NEC

应用笔记

# 78K0/Lx2

## 8 位单片机控制器

## Flash 存储器编程 (编程器)

μPD78F0361
μPD78F0362
μPD78F0363
μPD78F0363D
μPD78F0372
μPD78F0373
μPD78F0374
μPD78F0375
μPD78F0376
μPD78F0376D
μPD78F0382
μPD78F0383
μPD78F0384
μPD78F0385
μPD78F0386
μPD78F0386D
μPD78F0393
μPD78F0394
μPD78F0395
μPD78F0396
μPD78F0397
μPD78F0397D

文档编号：U18204CA1V0AN00 （第一版）
发行日期：2007 年 3 月 NS CP（K）

**[备忘录]**

## CMOS设备的注释

**① 输入引脚处的电压波形**

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

**② 未使用的输入引脚的处理**

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

**③ ESD防护措施**

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

**④ 初始化之前的状态**

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

**⑤ 电源开关顺序**

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

对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

**⑥ 电源关闭状态下的输入信号**

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

每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断。

- **本文档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。本文件所登载内容的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权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或为日本电气电子株式会社（定义如上）开发或制造的产品。

M8 02.11-1

# 引言

**读者对象**

本手册适用于那些希望了解78K0/Lx2功能，并设计开发应用系统和程序的工程师。

**目的**

本手册的目的是帮助用户去了解怎样用专用flash存储器编程器去重写78K0/Lx2的内部flash存储器。

本文档中的例子程序和电路图仅供参考，并非供给实际设计所使用。
因此，若使用这些例子程序，请用户自己承担风险。使用这些例子程序并不保证可以正确操作。

**组件**

本手册所包含的主要组件

- Flash 存储器编程
- 编程器操作环境
- 编程器基本操作
- 命令/数据帧格式
- 指令处理描述
- UART 通讯模式
- 3线串行 I/O 命令模式 (CSI)
- Flash 存储器编程参数特性

**手册使用方法**

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方面的一般知识。

- 如果读者要了解产品功能：
  → 请按**目录**顺序阅读本手册。
- 为了了解更多的关于78K0/Lx2的硬件功能：
  → 可参阅每个78K0/Lx2的用户手册。

**规定**

| | | |
|---|---|---|
| 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 | |
| 数的表示法: | 二进制 | ···××××或××××B |
| | 十进制 | ···×××× |
| 十六进制 | ××××H | |

**相关文档**　　本手册中指出的相关文档可能包括了初级的版本，但未注明。

设备文档

| 文档名称 | 文档编号 |
|---|---|
| 78K0/LE2用户手册 | U17734E |
| 78K0/LF2用户手册 | U17504E |
| 78K0/LG2用户手册 | U17473E |
| 78K/0系列指令用户手册 | U12326E |

注意事项　对以上列出的相关文档所做修改恕不另行通知，在设计时请使用每个文档的最新版本。

目录

# 第一章 FLASH 存储器编程

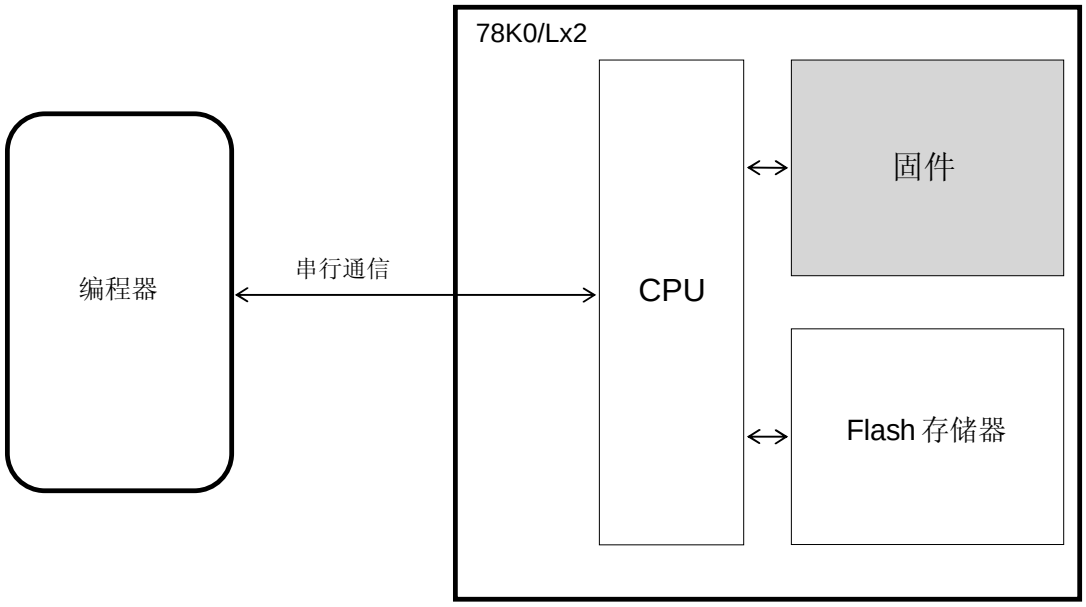
为了重写78K0/Lx2内部Flash 存储器的内容，通常使用一个专用Flash 存储器编程器（以后我们用“编程器”来代替）。

本应用笔记将说明如何去开发一个专用的编程器。

## 1.1 概述

78K0/ Lx2 含有控制 Flash 存储器编程的固件。内部 Flash 存储器编程执行发送/接收命令，使编程器和 78K0/ Lx2 进行串行通信。

**图 1-1. 78K0/ Lx2 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。

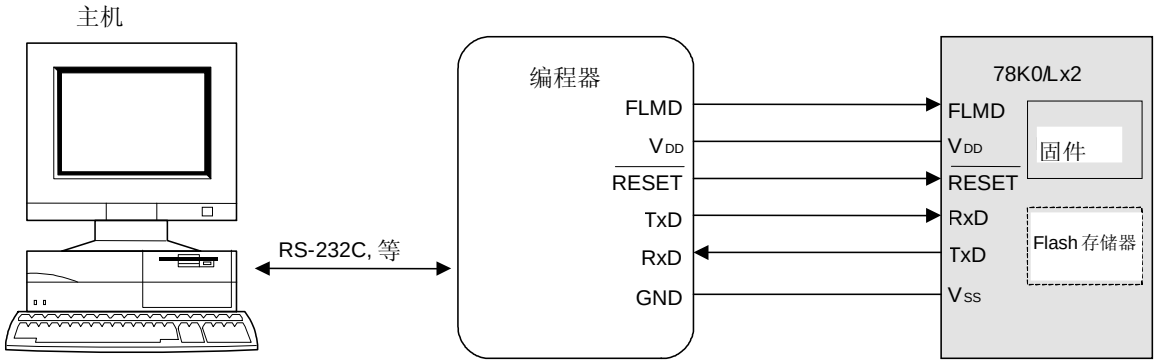
这些图例显示了主机控制下的编程器Flash 存储器编程。

若编程器已经连接，用户程序已下载到编程器，则编程器可以在无主机的独立模式下使用。

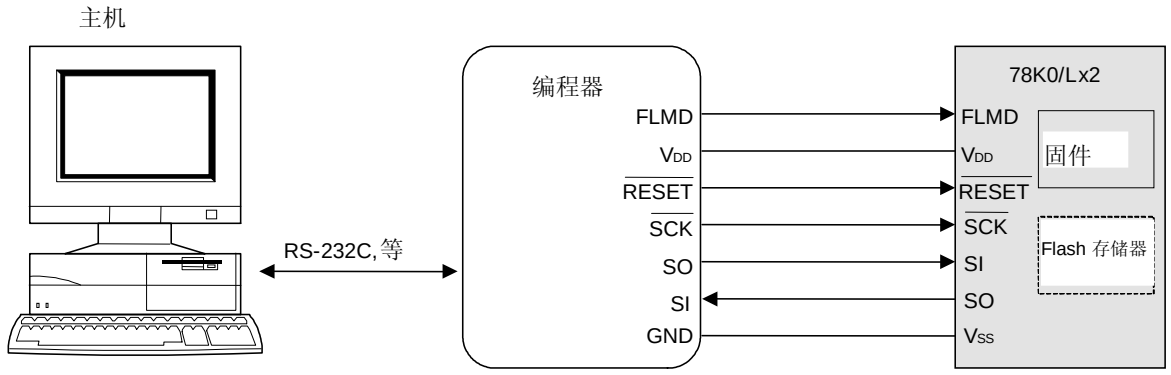
例如， NEC Electronics' Flash 存储器编程器 PG-FP4 可以在主机连接使用 GUI 软件或自己（独立的）执行编程。

**图 1-2. 系统配置示例**

**(1) UART 通信模式 (LSB 优先传输)**




**(2) 3 线串行 I/O 通信模式 (CSI) (MSB 优先传输)**

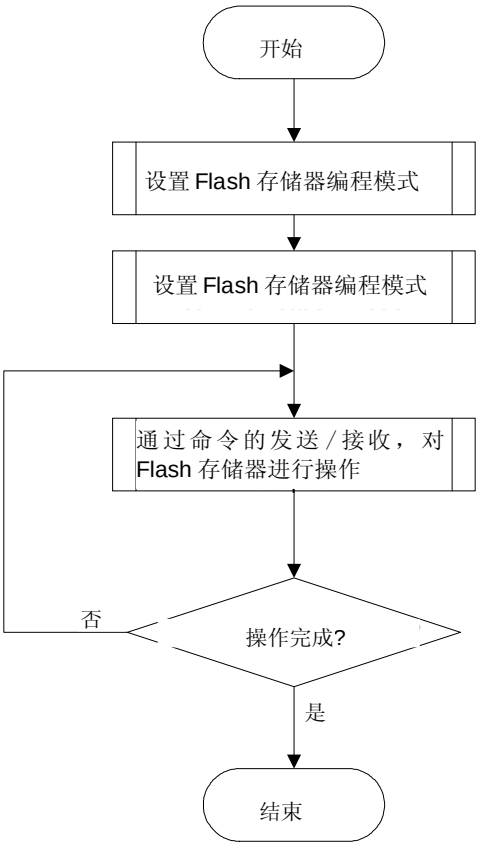


## 1.3 编程概述

为了使用编程器重写Flash存储器的内容，78K0/ Lx2 必须首先被设置为Flash存储器编程模式。然后，选择编程器与78K0/ Lx2 之间的通信模式，通过串行通信从编程器发送命令，然后重写Flash存储器。

图 1-3 为编程的流程图。

**图 1-3. 编程流程图**




### 1.3.1 设置 flash 存储器编程模式

提供一个特定电压给 78K0/ Lx2 Flash 存储器编程模式的设置引脚(FLMD0)，并复位一次，这样就设置了 Flash 存储器编程模式。

### 1.3.2 选择串行通信模式

为了选择一个串行通信模式，在 Flash 存储器编程模式下，通过改变 $V_{DD}$ 和 GND 电压间的 Flash 存储器编程模式设定引脚（FLMD0）的电压产生脉冲，根据脉冲计数判断通信模式。

### 1.3.3 通过命令发送/接收操作 Flash 存储器

具有 Flash 存储器功能的 78K0/Lx2 可以重写 Flash 存储器的内容。操作功能显示如下表 1-1 。

**表 1-1. Flash 存储器功能概述**

| 功能 | 概述 |
|---|---|
| 擦除 | 擦除Flash存储器内容 |
| 写入 | 向Flash存储器写入数据 |
| 校验 | 比较Flash存储器内容与数据，用于校验 |
| 获得信息 | 读取Flash存储器的相关信息 |

通过串行通信，编程器传输命令给78K0/Lx2来控制这些内容。78K0/Lx2 返回命令响应状态。Flash存储器通过反复进行串行通信来进行编程。

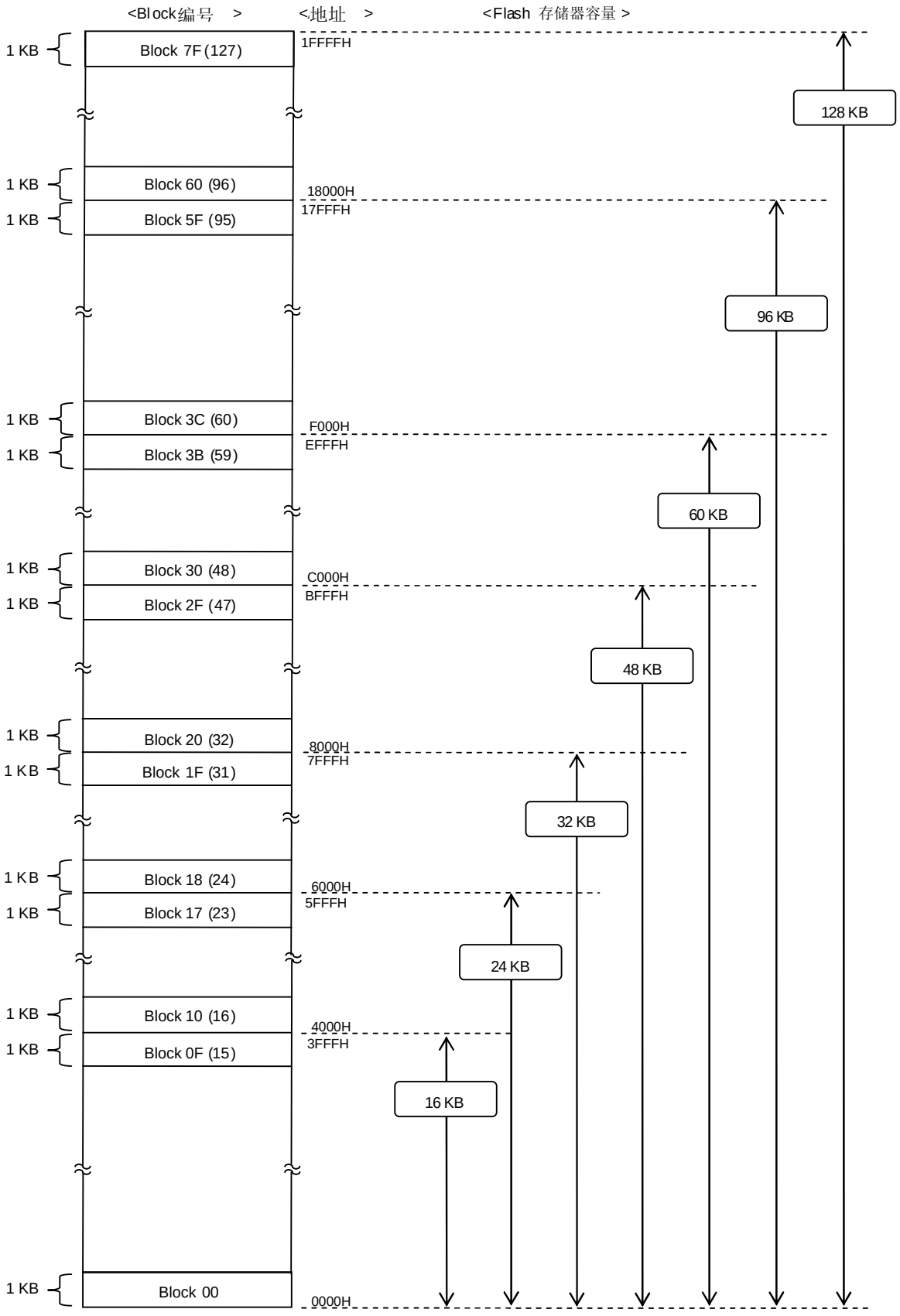
## 1.4 78K0/Lx2 信息细节

编程器必须管理产品特定信息 (例如器件名称和存储器信息)。
表 1-2 显示了 78K0/Lx2 的 Flash 存储器容量，图 1-4 显示了 Flash 存储器结构。

**表 1-2. 78K0/Lx2 的 Flash 存储器容量**

| 器件名称 | | Flash 存储器容量 |
|---|---|---|
| 78K0/LE2 | μPD78F0361 | 16 KB |
| | μPD78F0362 | 24 KB |
| | μPD78F0363, 78F0363D | 32 KB |
| 78K0/LF2 | μPD78F0372 | 24 KB |
| | μPD78F0373 | 32 KB |
| | μPD78F0374 | 48 KB |
| | μPD78F0375 | 60 KB |
| | μPD78F0376, 78F0376D | 96 KB |
| | μPD78F0382 | 24 KB |
| | μPD78F0383 | 32 KB |
| | μPD78F0384 | 48 KB |
| | μPD78F0385 | 60 KB |
| | μPD78F0386, 78F0386D | 96 KB |
| 78K0/LG2 | μPD78F0393 | 32 KB |
| | μPD78F0394 | 48 KB |
| | μPD78F0395 | 60 KB |
| | μPD78F0396 | 96 KB |
| | μPD78F0397, 78F0397D | 128 KB |

**图 1-4. Flash 存储器配置**




**备注** 每个 block 由 1 KB 组成(此图仅举例显示 Flash 存储器中整个 block 的一部分)。

# 第二章　编程器操作环境

## 2.1 编程器控制引脚

表 2-1 列出了在用户系统中必须控制的执行编程器功能的编程器引脚。引脚详细情况请参见下页。

**表 2-1. 引脚描述**

| 编程器 | | | 78K0/Lx2 | 目标通讯模式 | |
|---|---|---|---|---|---|
| 信号名称 | I/O | 引脚功能 | 引脚名称 | CSI | UART |
| FLMD0 | 输出 | 设置编程模式信号电平输出和选择通讯模式脉冲输出 | FLMD0 | ○ | ○ |
| $V_{DD}$ | 输出 | $V_{DD}$ 电压产生/监测 | $V_{DD}$<br>$LV_{DD}$<br>$AV_{REF}$ | △ | △ |
| GND | – | 地 | $V_{SS}$<br>$LV_{SS}$<br>$AV_{SS}$ | ○ | ○ |
| CLK | 输出 | 78K0/Lx2操作时钟输出 | EXCLK | ×注1 | △注2 |
| $\overline{RESET}$ | 输出t | 编程模式触发开关 | $\overline{RESET}$ | ○ | ○ |
| SO | 输出 | 78K0/Lx2命令传输 | SI10 | ○ | × |
| SI | 输入 | 78K0/Lx2状态响应和数据接收 | SO10 | ○ | × |
| $\overline{SCK}$ | 输出 | 78K0/Lx2串行时钟 | $\overline{SCK10}$ | ○ | × |
| TxD | 输出 | 78K0/Lx2命令传输 | RxD6 | × | ○ |
| RxD | 输入 | 78K0/Lx2状态响应和数据接收 | TxD6 | × | ○ |

**注 1.** 在使用CSI10时，78K0/Lx2使用内部高速振荡时钟 ($f_{RH}$) 。

**2.** 在使用UART6时，必须使用X1 时钟 ($f_X$) 或外部主系统时钟 ($f_{EXCLK}$)。
若要使用 Flash 编程器的时钟输出，连接编程器的 CLK 到 EXCLK。

**注意事项 1. 当使用振荡器时，P31/INTP2 要下拉。**
**2. 当不使用振荡器时，P31/INTP2 下拉并且 X1 要下拉或悬空。**

**备注**　○: 确保连接引脚。
×: 此引脚不需要连接。
△: 如果在目标板上产生信号，则此引脚不需要连接。

编程器的控制引脚电压，请参见 Flash 存储器编程设备用户手册。

## 2.2　控制引脚的详细资料

### 2.2.1　Flash 存储器编程模式设置引脚(FLMD0)

FLMD0引脚用来控制78K0/Lx2的操作模式。当给此引脚提供特殊电压或复位后，78K0/Lx2 以Flash 存储器编程模式操作 。

编程器和 78K0/Lx2 之间的串行通讯模式，由复位后 FLMD0 引脚的控制电压 $V_{DD}$ 与 GND 之间和输出脉冲决定。表 2-3 显示了 FLMD0 脉冲个数和通讯模式之间的关系。

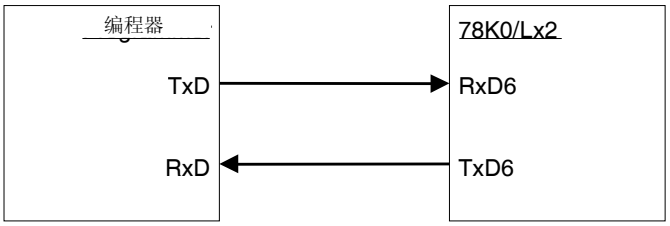
### 2.2.2　串行接口引脚 (TxD, RxD, SI, SO, $\overline{SCK}$)

串行接口引脚用来在编程器和78K0/Lx2间传输Flash存储器的写入命令。

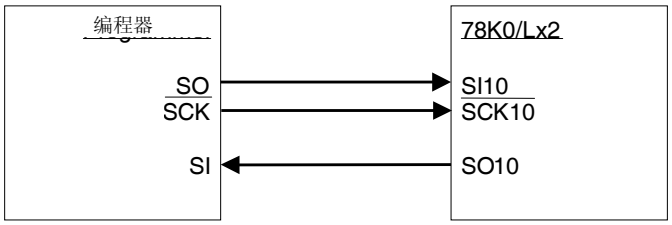
78K0/Lx2 的通讯模式可通过 UART 和 CSI 选择。以下图表举例说明了每种通讯模式所使用的引脚连接。

**图 2-1. 串行接口引脚**

**(1) UART 通信模式**



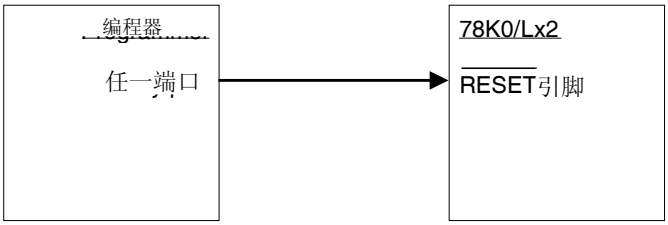

**(2) 3 线串行 I/O 通信模式 (CSI)**



### 2.2.3　复位控制引脚 (RESET)

复位控制引脚用来控制78K0/Lx2系统复位。给FLMD0引脚提供特殊电压并且复位，可以选择Flash存储器编程模式。

**图 2-2. 复位控制引脚**

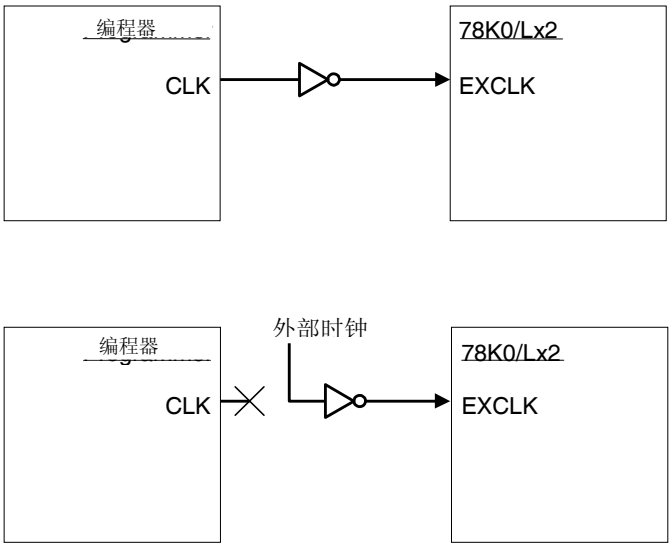



### 2.2.4　时钟控制引脚 (CLK)

时钟控制引脚仅用于从编程器提供时钟到78K0/Lx2。若不需要从编程器提供操作时钟给78K0/Lx2，则这些引脚可以不连接。

**(1) UART 通信模式**

**图 2-3. 时钟控制引脚**




**(2) 3线串行I/O通信模式 (CSI)**

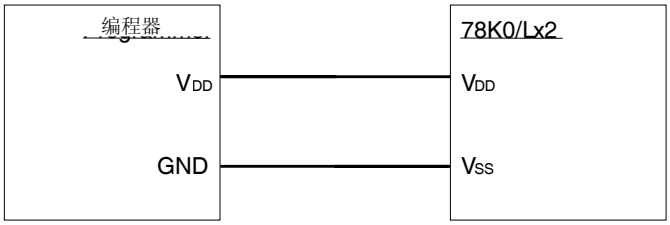
78K0/Lx2使用内部高速振荡时钟($f_{RH}$)。

### 2.2.5　$V_{DD}$/GND 控制引脚

$V_{DD}$ 控制引脚用来给78K0/Lx2提供电源。若不需要从编程器提供操作时钟给78K0/Lx2，则这些引脚可以不连接。因为专用编程器监控78K0/Lx2的电源状态，所以使用专用编程器时，不管是否需要从编程器提供电源，这些引脚都必须连接。

不管是否需要从编程器提供电源，GND 控制引脚必须连接到78K0/Lx2的$V_{SS}$ 。

**图2-4. $V_{DD}$/GND 控制引脚**



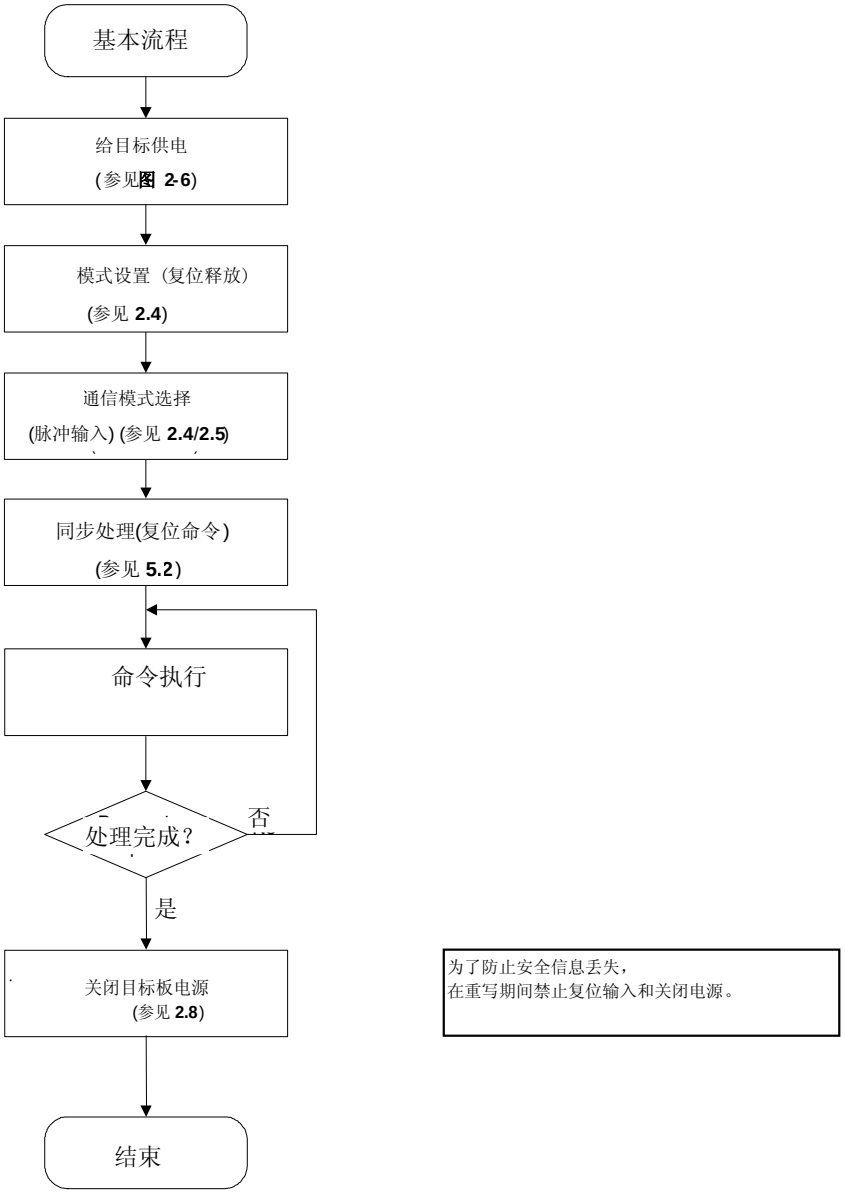

### 2.2.6　其他引脚

这些引脚不需要连接到编程器，请参见每个器件用户手册中的Flash存储器相关章节描述。

## 2.3　基本流程图

以下举例说明编程器重写Flash 存储器的基本流程。

**图 2-5. Flash 存储器重写处理基本流程图**

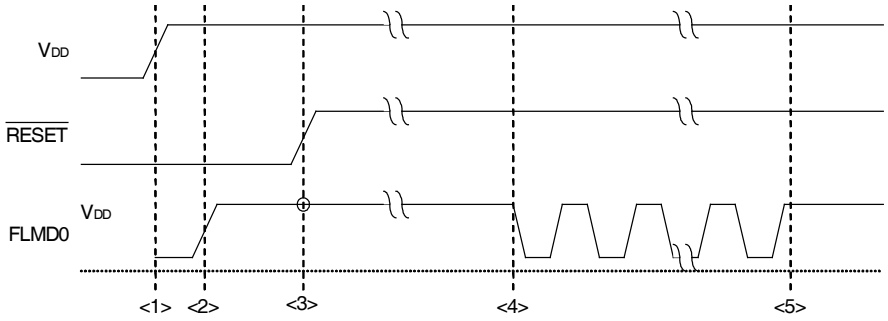


## 2.4　设置 Flash 存储器编程模式

通过编程器重写Flash 存储器的内容，必须先将78K0/Lx2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)提供特殊电压，并复位。

以下举例说明设置Flash 存储器编程模式和选择通讯模式的时序图。

**图 2-6. 设置Flash 存储器编程模式和选择通讯模式**




<1>: 供电电源(VDD)

<2>: FLMD0 = 高

<3>: 复位 (设置模式)

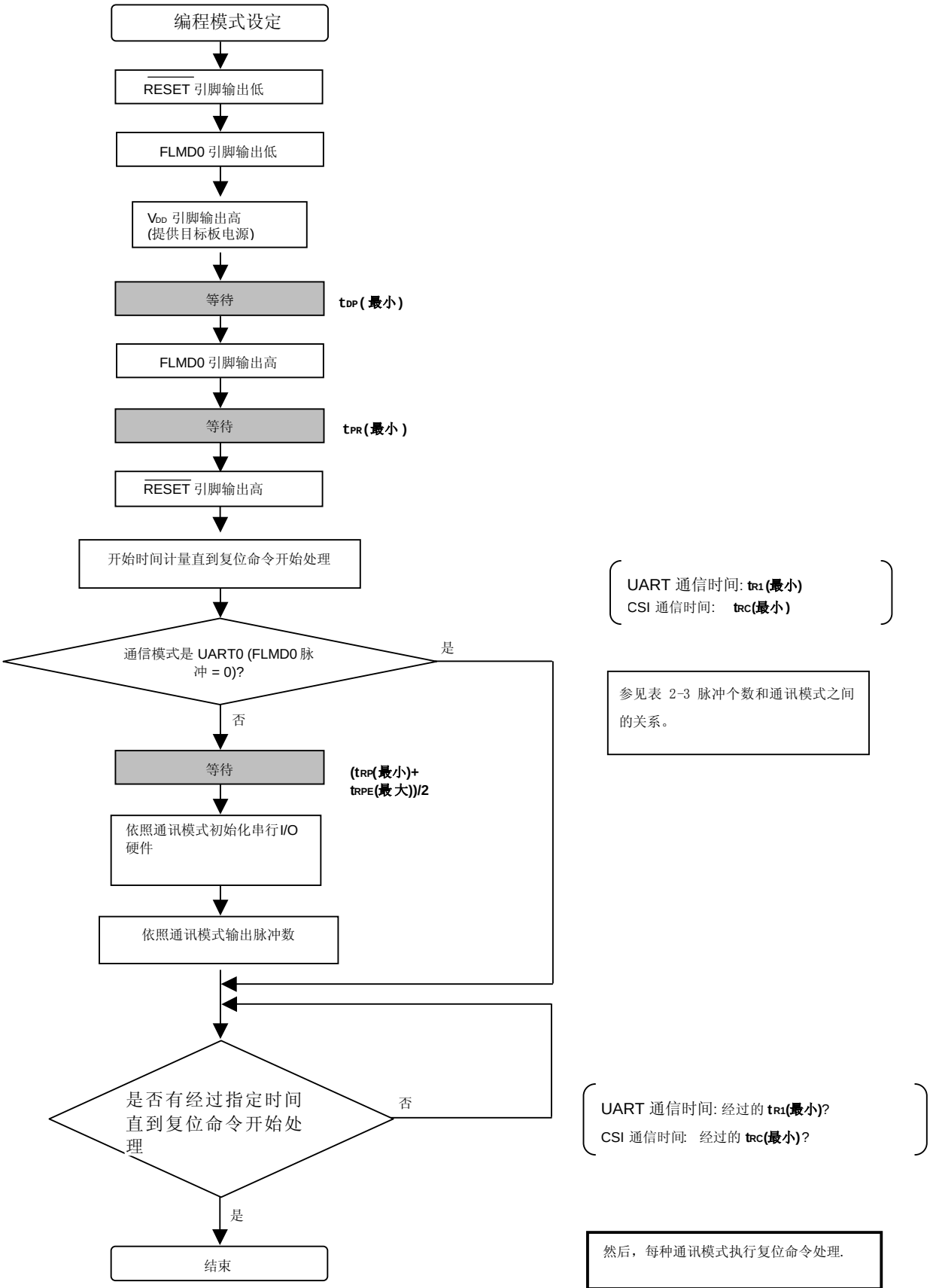
<4>: 脉冲输出开始

以下显示复位后FLMD0 引脚和操作模式之间的关系。

**表 2-2. 复位后FLMD0 引脚和操作模式之间的关系**

| FLMD0 | 操作模式 |
|---|---|
| 低 (GND) | 普通操作模式 |
| 高 (VDD) | Flash 存储器编程模式 |

### 2.4.1　模式设置流程图


编程模式设定
RESET 引脚输出低
FLMD0 引脚输出低
VDD 引脚输出高
(提供目标板电源)
等待
tDP(最小)
FLMD0 引脚输出高
等待
tPR(最小)
RESET 引脚输出高
开始时间计量直到复位命令开始处理
通信模式是 UART0 (FLMD0 脉冲 = 0)?
是
否
等待
(tRP(最小)+ tRPE(最大))/2
依照通讯模式初始化串行I/O硬件
依照通讯模式输出脉冲数
是否有经过指定时间直到复位命令开始处理
否
是
结束
UART 通信时间: tR1(最小)
CSI 通信时间: tRC(最小)
参见表 2-3 脉冲个数和通讯模式之间的关系。
UART 通信时间: 经过的 tR1(最小)?
CSI 通信时间: 经过的 tRC(最小)?
然后，每种通讯模式执行复位命令处理.

### 2.4.2　例子程序

以下例子程序显示了模式设置处理。

```
/*****************************************************************/
/*                                                               */
/* 连接到Flash器件                                               */
/*                                                               */
/*****************************************************************/
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

          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"等待定时器
          fl_wait((tRP+tRPE)/2);

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

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

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

        // 开始 RESET 命令 proc.

}
```

## 2.5　选择串行通信模式

78K0/Lx2通讯模式由FLMD0 引脚的输入脉冲决定，复位后设置Flash存储器编程模式。

FLMD0 的高低脉冲分别为$V_{DD}$和GND。

以下表格显示了通过78K0/Lx2选择的FLMD0脉冲个数（脉冲数）和通讯模式之间的关系。

**表2-3. FLMD0 脉冲个数和通讯模式之间的关系**

## 2.6　UART 通信模式

| 通信模式 | FLMD0脉冲计数 | 用于通信的端口 |
|---|---|---|
| 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) | |
| 3线串行I/O (CSI10) | 8 | SO10 (P12), SI10 (P11), $\overline{SCK10}$ (P10) |
| 设置禁止 | 其他 | ~ |

UART 通讯使用RxD和TxD引脚。通讯条件如下。

**表 2-4. UART 通信条件**

| 项目 | 说明 |
|---|---|
| 波特率 | 直到振荡频率设置命令已经传输，通讯一直以9,600 bps 执行。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 | None |
| 数据长度 | 8 位 (LSB 优先) |
| 停止位 | 1 位 |

CSI 通讯时，编程器总是以主设备运行，所以无论78K0/Lx2运行是否正常完成（例如写入或擦除），编程器都必须检查。另外, UART通讯期间主从设备状态有时候互换，所有此时通讯有可能是最适宜的时间。

**注意事项　执行UART通讯时，设置主从设备相同的波特率。**

## 2.7　3 线串行 I/O 通信模式 (CSI)

CSI 通讯使用SCK 、SO和SI 引脚。编程器始终为主设备，如果78K0/Lx2未读取经SCK引脚的传输数据，有可能通讯不能正常执行。

通讯数据格式为MSB优先，8位。保持时钟频率在2.5 MHz 或更低。

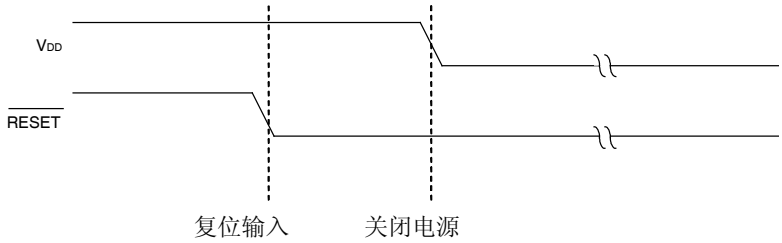
## 2.8　关闭目标板电源

每个命令执行完后，复位后关闭目标板的电源。

当关闭目标板的电源，设置其他引脚为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 2-7. Flash 存储器编程模式终止时序**



## 2.9 Flash 存储器的操作

含Flash存储器的78K0/Lx2具有操作Flash存储器的功能，如表2-5。编程器发送命令给78K0/Lx2控制这些功能，并且检查78K0/Lx2发出的响应状态，以此来操作Flash存储器。

**表 2-5. Flash存储器使用功能列表**

| 分类 | 功能名称 | 描述 |
|---|---|---|
| 擦除 | 芯片擦除 | 擦除全部Flash存储器区域。清除安全标志。 |
| | Block 擦除 | 擦除Flash存储器的指定block。 |
| 写入 | 写入 | 在Flash存储器指定区域写入数据。 |
| 校验 | 校验 | 比较Flash存储器指定地址获取的数据与编程器传输的数据 |
| 空白检测 | Block 空白检测 | 检查Flash存储器指定区域的擦除状态 |
| 信息获取 | 硅标记获取 | 获取写入协议信息 |
| | 版本获取 | 获取78K0/Lx2 和固件版本信息 |
| | 状态获取 | 获取当前的操作状态 |
| | 校验和获取 | 获取指定区域校验和数据 |
| 安全 | 安全设置 | 设置安全信息 |
| 其他 | 复位 | 侦察同步通信 |

## 2.10 命令列表

以下列表说明了编程器使用的命令及其功能。

**表 2-6. 编程器给 78K0/Lx2 传送的命令列表**

| 命令数值 | 命令名称 | 功能 |
|---|---|---|
| 70H | 状态 | 获取当前的操作状态 (状态数据) |
| 00H | 复位 | 通讯同步检测 |
| 90H | 设置振荡频率 | 指定78K0/Lx2振荡频率 |
| 20H | 芯片擦除 | 擦除全部Flash存储器区域 |
| 22H | Block 擦除 | 擦除Flash存储器的指定区域 |
| 40H | 编程 | 在Flash存储器指定区域写入数据 |
| 13H | 校验 | 比较Flash存储器指定区域的内容与编程器传输的数据 |
| 32H | Block 空白检测 | 检查Flash存储器指定区域的擦除状态 |
| C0H | 硅标记 | 获取78K0/Lx2 信息(产品编号、Flash 存储器结构、等) |
| C5H | 获取版本 | 获取78K0/Lx2 和固件版本信息 |
| B0H | 校验和 | 获取指定区域的校验和数据 |
| A0H | 安全设置 | 设置安全信息 |

## 2.11 状态列表

以下列表显示了编程器从78K0/Lx2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 | 如果接收到不支持的命令返回错误 |
| 05H | 参数错误 | 如果命令信息(参数)无效返回错误 |
| 06H | 常规应答<br>(ACK) | 常规应答 |
| 07H | 校验和错误 | 如果编程器传输帧的数据异常时返回错误 |
| 0FH | 校验错误 | 从编程器传送校验数据时产生校验误差返回错误 |
| 10H | 保护错误 | 由于安全设置命令而禁止执行处理返回错误 |
| 15H | 否定应答<br>(NACK) | 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 | 当读取安全信息失败时的错误返回 |
| FFH | 处理中<br>(BUSY) | 繁忙响应[注] |

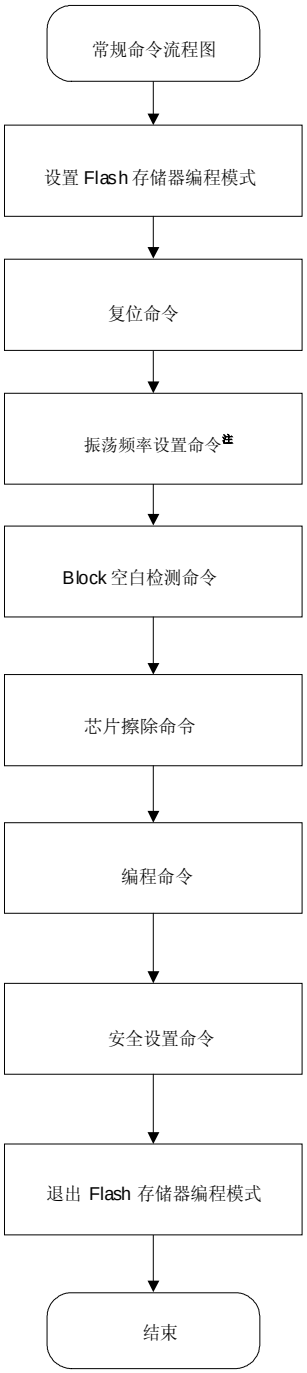
**注**　CSI通信期间，1字节 “FFH” 作为数据帧格式传送。

本手册将接收校验和错误或NACK作为非正常终止处理。当作为专用编程器时，传送命令前重审处理可能造成无问题等待的校验和错误或NACK。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(BUSY 超时或UART通讯期间数据帧接收超时)，将关闭78K0/Lx2供电电压并再次连接。 (详情参见**2.8 关闭目标电源**)

图 3-1 举例说明了当 Flash 存储器重写时编程器常规命令执行流程。

**图 3-1. Flash 存储器重写常规命令执行流程**




**注**　在 CSI 通讯模式期间，当 flash 存储器写入时，78K0/Lx2 不需要执行此命令。

**备注**　可支持校验命令和校验和命令。

# 第四章 命令/数据帧格式

编程器使用命令帧来传输命令给78K0/Lx2。78K0/Lx2使用数据帧传输写入数据或校验数据给编程器。每帧都附加头、尾、数据长度和校验和，以增加传输数据的可靠性。

以下显示了命令帧和数据帧的格式。

**图 4-1. 命令帧格式**

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

**图 4-2. 数据帧格式**

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

**表 4-1. 每帧符号描述**

| 符号I | 值 | 描述 |
|---|---|---|
| SOH | 01H | 命令帧头 |
| STX | 02H | 数据帧头 |
| LEN | ˜ | 数据长度信息 (00H 表示 256).<br>命令帧： COM + 命令信息长度<br>数据帧： 数据长度 |
| COM | ˜ | 命令编号 |
| SUM | ˜ | 帧校验和数据<br>获取所有计算目标数据从最初值1个字节(00H)开始(忽略借位)。计算内容如下。<br>命令帧： LEN + COM + 全部命令信息<br>数据帧： LEN + 全部数据 |
| ETB | 17H | 倒数第二帧数据 |
| 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]**

举例中的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，校验和数据按如下计算。

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

最后传输的命令帧如下。

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

**[数据帧]**

以下显示传输的数据帧，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，校验和数据按如下计算。

00H (初始值)˜ 04H (LEN)˜ FFH (D1)˜ 80H (D2)˜ 40H (D3)˜ 22H (D4)

= 1BH (忽略借位，仅低8位)

最后传输的数据帧如下。

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

当接收到数据帧，校验和数据以同样的方式计算，并且将获得的值用来判断校验和错误，无论此值是否与接收数据中的 SUM 区域存储的值相同。举例说明，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 |

如果正常，应该为 1BH

## 4.1 命令帧传输处理

阅读以下章节，获取每种通讯模式下传输命令帧，命令处理流程图的详细资料。

- UART通讯模式，阅读**6.1 命令帧传输处理流程图**
- 3线串行I/O通讯模式(CSI)，阅读**7.1 命令帧传输处理流程图**

## 4.2 数据帧传输处理

写入数据帧(用户程序)，校验数据帧(用户程序)，和安全数据帧(安全标志)按数据帧传输。
阅读以下章节，获取每种通讯模式下传输数据帧，命令处理流程图的详细资料。

- UART通讯模式，阅读**6.2 数据帧传输处理流程图**
- 3线串行I/O通讯模式(CSI)，阅读**7.2 数据帧传输处理流程图**

## 4.3 数据帧接收处理

状态帧，硅标记数据帧，版本数据帧，和校验和数据帧按数据帧接收。
阅读以下章节，获取每种通讯模式下接收数据帧，命令处理流程图的详细资料。

- UART通讯模式，阅读**6.3 数据帧接收处理流程图**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.3 数据帧接收处理流程图**

# 第五章 命令处理描述

## 5.1 状态命令

### 5.1.1 描述

此命令用来检查传输完每条命令（例如写入或擦除）的78K0/Lx2操作状态。

传输状态命令后，若由于通讯问题或类似问题，78K0/Lx2 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。

### 5.1.2 命令帧和状态帧

图 5-1 显示了状态命令的命令帧格式，图 5-2 显示了命令的状态帧。

**图 5-1. 状态命令帧 (从编程器到 78K0/Lx2)**

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

**图 5-2. 状态命令的状态帧 (从 78K0/Lx2 到编程器)**

| STX | LEN | 数据 | | | SUM | ETX |
|---|---|---|---|---|---|---|
| 02H | n | ST1 | … | STn | 02H | n |

**备注 1.** ST1～STn: 状态 #1～ 状态 #n

**2.** 根据传输给 78K0/Lx2 每条命令(例如写入或擦除)的不同而改变状态帧的长度。

阅读以下章节，获取每种通讯模式下编程器和78K0/Lx2间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式不能使用状态命令
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.4 状态命令**

**注意事项 UART 通讯中每条命令（例如写入或擦除）传输后，78K0/Lx2在指定时间内自动返回状态帧。因此不使用状态命令。**

**若 UART 通讯中传输状态命令，将返回命令编号错误。**

## 5.2 复位命令

### 5.2.1 描述

在通讯模式设置后，此命令用来检查编程器和78K0/Lx2间确立的通讯。

当 78K0/Lx2 选择 UART 通讯模式，编程器和 78K0/Lx2 必须设置相同的波特率。然而，78K0/Lx2 可以不需要自己的波特率发生时钟频率($f_X$ 或 $f_{EXCLK}$)，所以可以不必设置波特率。78K0/Lx2 的波特率发生时钟频率可能通过在 9,600 bps 下从编程器发送两次“00H”，测试到“00H”的低电平宽度，再计算两次发送信号的平均值来产生。波特率因此设置，允许通讯中同步检测。

### 5.2.2 命令帧和状态帧

图 5-3 显示了复位命令的命令帧格式，图 5-4 显示了命令的状态帧。

**图 5-3. 复位命令帧 (从编程器到 78K0/Lx2)**

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

**图 5-4. 复位命令的状态帧 (从 78K0/Lx2 到编程器)**

| STX | LEN | 数据 | SUM | ETX |
|---|---|---|---|---|
| 02H | 1 | ST1 | 校验和 | 03H |

**备注** ST1:同步检测结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART通讯模式，阅读**6.4 复位命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.5 复位命令**

## 5.3 波特率设置命令

78K0/Lx2 不支持波特率设置命令。

78K0/Lx2，UART通讯以9,600 bps执行直到振荡频率设置命令传送。
在接收到状态帧后，通讯速率转换成 115,200 bps。此后，通讯速率固定为 115,200 bps。

## 5.4 振荡频率设置命令

### 5.4.1 描述

此命令用来指定UART通讯期间$f_X$ 或$f_{EXCLK}$频率。

78K0/Lx2通过接收包里的频率数据来确定波特率为115,200 bps。

CSI 通讯不需要执行此命令 (若 CSI 通讯期间执行此命令要依据编程器指定，设置频率为 8 MHz)。

**注意事项 直到振荡频率设置命令传输，78K0/Lx2都以9,600 bps执行UART通讯。**
**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。**

### 5.4.2 命令帧和状态帧

图 5-5 显示了振荡频率设置命令的命令帧结构，图 5-6 显示了命令的状态帧。

**图 5-5. 振荡频率设置命令帧 (从编程器到 78K0/Lx2)**

| SOH | LEN | COM | 命令信息 | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|
| 01H | 05H | 90H (振荡频率设置) | D01 | D02 | D03 | 01H | 05H | 90H (振荡频率设置) |

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

可设置10 kHz~100 MHz，但实际传输命令时，设置值依照每种器件而定。

D01~D03 被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 × 0.1 × $10^4$ = 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 × 0.1 × $10^5$ = 10,000 kHz = 10 MHz

**图 5-6. 振荡频率设置命令的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1: 振荡频率设置结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.5 振荡频率设置命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.6 振荡频率设置命令**

## 5.5 芯片擦除命令

### 5.5.1 描述

此命令用来擦除 Flash 存储器的全部内容。另外，芯片擦除处理可以初始化安全设置处理的全部设置信息，只要安全设置未禁止擦除 (参见 **5.13 安全设置命令**)。

### 5.5.2 命令帧和状态帧

图 5-7 显示了芯片擦除命令的命令帧格式，图 5-8 显示了命令的状态帧。

**图 5-7. 芯片擦除命令帧 (从编程器到 78K0/Lx2)**

| SOH | LEN | COM | SUM | ETX |
|---|---|---|---|---|
| 01H | 01H | 20H<br>(芯片擦除) | 校验和 | 03H |

**图 5-8. 芯片擦除命令的状态帧(从 78K0/Lx2 到编程器)**

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

**备注** ST1: 芯片擦除结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.6 芯片擦除命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.7 芯片擦除命令**

## 5.6 Block 擦除命令

### 5.6.1 描述

此命令用来擦除 Flash 存储器的指定 blocks 内容，只要擦除未被安全设置禁止 (参见 **5.13 安全设置命令**)。

### 5.6.2 命令帧和状态帧

图 5-9 显示了 Block 擦除命令的命令帧格式，图 5-10 显示了命令的状态帧。

**图 5-9. Block 擦除命令帧(从编程器到 78K0/Lx2)**

| SOH | LEN | COM | 命令信息 | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|
| 01H | 07H | 22H<br>(Block擦除) | SAH | SAM | SAL | 01H | 07H | 22H<br>(Block擦除) | SAH | SAM |

**备注** SAH, SAM, SAL: Block 擦除开始地址 (任一 block 开始地址)

SAH: 开始地址，高 (位23~16)

SAM: 开始地址，中 (位15~8)

SAL: 开始地址，低 (位 7~0)

EAH, EAM, EAL: Block 擦除结束地址(任一 block 最后地址)

EAH: 结束地址，高 (位23~16)

EAM: 结束地址，中 (位15~8)

EAL: 结束地址，低 (位7~0)

**图 5-10. Block 擦除命令的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1: Block 擦除结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.7 Block 擦除命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.8 Block 擦除命令**

## 5.7 编程命令

### 5.7.1 描述

此命令在写入开始地址和结束地址传输完后，通过写入字节来传输数据。此命令给Flash存储器写入用户程序并内部校验。

写入开始/结束地址仅以block开始/结束地址单位来设置。

若传输最后数据后的状态帧(ST1 和 ST2) 显示 ACK，78K0/Lx2 固件自动执行内部校验。因此，内部校验的状态命令必须传输。

### 5.7.2 命令帧和状态帧

图 5-11 显示了编程命令的命令帧格式，图 5-12 显示了命令的状态帧。

**图 5-11. 编程命令帧 (从编程器到 78K0/Lx2)**

| SOH | LEN | COM | 命令信息 | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|
| 01H | 07H | 40H<br>（编程) | SAH | SAM | SAL | 01H | 07H | 40H<br>（编程) | SAH | SAM |

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

**图 5-12. 编程命令的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1 (a): 命令接收结果

### 5.7.3 数据帧和状态帧

图 5-13 显示了包括数据写入的帧格式，图 5-14 显示了数据的状态帧。

**图 5-13. 写入数据帧 (从编程器到 78K0/Lx2)**

| STX | LEN | 数据 | SUM | ETX/ETB |
|---|---|---|---|---|
| 02H | 00H to FFH<br>(00H = 256) | 写数据 | 校验和 | 03H/17H |

**备注** 写入数据: 要写入的用户程序

**图 5-14. 数据帧的状态帧 (从 78K0/Lx2 到编程器)**

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

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

### 5.7.4 所有数据和状态帧发送完成

图 5-15 显示了传递完全部数据后的状态帧。

**图 5-15. 所有数据发送完成后的状态帧(从 78K0/Lx2 到编程器)**

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

**备注** ST1 (c):内部校验结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.8 编程命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.9 编程命令**

## 5.8 校验命令

### 5.8.1 描述

此命令用来比较指定地址范围的编程器传输数据和78K0/Lx2读入数据，并且检查，无论它们是否匹配。

校验开始/结束地址仅以 block 开始/结束地址单位设置。

### 5.8.2 命令帧和状态帧

图 5-16 显示了校验命令的命令帧格式，图 5-17 显示了命令的状态帧。

**图 5-16. 校验命令帧（从编程器到 78K0/Lx2）**

| SOH | LEN | COM | 命令信息 | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|
| 01H | 07H | 13H<br>(校验) | SAH | SAM | SAL | 01H | 07H | 13H<br>(校验) | SAH | SAM |

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

**图 5-17. 校验命令的状态帧（从 78K0/Lx2 到编程器）**

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

**备注** ST1 (a): 命令接收结果

### 5.8.3 数据帧和状态帧

图 5-18 显示了包括校验数据的帧格式，图 5-19 显示了数据的状态帧。

**图 5-18. 校验数据的数据帧(从编程器到 78K0/Lx2)**

| STX | LEN | 数据 | SUM | ETX/ETB |
|---|---|---|---|---|
| 02H | 00H～ FFH<br>(00H = 256) | 校验数据 | 校验和 | 03H/17H |

**备注** 校验数据：要校验的用户程序

**图 5-19. 数据帧的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1 (b):数据接收检查结果
ST2 (b): 校验结果[注]

**注** 即使指定地址范围内有校验错误发生，ACK 始终返回校验结果。所有校验错误状态体现在最后数据的校验结果中。因此，当指定地址范围的所有校验处理完成，才能检查出发生的校验错误。

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.9 校验命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.10 校验命令**

## 5.9 Block 空白检测命令

### 5.9.1 描述

此命令用来检测Flash存储器指定block的空白状态(擦除状态)。

1 个指定 block 可以作为空白检测开始地址的开始 block 和结束地址的结束 block。继而指定多个 block。

### 5.9.2 命令帧和状态帧

图 5-20 显示了 Block 空白检测命令的命令帧格式，图 5-21 显示了命令的状态帧。

**图 5-20. Block 空白检测命令帧（从编程器到 78K0/Lx2）**

| SOH | LEN | COM | 命令信息 | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|
| 01H | 07H | 32H<br>(Block空白检测) | SAH | SAM | SAL | 01H | 07H | 32H<br>(Block空白检测) | SAH | SAM |

**备注** SAH, SAM, SAL: Block 空白检测开始地址 (任一 block 开始地址)
SAH: 开始地址，高 (位 23~16)
SAM: 开始地址, 中 (位 15~ 8)
SAL: 开始地址, 低 (位 7~0)
EAH, EAM, EAL: Block 空白检测结束地址 (任一 block 最后地址)
EAH: 结束地址, 高 (位 23 ~ 16)
EAM: 结束地址, 中 (位 15 ~ 8)
EAL: 结束地址, 低 (位 7 ~ 0)

**图 5-21. Block 空白检测命令的状态帧（从 78K0/Lx2 到编程器）**

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

**备注** ST1: Block 空白检测结果

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.10 Block 空白检测命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.11 Block 空白检测命令**

## 5.10 硅标记命令

### 5.10.1 描述

此命令用来读取设备写入协议信息(硅标记) 。

编程器支持的编程协议而 78K0/Lx2 不支持，例如，执行此命令来选择适当的协议来使第 2 和第 3 字节的值一致。

### 5.10.2 命令帧和状态帧

图 5-22 显示了硅标记命令的命令帧格式，图 5-23 显示了命令的状态帧。

**图 5-22. 硅标记命令帧 (从编程器到 78K0/Lx2)**

| SOH | LEN | COM | SUM | ETX |
|---|---|---|---|---|
| 01H | 01H | C0H<br>(硅标记) | 校验和 | 03H |

**图 5-23. 硅标记命令的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1: 命令接收结果

### 5.10.3 硅标记数据帧

图 5-24 显示了包括硅标记数据的帧格式。

**图 5-24. 硅标记数据帧 (从 78K0/Lx2 到编程器)**

| STX | LEN | 数据 | | | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 02H | n | VEN | MET | MSC | DEC | END | INVALID DATA | SCF | BOT | 校验和 | 03H |

**备注 1.** n (LEN): 数据长度

VEN: 厂商码 (NEC: 10H)

MET: 扩展码

MSC: 功能码

DEC: 设备扩展码

END: 内部 Flash 存储器结束地址

INVALID DATA: 10 字节长度的无效数据

SCF: 安全标志信息

BOT: 启动 block 编号(固定为 03H)

**2.** 厂商码(VEN)，扩展码(MET)，功能码(MSC)，设备扩展码(DEC)，内部 Flash 存储器结束地址(END)，设备名称(DEV)和安全标志信息(SCF)，低 7 位常用作数据，最高位常用作奇偶校验。以下举例说明。

**表 5-1. 硅标记数据的示例**

| 区域 | 内容 | 长度 (字节) | 硅标记数据举例[注1] | 实际值 | 奇偶校验 |
|---|---|---|---|---|---|
| VEN | 厂商码(NEC) | 1 | 10H (00010000B) | 10H | 加 |
| MET | 扩展码 (固定 78K0/Lx2) | 1 | 7FH (01111111B) | 7FH | 加 |
| MSC | 功能信息(固定 78K0/Lx2) | 1 | 04H (00000100B) | 04H | 加 |
| DEC | 设备扩展码(固定为 78K0/Lx2) | 1 | 7CH (01111100B) | 07H | 加 |
| END | 内部Flash存储器结束地址 (低字节获取) | 3 | 7FH (01111111B) | 005FFFH | 加[注2] |
| | | | BFH (11011111B) | | |
| | | | 01H (00000001B) | | |
| INVALID DATA | 无效数据 | 10 | – | – | – |
| SCF | 安全标志信息 | 1 | 任 | 任 | 加[注3] |
| BOT | 启动 block 编号(固定) | 1 | 03H (00000011B) | 03H | 不加 |

注 **1.** 0 和1 是奇校验(此值调整字节中 “1”的个数作为奇值)

**2.** 奇偶校验计算END区域按以下执行 (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> 奇校验位加最高位。

p0000001 p01111111 p1111111 (p =奇校验位)

= 0000001 10111111 01111111

= 01 BF 7F

<3> 高，中，低字节次序反转，如下。

7F BF 01

以下显示了 END 区域值从微处理器传输到有效地址的转换顺序。

<1> 高，中，低字节次序反转，如下。

7F BF 01
↓
01 BF 7F

<2> 检查每个字节 “1”的个数为奇数 (可按另一时序执行)。

<3> 去除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> 将值传输到以 8 位为单位构成的组里。

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

若 “7F BF 01” 给 END 区域，那么实际的最终地址是 005FFFH。

**注 3.** 当使用安全设置命令来设置安全标记信息时，最高位固定为“1”。若使用硅标记命令来读取安全标记信息，最高位为奇校验。

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.11 硅标记命令。**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.12 硅标记命令**。

### 5.10.4 78K0/Lx2 硅标记列表

**表 5-2. 78K0/Lx2 硅标记数据列表**

| 项目 | 描述 | 长度 (字节) | 数据 (Hex) |
|---|---|---|---|
| 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) X 3 | 3 | **注** |
| 无效数据 | – | 10 | – |
| 安全标志信息 | 安全标志信息 | 1 | 任一 |
| 启动block编号 | 当前选定的启动区域的最后block编号 | 1 | 03 |

**注** 内部 Flash 存储器结束地址列表

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

## 5.11 版本获取命令

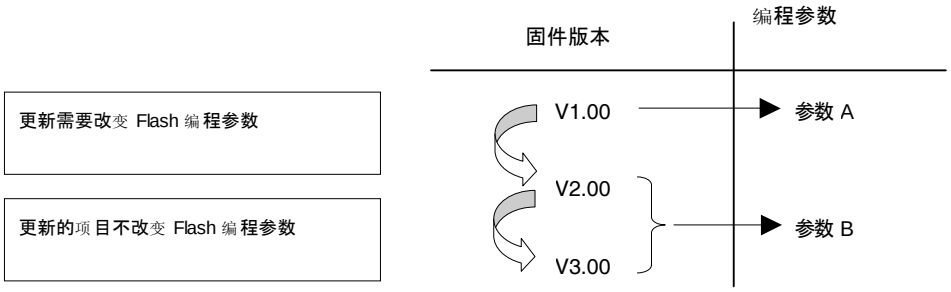
### 5.11.1 描述

此命令用来获取78K0/Lx2设备和固件版本信息。设备版本值固定为00H。

当编程参数必须改变为与 78K0/Lx2 和固件版本一致时，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** 固件版本和改变参数




### 5.11.2 命令帧和状态帧

图 5-26 显示了版本获取命令的命令帧格式，图 5-27 显示了命令的状态帧。

**图 5-26. 版本获取命令帧 (从编程器到 78K0/Lx2)**

| SOH | LEN | COM | SUM | ETX |
|---|---|---|---|---|
| 01H | 01H | C5H<br>(版本获取) | 校验和 | 03H |

**图 5-27. 版本获取命令的状态帧(从 78K0/Lx2 到编程器)**

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

**备注** ST1: 命令接收结果

### 5.11.3 版本数据帧

图 5-28 显示了版本数据的数据帧。

**图 5-28. 版本数据帧（从 78K0/Lx2 到编程器）**

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

**备注** DV1: 设备版本整数 (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

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.12 版本获取命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.13 版本获取命令**

## 5.12 校验和命令

### 5.12.1 描述

此命令用来获取指定区域的校验和数据。

为了校验和计算开始/结束地址，在Flash存储器开头一固定区域地址以block为单位(1 KB) 指定。校验和数据通过指定地址范围从初始值(00H)连续减 1 字节获得。

### 5.12.2 命令帧和状态帧

图 5-29 显示了校验和命令的命令帧格式，图 5-30 显示了命令的状态帧。

**图 5-29. 校验和命令帧（从编程器到 78K0/Lx2）**

| SOH | LEN | COM | 命令信息 | | | | | | SUM | ETX |
|---|---|---|---|---|---|---|---|---|---|---|
| 01H | 07H | B0H<br>(校验和) | SAH | SAM | SAL | EAH | EAM | EAL | 校验和 | 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 5-30. 校验和命令的状态帧（从 78K0/Lx2 到编程器）**

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

**备注** ST1: 命令接收结果

### 5.12.3 校验和数据帧

图 5-31 显示了包括校验和数据的帧格式。

**图 5-31. 校验和数据帧（从 78K0/Lx2 到编程器）**

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

**备注** CK1: 校验和数据的高 8 位
CK2: 校验和数据的低 8 位

阅读以下章节，获取每种通讯模式下编程器和 78K0/Lx2 间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART 通讯模式，阅读 **6.13 校验和命令**
- 3 线串行 I/O 通讯模式(CSI)，阅读 **7.14 校验和命令**

## 5.13 安全设置命令

### 5.13.1 描述

此命令用来执行安全设置 (允许或禁止写入，block 擦除，芯片擦除和启动 block 重写)。执行此命令限制重写 Flash 存储器非授权部分。

**注意事项 安全设置后，可以执行新的设置，从允许到禁止改变设置，但不允许设置从禁止到允许的改变。若执行类似的设置，将产生保护错误(10H)。若必须这样设置，首先要先通过芯片擦除命令将所有的安全标记都初始化 (Block擦除命令不能用来初始化安全标记)。**

**若禁止芯片擦除或启动 block 重写，芯片擦除自写也被禁止，那么编程器不能擦除这些设置。由于编程器规范，推荐芯片擦除禁止前执行重写安全设置。**

### 5.13.2 命令帧和状态帧

图 5-32 显示了安全设置命令的命令帧格式，图 5-33 显示了命令的状态帧。

**图 5-32. 安全设置命令帧(从编程器到 78K0/Lx2)**

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

**图 5-33. 安全设置命令的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1 (a): 命令接收结果

### 5.13.3 数据帧和状态帧

图 5-34 显示了安全数据帧的格式，图 5-35 显示了数据的状态帧。

**图 5-34. 安全数据帧 (从编程器到 78K0/Lx2)**

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

**备注** FLG: 安全标志
BOT: 启动区域中最后 block 编号(固定为 03H)

**图 5-35. 安全数据写入的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1 (b): 安全数据写入结果

### 5.13.4 内部校验检查和状态帧

图 5-36 显示了内部校验检查的状态帧。

**图 5-36. 内部校验检查的状态帧 (从 78K0/Lx2 到编程器)**

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

**备注** ST1 (c):内部校验结果

以下表格显示了安全标志域的内容。

**表 5-3. 安全标志域内容**

| 项目 | 内容 |
|---|---|
| 第 7位 | 固定为 “1” |
| 第 6位 | |
| 第 5位 | |
| 第 4位 | 启动block重写禁止标志 (1: 允许启动block重写，0: 禁止启动block重写) |
| 第 3位 | 固定为“1” |
| 第 2位 | 编程禁止标志(1:允许编程，0: 禁止编程) |
| 第 1位 | Block擦除禁止标志 (1: 允许block擦除，0: 禁止block擦除) |
| 第 0位 | 芯片擦除禁止标志 (1: 允许芯片擦除，0: 禁止芯片擦除) |

以下表格显示了每种操作下安全标记设置和允许/禁止状态的关系。

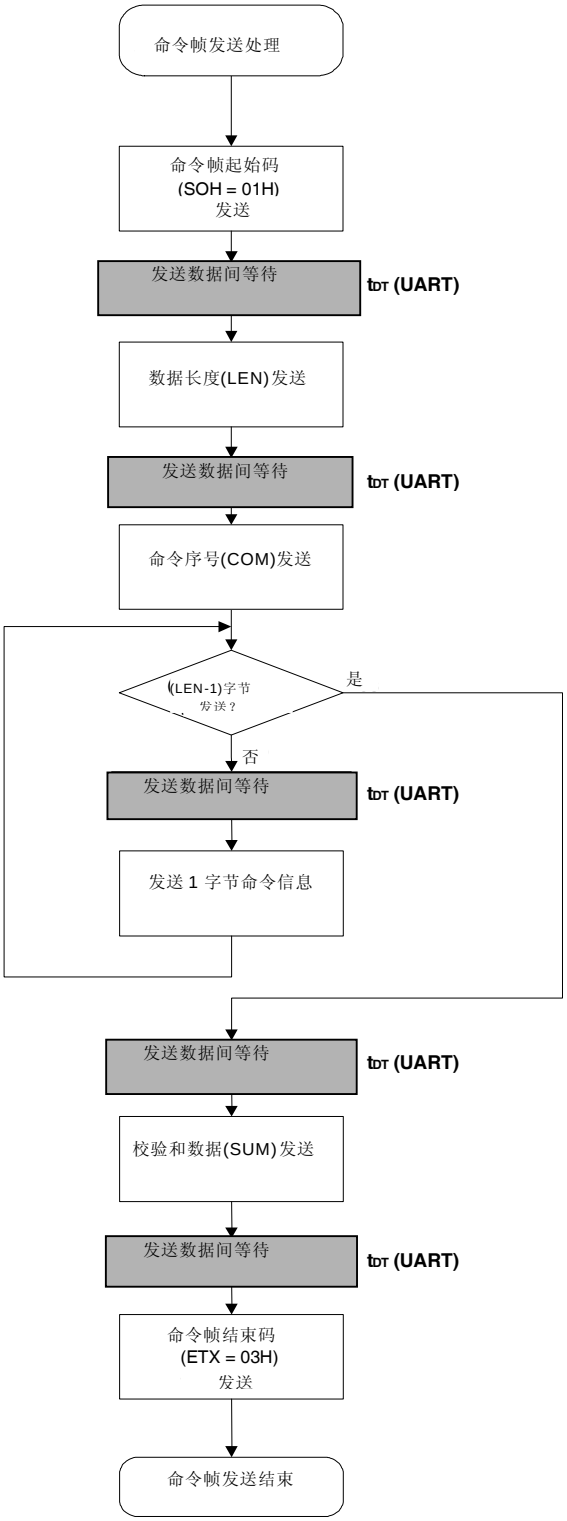
**表 5-4. 每种操作的安全标志域和允许/禁止状态**

<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>命令<br>安全设置项目</td><td colspan="3">安全设置后的命令操作<br>√:允许执行，X:禁止执行<br>△:禁止导入区域写入和block擦除</td><td>• 不管安全设置值所有命令都可以执行<br>• 仅保持安全设置值</td></tr>
<tr><td></td><td>编程</td><td>芯片擦除</td><td>Block 擦除</td><td></td></tr>
<tr><td>禁止编程</td><td>X</td><td>√</td><td>禁止编程</td><td rowspan="3">编程<br>X<br>√<br>√</td></tr>
<tr><td>禁止芯片擦除</td><td>√</td><td>X</td><td>禁止芯片擦除</td></tr>
<tr><td>禁止block擦除</td><td>√</td><td>√</td><td>禁止block擦除</td></tr>
<tr><td>启动block重写禁止标志</td><td>△</td><td>X</td><td>启动block重写禁止标志</td><td>△</td></tr>
</table>

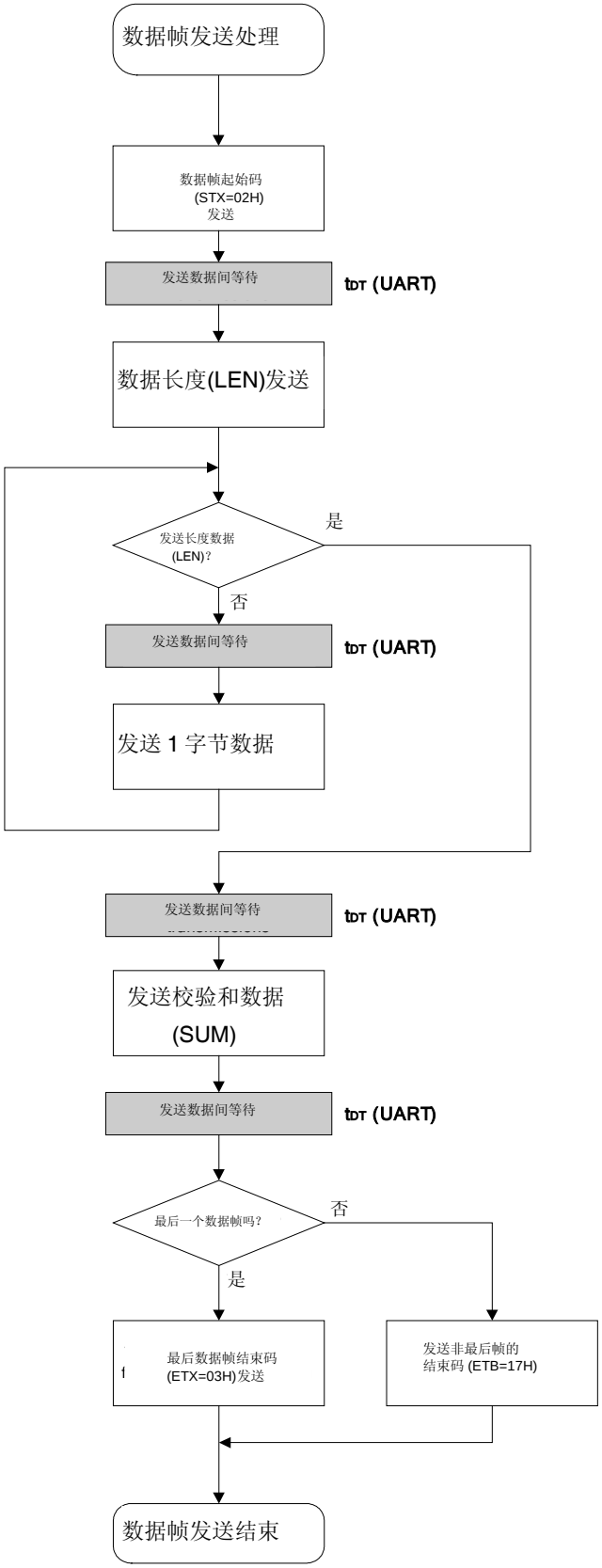
阅读以下章节，获取每种通讯模式下编程器和78K0/Lx2间的程序举例，处理次序流程图和命令处理流程图的详细资料。

- UART通讯模式，阅读**6.14 安全设置命令**
- 3线串行I/O通讯模式(CSI)，阅读**7.15 安全设置命令**

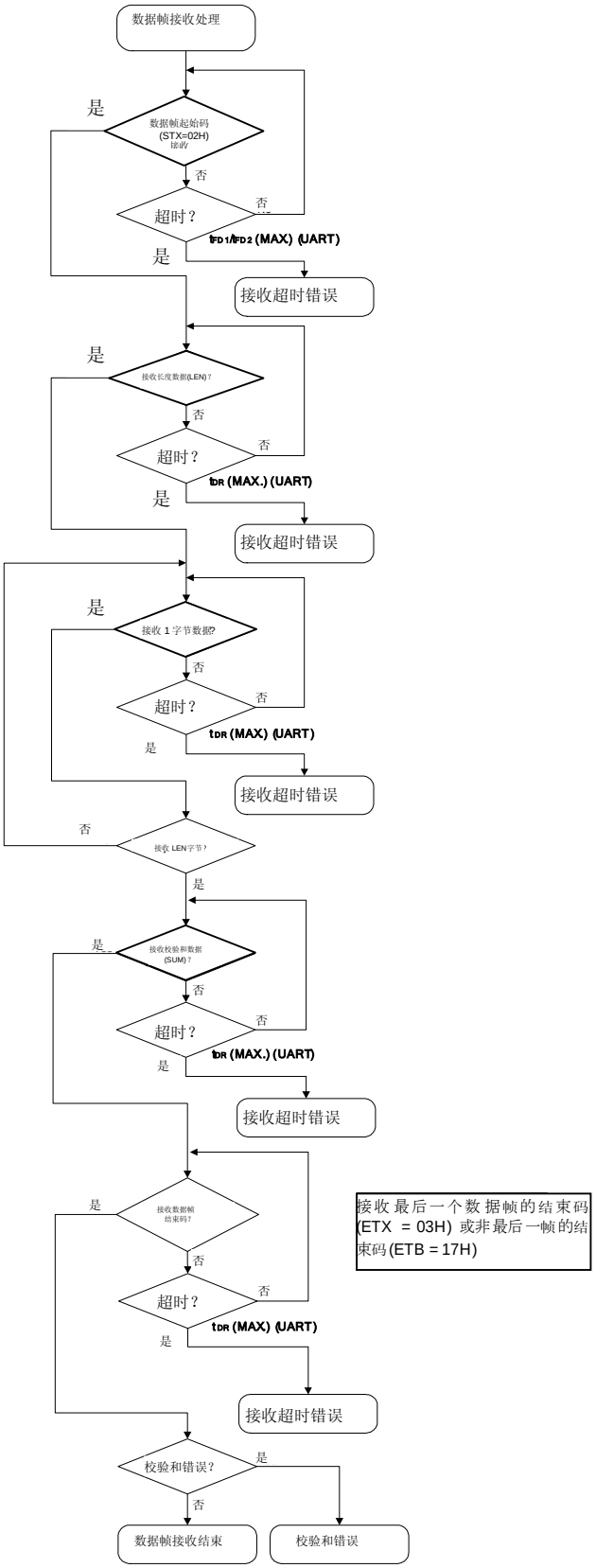
## 6.1 命令帧发送处理流程图



## 6.2 数据帧发送处理流程图



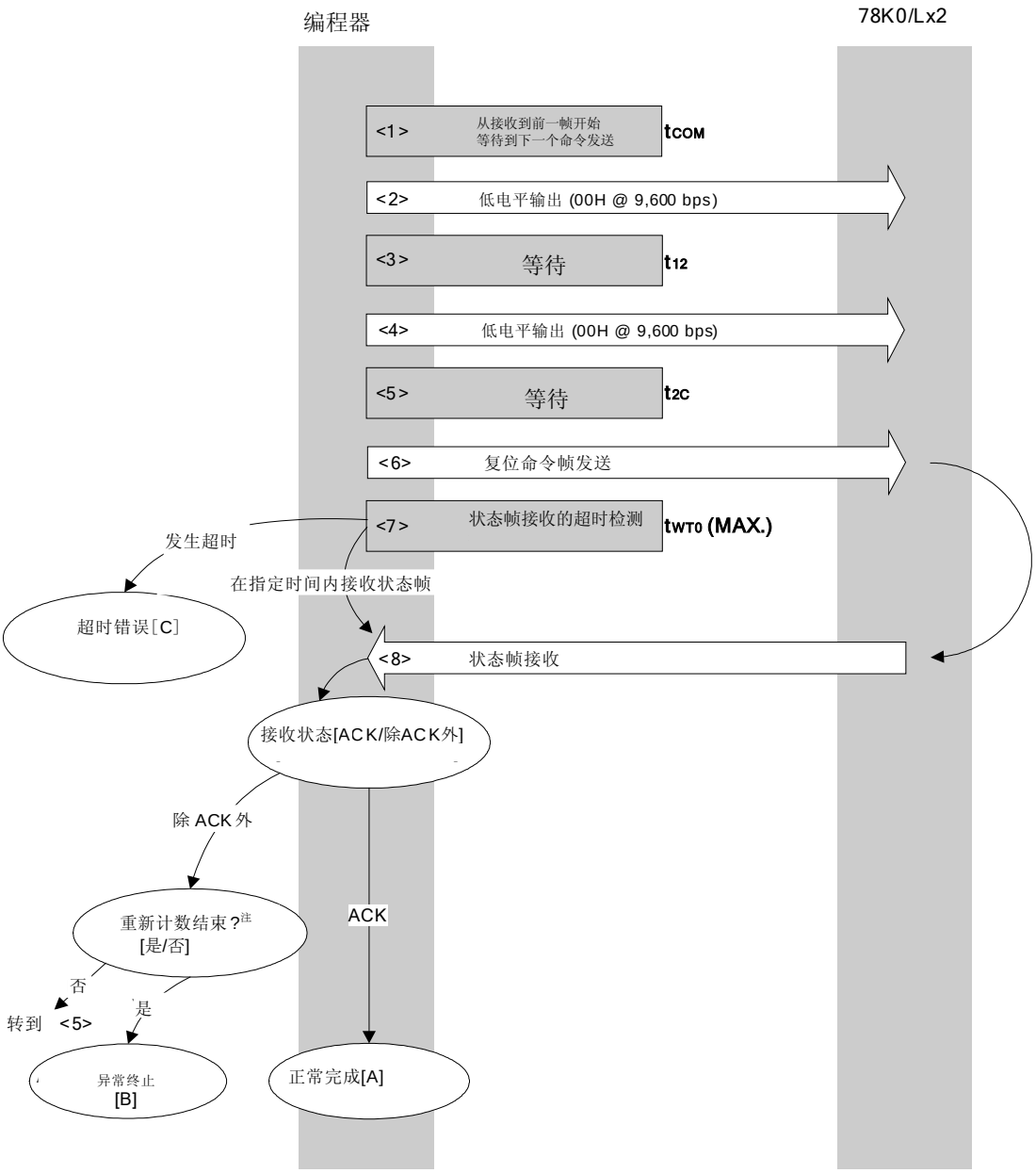
## 6.3 数据帧接收处理流程图



## 6.4 复位命令

### 6.4.1 处理流程图

复位命令处理流程




**注** 复位命令发送不要重复计数(至多16 次)。

### 6.4.2 处理序列描述

<1> 从接收到前一帧开始等待到下一个命令处理开始(等待时间 $t_{COM}$)。

<2> 输出低电平 (数据 00H以9,600 bps发送)。

<3> 等待状态 (等待时间 $t_{12}$)。

<4> 输出低电平(数据00H以 9,600 bps发送)。

<5> 等待状态 (等待时间 $t_{2C}$).

<6> 复位命令由命令帧发送程序发送。

<7> 从命令发送执行超时检测，直到状态帧接收。

如果发生超时，返回超时错误 [C] (超时时间$t_{WT0}$(MAX.))。

<8> 检测状态码。

当 ST1 = ACK: 正常完成 [A]

当 ST1 ¹ ACK: 检查重复计数($t_{RS}$) 。

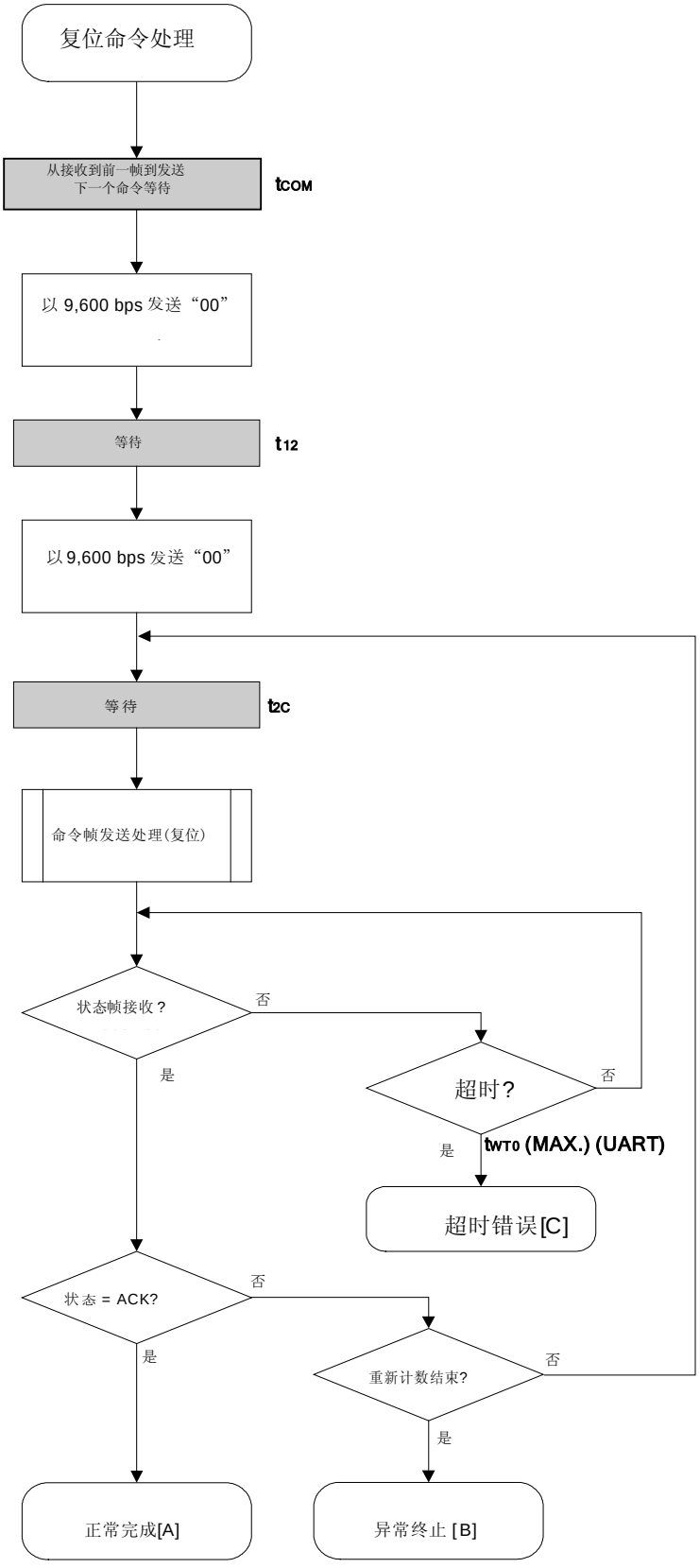
如果重复计数没有结束，从步骤<5> 开始重复执行。

如果重复计数结束，处理异常终止 [B]。

### 6.4.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常响应 (ACK) | 06H | 命令正常执行并且编程器与78K0/Lx2 同步。 |
| 异常终止 [B] | 校验和错误 | 07H | 发送的命令帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常(如无效数据长度(LEN) 或 无 ETX)。 |
| 超时错误 [C] | | ~ | 在指定时间内没有接收到状态帧。 |

### 6.4.4 流程图


复位命令处理
从接收到前一帧到发送下一个命令等待
tCOM
以 9,600 bps 发送“00”
等待
t12
以 9,600 bps 发送“00”
等待
t2C
命令帧发送处理(复位)
状态帧接收？
否
是
超时？
否
是
tWT0 (MAX.) (UART)
超时错误[C]
状态 = ACK?
否
是
重新计数结束?
否
是
正常完成[A]
异常终止 [B]

### 6.4.5 例子程序

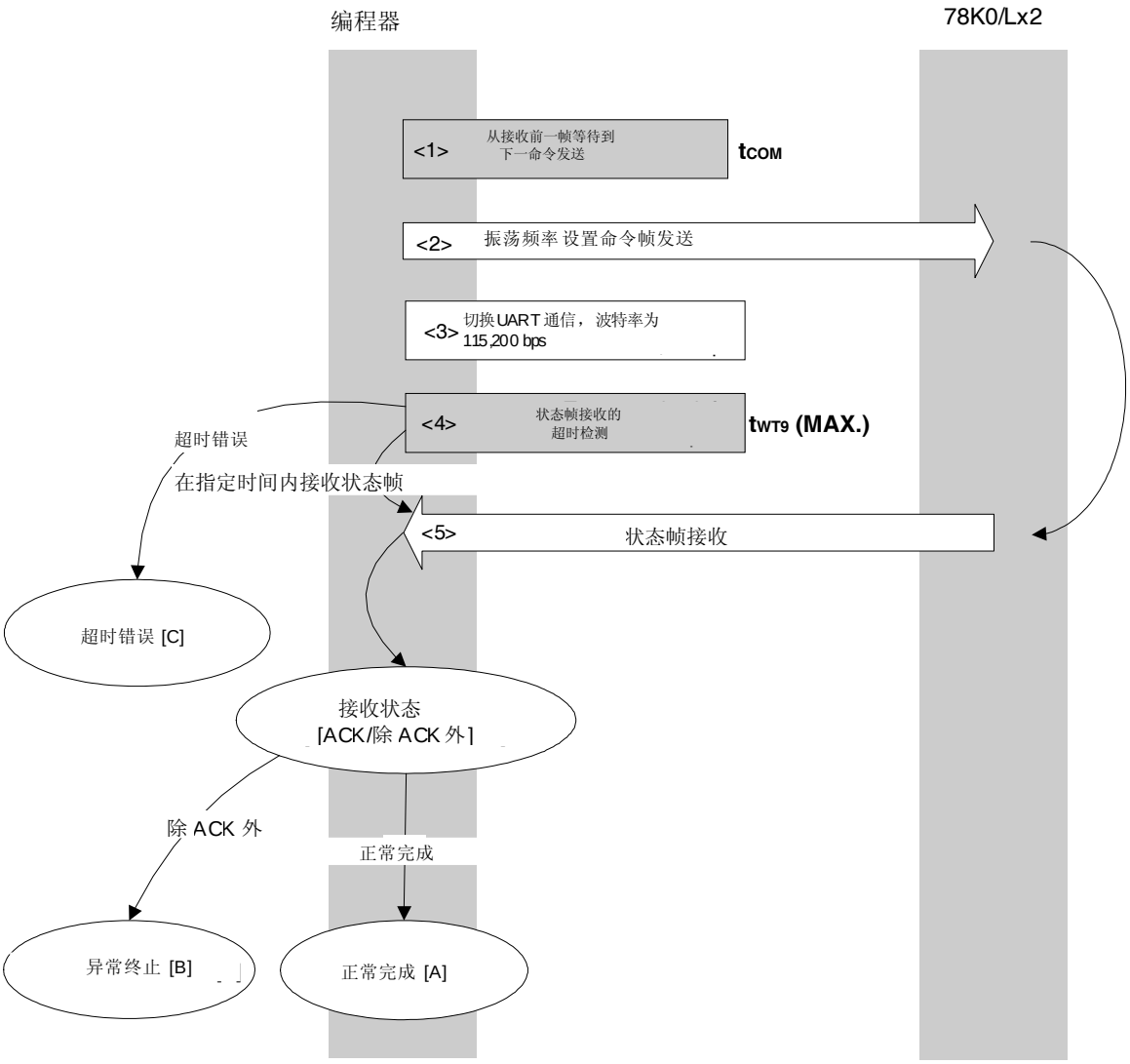
如下所示为复位命令处理的一个例子程序。

```
/*****************************************************************/
/*                                                               */
/*复位命令                                                       */
/*                                                               */
/*****************************************************************/
/* [r] u16      ... 错误代码                                     */
/*****************************************************************/
u16             fl_ua_reset(void)
{
        u16     rc;
        u32     retry;
        set_uart0_br(BR_9600);  //改为 9600bps

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

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

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

                fl_wait(t2C);       //等待

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

                rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
                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] (如果从循环中跳出)
        }
//      switch(rc) {
//
//              case    FLC_NO_ERR:   return    rc;       break;   // case [A]
//              case    FLC_DFTO_ERR:         return    rc;       break;   // case [C]
//              default:            return    rc;       break;   // case [B]
//      }
        return  rc;
}
```

## 6.5 振荡频率设置命令

### 6.5.1 处理流程图

振荡频率设置命令处理流程图



### 6.5.2 处理流程描述

<1> 从接收到前一帧开始到下一个命令发送等待 (等待时间$t_{COM}$).

<2> 振荡频率设置命令由命令帧发送处理发送。

<3> 接收到状态帧后，UART通讯速率变为115,200 bps。之后，通讯速率恒为115,200 bps。

<4> 从命令发送执行超时检测，直到状态帧接收。

如果发生超时，返回超时错误[C] (超时时间 $t_{WT9}$(MAX.))。

<5> 检查状态码。

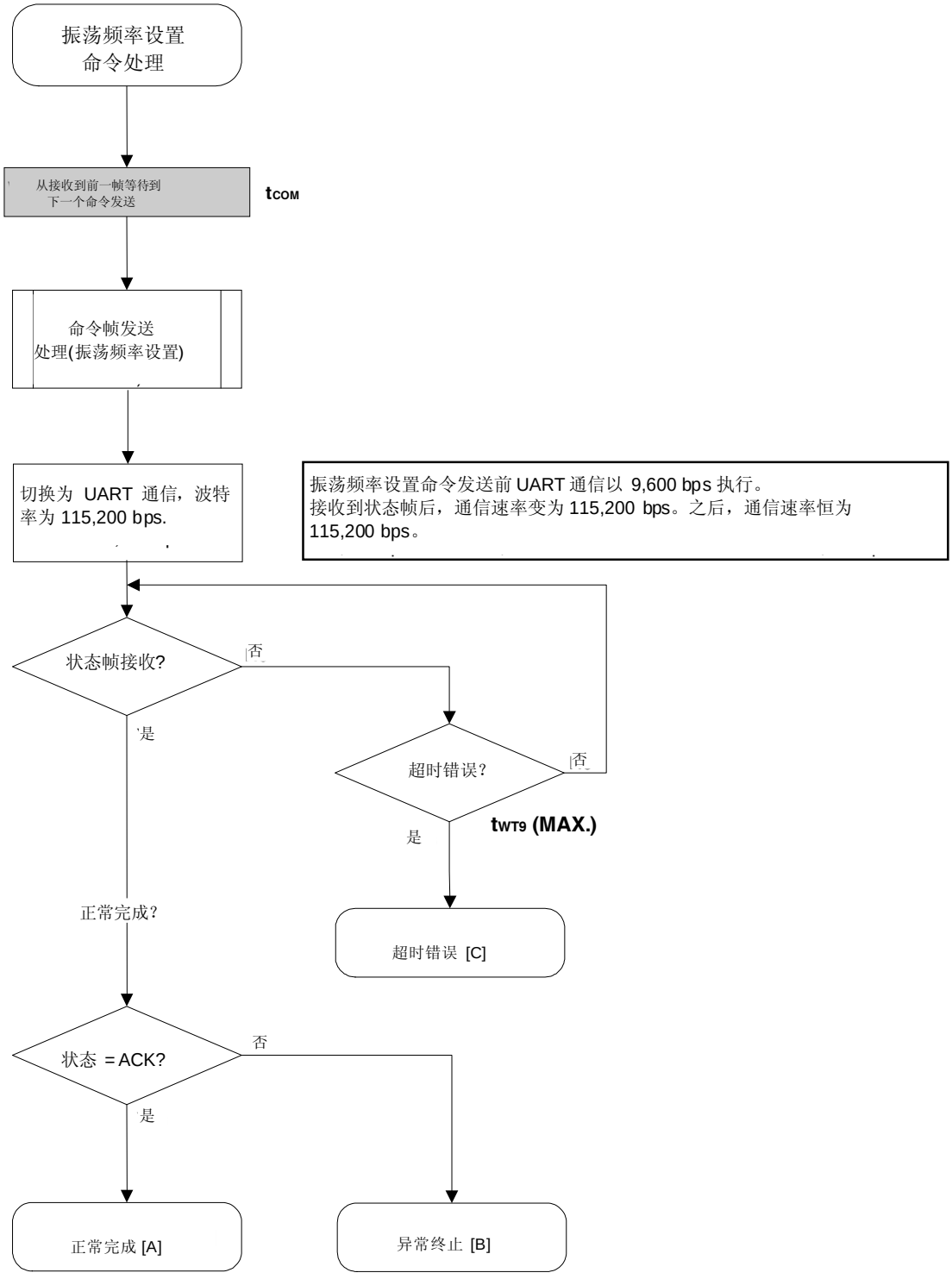
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [B]

### 6.5.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常响应 (ACK) | 06H | 命令正常执行， 78K0/Lx2 设置的频率正确。 |
| 异常终止 [B] | 参数错误 | 05H | 振荡频率值超出范围。 |
| | 校验和错误 | 07H | 发送的命令帧校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN) 或无 ETX)。 |
| 超时错误 [C] | | - | 状态帧没有在指定时间内接收到。 |

### 6.5.4 流程图


振荡频率设置
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(振荡频率设置)
切换为 UART 通信，波特率为 115,200 bps.
振荡频率设置命令发送前 UART 通信以 9,600 bps 执行。
接收到状态帧后，通信速率变为 115,200 bps。之后，通信速率恒为 115,200 bps。
状态帧接收?
否
是
超时错误?
否
tWT9 (MAX.)
是
正常完成?
超时错误 [C]
状态 = ACK?
否
是
正常完成 [A]
异常终止 [B]

### 6.5.5 例子程序

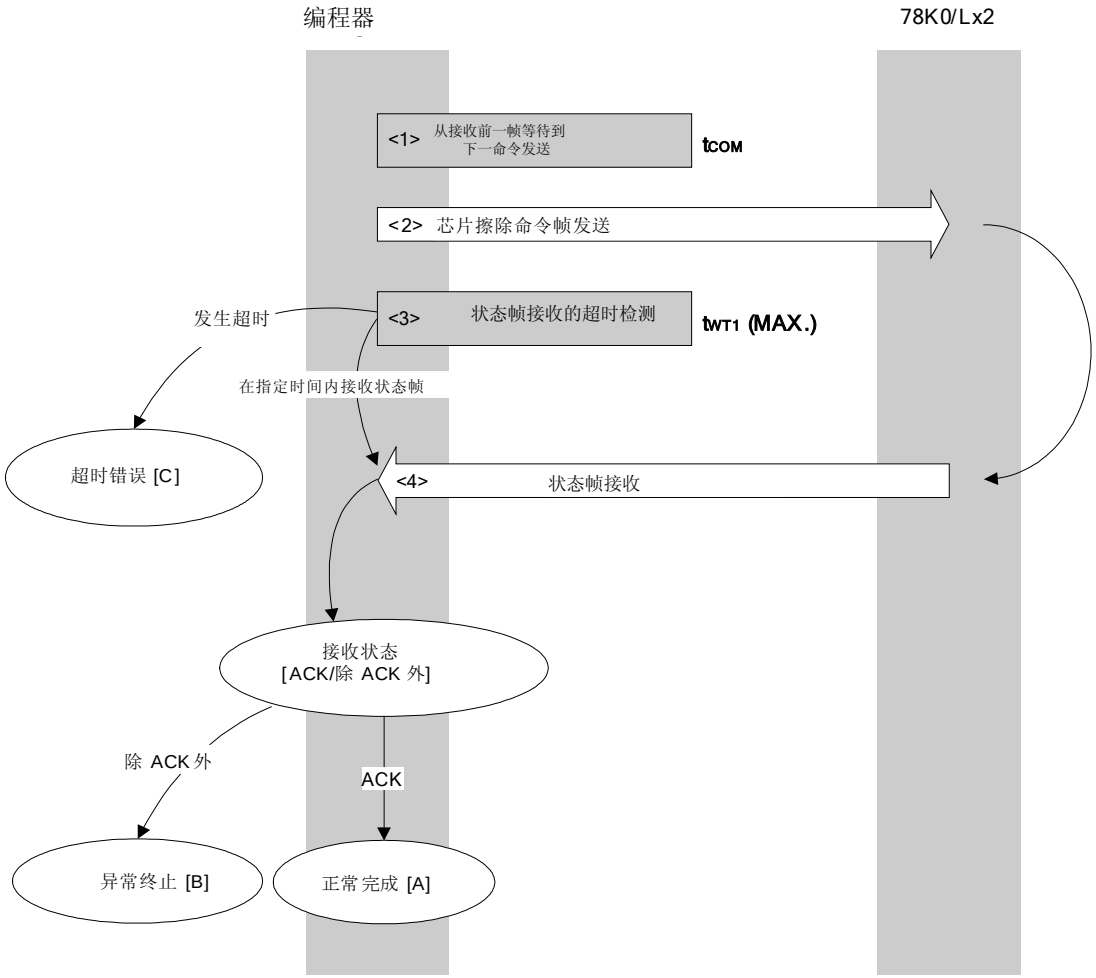
如下所示为振荡频率设置命令的一个例子程序。

```
/*********************************************************************/
/*                                                                   */
/*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);          //改变波特率
        set_uart0_br(BR_115200);        //改变波特率(h.w.)


        rc = get_sfrm_ua(fl_ua_sfrm, tWT9_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;
}
```

## 6.6 芯片擦除命令

### 6.6.1 处理流程图

芯片擦除命令处理流程图



### 6.6.2 处理流程描述

<1> 从接收到前一帧开始到下一个命令发送等待(等待时间 $t_{COM}$)。

<2> 芯片擦除命令由命令帧发送处理发送。

<3> 从命令发送执行超时检测，直到状态帧接收。

如果发生超时，返回超时错误[C] (超时时间 $t_{WT1}$(MAX.))。

<4> 检查状态码。

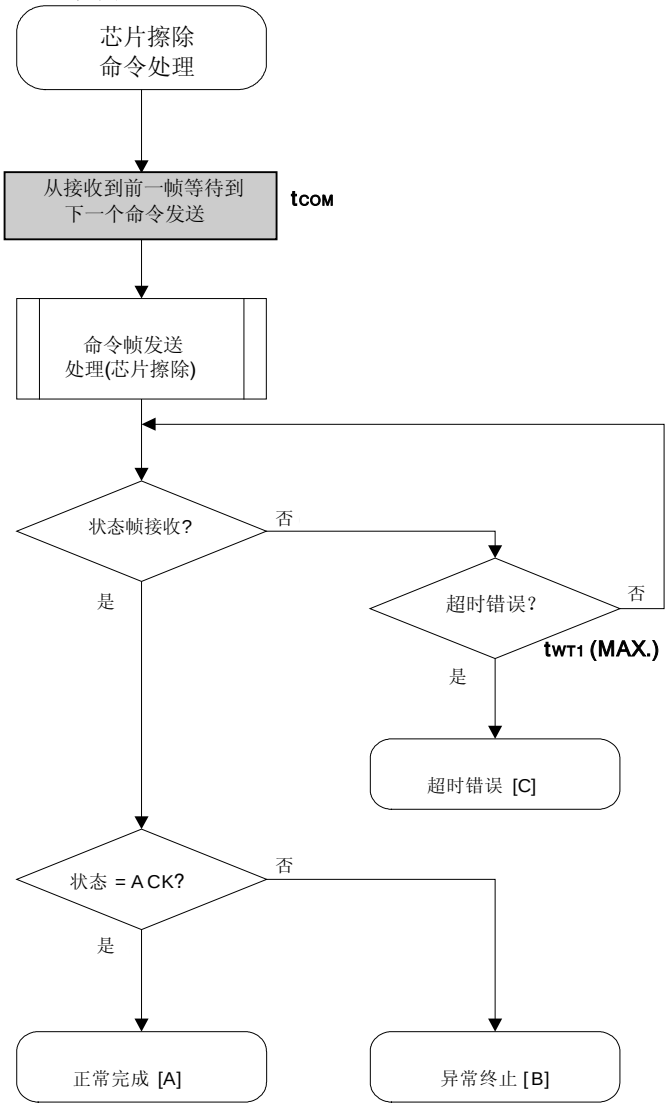
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [B]

### 6.6.3 命令完成时的状态

| 处理完成时状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常响应 (ACK) | 06H | 命令正常执行，芯片擦除正常完成。 |
| 异常终止 [B] | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 保护错误 | 10H | 安全设置禁止芯片擦除。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常(如无效数据长度(LEN) 或无 ETX)。 |
| | 擦除错误 | 1AH | 发生芯片擦除错误。 |
| 超时错误 [C] | | - | 没有在指定时间内接收到状态帧。 |

### 6.6.4 流程图


芯片擦除
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(芯片擦除)
状态帧接收?
否
是
超时错误?
否
tWT1 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
正常完成 [A]
异常终止 [B]

### 6.6.5 例子程序

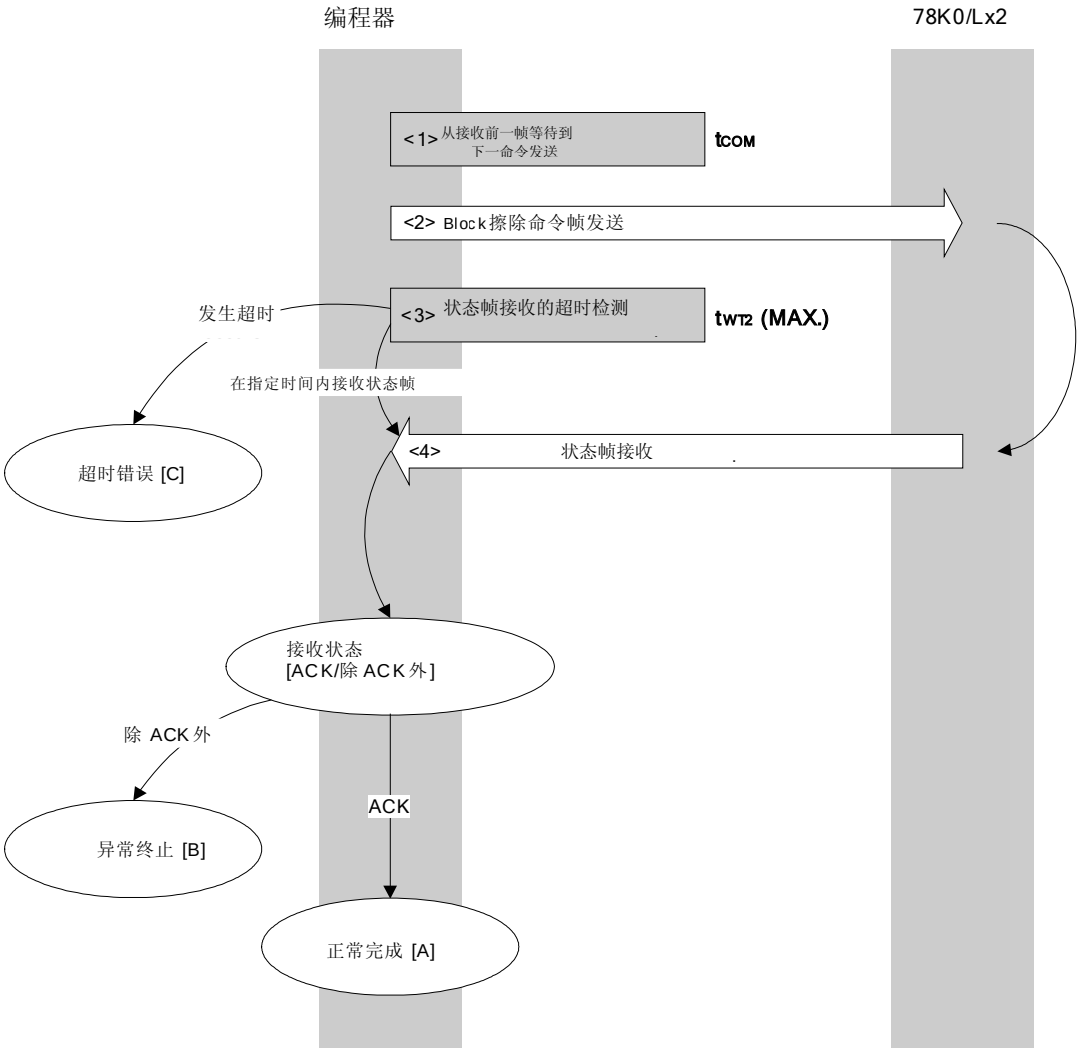
如下所示为芯片擦除命令的一个例子程序。

```
/*********************************************************************/
/*                                                                   */
/*全部(芯片)擦除命令                                                 */
/*                                                                   */
/*********************************************************************/
/* [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); //发送芯片擦除命令

        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;

}
```

## 6.7 Block 擦除命令

### 6.7.1 处理流程图

Block 擦除命令处理流程图



### 6.7.2 流程描述

<1> 从接收到前一帧开始到下一个命令发送等待(等待时间$t_{COM}$)。

<2> Block擦除命令由命令帧发送处理发送。

<3> 从命令发送执行超时检测，直到状态帧接收。

如果发生超时，返回超时错误[C] (超时时间$t_{WT2}$(MAX.))。

<4> 检查状态码。

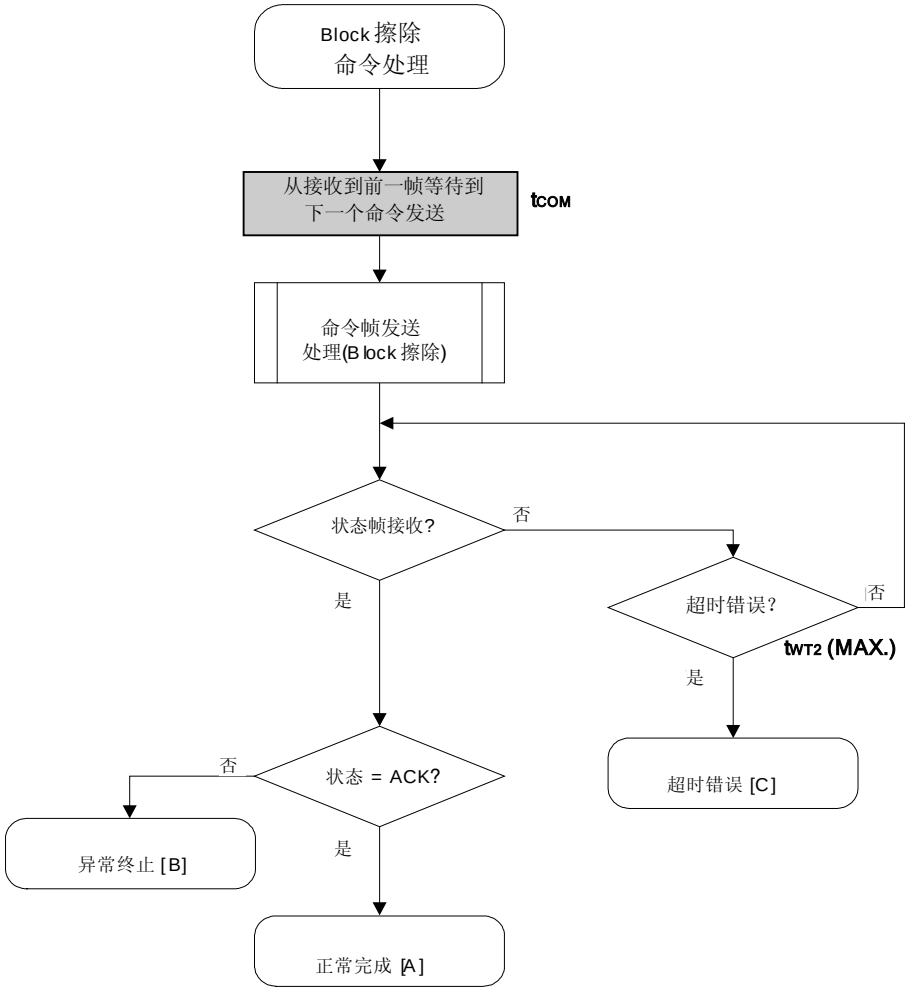
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [B]

### 6.7.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常响应(ACK) | 06H | 命令正常执行，block擦除正常完成。 |
| 异常终止 [B] | 参数错误 | 05H | block的序号超出范围。 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 保护错误 | 10H | 在安全设置中禁止写、block擦除、或芯片擦除操作。 |
| | 异常应答(NACK) | 15H | 命令帧数据异常(如无效数据长度(LEN) 或无ETX)。 |
| | 擦除错误 | 1AH | 发生擦除错误。 |
| 超时错误 [C] | | - | 状态帧没有在指定时间内接收到。 |

### 6.7.4 流程图


Block 擦除
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(Block 擦除)
状态帧接收?
否
是
超时错误?
否
tWT2 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
正常完成 [A]

### 6.7.5 例子程序

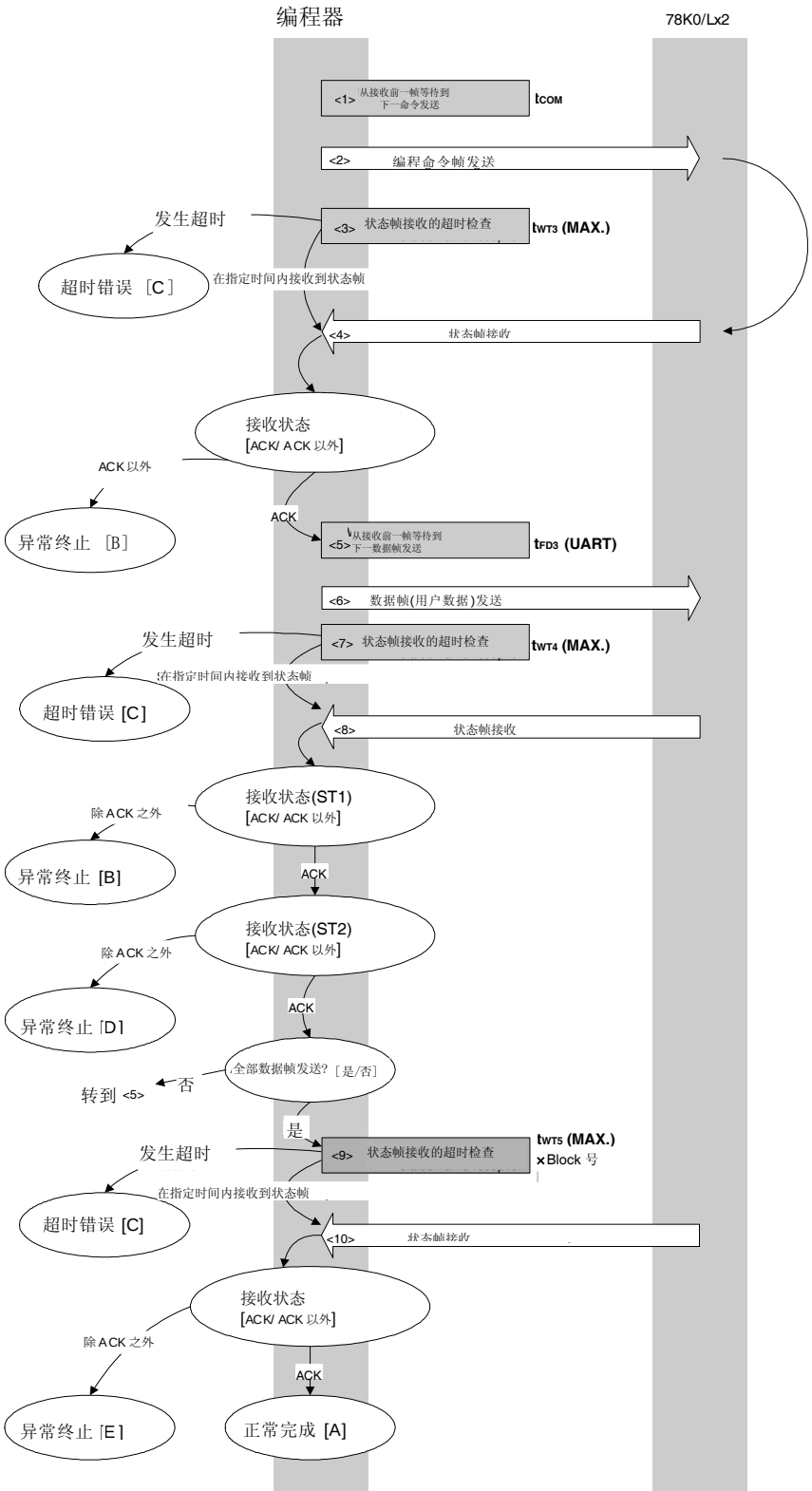
如下所示为擦除一个block 的block 擦除命令处理的例子程序。

```
/*****************************************************************/
/*                                                               */
/*擦除 block 命令                                                */
/*                                                               */
/*****************************************************************/
/* [i] u16 sblk   ... 待擦除的开始 block (0...255)          */
/* [i] u16 eblk   ... 待擦除的结束 block   (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 block 的起始地址
        bottom = get_bottom_addr(eblk);     //获取最后一个 block 的最后一个地址

        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); //发送芯片擦除命令

        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;

}
```

## 6.8 编程命令

### 6.8.1 处理流程图

编程命令处理流程图



### 6.8.2 流程描述

<1> 从接收到前一帧开始到下一个命令发送等待(等待时间 $t_{COM}$)。

<2> 编程命令由命令帧发送处理发送。

<3> 从命令发送执行超时检测，直到状态帧接收。

如果发生超时，返回超时错误[C] (超时时间 $t_{WT3}$(MAX.))。

<4> 检查状态码。

当 ST1 = ACK: 进行 <5>

当 ST1 ≠ ACK: 异常终止 [B]

<5> 从接收到前一帧开始到下一个数据帧发送等待(等待时间$t_{FD3}$(UART))。

<6> 用户数据由数据帧发送处理发送。

<7> 从用户数据发送执行超时检测，直到数据帧接收。

如果发生超时，返回超时错误[C] (超时时间$t_{WT4}$(MAX.))。

<8> 检查状态码 (ST1/ST2) (参考流程图)。

当 ST1 ≠ ACK: 异常终止 [B]

当 ST1 = ACK: 如下处理根据ST2的值执行。

· 当 ST2 = ACK: 当所有数据帧发送完毕运行<9>。

如果仍然有保留数据帧需要发送，处理重新从步骤<5>执行。

• 当 ST2 ≠ ACK: 异常终止 [D]

<9> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间 $t_{WT5}$(MAX.)Xblock数)。

<10> 检查状态码。

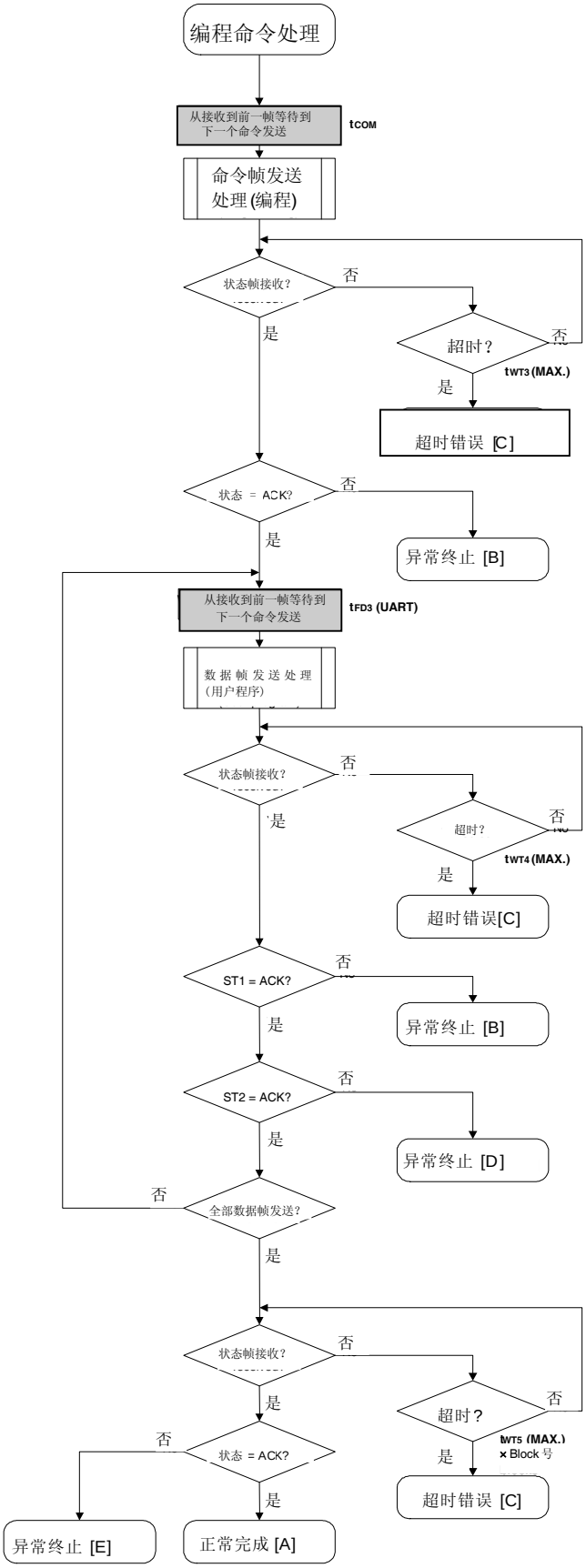
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [E]

### 6.8.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 执行命令正常并且用户数据正常写入。 |
| 异常终止 [B] | 参数错误 | 05H | 特定开始/结束地址在flash存储器范围之外，或者不是8的倍数。 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 保护错误 | 10H | 安全设置禁止写入。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN) 或无 ETX)。 |
| 超时错误 [C] | | ~ | 在指定时间内没有接收到状态帧。 |
| 异常终止 [D] | 写错误 | 1CH (ST2) | 产生写入错误。 |
| 异常终止 [E] | MRG11 错误 | 1BH | 产生内部校验错误。 |

### 6.8.4 流程图


编程命令处理
从接收到前一帧等待到下一个命令发送
tCOM
命令帧发送处理(编程)
状态帧接收?
否
是
超时?
否
tWT3 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
从接收到前一帧等待到下一个命令发送
tFD3 (UART)
数据帧发送处理(用户程序)
状态帧接收?
否
是
超时?
否
tWT4 (MAX.)
是
超时错误[C]
ST1 = ACK?
否
是
异常终止 [B]
ST2 = ACK?
否
是
异常终止 [D]
全部数据帧发送?
否
是
状态帧接收?
否
是
超时?
否
tWT5 (MAX.) × Block 号
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [E]
正常完成 [A]

### 6.8.5 例子程序

如下所示为编程命令处理的一个例子程序。

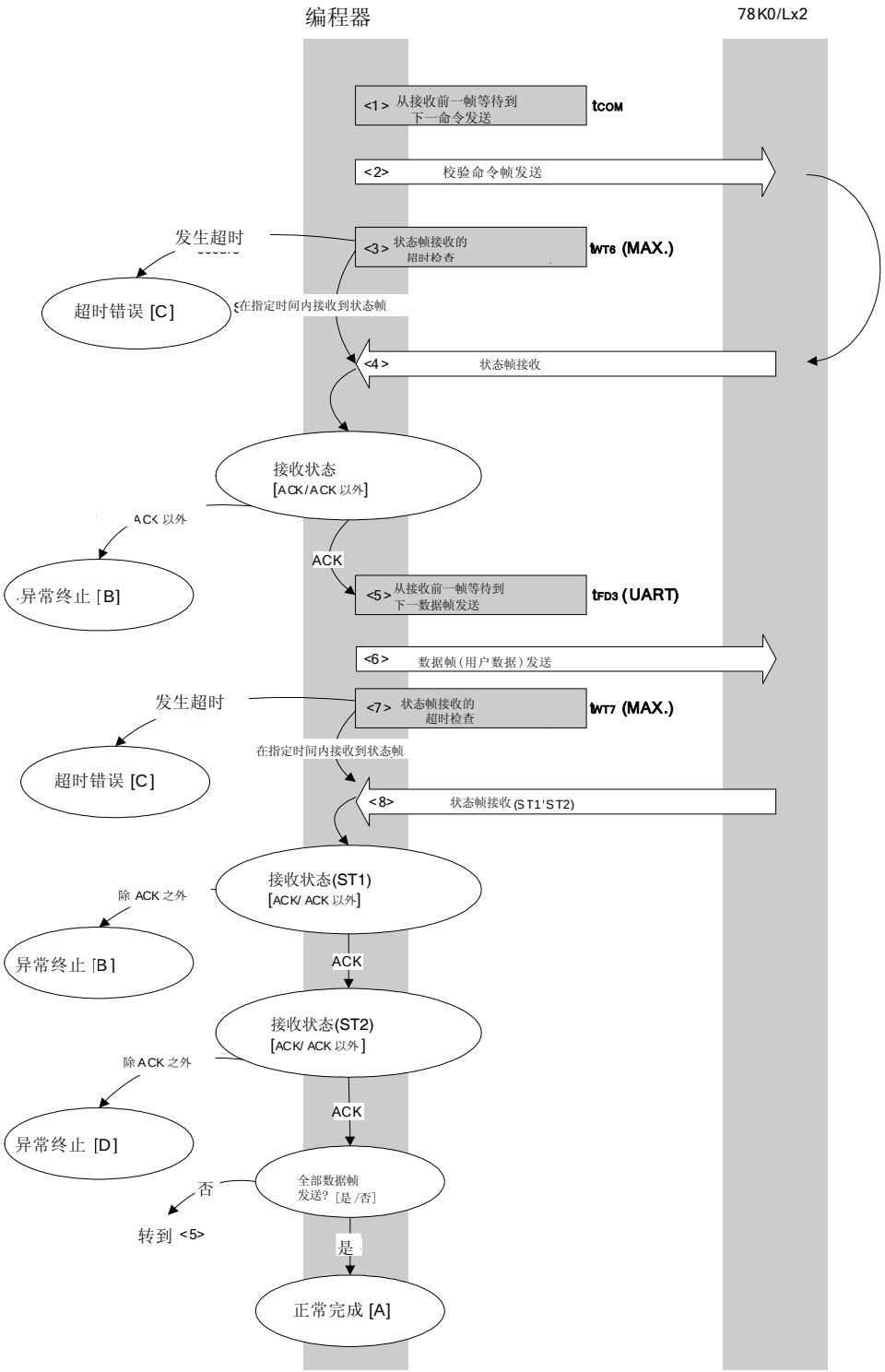
```
/*****************************************************************/
/*                                                               */
/*写命令                                                         */
/*                                                               */
/*****************************************************************/
/* [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;

        /*************************************************/
        /*     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 block num

        /*************************************************/
        /*     发送命令 & 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 "Programming" 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT3_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]
        }

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

```
while(1){
        // make send data frame
                if ((bottom - send_head) > 256){                // rest 大小 > 256 ?
                        is_end = false;                         //是, 不是 is_end 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);    //建立数据帧
                                                                                //有效载荷
                send_head += send_size;

                fl_wait(tFD3_UA);                       //发送数据帧前等待

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

                rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX);             //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]
                }
                if (fl_st2_ua != FLST_ACK){                     // ST2 = ACK ?
                        rc = decode_status(fl_st2_ua);      // 否
                        return  rc;                                 // case [D]
                }
                if (is_end)
                        break;

        }
        /*************************************************/
        /*      检查内部校验            */
        /*************************************************/
        rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX * block_num));       //再次取状态帧
//      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;
}
```

## 6.9 校验命令

### 6.9.1 处理流程

校验命令处理流程图



### 6.9.2 流程描述

<1> 从接收到前一帧到发送下一命令帧等待(等待时间$t_{COM}$)。

<2> 校验命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT6}$(MAX.))。

<4> 检查状态码。

当 ST1 = ACK: 执行 <5>.

当 ST1 ≠ ACK: 异常终止 [B]

<5> 从接收到前一帧到发送下一数据帧等待(等待时间 $t_{FD3}$(UART))。

<6> 校验的用户数据由数据帧发送处理发送。

<7> 从用户数据发送执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT7}$(MAX.))。

<8> 检查状态码(ST1/ST2) (也可参考过程流程图)。

当 ST1 ≠ ACK: 异常终止 [B]

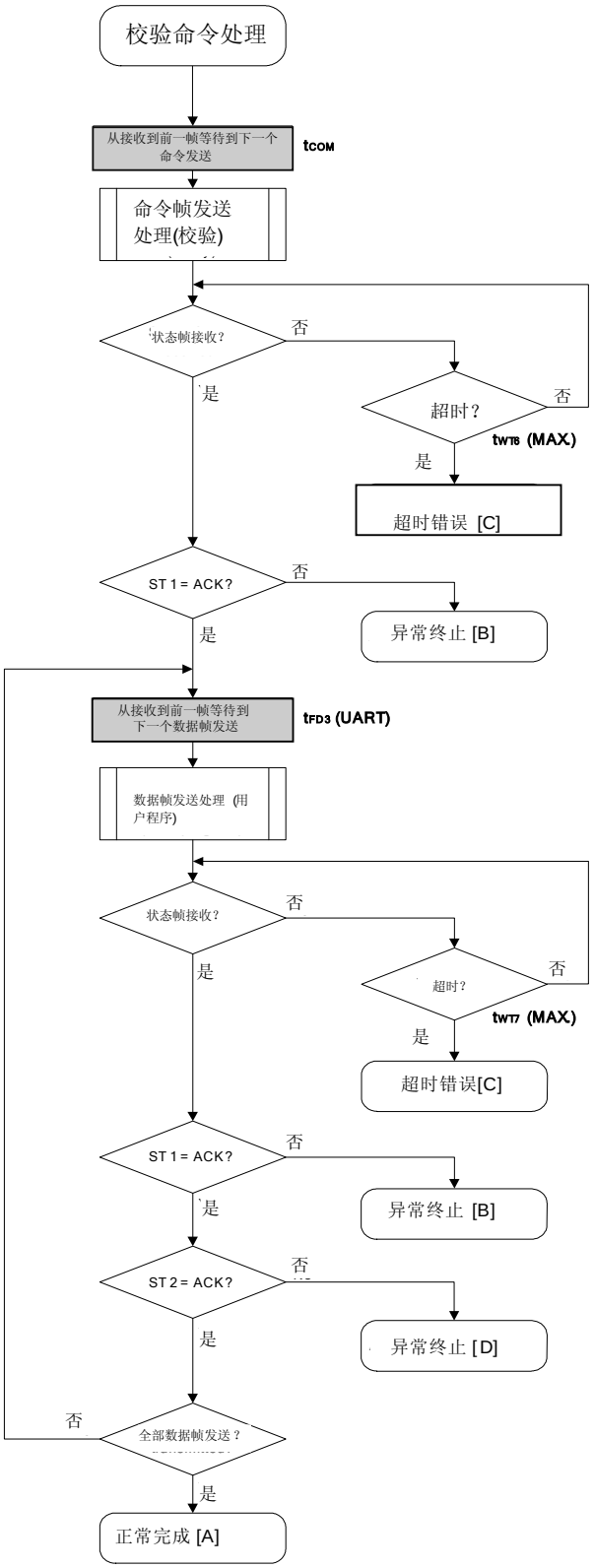
当 ST1 = ACK: 下面的处理根据ST2 的值执行。

- 当 ST2 = ACK: 如果所有的数据帧发送完毕，处理正常结束[A]。
  如果仍有保留的数据帧需要发送，处理从步骤<5>重新开始执行。
- 当 ST2 ≠ ACK: 异常终止 [D]

### 6.9.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 命令正常执行，并且校验正常完成。 |
| 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 | 发送命令帧或数据帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常(如无效数据长度(LEN)或无 ETX)。 |
| 超时错误 [C] | | - | 没有在指定时间内接收到状态帧。 |
| 异常终止 [D] | 校验错误 | 0FH (ST2) | 校验失败，或发生其他错误。 |

### 6.9.4 流程图


校验命令处理
从接收到前一帧等待到下一个命令发送
tCOM
命令帧发送处理(校验)
状态帧接收?
否
是
超时?
否
tWT8 (MAX.)
是
超时错误 [C]
ST 1 = ACK?
否
是
异常终止 [B]
从接收到前一帧等待到下一个数据帧发送
tFD3 (UART)
数据帧发送处理 (用户程序)
状态帧接收?
否
是
超时?
否
tWT7 (MAX.)
是
超时错误[C]
ST 1 = ACK?
否
是
异常终止 [B]
ST 2 = ACK?
否
是
异常终止 [D]
否
全部数据帧发送?
是
正常完成 [A]

### 6.9.5 例子程序

如下所示为校验命令处理的一个例子程序。

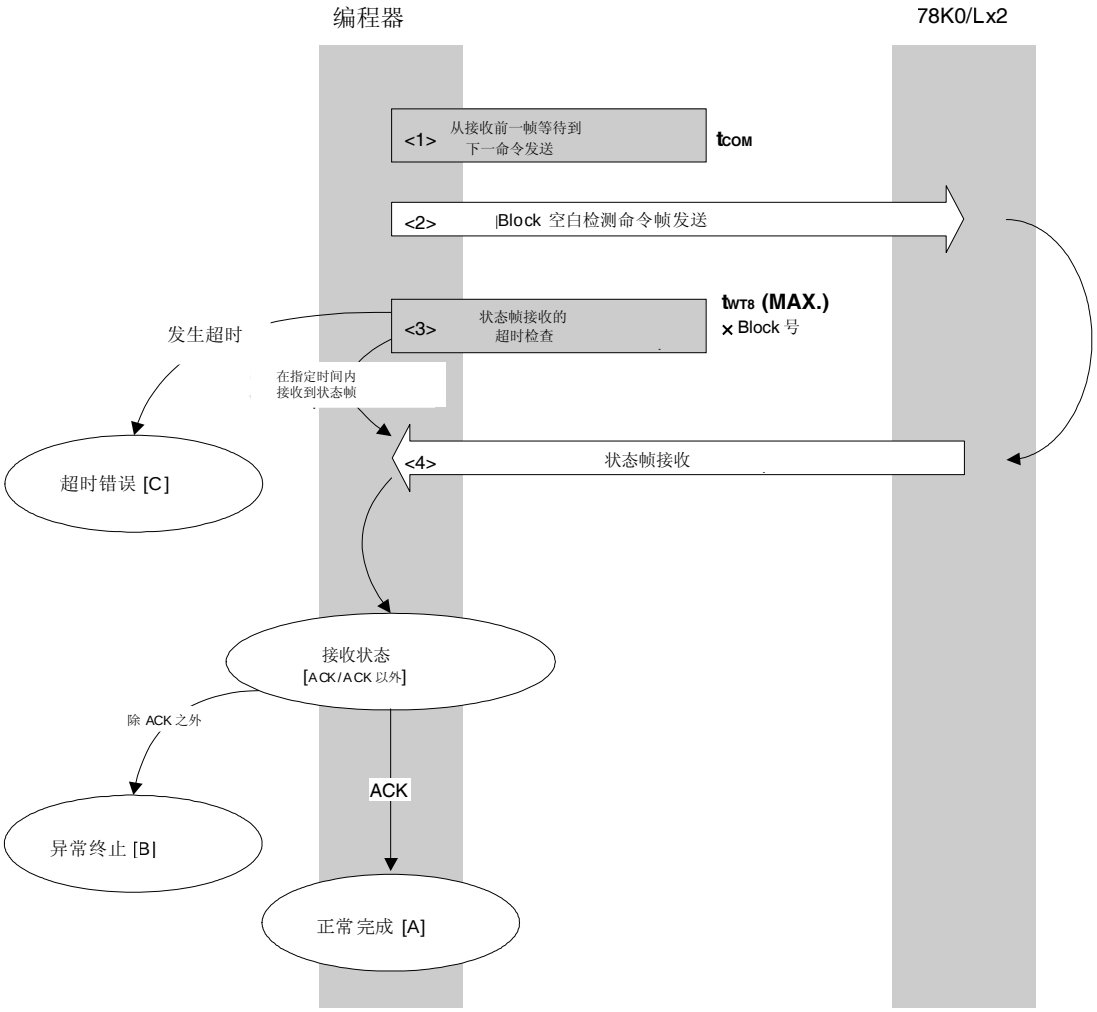
```
/*****************************************************************/
/*                                                               */
/*校验命令                                                       */
/*                                                               */
/*****************************************************************/
/* [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 & 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 VERIFY 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT6_MAX);                 //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){                // rest 大小 > 256 ?
```

```
                is_end = false;                 //是, 不是 is_end 帧
                send_size  = 256;               //发送大小= 256 byte
        }
        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
                                                                //有效载荷
        send_head += send_size;

        fl_wait(tFD3_UA);
        put_dfrm_ua(send_size, fl_txdata_frm, is_end); //发送用户数据

        rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX);         //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]
        }
        if (fl_st2_ua != FLST_ACK){                     // ST2 = ACK ?
                rc = decode_status(fl_st2_ua);          //否
                return  rc;                             // case [D]
        }
        if (is_end)                             //发送所有用户数据?
                break;                          //是
        //continue;
    }
    return  FLC_NO_ERR;  // case [A]
}
```

## 6.10 Block 空白检测命令

### 6.10.1 处理流程图

Block 空白检测命令处理流程图



### 6.10.2 流程描述

<1> 从前一帧接收到下一命令帧发送等待(等待时间 $t_{COM}$)。

<2> Block 空白检测命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT8}$(MAX.)X block数)。

<4> 检查状态码。

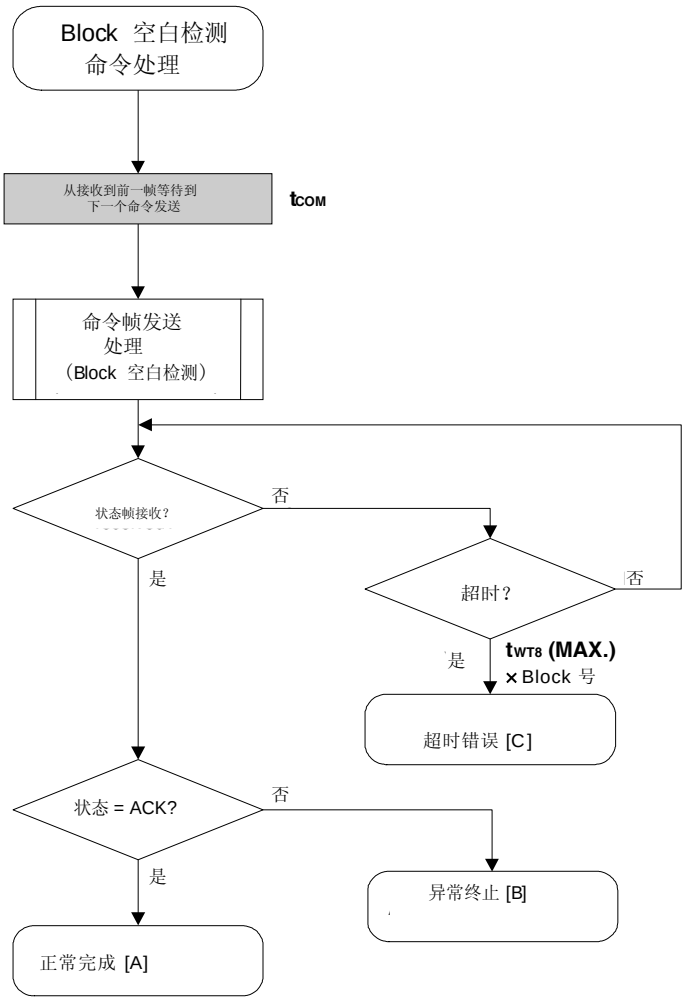
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [B]

### 6.10.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答(ACK) | 06H | 命令正常执行，并且所有指定的block都是空白。 |
| 异常终止 [B] | 参数错误 | 05H | block的序号超出范围。 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 异常应答(NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN)或无ETX)。 |
| | MRG11 错误 | 1BH | flash存储器中指定的block不是空白。 |
| 超时错误 [C] | | ~ | 没有在指定时间内接收到状态帧。 |

### 6.10.4 流程图


Block 空白检测
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理
（Block 空白检测）
状态帧接收?
否
是
超时?
否
是
tWT8 (MAX.)
× Block 号
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
正常完成 [A]

‘

### 6.10.5 例子程序

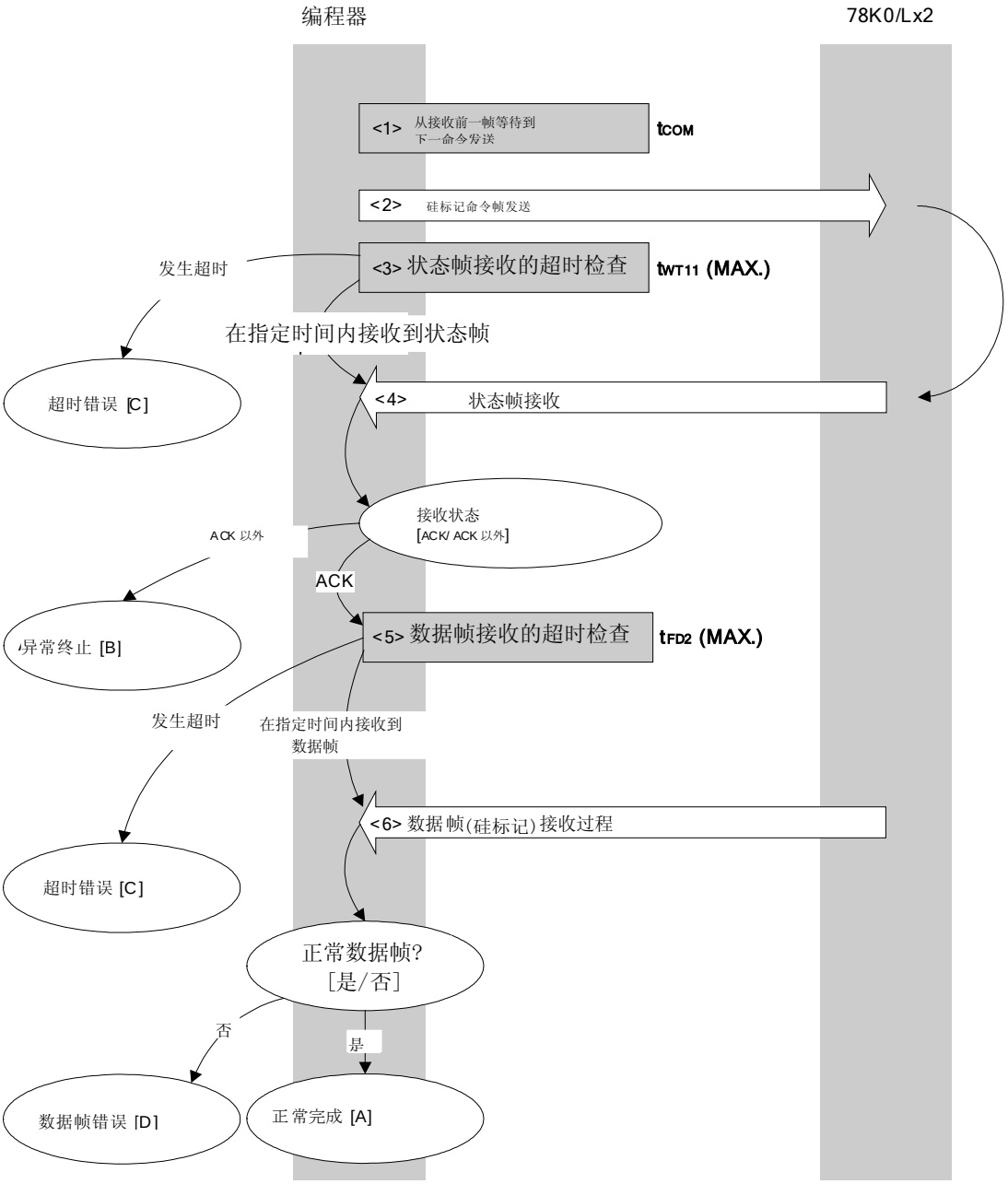
如下所示为Block 空白检测命令处理的一个例子程序。

```
/*****************************************************************/
/*                                                               */
/* Block 空白检测命令                                            */
/*                                                               */
/*****************************************************************/
/* [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 block num

        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);         //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;

}
```

## 6.11 硅标记命令

### 6.11.1 处理流程图

硅标记命令处理流程图



### 6.11.2 流程描述

<1> 从接收到前一帧到发送下一个命令帧等待(等待时间 $t_{COM}$)。

<2> 硅标记命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT11}$(MAX.))。

<4> 检查状态码

当 ST1 = ACK: 执行 <5>

当 ST1 ≠ ACK: 异常终止 [B]

<5> 执行超时检测直到接收到数据帧（硅标记数据）。

如果发生超时，返回超时错误[C] (超时时间$t_{FD2}$(MAX.))。

<6> 检查接收到的数据帧(硅标记数据) 。

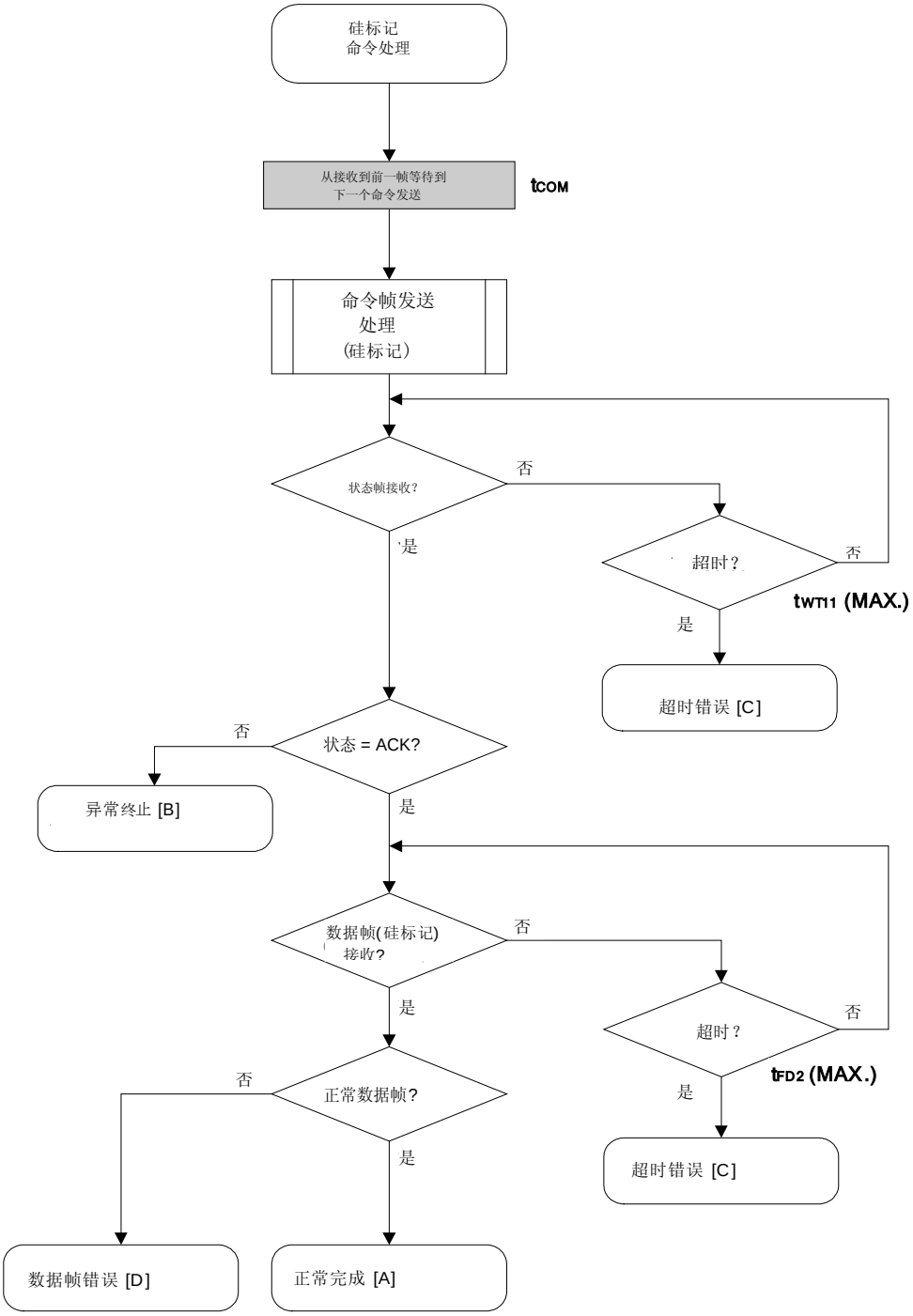
如果数据帧正常: 正常完成 [A]

如果数据帧异常: 数据帧错误[D]

### 6.11.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答(ACK) | 06H | 执行命令正常并且获取硅标记正常。 |
| 异常终止 [B] | 校验和错误 | 07H | 发送命令帧的校验和不相等 |
| | 异常应答(NACK) | 15H | 命令帧数据异常（例如无效数据长度（LEN）或无ETX）。 |
| | 读错误 | 20H | 读取安全信息失败 |
| 超时错误 [C] | | ~ | 没有在指定时间内接收到状态帧或数据帧。 |
| 数据帧错误 [D] | | ~ | 接收到的硅型号数据帧的校验和不相等。 |

### 6.11.4 流程图


硅标记
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理
(硅标记)
状态帧接收?
否
是
超时?
否
tWT11 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
异常终止 [B]
是
数据帧(硅标记)
接收?
否
是
超时?
否
tFD2 (MAX.)
是
超时错误 [C]
正常数据帧?
否
是
数据帧错误 [D]
正常完成 [A]

### 6.11.5 例子程序

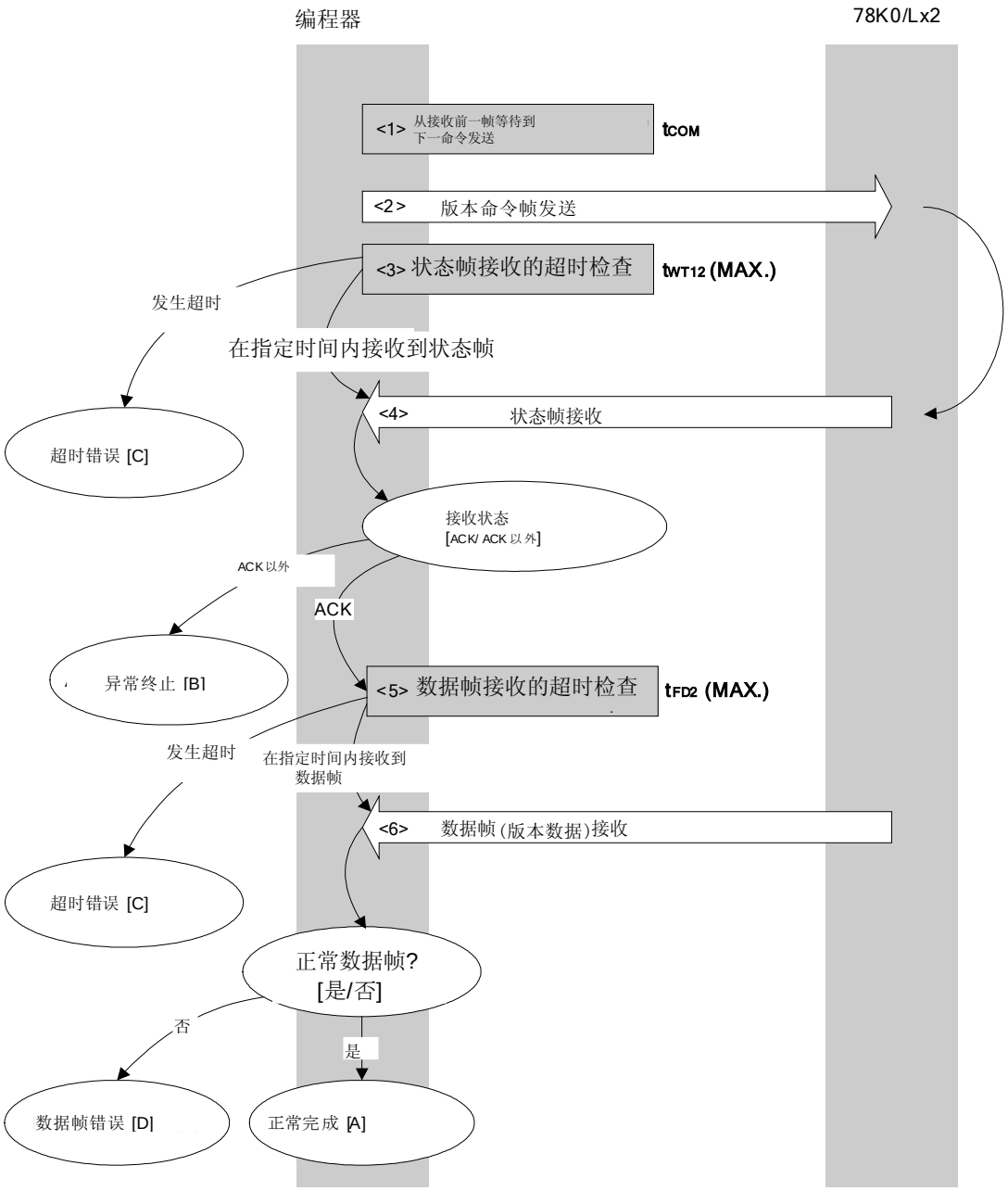
如下所示为硅标记命令处理的一个例子程序。

```
/*****************************************************************/
/*                                                               */
/*取硅标记命令                                                   */
/*                                                               */
/*****************************************************************/
/* [i] u8 *sig     ... 标记保存区域          */
/* [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 GET SIGNATURE 命令

        rc = get_sfrm_ua(fl_ua_sfrm, tWT11_MAX);  //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]
        }


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

## 6.12 版本获取命令

### 6.12.1 处理流程图

版本获取命令处理流程图



### 6.12.2 流程描述

<1> 从接收到上一帧到下一个命令帧发送等待(等待时间 $t_{COM}$)。

<2> 版本获得命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT12}$(MAX.))。

<4> 检查状态码

当 ST1 = ACK: 执行 <5>.

当 ST1 ≠ ACK: 异常终止[B]

<5> 执行超时检测直到接收到数据帧（版本数据）。

如果发生超时，返回超时错误[C] (超时时间$t_{FD2}$(MAX.))。

<6> 检查接收到的数据帧 (版本数据)。

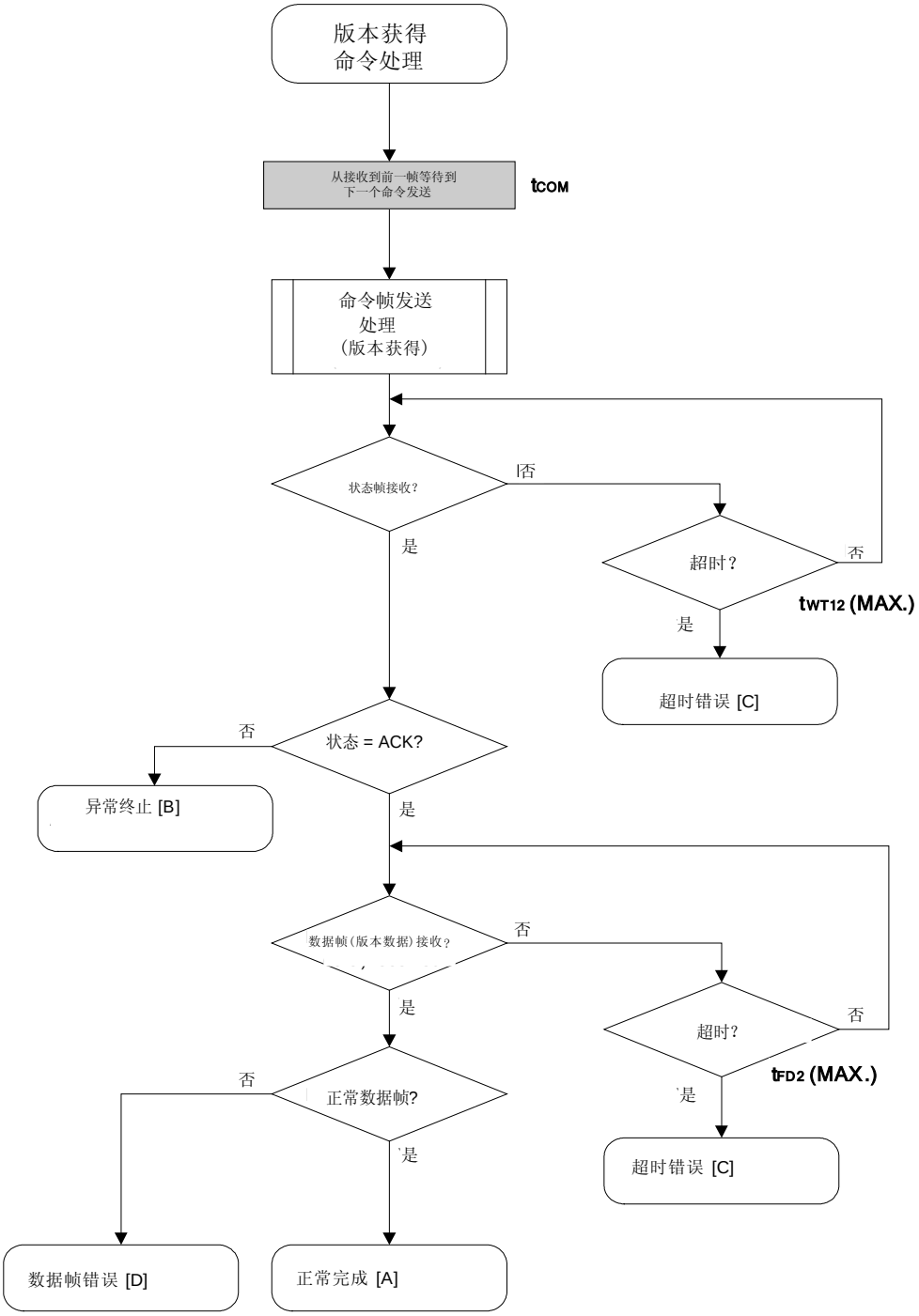
如果数据帧正常: 正常完成 [A]

如果数据帧异常: 数据帧错误 [D]

### 6.12.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常应答 (ACK) | 06H | 命令正常执行，并且正常获得版本数据。 |
| 异常终止 [B] | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (如无效数据长度 (LEN) 或无 ETX)。 |
| 超时错误 [C] | | - | 没有在指定时间内接收到状态帧或数据帧。 |
| 数据帧错误 [D] | | - | 接收到的版本数据的数据帧的校验和不相等。 |

### 6.12.4 流程图


版本获得
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理
（版本获得）
状态帧接收?
否
是
超时?
否
tWT12 (MAX.)
是
超时错误 [C]
否
状态 = ACK?
异常终止 [B]
是
数据帧(版本数据)接收?
否
是
超时?
否
tFD2 (MAX.)
是
超时错误 [C]
否
正常数据帧?
是
数据帧错误 [D]
正常完成 [A]

### 6.12.5 例子程序

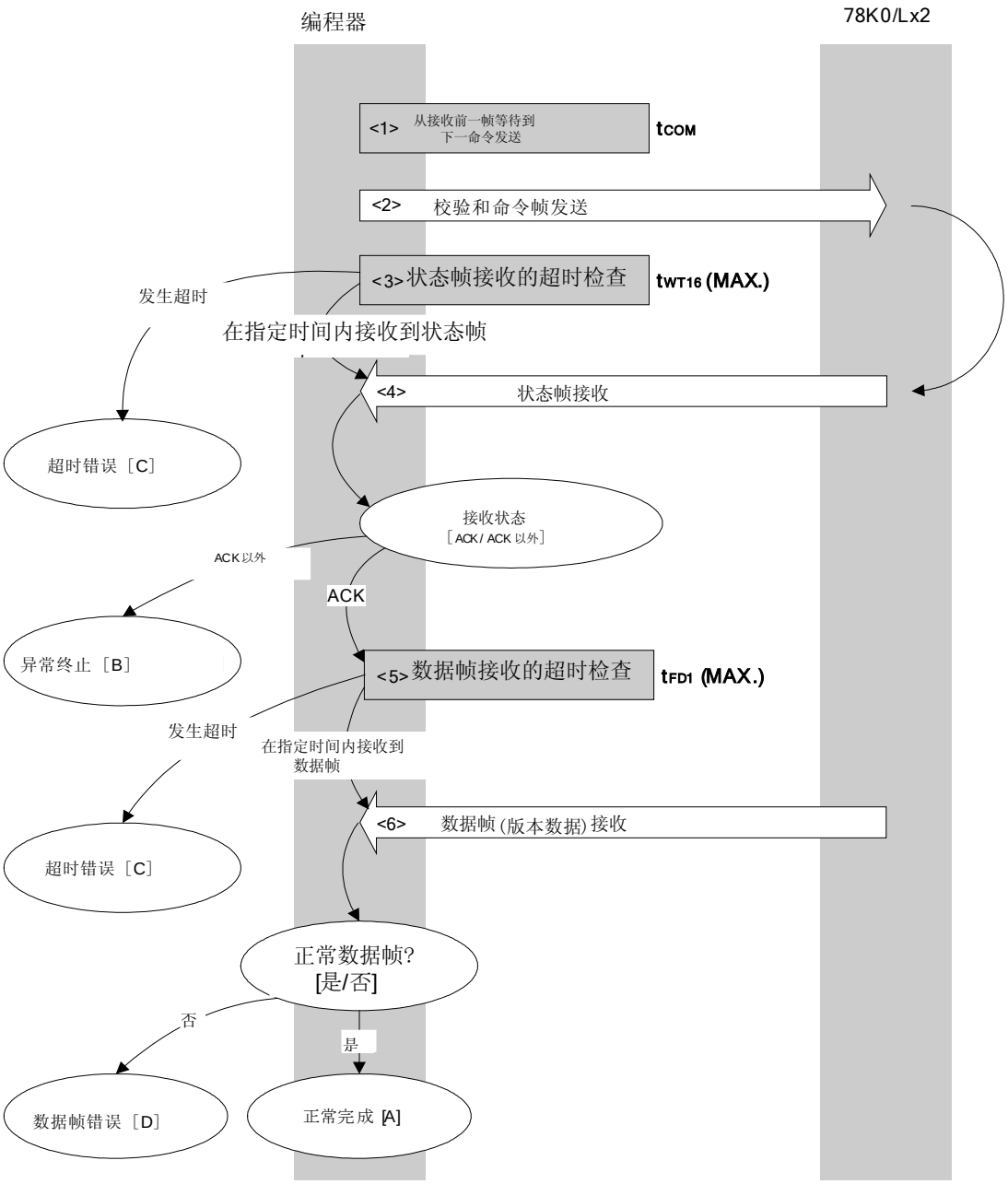
如下所示为版本获得命令处理的一个例子程序。

```
/*********************************************************************/
/*                                                                   */
/*获得设备/固件版本命令                                               */
/*                                                                   */
/*********************************************************************/
/* [i] u8 *buf     ... 版本数据保存区域        */
/* [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 GET VERSION 命令

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

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

## 6.13 校验和命令

### 6.13.1 处理流程

校验和命令处理流程图



### 6.13.2 流程描述

<1> 从接收到前一帧到下一个命令帧发送等待(等待时间 $t_{COM}$)。

<2> 校验和命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT16}$(MAX.))。

<4> 检查状态码。

当 ST1 = ACK: 执行<5>.

当 ST1 ≠ ACK: 异常终止 [B]

<5> 执行超时检测直到接收到数据帧（校验和数据）。

如果发生超时，返回超时错误[C] (超时时间$t_{FD1}$(MAX.))。

<6> 检查接收到的数据帧(校验和数据)。

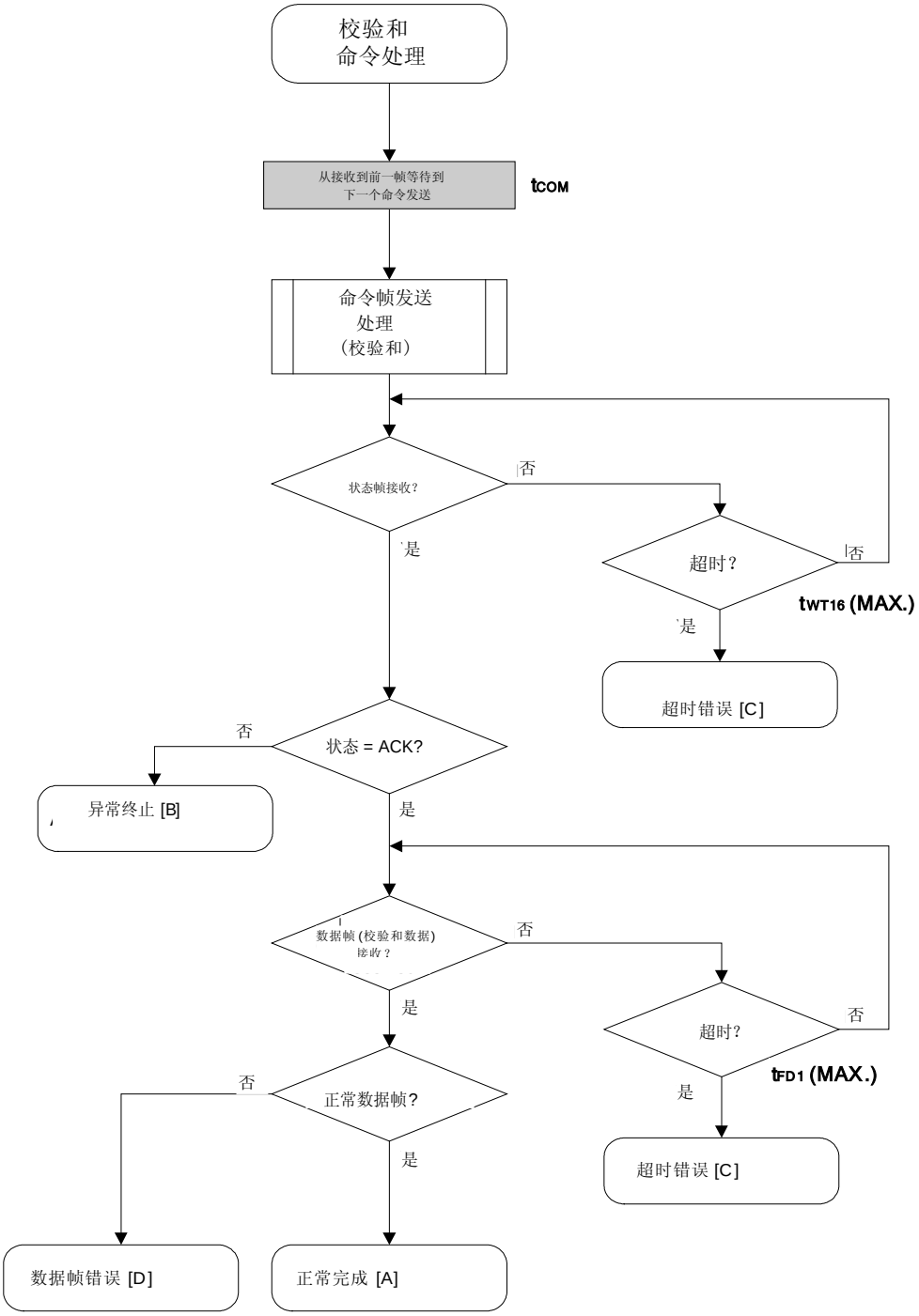
如果数据帧正常: 正常完成 [A]

如果数据帧异常: 数据帧错误 [D]

### 6.13.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答(ACK) | 06H | 命令正常运行，并且校验和数据正常获得。 |
| 异常终止 [B] | 参数错误 | 05H | 指定的起始/结束地址超出flash存储器的范围，或指定的地址不是以 2 KB 为单位的。 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 异常应答(NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN) 或无 ETX)。 |
| 超时错误 [C] | | - | 没有在指定时间内接收到状态帧或数据帧。 |
| 数据帧错误 [D] | | - | 接收到的版本数据的数据帧的校验和不相等。 |

### 6.13.4 流程图


校验和
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理
（校验和）
状态帧接收？
否
是
超时？
否
tWT16 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
异常终止 [B]
是
数据帧 (校验和数据)
接收？
否
是
超时？
否
tFD1 (MAX.)
是
超时错误 [C]
正常数据帧?
否
是
数据帧错误 [D]
正常完成 [A]

### 6.13.5 例子程序

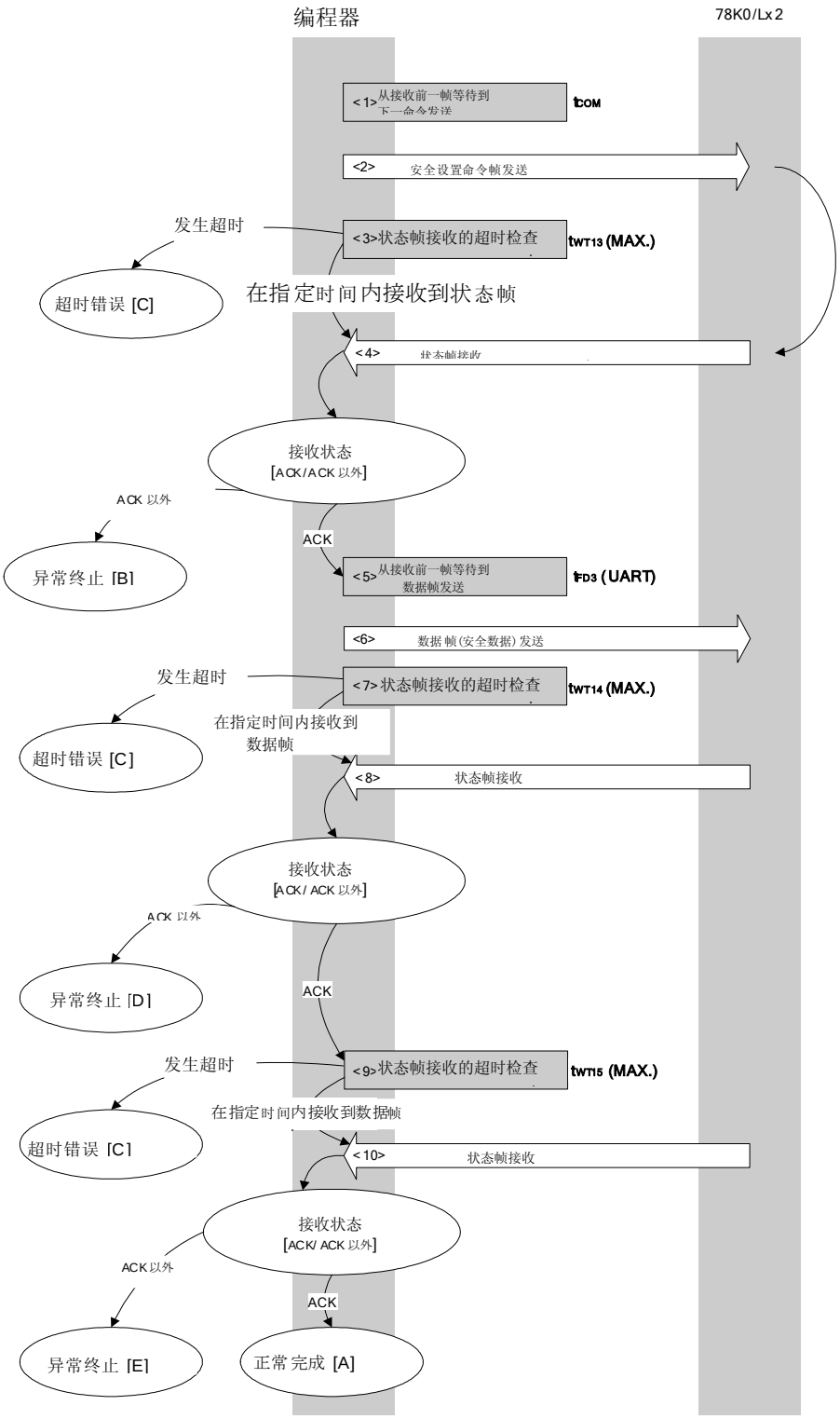
如下所示为校验和命令处理的一个例子程序。

```
/*****************************************************************/
/*                                                               */
/*获取校验和命令                                                 */
/*                                                               */
/*****************************************************************/
/* [i] u16 *sum    ... 校验和保存区域                            */
/* [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                                 */
        /*************************************************/
        // set params
        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL, EAH/EAM/EAL


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

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

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

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

        /*************************************************/
        /*      获取数据帧(检验和数据)                   */
        /*************************************************/
        rc = get_dfrm_ua(fl_rxdata_frm, tFD1_MAX);//获取状态帧
        if (rc){                                                //如果没有错误,
                return rc;                              // 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]
}
```

## 6.14 安全设置命令

### 6.14.1 处理流程图

安全设置命令处理流程图



### 6.14.2 流程描述

<1> 从接收到前一帧到下一个命令帧发送等待(等待时间$t_{COM}$)。

<2> 安全设置命令由命令帧发送处理发送。

<3> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT13}$(MAX.))。

<4> 检查状态码

当 ST1 = ACK: 执行 <5>.

当 ST1 ≠ ACK: 异常终止 [B]

<5> 从接收到前一帧到下一个数据帧发送等待 (等待时间 $t_{FD3}$(UART))。

<6> 数据帧(安全设置数据)由数据帧发送处理发送。

<7> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT14}$(MAX.))。

<8> 检查状态码

当 ST1 = ACK: 执行<9>

当 ST1 ≠ ACK: 异常终止 [D]

<9> 执行超时检测直到接收到状态帧。

如果发生超时，返回超时错误[C] (超时时间$t_{WT15}$(MAX.))。

<10> 检查状态码。

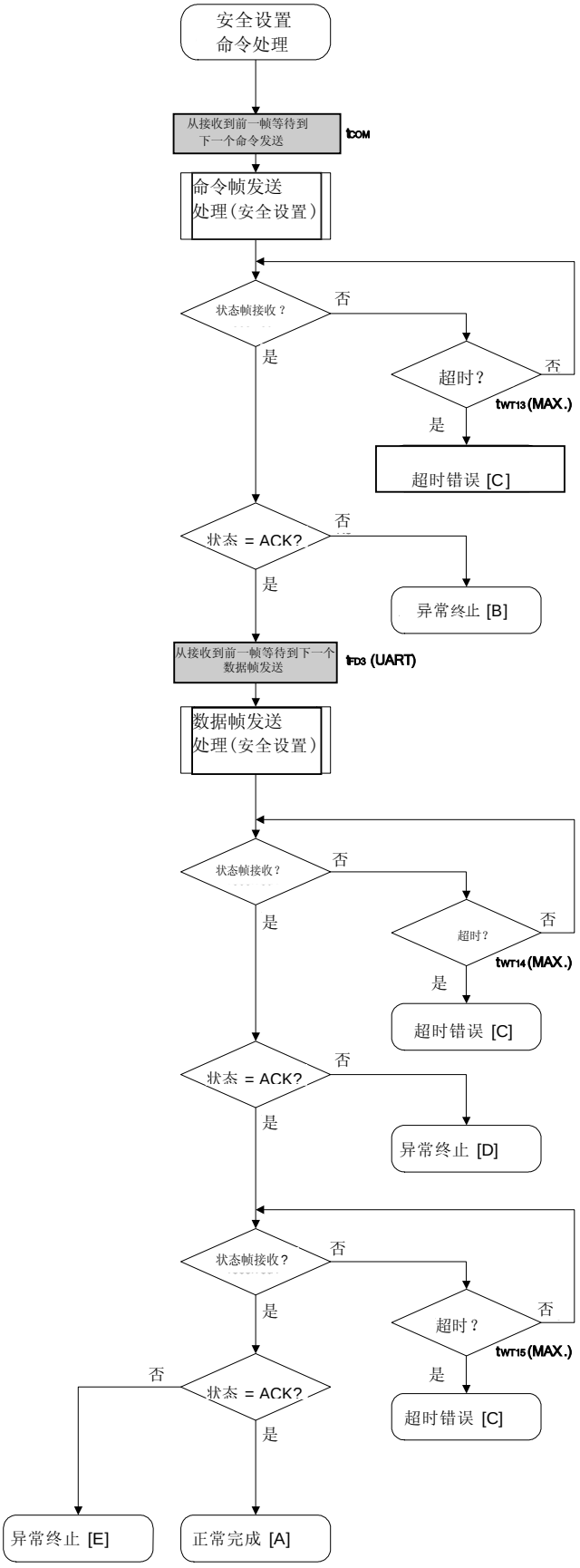
当 ST1 = ACK: 正常完成 [A]

当 ST1 ≠ ACK: 异常终止 [E]

### 6.14.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 命令正常执行，并且安全设置正常完成。 |
| 异常终止 [B] | 参数错误 | 05H | BLK 或 PAG不为00H。 |
| | 校验和错误 | 07H | 发送的命令帧或数据帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN) 或无ETX)。 |
| 超时错误 [C] | | - | 没有在指定时间内接收到状态帧或数据帧。 |
| 异常终止 [D] | FLMD 错误 | 18H | 发生写错误。 |
| | 写入错误 | 1CH | 发生写错误 (包括安全数据已被设置的情况)。 |
| 异常终止 [E] | MRG11 错误 | 1BH | 发生内部校验错误。 |

### 6.14.4 流程图


安全设置
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(安全设置)
状态帧接收？
否
是
超时？
否
tWT13 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
从接收到前一帧等待到下一个
数据帧发送
tFD3 (UART)
数据帧发送
处理(安全设置)
状态帧接收？
否
是
超时？
否
tWT14 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [D]
状态帧接收？
否
是
超时？
否
tWT15 (MAX.)
是
超时错误 [C]
状态 = ACK?
否
是
异常终止 [E]
正常完成 [A]

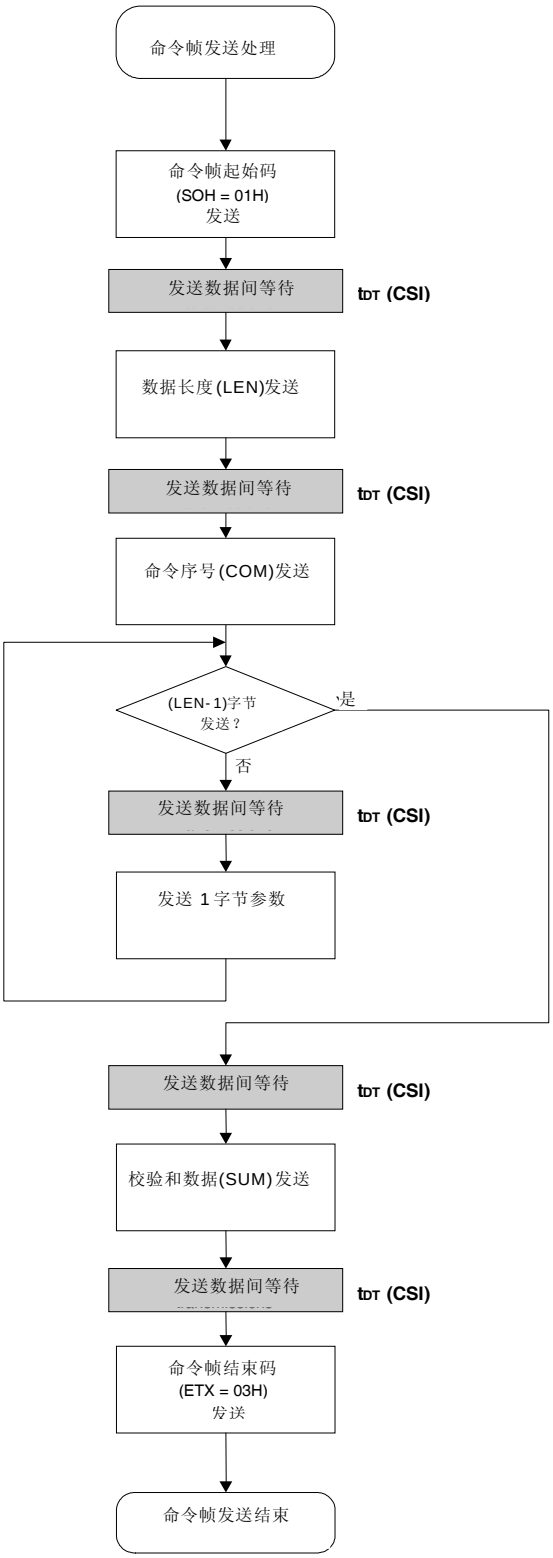
### 6.14.5 例子程序

如下所示为安全设置命令处理的一个例子程序。

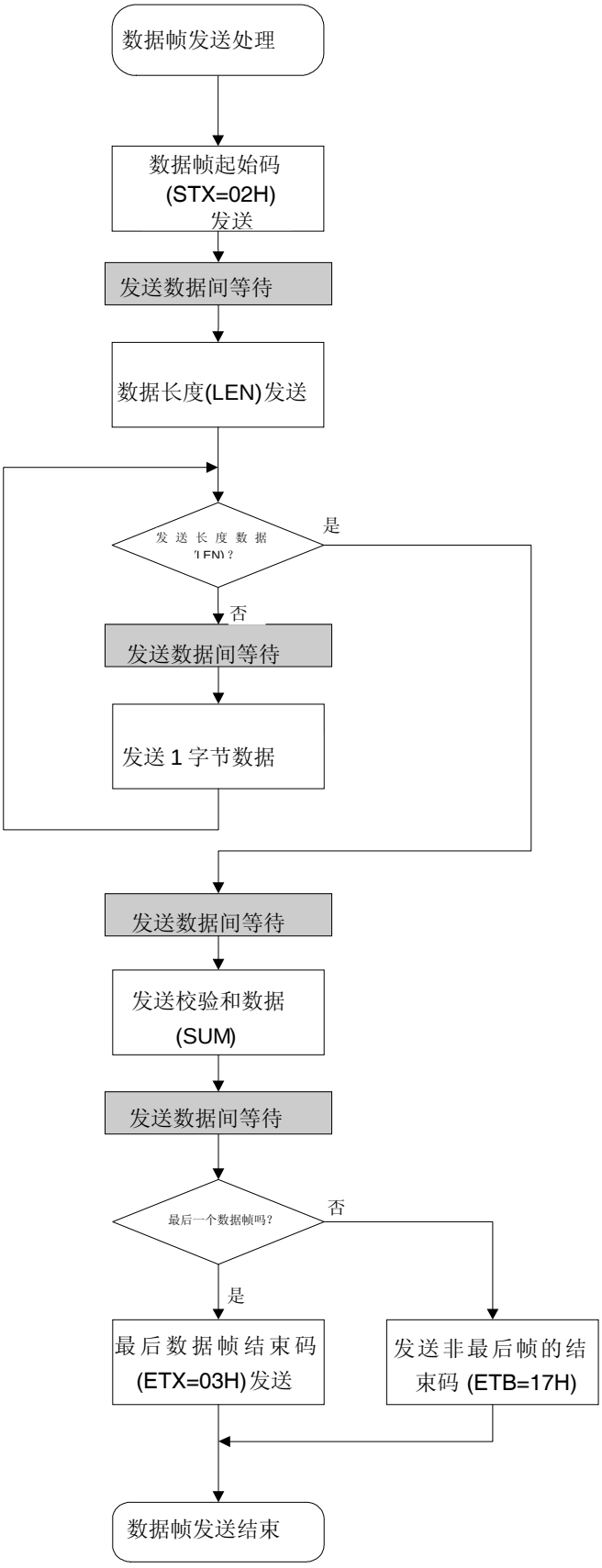
```
/*****************************************************************/
/*                                                               */
/*设置安全标志命令                                               */
/*                                                               */
/*****************************************************************/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

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

        fl_txdata_frm[1] = 0x03;                // "BOT" (恒为 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_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]
        }


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

        fl_wait(tFD3_UA);
```

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


    rc = get_sfrm_ua(fl_ua_sfrm, 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]
    }

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

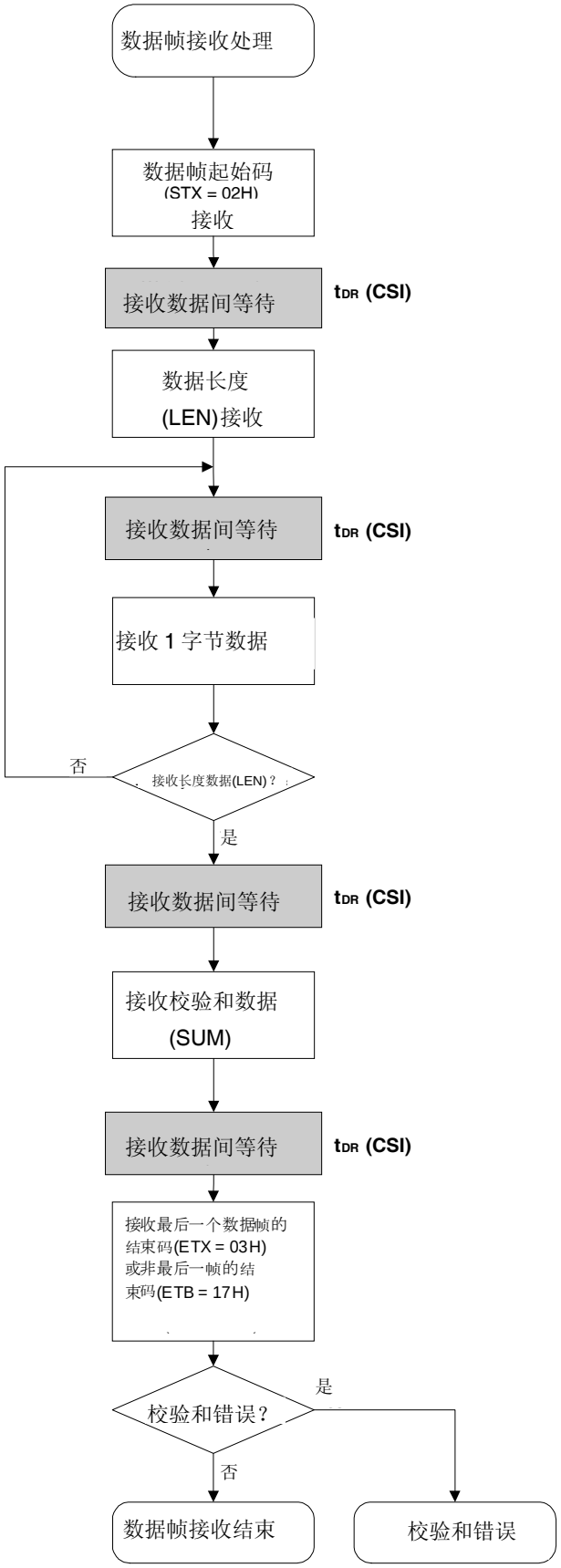
## 7.1 命令帧发送处理流程



## 7.2 数据帧发送处理流程



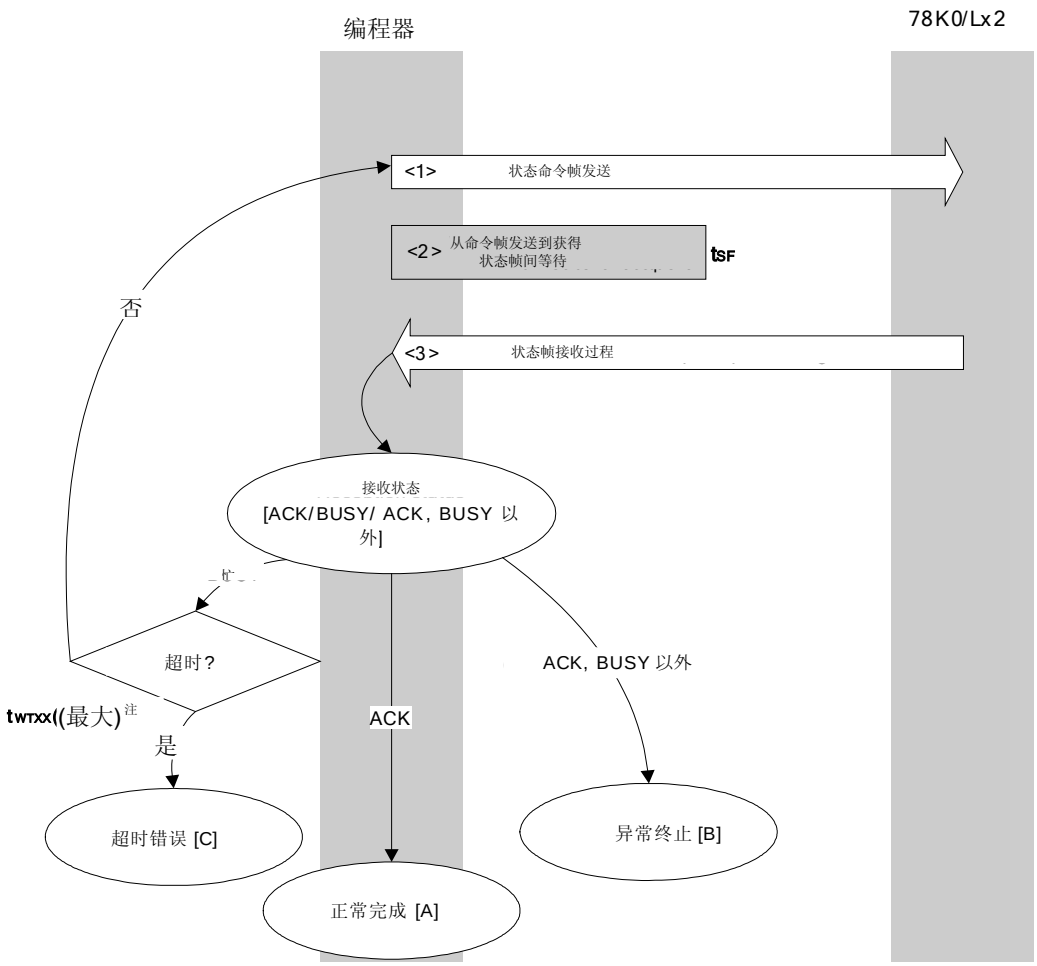
## 7.3 数据帧接收处理流程图



## 7.4 状态命令

### 7.4.1 处理流程图

状态命令处理流程图




**注 根据执行的命令不同，应用说明不同。**

### 7.4.2 流程描述

<1> 状态命令由命令帧发送处理发送。

<2> 从命令发送等待到状态帧接收(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])。
如果没有超时，从步骤<1>开始重复执行。
如果超时，返回超时错误[C]。

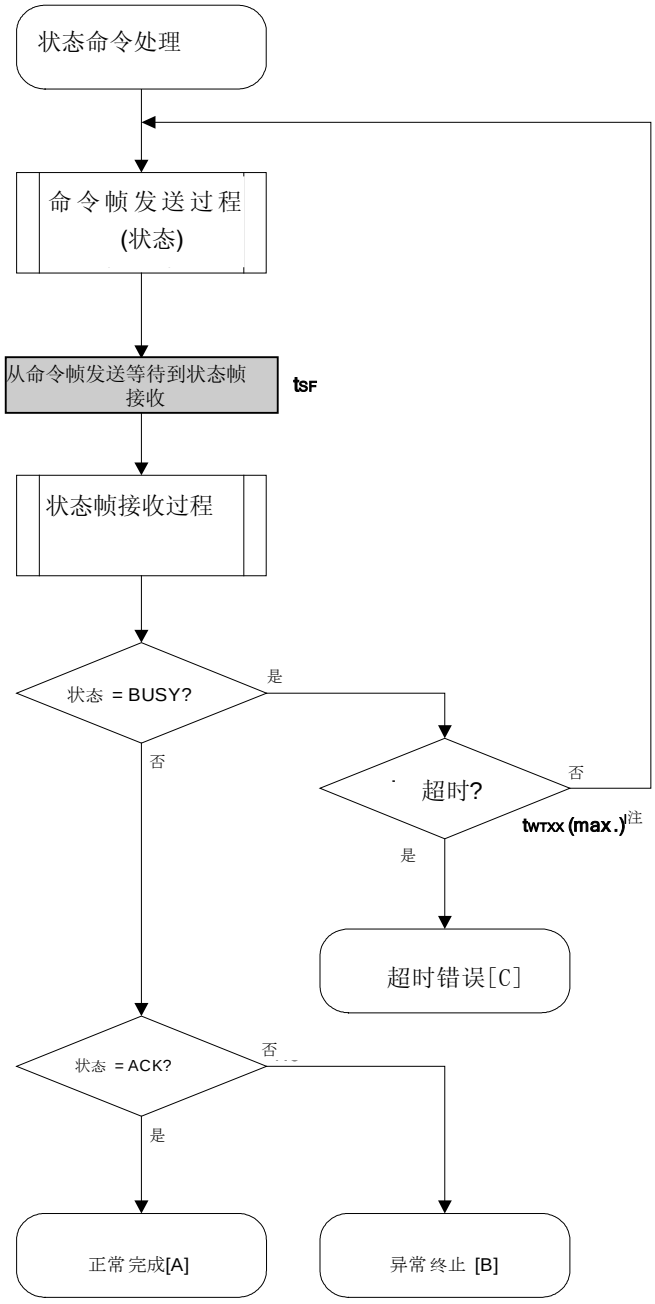
当 ST1 ≠ ACK, BUSY: 异常终止 [B]

**注 根据执行命令不同，应用说明不同。**

### 7.4.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 从78K0/Lx2发送的状态帧已经正常收到 |
| 无效异常终止 [B] | 命令错误 | 04H | 收到未支持命令或异常帧 |
| | 参数错误 | 05H | 命令信息（参数）无效 |
| | 校验和错误 | 07H | 从编程器发送的数据帧异常 |
| | 校验错误 | 0FH | 从编程器发送的数据帧产生校验错误 |
| | 保护错误 | 10H | 试图执行被安全设置命令禁止的处理 |
| | 异常应答 (NACK) | 15H | 否定应答 |
| | FLMD 错误 | 18H | 产生写错误 |
| | MRG10错误 | 1AH | 产生擦除错误 |
| | MRG11错误 | 1BH | 在写入数据期间产生内部校验错误或空白检测错误 |
| | 写错误 | 1CH | 产生写错误 |
| 超时错误 [C] | | ~ | 命令发送后，在指定的时间过去后仍返回BUSY响应。 |

### 7.4.4 流程图




**注** 根据执行命令不同，应用说明不同。

### 7.4.5 例子程序

如下所示为状态命令处理的一个例子程序。

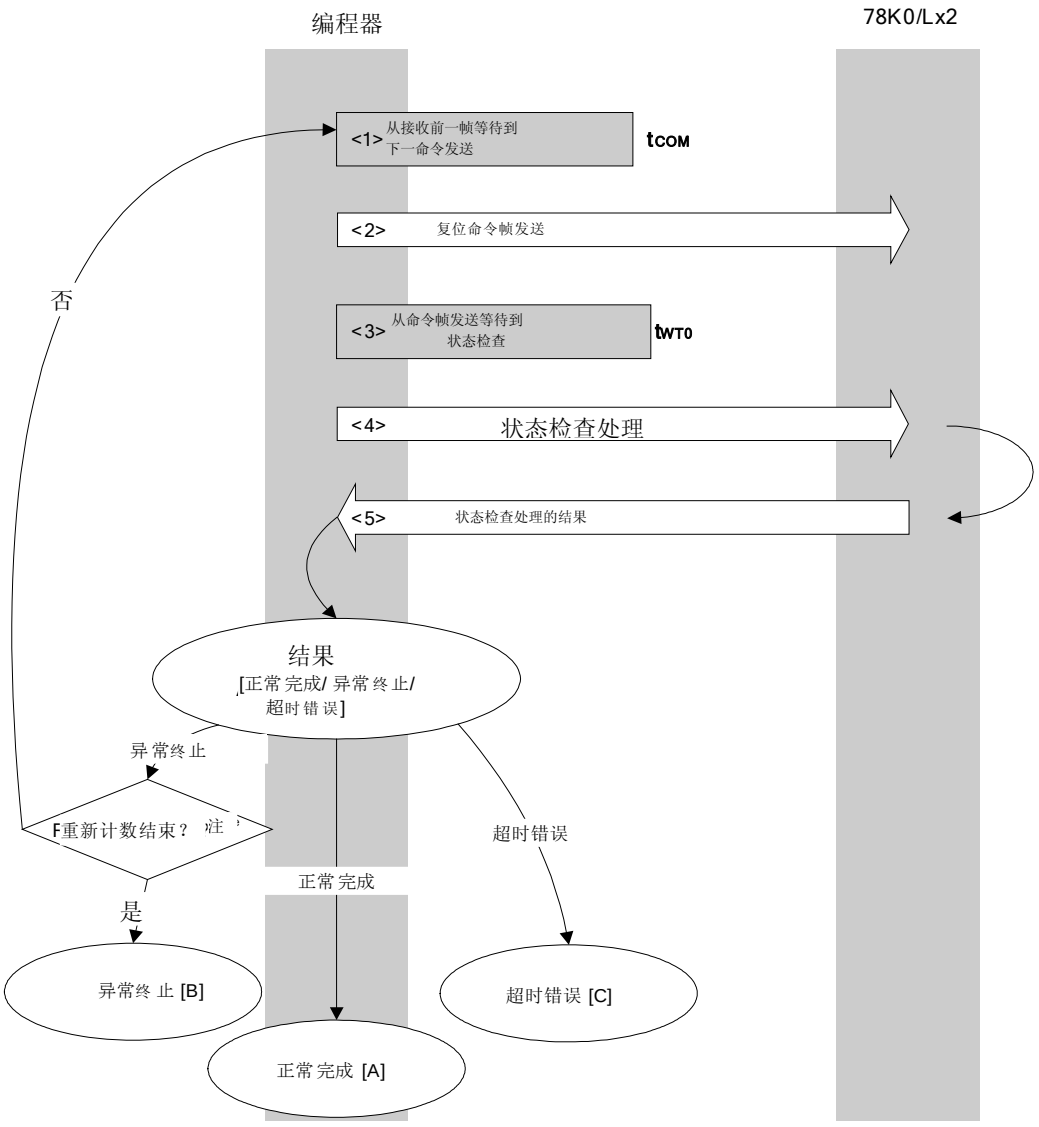
```
/*****************************************************************/
/*                                                               */
/*获取状态命令(CSI)                                              */
/*                                                               */
/*****************************************************************/
/* [r] u16                  ... 解码状态或错误代码               */
/*                                                               */
/* (see fl.h/fl-proto.h &                                        */
/*                  definition of decode_status() in fl.c)       */
/*****************************************************************/
static u16      fl_csi_getstatus(u32 limit)
{
        u16     rc;
        start_flto(limit);

        while(1){

        put_cmd_csi(FL_COM_GET_STA, 1, fl_cmd_prm);    // 发送 "Status" 命令
                                                       // 帧
                fl_wait(tSF);                          // 等待

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

                switch(rc){
                        case    FLC_BUSY:
                                if (check_flto())      //超时?
                                        return  FLC_DFTO_ERR; //是，超时// case [C]
                                continue;              //不, 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;               //不, 重试
                }
                break;                          // ACK 或其他错误(BUSY 除外)
        }
```

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

}
```

## 7.5 复位命令

### 7.5.1 处理流程图

状态命令处理流程图




**注** 复位命令发送不要执行重复计数(至多 16 次)。

### 7.5.2 流程描述

<1> 从接收到前一帧开始等待到下一个命令发送(等待时间$t_{COM}$)。

<2> 复位命令由命令帧发送处理发送。

<3> 从发送命令开始等待到状态检查处理(等待时间 $t_{WT0}$)。

<4> 状态帧由状态检查处理获取。

<5> 下面的过程根据状态检查处理的结果执行。

当处理正常结束: 正常完成 [A]

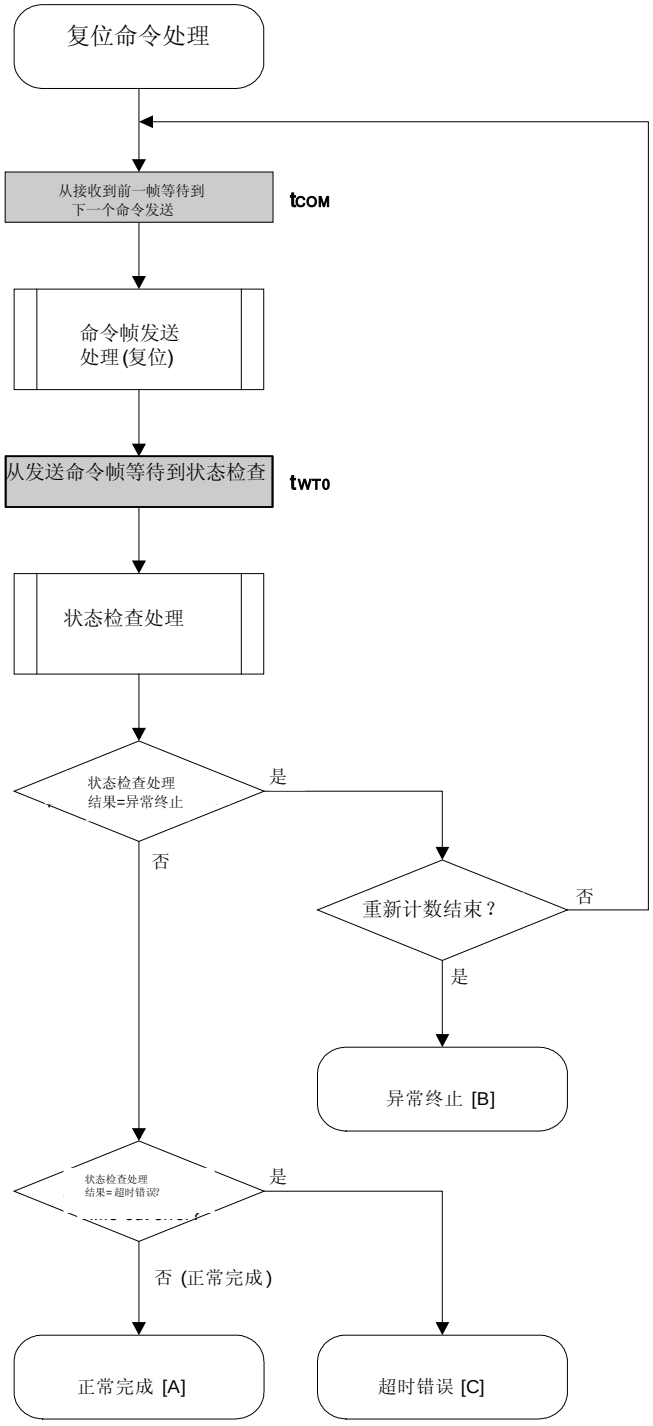
当处理异常结束: 如果重复计数没有结束从步骤<1> 开始重复执行。
如果重复计数结束处理异常终止[B]。

当出现超时错误: 返回超时错误[C]。

### 7.5.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 命令正常执行，并且编程器与78K0/Lx2 同步。 |
| 异常终止 [B] | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常(如无效数据长度(LEN) 或无 ETX)。 |
| 超时错误 [C] | | - | 状态检测处理超时。 |

### 7.5.4 流程图


复位命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(复位)
从发送命令帧等待到状态检查
tWT0
状态检查处理
状态检查处理
结果=异常终止
是
否
重新计数结束？
否
是
异常终止 [B]
状态检查处理
结果=超时错误?
是
否 (正常完成)
正常完成 [A]
超时错误 [C]

### 7.5.5 例子程序

如下所示为复位命令处理的一个例子程序。

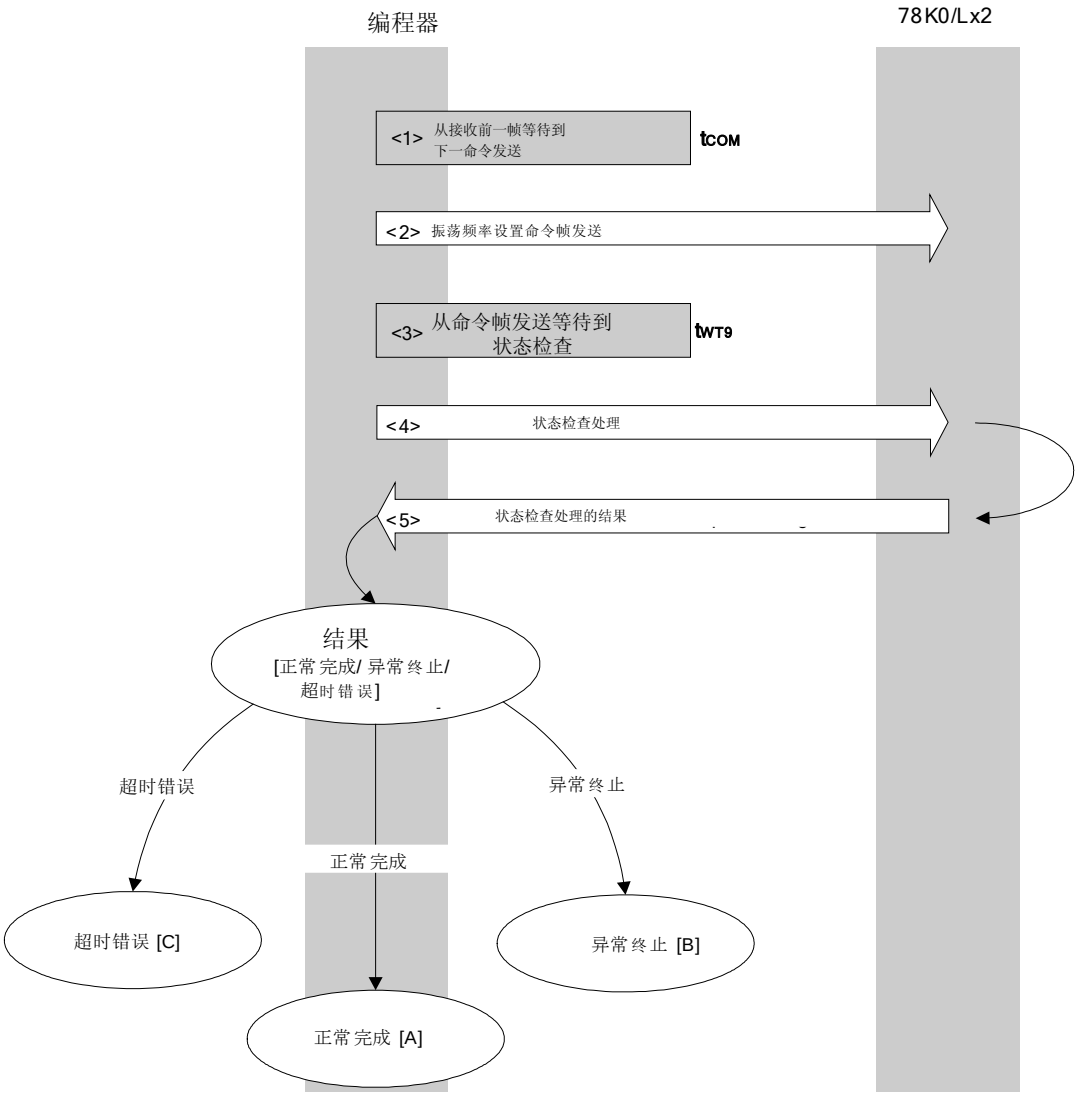
```
/*****************************************************************/
/*                                                               */
/*复位命令(CSI)                                                  */
/*                                                               */
/*****************************************************************/
/* [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"Reset" 命令帧

                fl_wait(tWT0);

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

                if (rc == FLC_DFTO_ERR)//超时错误 ?
                        break;                  //是// case [C]
                if (rc == FLC_ACK)              // Ack ?
                        break;                  //是// case [A]
                //continue;                                 // case [B] (如果跳出循环)
        }
//      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;
}
```

## 7.6 振荡频率设置命令

在CSI通信期间此命令没有必要执行(如果根据编程者的指定需要在CSI通信期间执行此命令，将频率设置为8 MHz)。

### 7.6.1 处理流程图

振荡频率设置命令处理流程图



### 7.6.2 流程描述

<1> 从接收到前一帧等待到下一个命令发送 (等待时间$t_{COM}$)。

<2> 振荡频率设置命令由命令帧发送处理发送。

<3> 从命令发送开始等待到状态检查处理 (等待时间$t_{WT9}$)。

<4> 状态帧由状态检查处理获取。

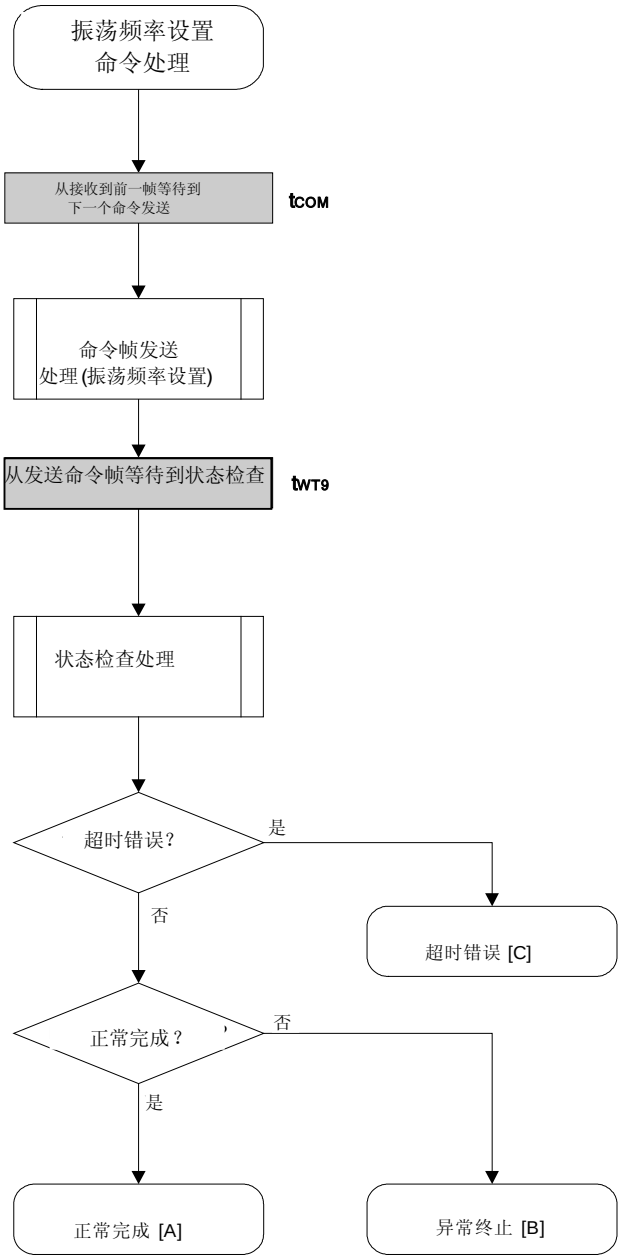
<5> 如下处理根据状态检查过程的结果执行。

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

### 7.6.3 处理完成时的状态

| 处理完成时的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成 [A] | 正常应答 (ACK) | 06H | 命令正常执行，并且78K0/Lx2的操作频率设置正确。 |
| 异常终止 [B] | 参数错误 | 05H | 振荡频率值超出范围。 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等。 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (如无效数据长度(LEN)或无 ETX)。 |
| 超时错误 [C] | | ~ | 没有在指定时间内接收到状态帧。 |

### 7.6.4 流程图


振荡频率设置
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(振荡频率设置)
从发送命令帧等待到状态检查
tWT9
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]

### 7.6.5 例子程序

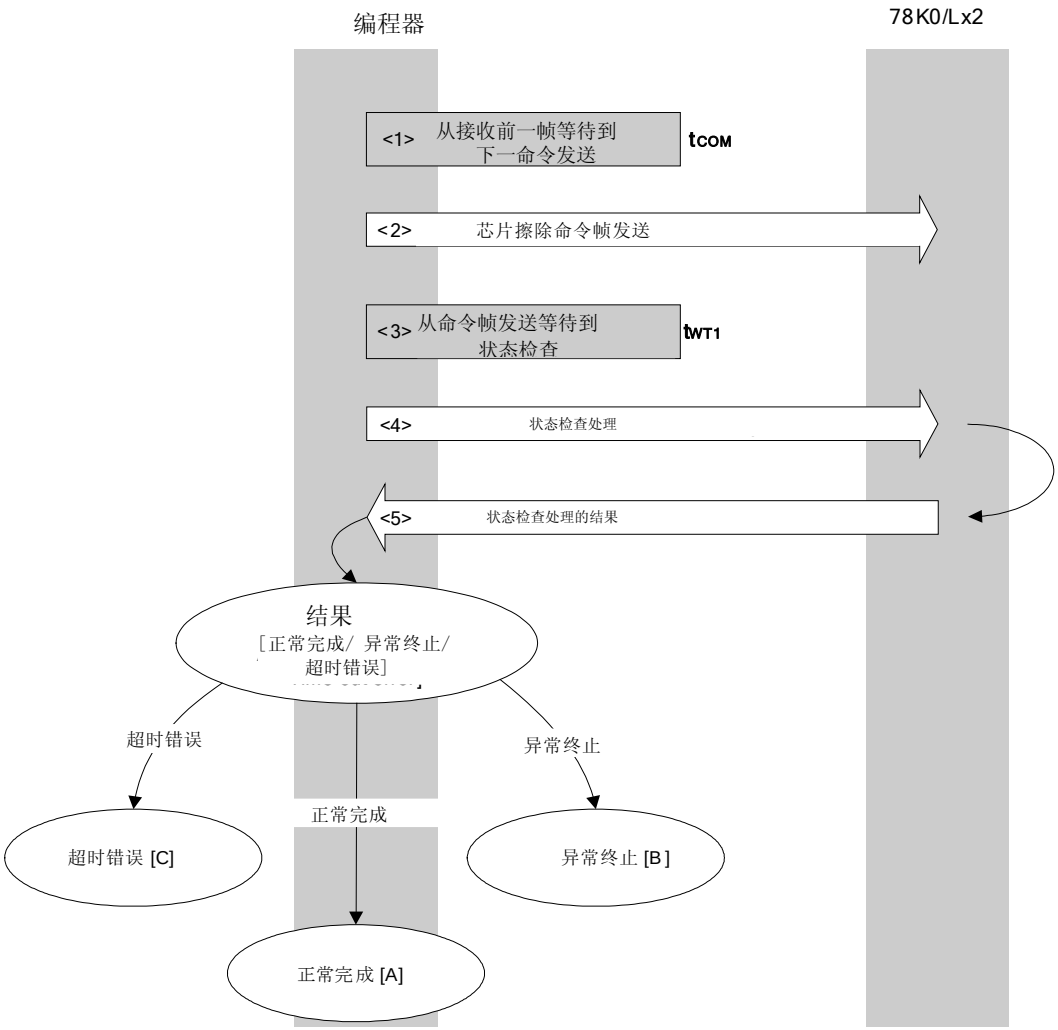
如下所示为振荡频率设置命令处理的一个例子程序。

```
/*****************************************************************/
/*                                                               */
/*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 "Oscillation Frequency Set" 命令

        fl_wait(tWT9);

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

## 7.7 芯片擦除命令

### 7.7.1 处理流程图

芯片擦除命令处理流程图



### 7.7.2 处理流程的描述

<1> 从上一帧接收到下一个命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送芯片擦除命令。

<3> 从命令发送到状态检测处理的等待 (等待时间 $t_{WT1}$)。

<4> 通过状态检测处理获得状态帧。

<5> 接下来的处理根据状态检测处理的结果而定。

当处理正常结束： 正常完成[A]

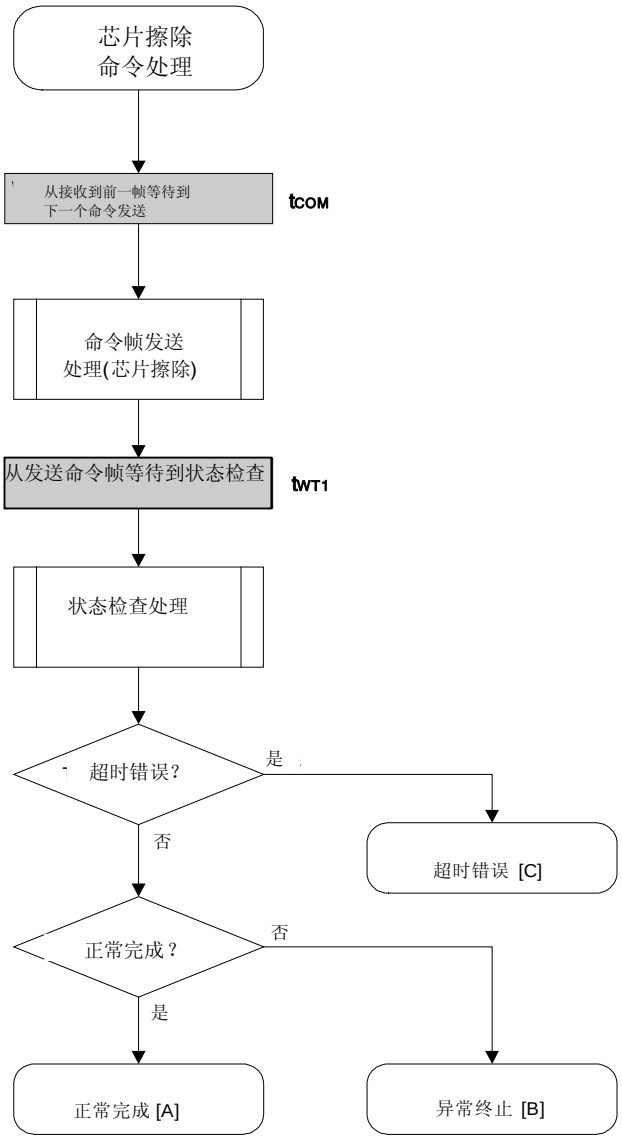
当处理异常结束： 异常终止[B]

当发生超时错误： 返回一个超时错误 [C]

### 7.7.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常应答 (ACK) | 06H | 命令正常执行，芯片擦除正常执行 |
| 异常终止[B] | 校验和错误 | 07H | 发送命令帧中的校验和不相等 |
| | 保护错误 | 10H | 安全设置里面设定了禁止芯片擦除 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (例如数据长度不对 (LEN) 或者没有 ETX) |
| | 擦除错误 | 1AH | 发生一个擦除错误 |
| 超时错误[C] | | - | 在指定时间内没有接收到状态帧 |

### 7.7.4 流程图


芯片擦除
命令处理
从接收到前一帧等待到
下一个命令发送
tCOM
命令帧发送
处理(芯片擦除)
从发送命令帧等待到状态检查
tWT1
状态检查处理
超时错误?
是
超时错误 [C]
否
正常完成?
否
是
正常完成 [A]
异常终止 [B]

### 7.7.5 例子程序

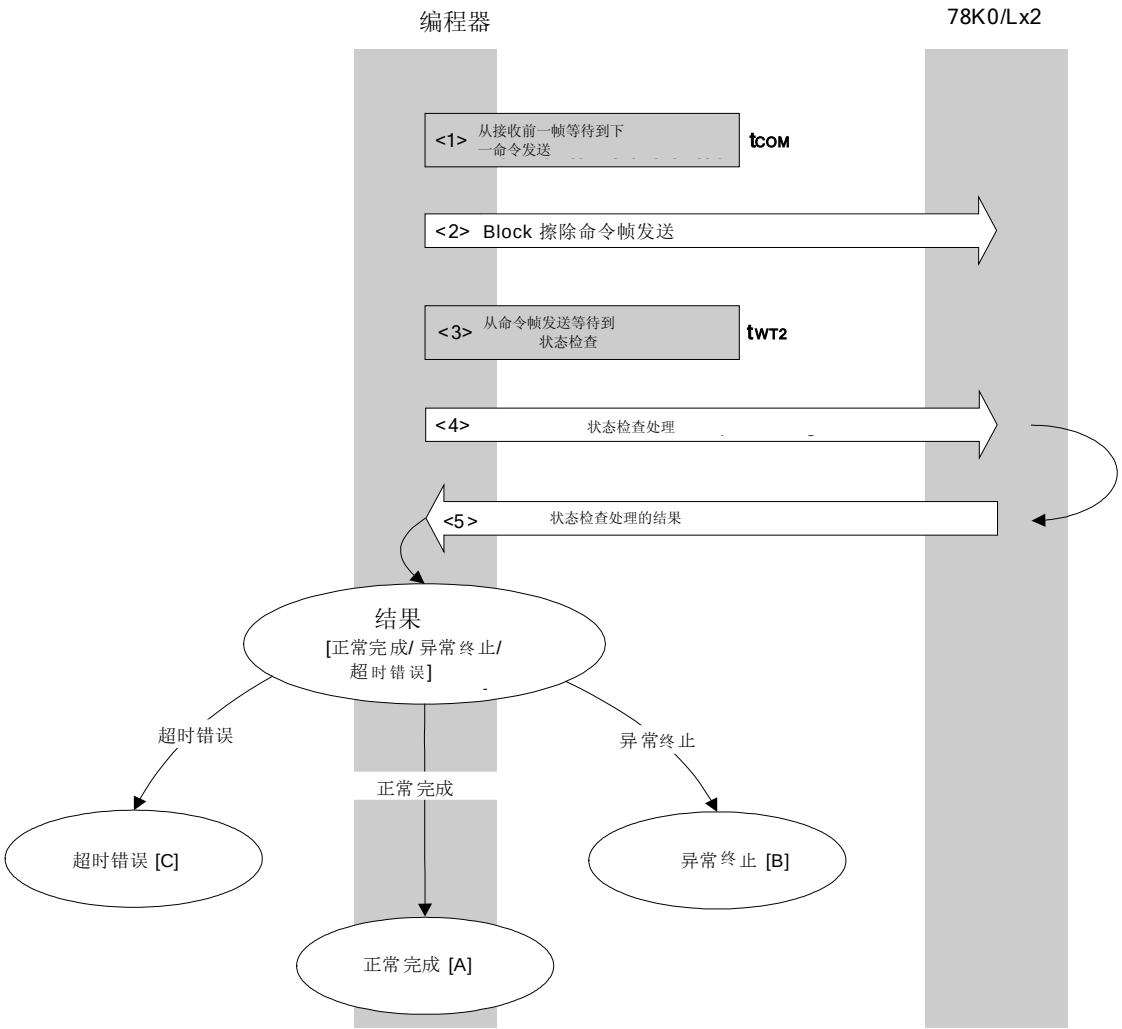
下面是芯片擦除命令处理的例程。

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


        fl_wait(tCOM);                          // 命令帧发送前的等待

        put_cmd_csi(FL_COM_ERASE_CHIP, 1, fl_cmd_prm);          //发送“芯片擦除”命令

        fl_wait(tWT1);

        rc = fl_csi_getstatus(tWT1_MAX);                //获得状态帧
//      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;

}
```

## 7.8 Block 擦除命令

### 7.8.1 处理流程图

Block 擦除命令处理流程图



### 7.8.2 处理流程的描述

<1> 从上一帧接收到下一个命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送block擦除命令。

<3> 到获得状态帧的等待 (等待时间 $t_{WT1}$)。

<4> 通过状态检测处理获得状态帧。

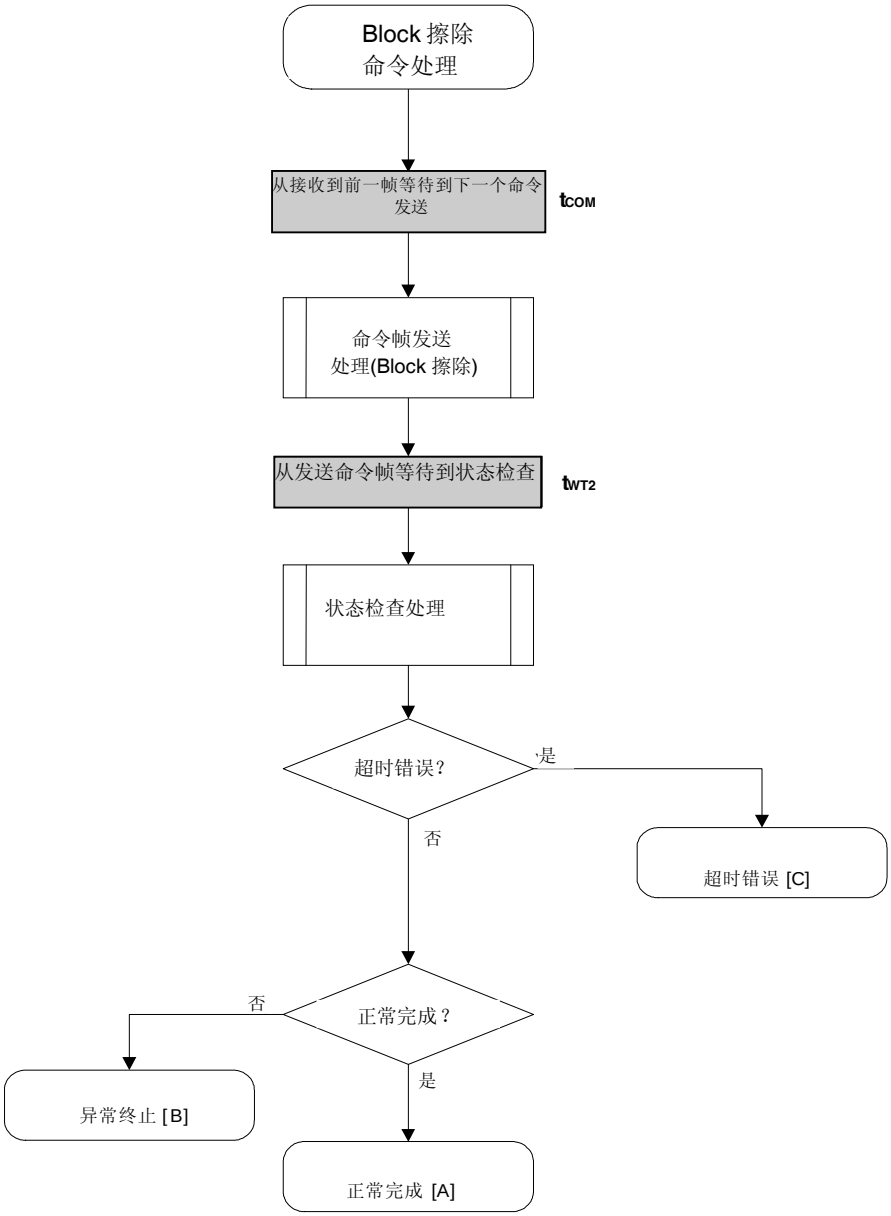
<5> 接下来的处理根据状态检测处理的结果而定。

当处理正常结束： 正常完成 [A]
当处理异常结束： 异常终止 [B]
当发生超时错误： 返回一个超时错误 [C]

### 7.8.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常应答 (ACK) | 06H | 命令正常执行，block擦除正常执行 |
| 异常终止[B] | 参数错误 | 05H | Block数超过范围 |
| | 校验和错误 | 07H | 发送命令帧中的校验和不相等 |
| | 保护错误 | 10H | 安全设置中禁止写入，block擦除或芯片擦除 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (例如数据长度不对 (LEN) 或者没有 ETX) |
| | 擦除错误 | 1AH | 发生一个擦除错误 |
| 超时错误[C] | | ~ | 在指定时间内没有接收到状态帧 |

### 7.8.4 流程图


Block 擦除
命令处理
从接收到前一帧等待到下一个命令发送
tCOM
命令帧发送
处理(Block 擦除)
从发送命令帧等待到状态检查
tWT2
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
正常完成 [A]

### 7.8.5 例子程序

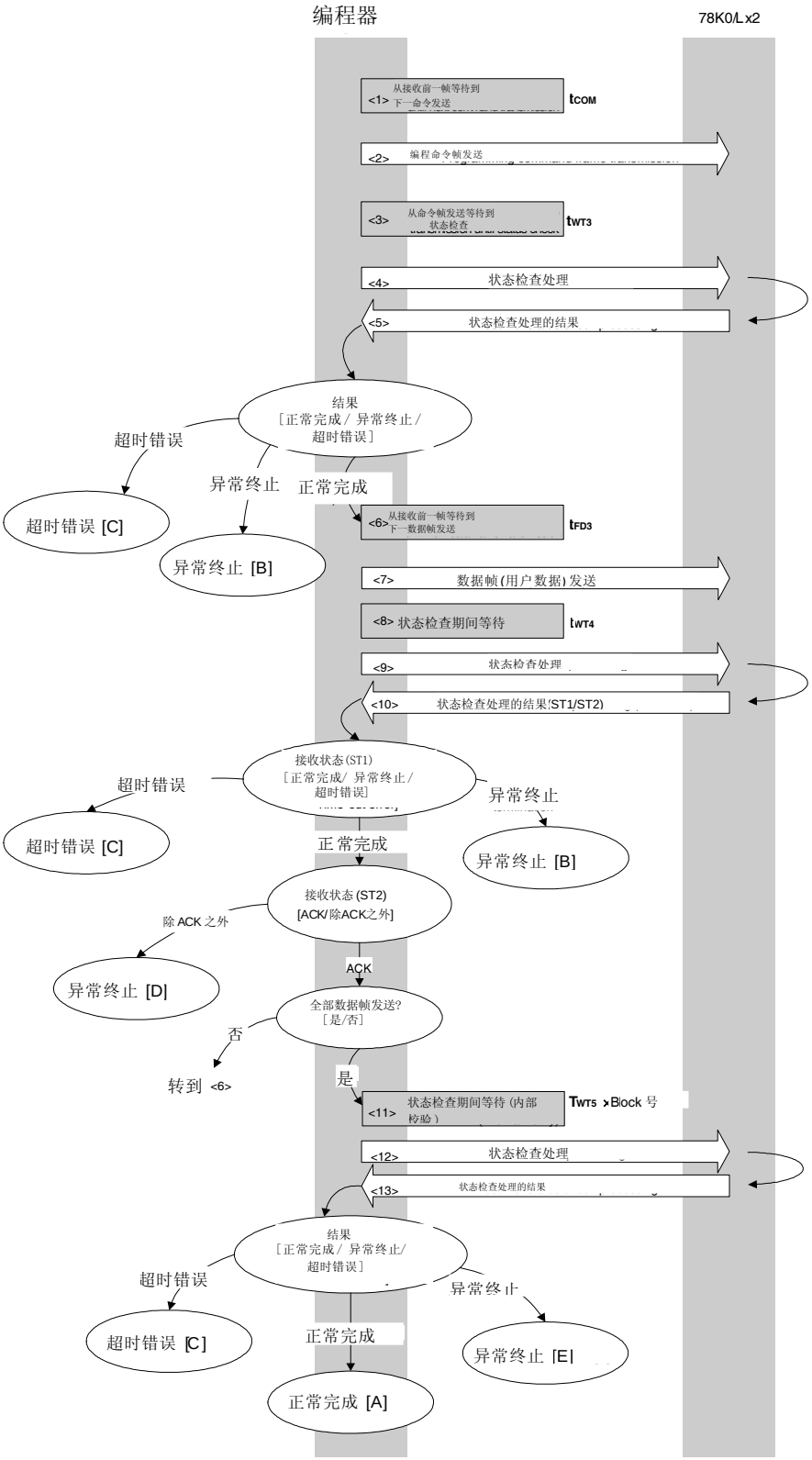
下面是block擦除命令处理的例子程序。

```
/*****************************************************************/
/*                                                               */
/* block擦除命令(CSI)                                            */
/*                                                               */
/*****************************************************************/
/* [i] u16 sblk   ... 要擦除的起始 block (0...255)        */
/* [i] u16 eblk  ...  要擦除的结束 block (0...255)          */
/* [r] u16       ... 错误码                          */
/*****************************************************************/
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 block 的起始地址
        bottom = get_bottom_addr(eblk);     //获取结束 block 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);                                  // 发送命令帧前的等待

        put_cmd_csi(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm);   // 发送“Block 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;
}
```

## 7.9 编程命令

### 7.9.1 处理流程图

编程命令处理流程图



### 7.9.2 流程描述

<1> 从上一帧接收到下一个命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送编程命令。

<3> 从命令发送到状态检测处理的等待 (等待时间$t_{WT3}$)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常完成： 继续进行<6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回超时错误 [C]

<6> 到下一个数据帧发送前的等待 (等待时间 $t_{FD3}$)。

<7> 通过数据帧发送处理把用户数据写入到 78K0/Lx2 flash存储器。

<8> 从数据帧(用户数据)发送到状态检测处理的等待 (等待时间$t_{WT4}$)。

<9> 通过状态检测处理获得状态帧。

<10> 根据状态检测处理的结果(状态码(ST1/ST2))来确定下列处理 (在处理流程图中也有涉及)。

当 ST1 = 异常终止： 异常终止 [B]

当 ST1 = 超时错误： 返回一个超时错误 [C]

当 ST1 = 正常完成： 根据ST2的值来确定下列处理

- 当 ST2 ≠ ACK: 异常终止 [D]
- 当 ST2 = ACK: 当全部用户数据发送完成进入<11>
  如果仍有用户数据要发送，处理重新从<6>开始执行

<11> 到状态检测处理的等待(超时时间 $t_{WT5}$X block数)。

<12> 通过状态检测处理来获得状态帧。

<13> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 正常完成 [A]
(完成写入之后执行内部校验检测标志)

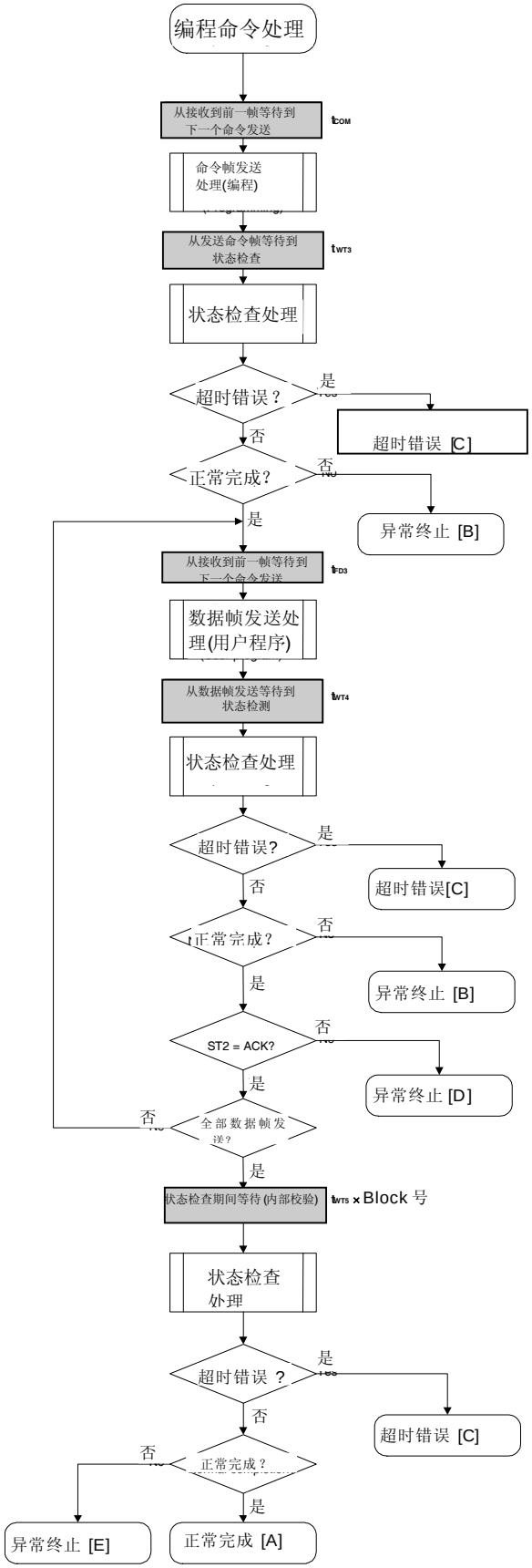
当处理异常结束： 异常终止 [E]
(完成写入之后不执行内部校验检测标志)

当发生一个超时错误： 返回一个超时错误 [C]

### 7.9.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常结束[A] | 正常应答(ACK) | 06H | 命令正常执行并且用户数据正常写入 |
| 异常终止[B] | 参数错误 | 05H | 指定的开始/结束地址超出flash存储器范围，或者不是8的倍数 |
| | 校验和错误 | 07H | 发送命令帧中的校验不相等 |
| | 保护错误 | 10H | 安全设置里设定了禁止写入 |
| | 异常应答 (NACK) | 15H | 命令帧数据异常 (例如数据长度(LEN)或者没有ETX) |
| 超时错误[C] | | ~ | 在指定时间内没有接收到状态帧 |
| 异常终止[D] | 写入错误 | 1CH (ST2) | 发生一个写入错误 |
| 异常终止[E] | MRG11错误 | 1BH | 发生一个内部校验错误 |

### 7.9.4 流程图


编程命令处理
从接收到前一帧等待到下一个命令发送
tCOM
命令帧发送处理(编程)
从发送命令帧等待到状态检查
tWT3
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从接收到前一帧等待到下一个命令发送
tFD3
数据帧发送处理(用户程序)
从数据帧发送等待到状态检测
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
全部数据帧发送?
否
是
状态检查期间等待 (内部校验)
tWT5 x Block 号
状态检查处理
超时错误 ?
是
超时错误 [C]
否
正常完成？
否
异常终止 [E]
是
正常完成 [A]

### 7.9.5 例子程序

下面是编程命令处理的例子程序

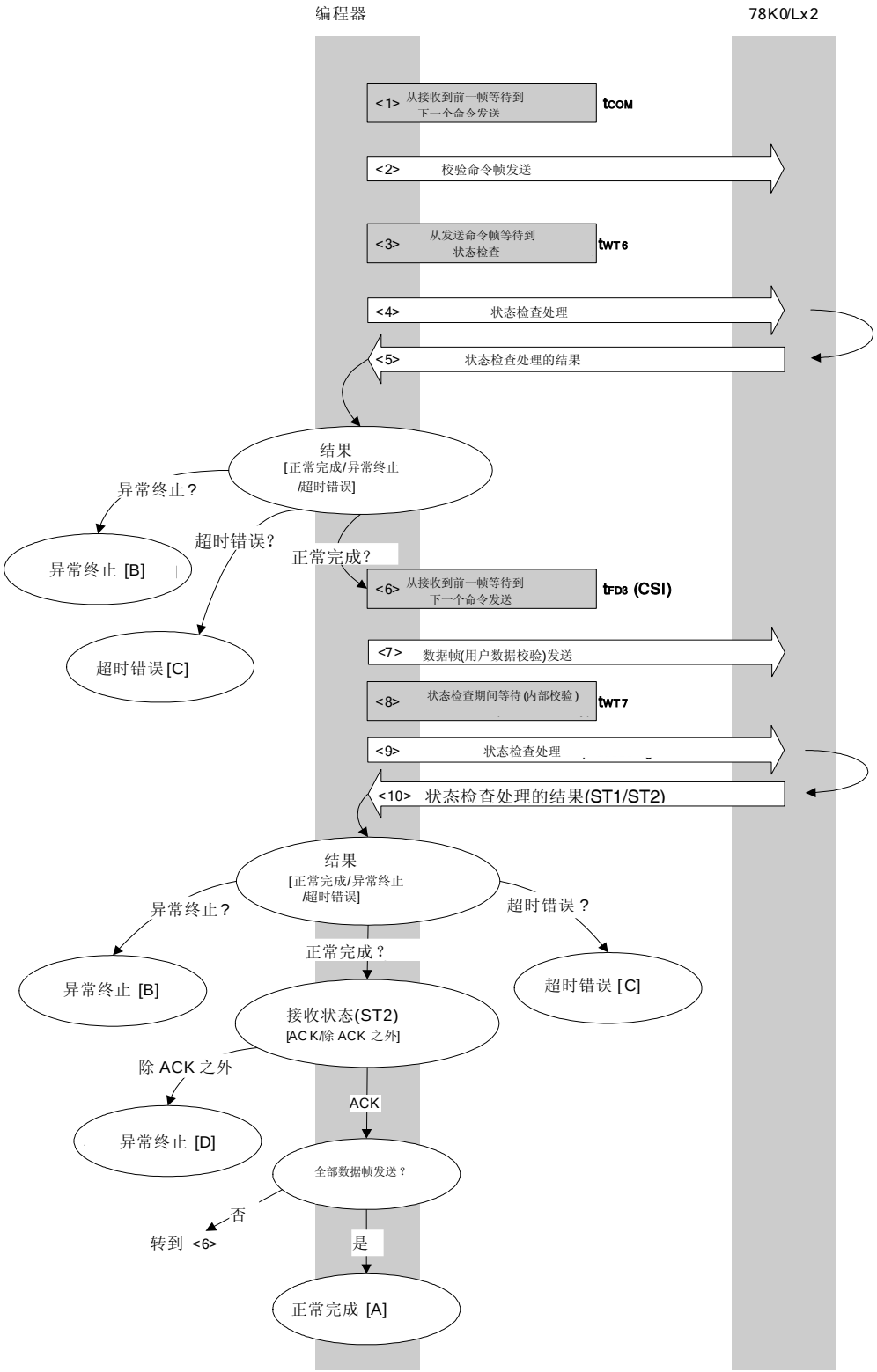
```
/*****************************************************************/
/*                                                               */
/*写入命令 (CSI)                                                 */
/*                                                               */
/*****************************************************************/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

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

        block_num = get_block_num(top, bottom);    //获得 block 数

        /*************************************************/
        /*     发送命令&检测状态                         */
        /*************************************************/

        fl_wait(tCOM);
        put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm);       //发送“编程”命令
        fl_wait(tWT3);

        rc = fl_csi_getstatus(tWT3_MAX);                //获得状态帧
        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){

                if ((bottom - send_head) > 256){      // rest size > 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);
                                                        //设置数据帧有效载荷
            send_head += send_size;


            fl_wait(tFD3_CSI);          // wait before sending data frame
            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;  // continue
            //      case    FLC_DFTO_ERR:       return  rc;     break;  // case [C]
                    default:        return  rc;     break;  // case [B]
            }
            if (fl_st2 != FLST_ACK){                    // ST2 = ACK ?
                    rc = decode_status(fl_st2);//否
                    return  rc;                         // case [D]
            }

            if (is_end)                         //发送全部用户数据?
                    break;                      //是
            //continue;
    }
    /************************************************/
    /*      检测内部校验            */
    /************************************************/

    fl_wait(tWT5 * block_num);                          //等待

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

## 7.10 校验命令

### 7.10.1 处理流程图

校验命令处理流程



### 7.10.2 流程描述

<1> 从上一帧接收到下一个命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送校验命令。

<3> 从命令发送到状态检测处理的等待 (等待时间 $t_{WT6}$)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 前往 <6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误[C]

<6> 从上一帧接收到下一数据帧发送的等待 (等待时间 $t_{FD3}$)。

<7> 通过数据帧发送处理发送用于校验的用户数据。

<8> 从数据帧发送到状态检测处理的等待 (等待时间 $t_{WT7}$)。

<9> 通过状态检测处理来获得状态帧。

<10> 根据状态检测处理的结果(状态码(ST1/ST2))来确定下列处理 (在处理流程图中也有涉及)。

当 ST1 = 异常终止： 异常终止 [B]

当 ST1 = 超时错误： 返回一个超时错误 [C]

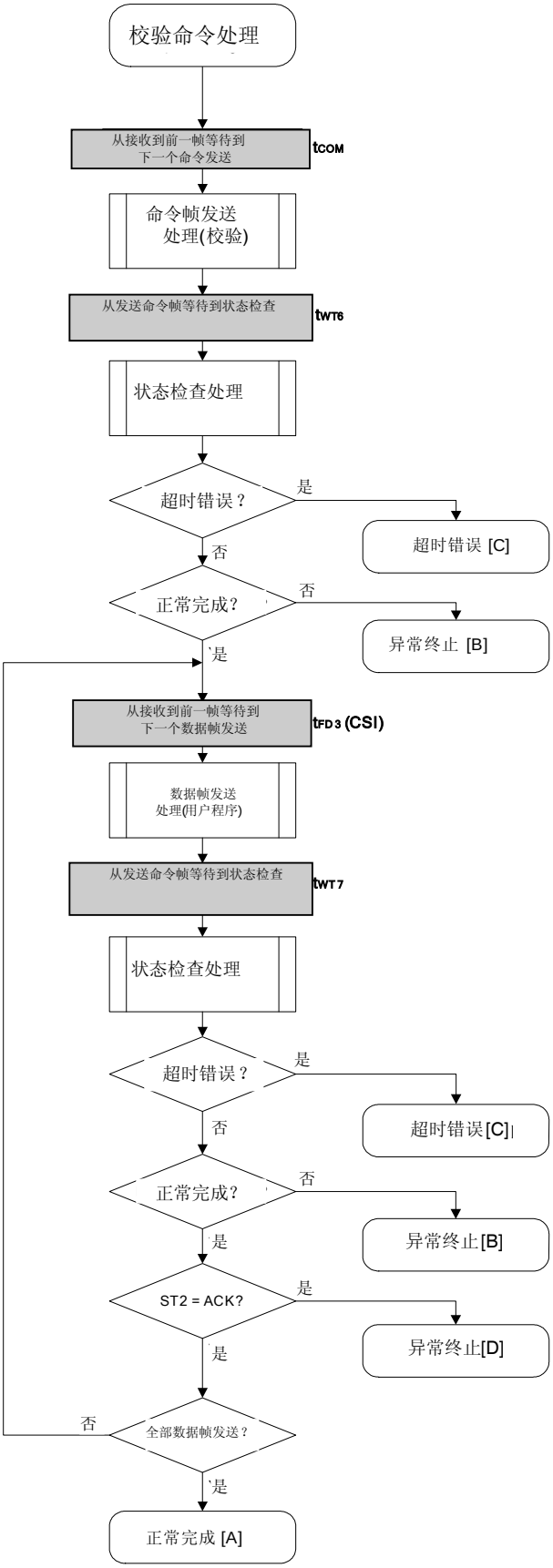
当 ST1 = 正常完成： 根据ST2的值来确定下列处理

- 当 ST2 ≠ ACK: 异常终止 [D]
- 当 ST2 = ACK: 如果全部数据帧发送完毕，处理正常结束 [A]
  如果还有数据帧需要发送，处理返回并重新执行<6>

### 7.10.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且校验正常完成。 |
| 异常终止[B] | 参数错误 | 05H | 指定的开始/结束地址超出flash存储器范围，或者指定地址不是2KB单位的一个确定地址。 |
| | 校验和错误 | 07H | 发送命令帧或者数据帧的校验和不相等。 |
| | 异常响应 (NACK) | 15H | 命令帧数据异常 (例如输出长度不正常(LEN) 或者不是 ETX)。 |
| 超时错误[C] | | - | 在指定时间内没有接收到状态帧。 |
| 异常终止[D] | 校验错误 | 0FH (ST2) | 校验失败，或者发生其他错误。 |

### 7.10.4 流程图


校验命令处理
从接收到前一帧等待到下一个命令发送
tCOM
命令帧发送处理(校验)
从发送命令帧等待到状态检查
tWT6
状态检查处理
超时错误？
是
超时错误 [C]
否
正常完成?
否
异常终止 [B]
是
从接收到前一帧等待到下一个数据帧发送
tFD3 (CSI)
数据帧发送处理(用户程序)
从发送命令帧等待到状态检查
tWT7
状态检查处理
超时错误？
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
ST2 = ACK?
是
异常终止[D]
是
否
全部数据帧发送？
是
正常完成 [A]

### 7.10.5 例子程序

下面是校验命令处理的例子程序。

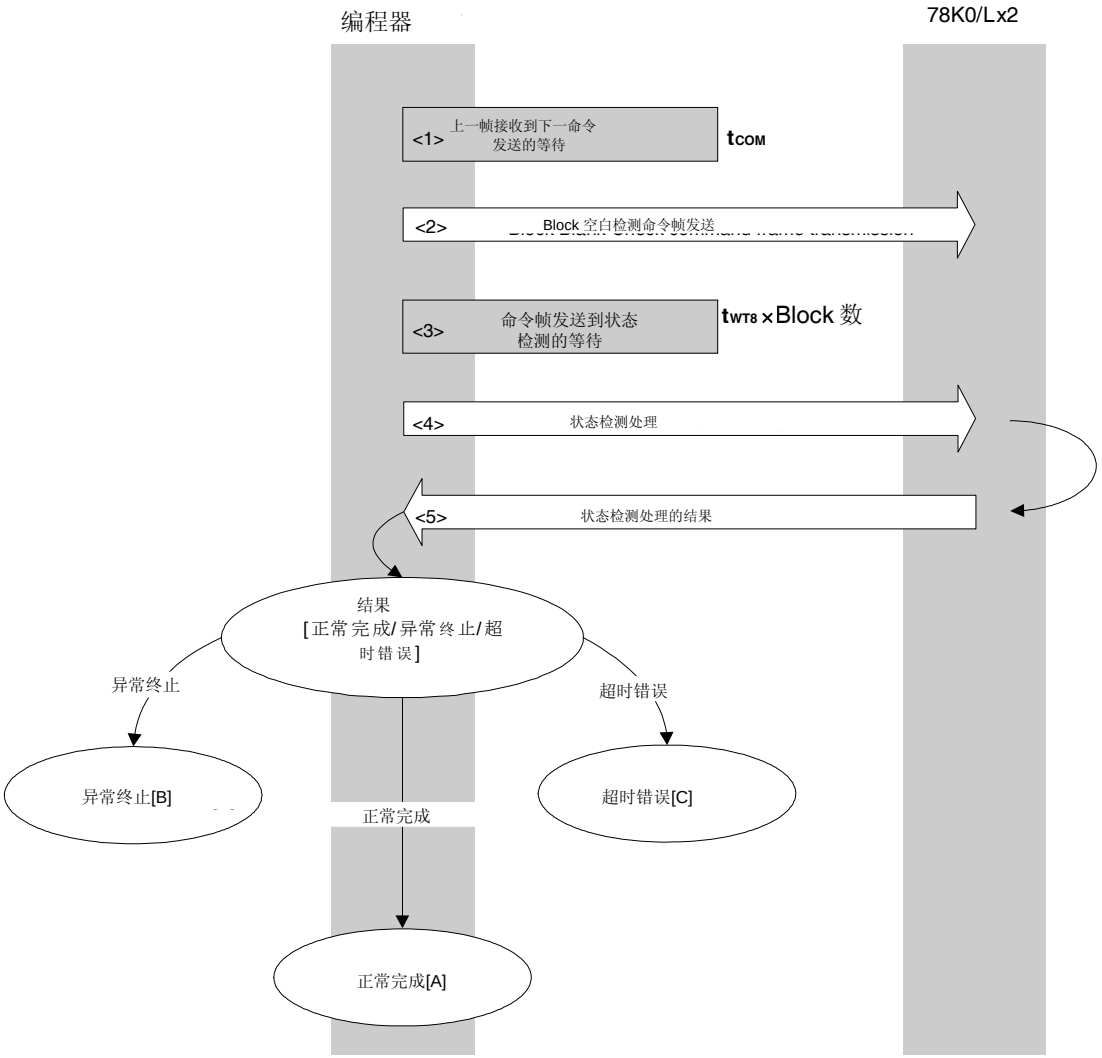
```
/*****************************************************************/
/*                                                               */
/*校验命令 (CSI)                                                 */
/*                                                               */
/*****************************************************************/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

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


        /*************************************************/
        /*      send command & check status              */
        /*************************************************/
        fl_wait(tCOM);
        put_cmd_csi(FL_COM_VERIFY, 7, fl_cmd_prm);      // 发送“校验”命令
        fl_wait(tWT6);

        rc = fl_csi_getstatus(tWT6_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]
        }


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

        while(1){

                if ((bottom - send_head) > 256){                // rest size > 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
                                                                      //帧有效载荷
            send_head += send_size;

            fl_wait(tFD3_CSI);                                  // 发送数据帧前的等待
            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;  // continue
            //  case    FLC_DFTO_ERR:       return  rc;         break;  // case [C]
                default:            return  rc;     break;  // case [B]
            }
            if (fl_st2 != FLST_ACK){            // ST2 = ACK ?
                rc = decode_status(fl_st2);//否
                return  rc;                             // case [D]
            }

            if (is_end)                         //发送全部用户数据?
                break;                          //是
            //continue;

        }
        return  FLC_NO_ERR;  // case [A]

}
```

## 7.11 Block 空白检测命令

### 7.11.1 处理流程图

Block 空白检测命令处理流程



### 7.11.2 流程描述

<1> 从上一帧接收到下一命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送Block空白检测命令。

<3> 从命令发送到状态检测处理的等待(等待时间 $t_{WT8}$ X block数)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当发生超时错误： 返回一个超时错误 [C] 当

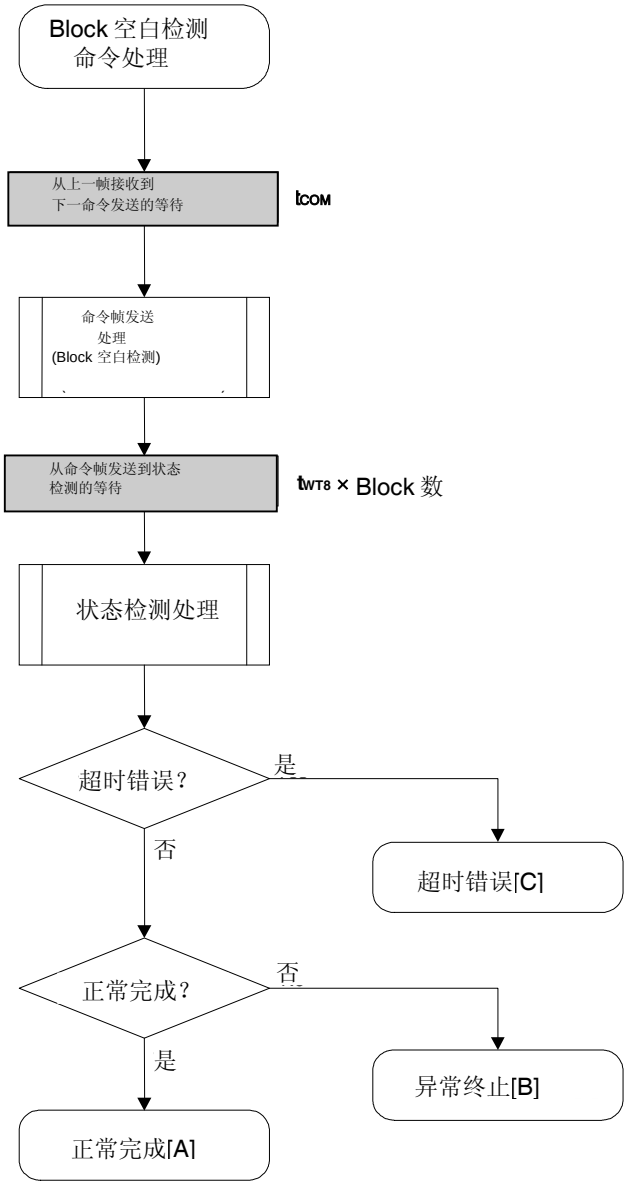
处理异常结束： 异常终止 [B]

当处理正常结束： 正常结束 [A]

### 7.11.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且全部指定block被清空 |
| 异常终止[B] | 参数错误 | 05H | Blcok数超出范围 |
| | 校验和错误 | 07H | 发送命令帧的校验和不相等 |
| | 异常响应 (NACK) | 15H | 命令帧和数据异常 (例如数据长度异常(LEN) 或者没有ETX) |
| | MRG11错误 | 1BH | Flash存储器的指定block不为空 |
| 超时错误[C] | | ~ | 指定时间内没有接收到状态帧 |

### 7.11.4 流程图


Block 空白检测
命令处理
从上一帧接收到
下一命令发送的等待
tCOM
命令帧发送
处理
(Block 空白检测)
从命令帧发送到状态
检测的等待
tWT8 × Block 数
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
正常完成[A]

### 7.11.5 例子程序

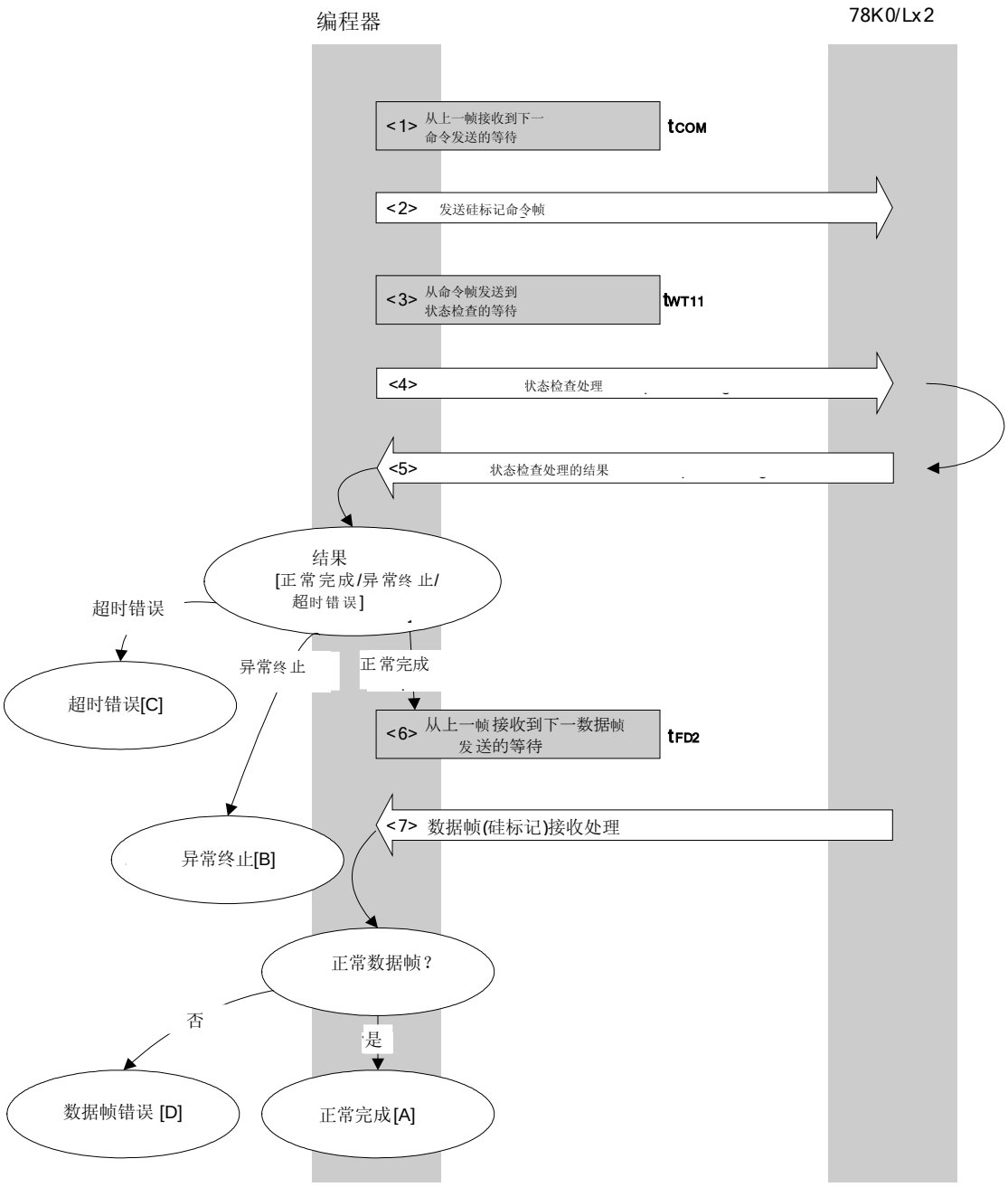
下面是一个block空白检测命令的例子程序。

```
/*****************************************************************/
/*                                                               */
/* Block 空白检测命令 (CSI)                                      */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... 起始地址                                  */
/* [i] u32 bottom  ... 结束地址                                  */
/* [r] u16         ... 错误码                                    */
/*****************************************************************/
u16             fl_csi_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 block 数

        fl_wait(tCOM);                             // 发送命令前的等待

        put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);
                                                   // 发送“Block 空白检测”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;
}
```

## 7.12 硅标记命令

### 7.12.1 处理流程图

硅标记命令处理流程



### 7.12.2 流程描述

<1> 从上一帧接收到下一命令帧发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送硅标记命令。

<3> 从命令发送到状态检测处理的等待 (等待时间 $t_{WT11}$)。

<4> 通过状态检测处理来获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 转到 <6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

<6> 从上一帧接收到下一命令帧发送的等待 (等待时间 $t_{FD2}$)。

<7> 检查接收数据帧 (硅标记数据)。

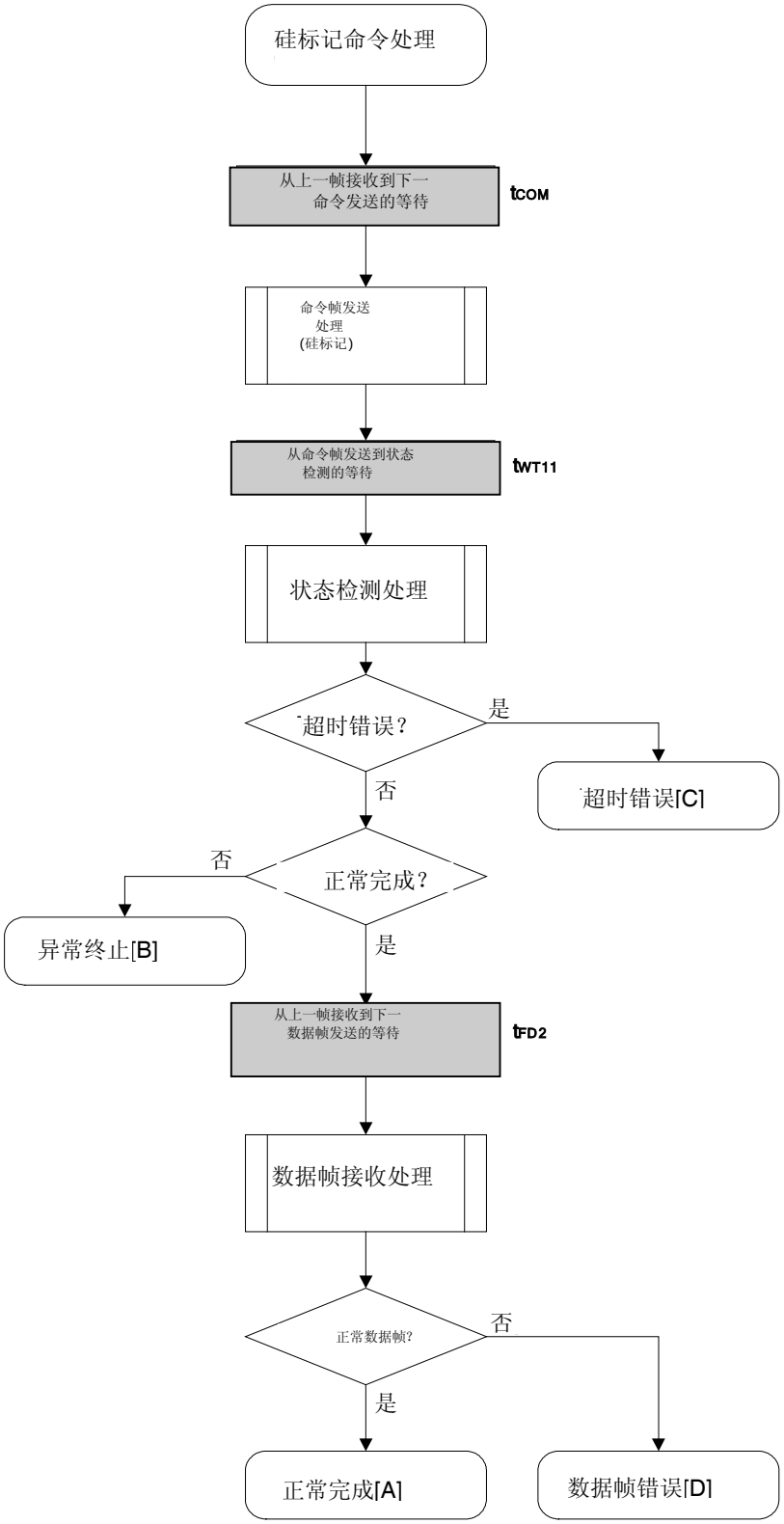
如果数据帧正常： 正常完成 [A]

如果数据帧异常： 数据帧错误 [D]

### 7.12.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且硅标记正常获得 |
| 异常中止[B] | 校验和错误 | 07H | 发送命令帧得校验和不相等 |
| | 异常响应 (NACK) | 15H | 命令帧数据异常(例如数据长度异常(LEN) 或者没有 ETX) |
| | 读错误 | 20H | 读取安全信息失败 |
| 超时错误[C] | | - | 指定时间内没有接收到状态帧 |
| 数据帧错误[D] | | - | 数据帧的校验和和硅标记数据不相等 |

### 7.12.4 流程图


硅标记命令处理
从上一帧接收到下一命令发送的等待
tCOM
命令帧发送处理(硅标记)
从命令帧发送到状态检测的等待
tWT11
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
从上一帧接收到下一数据帧发送的等待
tFD2
数据帧接收处理
正常数据帧?
否
是
正常完成[A]
数据帧错误[D]

### 7.12.5 例子程序

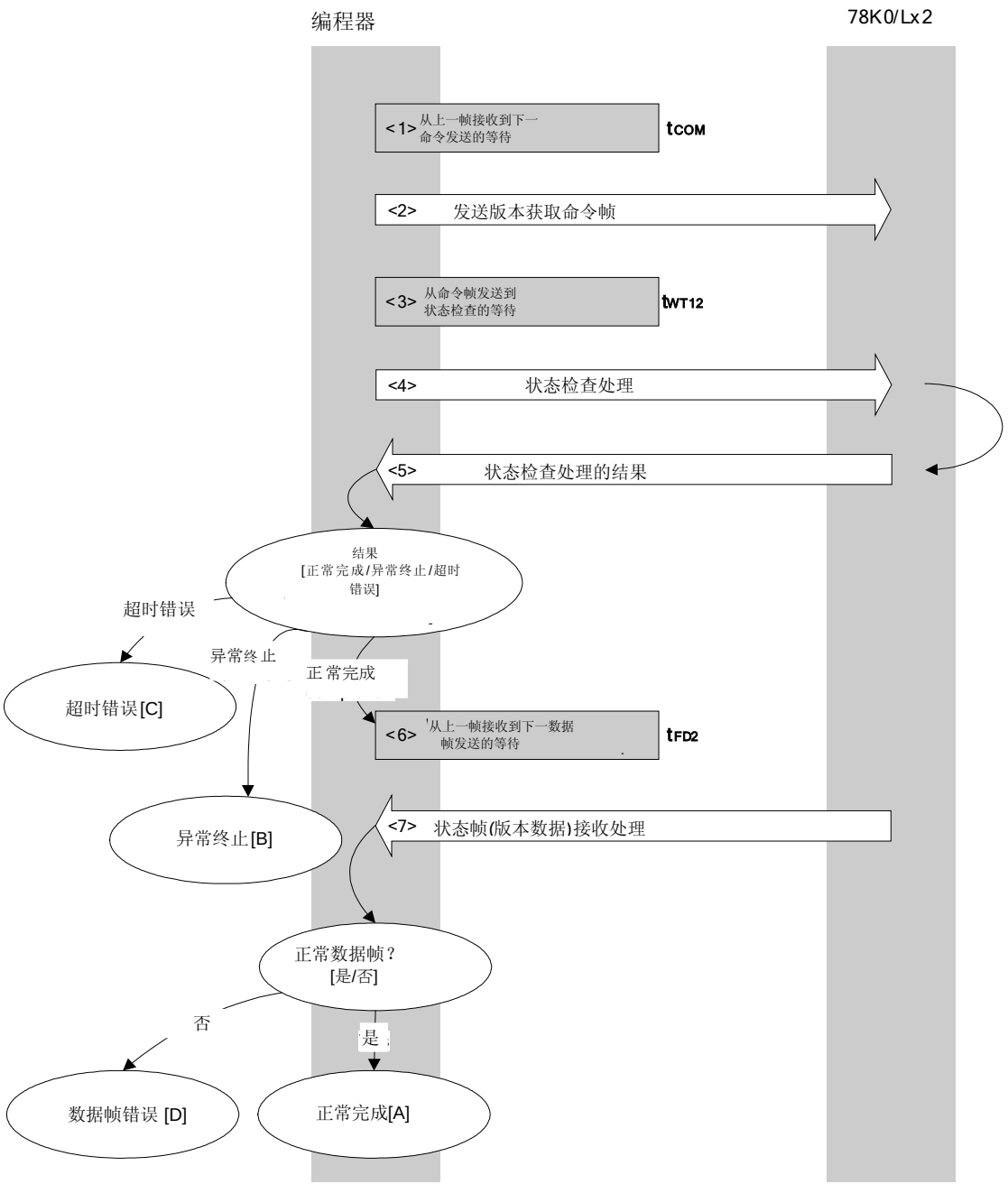
下面是一个硅标记命令处理的例子程序。

```
/*********************************************************************/
/*                                                                   */
/* 获取硅标记命令(CSI)                                               */
/*                                                                   */
/*********************************************************************/
/* [i] u8 *sig              ... 指向存储区域信号                     */
/* [r] u16                  ... 错误码                               */
/*********************************************************************/

u16         fl_csi_getsig(u8 *sig)
{
    u16     rc;

    fl_wait(tCOM);                          // 发送命令帧之前的等待

    put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);
                                                    //发送“硅标记”命令

    fl_wait(tWT11);

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

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

    rc = get_dfrm_csi(fl_rxdata_frm);   //获取数据帧(信号数据)

    if (rc){                                        //如果没错误,
        return rc;                          // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                            //复制标记数据
    return  rc;                             // case [A]
}
```

## 7.13 版本获取命令

### 7.13.1 处理流程图

版本获取命令处理流程



### 7.13.2 流程描述

<1> 从上一帧接收到下一名发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送版本获得命令。

<3> 从命令发送到状态检测处理的等待 (等待时间 $t_{WT12}$)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 转到<6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

<6> 从上一帧接收到下一命令发送的等待 (等待时间 $t_{FD2}$)。

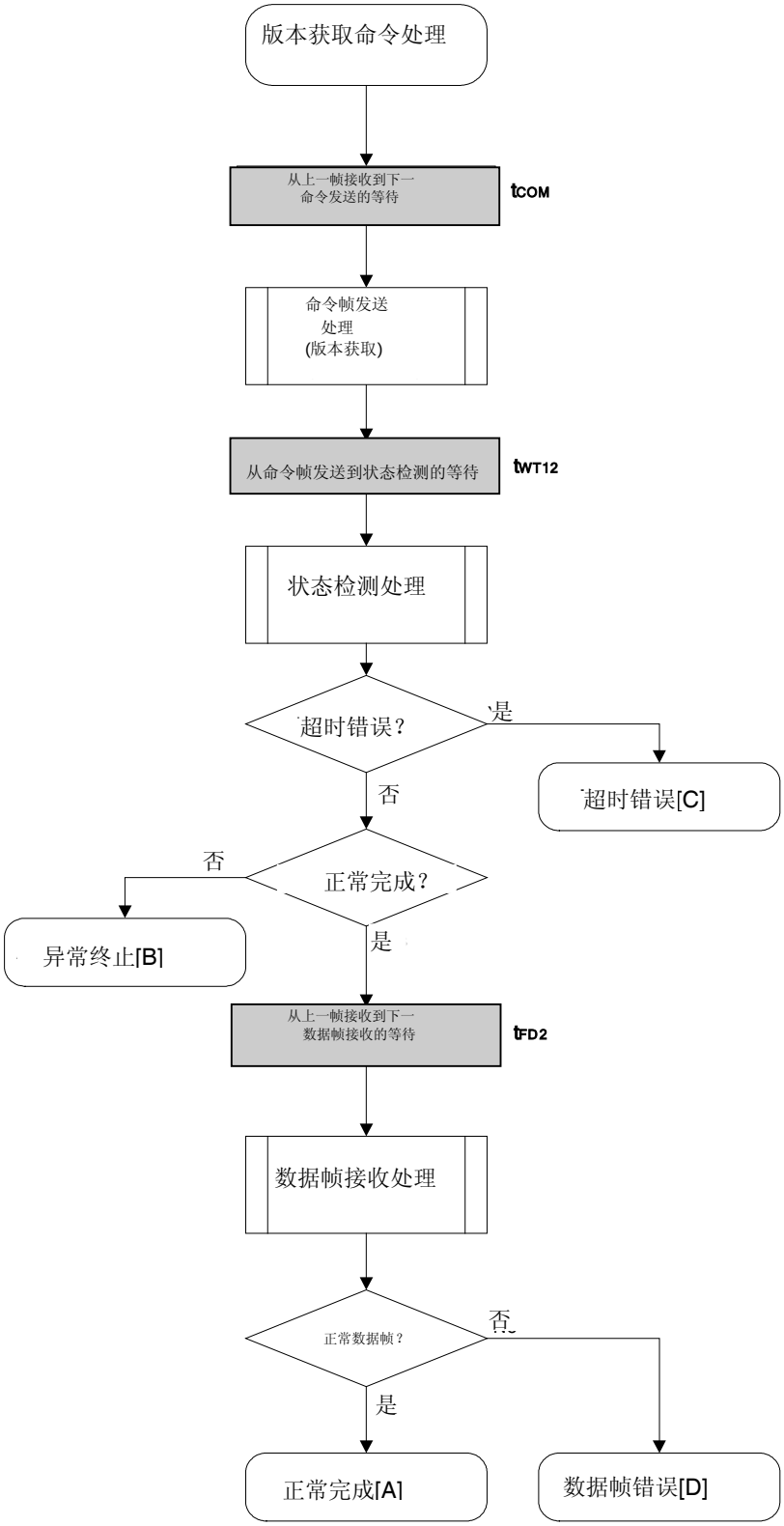
<7> 检查接收数据帧 (版本数据)。

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]

### 7.13.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且版本数据正常获取 |
| 异常结束[B] | 校验和错误 | 07H | 发送的命令帧的校验和不相等 |
| | 异常响应 (NACK) | 15H | 命令帧数据异常 (例如数据长度异常(LEN) 或者没有 ETX) |
| 超时错误[C] | | ~ | 指定时间内没有接收到状态帧 |
| 数据帧错误[D] | | ~ | 接收的数据帧的校验和不相等 |

### 7.13.4 流程图


版本获取命令处理
从上一帧接收到下一命令发送的等待
tCOM
命令帧发送处理(版本获取)
从命令帧发送到状态检测的等待
tWT12
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
从上一帧接收到下一数据帧接收的等待
tFD2
数据帧接收处理
正常数据帧?
否
是
正常完成[A]
数据帧错误[D]

### 7.13.5 例子程序

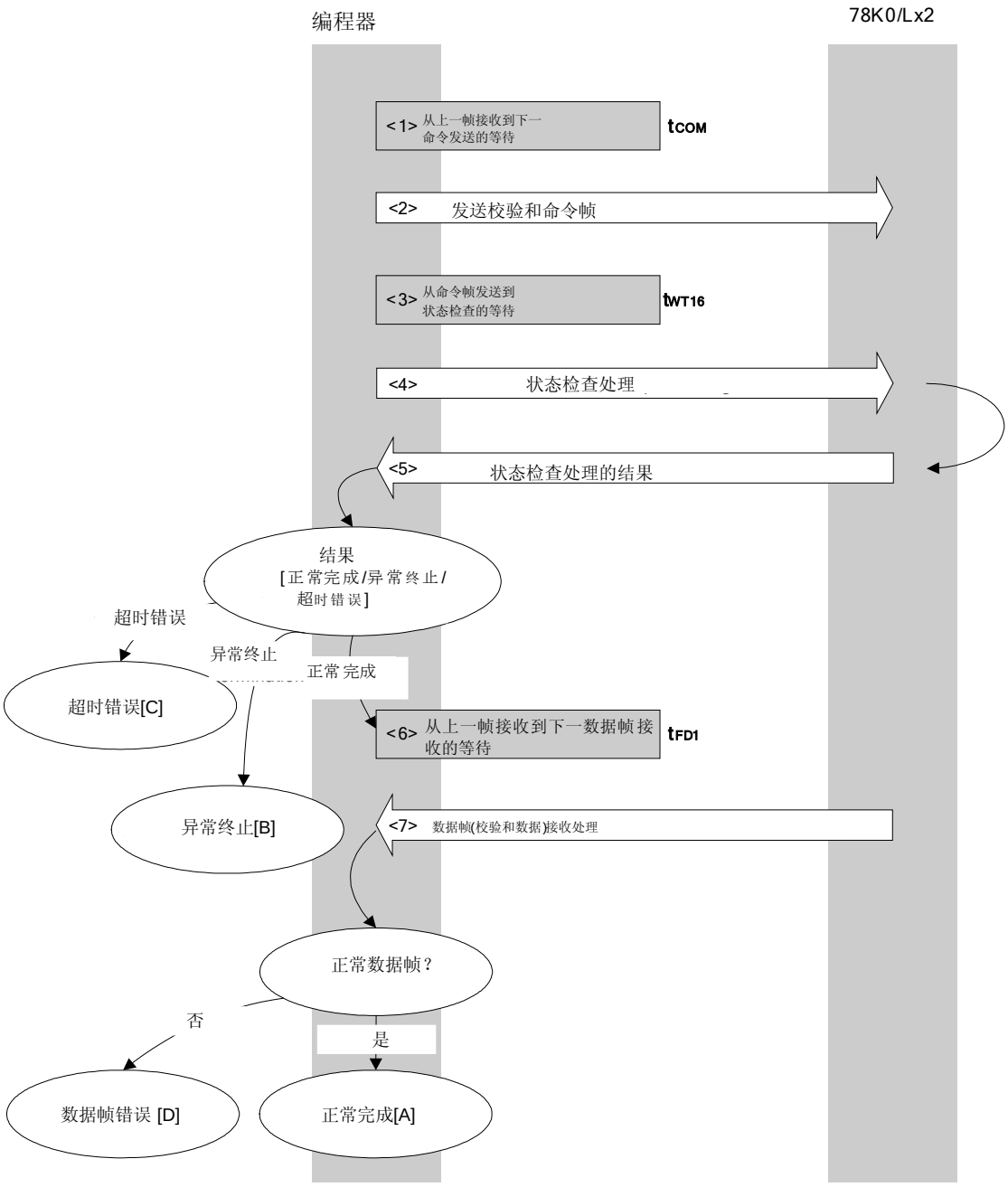
下面是一个版本获得命令处理的例子程序。

```
/*****************************************************************/
/*                                                               */
/* 获得设备/固件版本命令(CSI)                                    */
/*                                                               */
/*****************************************************************/
/* [i] u8 *buf          ... 指向版本数据存储区域                 */
/* [r] u16              ... 错误码                               */
/*****************************************************************/
u16             fl_csi_getver(u8 *buf)
{
        u16     rc;


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

        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_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]
        }

        fl_wait(tFD2_VG);               //获取状态帧前的等待

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

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

## 7.14 校验和命令

### 7.14.1 处理流程图

校验和命令处理流程



### 7.14.2 流程描述

<1> 从上一帧接收到下一命令帧发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送校验和命令。

<3> 从命令发送到状态检测处理的等待 (等待时间 $t_{WT16}$)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 转到 <6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

<6> 从上一帧接收到下一命令帧发送的等待 (等待时间 $t_{FD1}$)。

<7> 检测接收数据帧 (校验和数据)。

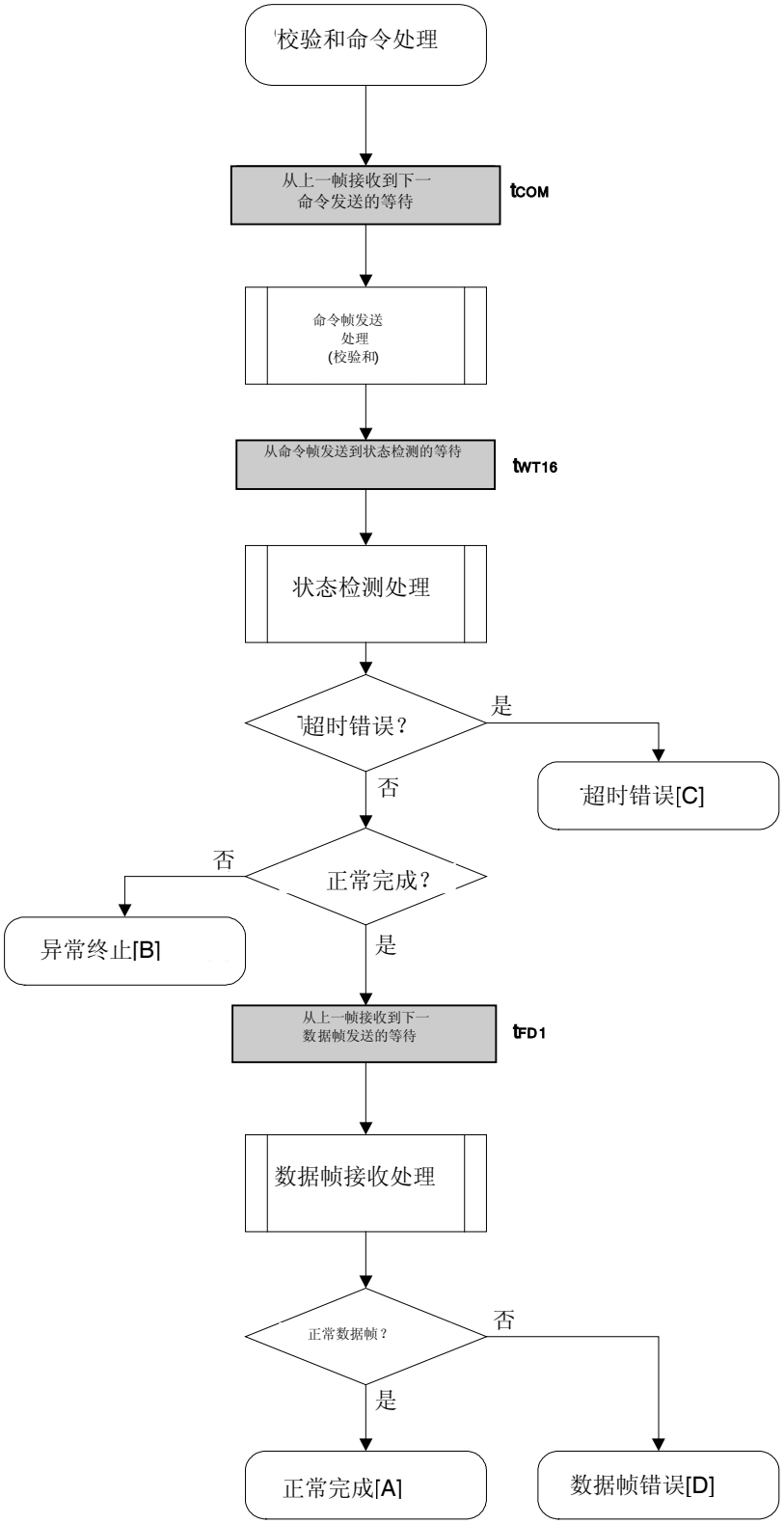
如果数据帧正常： 正常完成 [A]

如果数据帧异常： 数据帧错误 [D]

### 7.14.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且校验和数据正常获取 |
| 异常终止[B] | 参数错误 | 05H | 指定的开始/结束地址超出flash存储器范围，或者指定地址不是以2KB为单位的一个确定地址 |
| | 校验和错误 | 07H | 发送命令帧数据的校验和不相等 |
| | 异常响应 (NACK) | 15H | 命令帧数据异常 (例如数据长度异常 (LEN) 或者没有 ETX) |
| 超时错误[C] | | ˜ | 指定时间之内没有接收到状态帧 |
| 数据帧错误[D] | | ˜ | 接收数据帧校验和同版本数据不相等 |

### 7.14.4 流程图


校验和命令处理
从上一帧接收到下一
命令发送的等待
tCOM
命令帧发送
处理
(校验和)
从命令帧发送到状态检测的等待
tWT16
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
从上一帧接收到下一
数据帧发送的等待
tFD1
数据帧接收处理
正常数据帧?
否
是
正常完成[A]
数据帧错误[D]

### 7.14.5 例子程序

下面是一个校验和命令处理的例子程序。

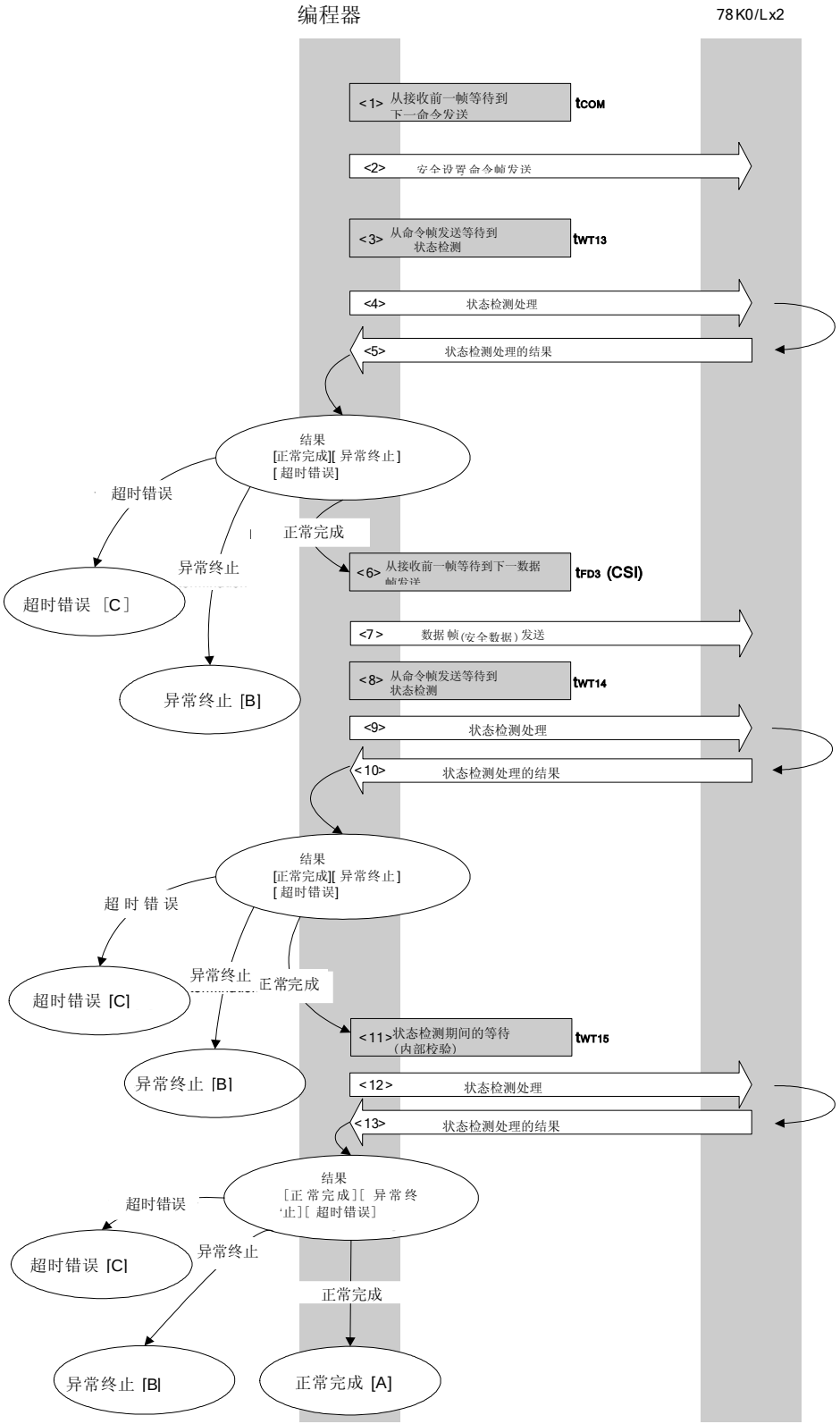
```
/*****************************************************************/
/*                                                               */
/* 获得校验和命令(CSI)                                           */
/*                                                               */
/*****************************************************************/

/* [i] u16 *sum    ... 指向校验和存储区域          */
/* [i] u32 top     ... 起始地址                    */
/* [i] u32 bottom  ... 结束地址                    */
/* [r] u16         ... 错误码                      */
/*****************************************************************/
u16             fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
        u16     rc;
        u16     block_num;

        /*************************************************/
        /*      set params                               */
        /*************************************************/
        // 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 block 数

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

        put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm);   // 发送“校验和”命令

        fl_wait(tWT16);

        rc = fl_csi_getstatus(tWT16_MAX);          //获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                        break;   // 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);                 //获取数据帧前的等待
```

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

    if (rc){                             //如果错误,
            return rc;                          // 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]
}
```

## 7.15 安全设置命令

### 7.15.1 处 理流程图

安全设置命令处理流程图



### 7.15.2 流程描述

<1> 从上一帧接收到下一命令发送的等待 (等待时间 $t_{COM}$)。

<2> 通过命令帧发送处理发送安全设置命令。

<3> 从命令发送到状态检测的等待 (等待时间 $t_{WT13}$)。

<4> 通过状态检测处理获得状态帧。

<5> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 转到<6>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

<6> 从上一帧接收到数据帧发送的等待 (等待时间 $t_{FD3}$(CSI))。

<7> 通过数据帧发送处理发送数据帧 (安全设置数据)

<8> 从数据帧发送到状态检测处理的等待 (等待时间 $t_{WT14}$)。

<9> 通过状态检测处理获得状态帧。

<10> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 转到<11>

当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

<11> 直到状态获得(完成内部校验)的等待 (等待时间 $t_{WT15}$)

<12> 通过状态检测处理获得状态帧。

<13> 根据状态检测处理的结果来确定下列处理。

当处理正常结束： 正常完成[A]

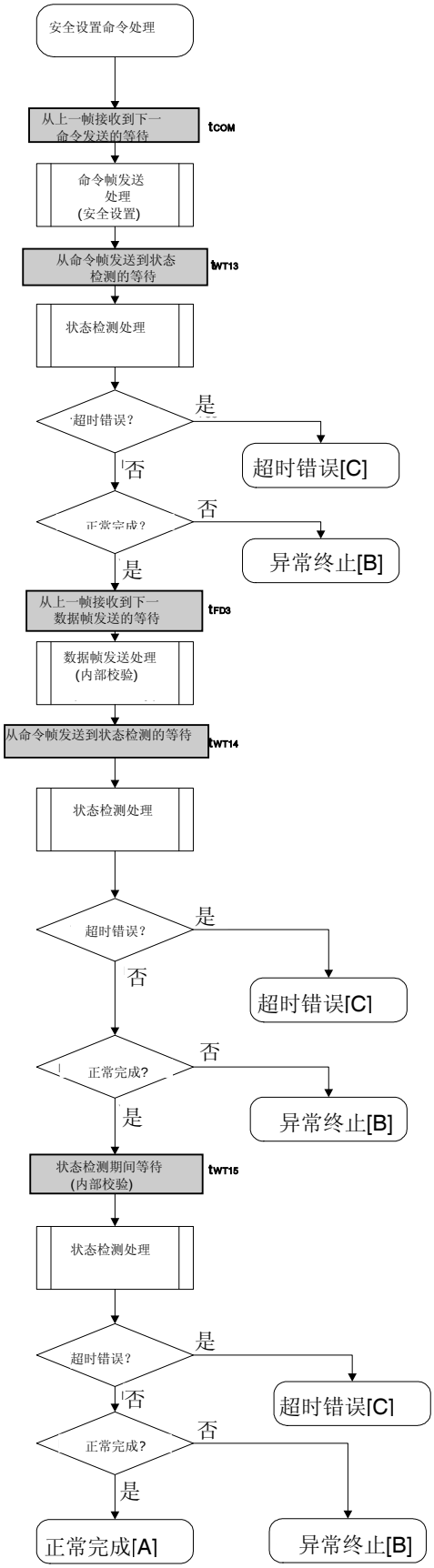
当处理异常结束： 异常终止 [B]

当发生超时错误： 返回一个超时错误 [C]

### 7.15.3 处理结束的状态

| 处理结束的状态 | | 状态码 | 描述 |
|---|---|---|---|
| 正常完成[A] | 正常响应 (ACK) | 06H | 命令正常执行并且安全设置正常执行 |
| 异常终止[B] | 参数错误 | 05H | 命令信息(参数)不是00H |
| | 校验和错误 | 07H | 发送命令帧或数据帧的校验和不相等 |
| | 写入错误 | 1CH | 安全数据已经写入，或者发生一个写入错误 |
| | 异常响应 (NACK) | 15H | 命令帧数据异常 (例如数据长度异常 (LEN) 或者没有 ETX) |
| 超时错误[C] | | - | 指定时间之内没有接收到状态帧 |

### 7.15.4 流程图


安全设置命令处理
从上一帧接收到下一命令发送的等待
tCOM
命令帧发送处理(安全设置)
从命令帧发送到状态检测的等待
tWT13
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
从上一帧接收到下一数据帧发送的等待
tFD3
数据帧发送处理(内部校验)
从命令帧发送到状态检测的等待
tWT14
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
状态检测期间等待(内部校验)
tWT15
状态检测处理
超时错误?
是
超时错误[C]
否
正常完成?
否
异常终止[B]
是
正常完成[A]

### 7.15.5 例子程序

下面是一个安全设置命令处理的例子程序。

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

        /*********************************************************/
        /*    set params                                         */
        /*********************************************************/
        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" (高 5位必须为'1' (确保))

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

        /*************************************************/
        /*     send command                              */
        /*************************************************/
        fl_wait(tCOM);                      // 发送命令帧前的等待

        put_cmd_csi(FL_COM_SET_SECURITY, 3, fl_cmd_prm);  // 发送“安全设置”命令

        fl_wait(tWT13);                     //等待

        rc = fl_csi_getstatus(tWT13_MAX);   //获取状态帧
        switch(rc) {
                case    FLC_NO_ERR:                 break;  // 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);                  //获取数据帧前的等待
```

```
        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]
        }


        /************************************************/
        /*    检测内部校验          */
        /************************************************/
        fl_wait(tWT15);

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

# 第八章 FLASH 存储器编程参数特性

这一章描述了在Flash存储器编程模式下编程器和78K0/Lx2间的参数特性。

当您设计一个编程器时，请务必参看 78K0/Lx2 用户手册的电气特性。

## 8.1 基本特性

| 参数 | 条件 | 符号 | 最小 | 典型 | 最大 | 单位 |
|---|---|---|---|---|---|---|
| Flash存储器编程模式下的78K0/Lx2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 | |

## 8.2 Flash 存储器编程模式设置时间

| 参数 | | 符号 | 最小 | 典型 | 最大 |
|---|---|---|---|---|---|
| $V_{DD}$ ~FLMD0 | | $t_{DP}$ | 1 ms | | |
| FLMD0 ~RESET | | $t_{PR}$ | 2 ms | | |
| RESET ~ FLMD0[注1]的计数开始时间 | | $t_{RP}$ | 59,327/$f_{RH}$ | | |
| RESET ~FLMD0[注1]的计数结束时间 | | $t_{RPE}$ | | | 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) | | $t_{RC}$ | 444,463/$f_{RH}$ | | 3 s |
| 等待低电平数据1 (UART) | X1 时钟 | $t_{R1}$ | 444,463/$f_{RH}$ + $2^{16}$/$f_X$ | | 等待低电平数据1 (UART) |
| | **外部主系统时钟** | | **444,463/$f_{RH}$** | | |
| 等待低电平数据2 (UART) | | $t_{I2}$ | 15,000/$f_{RH}$ | | 3 s |
| 等待读命令(UART) | | $t_{2C}$ | 15,000/$f_{RH}$ | | 3 s |
| 低电平数据1/2的宽度[注2] | | $t_{L1}$, $t_{L2}$ | | **注 2** | |
| FLMD0计数器上升/下降时间 | | ~ | | | 1 µs |

**注意 1.** (59,327/$f_{RH}$ + 238,414/$f_{RH}$)/2推荐作为FLMD0脉冲输入时序的标准值。

**2.** 低电平的宽度和在9600bps下数据00H是一样的，这里描述的宽度是数据宽度的一半。

**备注 1.** 在$f_{RH}$ = 8 MHz条件下计算参数

**2.** 等待按照下面定义

<$t_{R1}$ (最小.)>

UART的波特率通过外部时钟产生。

本规范允许使用的输入脉冲和使用的外部时钟振荡稳定时间。

## 8.3 编程特性

| 等待 | 条件 | 符号 | 串行I/F | 最小 | 最大 |
|---|---|---|---|---|---|
| 发送/接收数据帧之间 | 接收数据帧 | $t_{DR}$ | CSI | 64/$f_{RH}$ | 3 s |
| | | | **UART** | **74/$f_{RH}$** | **3 s** |
| | **发送数据帧** | **$t_{DT}$** | **CSI** | **88/$f_{RH}$** | **3 s** |
| | | | **UART** | **0** 注 1 | **3 s** |
| 从状态命令帧到状态发送帧 | ~ | $t_{SF}$ | CSI | 166/$f_{RH}$ | 3 s |
| 从状态发送帧到数据帧发送(1) | ~ | $t_{FD1}$ 注 2 | CSI | 54,368/$f_{RH}$ | 3 s |
| | | | **UART** | **0** 注 1 | **3 s** |
| 从状态发送帧到数据帧发送(2) | 硅标记数据 | $t_{FD2}$ | CSI | 321/$f_{RH}$ | 3 s |
| | **板本数据** | | | **136/$f_{RH}$** | **3 s** |
| | ~ | | **UART** | **0** 注 1 | **3 s** |
| 从状态发送帧到数据接收帧 | ~ | $t_{FD3}$ | CSI | 163/$f_{RH}$ | 3 s |
| | | | **UART** | **101/$f_{RH}$** | **3 s** |
| 从状态发送帧到命令接收帧 | ~ | $t_{COM}$ | CSI | 64/$f_{RH}$ | 3 s |
| | | | **UART** | **71/$f_{RH}$** | **3 s** |

**注** **1.** 当编程器可以成功接收时。

**2.** 每个block发送的时间

**备注** **1.** 在$f_{RH}$ = 8 MHz条件下计算参数

**2.** 等待按照下面定义

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

78K0/Lx2在完成上次通讯后的最小时间内准备好下次通讯。

编程器必须在完成上次通讯后的最小和最大时间之间发送下次数据。

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

78K0/Lx2在完成上次通讯后的最小时间内准备好下次通讯。

编程器必须在完成上次通讯后的最小和最大时间之间发送下次数据。

| 命令 | 符号 | 串行 I/F | 最小 | 最大 |
|---|---|---|---|---|
| 复位 | $t_{WT0}$ | CSI | 172/$f_{RH}$ | 3 s |
| | | **UART** | 注 1 | **3 s** |
| 芯片擦除 | $t_{WT1}$ | ~ | 857,883/$f_{RH}$ + 88,320 X block总数/$f_{RH}$ | 186,444,400/$f_{RH}$ + 22,609,920 X block总数/$f_{RH}$ |
| Block擦除 | $t_{WT2}$ 注 2 | ~ | 214,714/$f_{RH}$ X执行同时选择和擦除的数量+ 44,160/$f_{RH}$ X需要擦除的block数量 | 54,582,372/$f_{RH}$ X执行同时选择和擦除的数量+ 11,304,960/$f_{RH}$ X需要擦除的block数量 |
| 编程 | $t_{WT3}$ | CSI | 1,348/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| | **$t_{WT4}$** 注 3 | ~ | **68,118/$f_{RH}$** | **397,587/$f_{RH}$** |
| | 注 4 **$t_{WT5}$** | **CSI** | **100,407/$f_{RH}$** | **132,144,427/$f_{RH}$** |
| | | **UART** | 注1 | **132,144,427/$f_{RH}$** |
| 校验 | $t_{WT6}$ | CSI | 686/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| | 注 3 **$t_{WT7}$** | **CSI** | **12,827/$f_{RH}$** | **3 s** |
| | | **UART** | 注1 | **3 s** |
| Block空白检测 | $t_{WT8}$ 注 4 | CSI | 45,835/$f_{RH}$ | 55,004/$f_{RH}$ |
| | | **UART** | 注1 | **55,004/$f_{RH}$** |
| 振荡频率设置 | $t_{WT9}$ | CSI | 1,127/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| 硅标记 | $t_{WT11}$ | CSI | 1,233/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| 版本获得 | $t_{WT12}$ | CSI | 242/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| 安全设置 | $t_{WT13}$ | CSI | 923/$f_{RH}$ | 3 s |
| | | **UART** | 注1 | **3 s** |
| | **$t_{WT14}$** | ~ | **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/$f_{RH}$ | 3 s |
| | | **UART** | 注 1 | **3 s** |

**注** **1.** 必须在命令发送之前设置编程器允许接收。

**2.** 参见下面页中的关于执行同时选择和擦除的数量的计算方法补充。

**3.** 256字节数据发送的时间

**4.** 一个block发送的时间

**备注** **1.** 在 $f_{RH}$ = 8 MHz 条件下计算参数

**2.** 等待按照下面定义

<$t_{WT0}$ ~ $t_{WT16}$>78K0/Lx2命令在最大和最小时间之内完成处理。编程器必须在最大时间过后重复进行状态检测。

**补充**　　通过block擦除命令同时完成执行选择和擦除。

78K0/Lx2的block擦除命令重复执行“同时选择和擦除”，可同时擦除多个block。
所以block擦除命令的等待时间等于“同时选择和擦除”的总时间。

要计算“同时选择和擦除总的执行时间”，首先要计算同时选择和擦除的执行数(M)。

通过获得需要同时擦除的block的数量(同时选择和擦除的block的数量)来计算“M”。

下面描述了计算同时选择和擦除的block的数量和执行次数(M)的方法。

**(1) 计算同时选择和擦除的block的数量**

同时选择和擦除的block的数量应当是1，2，4，8，16，32，64或128，由下列条件决定。

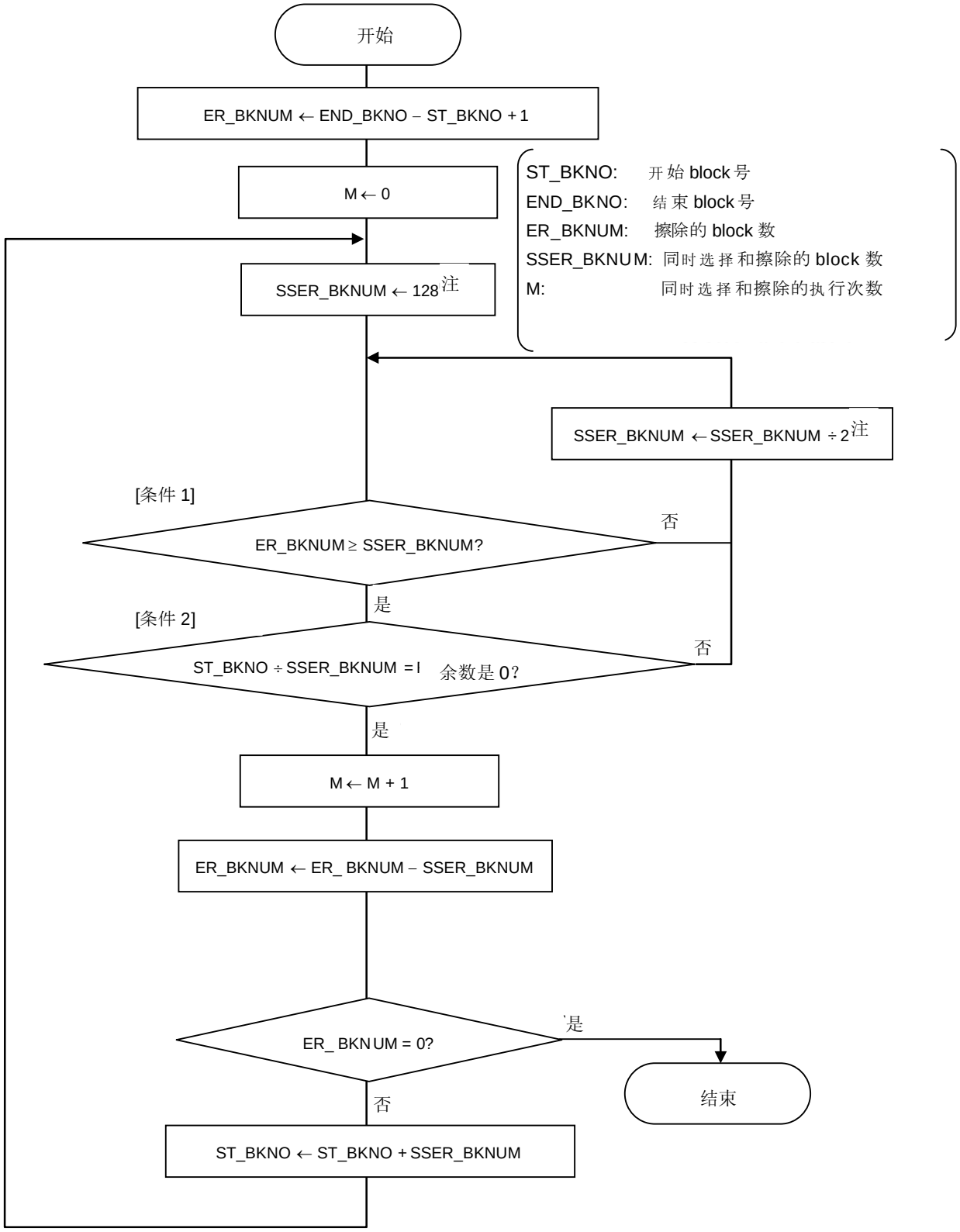
[条件 1]
(擦除block的数量)　　(同时选择和擦除的block的数量)

[条件 2]
(开始block数量)　　(同时选择和擦除的block的数量) = 余数为0

[条件 3]
最大值满足条件1和2中的值

**(2) 计算同时选择和擦除的执行次数(M)**

按照下面流程图计算执行次数 (M)




**注** 根据 SSER_BKNUM (128)的最大值，通过执行 SSER_BKNUM 2 获得满足条件 1 和 2 的值，条件 3 也满足。

**举例 1** 擦除block 1 ～ 127 (N (要擦除的block数) = 127)

<1> 首先开始的block数是1，需要擦除的数是127；因此满足条件1的值是1，2，4，8，16，32，64和128。

此外，满足条件2的值是1，满足条件3的值是1，所以同时选择和擦除的block是1；只有block 1擦除。

<2> block 1擦除后，接下来开始block数是2，需要擦除的数是126；因此满足条件1的值是1，2，4，8，16，32和64。

此外，满足条件2的值是1和2，满足条件3的值是2，所以同时选择和擦除得block是2；block 2和3擦除。

<3> block 2和3擦除后，接下来开始block数是4，需要擦除的数是124；因此满足条件1的值是1，2，4，8，16，32和64。

此外，满足条件2的值是1，2和4，满足条件3的值是4，所以同时选择和擦除得block是4；block 4～7擦除。

<4> block 4～7擦除后，接下来开始block数是8，需要擦除的数是120；因此满足条件1的值是1，2，4，8，16，32和64。

此外，满足条件2的值是1，2，4和8，满足条件3的值是8，所以同时选择和擦除得block是8；block 8～15擦除。

<5> block 8～15擦除后，接下来开始block数是16，需要擦除的数是112；因此满足条件1的值是1，2，4，8，16，32和64。

此外，满足条件2的值是1，2，4，8和16，满足条件3的值是16，所以同时选择和擦除得block是16；block 16～31擦除。

block 16～31擦除后，接下来开始block数是32，需要擦除的数是96；因此满足条件1的值是1，2，4，8，16，32和64。

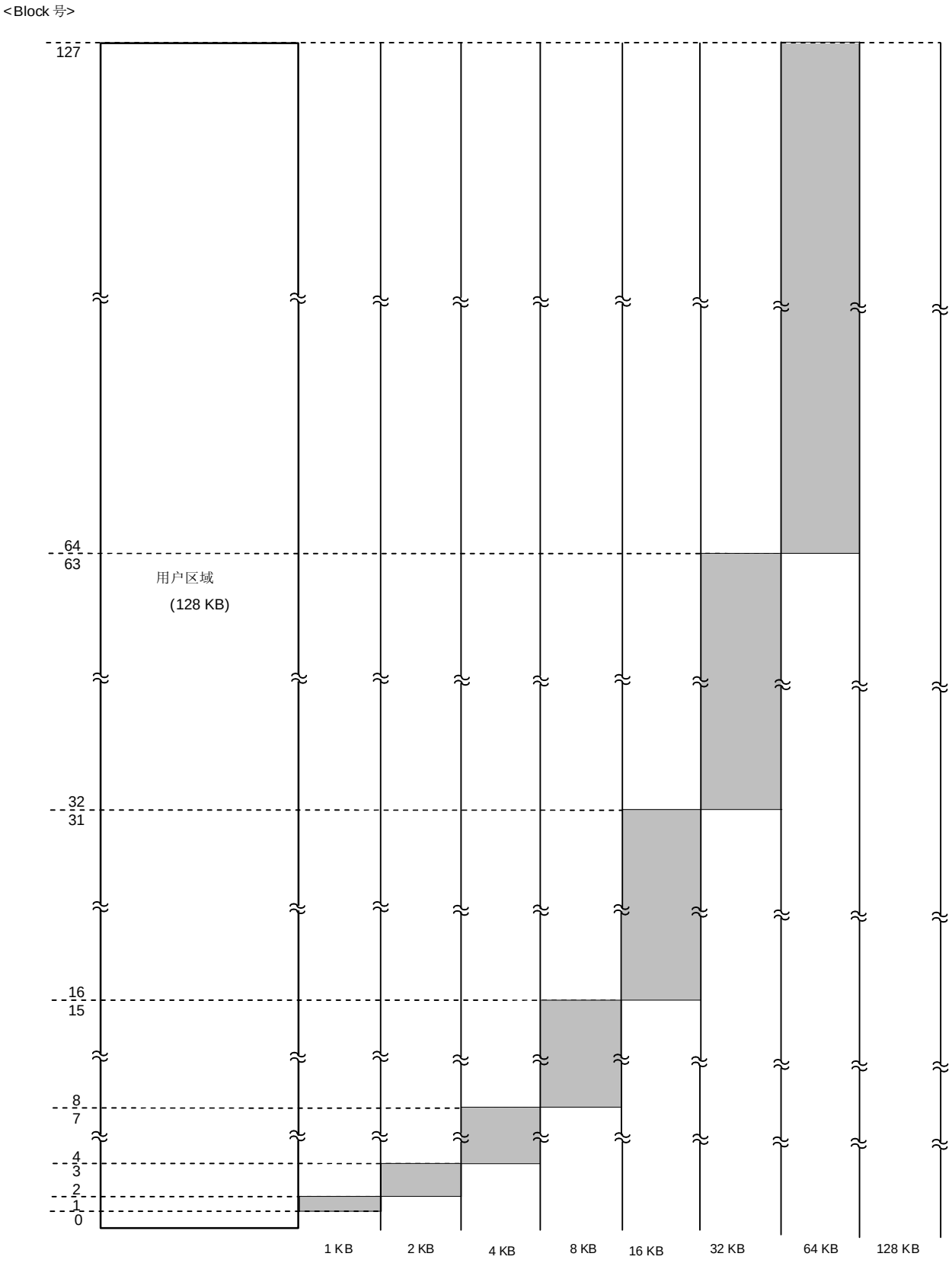
此外，满足条件2的值是1，2，4，8，16和32，满足条件3的值是32，所以同时选择和擦除得block是32；block 32～63擦除。

<6> block 32～63擦除后，接下来开始block数是64，需要擦除的数是64；因此满足条件1的值是1，2，4，8，16，32和64。

此外，满足条件2的值是1，2，4，8，16，32和64，满足条件3的值是64，所以同时选择和擦除得block是64；block 64～127擦除。

因此，同时选择和擦除执行 7 次 (1, 2 和 3, 4 to 7, 8 ～ 15, 16 ～ 31, 32 ～ 63, 和 64 ～ 127) 擦除了block 1 ～ 127，所以获得 M = 7。

执行同时选择和擦除的 block 配置 (当擦除 block 1 ～ 127)




<可以同时选择和擦除的 block 范围>

**举例2** 擦除block 5 ～ 10 (N (要擦除的block号) = 6)

<1> 首先开始的block数是5，需要擦除的数是6；因此满足条件1的值是1，2和4。

此外，满足条件2的值是1，满足条件3的值是1，所以同时选择和擦除的block是1；只有block 5擦除。

<2> block 5擦除后，接下来开始block数是6，需要擦除的数是5；因此满足条件1的值是1，2和4。

此外，满足条件2的值是1和2，满足条件3的值是2，所以同时选择和擦除得block是2；block 6和7擦除。

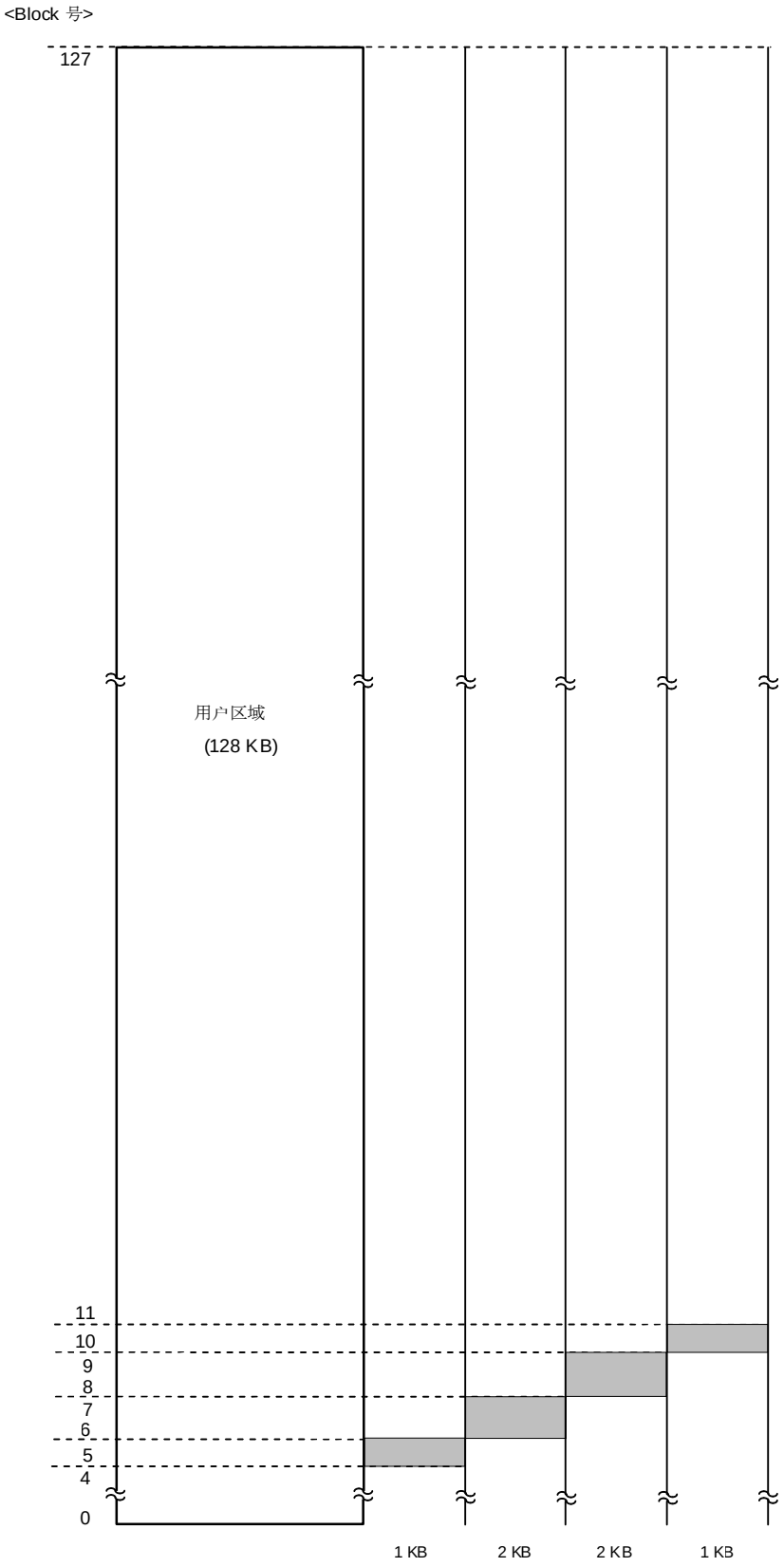
<3> block 6和7擦除后，接下来开始block数是8，需要擦除的数是3；因此满足条件1的值是1和2。

此外，满足条件2的值是1和2，满足条件3的值是2，所以同时选择和擦除得block是2；block 8和9擦除。

<4> block 8和9擦除后，接下来开始block数是10，需要擦除的数是1；因此满足条件1的值是1。这也同时满足条件2和3，所以同时选择和擦除得block是1；block 10被擦除。

因此，同时选择和擦除执行 4 次 (5, 6 和 7, 8 和 9, 和 10) 擦除了 block 5 ～ 10，所以获得 M = 4。

执行同时选择和擦除的 block 配置 (当擦除 block 5 ～ 10)




<可以同时选择和擦除的 block 范围>

**举例3** 擦除block 25 ～ 73 (N (要擦除的block数) = 49)

<1> 首先开始的block数是25，需要擦除的数是49；因此满足条件1的值是1，2，4，8，16和32。

此外，满足条件2的值是1，满足条件3的值是1，所以同时选择和擦除的block是1；只有block 25被擦除。

<2> block 25擦除后，接下来开始block数是26，需要擦除的数是48；因此满足条件1的值是1，2，4，8，16和32。

此外，满足条件2的值是1和2，满足条件3的值是2，所以同时选择和擦除的block是2；block 26和27擦除。

<3> block 26和27擦除后，接下来开始block数是28，需要擦除的数是46；因此满足条件1的值是1，2，4，8，16和32。

此外，满足条件2的值是1，2和4，满足条件3的值是4，所以同时选择和擦除的block是4；block 28 ～ 31擦除。

<4> block 28 ～ 31擦除后，接下来开始block数是32，需要擦除的数是42；因此满足条件1的值是1，2，4，8，16和32。

此外，满足条件2的值是1，2，4，8，16和32，满足条件3的值是32，所以同时选择和擦除的block是32；block 32 ～ 63擦除。

<5> block 32 ～ 63擦除后，接下来开始block数是64，需要擦除的数是10；因此满足条件1的值是1，2，4和8。

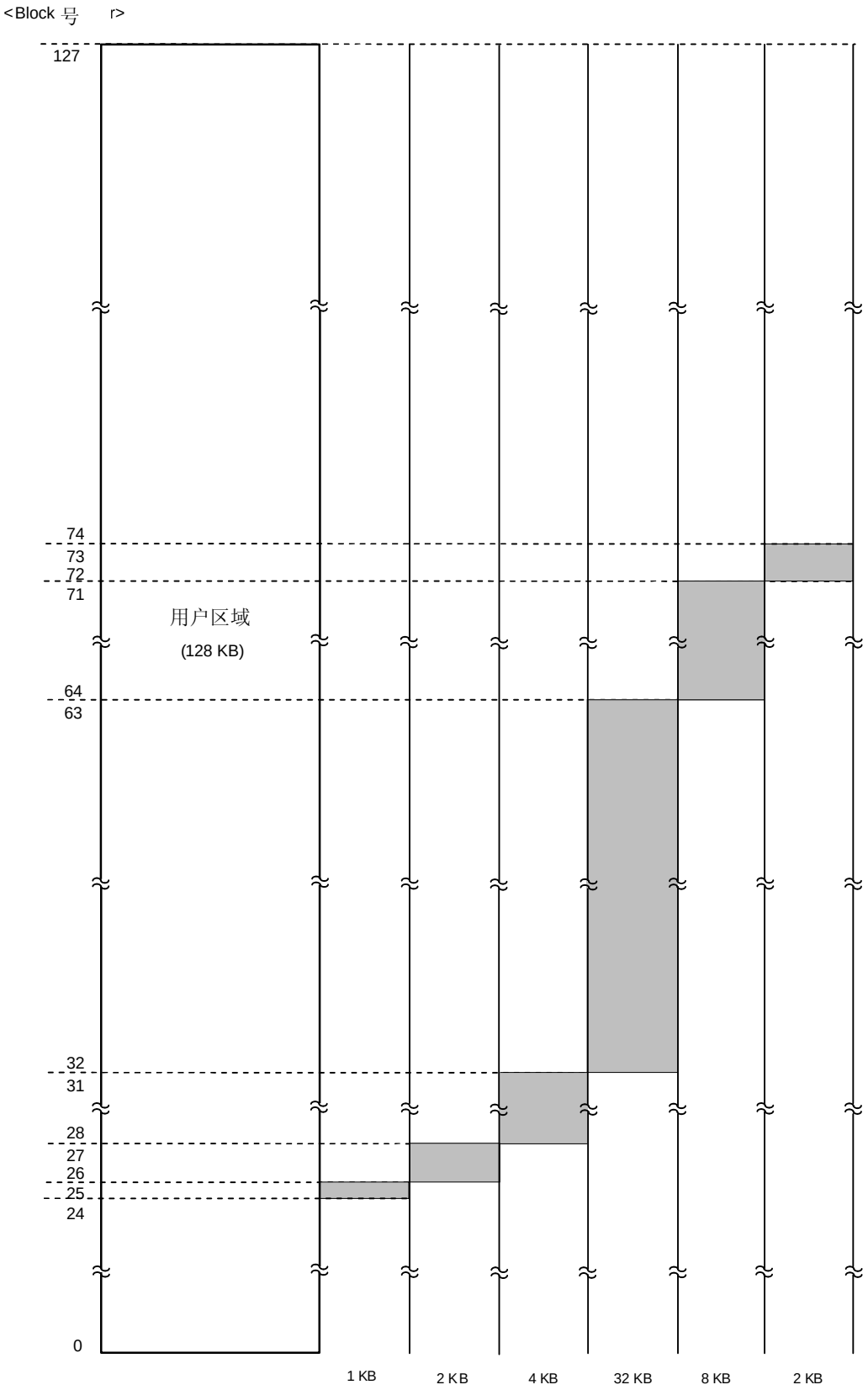
此外，满足条件2的值是1，2，4和8，满足条件3的值是8，所以同时选择和擦除的block是8；block 64 ～ 71擦除。

<6> block 64 ～ 71擦除后，接下来开始block数是72，需要擦除的数是2；因此满足条件1的值是1和2。

此外，满足条件2的值是1和2，满足条件3的值是2，所以同时选择和擦除的block是2；block 72和 73擦除。

因此，同时选择和擦除执行 6 次 (25, 26 和 27, 28 ～ 31, 32 ～ 63, 64 ～ 71,和 72 和 73) 擦除了 block 25 ～ 73，所以获得 M = 6。

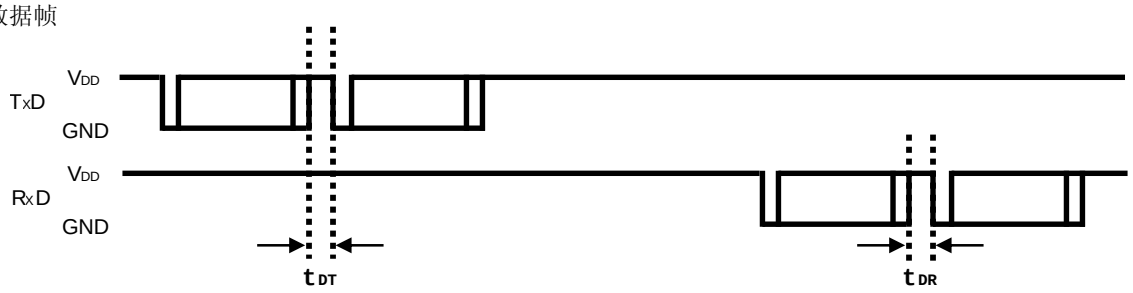
执行同时选择和擦除的block配置 (当擦除block 25 ～ 73)



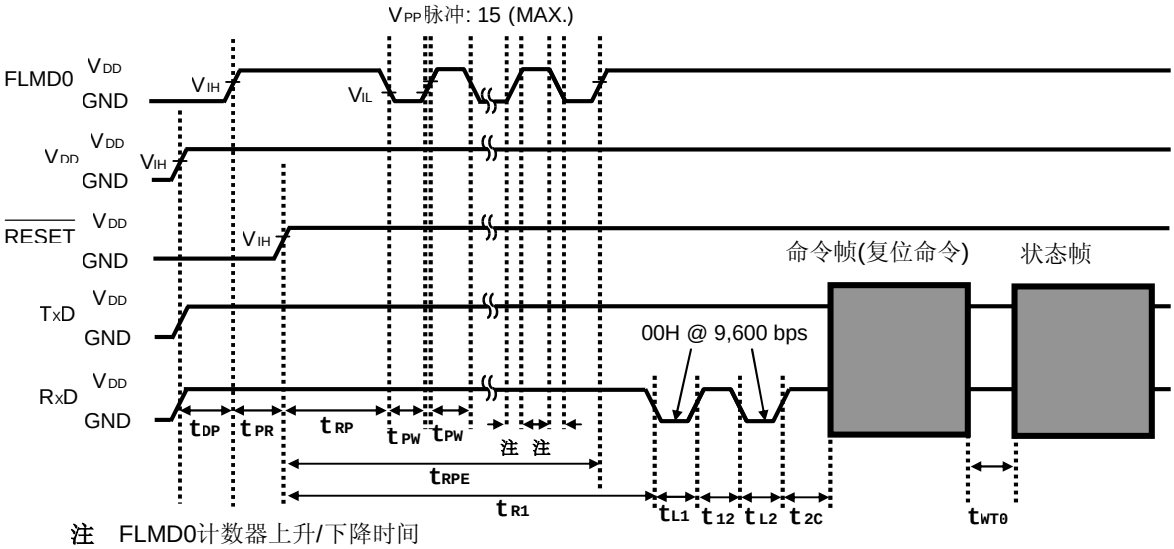

<可以同时选择和擦除的 block 范围>

## 8.4 UART 通讯模式

•数据帧

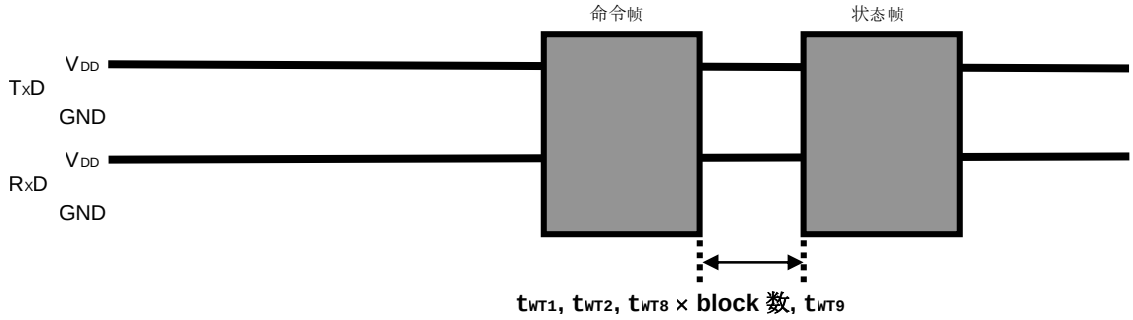



•编程模式设置/复位命令

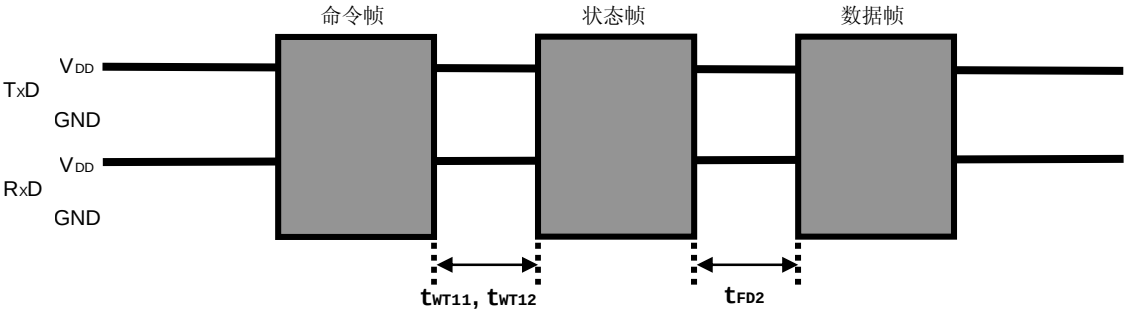



注 FLMD0计数器上升/下降时间

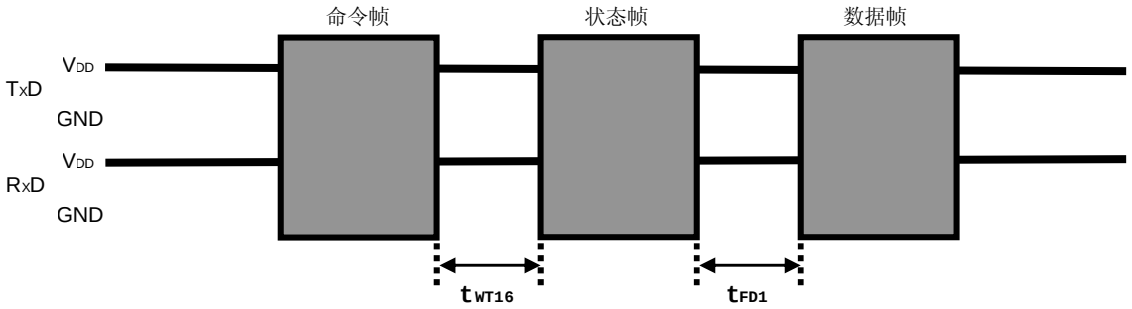
•芯片擦除命令/block 擦除命令/block 空白检测命令/振荡频率设置命令



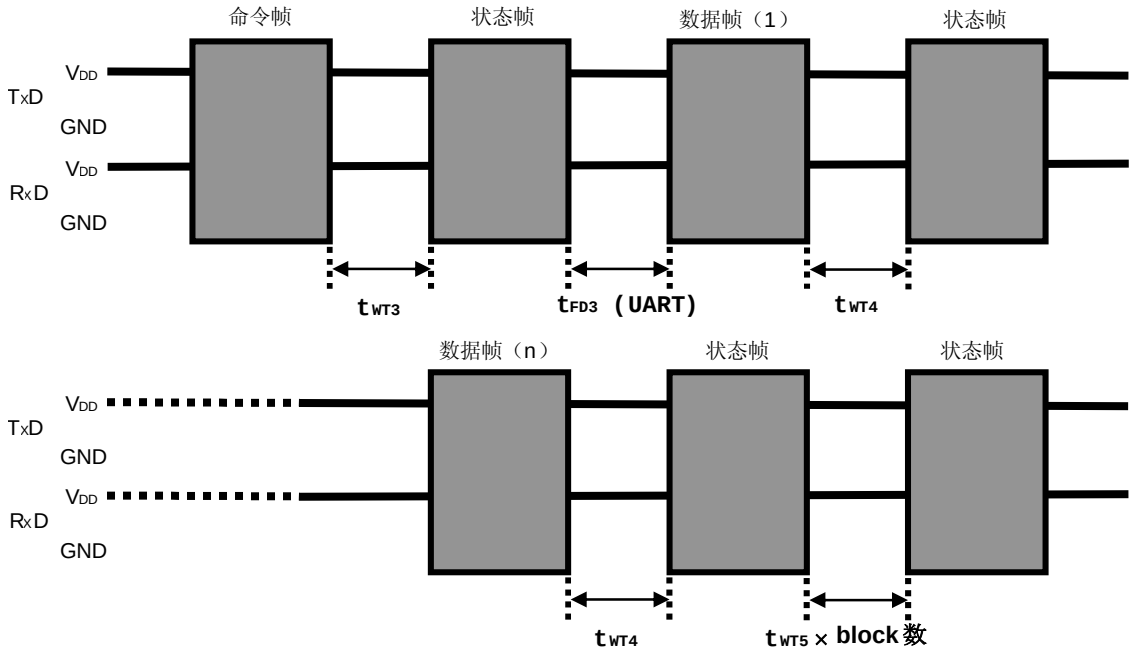

•硅标记命令/版本获取命令



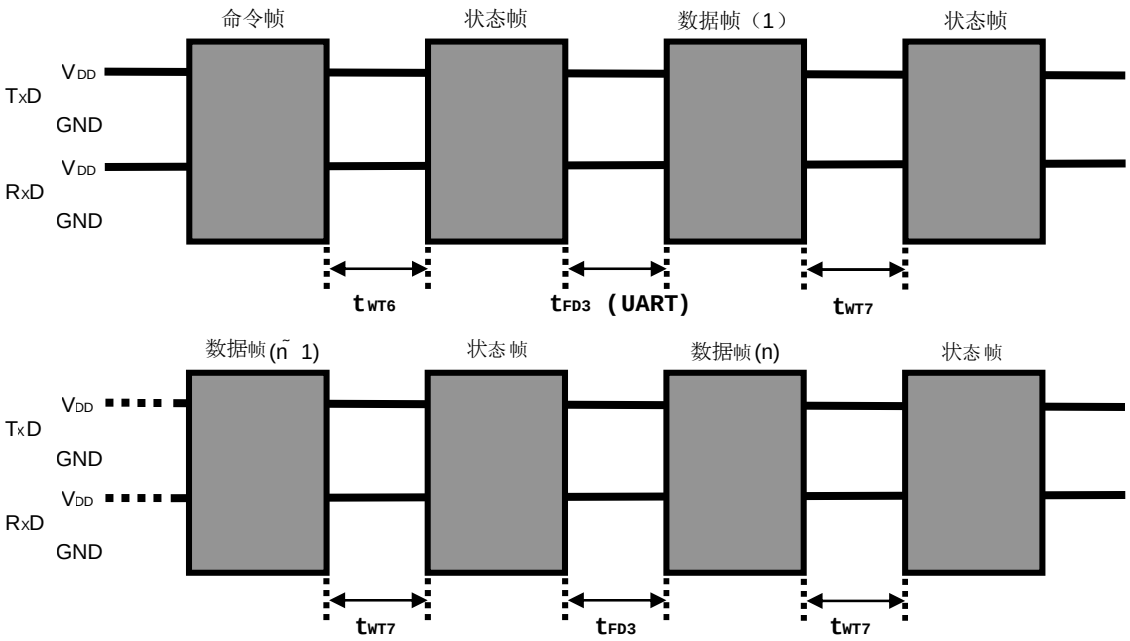
•校验和命令


命令帧
状态帧
数据帧
VDD
TXD
GND
VDD
RXD
GND
tWT16
tFD1

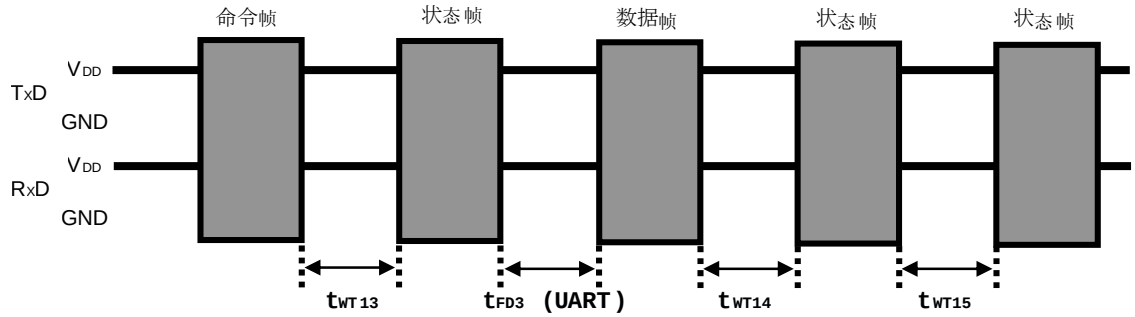
• 编程命令


命令帧
状态帧
数据帧（1）
状态帧
VDD
TXD
GND
VDD
RXD
GND
tWT3
tFD3 (UART)
tWT4
数据帧（n）
状态帧
状态帧
VDD
TXD
GND
VDD
RXD
GND
tWT4
tWT5 × block数

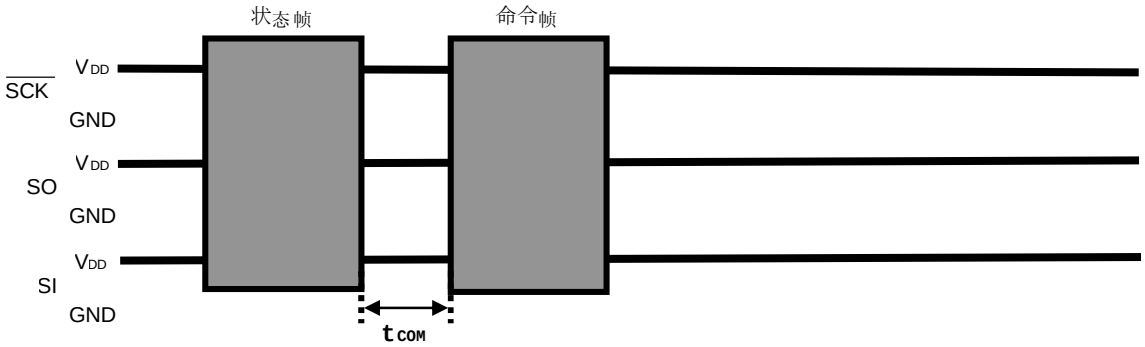
• 校验命令




•安全设置命令

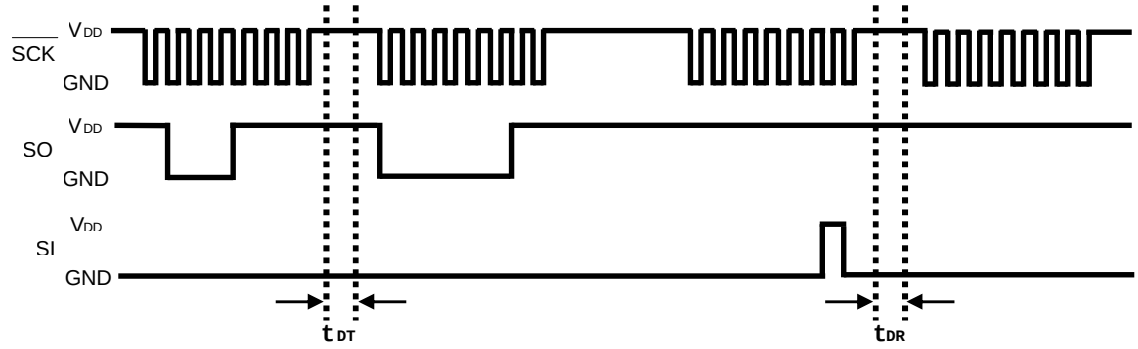



•命令帧发送前的等待

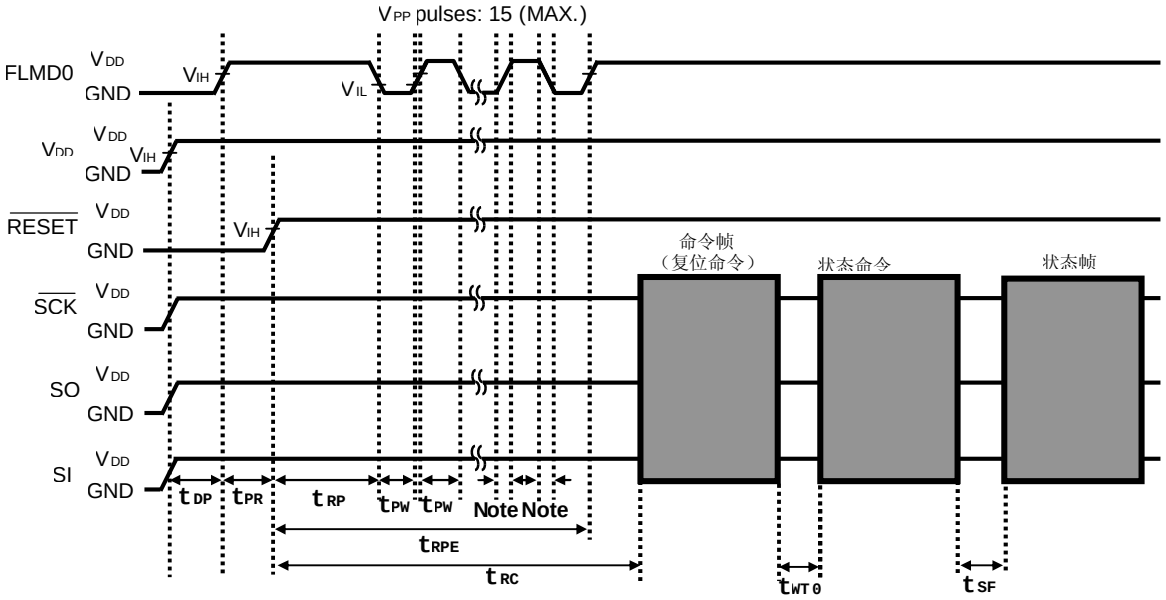


## 8.5 3 线串行 I/O 通讯模式

●数据帧

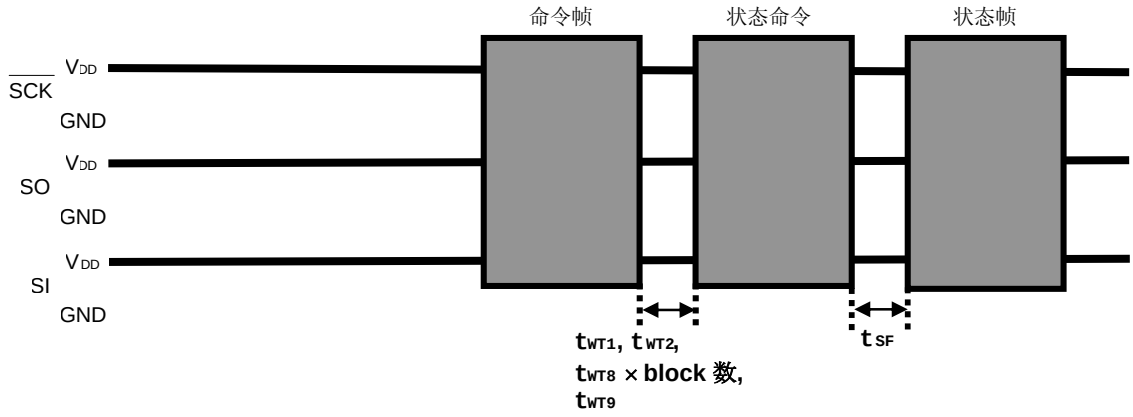



●编程模式设置/复位命令

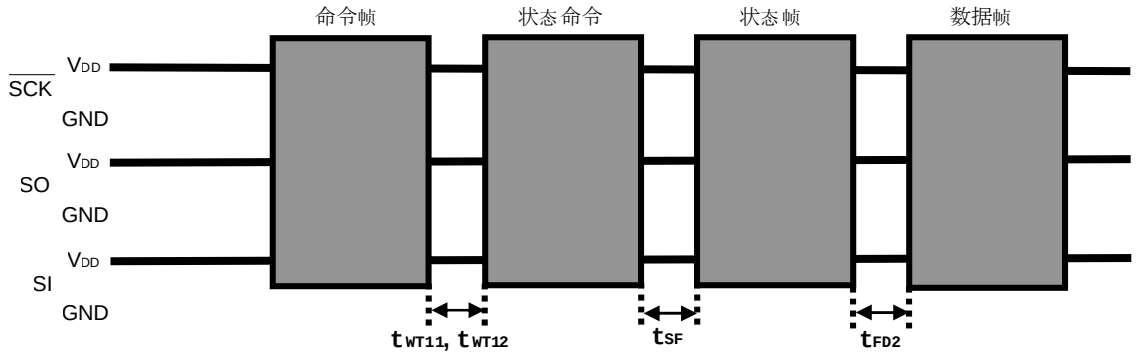



**注** FLMD0 计数器上升/下降时序

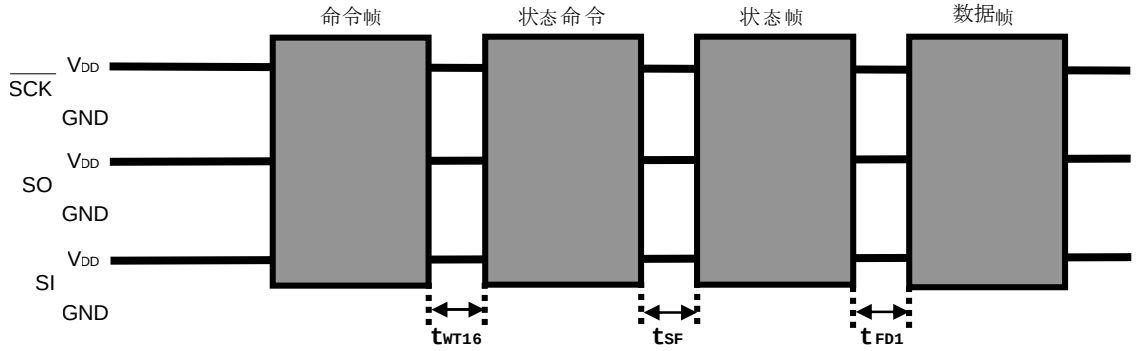
●芯片擦除命令/blcok 擦除命令/block 空白检测命令/振荡频率设置命令



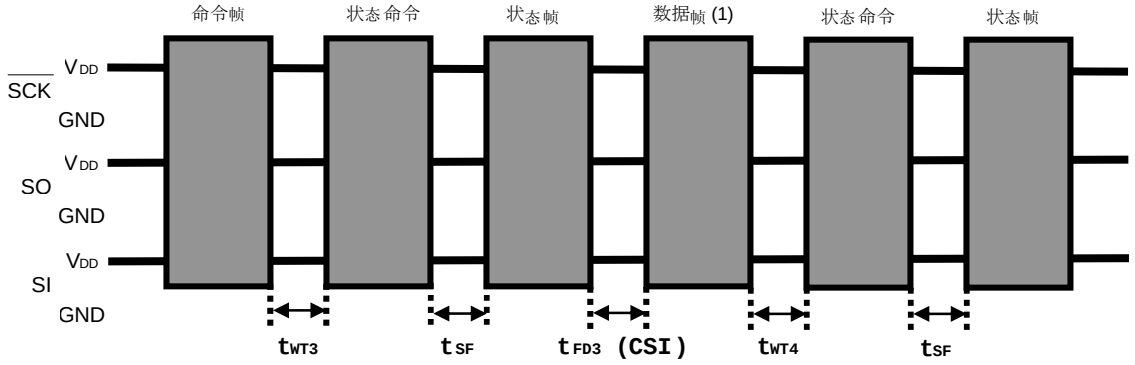
•硅标记命令/版本获取命令

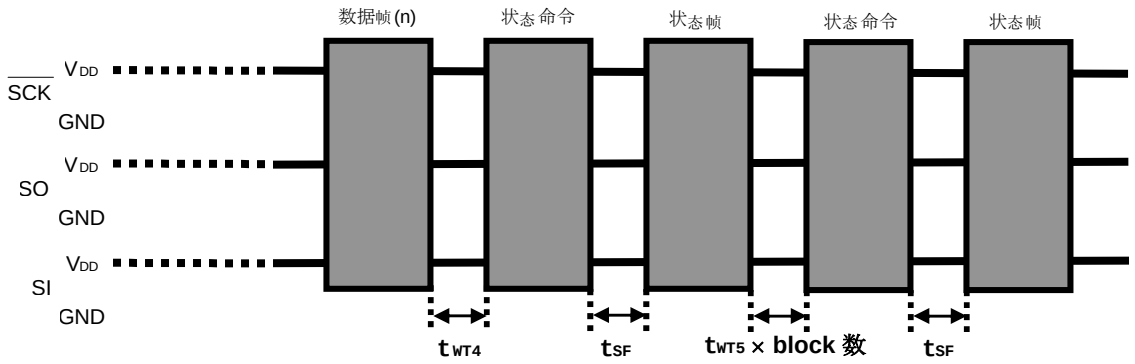



• 验和命令

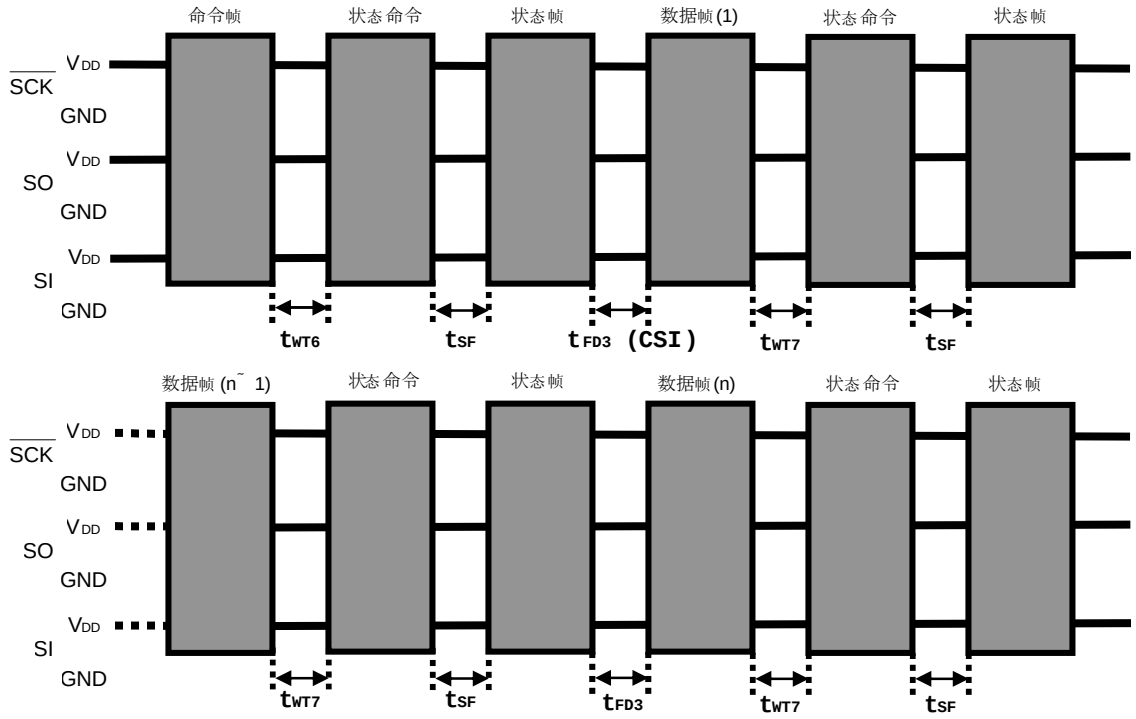



•编程命令

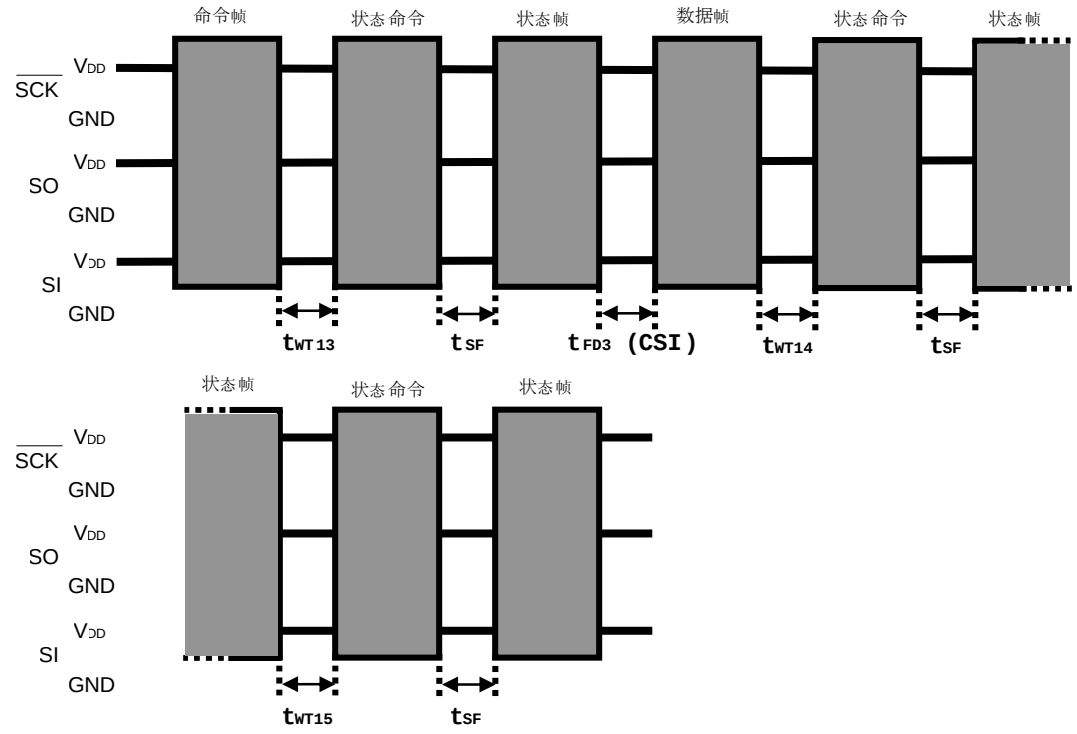


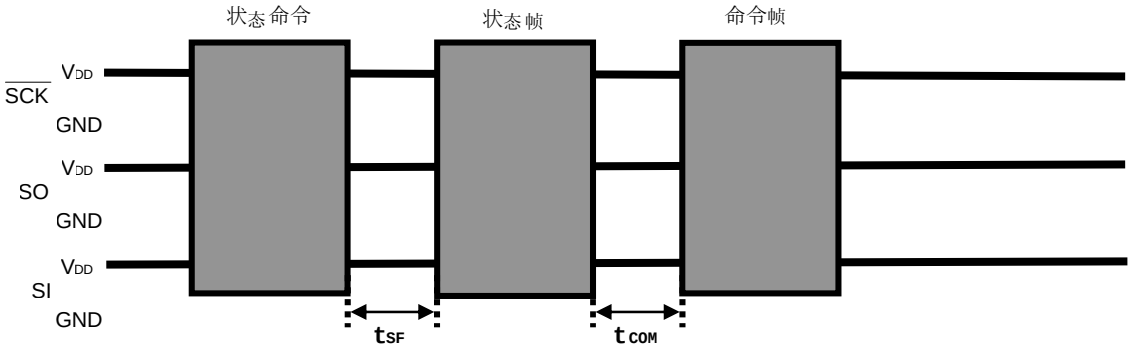
•校验命令


命令帧
状态 命令
状态 帧
数据帧 (1)
状态 命令
状态 帧
SCK
SO
SI
VDD
GND
tWT6
tSF
tFD3 (CSI)
tWT7
tSF
数据帧 (n˜ 1)
状态 命令
状态 帧
数据帧 (n)
状态 命令
状态 帧
tWT7
tSF
tFD3
tWT7
tSF

• 安全设置命令


命令帧
状态 命令
状态 帧
数据帧
状态 命令
状态 帧
SCK
SO
SI
VDD
GND
tWT13
tSF
tFD3 (CSI)
tWT14
tSF
状态帧
状态 命令
状态 帧
tWT15
tSF

•命令帧发送前的等待



要获取更多的信息

请联系：

**NEC Electronics Corporation**
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

**[America]**

**NEC Electronics America, Inc.**
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

**[Europe]**

**NEC Electronics (Europe) GmbH**
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

**Hanover Office**
Podbielskistrasse 164
30177 Hannover
Tel: 0 511 33 40 2-0

**Munich Office**
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

**Stuttgart Office**
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

**United Kingdom Branch**
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

**Succursale Française**
9, rue Paul Dautier, B.P. 52180
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

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

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

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

**Branch The Netherlands**
Limburglaan 5
5616 HR Eindhoven
The Netherlands
Tel: 040 265 40 10

**[Asia & Oceania]**

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

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

**NEC Electronics Hong Kong Ltd.**
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

**Seoul Branch**
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737

**NEC Electronics Taiwan Ltd.**
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-2719-2377

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