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

应用笔记

78K0/Kx2

8 位单片微控制器

Flash 存储器编程器 (编程器)

78K0/KB2:	μPD78F0500, 78F0501, 78F0502, 78F0503, 78F0503D, 78F0500A, 78F0501A, 78F0502A, 78F0503A, 78F0503DA
78K0/KC2:	μPD78F0511, 78F0512, 78F0513, 78F0514, 78F0515, 78F0513D, 78F0515D, 78F0511A, 78F0512A, 78F0513A, 78F0514A, 78F0515A, 78F0513DA, 78F0515DA
78K0/KD2:	μPD78F0521, 78F0522, 78F0523, 78F0524, 78F0525, 78F0526,78F0527, 78F0527D, 78F0521A, 78F0522A, 78F0523A, 78F0524A, 78F0525A, 78F0526A, 78F0527A, 78F0527DA
78K0/KE2:	μPD78F0531, 78F0532, 78F0533, 78F0534, 78F0535, 78F0536, 78F0537, 78F0537D, 78F0531A, 78F0532A, 78F0533A, 78F0534A, 78F0535A, 78F0536A, 78F0537A, 78F0537DA
78K0/KF2:	μPD78F0544, 78F0545, 78F0546, 78F0547, 78F0547D, 78F0544A, 78F0545A, 78F0546A, 78F0547A, 78F0547DA

文档编号. U17739CA3V0AN00 (第三版)
发行日期 2009 年 1 月 NS

[备注]

CMOS 设备注意事项

① 输入引脚处的电压波形

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

② 未使用的输入引脚的处理

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

③ ESD防护措施

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

④ 初始化之前的状态

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

⑤ 电源开关顺序

在一个设备的内部操作和外部接口使用不同的电源的情况下，按照规定，应先在接通内部电源之后再接通外部电源。当关闭电源时，按照规定，先关闭外部电源再关闭内部电源。如果电源开关顺序颠倒，可能会导致设备的内部组件过电压，产生异常电流，从而引起内部组件的误操作和性能的退化。
对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

⑥ 电源关闭状态下的输入信号

不要向没有加电的设备输入信号或提供I/O上拉电源。因为输入信号或提供I/O上拉电源将引起电流注入，从而引起设备的误操作，并产生异常电流，从而使内部组件退化。
每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断。

- **本文档所登载的内容有效期截止至 2009 年 01 月，信息先于产品的生产周期发布。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表或数据手册等相关资料以获取本公司产品的最新规格。**
- 并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。否则因本文档所登载内容引发的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权作出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：

“标准等级”、“专业等级”以及“特殊等级”三种质量等级。

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

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

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

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

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

（注）

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

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

M5 02.11-1

引言

读者

本手册适用于那些希望了解 78K0/Kx2 功能，并设计开发应用系统和程序的工程师。

目的

本手册旨在提供帮助用户理解怎样开发用于重写 78K0/Kx2 的 flash 存储器的 flash 编程器

本文档中的样例程序和电路图仅供参考，并非供实际设计所使用

因此，用户需自行承担使用这些样例的的风险。这些样例程序不保证能正确操作。

组织

以下是本手册包含的主要部分.

- Flash 存储器编程器
- 命令/数据帧格式
- 命令处理描述
- UART 通讯模式
- 3 线串行 I/O 通讯模式 (CSI)
- Flash 存储器编程参数特性

怎样阅读本手册

在阅读本手册前，读者应掌握电子工程、逻辑电路和微控制器等电子工程方面的基础知识。

- 要了解基本功能:
  → 请按目录顺序阅读本手册。标注 “<R>” 表示主要修订部分。修订部分可以很方便地通过在 PDF 文件中拷贝 “<R>” 并通过在 “查找:” 中指定来搜索查询。
- 要了解更多 78K0/Kx2 的硬件功能
  → 请参阅 78K0/Kx2 产品的各个用户手册

规则

数据规则：	数据的高位部分在左边，低位部分在右边
有效低电平表示法：	$\overline{\times\times\times}$ (在引脚和信号名称上加划一条线)
注：	文中用**注**标注的相关术语的脚注
注意事项：	需要特别关注的信息
备注：	补充信息
数值的表示：	二进制 ... ×××× 或 ××××B
	十进制 ... ××××
	十六进制 ... ××××H

相关文档　　本手册中提到的相关文档可能包括有初稿版本。但是，初稿版本没有特别注明。

设备相关文档

文档名称	文档编号
78K0/KB2 用户手册	U17328E
78K0/KC2 用户手册	U17336E
78K0/KD2 用户手册	U17312E
78K0/KE2 用户手册	U17260E
78K0/KF2 用户手册	U17397E
78K/0 系列指令用户手册	U12326E

注意事项　以上列出的相关文档可能会在无任何声明条件下修改。读者开发设计时，应该使用每个文档的最新版本。

目录

第 1 章 FLASH存储器编程

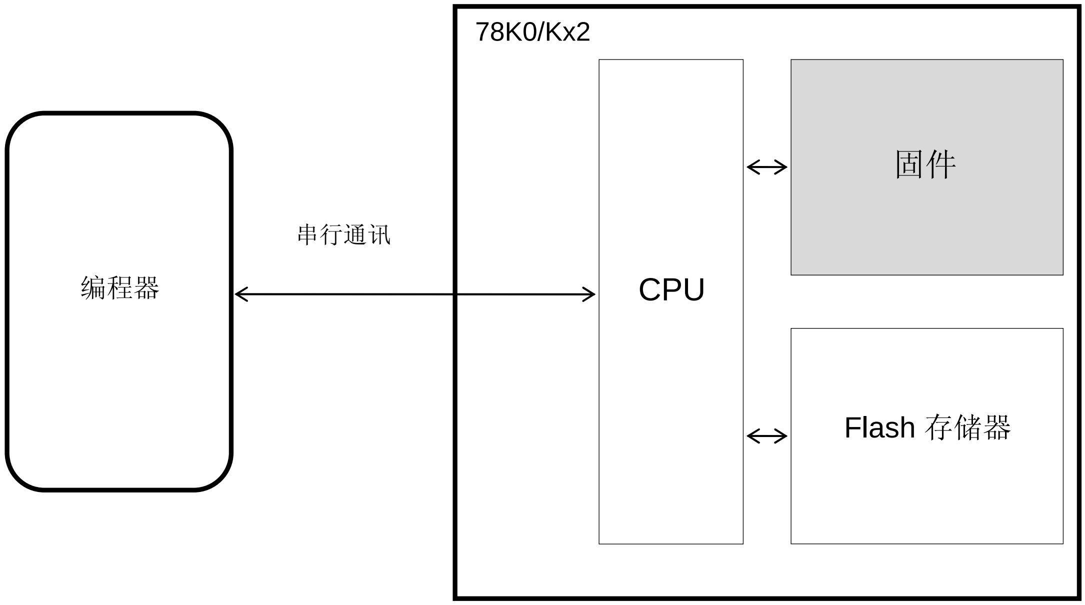
要重写78K0S/Kx2 内部Flash 存储器的内容，通常需要使用一个专用Flash 存储器编程器（以后我们用“编程器”来代替）。

本应用笔记将介绍如何开发专用的编程器。

1.1 概述

78K0/Kx2含有控制FLASH存储器编程的固件。通过编程器和78K0/Kx2 之间串口发送和接收通讯命令可以执行内部flash 存储器的编程。

图 1-1. 78K0/Kx2 Flash 存储器编程系统概要



1.2 系统配置

存储器编程的系统配置示例如图 1-2 所示。

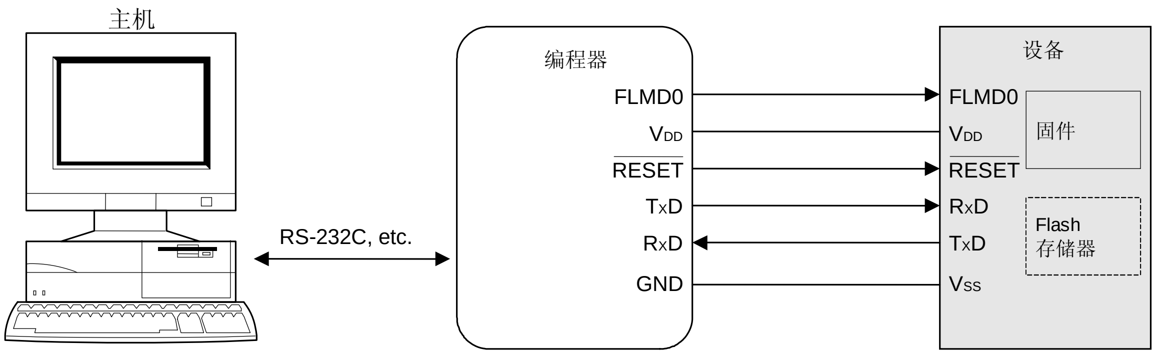
这些图例所示为主机控制下的编程器如何对 Flash 存储器编程。

根据编程器的连接方式，若事先已经将程序下载到编程器中，则编程器可以在无主机的独立模式下使用。

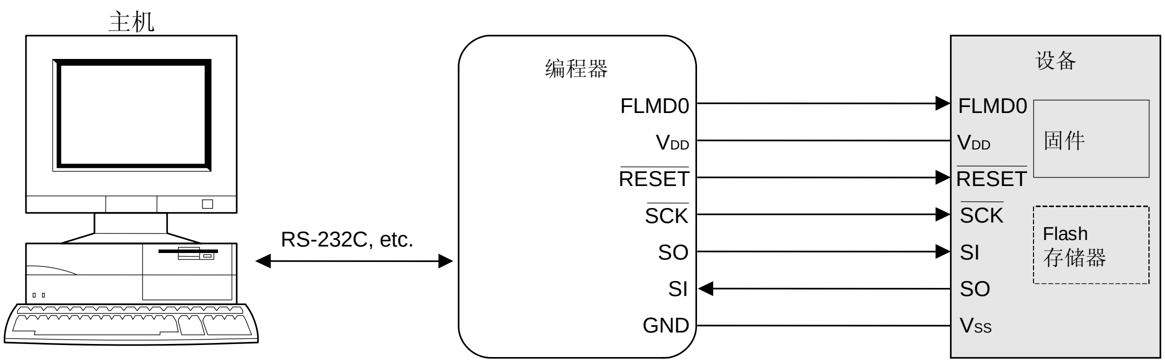
例如， NEC 电子的 Flash 存储器编程器 PG-FP4 可以在主机连接时使用 GUI 软件或独立的执行编程。

图 1-2.系统配置实例

(1) UART 通讯模式（**LSB** 优先传输）



(2) 三线串行输入**/**输出通讯模式（**CSI**）（**MSB** 优先传输）



备注 Flash 存储器编程所用的引脚及未用引脚的推荐连接，详见每个产品的用户手册。

1.3 Flash存储器配置

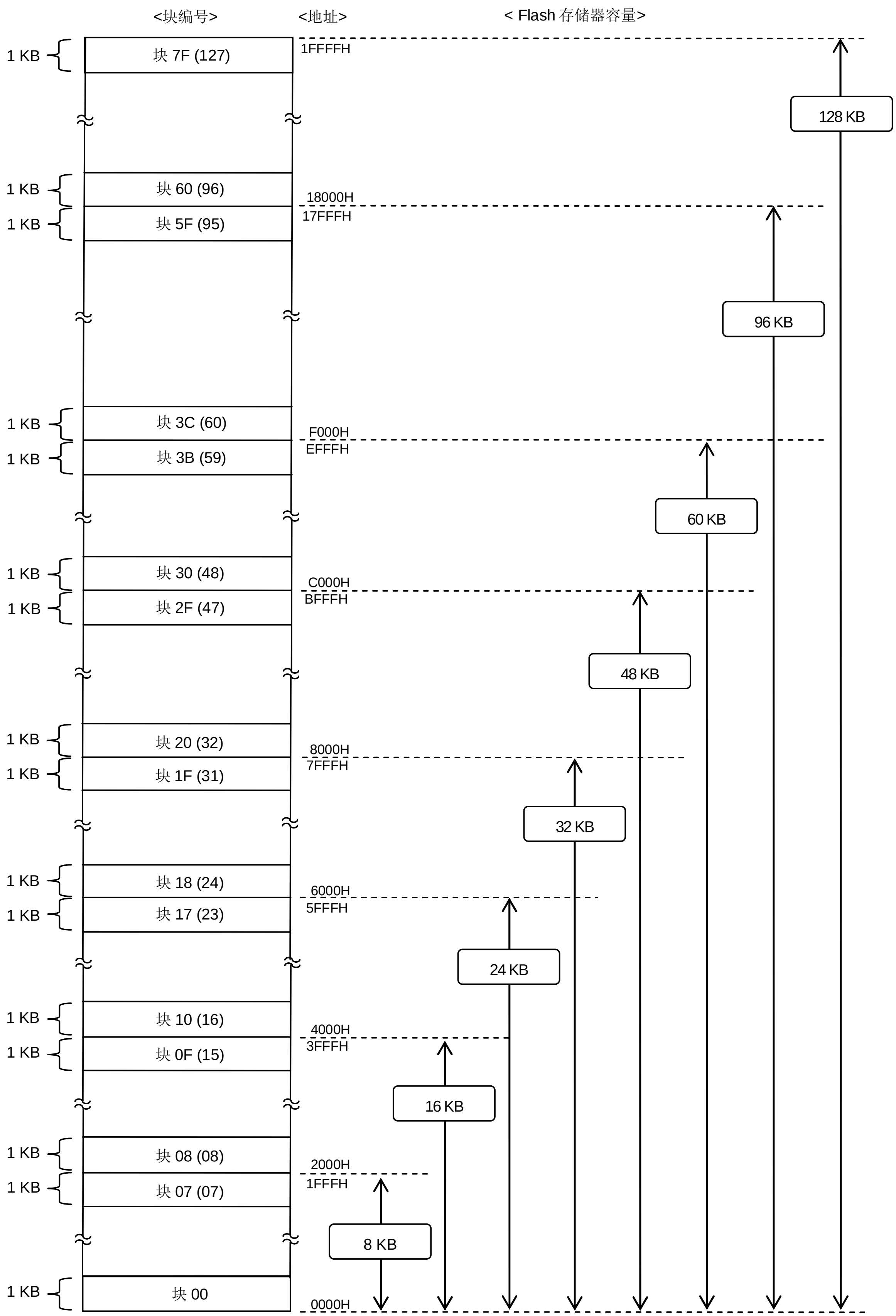
78K0/Kx2 必须管理产品的具体信息（例如设备名称与存储器信息）

表 1-1 所示为 78K0/Kx2 Flash 存储器容量，图 1-3 所示为 Flash 存储器结构。

表 1-1. 78K0/Kx2 Flash 存储器容量

设备名称		Flash 存储器容量
78K0/KB2	μPD78F0500, 78F0500A	8 KB
	μPD78F0501, 78F0501A	16 KB
	μPD78F0502, 78F0502A	24 KB
	μPD78F0503, 78F0503A, 78F0503D, 78F0503DA	32 KB
78K0/KC2	μPD78F0511, 78F0511A	16 KB
	μPD78F0512, 78F0512A	24 KB
	μPD78F0513, 78F0513A, 78F0513D, 78F0513DA	32 KB
	μPD78F0514, 78F0514A	48 KB
	μPD78F0515, 78F0515A, 78F0515D, 78F0515DA	60 KB
78K0/KD2	μPD78F0521, 78F0521A	16 KB
	μPD78F0522, 78F0522A	24 KB
	μPD78F0523, 78F0523A	32 KB
	μPD78F0524, 78F0524A	48 KB
	μPD78F0525, 78F0525A	60 KB
	μPD78F0526, 78F0526A	96 KB
	μPD78F0527, 78F0527A, 78F0527D, 78F0527DA	128 KB
78K0/KE2	μPD78F0531, 78F0531A	16 KB
	μPD78F0532, 78F0532A	24 KB
	μPD78F0533, 78F0533A	32 KB
	μPD78F0534, 78F0534A	48 KB
	μPD78F0535, 78F0535A	60 KB
	μPD78F0536, 78F0536A	96 KB
	μPD78F0537, 78F0537A, 78F0537D, 78F0537DA	128 KB
78K0/KF2	μPD78F0544, 78F0544A	48 KB
	μPD78F0545, 78F0545A	60 KB
	μPD78F0546, 78F0546A	96 KB
	μPD78F0547, 78F0547A, 78F0547D, 78F0547DA	128 KB

图 1-3 Flash 存储器结构



备注　　每个 block 由 1KB 组成（这个结构仅是 Flash 存储器中整个 block 的一部分举例）。

1.4 命令列表与状态列表

可以对组成 78K0/Kx2 的 Flash 存储器执行如表 1-2 所示的操作。编程器发送命令给 78K0/Kx2 来控制这些功能，并通过检查 78K0/Kx2 的响应状态来操作 Flash 存储器。

1.4.1 命令列表

下表所示为编程器使用的命令及其功能。

表 1-2. 编程器给 78K0/Kx2 传送的命令列表

命令数值	命令名称	功能名称	功能
20H	片擦除	擦除	擦除全部 Flash 存储器区域。
22H	块擦除		擦除 Flash 存储器的指定区域。
40H	编程	写入	在 Flash 存储器的指定区域写入数据。
13H	校验	校验	比较 Flash 存储器指定区域的内容与编程器发送的数据。
32H	块空白检查	空白检查	检查 Flash 存储器指定区域的擦除状态。
70H	状态	信息获取	获取当前的操作状态（状态数据）。
C0H	硅信号		获取 78K0/Kx2 信息（写入协议信息）。
C5H	获取版本		获取 78K0/Kx2 与固件的版本信息。
B0H	校验和		获取指定区域的校验和数据。
A0H	安全设置	安全	设置安全信息。
00H	复位	其他	检测同步通讯。
90H	设置振荡频率		指定 78K0/Kx2 的振荡频率。

1.4.2 状态列表

下表所示为编程器从 78K0/Kx2 接收到的状态码。

表 2-7. 状态码列表

状态码	状态	描述
04H	命令数值错误	接收到不支持的命令的错误返回
05H	参数错误	命令信息(参数)无效的错误返回
06H	(ACK)常规应答	常规应答
07H	校验和错误	编程器发送的帧数据反常时的错误返回
0FH	校验错误	编程器传送校验数据时校验错误返回
10H	保护错误	由于安全设置命令而禁止执行处理的错误返回
15H	(NACK)否定应答	否定应答
1AH	MRG10 错误	擦除校验错误
1BH	MRG11 错误	数据写入期间的内部校验错误或空白检测错误
1CH	写入错误	写入错误
20H	读取错误	读取安全信息失败时的错误返回
FFH	处理中（BUSY）	繁忙应答[注]

注　CSI 通讯期间，可以将 1 字节的“FFH”及“FFH”作为数据帧格式传送。

本手册将接收校验和错误或NACK视为非正常终止处理。当开发专用编程器时，引起校验和错误或NACK的命令发送前，该过程会一直执行直到没有错误发生。因此，推荐限制重查次数来防止无限次重复运行。

尽管上表并未列出，但若发生超时错误时(BUSY 超时或UART 通讯期间数据帧接收超时)，将关闭78K0/Kx2 供电电压并再次连接。(详情参见**1.6 关闭目标电源**)

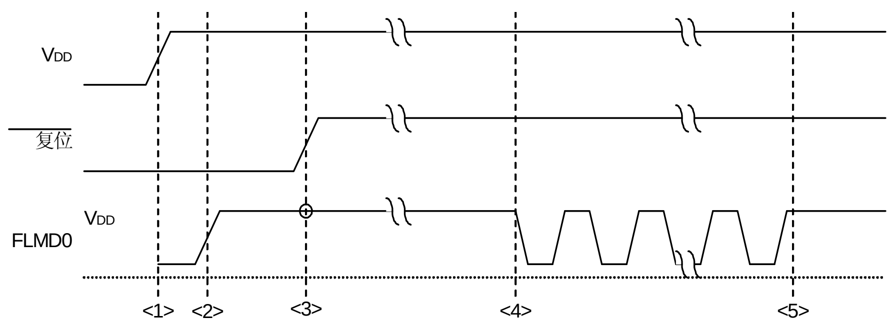
1.5 供电电压与设置Flash存储器编程模式

通过编程器重写Flash 存储器的内容，必须先将78K0/Kx2设置为Flash 存储器编程模式。要设置该模式，必须给Flash 存储器编程模式设置引脚(FLMD0)提供高电平，并复位。

在转到编程模式后，编程器接收到 FLMD0 引脚重写 Flash 存储器的脉冲输入。

以下举例说明设置 Flash 存储器编程模式与选择通讯模式的时序图。

图 1-4. 设置 Flash 存储器编程模式并选择通讯模式



<1>:供电电压(V_{DD})
<2>: FLMD0 =高电平
<3>:复位释放（模式设置）
<4>:脉冲输出开始
<5>:脉冲输出结束

下列表格所示为复位后 FLMD0 引脚设置与操作模式之间的关系。

表 1-4. 复位后 FLMD0 引脚设置与操作模式之间的关系。

FLMD0	操作模式
低(GND)	普通操作模式
高(V_{DD})	Flash 存储器编程模式

下列表格所示为通过 78K0/Kx2 选择的 FLMD0 脉冲个数（脉冲数）与通讯模式之间的关系。

表 1-5. FLMD0 脉冲个数与通讯模式之间的关系

通讯模式	FLMD0 脉冲个数	通讯使用端口
UART (UART6)	0（当使用 X1 时钟(f_X)时）	TxD6 (P13), RxD6 (P14)
	3（当使用外部主系统时钟(f_{EXCLK})时）	
三线串行 I/O(CSI10)	8	SO10 (P12), SI10 (P11), $\overline{SCK10}$ (P10)
禁止设置	其他	–

UART 通讯模式

UART 通讯使用 RxD 与 TxD 引脚。通讯条件如下。

UART 通讯条件

项目	描述
波特率	通讯一直以9,600 bps 执行，直到振荡频率设置命令已经传输。 状态帧接收后，通讯速率转换为115,200 bps。 此后，通讯速率固定为 115,200 bps。
校验位	无
数据长度	8 位（LSB 优先）
停止位	1 位

CSI 通讯时，编程器总是作为主设备运行，所以编程器必须检查78K0/Kx2 操作（例如写入或擦除）是否正常完成。另外, UART 通讯期间主从设备状态有时候互换，所以必须在最适宜的时间通讯。

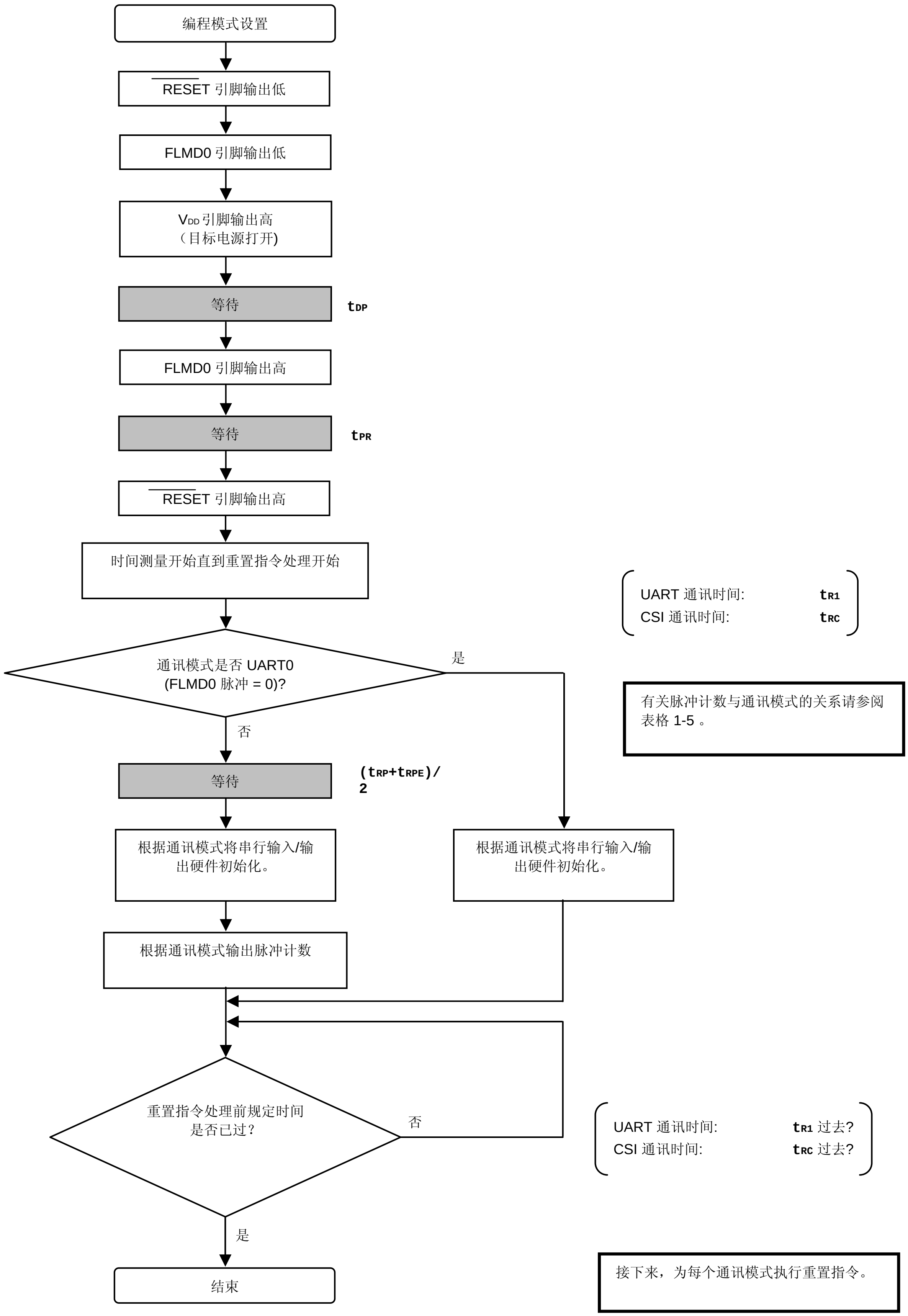
注意事项　执行 UART 通讯时，设置主从设备相同的波特率。

三线串行 **I/O** 通讯模式（**CSI**）

CSI 通讯使用SCK 、SO和SI 引脚。编程器始终为主设备，当78K0/Kx2没有准备好传送/接收时如果编程器经SCK引脚传输数据，有可能通讯不能正常执行。

通讯数据格式为 MSB 优先，8 位。保持时钟频率为 2.5 MHz 或更低。

1.5.1 模式设置流程图


编程模式设置
RESET 引脚输出低
FLMD0 引脚输出低
VDD 引脚输出高
（目标电源打开)
等待
tDP
FLMD0 引脚输出高
等待
tPR
RESET 引脚输出高
时间测量开始直到重置指令处理开始
UART 通讯时间: tR1
CSI 通讯时间: tRC
通讯模式是否 UART0
(FLMD0 脉冲 = 0)?
是
否
有关脉冲计数与通讯模式的关系请参阅表格 1-5 。
等待
(tRP+tRPE)/2
根据通讯模式将串行输入/输出硬件初始化。
根据通讯模式将串行输入/输出硬件初始化。
根据通讯模式输出脉冲计数
重置指令处理前规定时间是否已过?
否
UART 通讯时间: tR1 过去?
CSI 通讯时间: tRC 过去?
是
结束
接下来，为每个通讯模式执行重置指令。

1.5.2　示例程序

以下为模式设置处理示例程序

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

            int      n;
            int      pulse;

            SRMK0 = true;
            UARTE0 = false;

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

            }

            pFL_RES       = low;              // RESET = 低
            pmFL_FLMD0 = PM_OUT;              // FLMD0 = 输出模式
            pFL_FLMD0     = low;
            FL_VDD_HI(); // VDD = 高

            fl_wait(tDP);     // wait 等待

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

            pFL_RES       = hi;               // RESET = 高
            start_flto(fl_if == FLIF_CSI ? tRC : tR1);          //
                                                                  开始"tRC"等待计时
            fl_wait((tRP+tRPE)/2);

            if (fl_if == FLIF_UART){
              init_fl_uart();//
                  初始化UART硬件(控制Flash设备)
              UARTE0 = true;
              SRIF0 = false;
              SRMK0 = false;
            }
            else{
              init_fl_csi(); //初始化CSI硬件
            }

            for (n = 0; n < pulse; n++){          //脉冲输出
```

```
        pFL_FLMD0 = low;
        fl_wait(tPW);
        pFL_FLMD0 = hi;
        fl_wait(tPW);
      }
      while(!check_flto())              // tRC超时？
              ;                         // 否

      //开始执行RESET命令

}
```

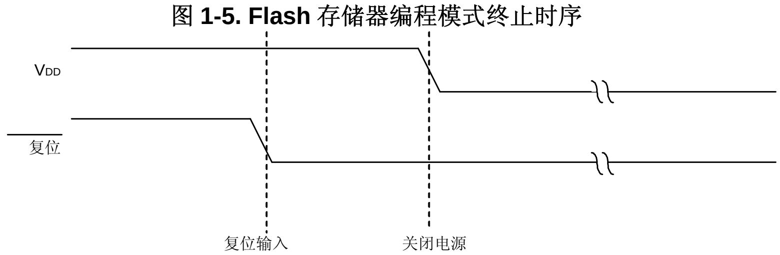
1.6 关闭目标板电源

每个命令执行完毕后，按以下所示设置复位 RESET 引脚至低后关闭目标板电源。

当关闭目标板电源时，将其他引脚设置为 Hi-Z。

注意事项 命令处理期间禁止关闭电源和输入复位信号。

图 1-5. Flash 存储器编程模式终止时序

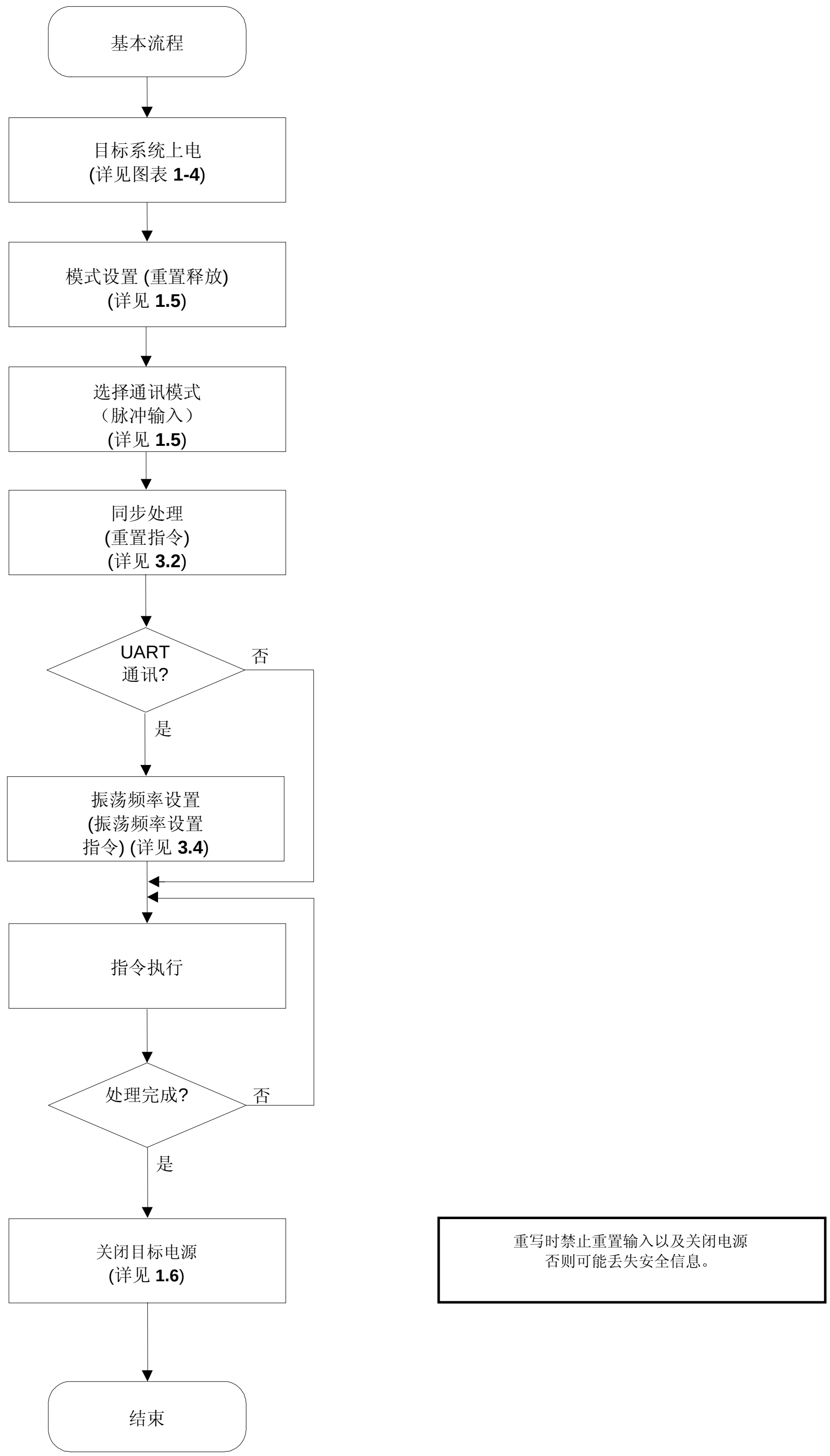


1.7 Flash存储器重写时命令执行流程

图 1-6 举例说明了当 Flash 存储器重写时编程器的基本流程。

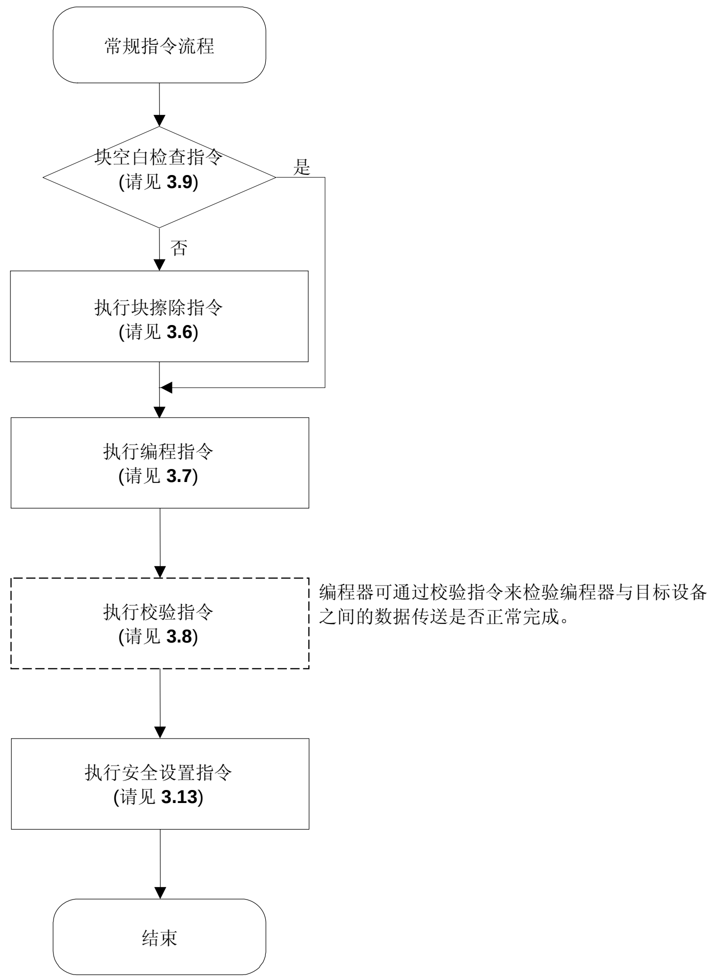
除图 1-6 显示的命令外，校验命令与校验和命令也能被支持。

图 1-6. Flash 存储器重写处理的基本流程



注　　图表 1-7 所示为每条命令的执行示例。

图表 1-7. Flash 存储器重写的常规命令执行流程



第 2 章 命令/数据帧格式

编程器使用命令帧来发送命令给 78K0/Kx2。78K0/Kx2 使用数据帧发送写入数据或校验数据给编程器。每帧都附加头、尾、数据长度、校验和以增加发送数据的可靠性。

以下所示为命令帧和数据帧的格式。

图 2-1. 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 2-2. 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 2-1. 每帧符号描述

符号	值	描述
SOH	01H	命令帧头
STX	02H	数据帧头
LEN	–	数据长度信息（00H 指示 256）。 命令帧： COM+命令信息长度 数据帧： 数据长度
COM	–	命令号
SUM	–	帧校验和数据，这些帧从最初值（00H）开始按照 1 个字节（忽略借位）顺序减去计算目标数据获得。计算目标如下。 命令帧：LEN+COM+全部命令信息 数据帧：LEN+全部数据
ETB	17H	倒数第二帧数据帧尾
ETX	03H	命令帧尾或数据帧尾

以下举例说明如何计算 1 帧的校验和(SUM)。

[命令帧]

下例中状态命令帧不包括命令信息，所以 LEN 和 COM 作为校验和计算目标。

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

此命令帧，校验和数据按如下所示。

00H (原始数值) – 01H (LEN) – 70H (COM) = 8FH (忽略借位。仅低 8 位。)

最后发送的命令帧如下。

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

[数据帧]

以下所示为发送的数据帧，LEN 和 D1~ D4 为校验和计算目标。

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

此数据帧，校验和数据按如下所示。

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

最后发送的数据帧如下。

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

当接收到数据帧时，校验和数据以同样的方式计算，并且通过判断获得值与接收数据中SUM 的值是否相同来检测校验和错误。例如，若接收到的数据帧如下，将检测到1个校验和错误。

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

↑ 若是正常的，应该为 1BH

2.1 命令帧发送处理

请参考以下章节关于每种通讯模式下发送命令帧，命令处理流程图的详细说明。

- UART 通讯模式，参考 **4.1 命令帧发送处理流程图**
- 3 线串行 I/O 通讯模式(CSI)，参考 **5.1 命令帧发送处理流程图图**

2.2 数据帧发送处理

数据帧(用户程序)、校验数据帧(用户程序)和安全数据帧(安全标志)按数据帧发送。

请参考以下章节关于每种通讯模式下发送数据帧，命令处理流程图的详细说明。

- UART 通讯模式，参考 **4.2 数据帧发送处理流程图**。
- 3 线串行 I/O 通讯模式(CSI)，参考 **5.2 数据帧发送处理流程图**

2.3 数据帧接收处理

状态帧、硅信号数据帧、版本数据帧，和校验和数据帧按数据帧接收。

请参考以下章节关于每种通讯模式下接收数据帧，命令处理流程图的详细说明。

- UART 通讯模式，参考 **4.3 数据帧接收处理流程图**
- 三线串行 I/O 通讯模式（CSI），参考 **5.3 数据帧接收处理流程图**

第 3 章 命令处理描述

3.1 状态命令

3.1.1 描述

此命令用来检查执行每条命令（例如写入或擦除）后 78K0/Kx2 的 操作状态

执行状态命令后，若由于通讯问题或类似问题，78K0/Kx2 不能正常接收状态命令帧，将不能执行状态设置。结果会接收到忙响应(FFH)，而不是状态帧。这样，将重发状态命令。

3.1.2 命令帧和状态帧

图 3-1 所示为状态命令的命令帧格式，图 3-2 所示为命令的状态帧。

图 3-1. 状态命令帧 (从编程器到 78K0/Kx2)

SOH	LEN	COM	SUM	ETX
01H	01H	70H（状态）	检查和	03H

图 3-2. 状态命令的状态帧 (从 78K0/Kx2 到编程器)

STX	LEN	Data （数据）			SUM	ETX
02H	n	ST1	…	STn	检查和	03H

备注 **1.** ST1 至 STn: 状态 #1 至 状态 #n

2. 根据发送给 78K0/Kx2 每条命令(例如写入或擦除)的不同而改变状态帧的长度。

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- UART 通讯模式不能使用状态命令。
- 三线串行 I/O 通讯模式（CSI），请参考 **5.4 状态命令**

注意事项 UART 通讯中每条命令（例如写入或擦除）发送后，78K0/Kx2 在指定时间内自动返回状态帧。因此不使用状态命令。

若在 UART 通讯中发送状态命令，将返回命令数错误。

3.2 复位命令

3.2.1 描述

在通讯模式设置后，此命令用来检查编程器与 78K0/Kx2 之间通讯是否建立。

当 78K0/Kx2 选择 UART 通讯模式时，编程器和 78K0/Kx2 必须设置相同的波特率。但是，78K0/Kx2 不能检测自己的波特率产生时钟频率（f_X or f_{EXCLK}），所以不能设置波特率。78K0/Kx2 波特率产生时钟频率可以通过从编程器以 9,600 bps 的速度发送两次“00H”来测得。测得“00H”的低电平宽度后，再计算两次发送信号的平均值。这样就可设置波特率，能够在通讯中同步侦测。

3.2.2 命令帧与状态帧

图 3-3 所示为复位命令的命令帧格式，图 3-4 所示为命令的状态帧。

图解 3-3. 复位命令帧 (从编程器到 78K0/Kx2)

SOH	LEN	COM	SUM	ETX
01H	01H	00H（复位）	检查和	03H

图 3-4. 复位命令的状态帧 (从 78K0/Kx2 到编程器)

STX	LEN	数据	SUM	ETX
02H	1	ST1	检查和	03H

备注 ST1：同步结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.4 复位命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.5 复位命令。**

3.3　波特率设置命令

78K0/Kx2 不支持波特率设置命令。

对于 78K0/Kx2，UART 通讯以 9,600 bps 的速度执行，直至发送振荡频率设置命令。在接收到状态帧后，通讯速率切换为 115,200 bps。之后，通讯速率固定为 115,200 bps。

3.4　振荡频率设置命令

3.4.1　描述

此命令用来指定 UART 通讯期间 f_X 或 f_{EXCLK}。

78K0/Kx2 通过接收包的频率数据来确定波特率为 115,200 bps。

注意事项　**78K0/Kx2 以 9,600 bps 的速度执行 UART 通讯，直到振荡频率设置命令传输。**
接收状态帧后，通讯速率转换为 115,200 bps。此后，通讯速率固定为 115,200 bps。

3.4.2　命令帧和状态帧

图 3-5 所示为振荡频率设置命令的命令帧格式，图 3-6 所示为命令的状态帧。

图 3-5. 振荡频率设置命令帧 (从编程器至 78K0/Kx2)

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

备注　D01 至 D04：

振荡频率 = $(D01\times0.1 + D02\times0.01 + D03\times0.001)\times10^{D04}$ (单位：kHz)

可以设置 10 kHz ~100 MHz，但是实际传输命令时，设置值按照每个设备的规格而定。D01~D03 由 BCDs 分开保存，D04 保存为 1 个带符号的整数。

设置举例：设置 6 MHz

D01 = 06H
D02 = 00H
D03 = 00H
D04 = 04H
振荡频率 = $6 \times 0.1 \times 10^4$ = 6,000 kHz = 6 MHz

设置举例：设置 10 MHz

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 3-6. 振荡频率设置命令的状态帧 (从 78K0/Kx2 至编程器)

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注　ST1: 震荡频率设置结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.5 振荡频率设置命令。**
- 关于三线串行 I/O 通讯模式 (CSI)，请参考 **5.6 振荡频率设置命令。**

3.5　片擦除命令

3.5.1　描述

此命令用来擦除 Flash 存储器的全部内容。另外，只要安全设置未禁止擦除(参见 **3.13 安全设置命令**)，片擦除处理可以初始化安全设置处理的全部设置信息。

3.5.2　命令帧与状态帧

图 3-7 所示为芯片擦除命令的命令帧格式，图 3-8 所示为命令的状态帧。

图 3-7. 片擦除命令帧（从编程器到 78K0/Kx2）

SOH	LEN	COM	SUM	ETX
01H	01H	20H （片擦除）	检查和	03H

图 3-8. 片擦除命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注　　ST1：片擦除结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.6 片擦除命令。**
- 关于三线串行 I/O 通讯模式，请参考 **5.7 芯片擦除命令。**

3.6　块擦除命令

3.6.1　描述

指定从擦除开始块的开始地址直至擦除结束块的结束地址的所有地址。可以指定多个连续块。

可是，只要擦除未被安全设置禁止，内容将不被擦除（参见 **3.13 安全设置命令**）。

3.6.2　命令帧与状态帧

图 3-9 所示为块擦除命令的命令帧格式，图 3-10 所示为命令的状态帧。

图 3-9. 块擦除命令帧 (从编程器到 78K0/Kx2)

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

备注 SAH, SAM, SAL: 块擦除开始地址 (任何块的开始地址)
SAH: 开始地址，高（23 至 16 位）（固定于 00H）
SAM: 开始地址，中（15 至 8 位）（固定于 00H）
SAL: 开始地址，低（7 至 0 位）（固定于 00H）
EAH, EAM, EAL: 块擦除结束地址（内部 Flash 存储器的最后地址）
EAH: 结束地址，高（23 至 16 位）
EAM: 结束地址，中（15 至 8 位）
EAL: 结束地址，低（7 至 0 位）

图 3-10. 块擦除命令的状态帧 （从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注　ST1：块擦除结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.7 块擦除命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.8 块擦除命令。**

3.7　编程命令

3.7.1　描述

此命令在写入开始地址和结束地址发送完成后,通过写入字节来发送数据。此命令将用户程序写入到 Flash 存储器并进行内部校验。

写入开始/结束地址仅能在块开始/结束地址单元中设置。

若最后数据传输后，两个状态帧（ST1 和 ST2）都显示 ACK，78K0/Kx2 固件自动执行内部校验。所以，内部校验需要进行状态代码确认。

3.7.2　命令帧与状态帧

图 3-11 所示为编程命令的命令帧格式，图 3-12 所示为命令的状态帧。

图 3-11. 编程命令帧（从编程器到 78K0/Kx2）

SOH	LEN	COM	通讯信息						SUM	ETX
01H	07H	40H （编程）	SAH	SAM	SAL	EAH	EAM	EAL	检查和	03H

备注　　SAH, SAM, SAL: 写入开始地址
　　　　EAH, EAM, EAL: 写入结束地址

图 3-12. 编程命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (a)	检查和	03H

备注　　ST1 (a)：命令接收结果

3.7.3　数据帧与状态帧

图 3-13 所示为数据写入的帧格式，图 3-14 所示为数据的状态帧。

图 3-13.　写入数据帧（从编程器到 78K0/Kx2）

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

备注　　写入数据：写入用户程序

图 3-14.　数据帧的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	检查和	03H

备注　　ST1 (b): 数据接收检查结果
　　　　ST2 (b): 写入结果

3.7.4 全部数据和状态帧传送完成

图 3-15 所示为发送完所有数据后的状态帧。

图 3-15. 传输完成全部数据后的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (c)	检查和	03H

备注 ST1 (c): 内部校验结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.8 编程命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.9 编程命令。**

3.8 校验命令

3.8.1 描述

此命令用来比较从指定地址范围内的编程器发送的数据和从 78K0/Kx2（读电平）读入的数据，并检查是否相配。

校验开始/结束地址只能在块开始/结束地址单元中设置。

3.8.2 命令帧与状态帧

图 3-16 所示为检验命令的命令帧格式，图 3-17 所示为命令的状态帧。

图 3-16. 校验命令帧（从编程器到 78K0/Kx2）

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

备注 SAH, SAM, SAL: 检验开始地址
EAH, EAM, EAL: 检验结束地址

图 3-17. 检验命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (a)	检查和	03H

备注 ST1 (a): 命令接收结果

3.8.3 数据帧与状态帧

图 3-18 所示为检验数据的框架格式，图 3-19 所示为数据的状态帧。

图 3-18. 检验数据的数据帧（从编程器到 78K0/Kx2）

STX	LEN	数据	SUM	ETX/ETB
02H	00H 至 FFH (00H = 256)	校验数据	检查和	03H/17H

备注 校验数据：将校验的用户程序

图 3-19. 数据帧的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	检查和	03H

备注　ST1 (b): 数据接收检验结果
　　　ST2 (b): 检验结果[注]

注　即使指定地址范围内有校验错误，校验结果 ACK 始终返回。所有校验错误的状态反映在最后数据的检验结果中。因此，只有在指定地址范围的所有校验处理完成后，才能检查出发生的校验错误。

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.9 检验命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.10 检验命令。**

3.9　块空白检测命令

3.9.1　描述

此命令用来检查 Flash 存储器内指定块的空白状态（擦除状态）。

指定从空白检测开始块的开始地址至空白检测结束块的结束地址。可以指定多个连续块。

3.9.2　命令帧与状态帧

图 3-20 所示为块空白检查命令的命令帧格式，图 3-21 所示为命令的状态帧。

图 3-20. 块空白检查命令帧（从编程器到 78K0/Kx2）

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

备注　SAH, SAM, SAL: 块空白检查开始地址（任何块的开始地址）
SAH: 开始地址，高（23 至 16 位）
SAM: 开始地址，中（15 至 8 位）
SAL: 开始地址，低（7 至 0 位）
EAH, EAM, EAL: 块空白检查结束地址（任何块的结束地址）
EAH: 结束地址，高（23 至 16 位）
EAM: 结束地址，中（15 至 8 位）
EAL: 结束地址，低（7 至 0 位）

图 3-21. 块空白检测命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注　ST1：块空白检查结果

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.10 块空白检验命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.11 块空白检验命令。**

3.10 硅信号命令

3.10.1 说明

此命令用来读取设备写入协议信息（硅信号）。

如果编程器支持 78K0/Kx2 不支持的编程协议，按照第二和第三字节的数值执行此命令并选择适当的协议。

3.10.2 命令帧与状态帧

图 3-22 所示为硅信号命令的命令帧格式，图 3-23 所示为命令的状态帧。

图 3-22. 硅信号命令帧（从编程器至 78K0/Kx2）

SOH	LEN	COM	SUM	ETX
01H	01H	C0H（硅信号）	检查和	03H

图 3-23. 硅信号命令的状态帧（从 78K0/Kx2 至编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注　　ST1: 命令接收结果

3.10.3 硅信号数据帧

图 3-24 所示为硅信号数据的帧格式。

图 3-24. 硅信号数据帧（从 78K0/Kx2 到编程器）

STX	LEN	Data								SUM	ETX
02H	N	VEN	MET	MSC	DEC	END	DEV	SCF	BOT	检查和	03H

备注　　**1.** n (LEN): 数据长度

VEN:　厂商码（NEC: 10H）

MET:　宏扩展码

MSC:　宏功能码

DEC:　设备扩展码

END:　内部 Flash 存储器的结束地址

DEV:　设备名称（μPDxx）

SCF:　安全标志信息

BOT:　引导块编号（固定为 03H）

2. 除引导块号码（BOT）的以上字段，低 7 位常用作数据，最高位用于奇偶校验。以下举例说明。

表 **3-1. 硅信号数据举例（以μPD78F0522 (78K0/KD2)为例)**

	字段	内容	长度（字节）	硅信号数据举例[注1]		实际值	奇偶校验
	VEN	厂商码（NEC）	1	10H	(00010000B)	10H	奇
	MET	扩展码 (固定于 78K0/Kx2)	1	7FH	(01111111B)	7FH	奇
	MSC	功能信息 (固定于 78K0/Kx2)	1	04H	(00000100B)	04H	奇
<R>	DEC	设备扩展码 (固定于 78K0/Kx2)	1	7CH	(01111100B)	07H	奇
	END	内部 Flash 存储器的结束地址（从低字节中提取）	3	7FH	(01111111B)	005FFFH	奇[注2]
				BFH	(11011111B)		
				01H	(00000001B)		
	DEV	设备名称	10	C4H	(11000100B = ‘D’)	‘D’	奇
				37H	(00110111B = ‘7’)	‘7’	
				38H	(00111000B = ‘8’)	‘8’	
				46H	(01000110B = ‘F’)	‘F’	
				B0H	(10110000B = ‘0’)	‘0’	
				B5H	(10110101B = ‘5’)	‘5’	
				32H	(00110010B = ‘2’)	‘2’	
				32H	(00110010B = ‘2’)	‘2’	
				20H	(00100000B = ‘ ’)	‘ ’	
				20H	(00100000B = ‘ ’)	‘ ’	
	SCF	安全标志信息	1	任意		任意	奇[注3]
<R>	BOT	引导块簇的最后块号（固定）	1	03H	(00000011B)	03H	未被加

注解 **1.** 0 与 1 为奇偶校验（调整字节中 “1”的个数作为奇值）

2. END 区域的奇偶校验计算按以下执行（当结束地址为 005FFFH）

<1> 从低位开始以 7 位为单位将 END 区域分开 (舍弃高 3 位)。

0 0 5 F F F

00000000 01011111 11111111

↓

~~000~~ 0000001 0111111 1111111

<2> 奇校验位加于最高位。

p0000001 p01111111 p1111111（p = 奇校验位）

= 00000001 10111111 01111111

= 01 BF 7F

<3> 高、中、与低字节的顺序按如下反转。

$\boxed{\text{7F BF 01}}$

以下所示为在 END 区域发送从微处理器得到的数据至实际地址的过程。

<1>　高、中、与低字节的顺序按如下反转。

7F BF 01
↓
01 BF 7F

<2>　检查在每个字节中“1”的个数为奇数（可在另一时间进行）。

<3>　去掉校验位并在最高位加 3 位 0。

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

<4>　将值以 8 位为一组进行发送。

000000000101111111111111
↓
00000000 01011111 11111111
↓
= 0 0 5 F F F

若 给 END 区域赋值“7F BF 01”，实际的最终地址为 005FFFH。

注　**3.　当使用安全设置命令来设置安全标志信息时，最高位固定为“1”。但是，若使用硅信号命令读取安全标志信息，最高位是奇校验。**

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.11 硅信号命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.12 硅信号命令。**

<R> **3.10.4 78K0/Kx2 硅信号列表**

表格 3-2. 78K0/Kx2 硅信号数据列表

项目	描述	长度（字节）	数据 （十六进制）
厂商码	NEC	1	10
扩展码	扩展码	1	7F
功能码	功能信息	1	04
设备信息	设备信息	1	7C
内部 Flash 存储器结束地址	(7 位数据 + 奇校验位) × 3	3	**注 1**
设备名称	78F0500, 78F0500A, 78F0501, 78F0501A, 78F0502, 78F0502A, 78F0503, 78F0503A, 78F0503D, 78F0503DA, 78F0511, 78F0511A, 78F0512, 78F0512A, 78F0513, 78F0513A, 78F0513D, 78F0513DA, 78F0514, 78F0514A, 78F0515, 78F0515A, 78F0515D, 78F0515DA, 78F0521, 78F0521A, 78F0522, 78F0522A, 78F0523, 78F0523A, 78F0524, 78F0524A, 78F0525, 78F0525A, 78F0526, 78F0526A, 78F0527, 78F0527A, 78F0527D, 78F0527DA, 78F0531, 78F0531A, 78F0532, 78F0532A, 78F0533, 78F0533A, 78F0534, 78F0534A, 78F0535, 78F0535A, 78F0536, 78F0536A, 78F0537, 78F0537A, 78F0537D, 78F0537DA, 78F0544, 78F0544A, 78F0545, 78F0545A, 78F0546, 78F0546A, 78F0547, 78F0547A, 78F0547D, 78F0547DA	10	**注 2**
安全信息	安全信息	1	任何值
引导块数	当前选定的引导簇的最后块号	1	03

备注 1.　内部 Flash 存储器结束地址列表

项目	描述	长度（字节）	数据（十六进制）
内部 Flash 存储器结束地址	8 KB (1FFFH)	3	7FBF80
	16 KB (3FFFH)		7F7F80
	24 KB (5FFFH)		7FBF01
	32 KB (7FFFH)		7F7F01
	48 KB (BFFFH)		7F7F02
	60 KB (EFFFH)		7FDF83
	96 KB (17FFFH)		7F7F85
	128 KB (1FFFFH)		7F7F07

（备注 2 列于下页）

备注 2. 设备名称如下表所列。

设备名称列表（1/4）

绰号	设备名称	长度（字节）	实际值 上行：地址码 下行：字符码									
78K0/KB2	D78F0500	10	C4	37	38	46	B0	B5	B0	B0	20	20
			D	7	8	F	0	5	0	0	-	-
	D78F0500A		C4	37	38	46	B0	B5	B0	B0	C1	20
			D	7	8	F	0	5	0	0	A	-
	D78F0501		C4	37	38	46	B0	B5	B0	31	20	20
			D	7	8	F	0	5	0	1	-	-
	D78F0501A		C4	37	38	46	B0	B5	B0	31	C1	20
			D	7	8	F	0	5	0	1	A	-
	D78F0502		C4	37	38	46	B0	B5	B0	32	20	20
			D	7	8	F	0	5	0	2	-	-
	D78F0502A		C4	37	38	46	B0	B5	B0	32	C1	20
			D	7	8	F	0	5	0	2	A	-
	D78F0503		C4	37	38	46	B0	B5	B0	B3	20	20
	D78F0503D		D	7	8	F	0	5	0	3	-	-
	D78F0503A		C4	37	38	46	B0	B5	B0	B3	C1	20
	D78F0503DA		D	7	8	F	0	5	0	3	A	-
78K0/KC2	D78F0511		C4	37	38	46	B0	B5	31	31	20	20
			D	7	8	F	0	5	1	1	-	-
	D78F0511A		C4	37	38	46	B0	B5	31	31	C1	20
			D	7	8	F	0	5	1	1	A	-
	D78F0512		C4	37	38	46	B0	B5	31	32	20	20
			D	7	8	F	0	5	1	2	-	-
	D78F0512A		C4	37	38	46	B0	B5	31	32	C1	20
			D	7	8	F	0	5	1	2	A	-
	D78F0513		C4	37	38	46	B0	B5	31	B3	20	20
	D78F0513D		D	7	8	F	0	5	1	3	-	-
	D78F0513A		C4	37	38	46	B0	B5	31	B3	C1	20
	D78F0513DA		D	7	8	F	0	5	1	3	A	-
	D78F0514		C4	37	38	46	B0	B5	31	34	20	20
			D	7	8	F	0	5	1	4	-	-
	D78F0514A		C4	37	38	46	B0	B5	31	34	C1	20
			D	7	8	F	0	5	1	4	A	-
	D78F0515		C4	37	38	46	B0	B5	31	B5	20	20
	D78F0515D		D	7	8	F	0	5	1	5	-	-
	D78F0515A		C4	37	38	46	B0	B5	31	B5	C1	20
	D78F0515DA		D	7	8	F	0	5	1	5	A	-

设备名称列表（2/4）

略称	设备名称	长度（字节）	实际数值（上行：地址码 下行：字符码）									
78K0/KD2	D78F0521	10	C4	37	38	46	B0	B5	32	31	20	20
			D	7	8	F	0	5	2	1	-	-
	D78F0521A		C4	37	38	46	B0	B5	32	31	C1	20
			D	7	8	F	0	5	2	1	A	-
	D78F0522		C4	37	38	46	B0	B5	32	32	20	20
			D	7	8	F	0	5	2	2	-	-
	D78F0522A		C4	37	38	46	B0	B5	32	32	C1	20
			D	7	8	F	0	5	2	2	A	-
	D78F0523		C4	37	38	46	B0	B5	32	B3	20	20
			D	7	8	F	0	5	2	3	-	-
	D78F0523A		C4	37	38	46	B0	B5	32	B3	C1	20
			D	7	8	F	0	5	2	3	A	-
	D78F0524		C4	37	38	46	B0	B5	32	34	20	20
			D	7	8	F	0	5	2	4	-	-
	D78F0524A		C4	37	38	46	B0	B5	32	34	C1	20
			D	7	8	F	0	5	2	4	A	-
	D78F0525		C4	37	38	46	B0	B5	32	B5	20	20
			D	7	8	F	0	5	2	5	-	-
	D78F0525A		C4	37	38	46	B0	B5	32	B5	C1	20
			D	7	8	F	0	5	2	5	A	-
	D78F0526		C4	37	38	46	B0	B5	32	B6	20	20
			D	7	8	F	0	5	2	6	-	-
	D78F0526A		C4	37	38	46	B0	B5	32	B6	C1	20
			D	7	8	F	0	5	2	6	A	-
	D78F0527		C4	37	38	46	B0	B5	32	37	20	20
	D78F0527D		D	7	8	F	0	5	2	7	-	-
	D78F0527A		C4	37	38	46	B0	B5	32	37	C1	20
	D78F0527DA		D	7	8	F	0	5	2	7	A	-

设备名称列表（3/4）

略称	设备名称	长度（字节）	实际数值（上行：地址码 下行：字符码）									
78K0/KE2	D78F0531	10	C4	37	38	46	B0	B5	B3	31	20	20
			D	7	8	F	0	5	3	1	-	-
	D78F0531A		C4	37	38	46	B0	B5	B3	31	C1	20
			D	7	8	F	0	5	3	1	A	-
	D78F0532		C4	37	38	46	B0	B5	B3	32	20	20
			D	7	8	F	0	5	3	2	-	-
	D78F0532A		C4	37	38	46	B0	B5	B3	32	C1	20
			D	7	8	F	0	5	3	2	A	-
	D78F0533		C4	37	38	46	B0	B5	B3	B3	20	20
			D	7	8	F	0	5	3	3	-	-
	D78F0533A		C4	37	38	46	B0	B5	B3	B3	C1	20
			D	7	8	F	0	5	3	3	A	-
	D78F0534		C4	37	38	46	B0	B5	B3	34	20	20
			D	7	8	F	0	5	3	4	-	-
	D78F0534A		C4	37	38	46	B0	B5	B3	34	C1	20
			D	7	8	F	0	5	3	4	A	-
	D78F0535		C4	37	38	46	B0	B5	B3	B5	20	20
			D	7	8	F	0	5	3	5	-	-
	D78F0535A		C4	37	38	46	B0	B5	B3	B5	C1	20
			D	7	8	F	0	5	3	5	A	-
	D78F0536		C4	37	38	46	B0	B5	B3	B6	20	20
			D	7	8	F	0	5	3	6	-	-
	D78F0536A		C4	37	38	46	B0	B5	B3	B6	C1	20
			D	7	8	F	0	5	3	6	A	-
	D78F0537		C4	37	38	46	B0	B5	B3	37	20	20
	D78F0537D		D	7	8	F	0	5	3	7	-	-
	D78F0537A		C4	37	38	46	B0	B5	B3	37	C1	20
	D78F0537DA		D	7	8	F	0	5	3	7	A	-

设备名称列表 (4/4)

略称	设备名称	长度（字节）	实际数值（上行：地址码；下行：字符码）									
78K0/KF2	D78F0544	10	C4	37	38	46	B0	B5	34	34	20	20
			D	7	8	F	0	5	4	4	-	-
	D78F0544A		C4	37	38	46	B0	B5	34	34	C1	20
			D	7	8	F	0	5	4	4	A	-
	D78F0545		C4	37	38	46	B0	B5	34	B5	20	20
			D	7	8	F	0	5	4	5	-	-
	D78F0545A		C4	37	38	46	B0	B5	34	B5	C1	20
			D	7	8	F	0	5	4	5	A	-
	D78F0546		C4	37	38	46	B0	B5	34	B6	20	20
			D	7	8	F	0	5	4	6	-	-
	D78F0546A		C4	37	38	46	B0	B5	34	B6	C1	20
			D	7	8	F	0	5	4	6	A	-
	D78F0547		C4	37	38	46	B0	B5	34	37	20	20
	D78F0547D		D	7	8	F	0	5	4	7	-	-
	D78F0547A		C4	37	38	46	B0	B5	34	37	C1	20
	D78F0547DA		D	7	8	F	0	5	4	7	A	-

3.11 版本获取命令

3.11.1 描述

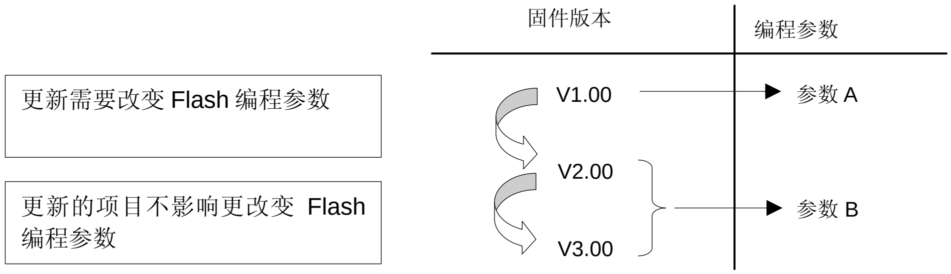
此命令用来获取 78K0/Kx2 设备版本和固件版本信息。

设备版本值固定为 00H。

根据 78K0/Kx2 固件版本必须更改编程参数时，使用此命令。

注意事项 固件版本可能更新，更新版本不会对 Flash 编程参数造成改变(此时，不公布更新的固件版本)。

举例 固件版本和重编程序参数



3.11.2 命令帧与状态帧

图 3-25 所示为版本获取命令的命令帧格式，图 3-26 所示为命令版本获取命令的状态帧。

图 3-25. 版本获取命令帧（从编程器到 78K0/Kx2）

SOH	LEN	COM	SUM	ETX
01H	01H	C5H(版本获取)	检查和	03H

图 3-26. 版本获取命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1	检查和	03H

备注 ST1: 命令接收结果

3.11.3 版本数据帧

图 3-27 所示为版本数据的数据帧。

图 3-27. 版本数据帧（从 **78K0/Kx2** 到编程器）

STX	LEN	数据						SUM	ETX
02H	06H	DV1	DV2	DV3	FV1	FV2	FV3	检查和	03H

备注　DV1: 设备版本整数 (固定为 00H)
DV2: 设备版本的第 1 小数位 (固定为 00H)
DV3: 设备版本的第 2 小数位(固定为 00H)
FV1: 固件版本整数
FV2: 固件版本的第 1 小数位
FV3: 固件版本的第 2 小数位

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.12 版本获取命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.13 版本获取命令。**

3.12 校验和命令

3.12.1 描述

此命令用来获取指定区域的校验和数据。

对于校验和计算的开始/结束地址，从 Flash 存储器开头指定块单元（1KB）的一个固定地址。

校验和数据通过指定地址从原始值（000H）以 1 字节为单位做累减获得。

3.12.2 命令帧与状态帧

图 3-28 所示为检查和命令的命令帧格式。图 3-29 所示为校验和命令的状态帧。

图 3-28. 校验和命令帧（从编程器到 **78K0/Kx2**）

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

备注　SAH, SAM, SAL: 校验和计算开始地址
EAH, EAM, EAL: 校验和计算结束地址

图 3-29. 校验和命令的状态帧（从 **78K0/Kx2** 到编程器）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	检查和	03H

备注　ST1: 命令接收结果

3.12.3 校验和数据帧

图 3-30 所示为包括校验和数据的帧格式。

图 3-30. 校验和数据帧（从 78K0/Kx2 到编程器）

STX	LEN	数据		SUM	ETX
02H	02H	CK1	CK2	检查和	03H

备注　　CK1: 检查和数据的高 8 位

　　　　CK2: 检查和数据的低 8 位

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.13 检查和命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.14 检查和命令。**

3.13 安全设置命令

3.13.1 描述

此命令用来进行安全设置（允许或禁止写入，块擦除，芯片擦除，与引导块簇重写）。用此命令进行设置，可限制未经许可者重写 Flash 存储器。

注意事项　**即使在安全设置后，可以执行从允许到禁止的更改设置，但不允许执行从禁止到允许的更改设置。若执行类似的设置，将产生保护错误(10H)。若必须这样设置，首先要先通过片擦除命令将所有的安全标记都初始化 (块擦除命令不能用来初始化安全标记)。**

若禁止片擦除或启动块重写，片擦除自身也被禁止，那么编程器不能擦除这些设置。根据编程器规范，推荐片擦除禁止前重新确认安全设置。

3.13.2 命令帧与状态帧

图 3-31 所示为安全设置命令的命令帧格式，图 3-32 所示为命令的状态帧。

安全设置命令帧包括块数与页数，但这些字段没有任何特别用处，所以固定设置为 00H。

图 3-31. 安全设置命令帧（从编程器到 78K0/Kx2）

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

图 3-32. 安全设置命令的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (a)	检查和	03H

备注　　ST1 (a): 命令接收结果

3.13.3 数据帧与状态帧

图 3-33 所示为安全数据帧的格式。图 3-34 所示为数据的状态帧。

图 3-33. 安全数据帧（从编程器到 78K0/Kx2）

STX	LEN	数据		SUM	ETX
02H	02H	FLG	BOT	检查和	03H

备注　FLG: 安全标记

BOT: 启动块簇的最后块数（固定为 03H）

图 3-34. 安全数据写入状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (b)	检查和	03H

备注　ST1 (b): 安全数据写入结果

3.13.4 内部校验检查和状态帧

图 3-35 所示为内部校验检查的状态帧。

图 3-35. 内部校验检查的状态帧（从 78K0/Kx2 到编程器）

STX	LEN	数据	SUM	ETX
02H	01H	ST1 (c)	检查和	03H

备注　ST1 (c): 内部校验结果

以下表格所示为安全标记的内容。

表格 3-3. 安全标记内容

项目	内容
Bit 7	固定为“1”
Bit 6	
Bit 5	
Bit 4	启动块簇重写禁止标志（1：允许启动块重写，0：禁止引导块重写）
Bit 3	固定为“1”
Bit 2	编程禁止标志（1：允许遍程，0：禁止编程）
Bit 1	块擦除禁止标志（1：允许块擦除，0：禁止块擦除）
Bit 0	片擦除禁止标志（1：允许片擦除，0：禁止片擦除）

以下表格所示为每个操作下安全标志字段设置和允许/禁止状态的关系。

表 3-4. 安全标志字段与每个操作的允许/禁止状态

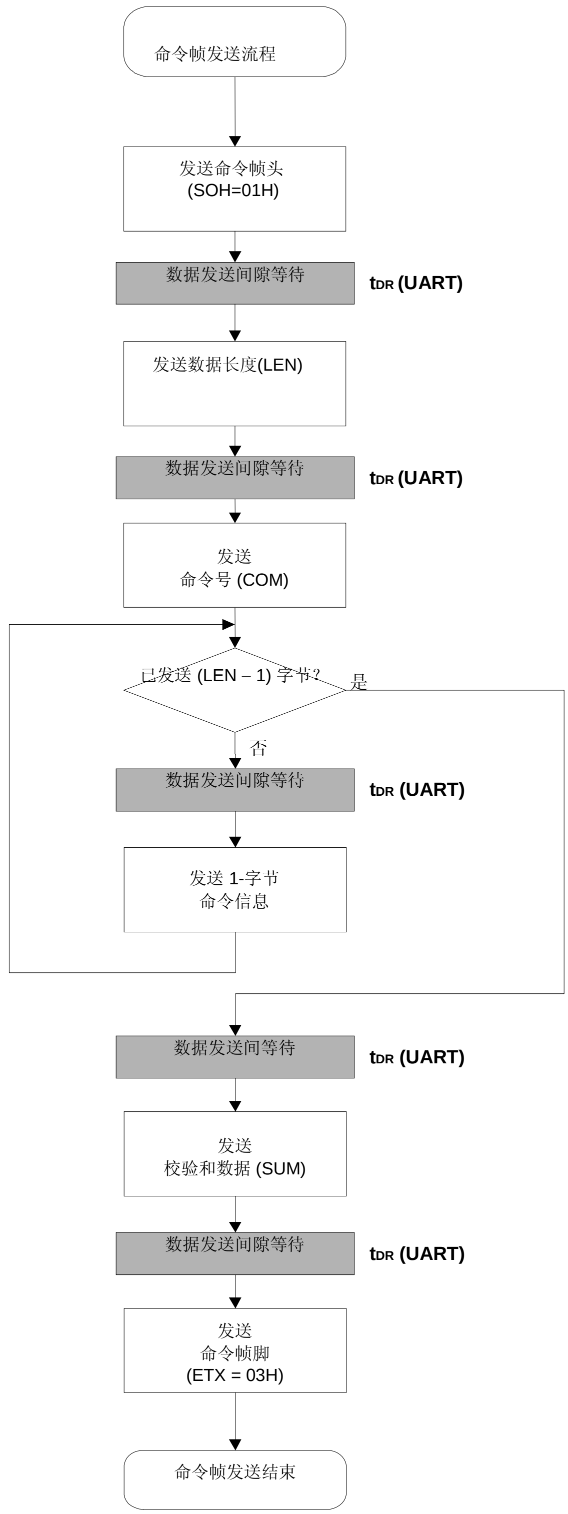
<table>
<tr><td>操作模式</td><td colspan="3">Flash 存储器编程模式</td><td>自编程模式</td></tr>
<tr><td rowspan="2">命令
安全设置项目</td><td colspan="3">安全设置后的命令操作
√:允许执行, 禁止执行
△: 无法在引导区域写入和块擦除
可以在除引导区域以外的区域写入或块擦除</td><td rowspan="5">• 不管安全设置数值，所有命令都可以执行
• 仅保持安全设置值</td></tr>
<tr><td>编程</td><td>片擦除</td><td>块擦除</td></tr>
<tr><td>禁止编程</td><td>×</td><td>√</td><td>×</td></tr>
<tr><td>禁止片擦除</td><td>√</td><td>×</td><td>×</td></tr>
<tr><td>禁止块擦除</td><td>√</td><td>√</td><td>×</td></tr>
<tr><td>禁止导入块重写标记</td><td>△</td><td>×</td><td>△</td><td>与 Flash 存储器编程模式条件相同（on-board/off-board 编程）</td></tr>
</table>

请参考以下章节关于编程器和 78K0/Kx2 间处理顺序流程图、命令处理流程图以及每种通讯模式下示例程序的详细资料。

- 关于 UART 通讯模式，请参考 **4.14 安全设置命令。**
- 关于三线串行 I/O 通讯模式（CSI），请参考 **5.15 安全设置命令。**

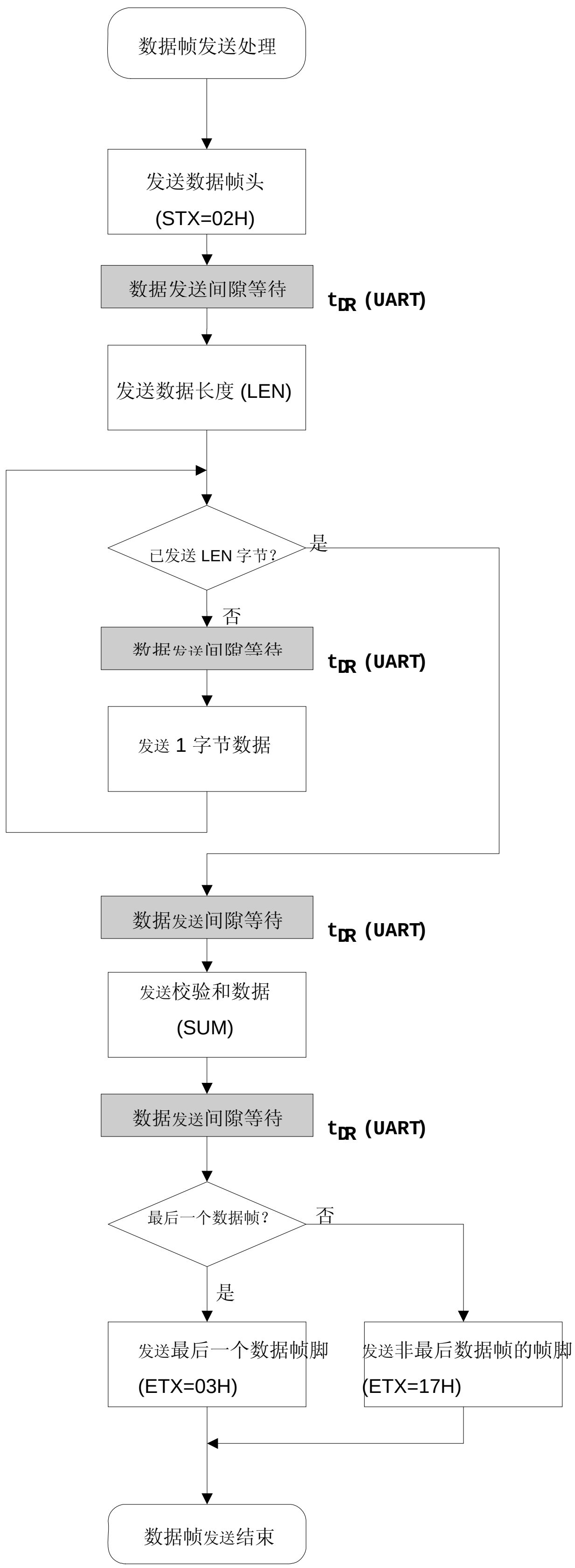
本章流程图中所示的每个符号（t_{xx} 与 t_{WTxx}）均为“**第 6 章 FLASH 存储器编程参数特性**”中项目的符号。
有关每个指定值的信息，请参阅“**第 6 章 FLASH 存储器编程参数特性**”。

<R> 4.1 命令帧发送处理流程图

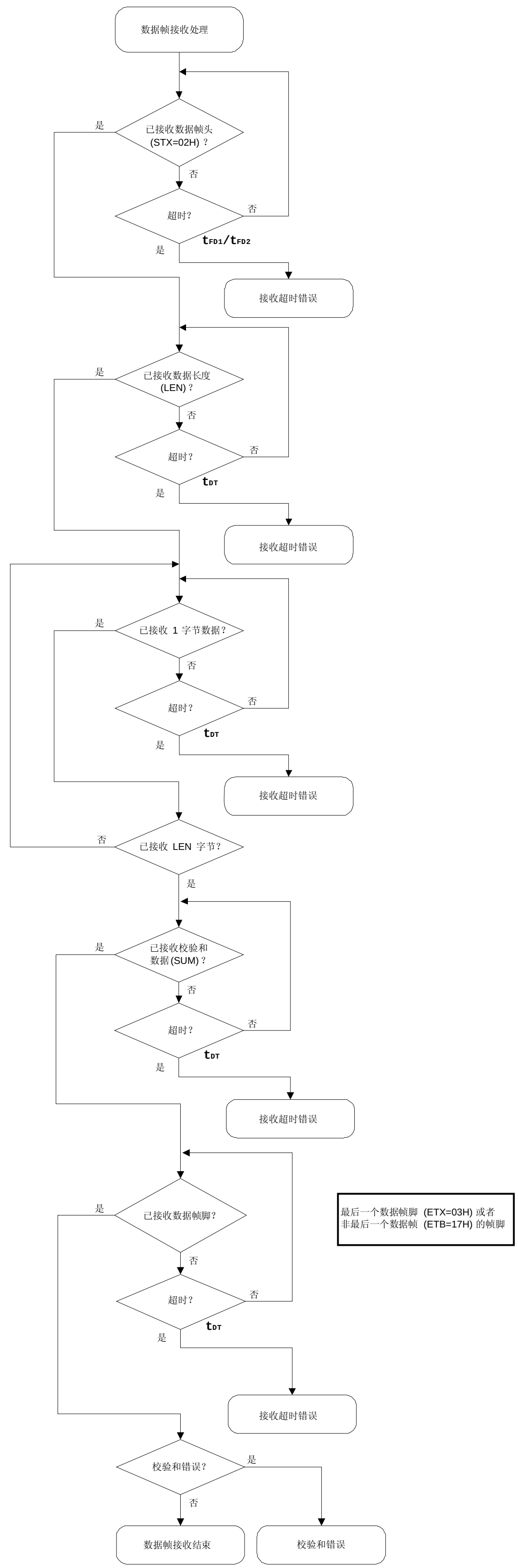


<R>

4.2 数据帧发送处理流程图



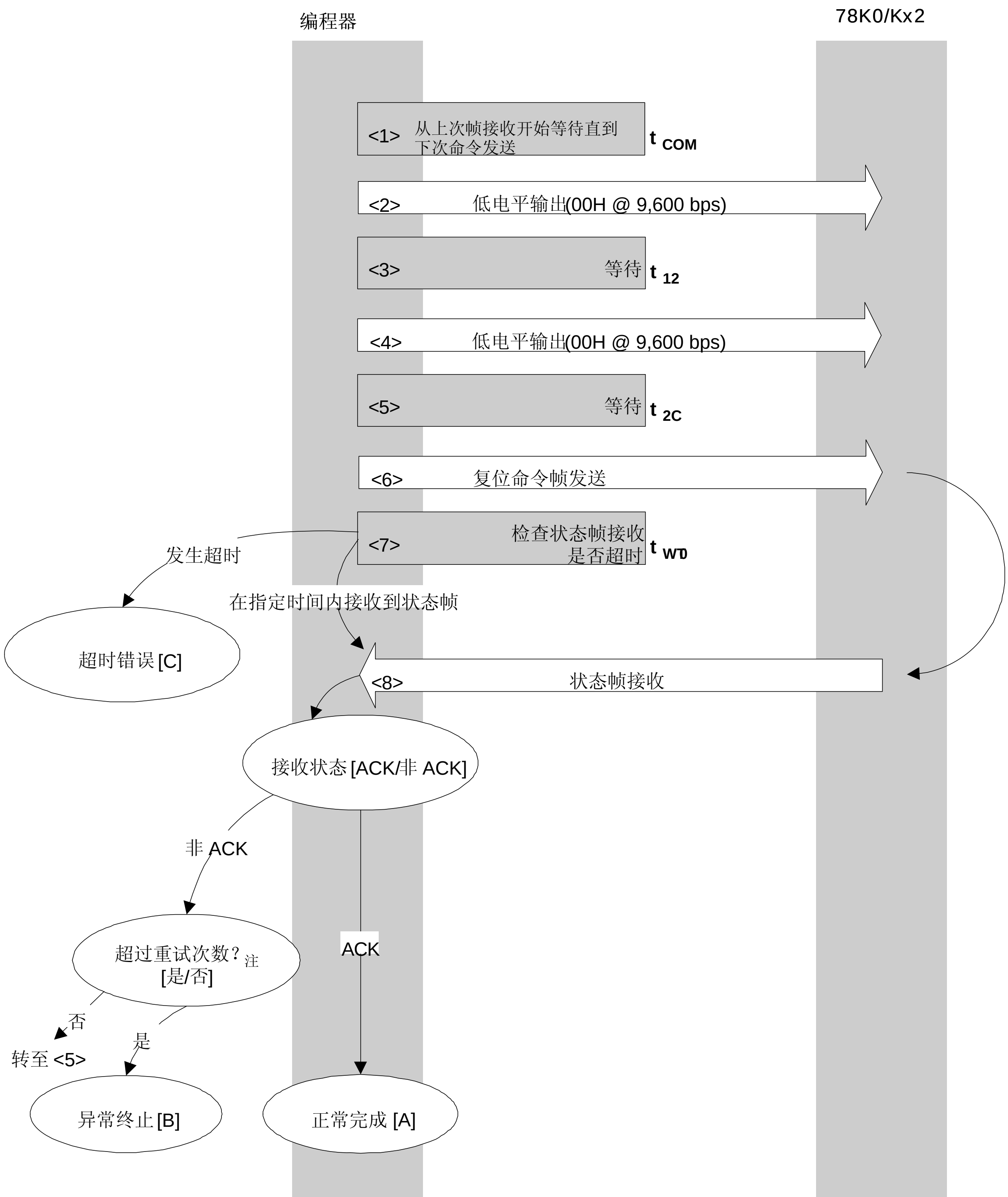
<R> 4.3 数据帧接收处理流程图



4.4　复位命令

4.4.1　处理流程图

复位命令处理流程



注　　请勿超过复位命令发送的重试次数（最多 16 次）。

4.4.2　处理流程描述

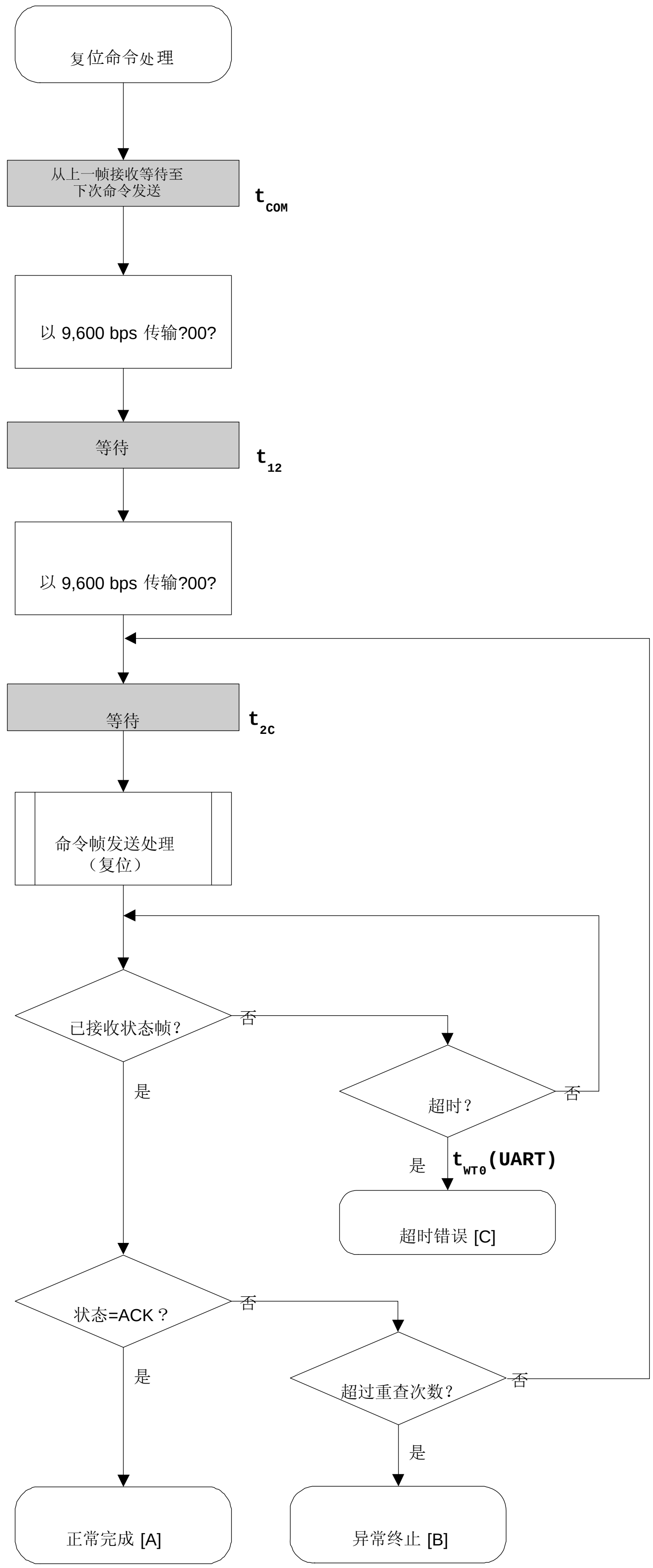
<1>　从上一帧接收开始等待直到下次命令处理开始（等待时间 t_{COM}）。
<2>　输出低电平（以 9,600 bps 发送数据 00H）。
<3>　等待状态（等待时间 t_{12}）。
<4>　输出低电平（以 9,600 bps 发送数据 00H）。
<5>　等待状态（等待时间 t_{2C}）。
<6>　"复位"命令由命令帧发送处理过程进行发送。
<7>　在命令发送到状态帧接收期间执行超时检查。
如果发生超时，返回超时错误 [C]（超时时间 t_{WT0}）。
<8>　检查状态码。

当 ST1 = ACK:　　正常完成 [A]
当 ST1 ≠ ACK:　　检查重查次数 (t_{RS})。
如果未超过重查次数，则从 <5> 开始重新顺序执行。
如果超过重查次数，则处理异常终止 [B]。

4.4.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已经建立编程器与 78K0/Kx2 之间的同步。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。

4.4.4　流程图


复位命令处理
从上一帧接收等待至
下次命令发送
tCOM
以 9,600 bps 传输?00?
等待
t12
以 9,600 bps 传输?00?
等待
t2C
命令帧发送处理
（复位）
已接收状态帧?
否
是
超时？
否
是
tWT0(UART)
超时错误 [C]
状态=ACK？
否
是
超过重查次数?
否
是
正常完成 [A]
异常终止 [B]

4.4.5　示例程序

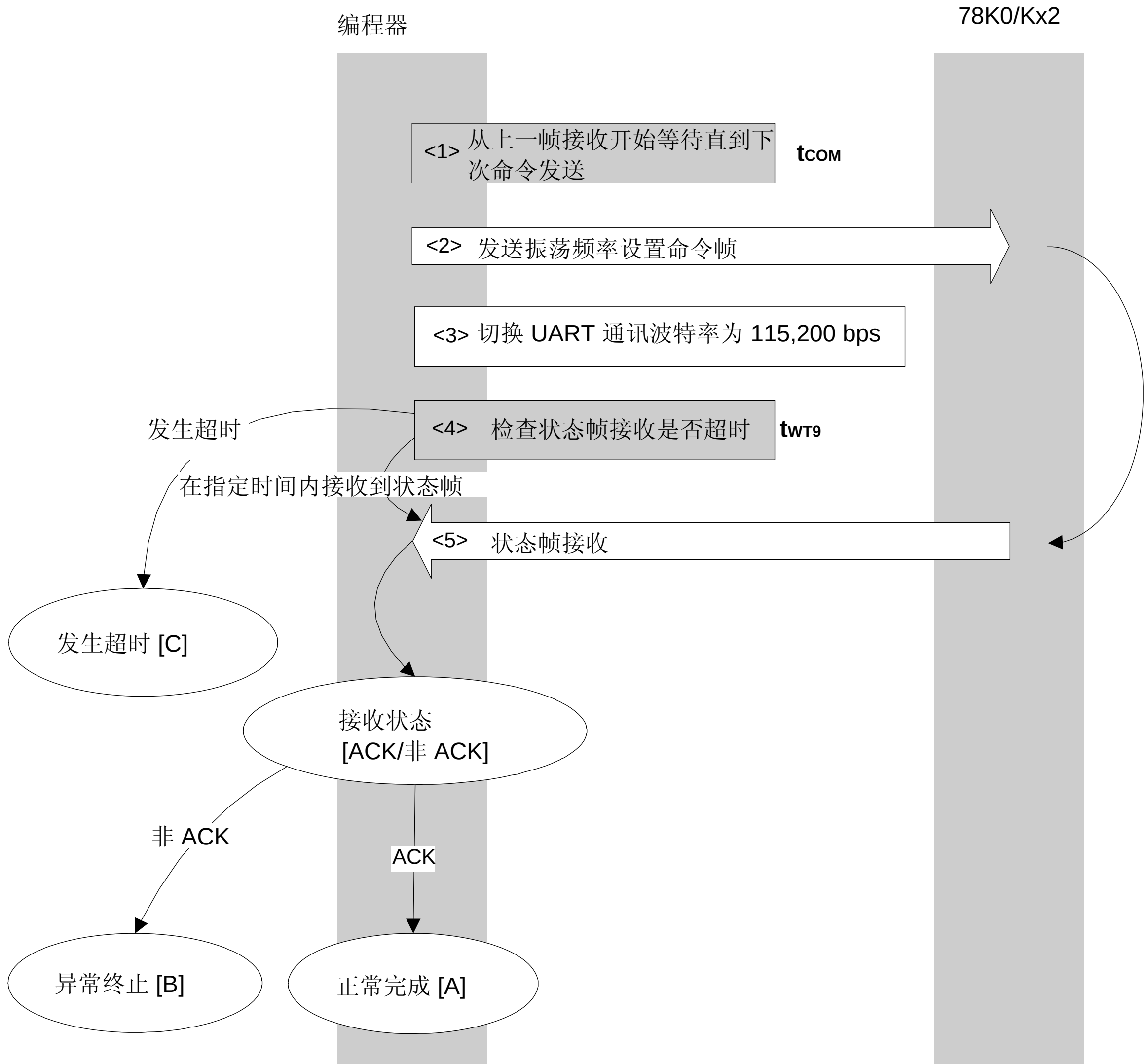
以下所示为“复位”命令处理的示例程序。

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

        set_uart0_br(BR_9600);   // 更改为 9600bps

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

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

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

                fl_wait(t2C);    // 等待

                put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);        // 发送“复位”命令

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

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

4.5　振荡频率设置命令

4.5.1　处理流程图

振荡频率设置命令处理顺序



4.5.2 处理顺序描述

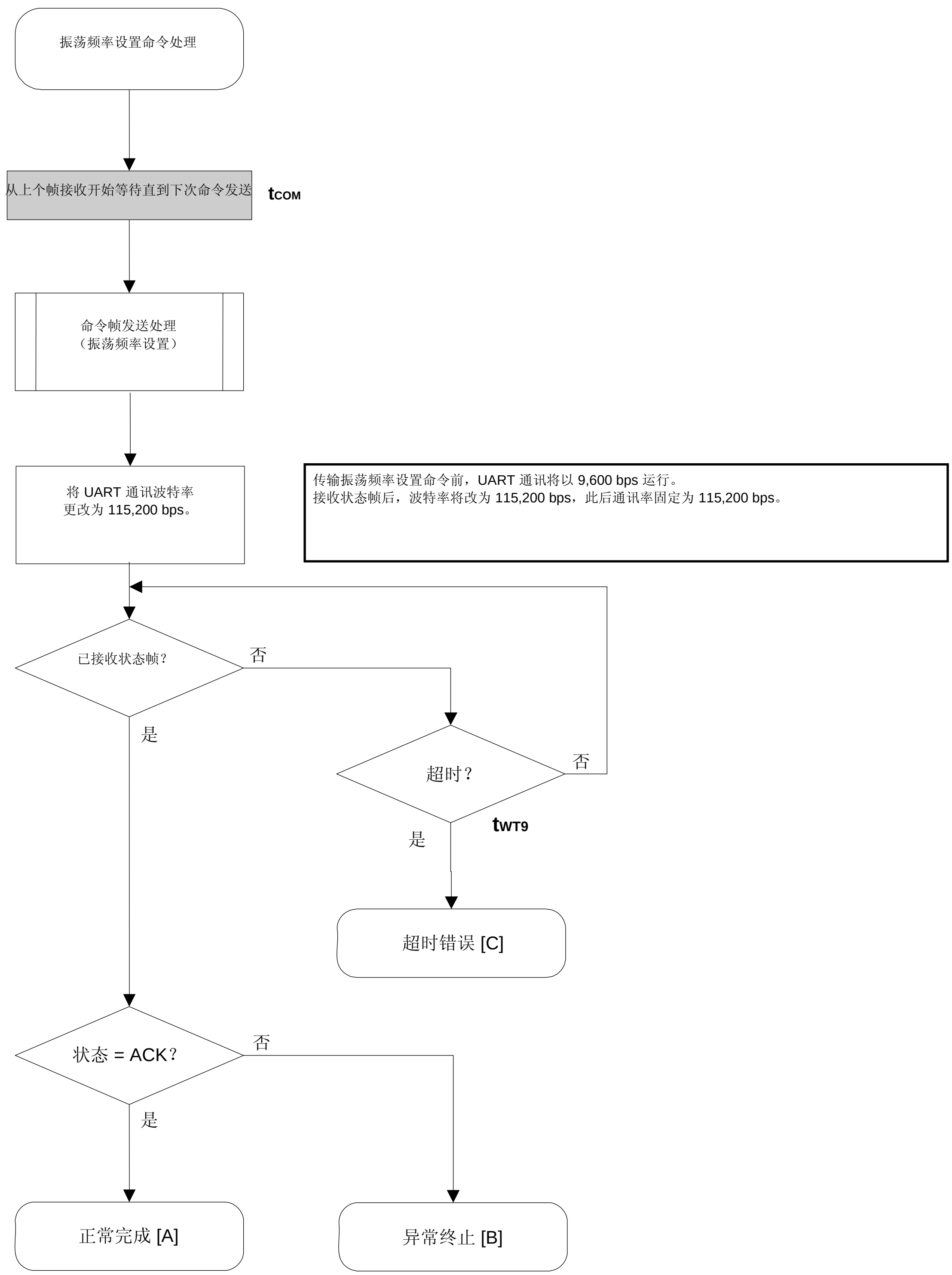
<1> 从上次帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 振荡频率设置命令由命令帧发送处理过程发送。
<3> 接收状态帧之后， UART 波特率更改为 115,200 bps。此后波特率固定为 115,200 bps
<4> 从命令发送开始执行超时检查，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT9}）。
<5> 检查状态码。

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

4.5.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已将操作频率正确地置入 78K0/Kx2。
异常终止 [B]	参数错误	05H	振荡频率值超过范围。
	校验和错误	07H	传输的命令帧校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。

4.5.4 流程图


振荡频率设置命令处理
从上个帧接收开始等待直到下次命令发送
tCOM
命令帧发送处理
（振荡频率设置）
将 UART 通讯波特率
更改为 115,200 bps。
传输振荡频率设置命令前，UART 通讯将以 9,600 bps 运行。
接收状态帧后，波特率将改为 115,200 bps，此后通讯率固定为 115,200 bps。
已接收状态帧？
否
是
超时？
否
tWT9
是
超时错误 [C]
状态 = ACK？
否
是
正常完成 [A]
异常终止 [B]

4.5.5 示例程序

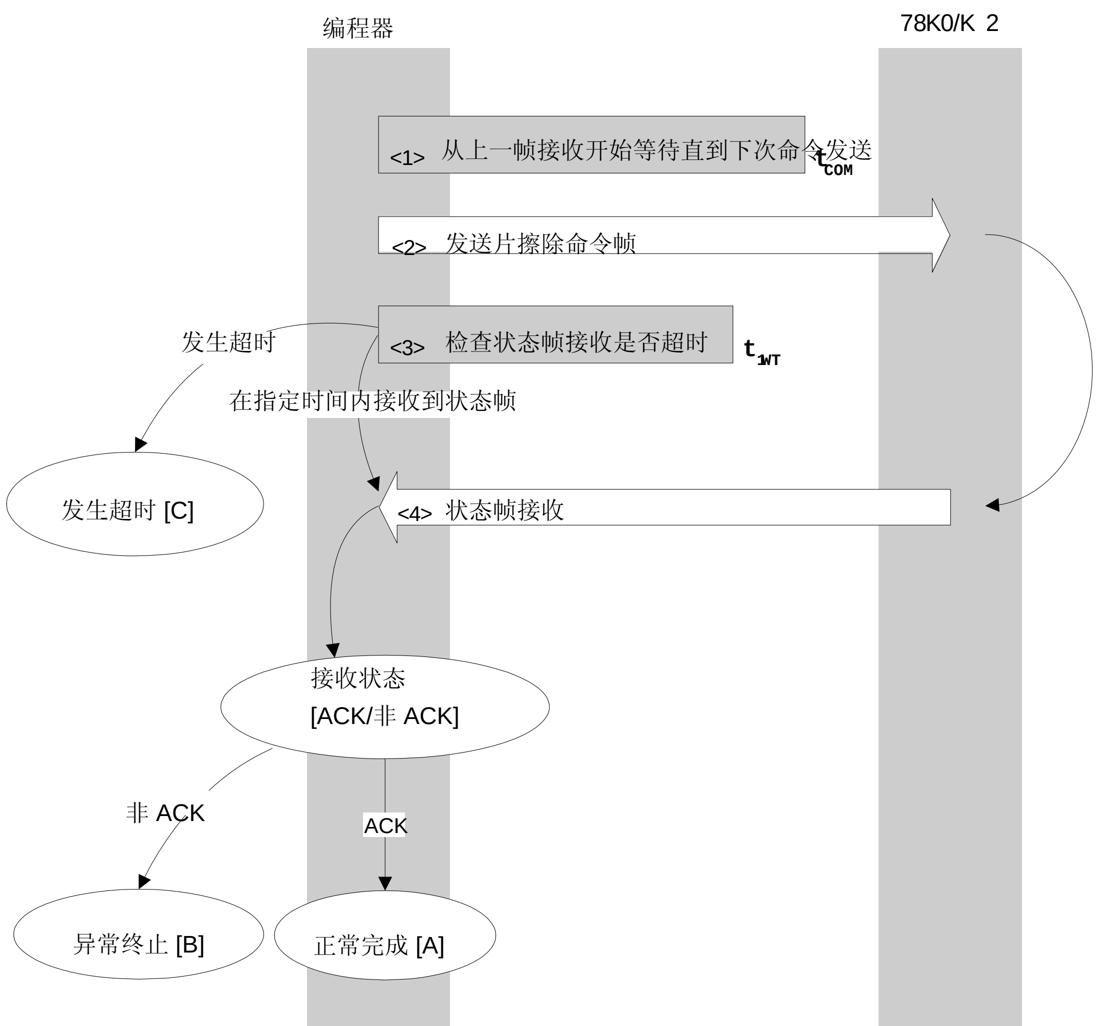
以下所示为振荡频率设置命令处理的示例程序。

```
/*****************************************************************/
/*                                                   */
/* 设置 Flash 设备时钟值命令                                */
/*                                                   */
/*****************************************************************/
/* [i] u8 clk[4]   ... 频率数据      (D1-D4)          */
/* [r] u16         ... 错误代码                 */
/*****************************************************************/
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);


        set_flbaud(BR_115200);              // 更改波特率
        set_uart0_br(BR_115200);            // 更改波特率 (h.w.)


        rc = get_sfrm_ua(fl_ua_sfrm, tWT9_TO);      // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:   return    rc;       break;    // case [A]
//              case    FLC_DFTO_ERR:           return    rc;       break;    // case [C]
//              default:              return    rc;       break;    // case [B]
//      }

        return rc;
}
```

4.6　片擦除命令

4.6.1　处理流程图

片擦除命令处理顺序



4.6.2 处理顺序描述

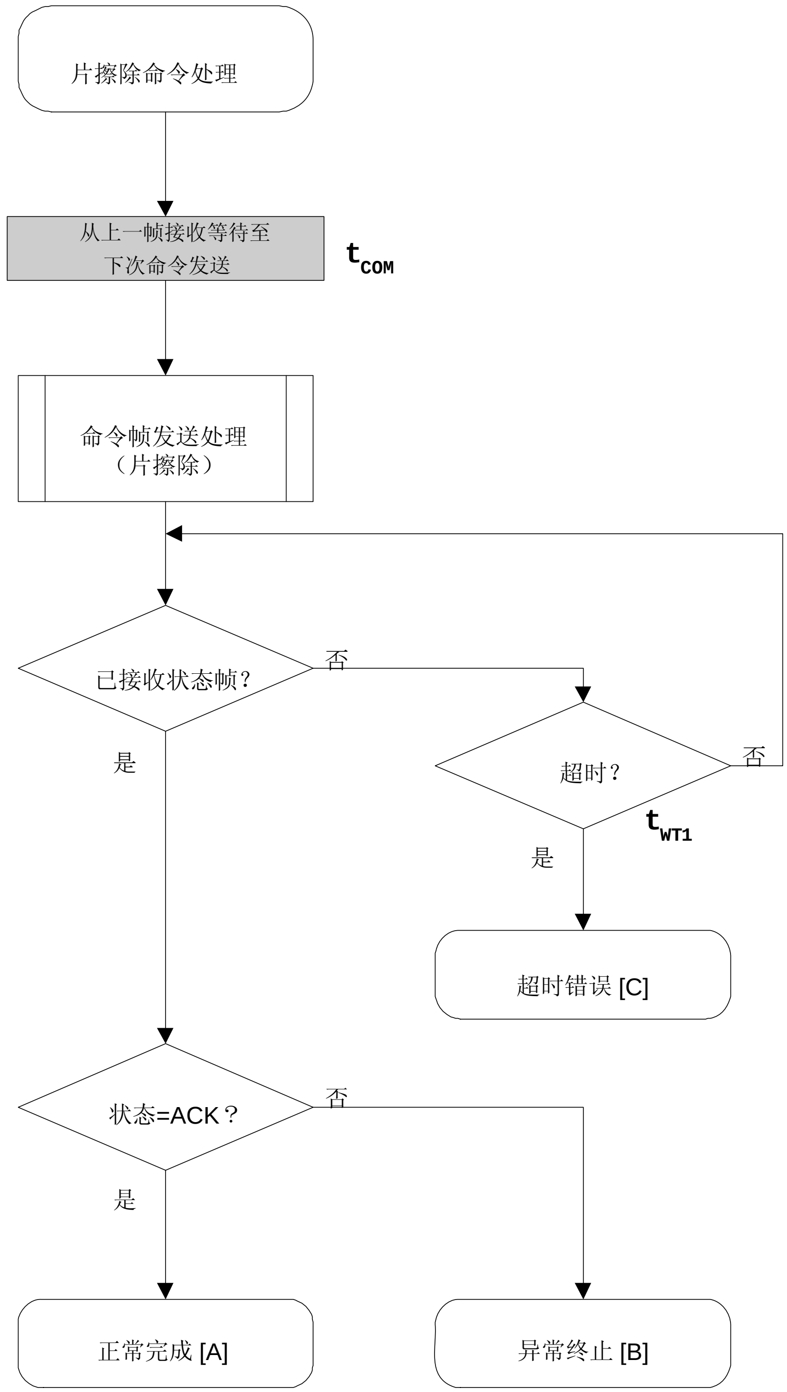
<1> 从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 片擦除命令由命令帧发送处理过程发送。
<3> 超时检查从命令传输时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT1}）。
<4> 检查状态码。

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

4.6.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 **[A]**	正常应答**(ACK)**	**06H**	已正常执行命令，并且已正常执行片擦除。
异常退出 **[B]**	校验和错误	**07H**	已发送命令帧的校验和发生异常。
	保护错误	**10H**	片擦除受安全设置的保护。
	否定应答 **(NACK)**	**15H**	命令帧数据发生异常（例如数据长度 **(LEN)** 无效或无 **ETX**）。
	擦除错误	**1AH**	发生擦除错误。
超时错误 **[C]**		–	未在指定时间内接收到状态帧。

4.6.4　流程图


片擦除命令处理
从上一帧接收等待至
下次命令发送
tCOM
命令帧发送处理
（片擦除）
已接收状态帧？
否
是
超时？
否
tWT1
是
超时错误 [C]
状态=ACK？
否
是
正常完成 [A]
异常终止 [B]

4.6.5　示例程序

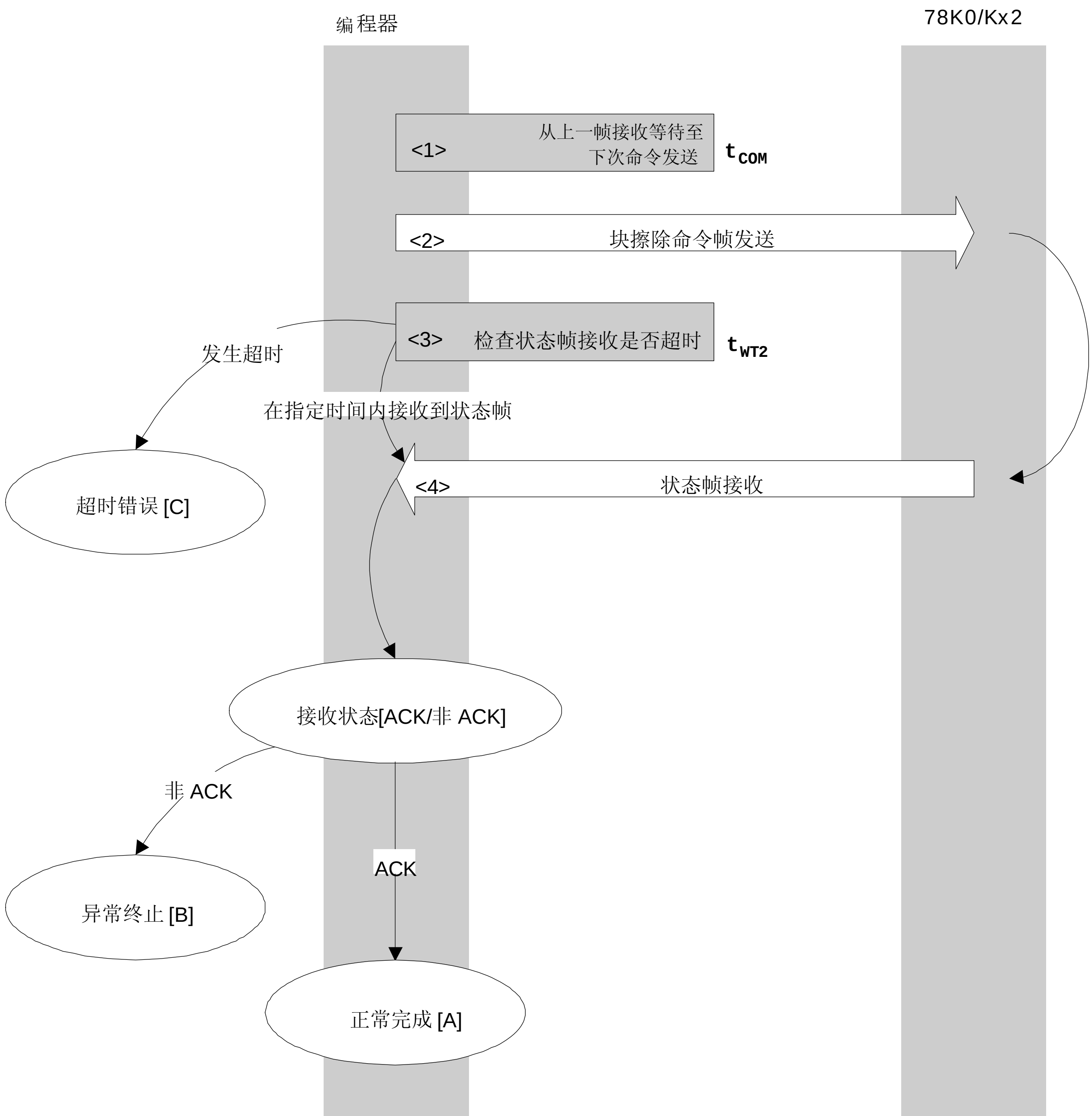
以下说明了片擦除命令处理的示例程序。

```
/****************************************************************/
/*                                                  */
/* 擦除所有（片）命令                                 */
/*                                                  */
/****************************************************************/
/* [r] u16          ... 错误代码                    */
/****************************************************************/
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);   // 获取状态帧
//      switch(rc) {
//
//              case    FLC_NO_ERR:   return   rc;      break;   // case [A]
//              case    FLC_DFTO_ERR:          return   rc;      break;   // case [C]
//              default:              return   rc;      break;   // case [B]
//      }
        return   rc;

}
```

4.7 块擦除命令

4.7.1 处理流程图

块擦除命令处理流程



4.7.2　处理顺序描述

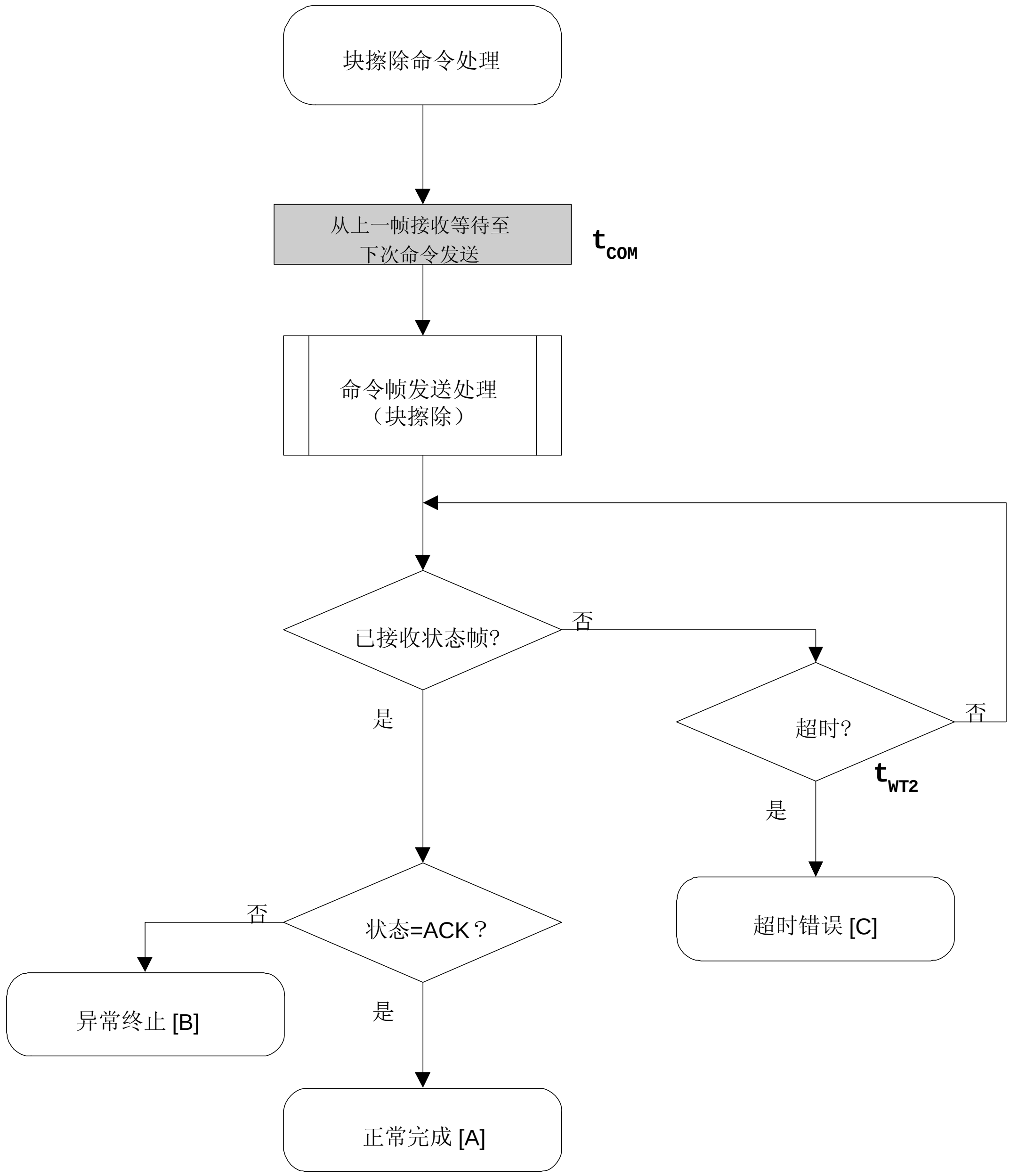
<1>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2>　块擦除命令由命令帧发送处理过程进行发送。
<3>　超时检查从命令发送时开始，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT2}）。
<4>　检查状态码。

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

4.7.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常执行块擦除。
异常终止 [B]	参数错误	05H	块数目超出范围。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	保护错误	10H	写入、块擦除或片擦除受安全设置的保护。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	擦除错误	1AH	发生擦除错误。
超时错误 [C]		–	未在指定时间内接收到状态帧。

4.7.4　流程图


块擦除命令处理
从上一帧接收等待至
下次命令发送
t_{COM}
命令帧发送处理
（块擦除）
已接收状态帧?
否
是
超时?
否
t_{WT2}
是
超时错误 [C]
否
状态=ACK？
异常终止 [B]
是
正常完成 [A]

4.7.5　示例程序

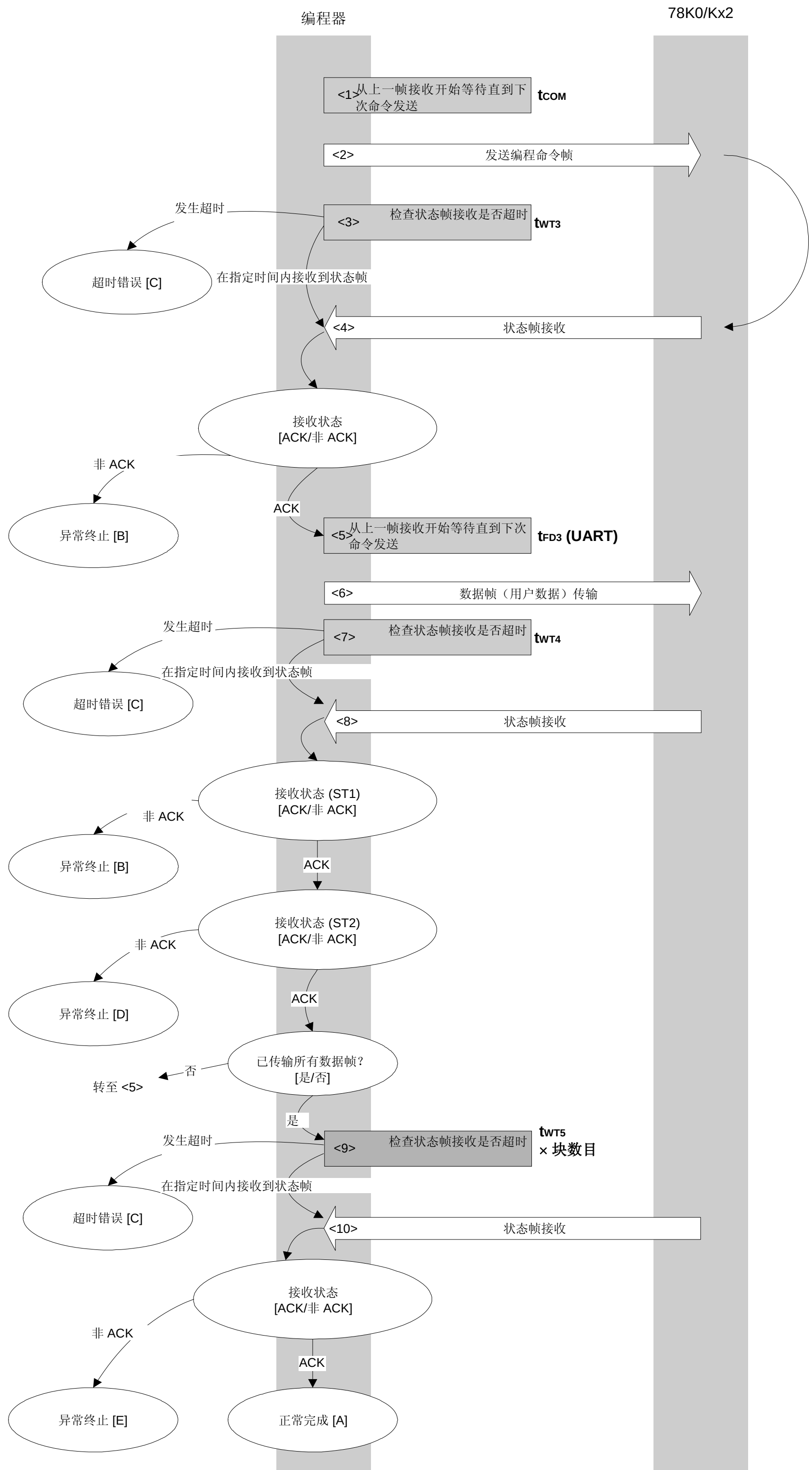
以下所示为一个块的块擦除命令处理的示例程序。

```
/****************************************************************/
/*                                                              */
/* 块擦除命令                                                    */
/*                                                              */
/****************************************************************/
/* [i] u16 sblk   ... 要擦除起始块        (0...255)        */
/* [i] u16 eblk   ... 要擦除结束块        (0...255)        */
/* [r] u16        ... 错误代码                              */
/****************************************************************/
u16         fl_ua_erase_blk(u16 sblk, u16 eblk)
{

        u16     rc;
        u32     wt2_max;
        u32     top, bottom;

        top = get_top_addr(sblk);           // 获取起始块的起始地址
        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

        wt2_max = make_wt2_max(sblk, eblk);

        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, wt2_max);      // 获取状态帧


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

        return rc;

}
```

4.8　编程命令

4.8.1　处理流程图

编程命令处理顺序



4.8.2 处理顺序描述

<1> 从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 编程命令由命令帧发送处理过程进行发送。
<3> 超时检查从命令发送时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT3}）。
<4> 检查状态码。

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

<5> 从上次帧接收等待至下次命令发送（等待时间 t_{COM}(UART)）。
<6> 用户数据由数据帧发送处理过程进行发送。
<7> 超时检查从用户数据发送时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT4}）。
<8> 检查状态码 (ST1/ST2)（也可参阅处理流程图与流程图）。

当 ST1 ≠ ACK: 异常终止 [B]
当 ST1 = ACK: 以下处理将按照 ST2 值进行操作。
- 当 ST2 = ACK: 所有数据帧发送完成后，转至 <9>。
  如果仍有数据帧未发送，该流程将从 <5> 重新顺序执行。
- 当 ST2 ≠ ACK: 异常终止 [D]

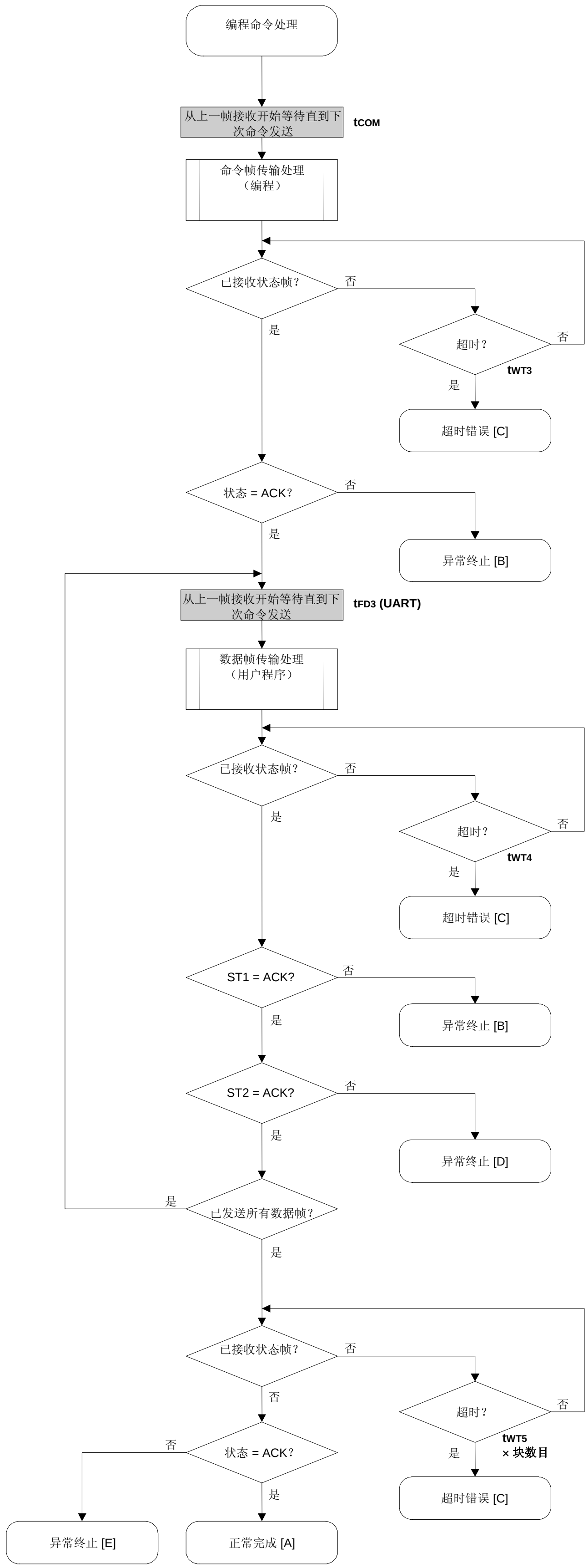
<9> 执行超时检查，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT5}× 块数目）。
<10> 检查状态码。

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

4.8.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常写入用户数据。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者不是 8 的倍数。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	保护错误	10H	写入操作受安全设置的保护。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
异常终止 [D]	写入错误	1CH (ST2)	发生写入错误。
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

4.8.4　流程图


编程命令处理
从上一帧接收开始等待直到下次命令发送
tCOM
命令帧传输处理（编程）
已接收状态帧？
否
超时？
tWT3
是
超时错误 [C]
状态 = ACK?
异常终止 [B]
从上一帧接收开始等待直到下次命令发送
tFD3 (UART)
数据帧传输处理（用户程序）
已接收状态帧？
超时？
tWT4
超时错误 [C]
ST1 = ACK?
异常终止 [B]
ST2 = ACK?
异常终止 [D]
已发送所有数据帧？
已接收状态帧？
超时？
tWT5
× 块数目
超时错误 [C]
状态 = ACK?
异常终止 [E]
正常完成 [A]

4.8.5 示例程序

以下所示为编程命令处理的示例程序。

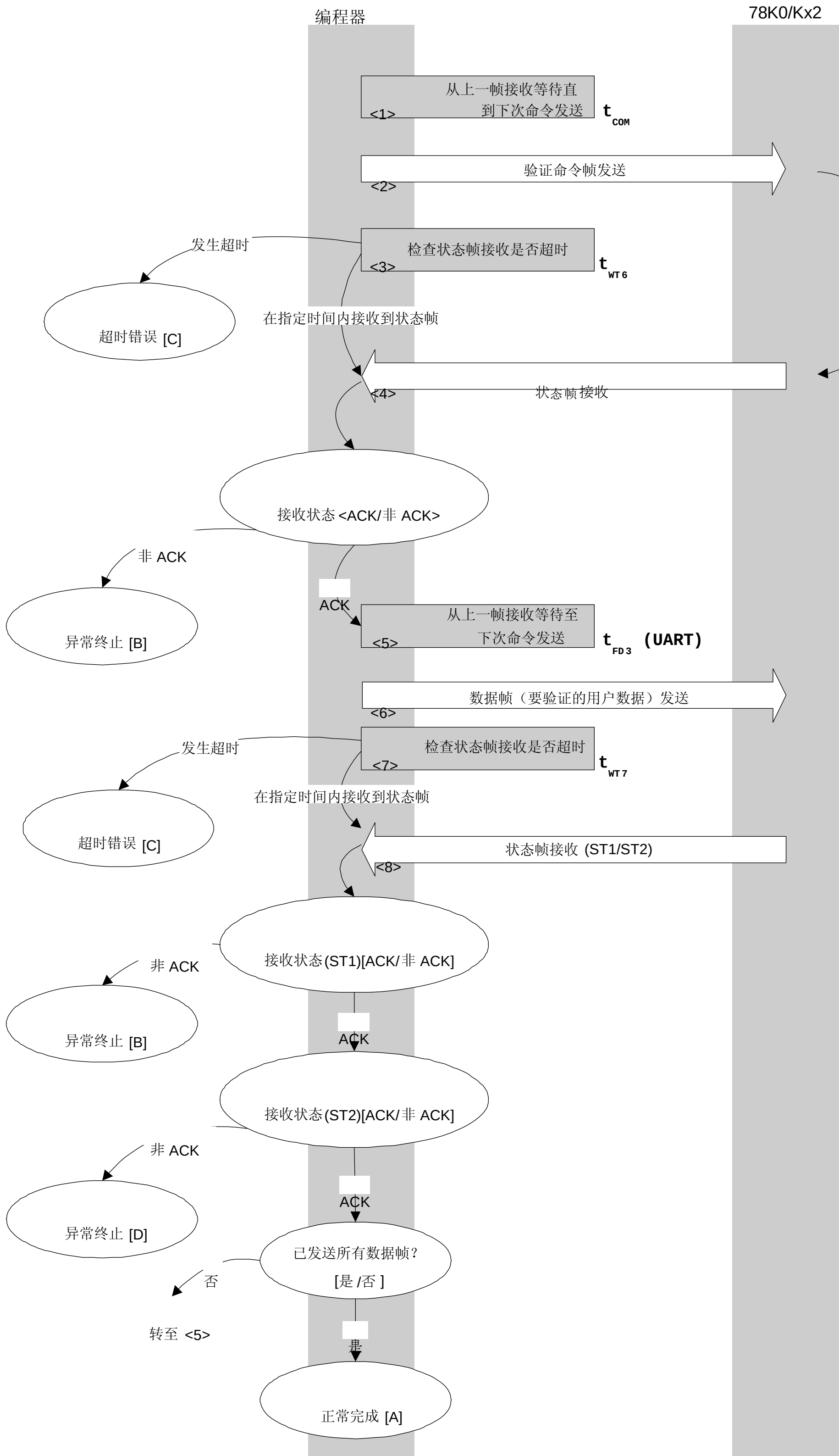
```
/******************************************************************/
/*                                                                */
/* 写入命令                                                       */
/*                                                                */
/******************************************************************/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

#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);     // 获取块数目

        /************************************************/
        /*      发送命令和检查状态                      */
        /************************************************/
        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_TO);      // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // continue
//              case    FLC_DFTO_ERR:       return  rc;         break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }

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

```
    while(1){

        // make send data frame
        if ((bottom - send_head) > 256){                // 其余大小大于 256 ?
            is_end = false;                             // "是"代表不是结束帧
            send_size  = 256;                           // 传输大小 = 256 字节
        }
        else{
            is_end = true;
            send_size = bottom - send_head + 1; // 传输大小 = (bottom –
                                                        // send_head)+1 字节

        }
        memcpy(fl_txdata_frm, rom_buf+send_head, send_size);        // 设置数据帧
                                                                    // 净荷
        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);             // 获取状态帧
        switch(rc) {
            case    FLC_NO_ERR:                     break;  // 继续
            case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
            default:            return  rc;     break;  // case [B]
        }
        if (fl_st2_ua != FLST_ACK){                 // ST2 = ACK ?
            rc = decode_status(fl_st2_ua);          // 否
            return  rc;                                     // case [D]
        }
        if (is_end)
            break;

    }
    /************************************************/
    /*  进行内部检验                        */
    /************************************************/
    rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX * block_num));       // 再次获取状态帧
//  switch(rc) {
//      case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//      case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
//      default:            return  rc;     break;  // case [E]
//  }
    return rc;
}
```

4.9　验证命令

4.9.1　处理流程图

验证命令处理顺序



4.9.2　处理顺序描述

<1>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2>　验证命令由命令帧发送处理过程进行发送。
<3>　超时检查从命令发送时开始，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT6}）。
<4>　检查状态码。

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

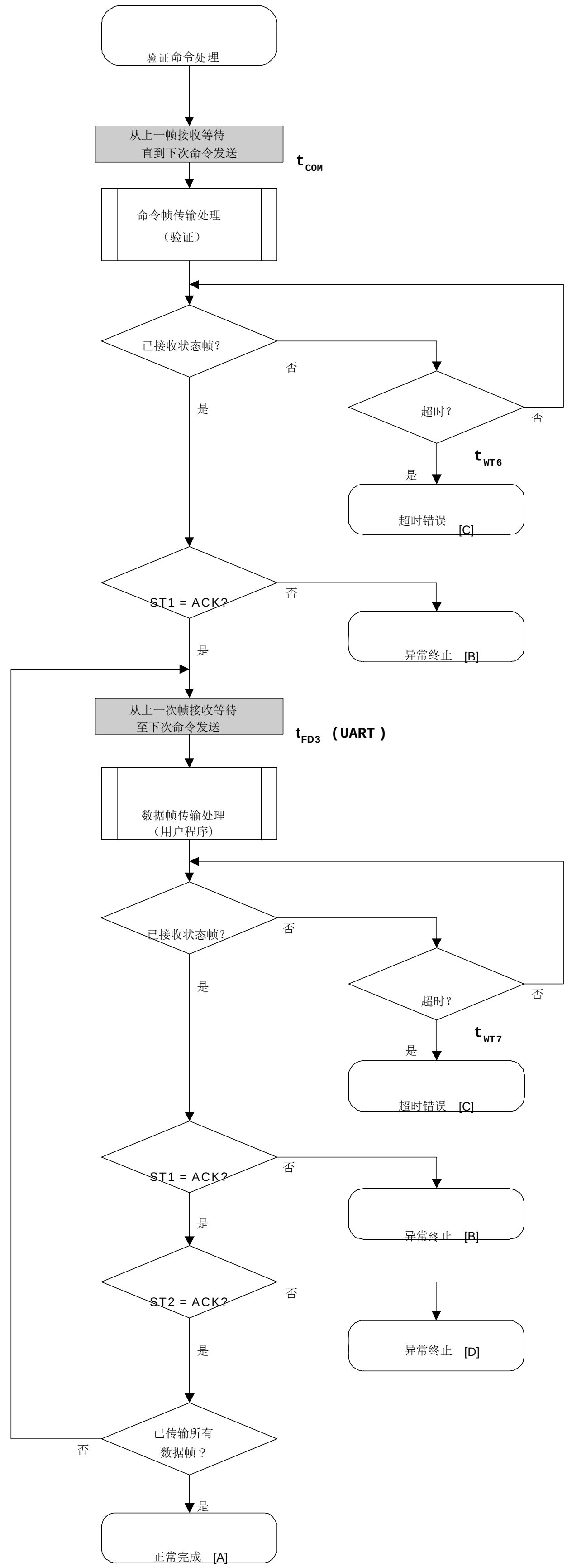
<5>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}(UART)）。
<6>　要验证的用户数据由数据帧发送处理过程进行发送。
<7>　超时检查从用户数据发送时开始，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT7}）。
<8> 检查状态码 (ST1/ST2)（也可参阅处理流程图与流程图）。

　　当 ST1 ≠ ACK:　　异常终止 [B]
　　当 ST1 = ACK:　　以下处理将按照 ST2 值进行操作。
- 当 ST2 = ACK:　　如果所有数据帧发送完成，则处理将正常结束 [A]。
  　　如果仍有数据帧未发送，该流程将从 <5> 重新顺序执行。
- 当 ST2 ≠ ACK:　　异常终止 [D]

4.9.3　处理完成时状态

处理完成时状态		状态码	说明
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常完成验证。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围。
	校验和错误	07H	已发送命令帧或数据帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
异常终止 [D]	验证错误	0FH (ST2)	验证失败或发生其他错误。

4.9.4　流程图



验证命令处理
从上一帧接收等待
直到下次命令发送
t_{COM}
命令帧传输处理
（验证）
已接收状态帧？
否
超时？
否
t_{WT6}
是
超时错误 [C]
是
ST1 = ACK?
否
异常终止 [B]
是
从上一次帧接收等待
至下次命令发送
t_{FD3} （UART）
数据帧传输处理
（用户程序）
已接收状态帧？
否
超时？
否
t_{WT7}
是
超时错误 [C]
是
ST1 = ACK?
否
异常终止 [B]
是
ST2 = ACK?
否
异常终止 [D]
是
已传输所有
数据帧？
否
是
正常完成 [A]

4.9.5 示例程序

以下所示为验证命令处理的示例程序。

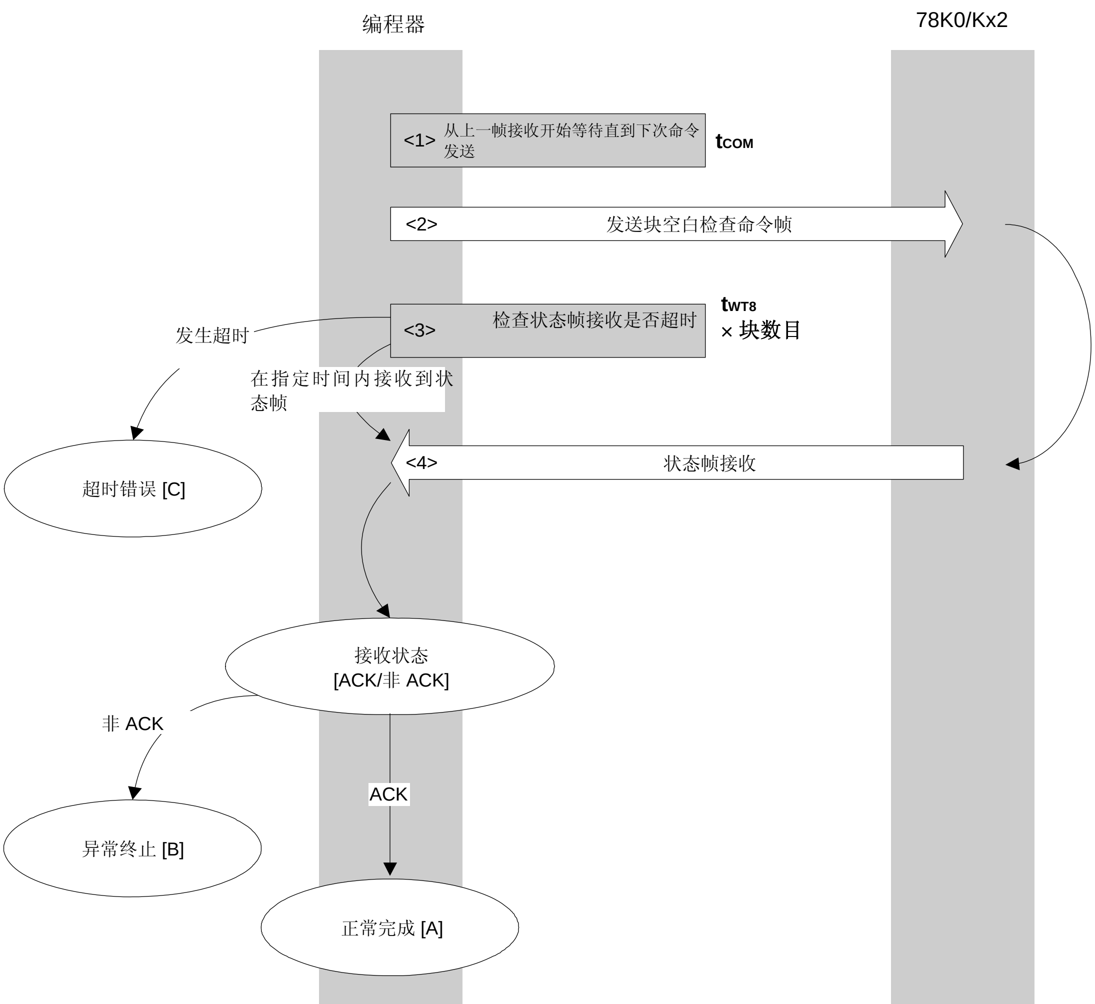
```
/****************************************************************/
/*                                                              */
/* 验证命令                                                     */
/*                                                              */
/****************************************************************/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

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


        /************************************************/
        /*      发送命令与检查状态                      */
        /************************************************/

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

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

        while(1){

                // make send data frame
                if ((bottom - send_head) > 256){        // 其他大小大于 256 ?
```

```
            is_end = false;                       // “是”代表非结束帧
            send_size  = 256;                     // 传输大小 = 256 字节
        }
        else{
            is_end = true;
            send_size = bottom - send_head + 1;           // 传输大小 = (bottom
                                                          // - send_head)+1 字节

        }
        memcpy(fl_txdata_frm, rom_buf+send_head, send_size);      // 设置数据帧
                                                                  // 净荷
        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_TO);            // 获取状态帧
        switch(rc) {
            case    FLC_NO_ERR:                     break;  // 继续
//          case    FLC_DFTO_ERR:       return  rc;     break;  // case [C]
            default:        return  rc;     break;  // case [B]
        }
        if (fl_st2_ua != FLST_ACK){               // ST2 = ACK ?
            rc = decode_status(fl_st2_ua);        // 否
            return  rc;                                   // case [D]
        }
        if (is_end)                           // 已发送所有用户数据？
            break;                            // 是
        //continue;
    }
    return  FLC_NO_ERR;    // case [A]
}
```

4.10 块空白检查命令

4.10.1 处理流程图

块空白检查名称处理顺序



4.10.2 处理顺序描述

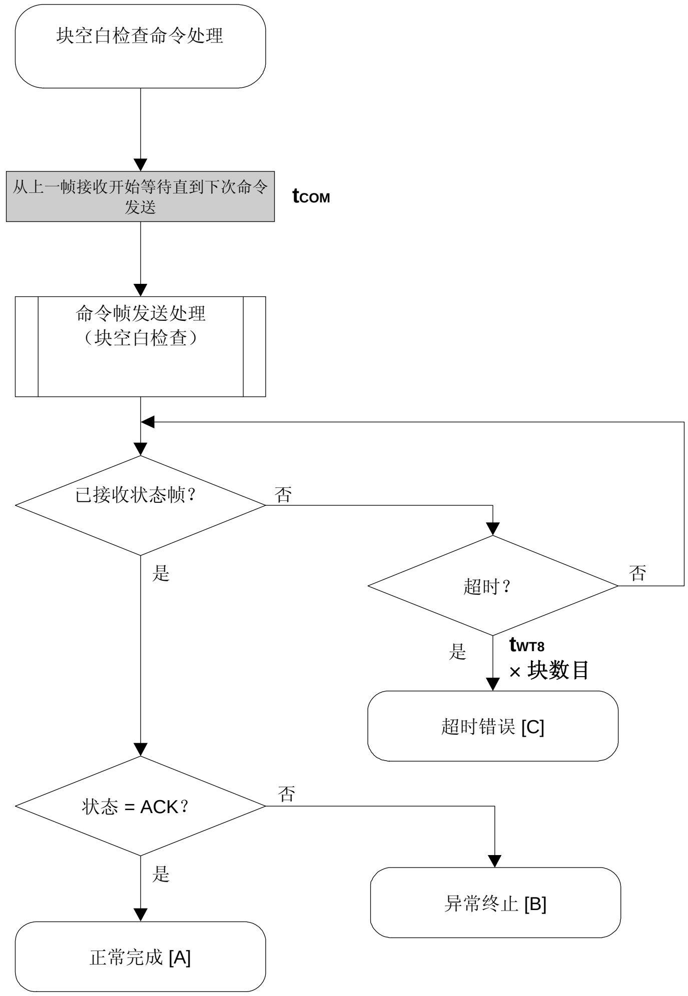
<1>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2>　块空白检查命令由命令帧发送处理过程进行发送。
<3>　超时检查从命令发送时开始，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 $t_{WT8}\times$ 块数目）。
<4>　检查状态码。

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

4.10.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且所有指定块为空白。
异常终止 [B]	参数错误	05H	块数目超出范围。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	MRG11 错误	1BH	Flash 存储器内指定的块不为空白。
超时错误 [C]		–	状态帧接收发生超时错误。

4.10.4 流程图



块空白检查命令处理
从上一帧接收开始等待直到下次命令发送
t_{COM}
命令帧发送处理
（块空白检查）
已接收状态帧？
否
是
超时？
否
是
t_{WT8}
× 块数目
超时错误 [C]
状态 = ACK？
否
是
异常终止 [B]
正常完成 [A]

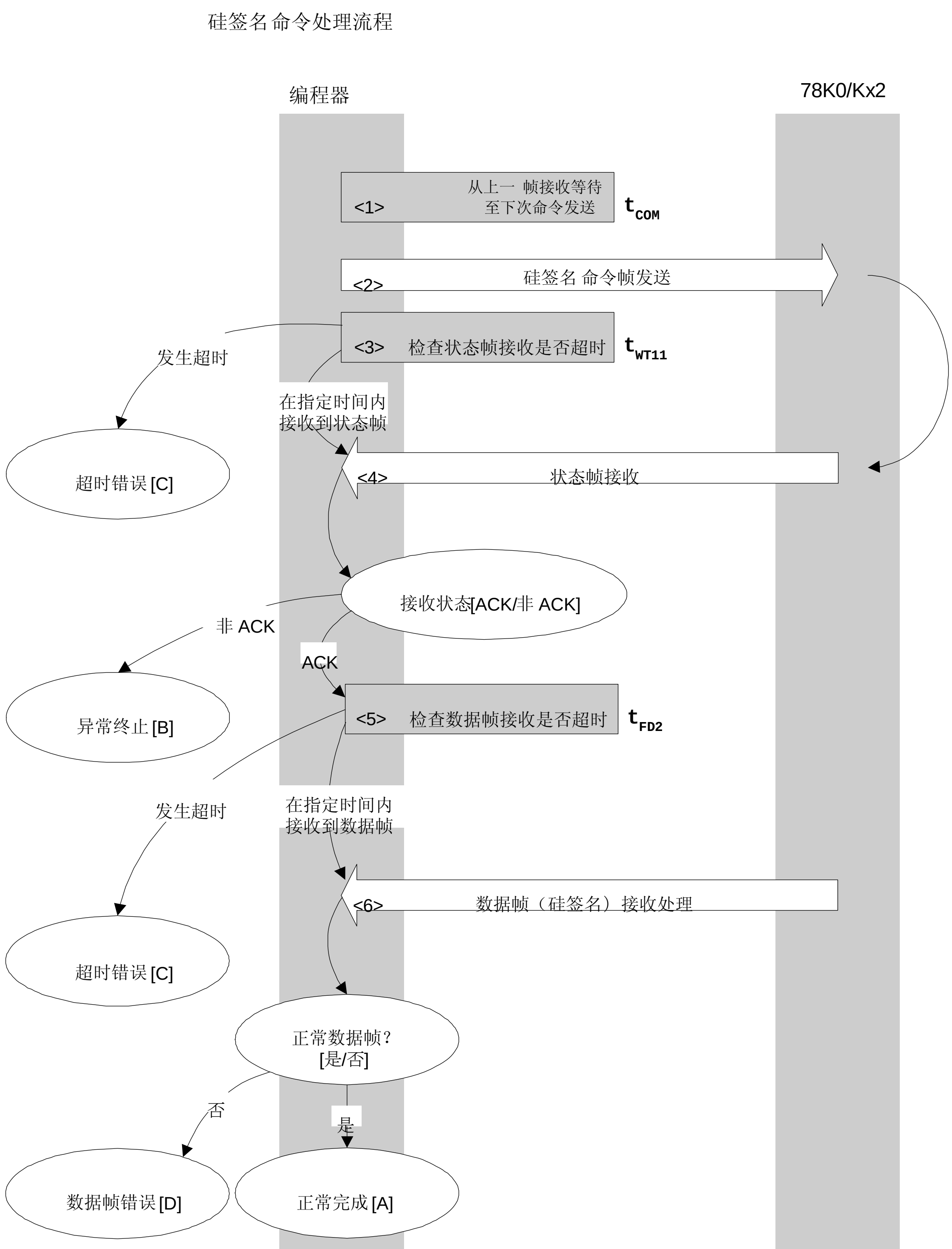
4.10.5 示例程序

以下所示为块空白检查命令处理的示例程序。

```
/***************************************************************/
/*                                                             */
/*块空白检查命令                                               */
/*                                                             */
/***************************************************************/
/* [i] u32 top     ... 起始地址                                */
/* [i] u32 bottom  ... 结束地址                                */
/* [r] u16         ... 错误代码                                */
/***************************************************************/
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);    // 获取块数目

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

}
```

4.11 硅签名命令

4.11.1 处理流程图



4.11.2 处理顺序描述

<1> 从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 硅签名命令由命令帧发送处理过程进行发送。
<3> 超时检查从命令发送时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT11}）。
<4> 检查状态码。

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

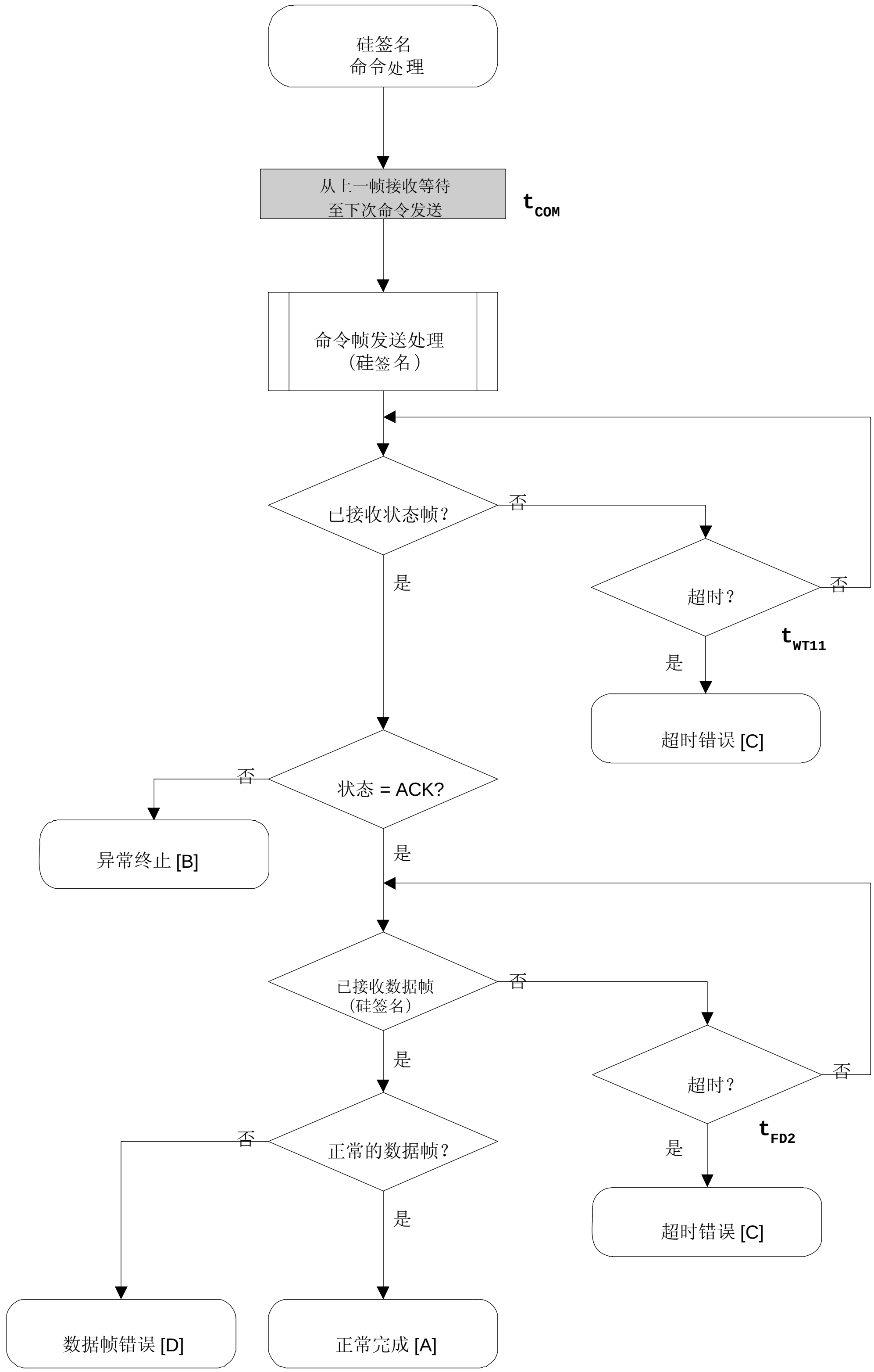
<5> 执行超时检查，直至数据帧（硅签名数据）接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{FD2}）。
<6> 检查已接收数据帧（硅签名数据）。

如果数据帧正常： 正常完成 [A]
如果数据帧异常： 数据帧错误 [D]

4.11.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取硅签名。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	读取错误	20H	读取安全信息失败。
超时错误 [C]		–	状态帧接收或数据帧接收发生超时错误。
数据帧错误 [D]		–	接收到数据帧校验和，因为硅签名数据异常。

4.11.4 流程图


硅签名
命令处理
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧发送处理
（硅签名）
已接收状态帧？
否
是
超时？
否
t_{WT11}
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
已接收数据帧
（硅签名）
否
是
超时？
否
t_{FD2}
是
超时错误 [C]
正常的数据帧？
否
是
数据帧错误 [D]
正常完成 [A]

4.11.5 示例程序

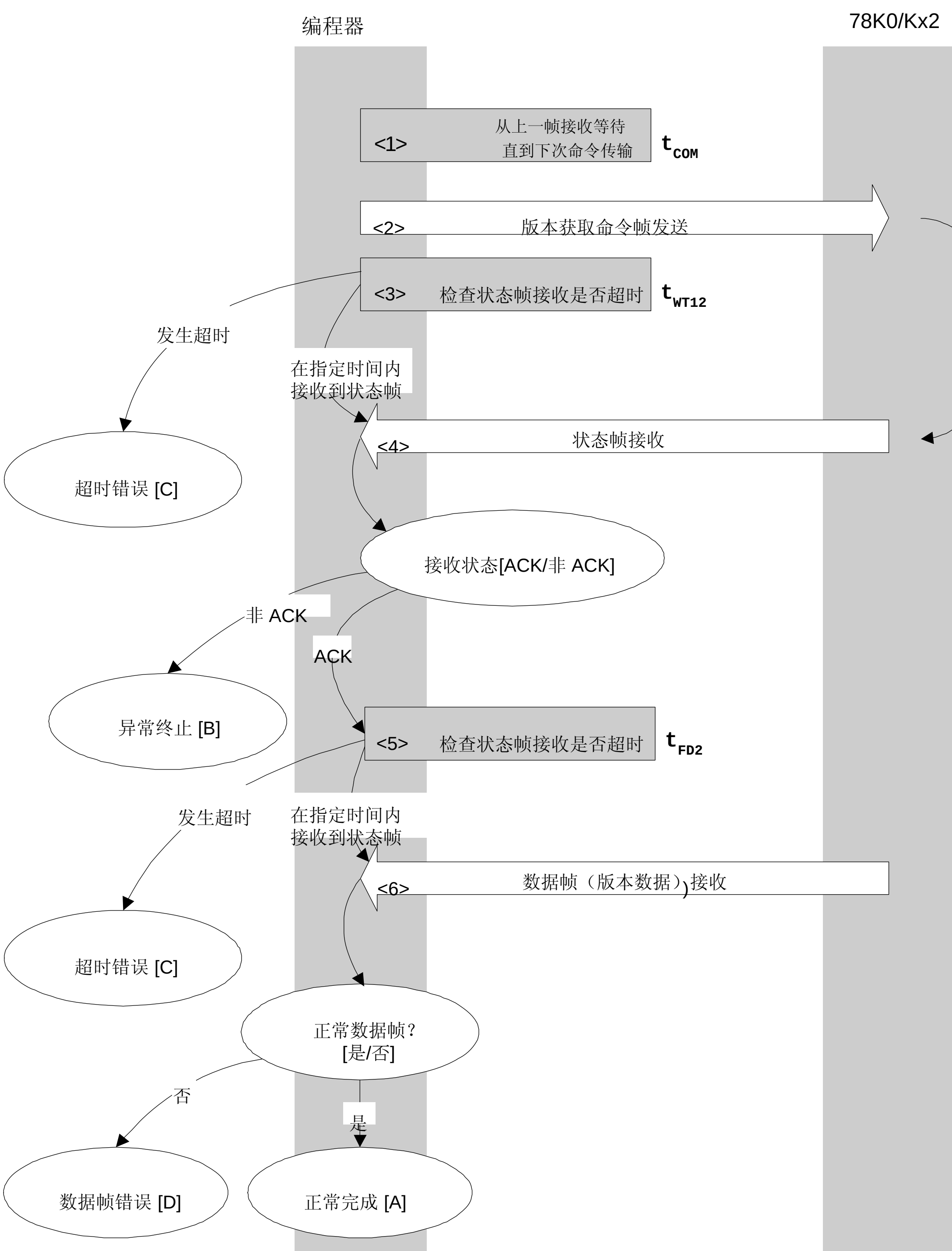
以下说明了硅签名命令处理的示例程序。

```
/***************************************************************/
/*                                                             */
/* 获取硅签名命令                                              */
/*                                                             */
/***************************************************************/
/* [i] u8 *sig    ... 指向签名保存区域的指针           */
/* [r] u16        ... 错误代码                         */
/***************************************************************/
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“获取签名”命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT11_TO);    // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
//              case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }


        rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO);  // 获取状态帧
        if (rc){                                   // 如果发生错误
                return rc;                                  // case [D]
        }
        memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                                    // 复制签名数据
        return  rc;                                // case [A]
}
```

4.12 版本获取命令

4.12.1 处理流程图

版本获取命令处理流程



4.12.2 处理顺序描述

<1> 从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 版本获取命令由命令帧发送处理过程进行发送。
<3> 超时检查从命令发送时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT12}）。
<4> 检查状态码。

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

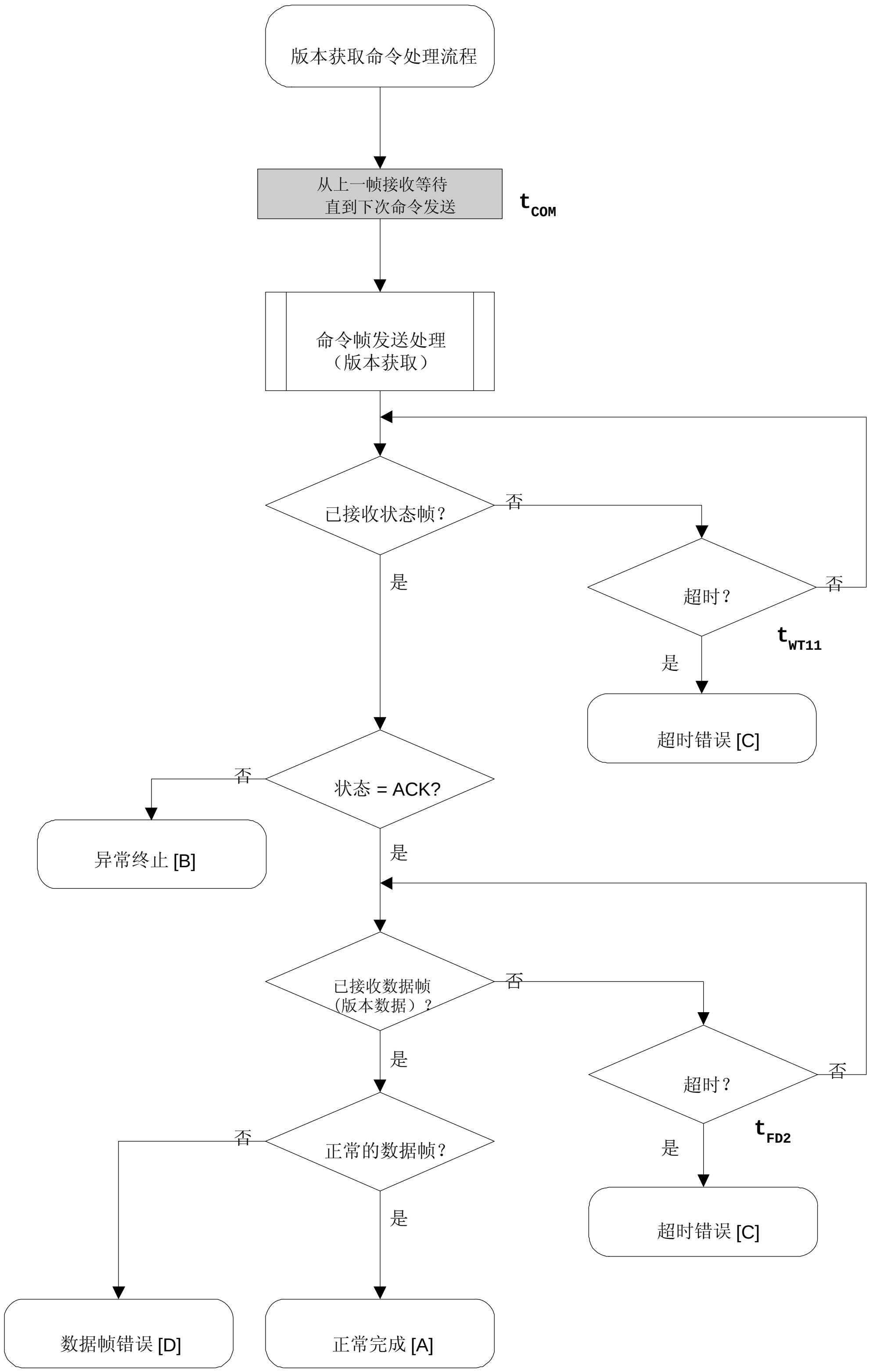
<5> 执行超时检查，直至数据帧（版本数据）接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{FD2}）。
<6> 检查已接收数据帧（版本数据）。

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]

4.12.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取版本数据。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	状态帧接收或数据帧接收发生超时错误。
数据帧错误 [D]		–	接收到数据帧校验和，因为版本数据异常。

4.12.4 流程图



版本获取命令处理流程
从上一帧接收等待
直到下次命令发送
t_{COM}
命令帧发送处理
（版本获取）
已接收状态帧？
否
是
超时？
否
t_{WT11}
是
超时错误 [C]
状态 = ACK?
否
异常终止 [B]
是
已接收数据帧
（版本数据）？
否
是
超时？
否
t_{FD2}
是
超时错误 [C]
正常的数据帧？
否
是
数据帧错误 [D]
正常完成 [A]

4.12.5 示例程序

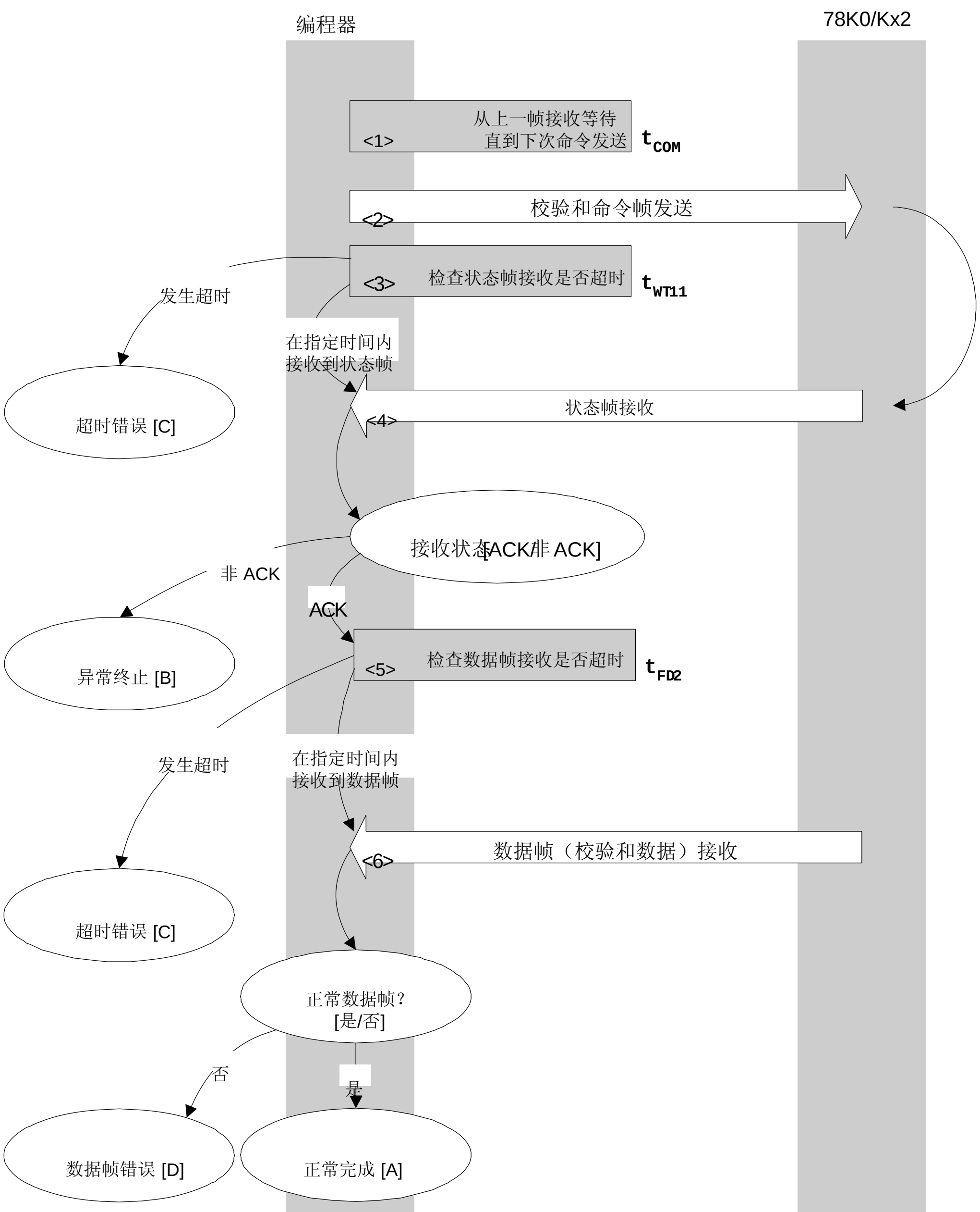
以下所示为版本获取命令处理的示例程序。

```
/******************************************************************/
/*                                                                */
/* 获取设备/固件版本命令                                          */
/*                                                                */
/******************************************************************/
/* [i] u8 *buf   ... 指向版本数据保存区域的指针                   */
/* [r] u16       ... 错误代码                                     */
/******************************************************************/
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”命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT12_TO);    // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
//              case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }

        rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO);  // 获取数据帧
        if (rc){
                return rc;                      // case [D]
        }

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

4.13 校验和命令

4.13.1 处理流程图

校验和命令处理流程



4.13.2 处理顺序描述

<1> 2.从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2> 校验和命令由命令帧发送处理过程进行发送。
<3> 超时检查从命令发送时开始，直至状态帧接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{WT16}）。
<4> 检查状态码。

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

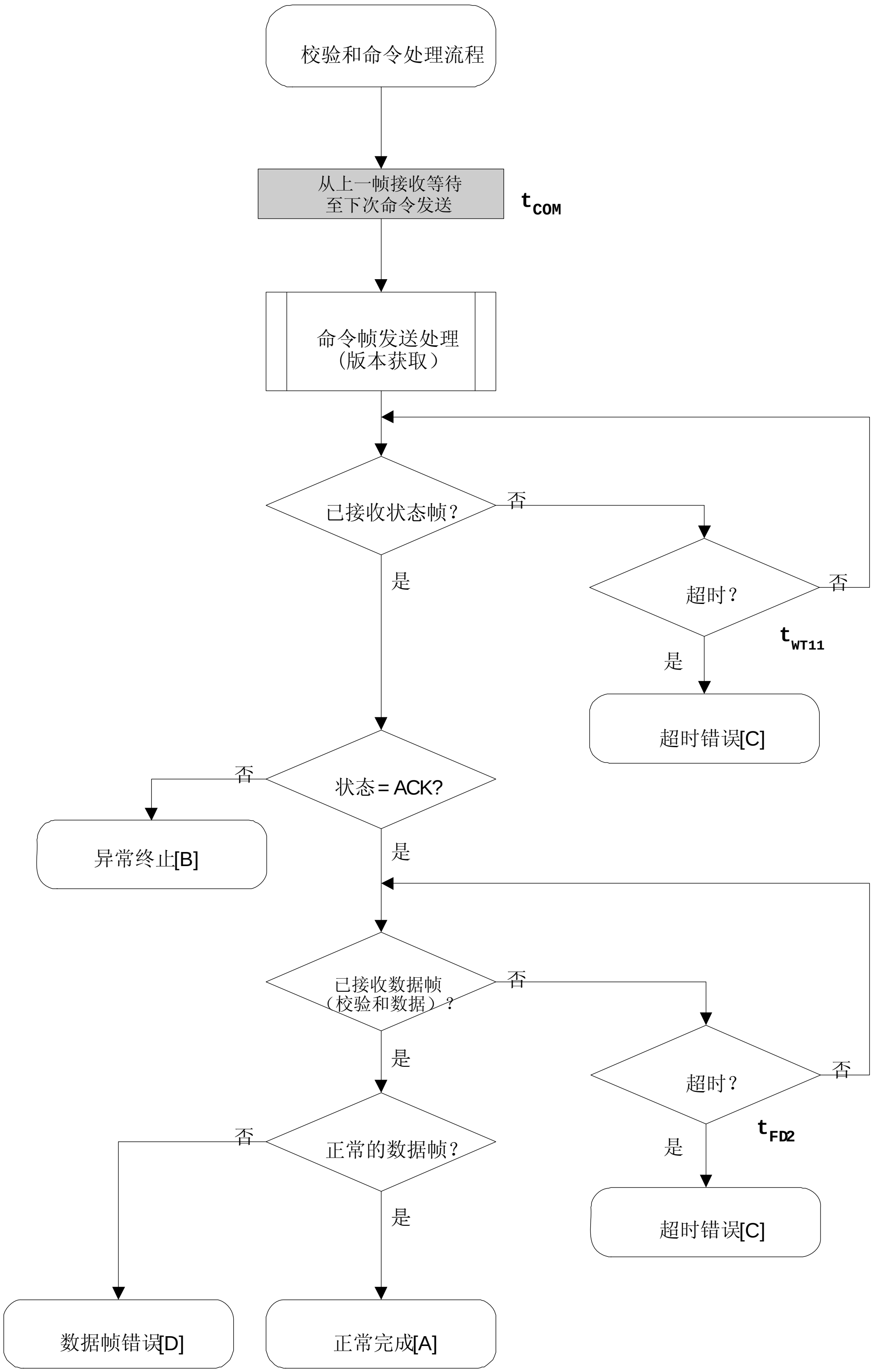
<5> 执行超时检查，直至数据帧（校验和数据）接收。
如果发生超时，则返回超时错误 [C]（超时时间 t_{FD1}）。
<6> 检查已接收数据帧（校验和数据）。

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]

4.13.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取校验和数据。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者指定的地址不是 2 KB 单位内的固定地址。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	状态帧接收或数据帧接收发生超时错误。
数据帧错误 [D]		–	接收到数据帧校验和，因为版本数据异常。

4.13.4 流程图


校验和命令处理流程
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧发送处理
（版本获取）
已接收状态帧？
否
是
超时？
否
是
t_{WT11}
超时错误[C]
状态＝ACK?
否
是
异常终止[B]
已接收数据帧
（校验和数据）？
否
是
超时？
否
是
t_{FD2}
超时错误[C]
正常的数据帧？
否
是
数据帧错误[D]
正常完成[A]

4.13.5 示例程序

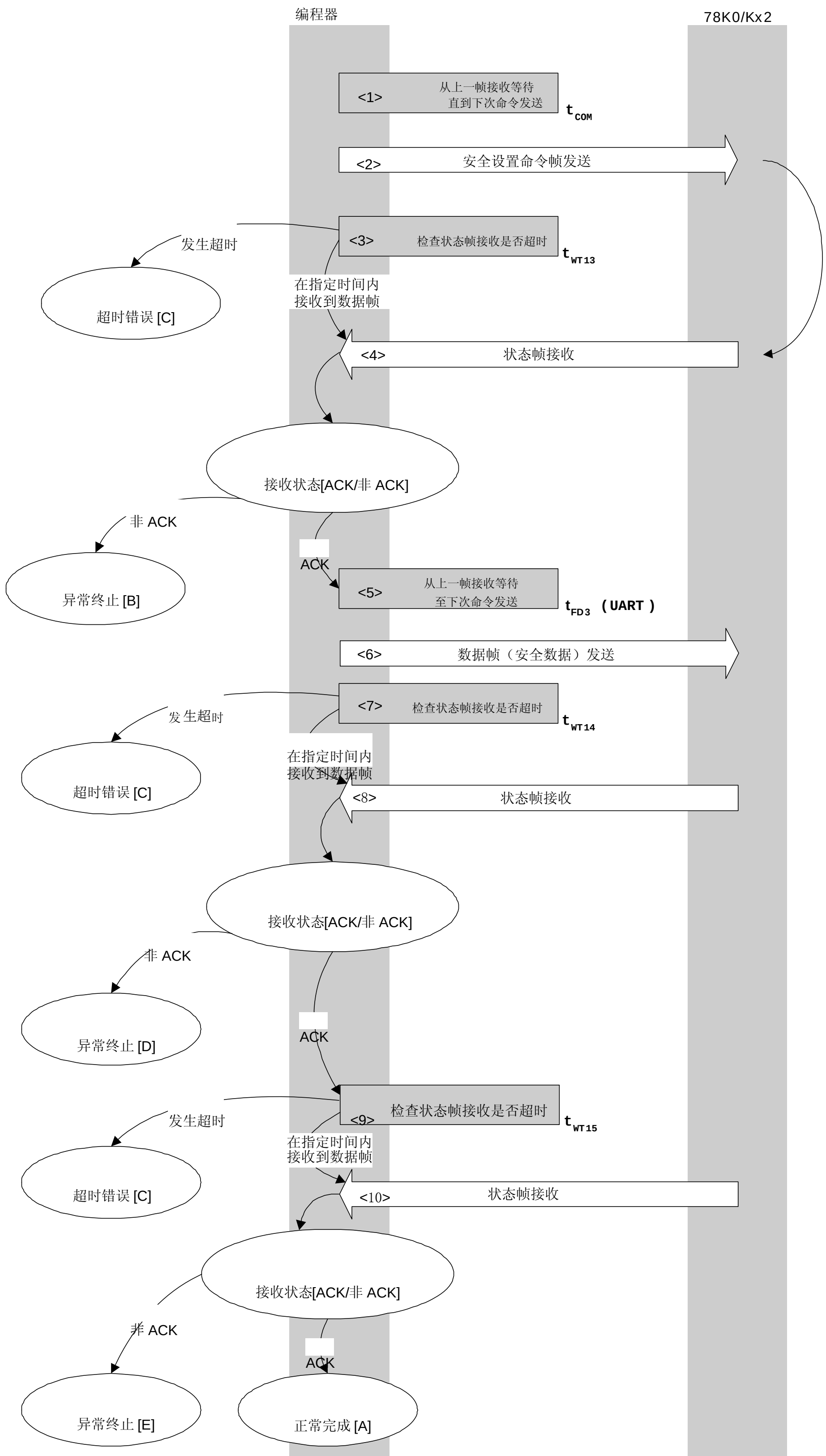
以下所示为校验和命令处理的示例程序。

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

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


    /***********************************************/
    /*   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"命令

    rc = get_sfrm_ua(fl_ua_sfrm, tWT16_TO);   // 获取状态帧
    switch(rc) {
        case    FLC_NO_ERR:                 break;  // 继续
//      case    FLC_DFTO_ERR:       return  rc;     break;  // case [C]
        default:            return  rc;     break;  // case [B]
    }

    /***********************************************/
    /*   获取数据帧（校验和数据）                  */
    /***********************************************/
    rc = get_dfrm_ua(fl_rxdata_frm, tFD1_TO);  // 获取状态帧
    if (rc){                                          // 如果没有错误，
        return rc;                      // case [D]
    }

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

}
```

4.14 安全设置命令

4.14.1　　　处理流程图

安全设置命令处理流程



4.14.2 处理顺序描述

<1>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}）。
<2>　安全设置命令由命令帧发送处理过程进行发送。
<3>　超时检查从命令发送时开始，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT13}）。
<4>　检查状态码。

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

<5>　从上一帧接收开始等待直到下次命令发送（等待时间 t_{COM}(UART)）。
<6>　数据帧（安全设置数据）由数据帧发送处理过程进行发送。
<7>　执行超时检查，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT14}）。
<8>　检查状态码。

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

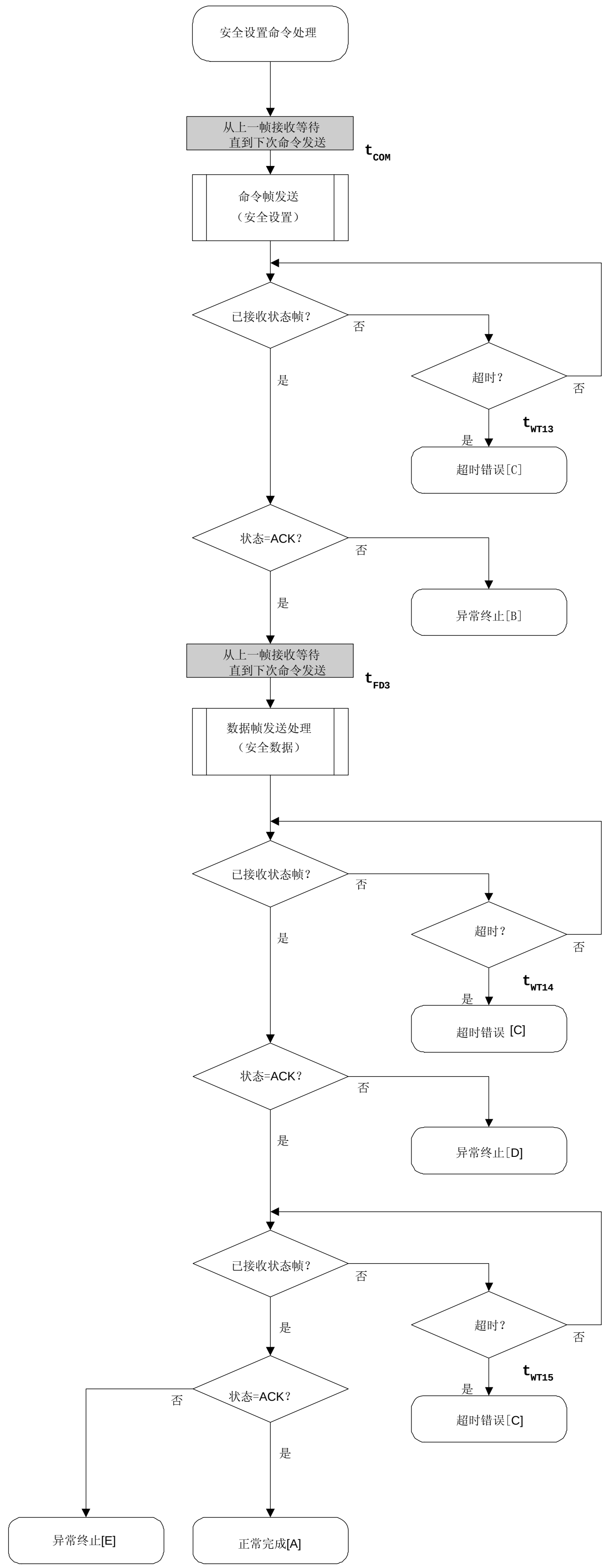
<9>　执行超时检查，直至状态帧接收。
　　如果发生超时，则返回超时错误 [C]（超时时间 t_{WT15}）。
<10> 检查状态码。

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

4.14.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常执行安全设置。
异常终止 [B]	参数错误	05H	命令信息（参数）不是 00H。
	校验和错误	07H	已发送命令帧或数据帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	状态帧接收或数据帧接收发生超时错误。
异常终止 [D]	FLMD 错误	18H	发生写入错误。
	写入错误	1CH	发生写入错误（包括已经设置安全数据的情况）。
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

4.14.4 流程图


安全设置命令处理
从上一帧接收等待
直到下次命令发送
t_{COM}
命令帧发送
（安全设置）
已接收状态帧？
否
超时？
否
是
t_{WT13}
是
超时错误[C]
状态=ACK？
否
异常终止[B]
是
从上一帧接收等待
直到下次命令发送
t_{FD3}
数据帧发送处理
（安全数据）
已接收状态帧？
否
超时？
否
是
t_{WT14}
是
超时错误 [C]
状态=ACK？
否
异常终止[D]
是
已接收状态帧？
否
超时？
否
是
t_{WT15}
是
超时错误[C]
状态=ACK？
否
是
异常终止[E]
正常完成[A]

4.14.5 示例程序

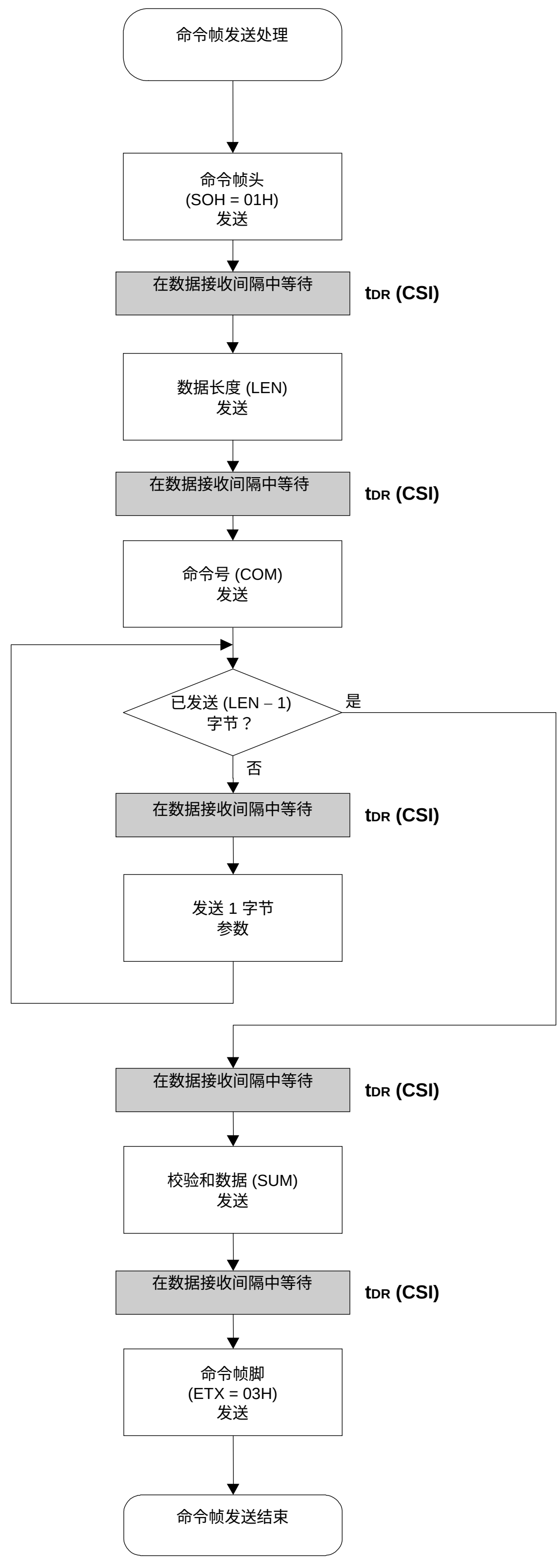
以下所示为安全设置命令处理的示例程序。

```
/****************************************************************/
/*                                                  */
/* 设置安全旗标命令                                       */
/*                                                  */
/****************************************************************/
/* [i] u8 scf      ... 安全旗标数据                         */
/* [r] u16         ... 错误代码                          */
/****************************************************************/
u16             fl_ua_setscf(u8 scf)
{
        u16         rc;

        /**********************************************/
        /*    设置参数                          */
        /**********************************************/
        fl_cmd_prm[0] = 0x00;                       // "BLK" (必须为 0x00)
        fl_cmd_prm[1] = 0x00;                       // "PAG" (必须为 0x00)
        fl_txdata_frm[0] = (scf |= 0b11101000);
                                                // "FLG" (位 7, 6, 5, 3 必须为“1” (务必))

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


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


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

        fl_wait(tFD3_UA);
```

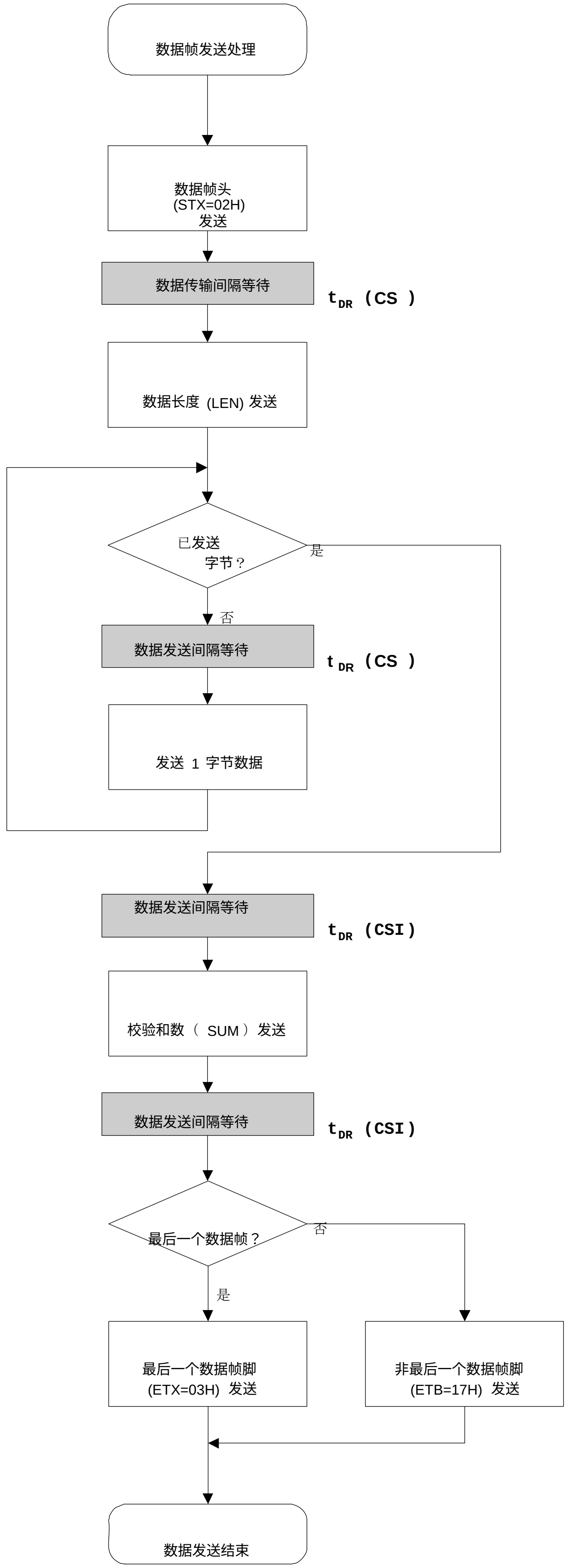
```
        put_dfrm_ua(2, fl_txdata_frm, true);   // 发送安全设置（旗标）与 BOT 数据


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

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

本章流程图中所示的每个符号（t_{XX} 与 t_{WTXX}）均为“**第 6 章 FLASH 存储器编程参数特性**”中项目的符号。
有关每个指定值的信息，请参阅**“第 6 章 FLASH 存储器编程参数特性**”。

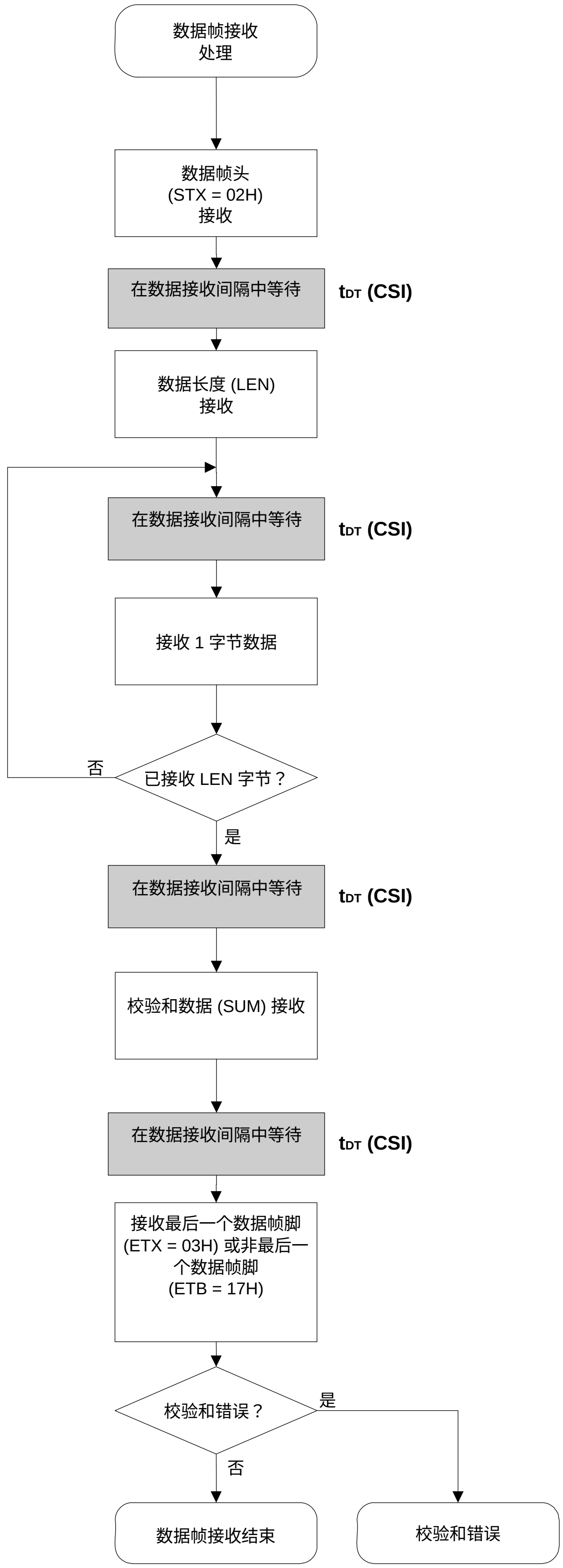
<R>　5.1　命令帧发送处理流程图



<R>　5.2　数据帧发送处理流程图



<R>　5.3　数据帧接收处理流程图

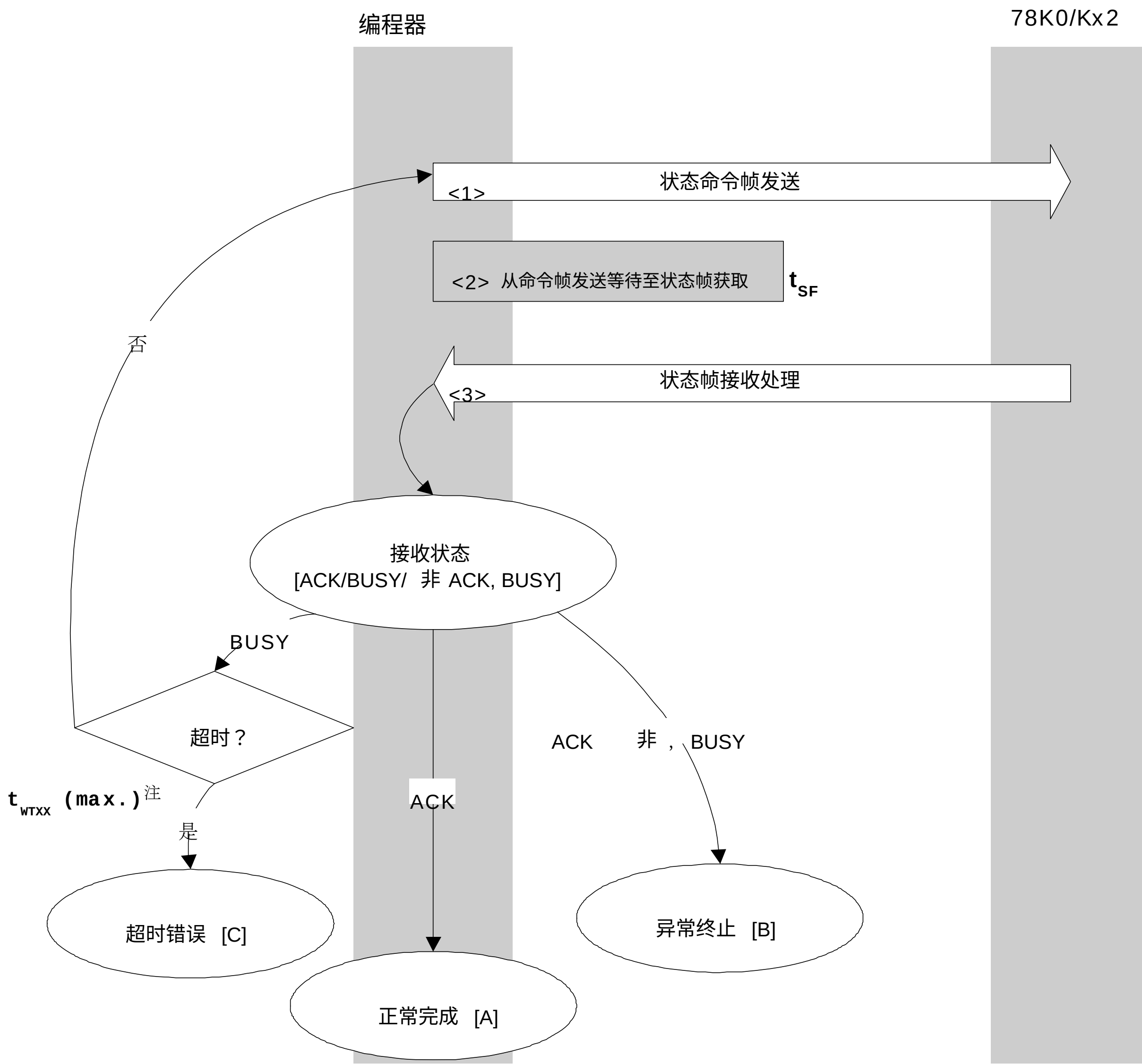


5.4　状态命令

5.4.1　处理流程图

状态命令处理流程

注　　根据执行命令的不同，应用程序参数亦有所不同。



5.4.2　处理顺序描述

<1>　状态命令由命令帧发送处理进行发送。
<2>　从命令发送等待至状态帧接收（等待时间 t_{SF}）。
<3>　检查状态码。

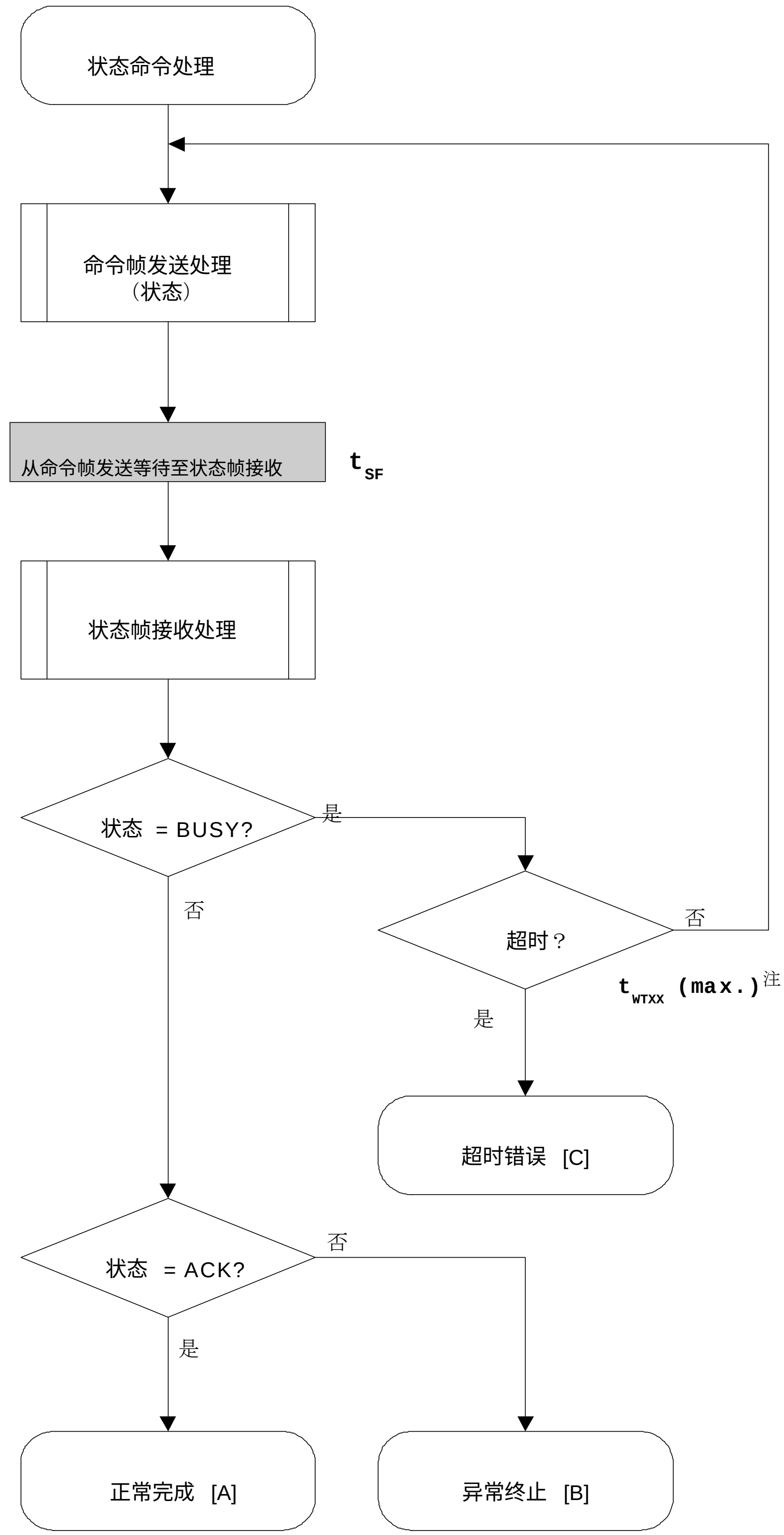
当 ST1 = ACK:　　正常完成 [A]
当 ST1 = BUSY:　　执行超时检查 (t_{WTXX}(MAX.)[注])。
　　如果处理操作未超时，流程将从 <1> 重新执行。
　　如果发生超时，则返回超时错误 [C]。
当 ST1 ≠ ACK, BUSY:　　异常终止 [B]

注　根据执行命令的不同，应用程序参数亦有所不同。

5.4.3　处理完成时状态

处理完成时状态		状态码	说明
正常完成 [A]	正常应答 (ACK)	06H	已正常接收发送自 78K0/Kx2 的状态帧。
异常终止 [B]	命令错误	04H	接收到不支持的命令或异常的帧。
	参数错误	05H	命令信息（参数）无效。
	校验和错误	07H	发送自编程器的帧的数据发生异常。
	验证错误	0FH	发送自编程器的帧的数据发生验证错误。
	保护错误	10H	已尝试执行受安全设置命令保护的进程。
	否定应答 (NACK)	15H	否定应答
	读取错误	20H	读取安全信息失败。
	MRG10　错误	1AH	发生擦除错误。
	MRG11 错误	1BH	数据写入期间发生内部验证错误，或者发生空白检查错误。
<R>	写入错误	1CH	发生写入错误。
超时错误 [C]		–	命令发送后，指定时间已过但是仍返回 BUSY 响应。

5.4.4　流程图



注　根据执行命令的不同，应用程序参数亦有所不同。

5.4.5　示例程序

以下所示为状态命令处理的示例程序。

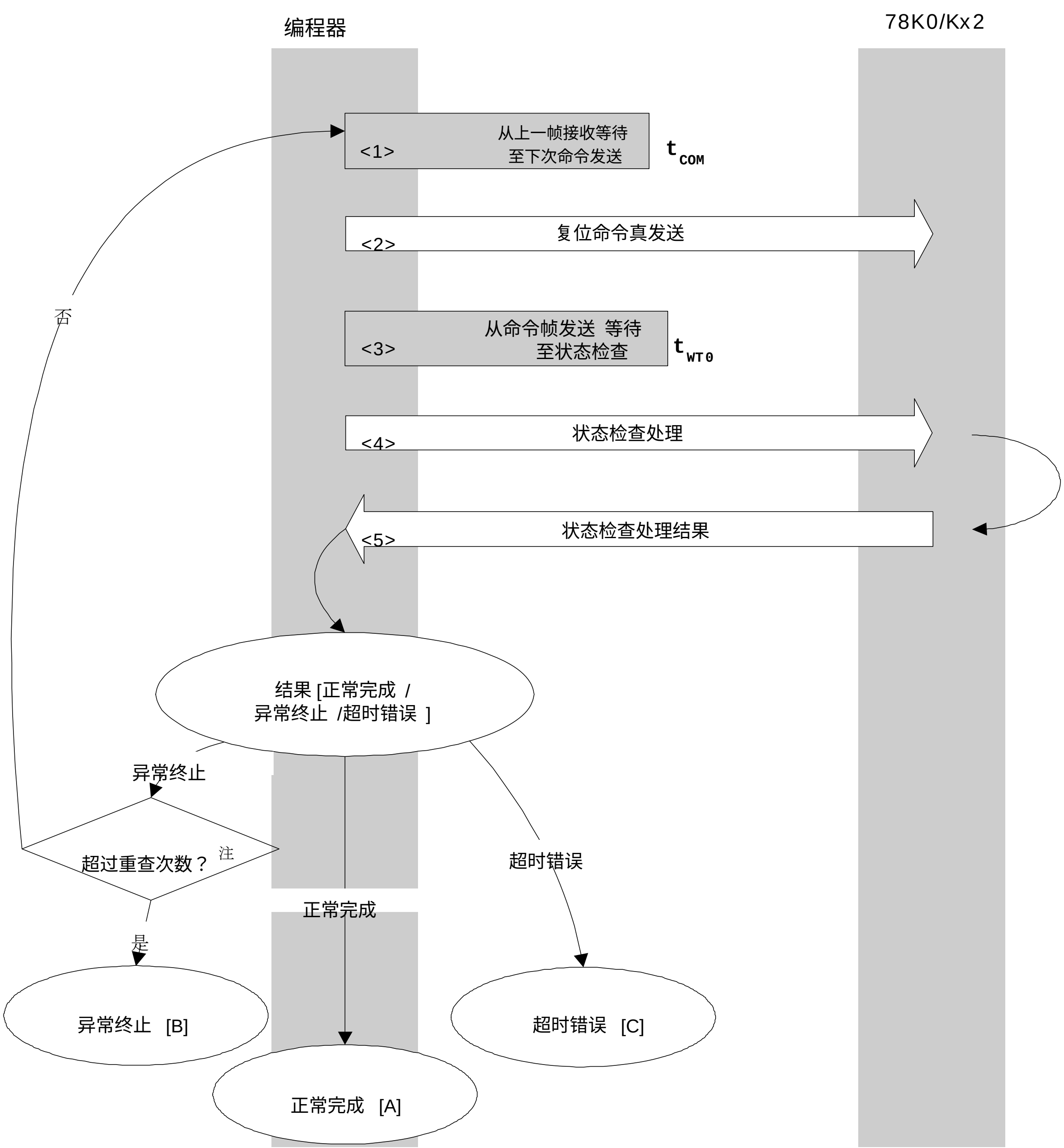
```
/***************************************************************/
/*                                                 */
/* 获取状态命令 (CSI)                                 */
/*                                                 */
/***************************************************************/
/* [r] u16        ... 已解码状态或错误代码                    */
/*                                                 */
/* （请参阅 fl.c 中 fl.h/fl-proto.h 与                  */
/*       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 // case [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// case [C]
                        continue;                 //否，重试
                }
                break;                    // ACK 或其他错误（除 BUSY 之外）
        }
```

```
        rc = decode_status(fl_st1); // 解码状态以返回代码
//      switch(rc) {
//
//              case    FLC_NO_ERR:   return   rc;      break;   // case [A]
//              default:              return   rc;      break;   // case [B]
//      }
        return rc;

}
```

5.5　复位命令

5.5.1　处理流程图

复位命令处理流程图



注　请勿超过复位命令发送的重查次数（最多 16 次）。

5.5.2　处理流程描述

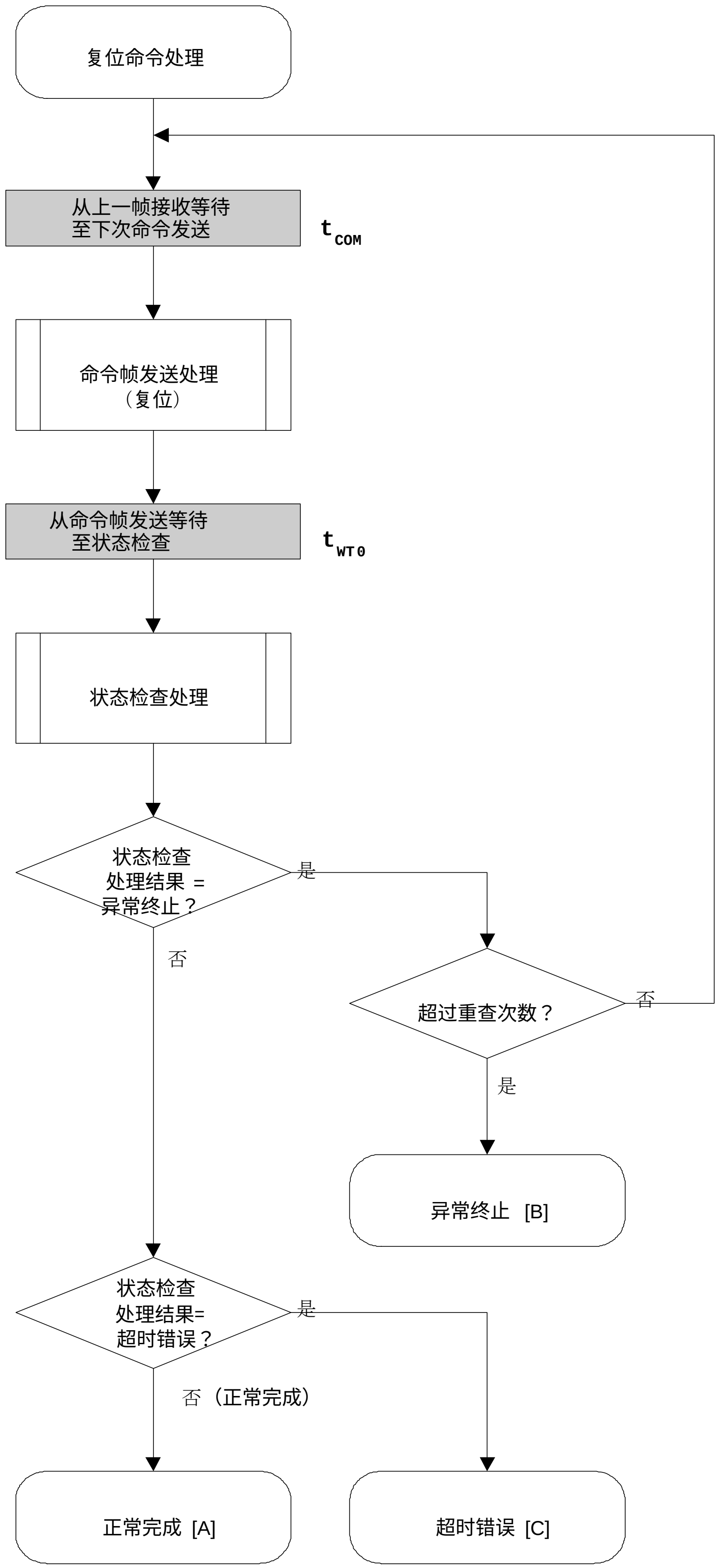
<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　复位命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT0}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	正常完成 [A]
当处理异常结束时：	如果未超过重查次数，该流程从 <1> 开始重新执行。 如果超过重查次数，该处理异常结束 [B]。
当发生超时错误时：	返回超时错误 [C]。

5.5.3　处理完成时状态

处理完成时状态		状态码	说明
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已建立编程器与 78K0/Kx2 之间的同步。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	状态检查处理由于超时而终止。

5.5.4　流程图


复位命令处理
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧发送处理
（复位）
从命令帧发送等待
至状态检查
t_{WT0}
状态检查处理
状态检查
处理结果 =
异常终止？
是
否
超过重查次数？
否
是
异常终止　[B]
状态检查
处理结果=
超时错误？
是
否（正常完成）
正常完成 [A]
超时错误 [C]

5.5.5　示例程序

以下所示为复位命令处理的示例程序。

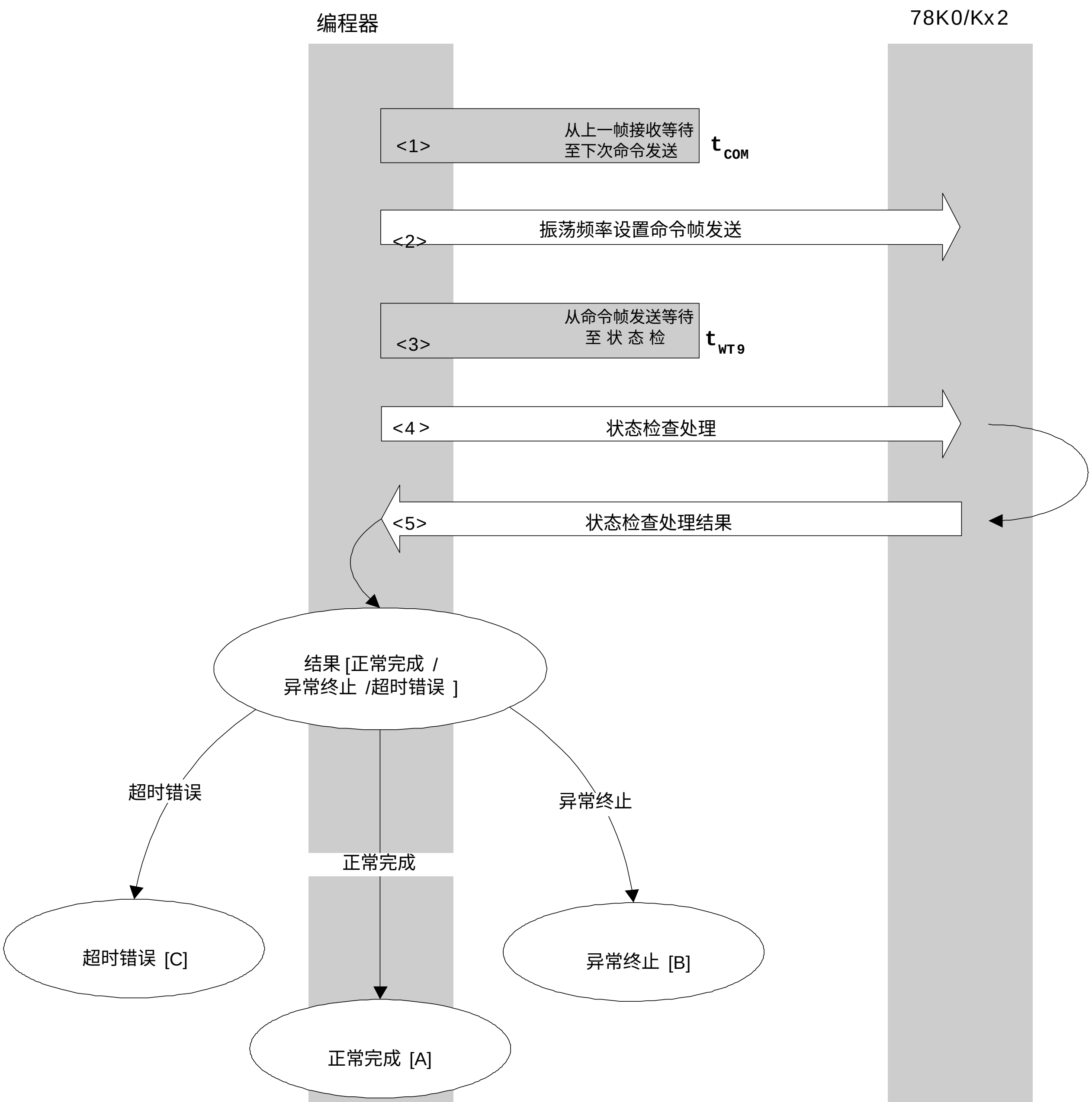
```
/***************************************************************/
/*                                                             */
/* 复位命令 (CSI)                                              */
/*                                                             */
/***************************************************************/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

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

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

                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_TO);         // 获取状态

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

5.6　振荡频率设置命令

CSI 通讯期间，执行该命令并非必需的步骤（如果 CSI 通讯期间要求根据编程器规范执行该命令，请将频率设置为 8 MHz）。

5.6.1　处理流程图

振荡频率设置命令处理流程



5.6.2　处理流程描述

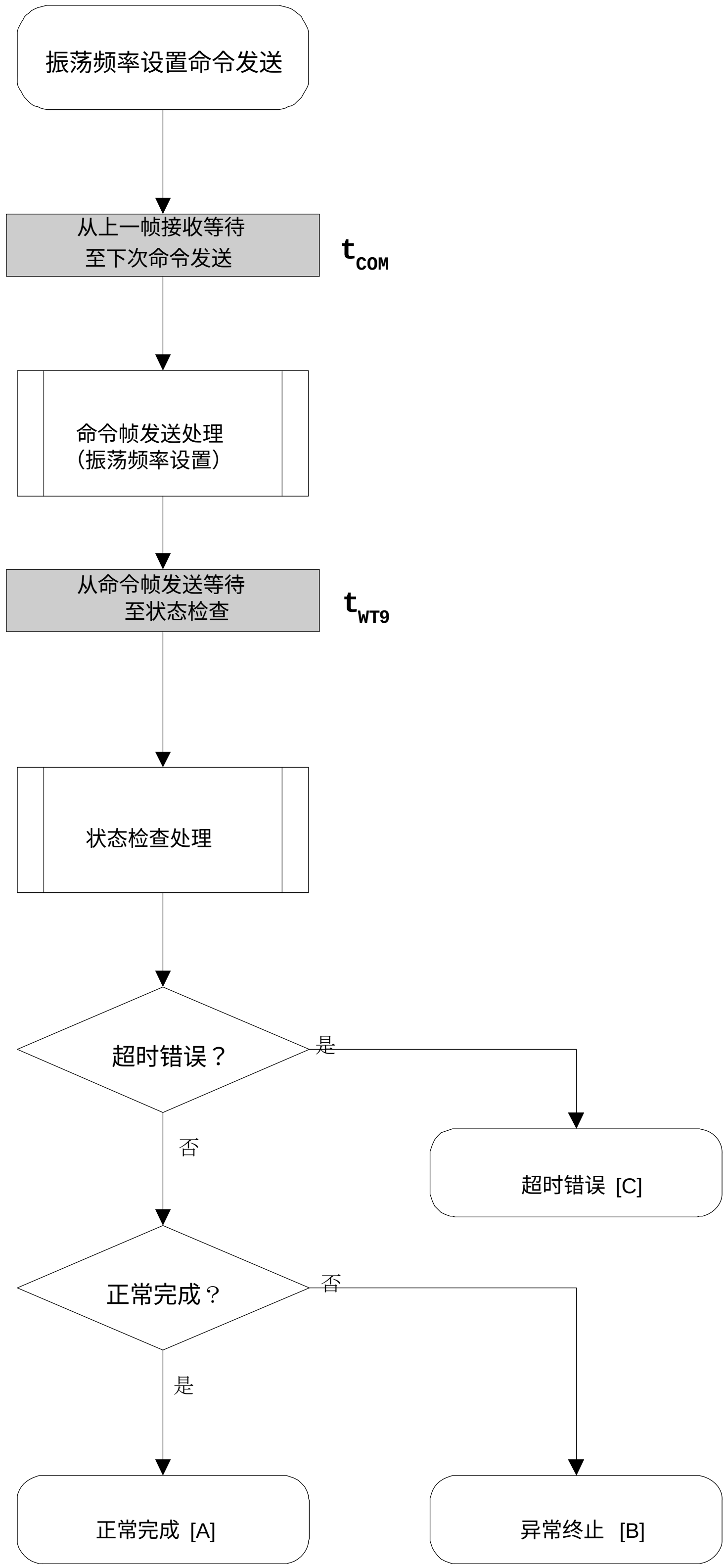
<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　振荡频率设置命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT9}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	正常完成 [A]
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误 [C]。

5.6.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已将操作频率正确置入 78K0/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	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。

5.6.4　流程图


振荡频率设置命令发送
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧发送处理
（振荡频率设置）
从命令帧发送等待
至状态检查
t_{WT9}
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
正常完成 [A]
异常终止 [B]

5.6.5　示例程序

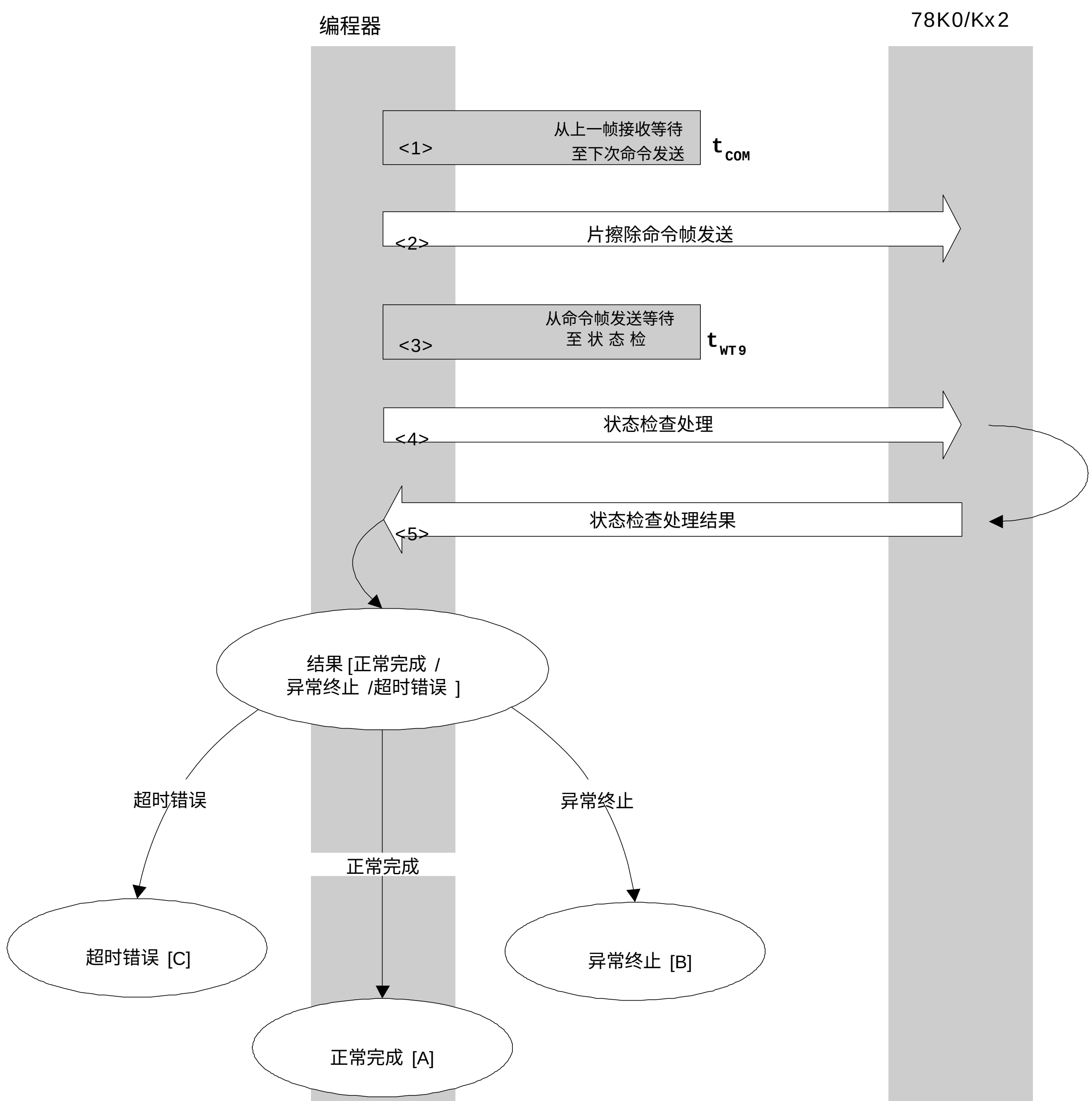
以下所示为振荡频率设置命令处理的示例程序。

```
/**************************************************************/
/*                                                            */
/* 设置 Flash 设备时钟值命令 (CSI)                              */
/*                                                            */
/**************************************************************/
/* [i] u8 clk[4]   ... 频率数据 (D1-D4)                         */
/* [r] u16         ... 错误代码                                 */
/**************************************************************/
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);                          // 发送命令帧前等待

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

5.7　片擦除命令

5.7.1　处理流程图

片擦除命令处理流程



5.7.2　处理流程描述

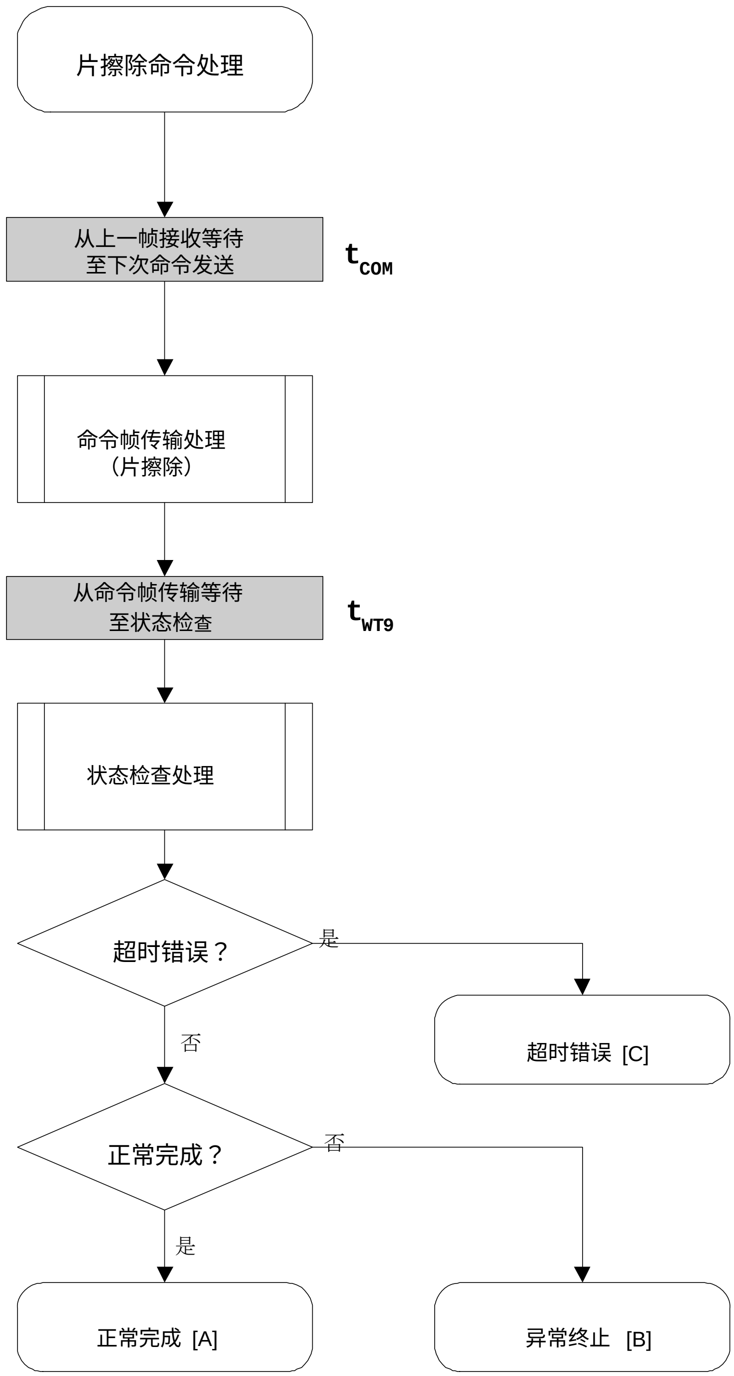
<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　片擦除命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT1}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	正常完成 [A]
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误 [C]。

5.7.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常执行片擦除。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	保护错误	10H	片擦除受安全设置的保护。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	擦除错误	1AH	发生擦除错误。
超时错误 [C]		–	未在指定时间内接收到状态帧。

5.7.4　流程图


片擦除命令处理
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧传输处理
（片擦除）
从命令帧传输等待
至状态检查
t_{WT9}
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成？
否
是
正常完成 [A]
异常终止 [B]

5.7.5　示例程序

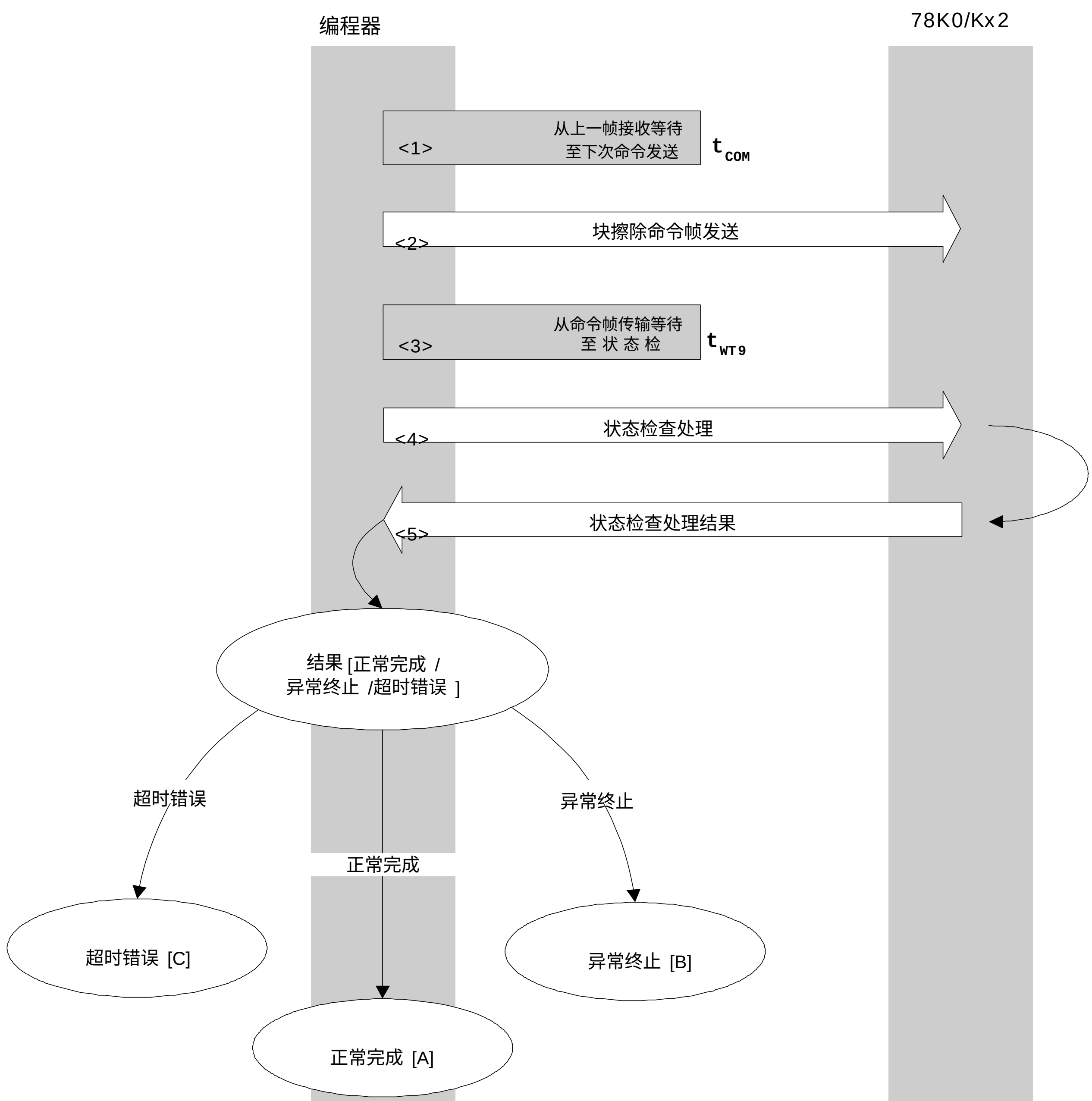
以下所示为片擦除命令处理的示例程序。

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


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

        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;    // case [A]
//              case    FLC_DFTO_ERR:           return    rc;       break;    // case [C]
//              default:              return    rc;       break;    // case [B]
//      }
        return rc;

}
```

5.8　块擦除命令

5.8.1　处理流程图

块擦除命令处理流程



5.8.2　处理流程描述

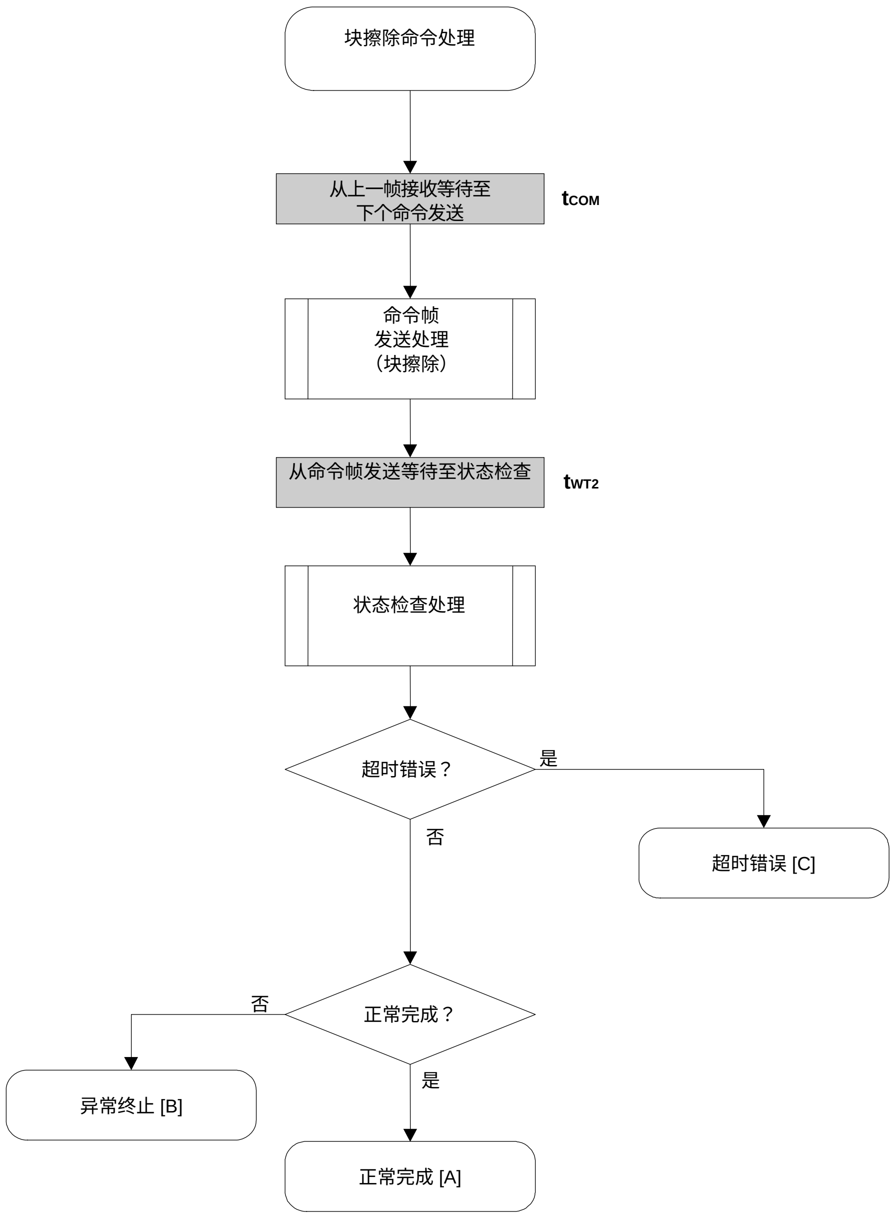
<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　块擦除命令由命令帧发送处理进行发送。
<3>　等待至状态帧获取（等待时间 t_{WT2}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	正常完成 [A]
当处理异常结束时：	异常终止 [B]
当发生超时错误时：	返回超时错误 [C]。

5.8.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常执行块擦除。
异常终止 [B]	参数错误	05H	块数目超出范围。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	保护错误	10H	写入、块擦除或片擦除受安全设置的保护。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	擦除错误	1AH	发生擦除错误。
超时错误 [C]		–	未在指定时间内接收到状态帧。

5.8.4　流程图


块擦除命令处理
从上一帧接收等待至
下个命令发送
tCOM
命令帧
发送处理
（块擦除）
从命令帧发送等待至状态检查
tWT2
状态检查处理
超时错误？
是
否
超时错误 [C]
正常完成？
否
是
异常终止 [B]
正常完成 [A]

5.8.5　示例程序

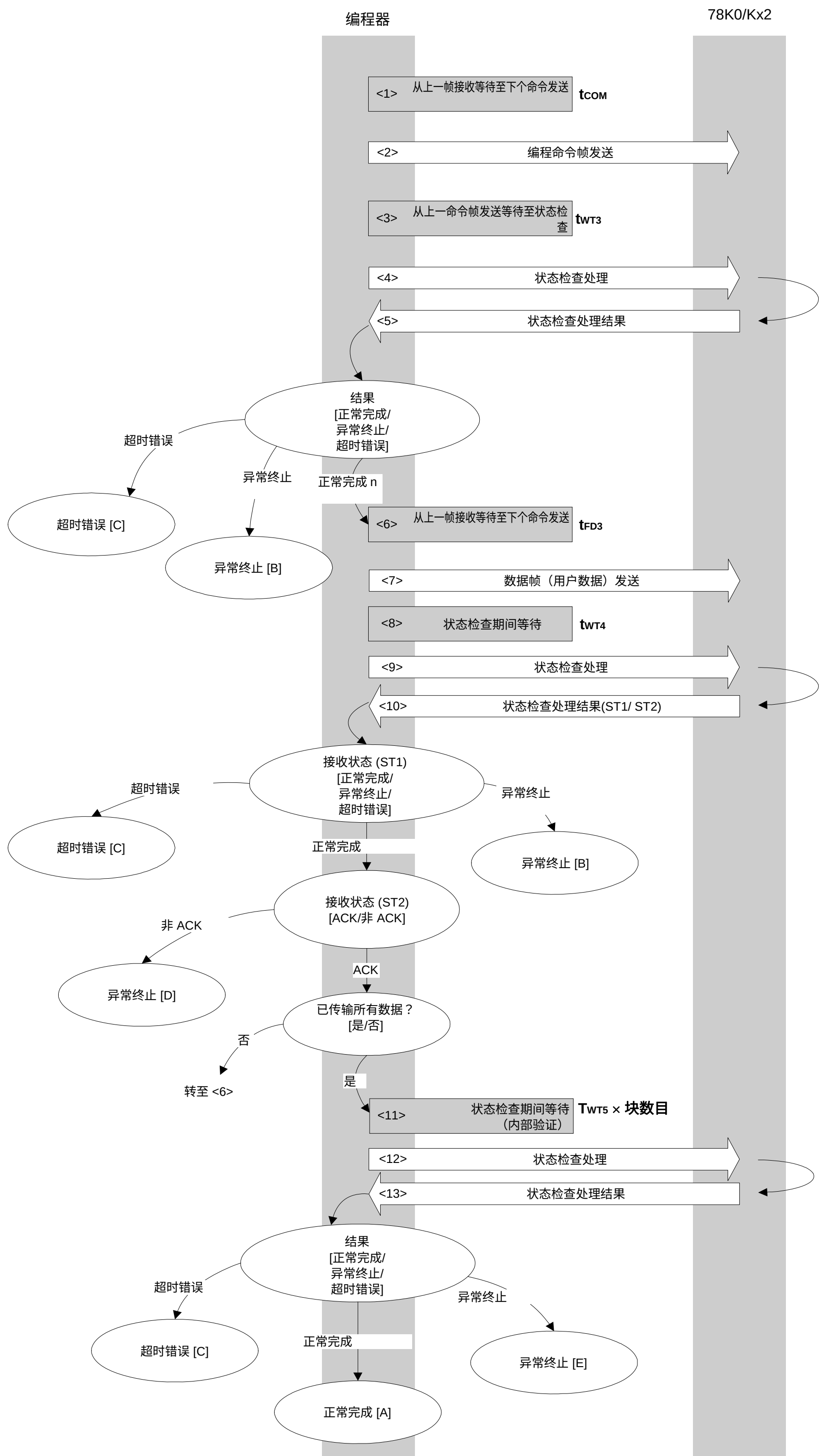
以下所示为块擦除命令处理的示例程序。

```
/****************************************************************/
/*                                                              */
/* 擦除块命令 (CSI)                                             */
/*                                                              */
/****************************************************************/
/* [i] u16 sblk   ... 要擦除的起始块 (0...255)                  */
/* [i] u16 eblk  ... 要擦除的结束块   (0...255)                 */
/* [r] u16       ... 错误代码                                   */
/****************************************************************/
u16             fl_csi_erase_blk(u16 sblk, u16 eblk)
{

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

        top = get_top_addr(sblk);               // 获取起始块的起始地址
        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

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

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

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

        fl_wait(wt2);


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

5.9　编程命令

5.9.1　处理流程图

编程命令处理流程



5.9.2　处理流程描述

<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　编程命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT3}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	转至 <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>　等待至下一个数据帧发送（等待时间 t_{FD3}）。
<7>　要写入 78K0/Kx2 Flash 存储器的用户数据由数据帧发送处理进行发送。
<8>　从数据帧（用户数据）发送等待至状态检查处理（等待时间 t_{WT4}）。
<9>　状态帧由状态检查处理进行获取。
<10> 根据状态检查处理（状态码 (ST1/ST2)）的结果执行以下处理（也可参阅处理流程图）。

当 ST1 = 异常终止： 异常终止 [B]
当 ST1 = 超时错误： 返回超时错误 [C]。
当 ST1 = 正常完成： 以下处理将按照 ST2 值进行操作。

- 当 ST2 ≠ ACK:　异常终止 [D]
- 当 ST2 = ACK:　所有用户数据发送完成后，转至 <11>。
  如果仍有用户数据未发送，该流程将从 <6> 重新执行该顺序。

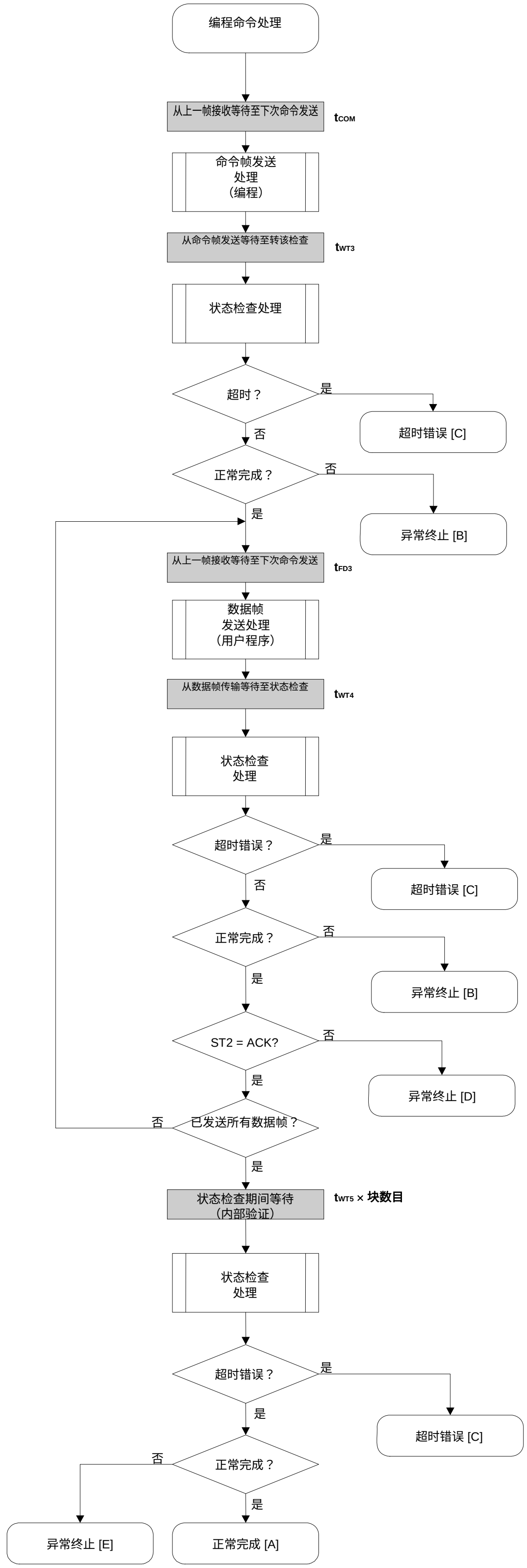
<11> 等待至状态检查处理（超时时间 t_{WT5} × 块数目）。
<12> 状态帧由状态检查处理进行获取。
<13> 根据状态检查处理的结果执行以下处理。

当处理正常结束时：	正常完成 [A] （指明写入完成后已正常执行内部验证检查）
当处理异常结束时：	异常终止 [E] （指明写入完成后未正常执行内部验证检查）
当发生超时错误时：	返回超时错误 [C]。

5.9.3　处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常写入用户数据。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者不是 8 的倍数。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	保护错误	10H	写入操作受安全设置的保护。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
异常终止 [D]	写入错误	1CH (ST2)	发生写入错误。
异常终止 [E]	MRG11 错误	1BH	发生内部验证错误。

5.9.4　流程图


编程命令处理
从上一帧接收等待至下次命令发送
tCOM
命令帧发送处理（编程）
从命令帧发送等待至转该检查
tWT3
状态检查处理
超时？
是
超时错误 [C]
否
正常完成？
否
异常终止 [B]
是
从上一帧接收等待至下次命令发送
tFD3
数据帧发送处理（用户程序）
从数据帧传输等待至状态检查
tWT4
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成？
否
异常终止 [B]
是
ST2 = ACK?
否
异常终止 [D]
是
已发送所有数据帧？
否
是
状态检查期间等待（内部验证）
tWT5 × 块数目
状态检查处理
超时错误？
是
超时错误 [C]
是
正常完成？
否
异常终止 [E]
是
正常完成 [A]

5.9.5　示例程序

以下所示为编程命令处理的示例程序。

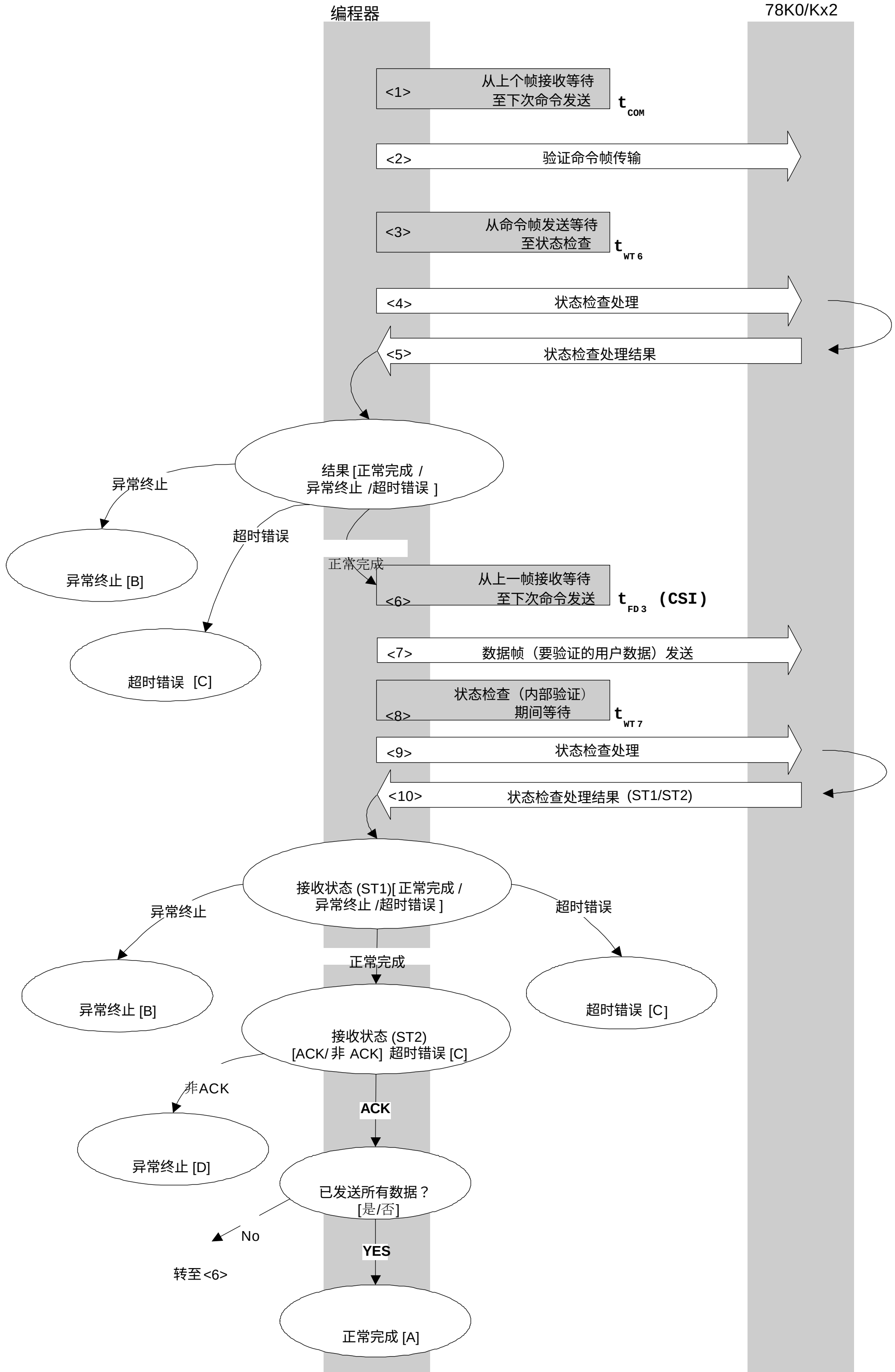
```
/*************************************************************/
/*                                                           */
/* 写入命令 (CSI)                                            */
/*                                                           */
/*************************************************************/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

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

        block_num = get_block_num(top, bottom);     // 获取块数目

        /**********************************************/
        /*      发送命令与检查状态                    */
        /**********************************************/

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

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

        while(1){

                if ((bottom - send_head) > 256){    // 其余大小 > 256 ?
                        is_end = false;             // 是，非结束帧
```

```
                send_size  = 256;               // 发送大小 = 256 字节
            }
            else{
                is_end = true;
                send_size = bottom - send_head + 1;
                                                // 发送大小 = (bottom - send_head)+1 字节
            }

            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                                                                // 设置有效数据帧
            send_head += send_size;


            fl_wait(tFD3_CSI);          // 发送数据帧前等待
            put_dfrm_csi(send_size, fl_txdata_frm, is_end);
                                                        // 发送数据帧（用户数据）
            fl_wait(tWT4);                              // 等待


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

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

        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;  // case [A]
//          case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
//          default:        return  rc;     break;  // case [E]
//      }
        return  rc;
}
```

5.10 验证命令

5.10.1 处理流程图

验证命令处理流程



5.10.2 处理流程描述

<1> 从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2> 验证命令由命令帧发送处理进行发送。
<3> 从命令发送等待至状态检查处理（等待时间 t_{WT6}）。
<4> 状态帧由状态检查处理进行获取。
<5> 根据状态检查处理的结果执行以下处理。

当处理正常结束时：　转至 <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]。

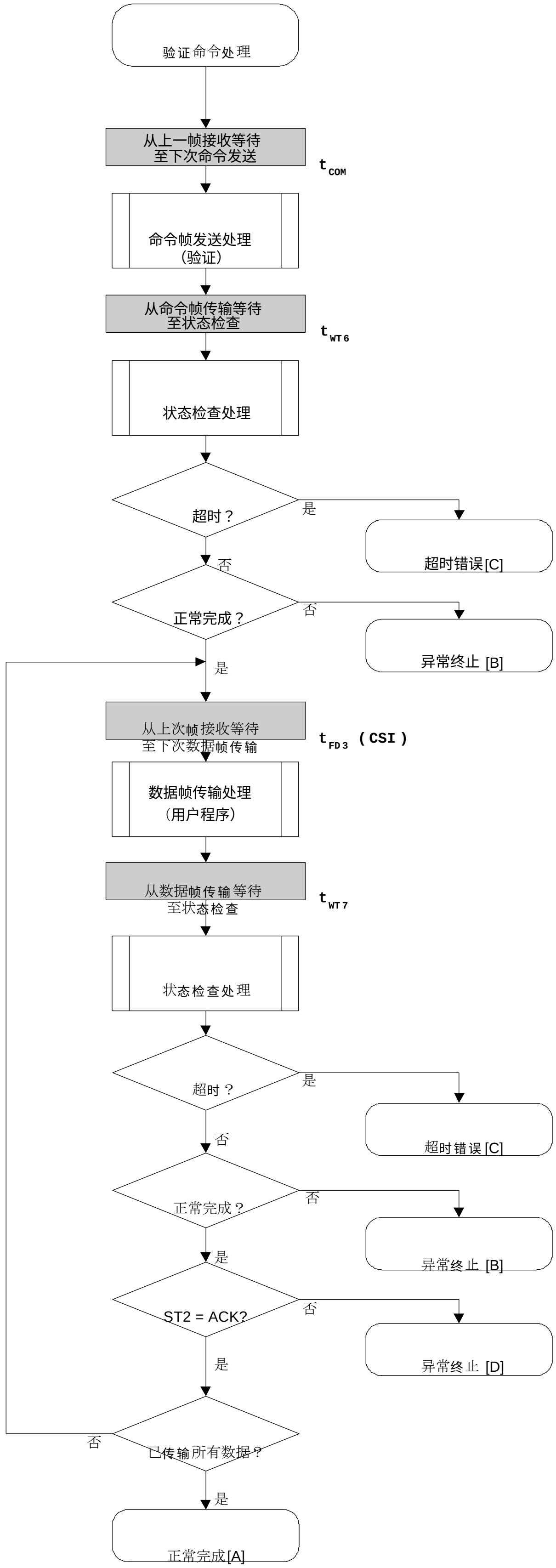
<6> 从上一帧接收等待至下次数据帧发送（等待时间 t_{FD3}）。
<7> 要验证的用户数据由数据帧发送处理进行发送。
<8> 从数据帧发送等待至状态检查处理（等待时间 t_{WT7}）。
<9> 状态帧由状态检查处理进行获取。
<10> 根据状态检查处理（状态码 (ST1/ST2)）的结果执行以下处理（也可参阅处理流程图）。

当 ST1 = 异常终止：异常终止 [B]
当 ST1 = 超时错误：返回超时错误 [C]。
当 ST1 = 正常完成：以下处理将按照 ST2 值进行操作。
- 当 ST2 ≠ ACK:　异常终止 [D]
- 当 ST2 = ACK:　如果所有数据帧发送完成，则处理将正常结束 [A]。
  如果仍有数据帧未发送，该流程将从 <6> 重新执行该顺序。

5.10.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常完成验证。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者指定的地址不是 2 KB 单位内的固定地址。
	校验和错误	07H	已发送命令帧或数据帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
异常终止 [D]	验证错误	0FH (ST2)	验证失败或发生其他错误。

5.10.4 流程图


验证命令处理
从上一帧接收等待
至下次命令发送
t_{COM}
命令帧发送处理
（验证）
从命令帧传输等待
至状态检查
t_{WT6}
状态检查处理
超时？
是
超时错误[C]
否
正常完成？
否
异常终止 [B]
是
从上次帧接收等待
至下次数据帧传输
t_{FD3}（CSI）
数据帧传输处理
（用户程序）
从数据帧传输等待
至状态检查
t_{WT7}
状态检查处理
超时？
是
超时错误 [C]
否
正常完成？
否
异常终止 [B]
是
ST2 = ACK?
否
异常终止 [D]
是
已传输所有数据？
否
是
正常完成[A]

5.10.5 示例程序

以下所示为验证命令处理的示例程序。

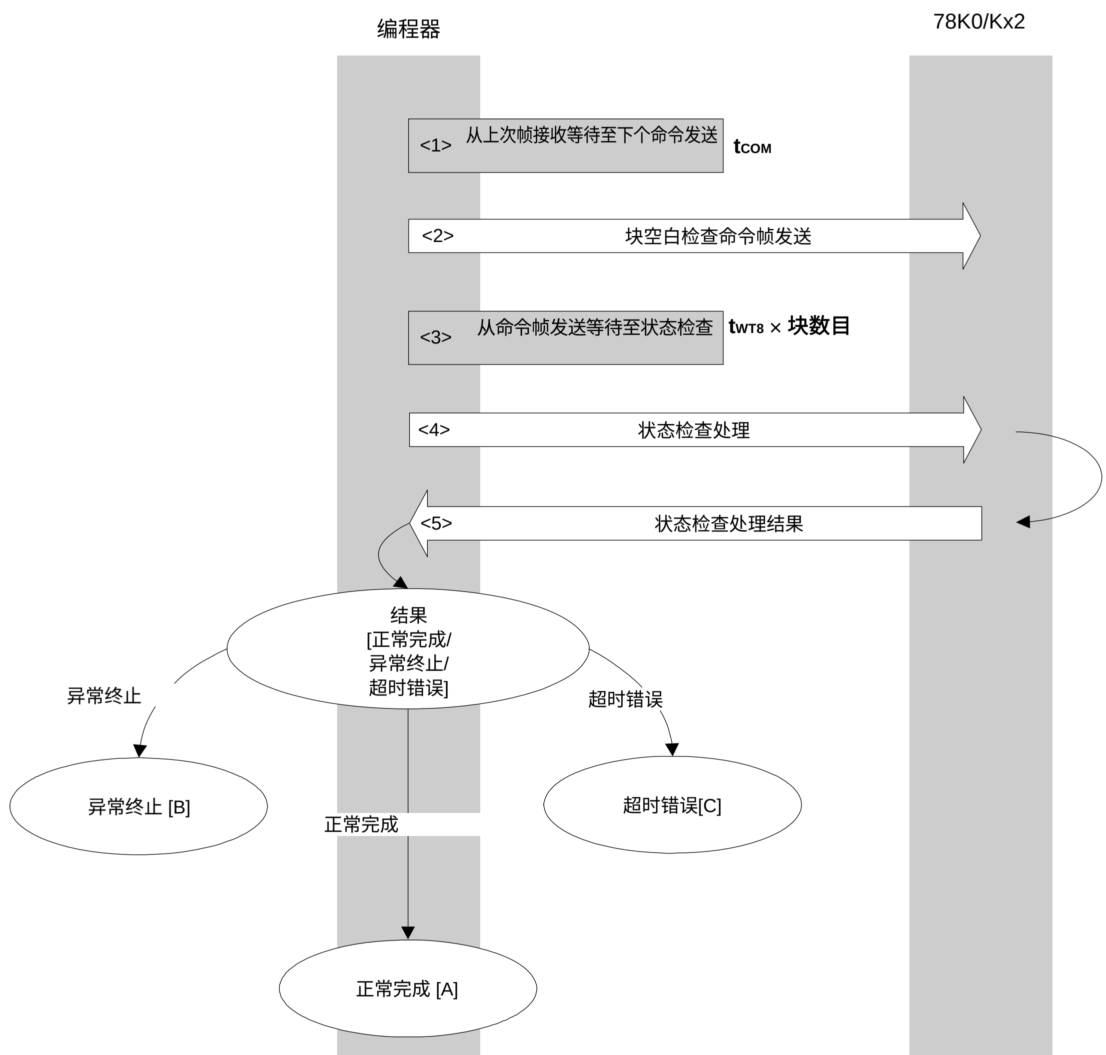
```
/***************************************************************/
/*                                                             */
/* 验证命令 (CSI)                                               */
/*                                                             */
/***************************************************************/
/* [i] u32 top     ... 起始地址                                 */
/* [i] u32 bottom  ... 结束地址                                 */
/* [r] u16         ... 错误代码                                 */
/***************************************************************/
u16             fl_csi_verify(u32 top, u32 bottom)
{
        u16     rc;
        u32     send_head, send_size;
        bool    is_end;

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


        /**********************************************/
        /*      发送命令与检查状态                      */
        /**********************************************/
        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_TO);                  // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                      break;   // 继续
        //      case    FLC_DFTO_ERR:           return   rc;      break;   // case [C]
                default:        return  rc;      break;  // case [B]
        }


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

        while(1){

                if ((bottom - send_head) > 256){         // 其余大小 > 256 ?
                        is_end = false;                  // 是，非结束帧
                        send_size  = 256;                // 发送大小 = 256 字节
                }
```

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

            }

            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);        // 设置数据
                                                                                    // 帧净荷
            send_head += send_size;

            fl_wait(tFD3_CSI);                                      // 发送数据帧前等待
            put_dfrm_csi(send_size, fl_txdata_frm, is_end);         // 发送数据帧
            fl_wait(tWT7);                                  // 等待

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

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

        }
        return      FLC_NO_ERR;     // case [A]

}
```

5.11 块空白检查命令

5.11.1 处理流程图

块空白检查命令处理流程图



5.11.2 处理流程描述

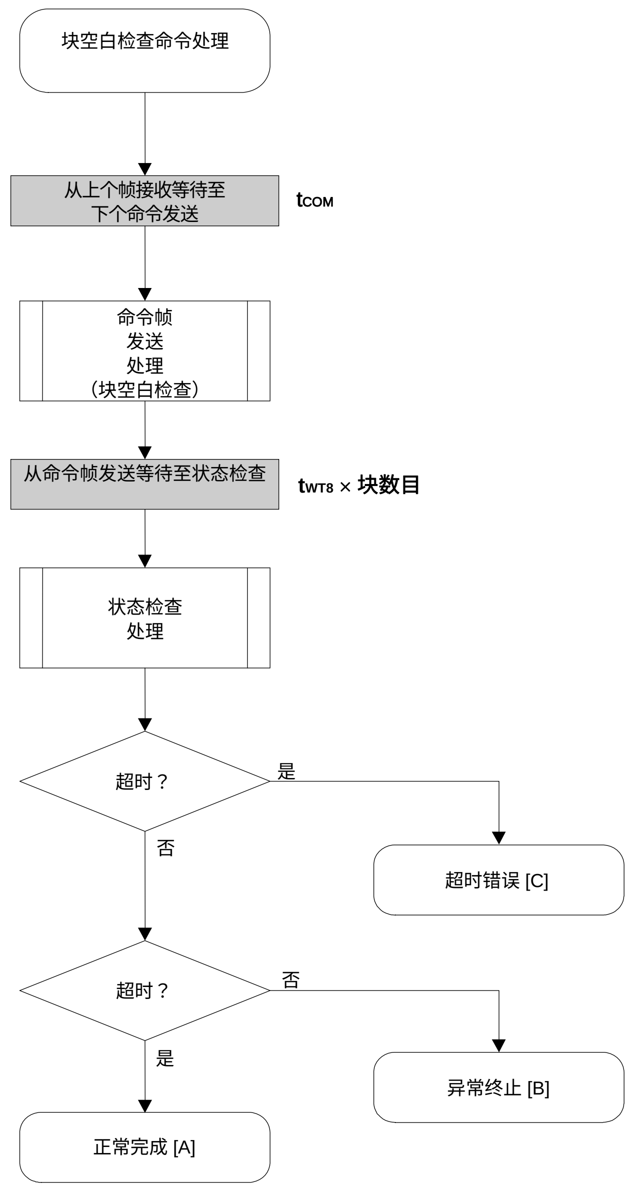
<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　块空白检查命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 $t_{WT8}\times$ 块数目）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

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：	正常完成 [A]

5.11.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且所有指定块为空白。
异常终止 [B]	参数错误	05H	块数目超出范围。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	MRG11 错误	1BH	Flash 存储器内指定的块不为空白。
超时错误 [C]		–	未在指定时间内接收到状态帧。

5.11.4 流程图


块空白检查命令处理
从上个帧接收等待至
下个命令发送
tCOM
命令帧
发送
处理
（块空白检查）
从命令帧发送等待至状态检查
tWT8 × 块数目
状态检查
处理
超时？
是
否
超时错误 [C]
超时？
否
是
异常终止 [B]
正常完成 [A]

5.11.5 示例程序

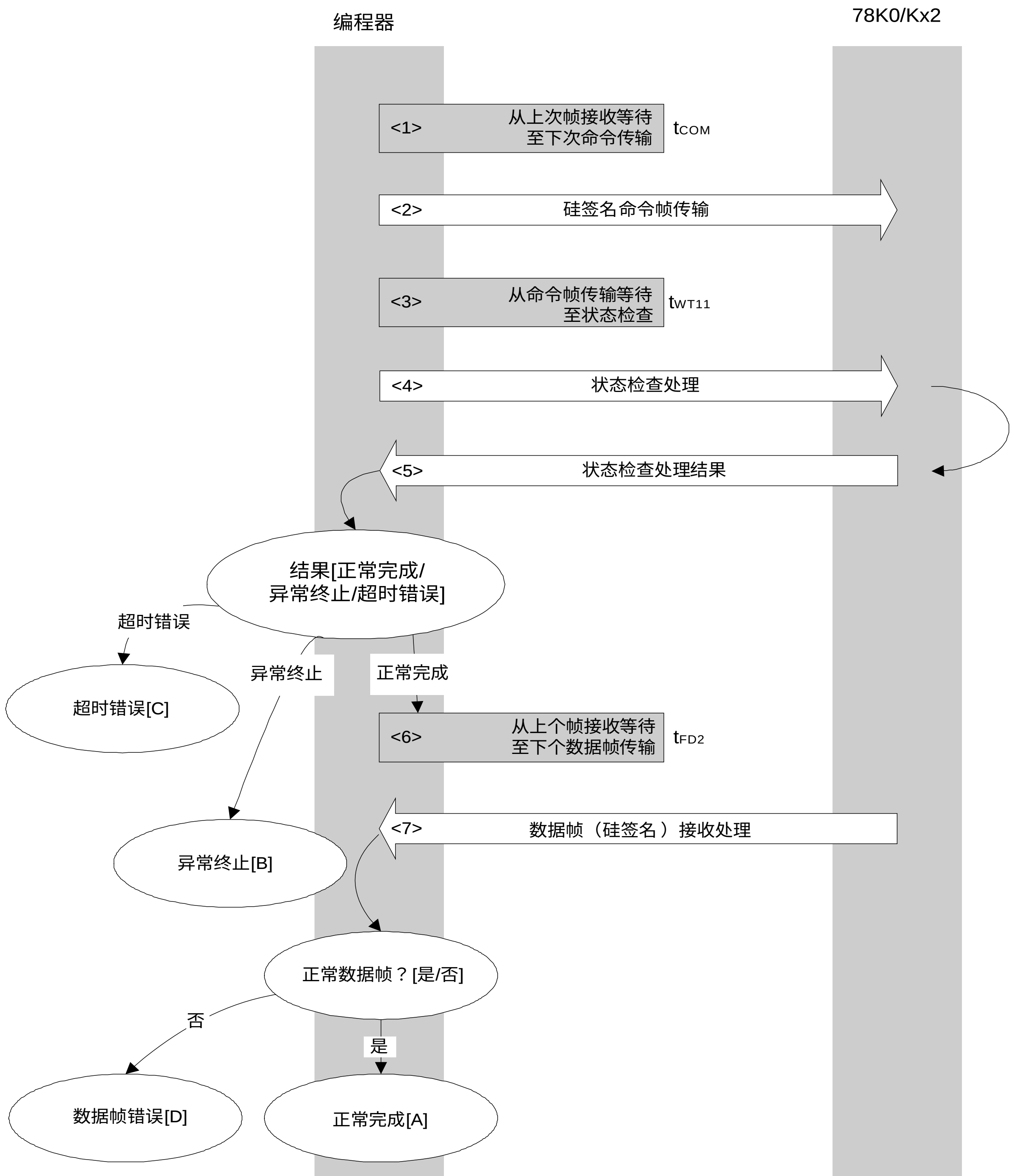
以下所示为块空白检查命令处理的示例程序。

```
/************************************************************/
/*                                                          */
/* 块空白检查命令 (CSI)                                     */
/*                                                          */
/************************************************************/
/* [i] u32 top     ... 起始地址                             */
/* [i] u32 bottom  ... 结束地址                             */
/* [r] u16         ... 错误代码                             */
/************************************************************/
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);     // 获取块数目

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

        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;  // case [A]
//              case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
//              default:        return  rc;     break;  // case [B]
//      }
        return rc;
}
```

5.12 硅签名命令

5.12.1 处理流程图

硅签名命令处理流程



5.12.2 处理流程描述

<1>　从上一次帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　硅签名命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT11}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	转至 <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]。

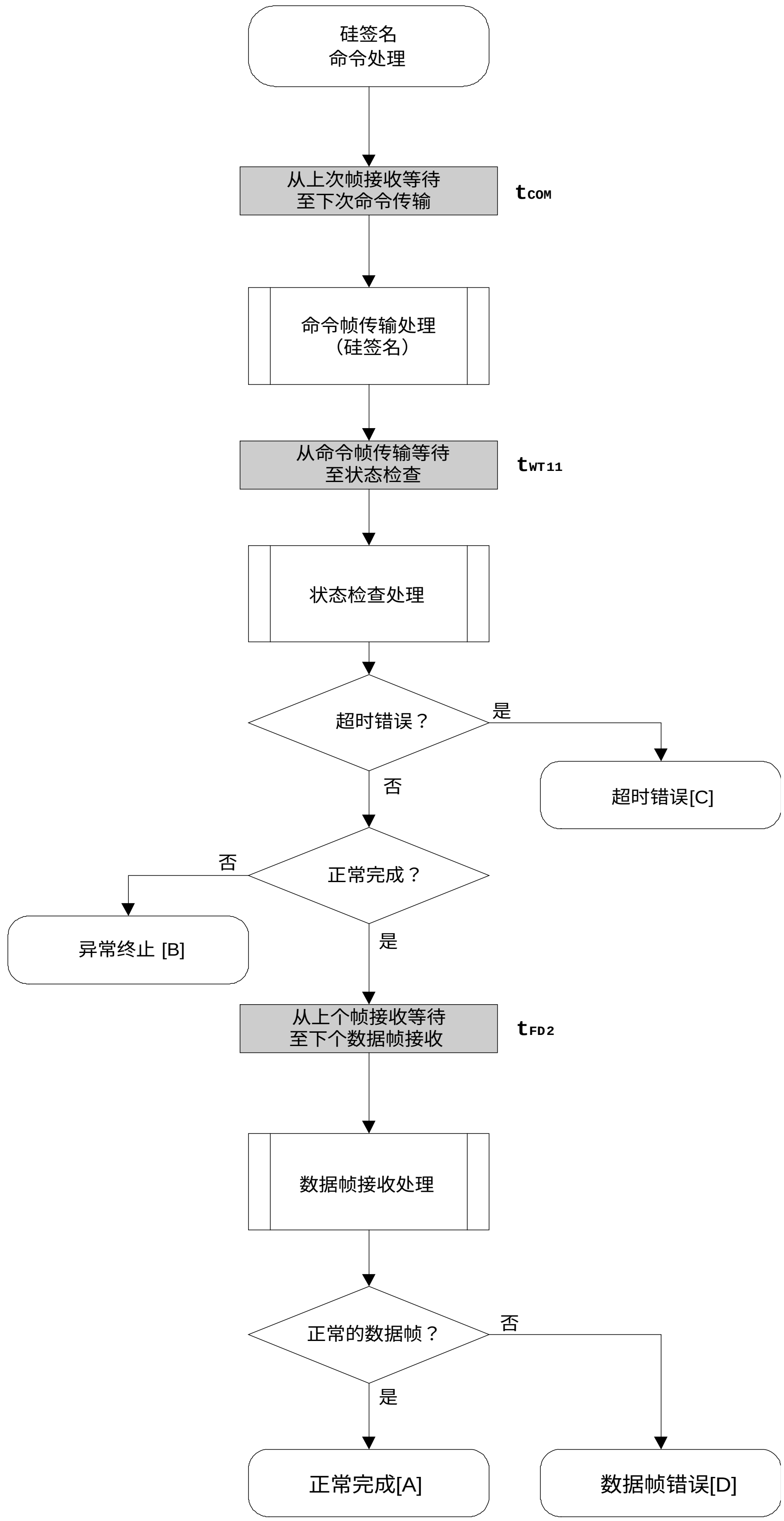
<6>　从上一帧接收等待至下次命令发送（等待时间 t_{FD2}）。
<7>　检查已接收数据帧（硅签名数据）。

如果数据帧正常：	正常完成 [A]
如果数据帧异常：	数据帧错误 [D]

5.12.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取硅签名。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
	读取错误	20H	读取安全信息失败。
超时错误 [C]		–	未在指定时间内接收到状态帧。
数据帧错误 [D]		–	接收到数据帧校验和，因为硅签名数据异常。

5.12.4 流程图


硅签名
命令处理
从上次帧接收等待
至下次命令传输
tCOM
命令帧传输处理
（硅签名）
从命令帧传输等待
至状态检查
tWT11
状态检查处理
超时错误？
是
超时错误[C]
否
正常完成？
否
异常终止 [B]
是
从上个帧接收等待
至下个数据帧接收
tFD2
数据帧接收处理
正常的数据帧？
否
是
正常完成[A]
数据帧错误[D]

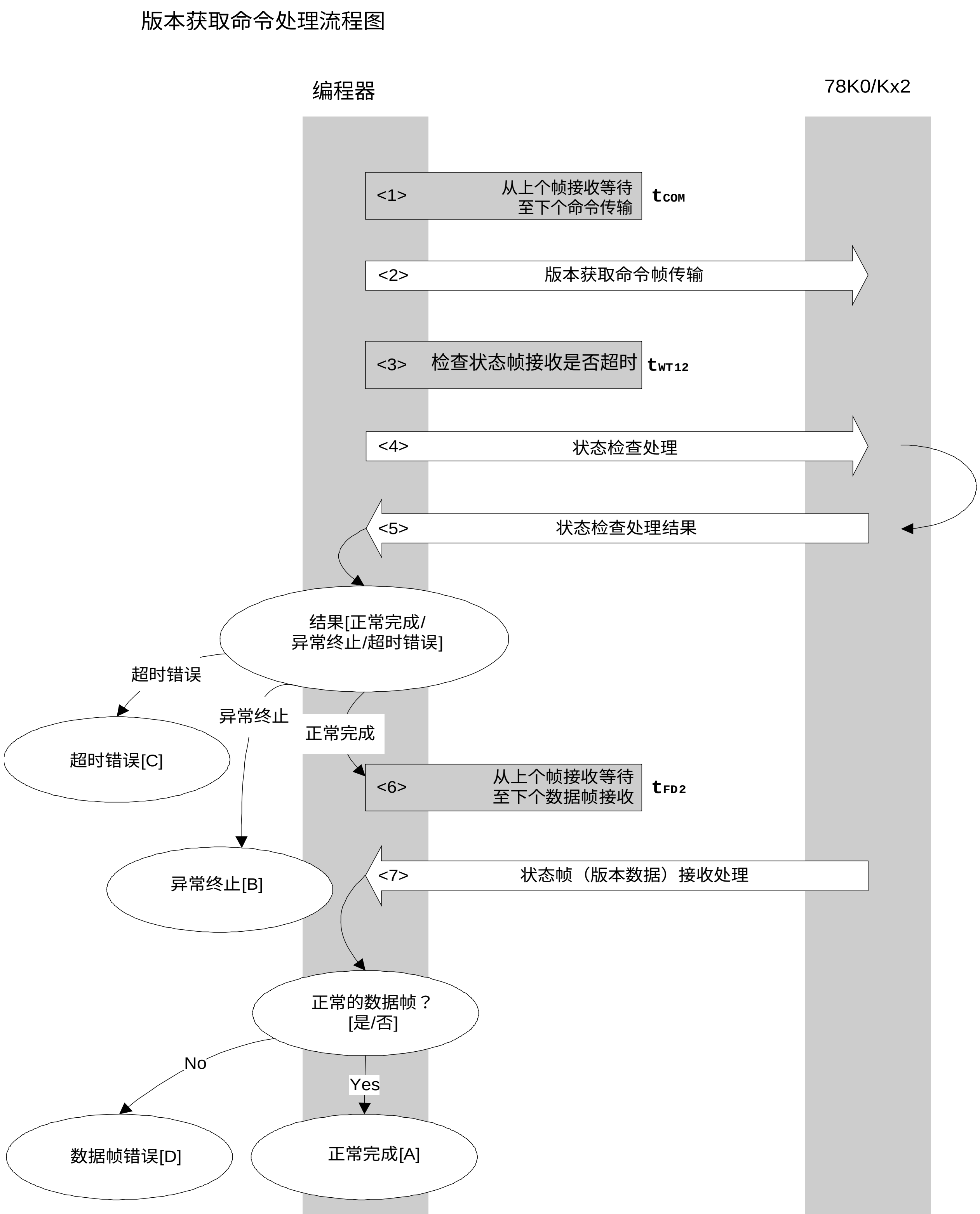
5.12.5 示例程序

以下所示为硅签名命令处理的示例程序。

```
/*************************************************************/
/*                                                          */
/* 获取硅签名命令 (CSI)                                      */
/*                                                          */
/*************************************************************/
/* [i] u8 *sig      ... 指向签名保存数据的指针               */
/* [r] u16          ... 错误代码                            */
/*************************************************************/
u16             fl_csi_getsig(u8 *sig)
{
        u16     rc;

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

        put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);
                                                        // 发送"硅签名"命令

        fl_wait(tWT11);

        rc = fl_csi_getstatus(tWT11_TO);        // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // continue
        //      case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }

        fl_wait(tFD2_SIG);                      // 获取数据帧前等待

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

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

5.13 版本获取命令

5.13.1 处理流程图



5.13.2 处理流程描述

<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　版本获取命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT12}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	转至 <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]。

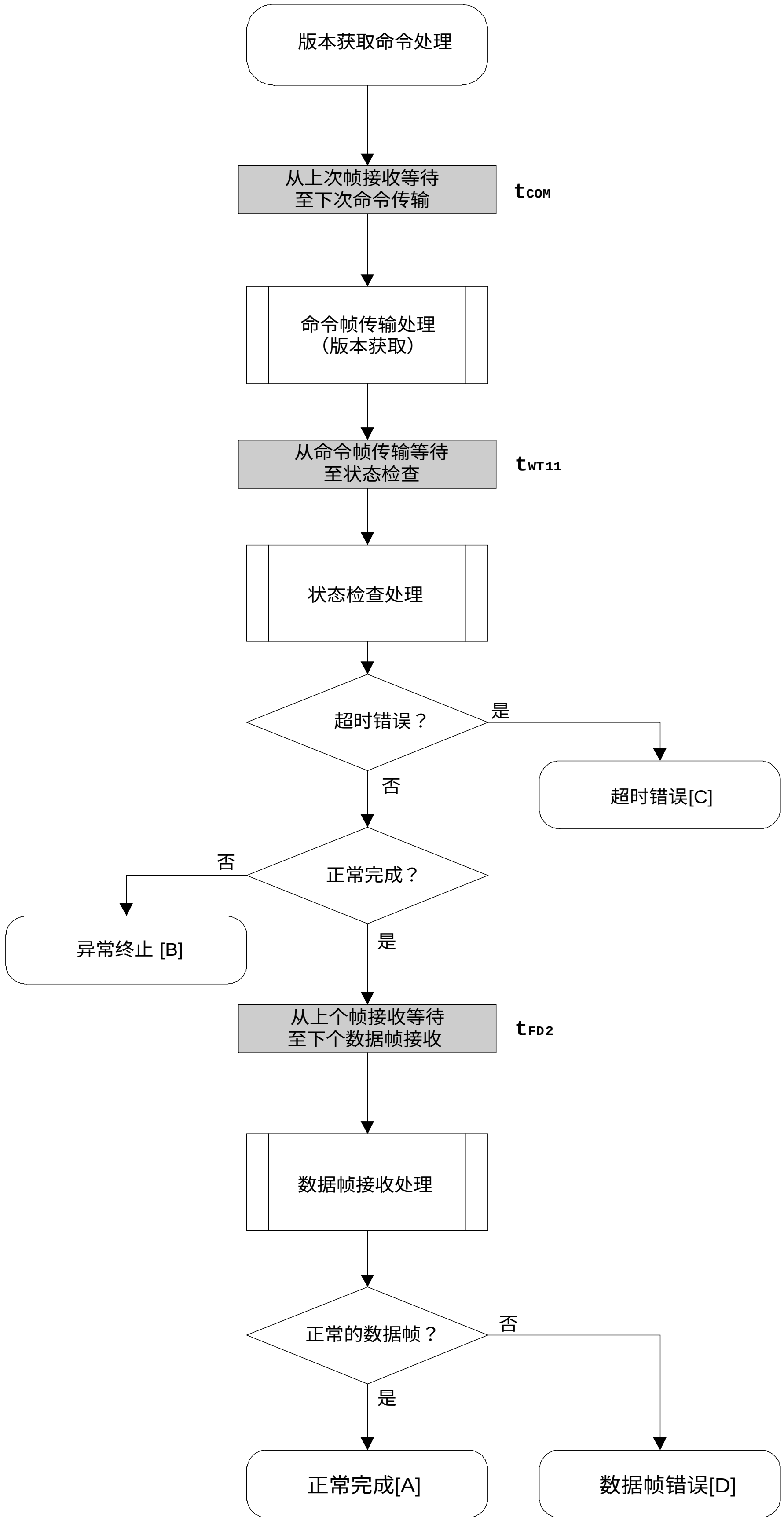
<6>　从上一帧接收等待至下次命令发送（等待时间 t_{FD2}）。
<7>　检查已接收数据帧（版本数据）。

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]

5.13.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取版本数据。
异常终止 [B]	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
数据帧错误 [D]		–	接收到数据帧校验和，因为版本数据异常。

5.13.4 流程图



版本获取命令处理
从上次帧接收等待
至下次命令传输
tCOM
命令帧传输处理
（版本获取）
从命令帧传输等待
至状态检查
tWT11
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]
从上个帧接收等待
至下个数据帧接收
tFD2
数据帧接收处理
正常的数据帧？
否
是
正常完成[A]
数据帧错误[D]

5.13.5 示例程序

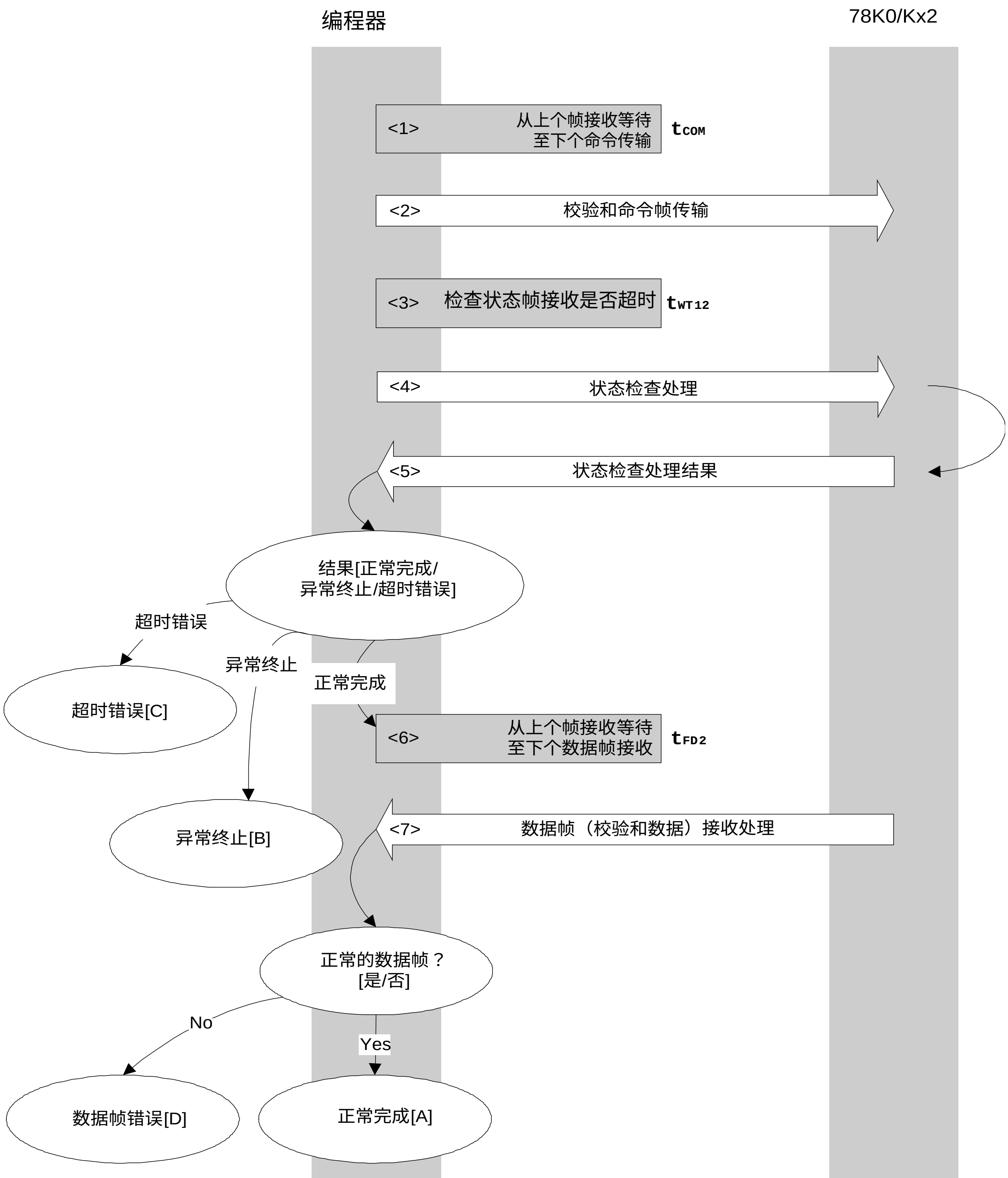
以下所示为版本获取命令处理的示例程序。

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


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

        put_cmd_csi(FL_COM_GET_VERSION, 1, fl_cmd_prm);   // 发送“版本获取”命令

        fl_wait(tWT12);

        rc = fl_csi_getstatus(tWT12_TO);        // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // continue
//              case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }

        fl_wait(tFD2_VG);               // 获取数据帧前等待

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

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

5.14 校验和命令

5.14.1 处理流程图

校验和命令处理流程



5.14.2 处理流程描述

<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　校验和命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT16}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：	转至 <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]。

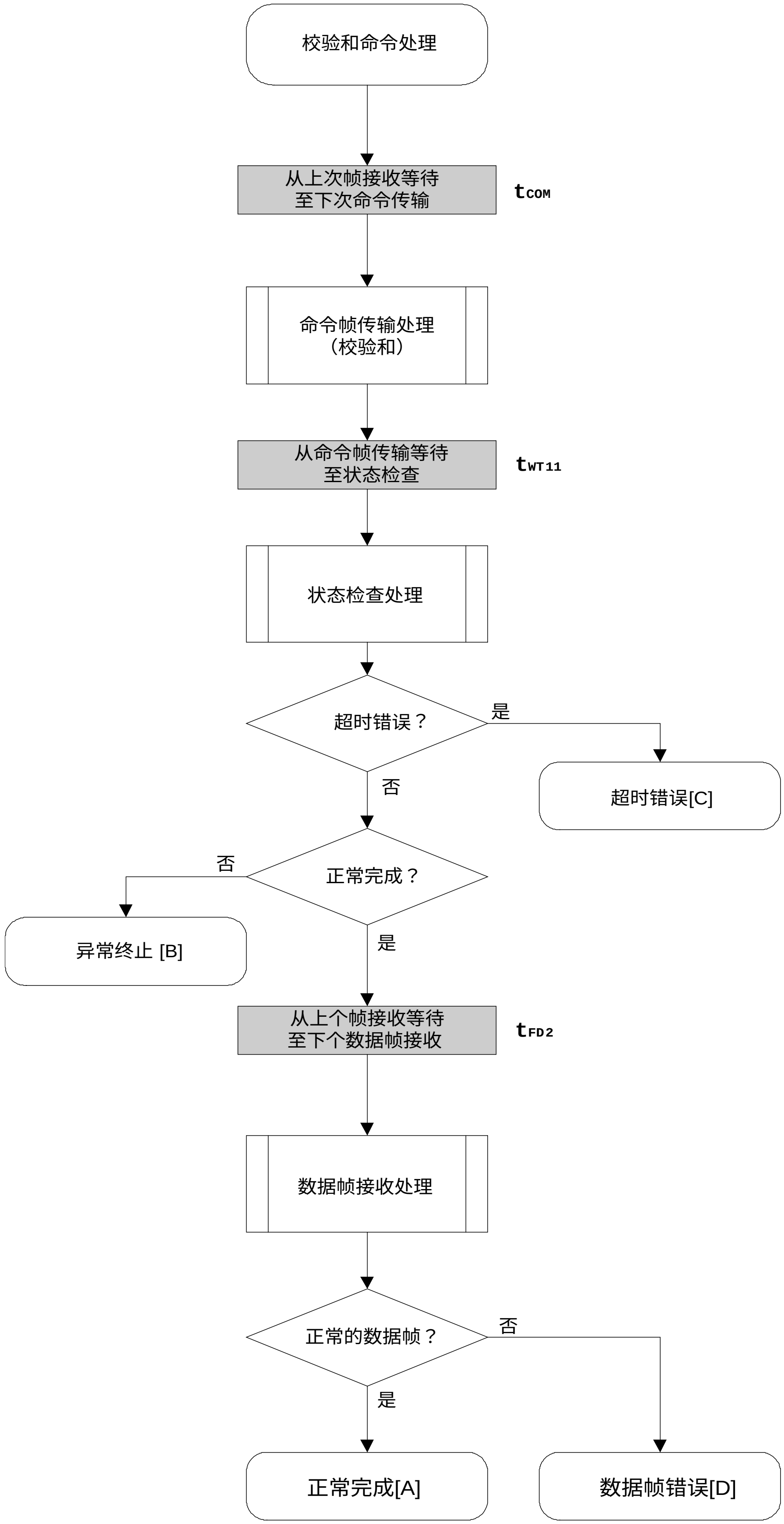
<6>　从上一帧接收等待至下次命令发送（等待时间 t_{FD1}）。
<7>　检查已接收数据帧（校验和数据）。

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]

5.14.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常获取校验和数据。
异常终止 [B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者指定的地址不是 2 KB 单位内的固定地址。
	校验和错误	07H	已发送命令帧的校验和发生异常。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。
数据帧错误 [D]		–	接收到数据帧校验和，因为版本数据异常。

5.14.4 流程图


校验和命令处理
从上次帧接收等待
至下次命令传输
tCOM
命令帧传输处理
（校验和）
从命令帧传输等待
至状态检查
tWT11
状态检查处理
超时错误？
是
超时错误[C]
否
正常完成？
否
异常终止 [B]
是
从上个帧接收等待
至下个数据帧接收
tFD2
数据帧接收处理
正常的数据帧？
否
是
正常完成[A]
数据帧错误[D]

5.14.5 示例程序

以下所示为校验和命令处理的示例程序。

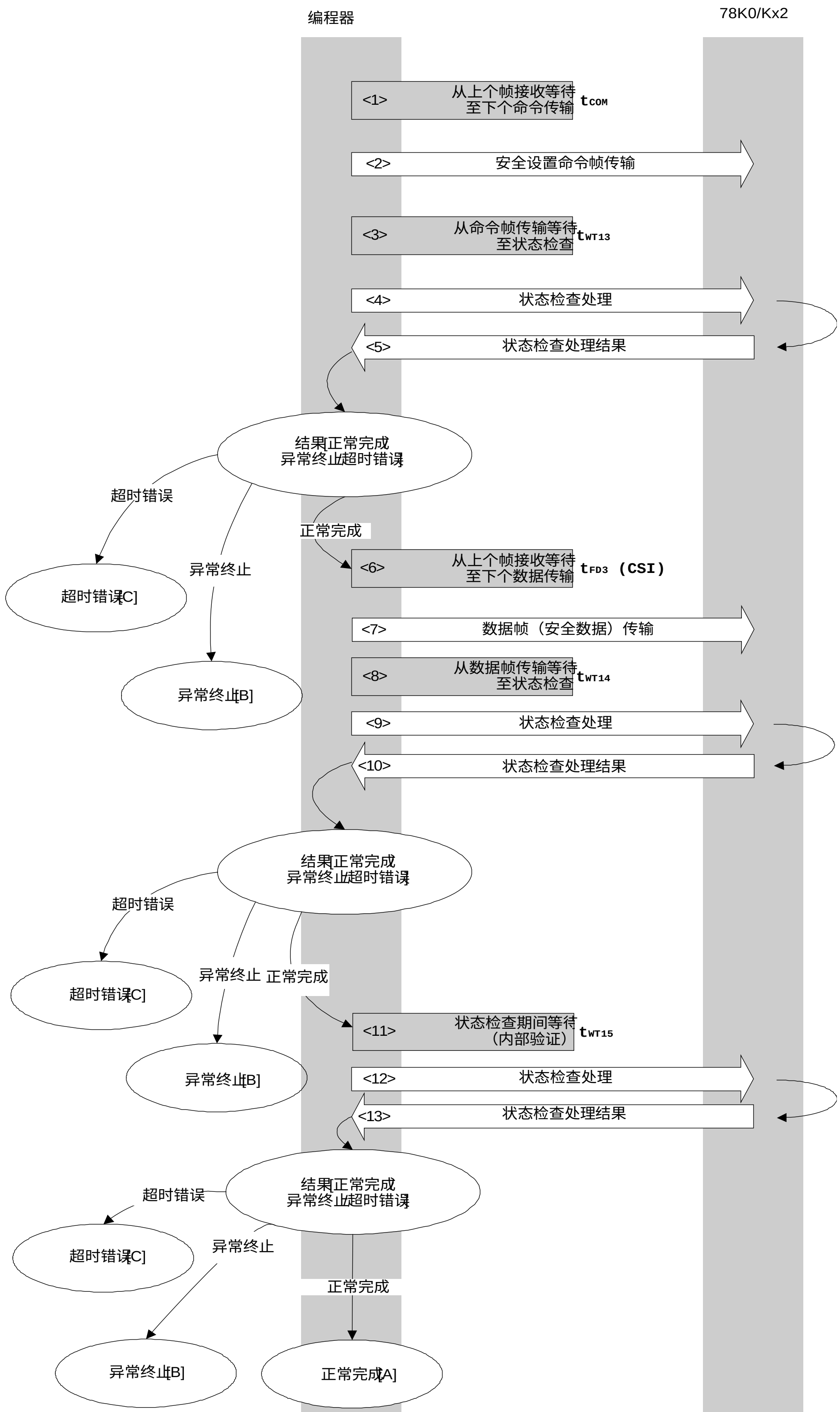
```
/**************************************************************/
/*                                                            */
/* 获取校验和命令 (CSI)                                        */
/*                                                            */
/**************************************************************/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

    /**********************************************/
    /*    设置参数                                 */
    /**********************************************/
    // 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);    // 获取块数目

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

    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_TO);           // 获取状态帧
    switch(rc) {
          case    FLC_NO_ERR:                         break;   // continue
    //    case    FLC_DFTO_ERR:          return  rc;  break;   // case [C]
          default:          return   rc;    break;  // case [B]
    }


    /**********************************************/
    /*    获取数据帧（校验和数据）                    */
    /**********************************************/
    fl_wait(tFD1 * block_num);                 // 获取数据帧前等待
```

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

    if (rc){                              // 如果存在错误，
        return rc;                        // case [D]
    }

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

5.15 安全设置命令

5.15.1 处理流程图

安全设置命令处理流程图



5.15.2 处理流程描述

<1>　从上一帧接收等待至下次命令发送（等待时间 t_{COM}）。
<2>　安全设置命令由命令帧发送处理进行发送。
<3>　从命令发送等待至状态检查处理（等待时间 t_{WT13}）。
<4>　状态帧由状态检查处理进行获取。
<5>　根据状态检查处理的结果执行以下处理。

当处理正常结束时：　　转至 <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>　从上一帧接收等待至下次命令发送（等待时间 t_{FD3}(CSI)）。
<7>　数据帧（安全设置数据）由数据帧发送处理进行发送。
<8>　从数据帧发送等待至状态检查处理（等待时间 t_{WT14}）。
<9>　状态帧由状态检查处理进行获取。
<10> 根据状态检查处理的结果执行以下处理。

当处理正常结束时：　　转至 <11>。
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]。

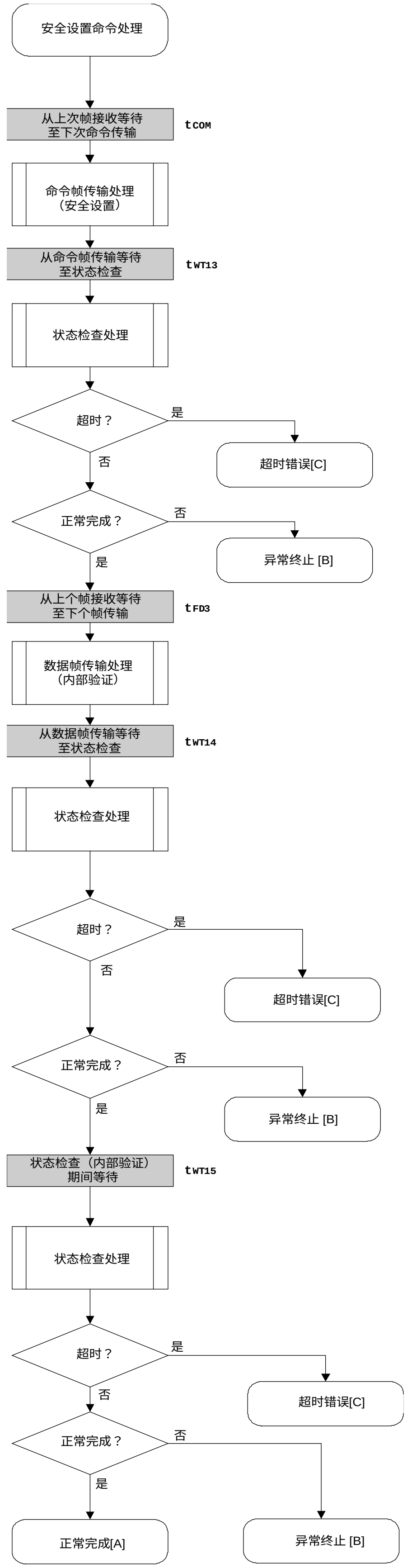
<11> 等待至状态获取（内部验证完成）（等待时间 t_{WT15}）。
<12> 状态帧由状态检查处理进行获取。
<13> 根据状态检查处理的结果执行以下处理。

当处理正常结束时：　　正常完成 [A]
当处理异常结束时：　　异常终止 [B]
当发生超时错误时：　　返回超时错误 [C]。

5.15.3 处理完成时状态

处理完成时状态		状态码	描述
正常完成 [A]	正常应答 (ACK)	06H	已正常执行命令，并且已正常执行安全设置。
异常终止 [B]	参数错误	05H	命令信息（参数）不是 00H。
	校验和错误	07H	已发送命令帧或数据帧的校验和发生异常。
	写入错误	1CH	已设置安全数据，或发生写入错误。
	否定应答 (NACK)	15H	命令帧数据发生异常（例如数据长度 (LEN) 无效或无 ETX）。
超时错误 [C]		–	未在指定时间内接收到状态帧。

5.15.4 流程图


安全设置命令处理
从上次帧接收等待
至下次命令传输
tCOM
命令帧传输处理
（安全设置）
从命令帧传输等待
至状态检查
tWT13
状态检查处理
超时？
是
超时错误[C]
否
正常完成？
否
异常终止 [B]
是
从上个帧接收等待
至下个帧传输
tFD3
数据帧传输处理
（内部验证）
从数据帧传输等待
至状态检查
tWT14
状态检查处理
超时？
是
超时错误[C]
否
正常完成？
否
异常终止 [B]
是
状态检查（内部验证）
期间等待
tWT15
状态检查处理
超时？
是
超时错误[C]
否
正常完成？
否
是
正常完成[A]
异常终止 [B]

5.15.5 示例程序

以下所示为安全设置命令处理的示例程序。

```
/***************************************************************/
/*                                                             */
/* 设置安全旗标命令 (CSI)                                        */
/*                                                             */
/***************************************************************/
/* [i] u8 scf     ... 安全旗标数据                               */
/* [r] u16        ... 错误代码                                   */
/***************************************************************/
u16             fl_csi_setscf(u8 scf)
{
        u16     rc;

        /*******************************************************/
        /*     设置参数                                          */
        /*******************************************************/
        fl_cmd_prm[0] = 0x00;           // "BLK" (必须是 0x00)
        fl_cmd_prm[1] = 0x00;           // "PAG" (必须是 0x00)
        fl_txdata_frm[0] = (scf |= 0b11101000);
                                        // "FLG" (上 5 位必须为“1”（务必）)

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



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

        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_TO);        // 获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // continue
//              case    FLC_DFTO_ERR:           return  rc;     break;  // case [C]
                default:        return  rc;     break;  // case [B]
        }

        /***********************************************/
        /*  发送数据帧（安全设置数据）                      */
        /***********************************************/
        fl_wait(tFD3_CSI);                      // 获取数据帧前等待
```

```
        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;  // continue
        //      case    FLC_DFTO_ERR:               return  rc;         break;  // case [C]
                default:                return  rc;     break;  // case [B]
        }


        /***********************************************/
        /*      检查内部验证                            */
        /***********************************************/
        fl_wait(tWT15);

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

第 6 章 FLASH 存储器编程参数特性

本章介绍在 Flash 存储器编程模式下编程器与 78K0/Kx2 之间的参数特性。
设计编程器时请务必参阅 78K0/Kx2 用户手册以了解相关电气技术规范。

<R> 6.1 扩展规范产品 (μPD78F05xxA) 的 Flash 存储器编程参数特性

6.1.1 基本特性

参数	状态	符号	MIN.	TYP.	MAX.	单位
Flash 存储器编程模式下 78K0/Kx2 操作时钟	内部高速振荡时钟	f_{RH}	7.6	8	8.4	MHz
X1 时钟	UART 通讯期间	f_X	2		20	
外部主系统时钟		f_{EXCLK}	2		20	

6.1.2 Flash 存储器编程模式设置时间

参数		符号	MIN.	TYP.	MAX.
V_{DD}↑ 至 FLMD0↑		t_{DP}	1 ms		
FLMD0↑ 至 $\overline{RESET}$↑		t_{PR}	2 ms		
从 $\overline{RESET}$↑ 至 FLMD0 [注 1] 计数开始时间		t_{RP}	59,327/f_{RH}		
从 $\overline{RESET}$↑ 至 FLMD0 [注 1] 计数完成时间		t_{RPE}			238,414/f_{RH}
FLMD0 计数器高电平/低电平宽度		t_{PW}	10 μs		100 μs
等待“复位”命令 (CSI) [注 1]		t_{RC}	444,463/f_{RH}		
等待低电平数据 1 (UART) [注 1]	X1 时钟	t_{R1}	444,463/f_{RH} + $2^{16}/f_X$		
	外部主系统时钟		444,463/f_{RH}		
等待低电平数据 2 (UART)		t_{12}	15,000/f_{RH}		
等待“读取”命令 (UART)		t_{2C}	15,000/f_{RH}		
低电平数据 1/2 [注 2] 的宽度		t_{L1}、t_{L2}		*注 2*	
FLMD0 计数器上升/下降时间		–			1 μs
复位低电平宽度（$\overline{RESET}$↑ 至 $\overline{RESET}$↓）[注 3]		t_{RST}	1,950 ms		

注 1. 建议将 (59,327/f_{RH} + 238,414/f_{RH})/2 作为 FLMD0 脉冲输入计时的标准值。

2. 低电平宽度与 00H 数据宽度相同，为 9,600 bps。

3. 单片机上电（复位释放）后，当模式从正常操作模式切换为 Flash 存储器编程模式时，请确保在加电（复位释放）后系统复位前等待该参数模式切换时所需的最小时间。

（下页所示备注。）

备注　**1.** 计算参数（假设 f_{RH} = 8 MHz）。
2. 等待时间定义如下。
<t_{R1} (MIN.)>
UART 的波特率由外部时钟生成。
按照技术规范和所使用的外部晶振振荡稳定时间设置所用的输入脉冲。

6.1.3　编程特性

等待	状态	符号	串行 I/F	MIN.	MAX.
数据帧发送/接收之间	数据帧接收	t_{DR}	CSI	$64/f_{RH}$	
			UART	$74/f_{RH}$	
	数据帧发送	t_{DT}	CSI	$88/f_{RH}$	
			UART	0 [注 1]	
从状态命令帧接收到状态帧发送	–	t_{SF}	CSI	$215/f_{RH}$	
从状态帧发送到数据帧发送 (1)	–	t_{FD1} [注 2]	CSI	$54,368/f_{RH}$	
			UART	0 [注 1]	
从状态帧发送到数据帧发送 (2)	硅签名数据	t_{FD2}	CSI	$321/f_{RH}$	
	版本数据			$206/f_{RH}$	
	–		UART	0 [注 1]	
从状态帧发送到数据帧接收	–	t_{FD3}	CSI	$163/f_{RH}$	
			UART	$101/f_{RH}$	
从状态帧发送到命令帧接收	–	t_{COM}	CSI	$106/f_{RH}$	
			UART	$106/f_{RH}$ [注 1]	

注 1. 当编程器启用连续接收时适用
2. 发送一个块所需时间

备注　**1.** 计算参数（假设 f_{RH} = 8 MHz）。
2. 等待时间定义如下。

<t_{DR}, t_{FD3}, t_{COM}>
前次通讯完成并且经过最小 (MIN.) 时间后，78K0/Kx2 准备好下次通讯。
前次通讯完成并且经过最小 (MIN.) 时间后，编程器可以发送下个数据。
最大 (MAX.) 时间未指定，在在大约 3 秒内发送下个数据。

<t_{DT}, t_{SF}, t_{FD1}, t_{FD2}>
前次通讯完成并且经过最小 (MIN.) 时间后，78K0/Kx2 准备好下次通讯。
前次通讯完成后并且经过最小 (MIN.) 时间前，编程器必须准备接收下个数据。
最大 (MAX.) 时间未指定，继续接收大约 3 秒钟，直到接收数据完成。

命令	符号	串行 I/F		MIN.	MAX.
复位	t_{WT0}	CSI		172/f_{RH}	
		UART		注 **1**	
片擦除	t_{WT1}	–		857,883/f_{RH} + 44,160 × 块总数目/f_{RH}	186,444,400/f_{RH} + 11,304,960 × 块总数目/f_{RH}
块擦除	t_{WT2} [注 2]	–		214,714/f_{RH} × 同时选择与擦除的执行计数 + 44,160/f_{RH} × 要擦除的块数目	54,582,372/f_{RH} × 同时选择与擦除的执行计数 + 11,304,960/f_{RH} × 要擦除的块数目
编程	t_{WT3}	CSI		1,506/f_{RH}	
		UART		注 **1**	
	t_{WT4} [注 3]	–		72,412/f_{RH}	893,355/f_{RH}
	t_{WT5} [注 4]	CSI	块 0	100,407/f_{RH}	132,144,427/f_{RH}
			块 1 至 127	100,407/f_{RH}	102,178/f_{RH}
		UART	块 0	注 **1**	132,144,427/f_{RH}
			块 1 至 127	注 **1**	102,178/f_{RH}
校验	t_{WT6}	CSI		686/f_{RH}	
		UART		注 **1**	
	t_{WT7} [注 3]	CSI		12,827/f_{RH}	
		UART		注 **1**	
块空白检查	t_{WT8} [注 4]	CSI		45,870/f_{RH}	55,044/f_{RH}
		UART		注 **1**	55,044/f_{RH}
振荡频率设置	t_{WT9}	CSI		1,238/f_{RH}	
		UART		注 **1**	
硅签名	t_{WT11}	CSI		1,233/f_{RH}	
		UART		注 **1**	
版本获取	t_{WT12}	CSI		252/f_{RH}	
		UART		注 **1**	
安全设置	t_{WT13}	CSI		975/f_{RH}	
		UART		注 **1**	
	t_{WT14}	–		275,518/f_{RH}	66,005,812/f_{RH}
	t_{WT15}	CSI		368,277/f_{RH}	66,018,156/f_{RH}
		UART		注 **1**	66,018,156/f_{RH}
校验和	t_{WT16}	CSI		583/f_{RH}	
		UART		注 **1**	

注 1. 命令发送前编程器必须启用接收。

2. 请参阅 **6.3 "块擦除"命令执行的同时选择与擦除**以了解有关同时选择与擦除的执行计数的计算方法。

3. 发送 256 字节数据所需时间

4. 发送一个块所需时间

备注 **1.** 计算参数（假设 f_{RH} = 8 MHz）。

2. 等待时间定义如下。

<t_{WT0} 至 t_{WT16}>

78K0/Kx2 完成命令处理所需的时间位于 MAX 和 MIN 之间，。

编程器必须以最大 (MAX.) 时间重复检查状态（如果未定义最大时间，则大约 3 秒钟）。

6.2 常规产品 (μPD78F05xx) 的 Flash 存储器编程参数特性

6.2.1 基本特性

参数	状态	符号	MIN.	TYP.	MAX.	单位
Flash 存储器编程模式下 78K0/Kx2 操作时钟	内部高速振荡时钟	f_{RH}	7.6	8	8.4	MHz
X1 时钟	UART 通讯期间	f_X	2		20	
外部主系统时钟		f_{EXCLK}	2		20	

<R> 6.2.2 Flash 存储器编程模式设置时间

参数		符号	MIN.	TYP.	MAX.
V_{DD}↑ 至 FLMD0↑		t_{DP}	1 ms		
FLMD0↑ 至 $\overline{\text{RESET}}$↑		t_{PR}	2 ms		
从 RESET↑ 至 FLMD0 计数开始时间[注 1]		t_{RP}	59,327/f_{RH}		
从 RESET↑ 至 FLMD0 计数完成时间[注 1]		t_{RPE}			238,414/f_{RH}
FLMD0 计数器高电平/低电平宽度		t_{PW}	10 μs		100 μs
等待“复位”命令 (CSI)		t_{RC}	444,463/f_{RH}		
等待低电平数据 1 (UART)	X1 时钟	t_{R1}	444,463/f_{RH} + 2^{16}/f_X		
	外部主系统时钟		444,463/f_{RH}		
等待低电平数据 2 (UART)		t_{12}	15,000/f_{RH}		
等待“读取”命令 (UART)		t_{2C}	15,000/f_{RH}		
低电平数据 1/2 [注 2] 的宽度		t_{L1}、t_{L2}		***注 2***	
FLMD0 计数器上升/下降时间		–			1 μs
复位低电平宽度（$\overline{\text{RESET}}$↑ 至 $\overline{\text{RESET}}$↓）[注 3]		t_{RST}	1,950 ms		

注 1. 建议将 (59,327/f_{RH} + 238,414/f_{RH})/2 作为 FLMD0 脉冲输入计时的标准值。

2. 低电平宽度与 9,600 bps 波特率下的 00H 数据宽度相同，此处描述的值为数据宽度的一半。

3. 单片机上电（复位释放）后，当模式从正常操作模式切换为 Flash 存储器编程模式时，请确保在加电（复位释放）后系统复位前等待该参数模式切换时所需的最小时间。

备注 **1.** 计算参数（假设 f_{RH} = 8 MHz）。

2. 等待时间定义如下。

<t_{R1} (MIN.)>

UART 的波特率由外部时钟生成。

按照技术规范和所使用的外部晶振振荡稳定时间设置所用的输入脉冲。

<R> 6.2.3 编程特性

等待	状态	符号	串行 I/F	MIN.	MAX.
数据帧发送/接收之间	数据帧接收	t_{DR}	CSI	$64/f_{RH}$	
			UART	$74/f_{RH}$	
	数据帧发送	t_{DT}	CSI	$88/f_{RH}$	
			UART	0 注 1	
从状态命令帧接收到状态帧发送	–	t_{SF}	CSI	$166/f_{RH}$	
从状态帧发送到数据帧发送 (1)	–	t_{FD1} 注 2	CSI	$54,368/f_{RH}$	
			UART	0 注 1	
从状态帧发送到数据帧发送 (2)	Silicon Signature 数据	t_{FD2}	CSI	$321/f_{RH}$	
	版本数据			$136/f_{RH}$	
	–		UART	0 注 1	
从状态帧发送到数据帧接收	–	t_{FD3}	CSI	$163/f_{RH}$	
			UART	$101/f_{RH}$	
从状态帧发送到命令帧接收	–	t_{COM}	CSI	$64/f_{RH}$	
			UART	$71/f_{RH}$	

注 1. 当编程器启用连续接收时适用

2. 发送一个块所需时间

备注 **1.** 计算参数（假设 f_{RH} = 8 MHz）。

2. 等待时间定义如下。

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

前次通讯完成并且经过最小 (MIN.) 时间后，78K0/Kx2 准备好下次通讯。

前次通讯完成并且经过最小 (MIN.) 时间后，编程器可以发送下个数据

最大 (MAX.) 时间未指定，在大约 3 秒内发送下个数据。

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

前次通讯完成并且经过最小 (MIN.) 时间后，78K0/Kx2 准备好下次通讯。

前次通讯完成后并且经过最小 (MIN.) 时间前，编程器必须准备接收下个数据。

最大 (MAX.) 时间未指定，继续接收大约 3 秒钟，直到接收数据完成。

命令	符号	串行 I/F		MIN.	MAX.
复位	t_{WT0}	CSI		172/f_{RH}	
		UART		**注 1**	
片擦除	t_{WT1}	–		857,883/f_{RH} + 44,160 × 块总数目/f_{RH}	186,444,400/f_{RH} + 11,304,960 × 块总数目/f_{RH}
块擦除	t_{WT2} [注 2]	–		214,714/f_{RH} × 同时选择与擦除的执行计数 + 44,160/f_{RH} × 要擦除的块数目	54,582,372/f_{RH} × 同时选择与擦除的执行计数 + 11,304,960/f_{RH} × 要擦除的块数目
编程	t_{WT3}	CSI		1,348/f_{RH}	
		UART		**注 1**	
	t_{WT4} [注 3]	–		68,118/f_{RH}	397,587/f_{RH}
	t_{WT5} [注 4]	CSI	块 0	100,407/f_{RH}	132,144,427/f_{RH}
			块 1 至 127	100,407/f_{RH}	102,178/f_{RH}
		UART	块 0	**注 1**	132,144,427/f_{RH}
			块 1 至 127	**注 1**	102,178/f_{RH}
校验	t_{WT6}	CSI		686/f_{RH}	
		UART		**注 1**	
	t_{WT7} [注 3]	CSI		12,827/f_{RH}	
		UART		**注 1**	
块空白检查	t_{WT8} [注 4]	CSI		45,835/f_{RH}	55,044/f_{RH}
		UART		**注 1**	55,044/f_{RH}
振荡频率设置	t_{WT9}	CSI		1,127/f_{RH}	
		UART		**注 1**	
Silicon Signature	t_{WT11}	CSI		1,233/f_{RH}	
		UART		**注 1**	
版本获取	t_{WT12}	CSI		242/f_{RH}	
		UART		**注 1**	
安全设置	t_{WT13}	CSI		923/f_{RH}	
		UART		**注 1**	
	t_{WT14}	–		275,518/f_{RH}	66,005,812/f_{RH}
	t_{WT15}	CSI		368,277/f_{RH}	66,018,156/f_{RH}
		UART		**注 1**	66,018,156/f_{RH}
校验和	t_{WT16}	CSI		583/f_{RH}	
		UART		**注 1**	

注 1. 命令发送前编程器必须启用接收。

2. 请参阅 **6.3 由“块擦除”命令执行的同时选择与擦除**以了解有关同时选择与擦除的执行计数的计算方法。

3. 发送 256 字节数据所需时间

4. 发送一个块所需时间

备注 **1.** 计算参数（假设 f_{RH} = 8 MHz）。

2. 等待时间定义如下。

<t_{WT0} 至 t_{WT16}>

78K0/Kx2 完成命令处理所需的时间位于 MAX 和 MIN 之间。

编程器必须以最大 (MAX.) 时间重复检查状态（如果未定义最大时间，则大约 3 秒钟）。

6.3 由块擦除命令执行的同时选择与擦除

78K0/Kx2 的块擦除命令通过重复"同时选择与擦除"方式执行，这种方式可以同时擦除多个块。
因此在"块擦除"命令执行期间插入的等待时间等于"同时选择与擦除"的总执行时间。
要计算"同时选择与擦除的总执行时间"，必须首先计算"同时选择与擦除"的执行计数 (M)。
"M"由要同时擦除的块数计算得出，此处的块数也指同时选择和擦除的块数。

以下介绍计算同时选择和擦除的块数和执行计数 (M) 的方法。

(1) 计算同时选择和擦除的块数

同时选择和擦除的块数应为 1、2、4、8、16、32、64, 或 128，这取决于以下条件的满足程度。

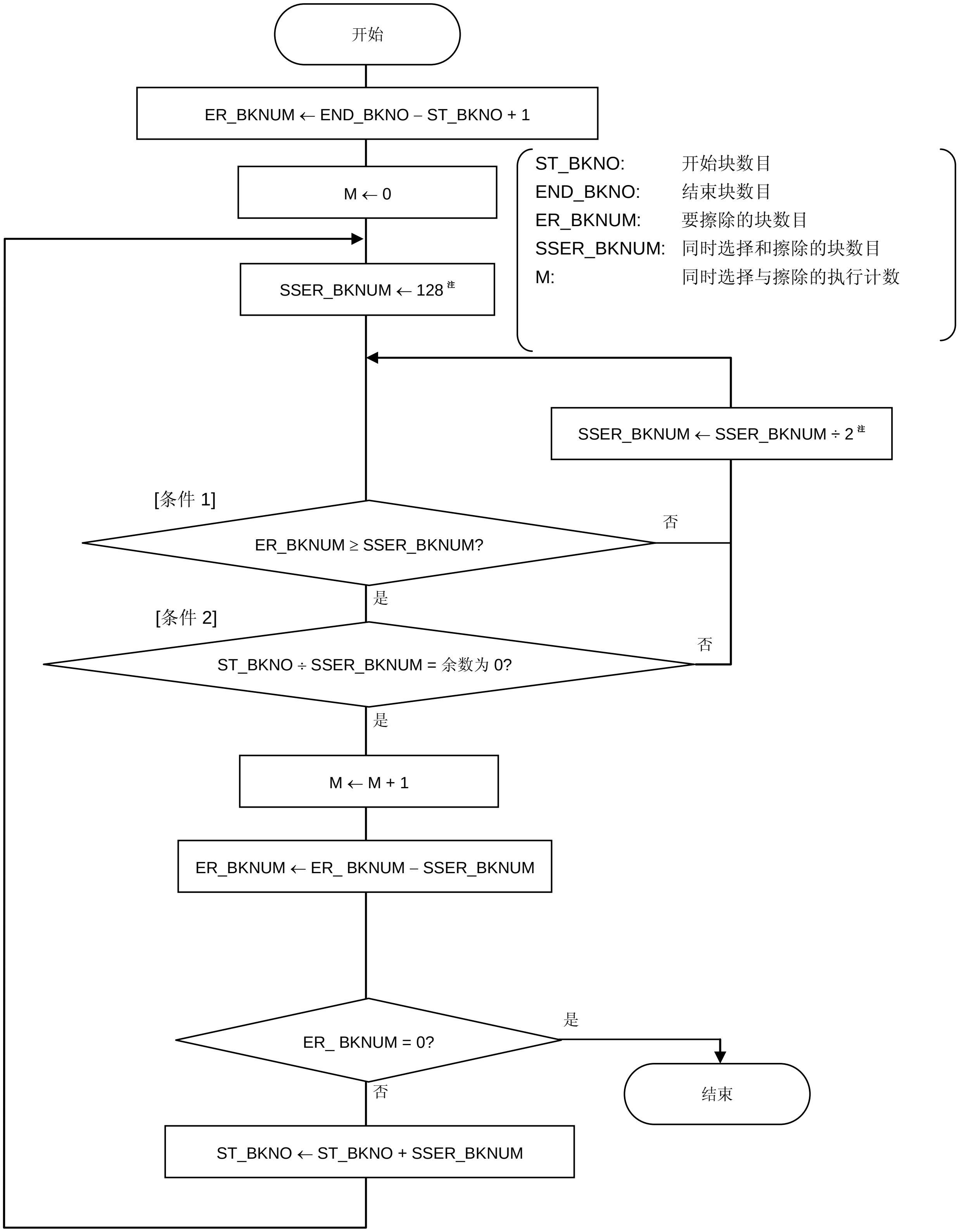
[条件 1]
（要擦除块数目）≥（同时选择和擦除的块数目）

[条件 2]
（开始块数目）÷（同时选择和擦除的块数目）= 余数为 0。

[条件 3]
满足条件 1 和 2 的所有值中最大值

(2) 同时选择与擦除的执行计数 (M) 计算

执行计数 (M) 如下方流程图中所示。



注　根据 SSER_BKNUM (128) 的最大值，执行 SSER_BKNUM ÷ 2，获得满足条件 1 和 2 的值；之后条件 3 获得满足。

示例 1　　擦除块 1 至 127（N（要擦除的块数目）= 127）

<1>　第一个起始块编号为 1，要擦除的块数目为 127；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1，满足条件 3 的值为 1，因此同时选择与擦除的块数目为 1；只有块 1 被擦除。

<2>　擦除块 1 后下一个开始的块编号为 2 并且要擦除的块数目为 126；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1 和 2，满足条件 3 的值为 2，因此同时选择与擦除的块数目为 2；块 2 和 3 被擦除。

<3>　擦除块 2 和块 3 后下一个开始的块编号为 4 并且要擦除的块数目为 124；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1、2 和 4，满足条件 3 的值为 4，因此同时选择与擦除的块数目为 4；块 4 至 7 被擦除。

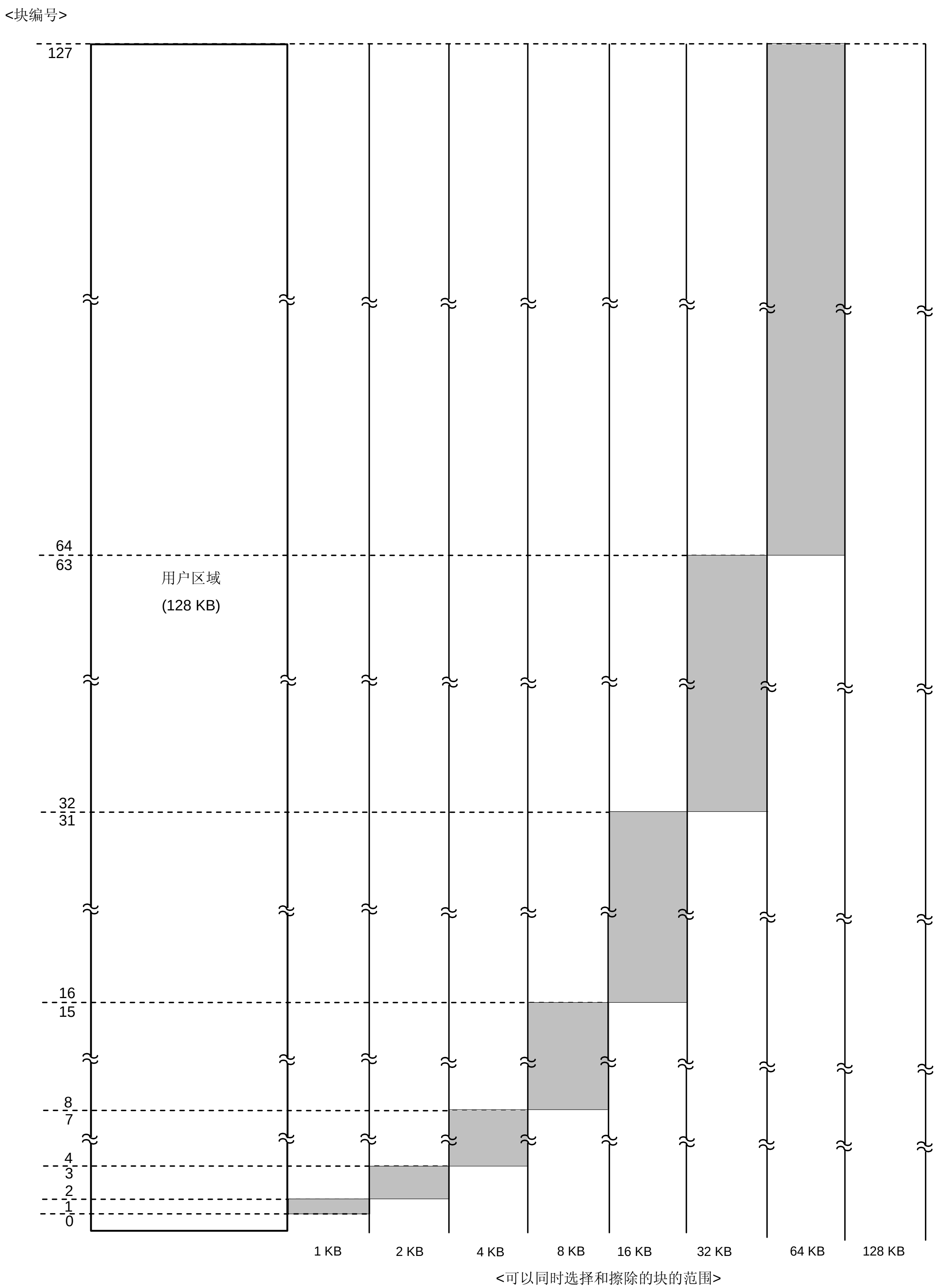
<4>　擦除块 4 至块 7 后下一个开始的块编号为 8 并且要擦除的块数目为 120；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1、2、4 和 8，满足条件 3 的值为 8，因此同时选择与擦除的块数目为 8；块 8 至 15 被擦除。

<5>　擦除块 8 至块 15 后下一个开始的块编号为 16 并且要擦除的块数目为 112；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1、2、4、8 和 16，满足条件 3 的值为 16，因此同时选择与擦除的块数目为 16；块 16 至 31 被擦除。
擦除块 16 至块 31 后下一个开始的块编号为 32 并且要擦除的块数目为 96；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1、2、4、8、16 和 32，满足条件 3 的值为 32，因此同时选择与擦除的块数目为 32；块 32 至 63 被擦除。

<6>　擦除块 32 至块 63 后下一个开始的块编号为 64 并且要擦除的块数目为 64；因此满足条件 1 的值为 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值为 1、2、4、8、16、32 和 64，满足条件 3 的值为 64，因此同时选择与擦除的块数目为 64；块 64 至 127 被擦除。

因此同时选择和擦除执行了七次（1，2 和 3，4 至 7，8 至 15，16 至 31，32 至 63 和 64 至 127），要擦除块 1 至 127，因此获得 M = 7。

执行同时选择与擦除时块配置（擦除块 1 至 127 时）



示例 2 擦除块 5 至 10（N（要擦除的块数目）= 6）

<1> 第一个起始块编号为 5，要擦除的块数目为 6；因此满足条件 1 的值为 1、2 和 4。
此外，满足条件 2 的值为 1，满足条件 3 的值为 1，因此同时选择与擦除的块数目为 1；只有块 5 被擦除。

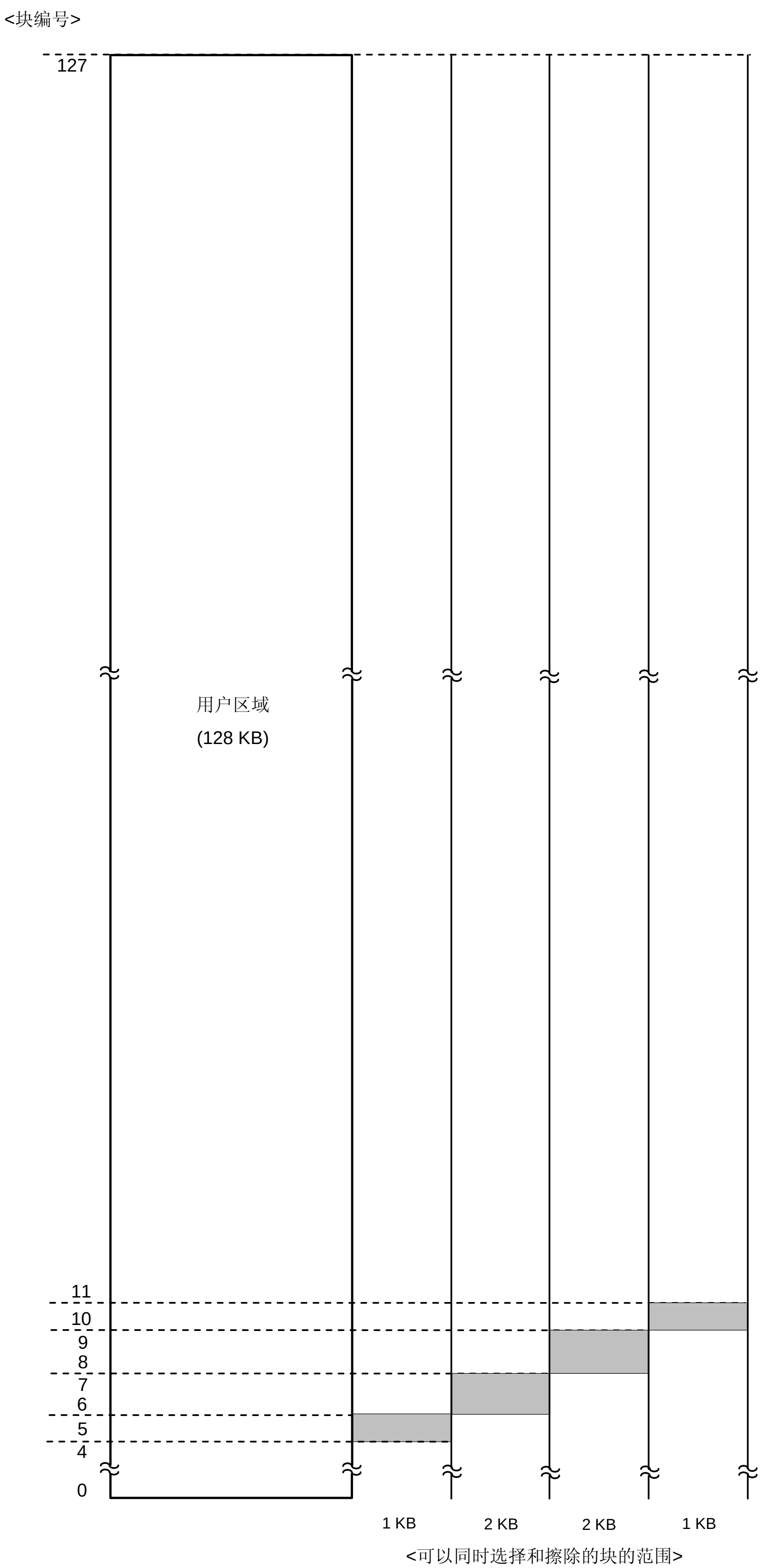
<2> 擦除块 5 后下一个开始的块编号为 6 并且要擦除的块数目为 5；因此满足条件 1 的值为 1、2 和 4。
此外，满足条件 2 的为 1 和 2，满足条件 3 的值为 2，因此同时选择与擦除的块数目为 2；块 6 和 7 被擦除。

<3> 擦除块 6 和块 7 后下一个开始的块编号为 8 并且要擦除的块数目为 3；因此满足条件 1 的值为 1 和 2。
此外，满足条件 2 的为 1 和 2，满足条件 3 的值为 2，因此同时选择与擦除的块数目为 2；块 8 和 9 被擦除。

<4> 擦除块 8 和块 9 后下一个开始的块编号为 10，并且要擦除的块数目为 1；因此满足条件 1 的值为 1。这同时也满足条件 2 和 3，因此要同时选择和擦除的块数目为 1；块 10 被擦除。

因此同时选择和擦除执行了四次（5，6 和 7，8 和 9，10），要擦除块 5 至 10，因此获得 M = 4。

执行同时选择与擦除时块配置（擦除块 5 至 10 时）



示例 **3**　　擦除块 25 至 73（N（要擦除的块数目）= 49）

<1> 第一个起始块编号为 25，要擦除的块数目为 49；因此满足条件 1 的值为 1、2、4、8、16 和 32。
此外，满足条件 2 的值为 1，满足条件 3 的值为 1，因此同时选择与擦除的块数目为 1；只有块 25 被擦除。

<2> 擦除块 25 后下一个开始的块编号为 26 并且要擦除的块数目为 48；因此满足条件 1 的值为 1、2、4、8、16 和 32。
此外，满足条件 2 的值为 1 和 2，满足条件 3 的值为 2，因此同时选择与擦除的块数目为 2；块 26 和 27 被擦除。

<3> 擦除块 26 和块 27 后下一个开始的块编号为 28 并且要擦除的块数目为 46；因此满足条件 1 的值为 1、2、4、8、16 和 32。
此外，满足条件 2 的值为 1、2 和 4，满足条件 3 的值为 4，因此同时选择与擦除的块数目为 4；块 28 至 31 被擦除。

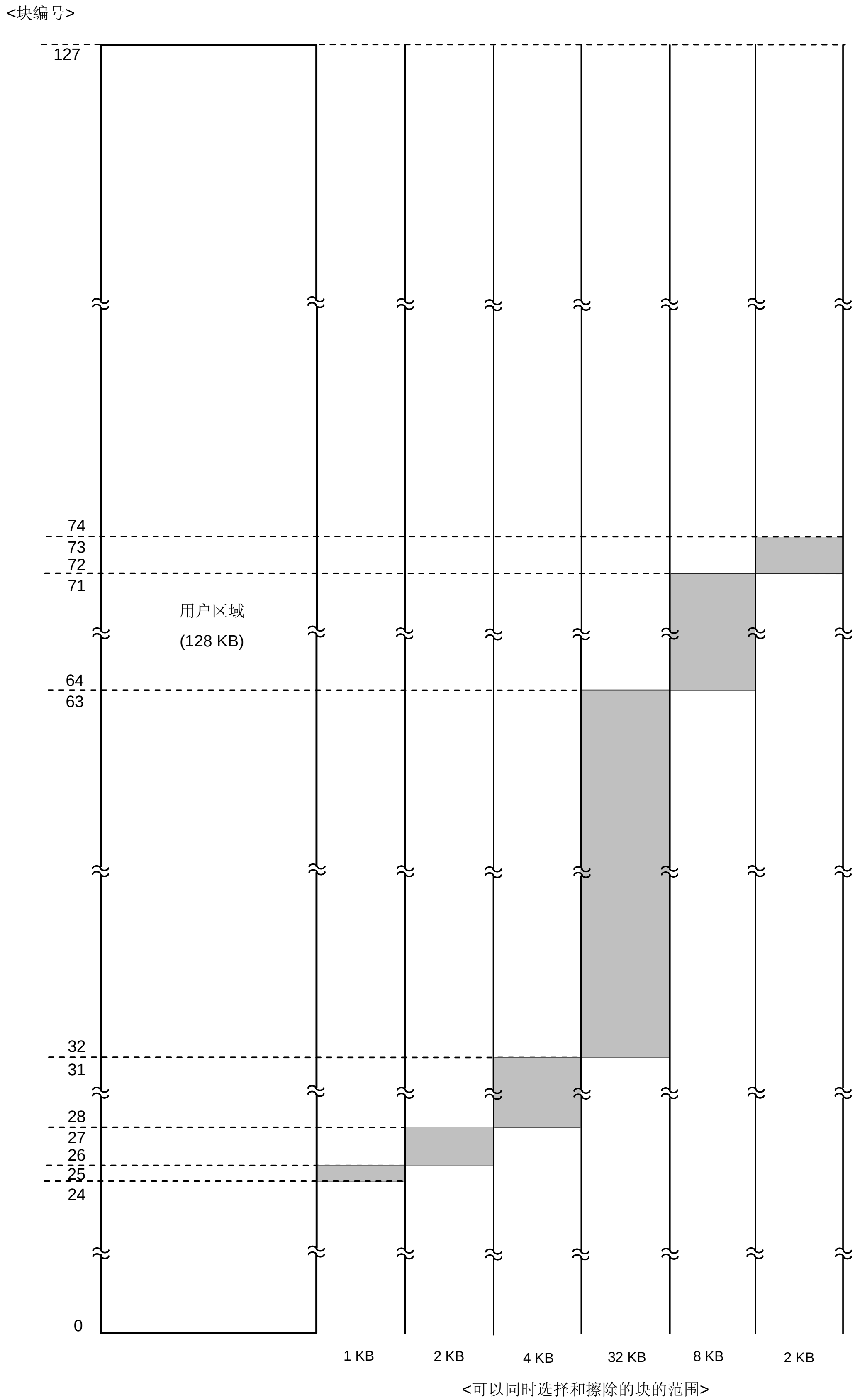
<4> 擦除块 28 至块 31 后下一个开始的块编号为 32 并且要擦除的块数目为 42；因此满足条件 1 的值为 1、2、4、8、16 和 32。
此外，满足条件 2 的值为 1、2、4、8 和 32，满足条件 3 的值为 32，因此同时选择与擦除的块数目为 32；块 32 至 63 被擦除。

<5> 擦除块 32 至块 63 后下一个开始的块编号为 64 并且要擦除的块数目为 10；因此满足条件 1 的值为 1、2、4 和 8。
此外，满足条件 2 的值为 1、2、4 和 8，满足条件 3 的值为 8，因此同时选择与擦除的块数目为 8；块 64 至 71 被擦除。

<6> 擦除块 64 至块 71 后下一个开始的块编号为 72 并且要擦除的块数目为 2；因此满足条件 1 的值为 1 和 2。
此外，满足条件 2 的值为 1 和 2，满足条件 3 的值为 2，因此同时选择与擦除的块数目为 2；块 72 和 73 被擦除。

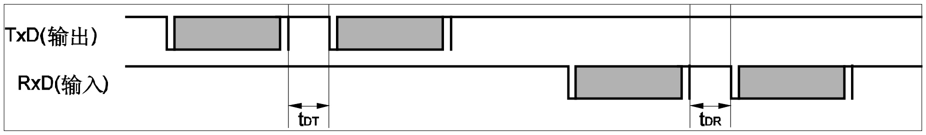
因此同时选择和擦除执行了六次（25，26 和 27，28 至 31，32 至 63，64 至 71，72，73），要擦除块 25 至 73，因此获得 M = 6。

执行同时选择与擦除时块配置（擦除块 25 至 73 时）

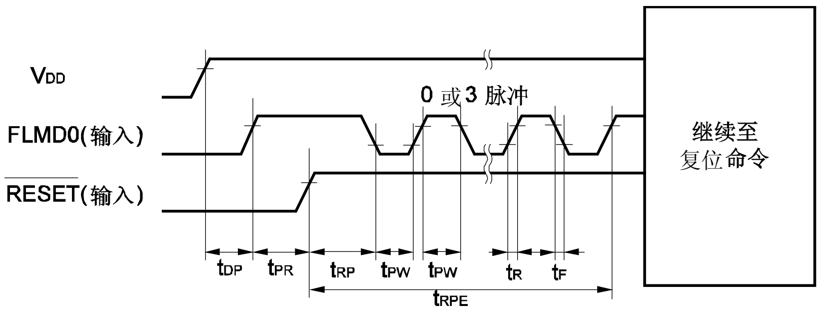


6.4 UART 通讯模式

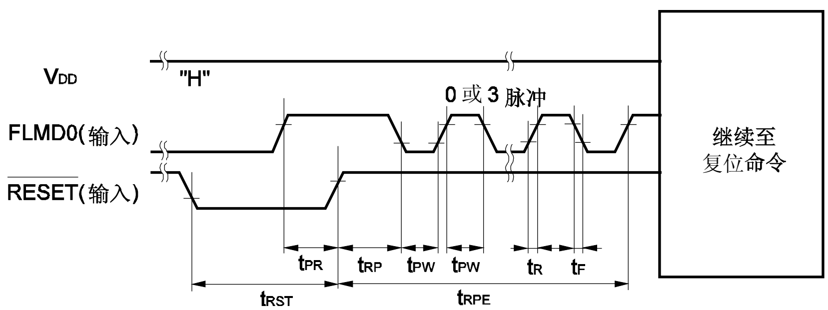
(a) 数据帧



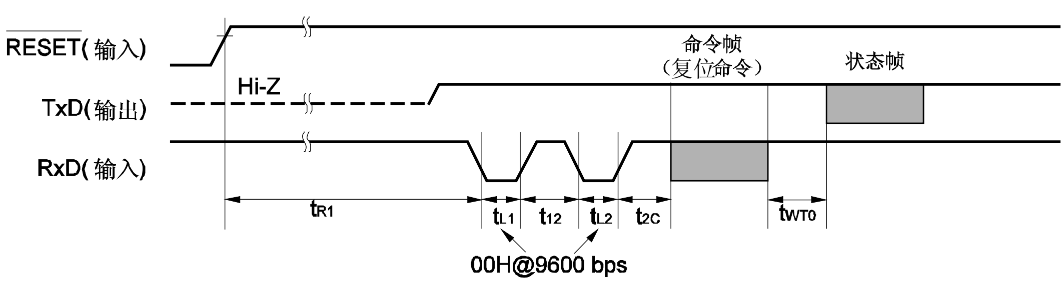
(b) 编程模式设置（加电时）



<R> (c) 编程模式设置（加电后）

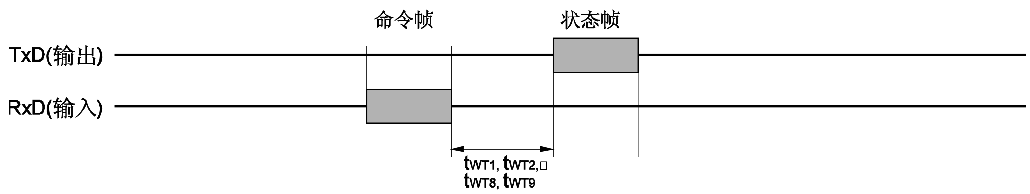


(d) 复位命令

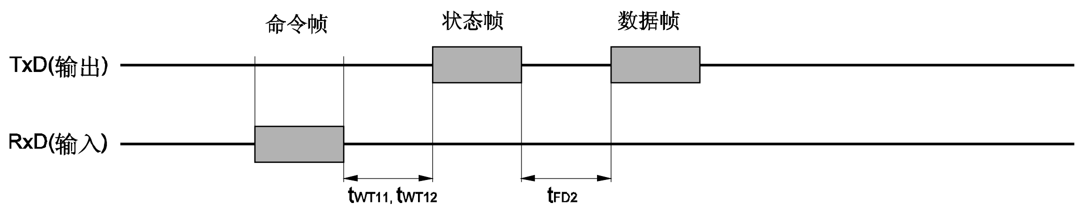


备注 TxD:TxD6
RxD:RxD6

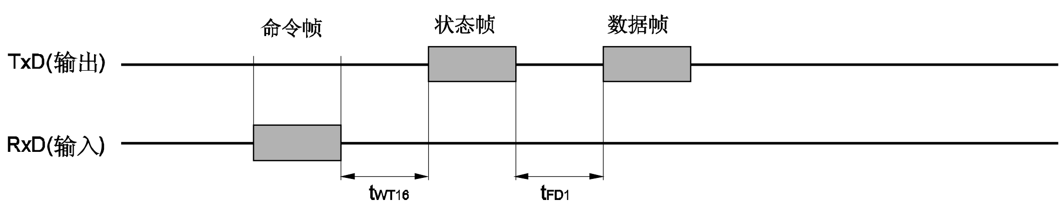
(e) 片擦除命令/块擦除命令/块空白检查命令/振荡设置命令



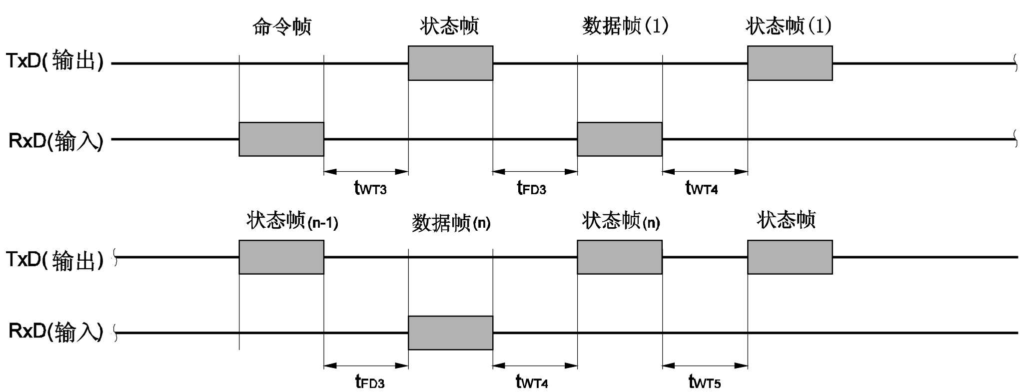
(f) 硅签名命令/版本获取命令



(g) 校验和命令



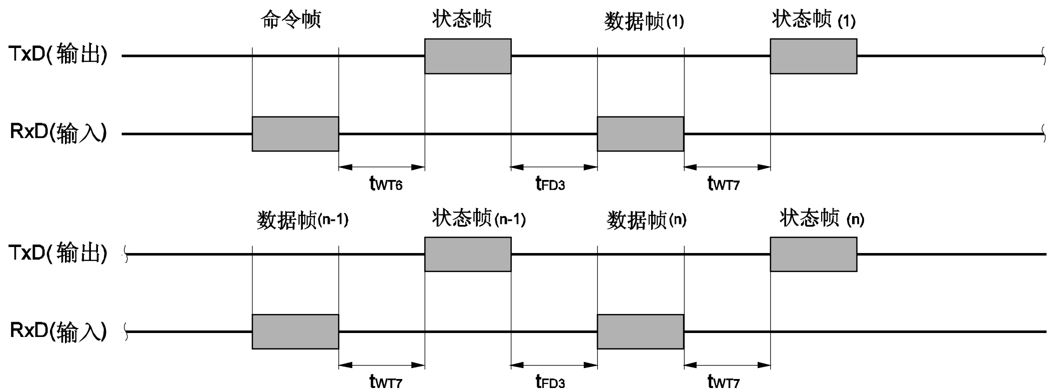
(h) 编程命令



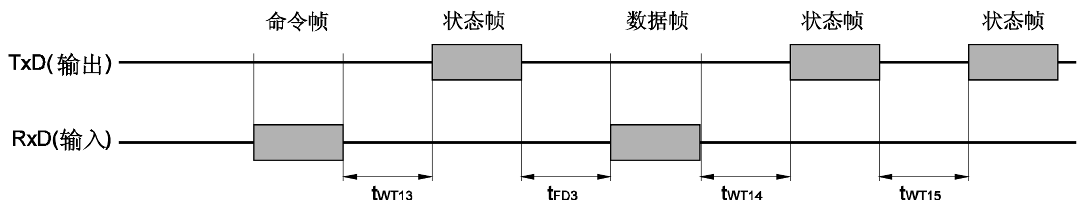
备注 TxD:TxD6

RxD:RxD6

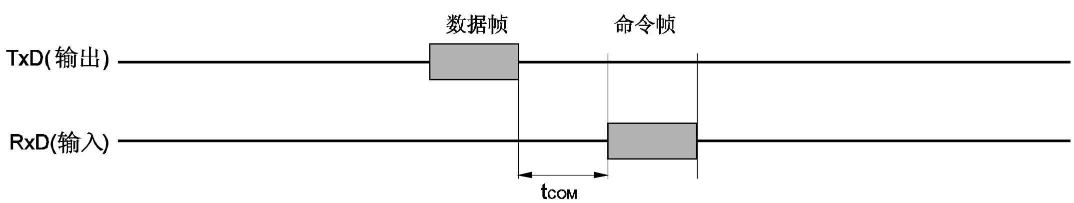
(i) 校验命令



(j) 安全设置命令



(k) 命令帧发送前等待

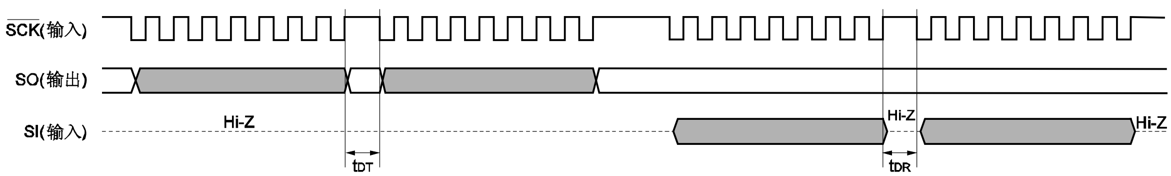


备注 TxD:TxD6

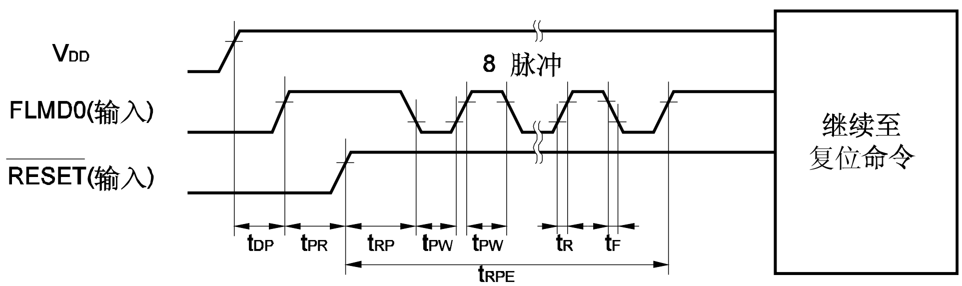
RxD:RxD6

6.5 3 线串行 I/O 通讯模式

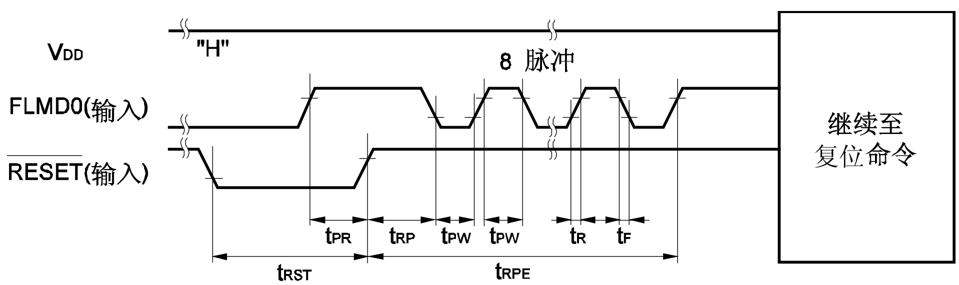
(a) 数据帧



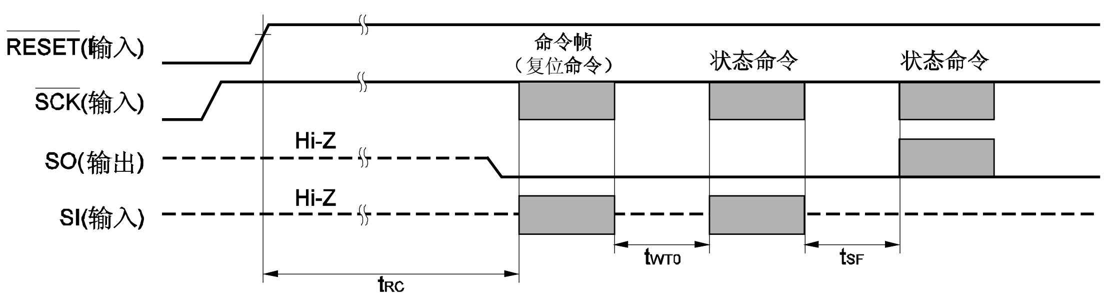
(b) 编程模式设置（加电时）



<R> (c) 编程模式设置（加电后）



(d) 复位命令

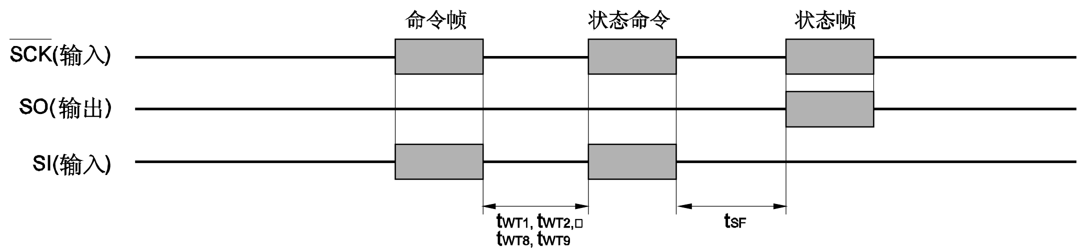


备注 SCK:SCK10

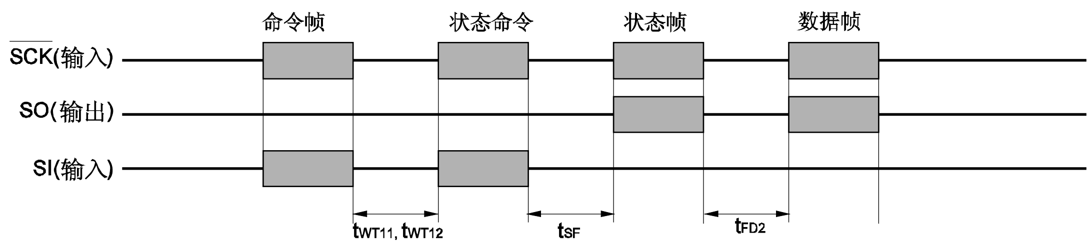
SC:SO10

SI:SI10

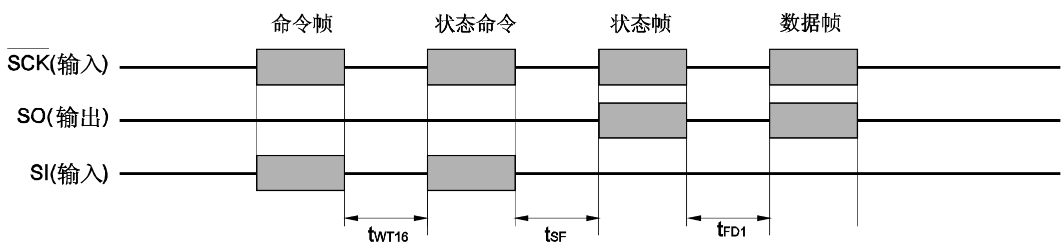
(e) 片擦除命令/块擦除命令/块空白检查命令/振荡设置命令



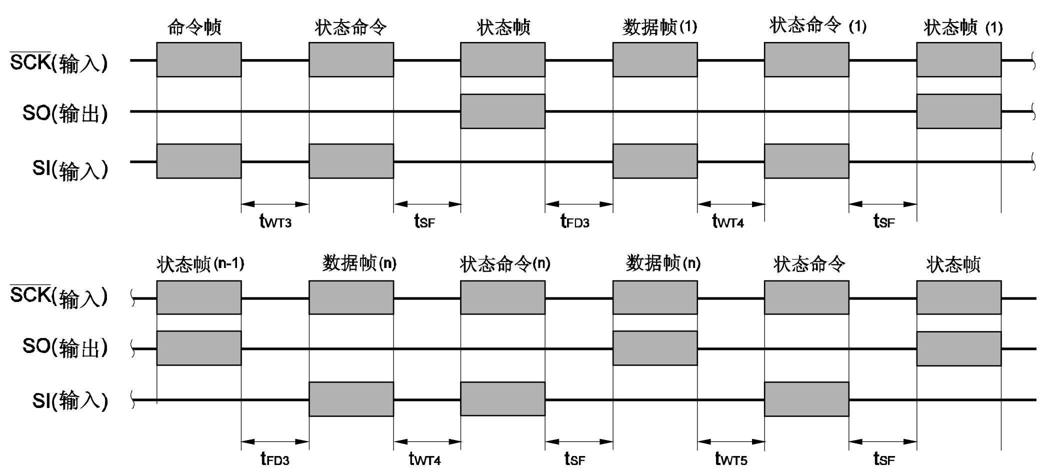
(f) 硅签名命令/版本获取命令



(g) 校验和命令

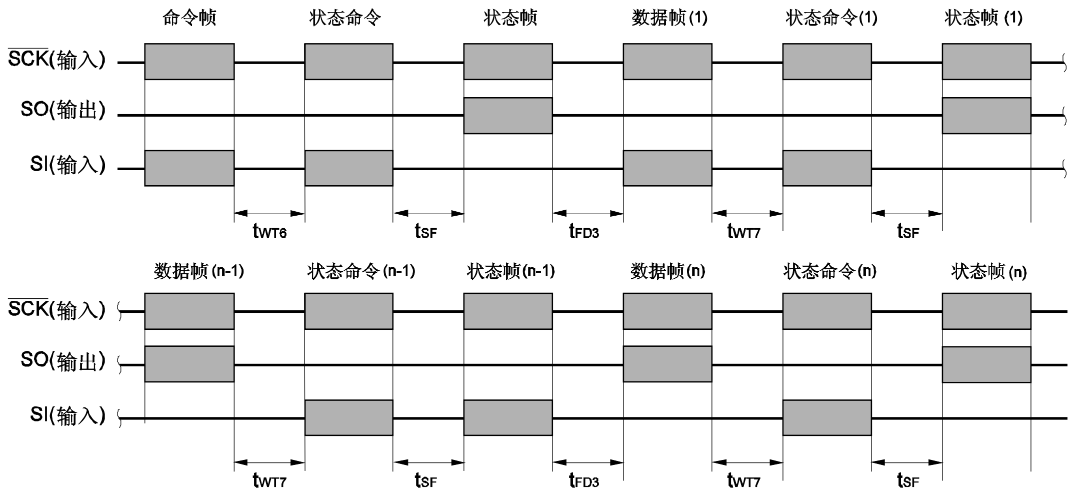


(h) 编程命令

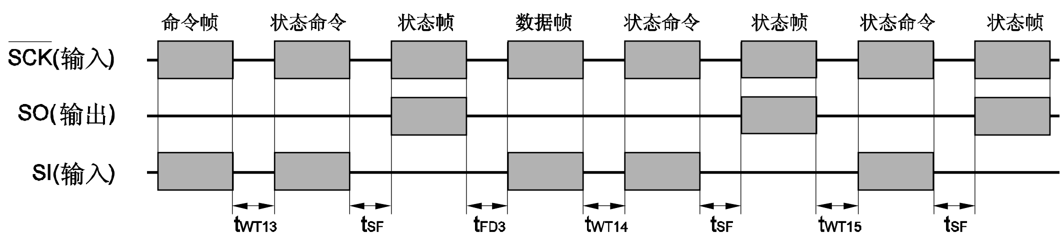


备注 SCK:SCK10
SC:SO10
SI:SI10

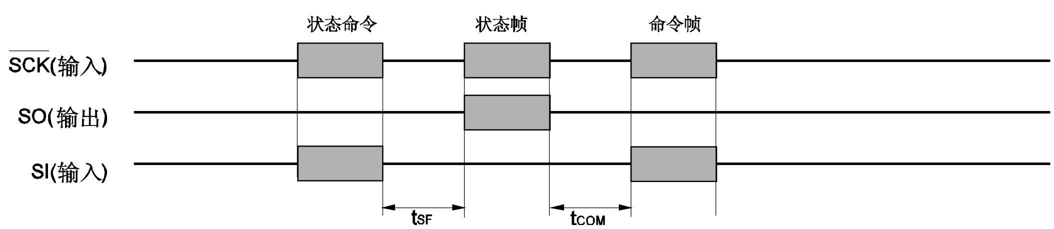
(i) 校验命令



(j) 安全设置命令



(k) 命令帧发送前等待



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

SC:SO10

SI:SI10

附录 A 电路图 (参考)

图 A-1 至 A-3 是 78K0/Kx2 编程器的参考电路图。

[备注]

图 A-1. 78K0/Kx2 编程器参考电路图 (UART 通信时: 使用 X1 时钟)

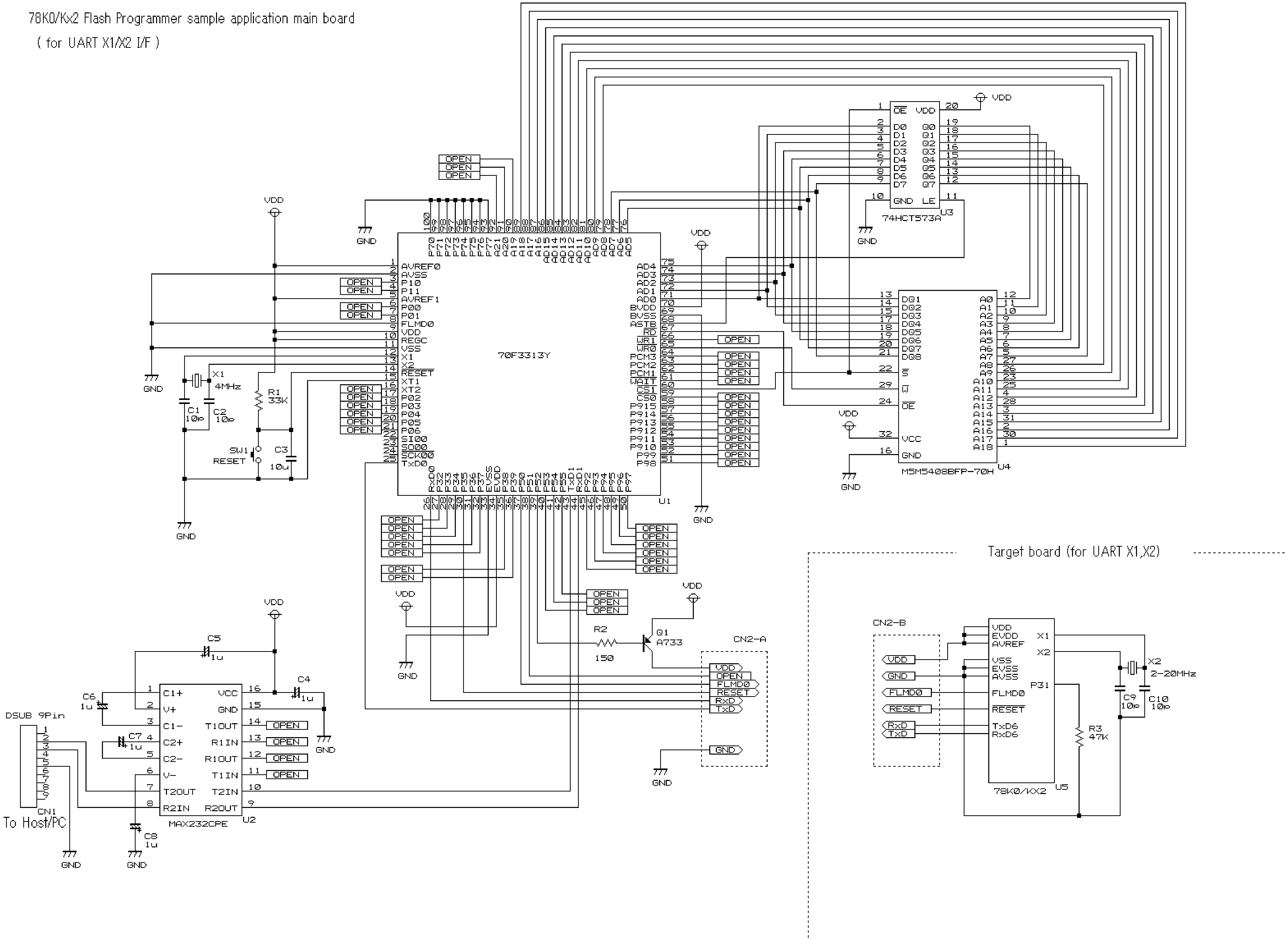


图 A-2. 78K0/Kx2 编程器参考电路图 (UART 通信时: 使用外部时钟)

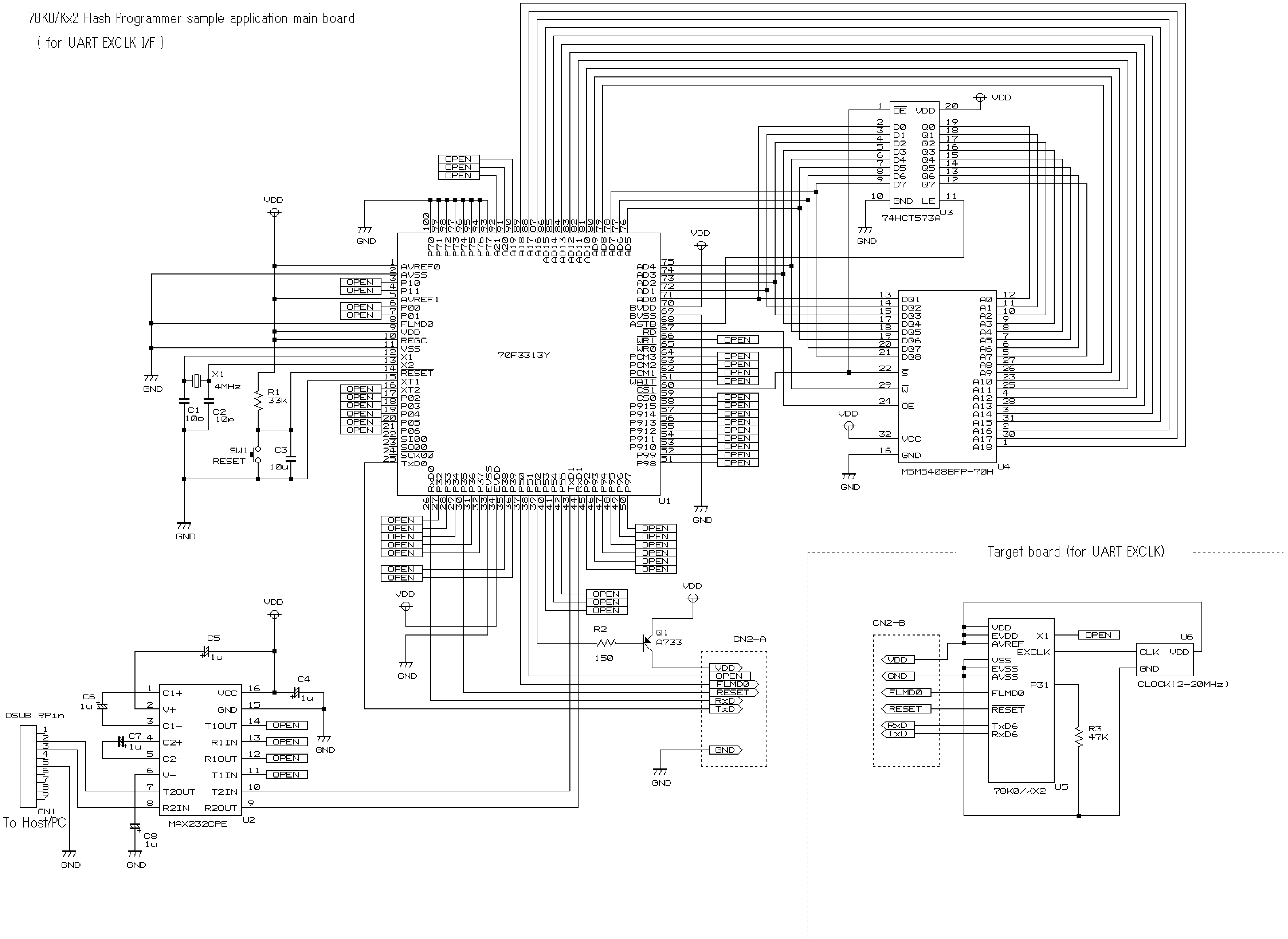
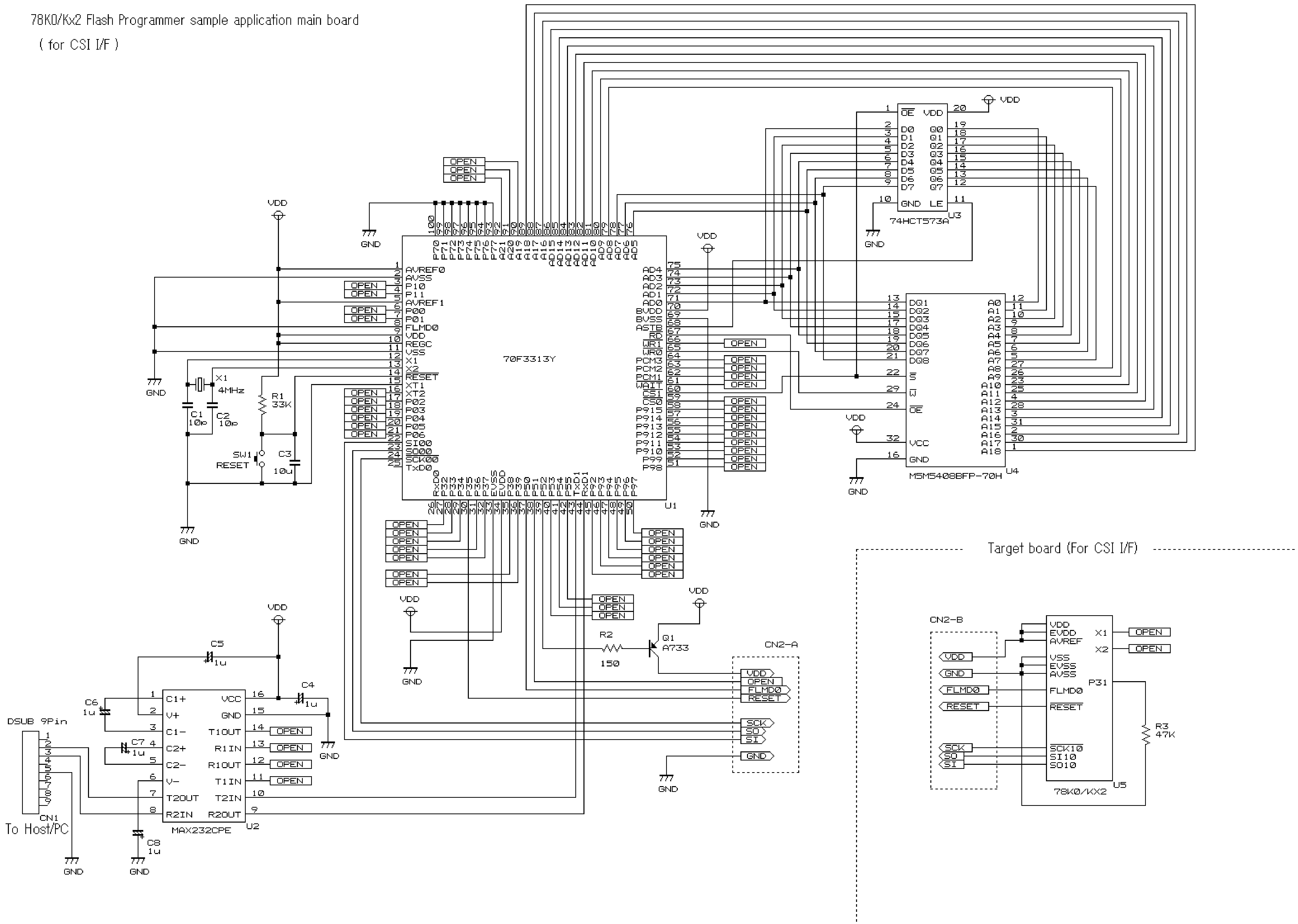


图 A-3. 78K0/Kx2 编程器参考电路图 (CSI 通信时)



附录B　修订历史

B.1 本版本主要修订内容

页码	说明
整篇	增加扩展规格产品
	μPD78F0500A, 78F0501A, 78F0502A, 78F0503A, 78F0503DA, 78F0511A, 78F0512A, 78F0513A, 78F0514A, 78F0515A, 78F0513DA, 78F0515DA, 78F0521A, 78F0522A, 78F0523A, 78F0524A, 78F0525A, 78F0526A,78F0527A, 78F0527DA, 78F0531A, 78F0532A, 78F0533A, 78F0534A, 78F0535A, 78F0536A, 78F0537A, 78F0537DA, 78F0544A, 78F0545A, 78F0546A, 78F0547A, 78F0547DA
	删除 **2.1** 编程器控制引脚 and **2.2** 控制引脚详细资料
	移动 9 个部分(从 **2.3** 基本流程图 到 **2.11** 状态列表) 到 第 **1** 章 **FLASH** 存储器编程器
p.32	修订 **3.5** 芯片擦除命令的描述
p.39	修订 表 **3-1.** 硅信号数据的举例 **(**在 **μPD78F0522 (78K0/KD2)**情况下**)**
pp.41 to 45	修订 **3.10.4 78K0/Kx2** 硅信号列表
pp.52 to 54	• 修订流程图中（从 **4.1** 命令帧发送处理流程图 到 **4.3** 数据帧接收处理流程图）的符号
pp.104 to 106	•修订流程图中（从 **5.1** 命令帧发送处理流程图到 **5.3** 数据帧接收处理流程图）的符号
p.108	修订 **5.4.3** 处理完成的状态
pp.161 to 164	增加 **6.1** 扩展规格产品的 **Flash** 存储器编程器参数特性**(μPD78F05xxA)**
p.164	修订 **6.2.2 Flash** 存储器编程器模式设置时间
pp.165 to 167	修订 **6.2.3** 编程器特性
p.175	增加 **6.4 (c)** 编程器模式设置 **(**上电后**)**
p.178	修订 **6.5 3** 线串行 **I/O** 通讯模式
p.190	增加 **B.2** 修订历史的以前版本

<R> B.2 旧版本的再版修订记录

以下表格所示为旧版本再版修订记录。应用范围表示每个版本修订的内容所在的章节。

(1/2)

版本	先前版本主要修订内容	应用范围:
第二版	修订 **图 2-5. Flash 存储器重写处理基本流程图**	第 2 章 编程器操作环境
	增加 **2.4.1 模式设置流程图**	
	增加 **2.4.2 样例程序**	
	6.8.3 处理完成状态 • 删除在异常终止[D]时 FLMD 错误	第 6 章 UART 通信模式
	6.9.3 处理完成状态 • 删除在异常终止[B]时参数错误的描述	
	6.11.3 处理完成状态 • 增加在异常终止[B]时读取错误	
	修订 **6.14.5 样例程序**	
	增加 7.4.1 处理顺序表的注	第 7 章 3 线串行 I/O 通讯模式 (CSI)
	修订 7.4.2 处理顺序说明 的说明和 **增加 注**	
	增加 7.4.4 流程图 注	
	修订 **7.4.5 样例程序**	
	修订 **7.5.5 样例程序**	
	修订 **7.6.5 样例程序**	
	修订 **7.7.5 样例程序**	
	修订 **7.8.5 样例程序**	
	7.9.3 处理完成状态 • 删除在异常终止[D]时 FLMD 错误	
	修订 **7.9.5 样例程序**	
	修订 **7.10.5 样例程序**	
	修订 **7.11.5 样例程序**	
	7.12.3 处理完成状态 • 增加在异常终止[B]时读取错误	
	修订 **7.12.5 样例程序**	
	修订 **7.13.5 样例程序**	
	修订 **7.14.5 样例程序**	
	修订 **7.15.5 样例程序**	
	修订 8.2 Flash 存储器编程器模式设置时间等待低电平数据 1 (UART)	第 8 章 FLASH 存储器编程器参数特性
	删除**第 9 章电气特性 (参考)**	旧版本 第 9 章电气特性 (参考)

(2/2)

版本	先前版本主要修订内容	应用范围:
第二版	修订 **图 A-1. 78K0/Kx2 编程器参考电路图 (UART 通信时: 使用 X1 时钟)** 到 **图 A-3. 78K0/Kx2 编程器参考电路图 (CSI 通信时)**	附录 A 电路图 (参考)
	增加 **附录 B 修订历史**	附录 B 修订历史

详细信息请联系：

中国区

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

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

[北京]

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

[上海]

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

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

[深圳]

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

[香港]

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

[成都]

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

[长春]

日电电子（中国）有限公司长春分公司
吉林省长春市朝阳区
西安大路 727 号中银大厦 A 座 1609 室
电话：(+86)431-8859-7533 / 8859-8533
传真：(+86)431-8680-2944

[大连]

日电电子（中国）有限公司长春分公司
大连市中山路 88 号天安国际大厦 2701 室
电话：(+86)411-8230-8815 / 8230-8825
传真：(+86)411-8230-8835