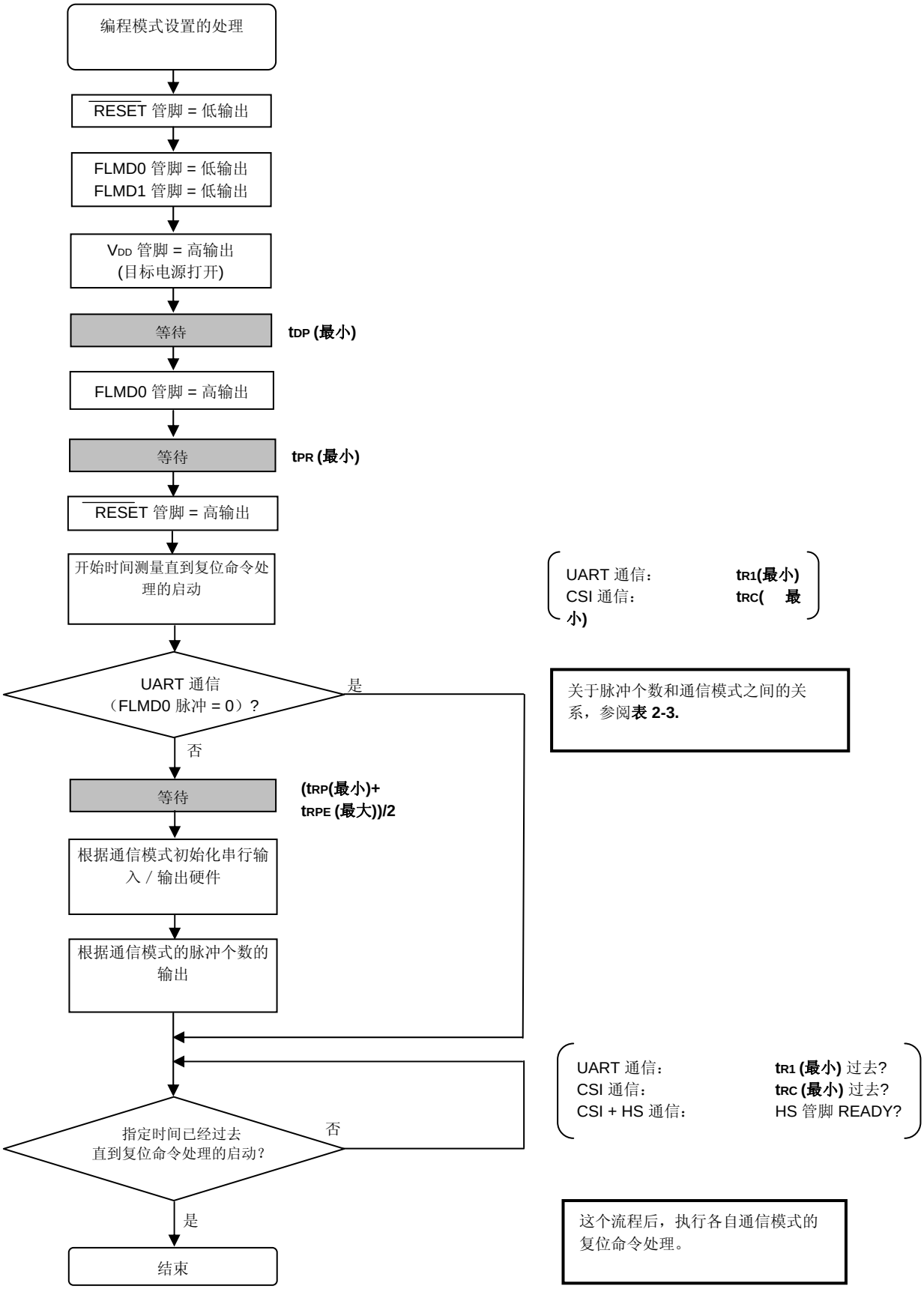
<1>: 电源应用(V_{DD})
<2>: FLMD0 = 高电平，FLMD1 = 低电平
<3>: 复位释放(模式设置)
<4>: 脉冲输出启动
<5>: 脉冲输出结束

复位释放后 FLMD0 和 FLMD1 管脚的设置和工作模式之间的关系如下所示。

表 2-2. 复位释放后 FLMD0 和 FLMD1 管脚的设置和工作模式之间的关系

FLMD0	FLMD1	工作模式
低（GND）	任意	正常工作模式
高（V_{DD}）	低（GND）	Flash 存储器编程模式
高（V_{DD}）	高（V_{DD}）	禁止设置

2.4.1 模式设置流程图


编程模式设置的处理
RESET 管脚 = 低输出
FLMD0 管脚 = 低输出
FLMD1 管脚 = 低输出
VDD 管脚 = 高输出
(目标电源打开)
等待
tDP (最小)
FLMD0 管脚 = 高输出
等待
tPR (最小)
RESET 管脚 = 高输出
开始时间测量直到复位命令处理的启动
UART 通信：tR1(最小)
CSI 通信：tRC(最小)
UART 通信
（FLMD0 脉冲 = 0）?
是
否
关于脉冲个数和通信模式之间的关系，参阅表 2-3.
等待
(tRP(最小)+ tRPE (最大))/2
根据通信模式初始化串行输入 / 输出硬件
根据通信模式的脉冲个数的输出
UART 通信：tR1 (最小) 过去?
CSI 通信：tRC (最小) 过去?
CSI + HS 通信：HS 管脚 READY?
指定时间已经过去
直到复位命令处理的启动?
否
是
这个流程后，执行各自通信模式的复位命令处理。
结束

2.4.2 样本程序

以下表示模式设置的一个样本程序。

```
/************************************************************/
/*                                                          */
/*   /* 连接到Flash设备                                    */
/*                                                          */
/************************************************************/
void
fl_con_dev（void）
{
extern    void    init_fl_uart（void）;          // [v1.01]
extern    void    init_fl_csi（void）;           // [v1.01]

          int     n;
          int     pulse;

          SRMK0 = true;
          UARTE0 = false;

          switch （fl_if）{
            default：
            case    FLIF_UART：   pulse = PULSE_UART;     break;
            case    FLIF_CSI：    pulse = UseCSIB3 ? PULSE_CSIB3 ： PULSE_CSI; break;
            case    FLIF_CSI_HS： pulse = UseCSIB3 ? PULSE_CSIB3HS：PULSE_CSIHS;     break;

          }

          pFL_RES       = low;                  // RESET/FLMD0 = 低
          pmFL_FLMD0 = PM_OUT;                  // FLMD0 = 低输出        //[v1.01]
          pFL_FLMD0       = low;
          pmFL_FLMD1 = PM_OUT;                  // FLMD1 = 低输出        //[v1.01]
          pFL_FLMD1       = low;
          FL_VDD_HI（）;                        // VDD = 高

          fl_wait（tDP）;                       // 等待

          pFL_FLMD0   = hi;                     // FLMD0 = 高
          fl_wait（tPR）;                       // 等待

          pFL_RES       = hi;                   // RESET = 高
          start_flto（tRC）;                    // 启动“tRC”等待定时器
          fl_wait（（tRP+tRPE）/2）;            // 等待

          if （fl_if == FLIF_UART）{
            init_fl_uart（）;                   // 初始化 UART 硬件（用于 Flash 设备控制）
                UARTE0 = true;
                SRIF0 = false;
                SRMK0 = false;
          }
          else{
              init_fl_csi（）;                  // 初始化 CSI 硬件
          }
          for （n = 0; n < pulse; n++）{        // 脉冲输出

           pFL_FLMD0 = low;
```

```
        fl_wait（tPW）;
        pFL_FLMD0 = hi;
        fl_wait（tPW）;
      }

      while（!check_flto（））          // 超时 tRC?          ;          // 无

      // 启动 RESET 命令处理
}
```

2.5 选择串行通信模式

通信模式通过在复位释放后输入一个脉冲到 V850ES/Jx2 中的 FLMD0 管脚来决定。
FLMD0 脉冲的高和低电平分别是 V_{DD} 和 GND。
下表表示 FLMD0 脉冲的个数（脉冲个数）和 V850ES/Jx2 可以选择的通信模式之间的关系。

表 2-3. FLMD0 脉冲个数和通信模式之间的关系

通信模式	FLMD0 脉冲个数	通信使用的端口
UART （UART0）	0	TXDA0 （P30），RXDA0 （P31）
3 线串行输入 / 输出（CSIB0）	8	SOB0 （P41），SIB0 （P40），$\overline{\text{SCKB0}}$ （P42）
3 线串行输入 / 输出（CSIB3）	9	SOB3 （P911），SIB3 （P910），$\overline{\text{SCKB3}}$ （P912）
支持握手的 3 线串行输入 / 输出通信模式（CSIB0 + HS）	11	SOB0 （P41），SIB0 （P40），$\overline{\text{SCKB0}}$ （P42），HS （PCM0）
支持握手的 3 线串行输入 / 输出通信模式（CSIB3 + HS）	12	SOB3 （P911），SIB3 （P910），$\overline{\text{SCKB3}}$ （P912），HS （PCM0）
禁止设置	其它	–

2.6 UART通信模式

RxD 和 TxD 管脚被用作 UART 通信。通信条件如下所示。

表 2-4. UART 通信条件

项目	说明
波特率	从 9,600，19,200，31,250，38,400，76,800 和 153,600 bps 中选择 （默认：9,600 bps）
奇偶位	无
数据长度	8 位 （LSB 在先）
停止位	1 位

在 CSI 通信过程中，编程器总是作为主设备，所以编程器必须检查 V850ES/Jx2 的处理，比如写入或擦除，被正常完成。另外，主和从的状态在 UART 通信过程中会偶尔交换，所以，不指定类似 CSI + HS 通信的一个管脚时的通信可能是最优通信时间。

注意事项　当执行 UART 通信时，为主和从设备设置相同的波特率。

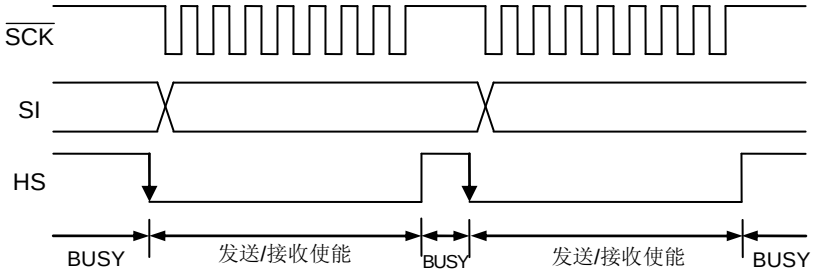
2.7 支持握手的 3 线串行输入 / 输出通信模式（CSI + HS）

在 CSI + HS 通信模式中，命令或数据的通信时序被优化。除 SI、SO 和 $\overline{SCK}$ 管脚外，HS（握手）管脚被用来实现有效通信。

当 V850ES/Jx2 准备好发送或接收数据时，HS 管脚的电平降低（低电平）。编程器在开始发送 / 接收到 V850ES/Jx2 的命令或数据前必须检查 HS 管脚信号的下降沿（低电平）。

通信数据格式时 MSB 在前，以 8 位一个单元。保持时钟频率为 2.5 MHz 或更低。

图 2-6. CSI + HS 通信的时序图



2.8 3 线串行输入 / 输出通信模式（CSI）

$\overline{SCK}$、SO 和 SI 管脚被用作 CSI 通信。编程器总是作为主设备工作，所以如果当 V850ES/Jx2 没有准备好发送/接收时数据通过 SCK 管脚被发送，通信可能不会被正常执行。

通信数据格式时 MSB 在前，以 8 位一个单元。保持时钟频率为 2.5 MHz 或更低。

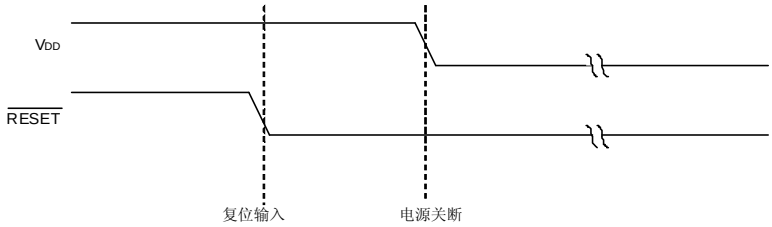
2.9 关断目标电源

在每个命令执行完成后，在设置 $\overline{RESET}$ 管脚到低电平后，关断目标的电源，如下所示。

当关断目标的电源时，设置其它管脚为高阻。

注意事项　在命令处理过程中，关断电源和输入复位被禁止。

图 2-7. 终止 Flash 存储器编程模式的时序



2.10 Flash存储器的操作

V850ES/Jx2 中集成的 flash 存储器有操作 flash 存储器的功能，如表 2-5 所示。 编程器发送命令到 V850ES/Jx2 来控制这些功能，并且检查从 V850ES/Jx2 发出的响应状态，来操作 flash 存储器。

表 2-5. Flash 存储器操作功能列表

分类	功能名	说明
擦除	片擦除	擦除整个 flash 存储器区域。清除安全标志。
	块擦除	擦除 flash 存储器中的指定块。
写入	写入	写入数据到 flash 存储器中的指定区域。
Verify	Verify	在 V850ES/Jx2 端，比较从 flash 存储器中指定地址获取的数据和从编程器发送的数据。
空白检查	块空白检查	检查 flash 存储器中指定区域的擦除状态。
信息获取	Silicon 签名获取	获取写入协议信息。
	版本获取	获取 V850ES/Jx2 和固件的版本信息。
	状态获取	获取当前的工作状态。
	校验和获取	获取指定区域的校验和数据。
安全	安全设置	设置安全信息。
其它	复位	删除通信中的同步。

2.11 命令列表

编程器使用的命令和它们的功能被列表如下。

表 2-6. 从编程器发送到 V850ES/Jx2 的命令列表

命令号	命令名	功能
70H	Status	获取当前的工作状态（状态数据）。
00H	Reset	删除通信中的同步。
90H	Oscillating Frequency Set	指定 V850ES/Jx2 的振荡频率。
9AH	Baud Rate Set	当 UART 通信模式被选择时，设置波特率。
20H	Chip Erase	擦除整个 flash 存储器区域。
22H	Block Erase	擦除 flash 存储器中的指定区域。
40H	Programming	写入数据到 flash 存储器中的指定区域。
13H	Verify	比较从 flash 存储器中指定区域的内容和从编程器发送的数据。
32H	Block Blank Check	检查 flash 存储器中指定区域的擦除状态。
C0H	Silicon Signature	获取 V850ES/Jx2 信息（部件号，flash 存储器配置，等等）。
C5H	Version Get	获取 V850ES/Jx2 和固件的版本信息。
B0H	Checksum	获取指定区域的校验和数据。
A0H	Security Set	设置安全信息。

2.12 状态列表

下面的表列表编程器从 V850ES/Jx2 接收的状态码。

表 2-7. 状态码列表

状态码	状态	说明
04H	命令号错误	如果一个不支持的命令被接收返回的错误
05H	参数错误	如果命令信息（参数）无效返回的错误
06H	正常响应（ACK）	正常响应
07H	校验和错误	如果从编程器发送的一帧中的数据异常返回的错误
08H	WWV1 错误	写入错误
0BH	EWV1 错误	擦除错误
0CH	EWV2 1 错误	擦除错误
0DH	EWV3 错误	擦除错误
0EH	校验错误	从编程器发送的帧的数据的校验错误发生。
0FH	校验错误	从编程器发送的帧的数据的校验错误发生。
10H	保护错误	试图执行被 Security Set 命令禁止的处理时返回的错误
11H	EWV4 错误	内部校验错误/空白错误
13H	压缩搜索错误	擦除错误
15H	消极响应（NACK）	消极响应
16H	程序错误	如果一个错误发生在 flash 控制宏中返回的错误
FFH	处理进行中（BUSY）	忙响应[注]

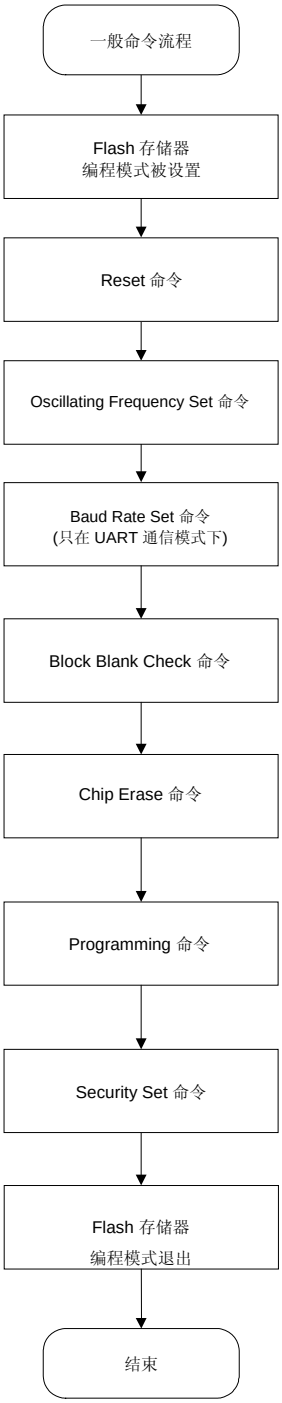
注 在 CSI 通信过程中，1 字节“FFH”可能被发送，“FFH”也作为数据帧格式。

在这个手册中，校验和错误或 NACK 的接收被认为一个立即异常结束。然而，当一个专用编程器被开发时，从导致校验和错误或 NACK 的命令的刚刚发送的等待，处理可能被重试。在这种情况下，建议限制重试次数来避免重试操作的无限重复。

尽管没有在上表列出，如果一个超时错误（在 UART 通信过程中的 BUSY 超时、HS 管脚超时或数据帧接收超时）发生，建议关断 V850ES/Jx2 的电源（参阅 **2.9 关断目标电源**）并且然后再次连接电源。

图 3-1 显示当使用编程器执行 flash 存储器重新写入时一般的命令执行流程。

图 3-1. Flash 存储器重新写入时的一般命令执行流程



注 建议执行安全设置来使读取无效作为编程器规范中的默认设置。

备注 Verify 命令和 Checksum 命令也可以被支持。

第 4 章 命令 / 数据帧格式

编程器使用命令帧来发送命令到 V850ES/Jx2。V850ES/Jx2 使用数据帧来发送写入数据或校验数据到编程器。头、脚、数据长度信息和校验和被附加到每帧来提高发送数据的可靠性。

以下表示命令帧和数据帧的格式。

图 4-1. 命令帧格式

SOH （1 字节）	LEN （1 字节）	COM （1 字节）	命令信息（可变长度） （最大 255 字节）	SUM （1 字节）	ETX （1 字节）

图 4-2. 数据帧格式

STX （1 字节）	LEN （1 字节）	数据（可变长度） （最大 256 字节）	SUM （1 字节）	ETX 或 ETB （1 字节）

表 4-1. 每帧中符号的说明

符号	值	说明
SOH	01H	命令帧头
STX	02H	数据帧头
LEN	–	数据长度信息（00H 表示 256）。 命令帧：　　COM + 命令信息长度 数据帧：　　数据区域长度
COM	–	命令号
SUM	–	一帧的校验和数据 通过从初始值（00H）中以 1 字节为单位按顺序减去所有计算目标数据来获得（借位被忽略）。计算目标如下所示。 命令帧：　　LEN + COM + 所有命令信息 数据帧：　　LEN + 所有数据
ETB	17H	除最后一帧以外的数据帧的脚
ETX	03H	命令帧脚或者最后数据帧的脚

以下表示对一帧计算校验和（SUM）的例子。

［命令帧］

命令信息没有被包含在下面状态命令帧的例子中，所以 LEN 和 COM 是校验和计算的目标。

SOH	LEN	COM	SUM	ETX
01H	01H	70H	校验和	03H
	校验和计算目标			

对这个命令帧，按照以下方法获得校验和数据。

00H （初始值） – 01H （LEN） – 70H （COM） = 8FH （借位被忽略。只有低 8 位。）

最终发送的命令帧如下所示。

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

［数据帧］

要发送如下所示的数据帧，LEN 和 D1 到 D4 是校验和计算目标。

STX	LEN	D1	D2	D3	D4	SUM	ETX
02H	04H	FFH	80H	40H	22H	校验和	03H
	校验和计算目标						

对这个数据帧，按照以下方法获得校验和。

00H （初始值） – 04H （LEN） – FFH （D1） – 80H （D2） – 40H （D3） – 22H （D4）
= 1BH （借位被忽略。只有低 8 位。）

最终发送的数据帧如下所示。

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

当一个数据帧被接收时，校验和数据以同样方式被计算，并且获得的值通过判断这个值与接收数据的 SUM 区域中保存的值是否一样被用来检测校验和错误。例如，当如下所示的一个数据帧被接收时，校验和错误被检测。

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

↑ 如果正常，应该是 1BH

4.1 命令帧发送处理

关于每种通信模式的发送命令帧的命令处理流程图，请阅读以下章。

- 对于 UART 通信模式，阅读 **6.1 命令帧发送处理的流程图。**
- 对于支持握手的 3 线串行输入 / 输出通信模式 （CSI + HS），阅读 **7.1 命令帧发送处理的流程图。**
- 对于 3 线串行输入 / 输出通信模式 （CSI），阅读 **8.1 命令帧发送处理的流程图。**

4.2 数据帧发送处理

写入数据帧（用户程序）、校验数据帧（用户程序）和安全数据帧（安全标志）被作为数据帧发送。
关于每种通信模式的发送数据帧的命令处理流程图，请阅读以下章。

- 对于 UART 通信模式，阅读 **6.2 数据帧发送处理的流程图。**
- 对于支持握手的 3 线串行输入 / 输出通信模式 （CSI + HS），阅读 **7.2 数据帧发送处理的流程图。**
- 对于 3 线串行输入 / 输出通信模式 （CSI），阅读 **8.2 数据帧发送处理的流程图。**

4.3 数据帧接收处理

状态帧、silicon 签名数据帧、版本数据帧和校验和数据帧被作为数据帧接收。
关于每种通信模式的接收数据帧的命令处理流程图，请阅读以下章。

- 对于 UART 通信模式，阅读 **6.3 数据帧接收处理的流程图。**
- 对于支持握手的 3 线串行输入 / 输出通信模式 （CSI + HS），阅读 **7.3 数据帧接收处理的流程图。**
- 对于 3 线串行输入 / 输出通信模式 （CSI），阅读 **8.3 数据帧接收处理的流程图。**

第 5 章 命令处理说明

5.1 Status 命令

5.1.1 说明

这个命令用来在每个命令，例如写入或擦除，发布后检查 V850ES/Jx2 的工作状态。

在 Status 命令执行后，如果由于通信或类似问题 Status 命令帧不能在 V850ES/Jx2 中被正常接收，在 V850ES/Jx2 中，状态设置不会执行。因此，一个忙响应（FFH）而不是状态帧可能被接收。在这种情况下，重试 Status 命令。

5.1.2 命令帧和状态帧

图 5-1 表示 Status 命令的命令帧格式，图 5-2 表示这个命令的状态帧。

图 5-1. Status 命令帧（从编程器到 V850ES/Jx2）

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

图 5-2. 对于 Status 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据			SUM	ETX
02H	n	ST1	…	STn	校验和	03H

备注 **1.** ST1 到 STn：Status #1 到 Status #n

2. 状态帧的长度根据要发送到 V850ES/Jx2 的每个命令（例如写入或擦除）而改变。

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 在 UART 通信中，Status 命令不被使用。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.4 Status 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.4 Status 命令。**

注意事项 在 UART 通信中，每个命令，例如写入或擦除，发送后，V850ES/Jx2 在特定时间内自动返回状态帧。Status 命令因此不被使用。

如果在 UART 通信中 Status 命令被发送，Command Number Error 被返回。

5.2 Reset 命令

5.2.1 说明

这个命令被用来在通信模式被设置后检查编程器和 V850ES/Jx2 之间的通信建立。

当 UART 被选择为与 V850ES/Jx2 通信的模式时，必须为编程器和 V850ES/Jx2 设置同样的波特率。然而，V850ES/Jx2 不能检测自己的工作频率，所以波特率不能被设置。通过从编程器以 9,600 bps 发送两次“00H”，测量“00H”的低电平宽度并且计算两个发送信号的平均，可以检测 V850ES/Jx2 的工作频率，因此，波特率可以被设置，从而使能通信中的同步检测。

5.2.2 命令帧和状态帧

图 5-3 表示 Reset 命令的命令帧的格式，图 5-4 表示这个命令的状态帧。

图 5-3. Reset 命令帧（从编程器到 V850ES/Jx2）

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

图 5-4. Status 命令的状态帧（从到编程器）

STX	LEN	数据	SUM	ETX
02H	1	ST1	校验和	03H

备注 ST1：同步检测结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.4 Reset 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.5 Reset 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.5 Reset 命令。**

5.3 Baud Rate Set 命令

5.3.1 说明

这个命令被用来更改 UART 的波特率（默认值：9,600 bps）。

在 Baud Rate Set 命令被执行后，Reset 命令必须被执行来检查在更改后的波特率上的同步。

Baud Rate Set 命令只有在 UART 通信模式下有效。设置波特率的数据以 1 字节数值表示。

如果在 UART 通信模式以外的模式下发送 Baud Rate Set 命令，V850ES/Jx2 忽略这个命令。

5.3.2 命令帧和状态帧

图 5-5 表示 Baud Rate Set 命令的命令帧格式，图 5-6 表示这个命令的状态帧。

图 5-5. Baud Rate Set 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息	SUM	ETX
01H	02H	9AH （Baud Rate Set）	D01	校验和	03H

备注 D01：波特率选择值

D01 Value	03H	04H	05H	06H	07H	08H
Baud rate （bps）	9600	19200	31250	38400	76800	153600

图 5-6. Baud Rate Set 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：同步检测结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.5 Baud Rate Set 命令**。
- 在支持握手的 3 线串行输入 / 输出通信模式（CSI + HS）下，Baud Rate Set 命令不被使用。
- 在 3 线串行输入 / 输出通信模式（CSI）下，Baud Rate Set 命令不被使用。

5.4 Oscillating Frequency Set命令

5.4.1 说明

这个命令用来设置 V850ES/Jx2 中的振荡频率数据。

指定实际输入到 V850ES/Jx2 的 X1 管脚的时钟的频率。

V850ES/Jx2 根据这个命令指定的时钟频率自动设置 COU 工作时钟的倍频率。因此，注意在这个命令执行前和执行后等待计算参考时钟的改变。

5.4.2 命令帧和状态帧

图 5-7 表示 Oscillating Frequency Set 命令的命令帧格式，图 5-8 表示这个命令的状态帧。

图 5-7. Oscillating Frequency Set 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息				SUM	ETX
01H	05H	90H（**Oscillating Frequency Set**）	D01	D02	D03	D04	校验和	03H

备注 D01 到 D04：振荡频率 = （D01 × 0.1 + D02 × 0.01 + D03 × 0.001） × 10^{D04} （单位：kHz）

设置可以从 10 kHz 到 100 MHz，但是当实际发送命令时，根据每个器件的规范设置数值。

D01 到 D03 保存未打包的 BCDs，D04 保存一个有符号的整数。

设置举例： 要设置 6 MHz

D01 = 06H

D02 = 00H

D03 = 00H

D04 = 04H

振荡频率 = 6 × 0.1 × 10^4 = 6,000 kHz = 6 MHz

设置举例： 要设置 10 MHz

D01 = 01H

D02 = 00H

D03 = 00H

D04 = 05H

振荡频率 = 1 × 0.1 × 10^5 = 10,000 kHz = 10 MHz

图 5-8. Oscillating Frequency Set 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：振荡频率设置结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.6 Oscillating Frequency Set 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.6 Oscillating Frequency Set 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.6 Oscillating Frequency Set 命令。**

5.5 Chip Erase 命令

5.5.1 说明

这个命令被用来擦除 flash 存储器的整个内容。此外，只要擦除没有被安全设置禁止，由安全设置处理设置的所有信息都会被片擦除处理初始化（见 **5.13 Security Set 命令**）。

5.5.2 命令帧和状态帧

图 5-9 表示 Chip Erase 命令的命令帧的格式，图 5-10 表示这个命令的状态帧。

图 5-9. Chip Erase 命令帧（从编程器到 V850ES/Jx2）

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

图 5-10. Chip Erase 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：片擦除结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.7 Chip Erase 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.7 Chip Erase 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.7 Chip Erase 命令。**

5.6 Block Erase命令

5.6.1 说明

只要擦除没有被安全设置禁止（见 **5.13 Security Set 命令**），这个命令用来擦除 flash 存储器中指定号的块的内容。

5.6.2 命令帧和状态帧

图 5-11 表示 Block Erase 命令的命令帧格式，图 5-12 表示这个命令的状态帧。

图 5-11. Block Erase 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息	SUM	ETX
01H	02H	22H （Block Erase）	BLK	校验和	03H

备注 BLK：要被擦除的块号

图 5-12. Block Erase 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：块擦除结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.8 Block Erase 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.8 Block Erase 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.8 Block Erase 命令。**

5.7 Programming命令

5.7.1 说明

这个命令用来在写入起始地址和写入结束地址被发送后发送写入字节个数的数据。这个命令然后写入用户程序到 flash 存储器并在内部检查。

写起始/结束 地址只能被设置为块起始/结束地址。

如果在最后一个数据被发送后两个状态帧（ST1 和 ST2）都表示 ACK，V850ES/Jx2 固件自动执行内部校验。因此，这个内部校验的 Status 命令必须被发送。

5.7.2 命令帧和状态帧

图 5-13 表示 Programming 命令的命令帧格式，图 5-14 表示这个命令的状态帧。

图 5-13. Programming 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	40H（**Programming**）	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注 SAH，SAM，SAL：写起始地址
EAH，EAM，EAL：写结束地址

图 5-14. Programming 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （a）	校验和	03H

备注 ST1 （a）：命令接收结果

5.7.3 数据帧和状态帧

图 5-15 表示包含要被写入的数据的帧的格式，图 5-16 表示数据的状态帧。

图 5-15. 要被写入的数据帧（从编程器到 V850ES/Jx2）

STX	LEN	数据	SUM	ETX/ETB
02H	00H 到 FFH（00H = 256）	写入数据	校验和	03H/17H

备注 写入数据：要被写入的用户程序

图 5-16. 数据帧的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据		SUM	ETX
02H	02H	ST1 （b）	ST2 （b）	校验和	03H

备注 ST1 （b）：数据接收检查结果
ST2 （b）：写入结果

5.7.4 发送所有数据的完成和状态帧

图 5-17 表示在所有数据发送完成后的状态帧。

图 5-17. 发送所有数据完成后的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （c）	校验和	03H

备注 ST1 （c）：内部校验结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.9 Programming 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.9 Programming 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.9 Programming 命令。**

5.8 Verify命令

5.8.1 说明

这个命令被用来比较从编程器发送的数据和从 V850ES/Jx2（读取级别）指定地址范围读取的数据，并且检查它们是否匹配。

校验起始/结束地址只能被设置为块起始/结束地址。

5.8.2 命令帧和状态帧

图 5-18 表示 Verify 命令的命令帧格式，图 5-19 表示这个命令的状态帧。

图 5-18. Verify 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	13H （校验）	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注 SAH，SAM，SAL：校验起始地址
EAH，EAM，EAL：校验结束地址

图 5-19. Verify 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （a）	校验和	03H

备注 ST1 （a）：命令接收结果

5.8.3 数据帧和状态帧

图 5-20 表示包含要被校验的数据的帧的格式，图 5-21 表示数据的状态帧。

图 5-20. 要被校验的数据帧（从编程器到 V850ES/Jx2）

STX	LEN	数据	SUM	ETX/ETB
02H	00H 到 FFH （00H = 256）	校验数据	校验和	03H/17H

备注 校验数据：要被校验的用户程序

图 5-21. 数据帧的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据		SUM	ETX
02H	02H	ST1（b）	ST2（b）	校验和	03H

备注 ST1（b）：数据接收检查结果
ST2（b）：校验结果[注]

注 即使在指定地址范围内一个校验错误发生，ACK 也会作为校验结果被返回。所有校验错误在最后一个数据的校验结果中被反映出来。因此，校验错误的发生只能在指定地址范围的所有校验处理完成时才能被检查。

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.10 Verify 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.10 Verify 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.10 Verify 命令。**

5.9 Block Blank Check命令

5.9.1 说明

这个命令被用来检查 flash 存储器中指定块号是否为空白（擦除状态）。

5.9.2 命令帧和状态帧

图 5-22 表示 Block Blank Check 命令的命令帧格式，图 5-23 表示这个命令的状态帧。

图 5-22. Block Blank Check 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息	SUM	ETX
01H	02H	32H （Block Blank Check）	BLK	校验和	03H

备注 BLK：要被空白检查的块号

图 5-23. Block Blank Check 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：块空白检查结果

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.11 Block Blank Check 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.11 Block Blank Check 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.11 Block Blank Check 命令。**

5.10 Silicon Signature命令

5.10.1 说明

这个命令用来读取器件的写入协议信息（silicon 签名）。

例如，如果编程器支持 V850ES/Jx2 不支持的编程协议，执行这个命令根据第二个和第三个字节的值来选择合适的协议。

5.10.2 命令帧和状态帧

图 5-24 表示 Silicon Signature 命令的命令帧格式，图 5-25 表示这个命令的状态帧。

图 5-24. Silicon Signature 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	SUM	ETX
01H	01H	C0H（**Silicon Signature**）	校验和	03H

图 5-25. Silicon Signature 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：命令接收结果

5.10.3 Silicon 签名数据帧

图 5-26 表示包含 silicon 签名数据的帧的格式

图 5-26. Silicon 签名数据帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据				SUM	ETX
02H	n	VEN	EXT	FNC	无效数据	校验和	03H

备注 **1.** n（LEN）： 数据长度
VEN： 提供商码（NEC：10H）
EXT： 扩展码
FNC： 功能信息
INVALID DATA： 90 到 198 字节长度的无效数据。

2. 对于提供商码（VEN），控制码（EXT）和功能信息（FNC），最高位被用作奇校验。下面表示一个例子。

表 5-1. Silicon 签名数据的例子

区域	内容	长度（字节）	Silicon 签名数据[注]的例子	实际值
VEN	提供商码（NEC）	1	10H （00010000B）	10H
EXT	控制码（固定）	1	7FH （01111111B）	7FH
FNC	功能信息（固定）	1	40H （01000000B）	40H

注 暗色的位是奇校验（调整一个字节中“1”的个数的值）。

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.12 Silicon Signature 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.12 Silicon Signature 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.12 Silicon Signature 命令。**

5.11 Version Get命令

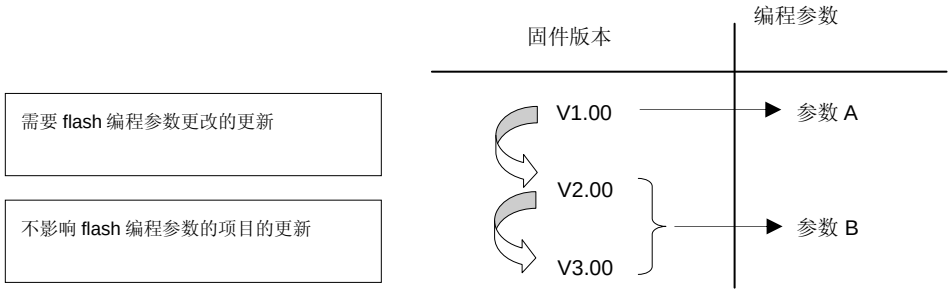
5.11.1 说明

这个命令被用来获取 V850ES/Jx2 设备版本和固件版本信息。

当编程参数必须根据 V850ES/Jx2 固件版本被更改时，使用这个命令。

注意事项 固件版本可能在固件更新过程中被更新，这不会影响 flash 编程参数的更改（这时，固件版本的更新没有被报告）。

例 固件版本和程序编程参数



5.11.2 命令帧和状态帧

图 5-28 表示 Version Get 命令的命令帧格式，图 5-29 表示这个命令的状态帧。

图 5-28. Version Get 命令帧（从编程器到 V850ES/Jx2）

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

图 5-29. Version Get 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：命令接收结果

5.11.3 版本数据帧

图 5-30 表示版本数据的数据帧。

图 5-30. 版本数据帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据						SUM	ETX
02H	06H	DV1	DV2	DV3	FV1	FV2	FV3	校验和	03H

备注 DV1：设备版本的整数
DV2：设备版本的第一个小数位置
DV3：设备版本的第二个小数位置
FV1：固件版本的整数
FV2：固件版本的第一个小数位置
FV3：固件版本的第二个小数位置

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.13 Version Get 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.13 Version Get 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.13 Version Get 命令。**

5.12 Checksum命令

5.12.1 说明

这个命令被用来获取指定区域中的校验和数据。

对于校验和计算起始/结束地址，从 flash 存储器的顶端开始以块为单位（2KB）指定一个固定地址。

通过从初始值（00H）中以 1 字节为单位按顺序减去指定地址中的数据来获得。

5.12.2 命令帧和状态帧

图 5-31 表示 Checksum 命令的命令帧格式，图 5-32 表示这个命令的状态帧。

图 5-31. Checksum 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	B0H （校验和）	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注 SAH，SAM，SAL：校验和计算起始地址

EAH，EAM，EAL：校验和计算结束地址

图 5-32. Checksum 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	校验和	03H

备注 ST1：命令接收结果

5.12.3 校验和数据帧

图 5-33 表示包含校验和数据的帧的格式。

图 5-33. Checksum 数据帧（从编程器到 V850ES/Jx2）

STX	LEN	数据		SUM	ETX
02H	02H	CK1	CK2	校验和	03H

备注 CK1：校验和的高 8 位

CK2：校验和的低 8 位

关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.14 Checksum 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.14 Checksum 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.14 Checksum 命令。**

5.13 Security Set命令

5.13.1 说明

这个命令用来执行安全设置（使能或无效写入、块擦除和芯片擦除）。通过使用这个命令执行这些设置，被一个未授权公司的 flash 存储器的重新写入可能被限制。

注意事项 一旦安全设置被执行，对安全标志的其它设置或从无效到使能的设置的更改将不再可能。如果这些设置被试图执行，一个写入错误（1CH）将发生。要重新设置安全标志，必须通过执行 Chip Erase 命令来初始化所有的安全标志（Block Erase 命令不能被用来初始化安全标志）。然而，如果芯片擦除被使无效，芯片擦除本身不可能，所以设置不能从编程器擦除。由于这个编程器规范，建议在使芯片擦除无效前，再次确认安全设置的执行。

5.13.2 命令帧和状态帧

图 5-34 表示 Security Set 命令的命令帧格式，图 5-35 表示这个命令的状态帧。

Security Set 命令帧包含块号区域和页号码区域，但是这些区域没有特殊用途，所以设置这些区域为 00H。

图 5-34. Security Set 命令帧（从编程器到 V850ES/Jx2）

SOH	LEN	COM	命令信息		SUM	ETX
01H	03H	A0H （Security Set）	00H （固定）	00H （固定）	校验和	03H

备注 BLK，PAG：固定为 00H

图 5-35. Security Set 命令的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （a）	校验和	03H

备注 ST1 （a）：命令接收结果

5.13.3 数据帧和状态帧

图 5-36 表示安全数据帧的格式，图 5-37 表示数据的状态帧。

图 5-36. 安全数据帧（从编程器到 V850ES/Jx2）

STX	LEN	数据	SUM	ETX
02H	01H	FLG	校验和	03H

备注 FLG：安全标志

图 5-37. 安全数据写入的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （b）	校验和	03H

备注 ST1 （b）：安全数据写入结果

5.13.4 内部校验检查和状态帧

图 5-38 表示内部校验检查的状态帧。

图 5-38. 内部校验检查的状态帧（从 V850ES/Jx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 （c）	校验和	03H

备注 ST1 （c）：内部校验结果

下表表示安全标志区域的内容。

表 5-2. 安全标志区域的内容

项目	内容
位 7	固定为“1”
位 6	
位 5	
位 4	
位 3	
位 2	编程无效标志（1：使能编程，0：使编程无效）
位 1	块擦除无效标志（1：使能块擦除，0：使块擦除无效）
位 0	芯片擦除无效标志（1：使能芯片擦除，0：使芯片擦除无效）

下表表示安全标志设置和每个操作的使能/使无效状态之间的关系。

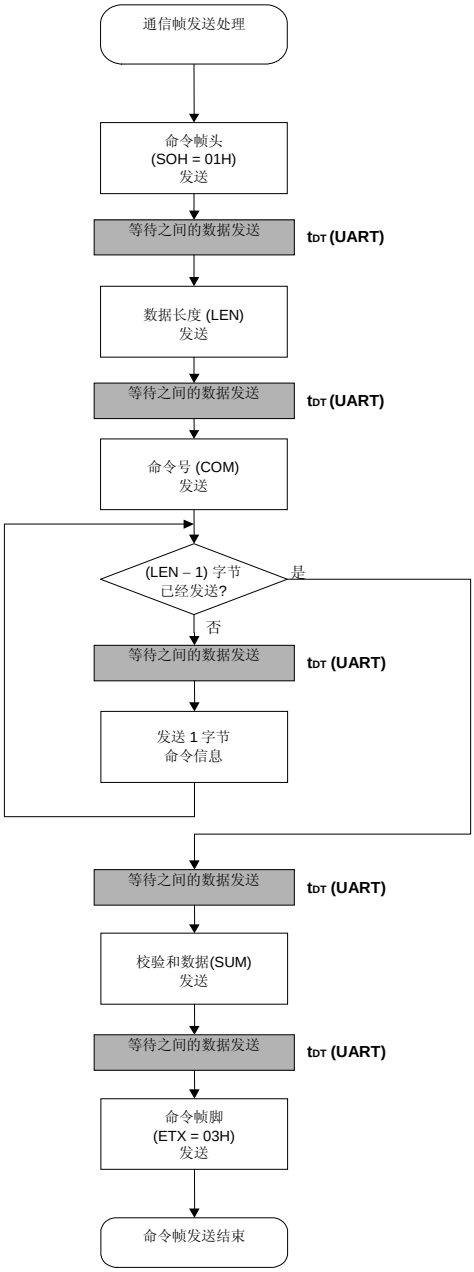
表 5-3. 安全标志区域和每个操作的使能/使无效状态

工作模式	Flash 存储器编程模式			自编程模式
命令 / 安全设置项目	安全设置后的命令操作 √：可以执行，×：不能执行			• 所有命令都可以被执行，而不管安全设置值 • 只有与安全设置值有关的值可能
	Programming	Chip Erase	Block Erase	
使编程无效	×	√	×	
使芯片擦除无效	√	×	×	
使块擦除无效	√	√	×	

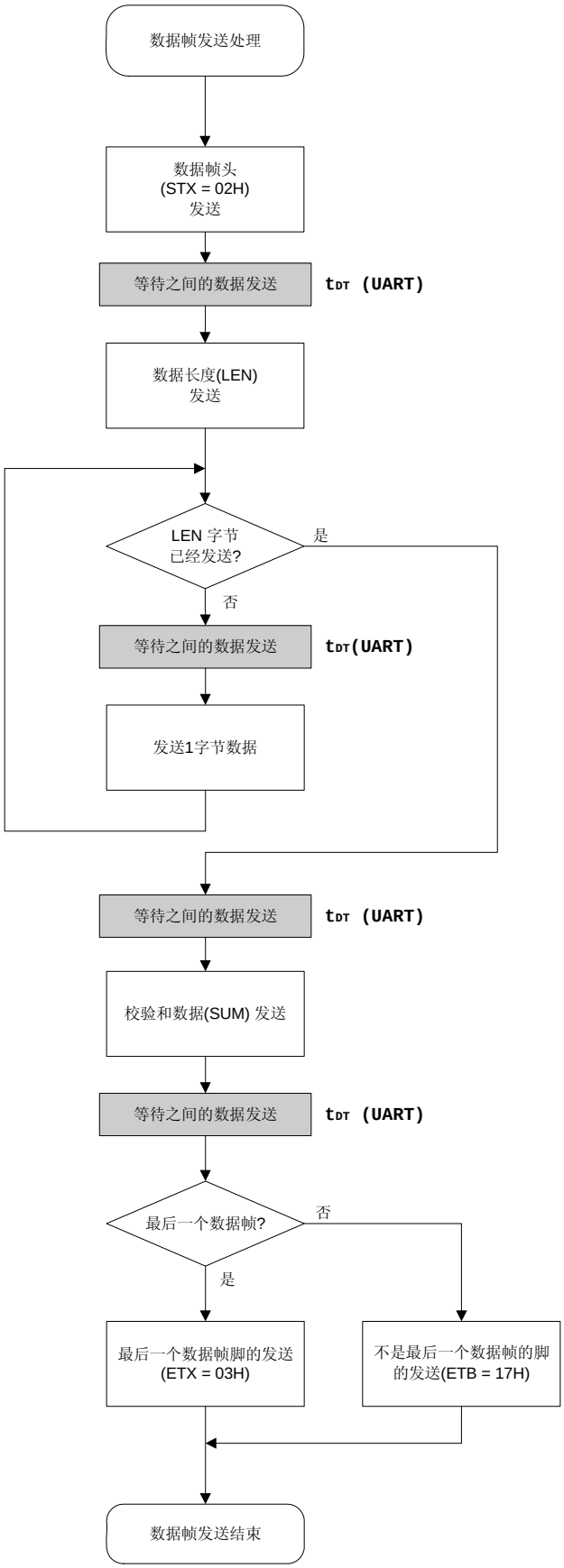
关于编程器和 V850ES/Jx2 之间的处理顺序的流程图、命令处理的流程图和每种通信模式的样本程序的细节，请阅读以下章。

- 关于 UART 通信模式，阅读 **6.15 Security Set 命令**。
- 关于支持握手的 3 线串行输入 / 输出通信模式（CSI + HS），阅读 **7.15 Security Set 命令。**
- 关于 3 线串行输入 / 输出通信模式（CSI ），阅读 **8.15 Security Set 命令。**

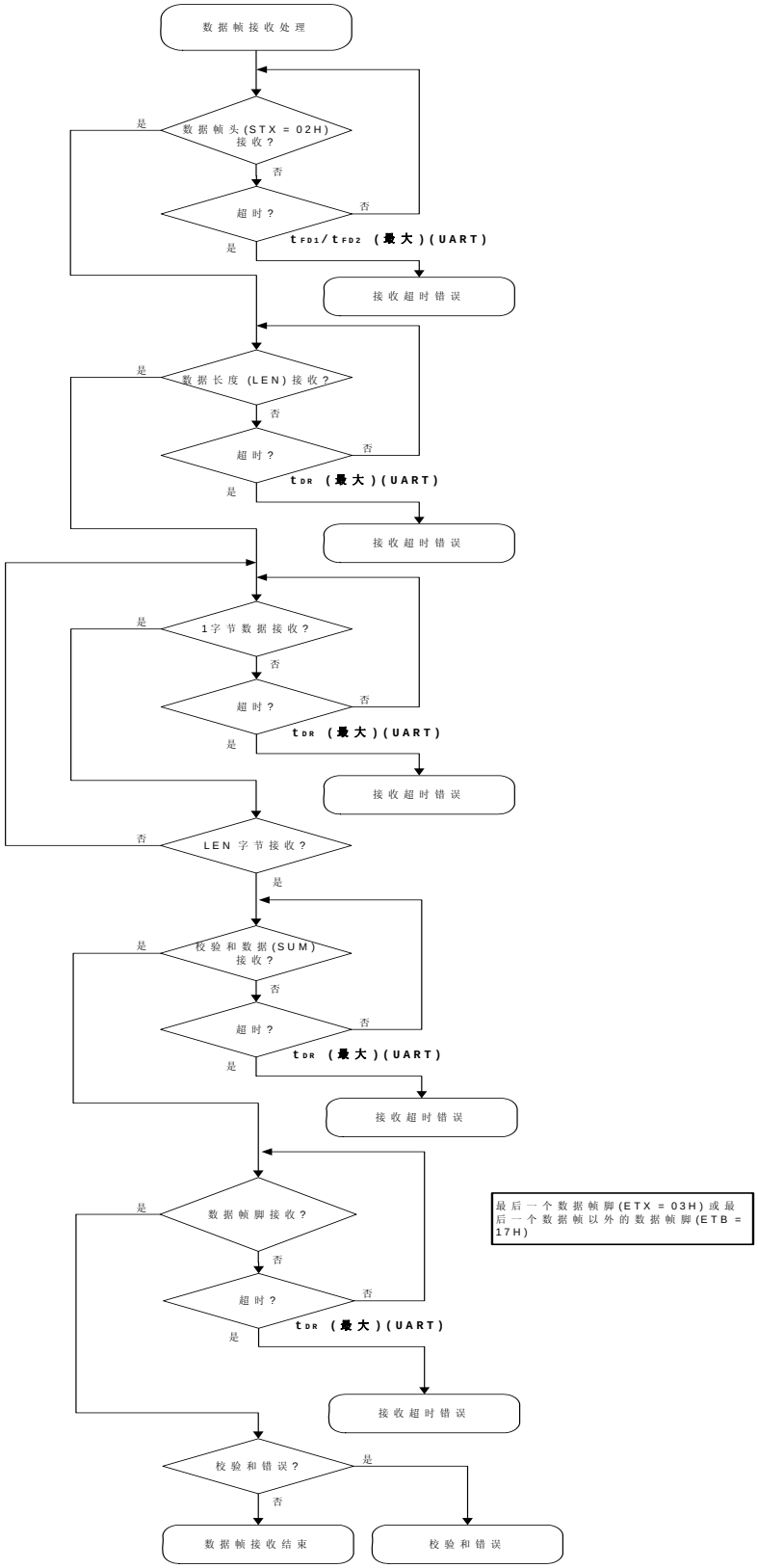
6.1 命令帧发送处理流程图



6.2 数据帧发送处理流程图



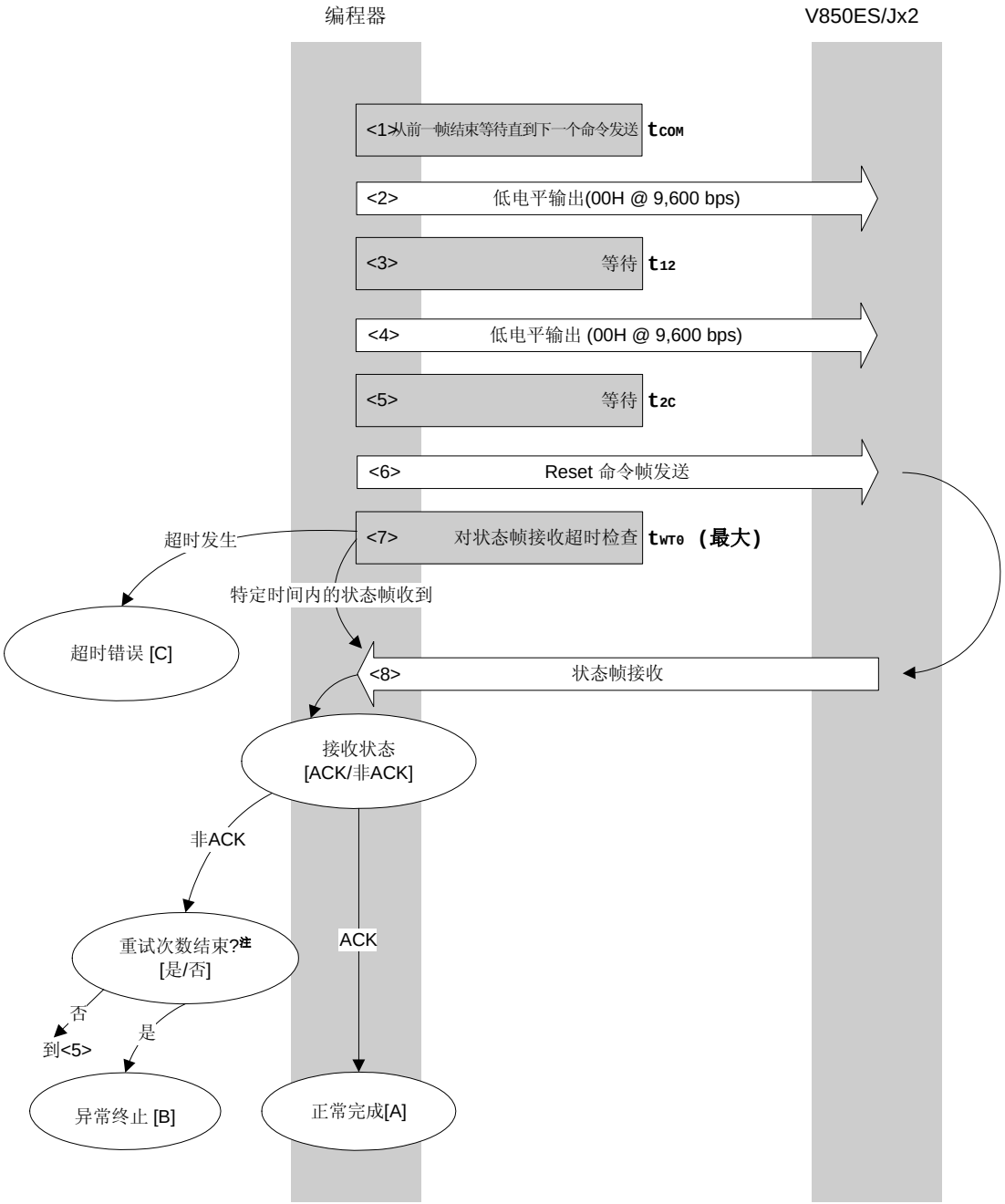
6.3 数据帧接收处理流程图



6.4 Reset 命令

6.4.1 处理顺序图

Reset 命令处理顺序



注 不要超过复位命令发送的重试次数（最多 16 次）。

6.4.2 处理顺序的说明

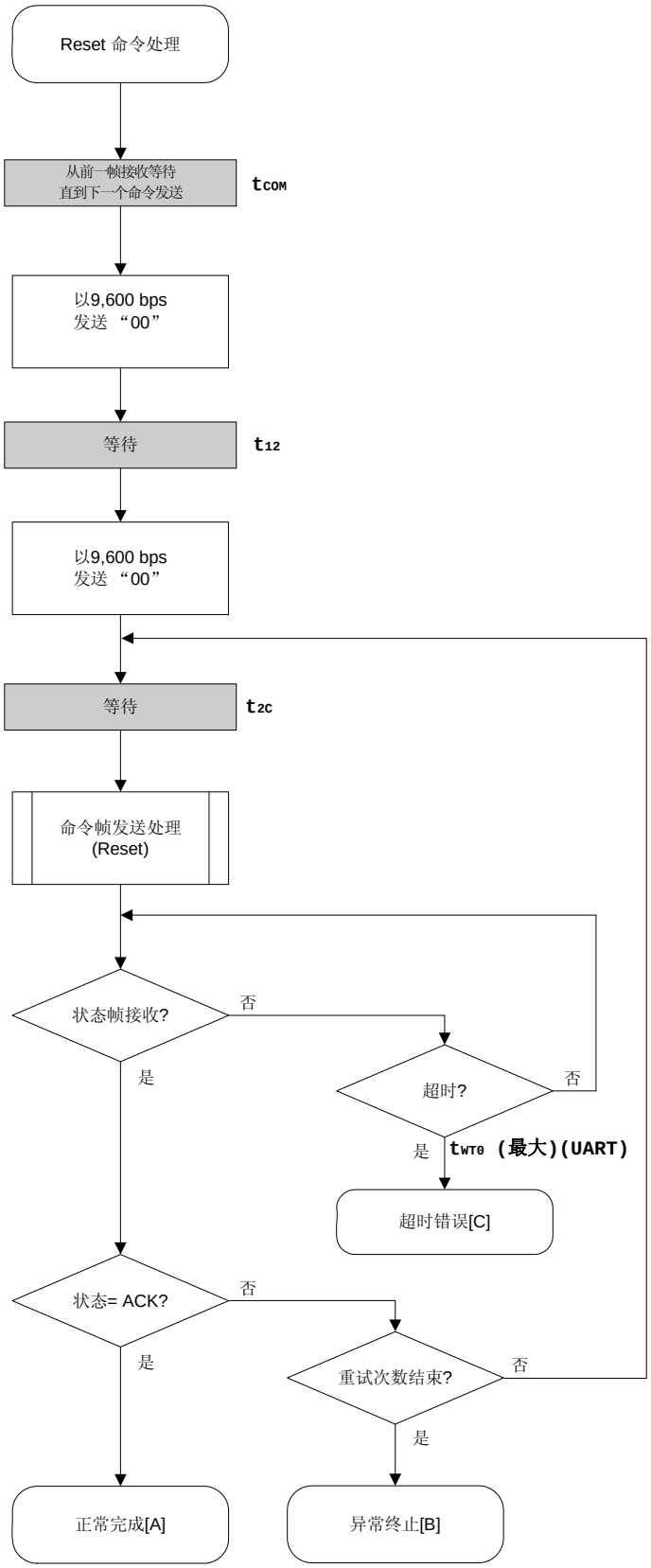
<1> 从前一帧接收等待直到下一个命令处理开始（等待时间 t_{COM}）。
<2> 输出低电平（数据 00H 以 9,600 bps 被发送）。
<3> 等待状态（等待时间 t_{12}）。
<4> 输出低电平（数据 00H 以 9,600 bps 被发送）。
<5> 等待状态（等待时间 t_{2C}）。
<6> 通过命令帧发送处理发送 Reset 命令。
<7> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT0}（最大））。
<8> 检查状态码。

当 ST1 = ACK 时： 正常完成［A］
当 ST1 ≠ ACK 时： 检查重试次数（t_{RS}）。
如果重试次数没有结束，顺序从<5>被重新执行。
如果重试次数结束，处理异常结束［B］。

6.4.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且编程器和 V850ES/Jx2 之间的同步已经被建立。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。

6.4.4 流程图


Reset 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
以9,600 bps
发送 “00”
等待
t12
以9,600 bps
发送 “00”
等待
t2C
命令帧发送处理
(Reset)
状态帧接收?
否
是
超时?
否
是
tWT0 (最大)(UART)
超时错误[C]
状态= ACK?
否
是
重试次数结束?
否
是
正常完成[A]
异常终止[B]

6.4.5 范例程序

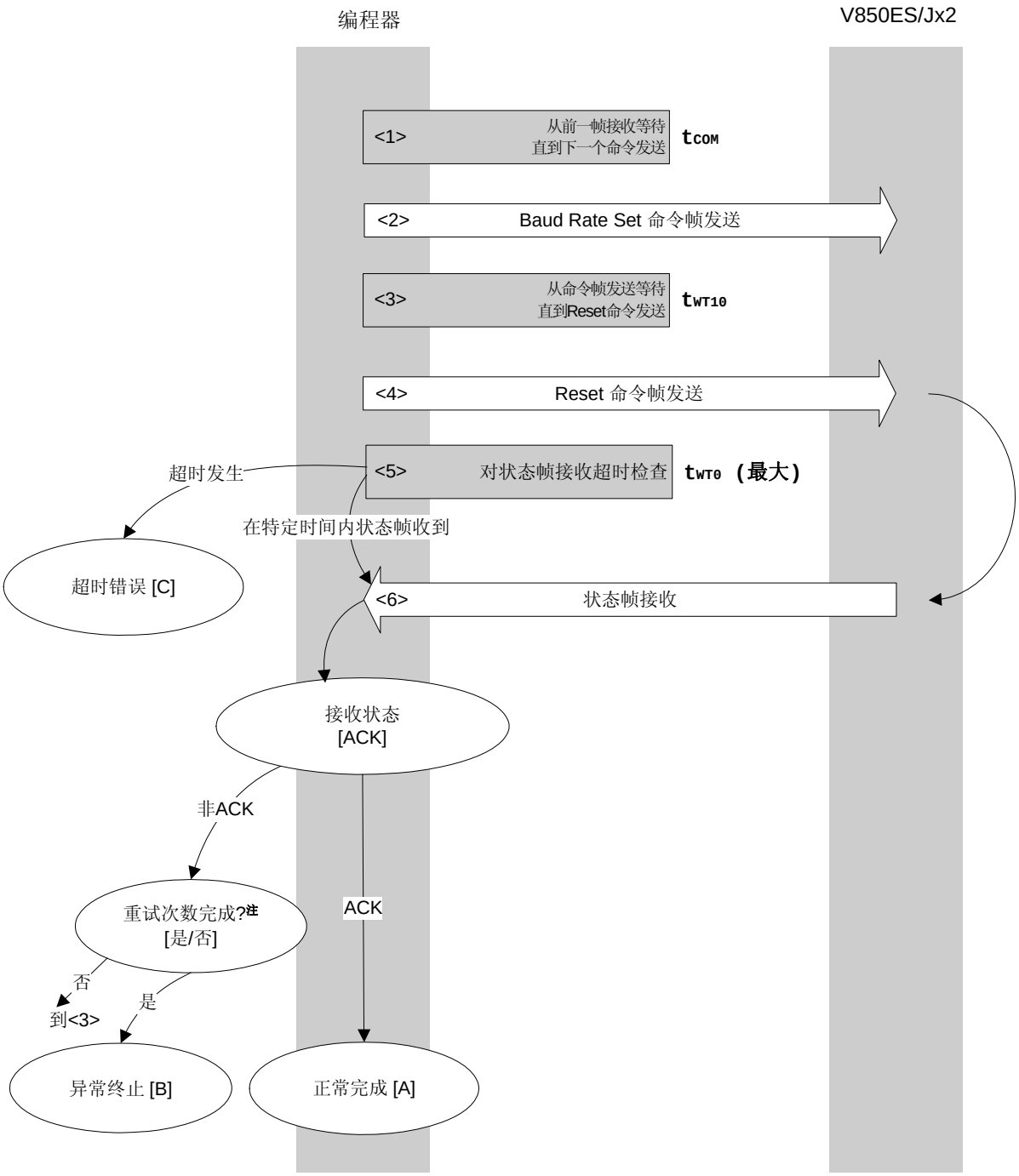
以下表示 Reset 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/*复位命令                                                       */
/*                                                               */
/*****************************************************************/
/* [r] u16                                                       */
/*****************************************************************/
u16   fl_ua_reset(void)
{
      u16      rc;
      u32      retry;

      set_uart0_br(BR_9600);    // 更改为 9600bps

      fl_wait(tCOM_UA);                 // 等待
      putc_ua(0x00);            // 发送 0x00 @ 9600bps

      fl_wait(t12);                     // 等待
      putc_ua(0x00);            // 发送 0x00 @ 9600bps

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

            fl_wait(t2C);       // 等待

            put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);      // 发送 RESET 命令

            rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
            if (rc == FLC_DFTO_ERR) // 超时. ?
                  break;                  // 是     // 如果 ［C］

            if (rc == FLC_ACK){             // ACK ?
                  break;                  // 是     // 如果 ［A］
            }
            else{
                  NOP();
            }
            //继续;                         // 如果 ［B］（如果从循环退出)
      }
//    switch(rc) {
//
//          case    FLC_NO_ERR:       return  rc;     break;  // 如果 ［A］
//          case    FLC_DFTO_ERR:     return  rc;     break;  // 如果 ［C］
//          default:                  return  rc;     break;  // 如果 ［B］
//    }
      return  rc;
}
```

6.5 Baud Rate Set 命令

6.5.1 处理顺序图

Baud Rate Set 命令处理顺序



注 不要超过复位命令发送的重试次数（最多 16 次）。

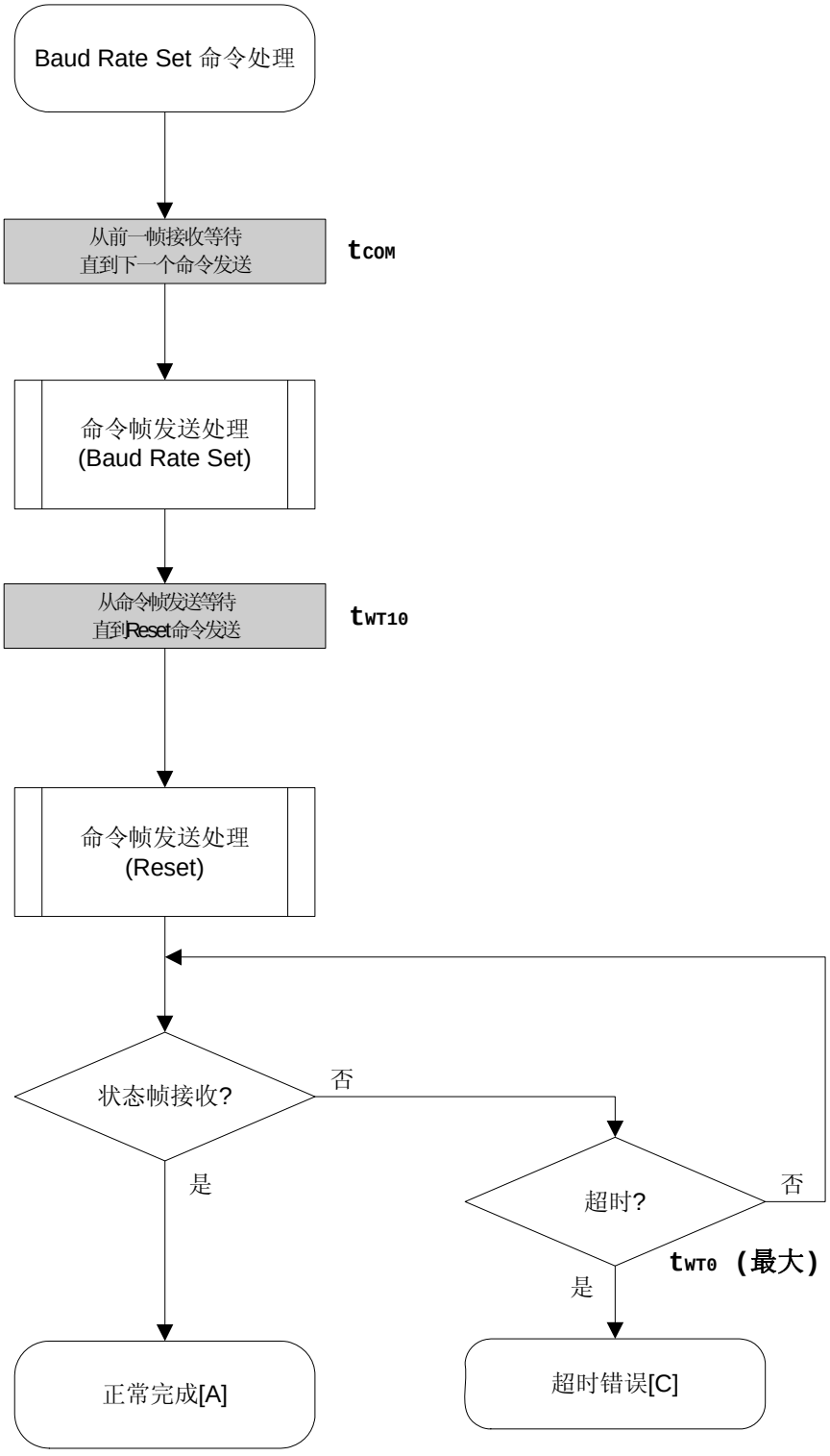
6.5.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Baud Rate Set 命令。
<3> 从命令发送等待直到 Reset 命令发送（等待时间 t_{WT10}）。
<4> Reset 命令通过命令帧发送处理被发送。
<5> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误［C］被返回（超时时间 t_{WT0}（最大））。
<6> 因为状态码应该是 ACK，处理正常结束［A］。

6.5.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且编程器和 V850ES/Jx2 之间的 UART 通信速度的同步已经被建立。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	数据帧接收超时。 对于 V850ES/Jx2，在以下情况下，这个命令也会导致错误。 • 命令信息（参数）无效。 • 命令帧包含校验和错误 • 命令帧的数据长度（LEN）无效 • 命令帧的脚（ETX）丢失 • 在设置波特率并且接收命令帧数据 16 次后，Reset 命令没有被检测到。

6.5.4 流程图


Baud Rate Set 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Baud Rate Set)
从命令帧发送等待
直到Reset命令发送
tWT10
命令帧发送处理
(Reset)
状态帧接收?
否
是
超时?
否
tWT0 (最大)
是
正常完成[A]
超时错误[C]

6.5.5 范例程序

以下表示 Baud Rate Set 命令处理的一个范例程序。

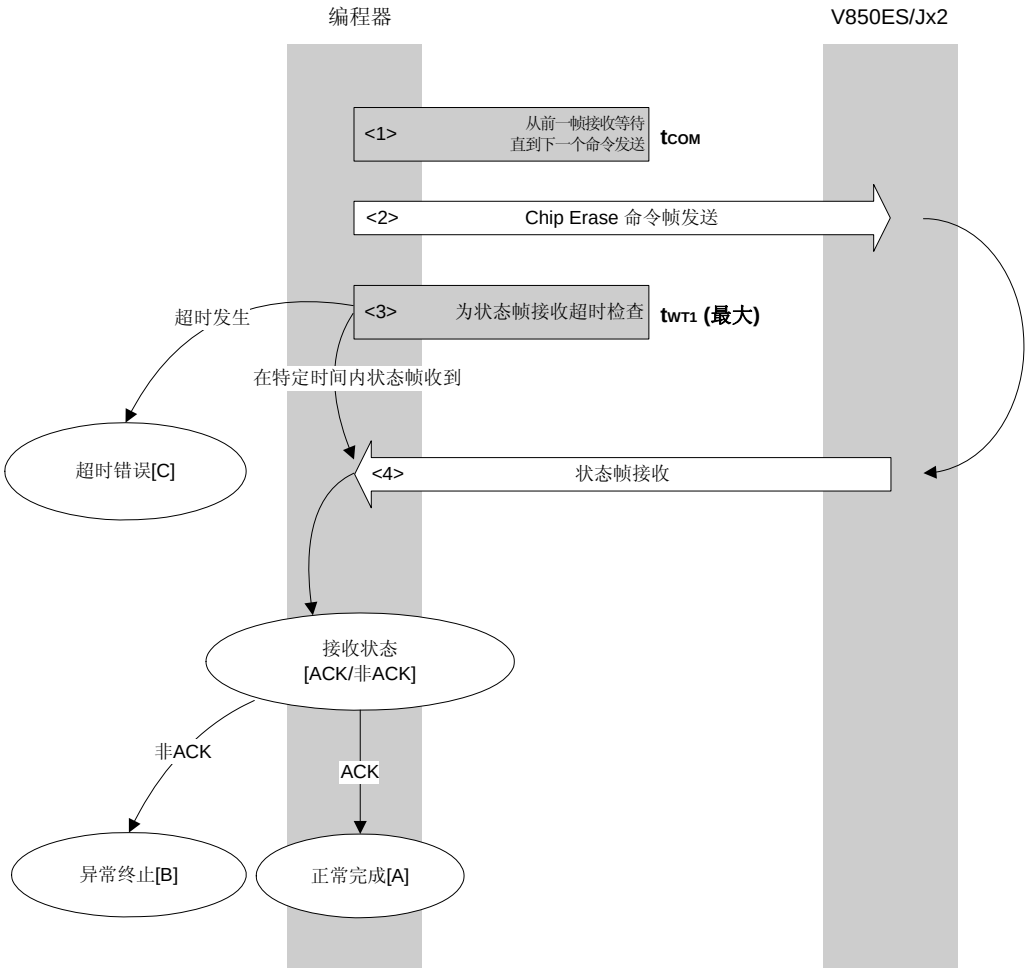
```
/*****************************************************************/
/*                                                               */
/*设置波特率命令                                                 */
/*                                                               */
/*****************************************************************/
/* [i]  u8 brid          ... 波特率 ID                           */
/* [r] u16               ... 错误码                              */
/*****************************************************************/
u16   fl_ua_setbaud(u8 brid)
{
        u16     rc;
        u8      br;
        u32     retry;

        switch(brid){
                default:
                case    BR_9600:        br = 0x03;      break;
                case    BR_19200:       br = 0x04;      break;
                case    BR_31250:       br = 0x05;      break;
                case    BR_38400:       br = 0x06;      break;
                case    BR_76800:       br = 0x07;      break;
                case    BR_153600:      br = 0x08;      break;
        }
        fl_cmd_prm[0] = br;         // "D01"


        fl_wait(tCOM_UA);           // 在发送命令前等待
        put_cmd_ua(FL_COM_SET_BAUDRATE, 2, fl_cmd_prm);// 发送“Baudrate Set”命令

        set_flbaud(brid);           // 更改波特率
        set_uart0_br(brid);                 // 更改波特率（硬件）

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

                put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);        // 发送 RESET 命令

                rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);         // 获取状态帧
                if (rc){
                        if (retry--)
                                continue;
                        else
                                return    rc;
                }
```

```
                break;              // 得到 ACK !!

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

        return rc;
}
```

6.6 Oscillating Frequency Set命令

6.6.1 处理顺序图

Chip Erase 命令处理顺序



6.6.2 处理顺序的说明

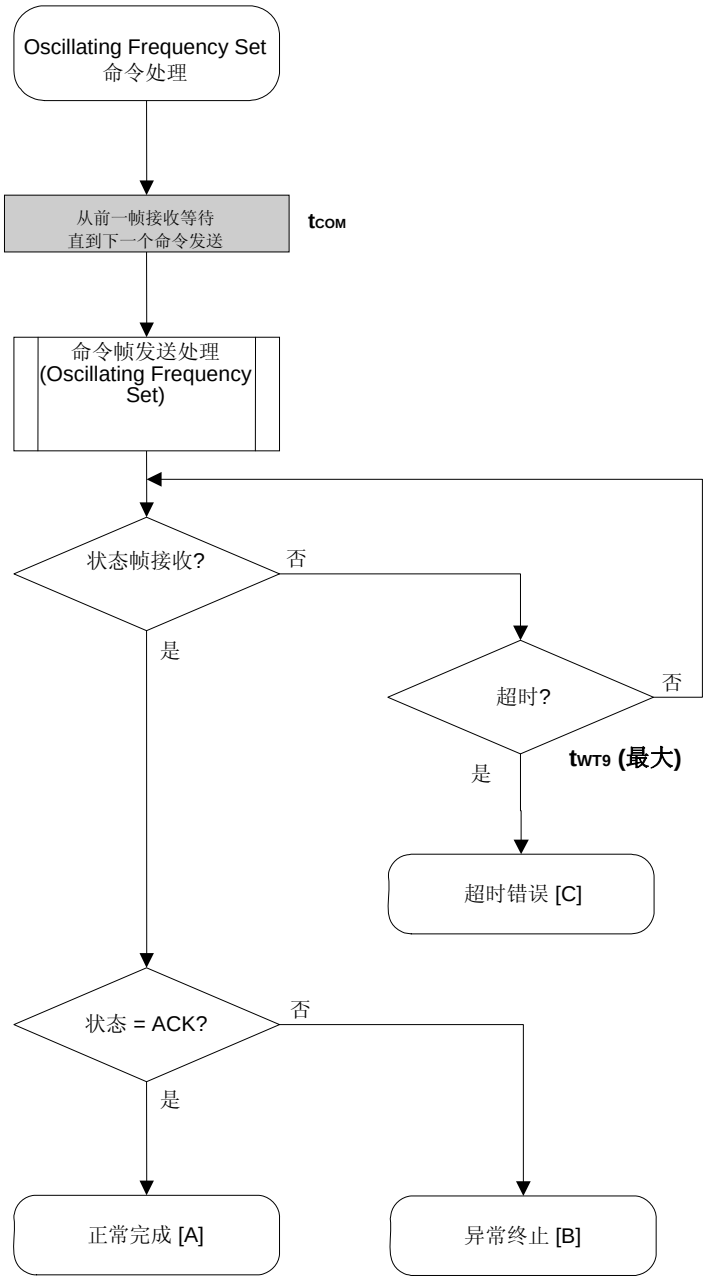
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Oscillating Frequency Set 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT9}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时：正常完成［A］
当 ST1 ≠ ACK 时：异常终止［B］

6.6.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且工作频率被正确设置到 V850ES/Jx2。
异常终止［B］	参数错误	05H	振荡频率值超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。

6.6.4 流程图


Oscillating Frequency Set
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Oscillating Frequency Set)
状态帧接收?
否
是
超时?
否
是
tWT9 (最大)
超时错误 [C]
状态 = ACK?
否
是
正常完成 [A]
异常终止 [B]

6.6.5 范例程序

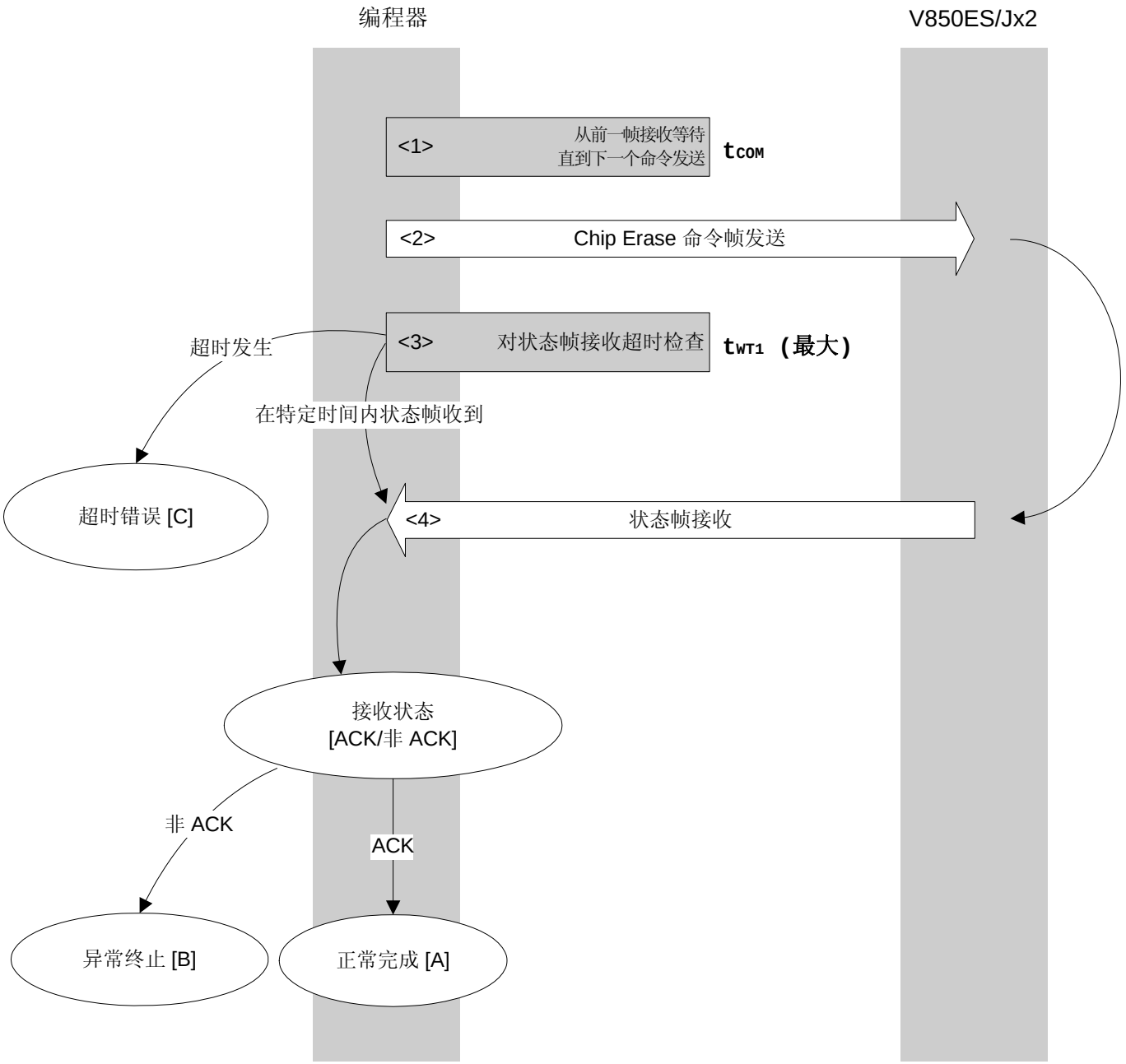
以下表示 Oscillating Frequency Set 命令处理的一个范例程序。

```
/*********************************************************************/
/*                                                                   */
/* 设置 Flash 器件时钟值命令                                          */
/*                                                                   */
/*********************************************************************/
/* [i] u8 clk[4]            ... 频率数据(D1-D4)                      */
/* [r] u16                  ... 错误码                               */
/*********************************************************************/
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_UA);                   // 在发送命令前等待
        put_cmd_ua(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);

        rc = get_sfrm_ua(fl_ua_sfrm, tWT9_MAX);    // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:         return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
//              default:                    return  rc;     break;  // 如果［B］
//      }

        return rc;
}
```

6.7 Chip Erase 命令

6.7.1 处理顺序图

Chip Erase 命令处理顺序



6.7.2 处理顺序的说明

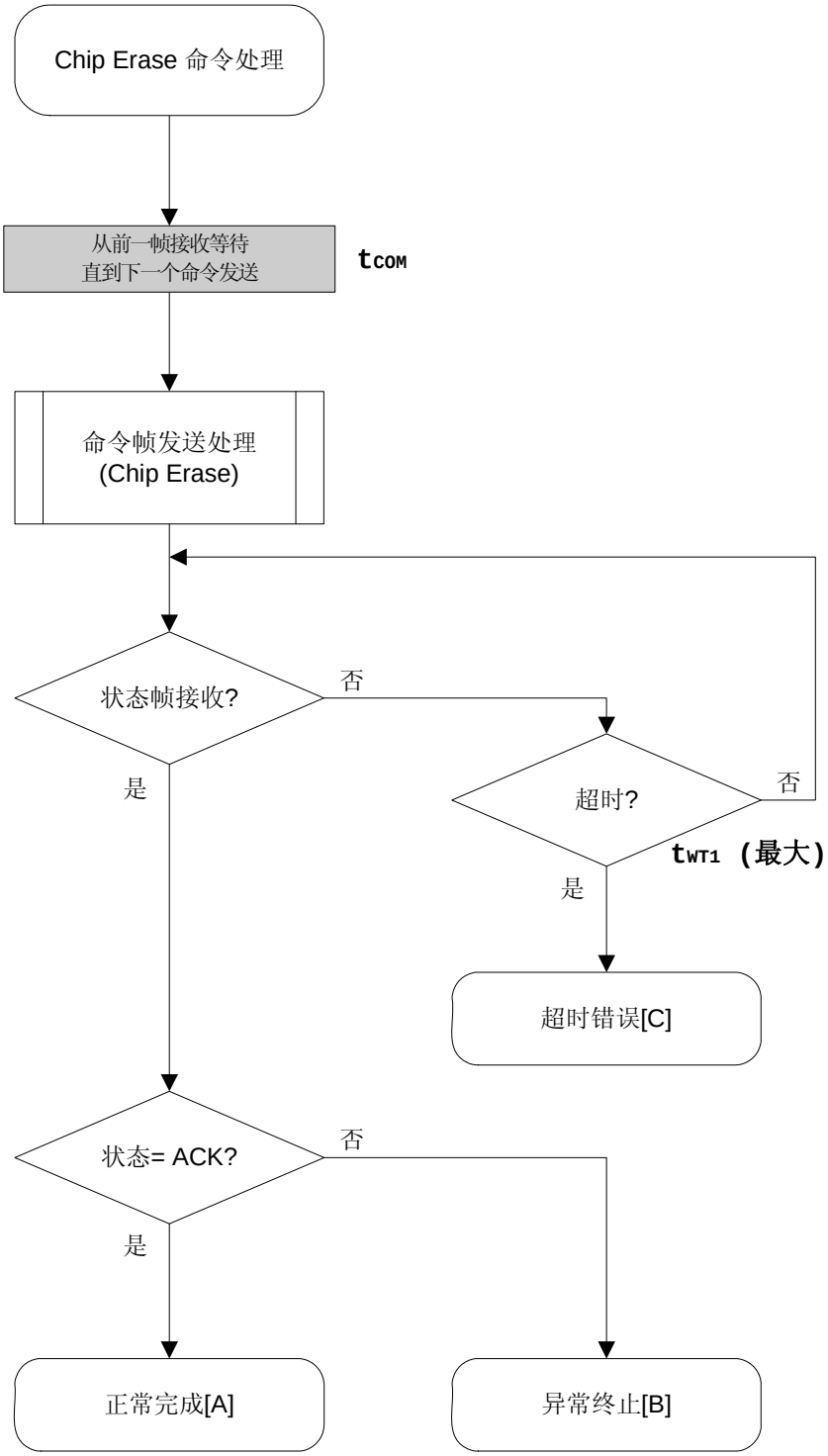
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Chip Erase 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT1}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时：正常完成［A］
当 ST1 ≠ ACK 时：异常终止［B］

6.7.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且正常完成芯片擦除。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

6.7.4 流程图


Chip Erase 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Chip Erase)
状态帧接收?
否
是
超时?
否
tWT1 (最大)
是
超时错误[C]
状态= ACK?
否
是
正常完成[A]
异常终止[B]

6.7.5 范例程序

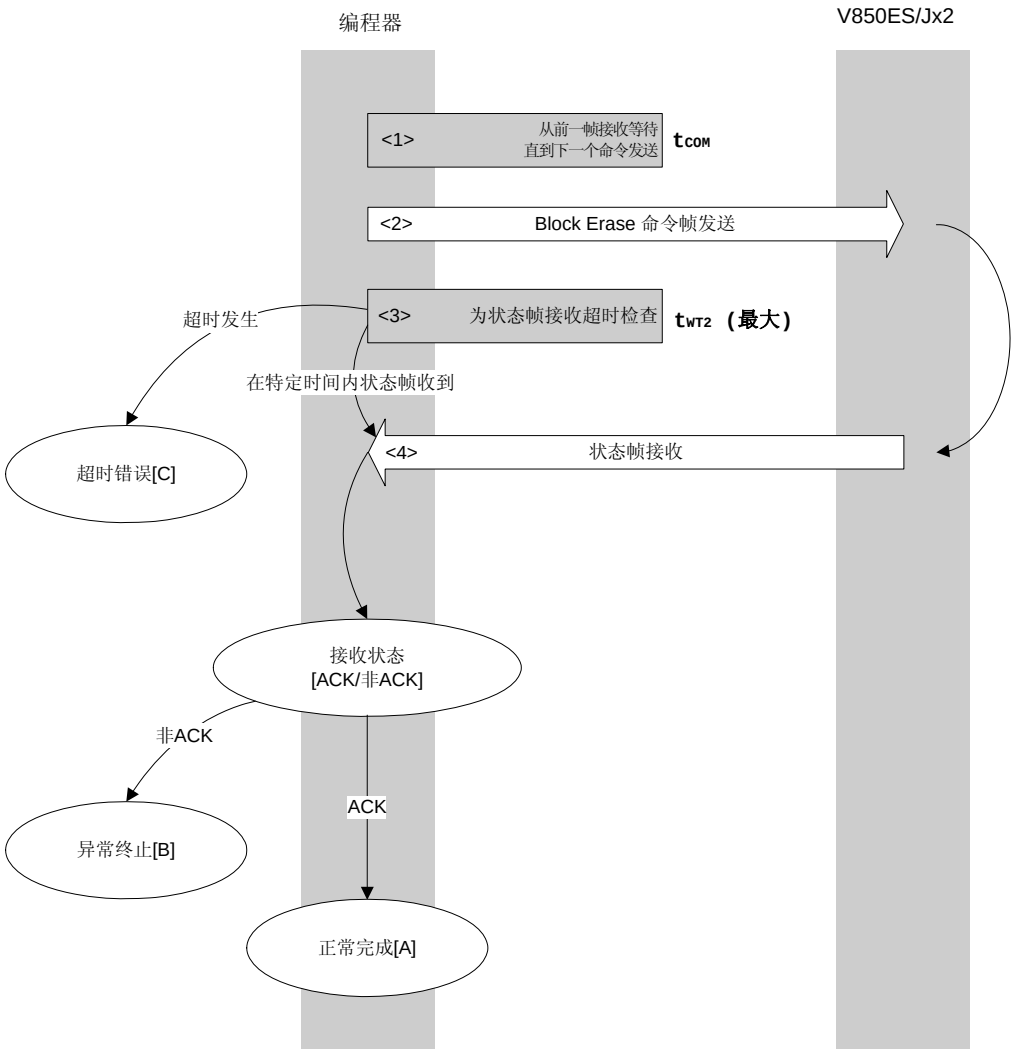
以下表示 Chip Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/*擦除全部（芯片）命令                                           */
/*                                                               */
/*****************************************************************/
/* [r] u16          ... 错误码                                   */
/*****************************************************************/
u16             fl_ua_erase_all(void)
{
        u16     rc;


        fl_wait(tCOM_UA);                   // 在发送命令前等待

        put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // 发送 ERASE CHIP 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX);    // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:         return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
//              default:                    return  rc;     break;  // 如果［B］
//      }
        return  rc;

}
```

6.8 Block Erase 命令

6.8.1 处理顺序图

Block Erase 命令处理顺序



6.8.2 处理顺序的说明

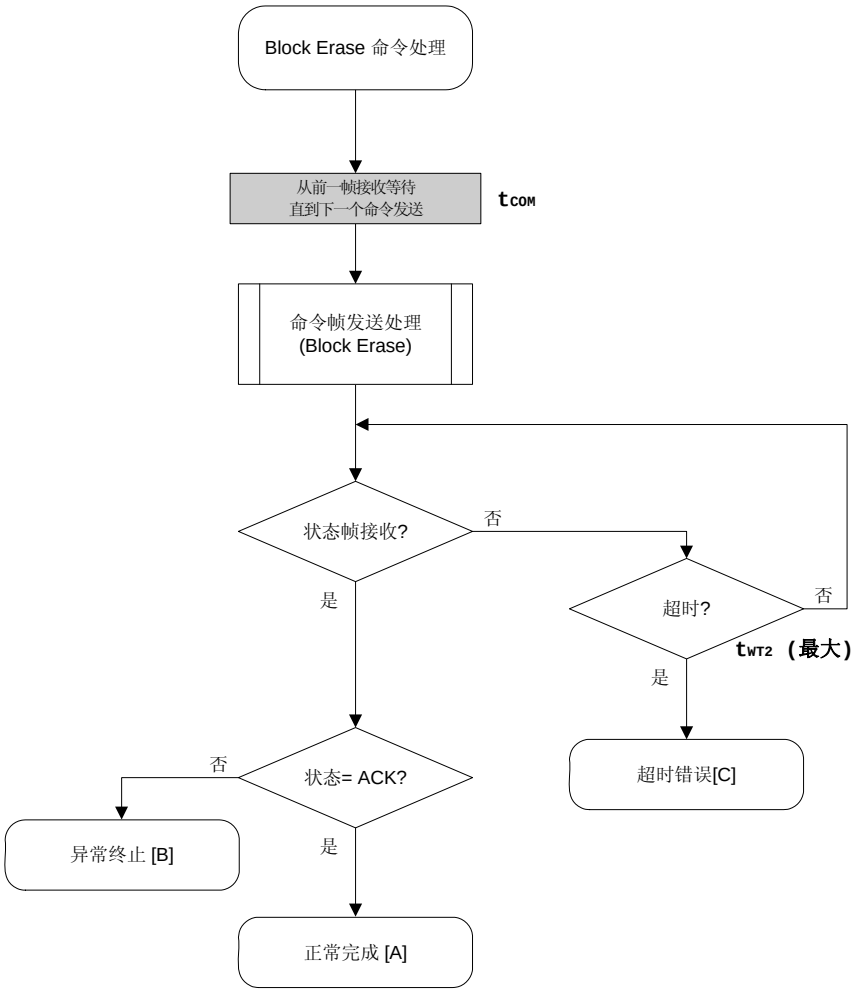
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Block Erase 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT2}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 当指定块的块擦除没有完成时，处理更改块号并按顺序从<1>重新执行。
当所有指定块的块擦除完成时，处理正常结束［A］。
当 ST1 ≠ ACK 时： 异常终止［B］

6.8.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且块擦除被正常完成。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	写入、块擦除或芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	重新错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

6.8.4 流程图


Block Erase 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Block Erase)
状态帧接收?
否
是
超时?
否
tWT2 （最大）
是
超时错误[C]
状态= ACK?
否
异常终止 [B]
是
正常完成 [A]

6.8.5 范例程序

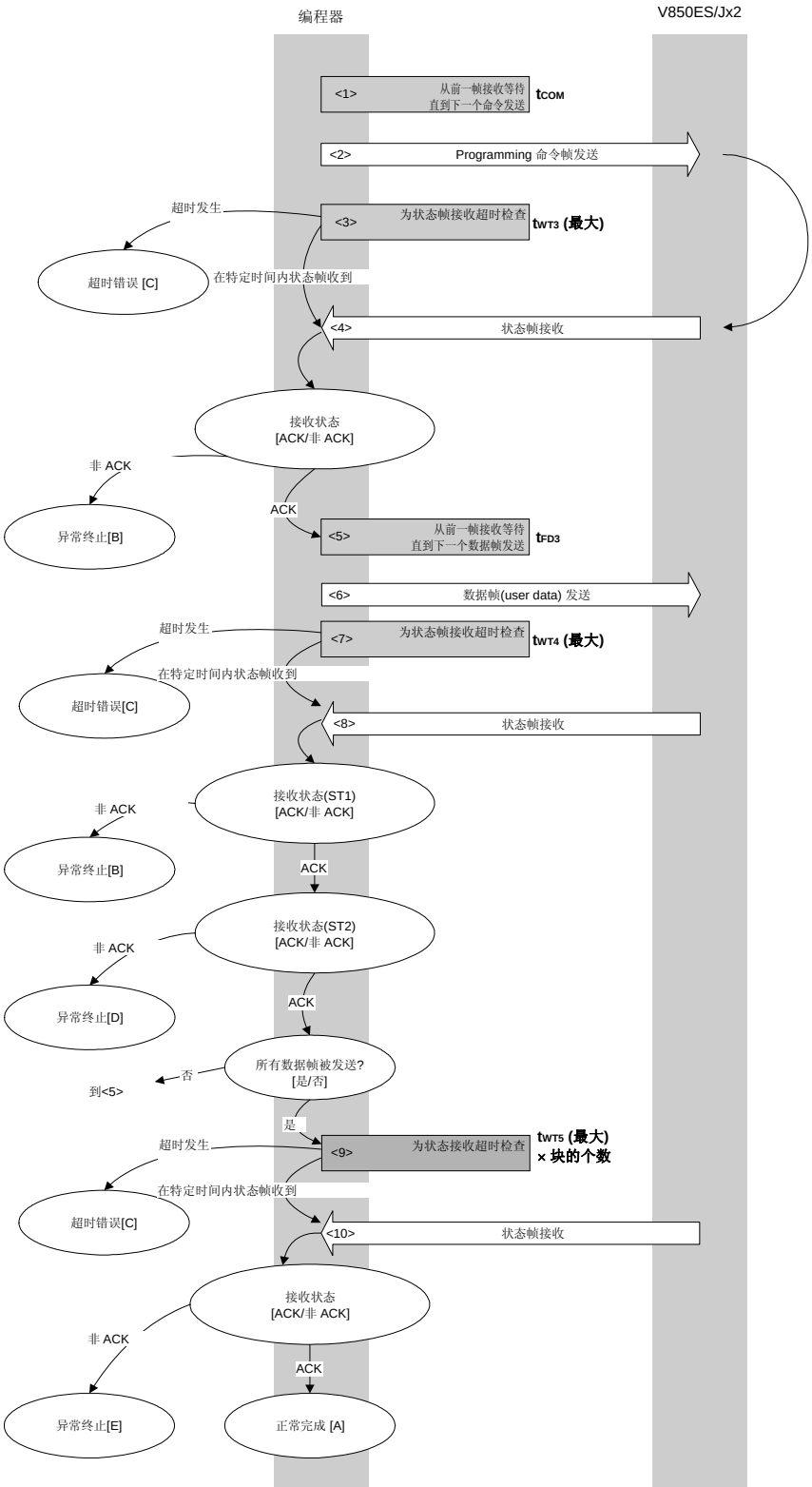
以下表示对一个块的 Block Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/*块擦除命令                                                     */
/*                                                               */
/*****************************************************************/
/* [i] u8 block              ... 块号                            */
/* [r] u16                   ... 错误码                          */
/*****************************************************************/
u16             fl_ua_erase_blk(u8 block)
{
        u16     rc;
        u32     wt2_max;

        fl_cmd_prm[0] = block;      // BLK
        wt2_max = get_wt2_max(get_block_size(block));

        fl_wait(tCOM_UA);                   // 在发送命令前等待

        put_cmd_ua(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm); // 发送 ERASE CHIP 命令

        rc = get_sfrm_ua(fl_ua_sfrm, wt2_max);      // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:         return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
//              default:                    return  rc;     break;  // 如果［B］
//      }

        return rc;

}
```

6.9 Programming 命令

6.9.1 处理时序图

Programming 命令处理顺序



6.9.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Programming 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT3}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

<5> 从前一帧接收等待直到下一个数据帧发送（等待时间 t_{FD3}）。
<6> 用户数据通过数据帧发送处理被发送。
<7> 从用户数据发送直到数据帧接收，超时检查被执行。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT4}（最大））。
<8> 状态码（ST1/ST2）被检查（参考处理顺序和流程图）。

当 ST1 ≠ ACK 时： 异常终止［B］
当 ST1 = ACK 时： 根据 ST2 的值，以下处理被执行。
- 当 ST2 = ACK 时： 当所有数据帧的发送完成后，到<9>。
  如果仍然存在数据帧要被发送，处理从<5>重新执行。
- 当 ST2 ≠ ACK 时： 异常终止［D］

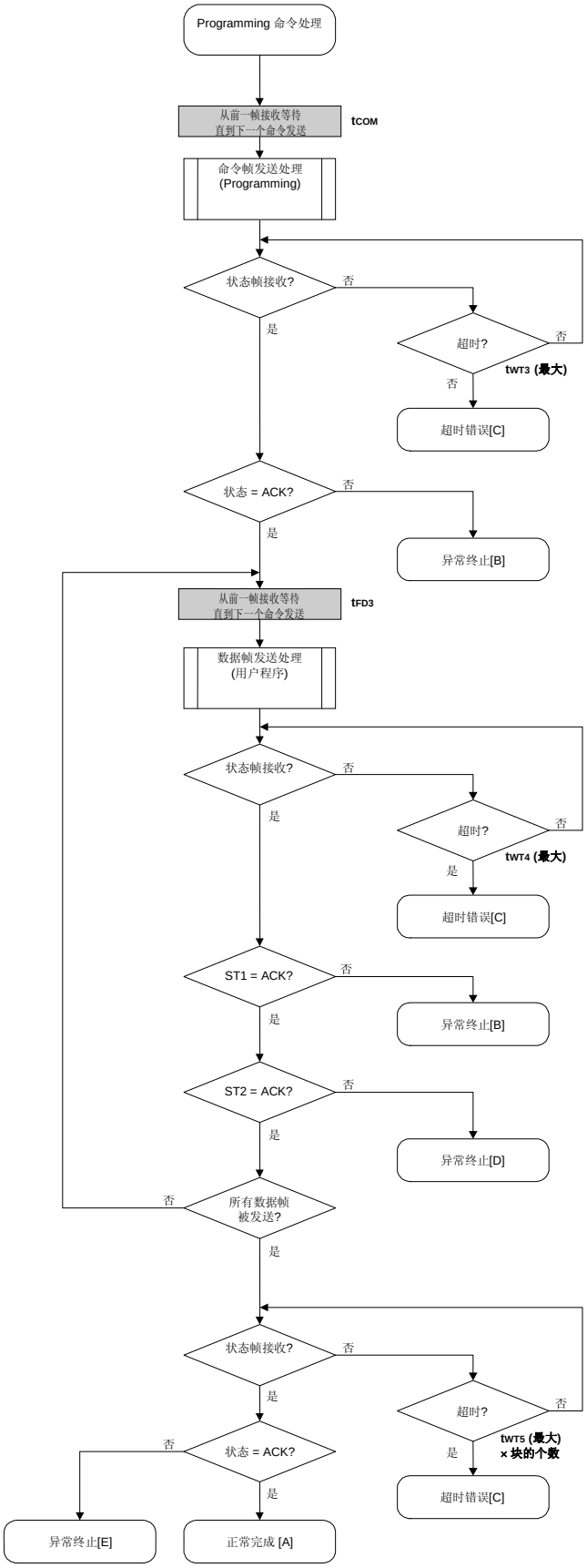
<9> 一个超时检查被执行直到状态帧接收。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT5}（最大））。
<10> 状态码被检查。

当 ST1 = ACK 时： 正常完成［A］
当 ST1 ≠ ACK 时： 异常终止［E］

6.9.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且用户数据被正常写入。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	写入在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
异常终止［D］	WWV1 错误	08H (ST2)	一个写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止［E］	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

6.9.4 流程图


Programming 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Programming)
状态帧接收?
否
是
超时?
否
tWT3 (最大)
否
超时错误[C]
状态 = ACK?
否
是
异常终止[B]
从前一帧接收等待
直到下一个命令发送
tFD3
数据帧发送处理
(用户程序)
状态帧接收?
否
是
超时?
否
tWT4 (最大)
是
超时错误[C]
ST1 = ACK?
否
是
异常终止[B]
ST2 = ACK?
否
是
异常终止[D]
所有数据帧
被发送?
否
是
状态帧接收?
否
是
超时?
否
tWT5 (最大)
× 块的个数
是
超时错误[C]
状态 = ACK?
否
是
异常终止[E]
正常完成 [A]

6.9.5 范例程序

以下表示 Programming 命令处理的一个范例程序。

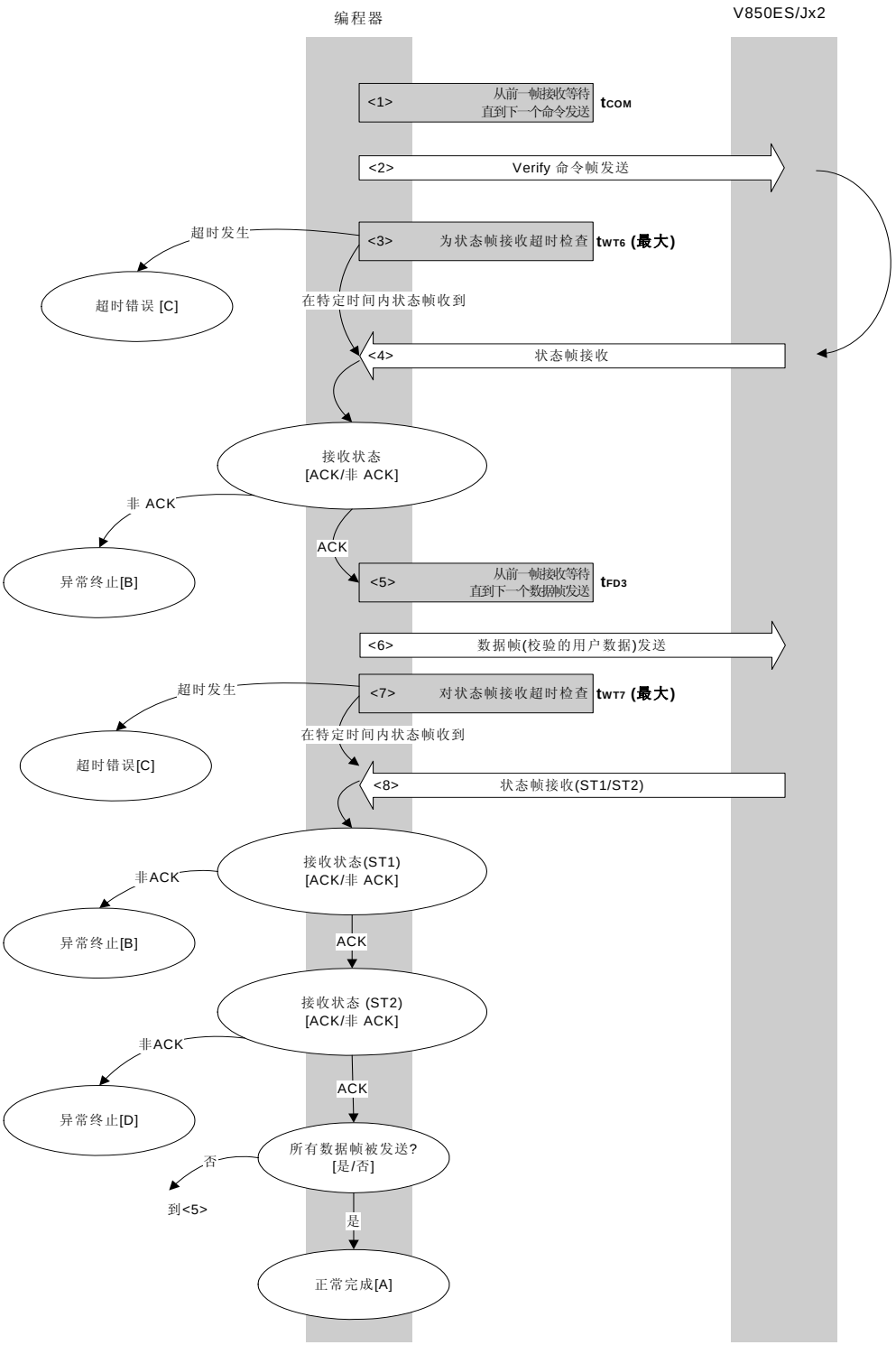
```
/*****************************************************************/
/*                                                               */
/*写入命令                                                       */
/*                                                               */
/*****************************************************************/
/* [i] u32 top              ... 起始地址                         */
/* [i] u32 bottom           ... 结束地址                         */
/* [r] u16                  ... 错误码                           */
/*****************************************************************/

#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;
        u32     wt5_max;

        /***********************************************/
       /*       设置参数                                */
        /***********************************************/
        set_range_prm(fl_cmd_prm, top, bottom);    // 设置 SAH/SAM/SAL, EAH/EAM/EAL
        wt5_max = get_wt5_max(bottom - top + 1);

        /***********************************************/
       /*       发送命令 & 检查状态                     */
        /***********************************************/
        fl_wait(tCOM);              // 在发送命令前等待

        put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm); // 发送 “Programming” 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT3_MAX);   // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
                default:                return  rc;     break;  // 如果［B］
        }

        /***********************************************/
       /*       发送用户数据                            */
        /***********************************************/
        send_head = top;

                while(1){
```

```
            // make send data frame
            if ((bottom - send_head) > 256){      // 剩余大小 > 256 ?
                    is_end = false;               // 是，不是最后一帧
                    send_size  = 256;             // 发送大小 = 256 字节
            }
            else{
                    is_end = true;
                    send_size = bottom - send_head + 1;          // 发送大小 = (底部 –
                                                                 // 发送头)+1 字节

            }
            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);        // 设置数据帧
                                                                        有效载荷
            send_head += send_size;

            fl_wait(tFD3);                                  // 在发送数据帧前等待

            put_dfrm_ua(send_size, fl_txdata_frm, is_end);          // 发送用户数据

            rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX);                 // 获得状态帧
            switch(rc) {
                            case    FLC_NO_ERR:                     break;  // 继续
                            case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
                            default:                    return  rc;     break;  // 如果［B］
            }
            if (fl_st2_ua != FLST_ACK){                             // ST2 = ACK ?
                            rc = decode_status(fl_st2_ua);          // 否
                            return  rc;                             // 如果［D］
            }
            if (is_end)
                            break;

    }
    /************************************************/
    /*      检查内部校验                            */
    /************************************************/
    rc = get_sfrm_ua(fl_ua_sfrm, wt5_max);                          // 再次获取状态帧

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

    return rc;
}
```

6.10 Verify 命令

6.10.1 处理顺序图

Verify 命令处理顺序



6.10.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Verify 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT6}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

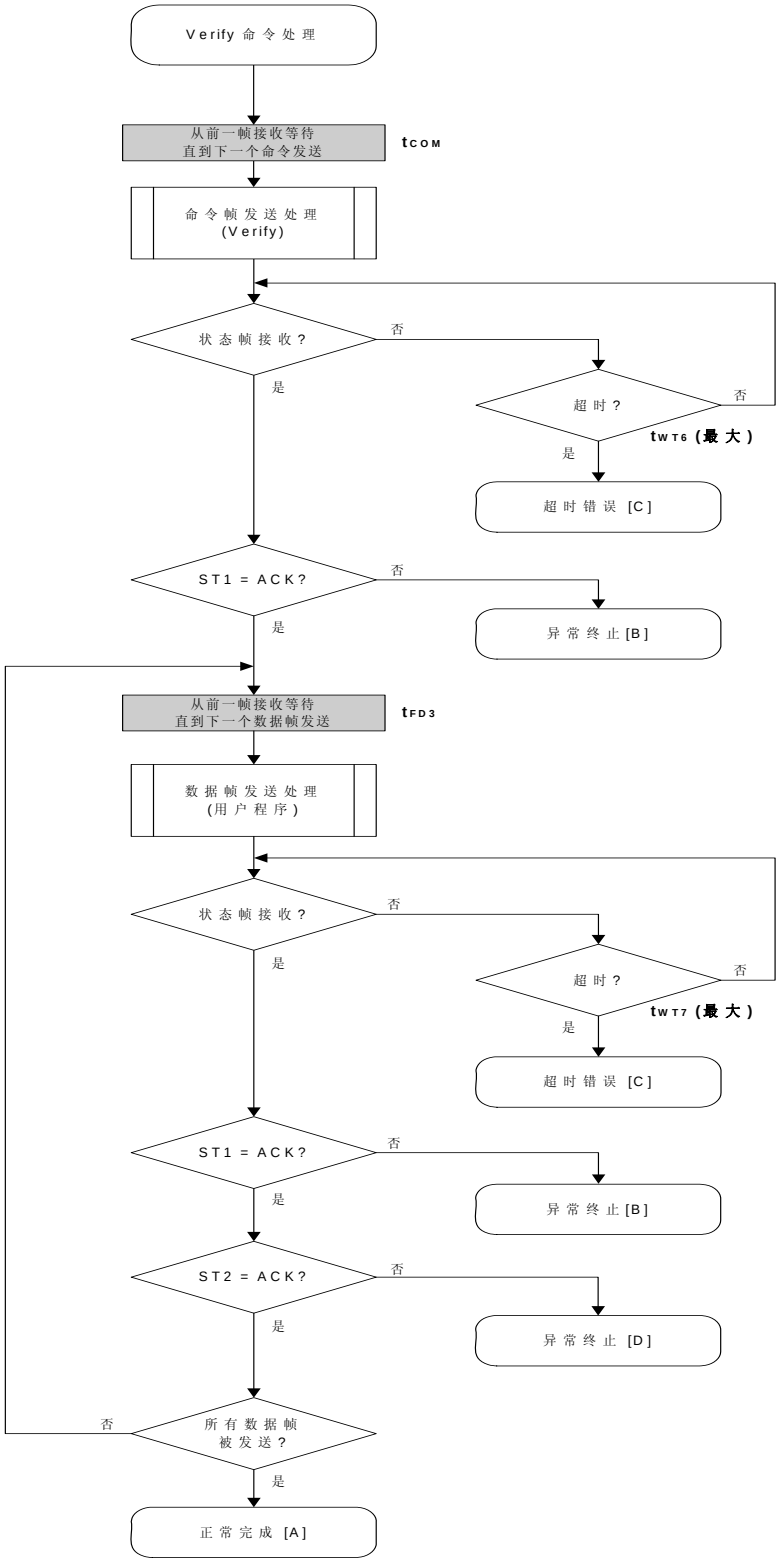
<5> 从前一帧接收等待直到下一个数据帧发送（等待时间 t_{FD3}）。
<6> 校验用的用户数据通过数据帧发送处理被发送。
<7> 从用户数据发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT7}（最大））。
检查状态码（ST1/ST2）（参考处理顺序和流程图）。

当 ST1 ≠ ACK 时： 异常终止［B］
当 ST1 = ACK 时： 根据 ST2 的值，执行以下处理。
- 当 ST2 = ACK 时： 如果所有数据帧的发送完成，处理正常结束［A］。
  如果仍然存在数据帧要被发送，处理从<5>重新执行。
- 当 ST2 ≠ ACK 时： 异常终止［D］

6.10.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验被正常完成。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
异常终止［D］	校验错误	0FH (ST2)	校验失败或者另一个错误发生。
	程序错误	16H	一个程序错误发生。

6.10.4 流程图


Verify 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Verify)
状态帧接收？
否
是
超时？
否
tWT6 (最大)
是
超时错误 [C]
ST1 = ACK?
否
是
异常终止 [B]
从前一帧接收等待
直到下一个数据帧发送
tFD3
数据帧发送处理
(用户程序)
状态帧接收？
否
是
超时？
否
tWT7 (最大)
是
超时错误 [C]
ST1 = ACK?
否
是
异常终止 [B]
ST2 = ACK?
否
是
异常终止 [D]
所有数据帧
被发送？
否
是
正常完成 [A]

6.10.5 范例程序

以下表示 Verify 命令处理的一个范例程序。

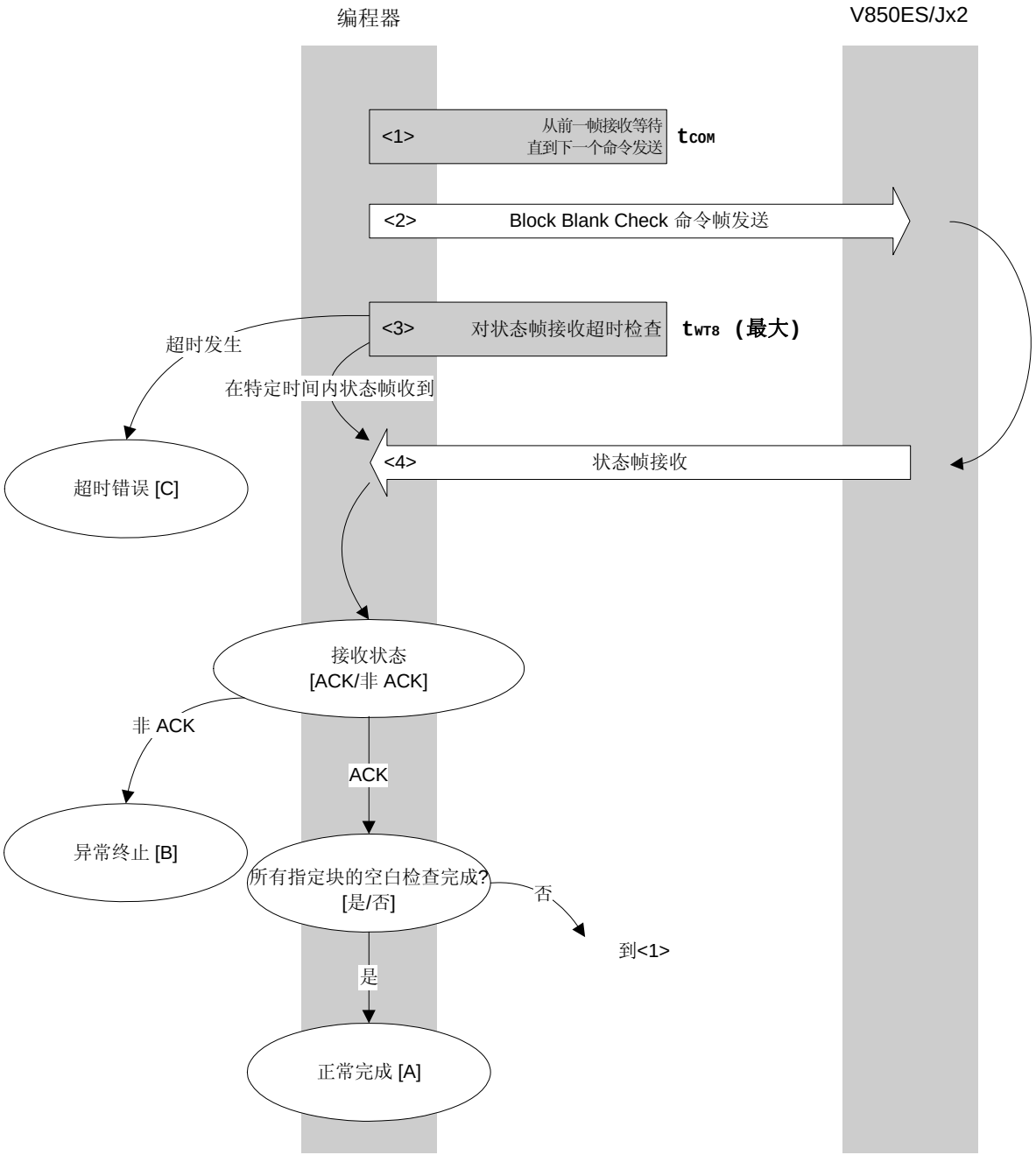
```
/*****************************************************************/
/*                                                               */
/*校验命令                                                       */
/*                                                               */
/*****************************************************************/
/* [i] u32 top              ... 起始地址                         */
/* [i] u32 bottom           ... 结束地址                         */
/* [r] u16                  ... 错误码                           */
/*****************************************************************/
u16   fl_ua_verify(u32 top, u32 bottom)
{
        u16     rc;
        u32     send_head, send_size;
        bool    is_end;

        /*************************************************/
       /*       设置参数                                */
        /*************************************************/
        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL, EAH/EAM/EAL


        /*************************************************/
       /*       发送命令 & 检查状态                     */
        /*************************************************/

        fl_wait(tCOM_UA);                   // 在发送命令前等待

        put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm);       // 发送 VERIFY 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT6_MAX);             // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
                default:                    return  rc;     break;  // 如果［B］
        }

        /*************************************************/
       /*       发送用户数据                            */
        /*************************************************/
        send_head = top;

        while(1){

                // make send data frame
                if ((bottom - send_head) > 256){      // 剩余大小 > 256 ?
```

```
                is_end = false;     // 是，不是最后一帧
                send_size  = 256; // 发送大小 = 256 字节
        }
        else{
                is_end = true;
                send_size = bottom - send_head + 1;          // 发送大小 =
                                                                // (底部 – 发送头)+1 字节


                }
        memcpy(fl_txdata_frm, rom_buf+send_head, send_size);        // 设置数据
                                                                        // 帧有效载荷
        send_head += send_size;

        fl_wait(tFD3);
        put_dfrm_ua(send_size, fl_txdata_frm, is_end); // 发送用户数据

        rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX);    // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_DFTO_ERR:       return  rc;     break;  // 如果 ［C］
                default:                    return  rc;     break;  // 如果 ［B］
        }
        if (fl_st2_ua != FLST_ACK){                         // ST2 = ACK ?
                        rc = decode_status(fl_st2_ua);      // 否
                        return  rc;                                 // 如果 ［D］
        }
        if (is_end)                                         // 发送所有用户数据？
                break;                                      // 是
        //继续;
    }
    return  FLC_NO_ERR;   // 如果 ［A］
}
```

6.11 Block Blank Check 命令

6.11.1 处理顺序图

Block Blank Check 命令处理顺序



6.11.2 处理顺序的说明

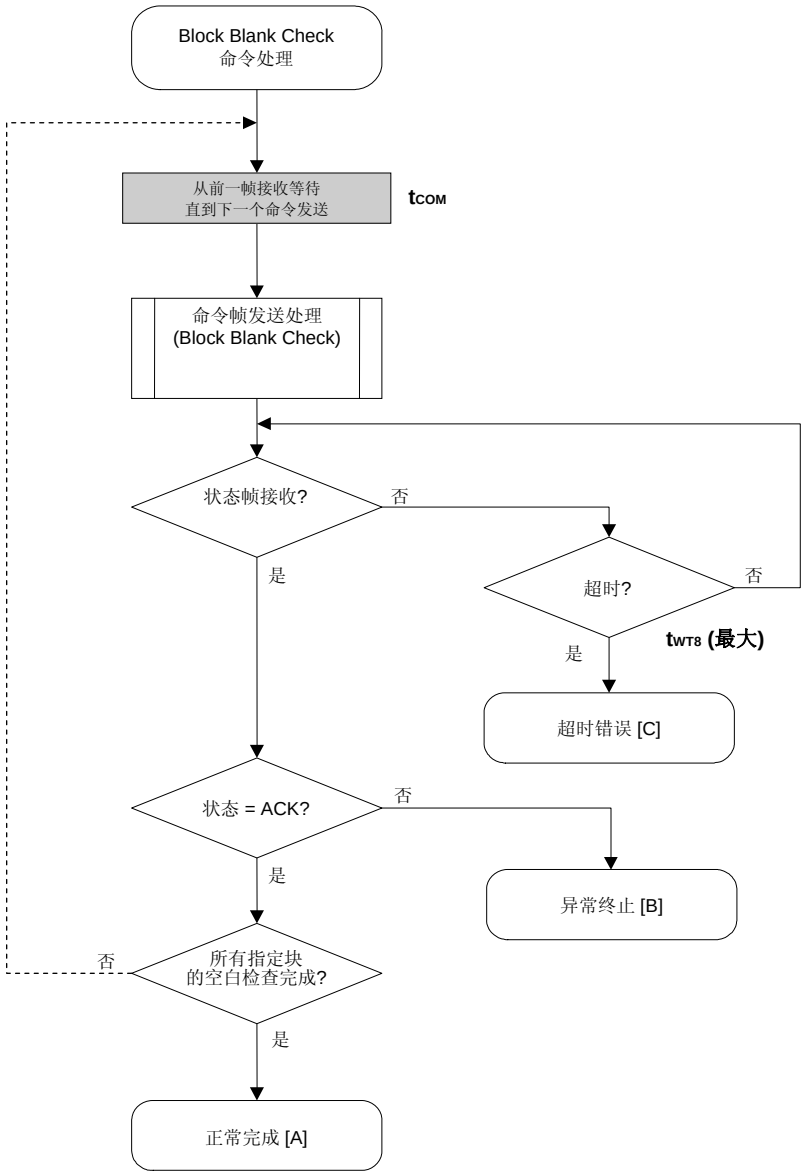
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Block Blank Check 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT8}（最大））。
<4> 检查状态码。

当 ST1 = ACK 时： 当指定块的空白检查没有完成时，处理更改块号并按顺序从<1>重新执行。
当所有指定块的空白检查完成时，处理正常结束［A］。
当 ST1 ≠ ACK 时： 异常终止［B］

6.11.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	这个命令被正常执行，并且所有指定的块空白。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	EWV4 错误	11H	指定块的 flash 存储器不是空白。
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

6.11.4 流程图


Block Blank Check
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Block Blank Check)
状态帧接收?
否
是
超时?
否
是
tWT8 (最大)
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
所有指定块
的空白检查完成?
否
是
正常完成 [A]

6.11.5 范例程序

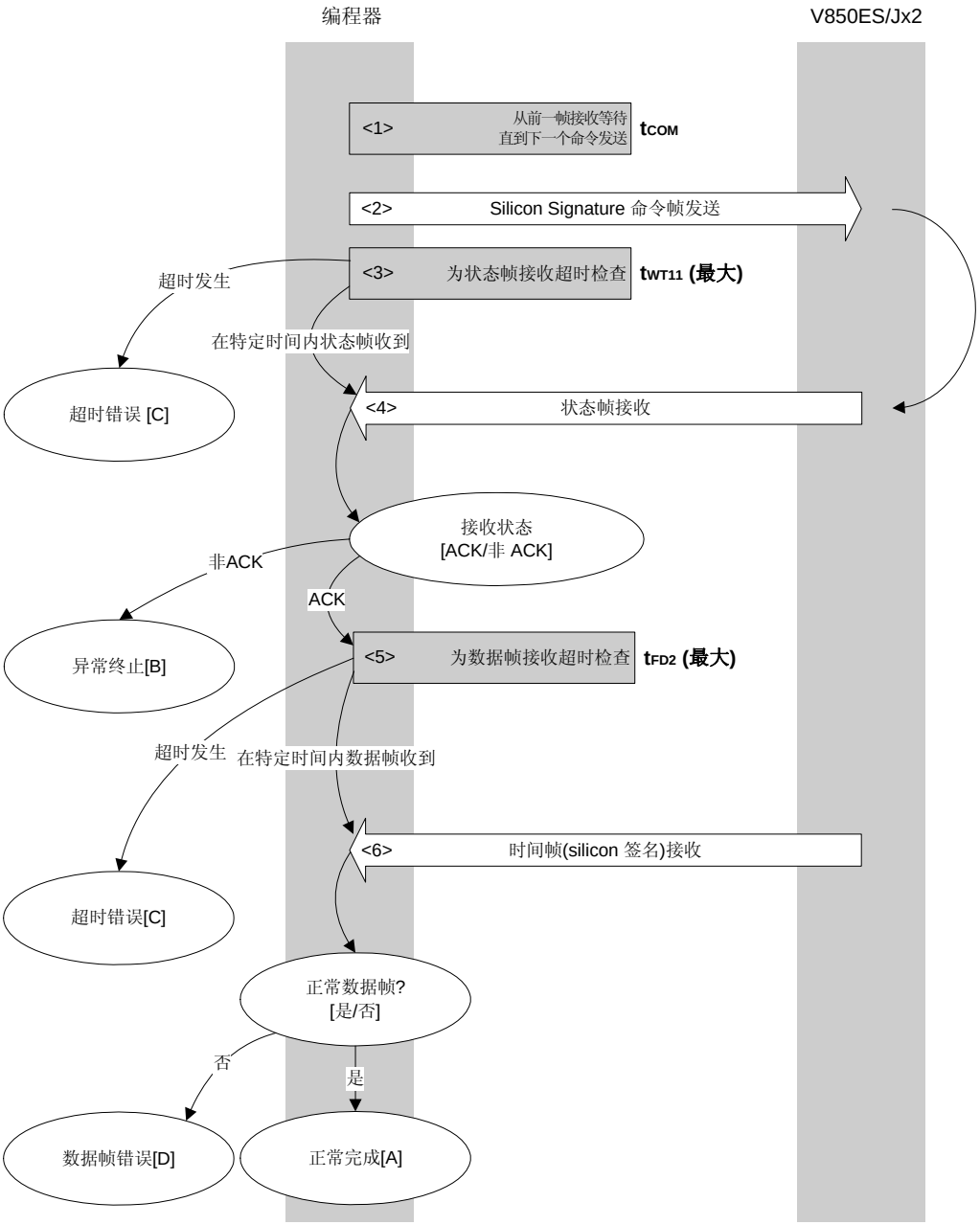
以下表示对一个块的 Block Blank Check 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 块空白检查命令                                                */
/*                                                               */
/*****************************************************************/
/* [i]  u8 block            ... 块号                             */
/* [r] u16                  ... 错误码                           */
/*****************************************************************/
u16             fl_ua_blk_blank_chk(u8 block)
{
        u16     rc;
        u32     wt8_max;

        fl_cmd_prm[0] = block;      // "BLK"
        wt8_max = get_wt8_max(get_block_size(block));

        fl_wait(tCOM_UA);                       // 在发送命令前等待

        put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);


        rc = get_sfrm_ua(fl_ua_sfrm, wt8_max);          // 获取状态帧
//      switch(rc) {
//
//              case            FLC_NO_ERR:         return  rc;     break;  // 如果［A］
//              case            FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
//              default:                            return  rc;     break;  // 如果［B］
//      }
        return rc;

}
```

6.12 Silicon Signature 命令

6.12.1 处理顺序图

Silicon Signature 命令处理顺序



6.12.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Silicon Signature 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT11}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

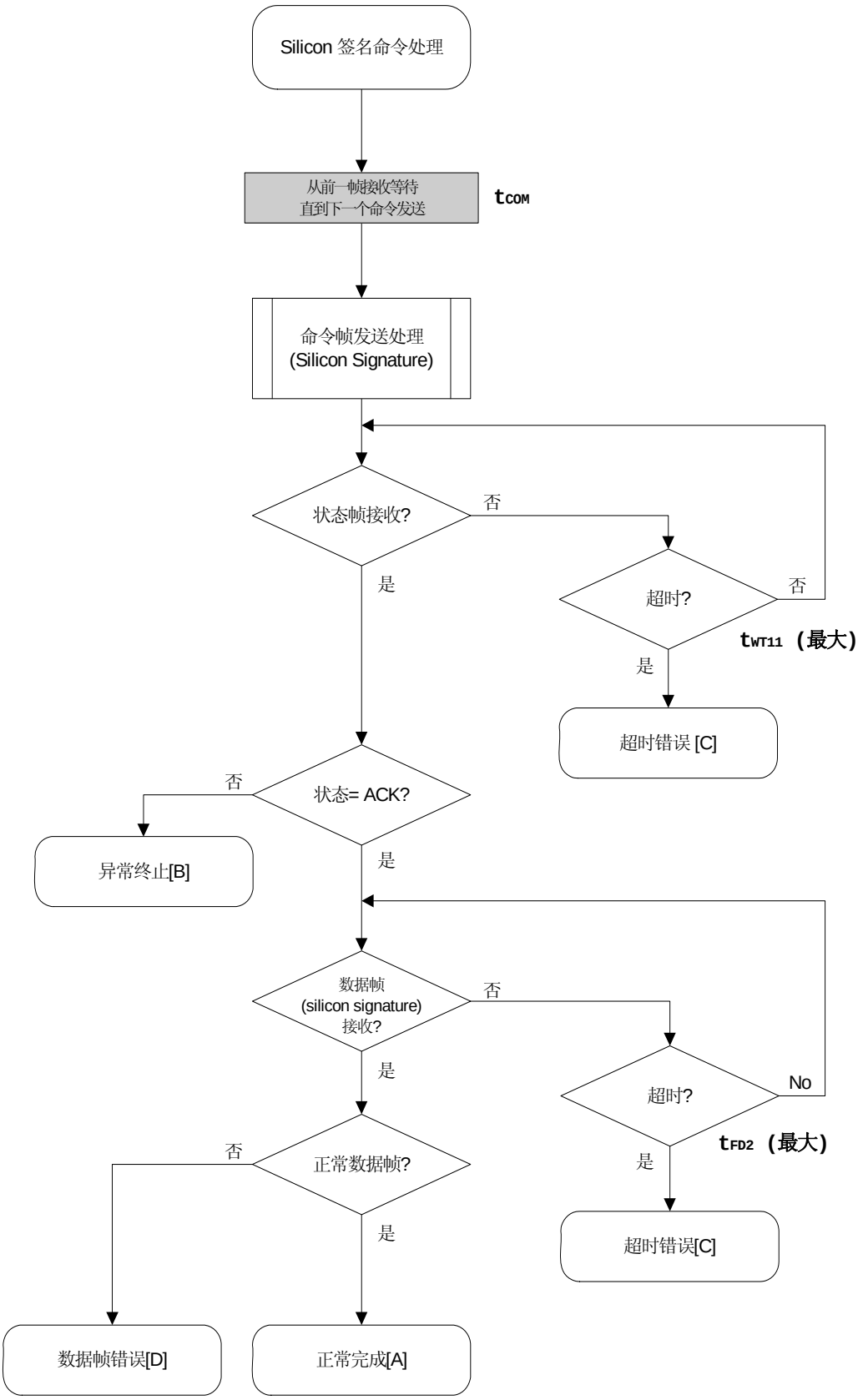
<5> 超时检查被执行直到数据帧（silicon 签名数据）接收。
如果一个超时发生，返回超时错误［C］（超时时间 t_{FD2}（最大））。
<6> 检查接收的数据帧（silicon 签名数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 ［D］

6.12.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且 silicon 签名被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧或数据帧在特定时间内没有被接收。
数据帧错误 ［D］		–	作为 silicon 签名数据接收到的数据帧的校验和不匹配。

6.12.4 流程图


Silicon 签名命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Silicon Signature)
状态帧接收?
否
是
超时?
否
tWT11 (最大)
是
超时错误 [C]
状态= ACK?
否
异常终止[B]
是
数据帧
(silicon signature)
接收?
否
是
超时?
No
tFD2 (最大)
是
超时错误[C]
正常数据帧?
否
是
数据帧错误[D]
正常完成[A]

6.12.5 范例程序

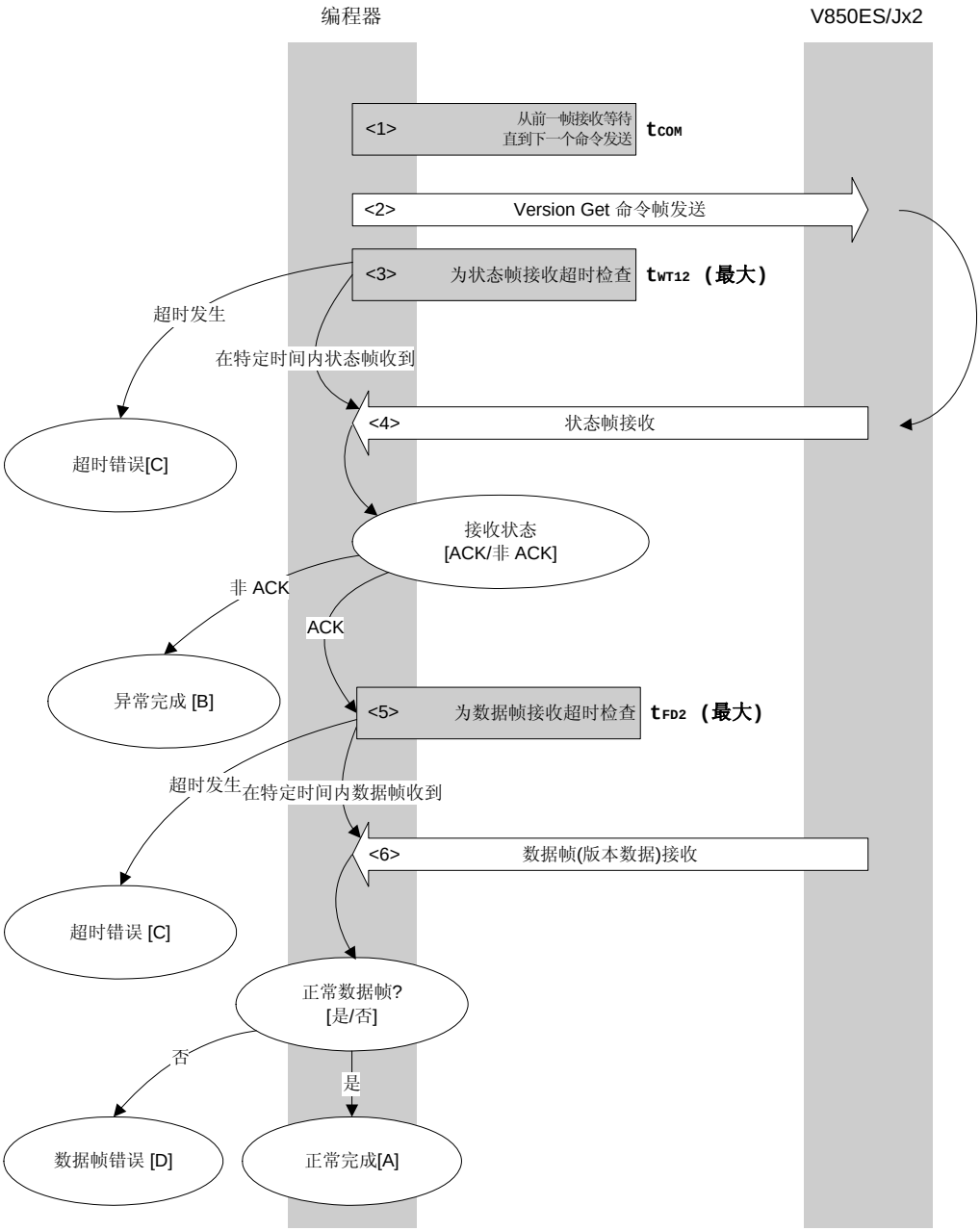
以下表示 Silicon Signature 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获得 silicon 签名命令                                          */
/*                                                               */
/*****************************************************************/
/* [i] u8 *sig          ... 签名保存区域的指针                    */
/* [r] u16              ... 错误码                                */
/*****************************************************************/
u16         fl_ua_getsig(u8 *sig)
{
    u16     rc;


    fl_wait(tCOM_UA);                   // 在发送命令前等待

    put_cmd_ua(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm); // 发送 GET SIGNATURE 命令

    rc = get_sfrm_ua(fl_ua_sfrm, tWT11_MAX);    // 获取状态帧
    switch(rc) {
        case    FLC_NO_ERR:                 break;  // 继续
    //  case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
        case    default:            return  rc;     break;  // 如果［B］
    }


    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX); // 获取状态帧
    if (rc){                            // 如果错误
        return rc;                      // 如果［D］
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                    // 复制签名数据
    return  rc;                         // 如果［A］
}
```

6.13 Version Get 命令

6.13.1 处理顺序图

Version Get 命令处理顺序



6.13.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Version Get 命令。
<3> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT12}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

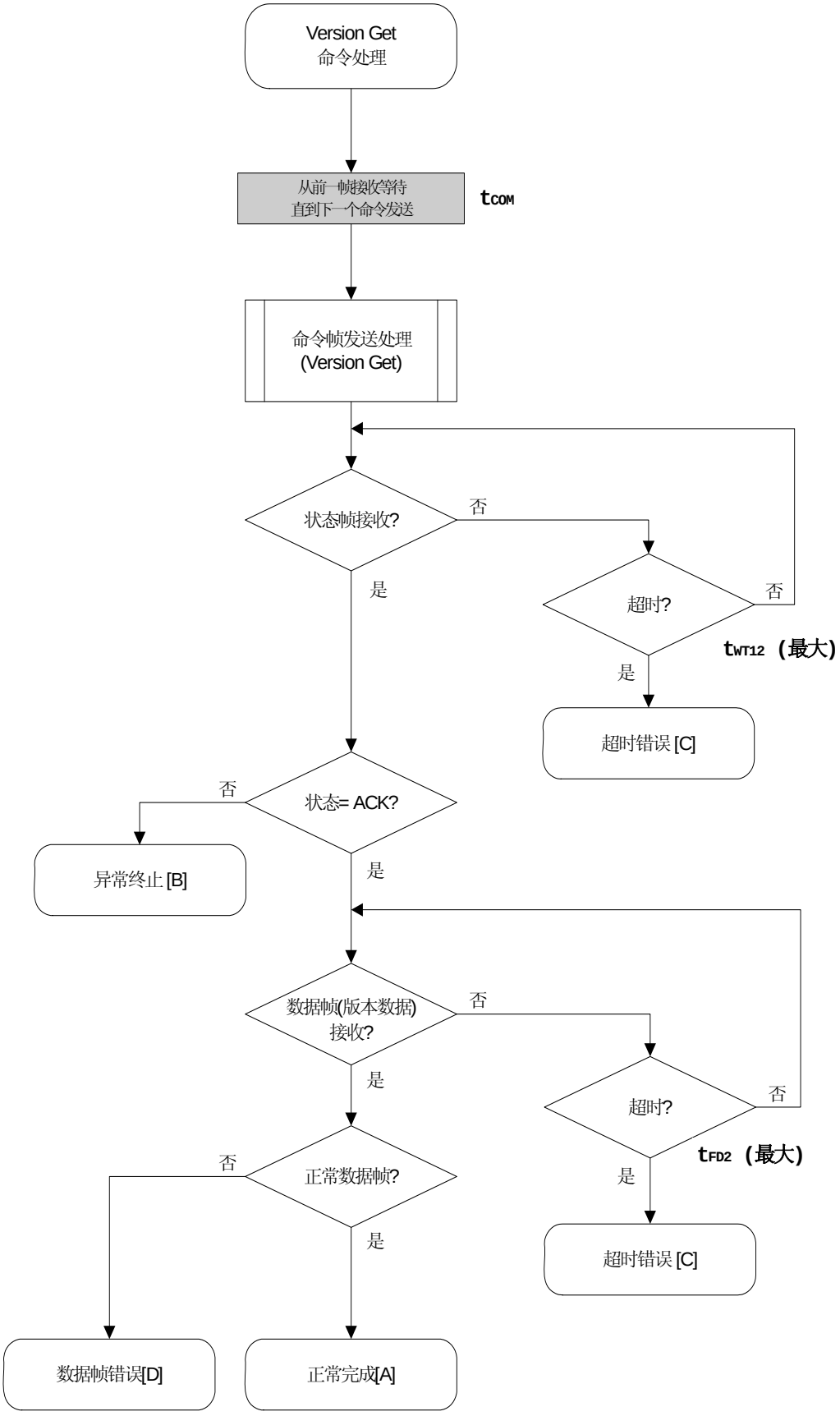
<5> 接收一个超时检查被执行直到数据帧（版本数据）。
如果一个超时发生，返回超时错误［C］（超时时间 t_{FD2}（最大））。
<6> 检查接收到的数据帧（版本数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 ［D］

6.13.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且版本数据被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧或数据帧在特定时间内没有被接收。
数据帧错误 ［D］		–	作为版本数据接收到的数据帧的校验和不匹配。

6.13.4 流程图


Version Get
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Version Get)
状态帧接收?
否
是
超时?
否
是
tWT12 (最大)
超时错误 [C]
状态= ACK?
否
异常终止 [B]
是
数据帧(版本数据)
接收?
否
是
超时?
否
是
tFD2 (最大)
超时错误 [C]
正常数据帧?
否
是
数据帧错误[D]
正常完成[A]

6.13.5 范例程序

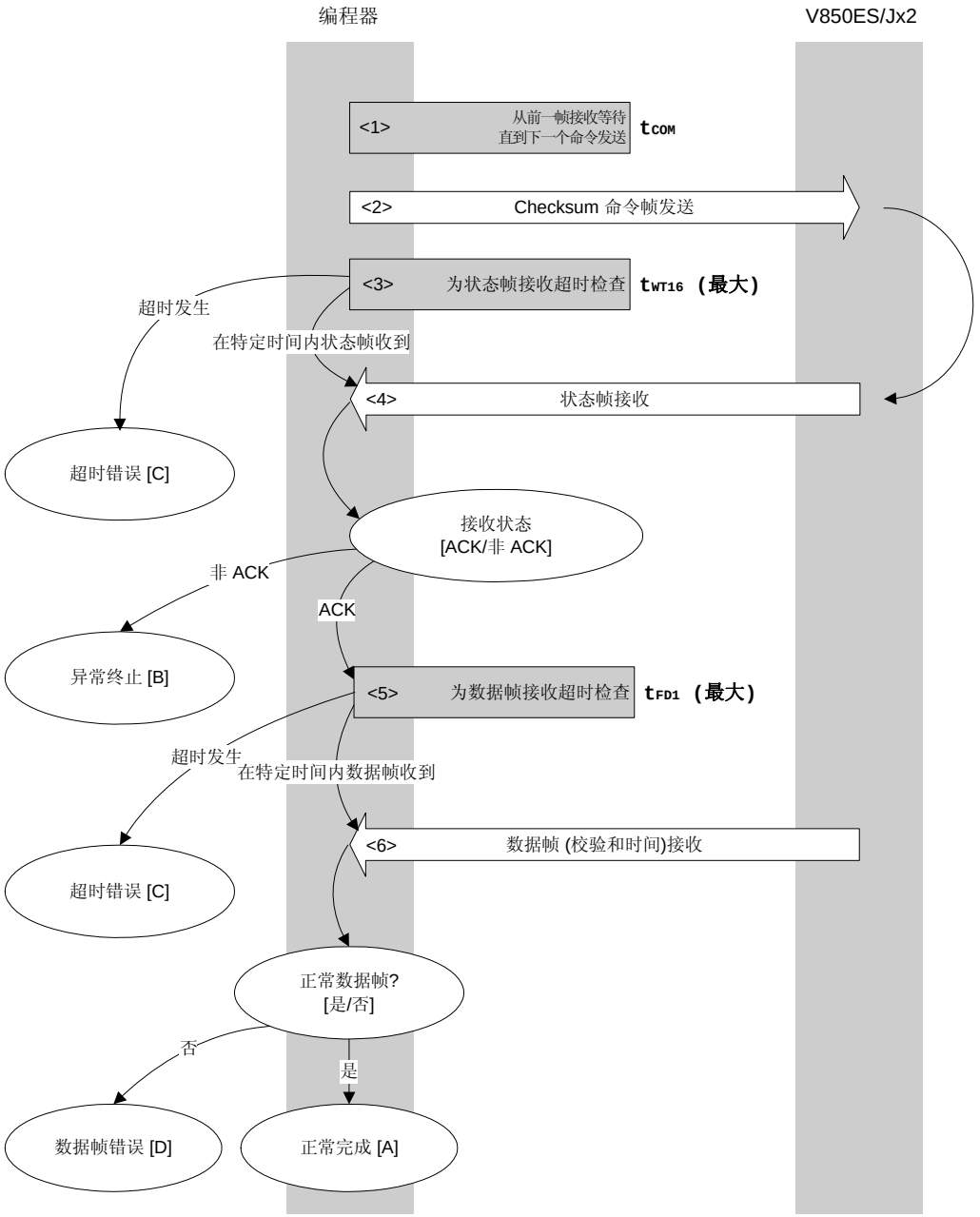
以下表示 Version Get 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获得设备/固件版本命令                                         */
/*                                                               */
/*****************************************************************/
/* [i] u8 *buf          ... 版本数据保存区域的指针               */
/* [r] u16              ... 错误码                               */
/*****************************************************************/
u16             fl_ua_getver(u8 *buf)
{
        u16     rc;


        fl_wait(tCOM_UA);                       // 在发送命令前等待


        put_cmd_ua(FL_COM_GET_VERSION, 1, fl_cmd_prm); // 发送 GET VERSION 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT12_MAX);  // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
                default:                        return  rc;     break;  // 如果［B］
        }

        rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX); // 获得数据帧
        if (rc){
                return rc;                      // 如果［D］
        }

        memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// 复制版本数据
        return rc;                              // 如果［A］
}
```

6.14 Checksum 命令

6.14.1 处理顺序图

Checksum 命令处理顺序



6.14.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Checksum 命令。
<3> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT16}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

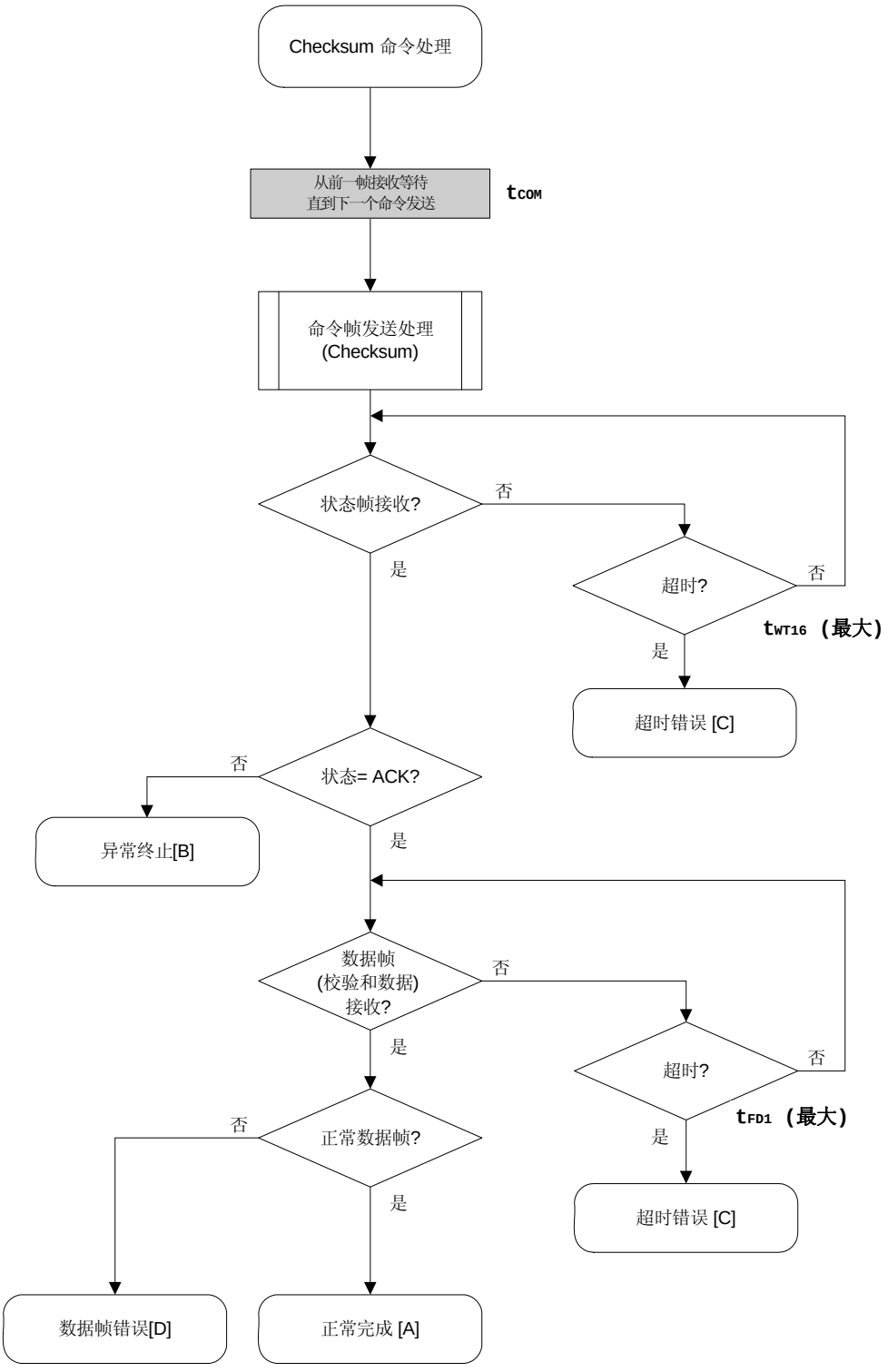
<5> 一个超时检查被执行直到数据帧（校验和数据）接收。
如果一个超时发生，返回超时错误［C］（超时时间 t_{FD1}（最大））。
<6> 检查接收的数据帧（校验和数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 ［D］

6.14.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验和数据被正常获取。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧或数据帧在特定时间内没有被接收。
数据帧错误 ［D］		–	作为版本数据接收到的数据帧的校验和不匹配。

6.14.4 流程图


Checksum 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Checksum)
状态帧接收?
否
是
超时?
否
tWT16 （最大）
是
超时错误 [C]
状态= ACK?
否
异常终止[B]
是
数据帧
(校验和数据)
接收?
否
是
超时?
否
tFD1 （最大）
是
超时错误 [C]
正常数据帧?
否
是
数据帧错误[D]
正常完成 [A]

6.14.5 范例程序

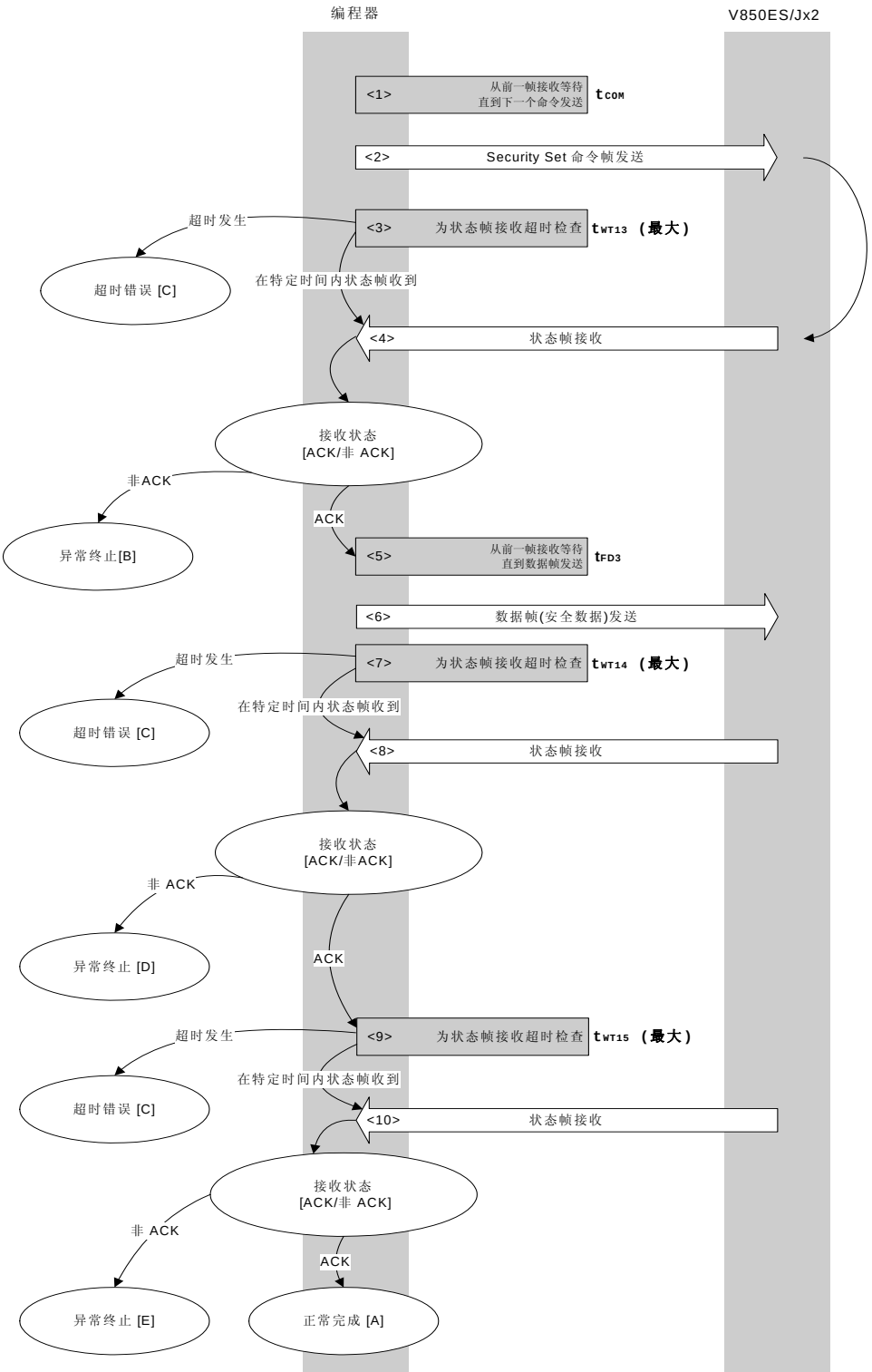
以下表示 Checksum 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/*获得校验和命令                                                 */
/*                                                               */
/*****************************************************************/
/* [i] u16 *sum          ... 校验和保存区域指针                  */
/* [i] u32 top           ... 起始地址                            */
/* [i] u32 bottom        ... 结束地址                            */
/* [r] u16               ... 错误码                              */
/*****************************************************************/
u16             fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
        u16     rc;

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


        /*************************************************/
        /*      发送命令                                 */
        /*************************************************/

        fl_wait(tCOM_UA);                           // 在发送命令前等待

        put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // 发送 GET VERSION 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT16_MAX);  // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
                default:                        return  rc;     break;  // 如果［B］
        }

        /*************************************************/
        /*      获得数据帧（校验和数据）    */             */
        /*************************************************/
        rc = get_dfrm_ua(fl_rxdata_frm, tFD1_MAX); // 获取状态帧
        if (rc){                                            // 如果没有错误，
                return rc;                              // 如果［D］
        }

        *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1];
                                                            // 设置 SUM 数据
        return rc;                                      // 如果［A］

}
```

6.15 Security Set 命令

6.15.1 处理顺序图

Security Set 命令处理顺序



6.15.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Security Set 命令。
<3> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT13}（最大））。
<4> 状态码被检查。

当 ST1 = ACK 时： 到<5>。
当 ST1 ≠ ACK 时： 异常终止［B］

<5> 从前一帧接收等待直到下一个数据帧发送（等待时间 t_{FD3}）。
<6> 通过数据帧发送处理发送数据帧（安全设置数据）命令。
<7> 执行一个超时检查直到状态帧接收。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT14}（最大））。
<8> 检查状态码。

当 ST1 = ACK 时： 到<9>。
当 ST1 ≠ ACK 时： 异常终止［D］

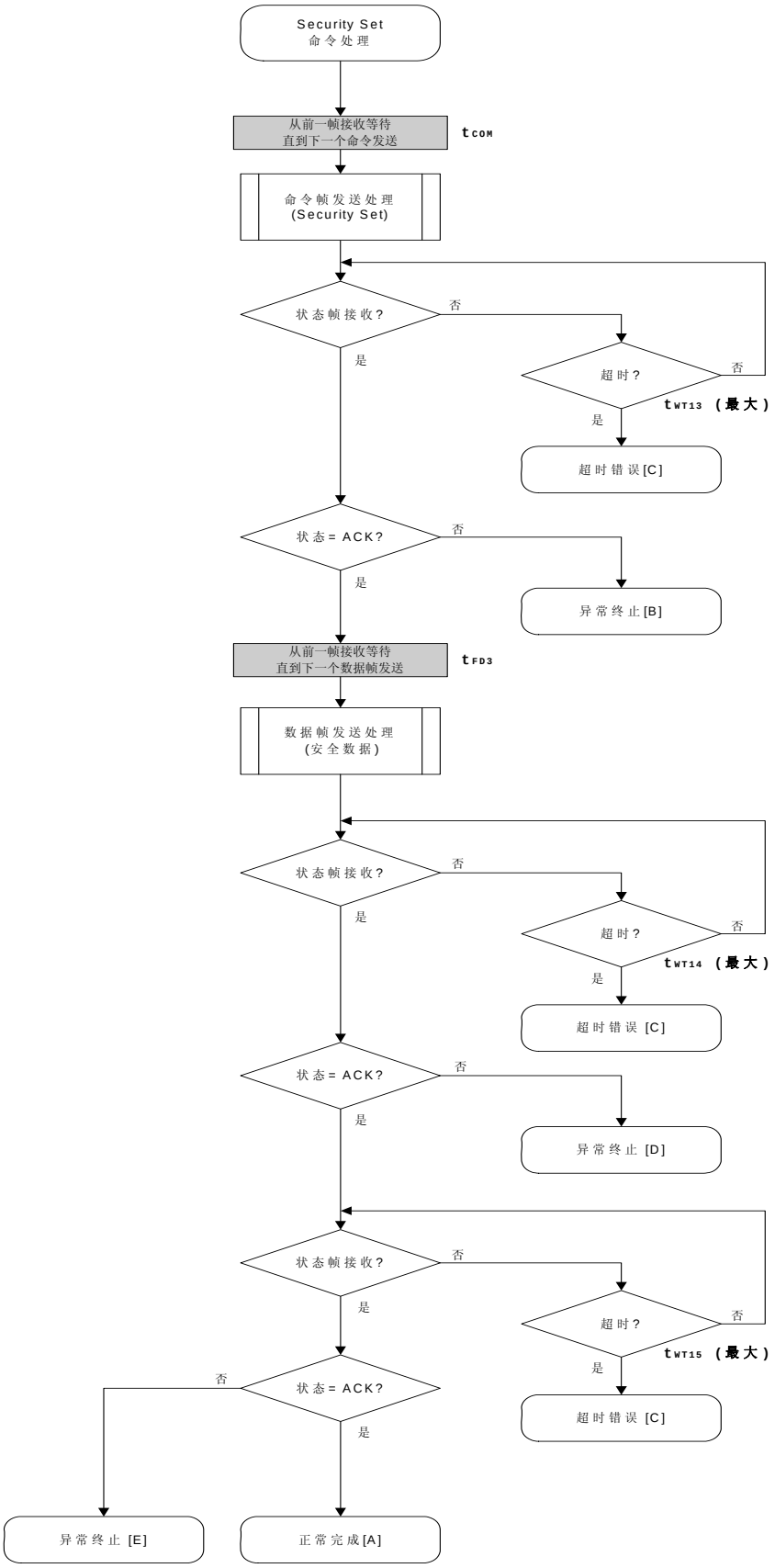
<9> 执行一个超时检查直到状态帧接收。
如果一个超时发生，返回超时错误［C］（超时时间 t_{WT15}（最大））。
<10> 检查状态码。

当 ST1 = ACK 时： 正常完成［A］
当 ST1 ≠ ACK 时： 异常终止［E］

6.15.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且安全设置被正常完成。
异常终止［B］	参数错误	05H	命令信息（参数）不是 00H。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	ID 码不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧或数据帧在特定时间内没有被接收。
异常终止［D］	WWV1 错误	08H	• 安全数据已经设置。 • 一个安全数据写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止［E］	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

6.15.4 流程图


Security Set
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Security Set)
状态帧接收?
否
是
超时?
否
tWT13 (最大)
是
超时错误[C]
状态= ACK?
否
是
异常终止[B]
从前一帧接收等待
直到下一个数据帧发送
tFD3
数据帧发送处理
(安全数据)
状态帧接收?
否
是
超时?
否
tWT14 (最大)
是
超时错误 [C]
状态= ACK?
否
是
异常终止 [D]
状态帧接收?
否
是
超时?
否
tWT15 (最大)
是
超时错误 [C]
状态= ACK?
否
是
异常终止 [E]
正常完成[A]

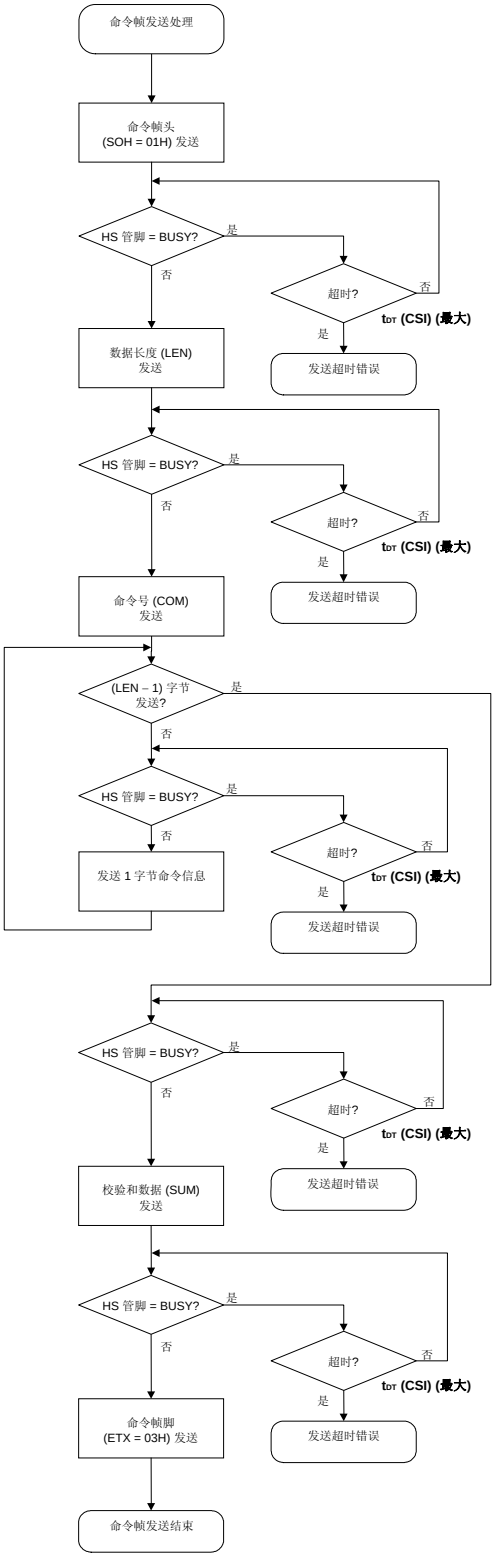
6.15.5 范例程序

以下表示 Security Set 命令处理的一个范例程序。

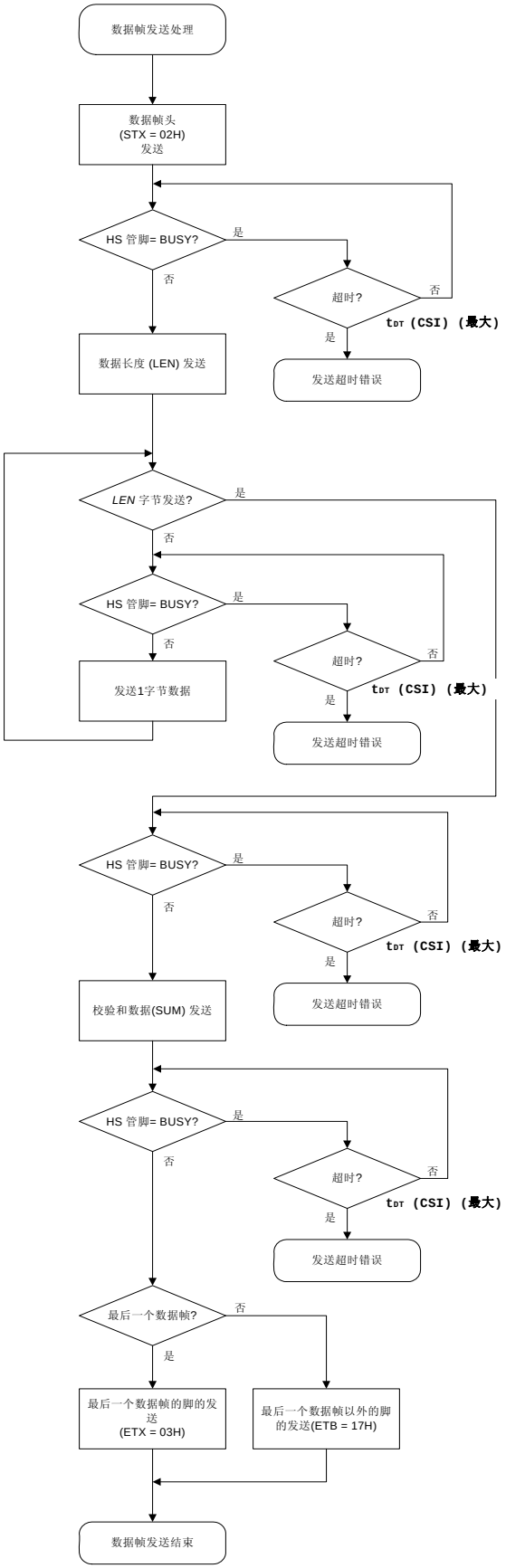
```
/*****************************************************************/
/*                                                               */
/*设置安全标志命令                                               */
/*                                                               */
/*****************************************************************/
/* [i] u8 scf           ... 安全标志数据                         */
/* [r] u16              ... 错误码                               */
/*****************************************************************/
u16             fl_ua_setscf(u8 scf)
{
        u16     rc;

        /*************************************************/
        /*      设置参数                                 */
        /*************************************************/
        fl_cmd_prm[0] = 0x00;                   // 第一个字节必须为 0x00
        fl_cmd_prm[1] = 0x00;                   // 第二个字节必须为 0x00
        fl_txdata_frm[0] = scf|= 0b11111000;
                                                // "FLG" (高 5 位必须为'1' (要确认))


        /*************************************************/
        /*      发送命令                                 */
        /*************************************************/
        fl_wait(tCOM_UA);               // 在发送命令前等待

        put_cmd_ua(FL_COM_SET_SECURITY, 3, fl_cmd_prm);

        rc = get_sfrm_ua(fl_ua_sfrm, tWT13_MAX);  // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
                default:                        return  rc;     break;  // 如果［B］
        }


        /*************************************************/
        /*      /*  发送数据帧（安全设置数据） */           */
        /*************************************************/

        fl_wait(tFD3);
        put_dfrm_ua(1, fl_txdata_frm, true); // 发送安全设置数据

        rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX);  // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
                default:                        return  rc;     break;  // 如果［B］
        }

        /*************************************************/
        /*      检查内部校验                             */
        /*************************************************/
```

```
        rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);  // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:                 return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:               return  rc;     break;  // 如果［C］
//              default:                            return  rc;     break;  // 如果［B］
//      }
        return rc;
}
```

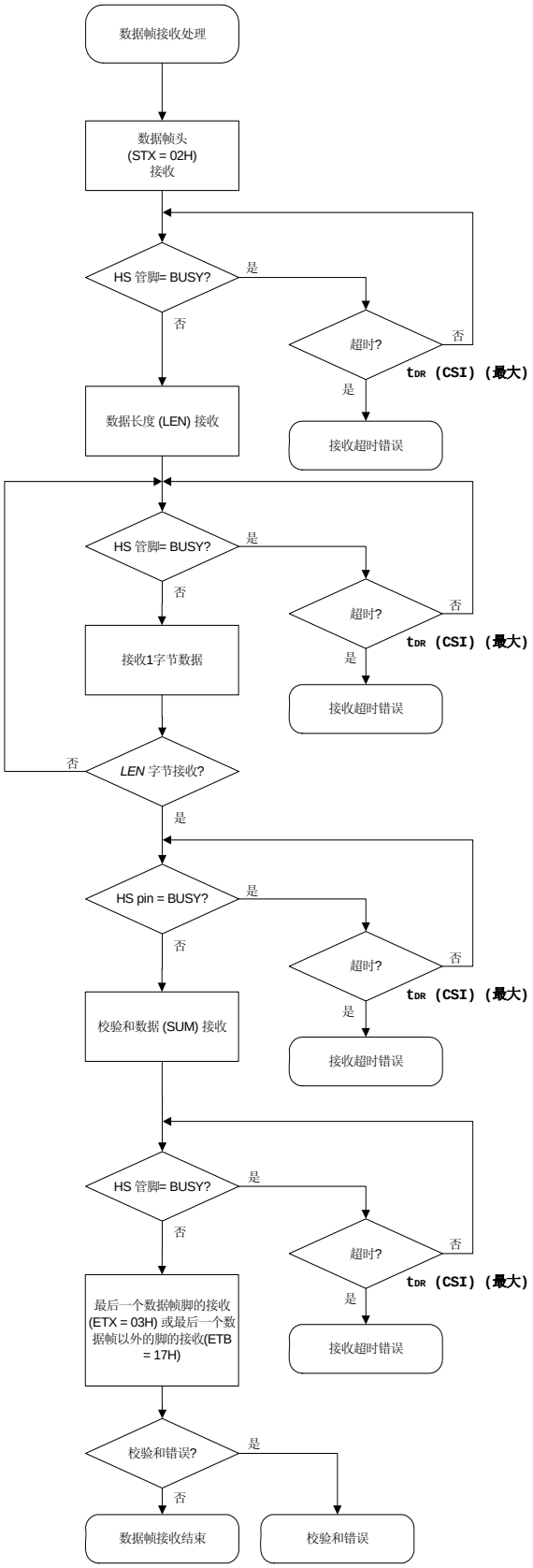
7.1 命令帧发送处理流程图



7.2 数据帧发送处理流程图



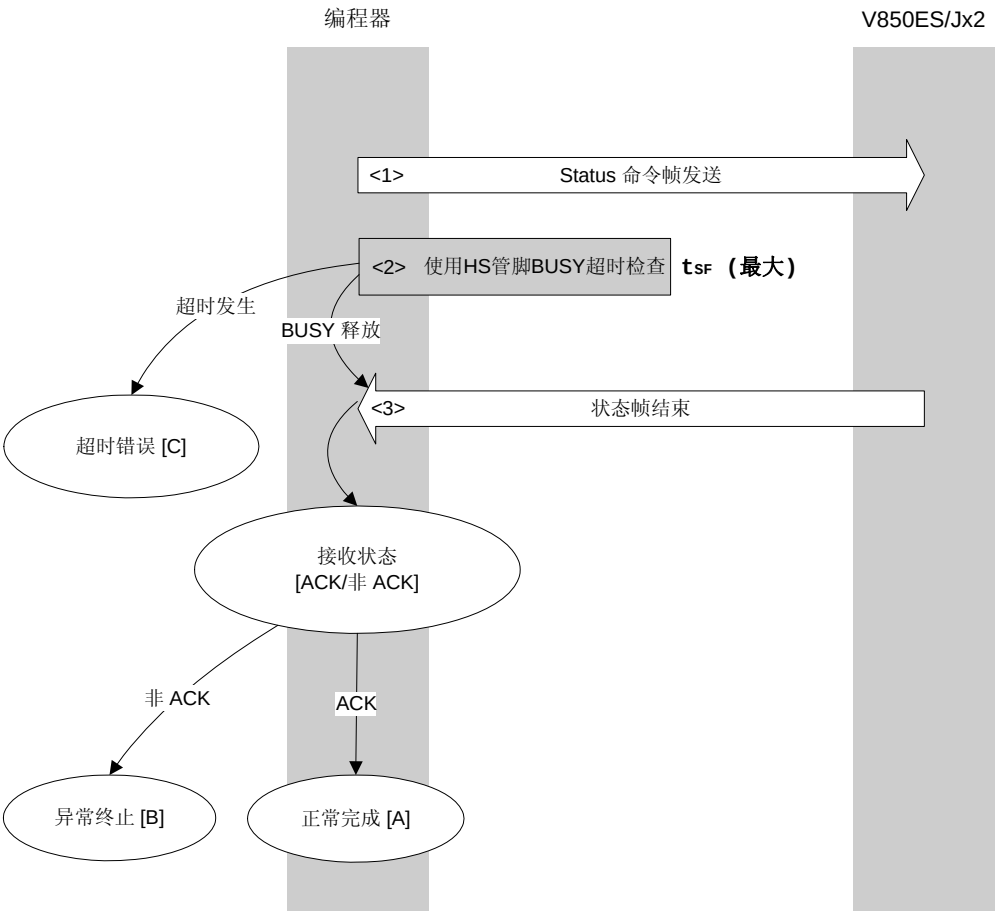
7.3 数据帧接收处理流程图



7.4 Status命令

7.4.1 处理顺序图

Status 命令处理顺序



7.4.2 处理顺序的说明

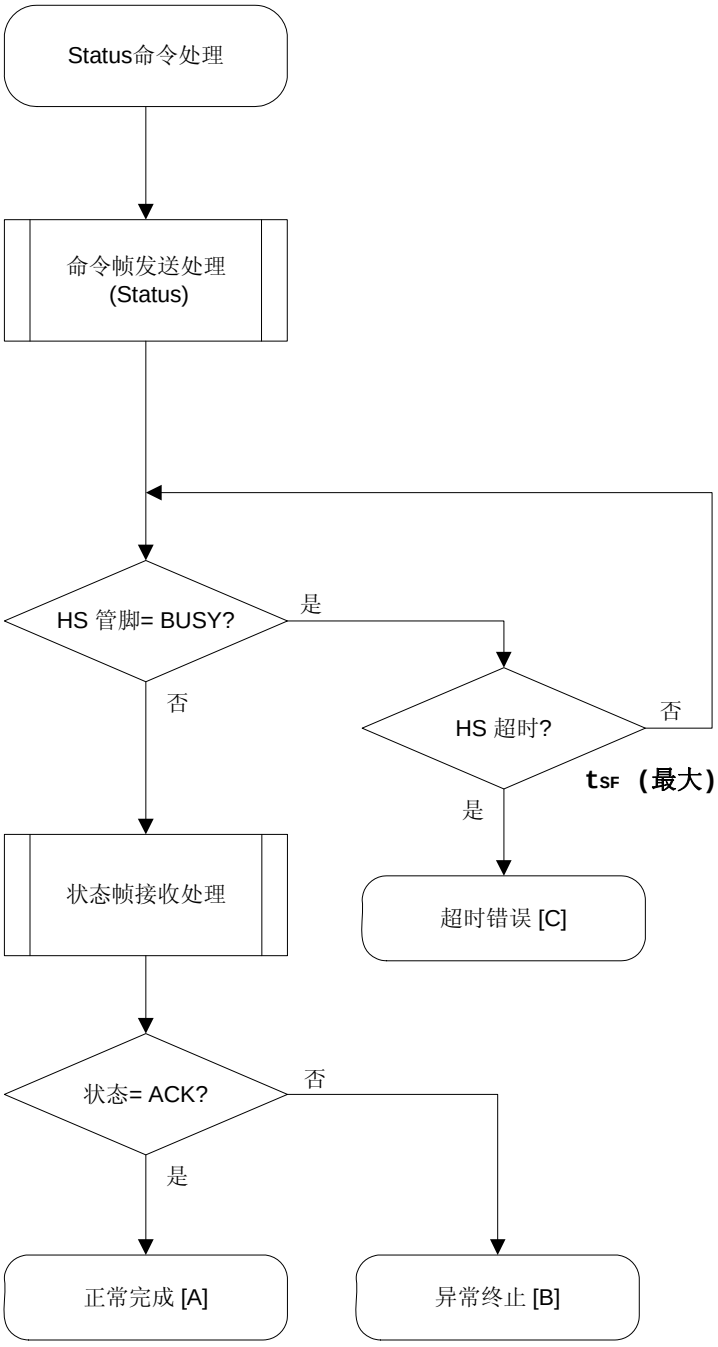
<1> 通过命令帧发送处理发送 Status 命令。
<2> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{SF}（最大））。
<3> 检查状态码。

当 ST1 = ACK 时： 正常完成［A］
当 ST1 ≠ ACK 时： 异常终止［B］

7.4.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	从 V850ES/Jx2 发送的状态帧被正常接收。
异常终止［B］	命令错误	04H	一个不支持的命令或异常帧被接收。
	参数错误	05H	命令信息（参数）无效。
	校验和错误	07H	从编程器发送的帧的数据异常。
	WWV1 错误	08H	写入错误
	EWV1 错误	0BH	擦除错误
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	校验错误	0EH	对于从编程器发送的帧的数据，一个校验错误发生。
		0FH	
	产品错误	10H	试图执行 Security Set 命令禁止的处理。
	EWV4 错误	11H	内部校验错误/空白错误
	压缩搜索错误	13H	擦除错误
	消极响应（NACK）	15H	消极响应
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。

7.4.4 流程图


Status命令处理
命令帧发送处理
(Status)
HS 管脚= BUSY?
是
否
HS 超时?
否
tSF （最大）
是
状态帧接收处理
超时错误 [C]
状态= ACK?
否
是
正常完成 [A]
异常终止 [B]

7.4.5 范例程序

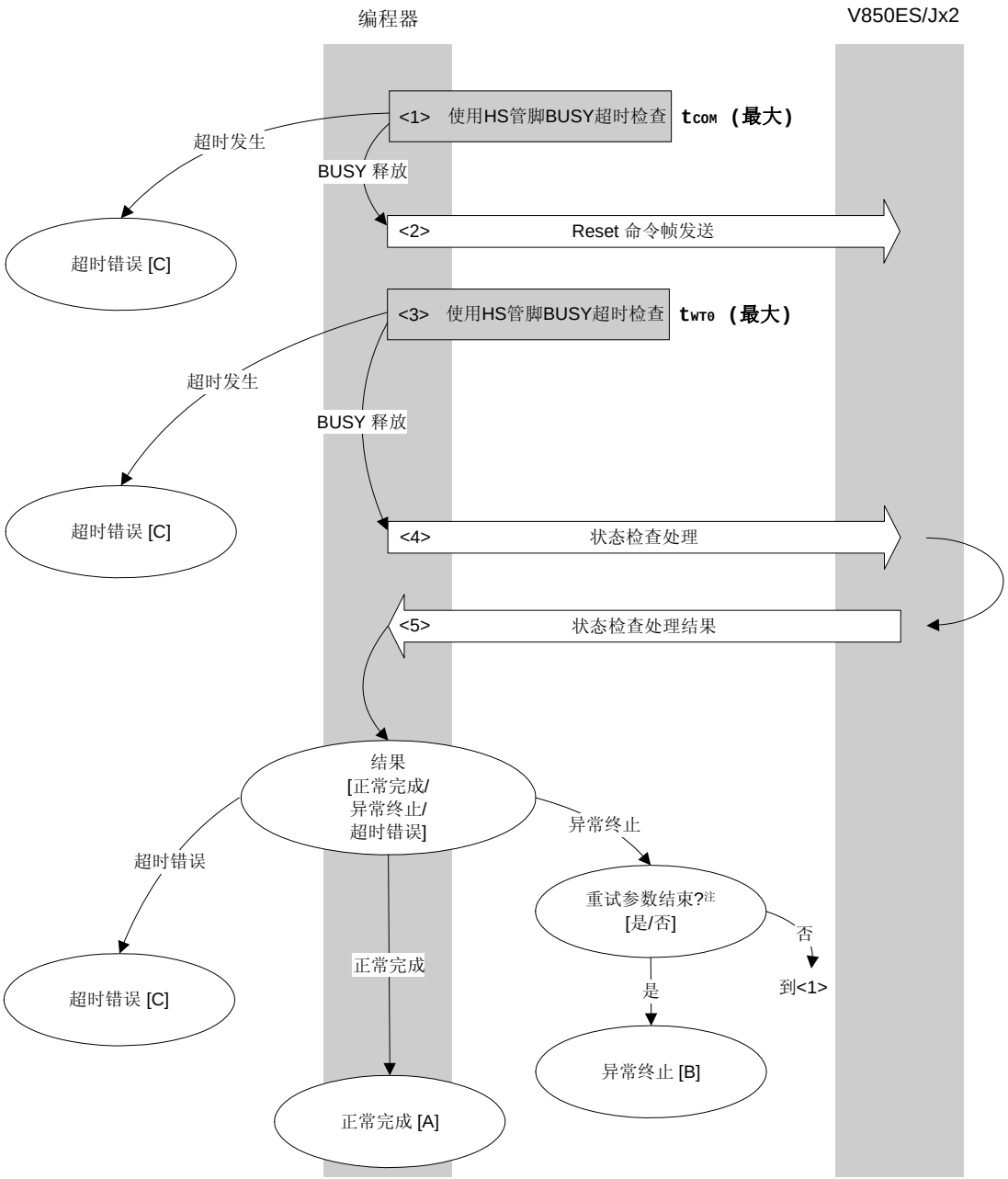
以下表示 Status 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/*  获取状态命令 (CSI-HS)                                        */
/*                                                               */
/*****************************************************************/
/*  [r] u16      ... 解码状态或错误码                             */
/*                                                                */
/*  (见 fl.h/fl-proto.h &                                         */
/*         fl.c 中 decode_status()的定义)                         */
/*****************************************************************/
static u16      fl_hs_getstatus(void)
{
        u16     rc;
        u32     retry = 0;



        rc = put_cmd_hs(FL_COM_GET_STA, 1, fl_cmd_prm);          // 发送 "Status" 命令
        if (rc)
                return  rc;       // case ［C］

        if (hs_busy_to(tSF_MAX))            // HS 忙超时?
                return  FLC_HSTO_ERR;                    // 超时检测：如果 ［C］

        if (rc = get_sfrm_hs(fl_rxdata_frm))
                return  rc;                 // 如果 ［C］ 或 [B(校验和错误)]


        rc = decode_status(fl_st1); // 解码返回码

        return rc;                  // 如果 ［A］ 或 ［B］
}
```

7.5 Reset 命令

7.5.1 处理顺序图

Reset 命令处理顺序



注 不要超过复位命令发送的重试次数（最多 16 次）。

7.5.2 处理顺序的说明

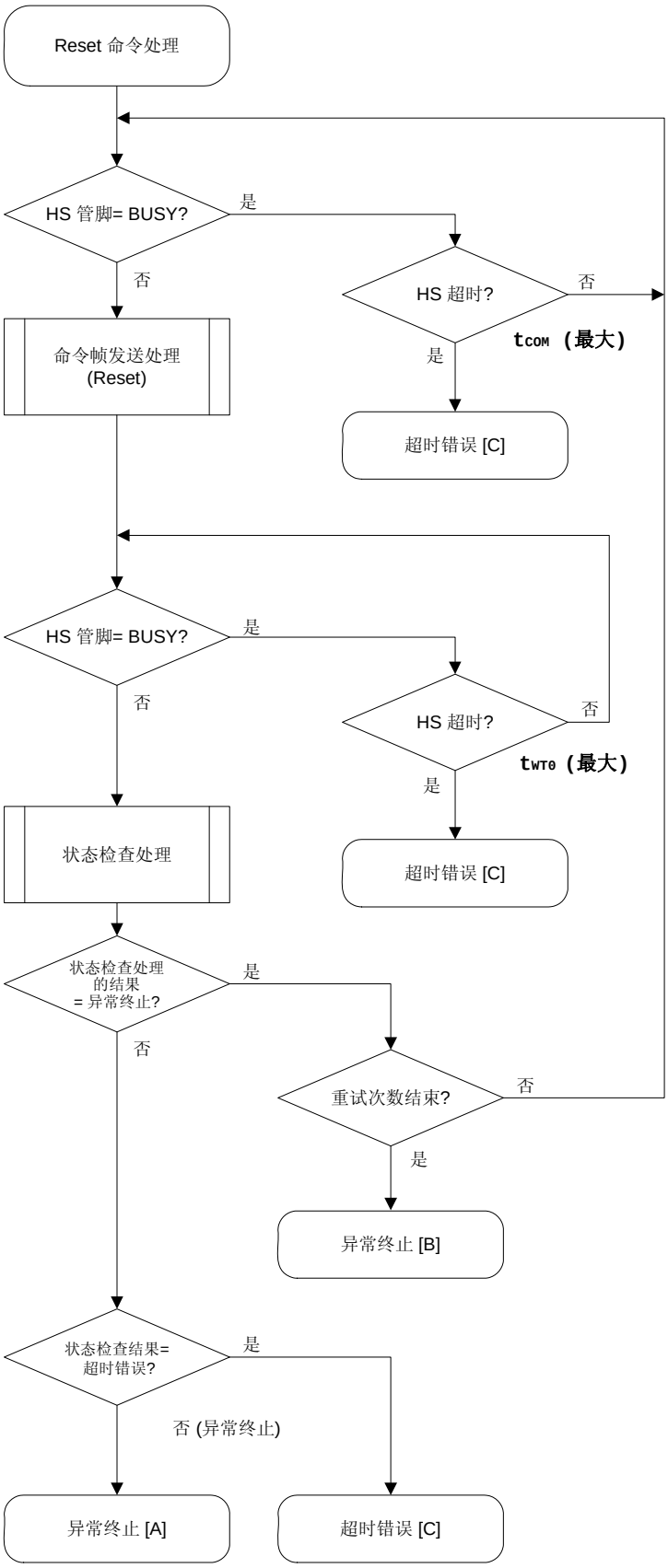
<1> 使用 HS 管脚，一个 V850ES/Jx2 BUSY 状态被检查。
如果一个超时发生，超时错误［C］被返回（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发出 Reset 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT0}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］
当处理异常结束时：	如果重试次数没有结束，顺序从<1>被重新执行。 如果重试次数结束，处理异常结束［B］。
当超时错误发生时：	一个超时错误［C］被返回。

7.5.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且已经建立编程器和 V850ES/Jx2 之间的同步。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态检查处理超时。 由于 HS 管脚的忙状态，处理超时。

7.5.4 流程图


Reset 命令处理
HS 管脚= BUSY?
是
否
HS 超时?
否
tCOM（最大）
是
命令帧发送处理
(Reset)
超时错误 [C]
HS 管脚= BUSY?
是
否
HS 超时?
否
tWT0（最大）
是
状态检查处理
超时错误 [C]
状态检查处理
的结果
= 异常终止?
是
否
重试次数结束?
否
是
异常终止 [B]
状态检查结果=
超时错误?
是
否 (异常终止)
异常终止 [A]
超时错误 [C]

7.5.5 范例程序

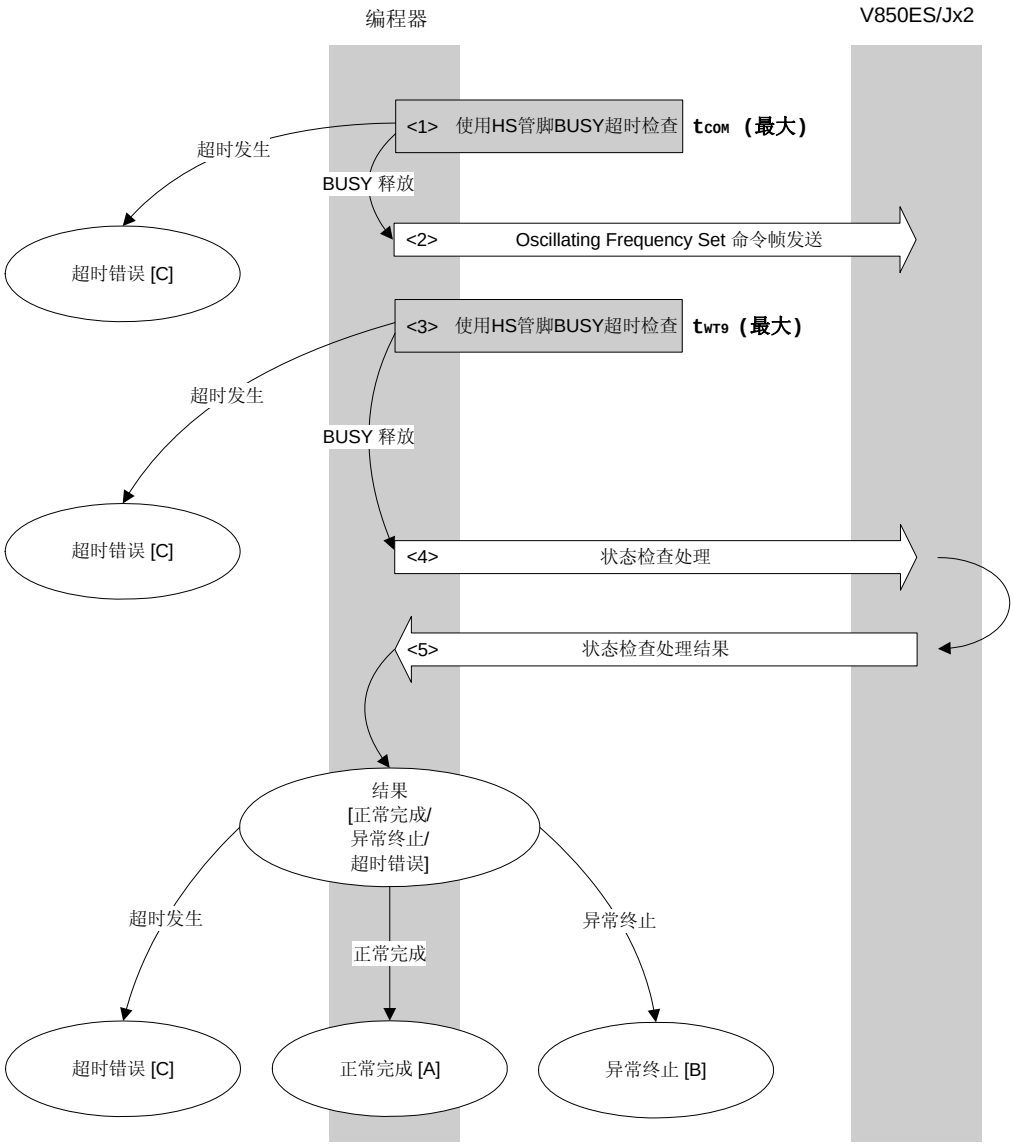
以下表示 Reset 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* Reset 命令 (CSI-HS)                                           */
/*                                                               */
/*****************************************************************/
/* [r] u16          ... 错误码                                   */
/*****************************************************************/
u16             fl_hs_reset(void)
{
        u16     rc;
        u32     retry;

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

                if (hs_busy_to(tCOM_MAX))
                        return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

                rc = put_cmd_hs(FL_COM_RESET, 1, fl_cmd_prm);   // 发送 "Reset" 命令
                if (rc)
                        return  rc;             // 如果 ［C］

                if (hs_busy_to(tWT0_MAX))
                        return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

                rc = fl_hs_getstatus();         // 获取状态帧
                if (rc == FLC_ACK)                      // ST1 = ACK ?
                        break;                  // 如果 ［A］
                //继续;                         // 如果 ［B］ (如果从循环退出)

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

7.6 Oscillating Frequency Set命令

7.6.1 处理顺序图

Oscillating Frequency Set 命令处理顺序



7.6.2 处理顺序的说明

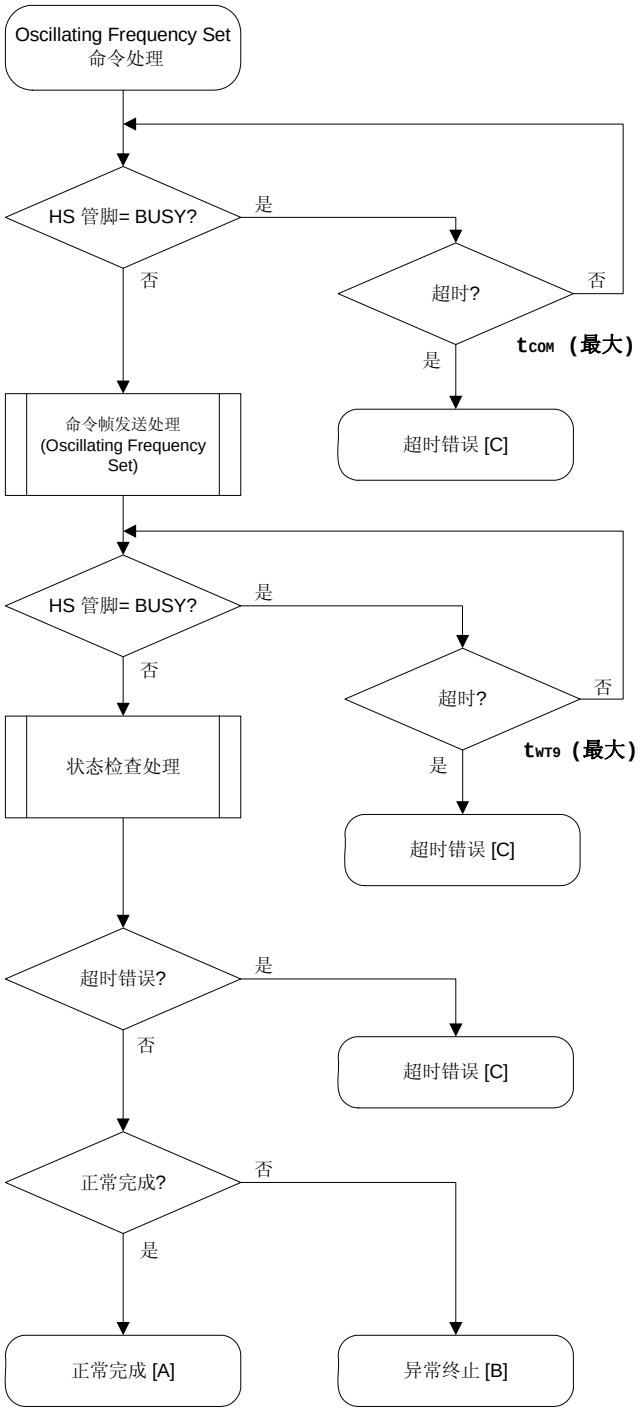
<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Oscillating Frequency Set 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT9}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］
当处理异常结束时：	异常终止［B］
当超时错误发生时：	返回一个超时错误［C］。

7.6.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且工作频率被正确设置到 V850ES/Jx2。
异常终止［B］	参数错误	05H	振荡频率值超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。

7.6.4 流程图


Oscillating Frequency Set
命令处理
HS 管脚= BUSY?
是
否
超时?
否
是
tCOM（最大）
超时错误 [C]
命令帧发送处理
(Oscillating Frequency
Set)
HS 管脚= BUSY?
是
否
超时?
否
是
tWT9（最大）
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
正常完成 [A]
异常终止 [B]

7.6.5 范例程序

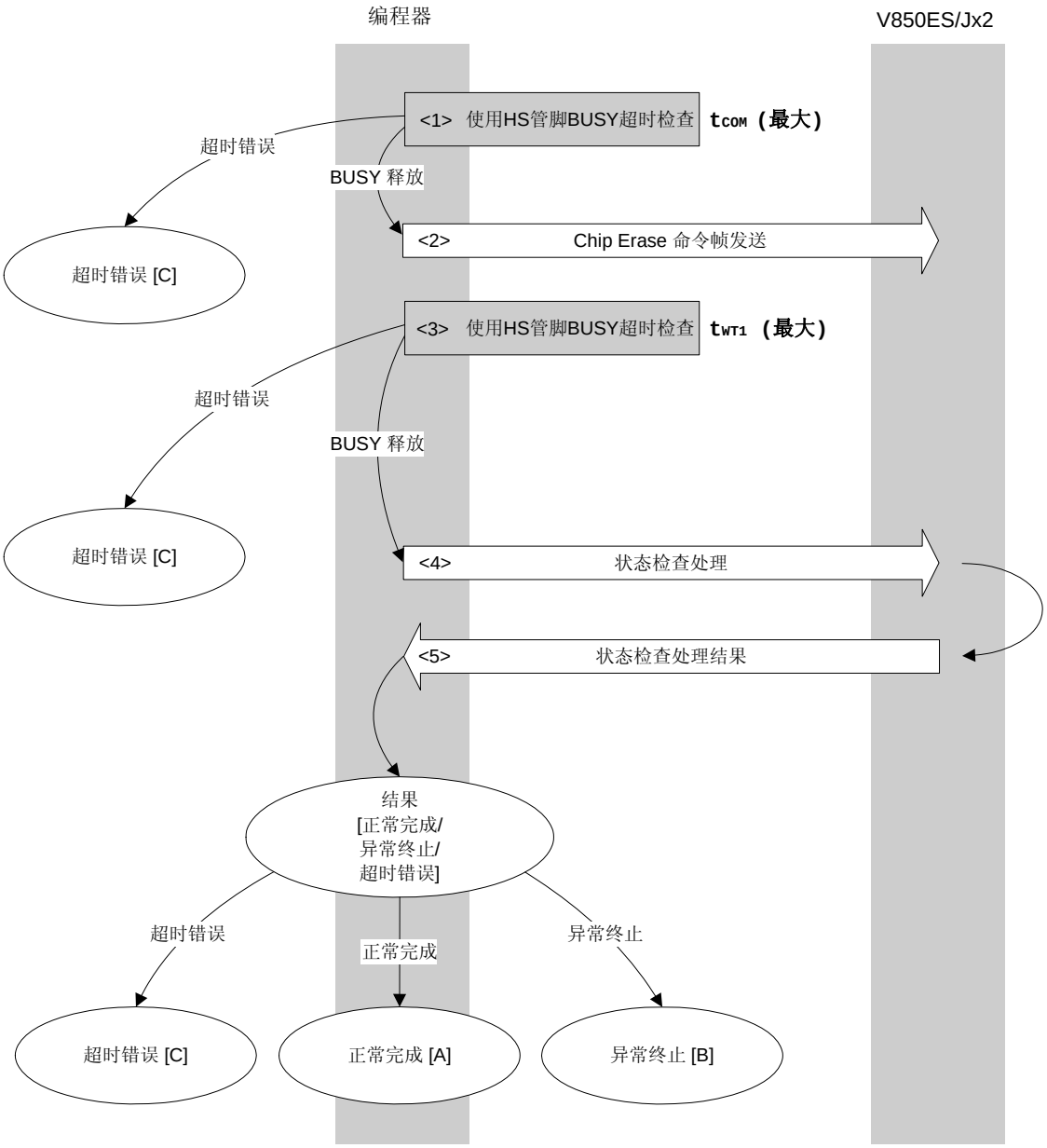
以下表示 Oscillating Frequency Set 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 设置 Flash 器件时钟值命令(CSI-HS)                             */
/*                                                               */
/*****************************************************************/
/* [i] u8 clk[4]  ... 频率数据(D1-D4)                             */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
/*****************************************************************/
u16             fl_hs_setclk(u8 clk[])
{
        u16     rc;

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


        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm))
                                                        // 发送 "Oscilating Frequency Set" 命令
                return  rc;                             // 如果 ［C］

        if (hs_busy_to(tWT9_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］


        rc = fl_hs_getstatus();                 // 获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果 ［A］
//              case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
//              default:                        return  rc;     break;  // 如果 ［B］
//      }
        return rc;
}
```

7.7 Chip Erase 命令

7.7.1 处理顺序图

Chip Erase 命令处理顺序



7.7.2 处理顺序的说明

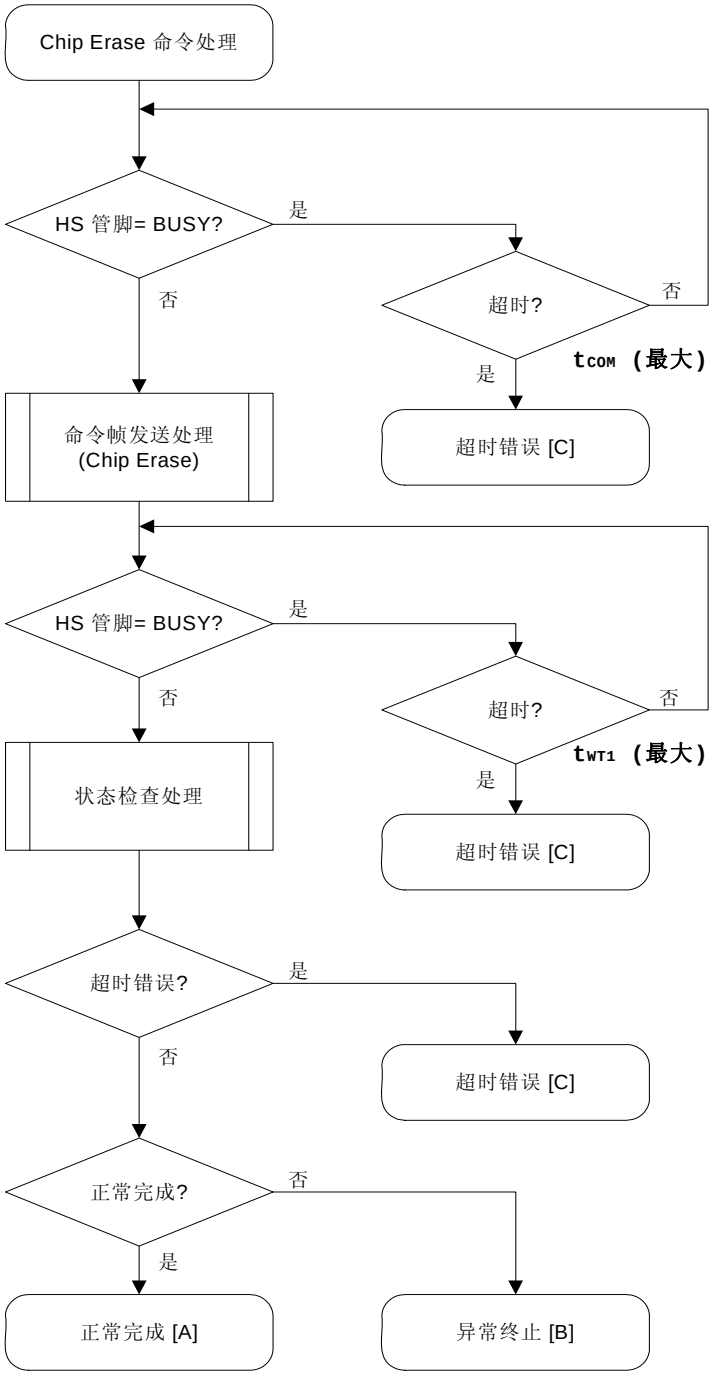
<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Chip Erase 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT1}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

7.7.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且正常完成芯片擦除。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。

7.7.4 流程图


Chip Erase 命令处理
HS 管脚= BUSY?
是
否
超时?
否
是
tCOM（最大）
超时错误 [C]
命令帧发送处理
(Chip Erase)
HS 管脚= BUSY?
是
否
超时?
否
是
tWT1（最大）
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
正常完成 [A]
异常终止 [B]

7.7.5 范例程序

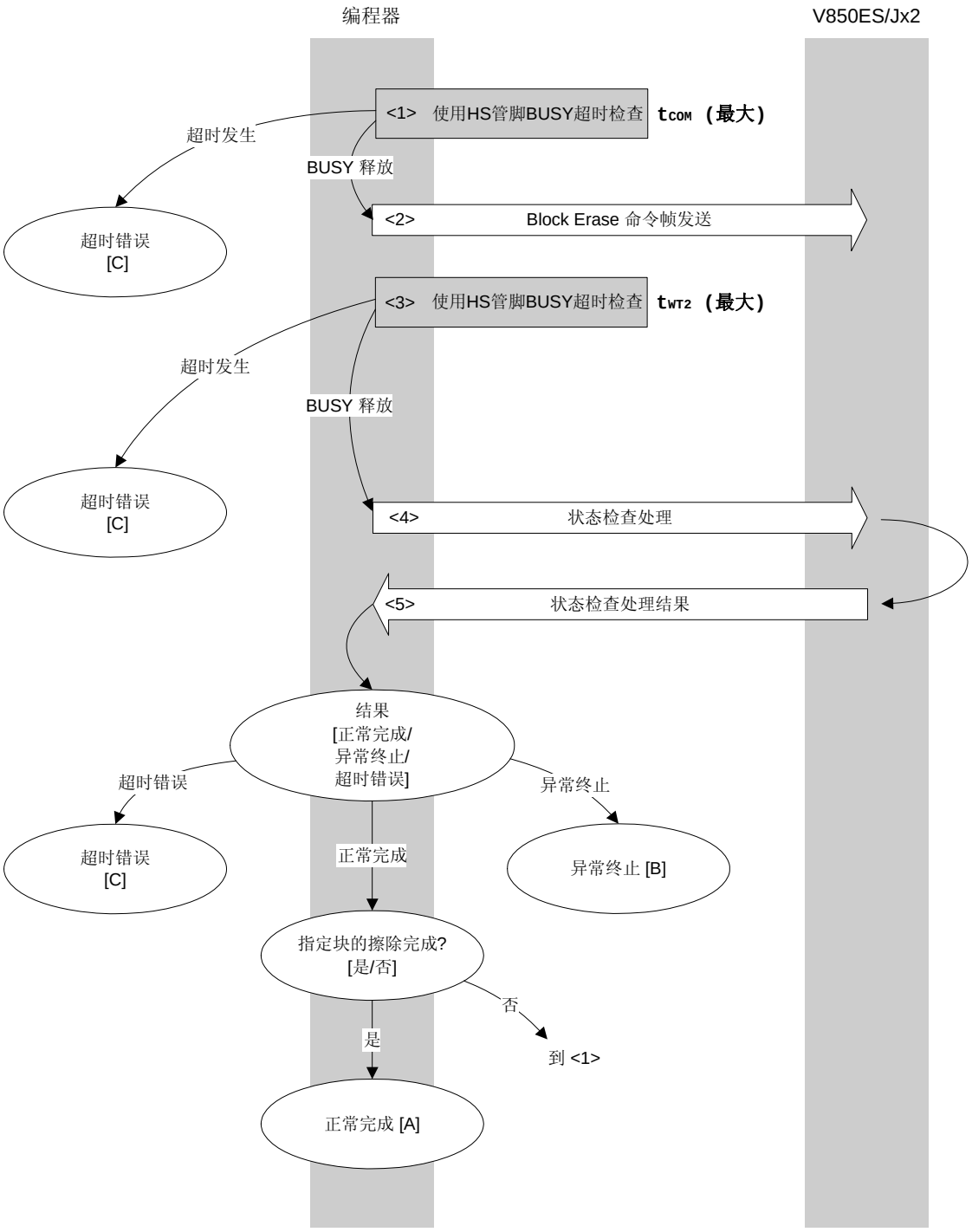
以下表示 Chip Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 擦除所有（整个芯片）命令（CSI-HS）                            */
/*                                                               */
/*****************************************************************/
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_hs_erase_all(void)
{
        u16     rc;

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测

        if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))
                                                        // 发送 "Chip Erase" 命令
                return  rc;                             // 如果 ［C］

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

        rc = fl_hs_getstatus();                 // 获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:     return  rc;     break;  // 如果 ［A］
//              case    FLC_HSTO_ERR:   return  rc;     break;  // 如果 ［C］
//              default:                return  rc;     break;  // 如果 ［B］
//      }
        return rc;
}
```

7.8 Block Erase 命令

7.8.1 处理顺序图

Block Erase 命令处理顺序



7.8.2 处理顺序的说明

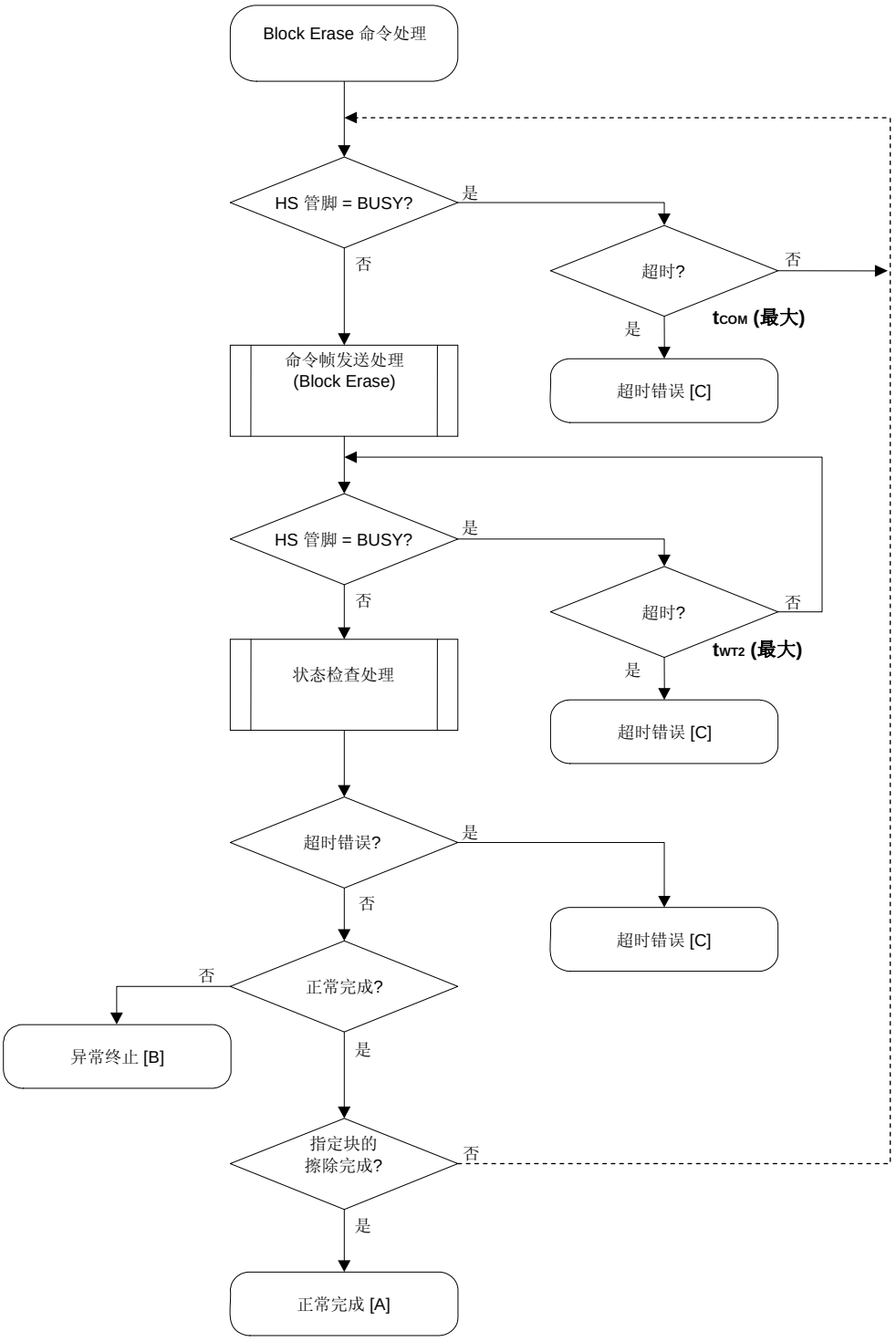
<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Block Erase 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT2}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	当指定块的块擦除没有完成时，处理更改块号并按顺序从<1>重新执行。 当所有指定块的块擦除完成时，处理正常结束［A］。
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

7.8.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且块擦除被正常完成。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。

7.8.4 流程图


Block Erase 命令处理
HS 管脚 = BUSY?
是
否
超时?
否
是
tCOM (最大)
超时错误 [C]
命令帧发送处理
(Block Erase)
HS 管脚 = BUSY?
是
否
超时?
否
是
tWT2 (最大)
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
异常终止 [B]
是
指定块的
擦除完成?
否
是
正常完成 [A]

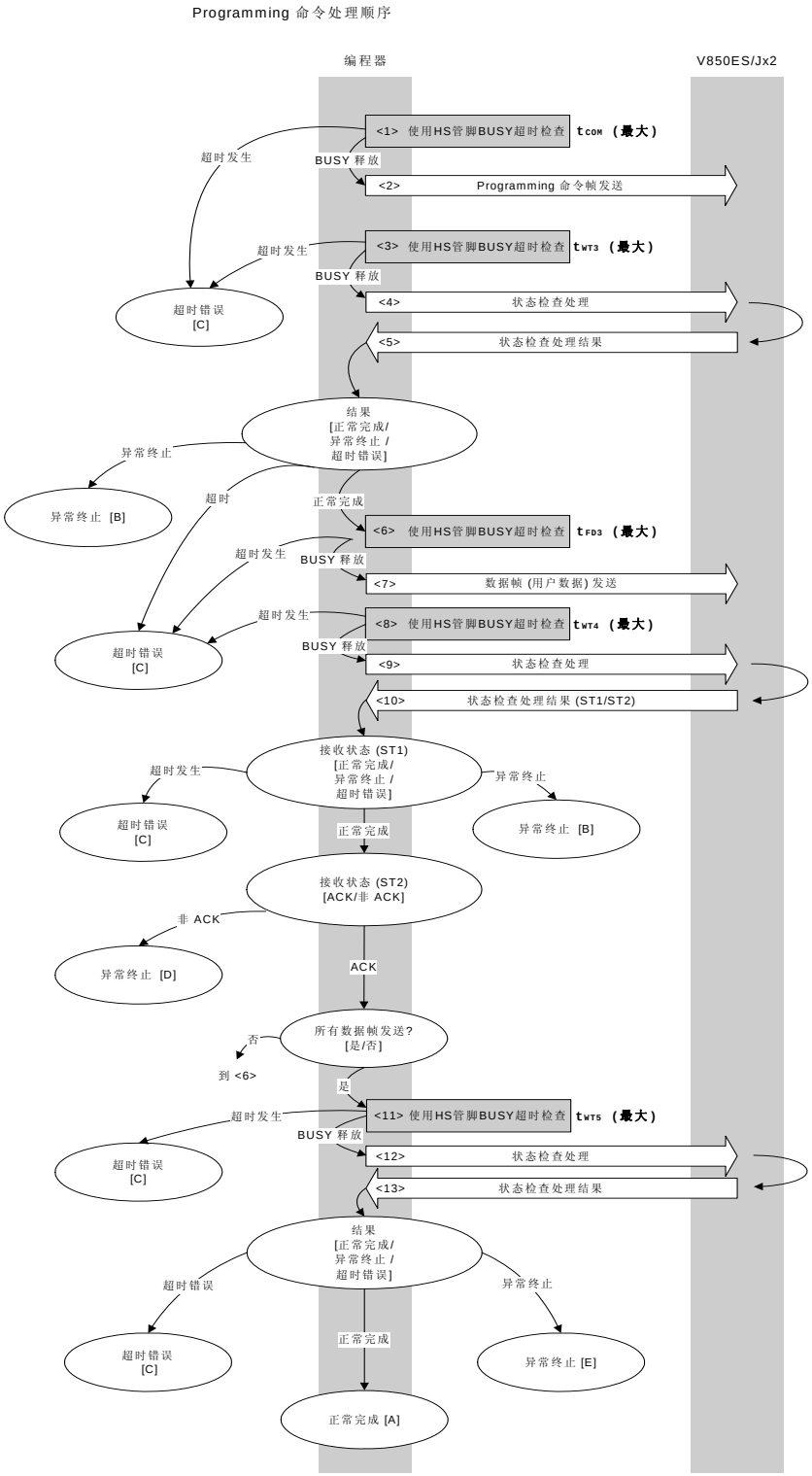
7.8.5 范例程序

以下表示对一个块的 Block Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 擦除块命令（CSI-HS）                                          */
/*                                                               */
/*****************************************************************/
/* [i] u8 block   ... 块号                                       */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_hs_erase_blk(u8 block)
{
        u16     rc;
        u32     wt2_max;

        fl_cmd_prm[0] = block;                  // 设置参数 (BLK)
        wt2_max = get_wt2_max(get_block_size(block));

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                           // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm))
                                                                // 发送 "Block Erase" 命令
                return  rc;                                     // 如果 ［C］

        if (hs_busy_to(wt2_max))
                return  FLC_HSTO_ERR;                           // 超时检测：如果 ［C］

        rc = fl_hs_getstatus();                 // 获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果 ［A］
//              case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
//              default:                        return  rc;     break;  // 如果 ［B］
//      }
        return rc;
}
```

7.9 Programming 命令

7.9.1 处理顺序图



7.9.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Programming 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT3}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	到<6>。
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD3}（最大））。
<7> 通过数据帧发送处理发送用户数据。
<8> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT4}（最大））。
<9> 通过状态检查处理，获取状态帧。
<10> 根据状态检查处理的结果（状态码（ST1/ST2））（同样参考处理顺序图和流程图），执行以下处理。

当 ST1 = 异常终止时：	异常终止［B］
当 ST1 = 超时错误时：	一个超时错误［C］被返回。
当 ST1 = 正常完成时：	根据 ST2 的值，以下处理被执行。

- 当 ST2 ≠ ACK 时： 异常终止[D]
- 当 ST2 = ACK 时： 当所有用户数据的发送完成后，到<11>。
  如果仍然存在用户数据要被发送，处理从<6>重新执行。

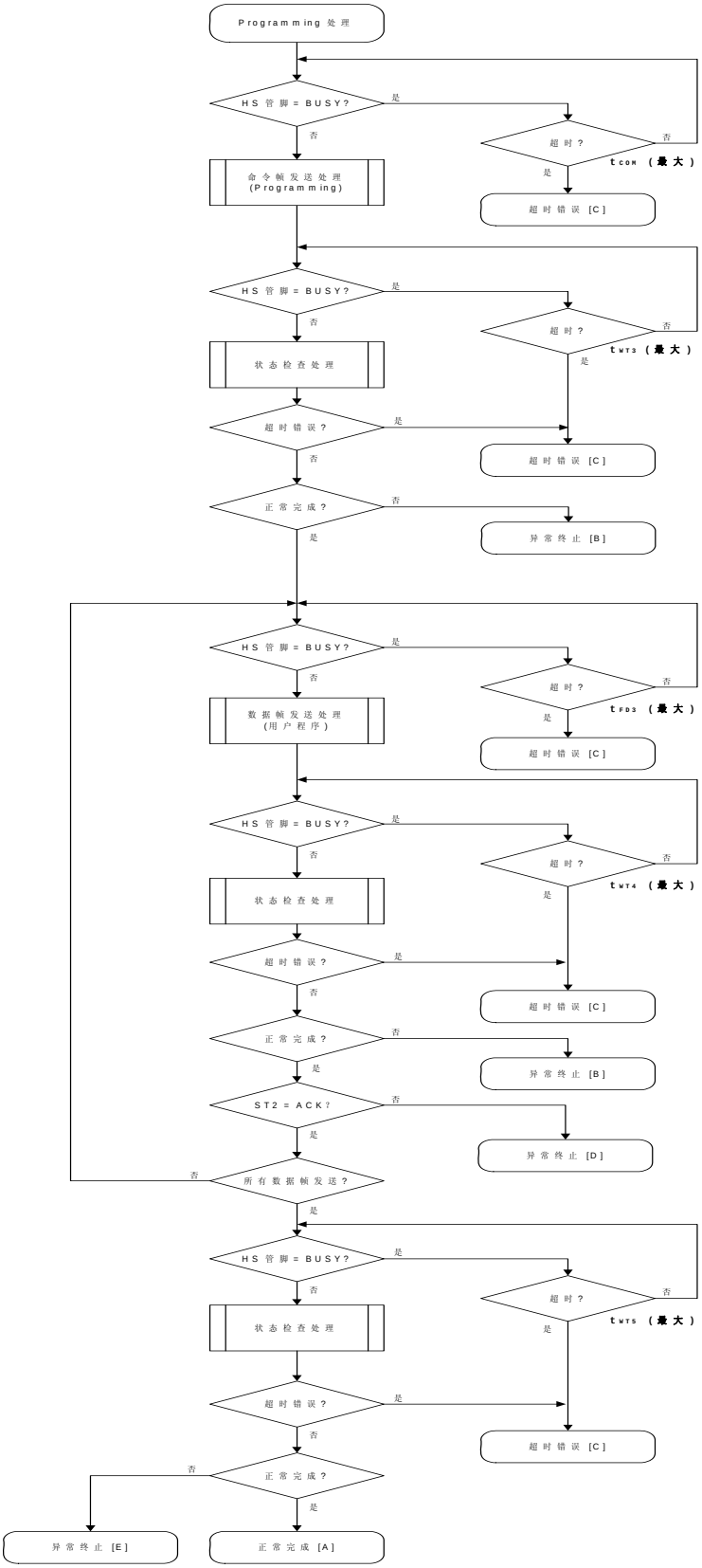
<11> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT5}（最大））。
<12> 通过状态检查处理，获取状态帧。
<13> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］ （表明内部校验检查在写入完成后正常执行）
当处理异常结束时：	异常终止[E] （表明内部校验检查在写入完成后没有正常执行）
当超时错误发生时：	一个超时错误［C］被返回。

7.9.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且用户数据被正常写入。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	写入在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
异常终止[D]	WWV1 错误	08H (ST2)	一个写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止[E]	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

7.9.4 流程图


Programming 处理
HS 管脚 = BUSY?
是
否
超时？
否
是
tCOM（最大）
命令帧发送处理
(Programming)
超时错误 [C]
HS 管脚 = BUSY?
是
否
超时？
否
是
tWT3（最大）
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [B]
HS 管脚 = BUSY?
是
否
超时？
否
是
tFD3（最大）
数据帧发送处理
(用户程序)
超时错误 [C]
HS 管脚 = BUSY?
是
否
超时？
否
是
tWT4（最大）
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [B]
ST2 = ACK？
否
是
异常终止 [D]
所有数据帧发送？
否
是
HS 管脚 = BUSY?
是
否
超时？
否
是
tWT5（最大）
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [E]
正常完成 [A]

7.9.5 范例程序

以下表示 Programming 命令处理的一个范例程序。

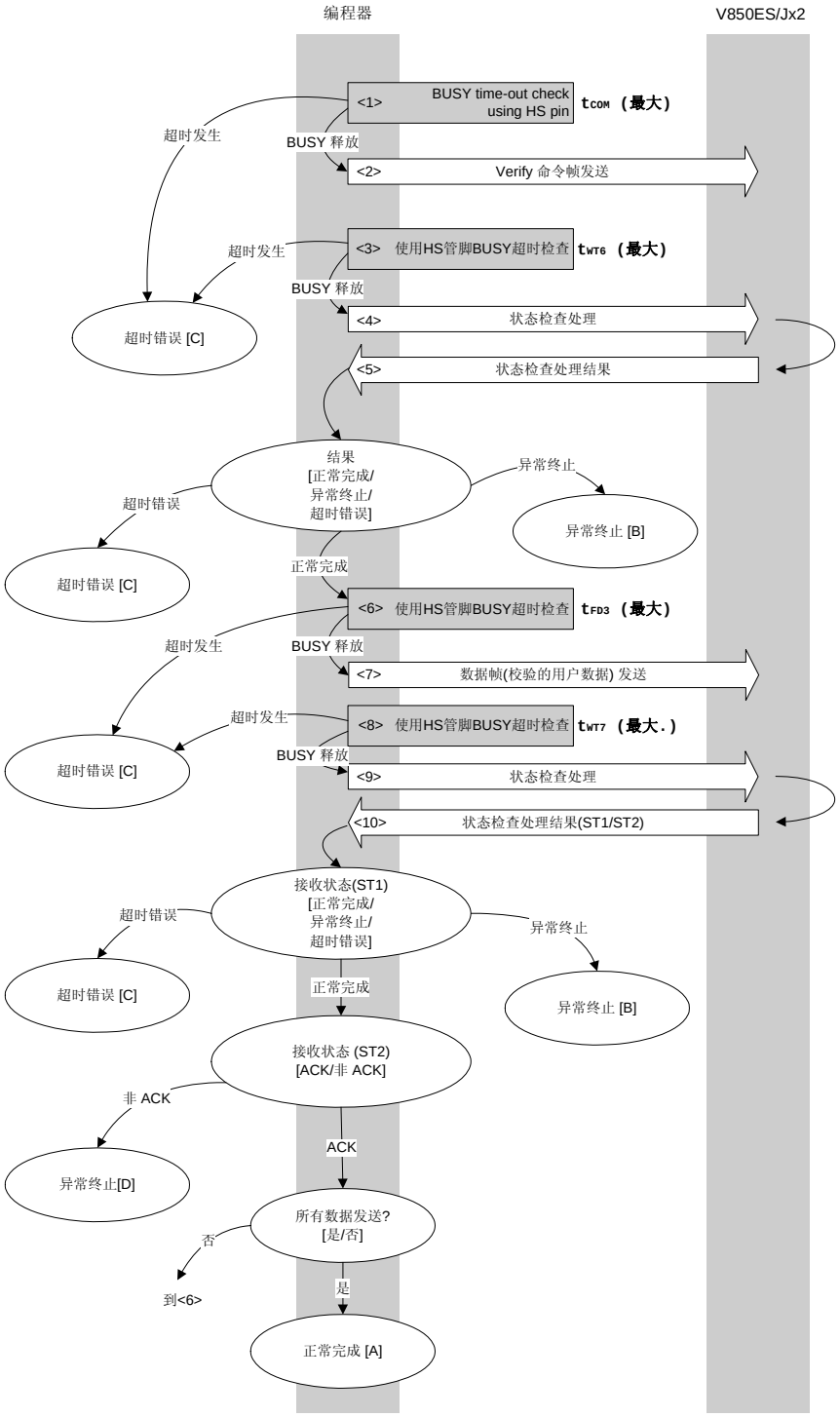
```
/*****************************************************************/
/*                                                               */
/* 写入命令 (CSI-HS)                                             */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... 起始地址                                  */
/* [i] u32 bottom  ... 结束地址                                  */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_hs_write(u32 top, u32 bottom)
{
        u16     rc;
        u32     send_head, send_size;
        bool    is_end;
        u32     wt5_max;


        /************************************************/
        /*      set params                              */
        /************************************************/
        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL
        wt5_max = get_wt5_max(bottom - top + 1);


        /************************************************/
        /*      发送命令 & 检查状态                     */
        /************************************************/
        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测

        if (rc = put_cmd_hs(FL_COM_WRITE, 7, fl_cmd_prm))  // 发送 "Programming" 命令
                return  rc;                                     // 超时检测

        if (hs_busy_to(tWT3_MAX))
                return  FLC_HSTO_ERR;                           // 超时检测

        rc = fl_hs_getstatus();                         // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_HSTO_ERR:       return  rc;     break;  // 如果［C］
                default:                    return  rc;     break;  // 如果［B］
        }

        /************************************************/
        /*      发送用户数据                            */
        /************************************************/
        send_head = top;

        while(1){
                // make send data frame
                if ((bottom - send_head) > 256){        // 剩余大小 > 256 ?
                        is_end = false;                 // 是，不是最后一帧
                        send_size  = 256;               // 发送大小 = 256 字节
```

```
                }
                else{
                        is_end = true;
                        send_size = bottom - send_head + 1;
                                                // 发送大小 = (底部 – 发送头)+1 字节

                }
                memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                                                // 设置数据帧有效载荷
                send_head += send_size;

                if (hs_busy_to(tFD3_MAX))       // 发送数据帧前超时检查
                        return  FLC_HSTO_ERR;   // 超时检测

                if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                                                // 发送用户数据
                        return  rc;             // 错误检测

                if (hs_busy_to(tWT4_MAX))
                        return  FLC_HSTO_ERR;           // 超时检测

                rc = fl_hs_getstatus();         // 获取状态帧
                switch(rc) {
                        case    FLC_NO_ERR:             break;  // 继续
                //      case    FLC_HSTO_ERR:   return  rc;     break;  // 如果 ［C］
                        default:        return  rc;     break;  // 如果 ［B］
                }
                if (fl_st2 != FLST_ACK){        // ST2 = ACK ?
                        rc = decode_status(fl_st2); // 否
                        return  rc;                     // 如果 [D]
                }
                if (is_end)                     // 发送所有用户数据 ?
                        break;                  // 是



        }
        /************************************************/
        /*      检查内部校验                            */
        /************************************************/
        if (hs_busy_to(wt5_max))
                return  FLC_HSTO_ERR;                   // 超时检测

        rc = fl_hs_getstatus();                 // 获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果 ［A］
//              case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
//              default:                        return  rc;     break;  // 如果 ［B］
//      }
        return rc;

}
```

7.10 Verify 命令

7.10.1 处理顺序图

Verify 命令处理顺序



7.10.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Verify 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT6}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时： 到<6>。
当处理异常结束时： 异常终止［B］
当超时错误发生时： 一个超时错误［C］被返回。

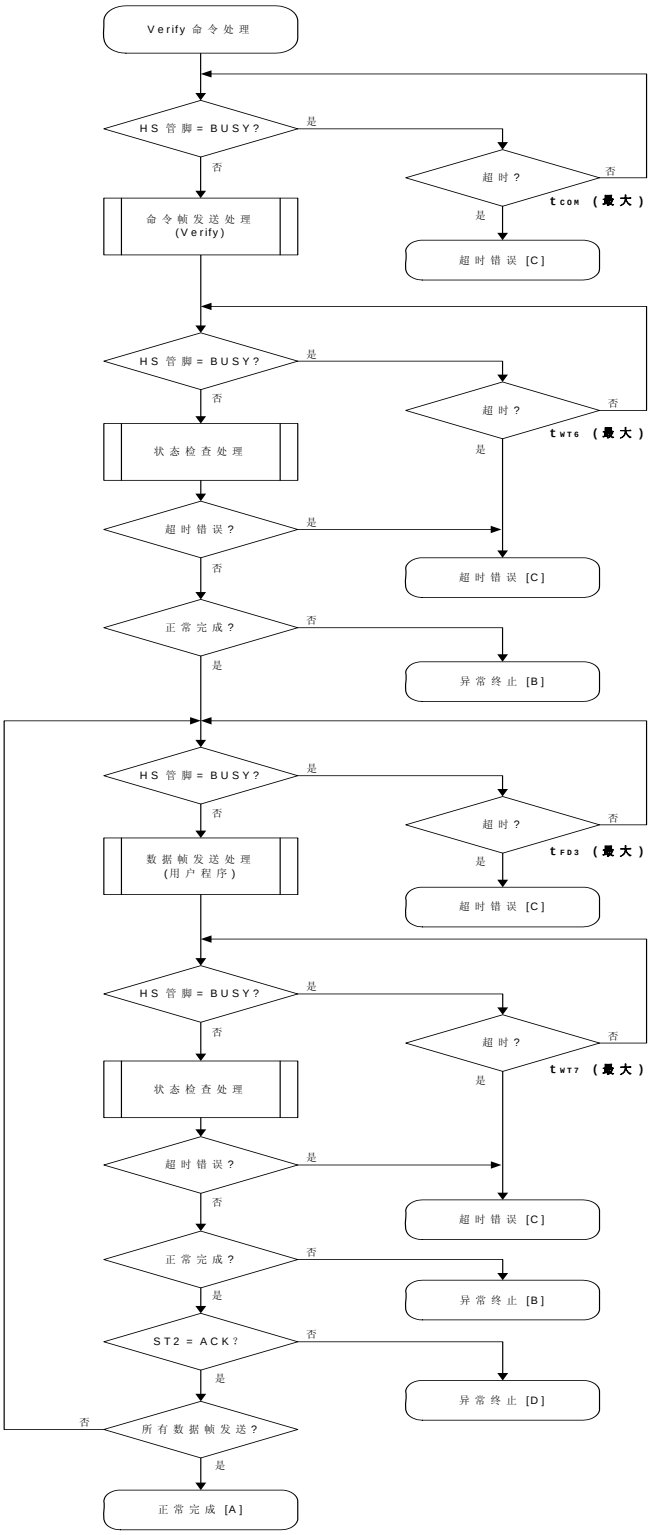
<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD3}（最大））。
<7> 通过数据帧发送处理发送校验用的用户数据。
<8> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT7}（最大））。
<9> 通过状态检查处理，获取状态帧。
根据状态检查处理的结果（状态码（ST1/ST2））（同样参考处理顺序图和流程图），以下处理被执行。

当 ST1 = 异常终止时： 异常终止［B］
当 ST1 = 超时错误时： 一个超时错误［C］被返回。
当 ST1 = 正常完成时： 根据 ST2 的值，以下处理被执行。
- 当 ST2 = ACK 时： 如果所有数据帧的发送完成，处理正常结束［A］。
  如果仍然存在数据帧要被发送，处理从<6>重新执行。
- 当 ST2 ≠ ACK 时： 异常终止[D]

7.10.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验被正常完成。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
异常终止[D]	校验错误	0FH (ST2)	校验失败或者另一个错误发生。
		0EH	
	程序错误	16H	一个程序错误发生。

7.10.4 流程图


Verify 命令处理
HS 管脚 = BUSY?
是
否
超时？
否
tCOM（最大）
是
超时错误 [C]
命令帧发送处理 (Verify)
HS 管脚 = BUSY?
是
否
超时？
否
tWT6（最大）
是
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [B]
HS 管脚 = BUSY?
是
否
超时？
否
tFD3（最大）
是
数据帧发送处理 (用户程序)
超时错误 [C]
HS 管脚 = BUSY?
是
否
超时？
否
tWT7（最大）
是
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [B]
ST2 = ACK？
否
是
异常终止 [D]
所有数据帧发送？
否
是
正常完成 [A]

7.10.5 范例程序

以下表示 Verify 命令处理的一个范例程序。

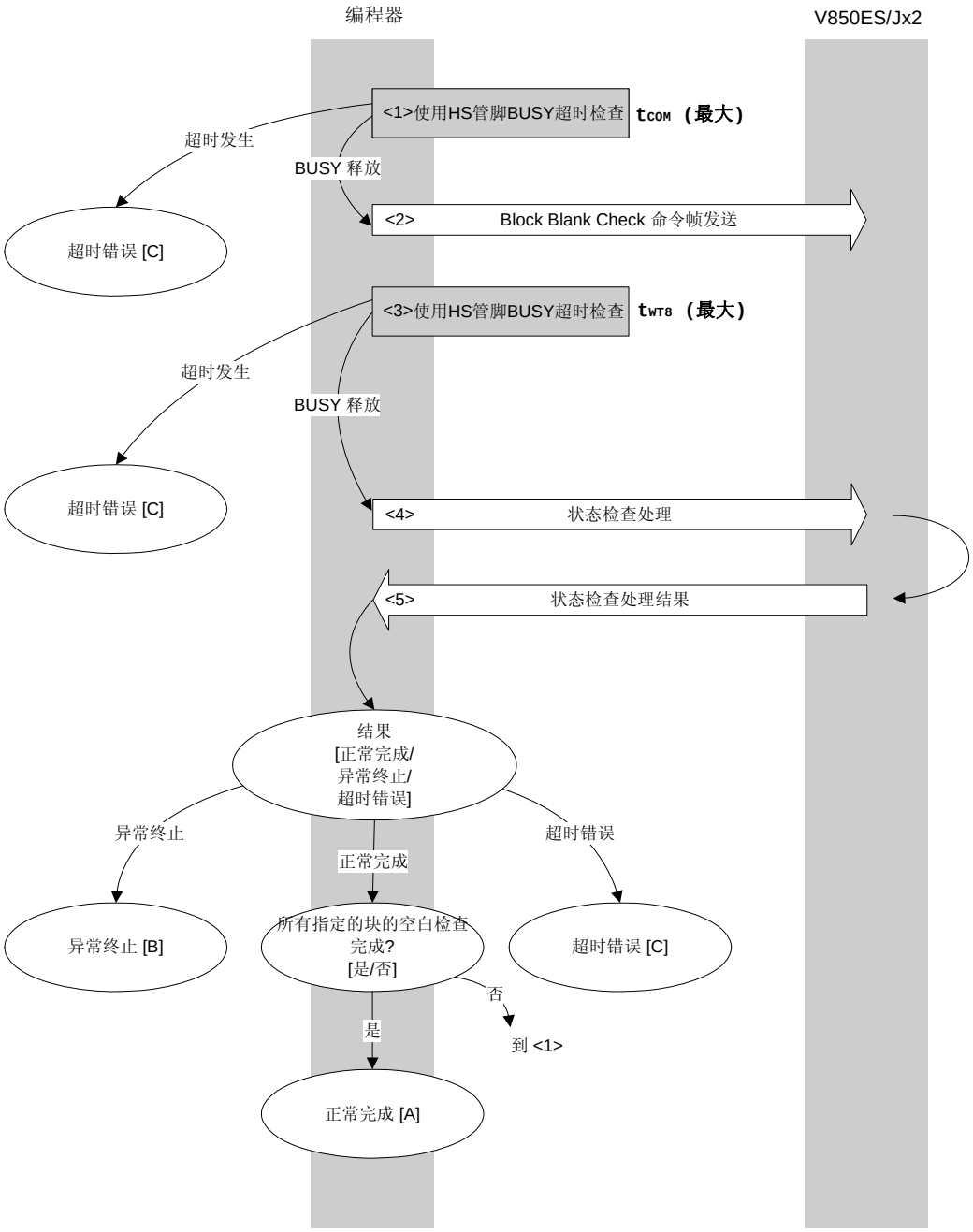
```
/*****************************************************************/
/*                                                               */
/* 校验命令 （CSI-HS）                                           */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... 起始地址                                  */
/* [i] u32 bottom  ... 结束地址                                  */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_hs_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);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL


        /*************************************************/
        /*      发送命令 & 检查状态                      */
        /*************************************************/

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测

        if (rc = put_cmd_hs(FL_COM_VERIFY, 7, fl_cmd_prm))          // 发送 "Verify" 命令
                return  rc;                             // 错误检测

        if (hs_busy_to(tWT6_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测

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


        /*************************************************/
        /*      发送用户数据                             */
        /*************************************************/
```

```
    send_head = top;

    while(1){

        // make send data frame
        if ((bottom - send_head) > 256){     // 剩余大小 > 256 ?
            is_end = false;                  // 是，不是最后一帧
            send_size  = 256;                // 发送大小 = 256 字节
        }
        else{
            is_end = true;
            send_size = bottom - send_head + 1;
                                             // 发送大小 = (底部 – 发送头)+1 字节

        }
        memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                                             // 设置数据帧有效载荷
        send_head += send_size;

        if (hs_busy_to(tFD3_MAX))
            return    FLC_HSTO_ERR;                  // 超时检测

        if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                                                     // 发送用户数据
            return    rc;                            // 错误检测

        if (hs_busy_to(tWT7_MAX))
            return    FLC_HSTO_ERR;                  // 超时检测

        rc = fl_hs_getstatus();              // 获取状态帧
        switch(rc) {
            case      FLC_NO_ERR:                    break;   // 继续
        //  case      FLC_HSTO_ERR:        return    rc;      break;   // 如果［C］
            default:            return    rc;      break;   // 如果［B］
        }
        if (fl_st2 != FLST_ACK){                     // ST2 = ACK ?
            rc = decode_status(fl_st2); // 否
            return    rc;                            // 如果 [D]
        }
        if (is_end)                          // 发送所有用户数据?
            break;                           // 是
    }
    return    FLC_NO_ERR;    // 如果［A］
}
```

7.11 Block Blank Check 命令

7.11.1 处理顺序图

Block Blank Check 命令处理顺序



7.11.2 处理顺序的说明

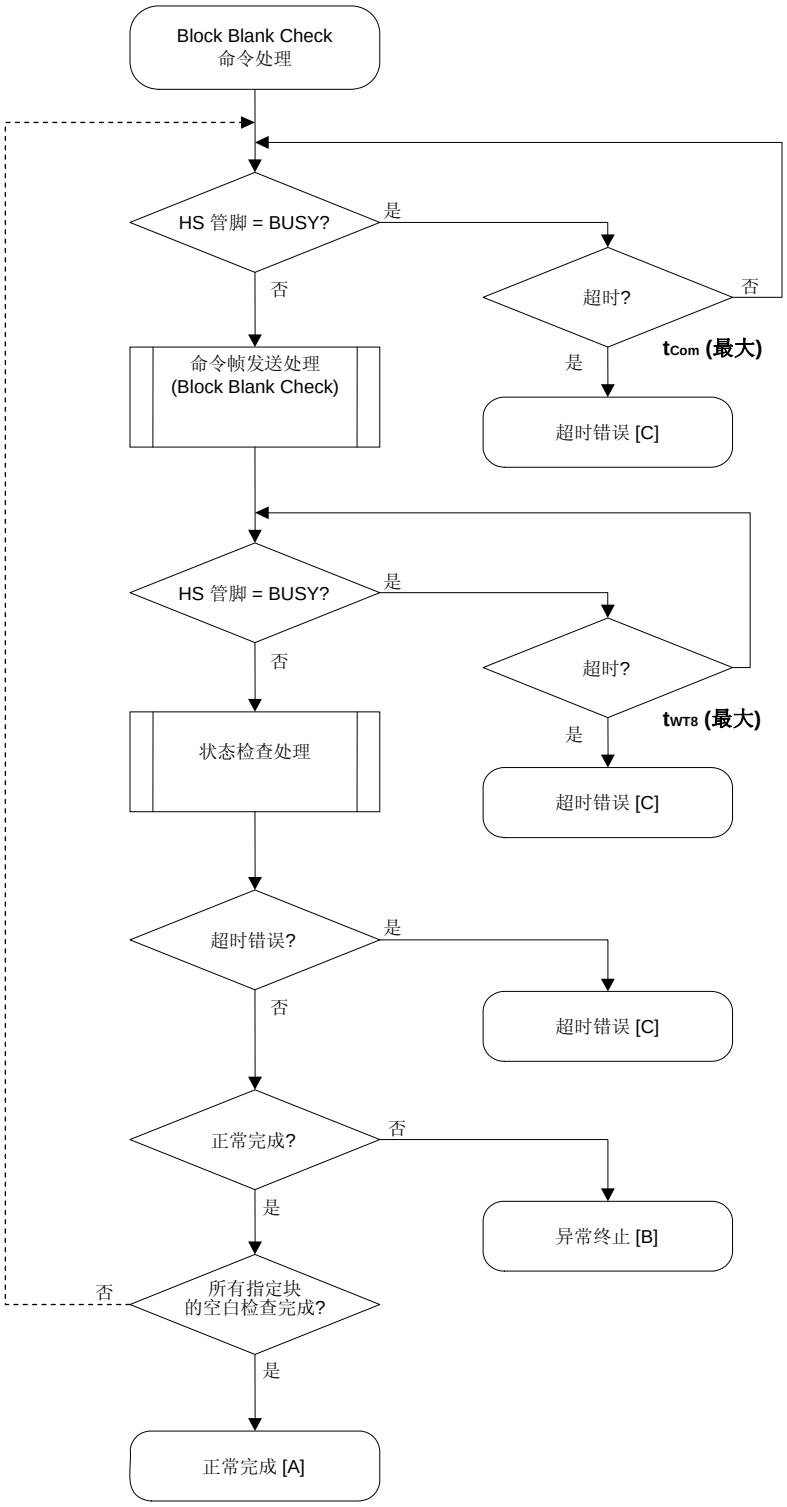
<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Block Blank Check 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT8}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理异常结束时：	异常终止［B］
当处理正常结束时：	当所有指定块的空白检查完成时，处理正常结束［A］。 当指定块的空白检查没有完成时，处理更改块号并按顺序从<1>重新执行。
当超时错误发生时：	一个超时错误［C］被返回。

7.11.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	这个命令被正常执行，并且所有指定的块空白。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	EWV4 错误	11H	指定块的 flash 存储器不是空白。
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。

7.11.4 流程图


Block Blank Check
命令处理
HS 管脚 = BUSY?
是
否
超时?
否
tCOM (最大)
是
超时错误 [C]
命令帧发送处理
(Block Blank Check)
HS 管脚 = BUSY?
是
否
超时?
tWT8 (最大)
是
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
异常终止 [B]
所有指定块
的空白检查完成?
否
是
正常完成 [A]

7.11.5 范例程序

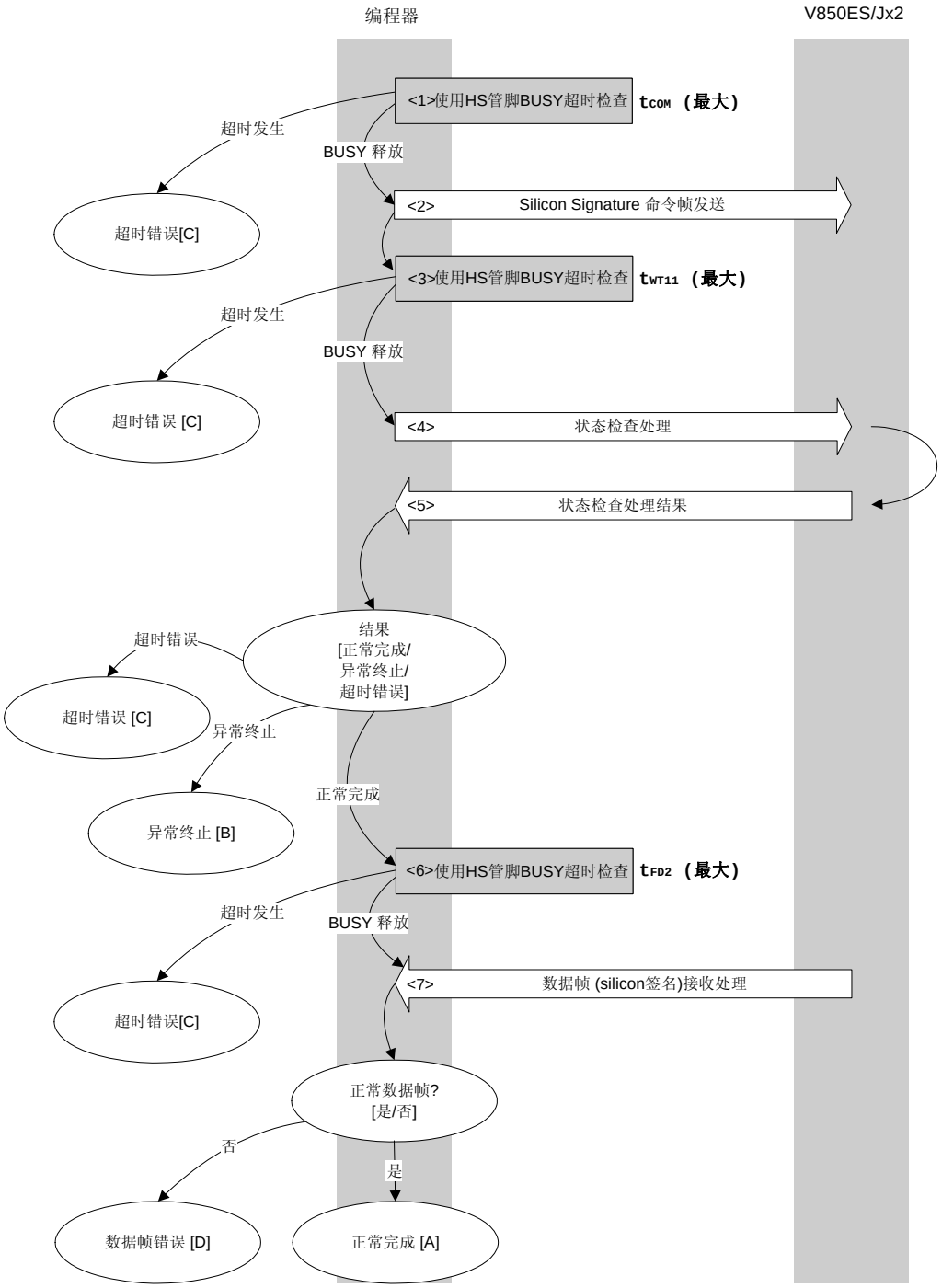
以下表示对一个块的 Block Blank Check 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 块空白检查命令（CSI-HS）                                      */
/*                                                               */
/*****************************************************************/
/* [i] u16 block  ... 块号                                       */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_hs_blk_blank_chk(u8 block)
{
        u16     rc;
        u32     wt8_max;

        fl_cmd_prm[0] = block;      // "BLK"
        wt8_max = get_wt8_max(get_block_size(block));


        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm))
                                                        // 发送 "Block Blank Check" 命令
                return  rc;                             // 如果 ［C］

        if (hs_busy_to(wt8_max))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        rc = fl_hs_getstatus();                 // 获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果 ［A］
//              case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
//              default:                        return  rc;     break;  // 如果 ［B］
//      }
        return rc;
}
```

7.12 Silicon Signature 命令

7.12.1 处理顺序图

Silicon Signature 命令处理顺序



7.12.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Silicon Signature 命令。
<3> 使用 HS 管脚，一个 V850ES/Jx2 BUSY 状态被检查。
如果一个 BUSY 超时发生，超时错误［C］被返回（超时时间 t_{WT11}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时： 到<6>。
当处理异常结束时： 异常终止［B］
当超时错误发生时： 一个超时错误［C］被返回。

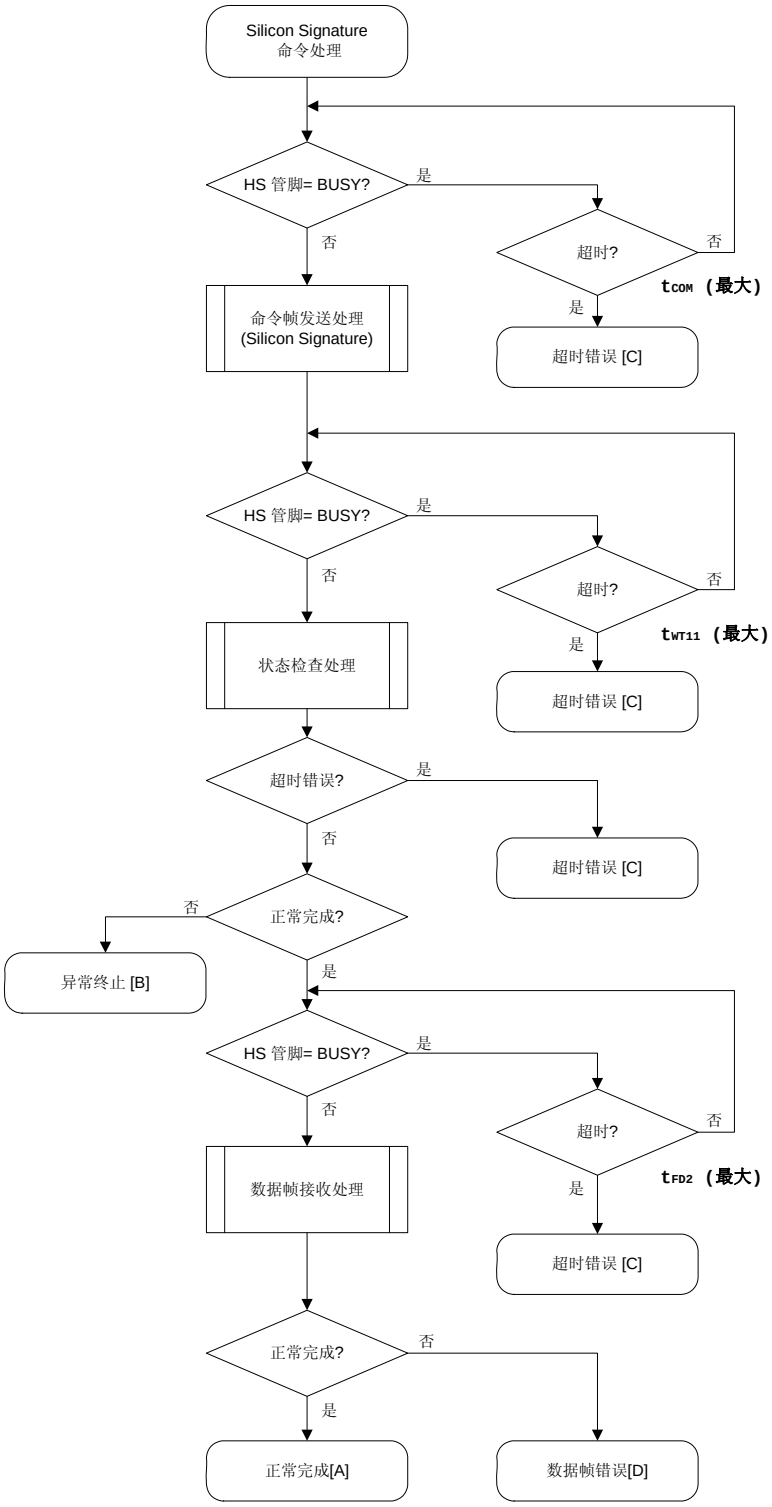
<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD2}（最大））。
<7> 检查接收的数据帧（silicon 签名数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 [D]

7.12.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且 silicon 签名被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
数据帧错误 [D]		–	作为 silicon 签名数据接收到的数据帧的校验和不匹配。

7.12.4 流程图


Silicon Signature
命令处理
HS 管脚= BUSY?
是
否
超时?
否
是
tCOM （最大）
命令帧发送处理
(Silicon Signature)
超时错误 [C]
HS 管脚= BUSY?
是
否
超时?
否
是
tWT11 （最大）
状态检查处理
超时错误 [C]
超时错误?
是
否
超时错误 [C]
否
正常完成?
异常终止 [B]
是
HS 管脚= BUSY?
是
否
超时?
否
是
tFD2 （最大）
数据帧接收处理
超时错误 [C]
正常完成?
否
是
正常完成[A]
数据帧错误[D]

7.12.5 范例程序

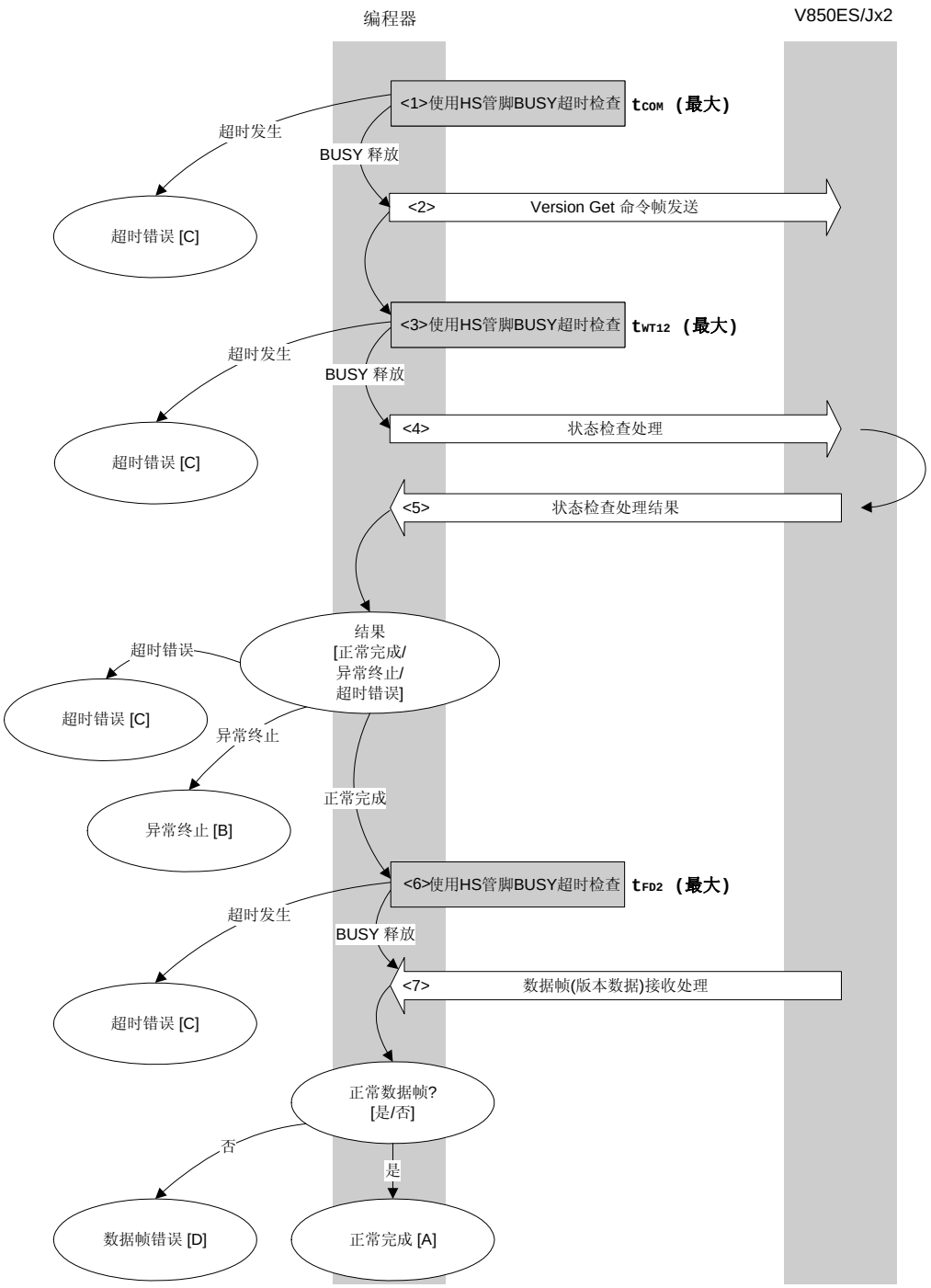
以下表示 Silicon Signature 命令处理的一个范例程序。

```
/*********************************************************************/
/*                                                                   */
/* 获取 silicon 签名命令（CSI-HS）                                    */
/*                                                                   */
/*********************************************************************/
/* [i] u8 *sig... 签名保存区域的指针                                  */
/* [r] u16    ... 错误码                                             */
/*********************************************************************/
u16             fl_hs_getsig(u8 *sig)
{
        u16     rc;

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm))
                                                        // 发送 "Silicon Signature" 命令
                return  rc;                             // 错误检测：如果 ［C］

        if (hs_busy_to(tWT11_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］


        rc = fl_hs_getstatus();                 // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_HSTO_ERR:   return  rc;     break;  // 如果 ［C］
                default:                return  rc;     break;  // 如果 ［B］
        }


        if (hs_busy_to(tFD2_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        rc = get_dfrm_hs(fl_rxdata_frm);    // 获取签名数据

        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_HSTO_ERR:   return  rc;     break;  // 如果 ［C］
                default:                return  rc;     break;  // 如果 [D]
        }

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

        return  rc;                                     // 如果 ［A］

}
```

7.13 Version Get 命令

7.13.1 处理顺序图

Version Get 命令处理顺序



7.13.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Version Get 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT12}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时： 到<6>。
当处理异常结束时： 异常终止［B］
当超时错误发生时： 一个超时错误［C］被返回。

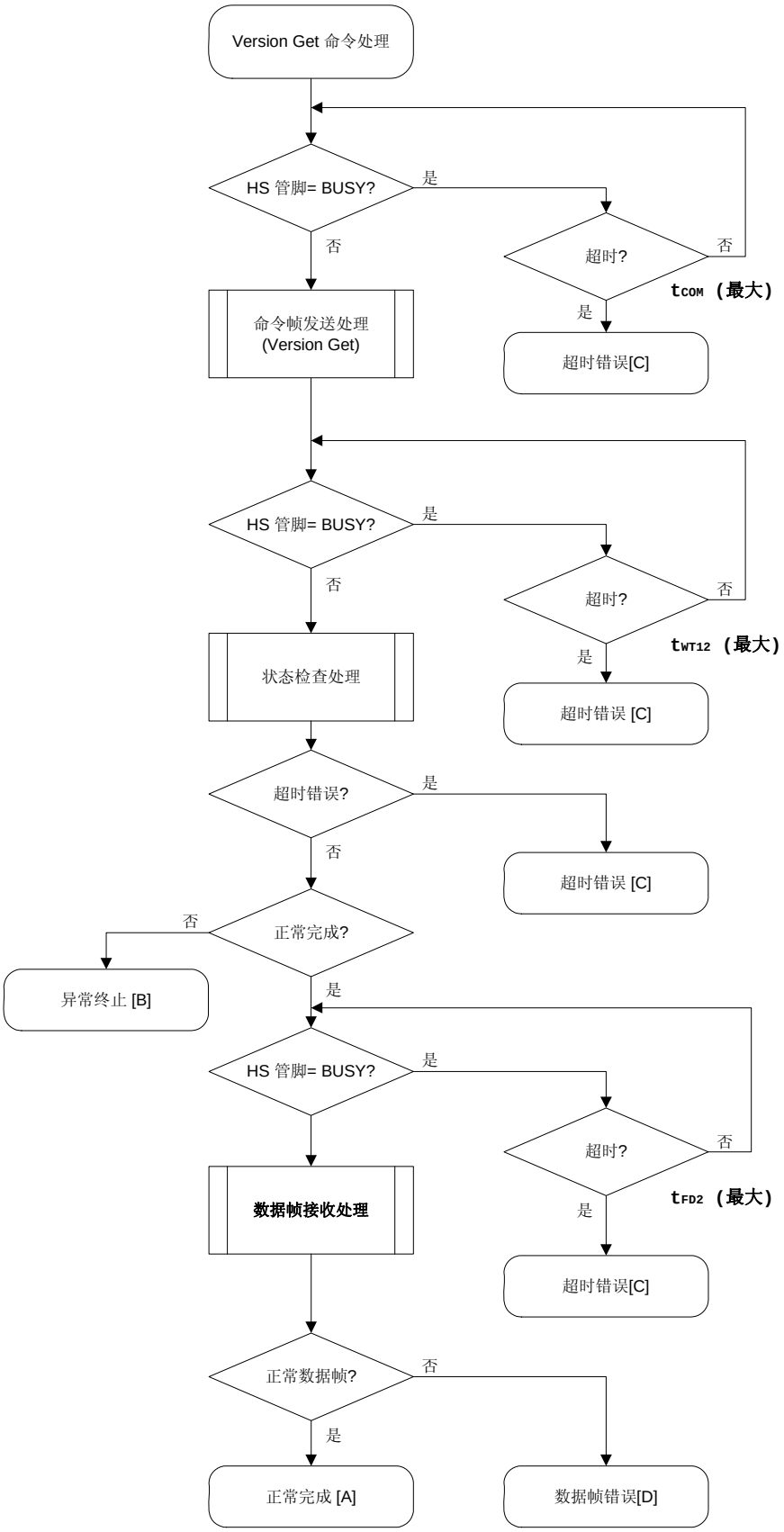
<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD2}（最大））。
<7> 检查接收到的数据帧（版本数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 [D]

7.13.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且版本数据被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
数据帧错误 [D]		–	作为版本数据接收到的数据帧的校验和不匹配。

7.13.4 流程图


Version Get 命令处理
HS 管脚= BUSY?
是
否
超时?
否
是
tCOM （最大）
命令帧发送处理
(Version Get)
超时错误[C]
HS 管脚= BUSY?
是
否
超时?
否
是
tWT12 （最大）
状态检查处理
超时错误 [C]
超时错误?
是
否
超时错误 [C]
否
正常完成?
异常终止 [B]
是
HS 管脚= BUSY?
是
否
超时?
否
是
tFD2 （最大）
数据帧接收处理
超时错误[C]
正常数据帧?
否
是
正常完成 [A]
数据帧错误[D]

7.13.5 范例程序

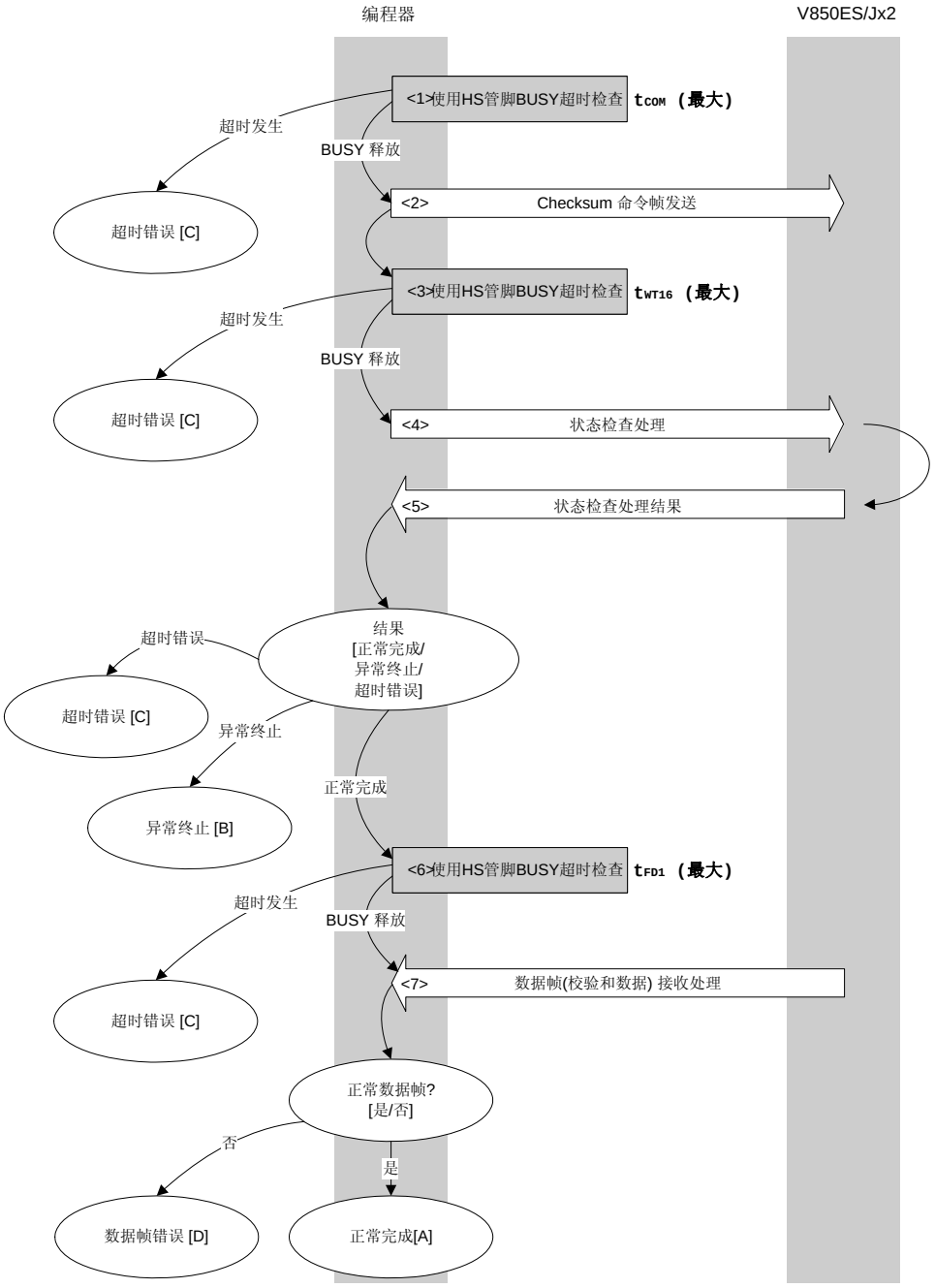
以下表示 Version Get 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获得设备/固件版本命令（CSI-HS）                               */
/*                                                               */
/*****************************************************************/
/* [i] u8 *buf     ... 版本数据保存区域的指针                    */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_hs_getver(u8 *buf)
{
        u16     rc;

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_GET_VERSION, 1, fl_cmd_prm))
                                                        // 发送 "Version Get" 命令
                return  rc;                             // 错误检测：如果 ［C］

        if (hs_busy_to(tWT12_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        rc = fl_hs_getstatus();                         // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
                default:                        return  rc;     break;  // 如果 ［B］
        }

        if (hs_busy_to(tFD2_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测：如果 ［C］

        rc = get_dfrm_hs(fl_rxdata_frm);                // 获取签名数据
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
                default:                        return  rc;     break;  // 如果 [D]
        }
        memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// 复制版本数据
        return  rc;                                     // 如果 ［A］

}
```

7.14 Checksum 命令

7.14.1 处理顺序图

Checksum 命令处理顺序



7.14.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Checksum 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT16}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时： 到<6>。
当处理异常结束时： 异常终止［B］
当超时错误发生时： 一个超时错误［C］被返回。

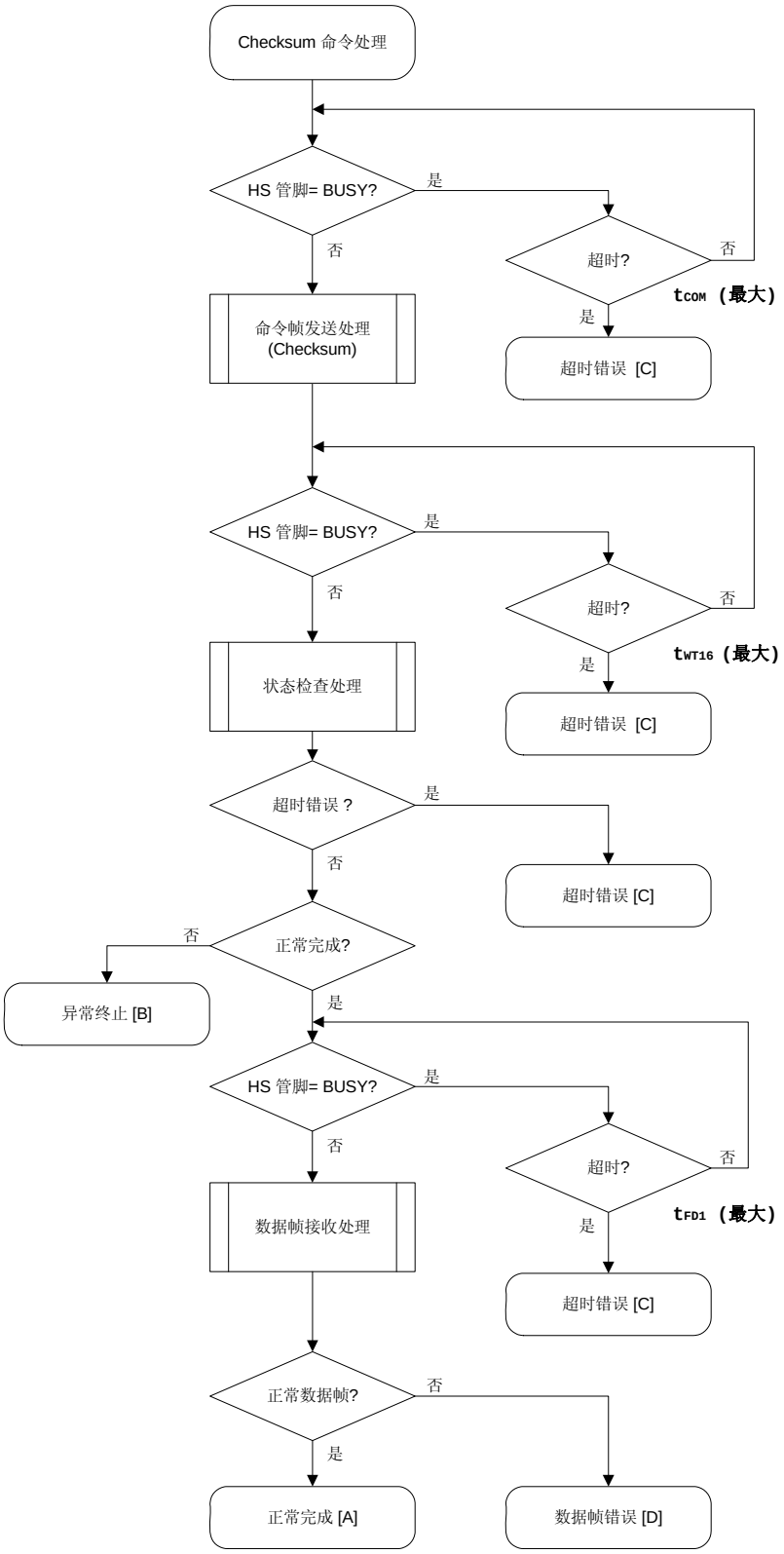
<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD1} （最大））。
<7> 检查接收的数据帧（校验和数据）。

如果数据帧正常： 正常完成［A］
如果数据帧异常： 数据帧错误 [D]

7.14.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验和数据被正常获取。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
数据帧错误 [D]		–	作为版本数据接收到的数据帧的校验和不匹配。

7.14.4 流程图


Checksum 命令处理
HS 管脚= BUSY?
是
否
超时?
否
是
tCOM（最大）
命令帧发送处理
(Checksum)
超时错误 [C]
HS 管脚= BUSY?
是
否
超时?
否
是
tWT16（最大）
状态检查处理
超时错误 [C]
超时错误 ?
是
否
超时错误 [C]
否
正常完成?
异常终止 [B]
是
HS 管脚= BUSY?
是
否
超时?
否
是
tFD1（最大）
数据帧接收处理
超时错误 [C]
正常数据帧?
否
是
正常完成 [A]
数据帧错误 [D]

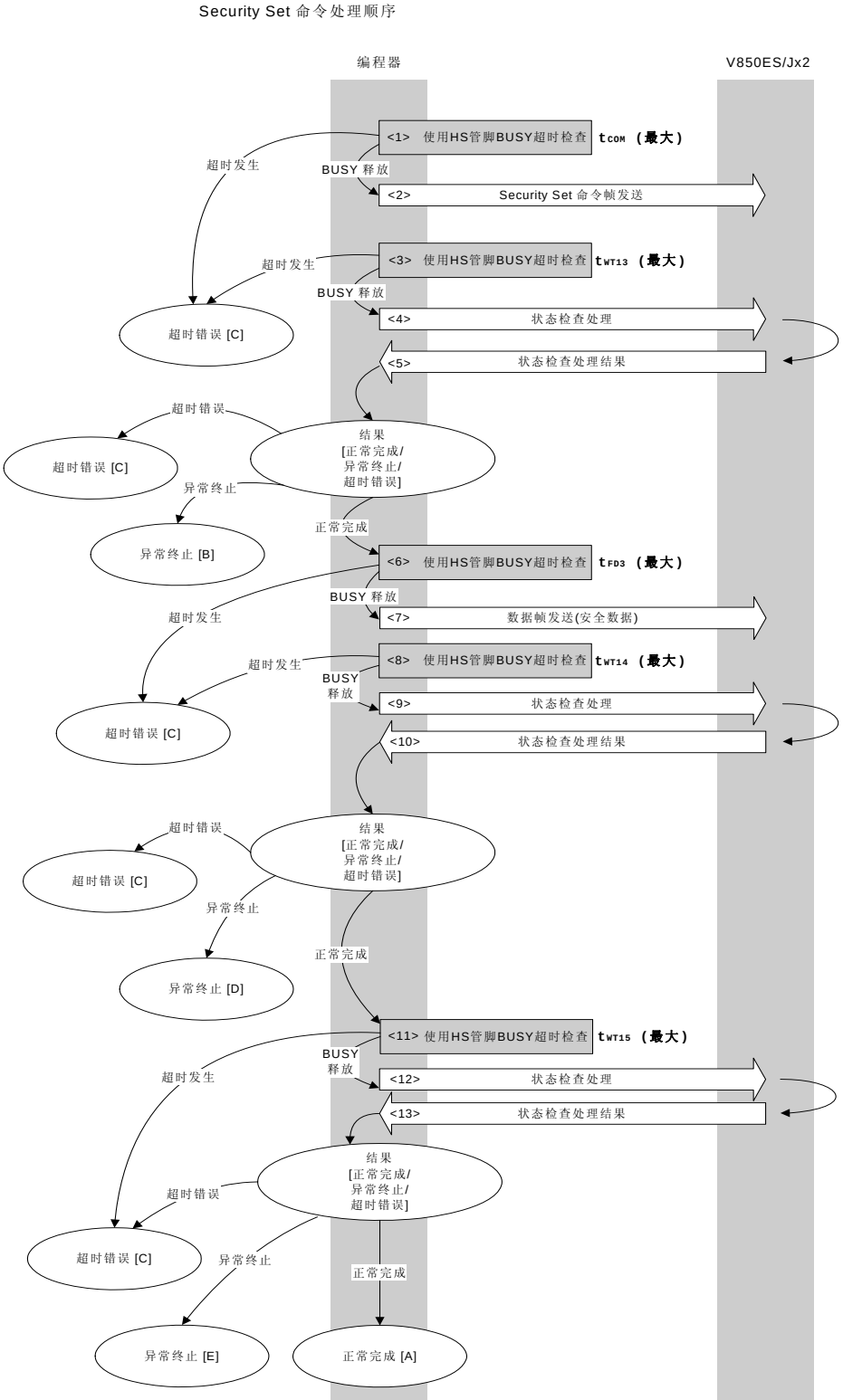
7.14.5 范例程序

以下表示 Checksum 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获取校验和命令（CSI-HS）                                      */
/*                                                               */
/*****************************************************************/
/* [i] u16 *sum    ... 校验和保存数据的指针                      */
/* [i] u32 top     ... 起始地址                                  */
/* [i] u32 bottom  ... 结束地址                                  */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_hs_getsum(u16 *sum, u32 top, u32 bottom)
{
        u16     rc;

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

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                           // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm))
                                                                // 发送 "Checksum" 命令
                return  rc;                                     // 错误检测：如果 ［C］

        if (hs_busy_to(tWT16_MAX))
                return  FLC_HSTO_ERR;                           // 超时检测：如果 ［C］

        rc = fl_hs_getstatus();                         // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
                default:                        return  rc;     break;  // 如果 ［B］
        }


        if (hs_busy_to(tFD1_MAX))
                return  FLC_HSTO_ERR;                           // 超时检测：如果 ［C］

        rc = get_dfrm_hs(fl_rxdata_frm);                // 获取签名数据

        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_HSTO_ERR:           return  rc;     break;  // 如果 ［C］
                default:                        return  rc;     break;  // 如果 [D]
        }

        *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1];
                                                        // 设置 SUM 数据
        return  rc;                                     // 如果 ［A］
}
```

7.15 Security Set 命令

7.15.1 处理顺序图



7.15.2 处理顺序的说明

<1> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个超时发生，返回超时错误［C］（超时时间 t_{COM}（最大））。
<2> 通过命令帧发送处理发送 Security Set 命令。
<3> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT13}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	到<6>。
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

<6> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{FD3}（最大））。
<7> 通过数据帧发送处理发送数据帧（安全设置数据）命令。
<8> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT1}（最大））。
<9> 通过状态检查处理，获取状态帧。
<10> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	到<11>。
当处理异常结束时：	异常终止[D]
当超时错误发生时：	一个超时错误［C］被返回。

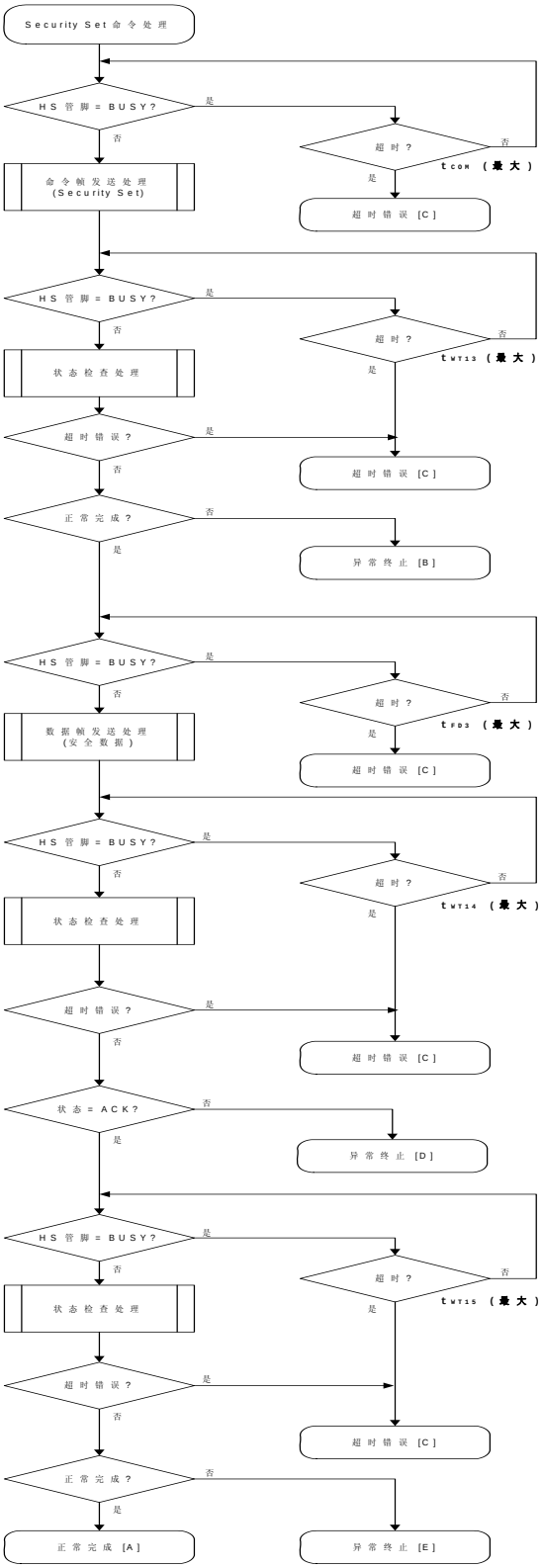
<11> 使用 HS 管脚，检查一个 V850ES/Jx2 BUSY 状态。
如果一个 BUSY 超时发生，返回超时错误［C］（超时时间 t_{WT15}（最大））。
<12> 通过状态检查处理，获取状态帧。
<13> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］
当处理异常结束时：	异常终止[E]
当超时错误发生时：	一个超时错误［C］被返回。

7.15.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且安全设置被正常完成。
异常终止［B］	参数错误	05H	命令信息（参数）不是 00H。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	ID 码不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	由于 HS 管脚的忙状态，处理超时。
异常终止[D]	WWV1 错误	08H	• 安全数据已经设置。 • 一个安全数据写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止[E]	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

7.15.4 流程图


Security Set 命令处理
HS 管脚 = BUSY?
是
否
超时?
否
tCOM（最大）
是
超时错误 [C]
命令帧发送处理
(Security Set)
HS 管脚 = BUSY?
是
否
超时?
否
tWT13（最大）
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
异常终止 [B]
HS 管脚 = BUSY?
是
否
超时?
否
tFD3（最大）
是
数据帧发送处理
(安全数据)
超时错误 [C]
HS 管脚 = BUSY?
是
否
超时?
否
tWT14（最大）
是
状态检查处理
超时错误?
是
否
超时错误 [C]
状态 = ACK?
否
是
异常终止 [D]
HS 管脚 = BUSY?
是
否
超时?
否
tWT15（最大）
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
正常完成 [A]
异常终止 [E]

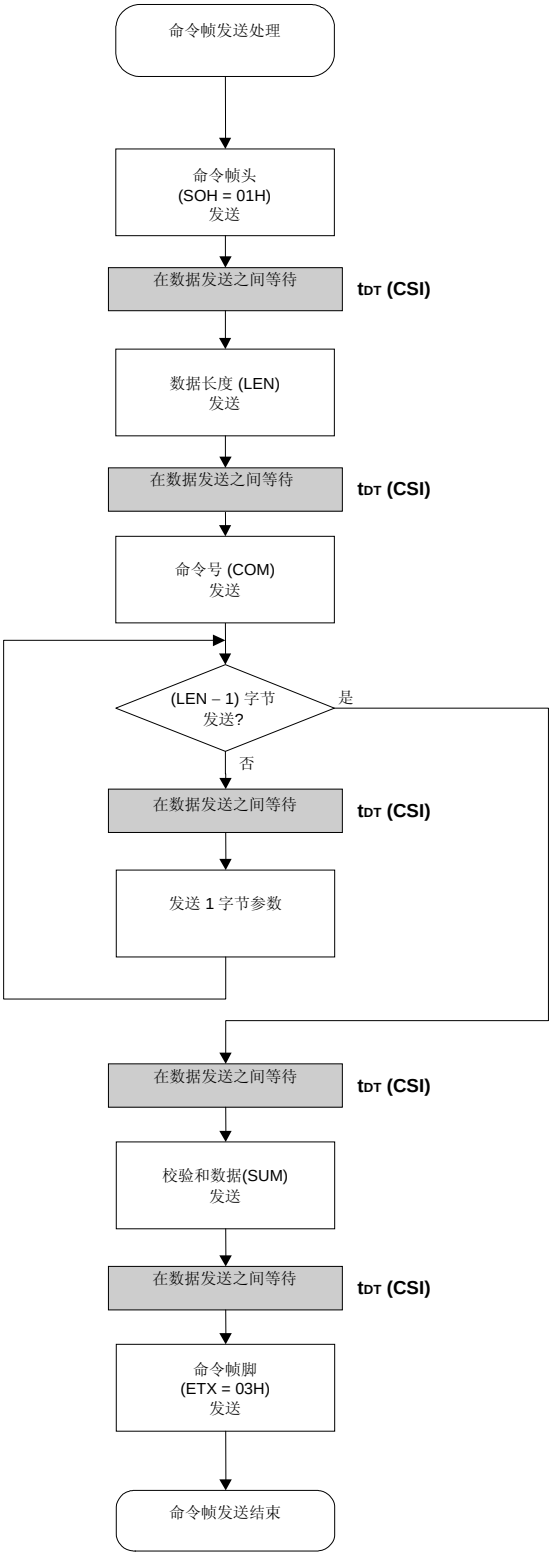
7.15.5 范例程序

以下表示 Security Set 命令处理的一个范例程序。

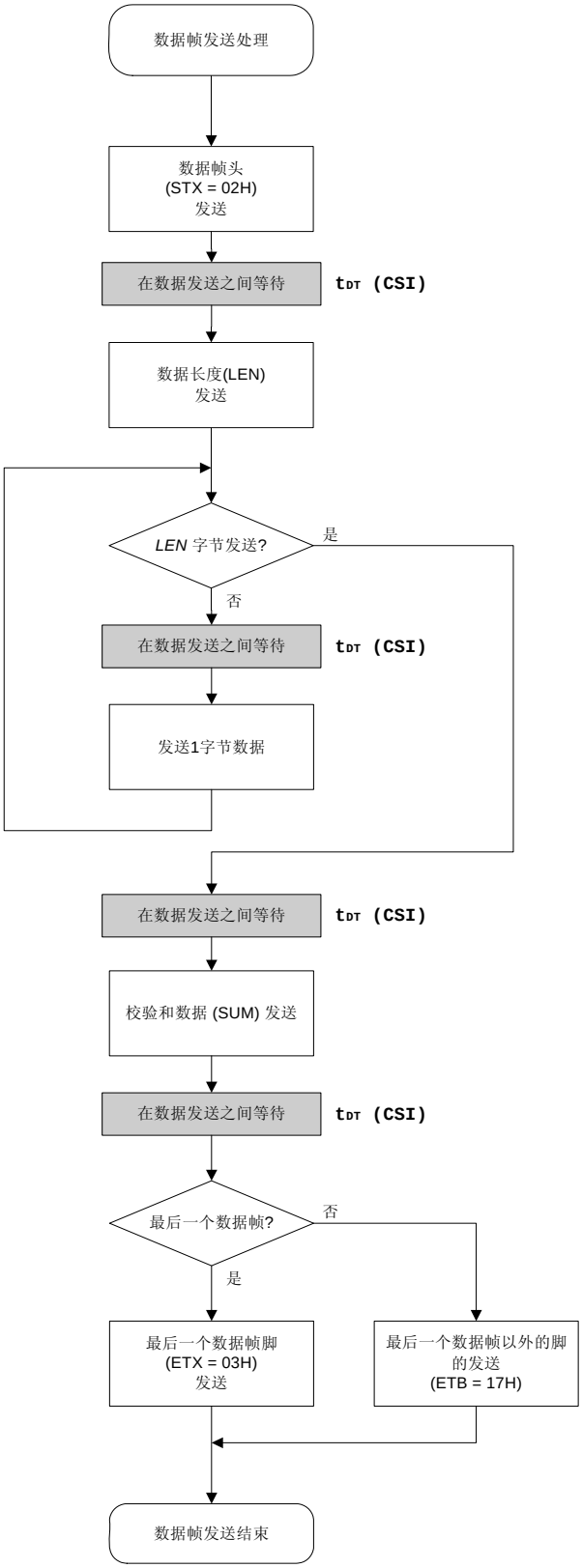
```
/*********************************************************************/
/*                                                                   */
/* 设置安全标志命令（CSI-HS）                                        */
/*                                                                   */
/*********************************************************************/
/* [i] u8 scf      ... 安全标志数据                                  */
/* [r] u16         ... 错误码                                        */
/*********************************************************************/
u16             fl_hs_setscf(u8 scf)
{
        u16     rc;

        fl_cmd_prm[0] = 0x00;               // 第一个字节必须为 0x00
        fl_cmd_prm[1] = 0x00;               // 第二个字节必须为 0x00
        fl_txdata_frm[0] = scf|= 0b11111000;
                                            // "FLG" (高 5 位必须为'1' (要确认))


        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;               // 超时检测：如果 ［C］

        if (rc = put_cmd_hs(FL_COM_SET_SECURITY, 3, fl_cmd_prm))
                                                    // 发送 "Security Set" 命令
                return  rc;                         // 错误检测：如果 ［C］

        if (hs_busy_to(tWT13_MAX))
                return  FLC_HSTO_ERR;               // 超时检测：如果 ［C］

        rc = fl_hs_getstatus();             // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                 break;  // 继续
        //      case    FLC_HSTO_ERR:       return  rc;     break;  // 如果 ［C］
                default:                    return  rc;     break;  // 如果 ［B］
        }


        if (hs_busy_to(tFD3_MAX))
                return  FLC_HSTO_ERR;               // 超时检测：如果 ［C］

        if (rc = put_dfrm_hs(1, fl_txdata_frm, true))   // 发送安全设置数据
                return  rc;                         // 错误检测：如果 ［C］

        if (hs_busy_to(tWT14_MAX))
                return  FLC_HSTO_ERR;               // 超时检测：如果 ［C］
```

```
        rc = fl_hs_getstatus();                 // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_HSTO_ERR:           return  rc;     break;  // 如果［C］
                default:        return  rc;     break;  // 如果［B］
        }


        if (hs_busy_to(tWT15_MAX))
                return  FLC_HSTO_ERR;                   // 超时检测

        rc = fl_hs_getstatus();                 // 再次获取状态帧
//      switch(rc) {
//              case    FLC_NO_ERR:     return  rc;     break;  // 如果［A］
//              case    FLC_HSTO_ERR:           return  rc;     break;  // 如果［C］
//              default:                        return  rc;     break;  // 如果［B］
//      }
        return  rc;
}
```

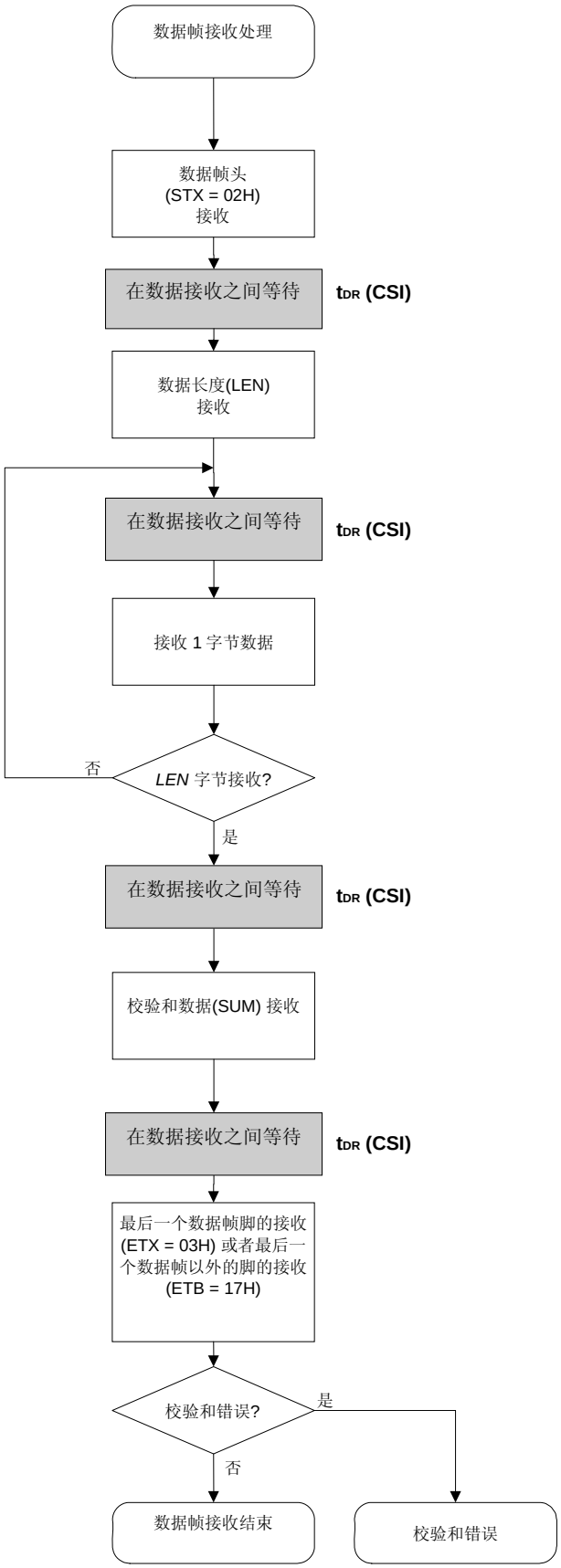
8.1　命令帧发送处理流程图



8.2　数据帧发送处理流程图



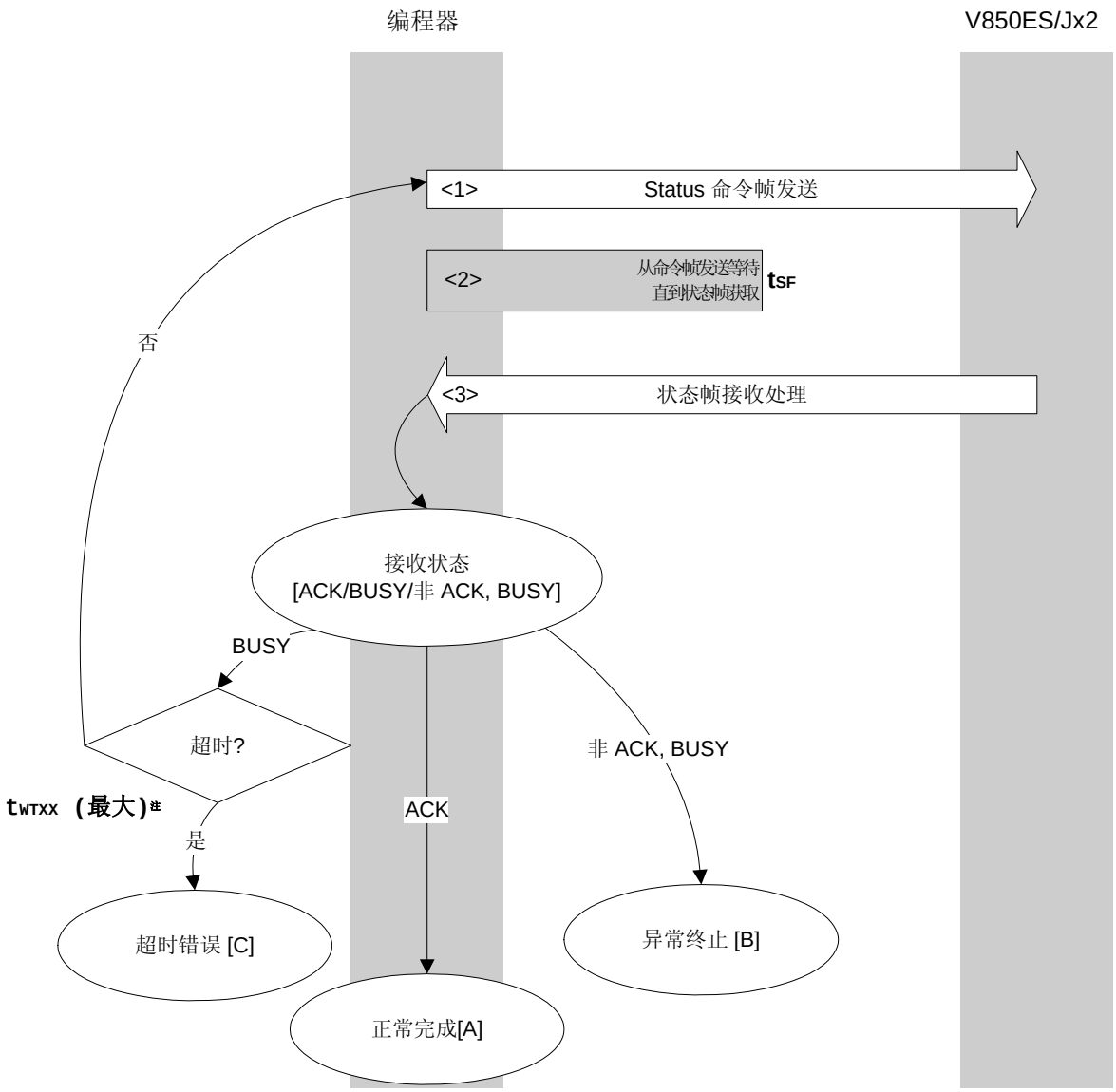
8.3　数据帧接收处理流程图



8.4　Status命令

8.4.1　处理顺序图

Status 命令处理顺序



注　应用的规范根据执行的命令而不同。

8.4.2　处理顺序的说明

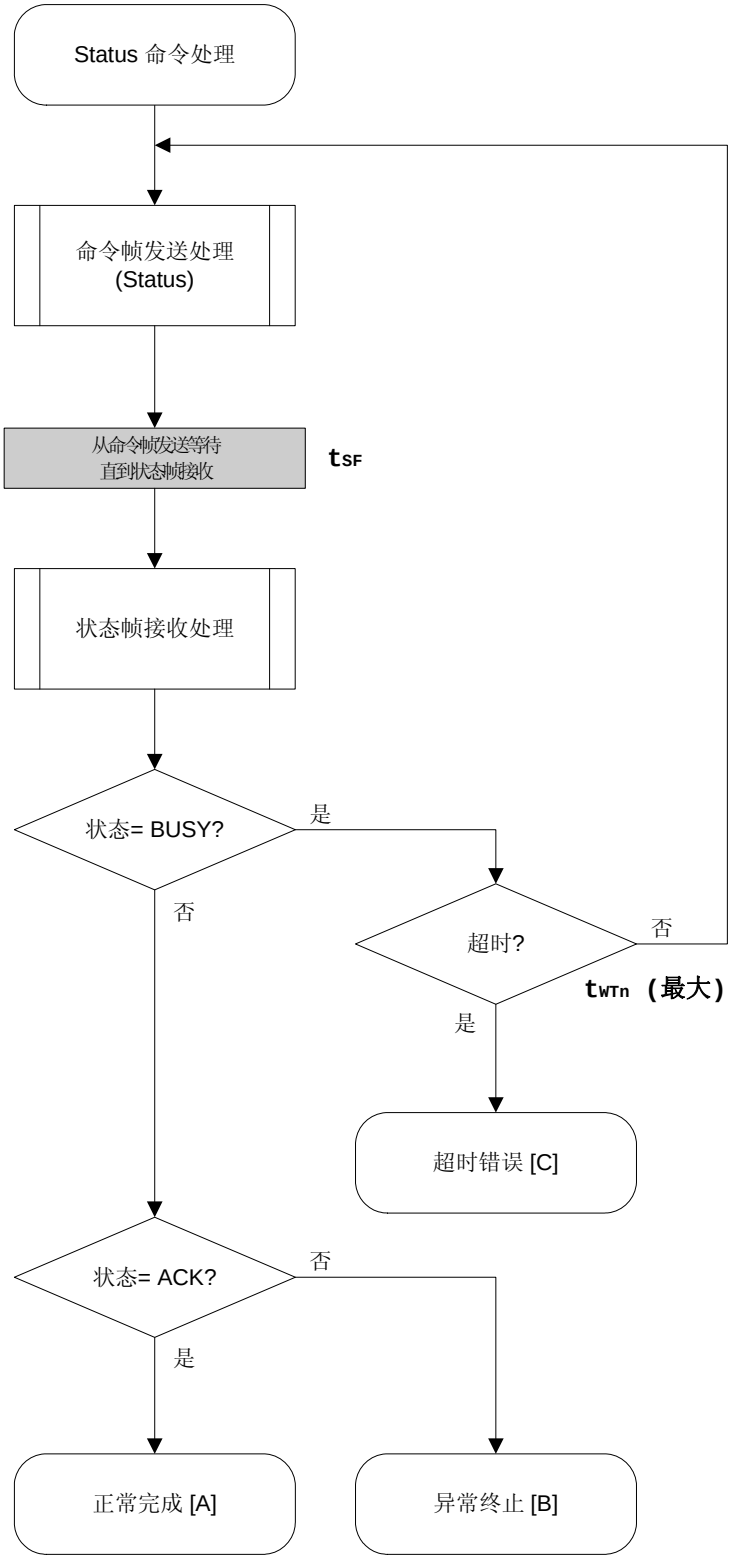
<1>　通过命令帧发送处理发送 Status 命令。
<2>　从命令发送等待直到状态帧接收（等待时间 t_{SF}）。
<3>　检查状态码。

当 ST1 = ACK 时：　　正常完成［A］
当 ST1 = BUSY 时：　　超时检查被执行。 超时时间（t_{WTn}（最大））被作为一个参数传递给这个处理。
如果处理没有超时，从<1>按顺序重新执行。
如果一个超时发生，返回超时错误［C］。
当 ST1 ≠ ACK，BUSY 时：　　异常终止［B］

8.4.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	从 V850ES/Jx2 发送的状态帧被正常接收。
异常终止［B］	命令错误	04H	一个不支持的命令或异常帧被接收。
	参数错误	05H	命令信息（参数）无效。
	校验和错误	07H	从编程器发送的帧的数据异常。
	WWV1 错误	08H	写入错误
	EWV1 错误	0BH	擦除错误
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	校验错误	0EH	对于从编程器发送的帧的数据，一个校验错误发生。
		0FH	
	产品错误	10H	试图执行 Security Set 命令禁止的处理。
	EWV4 错误	11H	内部校验错误/空白错误
	压缩搜索错误	13H	擦除错误
	消极响应（NACK）	15H	消极响应
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	命令发送后，指定的时间已经过去，但是 BUSY 响应仍然被返回。

8.4.4　流程图


Status 命令处理
命令帧发送处理
(Status)
从命令帧发送等待
直到状态帧接收
tSF
状态帧接收处理
状态= BUSY?
是
否
超时?
否
tWTn（最大）
是
超时错误 [C]
状态= ACK?
否
是
正常完成 [A]
异常终止 [B]

8.4.5　范例程序

以下表示 Status 命令处理的一个范例程序。

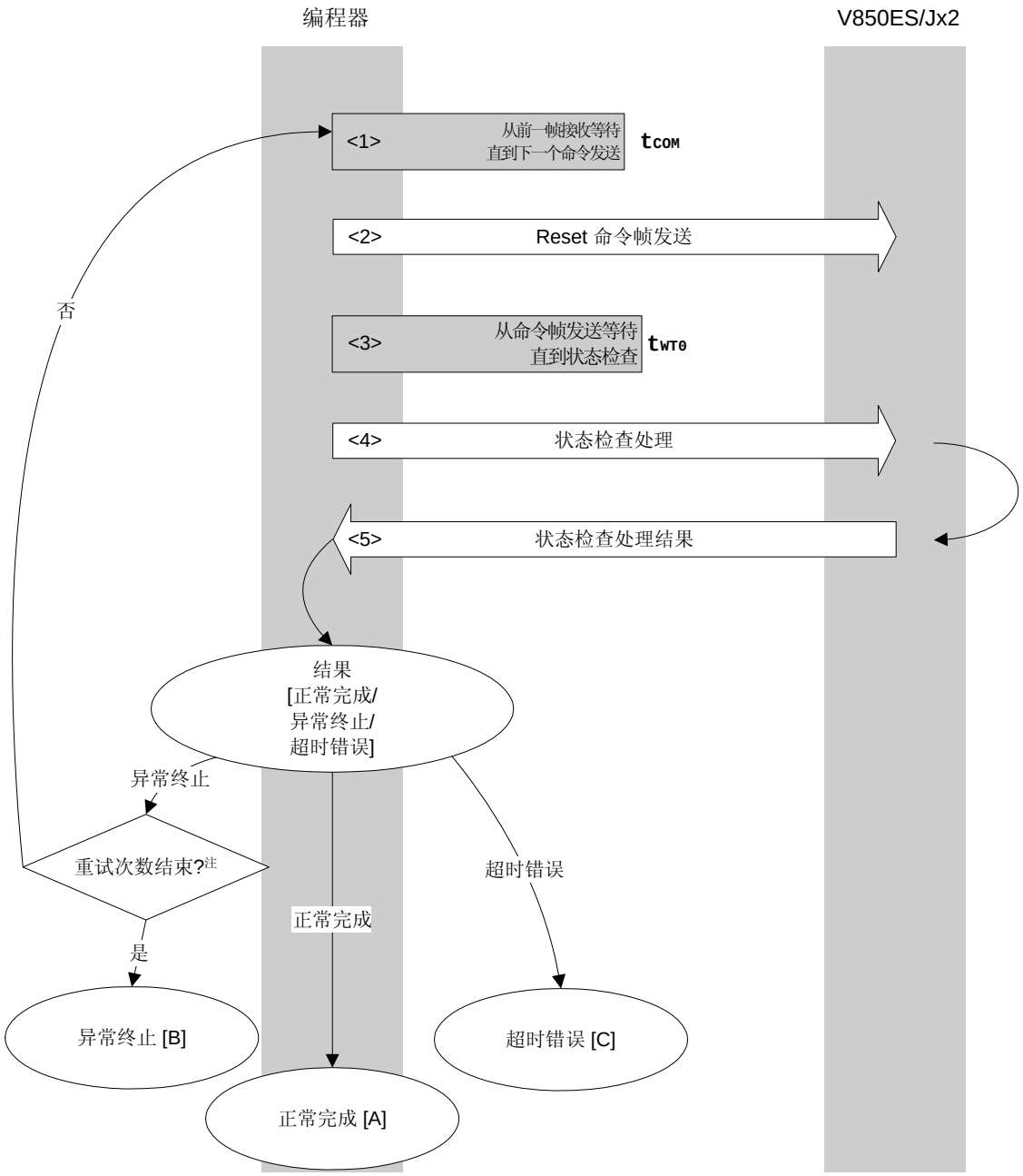
```
/*****************************************************************/
/*                                                               */
/* 获取状态帧（CSI）                                             */
/*                                                               */
/*****************************************************************/
/* [r] u16        ... 解码状态或错误码                           */
/*                                                               */
/* (见 fl.h/fl-proto.h &                                         */
/*       fl.c 中 decode_status()的定义)                          */
/*****************************************************************/
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);    // 发送 "Status"命令帧
                fl_wait(tSF);                                          // 等待

                rc = get_sfrm_csi(fl_rxdata_frm);                      // 获取状态帧

                switch(rc){
                        case    FLC_BUSY:
                                if (check_flto())                      // 超时?
                                        return  FLC_DFTO_ERR;                  // 是，超时 // 如果 ［C］
                                continue;                                      // 否，重试

                        default:                                       // 校验和错误
                                return  rc;

                        case    FLC_NO_ERR:                                    // 没有错误
                                break;

                }
                if (fl_st1 == FLST_BUSY){ // ST1 = BUSY
                        if (check_flto())               // 超时?
                                return  FLC_DFTO_ERR;           // 是，超时 // 如果 ［C］
                        continue;                               // 否，重试
                }

                if (fl_rxdata_frm[OFS_LEN] == 2 && fl_st1 == FLST_ACK && fl_st2 ==
                FLST_BUSY){
                        if (check_flto())               // 超时?
                                return  FLC_DFTO_ERR;           // 是，超时 // 如果 ［C］

                        continue;
                }

                break;                                  // ACK 或其它错误（但是 BUSY）
        }
```

```
        rc = decode_status(fl_st1); // 解码状态来返回码
//      switch(rc) {
//
//              case    FLC_NO_ERR:   return    rc;       break;   // 如果［A］
//              default:              return    rc;       break;   // 如果［B］
//      }
        return rc;

}
```

8.5　Reset 命令

8.5.1　处理顺序图

Reset 命令处理顺序



注　不要超过复位命令发送的重试次数（最多 16 次）。

8.5.2　处理顺序的说明

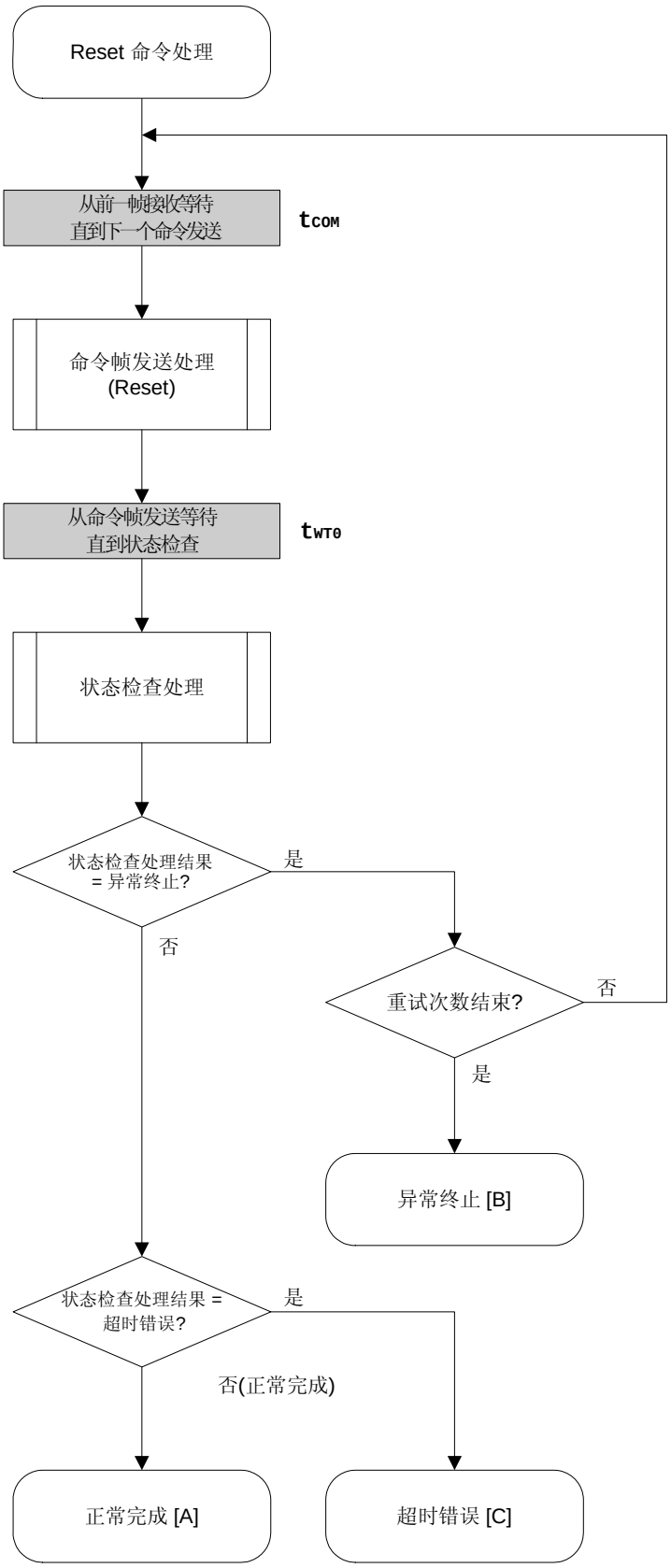
<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Reset 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT0}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　正常完成［A］
当处理异常结束时：　如果重试次数没有结束，顺序从<1>被重新执行。
　　如果重试次数结束，处理异常结束［B］。
当超时错误发生时：　一个超时错误［C］被返回。

8.5.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且编程器和 V850ES/Jx2 之间的同步已经被建立。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态检查处理超时。

8.5.4　流程图


Reset 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Reset)
从命令帧发送等待
直到状态检查
tWT0
状态检查处理
状态检查处理结果
= 异常终止?
是
否
重试次数结束?
否
是
异常终止 [B]
状态检查处理结果 =
超时错误?
是
否(正常完成)
正常完成 [A]
超时错误 [C]

8.5.5　范例程序

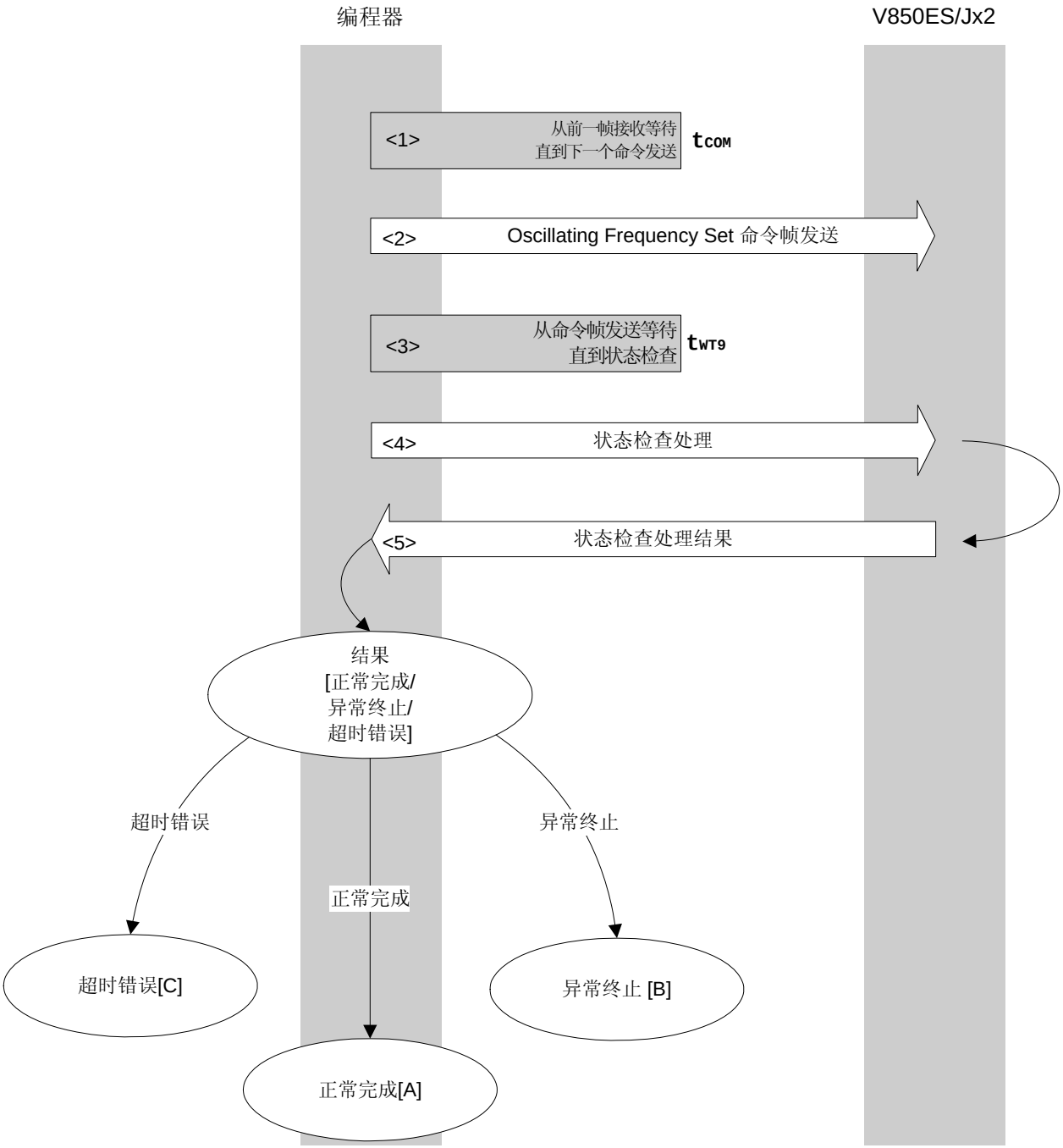
以下表示 Reset 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 复位命令（CSI）                                               */
/*                                                               */
/*****************************************************************/
/* [r] u16      ... 错误码                                       */
/*****************************************************************/
u16             fl_csi_reset(void)
{
        u16     rc;
        u32     retry;

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

                fl_wait(tCOM_CSI);                      // 在发送命令帧前等待

                put_cmd_csi(FL_COM_RESET, 1, fl_cmd_prm);  // 发送 "Reset" 命令帧

                fl_wait(tWT0);

                rc = fl_csi_getstatus(tWT0_MAX);        // 获取状态

                if (rc == FLC_DFTO_ERR)         // 超时错误?
                        break;                          // 是     // 如果［C］
                if (rc == FLC_ACK)                      // Ack ?
                        break;                          // 是     // 如果［A］
                //继续;                                 // 如果［B］（如果从循环退出）
        }
//      switch(rc) {
//
//              case    FLC_NO_ERR:     return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
//              default:                return  rc;     break;  // 如果［B］
//      }
        return  rc;
}
```

8.6　Oscillating Frequency Set命令

8.6.1　处理顺序图

Oscillating Frequency Set 命令处理顺序



8.6.2　处理顺序的说明

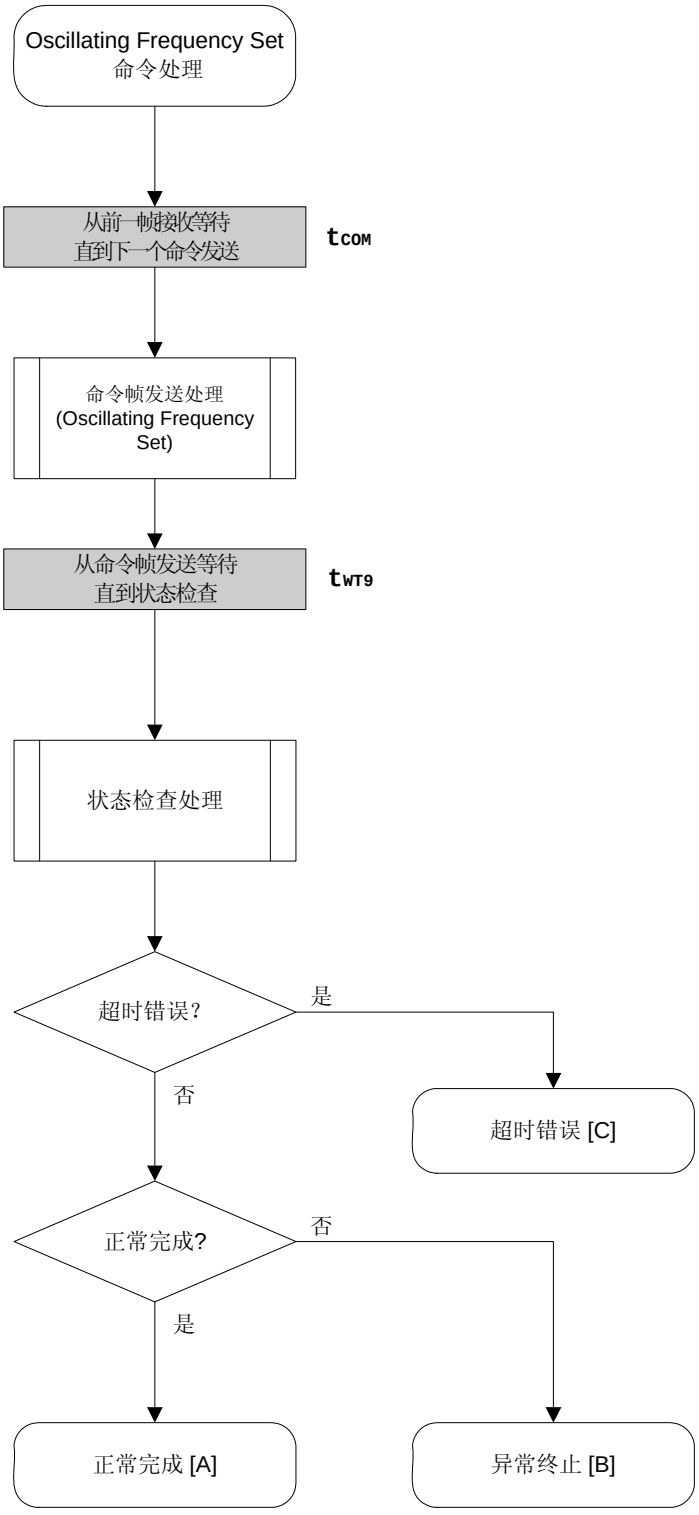
<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Oscillating Frequency Set 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT9}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　　正常完成［A］
当处理异常结束时：　　异常终止［B］
当超时错误发生时：　　一个超时错误［C］被返回。

8.6.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且工作频率被正确设置到 V850ES/Jx2。
异常终止［B］	参数错误	05H	振荡频率值超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。

8.6.4　流程图


Oscillating Frequency Set
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Oscillating Frequency
Set)
从命令帧发送等待
直到状态检查
tWT9
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
正常完成 [A]
异常终止 [B]

8.6.5　范例程序

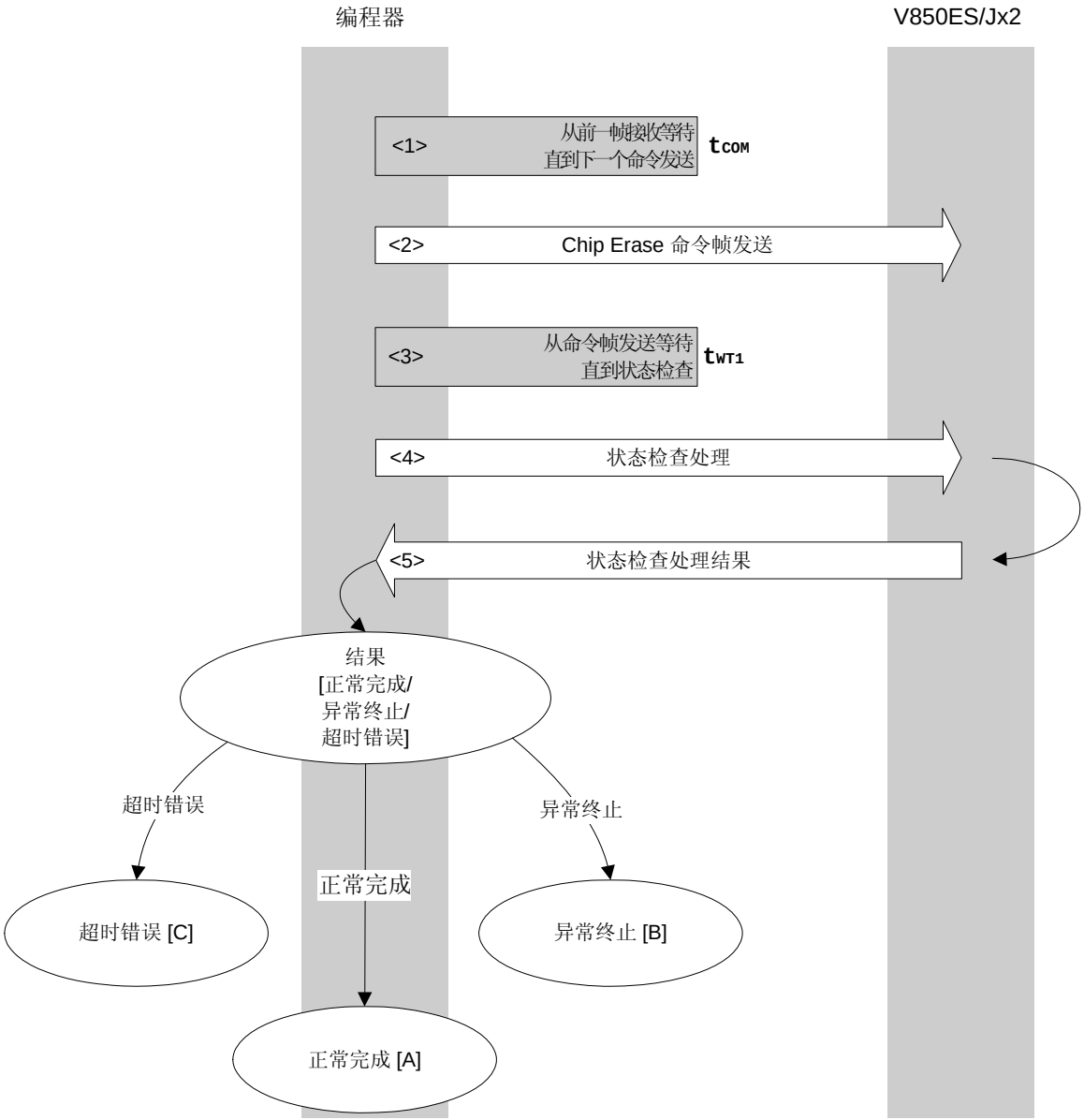
以下表示 Oscillating Frequency Set 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 设置 Flash 器件时钟值命令（CSI）                              */
/*                                                               */
/*****************************************************************/
/* [i] u8 clk[4]  ... 频率数据（D1-D4）                          */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
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_CSI);                                  // 在发送命令帧前等待

        put_cmd_csi(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);
                                                    // 发送 "Oscillation Frequency Set" 命令

        fl_wait(tWT9);

        rc = fl_csi_getstatus(tWT9_MAX);            // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:         return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:       return  rc;     break;  // 如果［C］
//              default:                    return  rc;     break;  // 如果［B］
//      }
        return rc;
}
```

8.7　Chip Erase 命令

8.7.1　处理顺序图

Chip Erase 命令处理顺序



8.7.2　处理顺序的说明

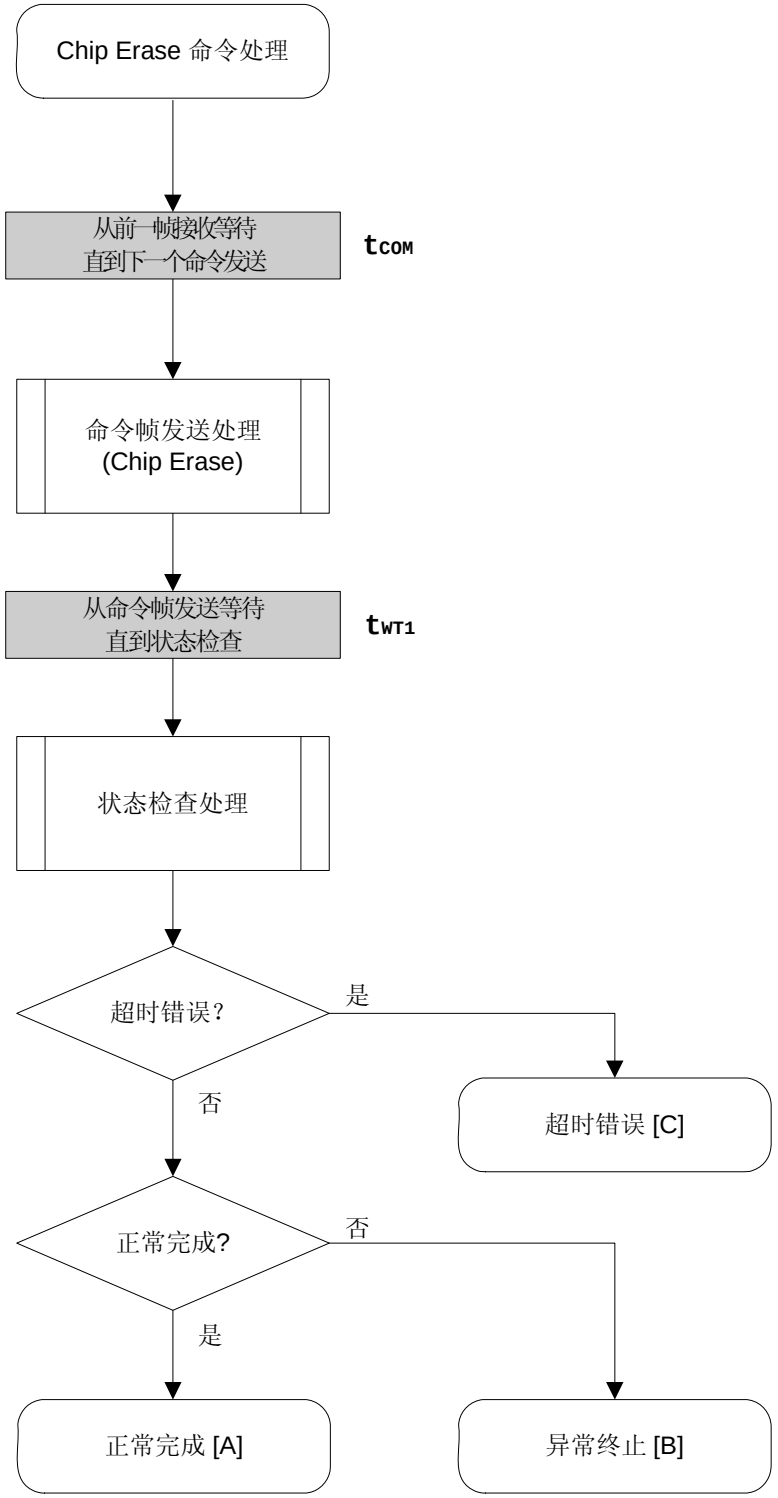
<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Chip Erase 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT1}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　　正常完成［A］
当处理异常结束时：　　异常终止［B］
当超时错误发生时：　　一个超时错误［C］被返回。

8.7.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且正常完成芯片擦除。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

8.7.4　流程图


Chip Erase 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Chip Erase)
从命令帧发送等待
直到状态检查
tWT1
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
是
正常完成 [A]
异常终止 [B]

8.7.5　范例程序

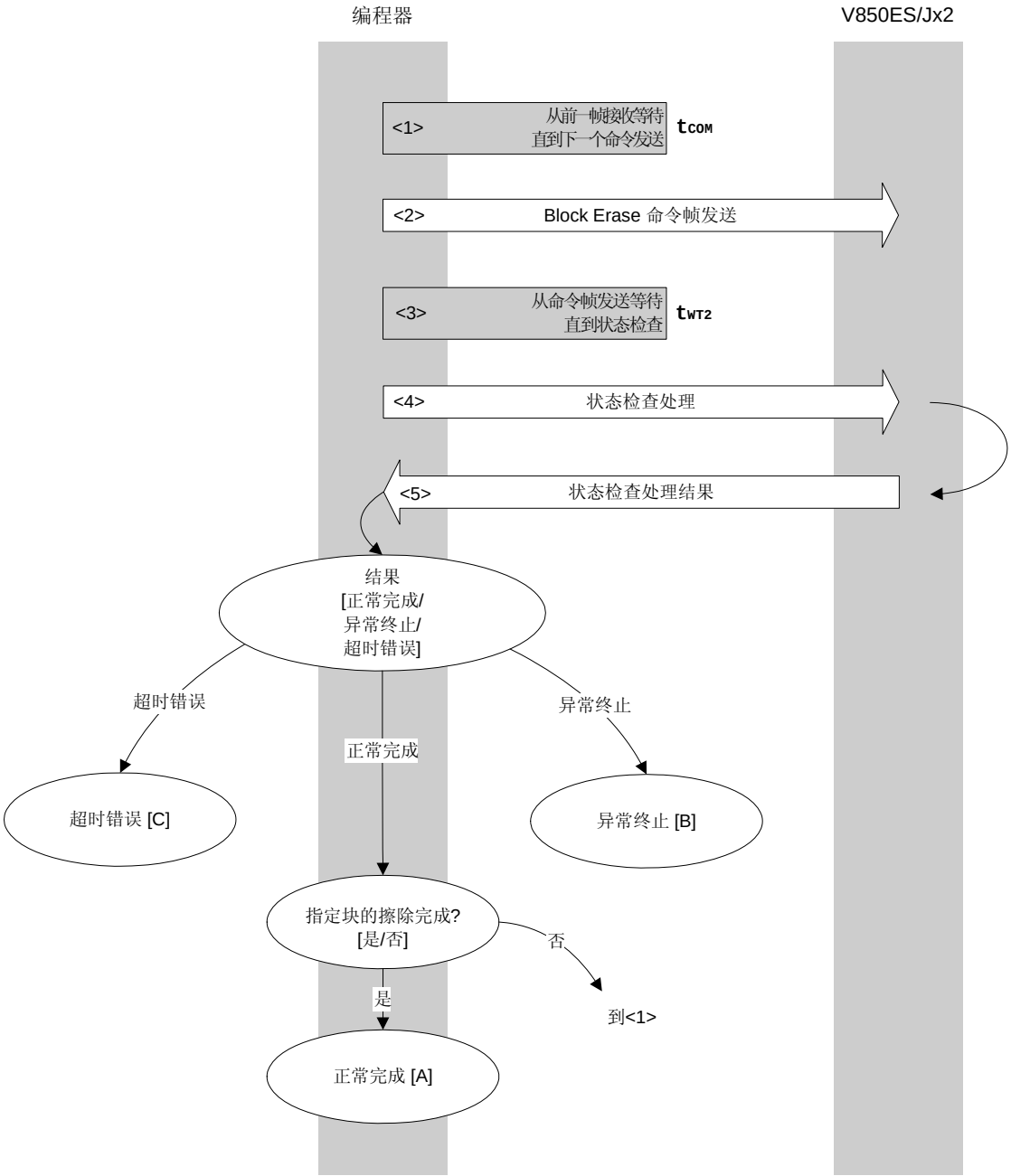
以下表示 Chip Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 擦除全部（整个芯片）命令（CSI）                              */
/*                                                               */
/*****************************************************************/
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_csi_erase_all(void)
{
        u16     rc;


        fl_wait(tCOM_CSI);                                      // 在发送命令帧前等待

        put_cmd_csi(FL_COM_ERASE_CHIP, 1, fl_cmd_prm);          // 发送 "Chip Erase" 命令

        fl_wait(tWT1);

        rc = fl_csi_getstatus(tWT1_MAX);                // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
//              default:                        return  rc;     break;  // 如果［B］
//      }
        return rc;

}
```

8.8　Block Erase 命令

8.8.1　处理顺序图

Block Erase 命令处理顺序



8.8.2　处理顺序的说明

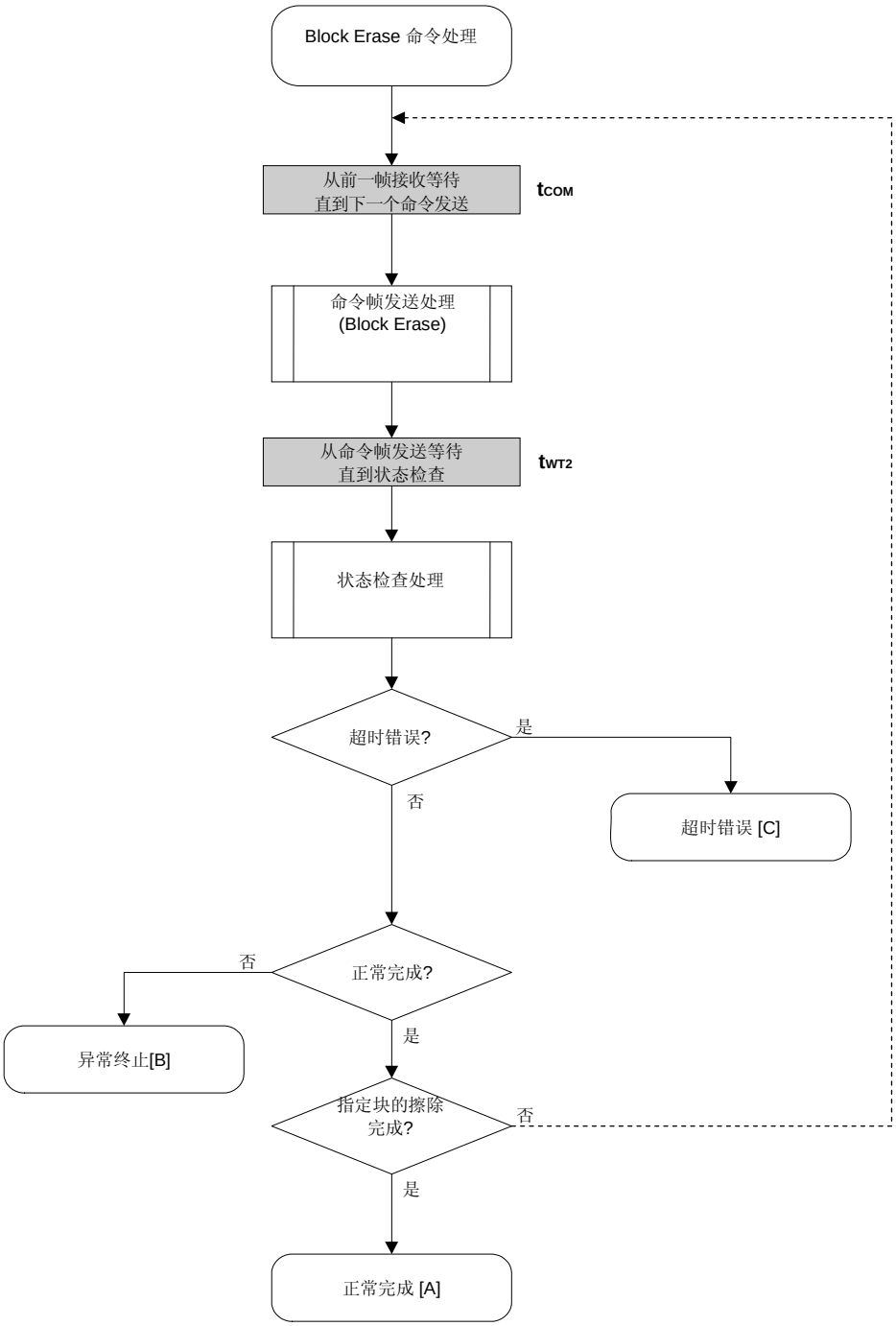
<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Block Erase 命令。
<3>　等待直到命令帧获取（等待时间 t_{WT2}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	当指定块的块擦除没有完成时，处理更改块号并按顺序从<1>重新执行。 当所有指定块的块擦除完成时，处理正常结束［A］。
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

8.8.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且块擦除被正常完成。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	产品错误	10H	芯片擦除在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	WWV1 错误	08H	一个擦除错误发生。
	EWV1 错误	0BH	
	EWV2 错误	0CH	
	EWV3 错误	0DH	
	压缩搜索错误	13H	
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

8.8.4　流程图


Block Erase 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Block Erase)
从命令帧发送等待
直到状态检查
tWT2
状态检查处理
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
异常终止[B]
指定块的擦除
完成?
否
是
正常完成 [A]

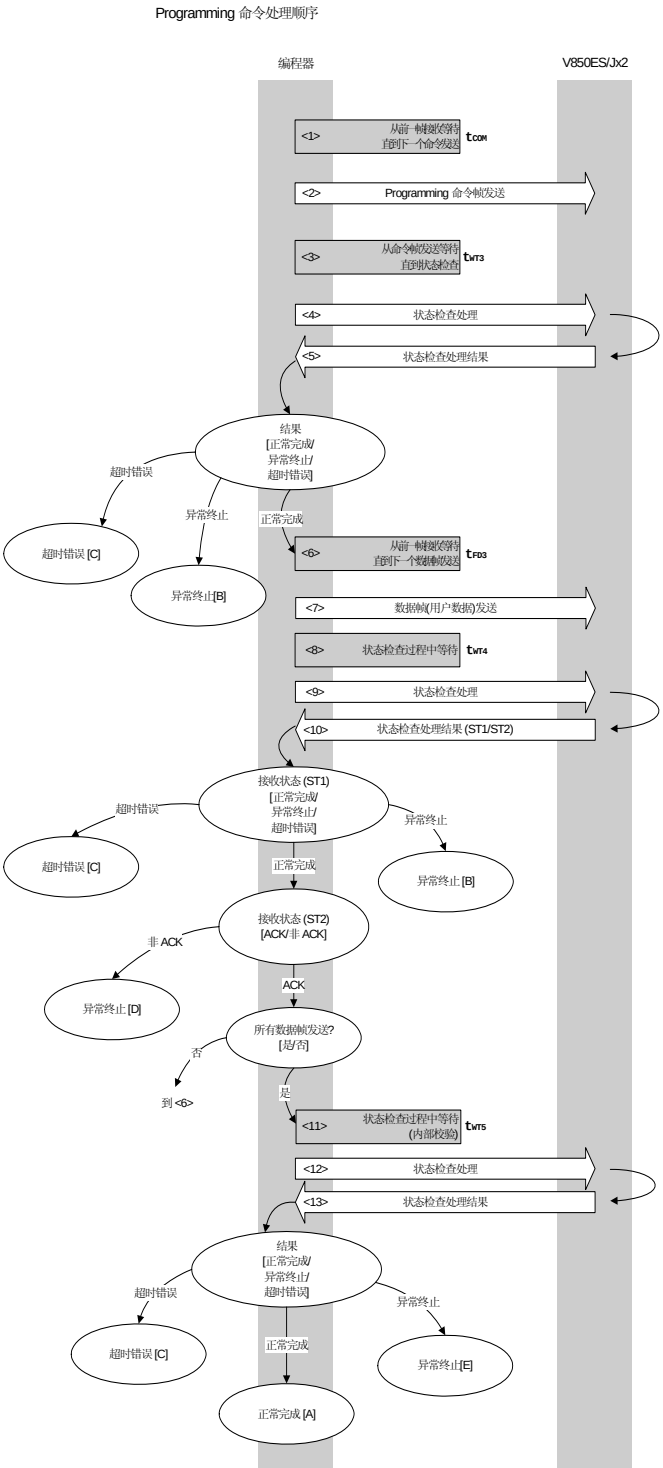
8.8.5　范例程序

以下表示对一个块的 Block Erase 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 擦除块命令（CSI）                                             */
/*                                                               */
/*****************************************************************/
/* [i] u8 block    ... 块号                                      */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_csi_erase_blk(u8 block)
{
        u16     rc;
        u32     wt2, wt2_max;

        fl_cmd_prm[0] = block;      // 设置参数
        wt2     = get_wt2(get_block_size(block));
        wt2_max = get_wt2_max(get_block_size(block));

        fl_wait(tCOM_CSI);                                      // 在发送命令帧前等待

        put_cmd_csi(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm);   // 发送 "Block Erase" 命令

        fl_wait(wt2);


        rc = fl_csi_getstatus(wt2_max);       // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
//              default:                        return  rc;     break;  // 如果［B］
//      }
        return rc;
}
```

8.9　Programming 命令

8.9.1　处理顺序图



8.9.2　处理顺序的说明

<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Programming 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT3}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	到<6>。
当处理异常结束时：	异常终止［B］
当超时错误发生时：	一个超时错误［C］被返回。

<6>　等待直到下一个数据帧发送（等待时间 t_{FD3}（最大））。
<7>　要被写入 V850ES/Jx2 flash 存储器的用户数据通过数据帧发送处理被发送。
<8>　从数据帧（用户数据）发送等待直到状态检查处理（等待时间 t_{WT4}（最大））。
<9>　通过状态检查处理，获取状态帧。
<10> 根据状态检查处理的结果（状态码（ST1/ST2））（同样参考处理顺序图和流程图），以下处理被执行。

当 ST1 = 异常终止时：	异常终止［B］
当 ST1 = 超时错误时：	一个超时错误［C］被返回。
当 ST1 = 正常完成时：	根据 ST2 的值，以下处理被执行。

- 当 ST2 ≠ ACK 时：异常终止［D］
- 当 ST2 = ACK 时：当所有用户数据的发送完成后，到<11>。
  如果仍然存在用户数据要被发送，处理从<6>重新执行。

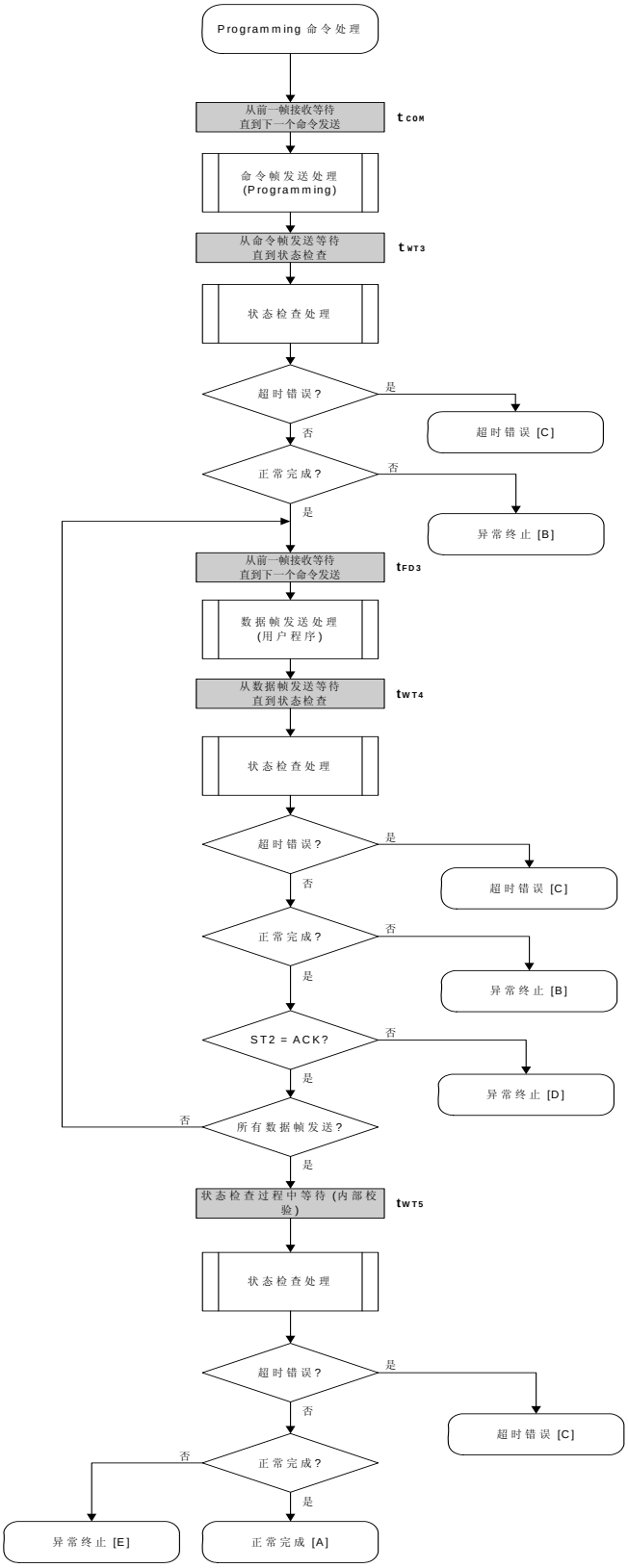
<11> 等待直到状态检查处理（超时时间 t_{WT5}（最大））。
<12> 通过状态检查处理，获取状态帧。
<13> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：	正常完成［A］ （表明内部校验检查在写入完成后正常执行）
当处理异常结束时：	异常终止［E］ （表明内部校验检查在写入完成后没有正常执行）
当超时错误发生时：	一个超时错误［C］被返回。

8.9.3　处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且用户数据被正常写入。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	写入在安全设置中被禁止。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
异常终止［D］	WWV1 错误	08H (ST2)	一个写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止［E］	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

8.9.4　流程图


Programming 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Programming)
从命令帧发送等待
直到状态检查
tWT3
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从前一帧接收等待
直到下一个命令发送
tFD3
数据帧发送处理
(用户程序)
从数据帧发送等待
直到状态检查
tWT4
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
ST2 = ACK?
否
异常终止 [D]
是
所有数据帧发送?
否
是
状态检查过程中等待 (内部校验)
tWT5
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [E]
是
正常完成 [A]

8.9.5　范例程序

以下表示 Programming 命令处理的一个范例程序。

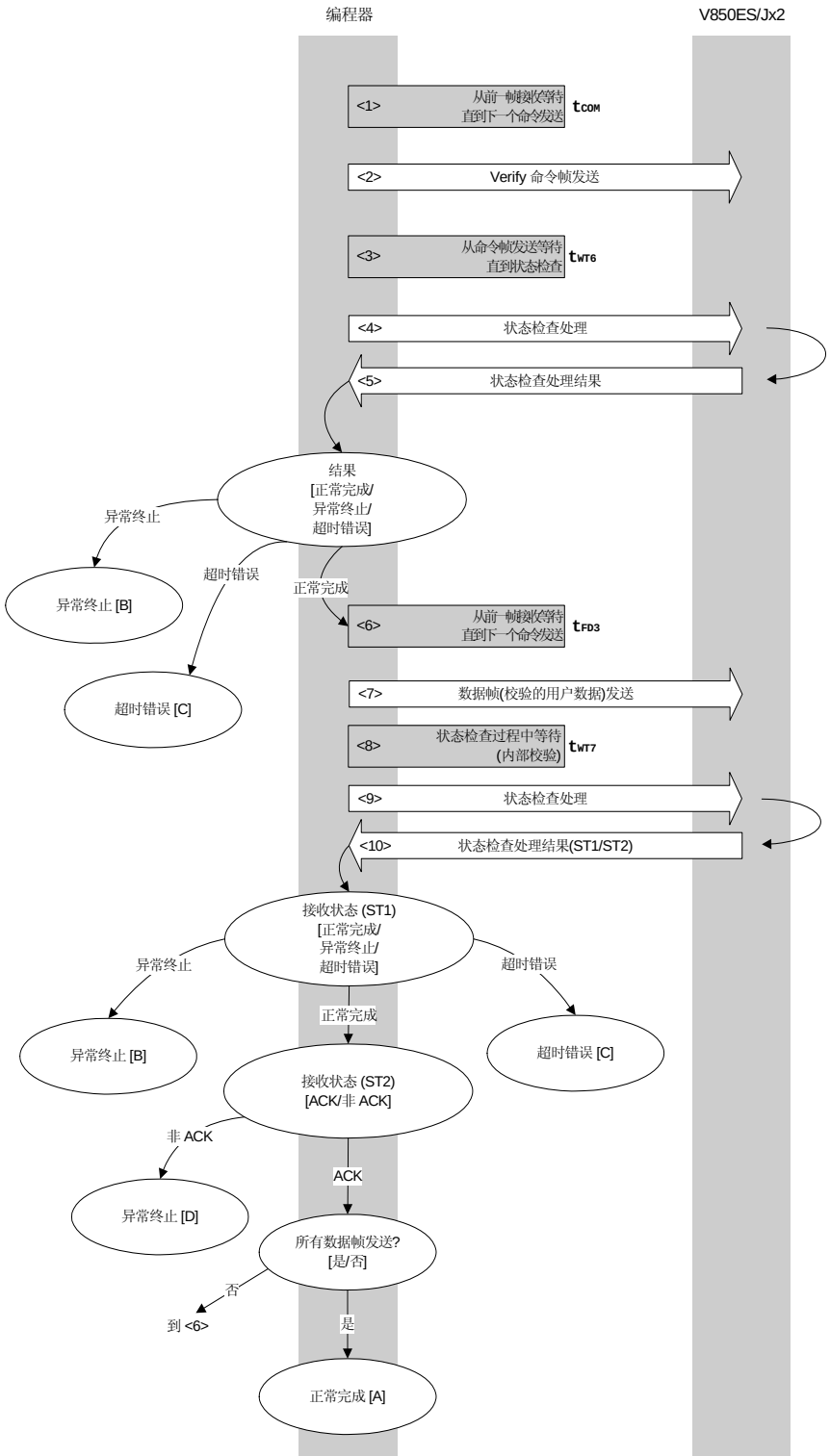
```
/*********************************************************************/
/*                                                                   */
/* 写入命令（CSI）                                                   */
/*                                                                   */
/*********************************************************************/
/* [i] u32 top     ... 起始地址                                      */
/* [i] u32 bottom  ... 结束地址                                      */
/* [r] u16         ... 错误码                                        */
/*********************************************************************/
u16             fl_csi_write(u32 top, u32 bottom)
{
        u16     rc;
        u32     send_head, send_size;
        bool    is_end;
        u32     wt5, wt5_max;

        // set params
        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL
        wt5     = get_wt5(bottom - top + 1);
        wt5_max = get_wt5_max(bottom - top + 1);


        /*************************************************/
        /*      发送命令 & 检查状态                      */
        /*************************************************/

        fl_wait(tCOM_CSI);
        put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm);       // 发送 "Programming" 命令
        fl_wait(tWT3);

        rc = fl_csi_getstatus(tWT3_MAX);                // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
                default:                return  rc;     break;  // 如果［B］
        }

        /*************************************************/
        /*      发送用户数据                             */
        /*************************************************/
        send_head = top;

        while(1){

                if ((bottom - send_head) > 256){        // 剩余大小 > 256 ?
                        is_end = false;                 // 是，不是最后一帧
                        send_size  = 256;               // 发送大小 = 256 字节
                }
                else{
```

```
                is_end = true;
                send_size = bottom - send_head + 1;
                                        // 发送大小 = (底部 – 发送头)+1 字节
            }

            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                                                    // 设置数据帧有效载荷
            send_head += send_size;


            fl_wait(tFD3);                          // 在发送数据帧前等待
            put_dfrm_csi(send_size, fl_txdata_frm, is_end);
                                                    // 发送数据帧（用户数据）
            fl_wait(tWT4);                          // 等待


            rc = fl_csi_getstatus(tWT4_MAX);        // 获取状态帧
            switch(rc) {
                case    FLC_NO_ERR:                         break;  // 继续
            //  case    FLC_DFTO_ERR:   return  rc;  break;  // 如果［C］
                default:                return  rc;  break;  // 如果［B］
            }
            if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
                rc = decode_status(fl_st2); // 否
                return  rc;                         // 如果［D］
            }

            if (is_end)                             // 发送所有用户数据 ?
                break;                              // 是
            //继续;
        }
        /************************************************/
        /*    内部检查校验                              */
        /************************************************/

        fl_wait(wt5);                               // 等待

        rc = fl_csi_getstatus(wt5_max);             // 获取状态帧
//      switch(rc) {
//          case    FLC_NO_ERR:     return  rc;  break;  // 如果［A］
//          case    FLC_DFTO_ERR:   return  rc;  break;  // 如果［C］
//          default:                return  rc;  break;  // 如果［E］
//      }
        return  rc;

}
```

8.10 Verify 命令

8.10.1 处理顺序图

Verify 命令处理顺序



8.10.2 处理顺序的说明

<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Verify 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT6}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　到<6>。
当处理异常结束时：　异常终止［B］
当超时错误发生时：　一个超时错误［C］被返回。

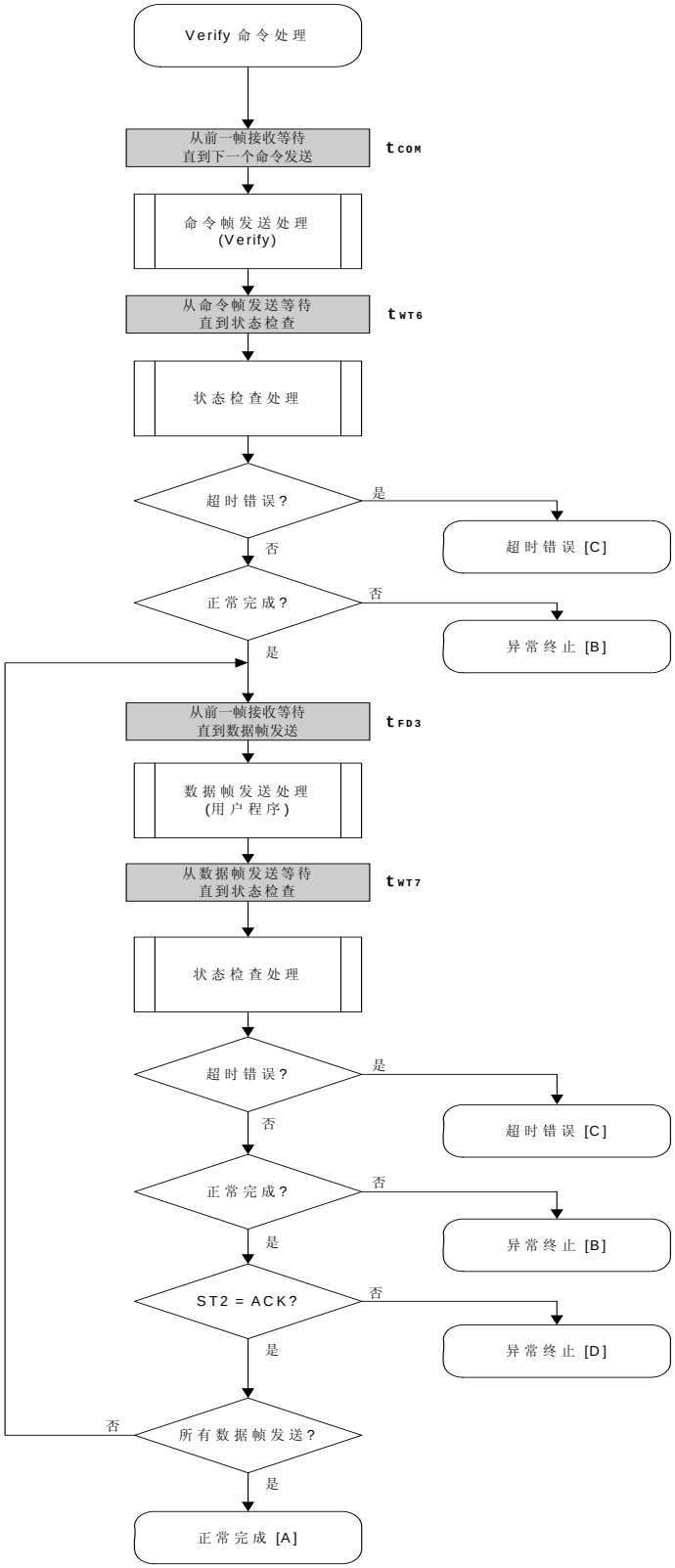
<6>　从前一帧接收等待直到下一个数据帧发送（等待时间 t_{FD3}）。
<7>　通过数据帧发送处理发送校验用的用户数据。
<8>　从数据帧发送等待直到状态检查处理（等待时间 t_{WT7}（最大））。
<9>　通过状态检查处理，获取状态帧被。
根据状态检查处理的结果（状态码（ST1/ST2））（同样参考处理顺序图和流程图），以下处理被执行。

当 ST1 = 异常终止时：　异常终止［B］
当 ST1 = 超时错误时：　一个超时错误［C］被返回。
当 ST1 = 正常完成时：　根据 ST2 的值，以下处理被执行。
- 当 ST2 ≠ ACK 时：　异常终止［D］
- 当 ST2 = ACK 时：　如果所有数据帧的发送完成，处理正常结束［A］。
　　如果仍然存在数据帧要被发送，处理从<6>重新执行。

8.10.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验被正常完成。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
异常终止［D］	校验错误	0FH (ST2)	校验失败或者另一个错误发生。
		0EH	
	程序错误	16H	一个程序错误发生。

8.10.4 流程图


Verify 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Verify)
从命令帧发送等待
直到状态检查
tWT6
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成？
否
异常终止 [B]
是
从前一帧接收等待
直到数据帧发送
tFD3
数据帧发送处理
(用户程序)
从数据帧发送等待
直到状态检查
tWT7
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成？
否
异常终止 [B]
是
ST2 = ACK?
否
异常终止 [D]
是
所有数据帧发送？
否
是
正常完成 [A]

8.10.5 范例程序

以下表示 Verify 命令处理的一个范例程序。

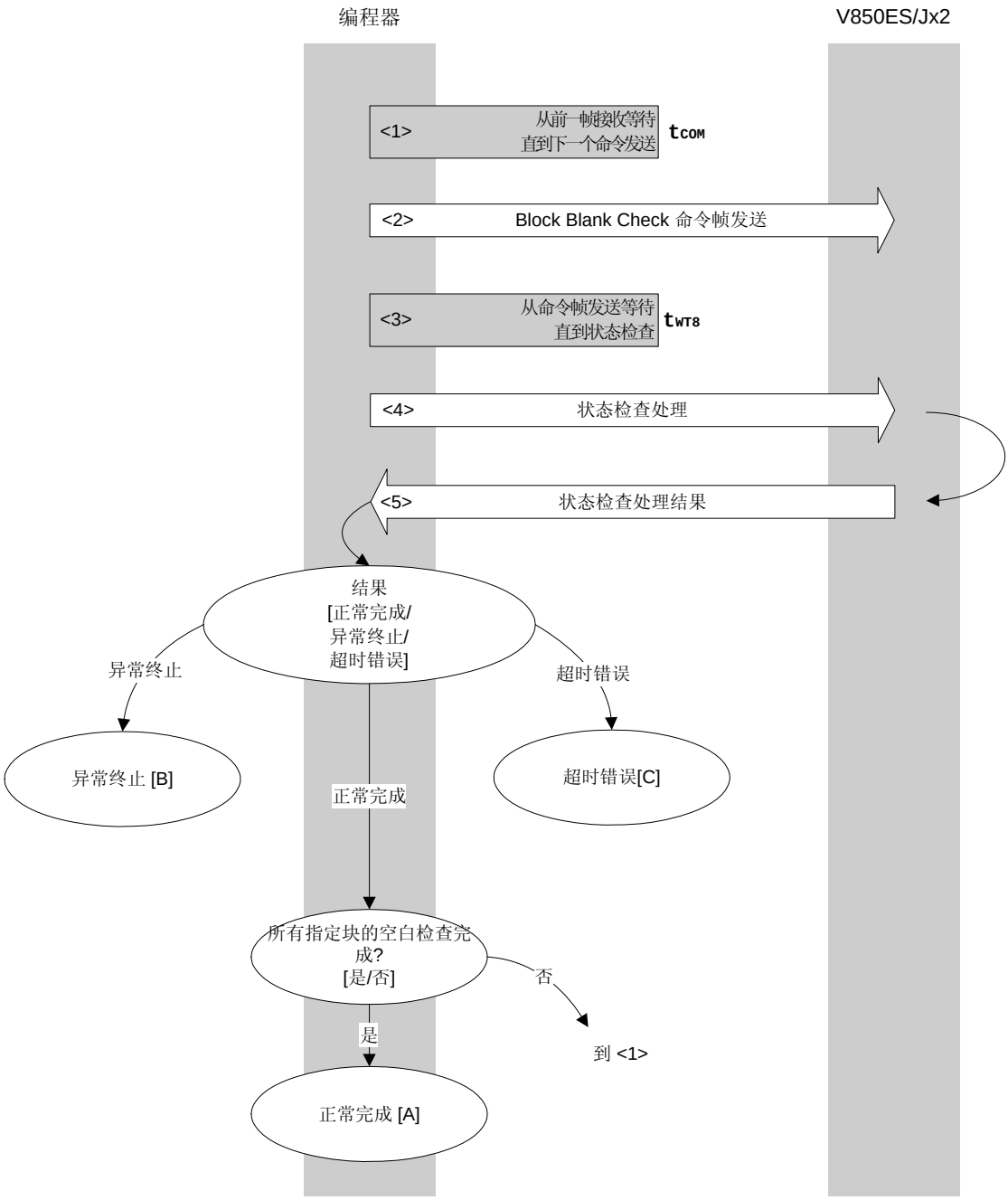
```
/*****************************************************************/
/*                                                               */
/* 校验命令（CSI）                                               */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... 起始地址                                  */
/* [i] u32 bottom  ... 结束地址                                  */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
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);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL


        /*************************************************/
        /*     发送命令 & 检查状态                       */
        /*************************************************/
        fl_wait(tCOM_CSI);
        put_cmd_csi(FL_COM_VERIFY, 7, fl_cmd_prm);      // 发送 "Verify" 命令
        fl_wait(tWT6);

        rc = fl_csi_getstatus(tWT6_MAX);                // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:       return  rc;         break;  // 如果［C］
                default:                    return  rc;         break;  // 如果［B］
        }


        /*************************************************/
        /*     发送用户数据                              */
        /*************************************************/
        send_head = top;

        while(1){

                if ((bottom - send_head) > 256){        // 剩余大小 > 256 ?
                        is_end = false;                 // 是，不是最后一帧
                        send_size  = 256;               // 发送大小 = 256 字节
                }
```

```
            else{
                is_end = true;
                send_size = bottom - send_head + 1;
                                        // 发送大小 = (底部 – 发送头)+1 字节

            }

            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                                                    // 设置数据帧有效载荷
            send_head += send_size;

            fl_wait(tFD3);                          // 在发送数据帧前等待
            put_dfrm_csi(send_size, fl_txdata_frm, is_end);     // 发送数据帧
            fl_wait(tWT7);                          // 等待

            rc = fl_csi_getstatus(tWT7_MAX);        // 获取状态帧
            switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
            //  case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
                default:                return  rc;     break;  // 如果［B］
            }
            if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
                rc = decode_status(fl_st2); // 否
                return  rc;                             // 如果［D］
            }

            if (is_end)                             // 发送所有用户数据 ?
                break;                              // 是
            //继续;

        }
        return  FLC_NO_ERR;  // 如果［A］

}
```

8.11 Block Blank Check 命令

8.11.1 处理顺序图

Block Blank Check 命令处理顺序



8.11.2 处理顺序的说明

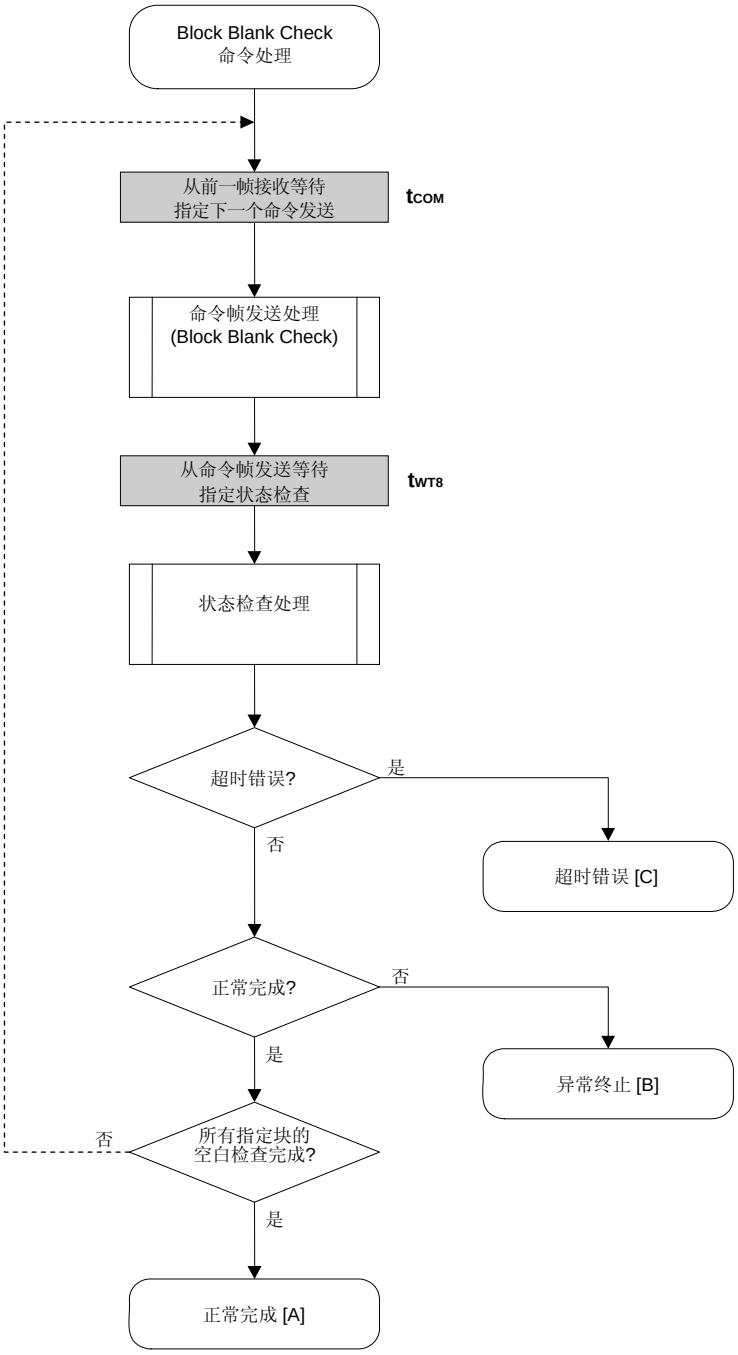
<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Block Blank Check 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT8}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当超时错误发生时：	一个超时错误［C］被返回。
当处理异常结束时：	异常终止［B］
当处理正常结束时：	当指定块的空白检查没有完成时，处理更改块号并按顺序从<1>重新执行。 当所有指定块的空白检查完成时，处理正常结束［A］。

8.11.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	这个命令被正常执行，并且所有指定的块空白。
异常终止［B］	参数错误	05H	块号超出范围。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	EWV4 错误	11H	指定块的 flash 存储器不是空白。
	程序错误	16H	一个程序错误发生。
超时错误［C］		–	状态帧在特定时间内没有被接收。

8.11.4 流程图


Block Blank Check
命令处理
从前一帧接收等待
指定下一个命令发送
tCOM
命令帧发送处理
(Block Blank Check)
从命令帧发送等待
指定状态检查
tWT8
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
所有指定块的
空白检查完成?
否
是
正常完成 [A]

8.11.5 范例程序

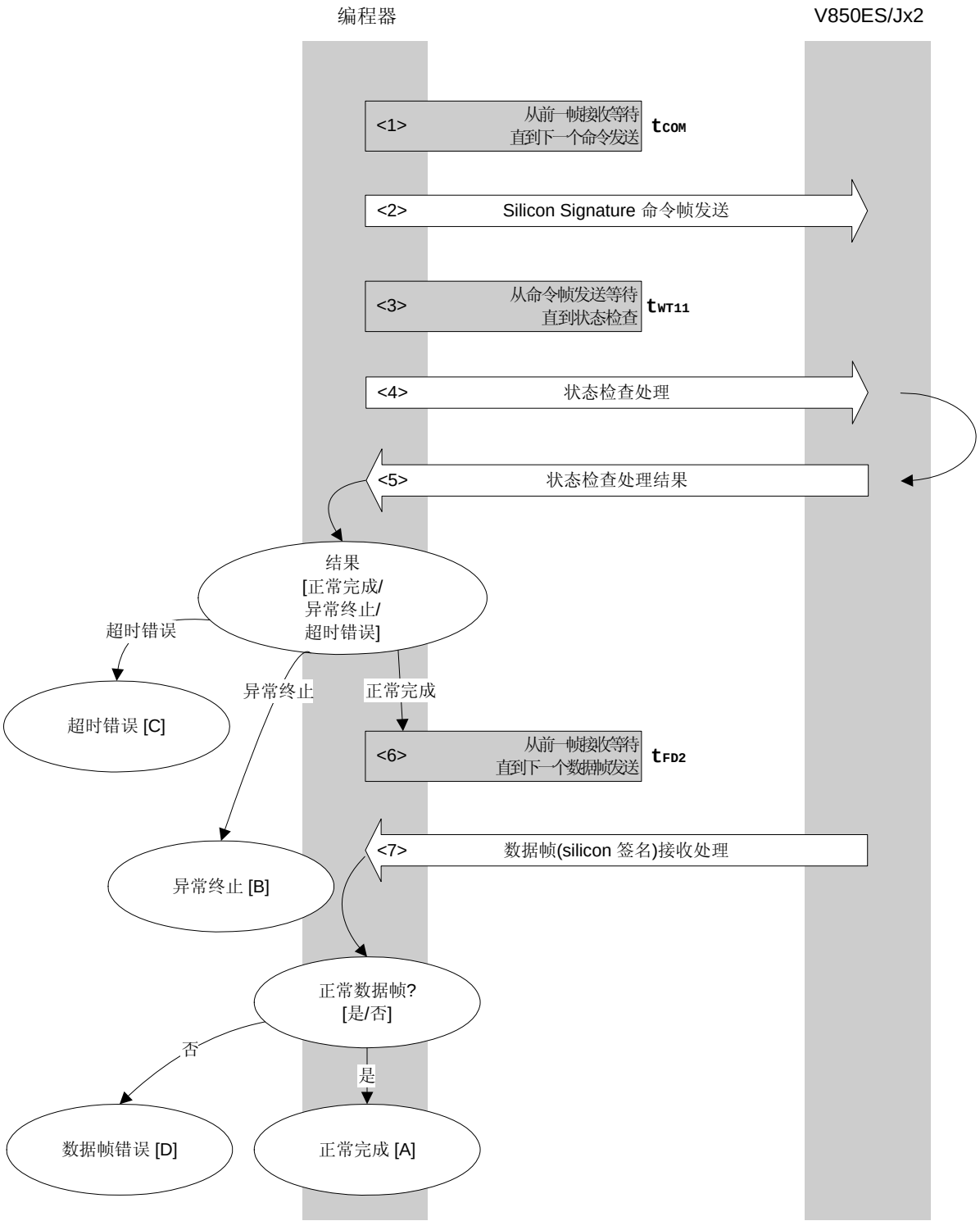
以下表示对一个块的 Block Blank Check 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 块空白检查命令（CSI）                                         */
/*                                                               */
/*****************************************************************/
/* [i]  u8 block  ... 块号                                       */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_csi_blk_blank_chk(u8 block)
{
        u16     rc;
        u32     wt8, wt8_max;

        fl_cmd_prm[0] = block;      // "BLK"
        wt8     = get_wt8(get_block_size(block));
        wt8_max = get_wt8_max(get_block_size(block));


        fl_wait(tCOM_CSI);                                  // 在发送命令帧前等待

        put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);
                                                            // 发送 "Block Blank Check" 命令

        fl_wait(wt8);

        rc = fl_csi_getstatus(wt8_max);       // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:             return  rc;     break;  // 如果［A］
//              case    FLC_DFTO_ERR:           return  rc;     break;  // 如果［C］
//              default:                        return  rc;     break;  // 如果［B］
//      }
        return rc;
}
```

8.12 Silicon Signature 命令

8.12.1 处理顺序图

Silicon Signature 命令处理顺序



8.12.2 处理顺序的说明

<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> 通过命令帧发送处理发送 Silicon Signature 命令。
<3> 从命令发送等待直到状态检查处理（等待时间 t_{WT11}（最大））。
<4> 通过状态检查处理，获取状态帧。
<5> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　到<6>。
当处理异常结束时：　异常终止［B］
当超时错误发生时：　一个超时错误［C］被返回。

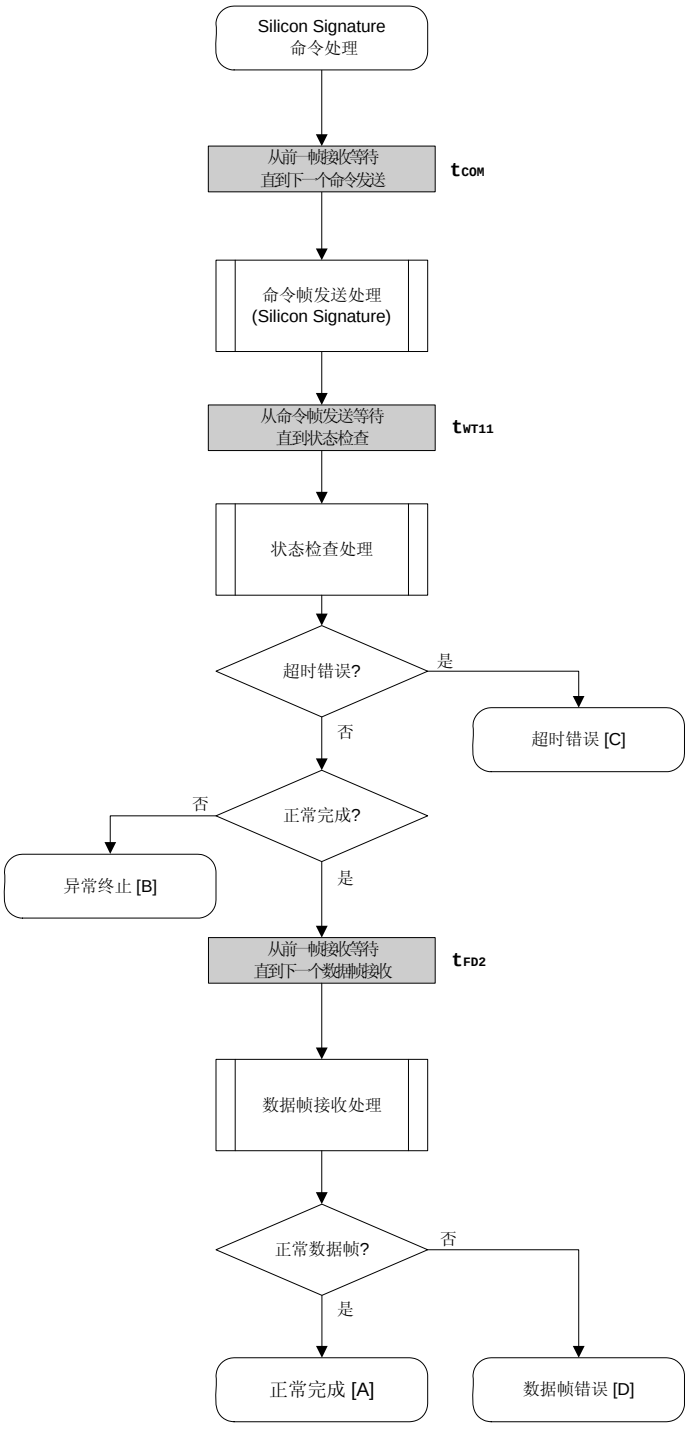
<6> 从前一帧接收等待直到下一个命令发送（等待时间 t_{FD2}）。
<7> 检查接收的数据帧（silicon 签名数据）。

如果数据帧正常：　正常完成［A］
如果数据帧异常：　数据帧错误 ［D］

8.12.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且 silicon 签名被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
数据帧错误 ［D］		–	作为 silicon 签名数据接收到的数据帧的校验和不匹配。

8.12.4 流程图


Silicon Signature
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Silicon Signature)
从命令帧发送等待
直到状态检查
tWT11
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从前一帧接收等待
直到下一个数据帧接收
tFD2
数据帧接收处理
正常数据帧?
否
是
正常完成 [A]
数据帧错误 [D]

8.12.5 范例程序

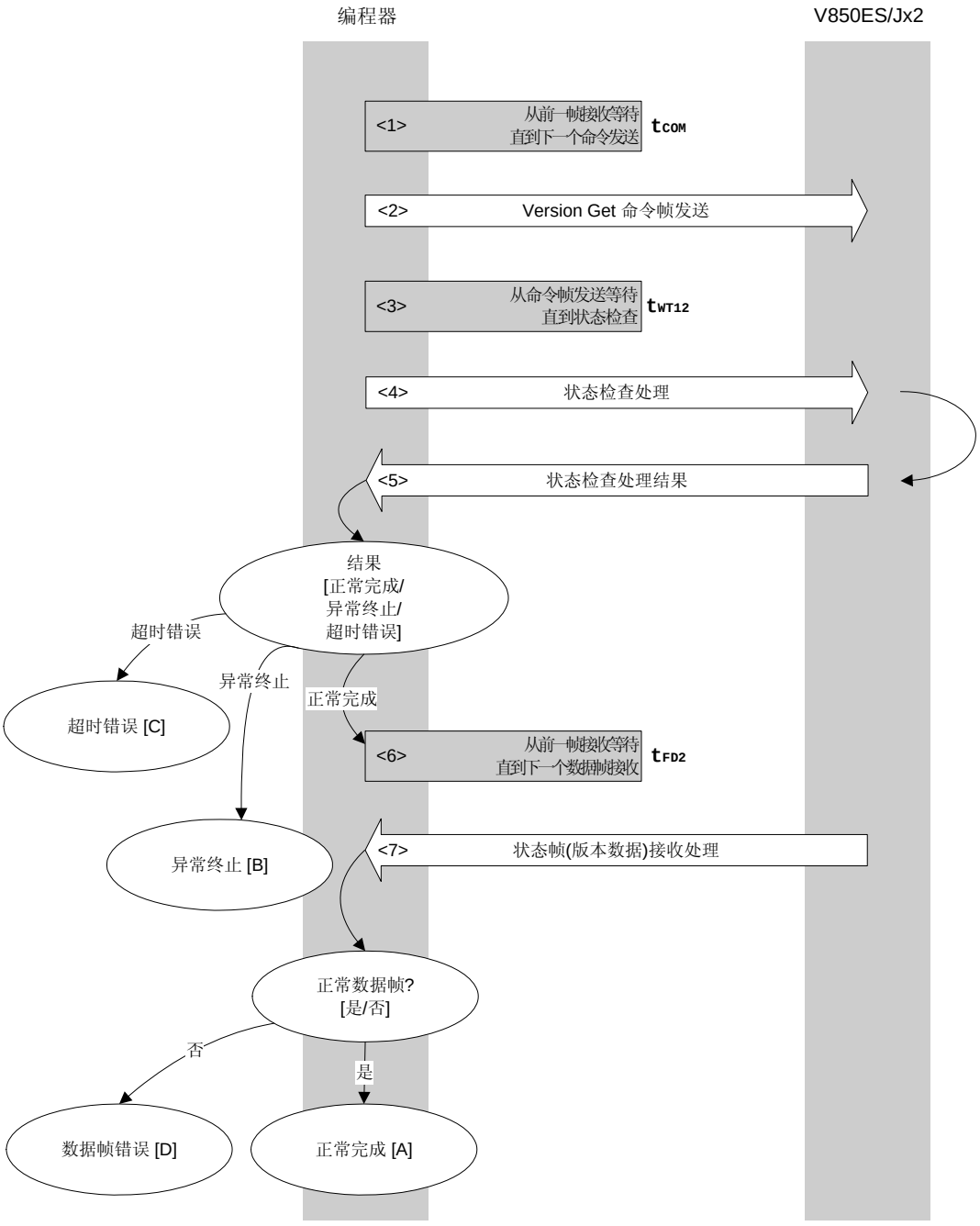
以下表示 Silicon Signature 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获得 silicon 签名命令（CSI）                                  */
/*                                                               */
/*****************************************************************/
/* [i] u8 *sig    ... 签名保存区域的指针                         */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_csi_getsig(u8 *sig)
{
        u16     rc;

        fl_wait(tCOM_CSI);                                      // 在发送命令帧前等待

        put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);
                                                                // 发送 "Silicon Signature" 命令

        fl_wait(tWT11);

        rc = fl_csi_getstatus(tWT11_MAX);               // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
                default:                return  rc;     break;  // 如果［B］
        }

        fl_wait(tFD2_SIG);                                      // 在获得数据帧前等待

        rc = get_dfrm_csi(fl_rxdata_frm);               // 获取数据帧（签名数据）

        if (rc){                                                // 如果没有错误，
                return rc;                                      // 如果［D］
        }
        memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                                // 复制签名数据
        return  rc;                                             // 如果［A］
}
```

8.13 Version Get 命令

8.13.1 处理顺序图

Version Get 命令处理顺序



8.13.2 处理顺序的说明

<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Version Get 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT12}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　到<6>。
当处理异常结束时：　异常终止［B］
当超时错误发生时：　一个超时错误［C］被返回。

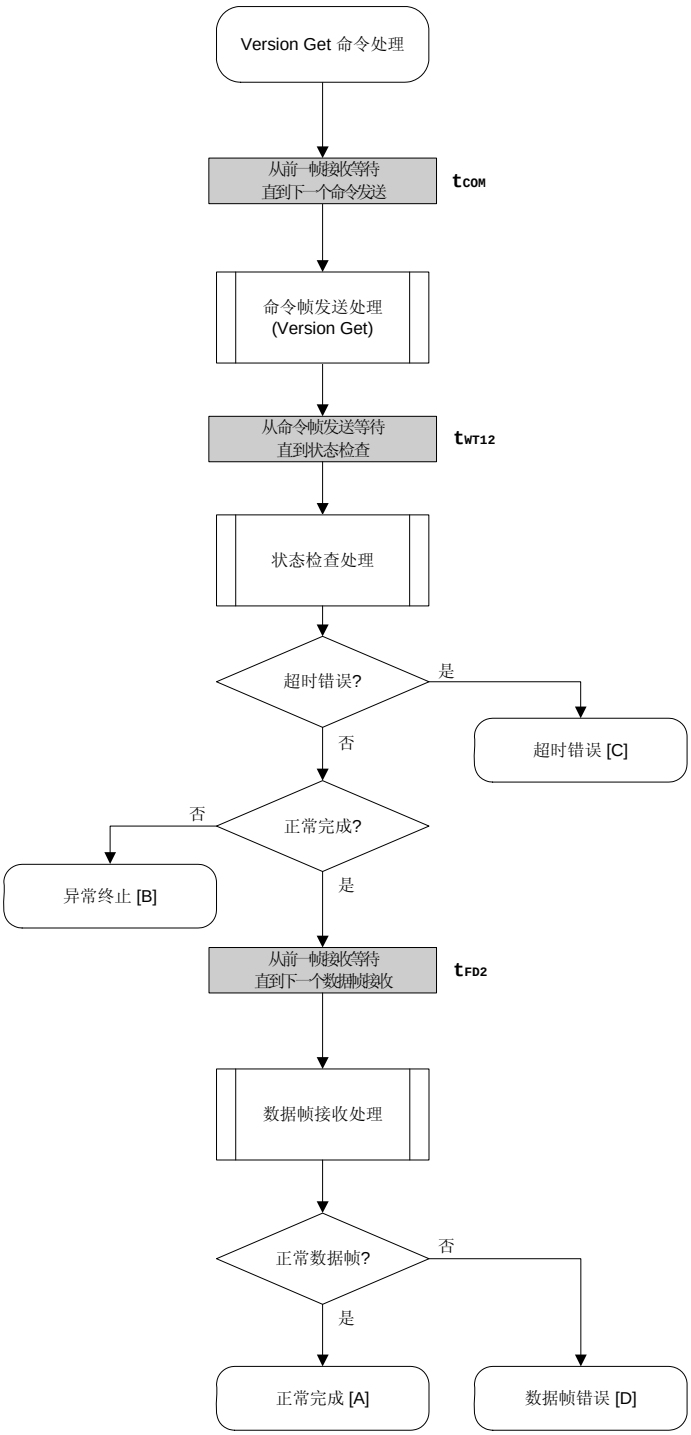
<6>　从前一帧接收等待直到下一个命令发送（等待时间 t_{FD2}）。
<7>　检查接收到的数据帧（版本数据）。

如果数据帧正常：　正常完成［A］
如果数据帧异常：　数据帧错误 ［D］

8.13.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且版本数据被正常获取。
异常终止［B］	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
数据帧错误 ［D］		–	作为版本数据接收到的数据帧的校验和不匹配。

8.13.4 流程图


Version Get 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Version Get)
从命令帧发送等待
直到状态检查
tWT12
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从前一帧接收等待
直到下一个数据帧接收
tFD2
数据帧接收处理
正常数据帧?
否
是
正常完成 [A]
数据帧错误 [D]

8.13.5 范例程序

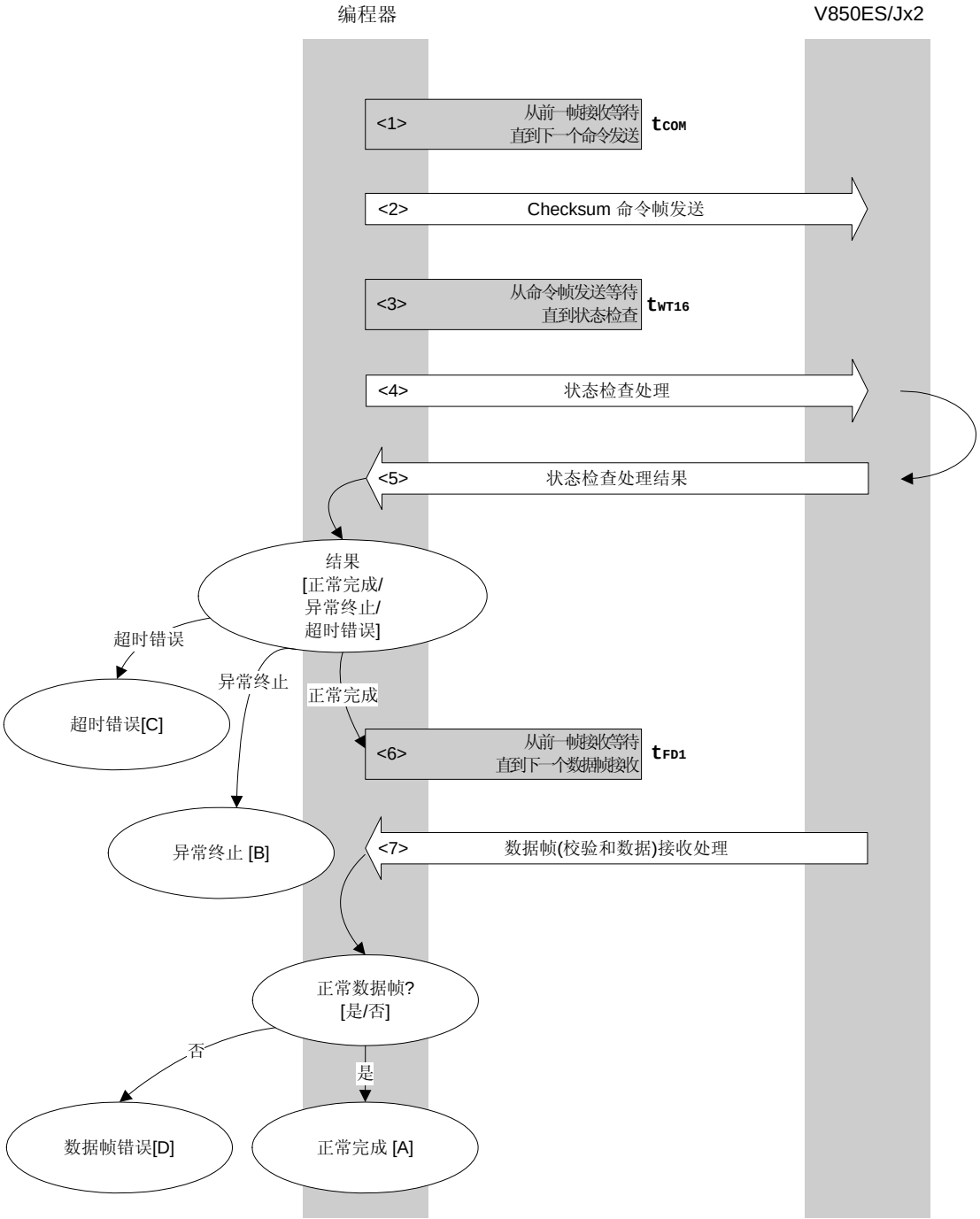
以下表示 Version Get 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获取设备 / 固件版本命令（CSI）                                */
/*                                                               */
/*****************************************************************/
/* [i] u8 *buf    ... 版本数据保存区域的指针                     */
/* [r] u16        ... 错误码                                     */
/*****************************************************************/
u16             fl_csi_getver(u8 *buf)
{
        u16     rc;


        fl_wait(tCOM_CSI);                                      // 在发送命令帧前等待

        put_cmd_csi(FL_COM_GET_VERSION, 1, fl_cmd_prm);   // 发送 "Version Get" 命令

        fl_wait(tWT12);

        rc = fl_csi_getstatus(tWT12_MAX);           // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:       return  rc;         break;  // 如果［C］
                default:                    return  rc;         break;  // 如果［B］
        }

        fl_wait(tFD2_VG);                           // 在获得数据帧前等待

        rc = get_dfrm_csi(fl_rxdata_frm);           // 获得版本数据

        if (rc){                                                // 如果没有错误，
                return rc;                                      // 如果［D］
        }
        memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// 复制版本数据
        return rc;                                              // 如果［A］
}
```

8.14 Checksum 命令

8.14.1 处理顺序图

Checksum 命令处理顺序



8.14.2 处理顺序的说明

<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Checksum 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT16}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　　到<6>。
当处理异常结束时：　　异常终止［B］
当超时错误发生时：　　一个超时错误［C］被返回。

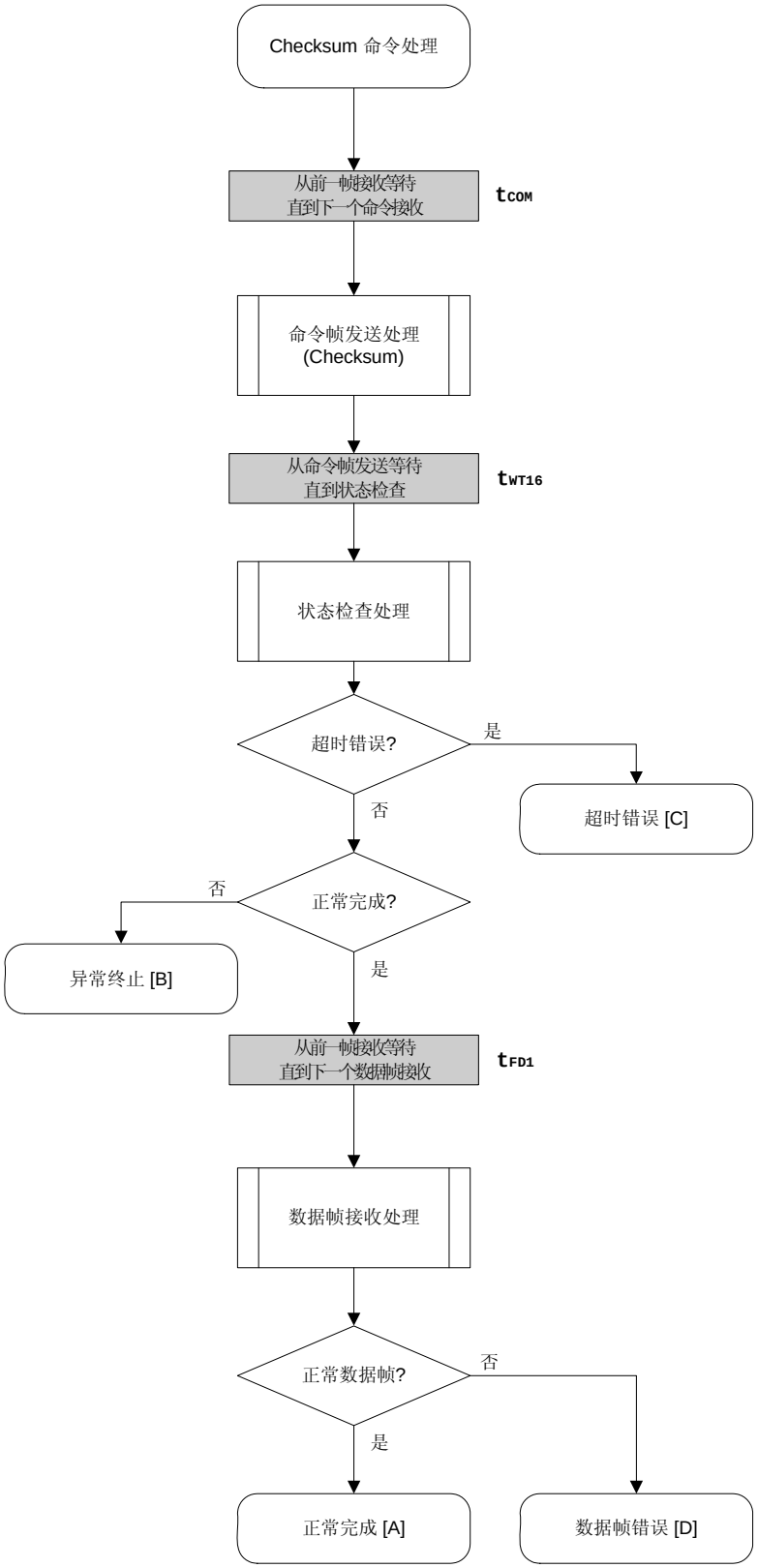
<6>　从前一帧接收等待直到下一个命令发送（等待时间 t_{FD1}）。
<7>　检查接收的数据帧（校验和数据）。

如果数据帧正常：　正常完成［A］
如果数据帧异常：　数据帧错误［D］

8.14.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且校验和数据被正常获取。
异常终止［B］	参数错误	05H	指定的起始/结束地址不是块的起始/结束地址。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	• 在处理过程中，Status 命令以外的命令被接收。 • 命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
数据帧错误［D］		–	作为版本数据接收到的数据帧的校验和不匹配。

8.14.4 流程图


Checksum 命令处理
从前一帧接收等待
直到下一个命令接收
tCOM
命令帧发送处理
(Checksum)
从命令帧发送等待
直到状态检查
tWT16
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从前一帧接收等待
直到下一个数据帧接收
tFD1
数据帧接收处理
正常数据帧?
否
是
正常完成 [A]
数据帧错误 [D]

8.14.5 范例程序

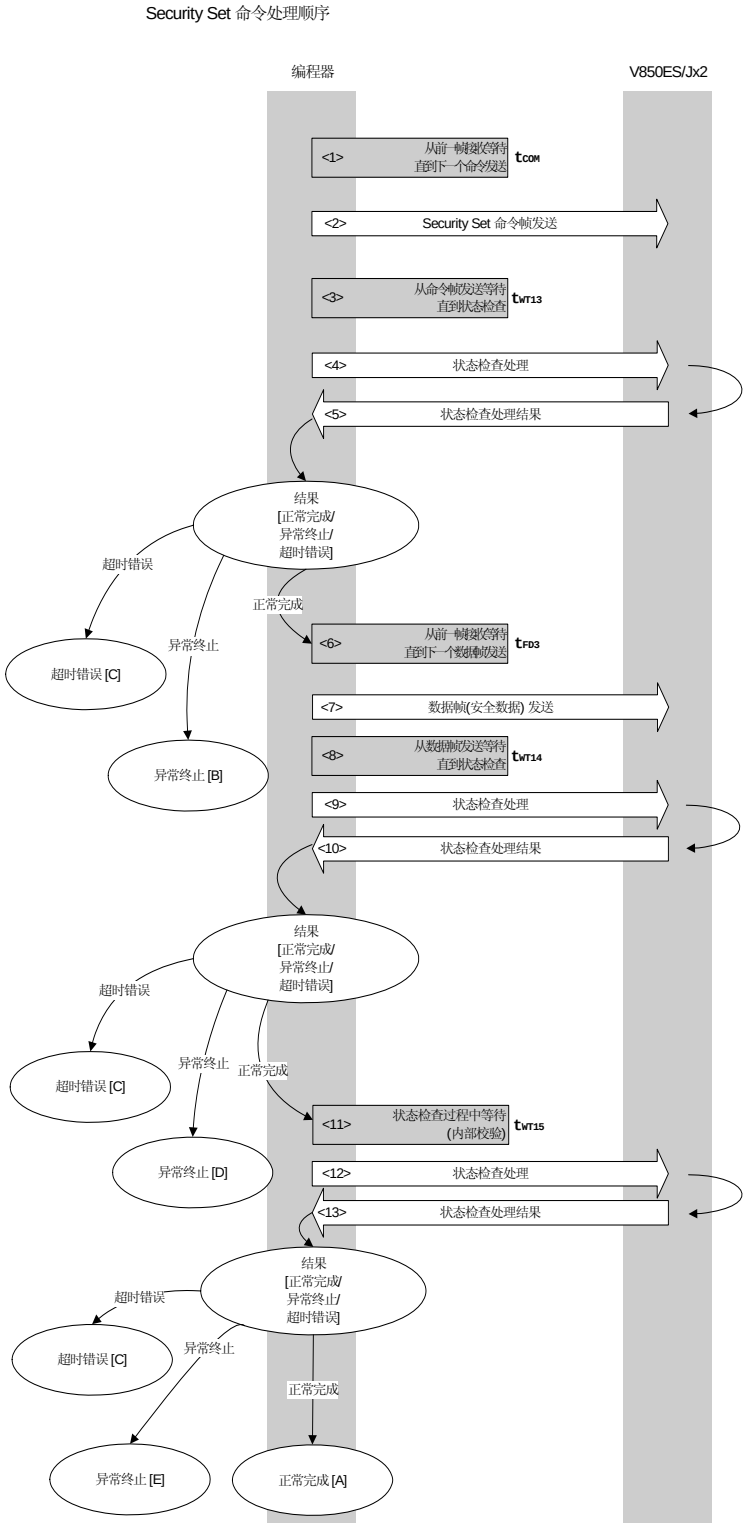
以下表示 Checksum 命令处理的一个范例程序。

```
/*****************************************************************/
/*                                                               */
/* 获取校验和命令（CSI）                                          */
/*                                                               */
/*****************************************************************/
/* [i] u16 *sum    ... 校验和保存区域的指针                       */
/* [i] u32 top     ... 起始地址                                   */
/* [i] u32 bottom  ... 结束地址                                   */
/* [r] u16         ... 错误码                                     */
/*****************************************************************/
u16             fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
        u16     rc;
        u32     fd1;

        /*************************************************/
        /*      设置参数                                 */
        /*************************************************/
        // set params
        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL
        fd1 = get_fd1(bottom - top + 1);

        /*************************************************/
        /*      发送命令                                 */
        /*************************************************/
        fl_wait(tCOM_CSI);                                  // 在发送命令帧前等待

        put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm);   // 发送 "Checksum" 命令

        fl_wait(tWT16);

        rc = fl_csi_getstatus(tWT16_MAX);           // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_DFTO_ERR:return     rc;     break;  // 如果［C］
                default:                return  rc;     break;  // 如果［B］
        }


        /*************************************************/
        /*      获取数据帧（校验和数据）                 */
        /*************************************************/
        fl_wait(fd1);

        rc = get_dfrm_csi(fl_rxdata_frm);     // 获取数据帧（版本数据）
```

```
    if (rc){                                    // 如果没有错误，
            return rc;                          // 如果 ［D］
    }

    *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1];
                                                                    // 设置 SUM 数据
    return rc;                                  // 如果 ［A］
}
```

8.15 Security Set 命令

8.15.1 处理顺序图



8.15.2 处理顺序的说明

<1>　从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2>　通过命令帧发送处理发送 Security Set 命令。
<3>　从命令发送等待直到状态检查处理（等待时间 t_{WT13}（最大））。
<4>　通过状态检查处理，获取状态帧。
<5>　根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　到<6>。
当处理异常结束时：　异常终止［B］
当超时错误发生时：　一个超时错误［C］被返回。

<6>　从前一帧接收等待直到数据帧发送（等待时间 t_{FD3}）。
<7>　通过数据帧发送处理发送数据帧（安全设置数据）命令。
<8>　从数据帧发送等待直到状态检查处理（等待时间 t_{WT14}（最大））。
<9>　通过状态检查处理，获取状态帧。
<10> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　到<11>。
当处理异常结束时：　异常终止［D］
当超时错误发生时：　一个超时错误［C］被返回。

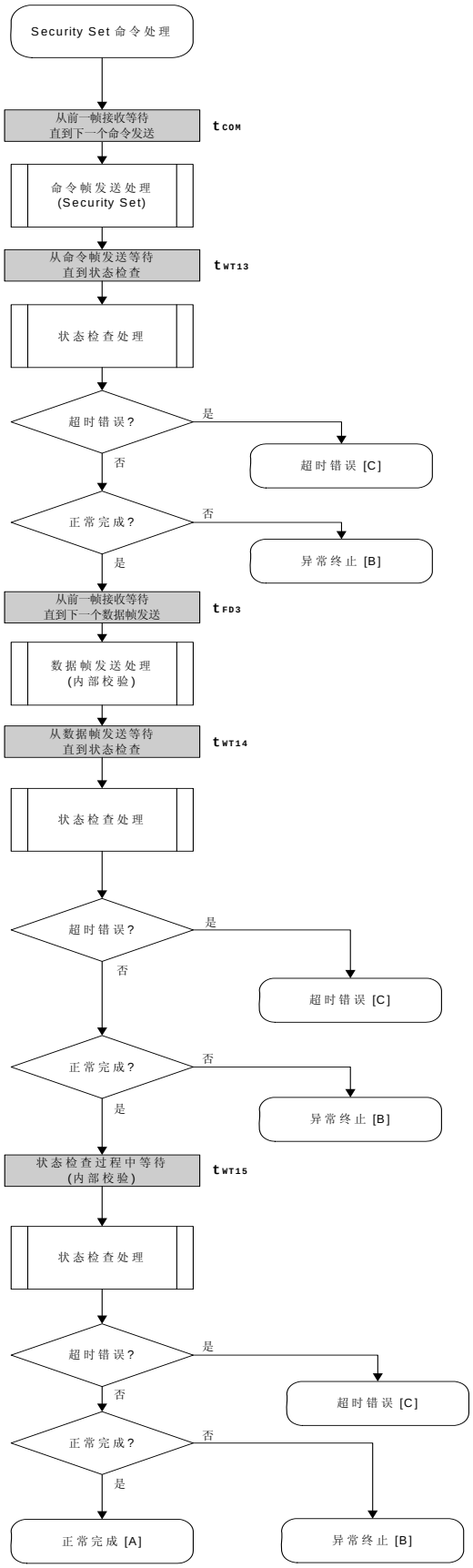
<11> 等待直到状态获取（内部校验完成）（等待时间 t_{WT15}（最大））。
<12> 通过状态检查处理，获取状态帧。
<13> 根据状态检查处理的结果，执行以下处理。

当处理正常结束时：　正常完成［A］
当处理异常结束时：　异常终止［E］
当超时错误发生时：　一个超时错误［C］被返回。

8.15.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成［A］	正常响应（ACK）	06H	命令被正常执行，并且安全设置被正常完成。
异常终止［B］	参数错误	05H	命令信息（参数）不是 00H。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	ID 码不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误［C］		–	状态帧在特定时间内没有被接收。
异常终止［D］	WWV1 错误	08H	• 安全数据已经设置。 • 一个安全数据写入错误发生。
	程序错误	16H	一个程序错误发生。
异常终止［E］	EWV4 错误	11H	一个校验错误发生。
	程序错误	16H	一个程序错误发生。

8.15.4 流程图


Security Set 命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Security Set)
从命令帧发送等待
直到状态检查
tWT13
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从前一帧接收等待
直到下一个数据帧发送
tFD3
数据帧发送处理
(内部校验)
从数据帧发送等待
直到状态检查
tWT14
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
状态检查过程中等待
(内部校验)
tWT15
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
是
正常完成 [A]
异常终止 [B]

8.15.5 范例程序

以下表示 Security Set 命令处理的一个范例程序。

```
/*********************************************************************/
/*                                                                   */
/* 设置安全标志命令（CSI）                                           */
/*                                                                   */
/*********************************************************************/
/* [i] u8 scf     ... 安全标志数据                                   */
/* [r] u16        ... 错误码                                         */
/*********************************************************************/
u16             fl_csi_setscf(u8 scf)
{
        u16     rc;

        /*************************************************/
        /*     设置参数                                  */
        /*************************************************/
        fl_cmd_prm[0] = 0x00;                   // 第一个字节必须为 0x00
        fl_cmd_prm[1] = 0x00;                   // 第二个字节必须为 0x00
        fl_txdata_frm[0] = scf|= 0b11111000;
                                        // "FLG" (高 5 位必须为'1' (要保证))


        /*************************************************/
        /*     发送命令                                  */
        /*************************************************/
        fl_wait(tCOM_CSI);                              // 在发送命令帧前等待

        put_cmd_csi(FL_COM_SET_SECURITY, 3, fl_cmd_prm);  // 发送 "Security Set" 命令

        fl_wait(tWT13);                 // 等待

        rc = fl_csi_getstatus(tWT13_MAX);       // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;     break;  // 如果［C］
                default:                        return  rc;     break;  // 如果［B］
        }

        /*************************************************/
        /*     发送数据帧（安全设置数据）                */
        /*************************************************/
        fl_wait(tFD3);                  // 在获得数据帧前等待

        put_dfrm_csi(1, fl_txdata_frm, true);// 发送数据帧（安全数据）

        fl_wait(tWT14);
```

```
        rc = fl_csi_getstatus(tWT14_MAX);           // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:       return  rc;         break;  // 如果［C］
                default:                    return  rc;         break;  // 如果［B］
        }



        /************************************************/
        /*    内部检查校验                              */
        /************************************************/
        fl_wait(tWT15);

        rc = fl_csi_getstatus(tWT15_MAX);           // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:         return  rc;         break;  // 如果［A］
//              case    FLC_DFTO_ERR:       return  rc;         break;  // 如果［C］
//              default:                    return  rc;         break;  // 如果［B］
//      }
        return rc;
}
```

第 9 章　FLASH存储器编程参数特性

9.1　Flash 存储器编程模式设置时间

<工作时钟（f_X）>

V850ES/Jx2 的内部时钟根据编程器通过 Oscillation Frequency Set 命令指定的指定频率（f_X）的值来更改。

当 2.0 MHz ≤ f_X ≤ 5.0 MHz：　$f_{XX} = f_X \times 4$

当 5.0 MHz < f_X ≤ 10.0 MHz：　$f_{XX} = f_X \times 2$

因此，直到 Oscillation Frequency Set 命令的 t_{WT9} 分配 fx，并且 t_{WT9} 分配 fxx 后，才能得到。

（T_A = –40 到 +85°C, $BV_{DD} \le V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$, $V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V）

参数	符号	最小	典型	最大
VDD↑ 到 FLMD0/FLMD1↑	t_{DP}	1 ms		
FLMD0/FLMD1↑ 到 $\overline{\text{RESET}}$↑	t_{PR}	2 ms		
从 $\overline{\text{RESET}}$↑ 到 FLMD0 注 1 的计数起始时间	t_{RP}	76,046/f_X		
从 $\overline{\text{RESET}}$↑ 到 FLMD0 注 1 的计数结束时间	t_{RPE}			212,148/f_X
FLMD0 计数器高电平 / 低电平宽度	t_{PW}	10　s		100　s
等待读取命令（CSI/CSI + HS）	t_{RC}	231,630/f_X		3 s
等待低电平数据 1 （UART）	t_{R1}	231,630/f_X		3 s
等待低电平数据 2 （UART）	t_{2C}	30,000/f_X		3 s
等待读取命令（UART）	t_{12}	30,000/f_X		3 s
低电平数据 1/2 注 2 的宽度	t_{L1}, t_{L2}		**注 2**	
FLMD0 计数器上升 / 下降时间	–			1　s

注　**1.**　（76,046/f_X + 212,148/f_X）/2 被建议作为 FLMD0 脉冲输入时序的标准值。

2.　低电平宽度与 9,600 bps 下的 00H 数据宽度一样。

9.2 编程特性

等待	条件	符号	串行 I/F	最小	最大
数据帧发送 / 接收之间	数据帧接收	t_{DR}	CSI	125/f_X	3 s
			UART	125/f_X	3 s
	数据帧发送	t_{DT}	CSI	127/f_X	3 s
			UART	0	3 s
从 Status 命令接收直到状态帧发送	–	t_{SF}	CSI	**注 1**	
从状态帧发送直到数据帧发送 （1）	–	t_{FD1} 注 2	CSI	55,920/f_X + 33,802/f_X × N	3 s
			UART	0	3 s
从状态帧发送直到数据帧发送 （2）	–	t_{FD2}	CSI	2,998/f_X	3 s
			UART	0 注 2	3 s
从状态帧发送直到数据帧接收 （2）	–	t_{FD3}	CSI	168/f_X	3 s
			UART	168/f_X	3 s
从状态帧发送直到命令帧接收	–	t_{COM}	CSI	154/f_X	3 s
			UART	120/f_X	3 s

注 1. 对于各自命令的 t_{SF} 最小值和最大值

命令	最小	最大
Reset, Oscillating Frequency Set, Checksum, Silicon Signature, Version Get	241/f_X	3 s
Chip Erase	399/f_X	3 s
Block Erase	428/f_X	3 s
Programming	59,520/f_X	3 s
Verify	4,656/f_X	3 s
Block Blank Check	428/f_X	3 s
Security Set	826/f_X	3 s

2. 对于编程器，当使能连续接收时

备注 **1.** N：命令执行块大小（KB）

2. 等待定义如下。

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

V850ES/Jx2 在前一个通信完成后最小时间过去后准备好下一个通信。

在前一个通信完成后，编程器必须在最小和最大时间之间发送下一个数据。

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

V850ES/Jx2 在前一个通信完成后最小时间过去后准备好下一个通信。

在前一个通信完成后，编程器必须在最小和最大时间之间发送下一个数据。

命令	符号	串行 I/F	最小	最大
Reset	t_{WT0}	CSI	199/f_X	3 s
		UART	**注**	3 s
Chip Erase	t_{WT1}	–	**[PD70F3718, 70F3719, 70F3723, 70F3724]** 159,944,112/f_{CX} + 4083.2 ms	**[PD70F3718, 70F3719, 70F3723, 70F3724]** 4,339,025,024/f_{CX} + 24,906/f_X + 175,918.4 ms
			[PD70F3715, 70F3716, 70F3717, 70F3720, 70F3721, 70F3722] 96,266,820/f_{CX} + 2462.4 ms	**[PD70F3715, 70F3716, 70F3717, 70F3720, 70F3721, 70F3722]** 2,288,923,488/f_{CX} + 22,018/f_X + 106,230 ms
Block Erase	t_{WT2}	–	（5.2 ms + 125,147/f_{CX}） + （6.25 ms + 246,784/f_{CX}） × N	25,570/f_X + (282.9 ms + 1,836,104/f_{CX}) + （267.8 ms + 3,619,584/f_{CX}） × N
Programming	t_{WT3}	CSI	58,872/f_X	3 s
		UART	**注**	3 s
	t_{WT4}	–	971.3 s + 29,818/f_{CX}	49.5 ms + 367,079/f_{CX}
	t_{WT5}	CSI	98,400/f_{CX} × N	123,000/f_{CX} × N
		UART	**注**	123,000/f_{CX} × N
Verify	t_{WT6}	CSI	407/f_{CX}	3 s
		UART	**注**	3 s
	t_{WT7}	CSI	28,128/f_X	3 s
		UART	**注**	3 s
Block Blank Check	t_{WT8}	–	44,904/f_{CX} × N	(811,400 + 2,777,554 × N)/f_{CX} + 21,611/f_X
Oscillating Frequency Set	t_{WT9}	CSI	6,199/f_X + 2^{13}/f_X	3 s
		UART	**注**	3 s
Baud Rate Set	t_{WT10}	UART	4,488/f_X	3 s
Silicon Signature	t_{WT11}	CSI	557/f_X	3 s
		UART	**注**	3 s
Version Get	t_{WT12}	CSI	570/f_X	3 s
		UART	**注**	3 s
Security Set	t_{WT13}	CSI	461/f_X	3 s
		UART	**注**	3 s
	t_{WT14}	–	60,990/f_X + 364.3 s + 11,295/f_{CX}	62,568/f_X + 49.5 ms + 367,079/f_{CX}
	t_{WT15}	CSI	8,284,080/f_{CX}	103,551,672/f_{CX}
		UART	**注**	103,551,672/f_{CX}
Checksum	t_{WT16}	CSI	691/f_X	3 s
		UART	**注**	3 s

注 在命令发送前，编程器的接收必须被使能。

备注 **1.** N：命令执行块大小 （K 字节）

f_{CX}：f_X：

2. 等待被定义如下。

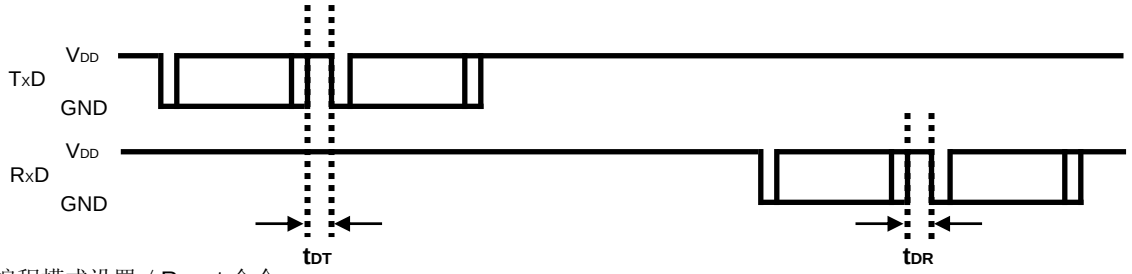
<t_{WT0} 到 t_{WT16}>

V850ES/Jx2 在最小到最大时间内完成命令处理。

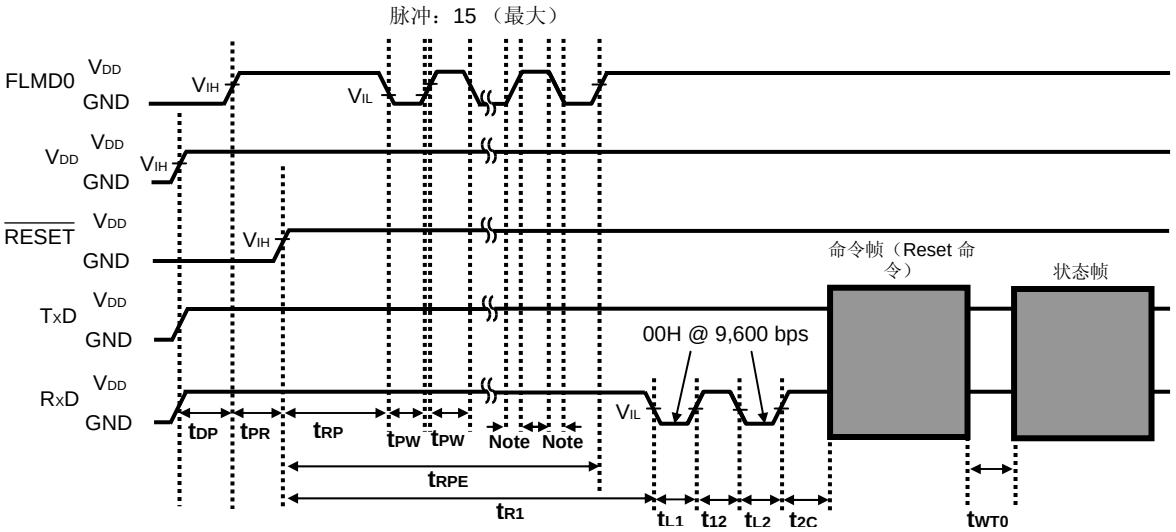
编程器必须重复状态检查，直到最大时间过去。

9.3 UART 通信模式

- 数据帧

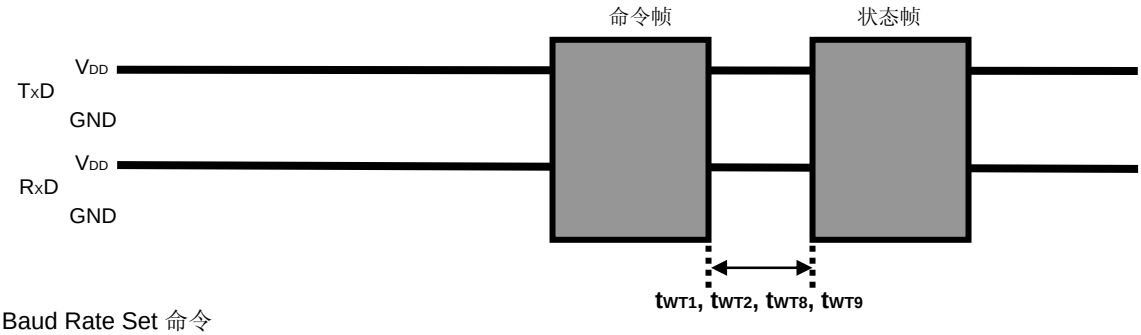


- 编程模式设置 / Reset 命令

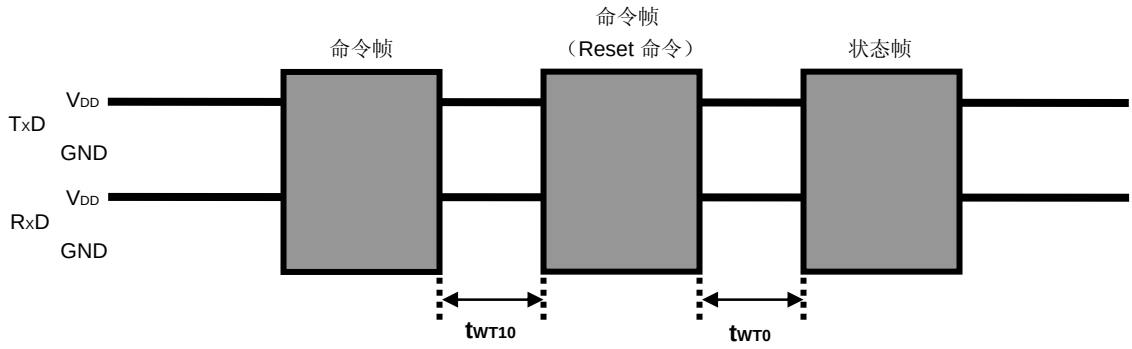


注 FLMD0 计数器上升 / 下降时间

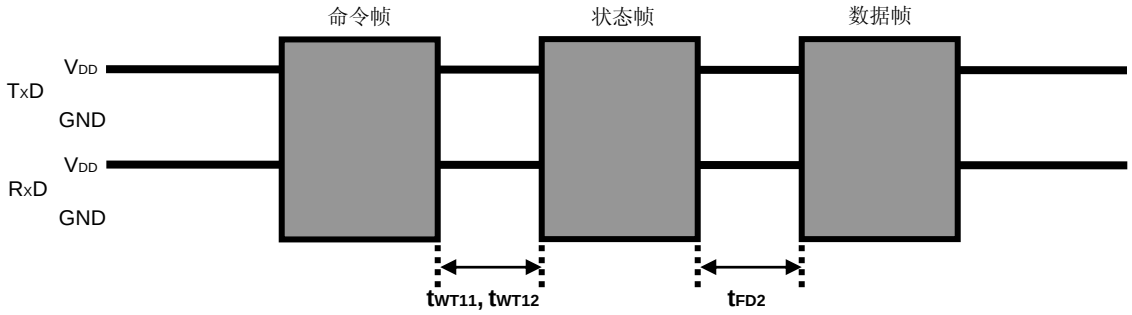
- Chip Erase 命令/Block Erase 命令/Block Blank Check 命令/Oscillating Frequency Set 命令



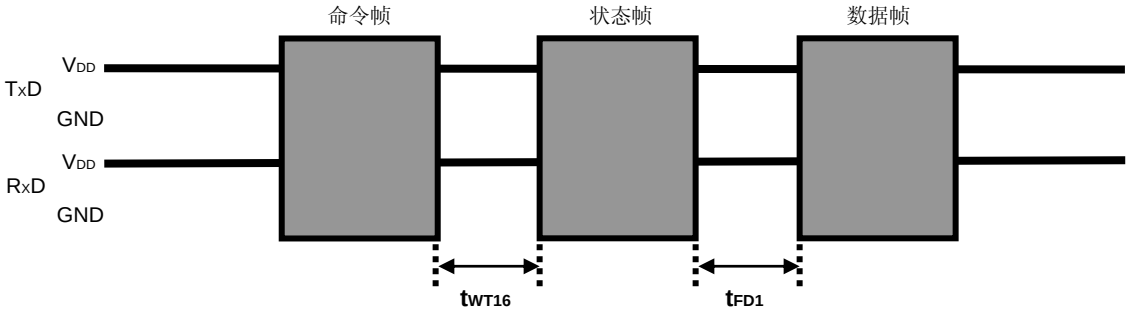
- Baud Rate Set 命令



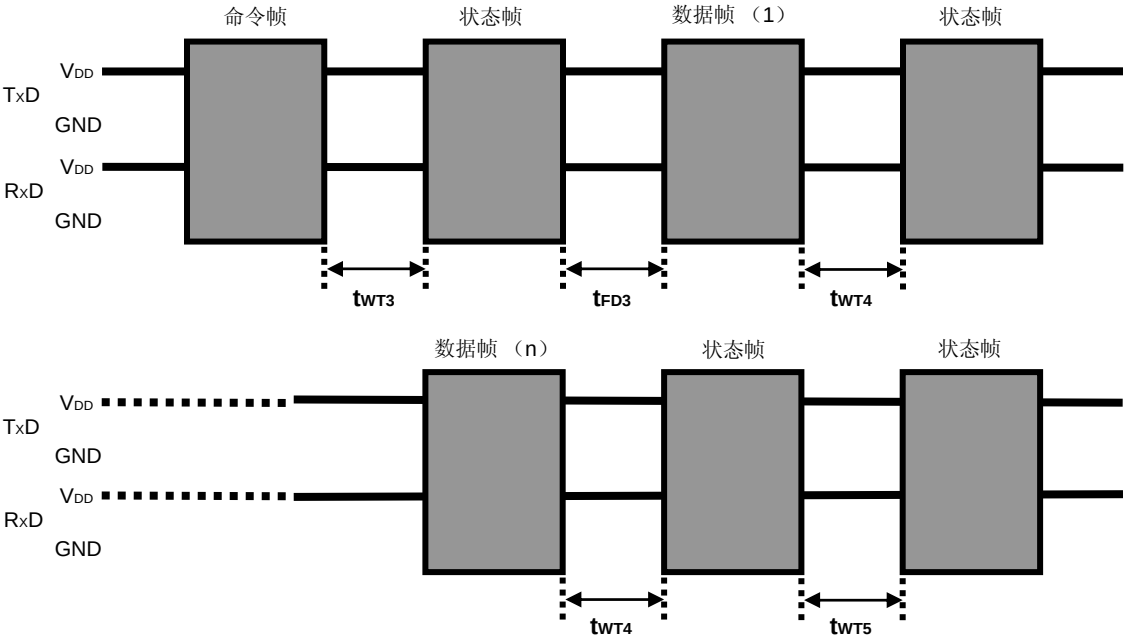
• Silicon Signature 命令 / Version Get 命令



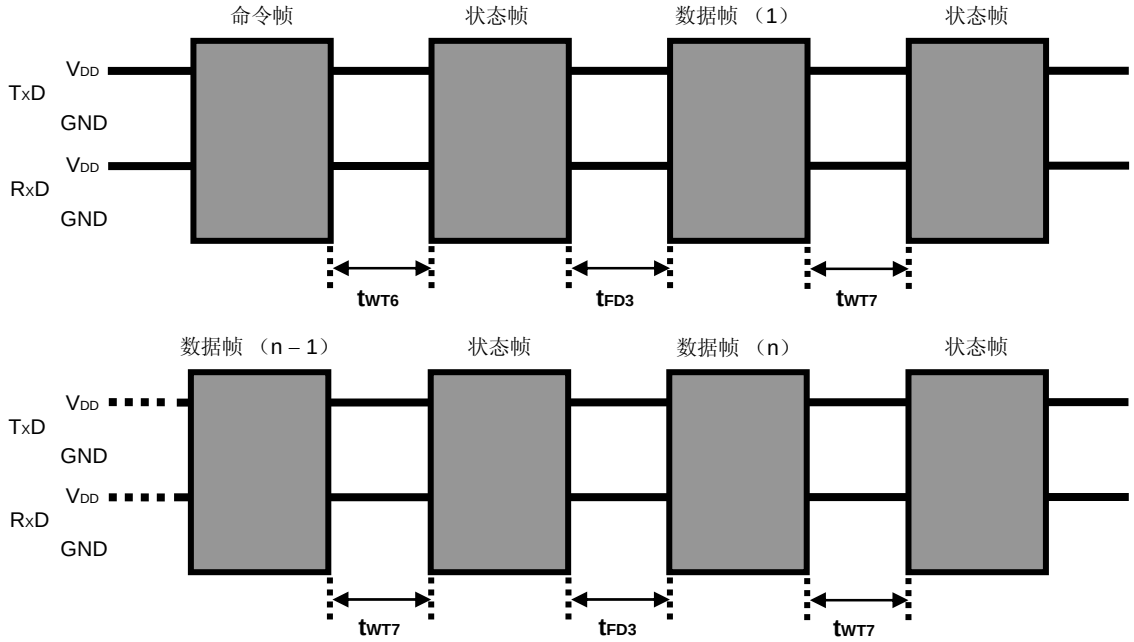
• Checksum 命令



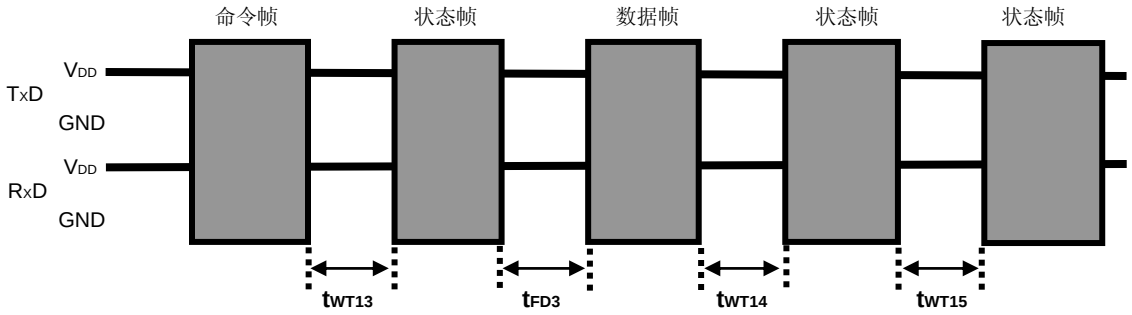
• Programming 命令



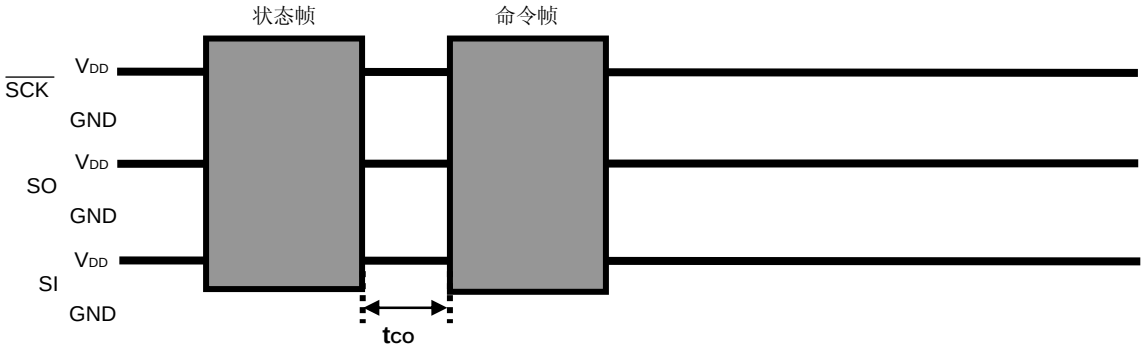
• Verify 命令



• Security Set 命令

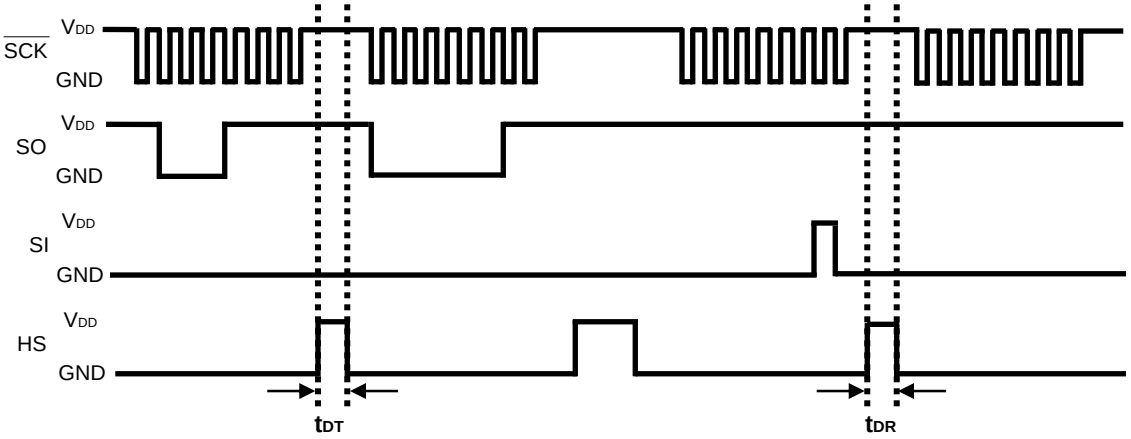


• 命令帧发送前的等待

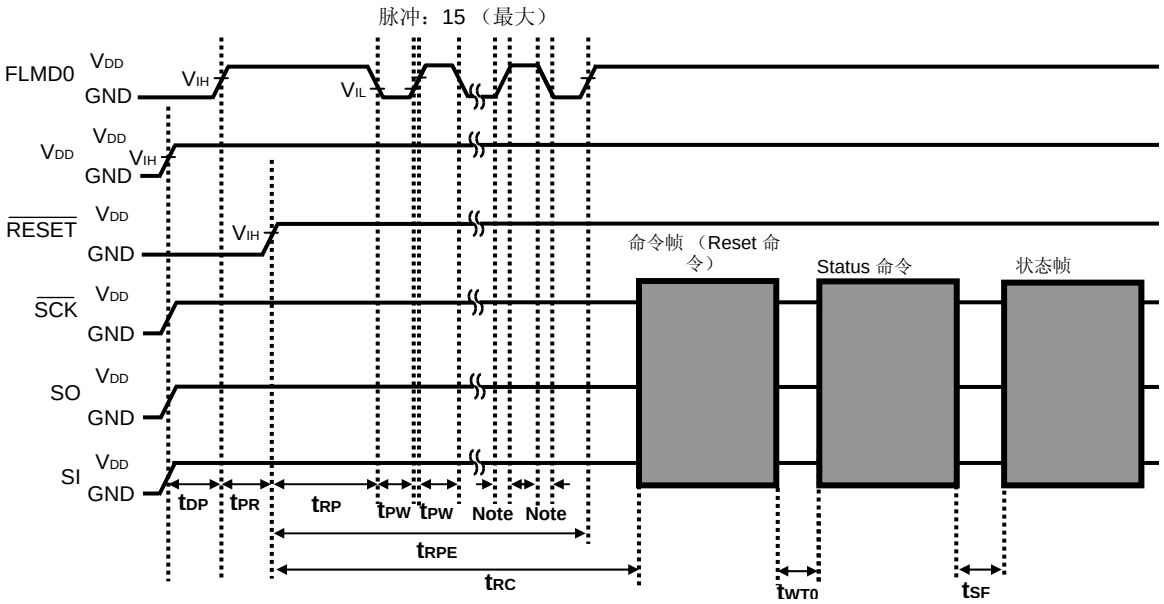


9.4 3 线串行输入 / 输出通信模式

• 数据帧

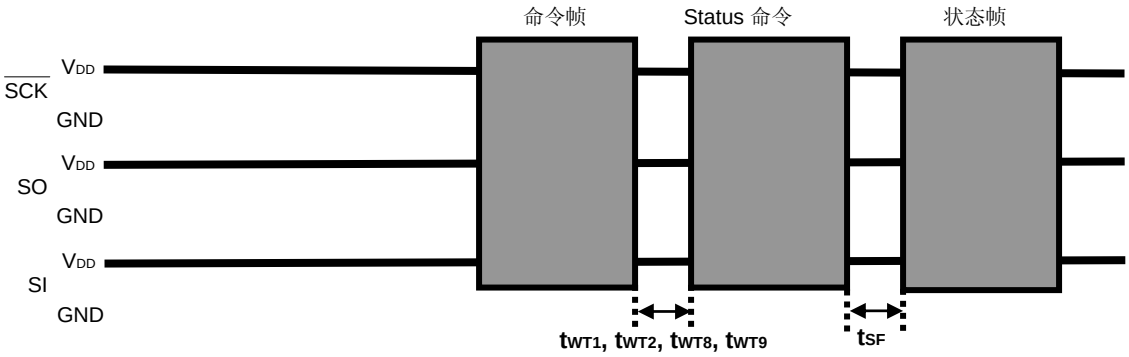


• 编程模式设置 / Reset 命令

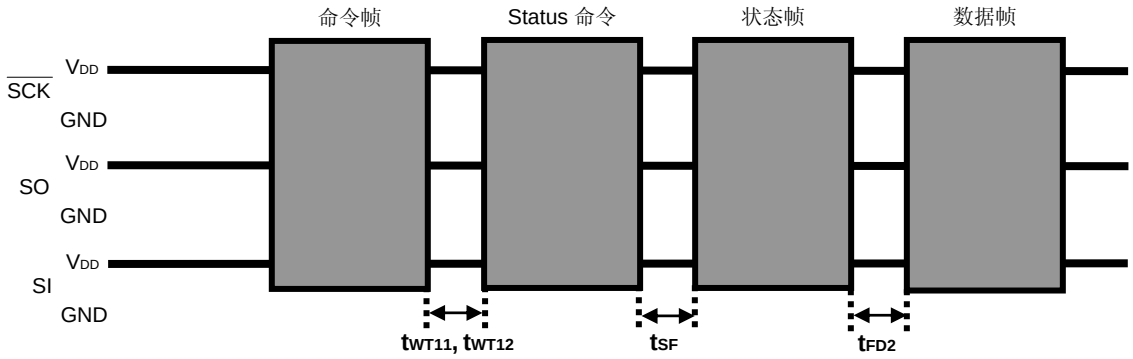


注　　FLMD0 计数器上升 / 下降时间

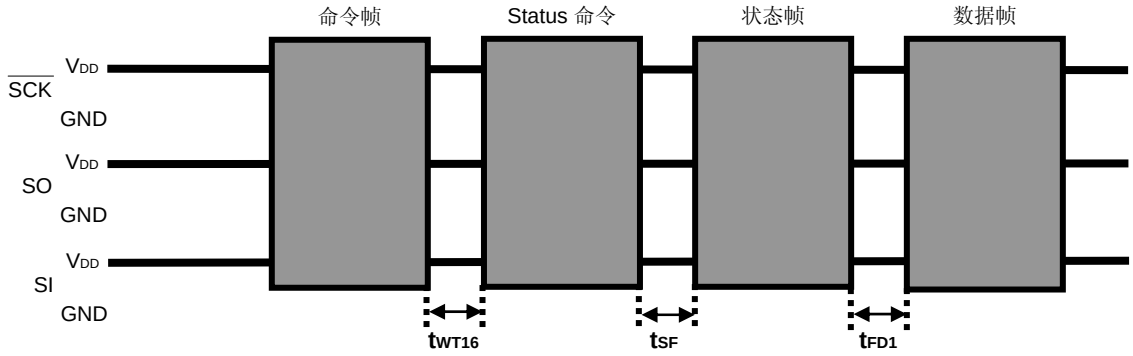
• Chip Erase 命令 / Block Erase 命令 / Block Blank Check 命令 / Oscillating Frequency Set 命令



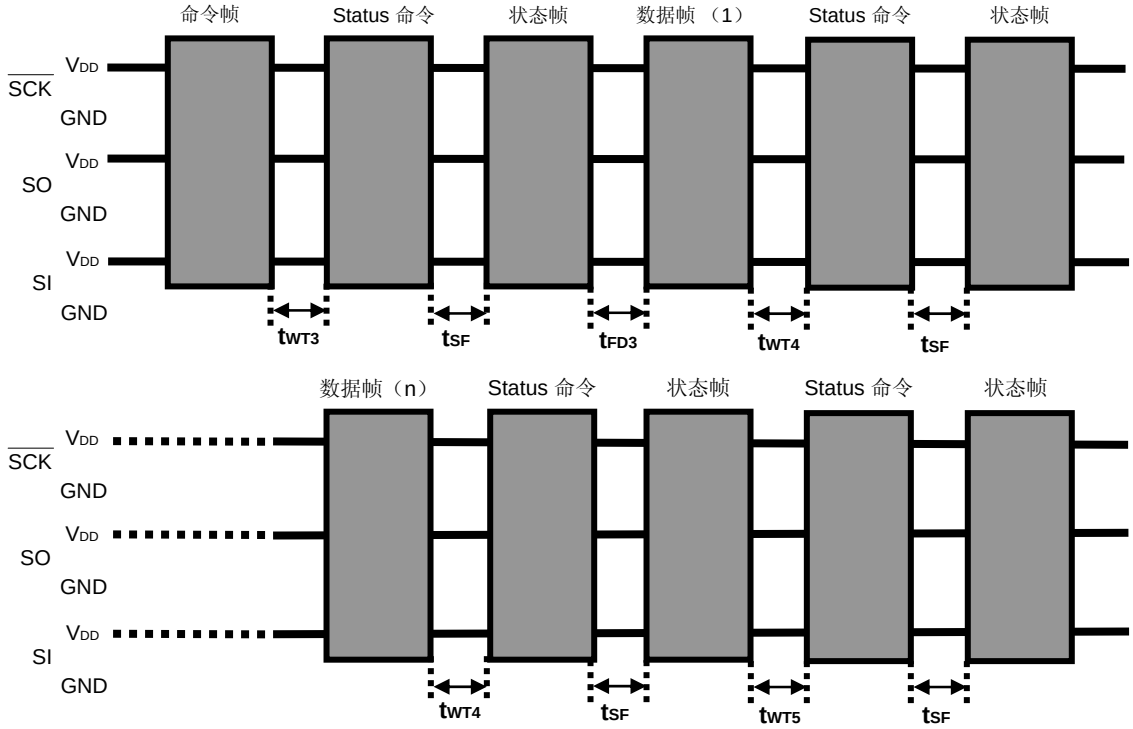
- Silicon Signature 命令 / Version Get 命令



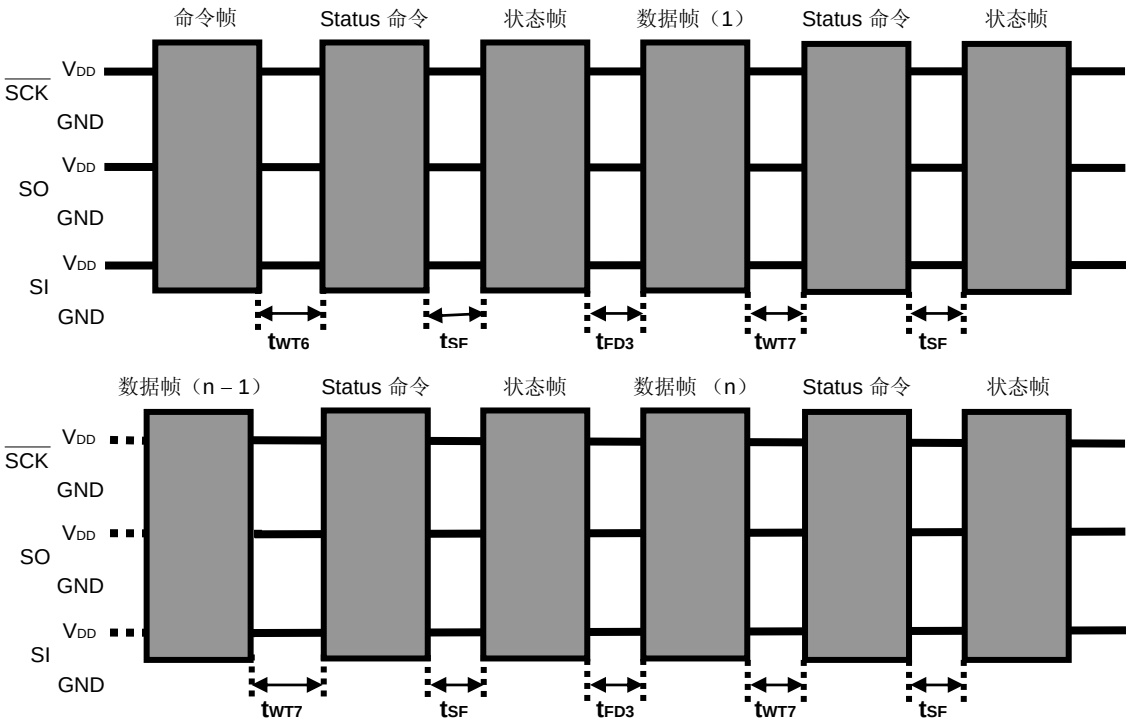
- Checksum 命令



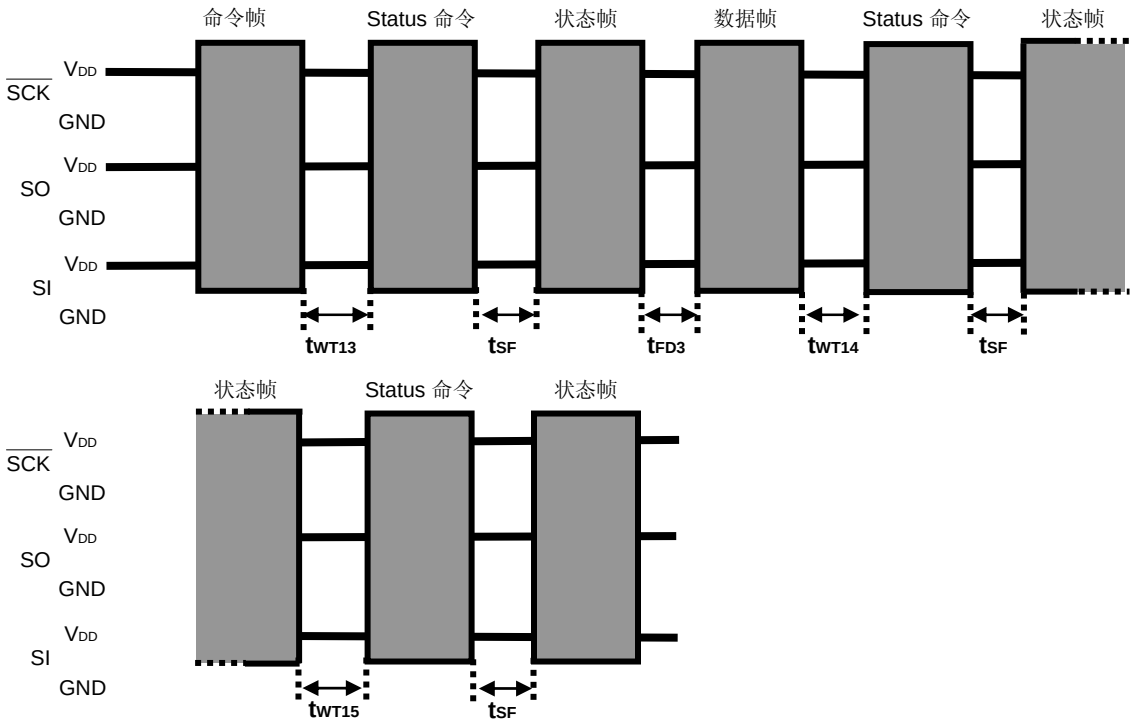
- Programming 命令



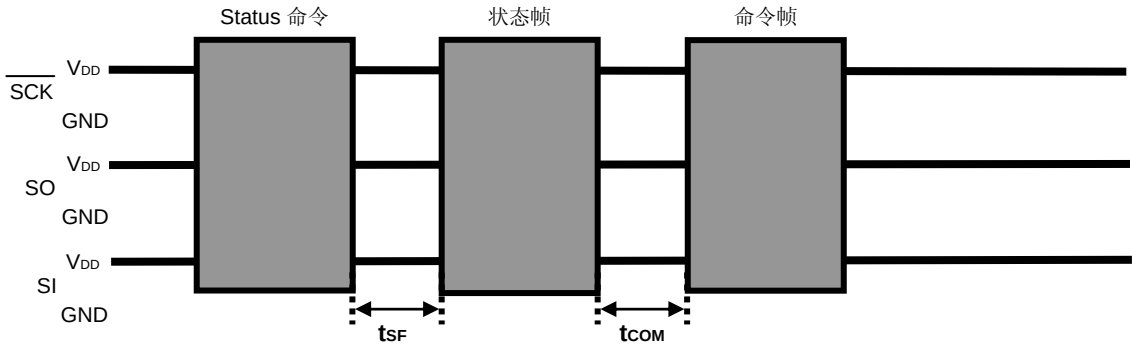
• Verify 命令



• Security Set 命令



- 命令帧发送前的等待



第 10 章　电气规范（参考）

以下规范只能作为您的参考。
当设计一个编程器时，对于电气规范，确认参阅每个产品的最新用户手册。

Flash 存储器编程特性

（T_A = –40 到 +85°C，$BV_{DD} \leq V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V）

（1） V850ES/JG2

项目	返回	条件			最小	典型	最大	单位
供电电流[注1]	I_{DD}	Flash 存储器编程模式	f_{XX} = 20 MHz	**注 2**		35	54	mA
			（f_X = 5 MHz）	**注 3**		33	51	mA

注　**1.** V_{DD}、EV_{DD} 和 BV_{DD} 电流总和。流过输出缓冲、模 / 数转换器、数 / 模转换器和片上下拉电阻的电流不包括在内。
2. μPD70F3718，70F3719
3. μPD70F3715，70F3716，70F3717

备注　f_{XX}：主时钟频率 f_X：主时钟振荡器频率

（2） V850ES/JJ2

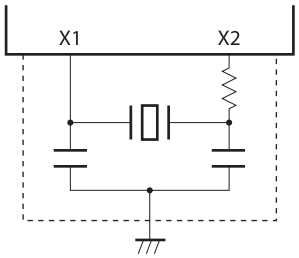
项目	返回	条件			最小	典型	最大	单位
供电电流[注1]	I_{DD}	Flash 存储器编程模式	f_{XX} = 20 MHz	**注 2**		38	61	mA
			（f_X = 5 MHz）	**注 3**		37	60	mA

注　**1.** V_{DD}、EV_{DD} 和 BV_{DD} 电流总和。流过输出缓冲、模 / 数转换器、数 / 模转换器和片上下拉电阻的电流不包括在内。
2. μPD70F3723，70F3724
3. μPD70F3720，70F3721，70F3722

备注　f_{XX}：主时钟频率 f_X：主时钟振荡器频率

主时钟振荡器特性

（T_A = –40 到 +85°C，$BV_{DD} \le V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V）

振荡器	电路举例	参数	条件	最小	典型	最大	单位
陶瓷振荡器 / 晶体振荡器		振荡频率（f_X）注 1		2.5		10	MHz
		振荡稳定时间注 2	复位释放后		$2^{16}/f_X$		s
			STOP 模式释放后	1 注 4	**注 3**		ms
			IDLE2 模式释放后	350 注 4	**注 3**		μs

注　**1.** 上面表示的振荡频率只表明振荡器特性。使用 V850ES/Jx2 来保证内部工作条件没有超过 **AC 特性**和 **DC 特性**中的额定值。

2. 从振荡启动直到振荡器稳定需要的时间。

3. 值根据 OSTS 寄存器的设置而改变。

4. 建立 flash 存储器需要的时间。使用 OSTS 寄存器来保护建立时间。

注意事项　1. 当使用主时钟振荡器时，对于上面图中的虚线包围的区域，按照以下走线来避免走线电容的不利影响。

- **保持走线长度尽量短。**
- **不要与其它信号线交叉走线。**
- **不要在通过高速变化电流的信号附近布线。**
- **总是使振荡器电容器的接地点与 V_{SS} 具有相同的电平。**
- **不要将电容器接地到大电流流动的地上。**
- **不要从振荡器取出信号。**

2. 当主时钟停止并且设备以子时钟工作时，在切换回主时钟前等待直到程序保护的振荡稳定时间过去。

3. 关于振荡器的选择和振荡器的常数，客户需要自己评估振荡或者向振荡器厂商申请评估。

DC 特性 （1/2）

（T_A = –40 到 +85°C，$BV_{DD} \leq V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V）

（1） V850ES/JG2

参数	符号	条件		最小	典型	最大	单位
输入电压，高	V_{IH1}	$\overline{RESET}$，FLMD0		0.8 EV_{DD}		EV_{DD}	V
	V_{IH2}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$		0.8 EV_{DD}		5.5	V
	V_{IH3}	SIB0/SDA01，SOB0/SCL01		0.7 EV_{DD}		5.5	V
	V_{IH4}	$\overline{WAIT}$，FLMD1		0.7 BV_{DD}		BV_{DD}	V
输入电压，低	V_{IL1}	$\overline{RESET}$，FLMD0		EV_{SS}		0.2 EV_{DD}	V
	V_{IL2}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$		EV_{SS}		0.2 EV_{DD}	V
	V_{IL3}	SIB0/SDA01，SOB0/SCL01		EV_{SS}		0.3 EV_{DD}	V
	V_{IL4}	$\overline{WAIT}$，FLMD1		BV_{SS}		0.3 BV_{DD}	V
输入漏电流，高	I_{LIH}	$V_I = V_{DD} = EV_{DD} = BV_{DD} = AV_{REF0} = AV_{REF1}$				5	μA
输入漏电流，低	I_{LIL}	V_I = 0 V				–5	μA
输出漏电流，高	I_{LOH}	$V_O = V_{DD} = EV_{DD} = BV_{DD} = AV_{REF0} = AV_{REF1}$				5	μA
输出漏电流，低	I_{LOL}	V_O = 0 V				–5	μA
输出电压，高	V_{OH1}	TXDA0/SOB4，RXDA0/SIB4，SIB0/SDA01，SOB0/SCL01，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$	每个管脚 I_{OH} = –1.0 mA	EV_{DD} – 1.0		EV_{DD}	V
			每个管脚 I_{OH} = –100 μA	EV_{DD} – 0.5		EV_{DD}	V
	V_{OH2}	$\overline{WAIT}$，FLMD1	每个管脚 I_{OH} = –1.0 mA	BV_{DD} – 1.0		BV_{DD}	V
			每个管脚 I_{OH} = –100 μA	BV_{DD} – 0.5		BV_{DD}	V
输出电压，低	V_{OL1}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$	每个管脚 I_{OL} = 1.0 mA	0		0.4	V
	V_{OL2}	SIB0/SDA01，SOB0/SCL01	每个管脚 I_{OL} = 3.0 mA	0		0.4	V
	V_{OL3}	$\overline{WAIT}$，FLMD1	每个管脚 I_{OL} = 1.0 mA	0		0.4	V

备注　**1.** 替换功能管脚的特性与端口管脚的特性相同。

2. 当某个管脚的 I_{OH} 和 I_{OL} 条件不满足而所有管脚的条件满足时，只有那个管脚不满足 DC 特性。

DC 特性 （2/2）

（T_A = –40 到 +85°C，$BV_{DD} \le V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V）

（2） V850ES/JJ2

参数	符号	条件		最小	典型	最大	单位
输入电压，高	V_{IH1}	$\overline{RESET}$，FLMD0		0.8 EV_{DD}		EV_{DD}	V
	V_{IH2}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$		0.8 EV_{DD}		5.5	V
	V_{IH3}	SIB0/SDA01，SOB0/SCL01		0.7 EV_{DD}		5.5	V
	V_{IH4}	$\overline{WAIT}$，FLMD1		0.7 BV_{DD}		BV_{DD}	V
输入电压，低	V_{IL1}	$\overline{RESET}$，FLMD0		EV_{SS}		0.2 EV_{DD}	V
	V_{IL2}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$		EV_{SS}		0.2 EV_{DD}	V
	V_{IL3}	SIB0/SDA01，SOB0/SCL01		EV_{SS}		0.3 EV_{DD}	V
	V_{IL4}	$\overline{WAIT}$，FLMD1		BV_{SS}		0.3 BV_{DD}	V
输入漏电流，高	I_{LIH}	$V_I = V_{DD} = EV_{DD} = BV_{DD} = AV_{REF0} = AV_{REF1}$				5	μA
输入漏电流，低	I_{LIL}	V_I = 0 V				–5	μA
输出漏电流，高	I_{LOH}	$V_O = V_{DD} = EV_{DD} = BV_{DD} = AV_{REF0} = AV_{REF1}$				5	μA
输出漏电流，低	I_{LOL}	V_O = 0 V				–5	μA
输出电压，高	V_{OH1}	TXDA0/SOB4，RXDA0/SIB4，SIB0/SDA01，SOB0/SCL01，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$	每个管脚 I_{OH} = –1.0 mA	EV_{DD} – 1.0		EV_{DD}	V
			每个管脚 I_{OH} = –100 μA	EV_{DD} – 0.5		EV_{DD}	V
	V_{OH2}	$\overline{WAIT}$，FLMD1	每个管脚 I_{OH} = –1.0 mA	BV_{DD} – 1.0		BV_{DD}	V
			每个管脚 I_{OH} = –100 μA	BV_{DD} – 0.5		BV_{DD}	V
输出电压，低	V_{OL1}	TXDA0/SOB4，RXDA0/SIB4，$\overline{SCKB0}$，SIB3，SOB3，$\overline{SCKB3}$	每个管脚 I_{OL} = 1.0 mA	0		0.4	V
	V_{OL2}	SIB0/SDA01，SOB0/SCL01	每个管脚 I_{OL} = 3.0 mA	0		0.4	V
	V_{OL3}	$\overline{WAIT}$，FLMD1	每个管脚 I_{OL} = 1.0 mA	0		0.4	V

备注　**1.** 替换功能管脚的特性与端口管脚的特性相同。

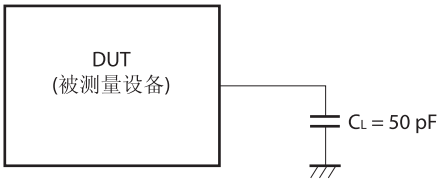
2. 当某个管脚的 I_{OH} 和 I_{OL} 条件不满足而所有管脚的条件满足时，只有那个管脚不满足 DC 特性。

AC 特性

AC 测试输入测量点 （V_{DD}，AV_{REF0}，AV_{REF1}，EV_{DD}，BV_{DD}）

AC 测试输出测量点

负载条件



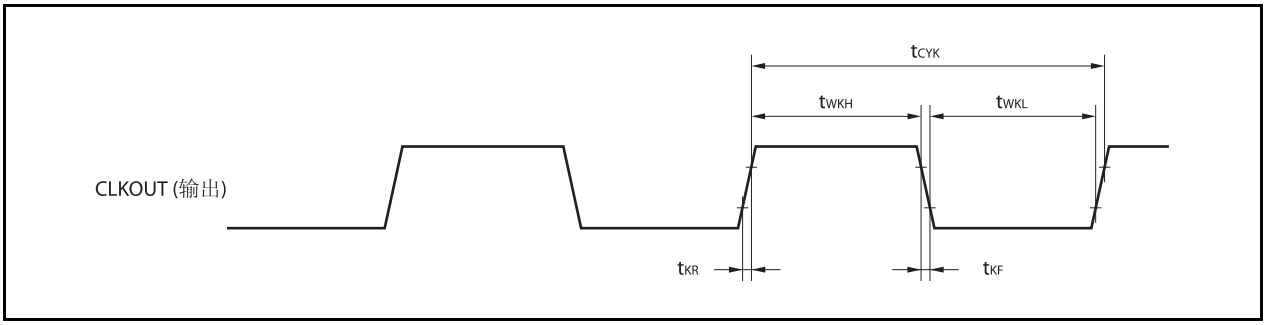
注意事项　如果由于电路配置负载电容超过 50 pF，通过初入一个缓冲或其它方式使器件的负载电容为 50 pF 或更低。

CLKOUT 输出时序

（T_A = –40 到 +85°C，$BV_{DD} \leq V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V，C_L = 50 pF）

参数	符号	条件	最小	最大	单位
输出周期	t_{CYK}		50 ns	31.25 μs	
高电平宽度	t_{WKH}		$t_{CYK}/2 - 10$		ns
低电平宽度	t_{WKL}		$t_{CYK}/2 - 10$		ns
上升时间	t_{KR}			10	ns
下降时间	t_{KF}			10	ns

时钟时序

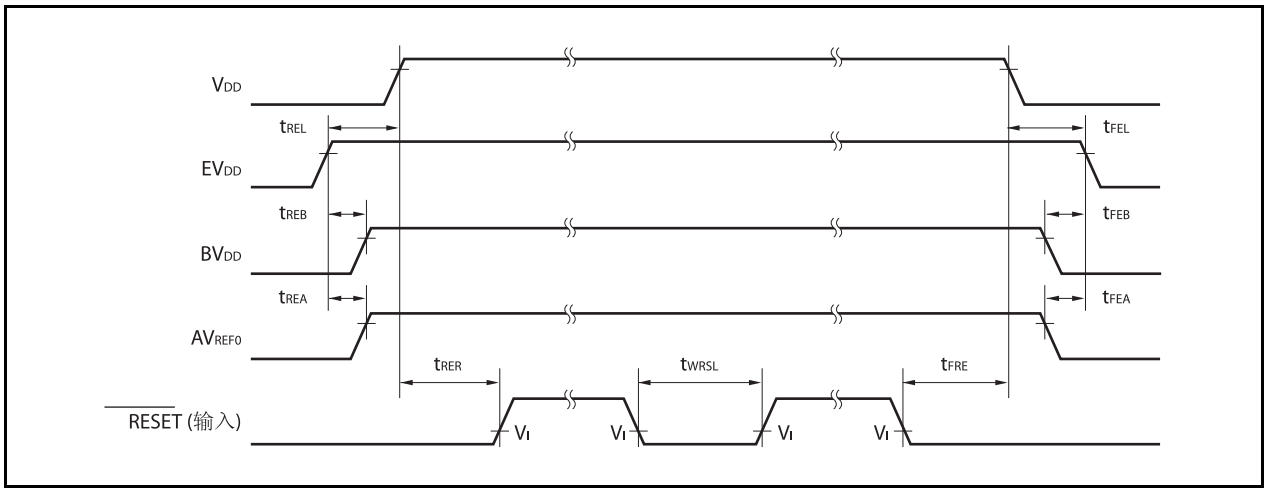


（1） 基本操作

（T_A = –40 到 +85°C，V_{SS} = AV_{SS} = BV_{SS} = EV_{SS} = 0 V，C_L = 50 pF）

参数	符号	条件	最小	最大	单位
EV_{DD}↑ → V_{DD}↑	t_{REL}		0		ns
EV_{DD}↑ → BV_{DD}↑	t_{REB}		0	t_{REL}	ns
EV_{DD}↑ → AV_{REF0}，AV_{REF1}↑	t_{REA}		0	t_{REL}	ns
EV_{DD}↑ → $\overline{RESET}$↑	t_{RER}		500 + t_{REG}注		ns
$\overline{RESET}$ 低电平宽度	t_{WRSL}	模拟噪声消除（flash擦除 / 写入过程中）	500		ns
		模拟噪声消除	500		ns
$\overline{RESET}$↓，V_{DD}↓	t_{FRE}		500		ns
V_{DD}↓　EV_{DD}↓	t_{FEL}		0		ns
BV_{DD}↓　EV_{DD}↓	t_{FEB}		0	t_{FEL}	ns
AV_{REF0}↓　EV_{DD}↓	t_{FEA}		0	t_{FEL}	ns

注　依赖于片上稳压器特性。



（2） 串行接口

UART 时序

（T_A = –40 到 +85°C，$BV_{DD} \leq V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V，C_L = 50 pF）

参数	符号	条件	最小	最大	单位
发送速率				312.5	kbps
ASCK0 周期时间				10	MHz

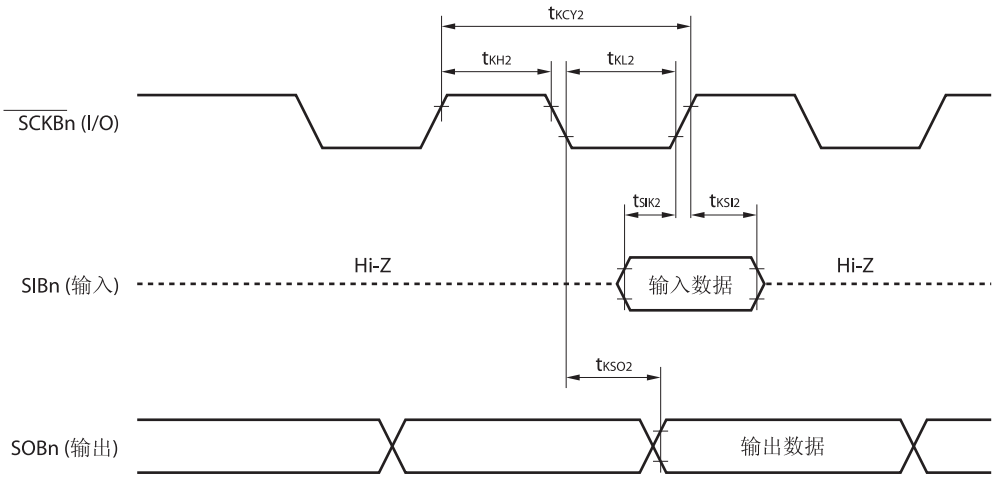
CSIB 时序

从模式

（T_A = –40 到 +85°C，$BV_{DD} \leq V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$，$V_{SS} = EV_{SS} = BV_{SS} = AV_{SS}$ = 0 V，C_L = 50 pF）

参数	符号	条件	最小	最大	单位
$\overline{SCKBn}$ 周期时间	t_{KCY2}		125		ns
$\overline{SCKBn}$ 高 / 低电平宽度	t_{KH2}，t_{KL2}		57.5		ns
SIBn 建立时间 （到 $\overline{SCKBn}$↑）	t_{SIK2}		30		ns
SIBn 保持时间 （从 $\overline{SCKBn}$↑）	t_{KSI2}		30		ns
从$\overline{SCKBn}$↓ 到 SOBn 输出的延时	t_{KSO2}			30	ns

备注　　n = 0 到 4 （V850ES/JG2），n = 0 到 5 （V850ES/JJ2）



备注　　n = 0 到 4 （V850ES/JG2），n = 0 到 5 （V850ES/JJ2）

[备忘录]

附录 A　电路图（参考）

图 A-1 和 A-2 表示编程器和 V850ES/Jx2 的电路图，供参考。

图 A-1. 编程器和 V850ES/Jx2 的参考电路图（母板）

V850ES/Jx2 Flash 编程器应用举例 - 母板

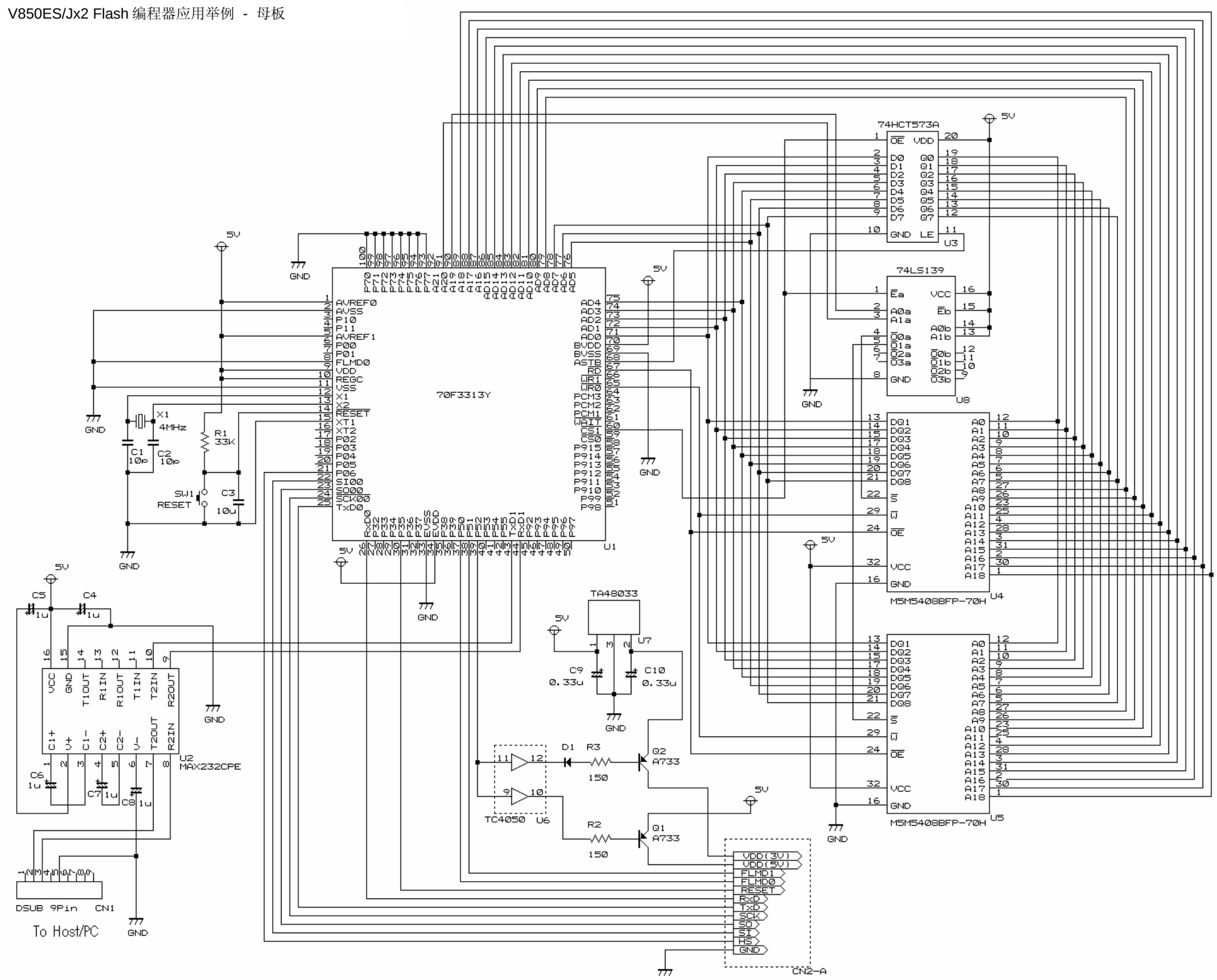
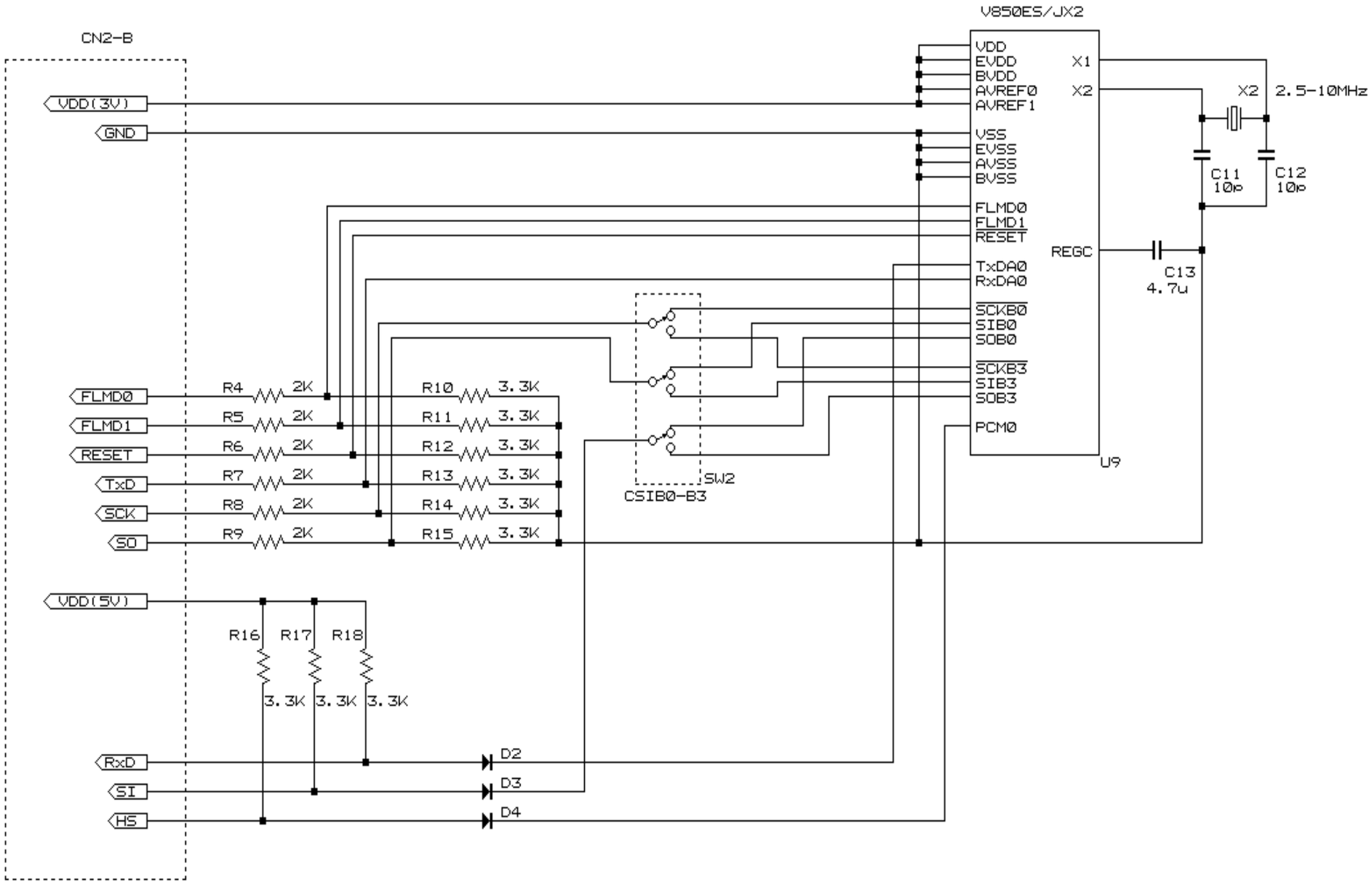


图 A-2. 编程器和 V850ES/Jx2 的参考电路图（目标板）

V850ES/Jx2 Flash 编程器应用举例 - 目标板



详细信息请联系：

中国区

MCU 技术支持热线：
电话：+86-400-700-0606 (普通话)
服务时间：9:00-12:00，13:00-17:00 （不含法定节假日）

网址：
http://www.cn.necel.com/ （中文）
http://www.necel.com/ （英文）

[北京]

日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[上海]

日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[深圳]

日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼
3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

[香港]

香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话： （+852）2886-9318
传真： （+852）2886-9022
2886-9044

[成都]

日电电子（中国）有限公司成都分公司
成都市二环路南三段 15 号天华大厦 7 楼 703 室
电话：(+86)28-8512-5224
传真：(+86)28-8512-5334