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.

RENESAS

使用说明

V850ES/Kx2

32 位单片机

Flash 存储器编程(编程器)

μPD70F3726
μPD70F3728
μPD70F3729
μPD70F3731
μPD70F3732
μPD70F3733
μPD70F3734

文档编号： U17885CA2V1AN00 (第二版)
发布日期： 2008 年 1 月 NS CP(K)

[备忘录]

关于CMOS器件的注意事项

① 输入引脚处的电压波形

输入噪声或由反射引起的波形失真可能导致故障发生。如果由于噪声等影响，使CMOS器件的输入电压范围处于在V_{IL}（MAX）和V_{IH}（MIN）之间，器件可能发生错误。在输入电平固定时以及输入电平从V_{IL}（MAX）到V_{IH}（MIN）的过渡期间，要谨防尖峰噪声影响器件。

② 未使用的输入引脚的处理

CMOS器件的输入端保持开路可能导致故障。如果一个输入引脚未被连接，则由于噪声等原因可能会产生内部输入电平，从而导致故障。CMOS器件的操作特性与双极性或NMOS器件不同。CMOS器件的输入电平必须借助上拉或下拉电路固定于高电平或低电平。每一个未使用引脚都应该连接到V_{DD}或GND，如果有可能作为输出引脚时，需要通过附加电阻连接到V_{DD}或GND。对未使用引脚的处理因器件而异，必须遵循与器件相关的规定和说明。

③ ESD防护措施

如果MOS器件周围有强电场，将会击穿氧化栅极，从而影响器件的工作。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。对于环境必须进行适当的控制。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。半导体器件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包装。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腕带以保证良好接地。不能用手直接接触半导体器件。对于装配有半导体器件的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器件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器件并没有被初始化。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。器件在收到复位信号后才进行初始化。具有复位功能的器件在上电后必须立即进行复位操作。

⑤ 电源开关顺序

一个器件内部工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，可能会导致器件的内部组件过电压，产生异常电流，从而引起内部组件的误操作和性能的劣化。

对每个器件电源的正确开关顺序，必须依据器件的规格说明分别进行判断。

⑥ 电源关闭状态下的输入信号

不要向没有加电的器件输入信号或提供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，从而引起器件的误操作，并产生异常电流，从而使内部组件劣化。

每个器件电源关闭时的信号输入必须依据器件的规格说明分别进行判断。

- **本文档信息发布于 2006 年 11 月。未来可能未经预先通知而进行更改。在实际进行生产设计时，请参阅各产品最新的数据规格书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货及其他信息。**
- 未经本公司事先书面许可，禁止采用任何方式复制或转载本文件中的内容。本文件所登载内容的错误，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，个人电气设备以及产业用机器人。

"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和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或为日本电气电子株式会社（如上定义）开发或制造的产品。

M8E 02.11-1

引言

读者对象

本使用说明供那些了解 V850ES/Kx2 单片机功能以及将使用本产品设计应用系统的人阅读。

目的

本使用说明旨在于帮助用户了解如何开发专用 Flash 编程器，用于重写 V850ES/Kx2 单片机内部的 Flash 存储器。

本文档中的例子程序与电路图仅可作为参考，并非供实际设计中使用。

因此，用户使用这些例子程序时风险自负，使用这些例子程序并不保证操作正确。

编排组织

本手册主要由以下部分组成：

- Flash 存储器编程
- 编程器工作环境
- 编程器基本操作
- 命令/数据帧格式
- 命令处理过程描述
- UART 通信模式
- 支持握手功能(CSI + HS)的 3 线串行 I/O 通信模式
- 3 线串行 I/O 通信模式(CSI)
- Flash 存储器编程的指标参数

如何阅读本手册

假定本手册的读者具备电子工程、逻辑电路以及单片机领域内的常识。

□ 如果希望对其功能作一般性的了解：

→ 根据**目录**阅读本手册内容， 标有 “<R>” 标志表示主要修订之处。将“<R>”复制到 PDF 文件中“查找”字段中，可以很方便搜索到修订的内容。

□ 如果希望更多地了解关于 V850ES/Kx2 的硬件功能：

→ 阅读各个 V850ES/Kx2 产品的用户手册。

惯例

数据有效位：	高位在左，低位在右
低电平有效表示：	$\overline{\text{xxx}}$（在引脚或信号名称之上有上划线）
注：	正文中用**注**标记的脚注
注意事项：	需要特别关注的信息
备注：	补充信息
数值表示：	二进制 ... xxxx 或 xxxxB
	十进制 ... xxxx
	十六进制 ... xxxxH

相关文档　　本次发布所指的相关文档也许包括早期版本，而早期版本并未标明如此。

和器件相关的文档

文档名称	文档编号
V850ES/KE2 用户手册	U17705E
V850ES/KF2 用户手册	U17704E
V850ES/KG2 用户手册	U17703E
V850ES/KJ2 用户手册	U17702E
V850ES 体系结构用户手册	U15943E

目录

第一章　FLASH 存储器编程

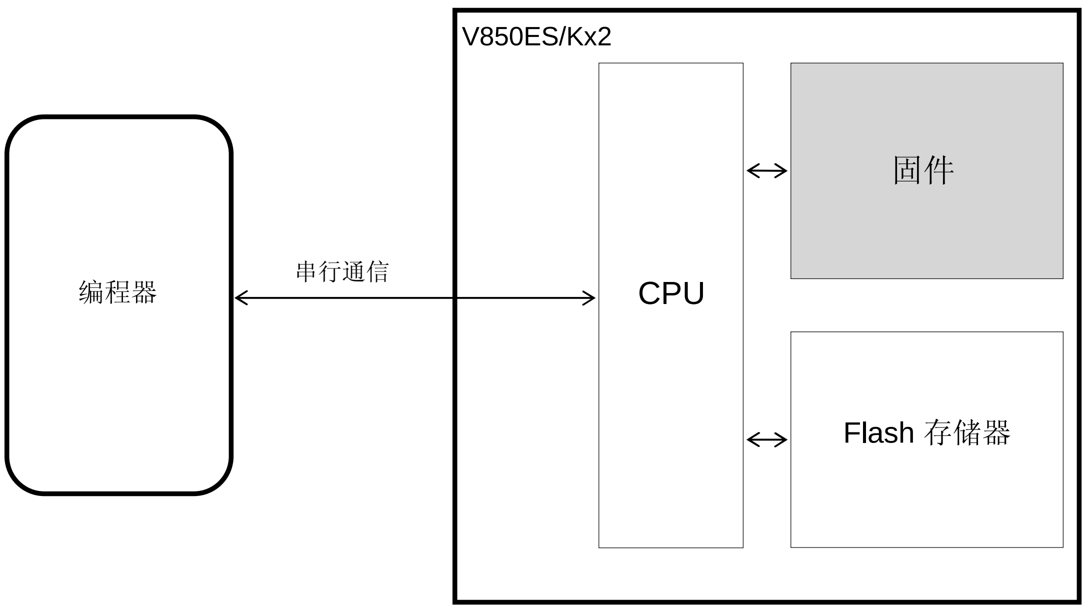
为了重写 V850ES/Kx2 单片机内部 flash 存储器的内容，通常使用一个专用的 flash 存储器编程器（以下简称“编程器”）。

本使用说明手册描述了如何开发一个专用编程器。

1.1　概要

V850ES/Kx2 单片机包含了控制 flash 存储器编程的固件。内部 flash 存储器的编程由编程器和 V850ES/Kx2 单片机之间通过串行通信的方式发送/接收命令来完成。

图 **1-1. V850ES/Kx2** 单片机中 **Flash** 存储器编程的系统框图



1.2 系统构成

flash 存储器编程的系统构成例子如图 1-2 所示。

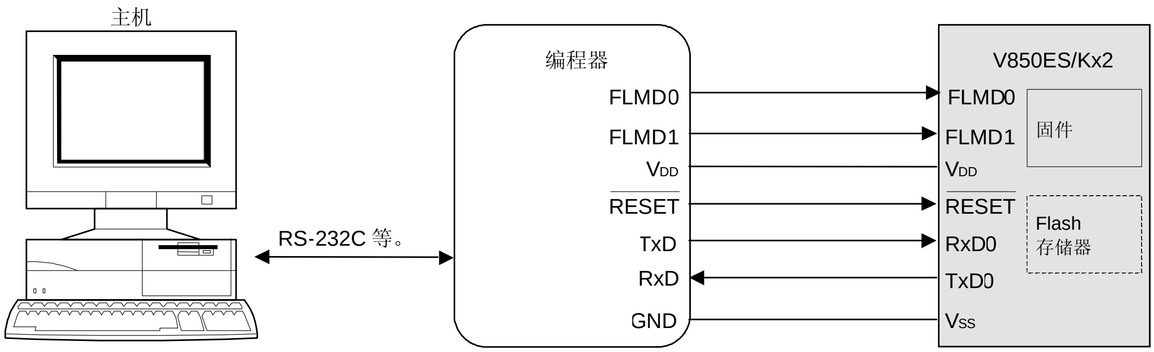
这些例图显示了在主机的控制下如何利用编程器对 flash 存储器进行编程。

根据编程器的连接方式，如果一个用户程序已经被预先下载到编程器，则编程器能够在不使用主机的独立模式下使用。

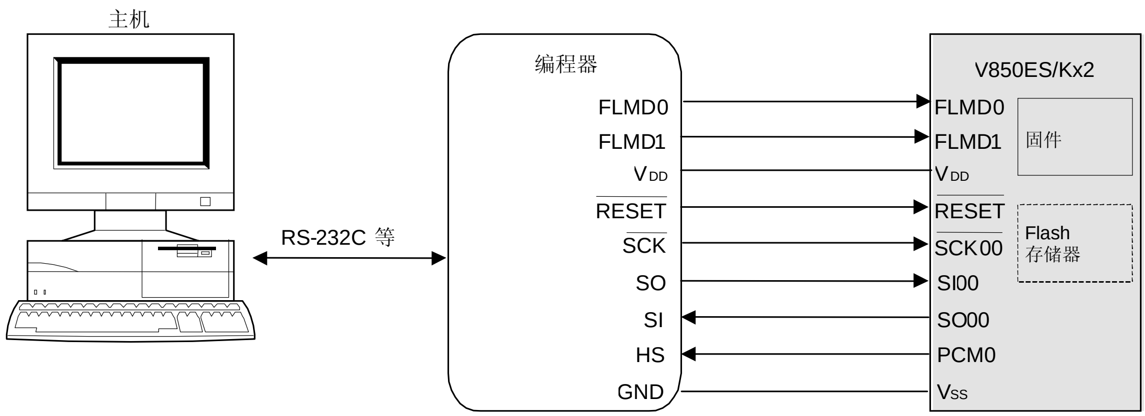
例如： NEC Electronics 的 flash 存储器编程器 PG-FP4 能够在连接主机时使用 GUI 软件进行编程，也能够自行编程（独立模式下）。

图 1-2. 系统构成举例

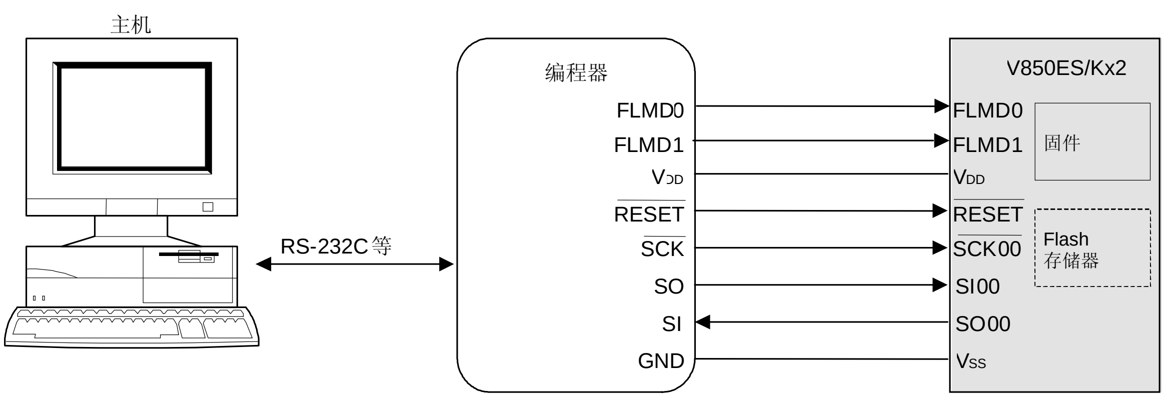
(1) UART 通信模式 (LSB 先行传输)



(2) 支持握手功能的 3 线串行 I/O 通信模式(CSI + HS) (MSB 先行传输)



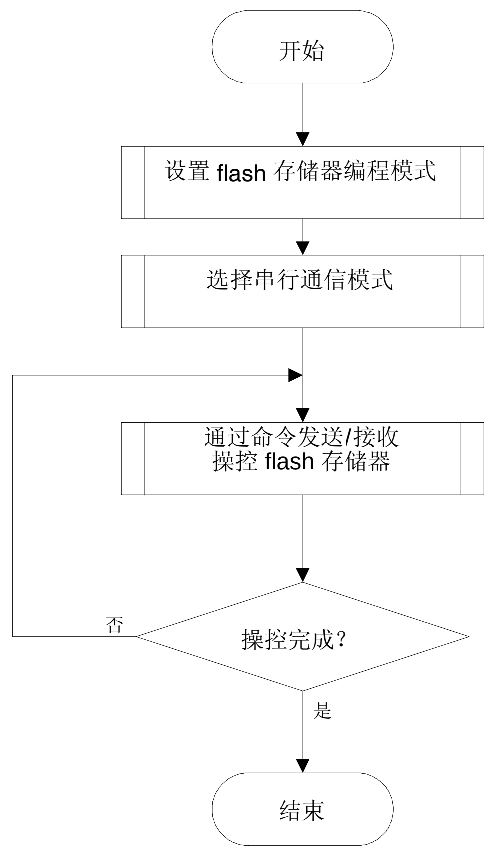
(3) 3 线串行 I/O 通信模式 (CSI) (MSB 先行传输)



1.3 编程概要

为使用编程器来重写 flash 存储器的内容，V850ES/Kx2 单片机必须首先设置为 flash 存储器编程模式。之后，选择编程器和 V850ES/Kx2 单片机之间的通信模式，以串行通信的方式从编程器发送命令，于是重写 flash 存储器。编程流程图如图 1-3 所示：

图 1-3. 编程流程图



1.3.1 设置 flash 存储器编程模式

在 V850ES/Kx2 单片机中，提供指定电压至 flash 存储器编程模式的设置引脚（FLMD0 和 FLMD1）并进行一次复位，则设置为 flash 存储器编程模式。

1.3.2 选择串行通信模式

为了选择串行通信模式，在 flash 存储器编程模式时，通过在 V_{DD} 和 GND 之间改变设置引脚（FLMD0）的电压来产生脉冲，并且根据脉冲数量确定通信模式。

1.3.3 利用命令发送/接收操控 flash 存储器

V850ES/Kx2 单片机中集成的 flash 存储器具有重写内容功能。flash 存储器可用的操作功能如下表 1-1 所示：

表 1-1. Flash 存储器功能概要

功能	概要
擦除	擦除 flash 存储器的内容
写入	向 flash 存储器中写入数据
校验	将 flash 存储器的内容与用于校验的数据作比较
信息采集	读取有关 flash 存储器的信息

为了控制这些功能，编程器以串行通信的方式发送命令至 V850ES/Kx2 单片机。V850ES/Kx2 单片机返回命令的应答状态。通过重复这些串行通信序列来实现对 flash 存储器的编程。

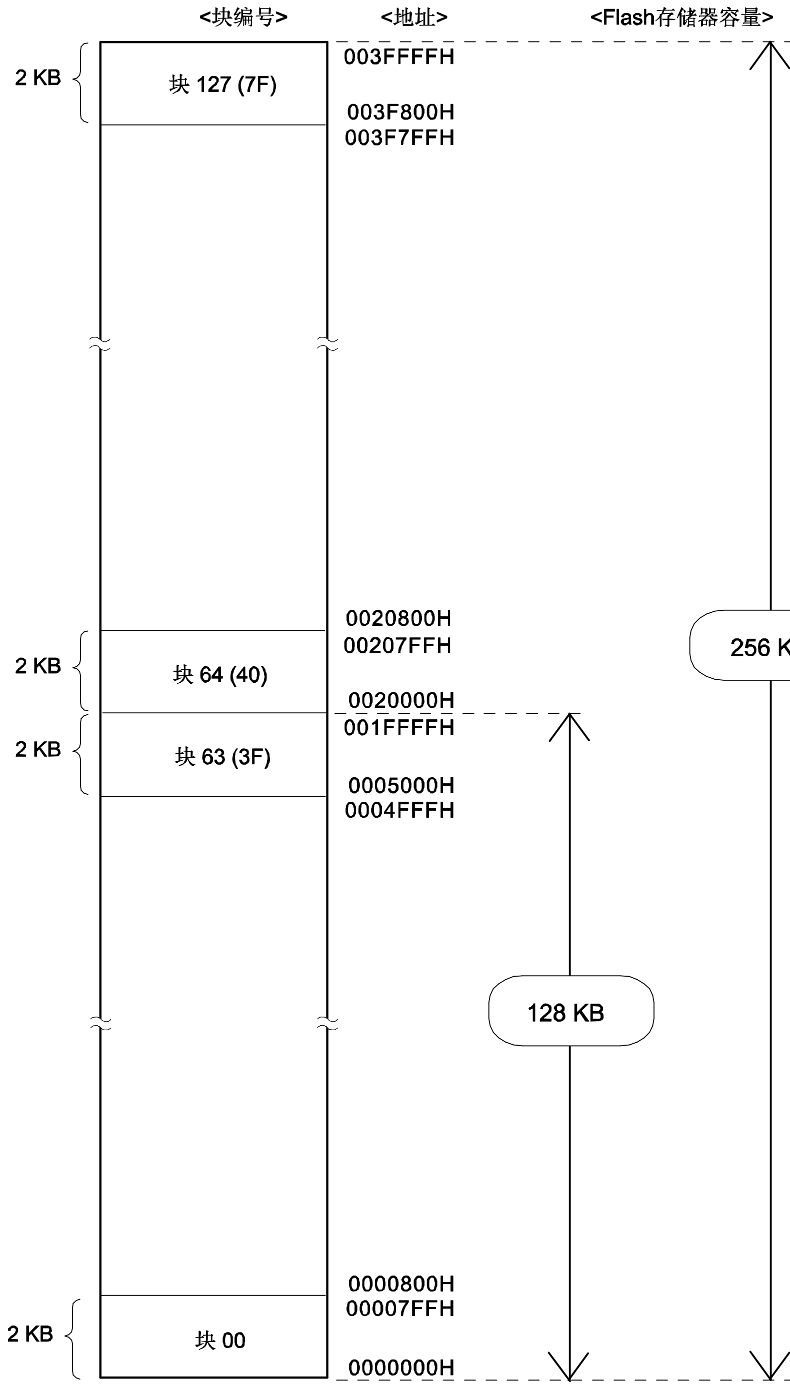
1.4 V850ES/Kx2 单片机的特定信息

编程器必须处理产品特定信息（如一个器件名和存储器信息）。表 1-2 显示了 V850ES/Kx2 单片机的 flash 存储器容量， 图 1-4 显示了 flash 存储器的结构。

表 1-2. V850ES/Kx2 单片机的 Flash 存储器容量

器件名		Flash 存储器容量
V850ES/KE2	μPD70F3726	128 KB
V850ES/KF2	μPD70F3728	128 KB
	μPD70F3729	256 KB
V850ES/KG2	μPD70F3731	128 KB
	μPD70F3732	256 KB
V850ES/KJ2	μPD70F3733	128 KB
	μPD70F3734	256 KB

图 **1-4. Flash** 存储器结构



备注　　各块由 2 KB 组成（本图仅显示了 flash 存储器所有块的若干部分）。

第二章　编程器工作环境

2.1　编程器控制引脚

表 2-1 列出了在用户系统中为实现编程器功能，编程器必须控制的引脚。关于各引脚的细节，参见以下各页。

表 2-1. 引脚说明

编程器			V850ES/Kx2	与目标通信的模式		
信号名称	I/O	引脚功能	引脚名称	CSI	CSI + HS	UART
FLMD0	输出	用于设置编程模式的信号电平输出和选择通信模式的脉冲输出	FLMD0	○	○	○
FLMD1	输出	用于设置编程模式的信号电平输出	FLMD1	○	○	○
V_{DD}	输出	V_{DD} 电压产生/监控	V_{DD}	△	△	△
GND	–	接地	V_{SS}	○	○	○
CLK	输出	输出至 V850ES/Kx2 单片机的工作时钟	X1、X2 [注]	△	△	△
$\overline{\text{RESET}}$	输出	编程模式转换触发	$\overline{\text{RESET}}$	○	○	○
SO	输出	至 V850ES/Kx2 单片机的命令发送	SI00	○	○	×
SI	输入	接收自 V850ES/Kx2 单片机的应答状态和数据	SO00	○	○	×
$\overline{\text{SCK}}$	输出	至 V850ES/Kx2 单片机的串行时钟供给	$\overline{\text{SCK00}}$	○	○	×
HS (握手信号)	输入	用于接收与 V850ES/Kx2 单片机串行通信的握手信号	PCM0	×	○	×
TxD	输出	至 V850ES/Kx2 单片机的命令发送	RXD0	×	×	○
RxD	输入	接收自 V850ES/Kx2 单片机的应答状态和数据	TXD0	×	×	○

注　　将 X1 的反相信号连接至 X2。

备注　　○：必须连接的引脚。
×：不必连接的引脚。
△：如果信号在目标板上产生，则该引脚不必连接。

关于由编程器控制的引脚电压，参见支持 flash 存储器编程的器件用户手册。

2.2　控制引脚的详情

2.2.1　Flash 存储器编程模式设置引脚(FLMD0、 FLMD1)

FLMD0 引脚和 FLMD1 引脚用于控制 V850ES/Kx2 单片机的工作模式。当提供指定电压至这两个引脚并进行一次复位后， 则 V850ES/Kx2 单片机工作在 flash 存储器编程模式下。

复位后，编程器和 V850ES/Kx2 单片机之间的串行通信模式，通过在 V_{DD} 和 GND 之间控制 FLMD0 引脚的电压，从而输出脉冲来确定。关于 FLMD0 引脚脉冲数量和通信模式之间的关系，参见表 2-3。

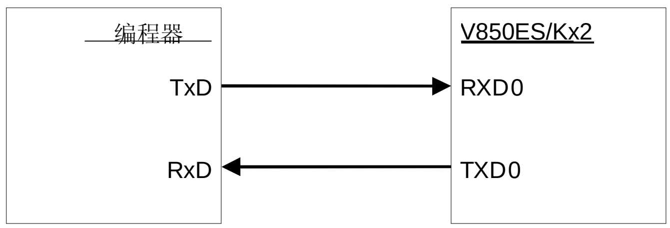
2.2.2 串行接口引脚 (TxD、 RxD、 SI、 SO、$\overline{\text{SCK}}$、 HS)

串行接口引脚用于传输编程器和 V850ES/Kx2 单片机之间的 flash 存储器写入命令。

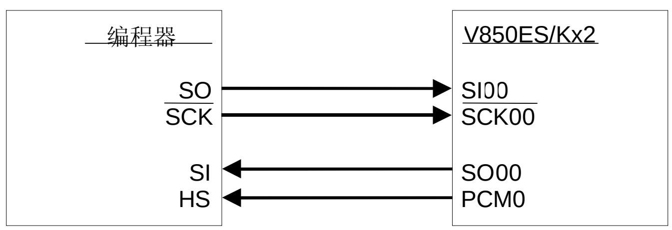
对于 V850ES/Kx2 单片机， 有 UART、 CSI + HS、以及 CSI 三种通信模式可供选择。下图显示了用于各通信模式引脚的连接：

图 **2-1.** 串行接口引脚

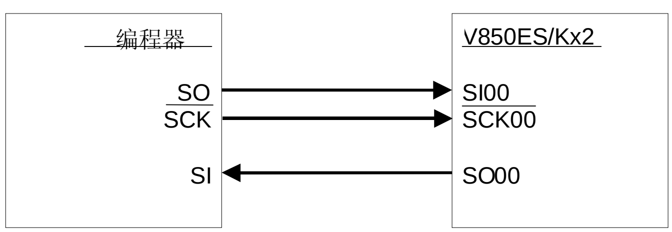
(1) UART 通信模式



(2) 支持握手功能的 **3** 线串行 **I/O** 通信模式**(CSI + HS)**



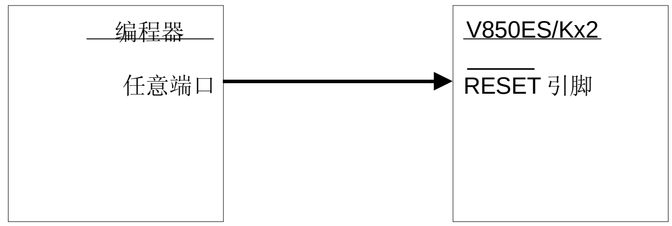
(3) 3 线串行 **I/O** 通信模式 **(CSI)**



2.2.3　复位控制引脚 ($\overline{\text{RESET}}$)

复位控制引脚用于由编程器控制 V850ES/Kx2 单片机的系统复位。当提供指定电压至 FLMD0 引脚和 FLMD1 引脚并进行一次复位后，则选择为 Flash 存储器编程模式。

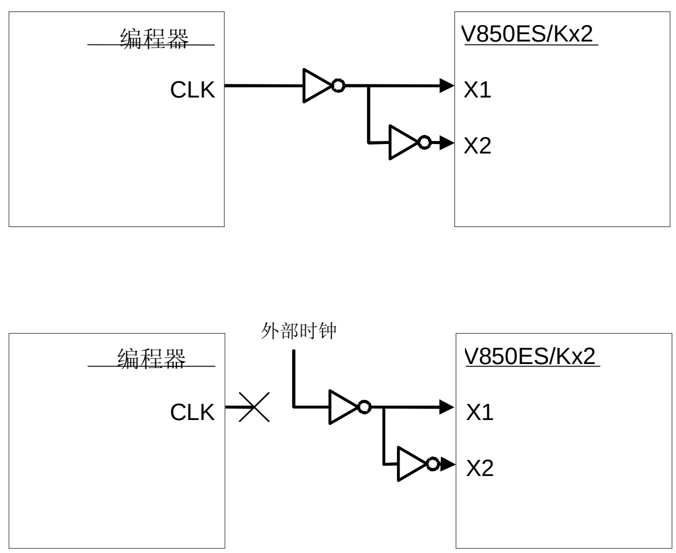
图 2-2.　复位控制引脚



2.2.4　时钟控制引脚 (CLK)

仅当由编程器提供时钟至 V850ES/Kx2 单片机时，使用该时钟控制引脚。当不必由编程器提供工作时钟至 V850ES/Kx2 单片机时，则该引脚不必连接。

图 2-3.　时钟控制引脚

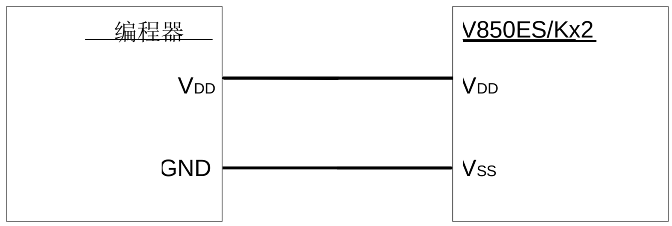


2.2.5　V_{DD}/GND 控制引脚

V_{DD} 控制引脚用于由编程器提供电源至 V850ES/Kx2 单片机。当不必从编程器提供电源至 V850ES/Kx2 单片机时，则该引脚不必连接。然而，当使用专用编程器时，因为专用编程器监控 V850ES/Kx2 单片机的电源供给状态，所以无论电源供给是否来自编程器，都必须连接该引脚。

无论电源供给是否来自编程器，GND 控制引脚必须连接至 V850ES/Kx2 单片机的 V_{SS} 引脚。

图 2-4. V_{DD}/GND 控制引脚



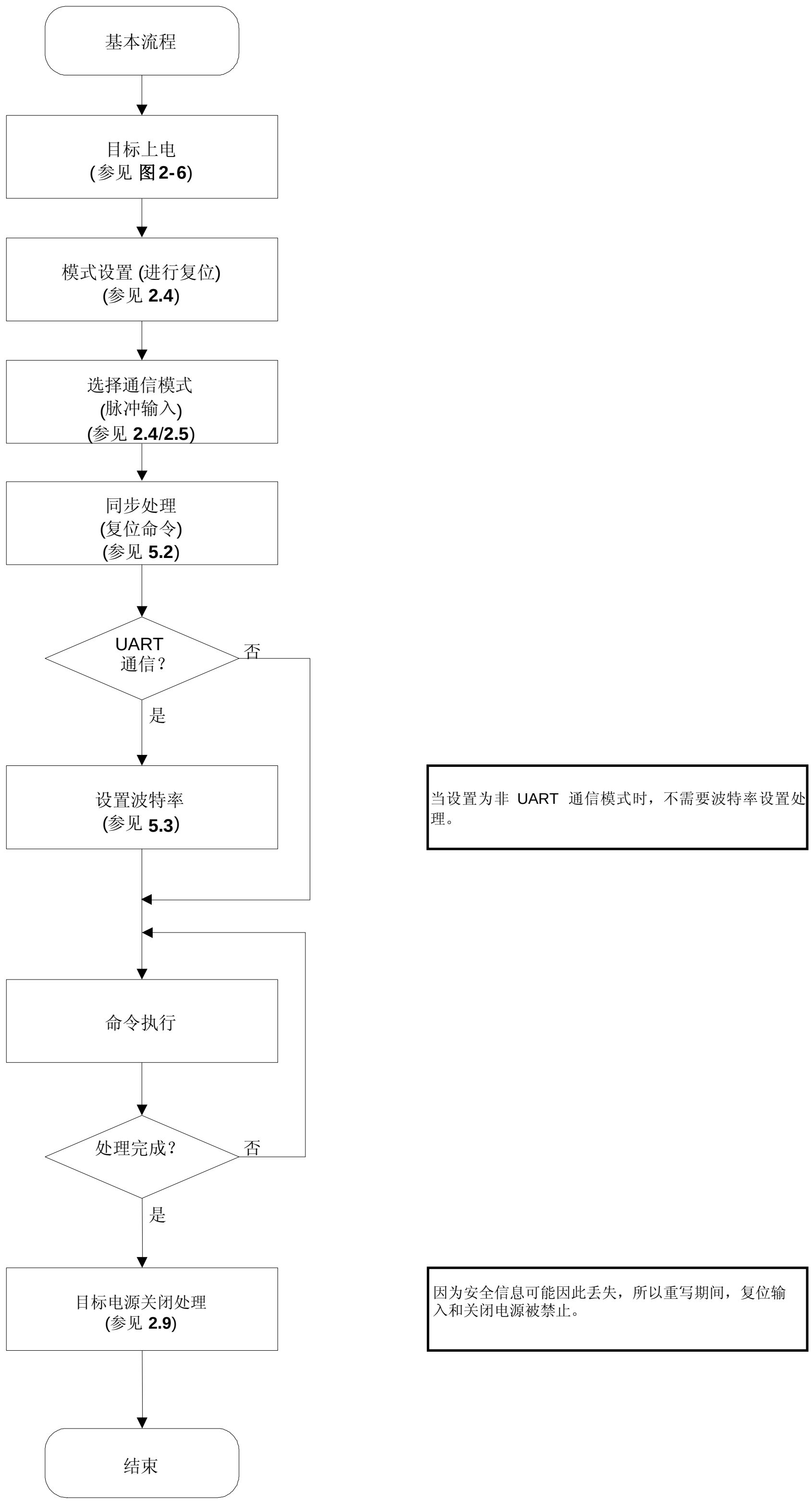
2.2.6　其它引脚

关于未连接至编程器的引脚连接，参见各器件用户手册中描述 flash 存储器的章节。

2.3　基本流程图

下图显示了利用编程器执行 flash 存储器重写功能的基本流程图：

图 2-5. Flash 存储器重写处理的基本流程图

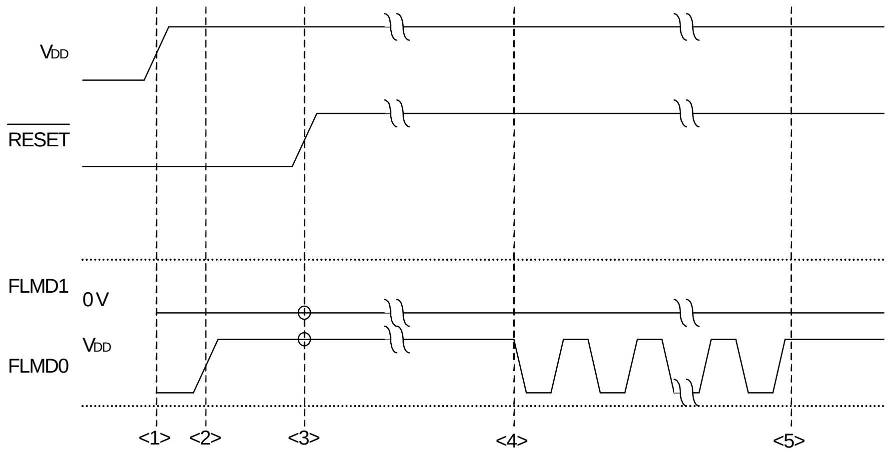


2.4　设置 Flash 存储器编程模式

为了利用编程器重写 flash 存储器的内容，必须首先通过提供指定电压至 V850ES/Kx2 单片机中 flash 存储器编程模式设置引脚（FLMD0、 FLMD1）并进行一次复位，从而将 V850ES/Kx2 单片机设置为 flash 存储器编程模式。

下图显示了设置 flash 存储器编程模式以及通信模式选择时序图：

图 2-6. 设置 flash 存储器编程模式以及通信模式选择



<1>： 上电 (V_{DD})
<2>： FLMD0 = 高电平， FLMD1 = 低电平
<3>： 进行复位 (模式设置)
<4>： 脉冲输出开始
<5>： 脉冲输出结束

复位后的 FLMD0 引脚和 FLMD1 引脚设置与工作模式之间的关系如下表所示：

表 2-2. 复位后的 FLMD0 引脚和 FLMD1 引脚设置与工作模式之间的关系

FLMD0	FLMD1	工作模式
低电平 (GND)	任意电平	正常工作模式
高电平 (V_{DD})	低电平(GND)	Flash 存储器编程模式
高电平(V_{DD})	高电平(V_{DD})	设置禁止

2.5　串行通信模式选择

通信模式由复位后输入 V850ES/Kx2 单片机中 FLMD0 引脚的脉冲数量来决定。
FLMD0 脉冲的高电平和低电平分别为 V_{DD} 和 GND。
下表显示了 FLMD0 脉冲数量与 V850ES/Kx2 单片机所能选择的通信模式之间的关系：

表 2-3. FLMD0 脉冲数量与通信模式之间的关系

通信模式	FLMD0 脉冲数量	用于通信的端口
UART (UART0)	0	TxD0 (P30)、RxD0 (P31)
3 线串行 I/O (CSI00)	8	SO00 (P41)、 SI00 (P40)、 $\overline{\text{SCK00}}$ (P42)
支持握手功能的 3 线串行 I/O (CSI00 + HS)	11	SO00 (P41)、 SI00 (P40)、 $\overline{\text{SCK00}}$ (P42)、 PCM0
设置禁止	其它	–

2.6　UART 通信模式

RxD 引脚和 TxD 引脚用于 UART 通信。通信状态如下表所示：

表 2-4. UART 通信状态

名称	说明
波特率	可从 9，600、 19，200、 31，250、 38，400、 76，800、以及 153，600 bps 中选择(缺省值：9，600 bps)
奇偶校验位	无
数据长度	8 位 (LSB 先发)
结束位	1 位

CSI 通信模式期间，编程器总是工作于主设备状态下，因此编程器必须检查 V850ES/Kx2 单片机的处理（如：写入或擦除）是否正常完成。而另一方面，在 UART 通信期间主从设备的状态偶尔会调换，因此不用象 CSI + HS 通信时一样另行分配一个引脚，也能进行时序上最为合理的通信。

注意事项　当执行 UART 通信时，设置相同的波特率至主从设备。

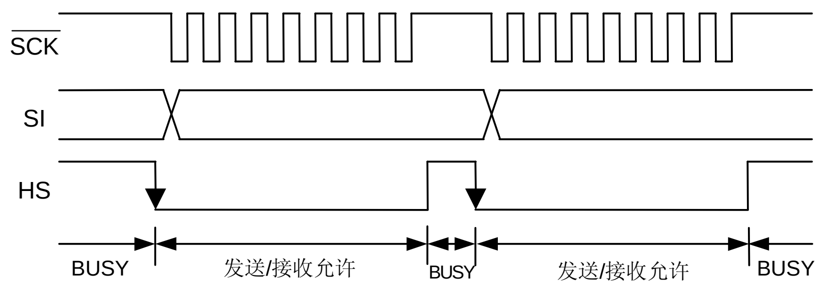
2.7　支持握手功能的 3 线串行 I/O 通信模式 (CSI + HS)

在 CSI + HS 通信模式下， 命令或数据的通信时序是最佳的。除 SI 引脚、SO 引脚以及 $\overline{\text{SCK}}$ 引脚之外，HS (握手) 引脚也用来使通信更有效率。

当 V850ES/Kx2 单片机准备好发送或接收数据时，HS 引脚信号的电平下降（变为低电平）。V850ES/Kx2 单片机的命令或数据开始发送/接收之前，编程器必须检测 HS 引脚信号的下降沿（低电平）。

通信数据格式为：以字节为单位 MSB 先发， 保证时钟频率为 2.5 MHz 或更低。

图 2-7. CSI + HS 通信时序图



2.8　3 线串行 I/O 通信模式(CSI)

$\overline{\text{SCK}}$ 引脚、 SO 引脚以及 SI 引脚用于 CSI 通信。 由于编程器总是工作于主设备状态下，因此，如果 V850ES/Kx2 单片机还没有准备好发送/接收时，数据就经由 $\overline{\text{SCK}}$ 引脚发送，通信可能不会正常地执行。

通信数据格式为：以字节为单位 MSB 先发， 保证时钟频率为 2.5 MHz 或更低。

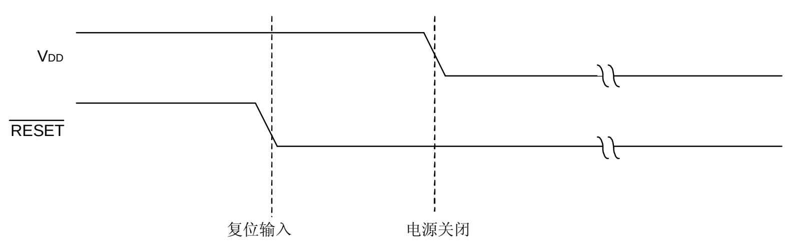
2.9　关闭目标供电电源

各命令执行完成之后，关闭至目标器件的供电电源之前设置 $\overline{\text{RESET}}$ 引脚为低电平，如下图所示：

当关闭至目标器件的供电电源时，设置其它引脚为 Hi-Z（高阻）。

注意事项　命令处理期间，关闭电源及输入复位被禁止。

图 2-8. 终止 Flash 存储器编程模式的时序



2.10 Flash 存储器的操控

V850ES/Kx2 单片机中集成的 flash 存储器有如表 2-5 所列的操控功能。为了控制这些功能，编程器发送命令至 V850ES/Kx2 单片机，并且检查自 V850ES/Kx2 单片机的应答状态，最终达到操控 flash 存储器的目的。

表 2-5. Flash 存储器操控功能列表

分类	功能名称	说明
擦除	芯片擦除	擦除整个 flash 存储区域。清除安全标志。
	块擦除	擦除 flash 存储器中的指定块。
写入	写入	至 flash 存储器指定区域中写入数据。
验证	验证	在 V850ES/Kx2 单片机一侧，将编程器发送的数据与从 flash 存储器指定地址中获取的数据相比较。
空白检查	块空白检查	检查 flash 存储器指定区域的擦除状态。
读取	读取	读取 flash 存储器指定区域的数据。
信息采集	‘硅标记’信息采集	获取写入协议信息。
	版本信息采集	获取 V850ES/Kx2 单片机及固件的版本信息。
	状态信息采集	获取当前操作状态。
	校验和信息采集	获取指定区域的校验和数据。
安全性	安全设置	设置安全信息。
其它	复位	检测通信同步。

2.11 命令列表

由编程器使用的命令，其功能如下表所示：

表 2-6. 由编程器发送至 V850ES/Kx2 单片机的命令列表

命令编号	命令名称	功能
70H	状态	获取当前操作状态(状态数据)。
00H	复位	检测通信同步。
90H	振荡频率设置	指定 V850ES/Kx2 单片机的振荡频率。
9AH	波特率设置	当 UART 通信模式被选择时，设置波特率。
20H	芯片擦除	擦除整个 flash 存储区域。
22H	块擦除	擦除 flash 存储器指定区域。
40H	编程	写入数据至 flash 存储器指定区域。
13H	验证	将 flash 存储器指定区域中的内容与由编程器发送的数据相比较。
32H	块空白检查	检查 flash 存储器中指定模块的擦除状态。
C0H	‘硅标记’	获取 V850ES/Kx2 单片机的信息 (器件编号、 flash 存储器结构等)。
C5H	获取版本信息	获取 V850ES/Kx2 单片机及固件的版本信息。
B0H	校验和	获取指定区域的校验和数据。
A0H	安全设置	设置安全信息。
50H	读取	读取 flash 存储器指定区域的数据。

2.12 状态列表

下表列出了编程器接收自 V850ES/Kx2 单片机的状态码：

表 2-7. 状态码列表

状态码	状态	说明
04H	命令编号错误	接收到一个不被支持的命令时，出错返回。
05H	参数错误	命令信息(参数) 无效时，出错返回。
06H	确认正常(ACK)	确认正常。
07H	校验和错误	由编程器发送的帧中数据异常时，出错返回。
0FH	验证错误	由编程器发送的验证数据中，出现验证错误时，出错返回。
10H	保护错误	试图执行由安全设置命令禁止的处理时，出错返回。
15H	确认异常 (NACK)	确认异常。
18H	FLMD 错误	出现写入错误时，出错返回 。
1AH	MRG10 错误	擦除出错 。
1BH	MRG11 错误	数据写入期间，内部验证出错或空白检查出错。
1CH	写入错误	写入出错。
FFH	处理进行中 (BUSY)	忙应答[注]。

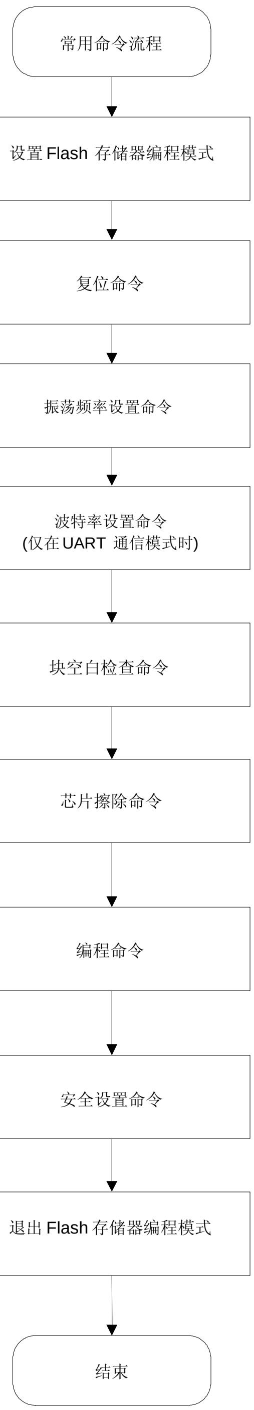
注　　CSI 通信期间， 1 字节的“FFH”可能被发送，同时又被当作数据帧格式。

本手册中，接收一个校验和错误或 NACK 都被视为立即异常终止。然而，当开发一个专用编程器时，由校验和错误或 NACK 错误或经由 HS 引脚的 BUSY 状态检查后引起的命令发送之前，毫无疑问，处理可能会立即由等待状态进行重试。 因此，建议限制重试次数以阻止该重试操作的无限重复。

尽管没有在上表列出，如果超时错误（BUSY 超时、HS 引脚超时、或者在 UART 通信期间数据帧接收超时）发生时，建议关闭至 V850ES/Kx2 单片机的电源供给(参见 **2.9 关闭目标供电电源**) ，然后再次接通电源。

图 3-1 显示了利用编程器执行 flash 存储器重写功能时的常用命令执行流程：

图 3-1. Flash 存储器重写时的常用命令执行流程



注 在编程器规格说明中，建议将执行安全设置使读禁止作为一项缺省设置。

备注 同样支持验证命令、校验和命令、以及读取命令。

第四章　命令/数据帧格式

编程器使用命令帧发送命令至 V850ES/Kx2 单片机。V850ES/Kx2 单片机使用数据帧发送写入数据或验证数据至编程器。为增强传输数据的可靠性，将帧头信息、帧尾信息、数据长度信息以及校验和信息附加到各帧。

以下显示了命令帧和数据帧的格式：

图 4-1. 命令帧格式

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

图 4-2. 数据帧格式

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

表 4-1. 各帧中符号说明

符号	值	说明
SOH	01H	命令帧帧头
STX	02H	数据帧帧头
LEN	–	数据长度信息 (00H 表示 256)。 命令帧：COM +命令信息长度 数据帧：数据字段长度
COM	–	命令编号
SUM	–	关于帧的校验和数据 以字节为单位，通过从初始值(00H)中连续减去所有计算对象数据而获得(忽略借位)。 计算对象如下： 命令帧：　LEN + COM +所有命令信息 数据帧：　LEN +所有数据
ETB	17H	除末尾帧之外的数据帧帧尾
ETX	03H	命令帧帧尾、或末尾数据帧帧尾

以下显示了计算帧的校验和例子。

[命令帧]

以下状态命令帧的例子中没有包含命令信息，因此，LEN 和 COM 是校验和计算的对象。

SOH	LEN	COM	SUM	ETX
01H	01H	70H	校验和	03H
	校验和计算对象			

该命令帧的校验和数据按照如下方法获得：

00H (初始值) – 01H (LEN) – 70H (COM) = 8FH （忽略借位。仅低 8 位。）

最终发送的命令帧如下：

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

[数据帧]

为了发送如下所示的数据帧，LEN 及 D1 至 D4 是校验和的计算对象。

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

该数据帧的校验和数据按照如下方法获得：

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

最终发送的数据帧如下：

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

当接收到一个数据帧时，校验和数据按同样的方法进行计算，通过判断所得值与存储在接收数据 SUM 字段中的值是否相同来检测校验和错误。例如，当接收到如下所示的一个数据帧时，则检测到一个校验和错误。

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

↑正常时，应为 1BH

4.1 命令帧发送处理

关于各通信模式时发送命令帧的命令处理流程图之详情，请阅读以下章节：

- 关于 UART 通信模式，阅读 **6.1 命令帧发送处理流程图**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.1 命令帧发送处理流程图**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.1 命令帧发送处理流程图**。

4.2 数据帧发送处理

写入数据帧（用户程序）、验证数据帧（用户程序）、以及安全数据帧（安全标志）作为一个数据帧被发送。
关于各通信模式时发送数据帧的命令处理流程图之详情，请阅读以下章节：

- 关于 UART 通信模式，阅读 **6.2 数据帧发送处理流程图**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.2 数据帧发送处理流程图**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.2 数据帧发送处理流程图**。

4.3 数据帧接收处理

状态帧、‘硅标记’数据帧、版本数据帧、以及校验和数据帧作为一个数据帧被接收。
关于各通信模式时接收数据帧的命令处理流程图之详情，请阅读以下章节：

- 关于 UART 通信模式，阅读 **6.3 数据帧接收处理流程图**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.3 数据帧接收处理流程图**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.3 数据帧接收处理流程图**。

第五章 命令处理描述

5.1 状态命令

5.1.1 描述说明

该命令用于检查各命令（如写入或擦除）发布之后 V850ES/Kx2 单片机的工作状态。

状态命令发布后，如果因为基于通信或类似的原因，在 V850ES/Kx2 单片机中，该状态命令帧不能被正常接收，则 V850ES/Kx2 单片机无法完成状态设置。结果，会收到一个忙应答（FFH）而不是状态帧。在这样的情况下，重试状态命令。

5.1.2 命令帧和状态帧

图 5-1 显示了状态命令的命令帧格式，图 5-2 显示了该命令的状态帧：

图 5-1. 状态命令帧（自编程器至 V850ES/Kx2 单片机）

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

图 5-2. 状态命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注　**1.** ST1 至 STn：状态#1 至状态#n

　　　2. 状态帧的长度随发送至 V850ES/Kx2 单片机的各命令（如写入或擦除）而改变。

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 在 UART 通信模式时，不使用状态命令。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.4 状态命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.4 状态命令**。

注意事项　在 UART 通信时，各命令（如写入或擦除）被发送后， V850ES/Kx2 单片机在指定时间内自动返回状态帧。因此不使用状态命令。

如果在 UART 通信时状态命令被发送，则返回命令编号错误。

5.2　复位命令

5.2.1　描述说明

该命令用于检查通信模式设置后编程器和 V850ES/Kx2 单片机之间的通信建立。

当 UART 被选择为与 V850ES/Kx2 单片机的通信模式时，编程器和 V850ES/Kx2 单片机的波特率设置值必须相同。 但是 V850ES/Kx2 单片机不能检测其自身的工作频率，因此不能设置波特率。通过以 9，600 bps 速率由编程器发送“00H”两次，并测量“00H”低电平的宽度，然后计算这两次所发送信号的平均值，则使 V850ES/Kx2 单片机中工作频率的检测成为可能。从而，可以设置波特率，并能够在通信中进行同步检测。

5.2.2　命令帧和状态帧

图 5-3 显示了复位命令的命令帧格式， 图 5-4 显示了该命令的状态帧：

图 5-3. 复位命令帧（自编程器至 V850ES/Kx2 单片机）

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

图 5-4. 复位命令的状态帧（自 V850ES/Kx2 单片机至编程器）

STX	LEN	数据	SUM	ETX
02H	1	ST1	校验和	03H

备注　　ST1：同步检测结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.4 复位命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.5 复位命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.5 复位命令**。

5.3　波特率设置命令

5.3.1　描述说明

该命令用于改变 UART 通信的波特率（缺省值：9，600 bps）。

波特率设置命令执行后，在新的波特率设置值条件下，为了确认同步，必须执行复位命令。

仅在 UART 通信模式时，波特率设置命令有效。设置的波特率数据由一个字节值代表。

如果在非 UART 通信模式下发送波特率设置命令，则 V850ES/Kx2 单片机忽略波特率设置命令。

5.3.2　命令帧和状态帧

图 5-5 显示了波特率设置命令的命令帧格式，图 5-6 显示了该命令的状态帧：

图 5-5. 波特率设置命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息	SUM	ETX
01H	02H	9AH (波特率设置)	D01	校验和	03H

备注　D01：波特率选择值 ：

D01 值	03H	04H	05H	06H	07H	08H
波特率 (bps)	9600	19200	31250	38400	76800	153600

图 5-6. 波特率设置命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注　ST1：同步检测结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 对于 UART 通信模式，阅读 **6.5 波特率设置命令**。
- 支持握手功能的 3 线串行 I/O 通信模式（CSI + HS）不使用波特率设置命令。
- 3 线串行 I/O 通信模式（CSI）不使用波特率设置命令 。

5.4 振荡频率设置命令

5.4.1 描述说明

该命令用于设置 V850ES/Kx2 单片机的振荡频率数据。

设置实际输入至 V850ES/Kx2 单片机 X1 引脚的时钟频率。

然而，当该命令成功执行且时钟频率在 2 至 5 MHz 的范围内时，目标器件通过内部 PLL 将 CPU 的工作时钟乘以 4。因此，时钟数乘以 4，用此计算值设置各时序参数。

5.4.2 命令帧和状态帧

图 5-7 显示了振荡频率设置命令的命令帧格式，图 5-8 显示了该命令的状态帧：

图 5-7. 振荡频率设置命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息				SUM	ETX
01H	05H	90H (振荡频率设置)	D01	D02	D03	D04	校验和	03H

备注 D01 至 D04：振荡频率 = $(D01 \times 0.1 + D02 \times 0.01 + D03 \times 0.001) \times 10^{D04}$ (单位： kHz)

设置范围： 10 kHz 至 100 MHz，但实际发送命令时，根据各器件的规格说明来设值该值。

D01 至 D03 内为原始 8 位的 BCD 码，而 D04 为一个带符号整数。

设置例子：要设置为 6 MHz，则：

D01 = 06H

D02 = 00H

D03 = 00H

D04 = 04H

振荡频率 = $6 \times 0.1 \times 10^4$ = 6，000 kHz = 6 MHz

设置例子：要设置为 10 MHz，则：

D01 = 01H

D02 = 00H

D03 = 00H

D04 = 05H

振荡频率= $1 \times 0.1 \times 10^5$ = 10，000 kHz = 10 MHz

图 5-8. 振荡频率设置命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注 ST1：振荡频率设置结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.6 振荡频率设置命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.6 振荡频率设置命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.6 振荡频率设置命令**。

5.5　芯片擦除命令

5.5.1　描述说明

该命令用于擦除整个 flash 存储器的内容。另外，只要安全设置没有禁止擦除（参见 **5.13 安全设置命令**），所有由安全设置处理所设置的信息也能够通过芯片擦除处理来初始化。

5.5.2　命令帧和状态帧

图 5-9 显示了芯片擦除命令的命令帧格式，图 5-10 显示了该命令的状态帧：

图 **5-9.** 芯片擦除命令帧（自编程器至 **V850ES/Kx2** 单片机）

SOH	LEN	COM	SUM	ETX
01H	01H	20H (芯片擦除)	校验和	03H

图 **5-10.**芯片擦除命令的状态帧（自 **V850ES/Kx2** 单片机至编程器）

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

备注　　ST1：芯片擦除结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.7　芯片擦除命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.7　芯片擦除命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.7　芯片擦除命令**。

5.6　块擦除命令

5.6.1　描述说明

只要安全设置没有禁止擦除（参见 **5.13 安全设置命令**），该命令就可用于擦除 flash 存储器中指定编号块的内容。

5.6.2　命令帧和状态帧

图 5-11 显示了块擦除命令的命令帧格式，图 5-12 显示了该命令的状态帧：

图 5-11. 块擦除命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	22H (块擦除)	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注　SAH、 SAM、SAL： 块擦除起始地址（指定块的起始地址）
　　　EAH、 EAM、EAL： 块擦除结束地址（指定块的结束地址）

图 5-12. 块擦除命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注　ST1： 块擦除结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.8 块擦除命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.8 块擦除命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.8 块擦除命令**。

5.7　编程命令

5.7.1　描述说明

写入起始地址和写入结束地址发送后，使用该命令发送待写入的字节数量。 然后将用户程序写至 flash 存储器并在内部对其进行验证。

写入开始/结束地址只能以块起始/结束地址为单位进行设置。

如果在末尾数据发送后两个状态帧（ST1 和 ST2）都指示 ACK ，则 V850ES/Kx2 单片机固件自动执行内部验证。 因此，必须发送该内部验证的状态命令。

编程命令执行以后，建议执行安全设置命令以禁止读出，并将其作为编程器规范的一项缺省设置。

5.7.2　命令帧和状态帧

图 5-13 显示了编程命令的命令帧格式，图 5-14 显示了该命令的状态帧：

图 5-13. 编程命令帧（自编程器至 V850ES/Kx2 单片机）

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

备注　SAH、SAM、SAL：写入起始地址。
　　　EAH、EAM、EAL：写入结束地址。

图 5-14. 编程命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 显示了包含写入数据的帧格式， 图 5-16 显示了该数据的状态帧：

图 5-15. 写入数据的帧（自编程器至 V850ES/Kx2 单片机）

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

备注　写入数据：被写入的用户程序。

图 5-16. 数据帧的状态帧（自 V850ES/Kx2 单片机至编程器）

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　所有数据传输完毕和状态帧

图 5-17 显示了所有数据传输完毕后的状态帧：

图 5-17. 所有数据传输完毕后的状态帧（自 **V850ES/Kx2** 单片机至编程器）

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

备注　ST1 (c)：内部验证结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.9 编程命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.9 编程命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.9 编程命令**。

5.8　验证命令

5.8.1　描述说明

该命令用于比较由编程器发送的数据和自 V850ES/Kx2 单片机指定地址范围内读取的数据，并检查它们是否匹配。
验证开始/结束地址只能以块起始/结束地址为单位进行设置。

5.8.2　命令帧和状态帧

图 5-18 显示了验证命令的命令帧格式， 图 5-19 显示了该命令的状态帧：

图 5-18. 验证命令帧（自编程器至 V850ES/Kx2 单片机）

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

备注　SAH、 SAM、 SAL：验证起始地址。
EAH、 EAM、 EAL：验证结束地址。

图 5-19. 验证命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 显示了包含验证数据的帧格式，图 5-21 显示了该数据的状态帧：

图 5-20. 验证数据的数据帧（自编程器至 V850ES/Kx2 单片机）

STX	LEN	数据	SUM	ETX/ETB
02H	00H 至 FFH (00H = 256)	验证数据	校验和	03H/17H

备注　验证数据： 被验证的用户程序。

图 **5-21. 数据帧的状态帧（自 V850ES/Kx2** 单片机至编程器）

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)：验证结果[注]

注　即使在指定地址范围内发生验证错误，ACK 也总是作为验证结果被返回。所有验证错误的状态都反映到末尾数据的验证结果上。因此，仅当所有指定地址范围的验证处理完成后，才能检查到所发生的验证错误。

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.10 验证命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.10 验证命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.10 验证命令**。

5.9 块空白检查命令

5.9.1 描述说明

该命令用于检查 flash 存储器中指定块编号的那个块是否空白（擦除状态）。

5.9.2 命令帧和状态帧

图 5-22 显示了块空白检查命令的命令帧格式，图 5-23 显示了该命令的状态帧：

图 5-22. 块空白检查命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	32H (块空白检查)	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注 SAH、 SAM、SAL： 块空白检查起始地址（所指定的块起始地址）。
EAH、 EAM、EAL： 块空白检查结束地址（所指定的块结束地址）。

图 5-23. 块空白检查命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注 ST1：块空白检查结果

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.11 块空白检查命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.11 块空白检查命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.11 块空白检查命令**。

5.10 ‘硅标记’命令

5.10.1 描述说明

该命令用于读取器件的写入协议信息（‘硅标记’）。

若编程器支持 V850ES/Kx2 单片机中所不支持的编程协议，可根据第二以及第三字节的值执行该命令来选择恰当的协议。

5.10.2 命令帧和状态帧

图 5-24 显示了 ‘硅标记’命令的命令帧格式， 图 5-25 显示了该命令的状态帧：

图 5-24. ‘硅标记’命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	SUM	ETX
01H	01H	C0H （‘硅标记’）	校验和	03H

图 5-25. ‘硅标记’命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 ‘硅标记’数据帧

图 5-26 显示了包含‘硅标记’数据的帧格式：

图 5-26. ‘硅标记’数据帧（自 V850ES/Kx2 单片机至编程器）

STX	LEN	数据							SUM	ETX
02H	n	VEN	EXT	MSC	DEC	无效数据	SCF	BOT	校验和	03H

备注　**1.** n (LEN)：　数据长度

VEN：　厂家代码 (NEC：10H)

EXT：　扩展代码

MSC：　模块功能代码

DEC：　器件扩展代码

INVALID DATA：　13 字节长度的无效数据

SCF：　安全标志设置

BOT：　引导模块编号

2. 对于厂家代码（VEN）、扩展代码（EXT）、以及模块功能代码（FNC），其最高位都用作奇校验位。举例显示如下：

表 5-1. ‘硅标记’数据举例

字段	内容	长度 (字节)	‘硅标记’数据注举例	实际值
VEN	厂家代码(NEC)	1	10H (00010000B)	10H
EXT	扩展代码(固定)	1	7FH (01111111B)	7FH
MSC	模块功能代码(固定)	1	01H (00000001B)	01H
DEC	器件扩展代码(固定)	1	FEH (11111110B)	7EH
INVALID DATA	无效数据	13	–	–
SCF	安全设置	1	7FH (01111111B)	7FH
BOT	引导模块编号	1	00H (无奇偶校验)	00H

注　　阴影位代表奇校验位（该值用来调整字节中“1”的数量）。

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.12　‘硅标记’命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.12　‘硅标记’命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.12　‘硅标记’命令**。

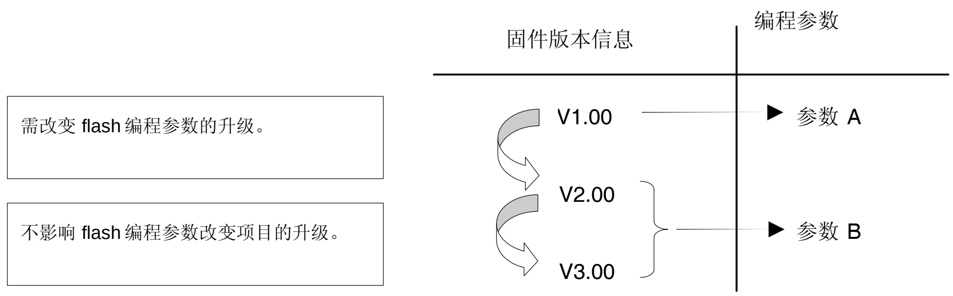
5.11 获取版本信息命令

5.11.1 描述说明

该命令用于获取有关 V850ES/Kx2 单片机的器件版本信息和固件版本信息。
当编程参数必须随固件版本信息而改变时，使用该命令。

注意事项　在不影响 flash 编程参数改变的固件更新期间，固件版本信息可能被修改（此时，不通告固件版本信息的修改）。

例如：　固件版本信息和重新编程参数



5.11.2 命令帧和状态帧

图 5-28 显示了获取版本信息命令的命令帧格式， 图 5-29 显示了该命令的状态帧：

图 5-28. 获取版本信息命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	SUM	ETX
01H	01H	C5H (获取版本信息)	校验和	03H

图 5-29. 获取版本信息命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 显示了版本数据的数据帧：

图 5-30. 版本数据帧（自 V850ES/Kx2 单片机至编程器）

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

备注　DV1：器件版本信息的整数部分

DV2：器件版本信息的第一个小数点位置

DV3：器件版本信息的第二个小数点位置

FV1：固件版本信息的整数部分

FV2：固件版本信息的第一个小数点位置

FV3：固件版本信息的第二个小数点位置

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.13 获取版本信息命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.13 获取版本信息命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.13 获取版本信息命令**。

5.12 校验和命令

5.12.1 描述说明

该命令用于获取指定区域的校验和数据。

关于校验和计算的起始/结束地址，即从 flash 存储器的顶部开始以块为单位（2 KB）所指定的一个固定地址。

校验和数据通过从初始值（00H）中以字节为单位连续减去指定地址范围内的数据而得到。

5.12.2 命令帧和状态帧

图 5-31 显示了校验和命令的命令帧格式， 图 5-32 显示了该命令的状态帧：

图 5-31. 校验和命令帧（自编程器至 V850ES/Kx2 单片机）

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. 校验和命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 显示了包含校验和数据的帧格式：

图 5-33. 校验和数据帧（自 V850ES/Kx2 单片机至编程器）

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

备注　CK1：校验和数据的高 8 位

　　　CK2：校验和数据的低 8 位

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.14 校验和命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.14 校验和命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.14 校验和命令**。

5.13 安全设置命令

5.13.1 描述说明

该命令用于执行安全设置（写入、块擦除、以及芯片擦除的允许或禁止）。用该命令执行这些设置，能够限制未经授权就对 flash 存储器进行重写。

执行编程命令后，在编程器规格说明中，建议执行安全设置命令使读禁止，并将其作为一项缺省设置。

注意事项　安全设置一旦完成，将不能再次设置安全标志或者由禁止改变为允许。如果试图进行这样的设置，则将会产生一个保护错误。为了重设安全标志，所有的安全标志必须通过执行芯片擦除命令（块擦除命令不能够用来初始化安全标志）来初始化。 但是，如果芯片擦除已被禁止，则无法擦除芯片自身，因此，此设置将不能从编程器中抹去。 所以，建议芯片禁止擦除之前，本编程器规范中应该对安全设置执行进行再确认。

5.13.2 命令帧和状态帧

图 5-34 显示了安全设置命令的命令帧格式，图 5-35 显示了该命令的状态帧：

安全设置命令帧包含块编号字段和页编号字段，但这些字段没有任何特殊用途，因此这些字段设置为 00H。

图 5-34. 安全设置命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息		SUM	ETX
01H	03H	A0H (安全设置)	00H (固定)	00H (固定)	校验和	03H

备注　　BLK、PAG：固定为 00H

图 5-35. 安全设置命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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 显示了安全数据帧的格式，图 5-37 显示了该数据的状态帧：

图 5-36 安全数据帧（自编程器至 V850ES/Kx2 单片机）

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

备注　　FLG：安全标志
　　　　BOT：引导模块编号
　　　　（当 FLG 中的位 4 = 1 时，其值固定为 00H。当该位为 0 时，其值任意）。

图 5-37. 安全数据写入状态帧（自 V850ES/Kx2 单片机至编程器）

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 内部验证检查和状态帧

图 5-38 显示了内部验证检查的状态帧：

图 5-38. 内部验证检查的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注　　ST1 (c)：内部验证结果

下表显示了安全标志字段中的内容：

表 5-2. 安全标志字段中的内容

名称	内容
位 7	固定为“1”
位 6	
位 5	
位 4	引导模块簇重写禁止标志 (1： 重写允许， 0： 重写禁止)
位 3	读取禁止标志 (1：读取允许， 0： 读取禁止)
位 2	编程禁止标志 (1：编程允许， 0： 编程禁止)
位 1	块擦除禁止标志 (1： 块擦除允许， 0： 块擦除禁止)
位 0	芯片擦除禁止标志 (1：芯片擦除允许， 0： 芯片擦除禁止)

下表显示了安全标志字段设置和各工作模式中允许/禁止状态之间的关系：

表 5-3. 安全标志字段和各工作模式中允许/禁止状态

<table>
<tr><td>工作模式</td><td colspan="4">Flash 存储器编程模式</td><td>自编程模式</td></tr>
<tr><td rowspan="2">命令
安全设置名称</td><td colspan="4">安全设置后的命令工作状态：
√：　可执行，×：　不可执行
△：引导区中写入和块擦除命令不可执行</td><td rowspan="6">• 所有命令都能执行，而与安全设置值无关
• 可以保留安全设置值以及从允许改变为禁止</td></tr>
<tr><td>编程</td><td>芯片擦除</td><td>块擦除</td><td>读取</td></tr>
<tr><td>编程禁止</td><td>×</td><td>√</td><td>×</td><td>√</td></tr>
<tr><td>芯片擦除禁止</td><td>√</td><td>×</td><td>×</td><td>√</td></tr>
<tr><td>块擦除禁止</td><td>√</td><td>√</td><td>×</td><td>√</td></tr>
<tr><td>读取禁止</td><td>√</td><td>√</td><td>√</td><td>×</td></tr>
<tr><td>引导模块重写禁止</td><td>△</td><td>×</td><td>△</td><td>√</td><td>处于与 flash 存储器编程模式时相同的状态 (板内/板外编程)</td></tr>
</table>

关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.15 安全设置命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS）时，阅读 **7.15 安全设置命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.15 安全设置命令**。

5.14 读取命令

5.14.1 描述说明

该命令用于从 V850ES/Kx2 单片机的 flash 存储器中读取数据。

写入开始/结束地址只能以块起始/结束地址为单位进行设置。

5.14.2 命令帧和状态帧

图 5-39 显示了读取命令的命令帧格式，图 5-40 显示了该命令的状态帧：

图 5-39. 读取命令帧（自编程器至 V850ES/Kx2 单片机）

SOH	LEN	COM	命令信息						SUM	ETX
01H	07H	50H (读取)	SAH	SAM	SAL	EAH	EAM	EAL	校验和	03H

备注　SAH、SAM、 SAL：读取起始地址（块的起始地址）
　　　EAH、EAM、 EAL：读取结束地址（块的结束地址）

图 5-40. 读取命令的状态帧（自 V850ES/Kx2 单片机至编程器）

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

备注　ST1 (a)：命令接收结果

5.14.3 数据帧和状态帧

图 5-41 显示了包含读取数据的帧格式，图 5-42 显示了该数据的状态帧：

图 5-41. 读取数据的数据帧（自编程器至 V850ES/Kx2 单片机）

STX	LEN	数据	SUM	ETX/ETB
02H	00H 至 FFH (00H = 256)	读取数据	校验和	03H/17H

备注　读取数据：读取自 V850ES/Kx2 单片机中的数据

图 5-42. 读取数据的状态帧（自 V850ES/Kx2 单片机至编程器）

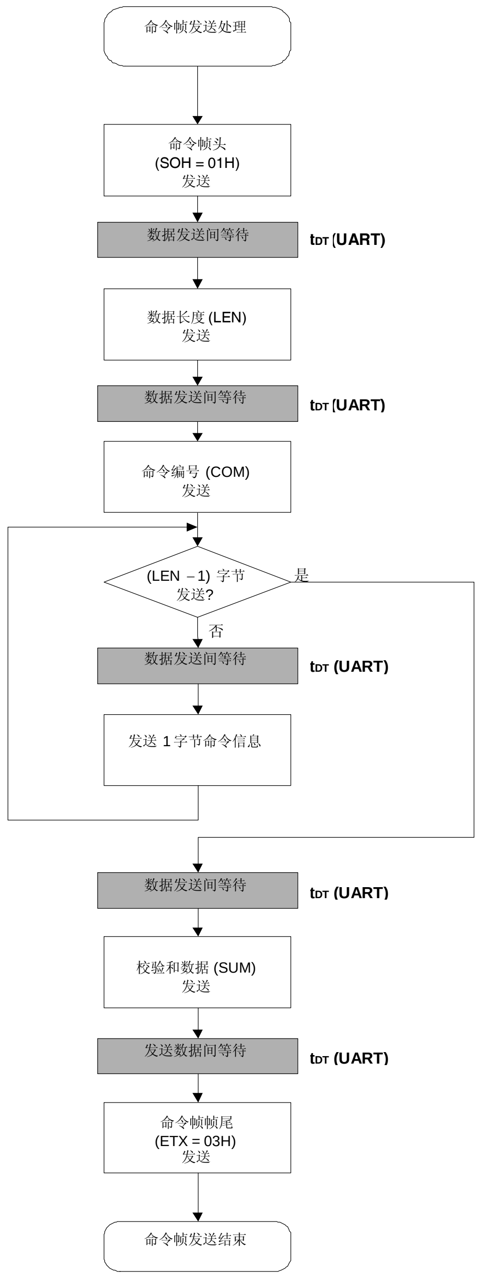
STX	LEN	数据	SUM	ETX
02H	00H 至 FFH (00H = 256)	ST1 (b)	校验和	03H/17H

备注　ST1 (b)：为了读取数据，由编程器发送 ACK（06H）或 NACK（15H）。

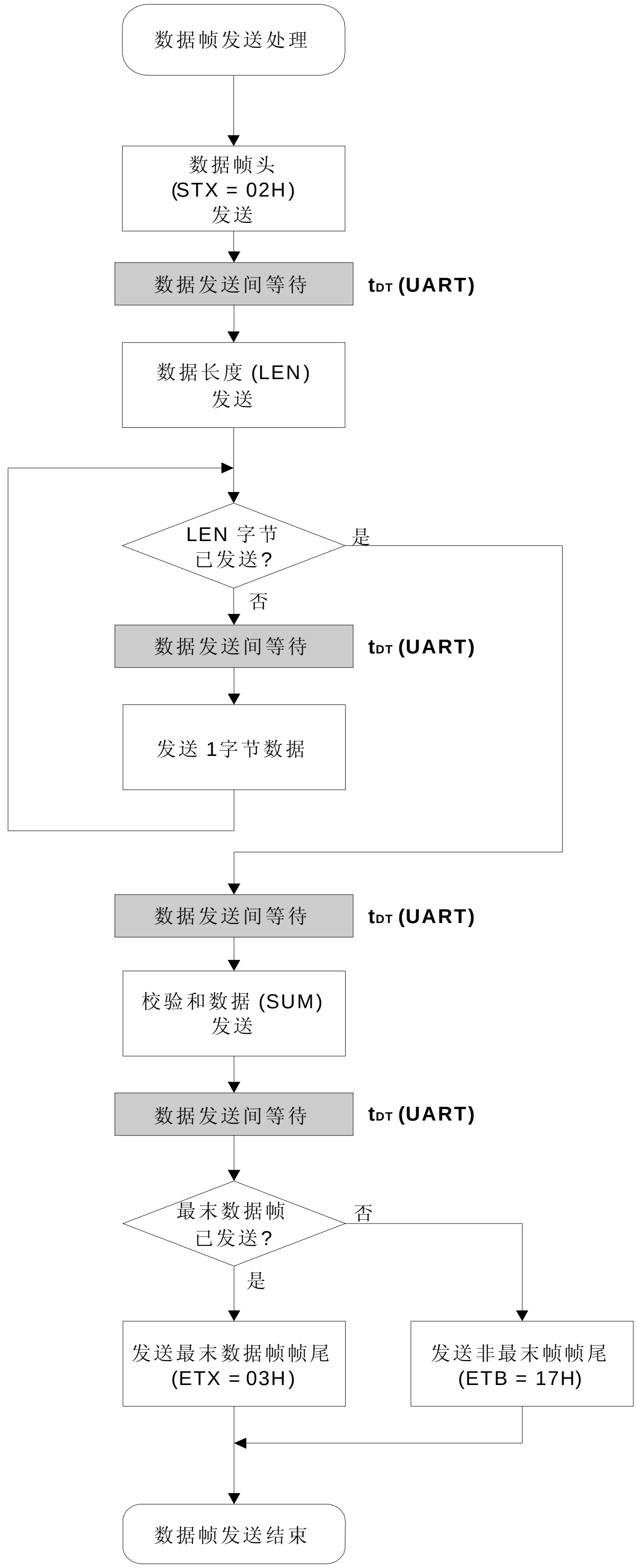
关于各通信模式时编程器和 V850ES/Kx2 单片机之间的处理顺序流程图、命令处理流程图、以及举例程序之详情，请阅读下述章节：

- 关于 UART 通信模式，阅读 **6.16　读取命令**。
- 关于支持握手功能的 3 线串行 I/O 通信模式（CSI + HS），阅读 **7.16　读取命令**。
- 关于 3 线串行 I/O 通信模式（CSI），阅读 **8.16 读取命令**。

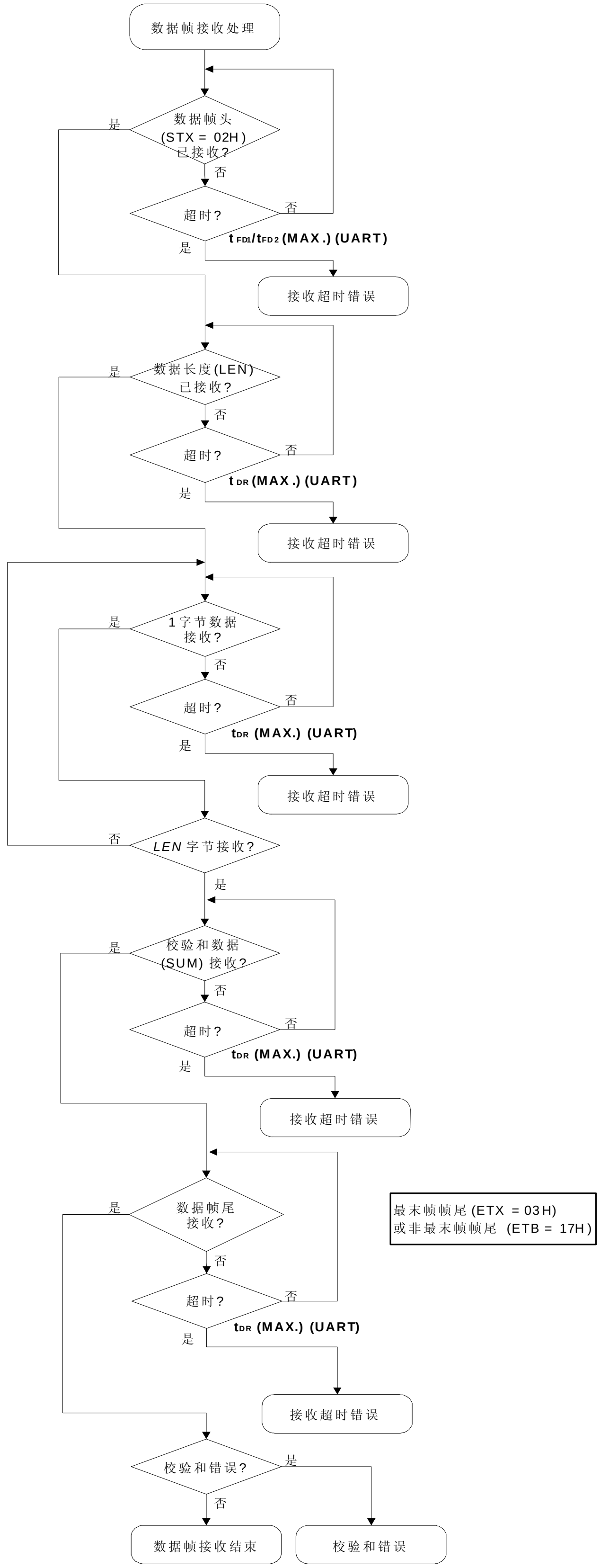
6.1　命令帧发送处理流程图



6.2 数据帧发送处理流程图



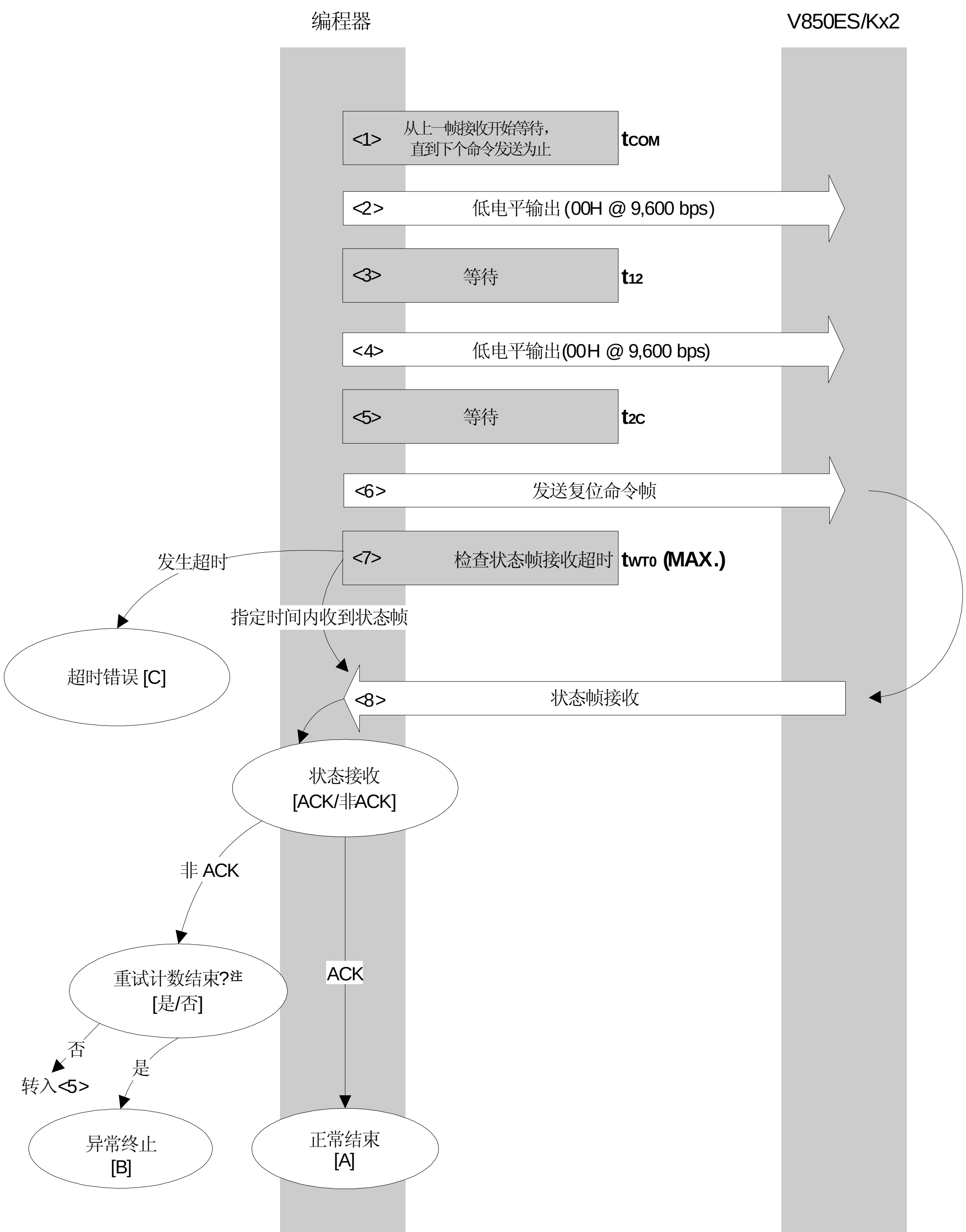
6.3　数据帧接收处理流程图



6.4 复位命令

6.4.1 处理程序流程图

复位命令处理程序



注 不要超过复位命令发送的重试计数。 (最大到 16 次)

6.4.2　处理程序的描述说明

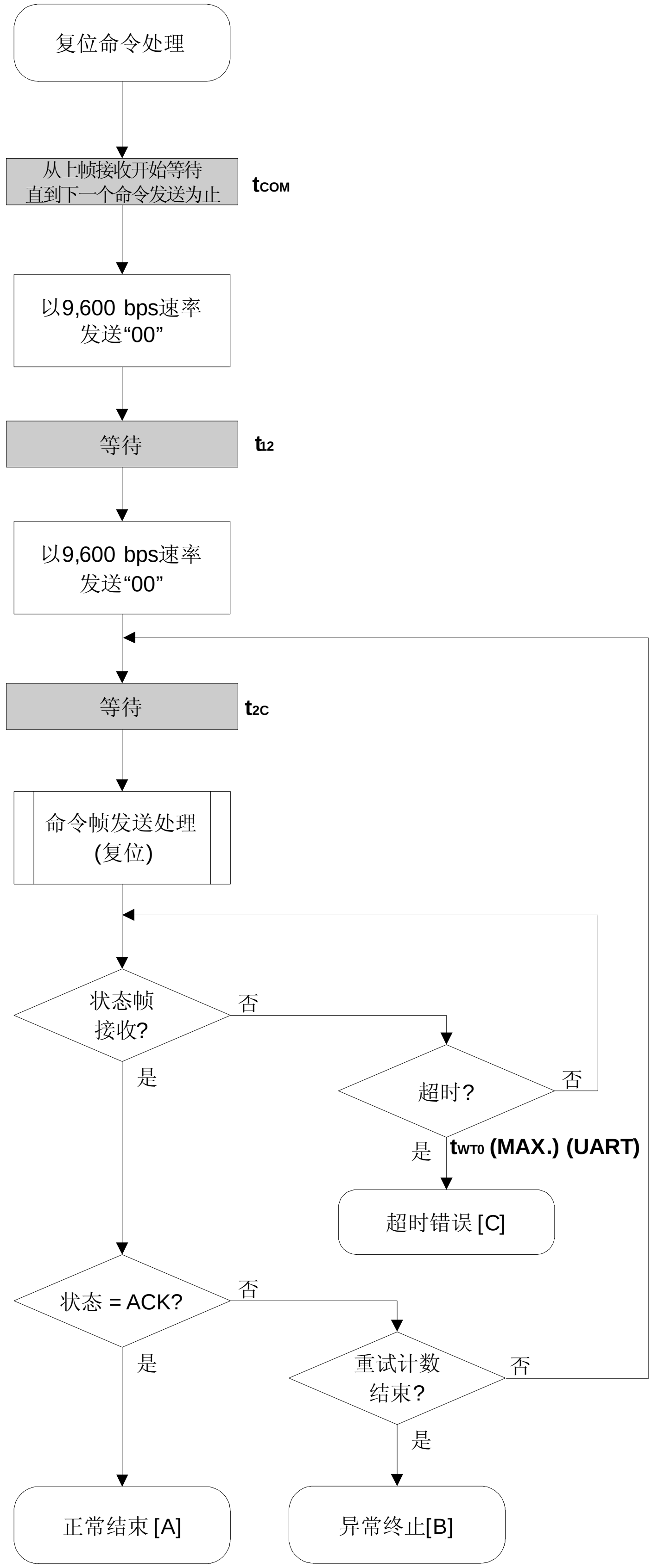
<1>　从上一个帧接收开始等待直到下一个命令处理开始为止(等待时间 t_{COM})。
<2>　输出低电平(以 9,600 bps 发送数据 00H)。
<3>　等待状态 (等待时间 t_{12})。
<4>　输出低电平(以 9,600 bps 发送数据 00H)。
<5>　等待状态(等待时间 t_{2C})。
<6>　命令帧发送处理发送复位命令。
<7>　从命令发送开始执行超时检查直到状态帧接收为止。
　　　如果发生超时，超时错误[C]被返回。 (超时时间 t_{WT0}(MAX.))
<8>　检查状态码。

　　　当 ST1 = ACK 时：　正常完成 [A]
　　　当 ST1 ≠ ACK 时：　检查重试计数(t_{RS})
　　　　　　　　　　　　如果没有超出重试计数，程序从<5> 重复执行。
　　　　　　　　　　　　如果超出重试计数, 处理异常终止 [B]。

6.4.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且编程器和 V850ES/Kx2 之间建立同步。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	• 处理过程中接收到一个除状态命令之外的其他命令。 • 命令帧数据异常(例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定时间内没有收到状态帧。

6.4.4　流程图


复位命令处理
从上帧接收开始等待
直到下一个命令发送为止
t_{COM}
以9,600 bps速率
发送“00”
等待
t_{12}
以9,600 bps速率
发送“00”
等待
t_{2C}
命令帧发送处理
(复位)
状态帧
接收?
否
是
超时?
否
是
t_{WT0} (MAX.) (UART)
超时错误 [C]
状态 = ACK?
否
是
重试计数
结束?
否
是
正常结束 [A]
异常终止[B]

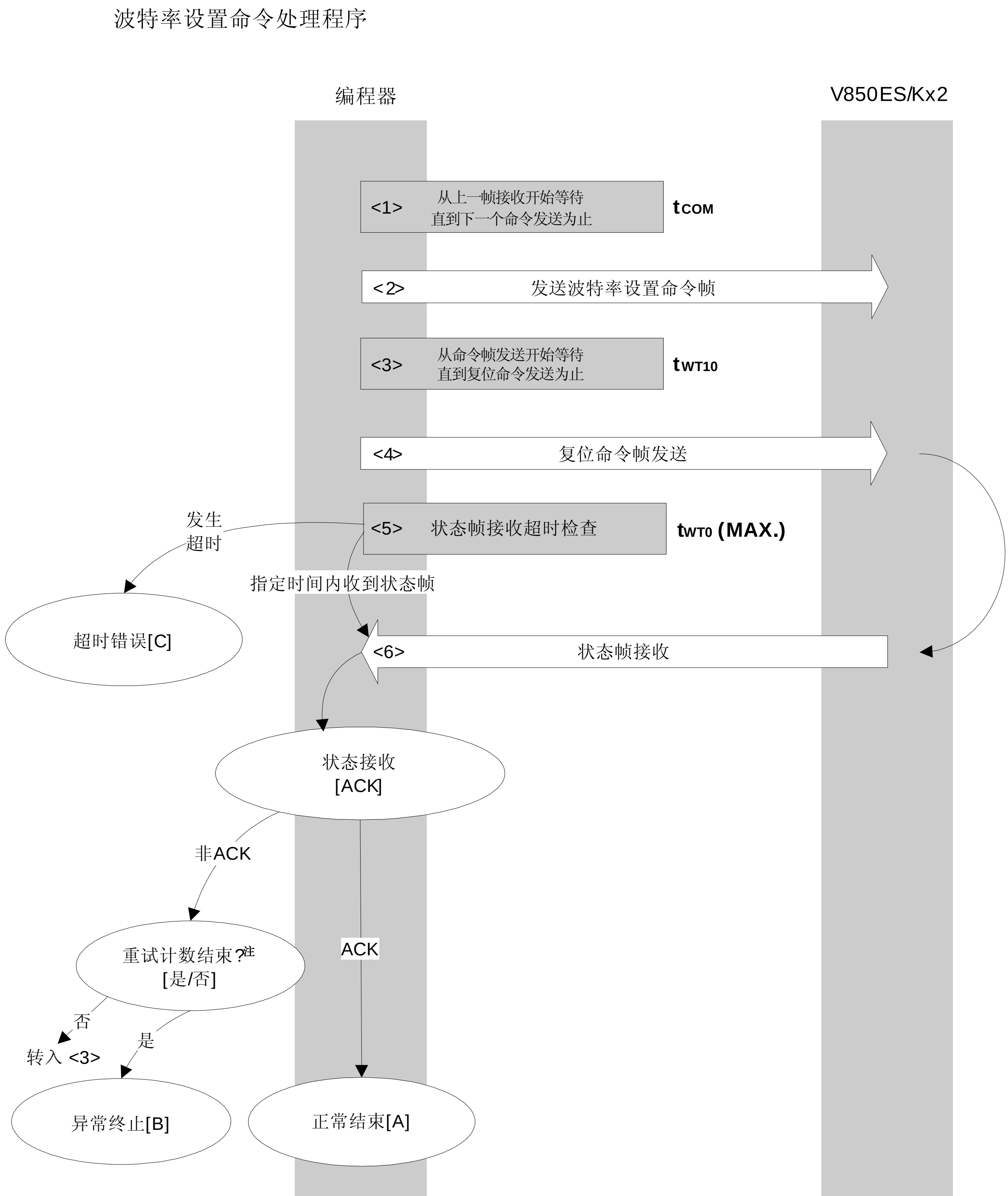
6.4.5 程序举例说明

以下是复位命令处理的程序举例说明：

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

      set_uart0_br(BR_9600);   // 改变为 9600bps

      fl_wait(tCOM);           // 等待
      putc_ua(0x00);           //以 9600bps 发送 0x00

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

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

            fl_wait(t2C);      //等待

            put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);      //发送复位命令

            rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
            if (rc == FLC_DFTO_ERR) // 超时错误 ?
                  break;                  //是      // 情况 [C]

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

6.5 波特率设置命令

6.5.1 处理程序流程图



注　不要超出复位命令发送的重试计数。(最大到 16 次)

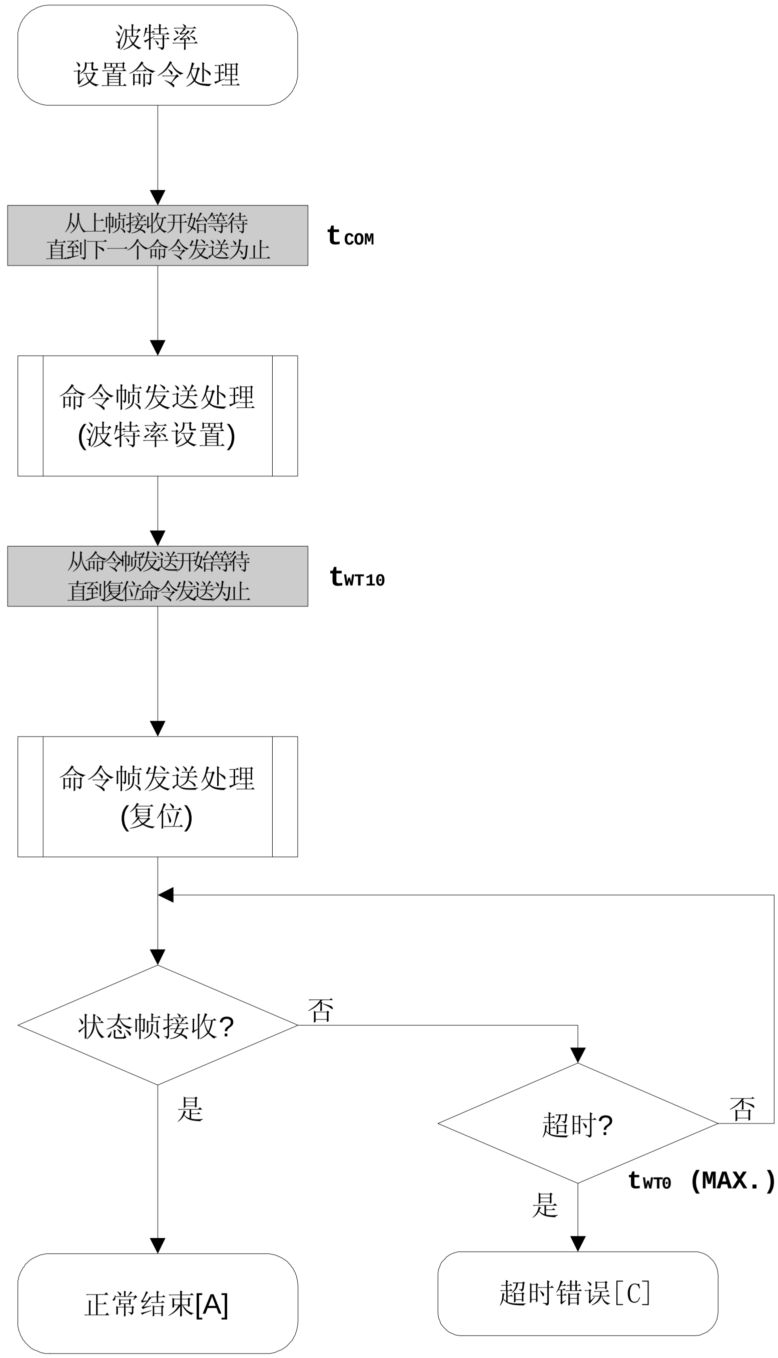
6.5.2　处理程序的描述说明

<1>　从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2>　命令帧发送处理发送波特率设置命令。
<3>　从命令发送开始等待直到复位命令发送为止(等待时间 t_{WT10})。
<4>　命令帧发送处理发送复位命令。
<5>　从命令发送开始执行超时检查直到状态帧接收为止。
　　如果发生超时，超时错误[C]被返回。 (超时时间 t_{WT0}(MAX.))
<6>　如果状态码是 ACK, 处理正常结束 [A]。

6.5.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常(ACK)	06H	命令正常执行，而且建立编程器和 V850ES/Kx2 之间的 UART 通信速度同步。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常(例如不正确的数据长度(LEN) 或没有 ETX)。
超时错误 [C]		–	数据帧接收超时 对于 V850ES/Kx2,该命令还会导致如下错误： • 命令信息(参数)无效。 • 命令帧包括校验和错误。 • 命令帧的数据长度 (LEN)不正确。 • 命令帧帧尾 (ETX)缺失。 • 设置波特率和接收命令帧数据 16 次后没有检测到复位命令。

6.5.4　流程图


波特率
设置命令处理
从上帧接收开始等待
直到下一个命令发送为止
tCOM
命令帧发送处理
(波特率设置)
从命令帧发送开始等待
直到复位命令发送为止
tWT10
命令帧发送处理
(复位)
状态帧接收?
否
是
超时?
否
tWT0 (MAX.)
是
正常结束[A]
超时错误[C]

6.5.5 程序举例说明

以下是波特率设置命令处理的程序举例说明：

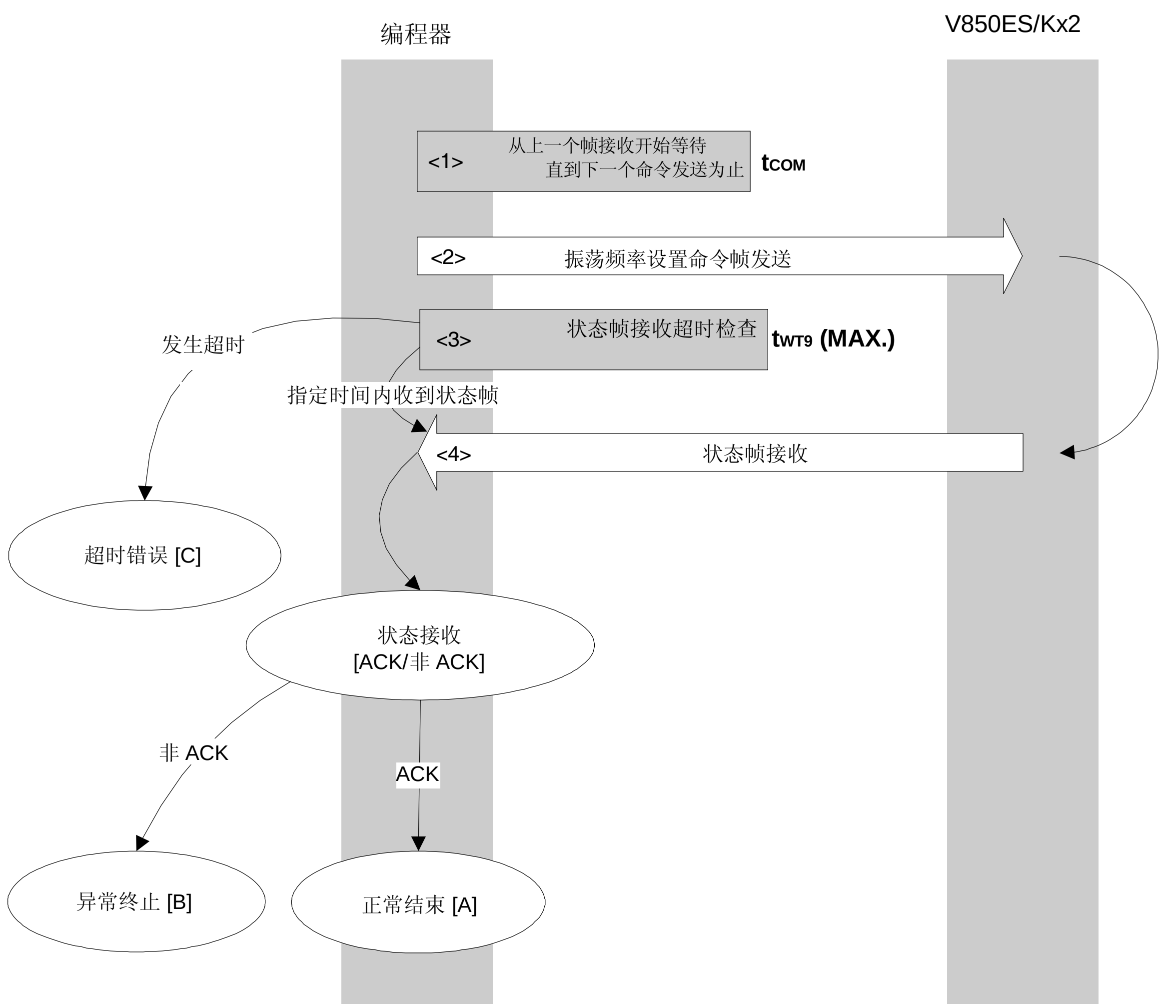
```
/***************************************************************/
/*                                                              */
/* 设置波特率命令                                               */
/*                                                              */
/***************************************************************/
/* [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);      // 发送命令前等待
        put_cmd_ua(FL_COM_SET_BAUDRATE, 2, fl_cmd_prm); // 发送"波特率设置"命令

        set_flbaud(brid);                   // 改变波特率
        set_uart0_br(brid);                 //改变波特率(h.w.)

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

                put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);        // 发送复位命令

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

```
            break;          // 获得 ACK !!

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

    return rc;
}
```

6.6　振荡频率设置命令

6.6.1　处理程序流程图

振荡频率设置命令处理程序



6.6.2　处理程序的描述说明

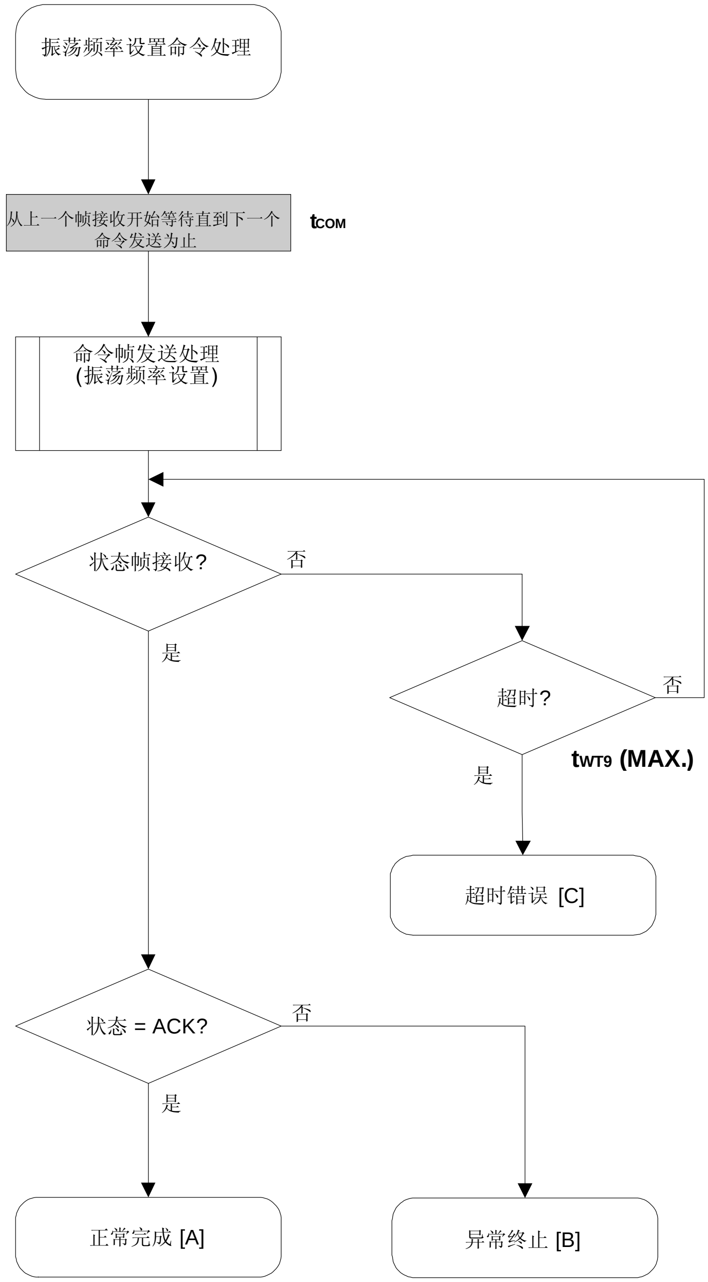
<1>　从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2>　命令帧发送处理发送振荡频率设置命令。
<3>　从命令发送开始执行超时检查直到状态帧接收为止。
　　如果发生超时，超时错误[C]被返回。(超时时间 t_{WT9}(MAX.))
<4>　检查状态码。

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

6.6.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成[A]	确认正常(ACK)	06H	命令正常执行，操作频率被正确设置到 V850ES/Kx2。
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。
	确认异常 (NACK)	15H	命令帧数据异常(例如不正确的数据长度(LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧。

6.6.4　流程图


振荡频率设置命令处理
从上一个帧接收开始等待直到下一个命令发送为止
t_{COM}
命令帧发送处理
(振荡频率设置)
状态帧接收?
否
是
超时?
否
t_{WT9} (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
正常完成 [A]
异常终止 [B]

6.6.5 程序举例说明

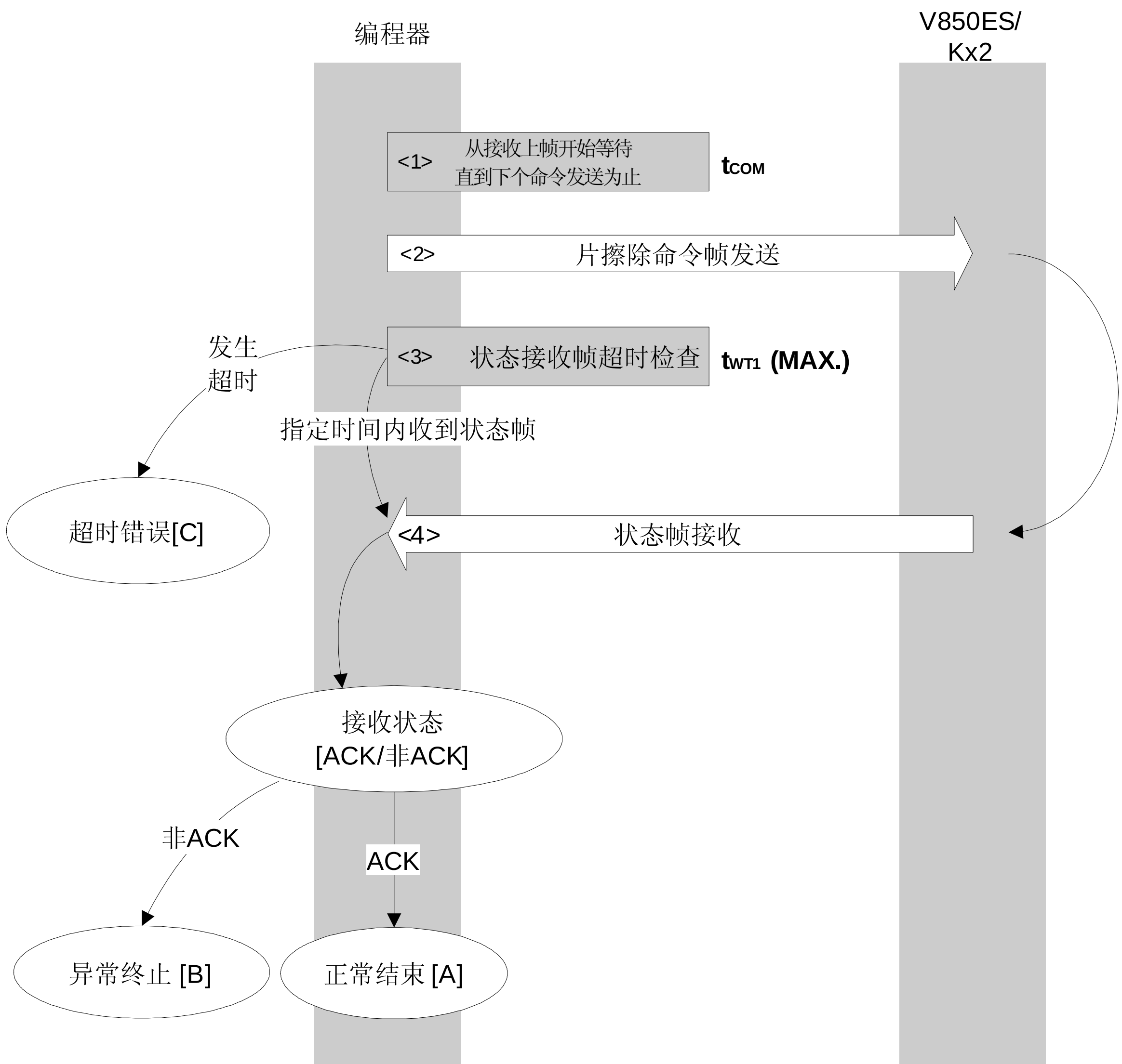
以下是振荡频率设置命令处理的程序举例说明：

```
/******************************************************************/
/*                                                                */
/* 设置 Flash 存储器器件时钟值命令                                    */
/*                                                                */
/******************************************************************/
/* [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);              // 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);    // 获得状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:     return  rc;     break;  // 情况 [A]
//              case    FLC_DFTO_ERR:           return  rc;     break;  // 情况 [C]
//              default:                return  rc;     break;  // 情况 [B]
//      }

        return rc;
}
```

6.7 片擦除命令

6.7.1 处理程序流程图

片擦除命令处理程序



6.7.2 处理程序的描述说明

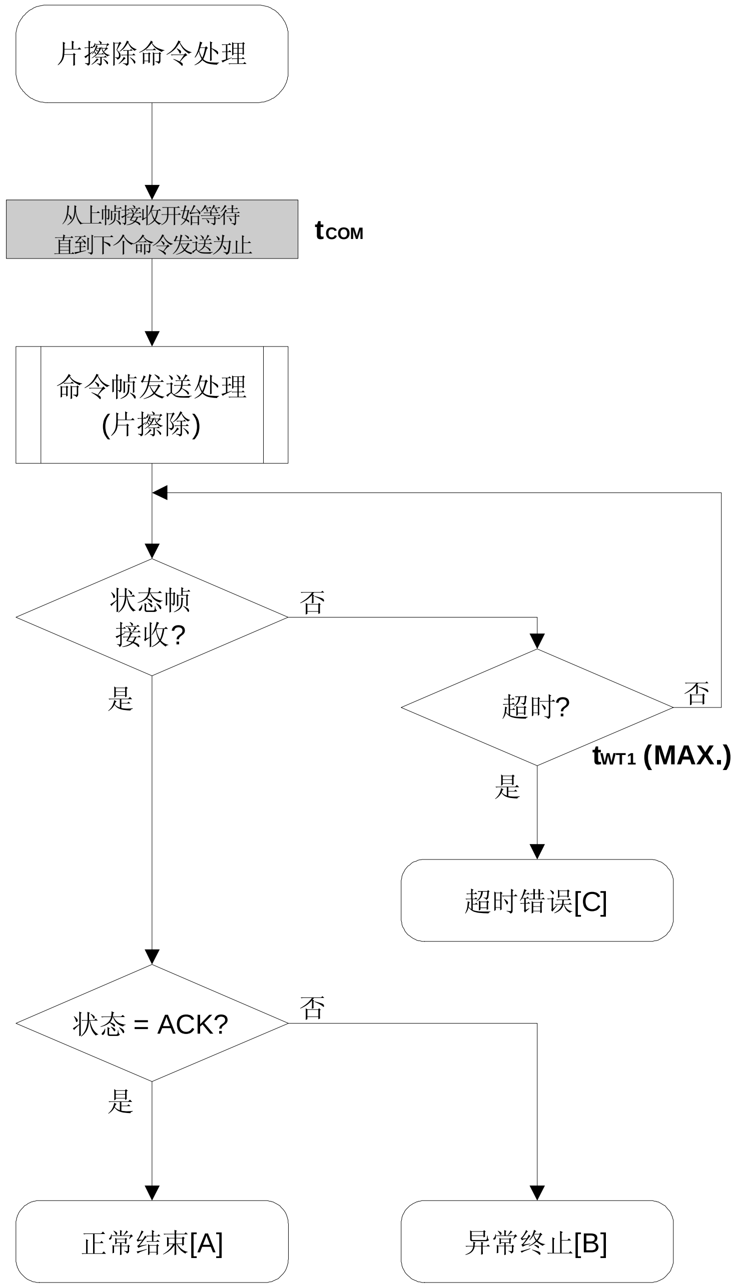
<1> 从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2> 命令帧发送处理发送片擦除命令。
<3> 从命令发送开始执行超时检查直到状态帧被接收为止。
如果发生超时错误，超时错误 [C] 被返回 (超时时间 t_{WT1}(MAX.))。
<4> 检查状态码。

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

6.7.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常(ACK)	06H	命令正常执行，片擦除被正常完成。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中片擦除被禁止。 • 在安全设置中引导区重写被禁止。
	确认异常(NACK)	15H	命令帧数据异常(例如不正确的数据长度(LEN) 或没有 ETX)。
	FLMD 错误	18H	发生擦除错误。
	写错误	1CH	
	MRG10 错误	1AH	
	MRG11 错误	1BH	
超时错误 [C]		–	在指定的时间内没有收到状态帧。

6.7.4 流程图


片擦除命令处理
从上帧接收开始等待
直到下个命令发送为止
tCOM
命令帧发送处理
(片擦除)
状态帧
接收?
否
是
超时?
否
tWT1 (MAX.)
是
超时错误[C]
状态 = ACK?
否
是
正常结束[A]
异常终止[B]

6.7.5 程序举例说明

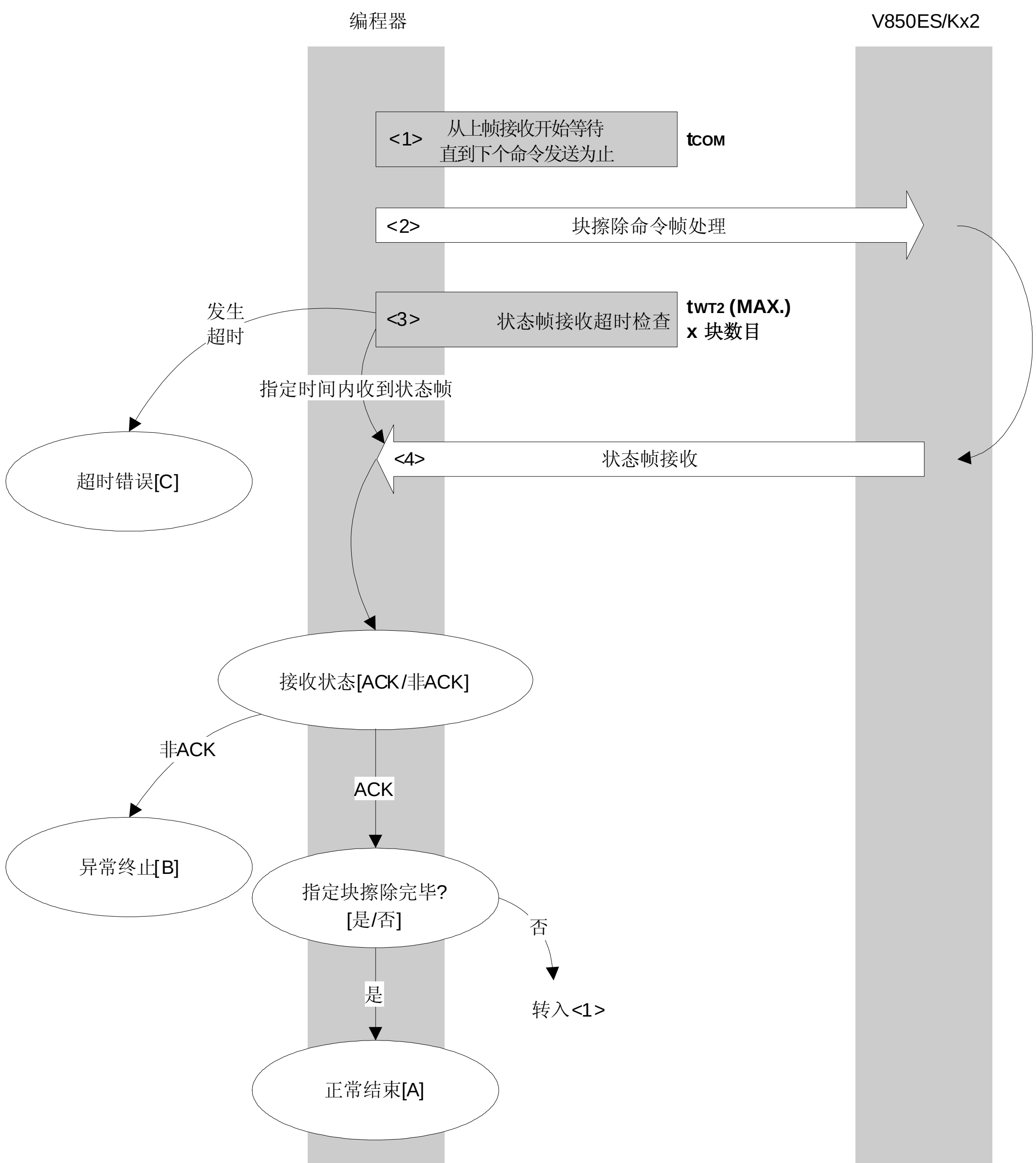
以下是片擦除命令处理的程序举例说明：

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


        fl_wait(tCOM);                  // 发送命令前等待

        put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // 发送擦除片命令

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

}
```

6.8　块擦除命令

6.8.1　处理程序流程图

块擦除命令处理程序



6.8.2 处理程序的描述说明

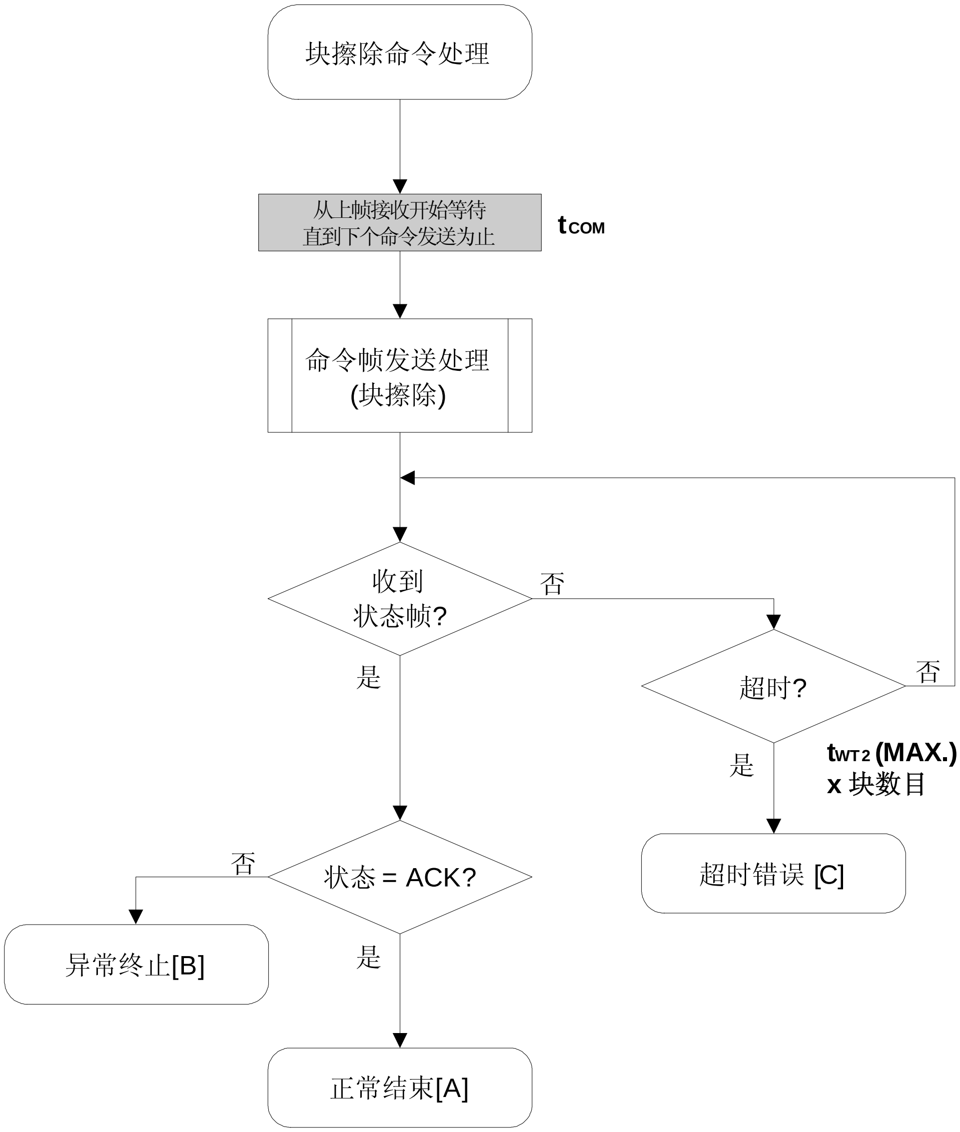
<1> 从上一个帧接收开始等待直到下一个命令发送为止。 (等待时间 t_{COM})
<2> 命令帧发送处理发送块擦除命令。
<3> 命令发送后开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误[C] 被返回(超时时间 t_{WT2}(MAX.) × 块的数目)。
<4> 检查状态码。

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

6.8.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，块擦除被正常完成。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址超出 Flash 存储器 范围。 • 指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中块擦除被禁止。 • 在安全设置中引导块重写被禁止时，引导区处于指定范围内。 • 在安全设置中片擦除被禁止。 • 在安全设置中编程被禁止。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度(LEN) 或没有 ETX)。
	MRG10 错误	1AH	发生擦除错误。
超时错误 [C]		–	在指定的时间内没有收到状态帧。

6.8.4　流程图


块擦除命令处理
从上帧接收开始等待
直到下个命令发送为止
tCOM
命令帧发送处理
(块擦除)
收到
状态帧?
否
是
超时?
否
是
tWT2 (MAX.)
x 块数目
超时错误 [C]
状态 = ACK?
否
是
异常终止[B]
正常结束[A]

6.8.5　程序举例说明

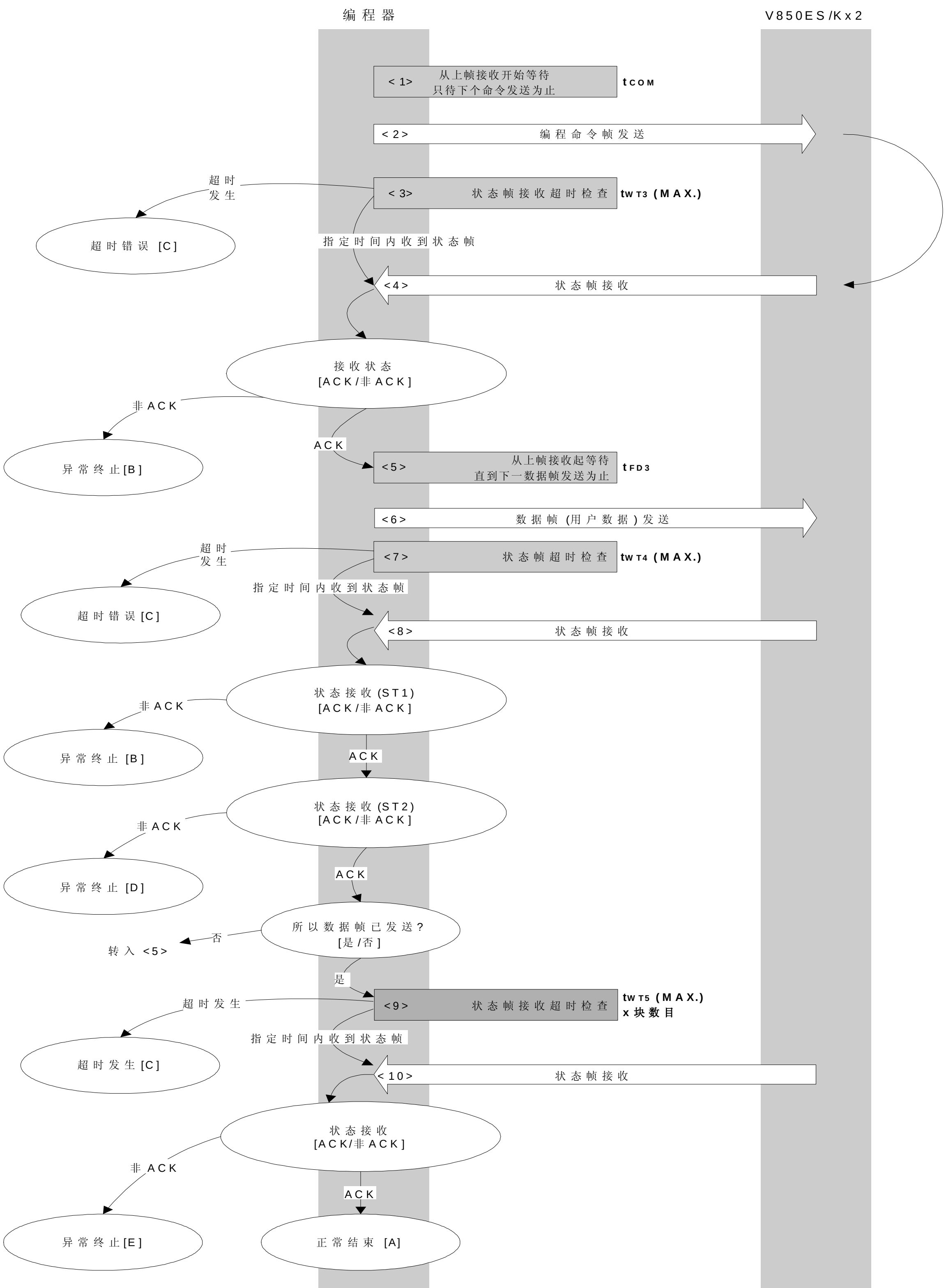
以下是对一个块的块擦除命令处理的程序举例说明：

```
/****************************************************************/
/*                                                              */
/*  块擦除命令                                                  */
/*                                                              */
/****************************************************************/
/*  [i] u16 sblk        ...开始块擦除(0...255)                  */
/*  [i] u16 eblk        ...结束块擦除(0...255)                  */
/*  [r] u16             ... 错误码                              */
/****************************************************************/
u16     fl_ua_erase_blk(u16 sblk, u16 eblk)
{

    u16     rc;
    u16     block_num;
    u32     top, bottom;

    top = get_top_addr(sblk);           // 获得开始块的开始地址
    bottom = get_bottom_addr(eblk);     // 获得结束块的结束地址

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

    block_num = eblk - sblk + 1;

    fl_wait(tCOM);                  // 发送命令前等待

    put_cmd_ua(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm); // 发送擦除片命令

    rc = get_sfrm_ua(fl_ua_sfrm, tWT2_MAX * block_num);  // 获得状态帧

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

    return rc;

}
```

6.9　编程命令

6.9.1　处理程序流程图

编 程 命 令 处 理 程 序



6.9.2 处理程序的描述说明

<1> 从上一个帧接收开始等待直到下一个命令发送为止。(等待时间 t_{COM})
<2> 命令帧发送处理发送编程命令。
<3> 从命令发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误[C] 被返回 (超时时间 t_{WT3}(MAX.))。
<4> 检查状态码。

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

<5> 从上一个帧接收开始等待直到下一个数据帧发送为止 (等待时间 t_{FD3})。
<6> 数据帧发送处理发送用户数据。
<7> 从用户数据发送开始执行超时检查直到数据帧接收为止。
如果发生超时，超时错误[C]被返回(超时时间 t_{WT4}(MAX.))。
<8> 检查状态码(ST1/ST2) (同样参见处理程序流程图和流程图)。

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： 当所有的数据帧发送完成时进入<9> 。
如果仍有剩余数据帧要发送，处理从<5>重复执行程序。
- 当 ST2 ≠ ACK 时： 异常终止 [D]

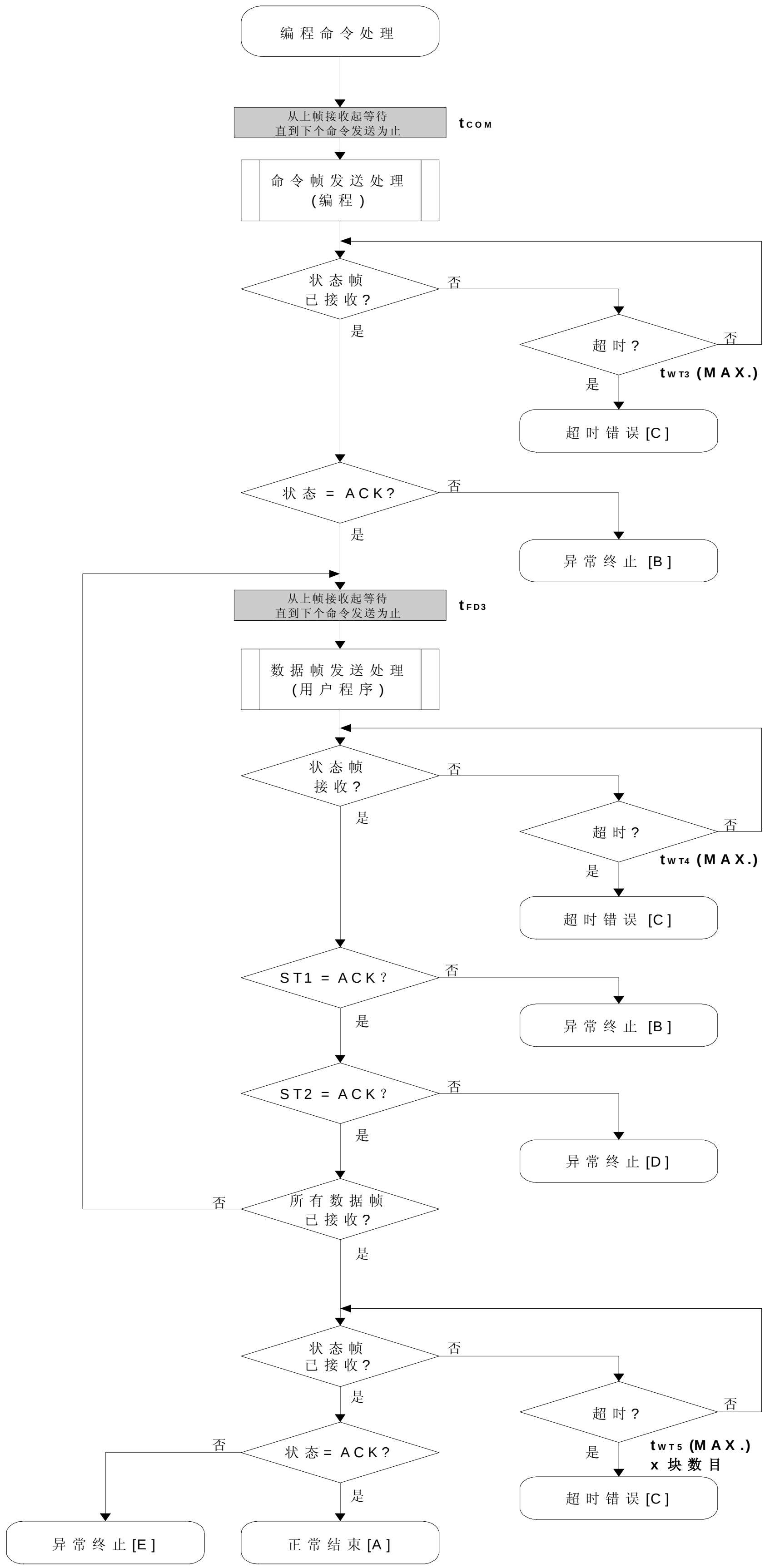
<9> 执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C] 被返回(超时时间 t_{WT5}(MAX.) × 块的数目)。
<10> 检查状态码。

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

6.9.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，用户数据被正常写入。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址不是该块的开始/结束地址。 • 数据长度不是 64 字单位。
	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中写入被禁止。 • 在安全设置中引导块重写被禁止。
	确认异常(NACK)	15H	命令帧数据异常(例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧。
异常终止 [D]	FLMD 错误	18H (ST2)	发生写错误。
	写错误	1CH (ST2)	发生写错误。
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

6.9.4　流程图



编 程 命 令 处 理
从上帧接收起等待
直到下个命令发送为止
tCOM
命 令 帧 发 送 处 理
(编 程)
状 态 帧
已 接 收?
否
是
超 时 ?
否
是
tWT3 (MAX.)
超 时 错 误 [C]
状 态 = ACK?
否
是
异 常 终 止 [B]
从上帧接收起等待
直到下个命令发送为止
tFD3
数 据 帧 发 送 处 理
(用 户 程 序)
状 态 帧
接 收?
否
是
超 时 ?
否
是
tWT4 (MAX.)
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
已 接 收?
否
是
状 态 帧
已 接 收?
否
是
超 时 ?
否
是
tWT5 (MAX.)
x 块 数 目
超 时 错 误 [C]
状 态 = ACK?
否
是
异 常 终 止 [E]
正 常 结 束 [A]

6.9.5 程序举例说明

以下是编程命令处理的程序举例说明：

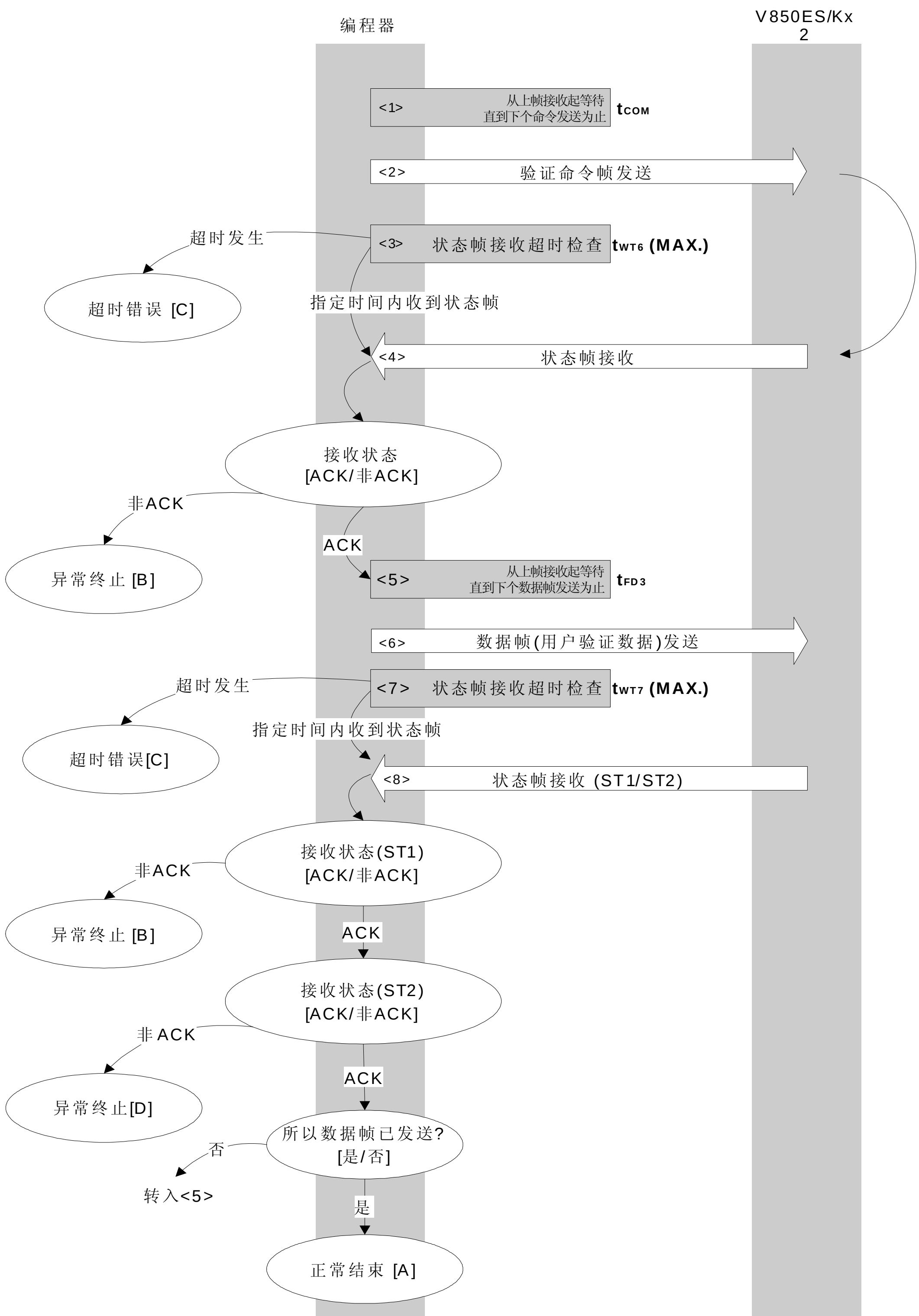
```
/*****************************************************************/
/*                                                               */
/*   写命令                                                      */
/*                                                               */
/*****************************************************************/
/*   [i] u32 top          ... 开始地址                           */
/*   [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;
  u16      block_num;

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

  block_num = get_block_num(top, bottom);    // 获得块编码

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

  put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm); // 发送 "编程" 命令

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

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

  while(1){

        // make send data frame
        if ((bottom - send_head) > 256){     // 剩余长度 > 256 ?
              is_end = false;                // 是，不是结束帧
              send_size  = 256;              // 发送长度= 256 字节
        }
```

```
            else{
                        is_end = true;
                        send_size = bottom - send_head + 1;
                                                        // 发送长度 = (bottom - send_head + 1) 字节

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

            fl_wait(tFD3_UA);                   // 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;    // 情况 [C]
                        default:            return      rc;       break;    // 情况 [B]
            }
            if (fl_st2_ua != FLST_ACK){                 // ST2 = ACK ?
                        rc = decode_status(fl_st2_ua);      //否
                        return      rc;                                 // 情况 [D]
            }
            if (is_end)
                        break;

  }
  /***********************************************/
 /*    检查内部验证                                                      */
  /***********************************************/
  rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX_UA * block_num));// 再次获得状态帧

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

  return rc;
}
```

6.10 验证命令

6.10.1 处理程序流程图

验证命令处理程序



6.10.2 处理程序的描述说明

<1> 从上一个帧接收开始等待直到下一个命令发送为止。(等待时间 t_{COM})
<2> 命令帧发送处理发送验证命令。
<3> 从命令发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT6}(MAX.))。
<4> 检查状态码。

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

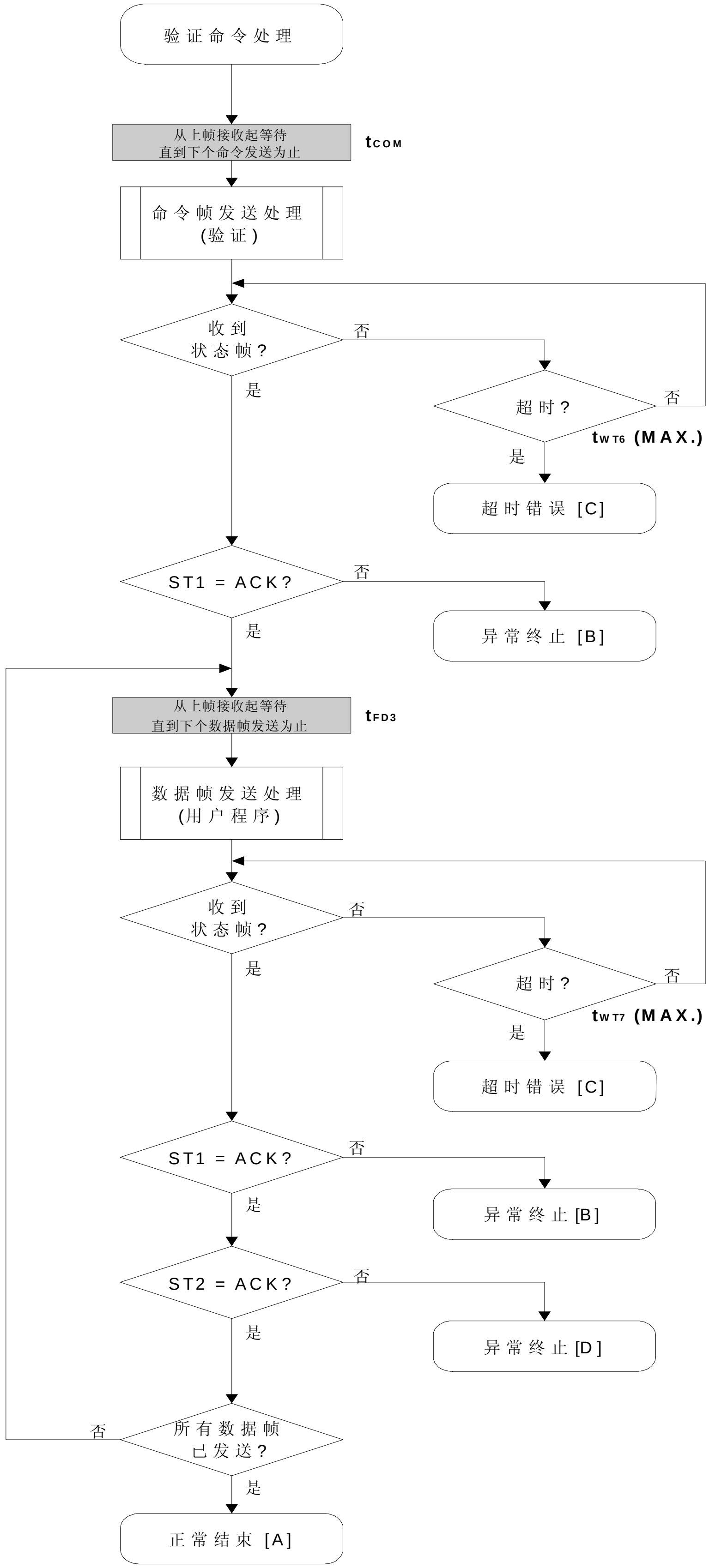
<5> 从上一个帧接收开始等待直到下一个数据帧发送为止。(等待时间 t_{FD3})
<6> 数据帧发送处理发送验证的用户数据。
<7> 从用户数据发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C] 被返回 (超时时间 t_{WT7}(MAX.))。
<8> 检查状态码(ST1/ST2) (同样参见处理程序流程图和流程图)。

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： 如果所有数据帧发送完成，处理正常结束 [A]。
  如果仍有剩余数据帧需要发送, 处理从<5>重复执行程序。
- 当 ST2 ≠ ACK 时： 异常终止 [D]

6.10.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，验证正常完成。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常(例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧。
异常终止 [D]	验证错误	0FH (ST2)	验证失败或发生其他错误。

6.10.4 流程图


验 证 命 令 处 理
从上帧接收起等待
直到下个命令发送为止
tCOM
命 令 帧 发 送 处 理
(验 证)
收 到
状 态 帧 ?
否
是
超 时 ?
否
tWT6 (MAX.)
是
超 时 错 误 [C]
ST1 = ACK?
否
是
异 常 终 止 [B]
从上帧接收起等待
直到下个数据帧发送为止
tFD3
数 据 帧 发 送 处 理
(用 户 程 序)
收 到
状 态 帧 ?
否
是
超 时 ?
否
tWT7 (MAX.)
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
已 发 送 ?
否
是
正 常 结 束 [A]

6.10.5 程序举例说明

以下是验证命令处理的程序举例说明：

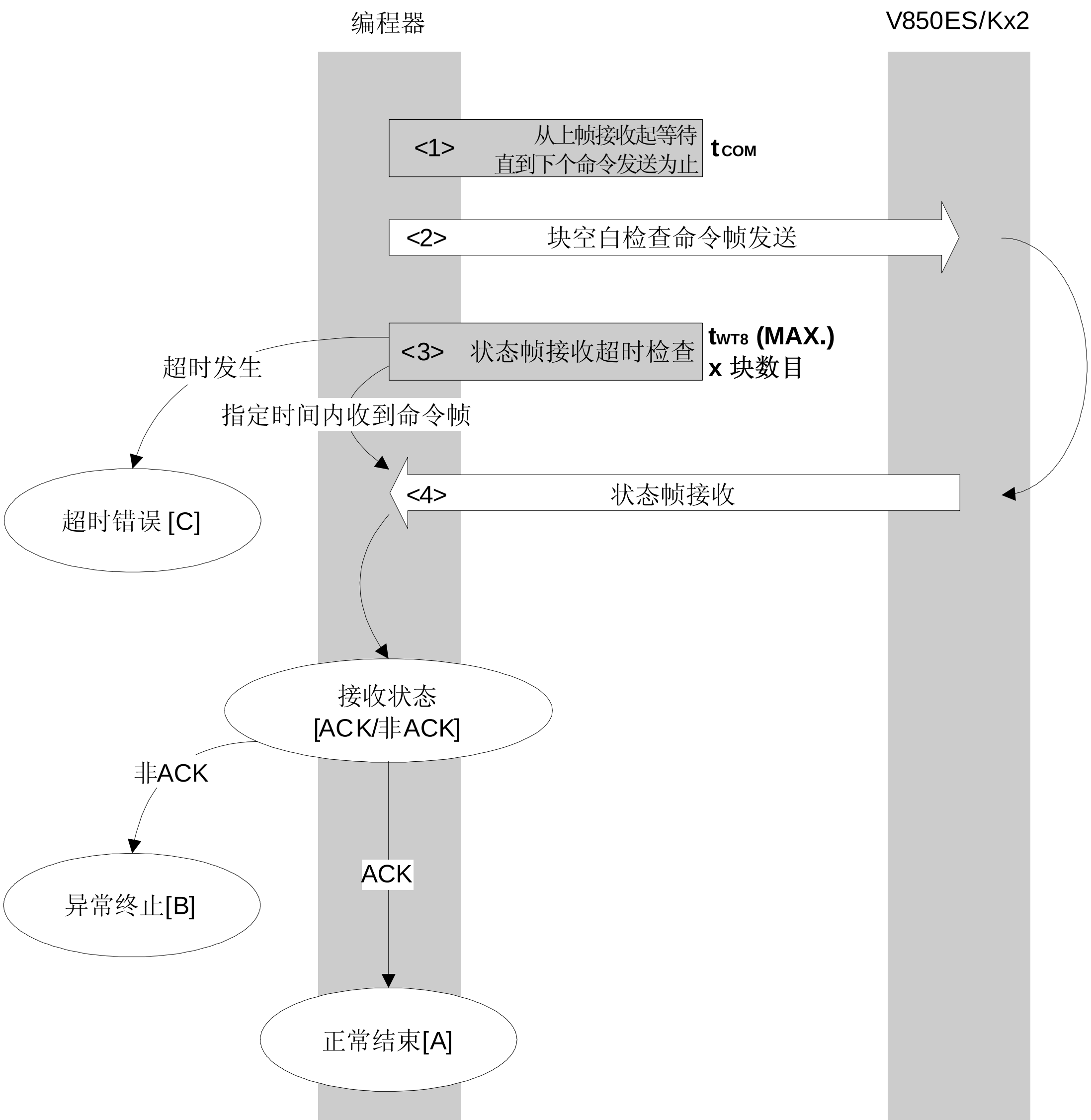
```
/*****************************************************************/
/*                                                               */
/*    验证命令                                                   */
/*                                                               */
/*****************************************************************/
/*   [i] u32 top          ... 开始地址                           */
/*   [i] u32 bottom       ... 结束地址                           */
/*   [i] u8  *buf         ... 指向验证数据缓冲区                 */
/*   [r] u16              ... 错误码                             */
/*****************************************************************/
u16       fl_ua_verify(u32 top, u32 bottom, u8 *buf)
{
  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

  /***********************************************/
  /*          发送命令并检查状态                 */
  /***********************************************/

  fl_wait(tCOM);              // 发送命令前等待

  put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm);        // 发送验证命令

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

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

  while(1){

          // make send data frame
          if ((bottom - send_head) > 256){           // 剩余长度> 256 ?
                  is_end = false;                    // 是，不是结束帧
                  send_size  = 256;                  // 发送长度= 256 字节
          }
          else{
                  is_end = true;
                  send_size = bottom - send_head + 1;
                                        //发送长度= (bottom - send_head) 字节
```

```
            }
            memcpy(fl_txdata_frm, buf+send_head, send_size); // 设置数据帧的有效载荷
            send_head += send_size;

            fl_wait(tFD3_UA);
            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);              // 获得状态帧
            switch(rc) {
                    case    FLC_NO_ERR:                          break;   // 继续
//                  case    FLC_DFTO_ERR:              return    rc;        break;   // 情况 [C]
                    default:            return    rc;         break;    // 情况 [B]
            }
            if (fl_st2_ua != FLST_ACK){                    // ST2 = ACK ?
                    rc = decode_status(fl_st2_ua);         //否
                    return    rc;                                // 情况 [D]
            }
            if (is_end)                              // 发送所有用户数据?
                    break;                           // 是
            //continue;
    }
    return    FLC_NO_ERR;    // 情况 [A]
}
```

6.11 块空白检查命令

6.11.1 处理程序流程图

块空白检查命令处理程序



6.11.2 处理程序的描述说明

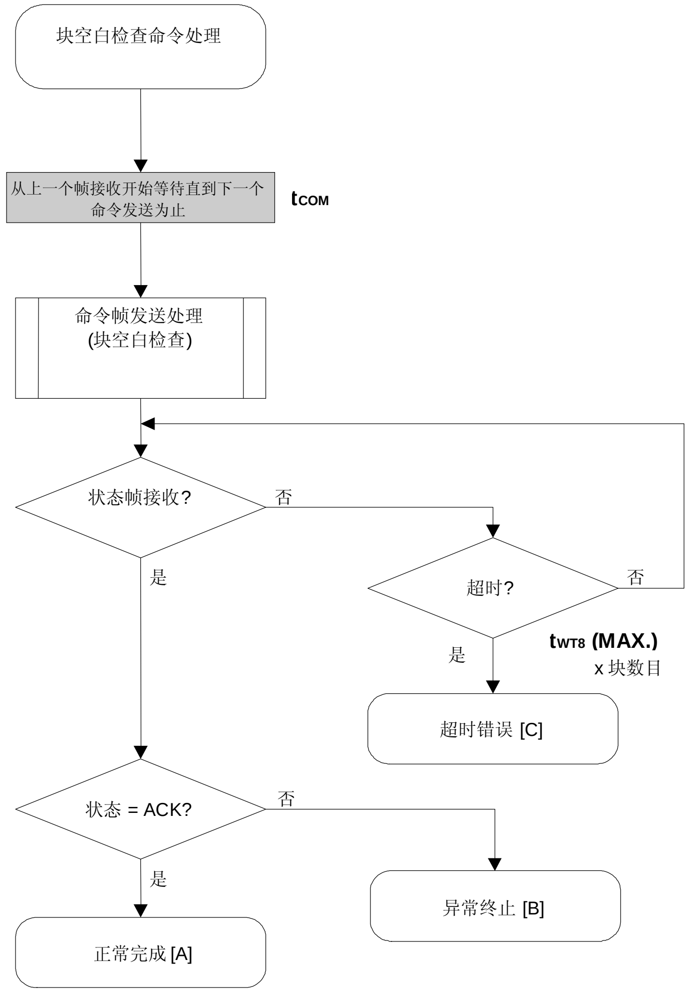
<1>　从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2>　命令帧发送处理发送块空白检查命令。
<3>　从命令发送开始执行超时检查直到状态帧接收为止。
　　　如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT8}(MAX.) × 块数目)。
<4>　检查状态码。

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

6.11.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，指定的所有块是空白。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址超出 Flash 存储器的范围。 • 指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度(LEN) 或没有 ETX)。
	MRG11 错误	1BH	Flash 存储器中的指定块不是空白块。
超时错误 [C]		–	在指定的时间内没有收到状态帧。

6.11.4 流程图


块空白检查命令处理
从上一个帧接收开始等待直到下一个命令发送为止
tCOM
命令帧发送处理
(块空白检查)
状态帧接收?
否
是
超时?
否
是
tWT8 (MAX.)
x 块数目
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
正常完成 [A]

6.11.5 程序举例说明

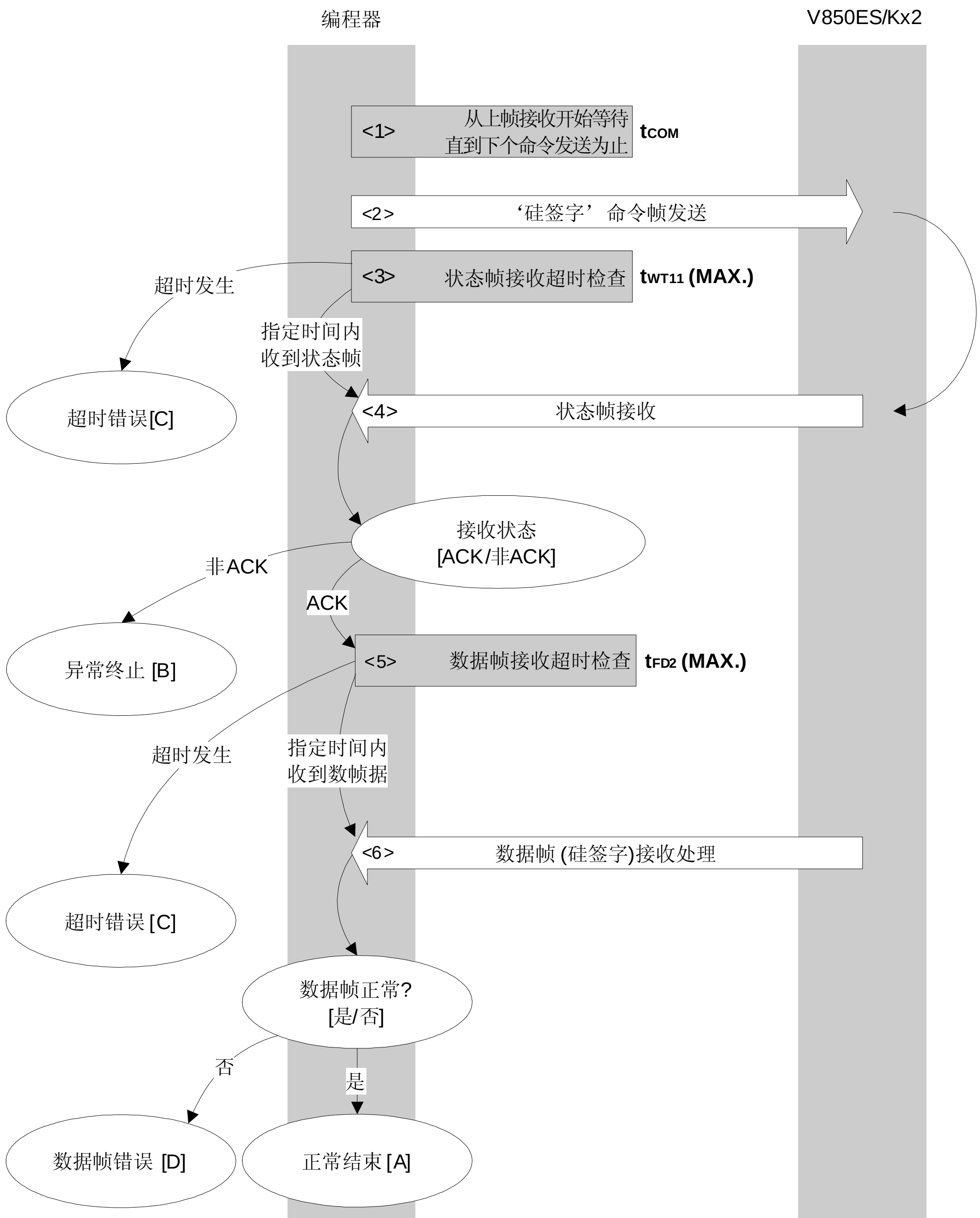
以下是块空白检查命令处理的程序举例说明：

```
/**************************************************************/
/*                                                            */
/*    块空白检查命令                                           */
/*                                                            */
/**************************************************************/
/*   [i] u32 top         ... 开始地址                          */
/*   [i] u32 bottom      ... 结束地址                          */
/*   [r] u16             ... 错误码                            */
/**************************************************************/
u16        fl_ua_blk_blank_chk(u32 top, u32 bottom)
{
  u16      rc;
  u16      block_num;

  set_range_prm(fl_cmd_prm, top, bottom);   // 设置 SAH/SAM/SAL, EAH/EAM/EAL
  block_num = get_block_num(top, bottom);   // 获得块编码

  fl_wait(tCOM);                 // 发送命令前等待

  put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);


  rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num);      // 获得状态帧
//switch(rc) {
//
//         case    FLC_NO_ERR:  return   rc;     break;   // 情况 [A]
//         case    FLC_DFTO_ERR:         return  rc;       break;  // 情况 [C]
//         default:         return  rc;     break;  // 情况 [B]
// }
  return rc;

}
```

6.12 ‘硅标记’命令

6.12.1 处理程序流程图

‘硅签字’命令处理程序



6.12.2 处理程序的描述说明

<1> 从上一个帧接收开始等待直到下一个命令发送为止 (等待时间 t_{COM})。
<2> 命令帧发送处理发送'硅标记'命令。
<3> 从命令发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT11}(MAX.))。
<4> 检查状态码。

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

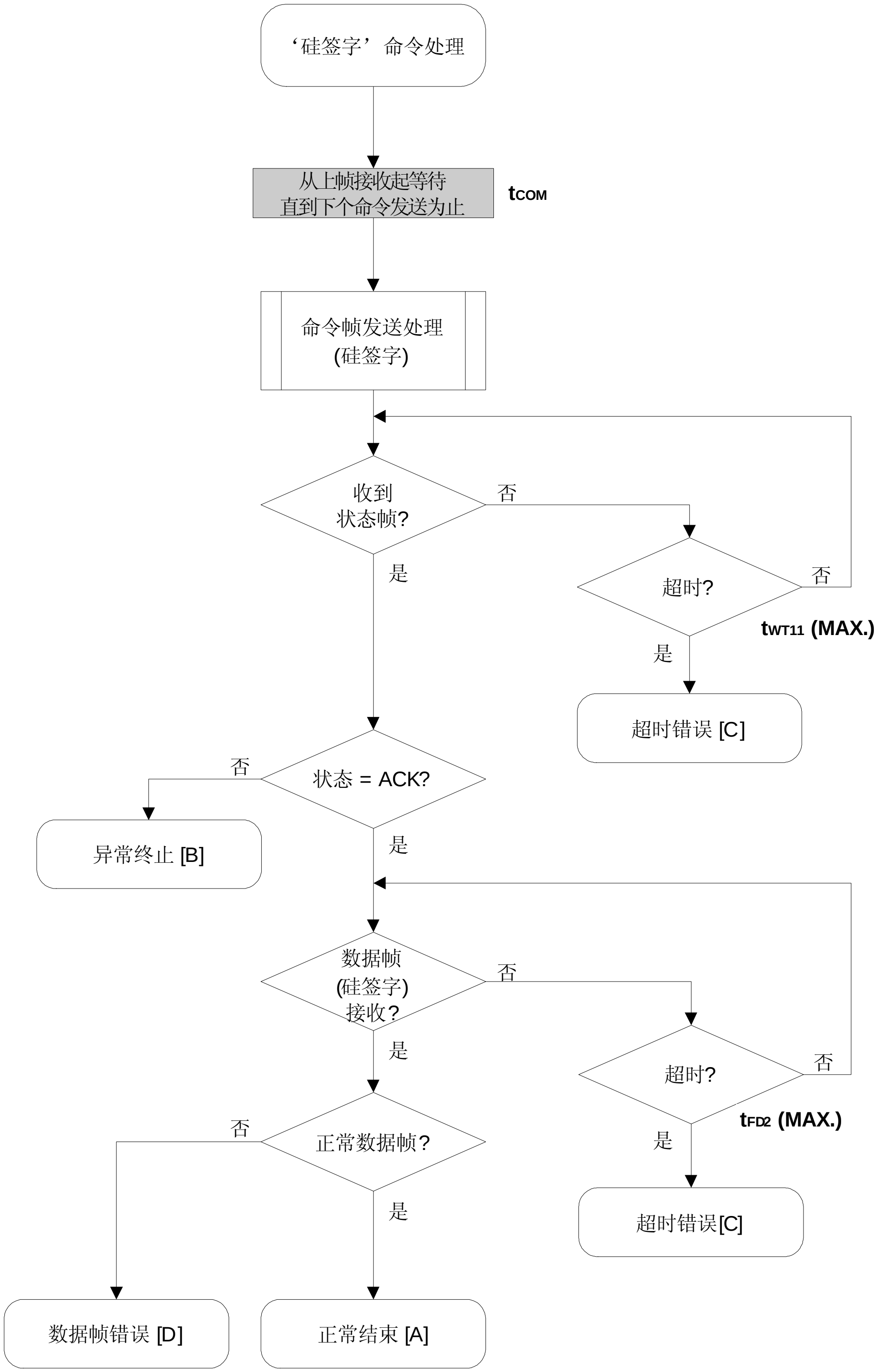
<5> 执行超时检查直到数据帧('硅标记'数据)接收为止。
如果发生超时，超时错误 [C] 被返回 (超时时间 t_{FD2}(MAX.))。
<6> 检查接收的数据帧('硅标记'数据)。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，'硅标记'被正常获得。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧或数据帧。
数据帧错误[D]		–	所接收的'硅标记'数据数据帧校验和不匹配。

6.12.4 流程图


‘硅签字’命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(硅签字)
收到
状态帧?
否
是
超时?
否
是
tWT11 (MAX.)
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
数据帧
(硅签字)
接收?
否
是
超时?
否
是
tFD2 (MAX.)
超时错误[C]
正常数据帧?
否
是
数据帧错误 [D]
正常结束 [A]

6.12.5 程序举例说明

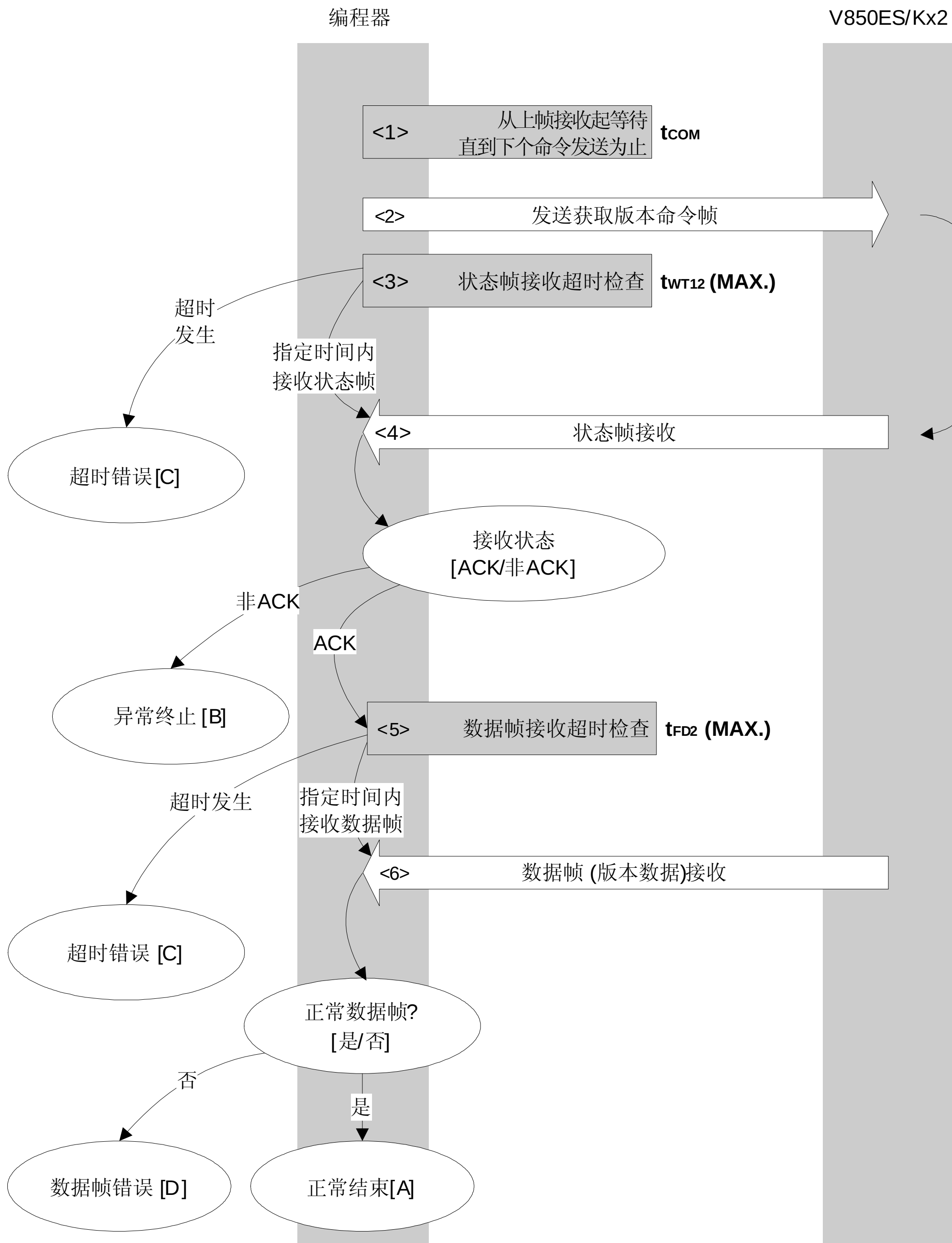
以下是'硅标记'命令处理的程序举例说明：

```
/***************************************************************/
/*                                                             */
/* 获得 '硅标记'命令                                            */
/*                                                             */
/***************************************************************/
/* [i] u8 *sig          ... 指向标记存储区                      */
/* [r] u16              ... 错误码                              */
/***************************************************************/
u16          fl_ua_getsig(u8 *sig)
{
        u16     rc;


        fl_wait(tCOM);              // 发送命令前等待

        put_cmd_ua(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm); // 发送获取标记命令

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


        rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX);//获得状态帧
        if (rc){                            // 如果错误
                return rc;                      // 情况 [D]
        }
        memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                        // 拷贝标记数据
        return  rc;                             // 情况 [A]
}
```

6.13 获取版本信息命令

6.13.1 处理程序流程图

获取版本命令处理程序



6.13.2 处理程序的描述说明

<1> 从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2> 命令帧发送处理发送获取版本信息命令。
<3> 从命令发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT12}(MAX.))。
<4> 检查状态码。

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

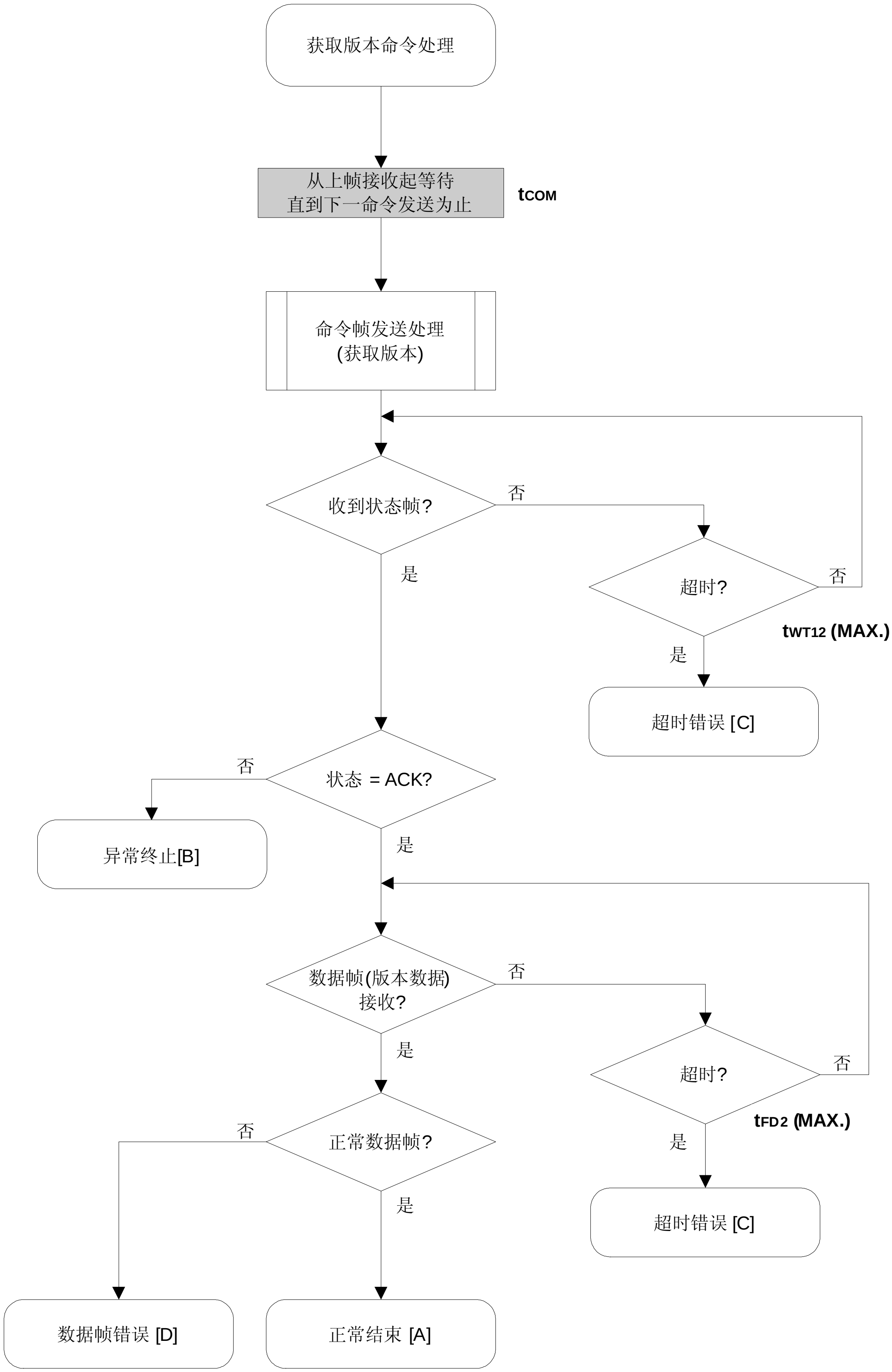
<5> 执行超时检查直到数据帧(版本数据)接收为止。
如果发生超时，超时错误 [C] 被返回 (超时时间 t_{FD2}(MAX.))。
<6> 检查接收的数据帧(版本数据)。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，版本数据被正常获得。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常(例如不正确的数据长度(LEN)或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧或数据帧。
数据帧错误[D]		–	所接收的版本数据数据帧校验和不匹配。

6.13.4 流程图


获取版本命令处理
从上帧接收起等待
直到下一命令发送为止
tCOM
命令帧发送处理
(获取版本)
收到状态帧?
否
是
超时?
否
tWT12 (MAX.)
是
超时错误 [C]
否
状态 = ACK?
异常终止[B]
是
数据帧(版本数据)
接收?
否
是
超时?
否
tFD2 (MAX.)
是
超时错误 [C]
否
正常数据帧?
是
数据帧错误 [D]
正常结束 [A]

6.13.5 程序举例说明

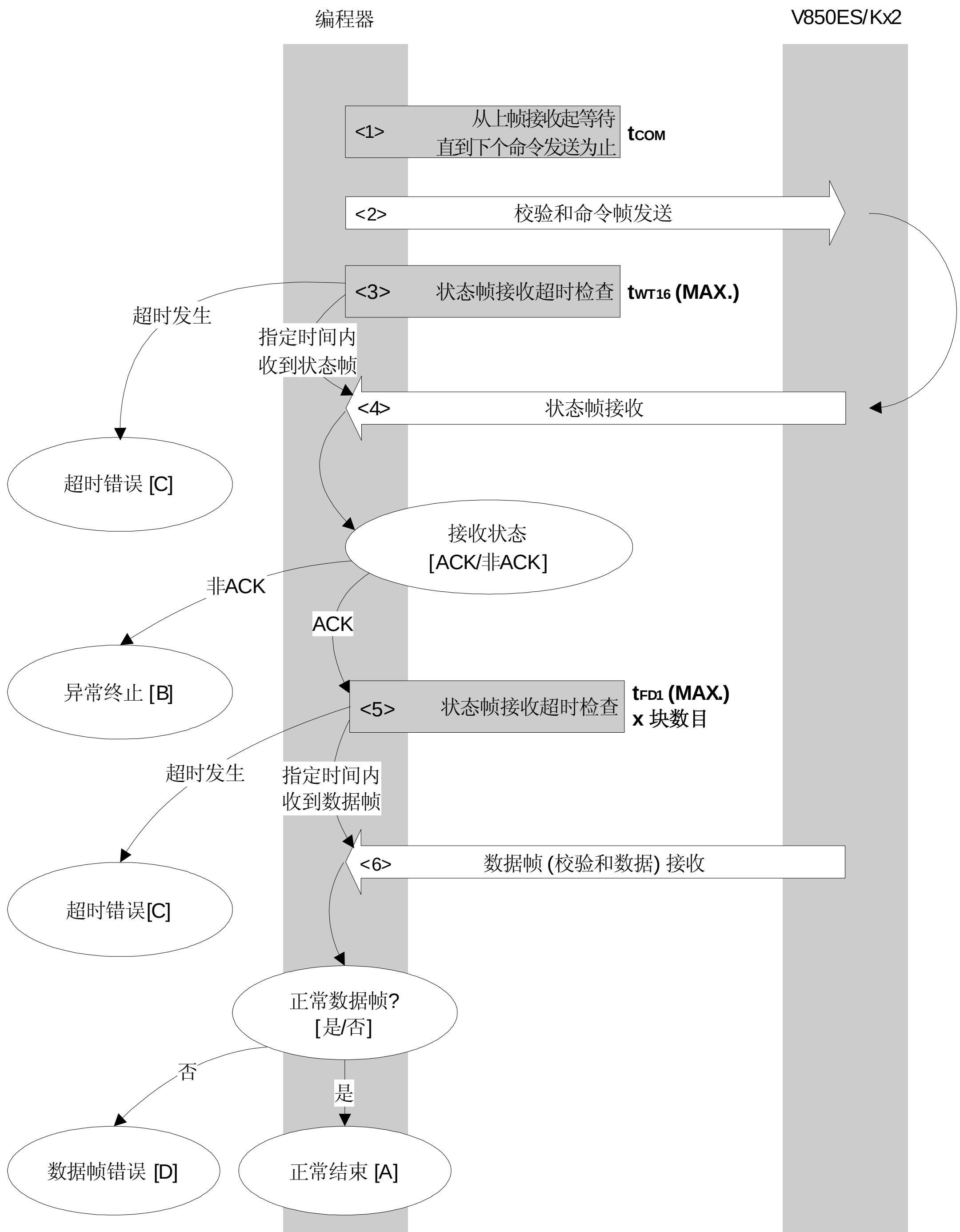
以下是获取版本信息命令处理的程序举例说明：

```
/**************************************************************/
/*                                                            */
/* 获得器件/固件版本信息命令                                   */
/*                                                            */
/**************************************************************/
/* [i] u8 *buf        ... 指向版本数据存储区                   */
/* [r] u16            ... 错误码                               */
/**************************************************************/
u16         fl_ua_getver(u8 *buf)
{
    u16     rc;


    fl_wait(tCOM);              // 发送命令前等待


    put_cmd_ua(FL_COM_GET_VERSION, 1, fl_cmd_prm); // 发送获取版本信息命令

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

    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX);// 获得状态帧
    if (rc){
            return rc;                      // 情况 [D]
    }

    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// 拷贝版本数据
    return rc;                              // 情况 [A]
}
```

6.14 校验和命令

6.14.1 处理程序流程图

校验和命令处理程序



6.14.2 处理程序的描述说明

<1>　从上一个帧接收开始等待直到下一个命令发送为止 (等待时间 t_{COM})。
<2>　命令帧发送处理发送校验和命令。
<3>　从命令发送开始执行超时检查直到状态帧接收为止。
　　如果发生超时，超时错误 [C] 被返回 (超时时间 t_{WT16}(MAX.))。
<4>　检查状态码。

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

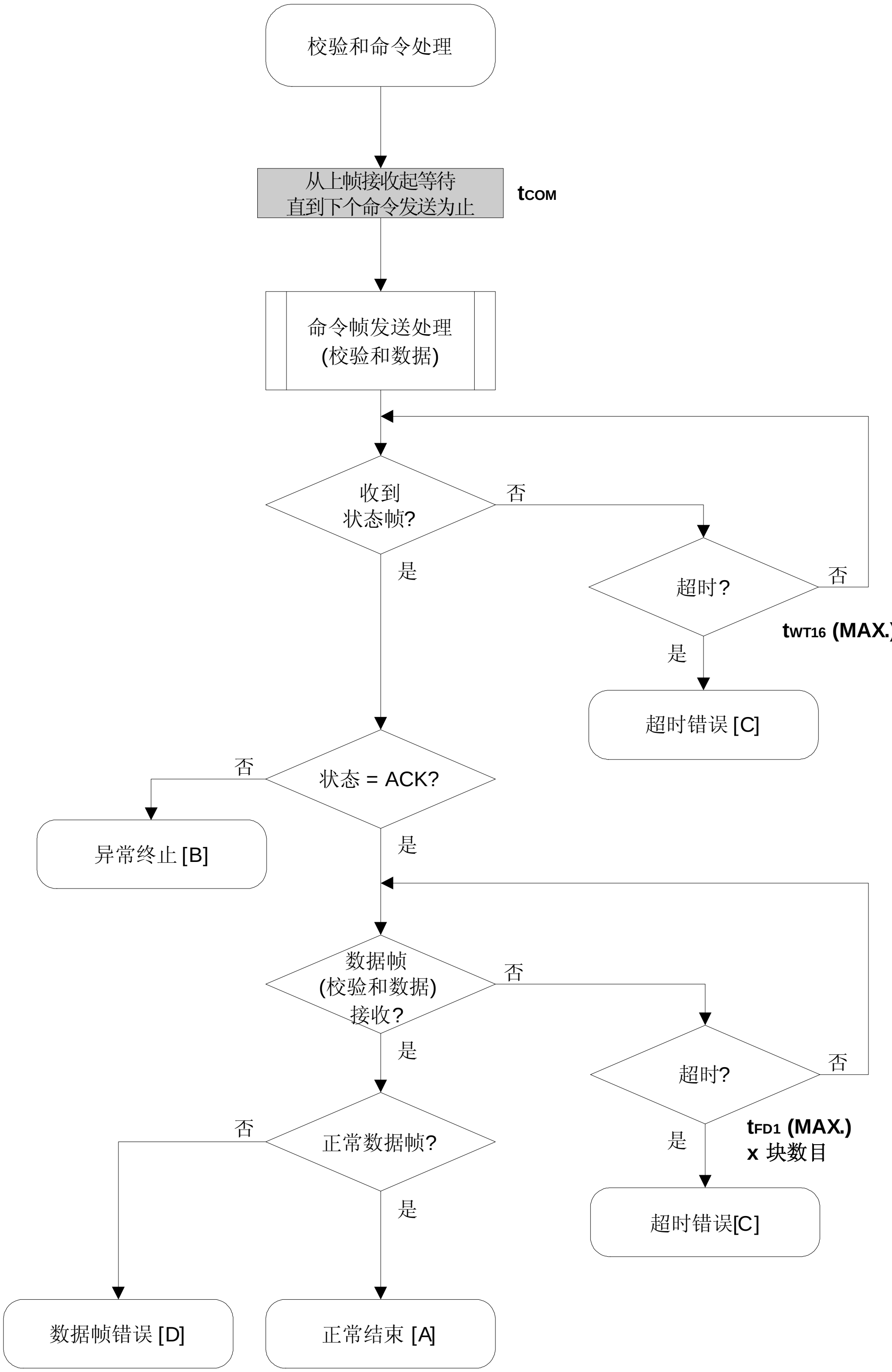
<5>　执行超时检查直到数据帧(校验和数据) 接收为止。
　　如果发生超时，超时错误 [C] 被返回 (超时时间 t_{FD1}(MAX.) × 块数目)。
<6>　检查接收的数据帧(校验和数据)。

　　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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，校验和数据被正常获得。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是从 Flash 存储器的顶端开始的块单元(2 KB)的一个固定地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度(LEN)或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧或数据帧。
数据帧错误 [D]		–	所接收的版本数据数据帧校验和不匹配。

6.14.4 流程图



校验和命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(校验和数据)
收到
状态帧?
否
是
超时?
否
tWT16 (MAX.)
是
超时错误 [C]
否
状态 = ACK?
异常终止 [B]
是
数据帧
(校验和数据)
接收?
否
是
超时?
否
tFD1 (MAX.)
x 块数目
是
超时错误[C]
否
正常数据帧?
是
数据帧错误 [D]
正常结束 [A]

6.14.5 程序举例说明

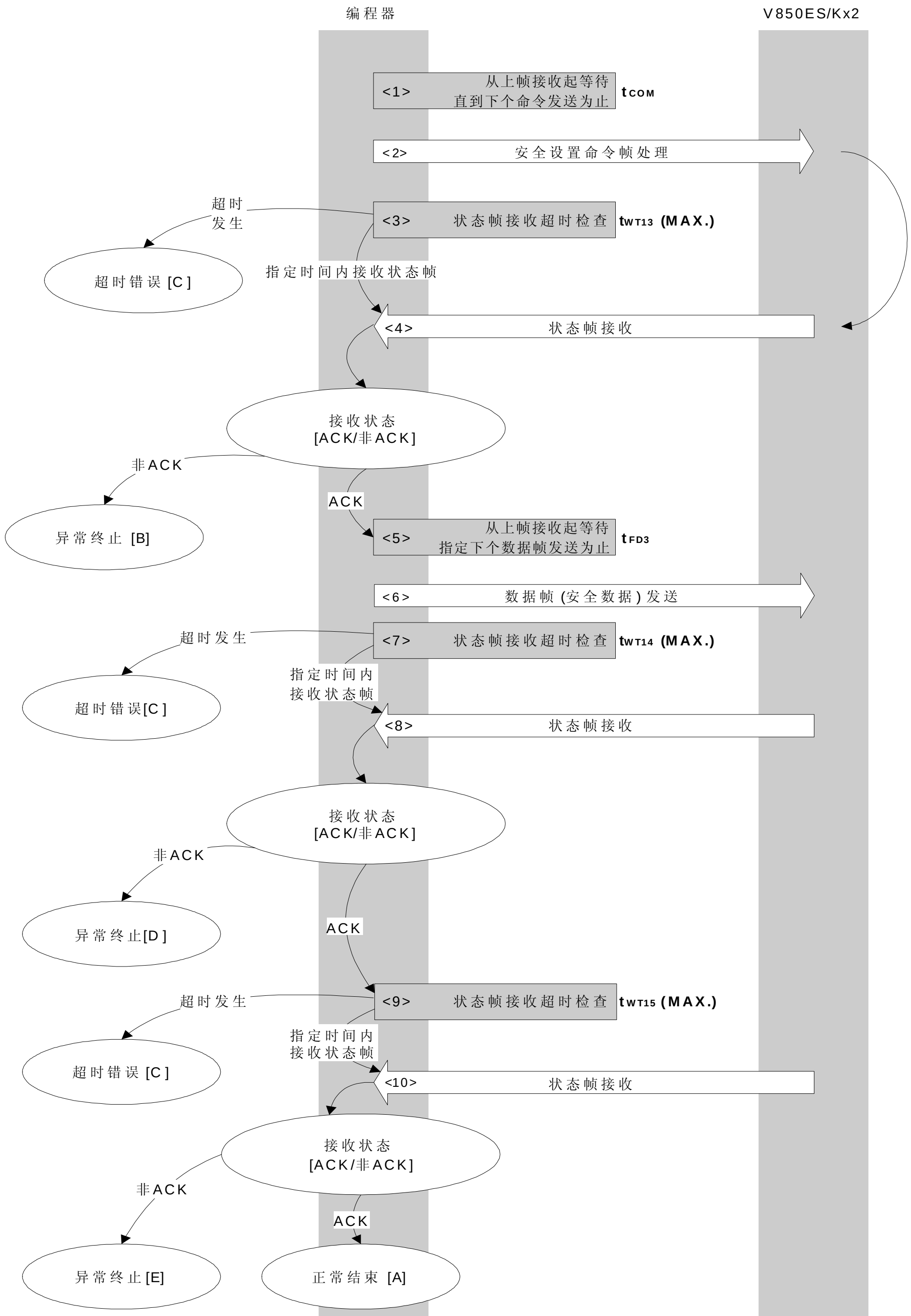
以下是校验和命令处理的程序举例说明：

```
/*****************************************************************/
/*                                                               */
/*    获得校验和命令                                              */
/*                                                               */
/*****************************************************************/
/*    [i] u16 *sum        ... 指向校验和存储区                      */
/*    [i] u32 top         ... 开始地址                             */
/*    [i] u32 bottom      ... 结束地址                             */
/*    [r] u16             ... 错误码                               */
/*****************************************************************/
u16     fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
  u16     rc;
  u16     block_num;

  /***********************************************/
  /*      设置参数                                  */
  /***********************************************/
  // set params
  set_range_prm(fl_cmd_prm, top, bottom);    // 设置 SAH/SAM/SAL, EAH/EAM/EAL
  block_num = get_block_num(top, bottom);    // 获得块编码


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

  fl_wait(tCOM);              // 发送命令前等待

  put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // 发送获取版本信息命令

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

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

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

}
```

6.15 安全设置命令

6.15.1 处理程序流程图

安全设置命令处理程序



6.15.2 处理程序的描述说明

<1>　从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2>　命令帧发送处理发送安全设置命令。
<3>　从命令发送开始执行超时检查直到状态帧接收为止。
　　　如果发生超时，超时错误 [C] 被返回 (超时时间 t_{WT13}(MAX.))。
<4>　检查状态码。

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

<5>　从上一个帧接收开始等待直到下一个数据帧发送为止(等待时间 t_{FD3})。
<6>　数据帧发送处理发送数据帧(安全设置数据) 。
<7>　执行超时检查直到状态帧接收为止。
　　　如果发生超时，超时错误 [C] 被返回 (超时时间 t_{WT14}(MAX.))。
<8>　检查状态码。

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

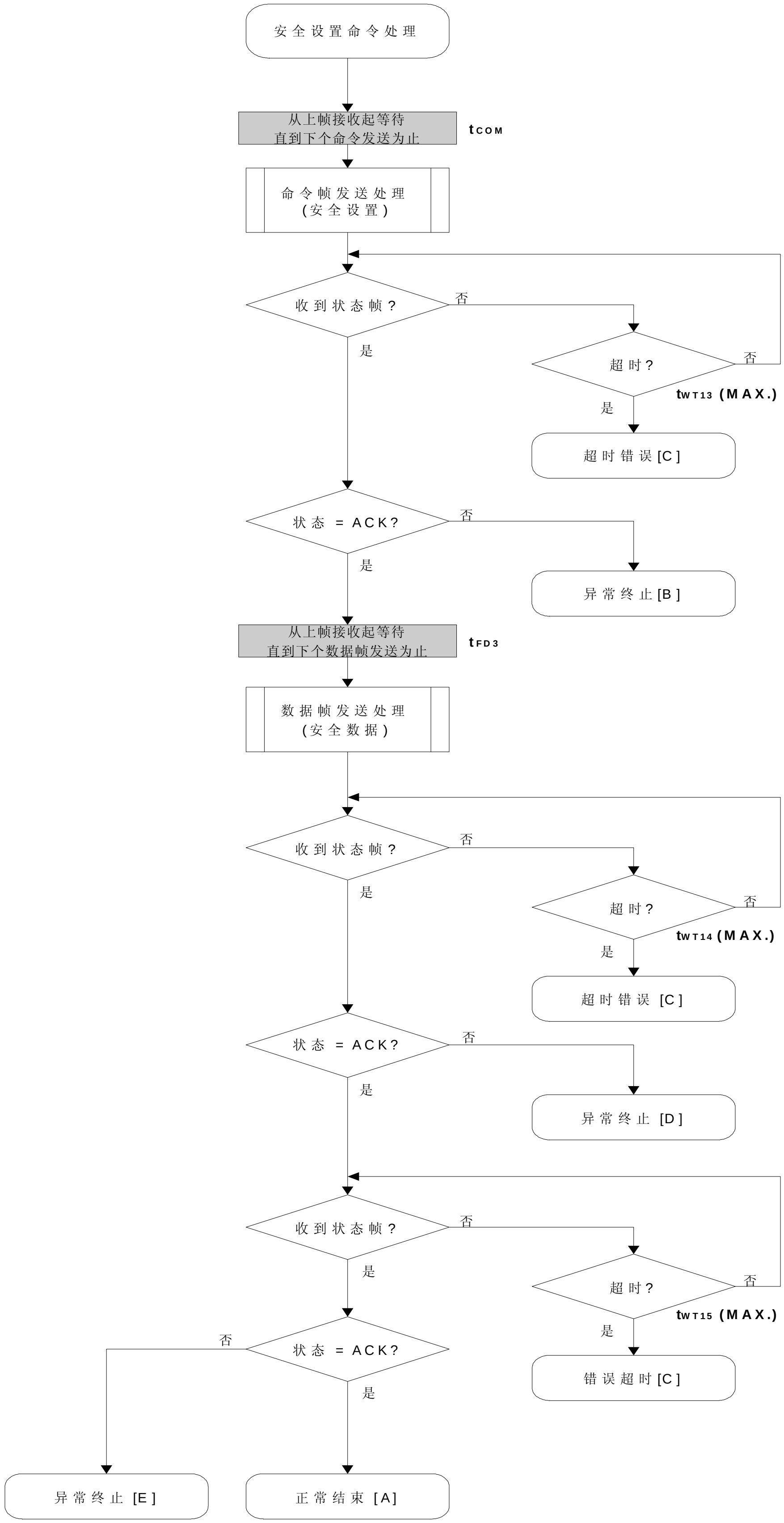
<9>　执行超时检查直到状态帧接收为止。
　　　如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT15}(MAX.))。
<10> 检查状态码。

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

6.15.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，安全设置被正常完成。
异常终止 [B]	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	• 试图使能已被设置禁止的标志。 • 在安全设置中重写引导块被禁止。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧或数据帧。
异常终止 [D]	FLMD 错误	18H	发生写错误。
	MGR10 错误	1AH	
	写错误	1CH	
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

6.15.4 流程图


安全设置命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(安全设置)
收到状态帧？
否
是
超时？
否
tWT13 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
从上帧接收起等待
直到下个数据帧发送为止
tFD3
数据帧发送处理
(安全数据)
收到状态帧？
否
是
超时？
否
tWT14 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [D]
收到状态帧？
否
是
超时？
否
tWT15 (MAX.)
是
错误超时 [C]
状态 = ACK?
否
是
异常终止 [E]
正常结束 [A]

6.15.5 程序举例说明

以下是安全设置命令处理的程序举例说明：

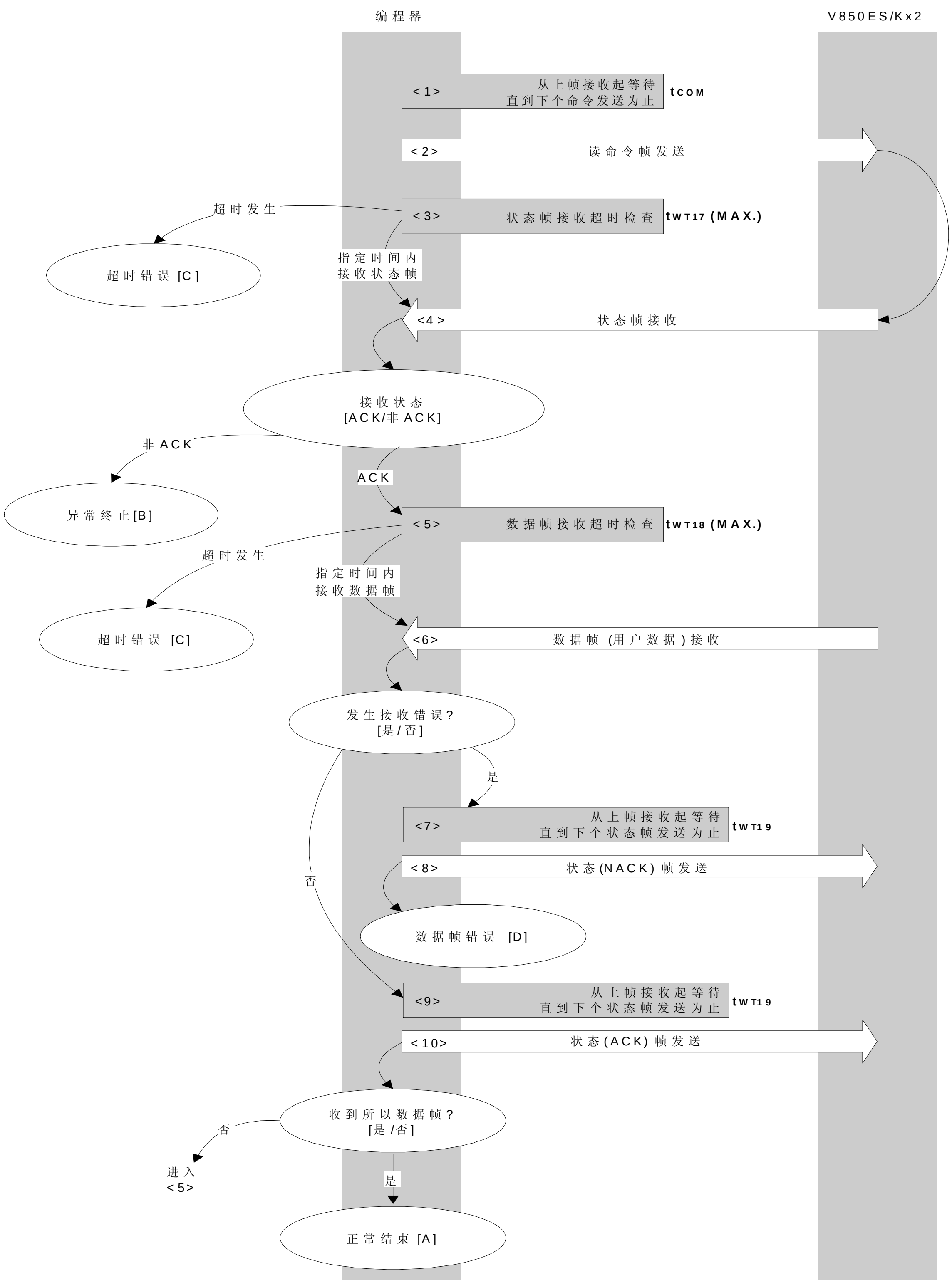
```
/*****************************************************************/
/*                                                               */
/*    设置安全标志命令                                            */
/*                                                               */
/*****************************************************************/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

  /***********************************************/
  /*        设置参数                              */
  /***********************************************/
  fl_cmd_prm[0] = 0x00;                    // 第1个字节必须是0x00
  fl_cmd_prm[1] = 0x00;                    // 第2个字节必须是0x00

  fl_txdata_frm[0] = (scf |= 0b11100000);
                                         // "FLG" (高 5位必须是 '1' (务必确定))
  fl_txdata_frm[1] = bot;                  // "BOT"

  /***********************************************/
  /*        发送命令                              */
  /***********************************************/
  fl_wait(tCOM);                // 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);  // 获得状态帧
  switch(rc) {
            case     FLC_NO_ERR:                          break;   // 继续
  //        case     FLC_DFTO_ERR:               return   rc;      break;   // 情况 [C]
            default:                return   rc;      break;   // 情况 [B]
  }

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

  fl_wait(tFD3_UA);

  put_dfrm_ua(2, fl_txdata_frm, true); // 发送安全设置 (FLAG) 并 BOT 数据

  rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX);  // 获得状态帧
  switch(rc) {
            case     FLC_NO_ERR:                          break;   // 继续
  //        case     FLC_DFTO_ERR:               return   rc;      break;   // 情况 [C]
            default:                return   rc;      break;   // 情况 [B]
  }
```

```
  /**************************************************/
  /*        检查内部验证                                              */
  /**************************************************/
  rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);  // 获得状态帧
//switch(rc) {
//
// case      FLC_NO_ERR:               return    rc;        break;    // 情况 [A]
// case      FLC_DFTO_ERR:                       return    rc;        break;    // 情况 [C]
// default:                      return    rc;        break;    // 情况 [B]
// }
  return rc;
}
```

6.16 读取命令

6.16.1 处理程序流程图

读 命 令 处 理 程 序



6.16.2 处理程序的描述说明

<1> 从上一个帧接收开始等待直到下一个命令发送为止(等待时间 t_{COM})。
<2> 命令帧发送处理发送读取命令。
<3> 从命令发送开始执行超时检查直到状态帧接收为止。
如果发生超时，超时错误 [C]被返回 (超时时间 t_{WT17}(MAX.))。
<4> 检查状态码。

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

<5> 执行超时检查直到数据帧接收为止（用户数据）。
如果发生超时，超时错误 [C] 被返回 (超时时间 t_{WT19}(MAX.))。
<6> 检查接收的数据帧。

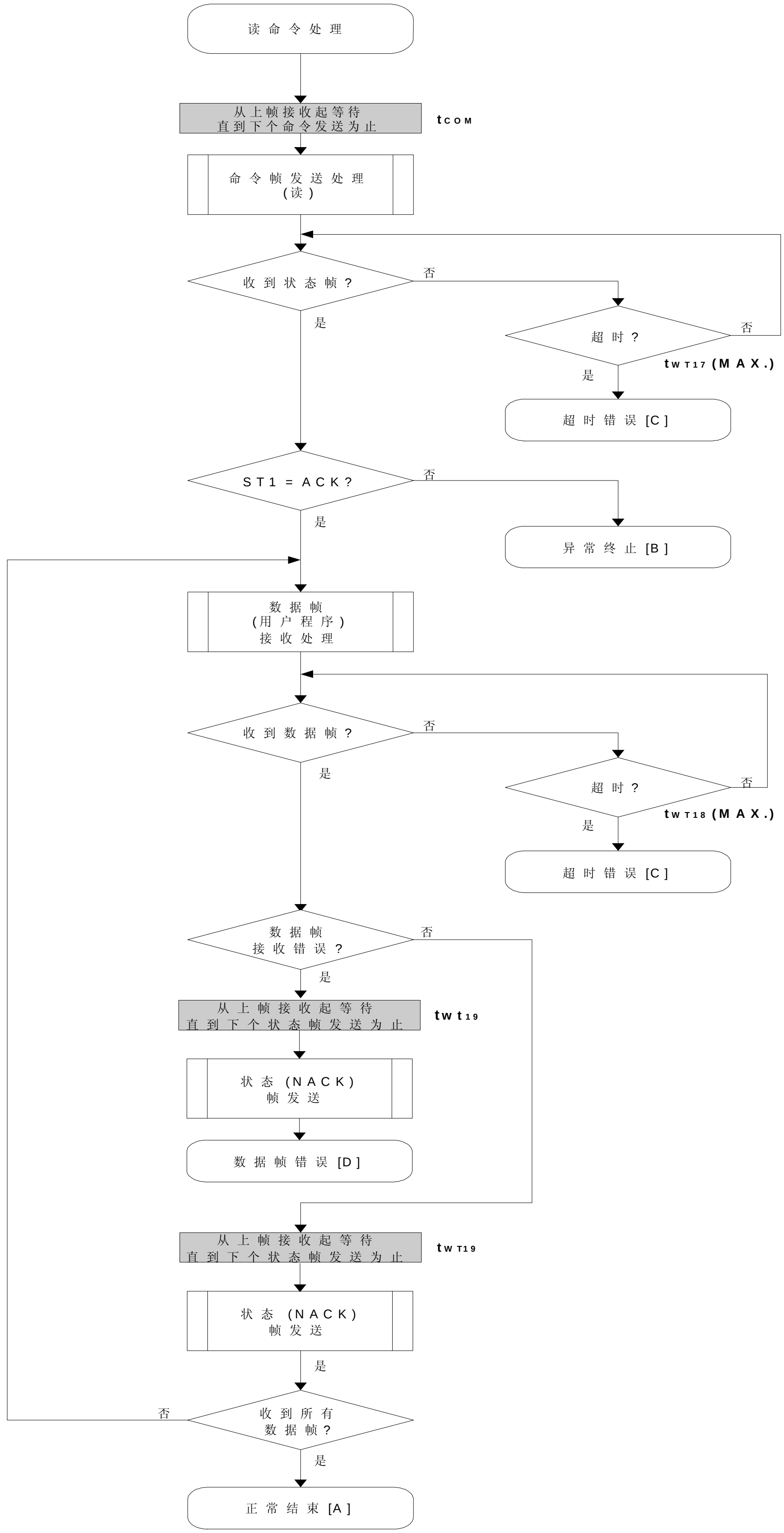
如果数据帧正常：　进入 <9>
如果数据帧异常：　进入 <7>

<7> 从上一个帧接收开始等待直到下一个状态(NACK) 帧发送为止(等待时间 t_{WT19}).
<8> 数据帧发送处理发送 NACK 帧。
→ 数据帧错误[D] 被返回。
<9> 从上一个帧接收开始等待直到下一个状态(ACK)帧发送为止 (等待时间 t_{WT19})。
<10> 数据帧发送处理发送 ACK 帧。
当所有数据帧接收完成时， 正常完成 [A] 被返回。
如果仍有剩余数据需要接收，处理从<5>重复执行程序。

6.16.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，读取数据被正常设置。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	在安全设置中读取被禁止。
	确认异常(NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误 [C]		–	在指定的时间内没有收到状态帧或数据帧。
数据帧错误 [D]		–	所接收的读取数据数据帧校验和不匹配。

6.16.4 流程图


读命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(读)
收到状态帧?
否
是
超时?
否
tWT17 (MAX.)
是
超时错误 [C]
ST1 = ACK?
否
是
异常终止 [B]
数据帧
(用户程序)
接收处理
收到数据帧?
否
是
超时?
否
tWT18 (MAX.)
是
超时错误 [C]
数据帧
接收错误?
否
是
从上帧接收起等待
直到下个状态帧发送为止
twt19
状态 (NACK)
帧发送
数据帧错误 [D]
从上帧接收起等待
直到下个状态帧发送为止
tWT19
状态 (NACK)
帧发送
是
收到所有
数据帧?
否
是
正常结束 [A]

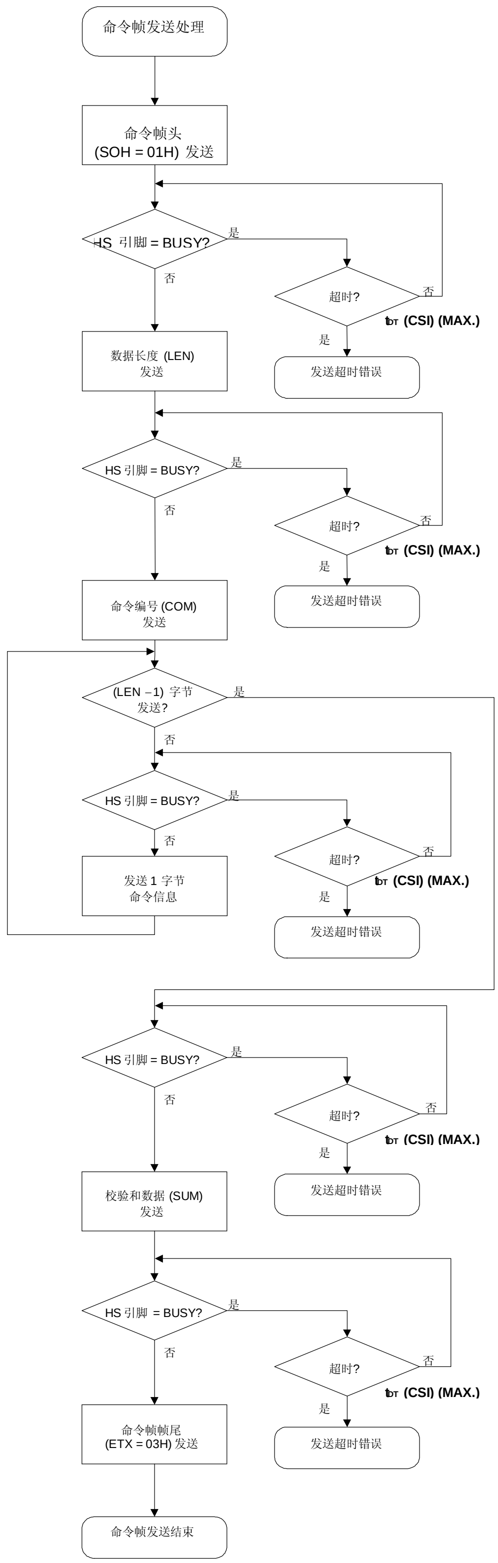
6.16.5 程序举例说明

以下是读取命令处理的程序举例说明：

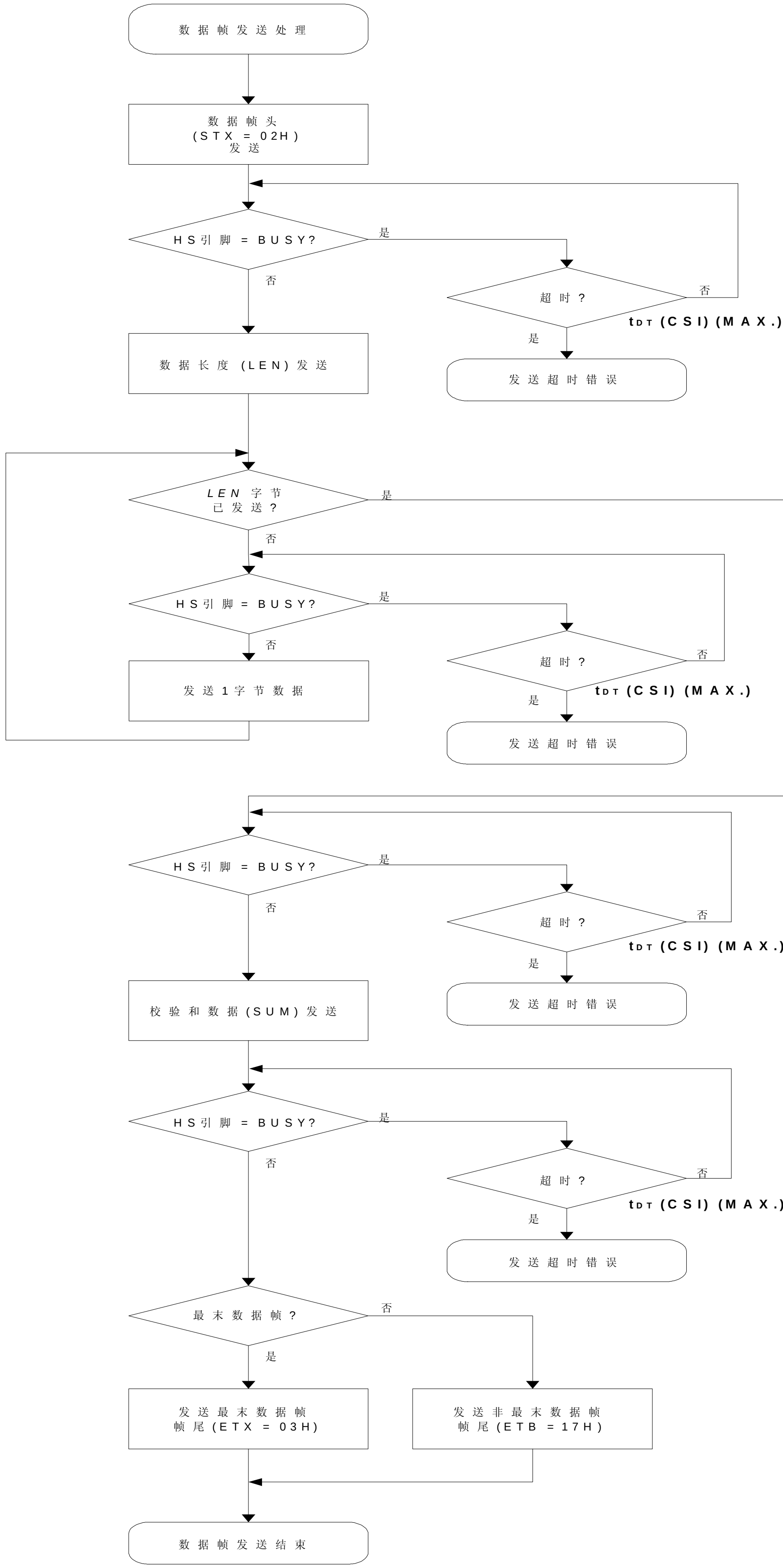
```
/**************************************************************/
/*                                                            */
/*    读取命令                                                */
/*                                                            */
/**************************************************************/
u16       fl_ua_read(u32 top, u32 bottom)
{
  u16     rc;
  u32     read_head;
  u16     len;
  u8      hooter;

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


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

  put_cmd_ua(FL_COM_READ, 7, fl_cmd_prm);

  rc = get_sfrm_ua(fl_ua_sfrm, tWT17_MAX); // 获得状态帧
  switch(rc) {
          case          FLC_NO_ERR:                     break;
  //      case          FLC_DFTO_ERR:         return    rc;       break;    // 情况 [C]
          default:                  return    rc;       break;    // 情况 [B]
  }

  /**********************************************/
  /*      接收用户数据                          */
  /**********************************************/
  read_head = top;

  while(1){

          rc = get_dfrm_ua(fl_rxdata_frm, tWT18_MAX);          //从 FLASH获得ROM数据

          switch(rc) {
                  case    FLC_NO_ERR:                     break;    // 继续
                  case    FLC_DFTO_ERR:         return    rc;       break;    // 情况 [C]
  //              case    FLC_RX_DFSUM_ERR:
                  default:                                // 情况 [B]

                          fl_wait(tWT19);
                  //      fl_wait(5000);
                          put_sfrm_ua(FLST_NACK);         // 发送状态 (NACK) 帧
                          return    rc;
                          break;
```

```
            }
            fl_wait(tWT19);
    //      fl_wait(5000);
            put_sfrm_ua(FLST_ACK);

            /************************************************/
            /*          储存 ROM数据                          */
            /************************************************/
            if ((len = fl_rxdata_frm[OFS_LEN]) == 0)     // 获得长度
                    len = 256;

            memcpy(read_buf+read_head, fl_rxdata_frm+2, len); // 存入外部RAM

            read_head += len;

            /************************************************/
            /*          结束检查                              */
            /************************************************/
            hooter = fl_rxdata_frm[len + 3];
            if (hooter == FL_ETB)              // 帧结束?
                    continue;                  // 否
            break;                             // 是
    }

    return  FLC_NO_ERR;

}
```

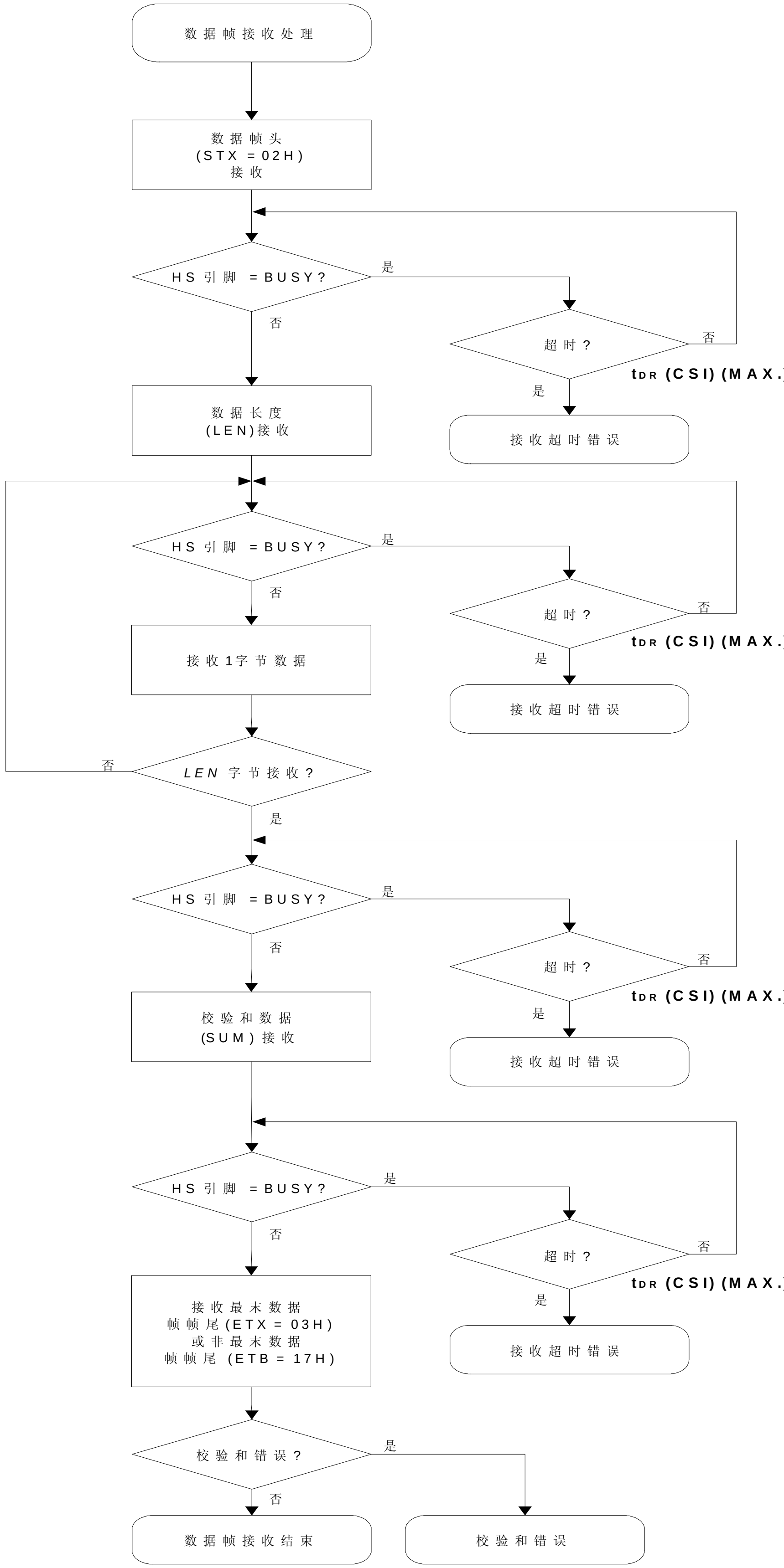
7.1　命令帧发送处理流程图



7.2 数据帧发送处理流程图



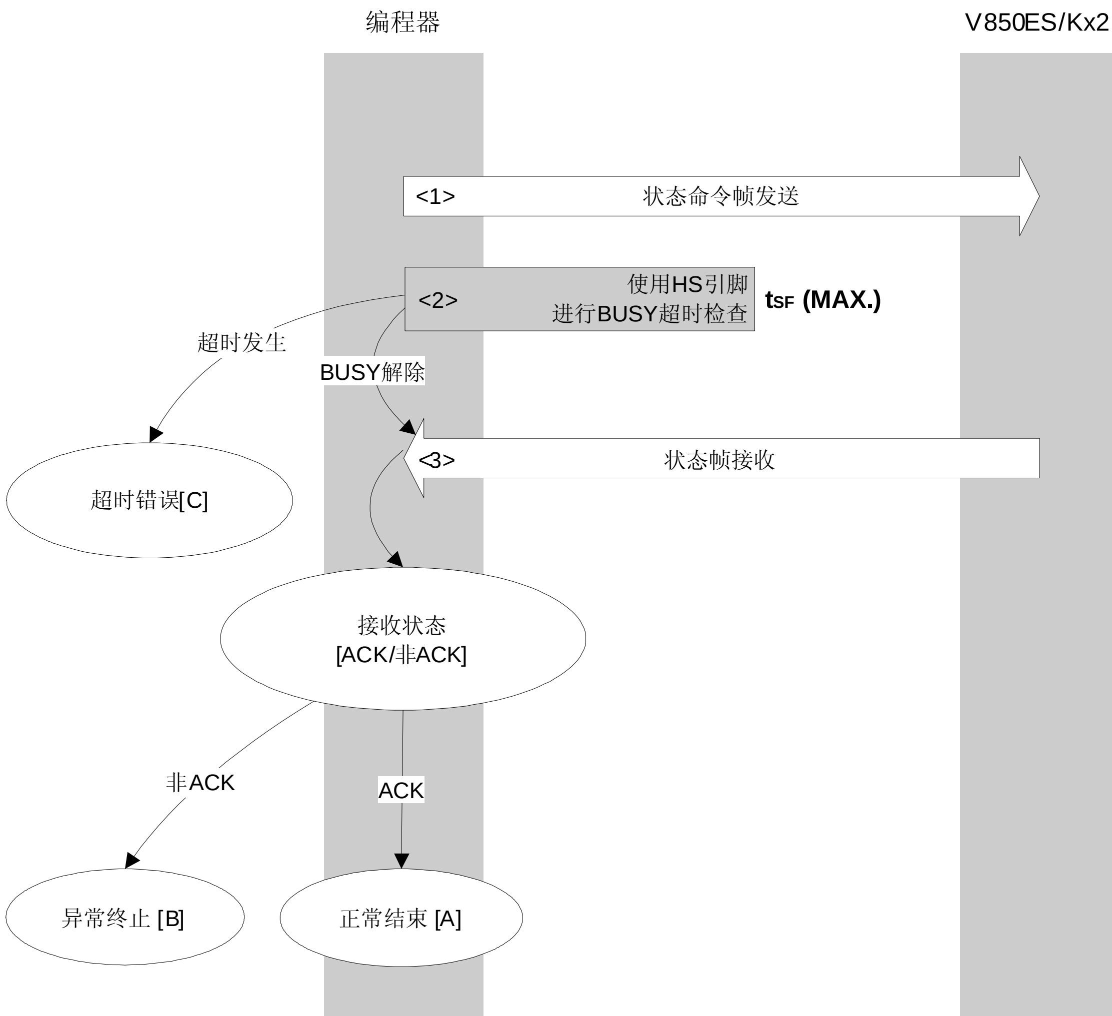
7.3　数据帧接收处理流程图



7.4　状态命令

7.4.1　处理程序流程图

状态命令处理程序



7.4.2 处理程序的描述说明

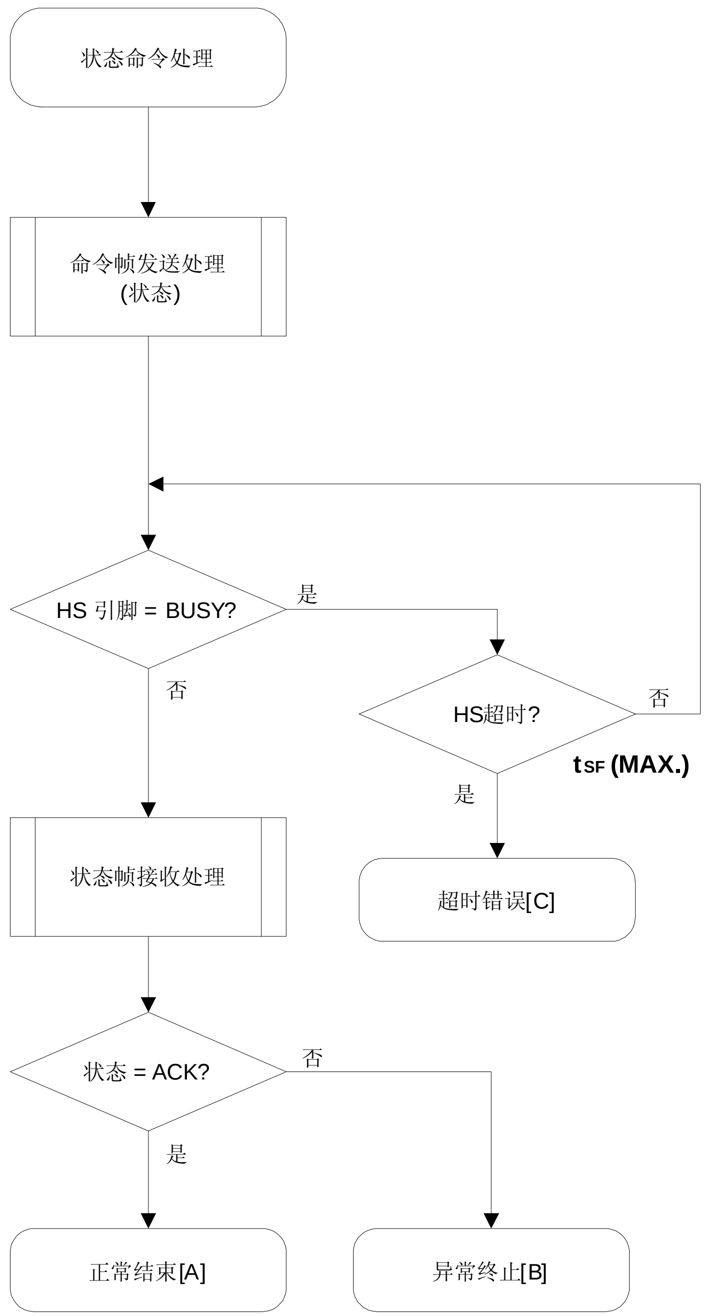
<1> 命令帧发送处理发送状态命令。
<2> 用 HS 引脚检查 V850ES/Kx2 BUSY 状态。
如果发生 BUSY 超时,超时错误 [C] 被返回 (超时时间 t_{SF}(MAX.))。
<3> 检查状态码。

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

7.4.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	从 V850ES/Kx2 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	编程器发送帧的数据异常。
	验证错误	0FH	编程器发送帧的数据发生验证错误。
	保护错误	10H	试图执行被安全设置命令禁止的处理。
	确认异常 (NACK)	15H	确认异常。
	FLMD 错误	18H	发生写错误。
	MRG10 错误	1AH	发生擦除错误
	MRG11 错误	1BH	数据写入过程中发生内部验证错误，或发生空白检查错误。
	写错误	1CH	发生写错误。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。

7.4.4 流程图


状态命令处理
命令帧发送处理
(状态)
HS 引脚 = BUSY?
是
否
HS超时?
否
t_{SF} (MAX.)
是
状态帧接收处理
超时错误[C]
状态 = ACK?
否
是
正常结束[A]
异常终止[B]

7.4.5 程序举例说明

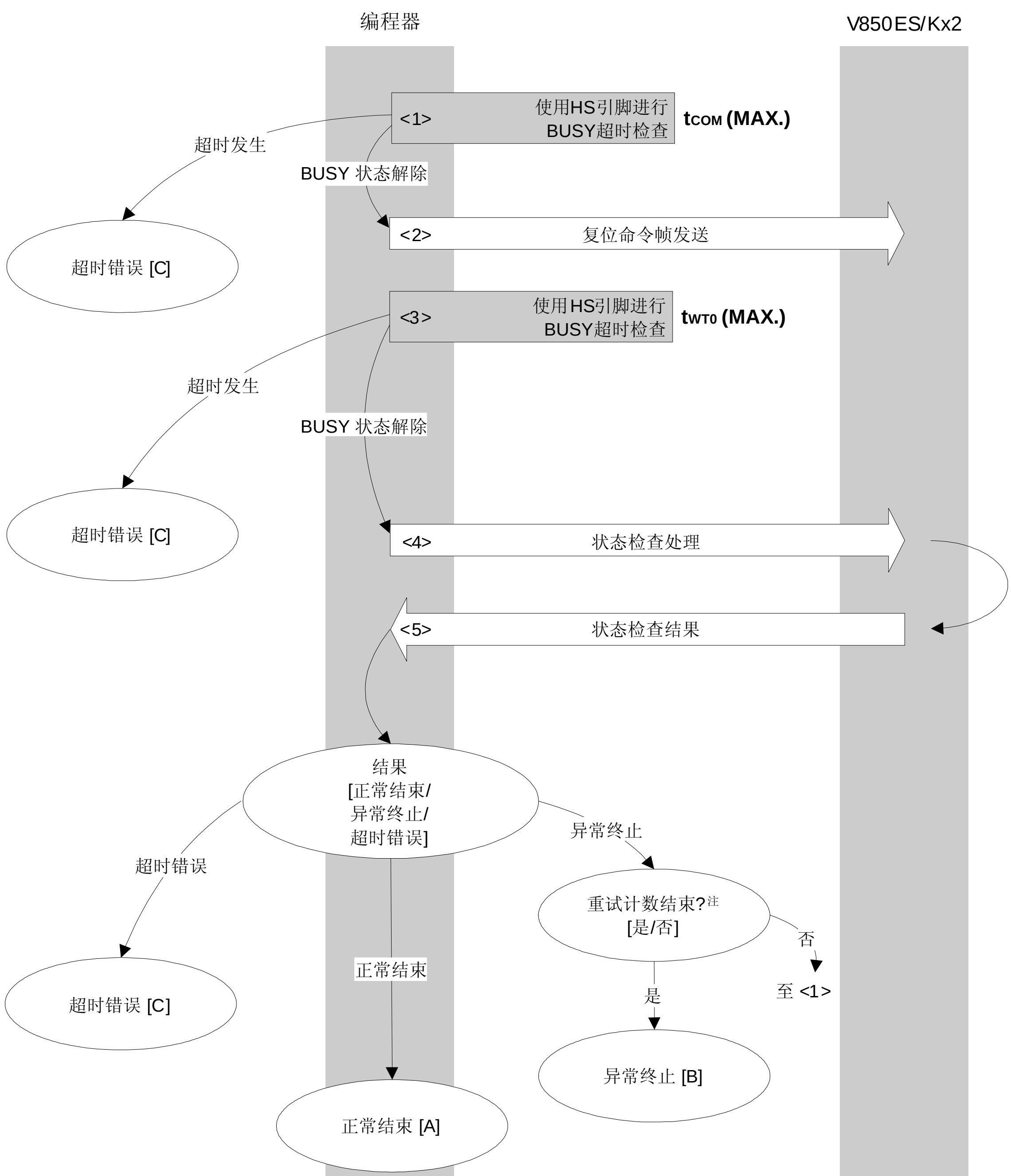
以下是状态命令处理的程序举例说明：

```
/****************************************************************/
/*                                                  */
/*  获得状态命令(CSI-HS)                                                                  */
/*                                                  */
/****************************************************************/
/*  [r] u16      ... 解码状态或错误码             */
/*                                                  */
/*  (见 fl.h/fl-proto.h 并                               */
/*        在 fl.c 里解码状态的定义 ())           */
/****************************************************************/
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 "状态" 命令
        if (rc)
                return    rc;       // 情况 [C]

        if (hs_busy_to(tSF_MAX))            // HS-Busy 超时 ?
                return    FLC_HSTO_ERR;                 // 检测到超时: 情况 [C]

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


        rc = decode_status(fl_st1);// 解码返回编码

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

7.5　复位命令

7.5.1　处理程序流程图

复位命令处理程序



注　　不要超出复位命令发送的重试计数(最大到 16 次)。

7.5.2　处理程序的描述说明

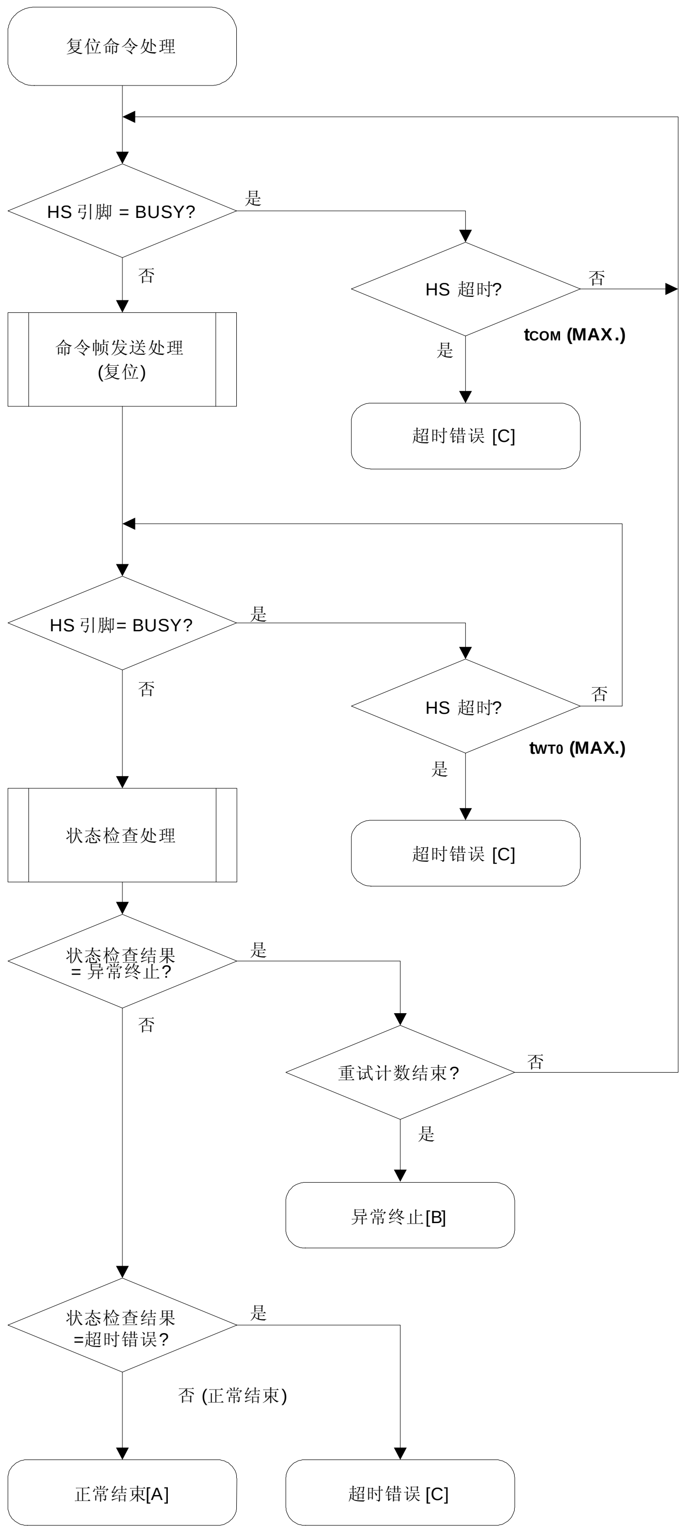
<1>　使用 HS 引脚进行 V850ES/Kx2 BUSY 状态检查。
　　如果发生 BUSY 超时,超时错误 [C] 被返回 (超时时间 t_{COM}(MAX.))。
<2>　命令帧发送处理发送复位命令。
<3>　使用 HS 引脚进行 V850ES/Kx2 BUSY 状态检查。
　　如果发生 BUSY 超时,超时错误 [C] 被返回(超时时间 t_{WT0}(MAX.))。
<4>　状态检查处理获得状态帧。
<5>　根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	如果没有超出重试计数，程序从<1>重复执行
	如果超出重试计数，处理异常终止 [B]
当发生超时错误时：	超时错误 [C] 被返回

7.5.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，编程器和 V850ES/Kx2 之间建立同步。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	• 处理中接收到一个除状态命令外的其他命令。 • 命令帧数据异常(例如不正确的数据长度(LEN) 或没有 ETX)。
超时错误 [C]		–	状态检查处理超时。 由于 HS 引脚的 busy 状态使处理超时。

7.5.4 流程图


复位命令处理
HS 引脚 = BUSY?
是
否
HS 超时?
否
是
tCOM (MAX.)
命令帧发送处理
(复位)
超时错误 [C]
HS 引脚= BUSY?
是
否
HS 超时?
否
是
tWT0 (MAX.)
状态检查处理
超时错误 [C]
状态检查结果
= 异常终止?
是
否
重试计数结束?
否
是
异常终止[B]
状态检查结果
=超时错误?
是
否 (正常结束)
正常结束[A]
超时错误 [C]

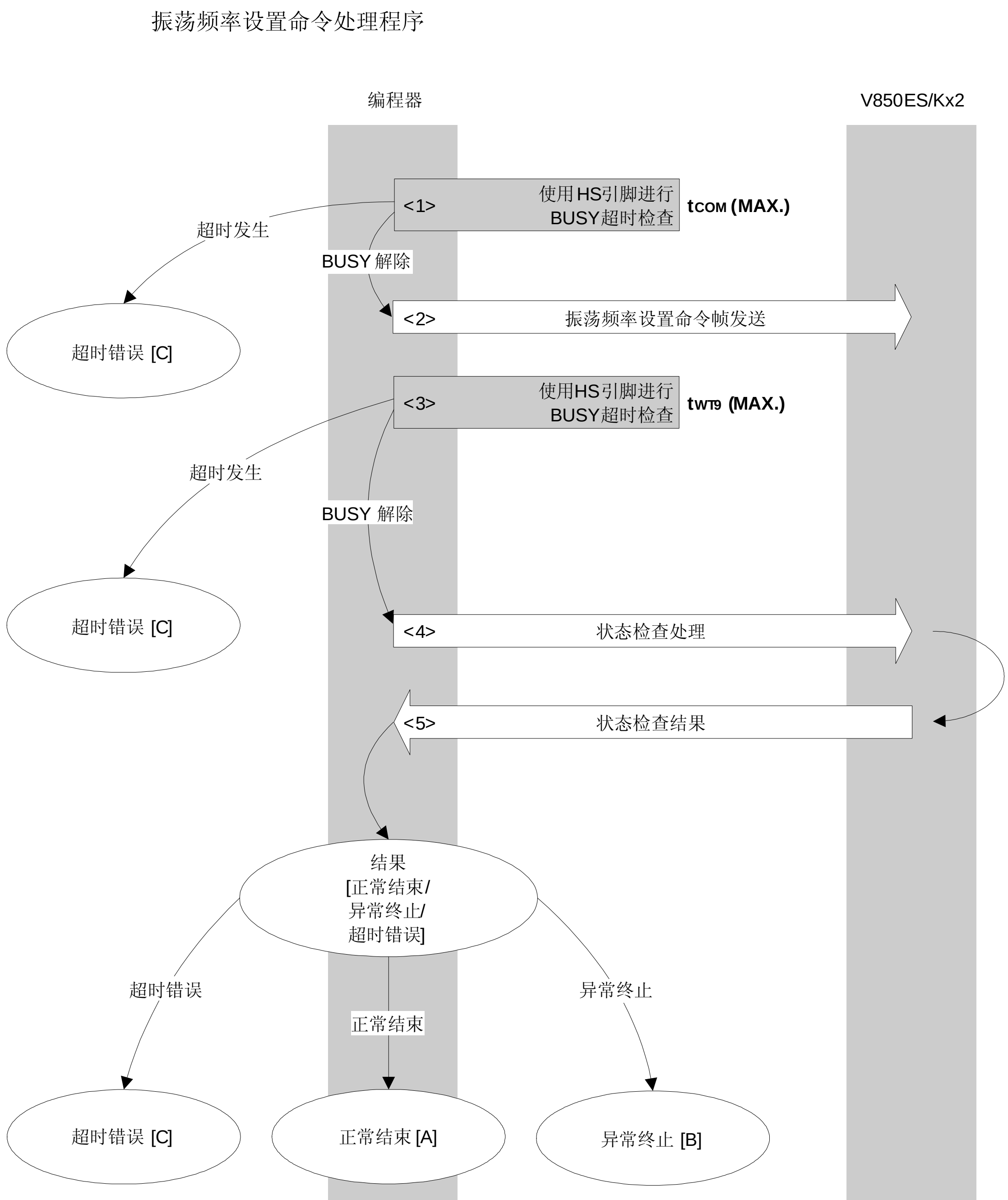
7.5.5　程序举例说明

以下是复位命令处理的程序举例说明：

```
/***************************************************************/
/*                                                        */
/* 复位命令 (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;                  // 检测到超时:情况 [C]

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

                if (hs_busy_to(tWT0_MAX))
                        return  FLC_HSTO_ERR;                  //检测到超时:情况 [C]

                rc = fl_hs_getstatus();     // 获得状态帧
                if (rc == FLC_ACK)              // ST1 = ACK ?
                        break;                  // 情况 [A]
                //continue;                     // 情况 [B] (如果从循环中退出)

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

7.6　振荡频率设置命令

7.6.1　处理程序流程图



7.6.2 处理程序的描述说明

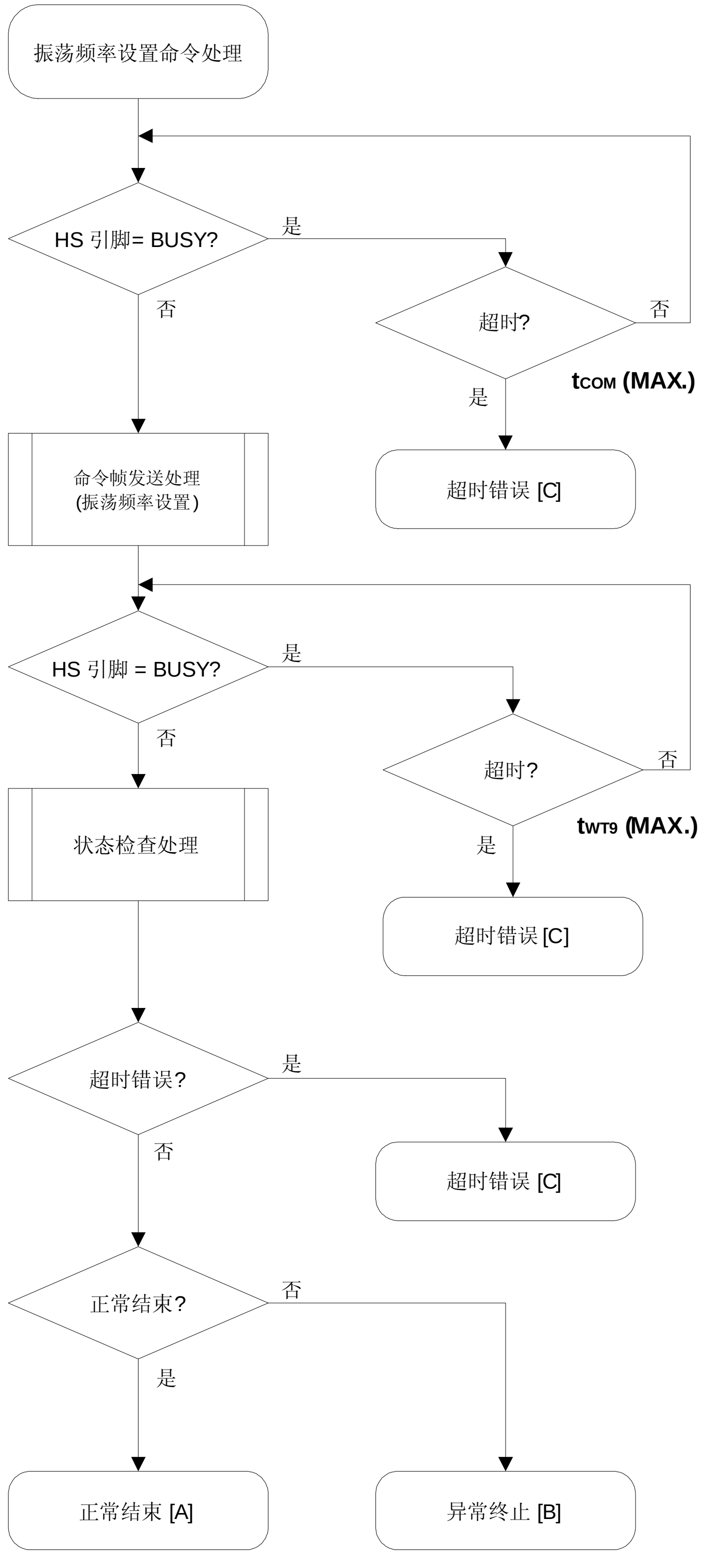
<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2> 命令帧发送处理发送振荡频率设置命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT9}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

7.6.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，操作频率被正确的设置到 V850ES/Kx2。
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	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。

7.6.4 流程图



振荡频率设置命令处理
HS 引脚= BUSY?
是
否
超时?
否
是
tCOM (MAX.)
命令帧发送处理
(振荡频率设置)
超时错误 [C]
HS 引脚 = BUSY?
是
否
超时?
否
是
tWT9 (MAX.)
状态检查处理
超时错误 [C]
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
正常结束 [A]
异常终止 [B]

7.6.5　程序举例说明

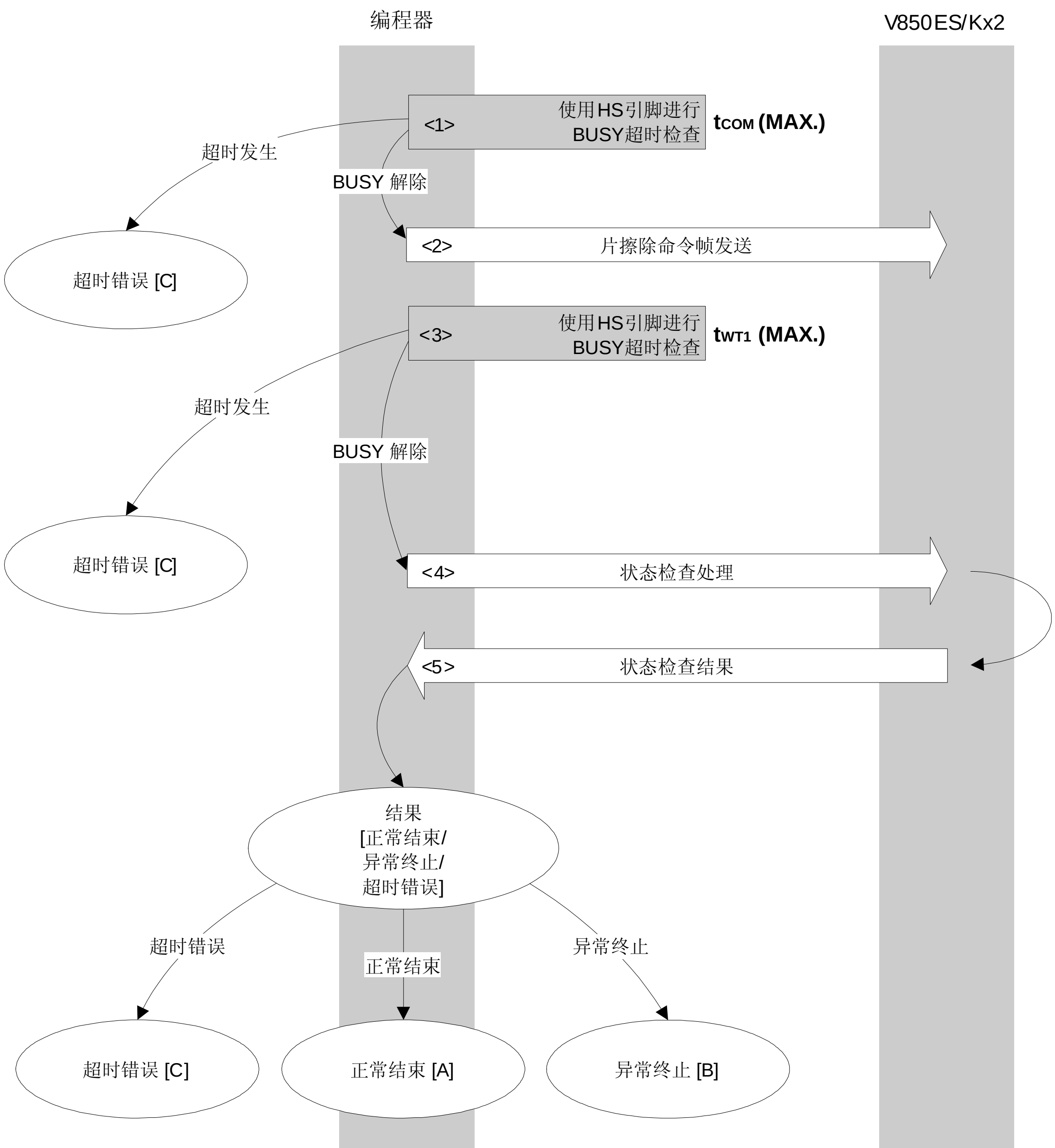
以下是振荡频率设置命令处理的程序举例说明：

```
/****************************************************************/
/*                                                    */
/* 设置 Flash 存储器器件时钟值命令(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;                 //检测到超时:情况 [C]

        if (rc = put_cmd_hs(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm))
                                                // 发送 "振荡频率设置" 命令
                return    rc;                           // 情况 [C]

        if (hs_busy_to(tWT9_MAX))
                return    FLC_HSTO_ERR;                 //检测到超时:情况 [C]


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

7.7　片擦除命令

7.7.1　处理程序流程图

片擦除命令处理程序



7.7.2　处理程序的描述说明

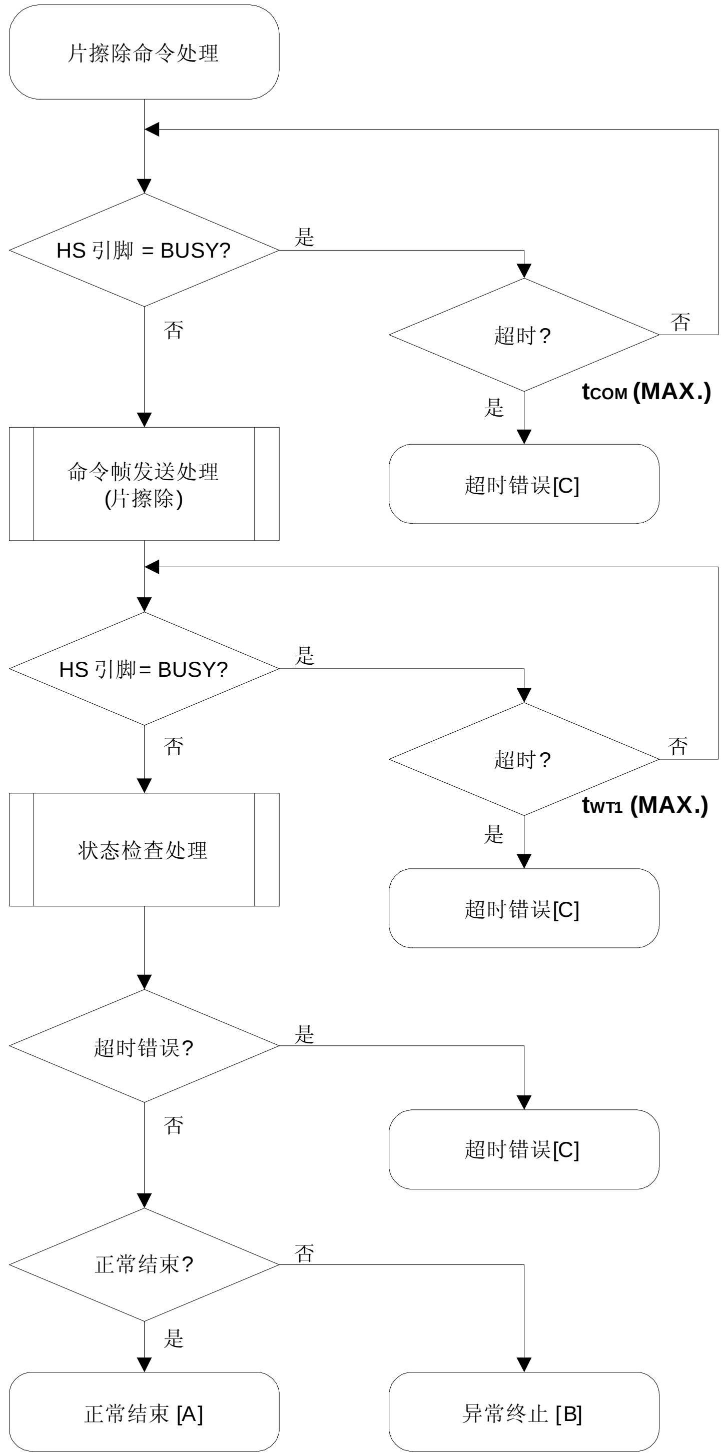
<1>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2>　命令帧发送处理发送片擦除命令。
<3>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT1}(MAX.))。
<4>　状态检查处理获得状态帧。
<5>　根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

7.7.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，片擦除被正常完成。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中片擦除被禁止。 • 在安全设置中重写引导块被禁止。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
	FLMD 错误	18H	发生擦除错误。
	写错误	1CH	
	MRG10 错误	1AH	
	MRG11 错误	1BH	
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。

7.7.4 流程图



片擦除命令处理
HS 引脚 = BUSY?
是
否
超时?
否
是
tCOM (MAX.)
命令帧发送处理
(片擦除)
超时错误[C]
HS 引脚= BUSY?
是
否
超时?
否
twT1 (MAX.)
是
状态检查处理
超时错误[C]
超时错误?
是
否
超时错误[C]
正常结束?
否
是
正常结束 [A]
异常终止 [B]

7.7.5　程序举例说明

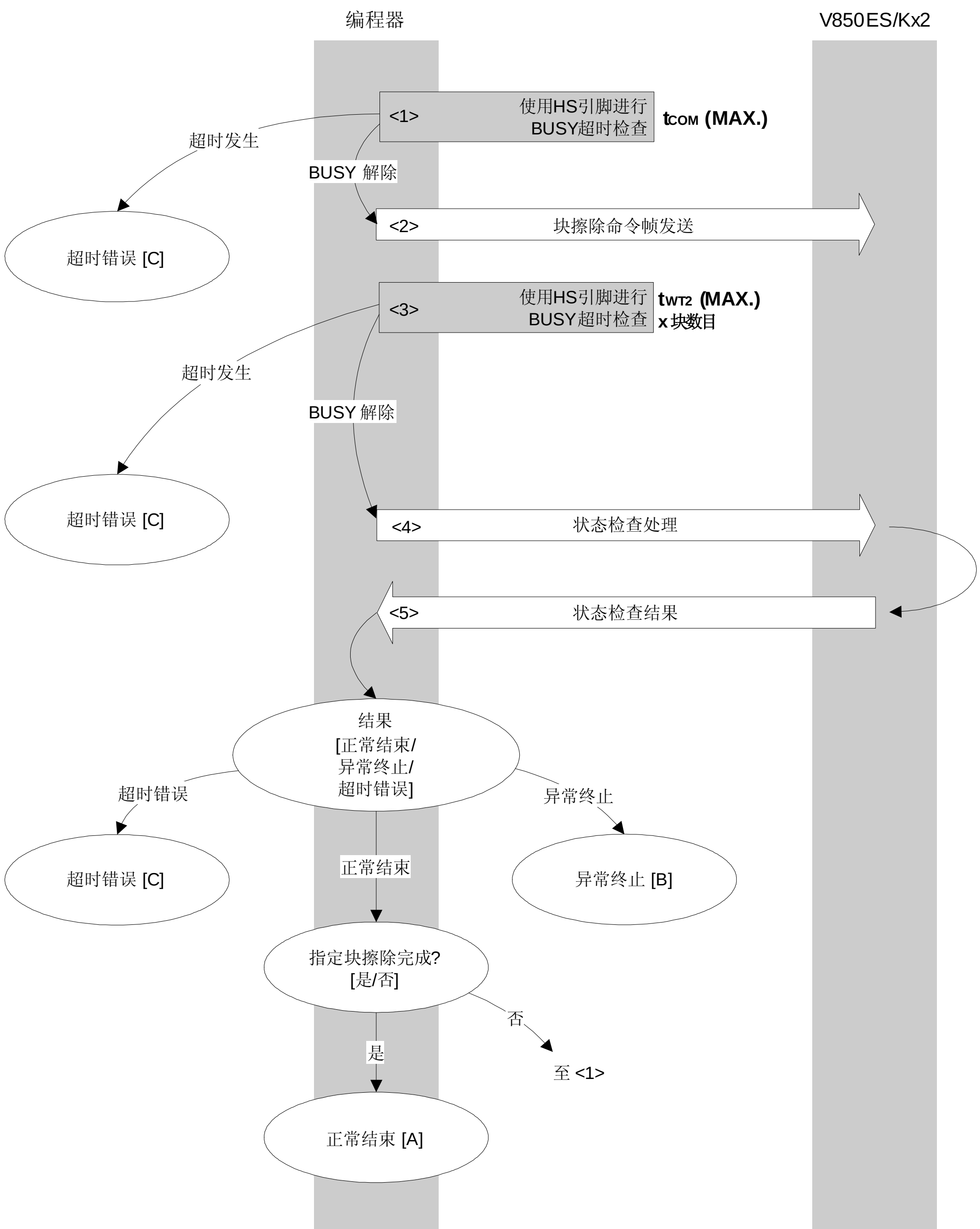
以下是片擦除处理的程序举例说明：

```
/*************************************************************/
/*                                                     */
/* 擦除全部 (整片) 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;                   //检测到超时

        if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))
                                                // 发送 "片擦除" 命令
                return  rc;                     // 情况 [C]

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

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

7.8　块擦除命令

7.8.1　处理程序流程图

块擦除命令处理程序



7.8.2　处理程序的描述说明

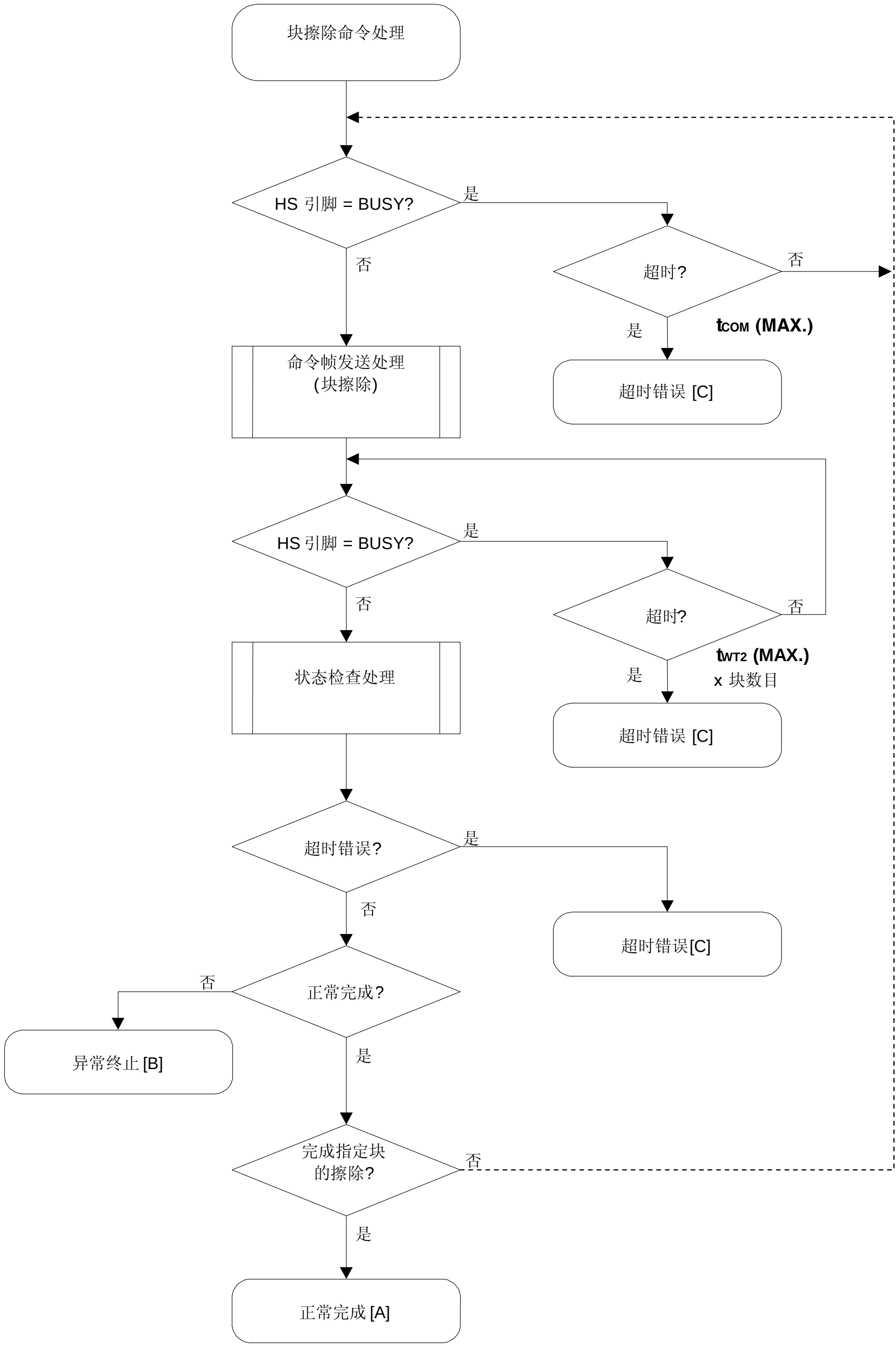
<1>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2>　命令帧发送处理发送块擦除命令。
<3>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT2}(MAX.) × 块数目)。
<4>　状态检查处理获得状态帧。
<5>　根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

7.8.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，块擦除被正常完成。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址超出 Flash 存储器的范围。 • 指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中块擦除被禁止。 • 在安全设置中引导块重写被禁止时, 引导区处于所指定范围内。 • 在安全设置中片擦除被禁止。 • 在安全设置中编程被禁止。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
	MRG10 错误	1AH	发生擦除错误。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。

7.8.4 流程图


块擦除命令处理
HS 引脚 = BUSY?
是
否
超时?
否
是
tCOM (MAX.)
超时错误 [C]
命令帧发送处理
(块擦除)
HS 引脚 = BUSY?
是
否
超时?
否
是
tWT2 (MAX.)
x 块数目
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]
完成指定块
的擦除?
否
是
正常完成 [A]

7.8.5　程序举例说明

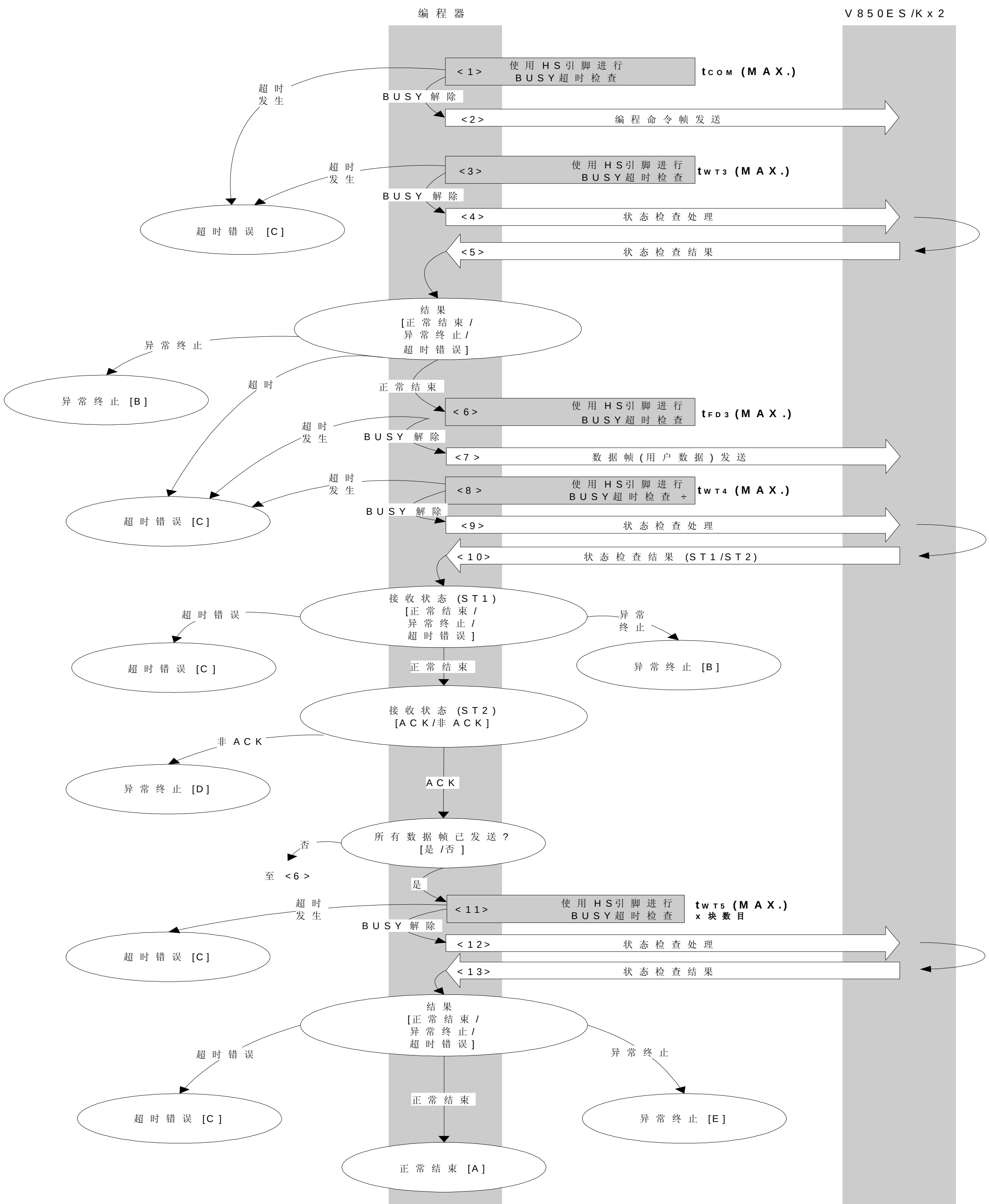
以下是针对一块的块擦除命令处理的程序举例说明：

```
/*****************************************************************/
/*                                                               */
/* 擦除块命令(CSI-HS)                                            */
/*                                                               */
/*****************************************************************/
/*   [i] u8 block          ... 块编码                            */
/*   [r] u16               ...错误码                             */
/*****************************************************************/
u16        fl_hs_erase_blk(u16 sblk, u16 eblk)
{
  u16      rc;
  u16      block_num;
  u32      top, bottom;

  top = get_top_addr(sblk);            // 获得开始块的开始地址
  bottom = get_bottom_addr(eblk);      // 获得结束块的结束地址

  set_range_prm(fl_cmd_prm, top, bottom);    // 设置 SAH/SAM/SAL, EAH/EAM/EAL
  block_num = eblk - sblk + 1;

  if (hs_busy_to(tCOM_MAX))
           return    FLC_HSTO_ERR;                          // 检测到超时 :情况 [C]

  if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm))
                                                   // 发送 "块擦除" 命令
           return    rc;                           // 情况 [C]

  if (hs_busy_to(tWT2_MAX * block_num))
           return    FLC_HSTO_ERR;                          // 检测到超时 :情况 [C]

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

7.9　编程命令

7.9.1　处理程序流程图

编程命令处理程序



7.9.2　处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2> 命令帧发送处理发送编程命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT3}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD3}(MAX.))。
<7> 数据帧发送处理发送用户数据。
<8> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT4}(MAX.))。
<9> 状态检查处理获得状态帧。
<10> 根据状态检查处理的结果执行以下处理 (状态码 (ST1/ST2)) (同样参见处理程序流程图和流程图)。

当 ST1 = 异常终止时：	异常终止 [B]
当 ST1 = 超时错误时：	超时错误[C]被返回
当 ST1 = 正常完成时：	根据 ST2 的值执行以下处理

- 当 ST2 ≠ ACK 时： 异常终止 [D]
- 当 ST2 = ACK 时： 当所有的用户数据发送完成时进入 <11>
  如果仍有剩余的用户数据需要发送, 处理从<6>重复执行程序。

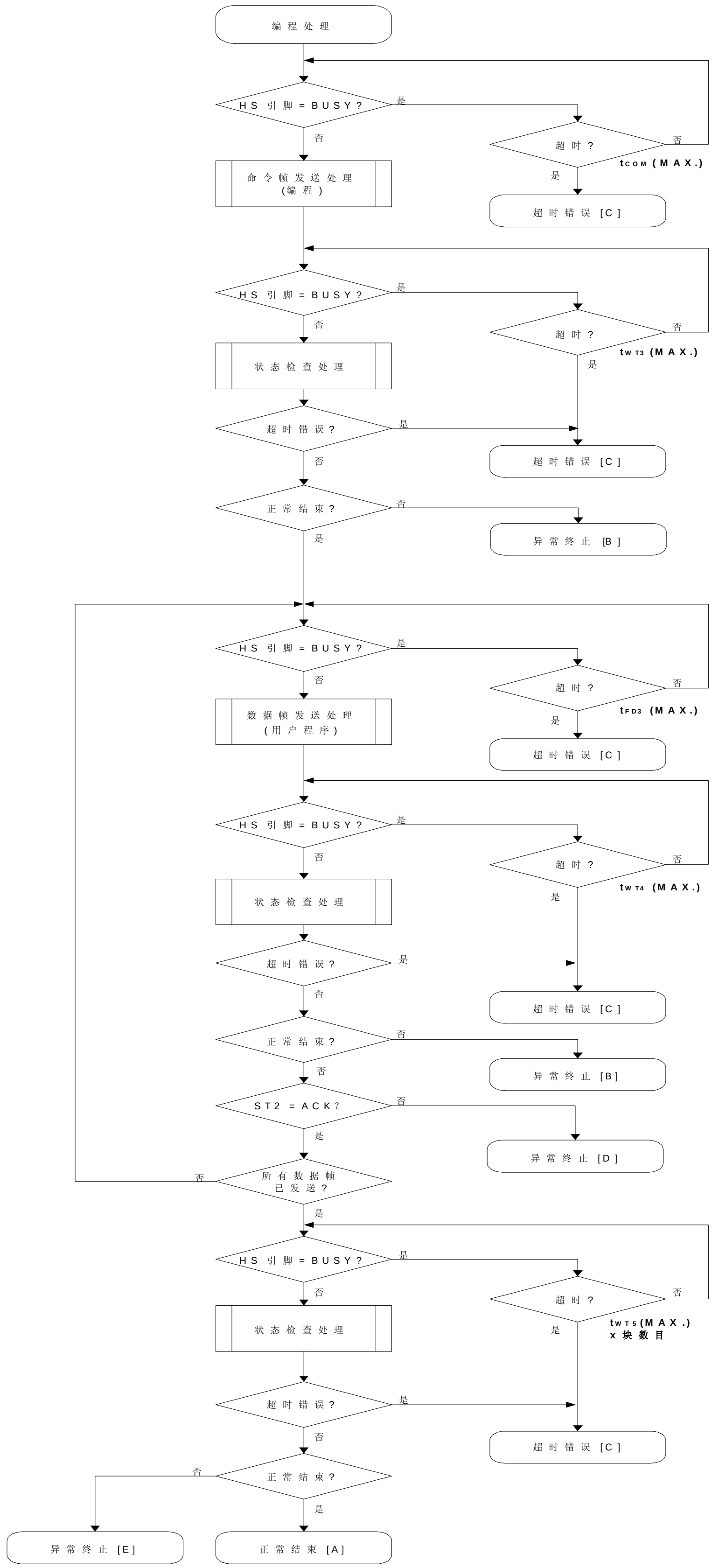
<11> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT5} (MAX.) × 块数目)。
<12> 状态检查处理获得状态帧。
<13> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A] (表明写完成后内部验证检查正常完成)
当处理结束异常时：	异常终止 [E] (表明写完成后内部验证检查没有正常完成)
当发生超时错误时：	超时错误[C]被返回

7.9.3　处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行用户数据被正常写入。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址不是该块的开始/结束地址。 • 数据长度不是 64 字单位 。
	校验和错误	07H	发送命令帧的校验和不匹配。
	保护错误	10H	• 在安全设置中写入被禁止。 • 在安全设置中重写引导块被禁止。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时 。
异常终止 [D]	FLMD 错误	18H (ST2)	发生写错误。
	写错误	1CH (ST2)	发生写错误。
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

7.9.4　流程图


编程处理
HS 引脚 = BUSY?
是
否
超时?
否
tCOM (MAX.)
是
命令帧发送处理
(编程)
超时错误 [C]
HS 引脚 = BUSY?
是
否
超时?
否
tWT3 (MAX.)
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止 [B]
HS 引脚 = BUSY?
是
否
超时?
否
tFD3 (MAX.)
是
数据帧发送处理
(用户程序)
超时错误 [C]
HS 引脚 = BUSY?
是
否
超时?
否
tWT4 (MAX.)
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
否
异常终止 [B]
ST2 = ACK?
否
是
异常终止 [D]
所有数据帧
已发送?
否
是
HS 引脚 = BUSY?
是
否
超时?
否
tWT5(MAX.)
x 块数目
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止 [E]
正常结束 [A]

7.9.5 程序举例说明

以下是编程命令处理的程序举例说明：

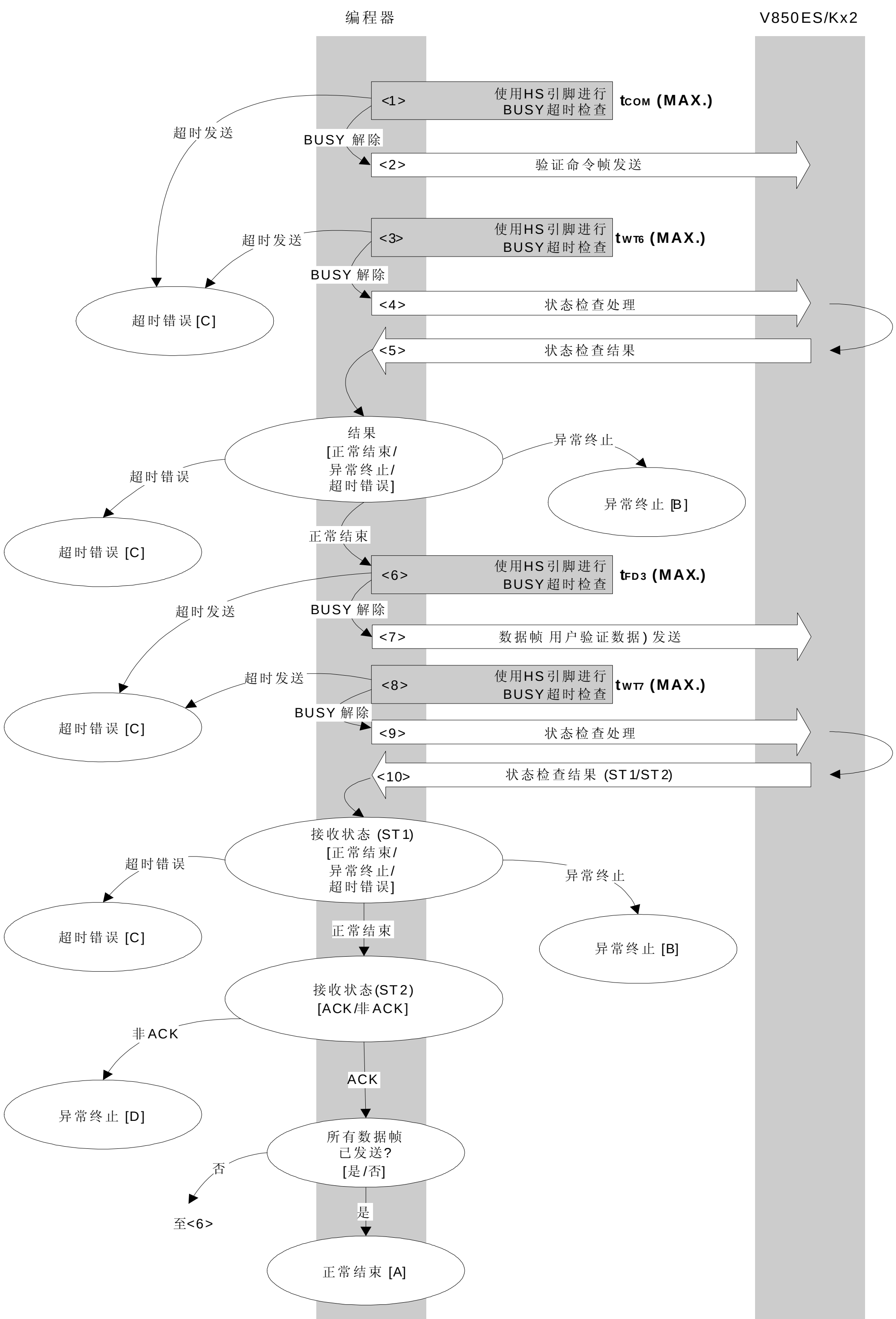
```
/****************************************************************/
/*                                                              */
/*    写命令(CSI-HS)                                            */
/*                                                              */
/****************************************************************/
/*    [i] u32 top          ... 开始地址                         */
/*    [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;
   u16      block_num;

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

   block_num = get_block_num(top, bottom);   // 获得块编码

   /*********************************************/
   /*    发送命令并检查状态                      */
   /*********************************************/
   if (hs_busy_to(tCOM_MAX))
            return    FLC_HSTO_ERR;                      // 检测到超时

   if (rc = put_cmd_hs(FL_COM_WRITE, 7, fl_cmd_prm))  // 发送 "编程" 命令
            return    rc;                                        // 检测到超时

   if (hs_busy_to(tWT3_MAX))
            return    FLC_HSTO_ERR;                      // 检测到超时

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

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

   while(1){
            // make send data frame
            if ((bottom - send_head) > 256){           // 剩余长度> 256 ?
                     is_end = false;                   // 是，不是结束帧
                     send_size  = 256;             // 发送长度= 256 字节
            }
```

```
            else{
                    is_end = true;
                    send_size = bottom - send_head + 1;
                                            // 发送长度= (bottom - send_head+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))           // 发送数据帧前检查超时
                    return    FLC_HSTO_ERR;                     // 检测到超时

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

            if (hs_busy_to(tWT4_MAX))
                    return    FLC_HSTO_ERR;                     // 检测到超时

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

    }
    /************************************************/
    /*            检查内部验证                                  */
    /************************************************/
    if (hs_busy_to(tWT5_MAX_HS * block_num))
            return    FLC_HSTO_ERR;                 // 检测到超时

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

  }
```

7.10 验证命令

7.10.1 处理程序流程图

验证命令处理程序



7.10.2 处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。

<2> 命令帧发送处理发送验证命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT6}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：　　进入 <6>
当处理结束异常时：　　异常终止 [B]
当发生超时错误时：　　超时错误[C]被返回

<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD3}(MAX.))。

<7> 数据帧发送处理发送用户验证数据。

<8> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT7}(MAX.))。
<9> 状态检查处理获得状态帧。
<10> 根据状态检查处理的结果执行以下处理 (状态码 (ST1/ST2)) (同时参见处理程序流程图和流程图)。

当 ST1 = 异常终止时：　　异常终止 [B]

当 ST1 = 超时错误时：　　超时错误[C]被返回

当 ST1 = 正常完成时：　　根据 ST2 的值执行以下处理

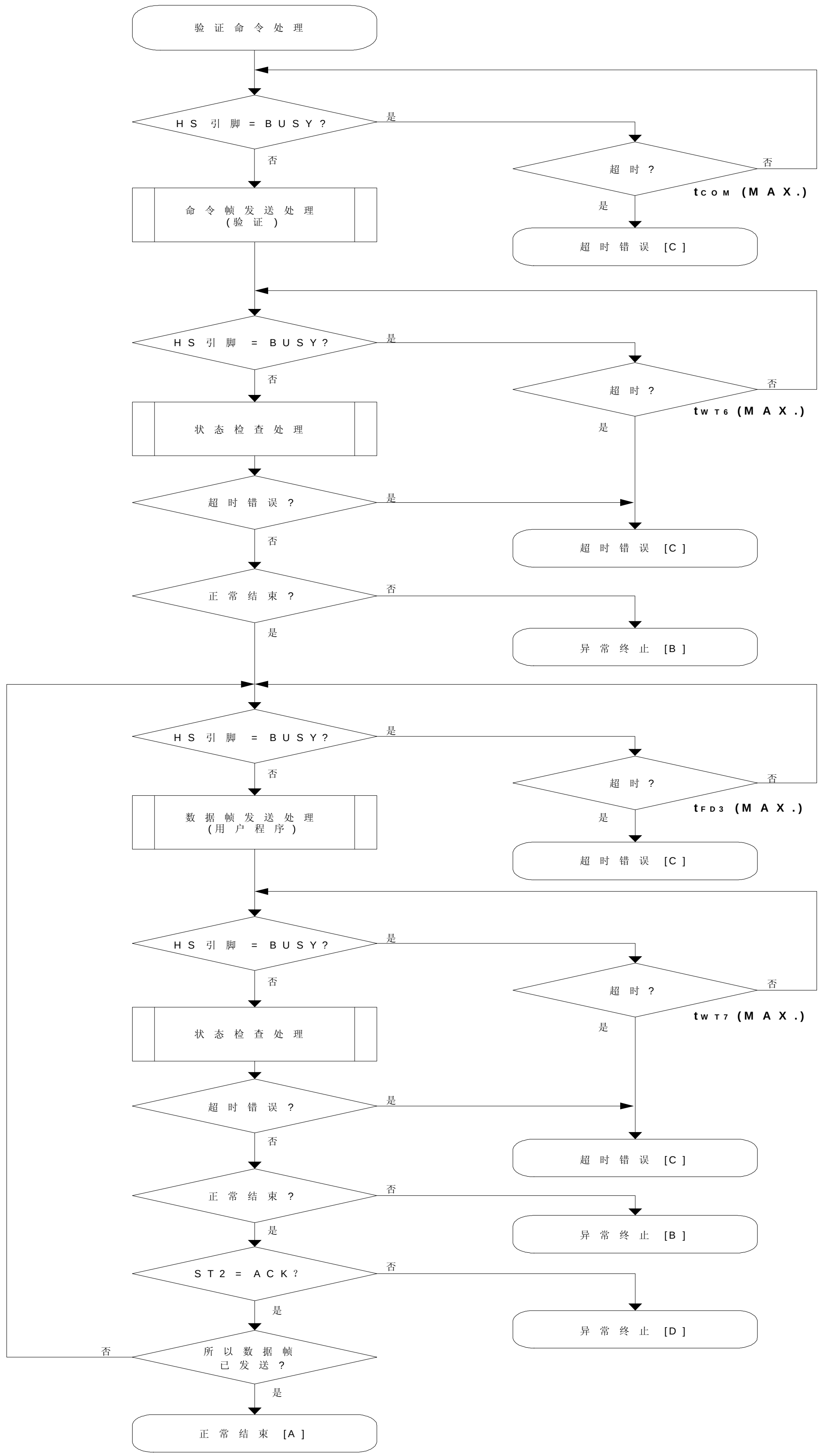
- 当 ST2 = ACK 时：　如果全部的数据帧发送完成，处理正常结束[A]
  如果仍有剩余数据帧需要发送，处理从<6>重复执行程序

- 当 ST2 ≠ ACK 时：　异常终止 [D]

7.10.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，验证被正常完成。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。
异常终止 [D]	验证错误	0FH (ST2)	验证失败或发生其他错误。

7.10.4 流程图


验证命令处理
HS 引脚 = BUSY？
是
否
超时？
否
tCOM (MAX.)
是
命令帧发送处理
(验证)
超时错误 [C]
HS 引脚 = BUSY？
是
否
超时？
否
tWT6 (MAX.)
是
状态检查处理
超时错误？
是
否
超时错误 [C]
正常结束？
否
是
异常终止 [B]
HS 引脚 = BUSY？
是
否
超时？
否
tFD3 (MAX.)
是
数据帧发送处理
(用户程序)
超时错误 [C]
HS 引脚 = BUSY？
是
否
超时？
否
tWT7 (MAX.)
是
状态检查处理
超时错误？
是
否
超时错误 [C]
正常结束？
否
是
异常终止 [B]
ST2 = ACK？
否
是
异常终止 [D]
所以数据帧
已发送？
否
是
正常结束 [A]

7.10.5 程序举例说明

以下是验证命令处理的程序举例说明：

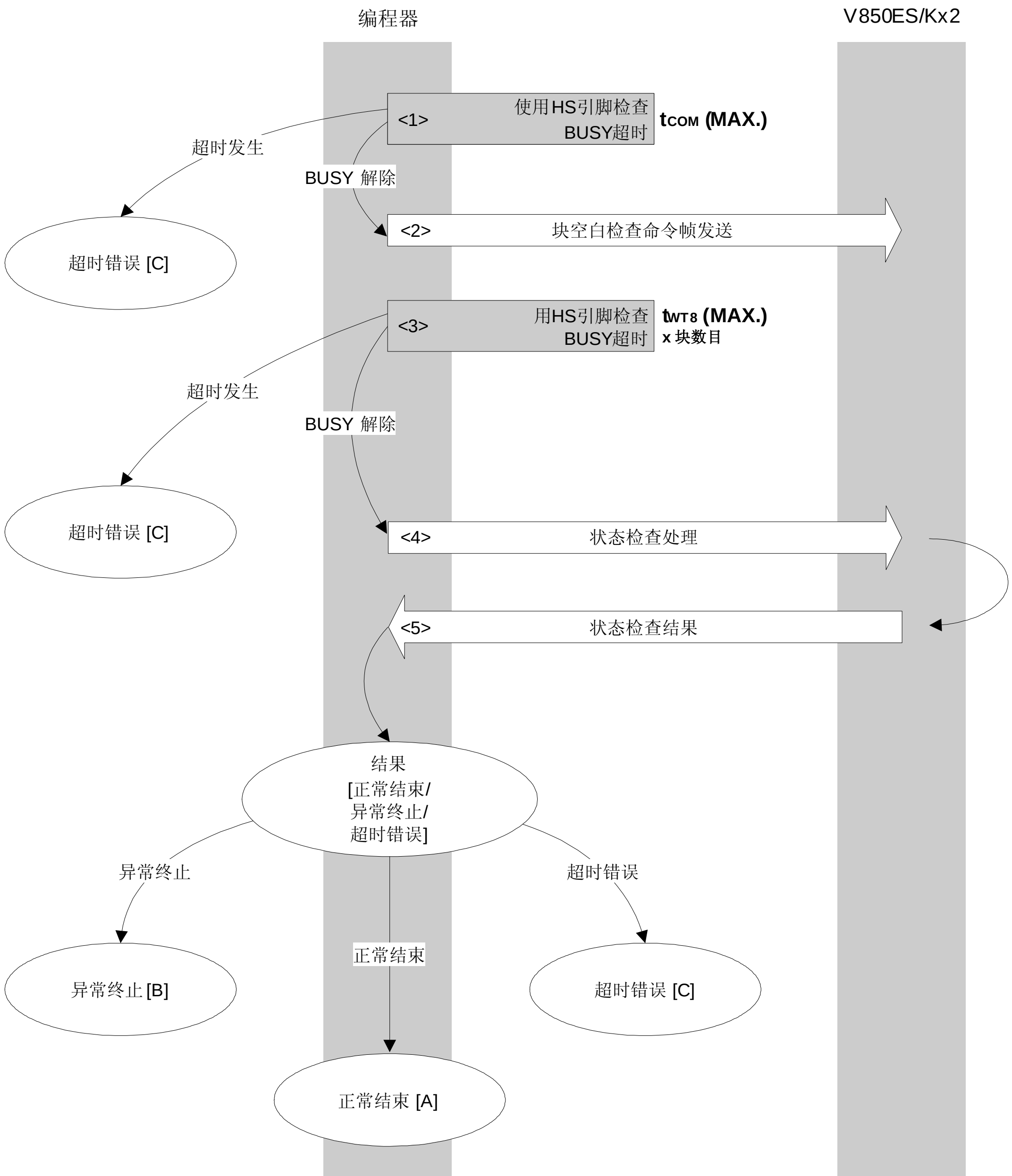
```
/**************************************************************/
/*                                                            */
/*   验证命令(CSI-HS)                                          */
/*                                                            */
/**************************************************************/
/*   [i] u32 top        ... 开始地址                           */
/*   [i] u32 bottom     ... 结束地址                           */
/*   [i] u8  *buf       ... 指向验证数据缓冲区                 */
/*[r] u16               ...错误码                              */
/**************************************************************/
u16     fl_hs_verify(u32 top, u32 bottom, u8 *buf)
{
  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

  /**********************************************/
  /*         发送命令并检查状态                 */
  /**********************************************/

  if (hs_busy_to(tCOM_MAX))
          return    FLC_HSTO_ERR;                   // 检测到超时

  if (rc = put_cmd_hs(FL_COM_VERIFY, 7, fl_cmd_prm))     // 发送 "验证" 命令
          return    rc;                      // 检测到错误

  if (hs_busy_to(tWT6_MAX))
          return    FLC_HSTO_ERR;                   // 检测到超时

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

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

  while(1){

          // make send data frame
          if ((bottom - send_head) > 256){          // 剩余长度> 256 ?
                  is_end = false;                   // 是，不是结束帧
                  send_size  = 256;               // 发送长度= 256 字节
          }
```

```
        else{
            is_end = true;
            send_size = bottom - send_head + 1;
                                // 发送长度= (bottom - send_head+1) 字节

        }
        memcpy(fl_txdata_frm, 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;                 // 检测到超时

        if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                                                      // 发送用户数据
            return    rc;                       // 检测到错误
        if (hs_busy_to(tWT7_MAX))
            return    FLC_HSTO_ERR;                 // 检测到超时

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

7.11 块空白检查命令

7.11.1 处理程序流程图

块空白检查命令处理程序



7.11.2 处理程序的描述说明

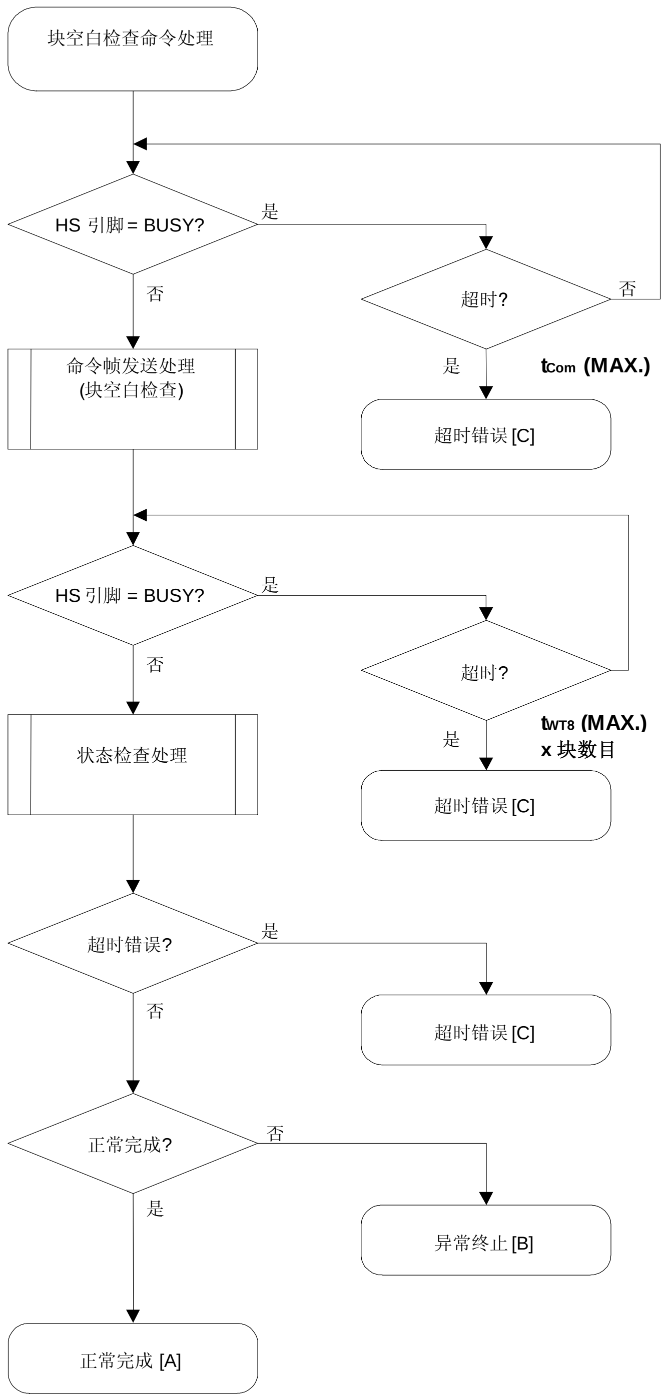
<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2> 命令帧发送处理发送块空白检查命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT8}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

7.11.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，所有的指定块是空白。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址超出 Flash 存储器的范围。 • 指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
	MRG11 错误	1BH	Flash 存储器中的指定块不是空白块。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。

7.11.4 流程图


块空白检查命令处理
HS 引脚 = BUSY?
是
否
超时?
否
是
t_{Com} (MAX.)
命令帧发送处理
(块空白检查)
超时错误 [C]
HS 引脚 = BUSY?
是
否
超时?
是
t_{WT8} (MAX.)
x 块数目
状态检查处理
超时错误 [C]
超时错误?
是
否
超时错误 [C]
正常完成?
否
是
异常终止 [B]
正常完成 [A]

7.11.5 程序举例说明

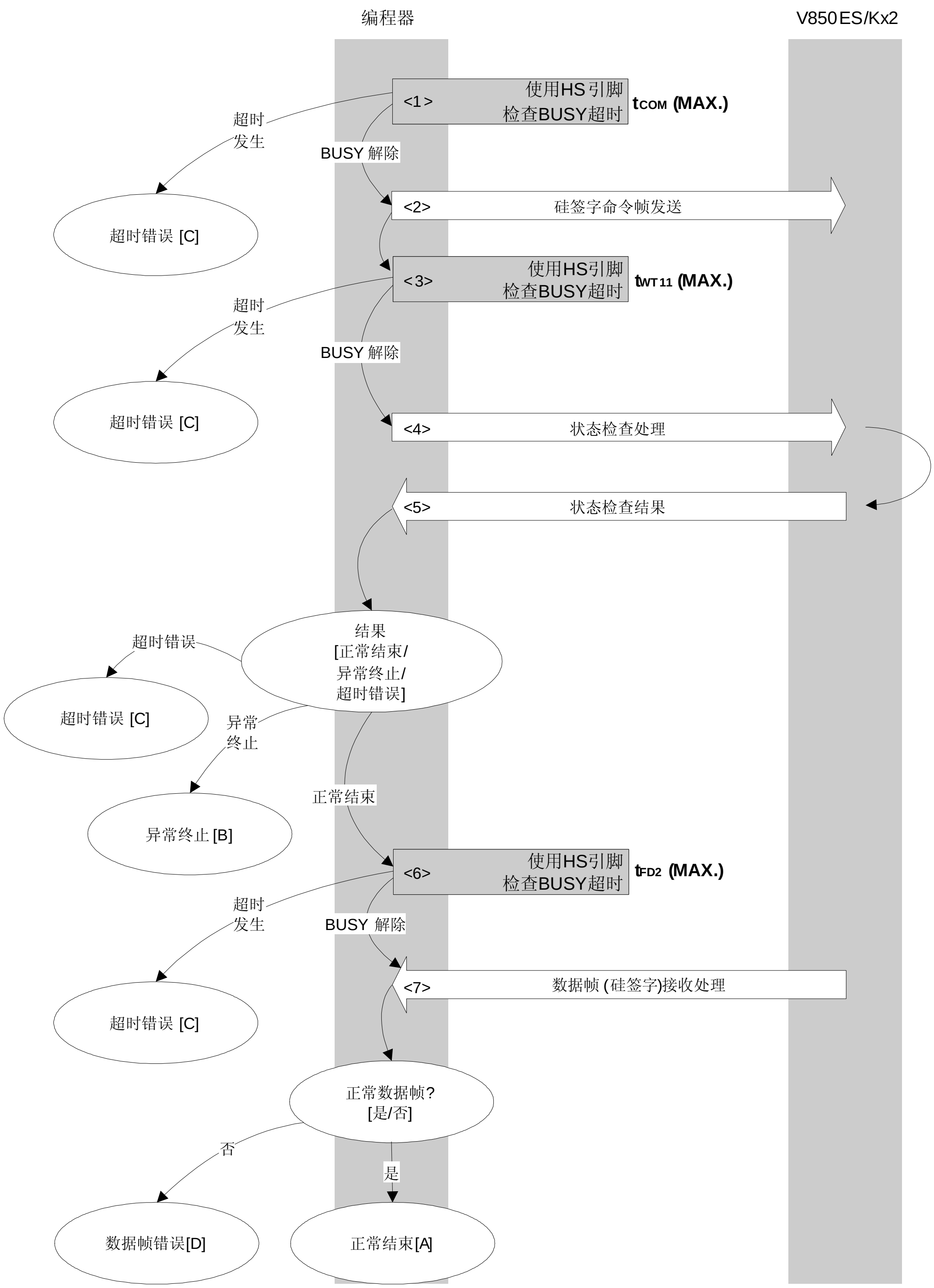
以下是针对一块的块空白检查命令处理的程序举例说明：

```
/*****************************************************************/
/*                                                               */
/*   块空白检查命令(CSI-HS)                                      */
/*                                                               */
/*****************************************************************/
/*   [i] u16 block       ... 块编码                              */
/*   [r] u16             ... 错误码                              */
/*****************************************************************/
u16        fl_hs_blk_blank_chk(u32 top, u32 bottom)
{
  u16      rc;
  u16      block_num;

  set_range_prm(fl_cmd_prm, top, bottom);    // 设置 SAH/SAM/SAL, EAH/EAM/EAL
  block_num = get_block_num(top, bottom);    // 获得块编码

  if (hs_busy_to(tCOM_MAX))
           return    FLC_HSTO_ERR;                    // 检测到超时 :情况 [C]

  if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm))
                                                  // 发送 "块空白检查" 命令
           return    rc;                  // 情况 [C]

  if (hs_busy_to(tWT8_MAX * block_num))
           return    FLC_HSTO_ERR;                    // 检测到超时 :情况 [C]

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

7.12 ‘硅标记’命令

7.12.1 处理程序流程图

硅签字命令处理程序



7.12.2 处理程序的描述说明

<1>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2>　命令帧发送处理发送'硅标记'命令。
<3>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT11}(MAX.))。
<4>　状态检查处理获得状态帧。
<5>　根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

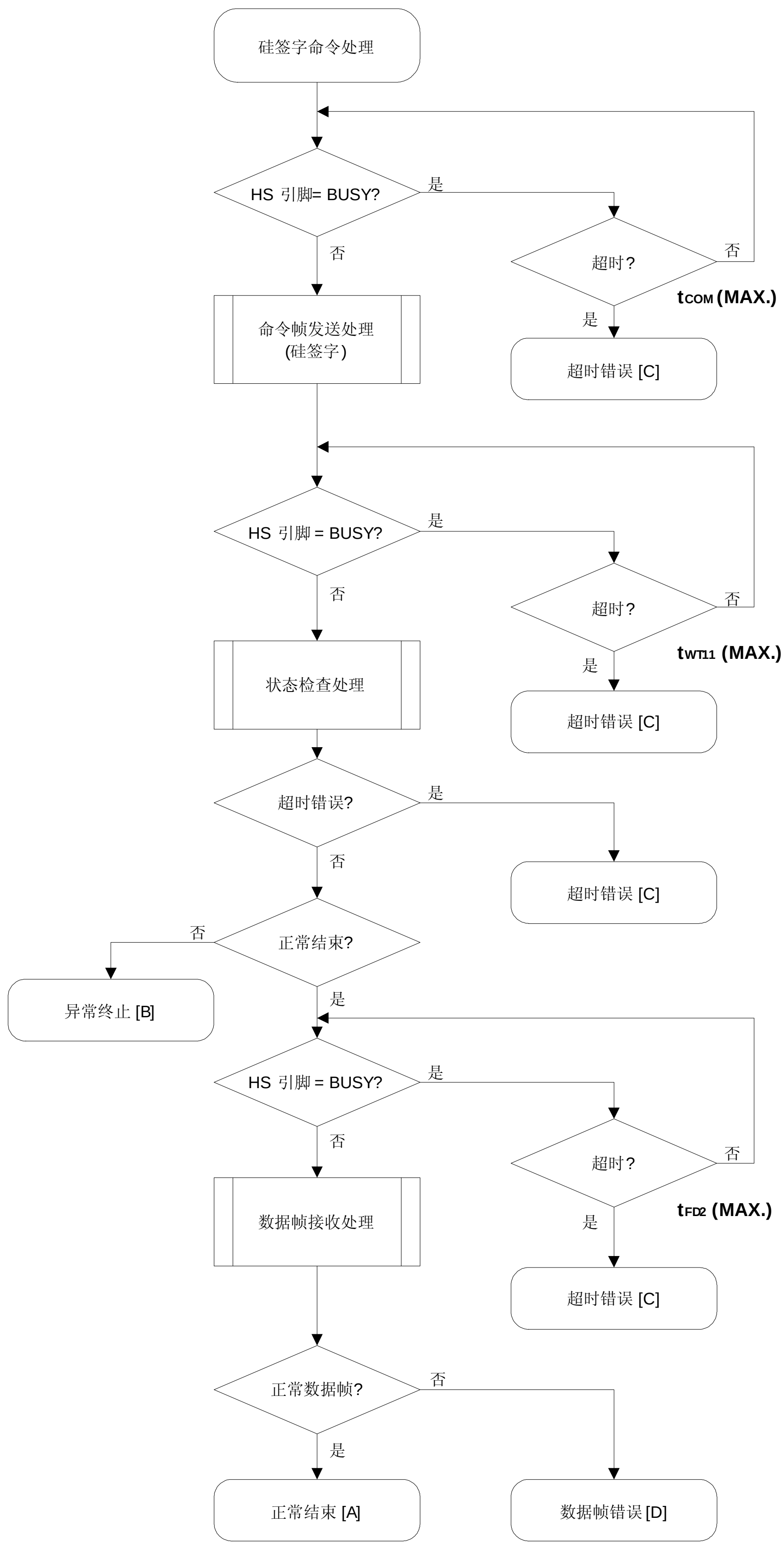
<6>　用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
　　如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD2}(MAX.))。
<7>　检查接收的数据('硅标记'数据)。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，'硅标记'被正常获得。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时 。
数据帧错误 [D]		–	所接收的'硅标记'数据帧校验和不匹配。

7.12.4 流程图


硅签字命令处理
HS 引脚= BUSY?
是
否
超时?
否
是
tCOM (MAX.)
超时错误 [C]
命令帧发送处理
(硅签字)
HS 引脚 = BUSY?
是
否
超时?
否
是
tWT11 (MAX.)
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止 [B]
HS 引脚 = BUSY?
是
否
超时?
否
是
tFD2 (MAX.)
超时错误 [C]
数据帧接收处理
正常数据帧?
否
是
正常结束 [A]
数据帧错误 [D]

7.12.5 程序举例说明

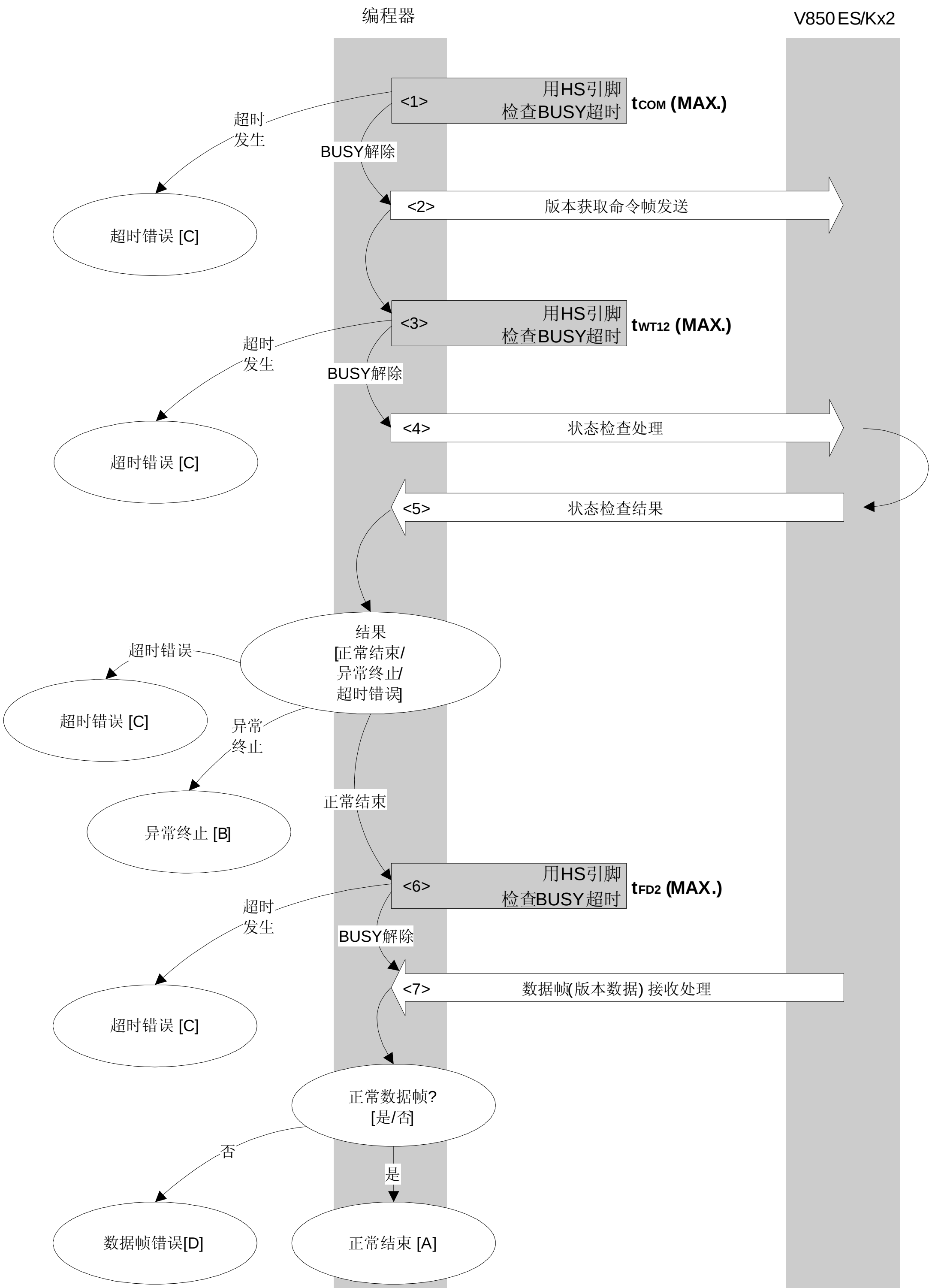
以下是‘硅标记’命令处理的程序举例说明：

```
/******************************************************************/
/*                                                          */
/* 获得‘硅标记’命令(CSI-HS)                                   */
/*                                                          */
/******************************************************************/
/* [i] u8 *sig... 指向标记存储区                             */
/* [r] u16    ... 错误码                                     */
/******************************************************************/
u16             fl_hs_getsig(u8 *sig)
{
        u16     rc;

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]

        if (rc = put_cmd_hs(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm))
                                                // 发送 "硅标记" 命令
                return  rc;                     // 检测到错误 :情况 [C]

        if (hs_busy_to(tWT11_MAX))
                return  FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]


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


        if (hs_busy_to(tFD2_MAX))
                return  FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]

        rc = get_dfrm_hs(fl_rxdata_frm);        // 获得标记数据

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

        memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                // 拷贝标记数据

        return  rc;                             // 情况 [A]

}
```

7.13 获取版本信息命令

7.13.1 处理程序流程图

获取版本命令处理程序



7.13.2 处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2> 命令帧发送处理发送获取版本信息命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT12}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

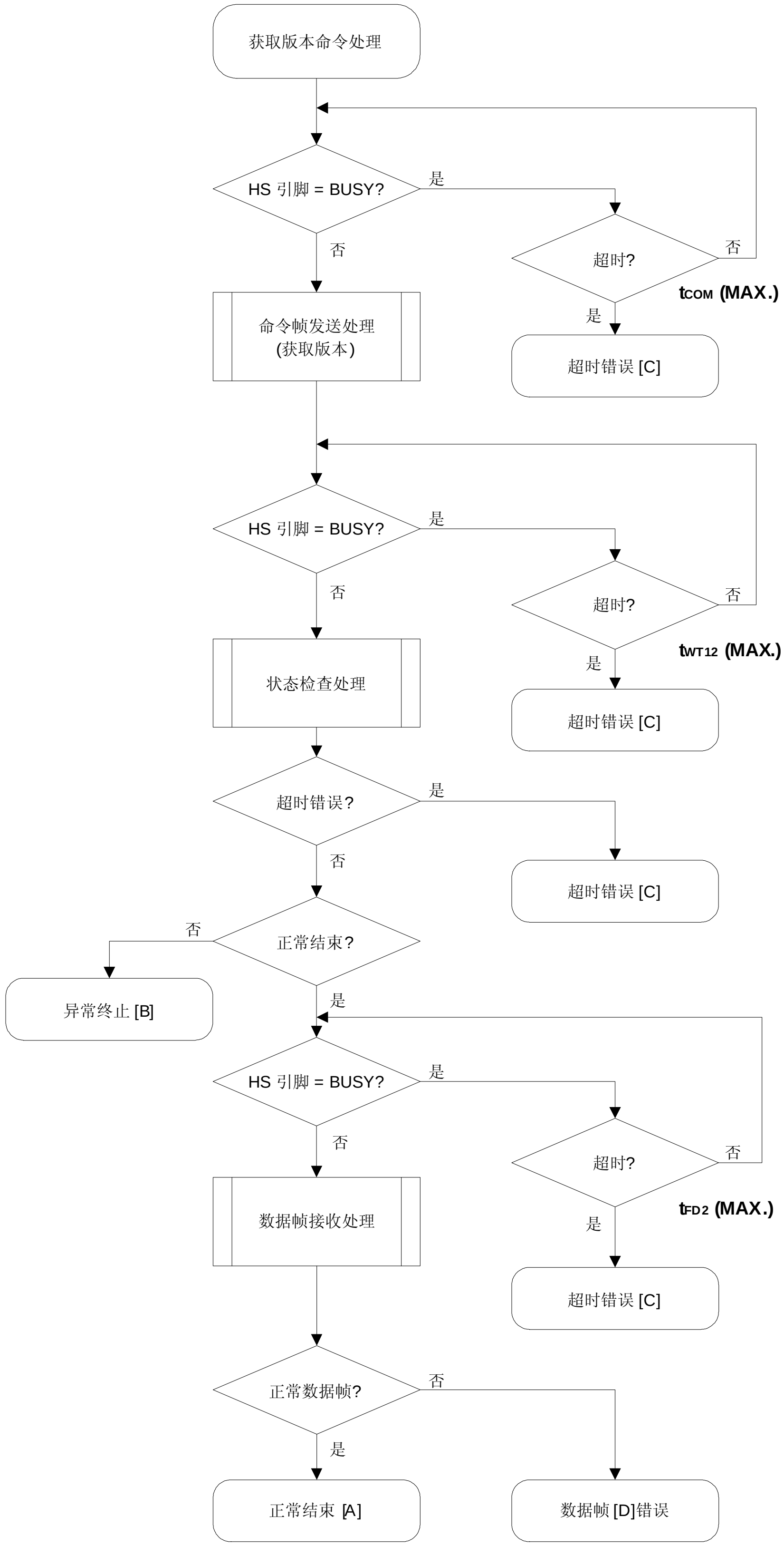
<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD2}(MAX.))。
<7> 检查接收的数据帧(版本数据) 。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，版本数据被正常获得。
异常终止 [B]	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时 。
数据帧错误 [D]		–	所接收的版本数据数据帧校验和不匹配。

7.13.4 流程图


获取版本命令处理
HS 引脚 = BUSY?
是
否
超时?
否
是
tCOM (MAX.)
命令帧发送处理
(获取版本)
超时错误 [C]
HS 引脚 = BUSY?
是
否
超时?
否
是
tWT12 (MAX.)
状态检查处理
超时错误 [C]
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止 [B]
HS 引脚 = BUSY?
是
否
超时?
否
是
tFD2 (MAX.)
数据帧接收处理
超时错误 [C]
正常数据帧?
否
是
正常结束 [A]
数据帧 [D]错误

7.13.5 程序举例说明

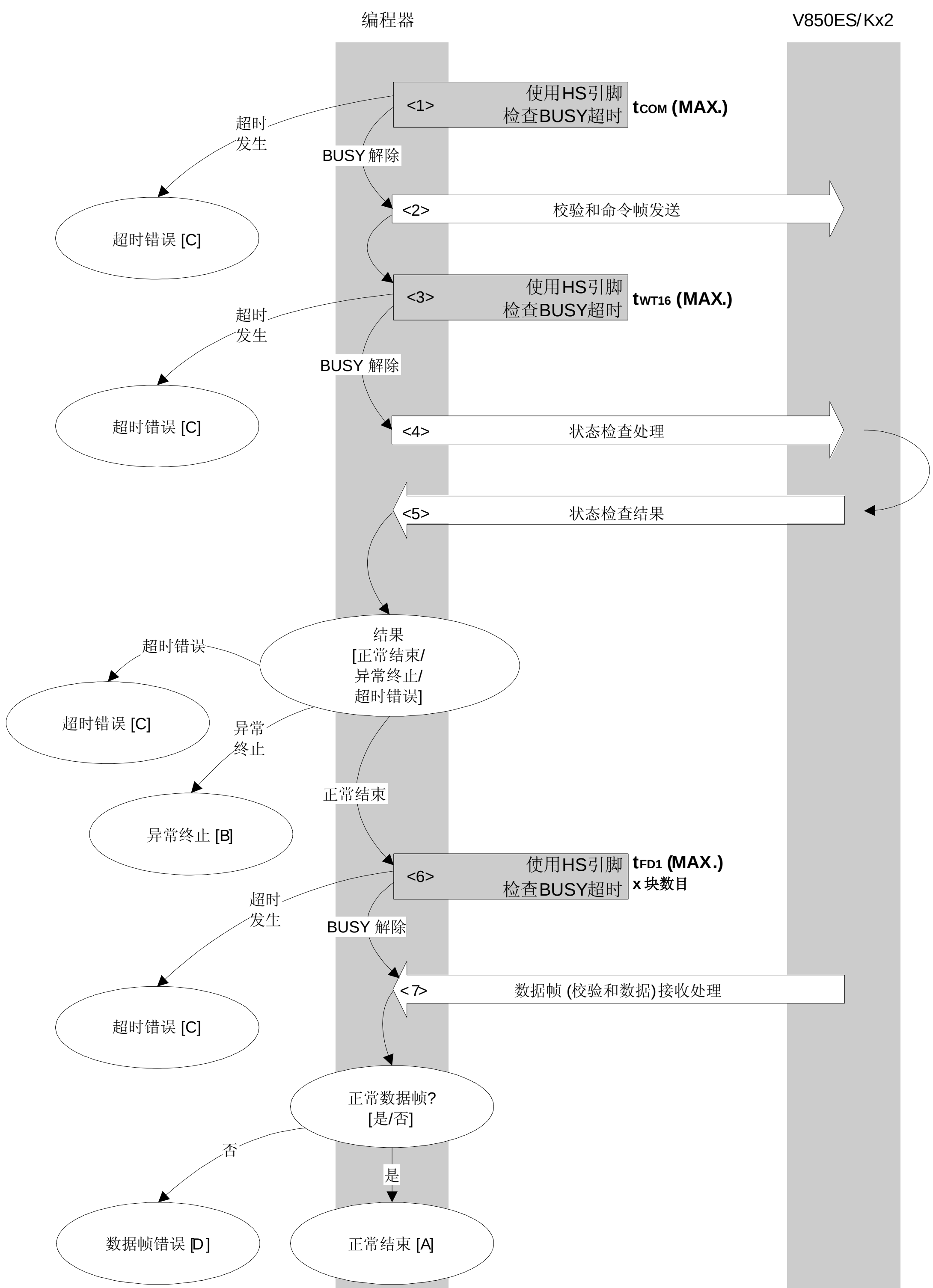
以下是获取版本信息命令处理的程序举例说明：

```
/****************************************************************/
/*                                                      */
/* 获取器件/固件版本信息命令 (CSI-HS)                    */
/*                                                      */
/****************************************************************/
/* [i] u8 *buf    ... 指向版本数据存储区      */
/* [r] u16        ... 错误码                  */
/****************************************************************/
u16          fl_hs_getver(u8 *buf)
{
        u16     rc;

        if (hs_busy_to(tCOM_MAX))
                return  FLC_HSTO_ERR;                 // 检测到超时 :情况 [C]

        if (rc = put_cmd_hs(FL_COM_GET_VERSION, 1, fl_cmd_prm))
                                              // 发送 "版本信息获取" 命令
                return  rc;                   // 检测到错误 :情况 [C]

        if (hs_busy_to(tWT12_MAX))
                return  FLC_HSTO_ERR;                 // 检测到超时 :情况 [C]

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

        if (hs_busy_to(tFD2_MAX))
                return  FLC_HSTO_ERR;                 // 检测到超时 :情况 [C]

        rc = get_dfrm_hs(fl_rxdata_frm);      // 获得标记数据
        switch(rc) {
                case    FLC_NO_ERR:                   break;  // 继续
//              case    FLC_HSTO_ERR:         return  rc;     break;  // 情况 [C]
                default:        return  rc;   break;  // 情况 [D]
        }
        memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// 拷贝版本数据
        return  rc;                           // 情况 [A]

}
```

7.14 校验和命令

7.14.1 处理程序流程图

校验和命令处理程序



7.14.2 处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。

<2> 命令帧发送处理发送校验和命令。

<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT16}(MAX.))。

<4> 状态检查处理获得状态帧。

<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD1} (MAX.) × 块数目)。

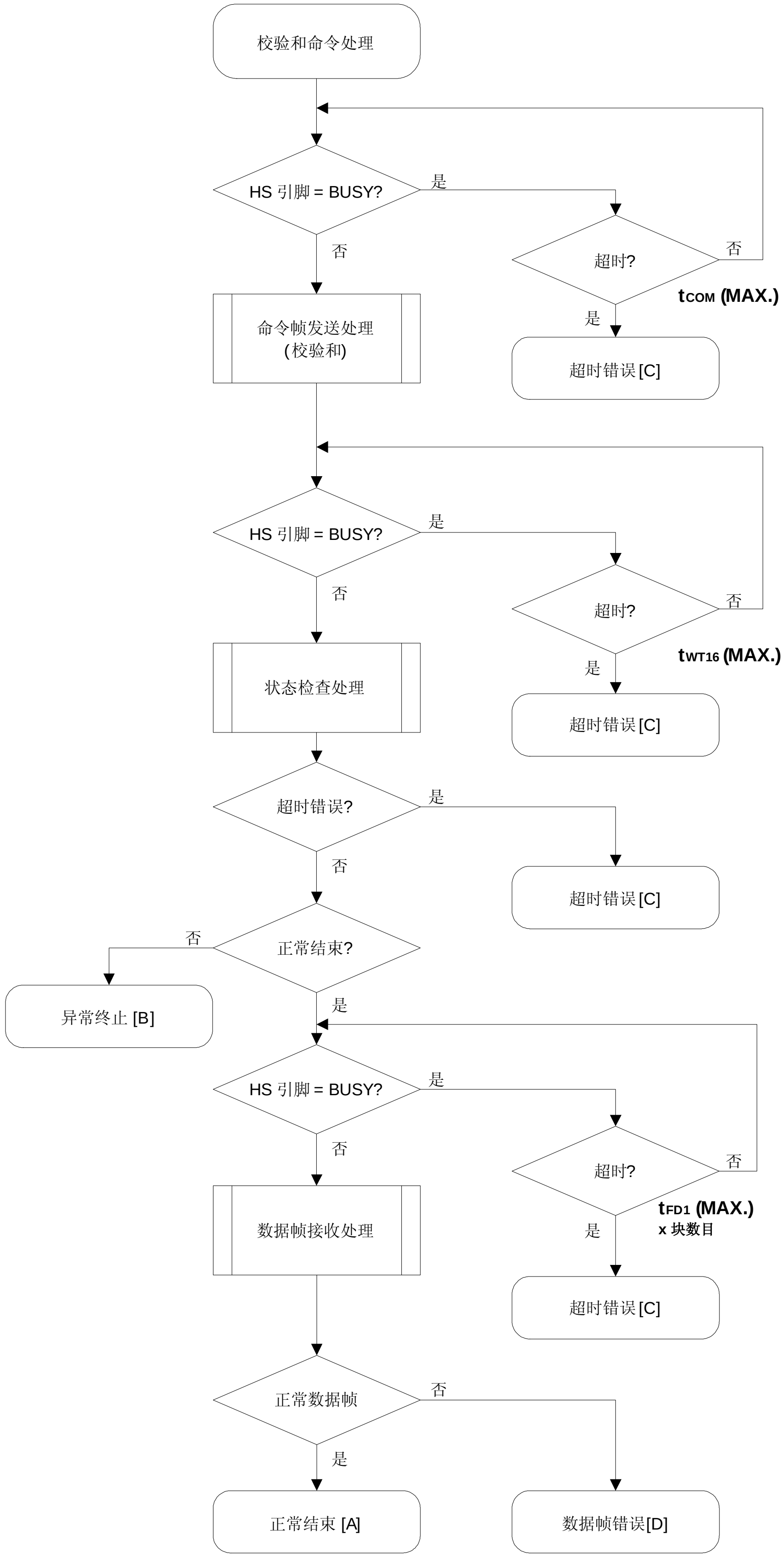
<7> 检查接收的数据帧(校验和数据) 。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，校验和数据被正常获得。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是从 Flash 存储器的顶端开始的块单元(2 KB)固定地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时 。
数据帧错误 [D]		–	所接收的版本数据数据帧的校验和不匹配。

7.14.4 流程图


校验和命令处理
HS 引脚 = BUSY?
是
否
超时?
否
tCOM (MAX.)
是
超时错误 [C]
命令帧发送处理
(校验和)
HS 引脚 = BUSY?
是
否
超时?
否
tWT16 (MAX.)
是
超时错误 [C]
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
异常终止 [B]
是
HS 引脚 = BUSY?
是
否
超时?
否
tFD1 (MAX.)
x 块数目
是
数据帧接收处理
超时错误 [C]
正常数据帧
否
是
正常结束 [A]
数据帧错误[D]

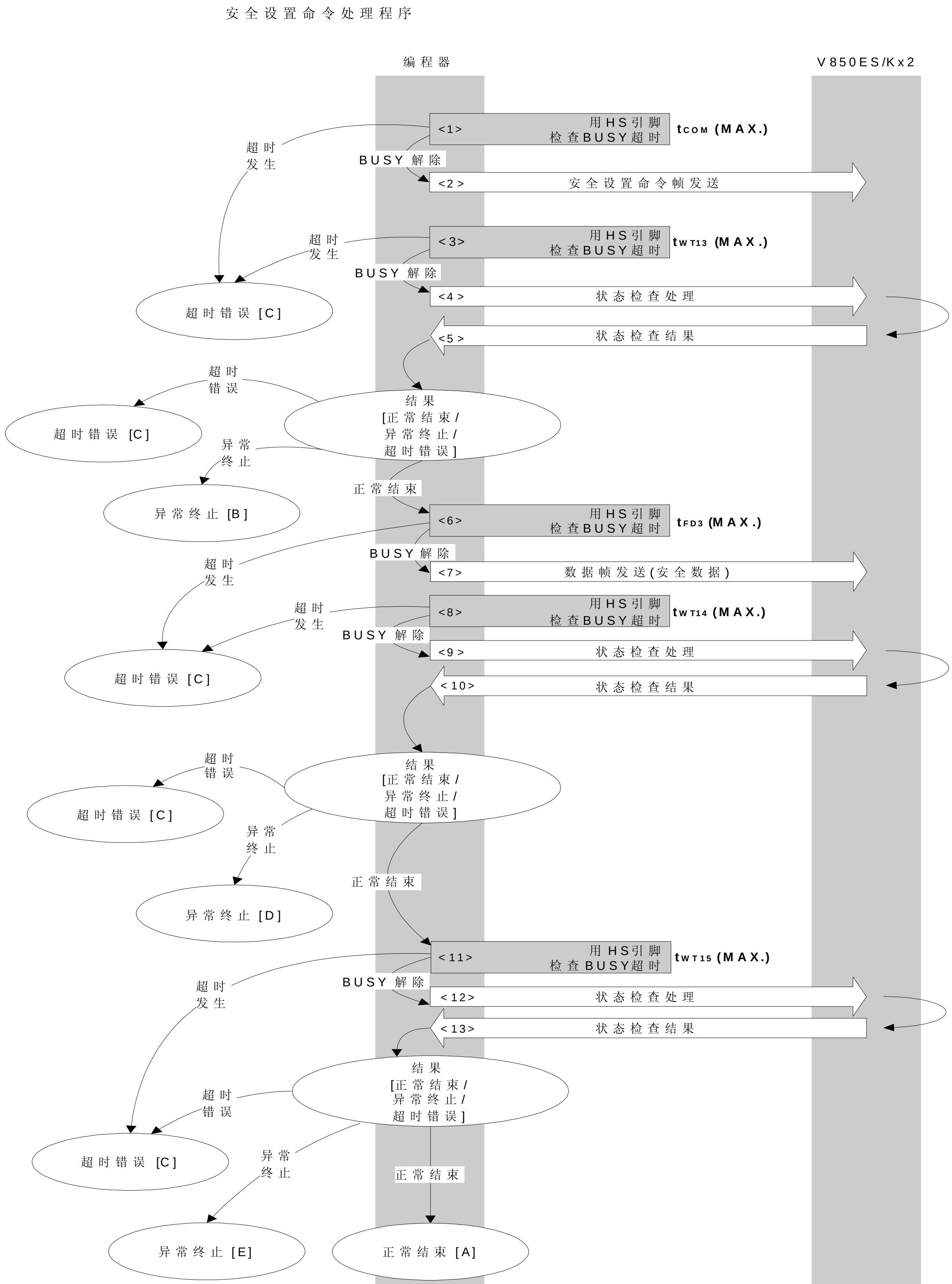
7.14.5 程序举例说明

以下是校验和命令处理的程序举例说明：

```
/*****************************************************************/
/*                                                               */
/*    获取校验和命令(CSI-HS)                                      */
/*                                                               */
/*****************************************************************/
/*   [i] u16 *sum        ... 指向校验和存储区                      */
/*   [i] u32 top         ... 开始地址                              */
/*   [i] u32 bottom      ... 结束地址                              */
/*   [r] u16             ... 错误码                                */
/*****************************************************************/
u16         fl_hs_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16     rc;
    u16     block_num;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom);    // 设置SAH/SAM/SAL, EAH/EAM/EAL
    block_num = get_block_num(top, bottom);    // 获得块编码

    if (hs_busy_to(tCOM_MAX))
            return  FLC_HSTO_ERR;                          // 检测到超时 :情况 [C]

    if (rc = put_cmd_hs(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm)) //发送 "校验和" 命令
            return  rc;                             // 检测到错误 :情况 [C]

    if (hs_busy_to(tWT16_MAX))
            return  FLC_HSTO_ERR;                          // 检测到超时 :情况 [C]

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

    if (hs_busy_to(tFD1_MAX * block_num))
            return  FLC_HSTO_ERR;                          // 检测到超时:情况 [C]

    rc = get_dfrm_hs(fl_rxdata_frm);           // 获得标记数据

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

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

7.15 安全设置命令

7.15.1 处理程序流程图



7.15.2 处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。
<2> 命令帧发送处理发送安全设置命令。
<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT13}(MAX.))。
<4> 状态检查处理获得状态帧。
<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{FD3}(MAX.))。
<7> 数据帧发送处理发送数据帧(安全设置数据) 。
<8> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT14}(MAX.))。
<9> 状态检查处理获得状态帧。
<10> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <11>
当处理结束异常时：	异常终止 [D]
当发生超时错误时：	超时错误[C]被返回

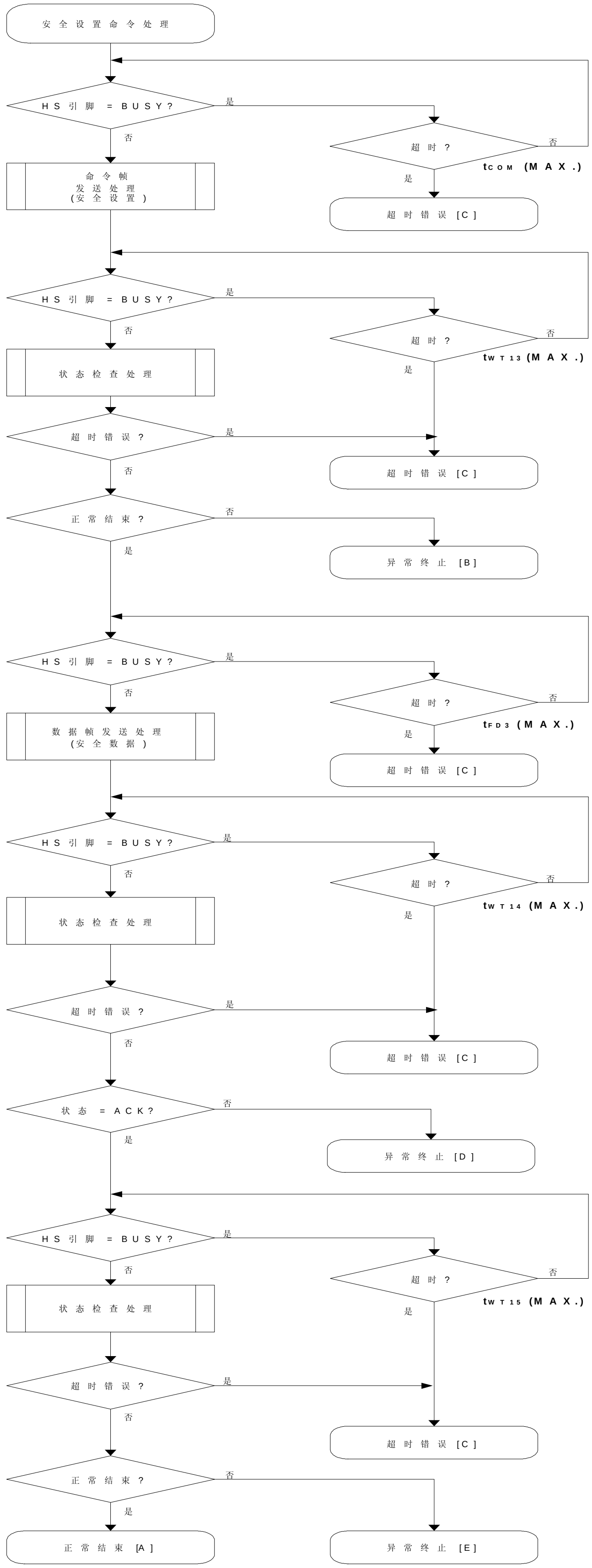
<11> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT15}(MAX.))。
<12> 状态检查处理获得状态帧。
<13> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	正常完成 [A]
当处理结束异常时：	异常终止 [E]
当发生超时错误时：	超时错误[C]被返回

7.15.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，安全设置被正常完成。
异常终止 [B]	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	• 试图使能已被设置禁止的标志。 • 在安全设置中重写引导块被禁止。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。
异常终止 [D]	FLMD 错误	18H	发生写错误。
	MGR10 错误	1AH	
	写错误	1CH	
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

7.15.4 流程图


安 全 设 置 命 令 处 理
HS 引 脚 = BUSY?
是
否
超 时 ?
t_{COM} (MAX.)
命 令 帧
发 送 处 理
(安 全 设 置)
超 时 错 误 [C]
t_{WT13} (MAX.)
状 态 检 查 处 理
超 时 错 误 ?
正 常 结 束 ?
异 常 终 止 [B]
t_{FD3} (MAX.)
数 据 帧 发 送 处 理
(安 全 数 据)
t_{WT14} (MAX.)
状 态 = ACK?
异 常 终 止 [D]
t_{WT15} (MAX.)
正 常 结 束 [A]
异 常 终 止 [E]

7.15.5 程序举例说明

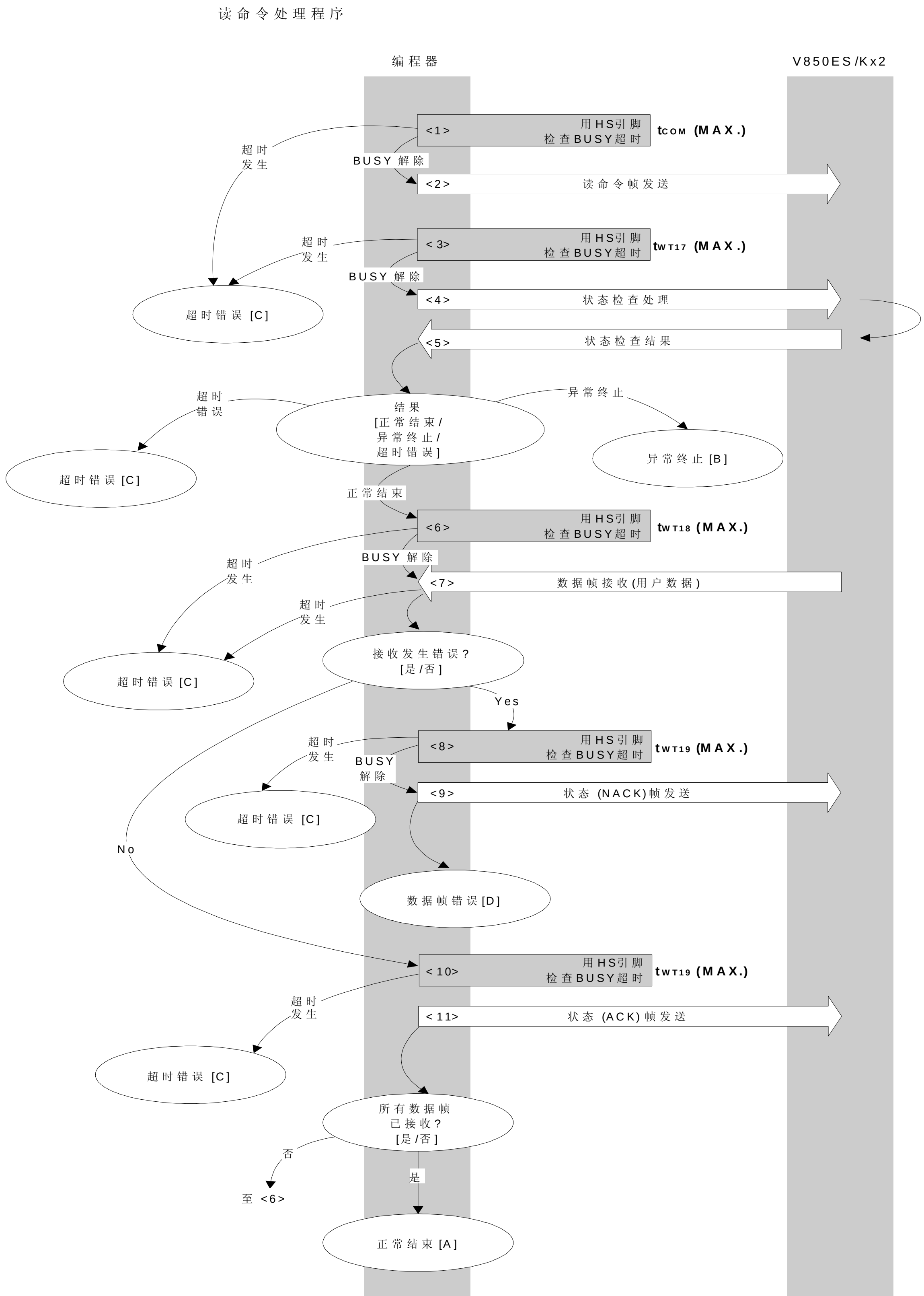
以下是安全设置命令处理的程序举例说明：

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

  /**********************************************************/
  /*            设置参数                                     */
  /**********************************************************/
  fl_cmd_prm[0] = 0x00;                   // 第1个字节必须是0x00
  fl_cmd_prm[1] = 0x00;                   // 第2个字节必须是0x00

  fl_txdata_frm[0] = (scf|= 0b11100000);
                                     // "FLG" (高3位必须是'1' (务必确保))
  fl_txdata_frm[1] = bot;                 // "BOT"

  /**************************************************/
  /*            发送命令                             */
  /**************************************************/
  if (hs_busy_to(tCOM_MAX))
        return    FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]

  if (rc = put_cmd_hs(FL_COM_SET_SECURITY, 3, fl_cmd_prm))
                                                  // 发送 "安全设置" 命令
        return    rc;                        // 检测到错误:情况 [C]

  if (hs_busy_to(tWT13_MAX))
        return    FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]

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

  /**************************************************/
  /*      发送数据帧(安全设置数据)                    */
  /**************************************************/
  if (hs_busy_to(tFD3_MAX))
        return    FLC_HSTO_ERR;                   // 检测到超时 :情况 [C]

  if (rc = put_dfrm_hs(2, fl_txdata_frm, true))   // 发送安全设置数据
        return    rc;                        // 检测到错误 :情况 [C]
```

```
if (hs_busy_to(tWT14_MAX))
        return   FLC_HSTO_ERR;                        // 检测到超时 :情况 [C]

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

  /***********************************************/
  /*         检查内部验证                              */
  /***********************************************/
  if (hs_busy_to(tWT15_MAX))
        return   FLC_HSTO_ERR;                        // 检测到超时

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

7.16 读取命令

7.16.1 处理程序流程图



7.16.2 处理程序的描述说明

<1> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{COM}(MAX.))。

<2> 命令帧发送处理发送读取命令。

<3> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT17}(MAX.))。

<4> 状态检查处理获得状态帧。

<5> 根据状态检查处理的结果执行以下处理。

当处理结束正常时：	进入 <6>
当处理结束异常时：	异常终止 [B]
当发生超时错误时：	超时错误[C]被返回

<6> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT18}(MAX.))。

<7> 数据帧接收处理接收 Flash 存储器中的数据帧(用户数据)。

当处理正常结束时：	进入 <10>
当发生校验和之类的错误时：	进入 <8>
当发生超时错误时：	超时错误[C]被返回

<8> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT19}(MAX.))。

<9> 数据帧发送处理发送 NACK 帧。
数据帧错误 [D] 被返回。

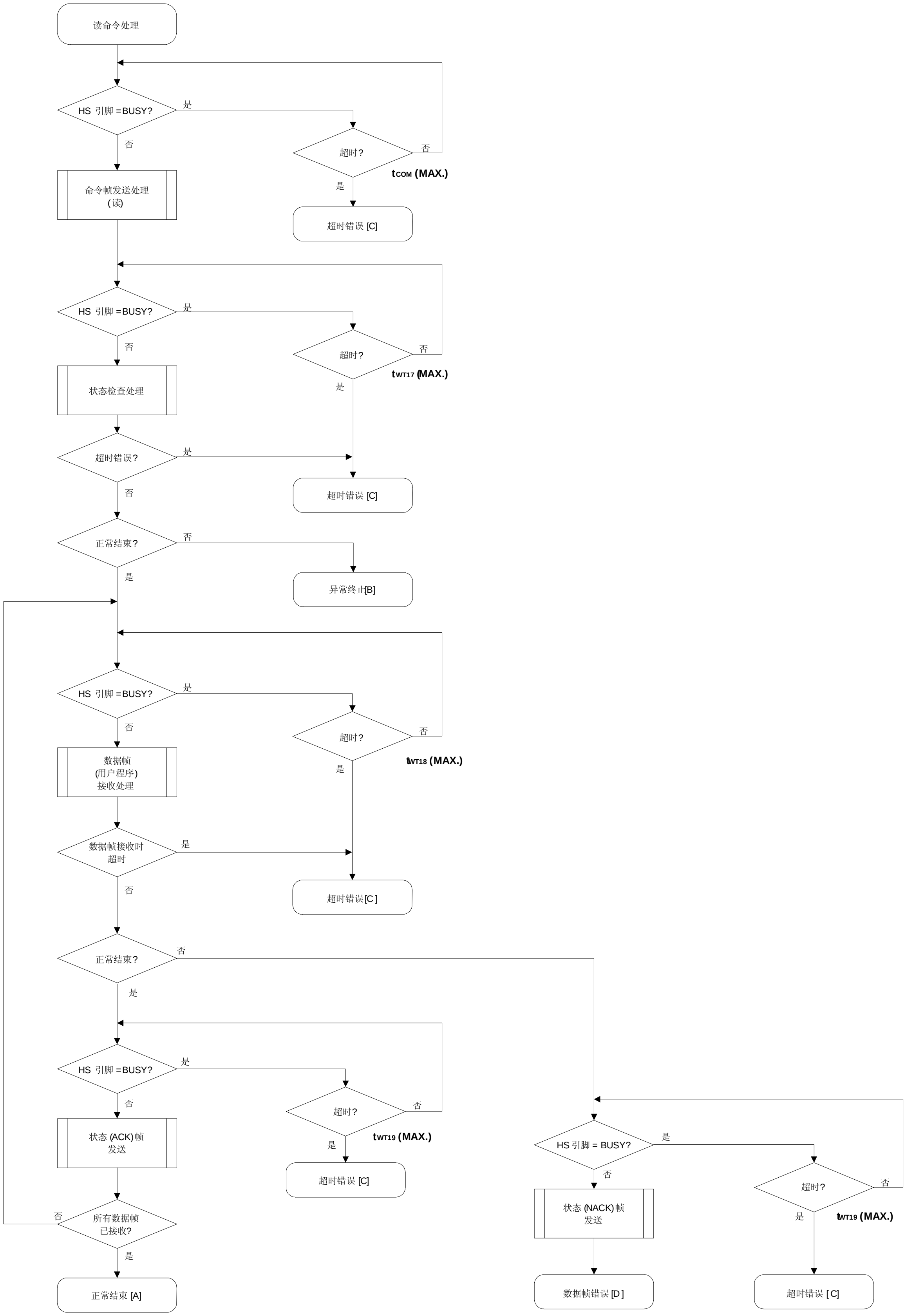
<10> 用 HS 引脚进行 V850ES/Kx2 BUSY 状态的检查。
如果发生 BUSY 超时，超时错误[C]被返回 (超时时间 t_{WT19}(MAX.))。

<11> 数据帧发送处理发送 ACK 帧。
当所有的数据帧接收完成时，正常完成状态[A] 被返回。
如果仍有剩余的数据帧需要接收，程序从<6>重复执行。

7.16.3 处理完毕后的状态

处理完毕后的状态		状态码	描述
正常完成 [A]	确认正常 (ACK)	06H	命令正常执行，读取数据被正常设置。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是该块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	在安全设置中读取被禁止。
	确认异常 (NACK)	15H	命令帧数据异常 (例如不正确的数据长度 (LEN) 或没有 ETX)。
超时错误[C]		–	由于 HS 引脚的 busy 状态使处理超时。
数据帧错误 [D]		–	所接收的读取数据数据帧校验和不匹配。

7.16.4 流程图


读命令处理
HS 引脚 =BUSY?
是
否
超时?
否
t_{COM} (MAX.)
是
命令帧发送处理
(读)
超时错误 [C]
HS 引脚 =BUSY?
是
否
超时?
否
t_{WT17} (MAX.)
是
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止[B]
HS 引脚 =BUSY?
是
否
超时?
否
t_{WT18} (MAX.)
是
数据帧
(用户程序)
接收处理
数据帧接收时
超时
是
否
超时错误[C]
正常结束?
否
是
HS 引脚 =BUSY?
是
否
超时?
否
t_{WT19} (MAX.)
是
状态 (ACK) 帧
发送
超时错误 [C]
所有数据帧
已接收?
否
是
正常结束 [A]
HS 引脚 = BUSY?
是
否
超时?
否
是
t_{WT19} (MAX.)
状态 (NACK) 帧
发送
数据帧错误 [D]
超时错误 [C]

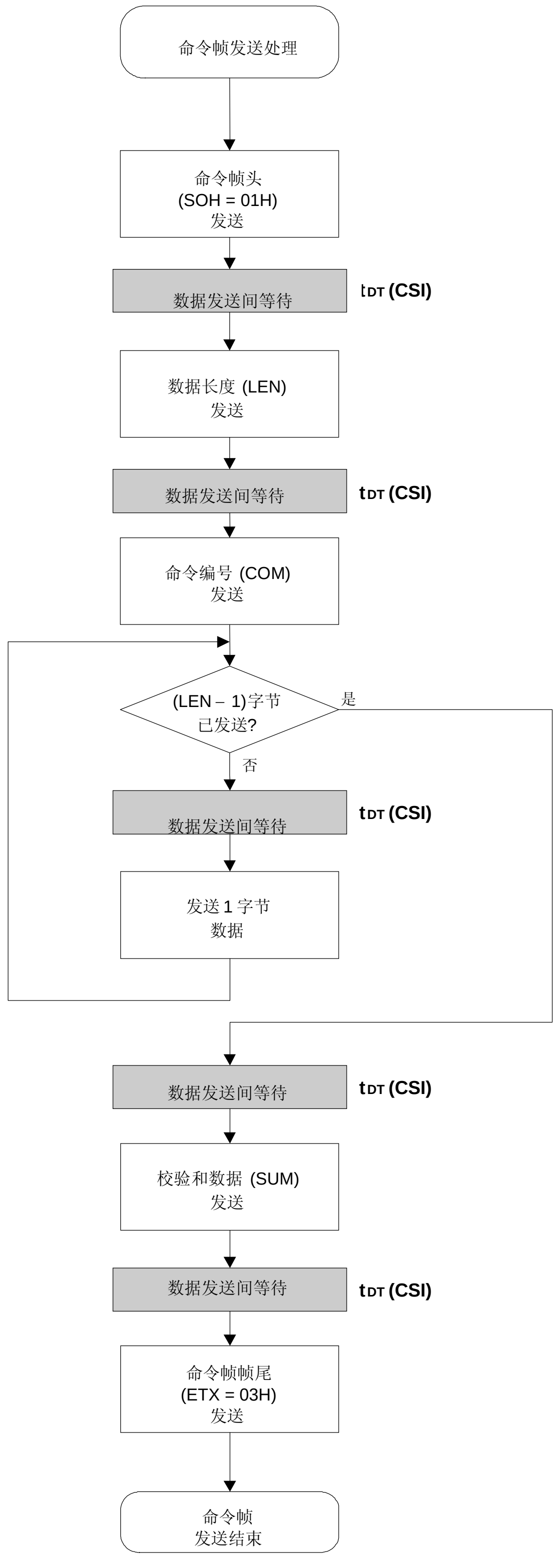
7.16.5 程序举例说明

以下时读取命令处理的程序举例说明：

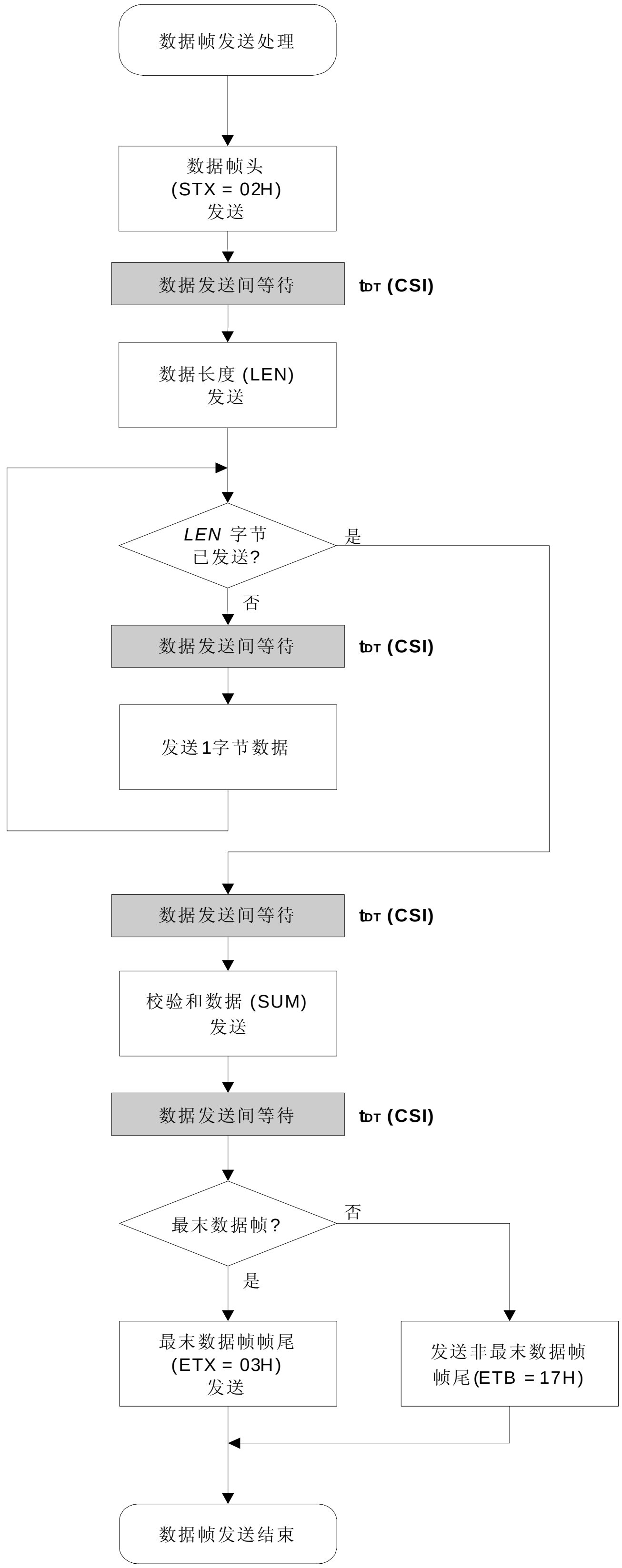
```
/*****************************************************************/
/*                                                               */
/*  读取命令                                                     */
/*                                                               */
/*****************************************************************/
u16         fl_hs_read(u32 top, u32 bottom)
{
  u16       rc;
  u32       read_head;
  u16       len;
  u8        hooter;

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


  /**********************************************/
  /*        发送命令并检查状态              */
  /**********************************************/
  if (hs_busy_to(tCOM_MAX))
            return    FLC_HSTO_ERR;                        // 检测到超时 :情况 [C]

  if (rc = put_cmd_hs(FL_COM_READ, 7, fl_cmd_prm))
            return    rc;

  if (hs_busy_to(tWT17_MAX))
            return    FLC_HSTO_ERR;                        // 检测到超时 :情况 [C]

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

  /**********************************************/
  /*        接收用户数据               */
  /**********************************************/
  read_head = top;

  while(1){

            if (hs_busy_to(tWT18_MAX))
                      return    FLC_HSTO_ERR;                        // 检测到超时 :情况 [C]

            rc = get_dfrm_hs(fl_rxdata_frm);           //从FLASH获取ROM数据
            switch(rc) {
                      case      FLC_NO_ERR:                          break;    // 继续
                      case      FLC_HSTO_ERR:              return    rc;       break;    // 情况 [C]
```

```
//          case    FLC_RX_DFSUM_ERR:
                    default:                        // 情况 [D]

                            if (hs_busy_to(tWT19_MAX))
                                    return  FLC_HSTO_ERR;           // 检测到超时

                                put_sfrm_hs(FLST_NACK);         // 发送状态 (NACK) 帧
                            return  rc;
                            break;

            }
            if (hs_busy_to(tWT19_MAX))
                    return  FLC_HSTO_ERR;                   // 检测到超时

            put_sfrm_hs(FLST_ACK);                  // 发送状态(ACK) 帧

            /**************************************************/
            /*              存储 ROM 数据                    */
            /**************************************************/
            if ((len = fl_rxdata_frm[OFS_LEN]) == 0)        // 获得长度
                    len = 256;

            memcpy(read_buf+read_head, fl_rxdata_frm+2, len); // 存储到外部 RAM

            read_head += len;

            /**************************************************/
            /*              结束检查                                         */
            /**************************************************/
            hooter = fl_rxdata_frm[len + 3];
            if (hooter == FL_ETB)                   // 帧结束?
                    continue;                               // 否
            break;                                          // 是
    }

    return  FLC_NO_ERR;
}
```

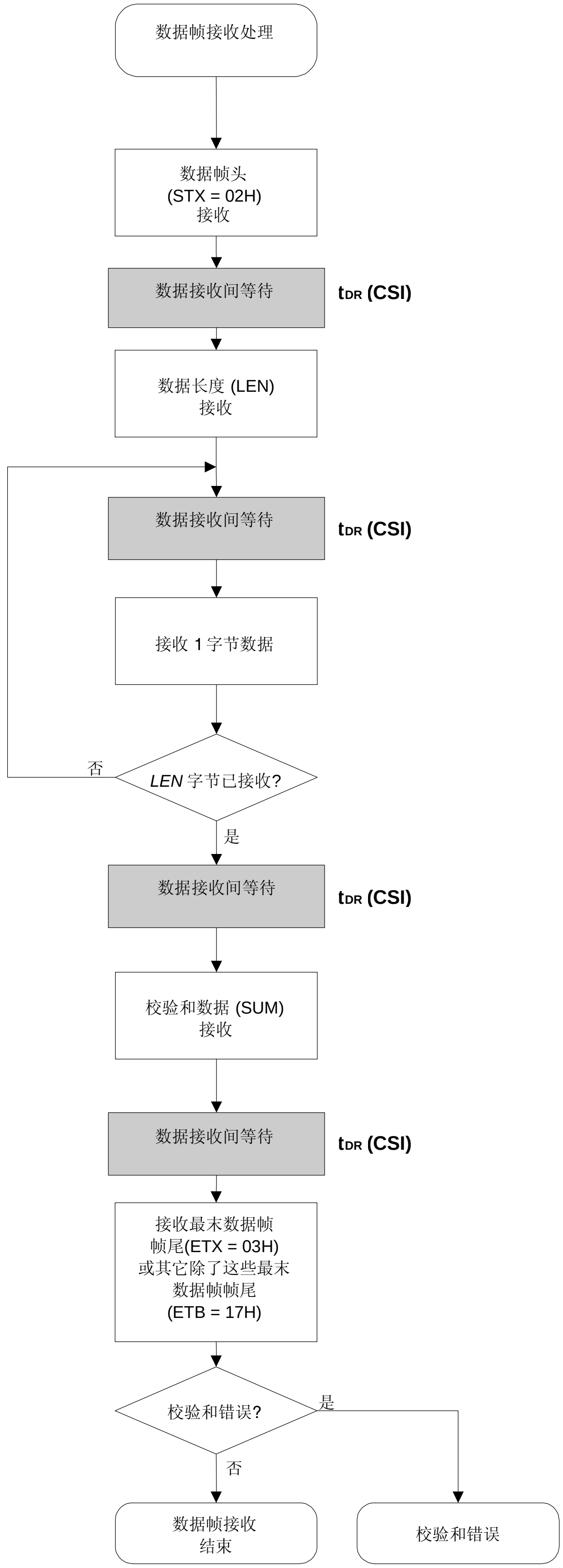
8.1　命令帧发送处理流程图



8.2 数据帧发送处理流程图



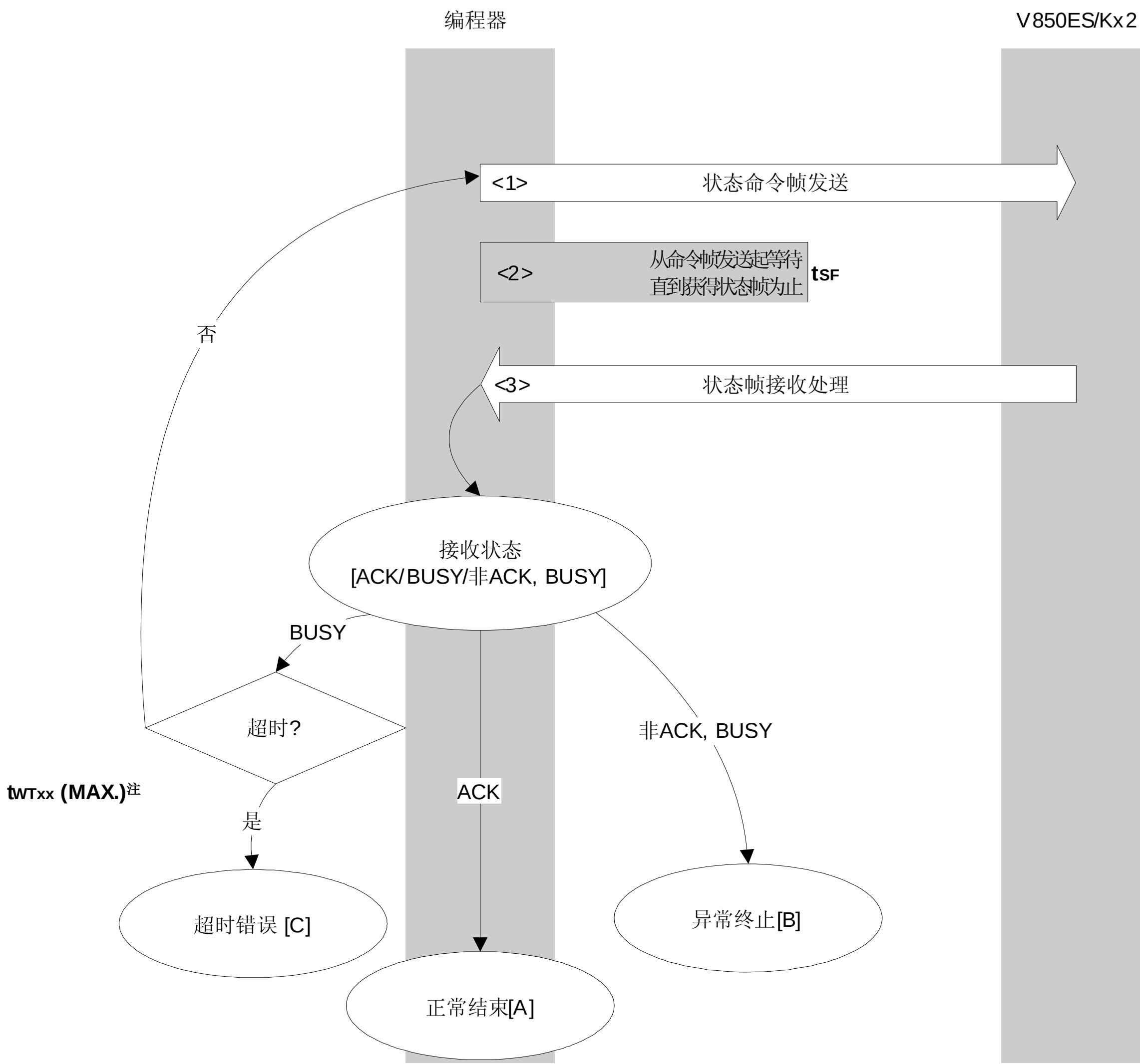
8.3　数据帧接收处理流程图



8.4　状态命令

8.4.1　处理程序流程图

状态命令处理程序



注　　依据执行的命令，使用不同参数。

8.4.2　处理程序的描述说明

<1>　由命令帧发送处理发送状态命令。
<2>　从命令发送起等待直到状态帧接收为止（等待时间 t_{SF}）。
<3>　状态码检验。

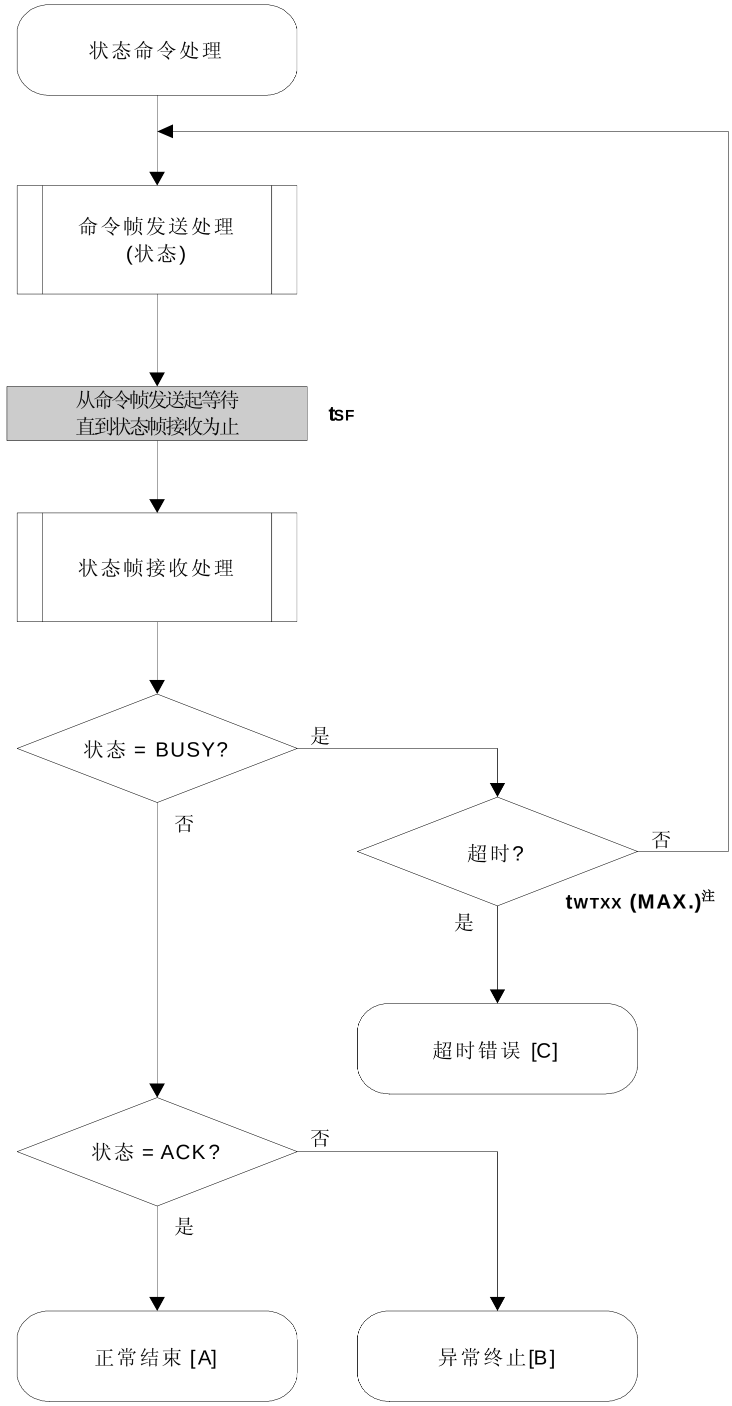
当 ST1 = ACK 时：　　正常完成 [A]
当 ST1 = BUSY 时：　　执行超时检验($t_{WT\times\times}$**(MAX.)**[注])。
　　　　　　　　　　如果处理没有超时，程序从 <1>重新执行。
　　　　　　　　　　如果发生超时，返回一个超时错误 [C]。
当 ST1 ≠ ACK 时，BUSY：异常终止 [B]

注　依据执行的命令，使用不同参数。

8.4.3　处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	从 V850ES/Kx2 发送的状态帧已经被正常接收。
异常终止 [B]	命令错误	04H	不支持的命令或异常帧已经被接收。
	参数错误	05H	命令信息（参数）无效。
	校验和错误	07H	从编程器发送的帧数据异常。
	校验错误	0FH	从编程器发送的帧数据发生校验错误。
	保护错误	10H	试图执行由安全设置命令禁止的处理。
	确认异常 (NACK)	15H	确认异常
	FLMD 错误	18H	发生写入错误。
	MRG10 错误	1AH	发生擦除错误。
	MRG11 错误	1BH	在数据写入期间发生内部校验错误，或发生空白检查错误。
	写错误	1CH	发生写入错误。
超时错误 [C]		–	在命令发送之后，经过了特定时间之后，仍会返回 BUSY 信号。

8.4.4　流程图



注　　依据执行的命令，使用不同参数。

8.4.5　程序举例说明

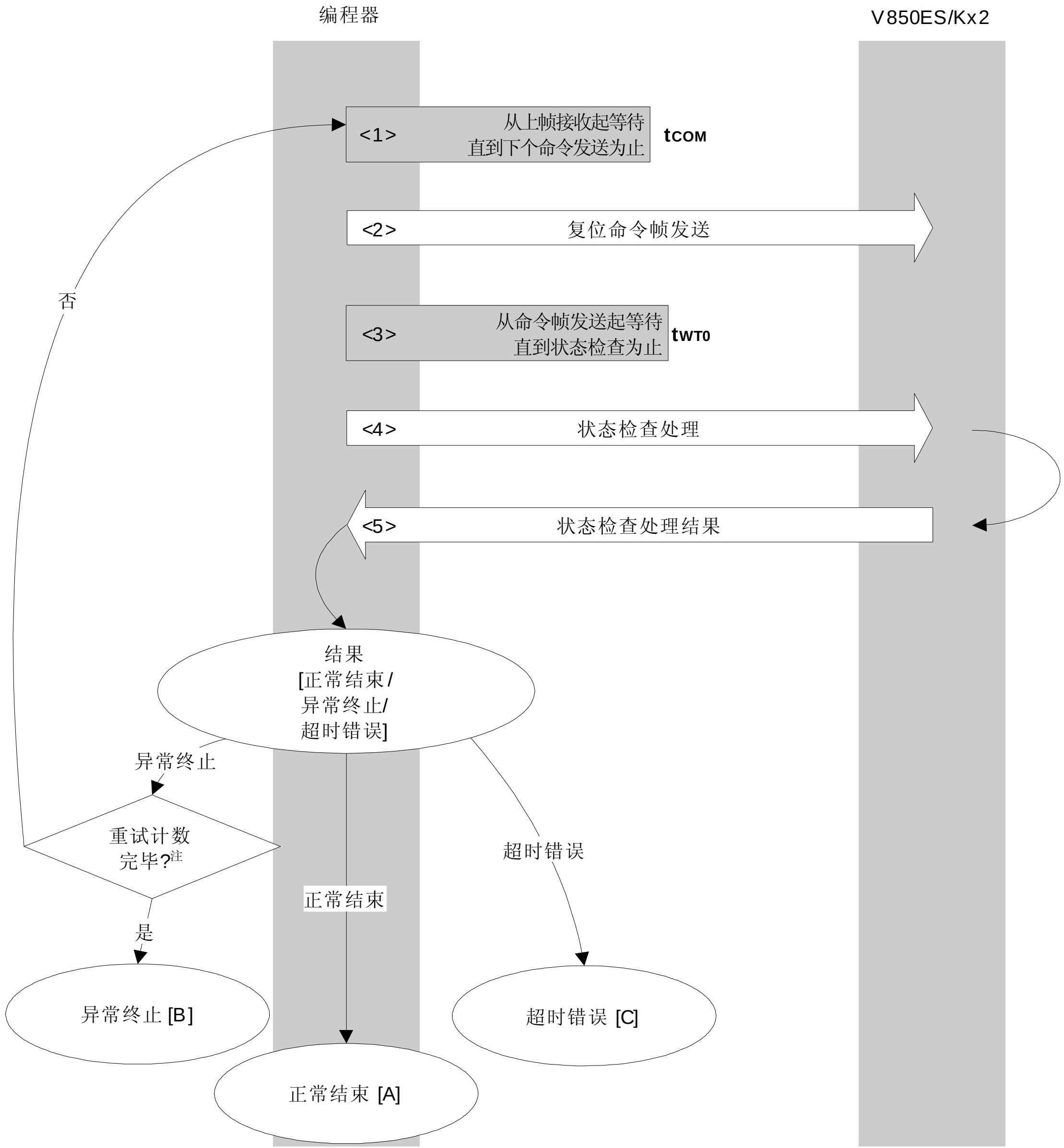
下面是状态命令处理的程序举例。

```
/**************************************************************************************************/
/*                                                                                                */
/*   获取状态命令 (CSI)                                                                             */
/*                                                                                                */
/**************************************************************************************************/
/*   [r] u16              ... 状态解码或错误编码                                                      */
/*                                                                                                */
/*   (见 fl.h/fl-proto.h 并                                                                        */
/*             在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“状态”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 // 情况 [C]
                          continue；                      // 否，重试

                  default:                                // 校验和错误
                          return   rc；

                  case    FLC_NO_ERR:                     // 无错误
                          break；

          }
          if (fl_st1 == FLST_BUSY){        // ST1 = BUSY
                  if (check_flto())                       // 超时?
                          return   FLC_DFTO_ERR；         // 是，超时 // 情况 [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 // 情况 [C]
                  continue；
          }
          break；                                   // ACK或其它错误(除BUSY之外)
  }

  rc = decode_status(fl_st1)；                      //状态解码返回编码
//switch(rc) {
//
// case     FLC_NO_ERR:   return   rc；    break；  // 情况 [A]
// default:               return   rc；    break；  // 情况 [B]
// }
  return rc；
}
```

8.5　复位命令

8.5.1　处理程序流程图

复位命令处理程序



注　　复位命令的发送次数不能超过重试次数（最多 16 次）。

8.5.2 处理程序的描述说明

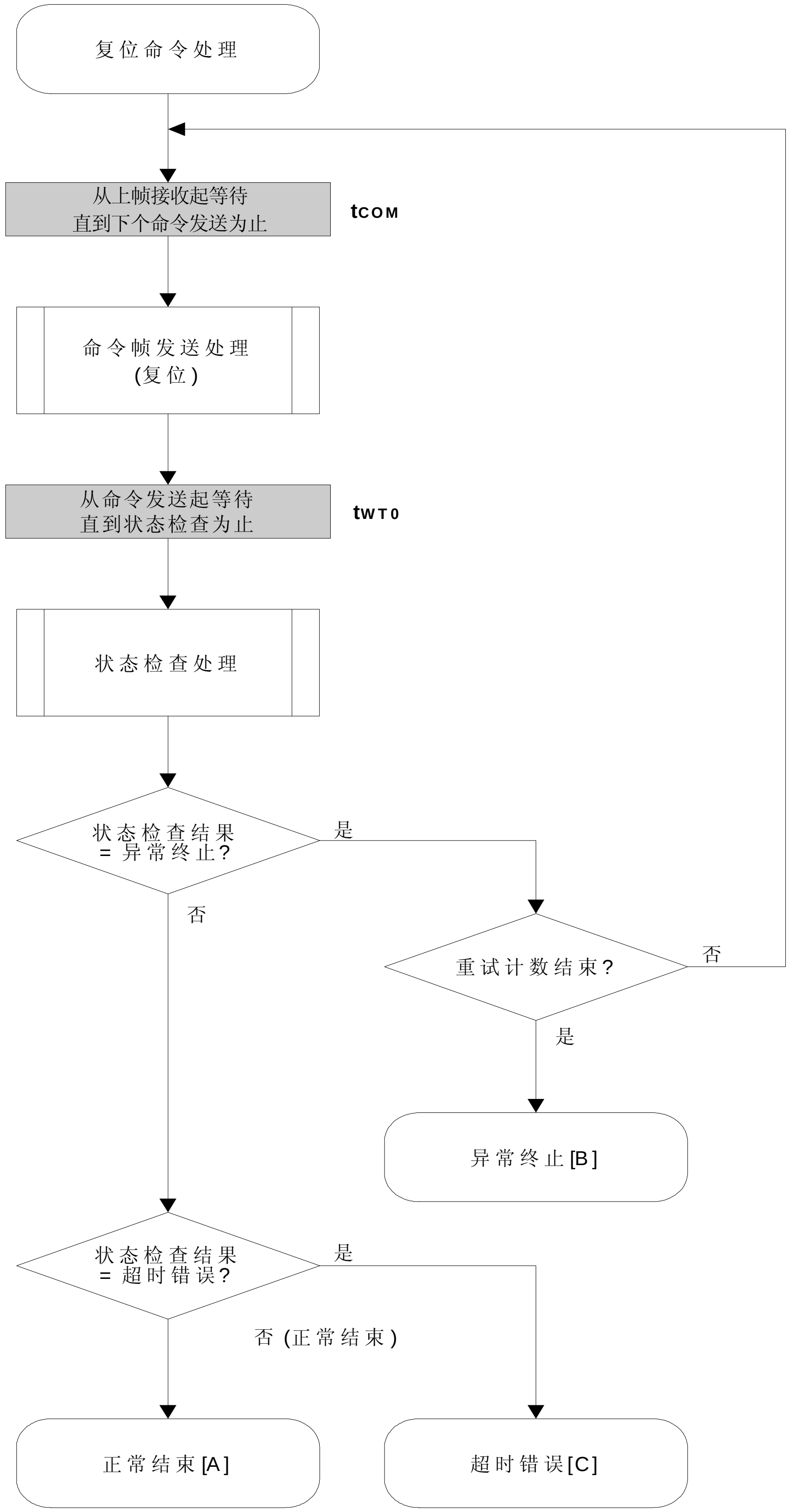
<1> 从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2> 由命令帧发送处理发送复位命令。
<3> 从命令发送起等待直到状态检查处理为止(等待时间 t_{WT0}(MAX.))。
<4> 由状态检查处理获得状态帧。
<5> 依据状态检查处理结果，执行下面的处理。

当处理正常结束时:	正常完成 [A]
当处理异常结束时:	如果没有超过重试次数，程序从<1>重新执行。
	如果超过了重试次数，处理异常结束 [B]。
当发生超时错误时:	返回超时错误 [C] 。

8.5.3 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行，而且在编程器和 V850ES/Kx2 之间已经建立同步。
异常终止 [B]	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	• 在处理期间，接收了除了状态命令之外的命令。 • 命令帧数据异常 (例如无效数据长度 (LEN)或无 ETX)。
超时错误 [C]		–	状态校验处理超时。

8.5.4　流程图


复位命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(复位)
从命令发送起等待
直到状态检查为止
tWT0
状态检查处理
状态检查结果
= 异常终止?
是
否
重试计数结束?
否
是
异常终止[B]
状态检查结果
= 超时错误?
是
否 (正常结束)
正常结束[A]
超时错误[C]

8.5.5　程序举例说明

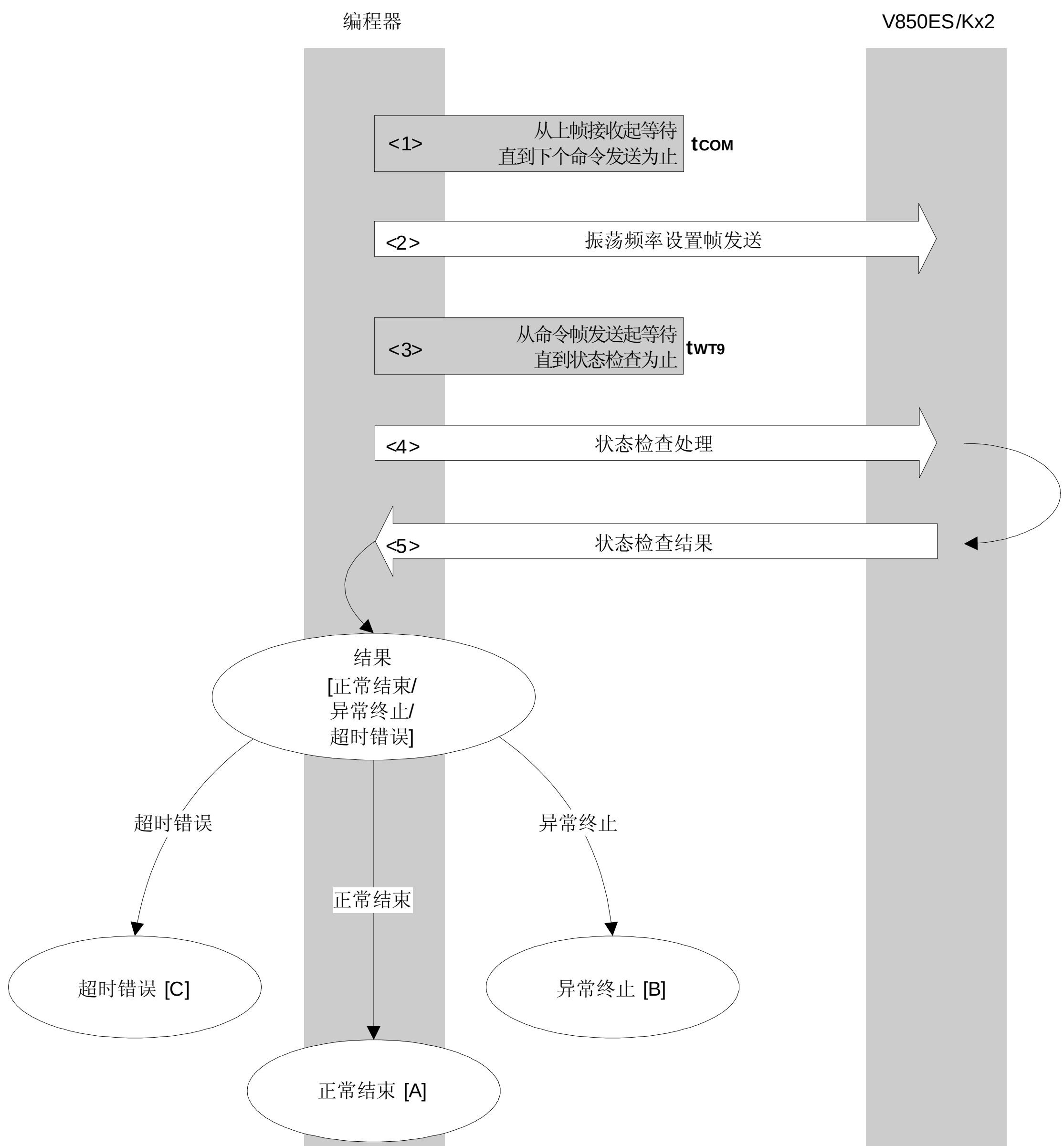
下面是复位命令处理的程序举例。

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

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

                fl_wait(tCOM)；                                     // 在发送命令帧之前等待

                put_cmd_csi(FL_COM_RESET， 1， fl_cmd_prm)；  //发送"复位"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      // 情况 [C]
            if (rc == FLC_ACK)                           // Ack ?
                     break；                             // 是      // 情况 [A]
            //continue；                                            // 情况 [B] (如果退出循环)
  }
//switch(rc) {
//
// case     FLC_NO_ERR:          return    rc；      break；  // 情况 [A]
// case     FLC_DFTO_ERR:        return    rc；      break；  // 情况 [C]
// default:                      return    rc；      break；  // 情况 [B]
//}
  return    rc；
}
```

8.6　振荡频率设置命令

8.6.1　处理程序流程图

振荡频率设置命令处理程序



8.6.2　处理程序的描述说明

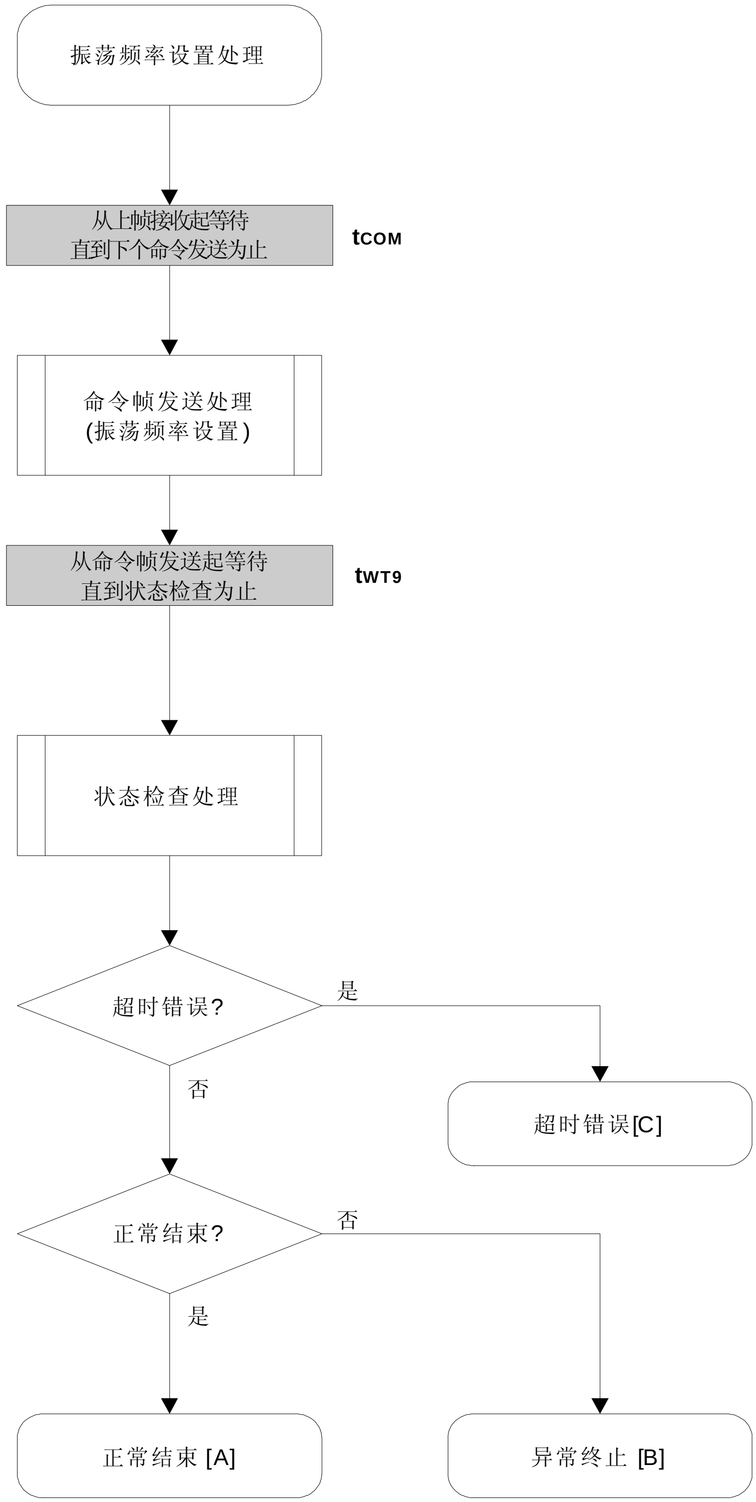
<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送振荡频率设置命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT9}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

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]
当发生超时错误时：	返回超时错误[C]。

8.6.3　处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且操作频率被正确的设置到 V850ES/Kx2。
异常终止 [B]	参数错误	05H	振荡频率的值超出范围。
	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。

8.6.4　流程图


振荡频率设置处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(振荡频率设置)
从命令帧发送起等待
直到状态检查为止
tWT9
状态检查处理
超时错误?
是
超时错误[C]
否
正常结束?
否
是
正常结束 [A]
异常终止 [B]

8.6.5　程序举例说明

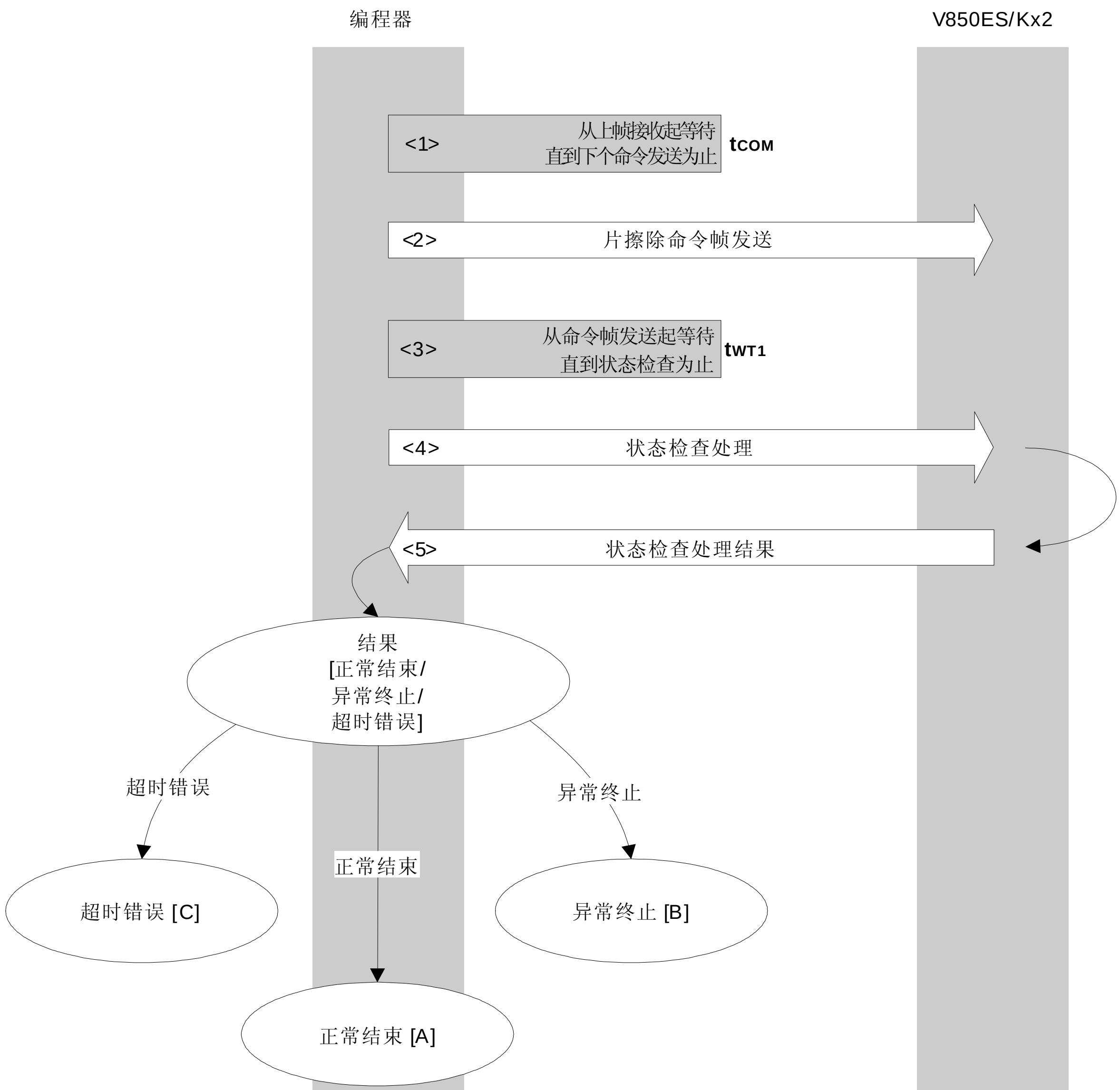
下面是振荡频率设置命令处理的程序举例。

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

     put_cmd_csi(FL_COM_SET_OSC_FREQ， 5， fl_cmd_prm)； // 发送"振荡频率设置" 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；  // 情况 [A]
// case      FLC_DFTO_ERR:           return    rc；      break；  // 情况 [C]
// default:                          return    rc；      break；  // 情况 [B]
//}
   return rc；
}
```

8.7　片擦除命令

8.7.1　处理程序流程图

片擦除命令处理程序



8.7.2　处理程序的描述说明

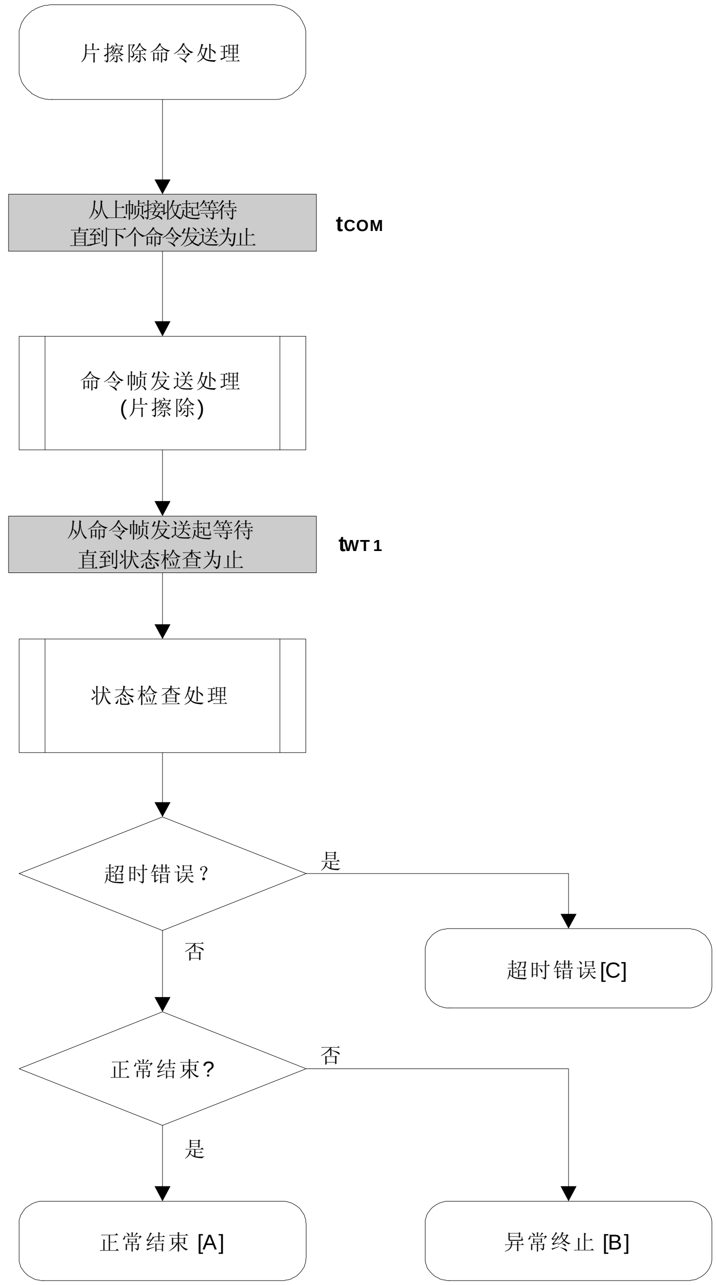
<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送片擦除命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT1}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理的结果，执行下面的处理。

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]
当发生超时错误时：	返回超时错误[C]。

8.7.3　处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且片擦除正常执行。
异常终止 [B]	校验和错误	07H	发送的命令帧校验和不匹配。
	保护错误	10H	• 在安全设置中片擦除被禁止。 • 在安全设置中引导块重写被禁止。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
	擦除错误	1AH	发生擦除错误。
	写错误	1CH	
	MRG10 错误	1AH	
	MRG11 错误	1BH	
超时错误 [C]		–	在特定的时间内没有接收到状态帧。

8.7.4　流程图


片擦除命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(片擦除)
从命令帧发送起等待
直到状态检查为止
tWT1
状态检查处理
超时错误？
是
超时错误[C]
否
正常结束?
否
是
正常结束 [A]
异常终止 [B]

8.7.5　程序举例说明

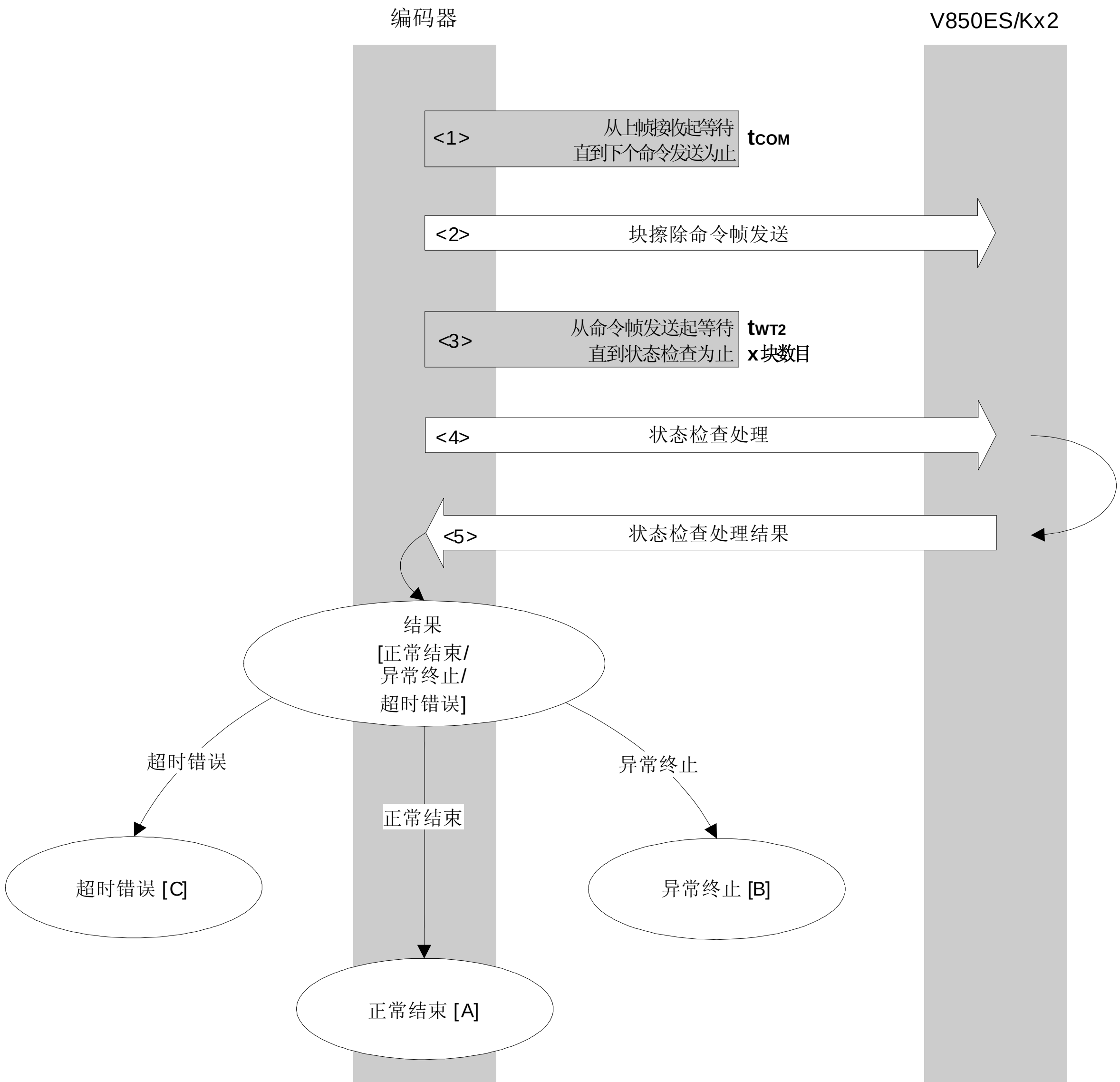
下面是片擦除命令处理的程序举例。

```
/**********************************************************************************************/
/*                                                                                            */
/*   擦除所有(片)命令(CSI)                                                                     */
/*                                                                                            */
/**********************************************************************************************/
/* [r] u16          ... 错误编码                                                               */
/**********************************************************************************************/
u16       fl_csi_erase_all(void)
{
  u16     rc；


  fl_wait(tCOM)；                                           // 在发送命令帧之前等待

  put_cmd_csi(FL_COM_ERASE_CHIP， 1， fl_cmd_prm)； // 发送"片擦除"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；  // 情况 [A]
// case     FLC_DFTO_ERR:        return   rc；     break；  // 情况 [C]
// default:                      return   rc；     break；  // 情况 [B]
//}
  return rc；

}
```

8.8　块擦除命令

8.8.1　处理程序流程图

块擦除命令处理顺序



8.8.2　处理程序的描述说明

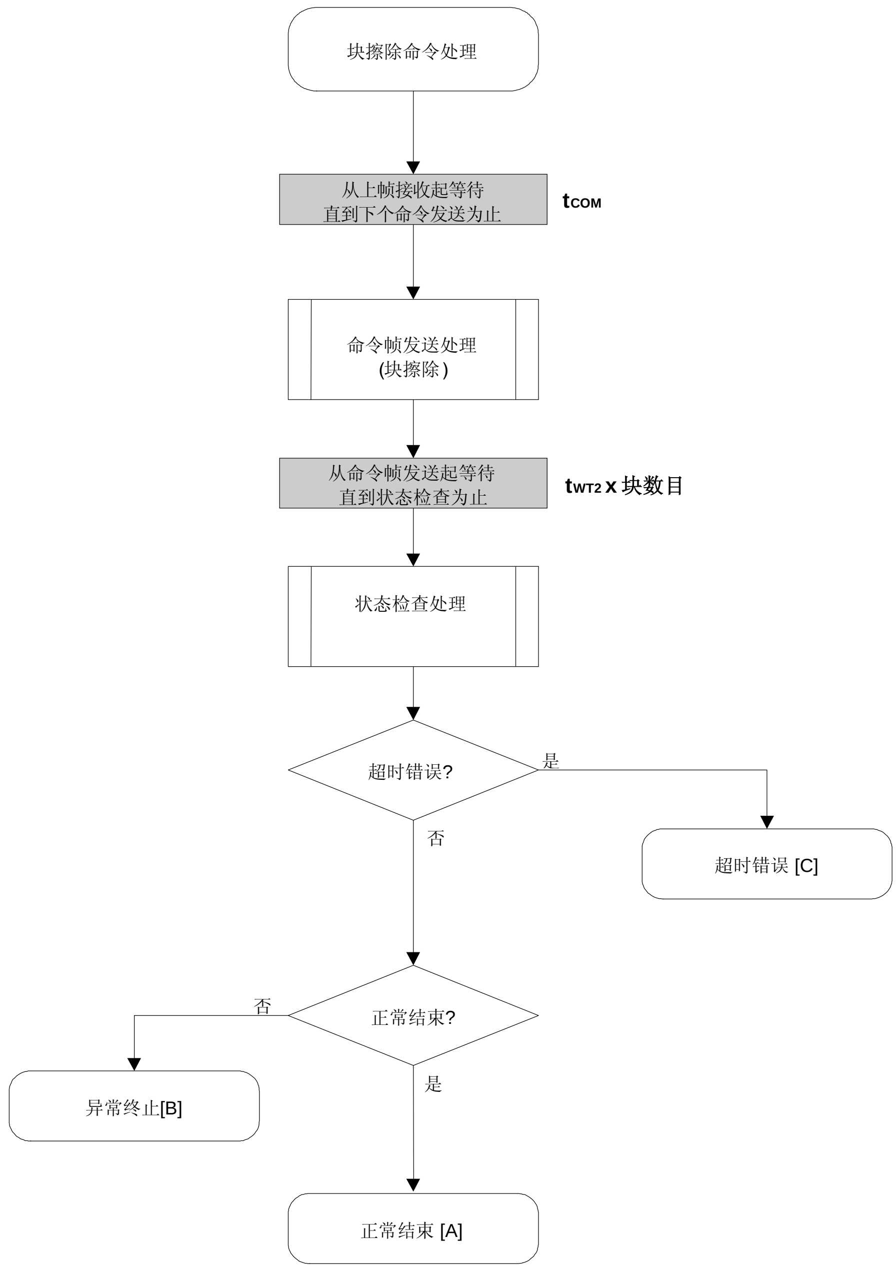
<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送块擦除命令。
<3>　等待直到获得状态帧为止(等待时间 t_{WT2}(MAX.) × 块数目)。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

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]
当发生超时错误时：	返回超时错误[C]。

8.8.3　处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且块擦除正常执行。
异常终止 [B]	参数错误	05H	• 在 flash 存储器范围之外指定开始/结束地址。 • 指定的开始/结束地址不是块的开始/结束地址。
	校验和错误	07H	发送的命令帧校验和不匹配。
	保护错误	10H	• 在安全设置中块擦除被禁止。 • 引导区域处于其指定范围内，而此时在安全设置中引导块重写被禁止。 • 在安全设置中片擦除被禁止。 • 在安全设置中编程被禁止。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
	MRG10 错误	1AH	发生擦除错误。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。

8.8.4　流程图


块擦除命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(块擦除)
从命令帧发送起等待
直到状态检查为止
tWT2 x 块数目
状态检查处理
超时错误?
是
否
超时错误 [C]
正常结束?
否
是
异常终止[B]
正常结束 [A]

8.8.5　程序举例说明

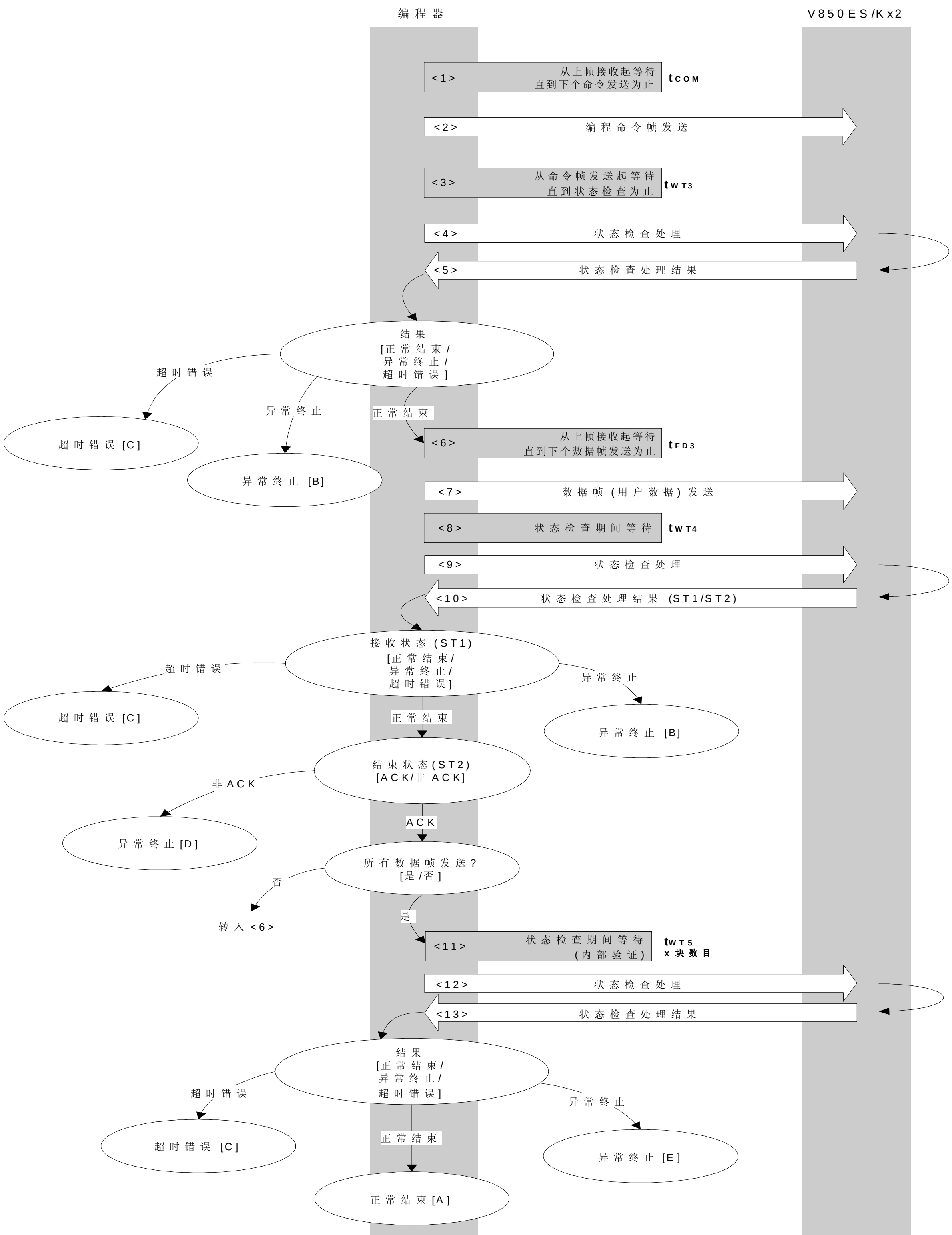
下面是块擦除命令对一个块进行处理的程序举例。

```
/**********************************************************************************************/
/*                                                                                            */
/*    擦除块命令(CSI)                                                                          */
/*                                                                                            */
/**********************************************************************************************/
/*   [i] u16 sblk            ... 开始块擦除   (0...255)                                       */
/*   [i] u16 eblk            ... 结束块擦除   (0...255)                                       */
/*   [r] u16                 ... 错误编码                                                     */
/**********************************************************************************************/
u16       fl_csi_erase_blk(u16 sblk， u16 eblk)
{

  u16       rc；
  u32       top， bottom；
  u16       block_num；

  top = get_top_addr(sblk)；                            // 获取开始块的开始地址
  bottom = get_bottom_addr(eblk)；                      // 获取结束块的结束地址

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

  block_num = eblk - sblk + 1；

  fl_wait(tCOM)；                                           // 在发送命令帧之前等待

  put_cmd_csi(FL_COM_ERASE_BLOCK， 7， fl_cmd_prm)；// 发送"块擦除"命令

  fl_wait(tWT2 * block_num)；

  rc = fl_csi_getstatus(tWT2_MAX * block_num)；        // 获取状态帧
//switch(rc) {
//
// case     FLC_NO_ERR:          return    rc；     break；  // 情况 [A]
// case     FLC_DFTO_ERR:        return    rc；     break；  // 情况 [C]
// default:                      return    rc；     break；  // 情况 [B]
//}
  return rc；
}
```

8.9　编程命令

8.9.1　处理程序流程图

编程命令处理程序



8.9.2　处理程序的描述说明

<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送编程命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT3}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	执行<6>。
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误[C]。

<6>　等待直到下个数据帧发送为止(等待时间 t_{FD3}(MAX.))。
<7>　由数据帧发送处理发送写入 V850ES/Kx2 flash 存储器的用户数据。
<8>　从数据帧（用户数据）发送起等待直到状态检查处理为止(等待时间 t_{WT4}(MAX.))。
<9>　由状态检查处理获得状态帧。
<10> 依据状态检查处理的结果，执行下面的处理 (状态码 (ST1/ST2)) (同样参考处理程序流程图和流程图)。

当 ST1 = 异常终止时：	异常终止 [B]
当 ST1 = 超时错误时：	返回超时错误[C]。
当 ST1 = 正常完成时：	依据 ST2 的值执行下面的处理。

- 当 ST2 ≠ ACK 时：异常终止 [D]
- 当 ST2 = ACK 时:　当所有的用户数据发送完成后执行<11>。
  如果还存在需要发送的用户数据，从<6>重新执行程序。

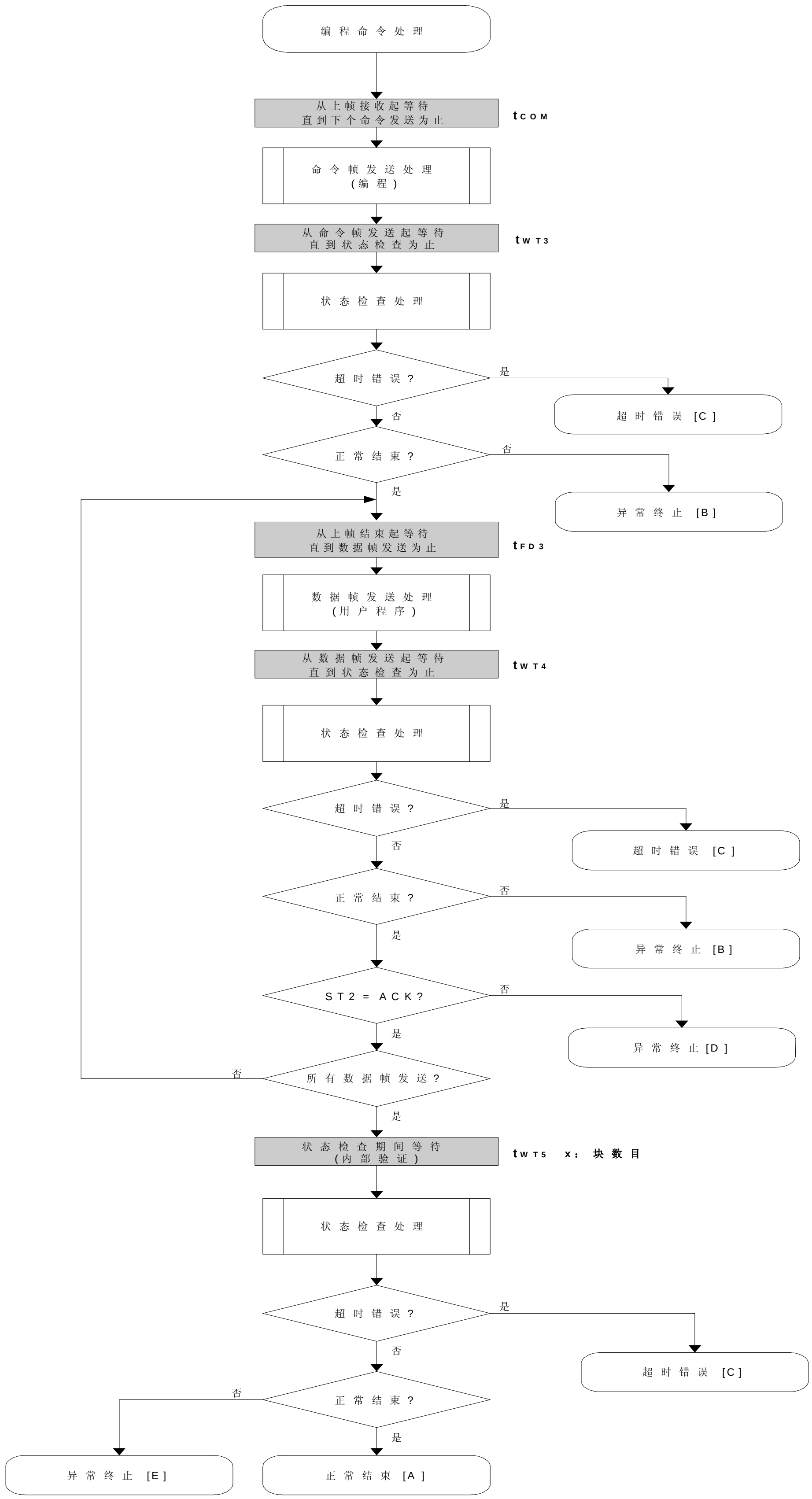
<11> 等待直到状态检查处理为止(超时时间 t_{WT5} (MAX.) × 块的数目)。
<12> 由状态检查处理获得状态帧。
<13> 依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	正常完成 [A] (指在写入完成后，内部校验检查已经正常完成)
当处理异常结束时：	异常终止 [E] (指在写入完成后，内部校验检查没有正常完成)
当发生超时错误时：	返回超时错误[C]。

8.9.3　处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且用户数据写入正常。
异常终止 [B]	参数错误	05H	• 指定的开始/结束地址不是块的开始/结束地址。 • 数据长度不是以 64 字为单位的。
	校验和错误	07H	发送的命令帧校验和不匹配。
	保护错误	10H	• 在安全设置中写入被禁止。 • 在安全设置中引导块重写被禁止。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。
异常终止 [D]	FLMD 错误	18H (ST2)	发生写入错误。
	写入错误	1CH (ST2)	发生写入错误。
异常终止 [E]	MRG11 错误	1BH	发生内部校验错误。

8.9.4　流程图


编 程 命 令 处 理
从上帧接收起等待
直到下个命令发送为止
tCOM
命 令 帧 发 送 处 理
（编 程）
从 命 令 帧 发 送 起 等 待
直 到 状 态 检 查 为 止
tWT3
状 态 检 查 处 理
超 时 错 误？
是
超 时 错 误 [C]
否
正 常 结 束？
否
异 常 终 止 [B]
是
从上帧结束起等待
直到数据帧发送为止
tFD3
数 据 帧 发 送 处 理
（用 户 程 序）
从 数 据 帧 发 送 起 等 待
直 到 状 态 检 查 为 止
tWT4
状 态 检 查 处 理
超 时 错 误？
是
超 时 错 误 [C]
否
正 常 结 束？
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 期 间 等 待
（内 部 验 证）
tWT5 x：块 数 目
状 态 检 查 处 理
超 时 错 误？
是
超 时 错 误 [C]
否
正 常 结 束？
否
异 常 终 止 [E]
是
正 常 结 束 [A]

8.9.5 程序举例说明

下面是编程命令处理的程序举例。

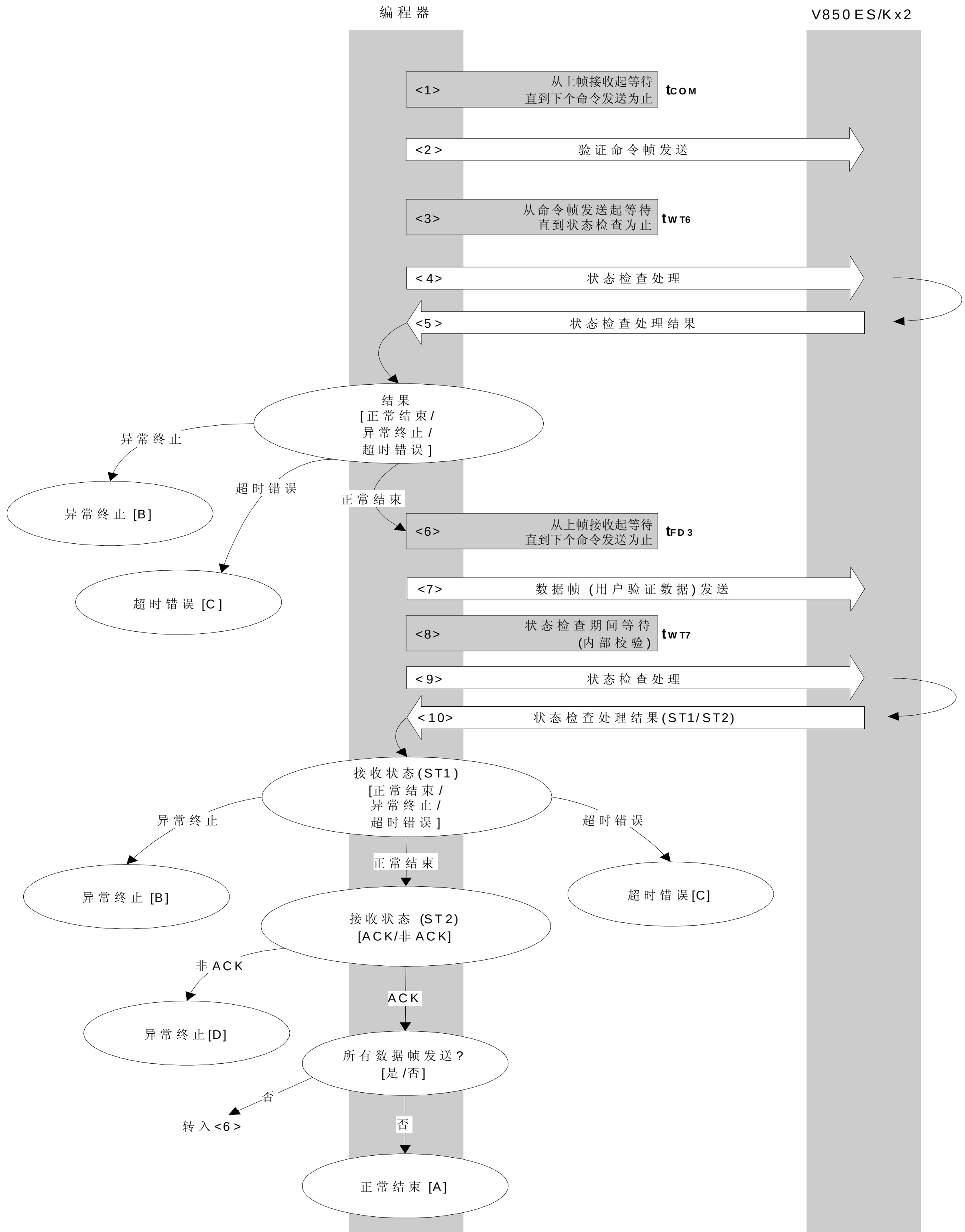
```
/**********************************************************************************************/
/*                                                                                            */
/*    写入命令(CSI)                                                                            */
/*      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

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

   block_num = get_block_num(top， bottom)；                // 获取块编号

   /********************************************************************************/
   /*              发送命令并检查状态                                              */
   /********************************************************************************/

   fl_wait(tCOM)；
   put_cmd_csi(FL_COM_WRITE， 7， fl_cmd_prm)；              // 发送"编程" 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;   // 情况 [C]
            default:                        return    rc;         break;   // 情况 [B]
   }

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

   while(1){

            if ((bottom - send_head) > 256){                   // 剩余长度 > 256 ?
                          is_end = false;                      // 是，非最末帧
                          send_size  = 256;                    // 发送长度＝256字节
            }
            else{
                     is_end = true;
                     send_size = bottom - send_head + 1； // 发送长度= (bottom - send_head)+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_CSI)；                          // 在发送数据帧之前等待
            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；  // 情况 [C]
                    default:                    return   rc；        break；  // 情况 [B]
            }
            if (fl_st2 != FLST_ACK){                        // ST2 = ACK ?
                    rc = decode_status(fl_st2)；            // 否
                    return    rc；                         // 情况 [D]
            }

            if (is_end)                                   // 发送所有用户数据?
                    break；                               // 是
            //continue；
    }
    /**************************************************************************************/
    /*        检查内部检验                                  */
    /**************************************************************************************/

    fl_wait(tWT5 * block_num)；                                          // 等待

    rc = fl_csi_getstatus(tWT5_MAX * block_num)；                         // 获取状态帧
    switch(rc) {
            case    FLC_NO_ERR:          return    rc；       break；  // 情况 [A]
            case    FLC_DFTO_ERR:        return    rc；       break；  // 情况 [C]
            default:                     return    rc；       break；  // 情况 [E]
    }
    return    rc；

}
```

8.10 验证命令

8.10.1 处理程序流程图

验 证 命 令 处 理 程 序



8.10.2 处理程序的描述说明

<1> 从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2> 由命令帧发送处理发送验证命令。
<3> 从命令发送起等待直到状态检查处理为止(等待时间 t_{WT6}(MAX.))。
<4> 由状态检查处理获得状态帧。
<5> 依据状态检查处理结果，执行下面的处理。

当处理正常结束时：　　执行<6>。
当处理异常结束时：　　异常终止 [B]
当发生超时错误时：　　返回超时错误[C]。

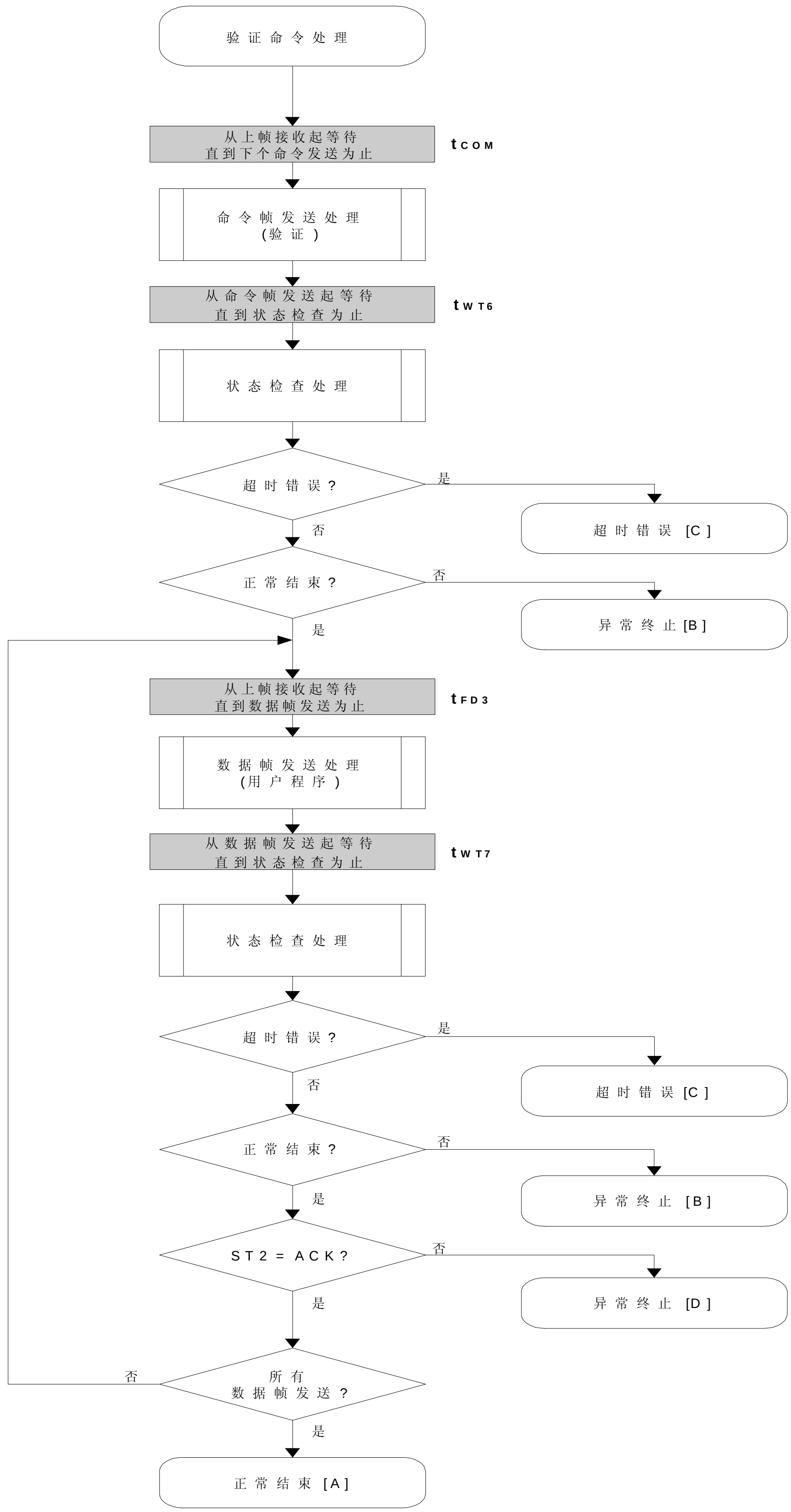
<6> 从上帧接收起等待直到下个数据帧发送为止 (等待时间 t_{FD3})。
<7> 由数据帧发送处理发送用户数据验证。
<8> 从数据帧发送起等待直到状态检查处理为止(等待时间 t_{WT7}(MAX.))。
<9> 由状态检查处理获得状态帧。
<10> 依据状态检查处理的结果执行下面的处理 (状态码 (ST1/ST2)) (同样参考处理程序流程图和流程图)。

当 ST1 = 异常终止时：　　异常终止 [B]
当 ST1 = 超时错误时：　　返回超时错误[C]。
当 ST1 = 正常完成：　　依据 ST2 的值，执行下面的处理。
- 当 ST2 ≠ ACK 时：　异常终止 [D]
- 当 ST2 = ACK 时：　如果所有的数据帧发送完成，处理正常结束[A]。
  如果仍存在需要发送的数据帧，从<6>重新执行程序处理。

8.10.3 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且验证正常结束。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。
异常终止 [D]	验证错误	0FH (ST2)	验证失败，或发生另一个错误。

8.10.4 流程图


验证命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(验证)
从命令帧发送起等待
直到状态检查为止
tWT6
状态检查处理
超时错误？
是
超时错误 [C]
否
正常结束？
否
异常终止 [B]
是
从上帧接收起等待
直到数据帧发送为止
tFD3
数据帧发送处理
(用户程序)
从数据帧发送起等待
直到状态检查为止
tWT7
状态检查处理
超时错误？
是
超时错误 [C]
否
正常结束？
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
正常结束 [A]

8.10.5 程序举例说明

下面是验证命令处理的程序举例。

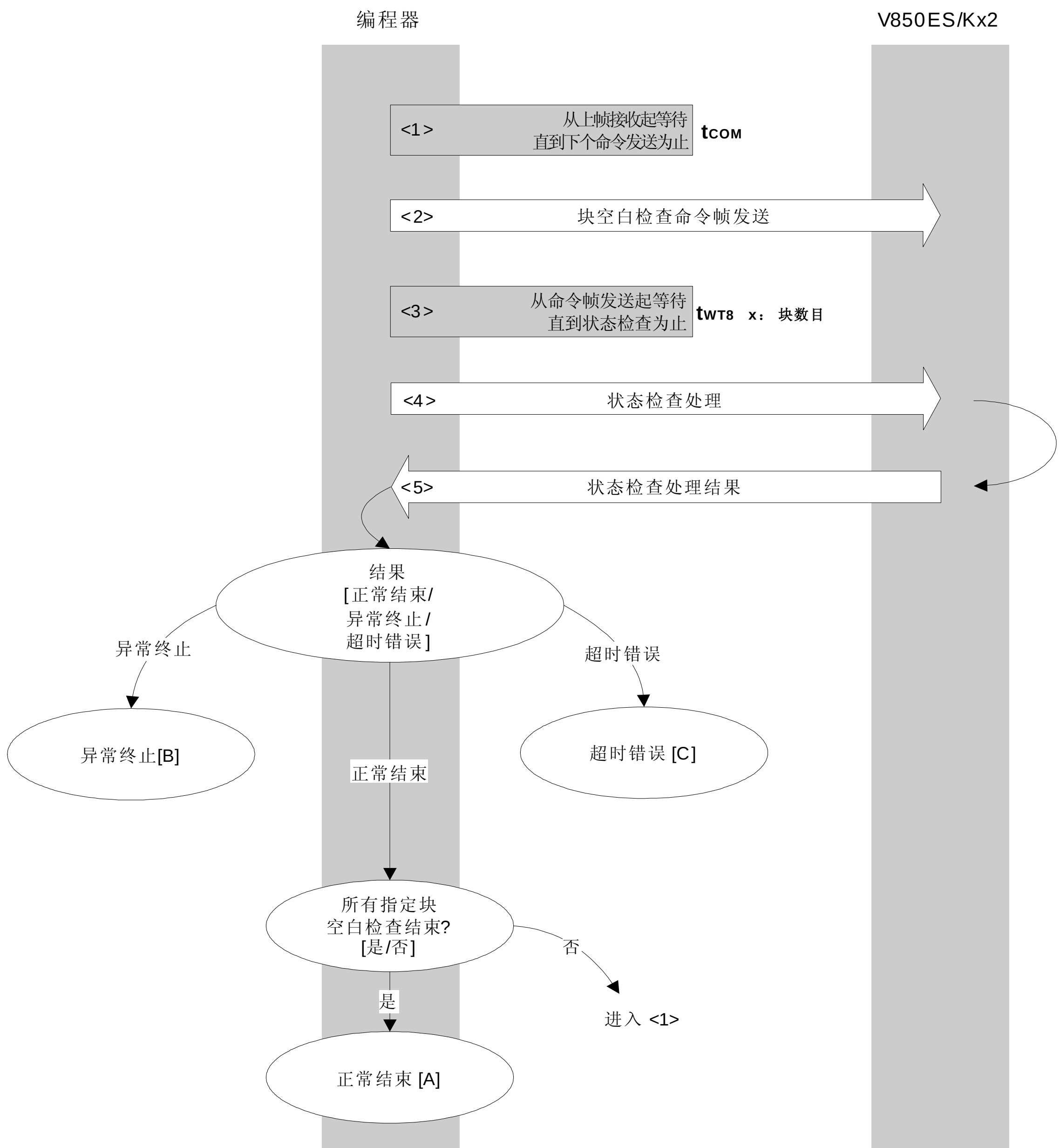
```
/*****************************************************************************************************/
/*                                                                                                   */
/*   验证命令(CSI)                                                                                   */
/*                                                                                                   */
/*****************************************************************************************************/
/*   [i] u32 top          ... 开始地址                                                               */
/*   [i] u32 bottom       ... 结束地址                                                               */
/*   [i] u8  *buf         ... 指向验证数据缓冲器                                                     */
/*   [r] u16              ... 错误编码                                                               */
/*****************************************************************************************************/
u16        fl_csi_verify(u32 top， u32 bottom， u8 *buf)
{
  u16      rc；
  u32      send_head， send_size；
  bool     is_end；

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


  /************************************************************************************/
  /*              发送命令并检查状态                                                  */
  /************************************************************************************/
  fl_wait(tCOM)；
  put_cmd_csi(FL_COM_VERIFY， 7， fl_cmd_prm)；          // 发送"验证"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； // 情况 [C]
           default:                        return    rc；          break； // 情况 [B]
  }


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

  while(1){

           if ((bottom - send_head) > 256){              // 剩余长度 > 256 ?
                    is_end = false；                      // 是，非最末帧
                    send_size  = 256；                    // 发送长度= 256字节
           }
           else{
                    is_end = true；
                    send_size = bottom - send_head + 1； // 发送长度 = (bottom - send_head)+1字节
                                                         //

           }
```

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

            fl_wait(tFD3_CSI)；                              // 在发送数据帧之前等待
            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； // 情况 [C]
                    default:                      return   rc；       break； // 情况 [B]
            }
            if (fl_st2 != FLST_ACK){                    // ST2 = ACK ?
                    rc = decode_status(fl_st2)；        // 否
                    return   rc；                       // 情况 [D]
            }

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

    }
    return    FLC_NO_ERR；                             // 情况 [A]

}
```

8.11 块空白检查命令

8.11.1 处理程序流程图

块空白检查命令处理程序



8.11.2 处理程序的描述说明

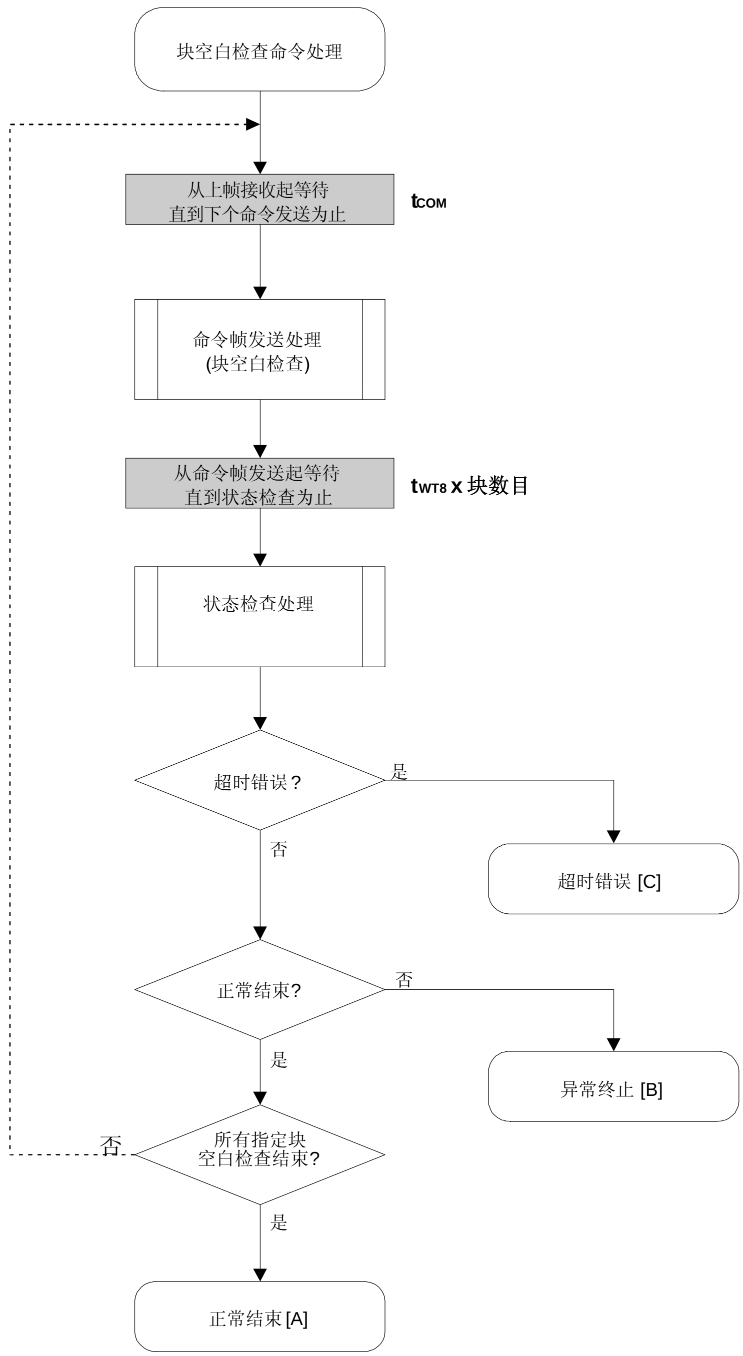
<1> 从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2> 由命令帧发送处理发送块空白检查命令。
<3> 从命令发送起等待直到状态检查处理为止(等待时间 t_{WT8}(MAX.) × 块的数目)。
<4> 由状态检查处理获得状态帧。
<5> 依据状态检查处理结果，执行下面的处理。

当发生超时错误时：	返回超时错误[C]。
当处理异常结束时：	异常终止 [B]
当处理正常结束时：	如果所有指定块的空白检查还没有完成，处理改变块编号，从<1>重新执行程序。 如果所有指定块的空白检查结束，处理正常结束[A]。

8.11.3 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且所有的指定块都是空白的。
异常终止 [B]	参数错误	05H	• 在 flash 存储器的范围之外指定的开始/结束地址。 • 指定的开始/结束地址不是块的开始/结束地址。
	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
	MRG11 错误	1BH	在 flash 存储器中指定的块不是空白的。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。

8.11.4 流程图


块空白检查命令处理
从上帧接收起等待
直到下个命令发送为止
t_{COM}
命令帧发送处理
(块空白检查)
从命令帧发送起等待
直到状态检查为止
t_{WT8} x 块数目
状态检查处理
超时错误？
是
否
超时错误 [C]
正常结束?
否
是
异常终止 [B]
所有指定块
空白检查结束?
否
是
正常结束 [A]

8.11.5 程序举例说明

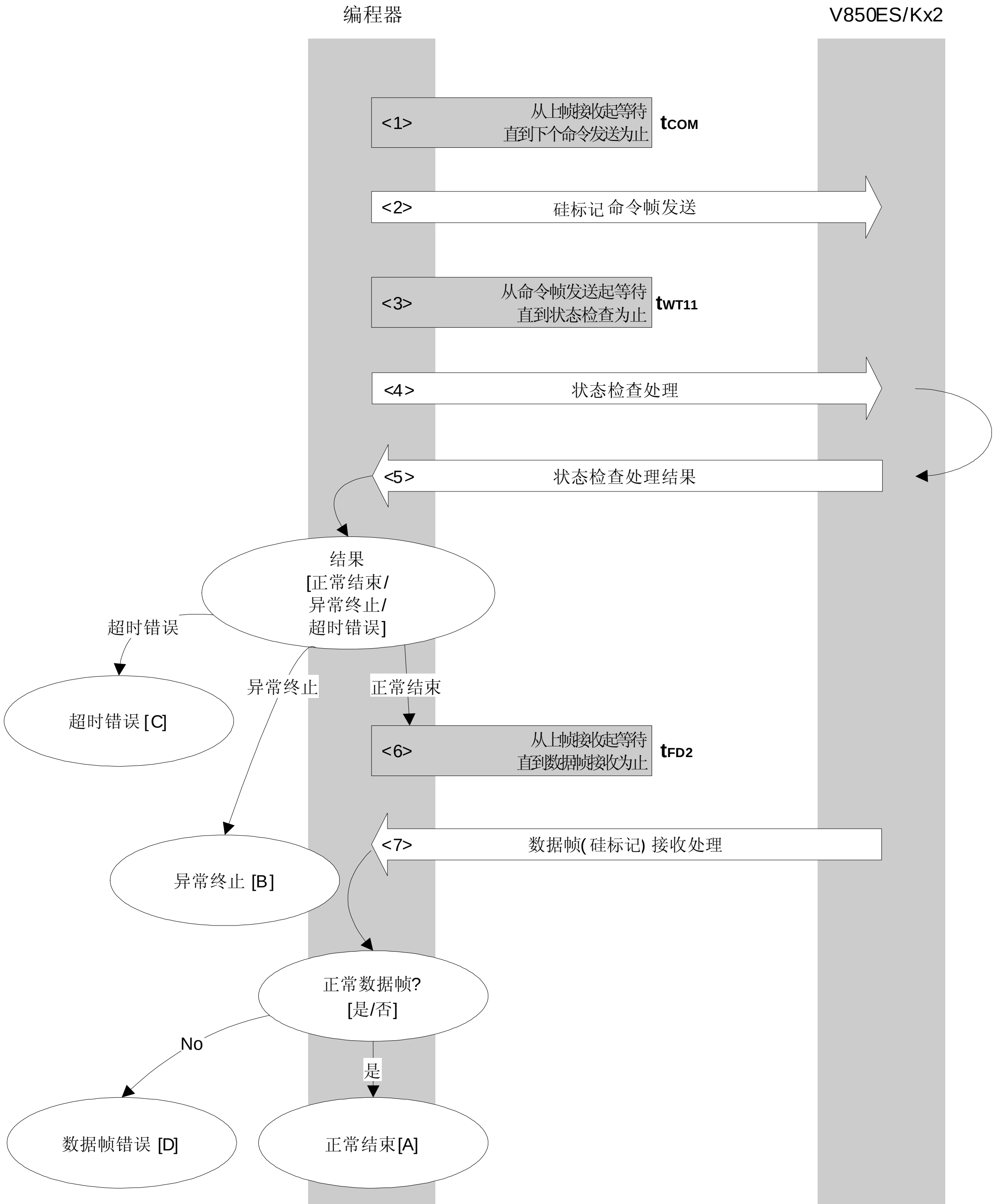
下面是块空白检查命令对一个存储块进行处理的程序举例。

```
/********************************************************************************************/
/*                                                                                          */
/*        块空白检查命令(CSI)                                                                */
/*                                                                                          */
/********************************************************************************************/
/*    [i] u32 top            ... 开始地址                                                   */
/*    [i] u32 bottom         ... 结束地址                                                   */
/*    [r] u16                ... 错误编码                                                   */
/********************************************************************************************/
u16         fl_csi_blk_blank_chk(u32 top， u32 bottom)
{
   u16        rc；
   u16        block_num；

   set_range_prm(fl_cmd_prm， top， bottom)；                    // 设置SAH/SAM/SAL， EAH/EAM/EAL
   block_num = get_block_num(top， bottom)；                      // 获取块编号

   fl_wait(tCOM)；                                         // 在发送命令帧之前等待

   put_cmd_csi(FL_COM_BLOCK_BLANK_CHK， 7， fl_cmd_prm)；// 发送"块空白检查"命令
                                                                //

   fl_wait(tWT8 * block_num)；

   rc = fl_csi_getstatus(tWT8_MAX * block_num)；           // 获取状态帧
//switch(rc) {
//
// case       FLC_NO_ERR:              return     rc；       break；   // 情况 [A]
// case       FLC_DFTO_ERR:            return     rc；       break；   // 情况 [C]
// default:                            return     rc；       break；   // 情况 [B]
//}
   return rc；
}
```

8.12 ‘硅标记’命令

8.12.1 处理程序流程图

‘硅标记’命令处理程序



8.12.2 处理程序的描述说明

<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送‘硅标记’命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT11}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	执行<6>。
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误[C]。

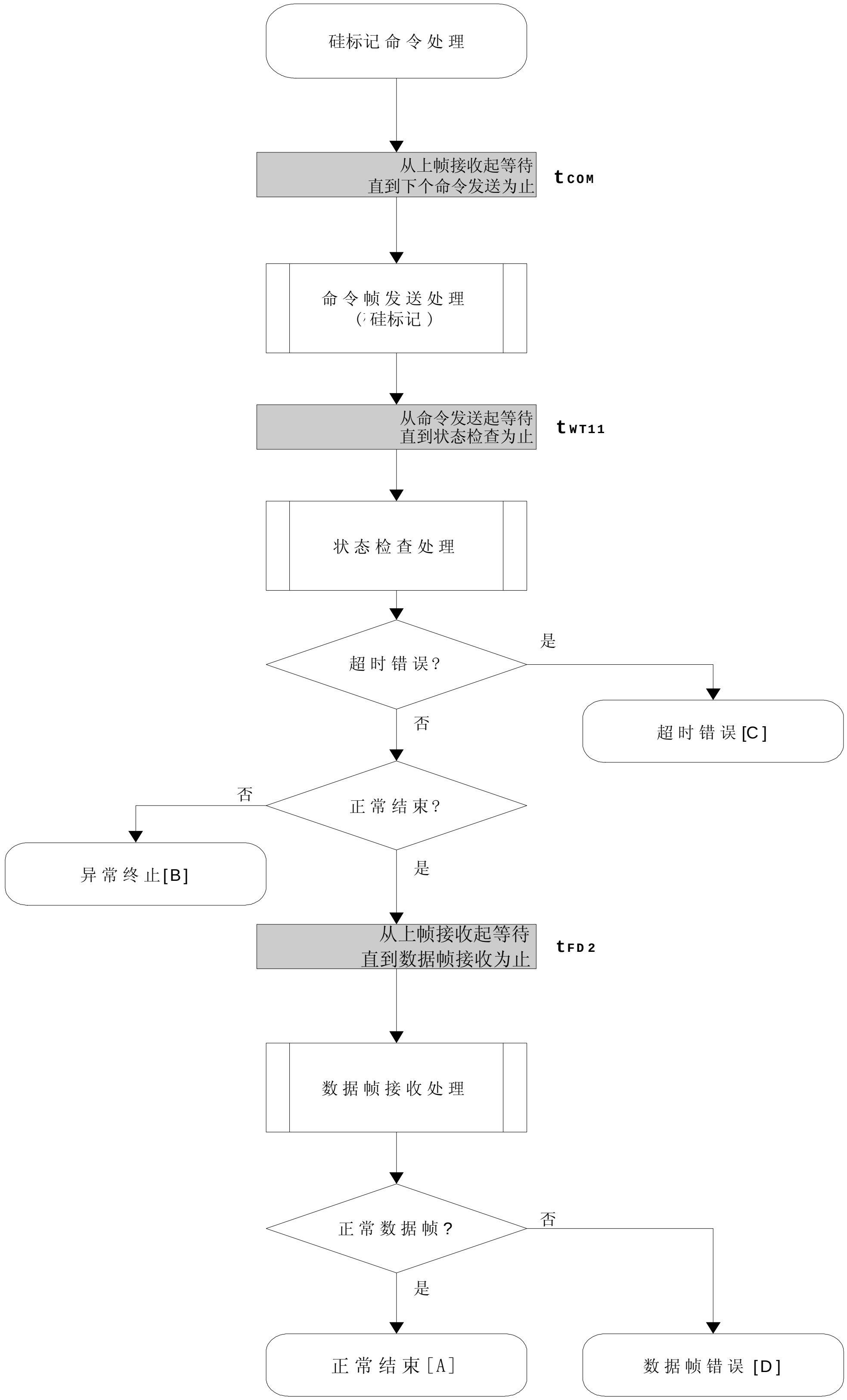
<6>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{FD2})。
<7>　检查接收的数据帧（‘硅标记’数据）。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且‘硅标记’正常获得。
异常终止 [B]	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。
数据帧错误 [D]		–	接收的‘硅标记’数据的数据帧校验和不匹配。

8.12.4 流程图



硅标记命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
（硅标记）
从命令发送起等待
直到状态检查为止
tWT11
状态检查处理
超时错误?
是
超时错误[C]
否
正常结束?
否
异常终止[B]
是
从上帧接收起等待
直到数据帧接收为止
tFD2
数据帧接收处理
正常数据帧?
否
是
正常结束[A]
数据帧错误[D]

8.12.5 程序举例说明

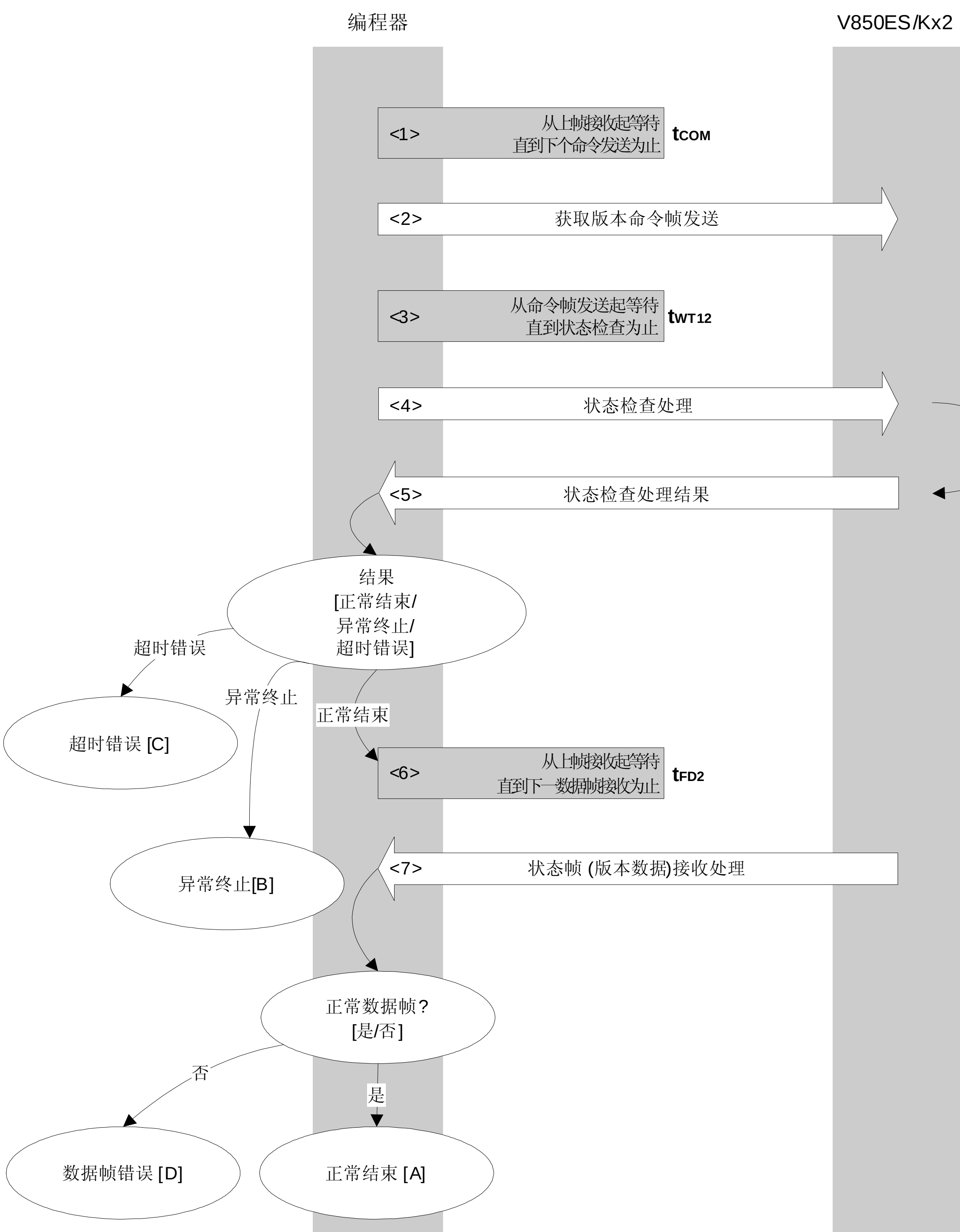
下面是'硅标记'命令处理的程序举例。

```
/*****************************************************************************************************/
/*                                                                                                   */
/*    获取'硅标记'命令(CSI)                                                                           */
/*                                                                                                   */
/*****************************************************************************************************/
/*    [i] u8 *sig               ... 指向'标记'保存区域                                               */
/*    [r] u16                   ... 错误编码                                                         */
/*****************************************************************************************************/
u16          fl_csi_getsig(u8 *sig)
{
   u16        rc;

   fl_wait(tCOM);                                                  // 在发送命令帧之前等待

   put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);   // 发送"硅标记" 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;   // 情况 [C]
              default:                return  rc;         break;   // 情况 [B]
   }

   fl_wait(tFD2_SIG);                                      // 在获取数据帧之前等待

   rc = get_dfrm_csi(fl_rxdata_frm);                       // 获取数据帧('标记' 数据)

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

8.13 获取版本信息命令

8.13.1 处理程序流程图

获取版本命令处理程序



8.13.2 处理程序的描述说明

<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送获取版本命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT12}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

当处理正常结束时：　执行<6>。
当处理异常结束时：　异常终止 [B]
当发生超时错误时：　返回超时错误[C]。

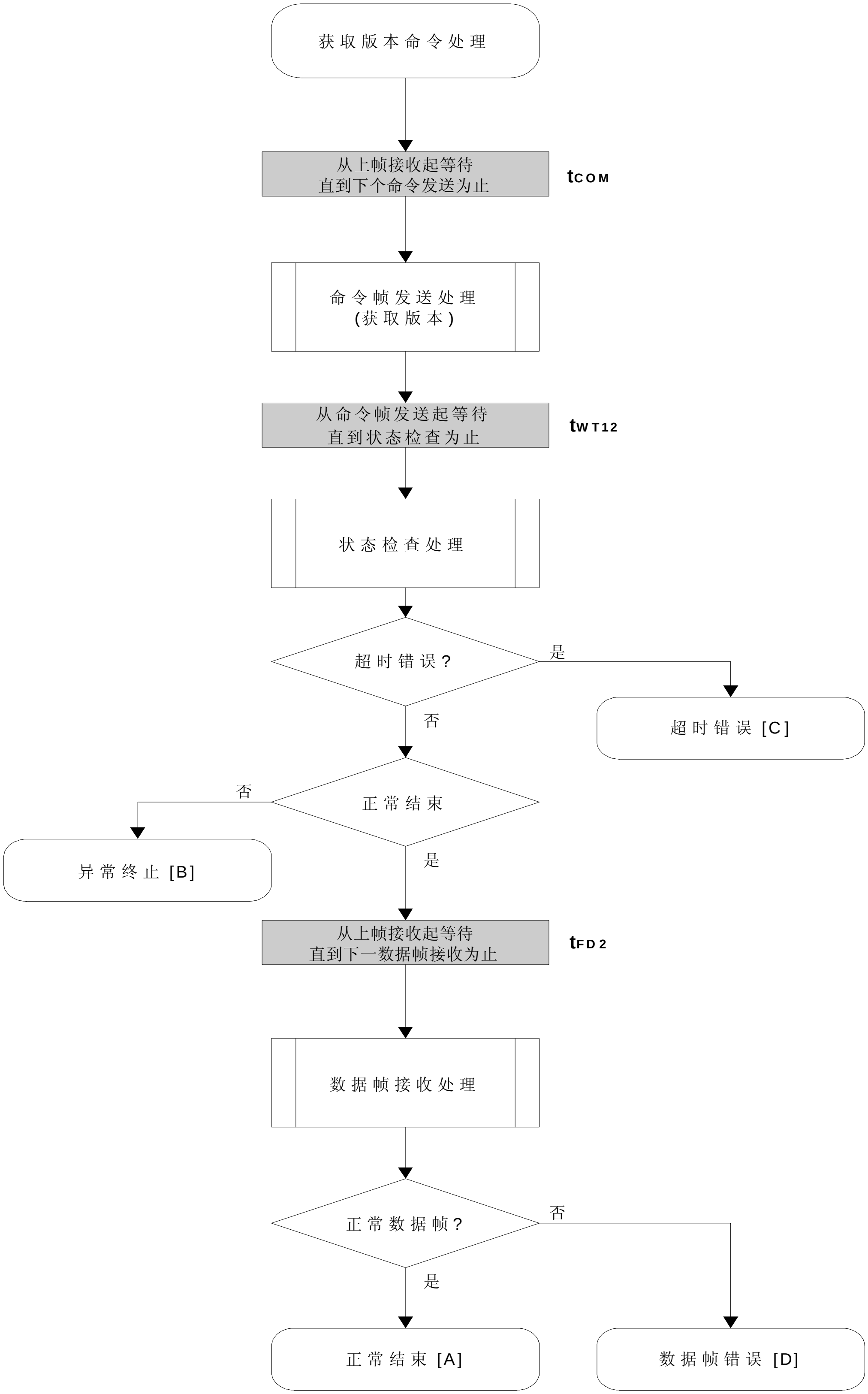
<6>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{FD2})。
<7>　检查接收的数据帧（版本数据）。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且版本数据获取正常。
异常终止 [B]	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。
数据帧错误 [D]		–	接收的版本数据的数据帧校验和不匹配。

8.13.4 流程图


获取版本命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(获取版本)
从命令帧发送起等待
直到状态检查为止
tWT12
状态检查处理
超时错误?
是
超时错误 [C]
否
正常结束
否
异常终止 [B]
是
从上帧接收起等待
直到下一数据帧接收为止
tFD2
数据帧接收处理
正常数据帧?
否
是
正常结束 [A]
数据帧错误 [D]

8.13.5 程序举例说明

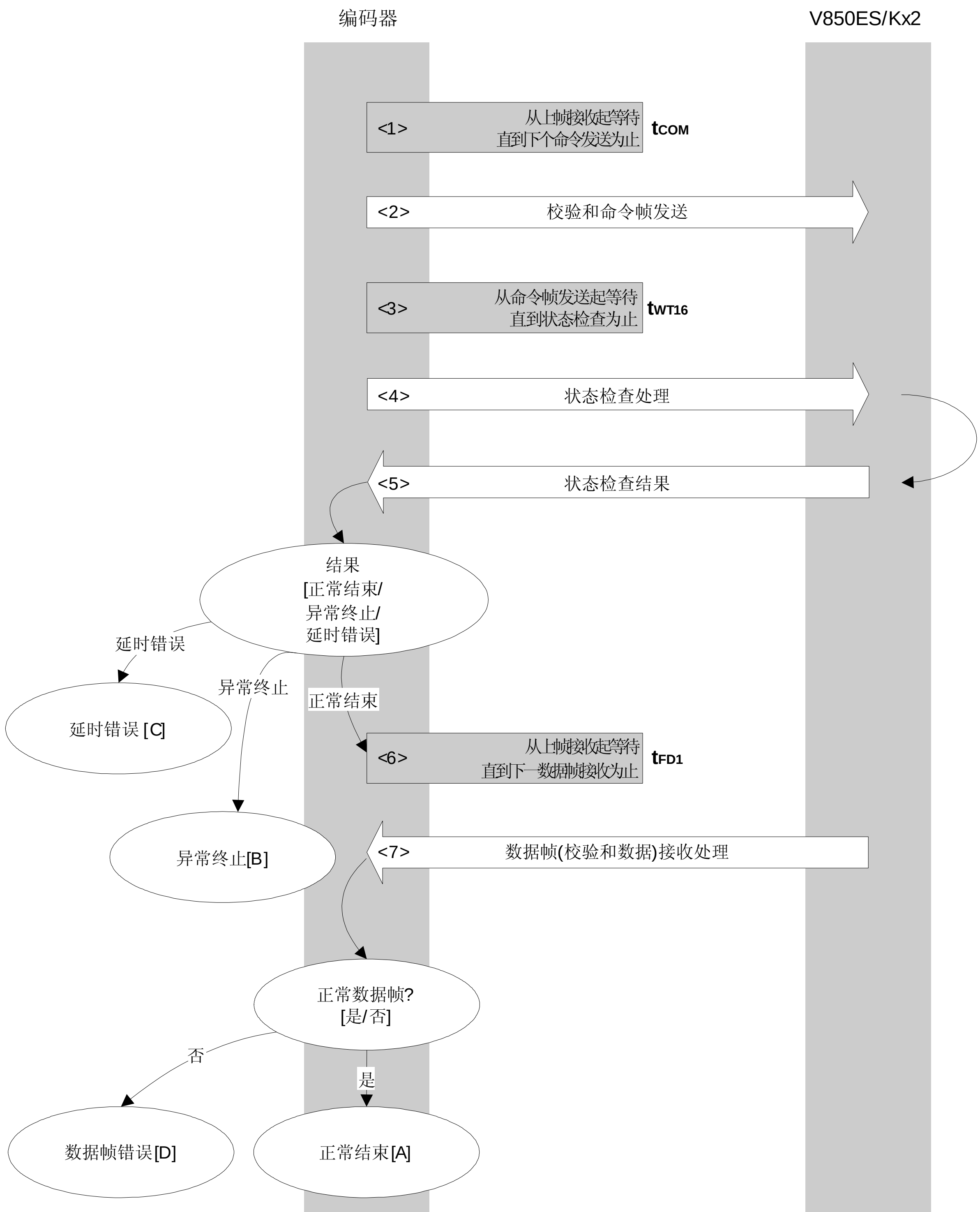
下面是获取版本命令处理程序举例。

```
/***************************************************************************************************/
/*                                                                                                 */
/*     获取器件/固件版本命令(CSI)                                                                    */
/*                                                                                                 */
/***************************************************************************************************/
/*   [i] u8 *buf              ... 指向版本数据存储区域                                               */
/*   [r] u16                  ... 错误编码                                                          */
/***************************************************************************************************/
u16          fl_csi_getver(u8 *buf)
{
   u16       rc；


   fl_wait(tCOM)；                                              // 在发送命令帧之前等待

   put_cmd_csi(FL_COM_GET_VERSION， 1， fl_cmd_prm)；  //发送"获取版本"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；  // 情况 [C]
             default:                 return  rc；      break；  // 情况 [B]
   }

   fl_wait(tFD2_VG)；                                          // 在获取数据帧之前等待

   rc = get_dfrm_csi(fl_rxdata_frm)；                          // 获取版本数据

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

8.14 校验和命令

8.14.1 处理程序流程图

校验和命令处理程序



8.14.2 处理程序的描述说明

<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送校验和命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT16}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

当处理正常结束时：　　执行<6>。
当处理异常结束时：　　异常终止 [B]
当发生超时错误时：　　返回超时错误[C]。

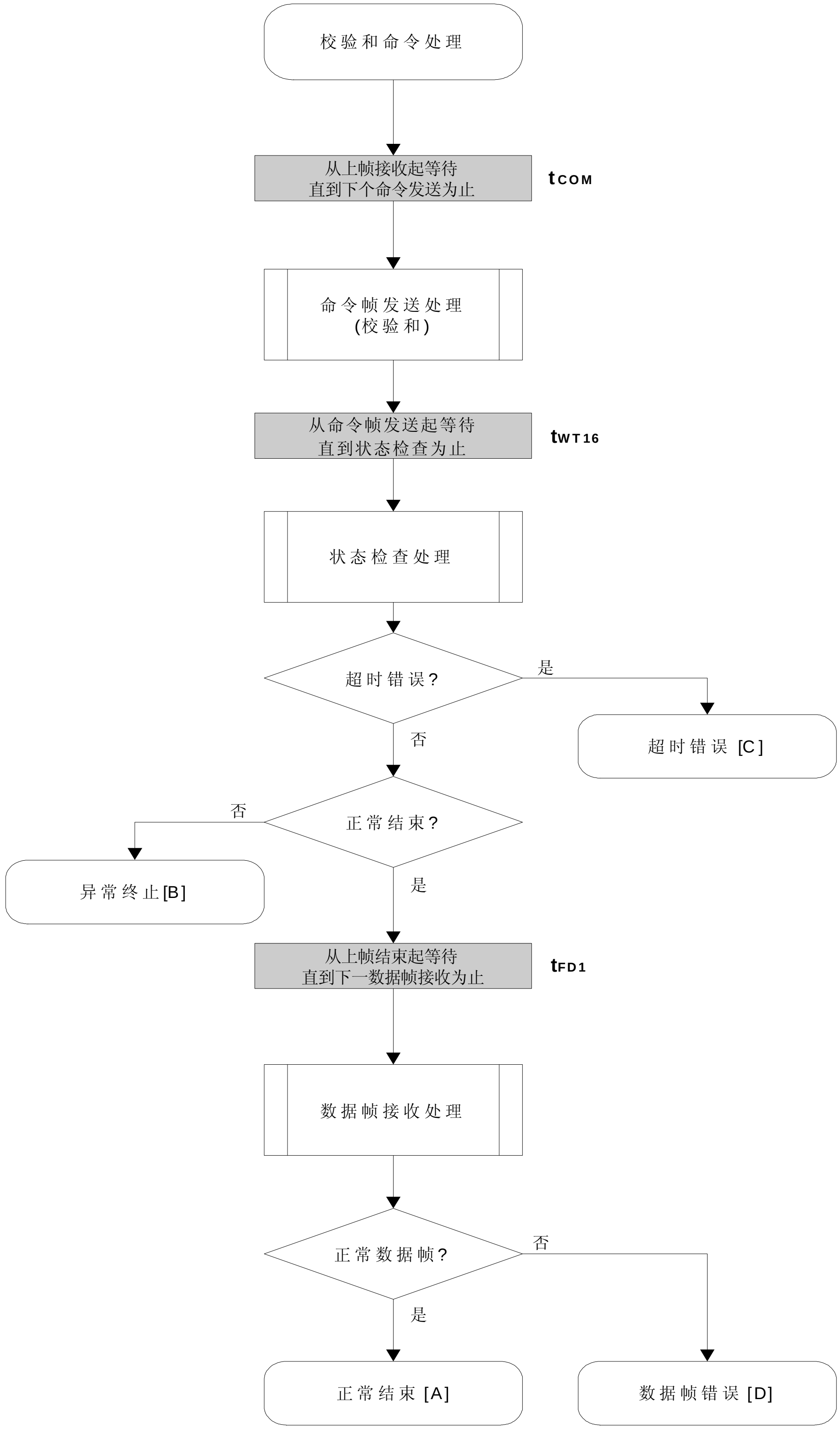
<6>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{FD1})。
<7>　检查接收的数据帧（校验和数据）。

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 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且校验和数据获取正常。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是在 flash 存储器内部以顶部作为开始地址的块(2 KB)的固定地址。
	校验和错误	07H	发送的命令帧校验和不匹配。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。
数据帧错误 [D]		–	接收的版本数据的数据帧校验和作为不匹配。

8.14.4 流程图



校验和命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(校验和)
从命令帧发送起等待
直到状态检查为止
tWT16
状态检查处理
超时错误?
是
超时错误 [C]
否
正常结束?
否
异常终止[B]
是
从上帧结束起等待
直到下一数据帧接收为止
tFD1
数据帧接收处理
正常数据帧?
否
是
正常结束 [A]
数据帧错误 [D]

8.14.5 程序举例说明

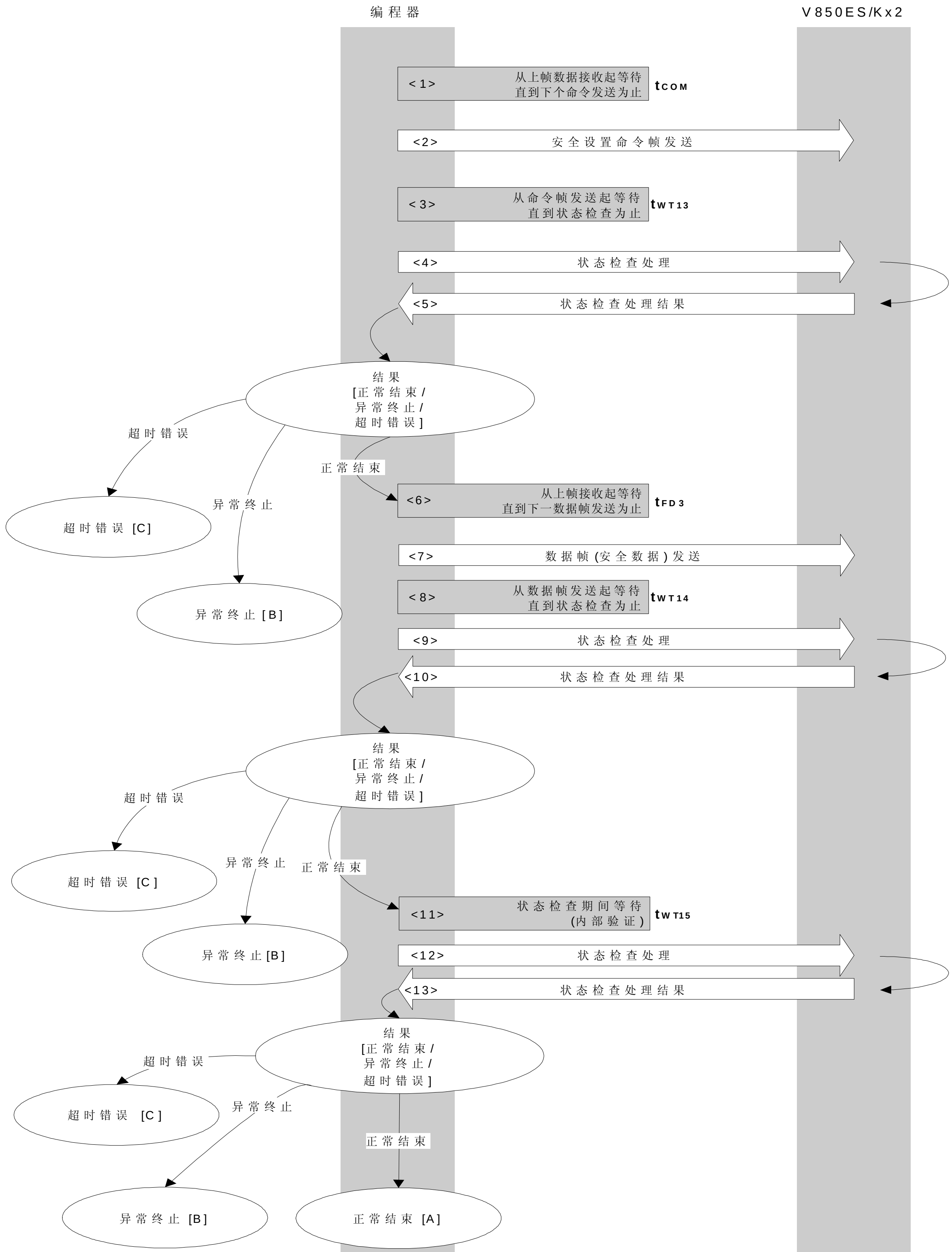
下面是校验和命令处理的程序举例。

```
/**********************************************************************************************/
/*                                                                                            */
/*   获取校验和命令(CSI)                                                                        */
/*                                                                                            */
/**********************************************************************************************/
/*   [i] u16 *sum          ... 指向校验和保存区域                                               */
/*   [i] u32 top           ... 开始地址                                                         */
/*   [i] u32 bottom        ... 结束地址                                                         */
/*   [r] u16               ... 错误编码                                                         */
/**********************************************************************************************/
u16        fl_csi_getsum(u16 *sum， u32 top， u32 bottom)
{
   u16       rc;
   u16       block_num;

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

   block_num = get_block_num(top， bottom)；               // 获取块编号

   /*****************************************************************************/
    /*          发送命令                                                       */
   /*****************************************************************************/
   fl_wait(tCOM)；                                        // 在发送命令帧之前等待

     put_cmd_csi(FL_COM_GET_CHECK_SUM， 7， fl_cmd_prm)；   // 发送"校验和"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;   // 情况 [C]
            default:                return  rc;       break;   // 情况 [B]
   }


   /*****************************************************************************/
   /*   获取数据帧(校验和数据)                                                    */
   /*****************************************************************************/
   fl_wait(tFD1 * block_num);                             // 在获取数据帧之前等待

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

   if (rc){                                         // 如果错误，
            return rc;                              // 情况 [D]
   }

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

8.15 安全设置命令

8.15.1 处理程序流程图

安全设置帧处理程序



8.15.2 处理程序的描述说明

<1>　从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2>　由命令帧发送处理发送安全设置命令。
<3>　从命令发送起等待直到状态检查处理为止(等待时间 t_{WT13}(MAX.))。
<4>　由状态检查处理获得状态帧。
<5>　依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	执行 <6>。
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误[C]。

<6>　从上帧接收起等待直到数据帧发送为止 (等待时间 t_{FD3})。
<7>　由数据帧发送处理发送数据帧（安全设置数据）。
<8>　从数据帧发送起等待直到状态检查处理为止 (等待时间 t_{WT14}(MAX.))。
<9>　由状态检查处理获得状态帧。
<10> 依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	执行 <11>。
当处理异常结束时：	异常终止 [D]
当发生超时错误时：	返回超时错误[C]。

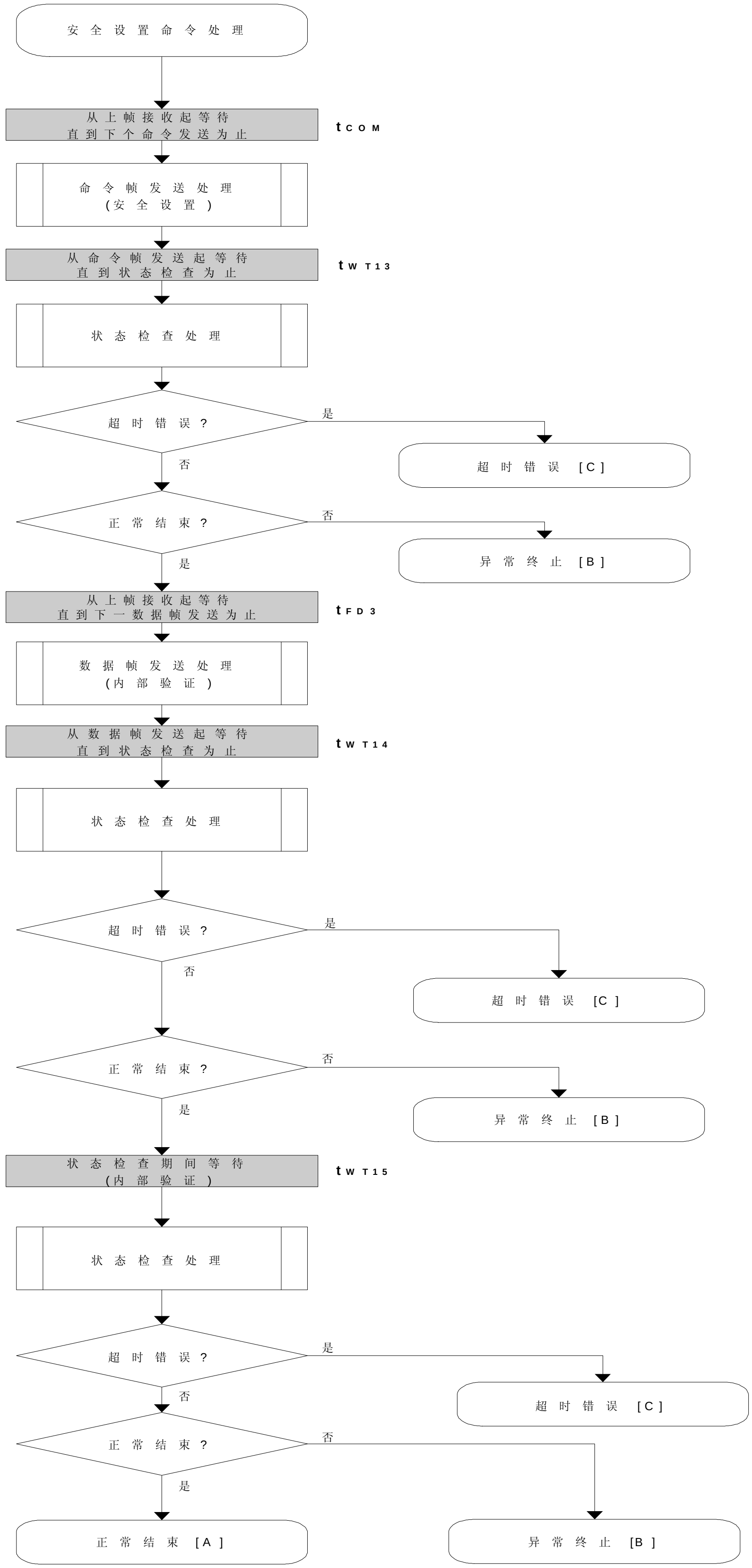
<11> 等待直到状态获取为止(内部校验完成) (等待时间 t_{WT15}(MAX.))。
<12> 由状态检查处理获得状态帧。
<13> 依据状态检查处理结果，执行下面的处理。

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]
当发生超时错误时：	返回超时错误[C]。

8.15.3 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且安全设置执行正常。
异常终止 [B]	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	• 在设置中试图使能标志已经被禁止。 • 在安全设置中引导块重写被禁止。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	在特定的时间内没有接收到状态帧。

8.15.4 流程图


安全设置命令处理
从上帧接收起等待
直到下个命令发送为止
tCOM
命令帧发送处理
(安全设置)
从命令帧发送起等待
直到状态检查为止
tWT13
状态检查处理
超时错误？
是
超时错误 [C]
否
正常结束？
否
异常终止 [B]
是
从上帧接收起等待
直到下一数据帧发送为止
tFD3
数据帧发送处理
(内部验证)
从数据帧发送起等待
直到状态检查为止
tWT14
状态检查处理
超时错误？
是
否
超时错误 [C]
正常结束？
否
异常终止 [B]
是
状态检查期间等待
(内部验证)
tWT15
状态检查处理
超时错误？
是
否
超时错误 [C]
正常结束？
否
是
正常结束 [A]
异常终止 [B]

8.15.5 程序举例说明

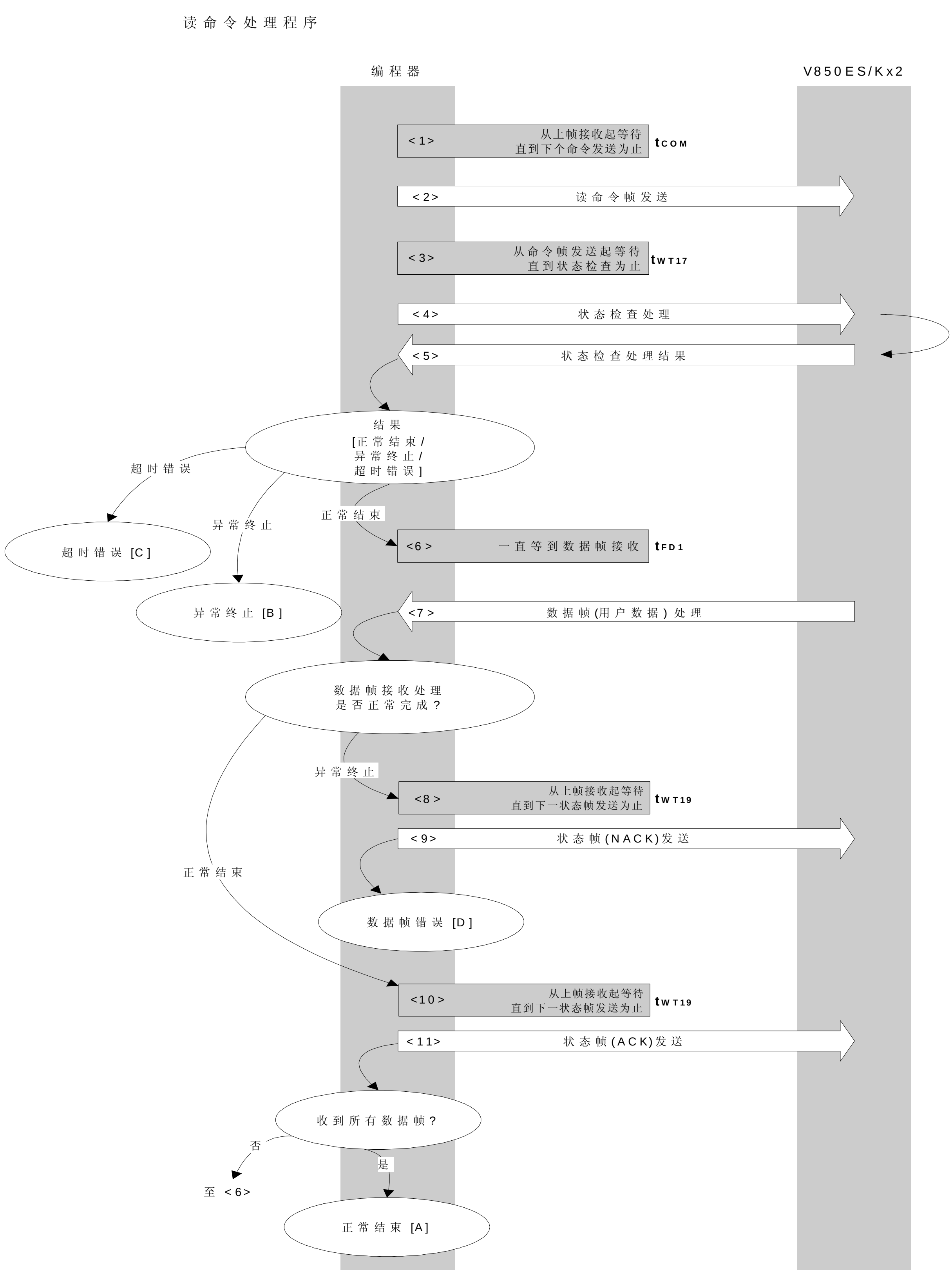
下面是安全设置命令处理的程序举例。

```
/*********************************************************************************************/
/*                                                                                           */
/*    设置安全标志命令(CSI)                                                                   */
/*                                                                                           */
/*********************************************************************************************/
/*    [i] u8 scf                ... 安全标志数据                                              */
/*    [i] u8 bot                ... 引导块编号                                                */
/*  [r] u16                     ... 错误编码                                                  */
/*********************************************************************************************/
u16          fl_csi_setscf(u8 scf， u8 bot)
{
   u16        rc；

   /******************************************************************************************/
   /*        设置参量                                                                        */
   /******************************************************************************************/
   fl_cmd_prm[0] = 0x00；                          // 第一个字节必须是0x00
   fl_cmd_prm[1] = 0x00；                          // 第二个字节必须是0x00
   fl_txdata_frm[0] = (scf |= 0b11100000)；        // "FLG" (高3位必须是'1'
                                                              // (必须确定))
   fl_txdata_frm[1] = bot；                        // "BOT"



   /*************************************************************************************/
   /*              发送命令                                                              */
   /*************************************************************************************/
   fl_wait(tCOM)；                                                  // 在发送命令帧之前等待

   put_cmd_csi(FL_COM_SET_SECURITY， 3， fl_cmd_prm)；  // 发送"安全设置"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；  // 情况 [C]
             default:                     return        rc；         break；  // 情况 [B]
   }

   /*************************************************************************************/
   /*              发送数据帧(安全设置数据)*/
   /*************************************************************************************/
   fl_wait(tFD3_CSI)；                                         // 在获取数据帧之前等待



   put_dfrm_csi(2， 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；  // 情况 [C]
          default:                  return        rc；     break；  // 情况 [B]
  }


  /*********************************************************************************/
  /*            检查内部校验                                       */
  /*********************************************************************************/
  fl_wait(tWT15)；

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

8.16 读取命令

8.16.1 处理程序流程图



8.16.2 处理程序的描述说明

<1> 从上帧接收起等待直到下个命令发送为止(等待时间 t_{COM})。
<2> 由命令帧发送处理发送读取命令。
<3> 从命令发送起等待直到状态检查处理为止(等待时间 t_{WT17})。
<4> 由状态检查处理获得状态帧。
<5> 依据状态检查处理结果，执行下面的处理。

当处理正常结束时：	执行<6>。
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误[C]。

<6> 从上帧接收起等待直到数据帧接收为止 (等待时间 t_{WT18})。
<7> 由数据帧接收处理接收数据帧（用户数据）。
依据接收处理的结果，执行下面的处理。

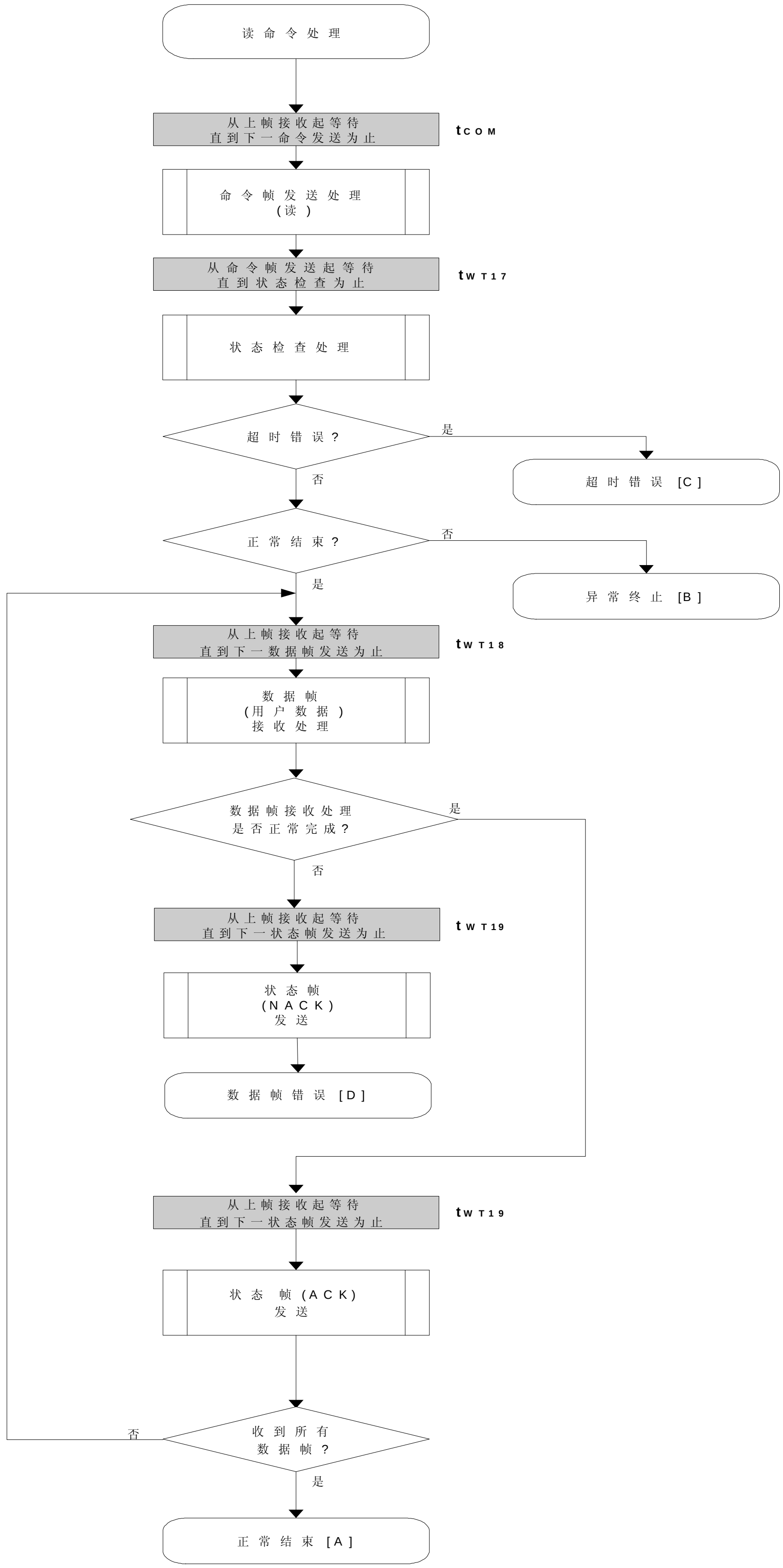
当处理正常结束时：	执行<10>。
当处理异常结束时：	执行<8>。

<8> 从上帧接收起等待直到下个状态帧(NACK)发送为止(等待时间 t_{WT19})。
<9> 由数据帧发送处理发送 NACK 帧。
返回数据帧错误[D]。
<10> 从上帧接收起等待直到下个状态(ACK)帧发送为止(等待时间 t_{WT19})。
<11> ACK 帧由数据帧发送处理发送。
当所有的数据帧接收完成时，返回正常完成状态[A]。
如果仍存在数据帧需要接收，程序从<5>重新执行。

8.16.3 处理完毕后的状态

处理完毕后的状态		状态码	描述说明
正常完成 [A]	确认正常(ACK)	06H	命令正常执行而且读出数据设置正常。
异常终止 [B]	参数错误	05H	指定的开始/结束地址不是块的开始/结束地址。
	校验和错误	07H	发送命令帧或数据帧的校验和不匹配。
	保护错误	10H	在安全设置中读取被禁止。
	确认异常 (NACK)	15H	命令帧数据异常（例如无效数据长度（LEN）或无 ETX）。
超时错误 [C]		–	状态帧或数据帧在特定时间内不被接收。
数据帧错误 [D]		–	接收的读出数据的数据帧校验和不匹配。

8.16.4 流程图


读 命 令 处 理
从 上 帧 接 收 起 等 待
直 到 下 一 命 令 发 送 为 止
tCOM
命 令 帧 发 送 处 理
（读 ）
从 命 令 帧 发 送 起 等 待
直 到 状 态 检 查 为 止
tWT17
状 态 检 查 处 理
超 时 错 误 ?
是
超 时 错 误 [C]
否
正 常 结 束 ?
否
异 常 终 止 [B]
是
从 上 帧 接 收 起 等 待
直 到 下 一 数 据 帧 发 送 为 止
tWT18
数 据 帧
（用 户 数 据 ）
接 收 处 理
数 据 帧 接 收 处 理
是 否 正 常 完 成 ?
是
否
从 上 帧 接 收 起 等 待
直 到 下 一 状 态 帧 发 送 为 止
tWT19
状 态 帧
（NACK）
发 送
数 据 帧 错 误 [D]
从 上 帧 接 收 起 等 待
直 到 下 一 状 态 帧 发 送 为 止
tWT19
状 态 帧 （ACK）
发 送
收 到 所 有
数 据 帧 ?
否
是
正 常 结 束 [A]

8.16.5 程序举例说明

下面是读取命令处理的程序举例。

```
/*****************************************************************************************/
/*                                                                                       */
/*    读取命令(CSI)                                                                      */
/*                                                                                       */
/*****************************************************************************************/
/*   [i] u32 top              ... 开始地址                                               */
/*   [i] u32 bottom           ... 结束地址                                               */
/*   [r] u16                  ... 错误编码                                               */
/*****************************************************************************************/
u16         fl_csi_read(u32 top， u32 bottom)
{
  u16       rc；
  u32       read_head；
  u16       len；
  u8        hooter；

  /*****************************************************************************/
  /*            设置参量                                                       */
  /*****************************************************************************/

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

  /*****************************************************************************/
  /*            发送命令并检查状态                                             */
  /*****************************************************************************/
  fl_wait(tCOM)；                                        // 在发送命令之前等待

  put_cmd_csi(FL_COM_READ， 7， fl_cmd_prm)；            // 发送"读取"命令

  fl_wait(tWT17)；                                       // 等待

  rc = fl_csi_getstatus(tWT17_MAX)；                              // 获取状态帧
  switch(rc) {
          case     FLC_NO_ERR:                          break；     // 继续
// case   FLC_DFTO_ERR:              return    rc；     break；     // 情况 [C]
          default:            return    rc；     break；            // 情况 [B]
  }
  /*****************************************************************************/
  /*            接收用户数据                                                   */
  /*****************************************************************************/
  read_head = top；

  while(1){
          fl_wait(tWT18)；

          rc = get_dfrm_csi(fl_rxdata_frm)；                  // 从FLASH获取ROM数据
          switch(rc) {
                  case     FLC_NO_ERR:           break；     // 继续
          //      case     FLC_RX_DFSUM_ERR:
                  default:                                    // 情况 [D]
                           fl_wait(tWT19)；
                           put_sfrm_csi(FLST_NACK)；          // 发送状态(NACK)帧
                           return    rc；
                           break；
          }
          fl_wait(tWT19)；
          put_sfrm_csi(FLST_ACK)；                               // 发送状态(ACK)帧
```

```
        /*************************************************************************************/
        /*                保存ROM数据                                                  */
        /*************************************************************************************/
        if ((len = fl_rxdata_frm[OFS_LEN]) == 0)                         // 获取长度
                len = 256;

        memcpy(read_buf+read_head， fl_rxdata_frm+2， len)；          // 保存到外部RAM

        read_head += len;

        /*************************************************************************************/
        /*                结束检查                                                     */
        /*************************************************************************************/
        hooter = fl_rxdata_frm[len + 3];
        if (hooter == FL_ETB)                                   // 帧尾?
                continue;                                       // 否
        break;                                                  // 是
    }

    return    FLC_NO_ERR;
}
```

第九章　FLASH 存储器编程的指标参数

这一章描述了在 flash 存储器编程模式下，编程器和 V850ES/Kx2 之间的参数特性。
当设计程序时，必须参考 V850ES/Kx2 用户手册的电气特性。

<操作时钟 (fx)>

V850ES/Kx2 的内部时钟依照编程器使用‘振荡频率设置’命令指定的振荡频率值(fx)而改变。

当 2.0 MHz $\le f_X \le$ 5.0 MHz 时：　$f_{XX} = f_X \times 4$ (PLL 模式)
当 5.0 MHz $< f_X \le$ 10.0 MHz 时：　$f_{XX} = f_X \times 1$ (时钟直通模式)

因此，在‘振荡频率设置’命令前 t_{WT9} 时间内，其值采用 f_X，而在‘振荡频率设置’命令 t_{WT9} 时间之后，其值采用 f_{XX}。

9.1　Flash 存储器编程模式设定时间

	参数	符号	最小值	典型值	最大值
	VDD↑ 至 FLMD0↑	t_{DP}	1 ms		
	FLMD0↑ 至 $\overline{\text{RESET}}$↑	t_{PR}	2 ms		
<R>	计数开始时间从 $\overline{\text{RESET}}$↑至 FLMD0 注 1	t_{RP}	33,740/f_X		
<R>	计数结束时间从 $\overline{\text{RESET}}$↑至 FLMD0 注 1	t_{RPE}			132,276/f_X
	FLMD0 计数器高电平/低电平宽度	t_{PW}	10 μs		100 μs
<R>	复位命令等待 (CSI/CSI + HS)	t_{RC}	181,787/f_X		3 s
	低电平数据 1 等待(UART)	t_{R1}			3 s
	低电平数据 2 等待(UART)	t_{2C}	30,000/f_X		3 s
	读取命令等待(UART)	t_{12}	30,000/f_X		3 s
	低电平数据 1/2 宽度 (UART)注 2	t_{L1}, t_{L2}		注 2	
	FLMD0 计数器上升/下降时间	–			1 μs

注　**1.** 推荐使用(66,612/f_X + 549,591/f_X)/2 作为 FLMD0 脉冲输入计时的标准值。
　　2. 低电平的宽度与 9,600 bps 速率时 00H 数据的宽度相同，而且这里描述的值是此数据宽度的一半。

9.2　编程特性

<R>

等待	状态	符号	串行 I/F	最小值	最大值
数据帧发送/接收之间	数据帧接收	t_{DR}	CSI	440/f_X	3 s
			UART	140/f_X	3 s
	数据帧发送	t_{DT}	CSI	560/f_X	3 s
			UART	0 [注1]	3 s
从状态命令帧接收起直到状态帧发送为止	片擦除	t_{SF}	CSI	2,160/f_X	3 s
	块擦除			1,800/f_X	
	编程			1,320/f_X	
	验证				
	块空白检查				
	安全设置			1,920/f_X	
	其它命令			1,320/f_X	
从状态帧发送起直到数据帧发送为止(1)	–	t_{FD1} [注2]	CSI	95,520/f_X + 24 μs	114,624/f_X + 29 μs
			UART	0 [注1]	114,624/f_X + 29 μs
从状态帧发送起直到数据帧发送为止(2)	‘硅标记’数据	t_{FD2}	CSI	2,900/f_X + 48 μs	3 s
	版本数据			280/f_X	3 s
	–		UART	0 [注1]	3 s
从状态帧发送起直到数据帧接收为止	–	t_{FD3}	CSI	1,160/f_X	3 s
			UART	6,720/f_X	3 s
从状态帧发送起直到命令帧接收为止	–	t_{COM}	CSI	158/f_X	3 s
			UART	154/f_X	3 s

注　**1.** 当编程器连续接收使能时

　　2. 一块数据的发送时间

备注　等待定义如下。

<t_{DR}，t_{FD3}，t_{COM}>

在完成上次通信后，经过最小时间以后 V850ES/Kx2 为下一次通信准备就绪。

在完成上次通信之后，编程器必须在最小时间和最大时间之间发送下一数据。

<t_{DT}，t_{SF}，t_{FD1}，t_{FD2}>

在完成上次通信后，经过最小时间以后 V850ES/Kx2 为下一次通信准备就绪。

在完成上次通信之后，编程器必须在最小时间和最大时间之间接收下一数据。

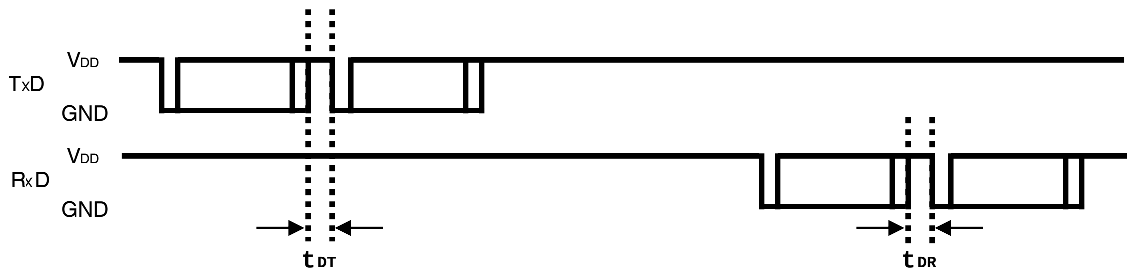
命令	符号	串行 I/F	最小值	最大值
复位	t_{WT0}	CSI	840/f_X	3 s
		UART	注 1	3 s
片擦除	t_{WT1}	–	**[V850ES/KE2]** 1,718,706/f_X + 135.4 ms	**[V850ES/KE2]** 36,374,804/f_X + 43,258.0 ms
			[V850ES/KF2,V850ES/KG2,V850ES/KJ2] 1,718,706/f_X + 237.8 ms	**[V850ES/KF2,V850ES/KG2,V850ES/KJ2]** 36,374,804/f_X + 76,787.6 ms
块擦除	t_{WT2}[注 2]	–	51,601/f_X + 13.7 ms	13,176,706/f_X + 3,497.6 ms
编程	t_{WT3}	CSI	1,500/f_X + 24 μs	3 s
		UART	注 1	3 s
	t_{WT4}[注 3]	–	26,980/f_X + 22.7 ms	623,736/f_X + 285.9 ms
	t_{WT5}[注 2]	CSI	129,207/f_X + 4.2 ms	176,809/f_X + 7.2 ms
		UART	注 1	176,809/f_X + 7.2 ms
验证	t_{WT6}	CSI	440/f_X	3 s
		UART	注 1	3 s
	t_{WT7}[注 3]	CSI	4,240/f_X + 404 μs	10,350/f_X + 423 μs
		UART	注 1	10,350/f_X + 423 μs
块空白检查	t_{WT8}[注 2]	CSI	54,778/f_X + 2.0 ms	75,899/f_X + 3.5 ms
		UART	注 1	75,899/f_X + 3.5 ms
振荡频率设置	t_{WT9}	CSI	154,000/f_X	3 s
		UART	注 1	3 s
波特率设置	t_{WT10}	UART	1,680/f_X	3 s
'硅标记'	t_{WT11}	CSI	520/f_X	3 s
		UART	注 1	3 s
获取版本	t_{WT12}	CSI	520/f_X	3 s
		UART	注 1	3 s
安全设置	t_{WT13}	CSI	460/f_X	3 s
		UART	注 1	3 s
	t_{WT14}	–	94,000/f_X + 4.8 ms	13,219,105/f_X + 3488.7 ms
	t_{WT15}	CSI	482,000/f_X + 16.9 ms	13,607,105/f_X + 3500.8 ms
		UART	注 1	13,607,105/f_X + 3500.8 ms
校验和	t_{WT16}	CSI	640/f_X	3 s
		UART	注 1	3 s
读取	t_{WT17}	CSI	1,520/f_X + 24 μs	3 s
		UART	注 1	3 s
	t_{WT18}[注 3]	CSI	13,920/f_X	3 s
		UART	注 1	3 s
	t_{WT19}	–	116/f_X	注 4

注　**1.** 在命令发送之前，编程器接收必须使能。
2. 一块数据的发送时间。
3. 256 字节数据的发送时间。
4. 直到编程器发送“ACK”为止的时间。

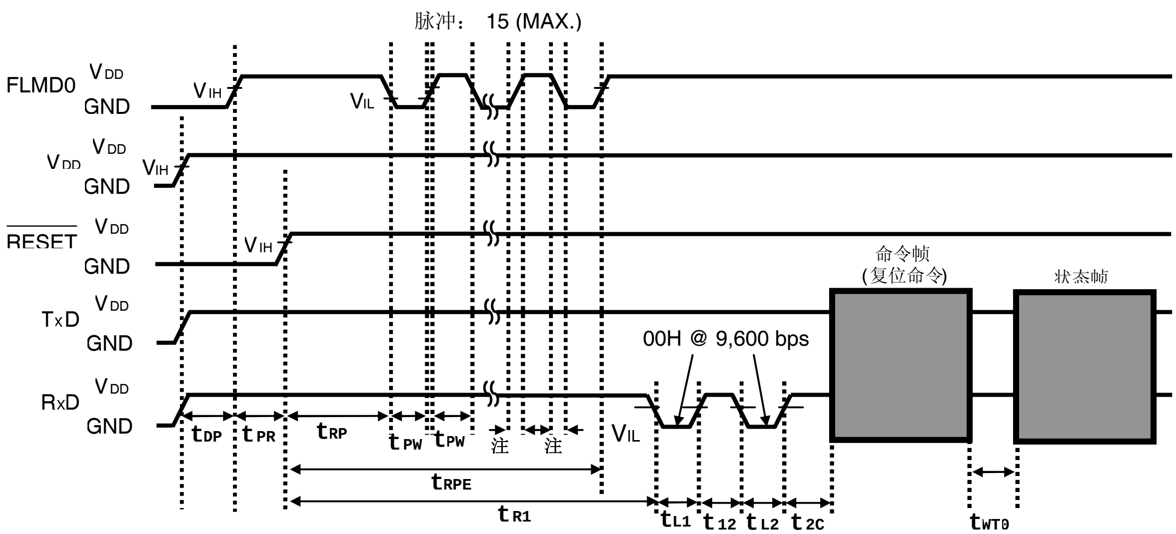
备注　等待时间定义如下：
<t_{WT0} 至 t_{WT19}>
V850ES/Kx2 在最大和最小时间之间完成命令处理。
编程器必须在最大时间逝去前重复进行状态检查。

9.3 UART 通信模式

• 数据帧

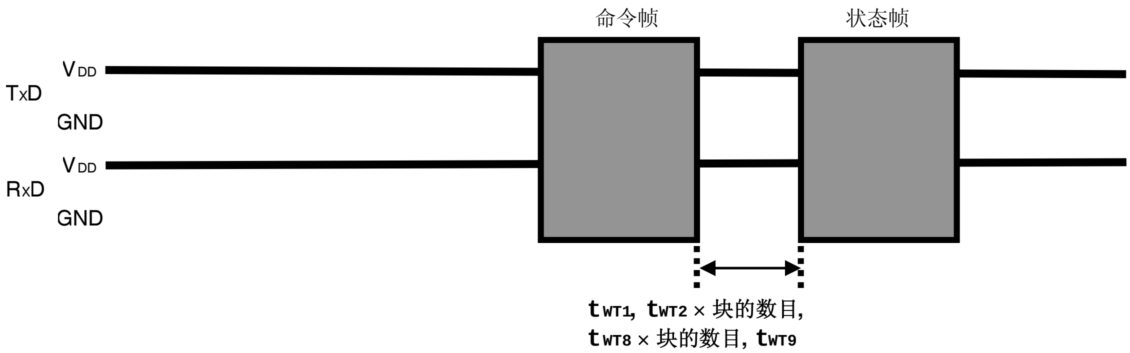


• 编程模式设置/复位命令

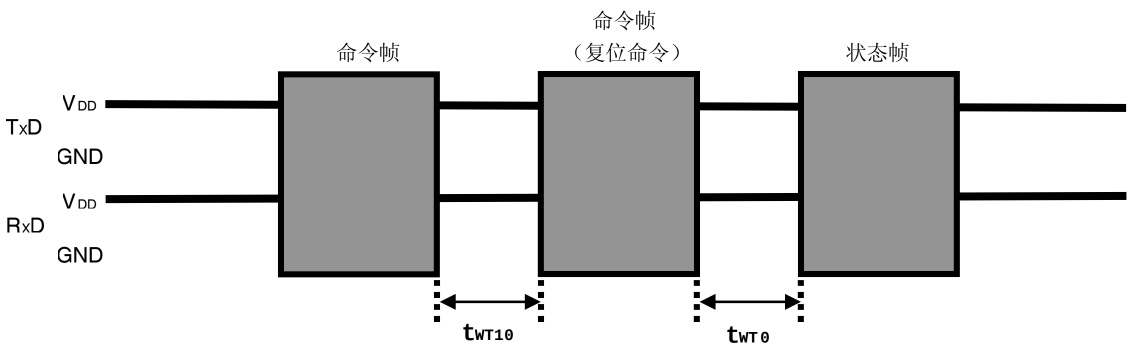


注　FLMD0 计数器上升/下降时间

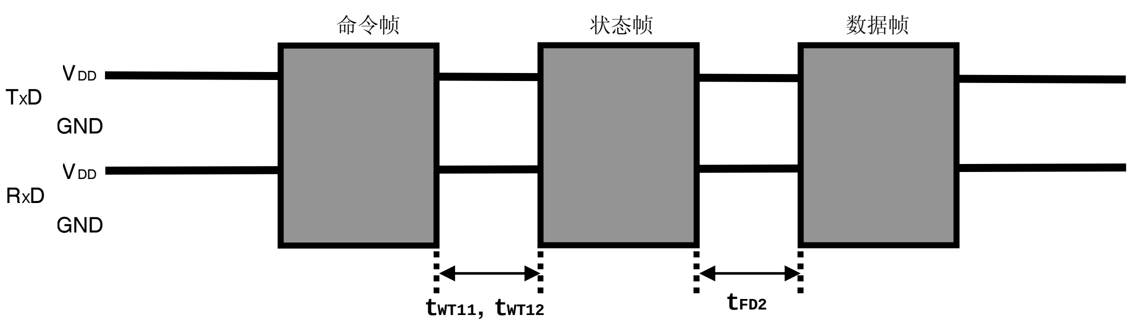
• 片擦除命令/块擦除命令/块空白检查命令/振荡频率设置命令



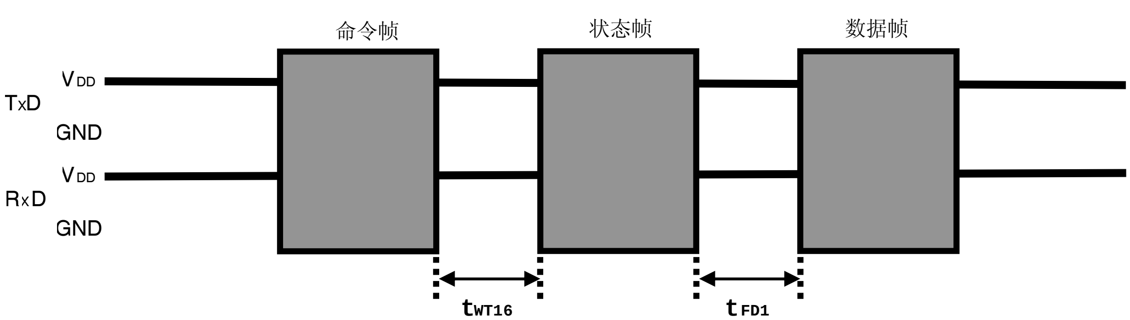
• 波特率设置命令



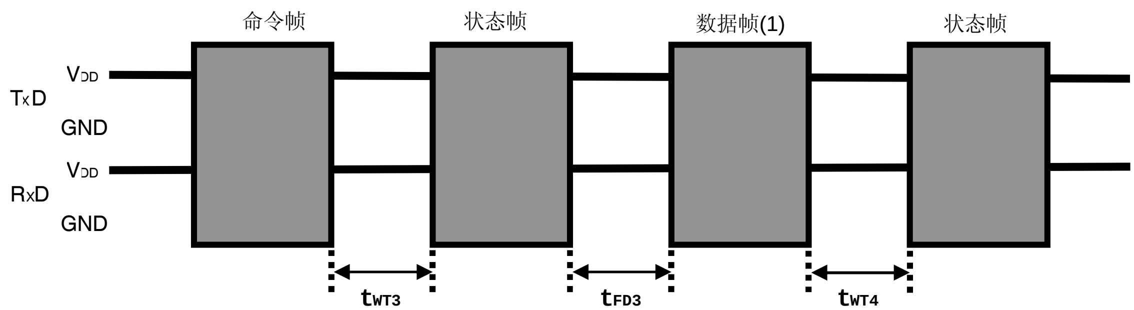
• ‘硅标记’命令/版本获取命令

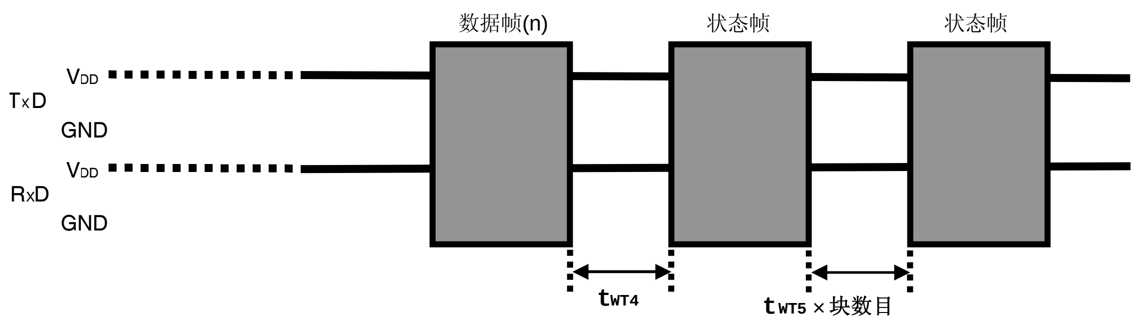


• 校验和命令

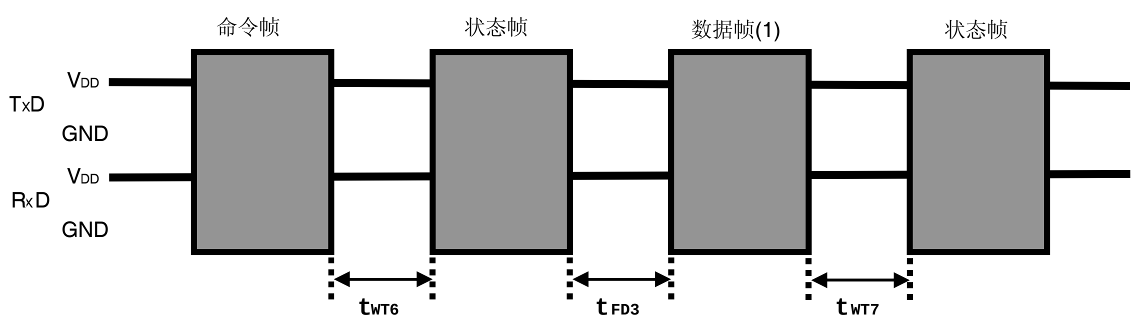


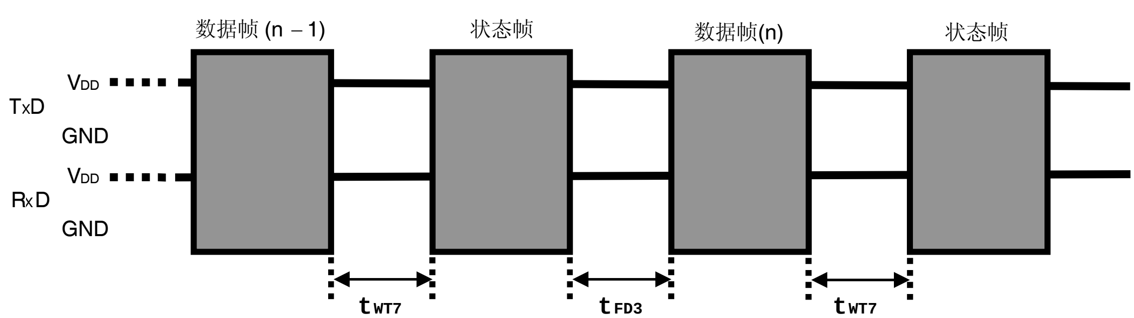
• 编程命令


命令帧
状态帧
数据帧(1)
状态帧
VDD
TxD
GND
VDD
RxD
GND
tWT3
tFD3
tWT4

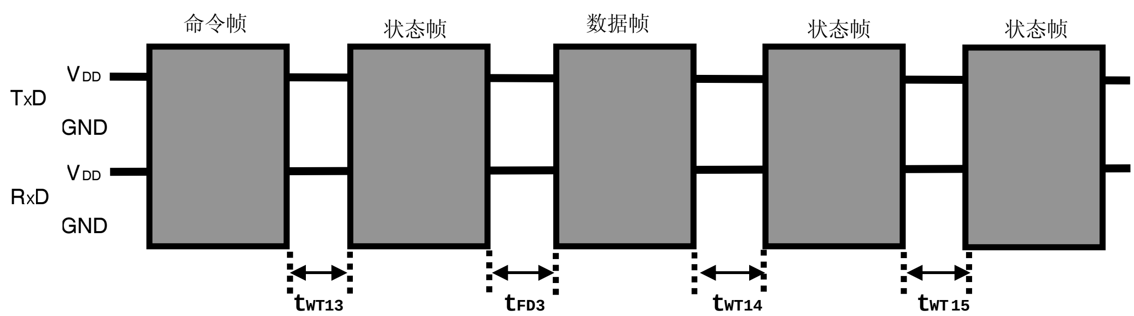

数据帧(n)
状态帧
状态帧
VDD
TxD
GND
VDD
RxD
GND
tWT4
tWT5 × 块数目

• 验证命令

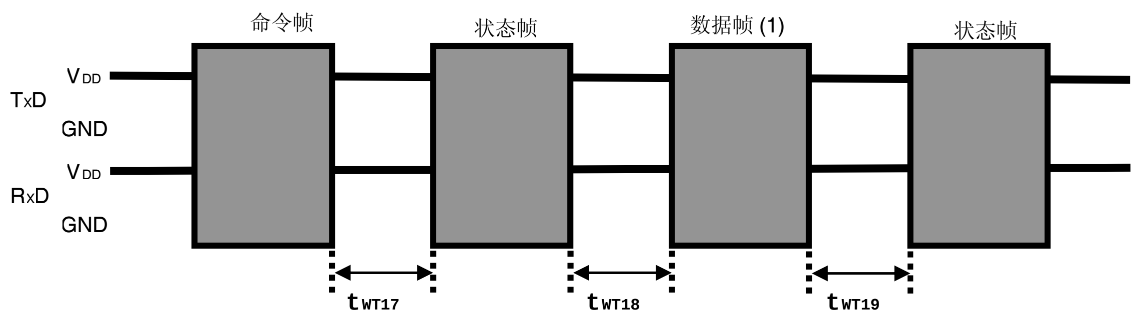

命令帧
状态帧
数据帧(1)
状态帧
VDD
TxD
GND
VDD
RxD
GND
tWT6
tFD3
tWT7

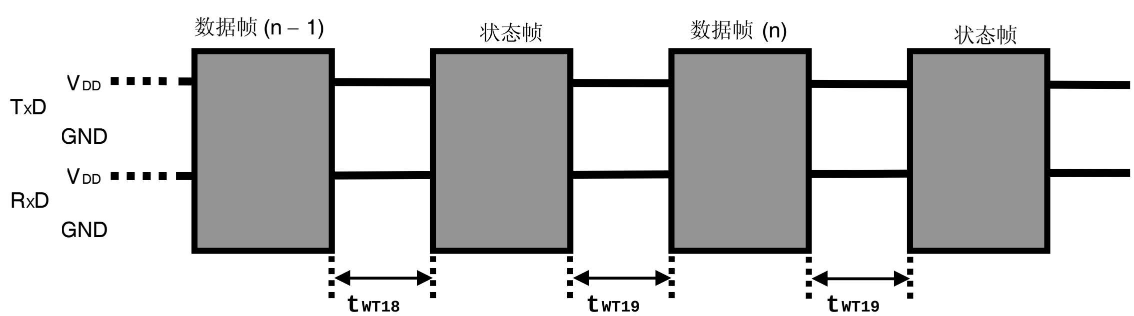

数据帧 (n – 1)
状态帧
数据帧(n)
状态帧
VDD
TxD
GND
VDD
RxD
GND
tWT7
tFD3
tWT7

• 安全设置命令

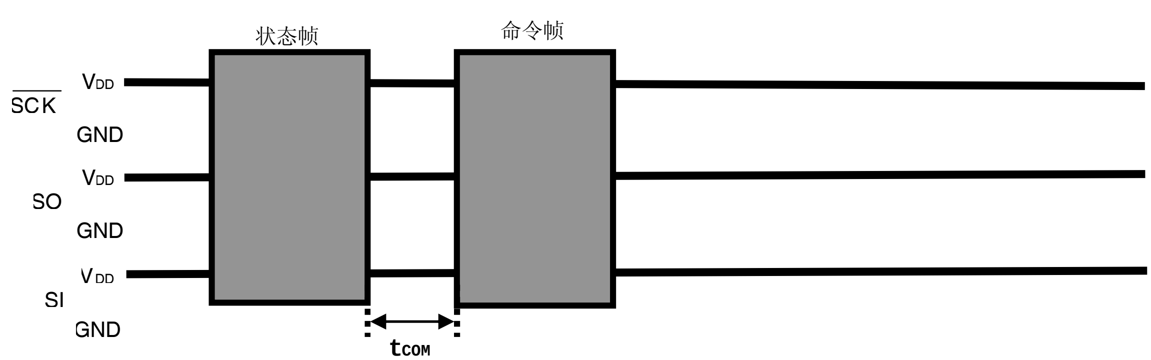


• 读取命令



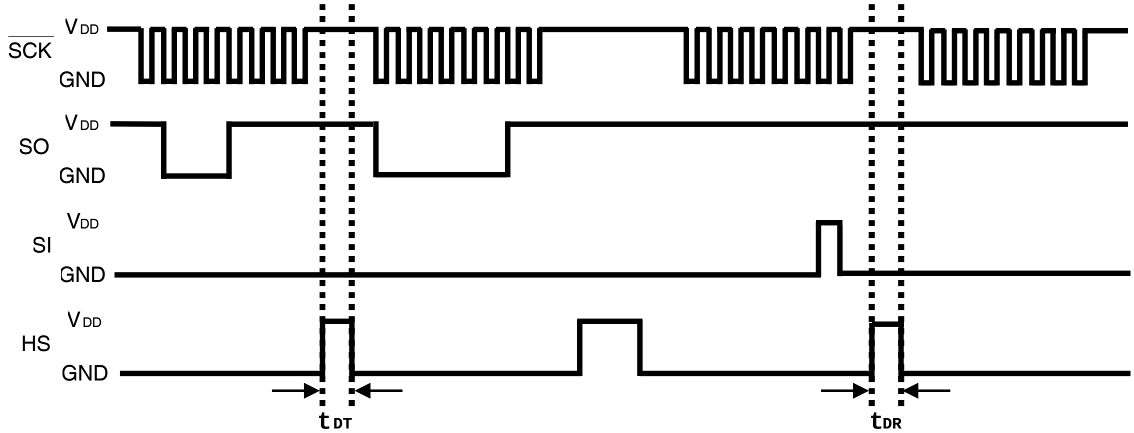


• 在命令帧发送之前等待

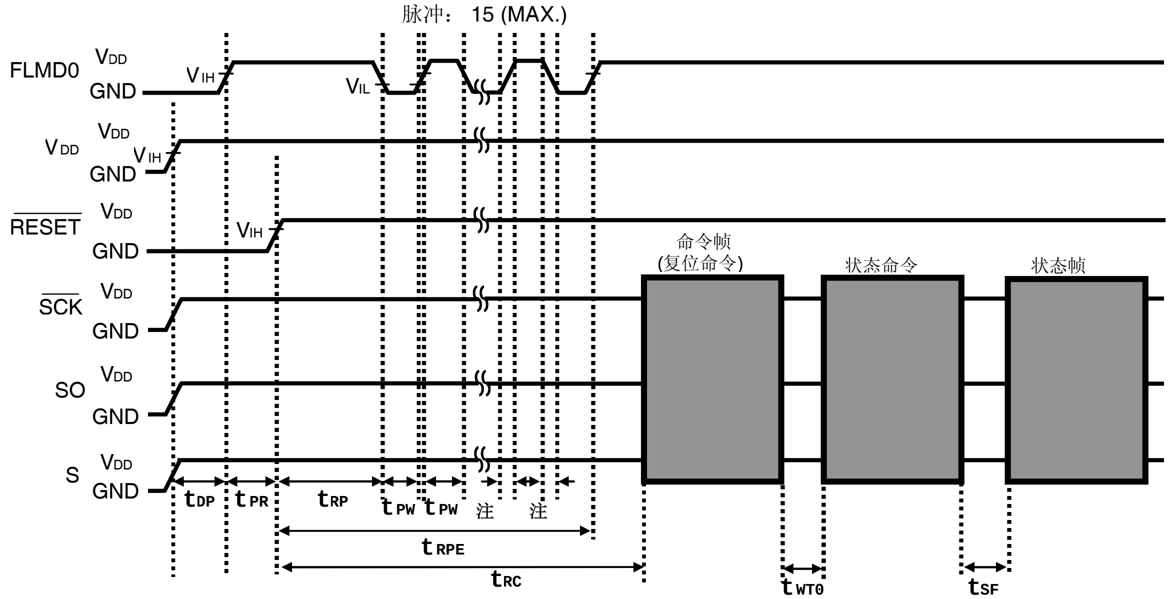


9.4 3 线串行 I/O 通信模式

• 数据帧

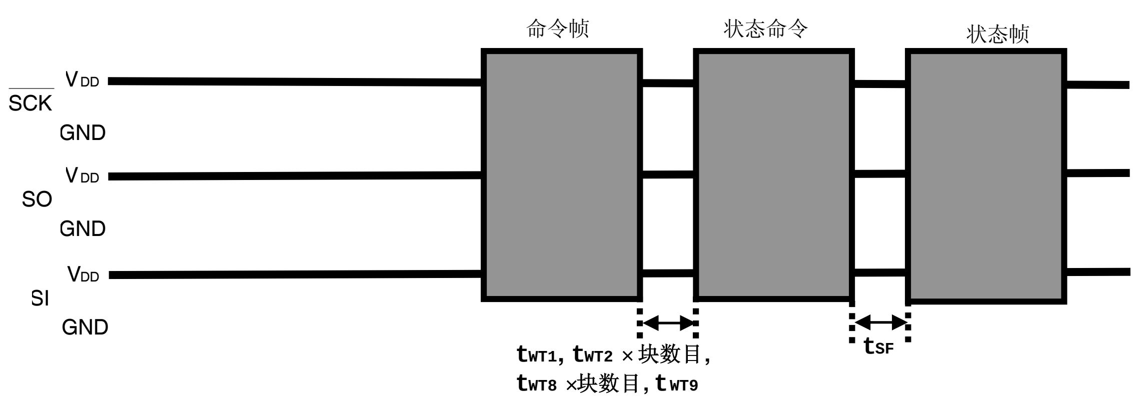


• 编程模式设置/复位命令

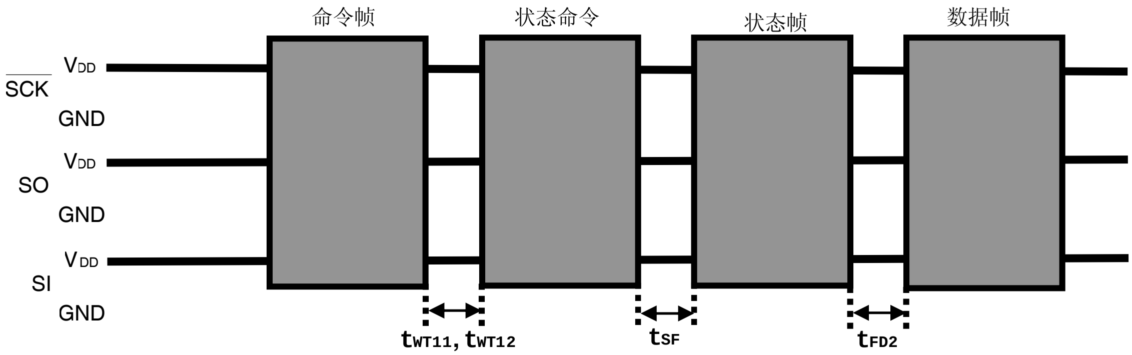


注 FLMD0 计数脉冲上升/下降时间

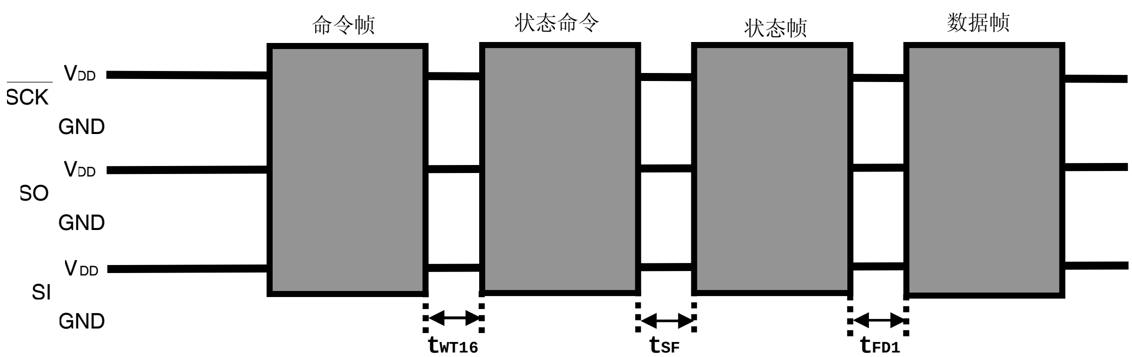
• 片擦除命令/块擦除命令/块空白检查命令/振荡频率设置命令



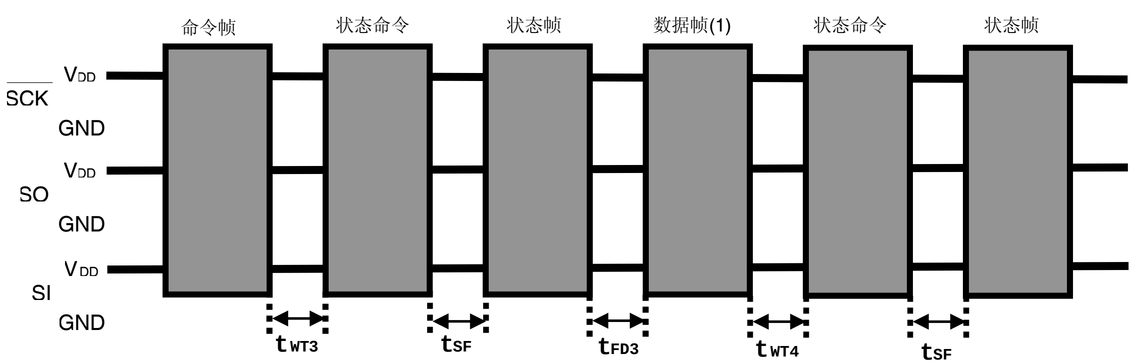
• ‘硅标记’命令/版本获取命令

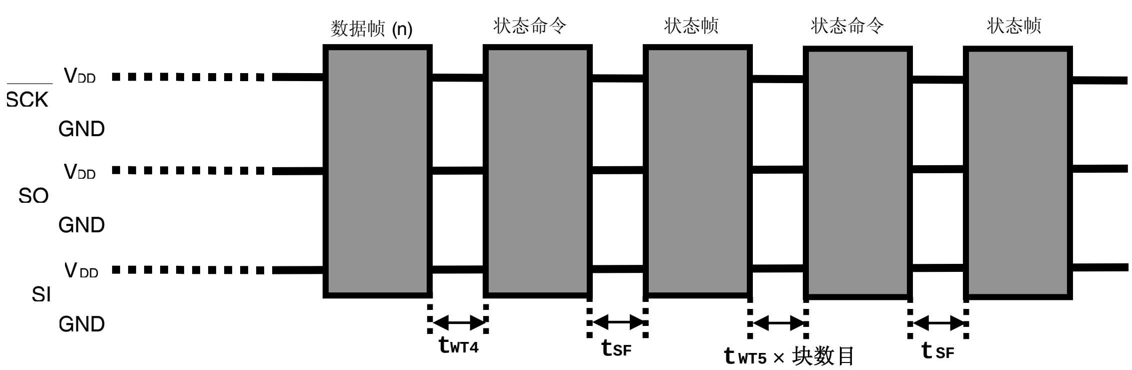


• 校验和命令

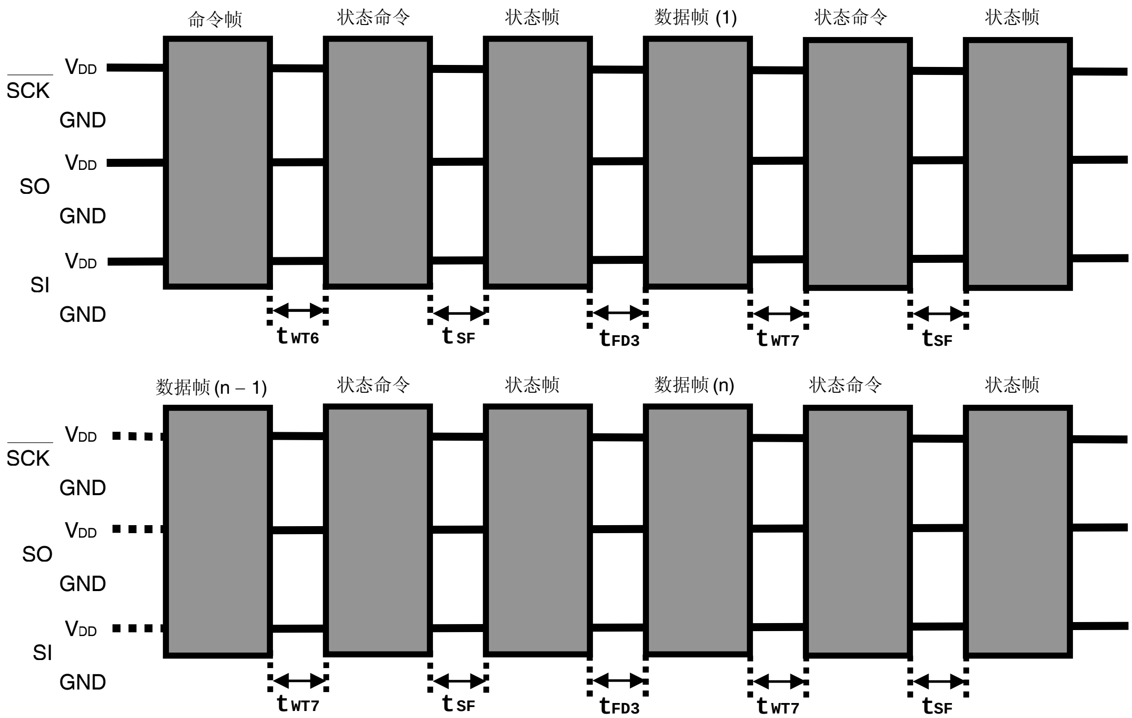


• 编程命令

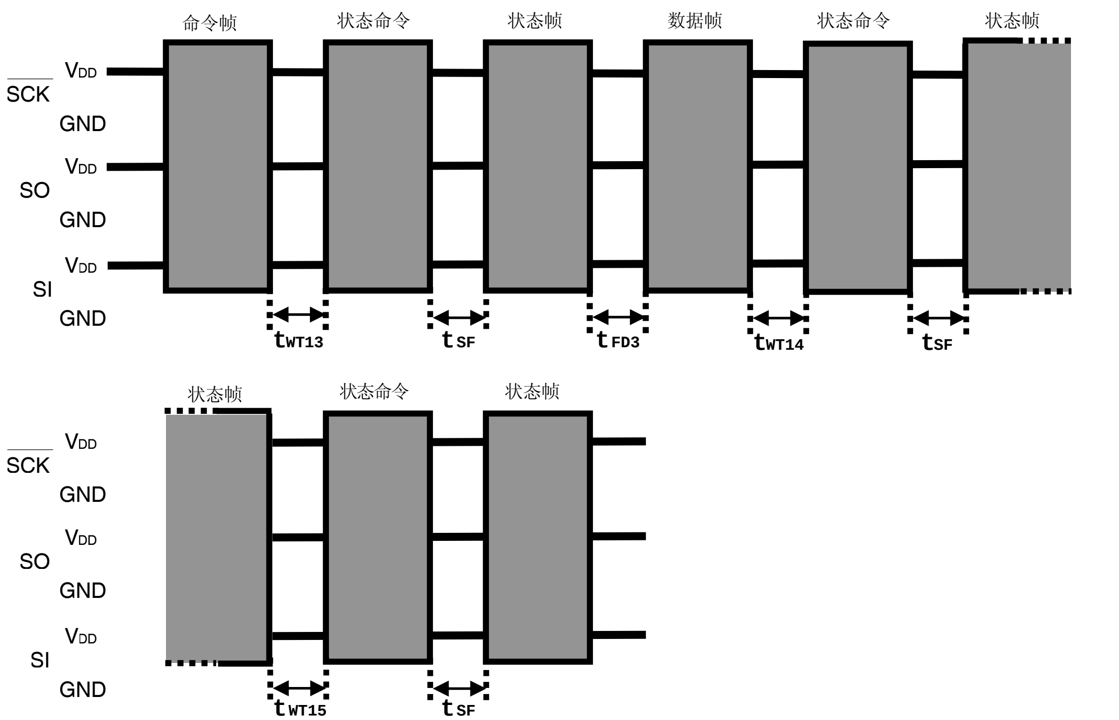




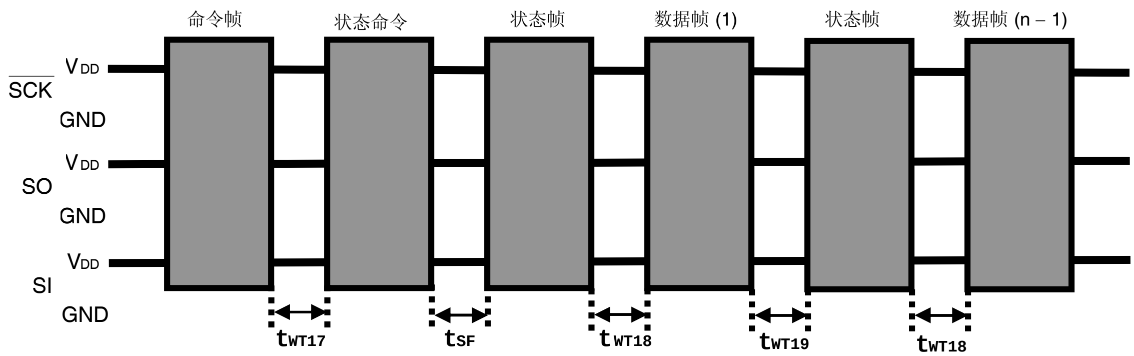
• 验证命令

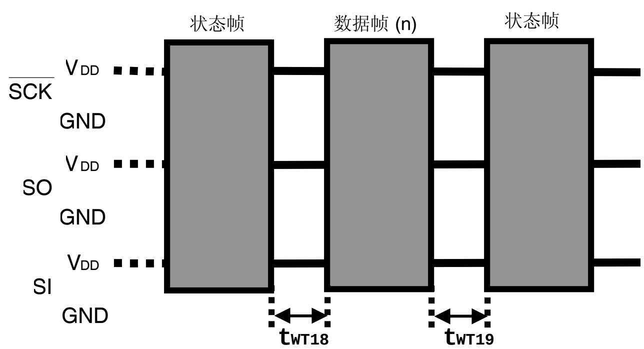

命令帧
状态命令
状态帧
数据帧 (1)
状态命令
状态帧
VDD
SCK
GND
VDD
SO
GND
VDD
SI
GND
tWT6
tSF
tFD3
tWT7
tSF
数据帧 (n – 1)
状态命令
状态帧
数据帧 (n)
状态命令
状态帧
tWT7
tSF
tFD3
tWT7
tSF

• 安全设置命令

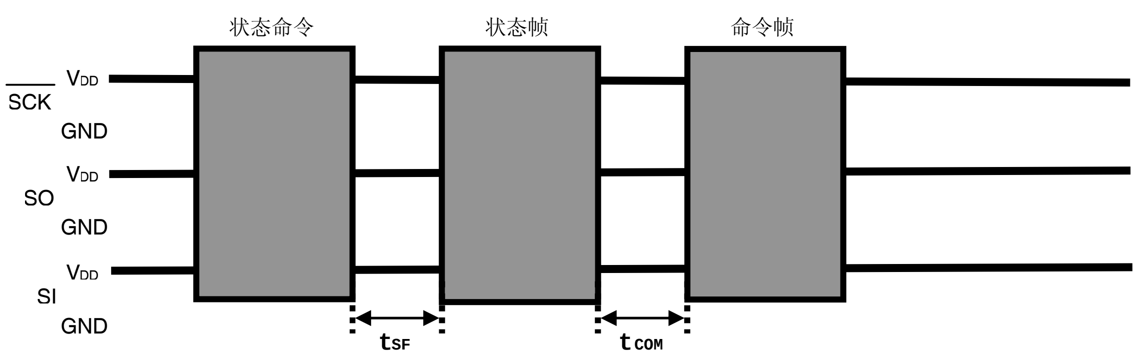

命令帧
状态命令
状态帧
数据帧
状态命令
状态帧
VDD
SCK
GND
VDD
SO
GND
VDD
SI
GND
tWT13
tSF
tFD3
tWT14
tSF
状态帧
状态命令
状态帧
tWT15
tSF

• 读取命令





• 在命令帧发送之前等待

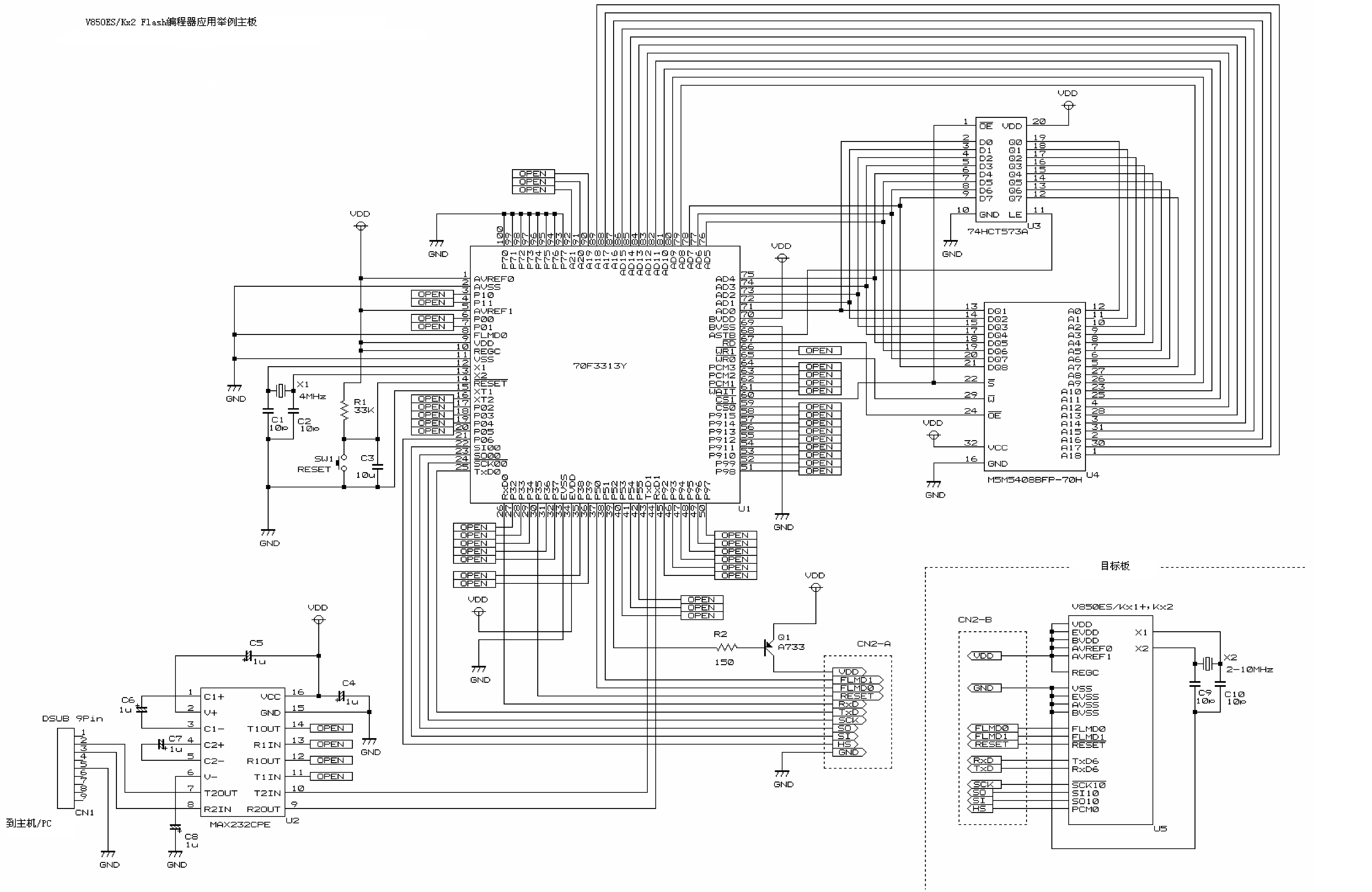


[备忘录]

附录 A 电路图（仅供参考）

图 A-1 显示了编程器和 V850ES/Kx2 的电路图，仅供参考。

图 **A-1.** 编程器和 **V850ES/Kx2** 的参考电路图



<R>

附录 B 修订历史

B.1 本版本主要修订之处

页码	说明
U17885CA2V0AN00 → U17885CA2V1AN00	
第 240 页	修正 **9.1 Flash 存储器编程模式设定时间**中的 t_{RP}， t_{RPE}， t_{RC} 和 t_{R1} 的最小值与最大值。
第 241 页	增加 “其它命令” 至 **9.2 编程特性**中的 t_{SF}。

详细信息请联系：

中国区

网址：
http://www.cn.necel.com/
http://www.necel.com/

[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