应用笔记

78K0R/Kx3

16 位单片微控制器

Flash 存储器编程（编程器）

μPD78F1142
μPD78F1143
μPD78F1144
μPD78F1145
μPD78F1146

μPD78F1152
μPD78F1153
μPD78F1154
μPD78F1155
μPD78F1156

μPD78F1162
μPD78F1163
μPD78F1164
μPD78F1165
μPD78F1166
μPD78F1167
μPD78F1168

文件编号．U18433CA2V0AN00（第二版）
出版日期　2007 年 12 月 N

日本印刷

[备忘录]

CMOS设备的注释

① 输入引脚处的电压波形

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

② 未使用的输入引脚的处理

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

③ ESD防护措施

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

④ 初始化之前的状态

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

⑤ 电源开关顺序

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

对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

⑥ 电源关闭状态下的输入信号

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

每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断。

- **本文档所登载的内容有效期截止至 2007 年 12 月，信息先于产品的生产周期发布。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表或数据手册等相关资料以获取本公司产品的最新规格。**
- 并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。否则因本文档所登载内容引发的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权作出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：

  “标准等级”、“专业等级”以及“特殊等级”三种质量等级。

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

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

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

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

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

（注）

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

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

M5 02.11-1

前言

目标读者

这个应用笔记是写给那些希望理解 78K0R/Kx3 的功能以及使用这个产品设计应用系统的用户。

目的

这个应用笔记的目的是帮助用户理解如何开发专用的 flash 存储器来重新写入 78K0R/Kx3 的内部 flash 存储器。

这个文档中的样本程序和电路图仅供参考，并且不是用于实际设计中的使用。

因此，这些样本程序必须在用户自己的风险下使用。如果这些样本程序被使用，正确工作不被保证。

结构

这个手册由以下主要部分组成。

- Flash 存储器编程
- 编程器工作环境
- 基本编程器操作
- 命令/数据帧格式
- 命令处理的说明
- UART 通信模式
- Flash 存储器编程参数特性

如何阅读这个手册

假定这个手册的读者具有电气工程、逻辑电路和微控制器的基本知识。

- 要获得功能的全面理解：
  → 按照**目录**的顺序阅读这个手册。
- 要学习更多的 78K0R/Kx3 微控制器硬件功能：
  → 见每个 78K0R/Kx3 产品的用户手册。

约定

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

目录

第 1 章 FLASH 存储器编程 10

1.1 综述 10
1.2 系统配置 11
1.3 编程综述 12
1.3.1 设置 flash 存储编程模式 12
1.3.2 通过命令发送/接收操作 flash 存储器 13
1.4 78K0R/Kx3 的特有信息 14

第 2 章 编程器工作环境 16

2.1 编程器控制管脚 16
2.2 控制管脚的细节 17
2.2.1 Flash 存储器编程模式设置管脚（FLMD0） 17
2.2.2 串行接口管脚（TOOL0） 17
2.2.3 复位控制管脚 ($\overline{\text{RESET}}$) 18
2.2.4 V_{DD}/GND 控制管脚 18
2.2.5 其它管脚 18
2.3 基本流程图 19
2.4 设置 Flash 存储器编程模式 20
2.4.1 模式设置流程图 21
2.4.2 范例程序 22
2.5 单线 UART 通信模式 23
2.6 关断目标电源 23
2.7 存储器的操作 24
2.8 命令列表 24
2.9 状态列表 25

第 3 章 基本编程器操作 26

第 4 章命令/ 数据帧格式 27

4.1 命令帧发送处理 29
4.2 数据帧发送处理 29
4.3 数据帧接收处理 29

第 5 章 命令处理说明 30

5.1 Status 命令 30
5.1.1 说明 30
5.1.2 状态帧 30
5.2 Reset 命令 31
5.2.1 说明 31
5.2.2 命令帧和状态帧 31
5.3 Baud Rate Set 命令 32
5.3.1 说明 32

5.3.2 命令帧和状态帧 32
5.4 Chip Erase 命令 34
5.4.1 说明 34
5.4.2 命令帧和状态帧 34
5.5 Block Erase 命令 35
5.5.1 说明 35
5.5.2 命令帧和状态帧 35
5.6 Programming 命令 36
5.6.1 说明 36
5.6.2 命令帧和状态帧 36
5.6.3 数据帧和状态帧 36
5.6.4 发送所有数据的完成和状态帧 37
5.7 Verify 命令 38
5.7.1 说明 38
5.7.2 命令帧和状态帧 38
5.7.3 数据帧和状态帧 38
5.8 Block Blank Check 命令 40
5.8.1 说明 40
5.8.2 命令帧和状态帧 40
5.9 Silicon Signature 命令 41
5.9.1 说明 41
5.9.2 命令帧和状态帧 41
5.9.3 Silicon 签名数据帧 42
5.9.4 78K0R/Kx3 silicon 签名列表 44
5.10 Version Get 命令 49
5.10.1 说明 49
5.10.2 命令帧和状态帧 49
5.10.3 版本数据帧 50
5.11 Checksum 命令 51
5.11.1 说明 51
5.11.2 命令帧和状态帧 51
5.11.3 校验和数据帧 51
5.12 Security Set 命令 52
5.12.1 说明 52
5.12.2 命令帧和状态帧 52
5.12.3 数据帧和状态帧 53
5.12.4 内部校验检查和状态帧 53

第 6 章 UART 通信模式 55

6.1 命令帧发送处理流程图 55
6.2 数据帧发送处理流程图 56
6.3 数据帧接收处理流程图 57
6.4 Reset 命令 58
6.4.1 处理顺序图 58
6.4.2 处理顺序的说明 59
6.4.3 处理完成时的状态 59
6.4.4 流程图 60
6.4.5 范例程序 61

6.5 Baud Rate Set 命令 62
6.5.1 处理顺序图 62
6.5.2 处理顺序的说明 63
6.5.3 处理完成时的状态 63
6.5.4 流程图 64
6.5.5 范例程序 65
6.6 Chip Erase 命令 66
6.6.1 处理顺序图 66
6.6.2 处理顺序的说明 67
6.6.3 处理完成时的状态 67
6.6.4 流程图 68
6.6.5 范例程序 69
6.7 Block Erase 命令 70
6.7.1 处理顺序图 70
6.7.2 处理顺序的说明 71
6.7.3 处理完成时的状态 71
6.7.4 流程图 72
6.7.5 范例程序 73
6.8 Programming 命令 74
6.8.1 处理顺序图 74
6.8.2 处理顺序的说明 75
6.8.3 处理完成时的状态 76
6.8.4 流程图 77
6.8.5 范例程序 78
6.9 Verify 命令 80
6.9.1 处理顺序图 80
6.9.2 处理顺序的说明 81
6.9.3 处理完成时的状态 81
6.9.4 流程图 82
6.9.5 范例程序 83
6.10 Block Blank Check 命令 85
6.10.1 处理顺序图 85
6.10.2 处理顺序的说明 86
6.10.3 处理完成时的状态 86
6.10.4 流程图 87
6.10.5 范例程序 88
6.11 Silicon Signature 命令 89
6.11.1 处理顺序图 89
6.11.2 处理顺序的说明 90
6.11.3 处理完成时的状态 90
6.11.4 流程图 91
6.11.5 范例程序 92
6.12 Version Get 命令 93
6.12.1 处理顺序图 93
6.12.2 处理顺序的说明 94
6.12.3 处理完成时的状态 94
6.12.4 流程图 95
6.12.5 范例程序 96

6.13 Checksum 命令 97
6.13.1 处理顺序图 97
6.13.2 处理顺序的说明 98
6.13.3 处理完成时的状态 98
6.13.4 流程图 99
6.13.5 范例程序 100
6.14 Security Set 命令 101
6.14.1 处理顺序图 101
6.14.2 处理顺序的说明 102
6.14.3 处理完成时的状态 102
6.14.4 流程图 103
6.14.5 范例程序 104

第 7 章 FLASH 存储器编程参数特性 106

附录 A 电路图（参考） 122

第 1 章 FLASH存储器编程

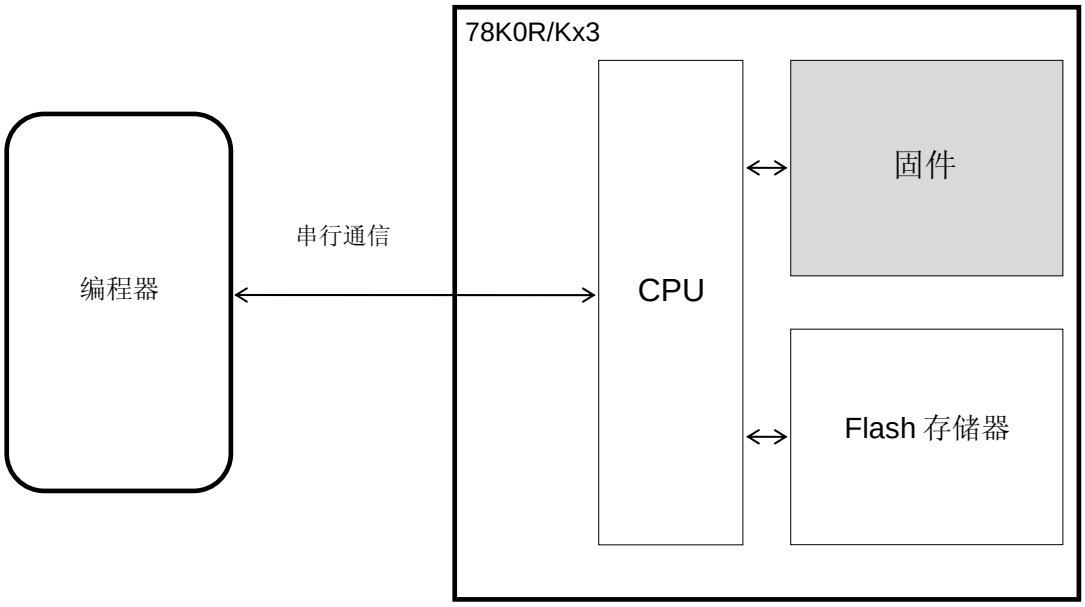
要写入 78K0R/Kx3 的内部 flash 存储器的内容，一个专用的 flash 存储器编程器（今后叫做“编程器”）一般被使用。

这个应用笔记解释如何开发一个专用的编程器。

1.1 综述

78K0R/Kx3 集成控制 flash 存储器编程的固件。通过串行通信在编程器和 78K0R/Kx3 之间发送/接收命令，内部 flash 存储器的编程被执行。

图 1-1. 78K0R/Kx3 中 Flash 存储器编程的系统概要



1.2 系统配置

编程 flash 存储器的系统配置的例子在图 1-2 中被显示。

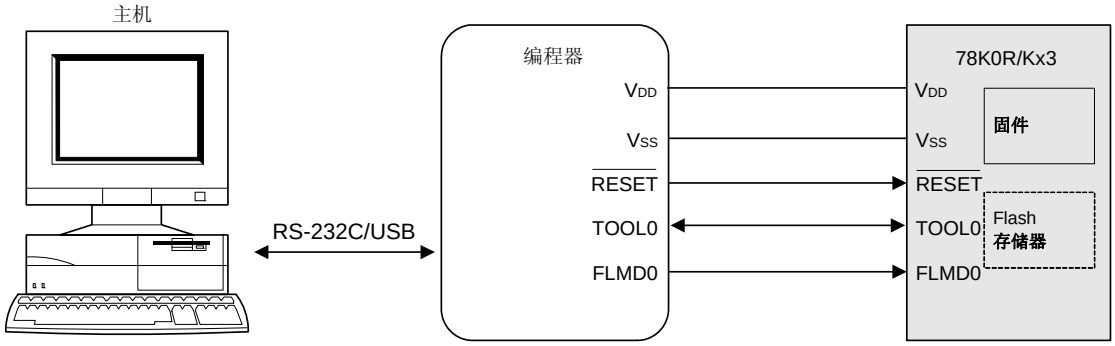
这个图表示在一台主机的控制下使用编程器如何编程 flash 存储器。

根据编程器如何被连接，如果用户程序已经事先被下载到编程器中，编程器可以在单独模式下使用，而不用主机。

例如，日电电子的 flash 存储器编程器 PG-FP4 可以通过连接的主机的 GUI 软件或者通过它自己（单独）被执行。

图 1-2. 系统配置

单线 UART 通信模式（LSB 首先发送）

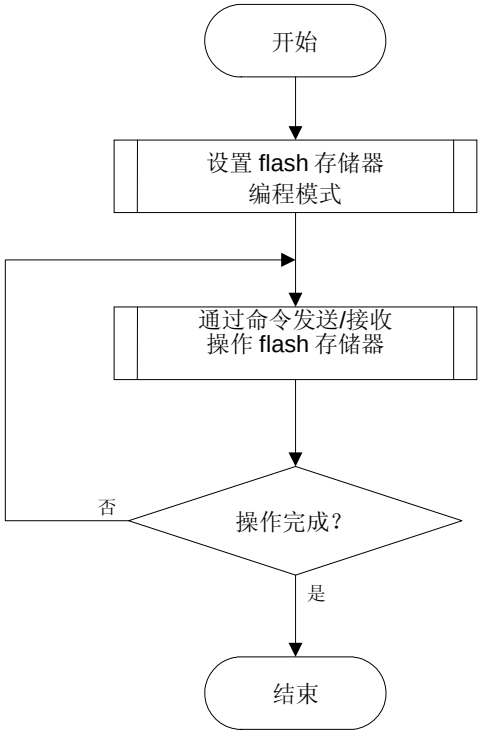


备注 78K0R/Kx3 只能通过单线 UART 通信模式通信。

1.3 编程综述

要使用编程器重新写入 flash 存储器的内容，78K0R/Kx3 必须首先被设置为 flash 存储器编程模式。然后，通过串行通信从编程器发送命令，并且重新写 flash 存储器。编程的流程图如图 1-3 所示。

图 1-3. 编程流程图



1.3.1 设置flash存储编程模式

提供一个特定的电压到 78K0R/Kx3 中的 flash 存储器编程模式设定管脚（FLMD0）并释放复位，然后，flash 存储器编程模式被设置。

1.3.2 通过命令发送/接收操作flash存储器

集成在 78K0R/Kx3 中的 flash 存储器有重新写入 flash 存储器内容的功能。表 1-1 中显示的 flash 存储器操作功能可以使用。

表 1-1. Flash 存储器功能要点

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

要控制这些功能，编程器通过串行通信发送命令到 78K0R/Kx3。78K0R/Kx3 返回命令的响应状态。flash 存储器编程通过重复串行通信的一系列命令被执行。

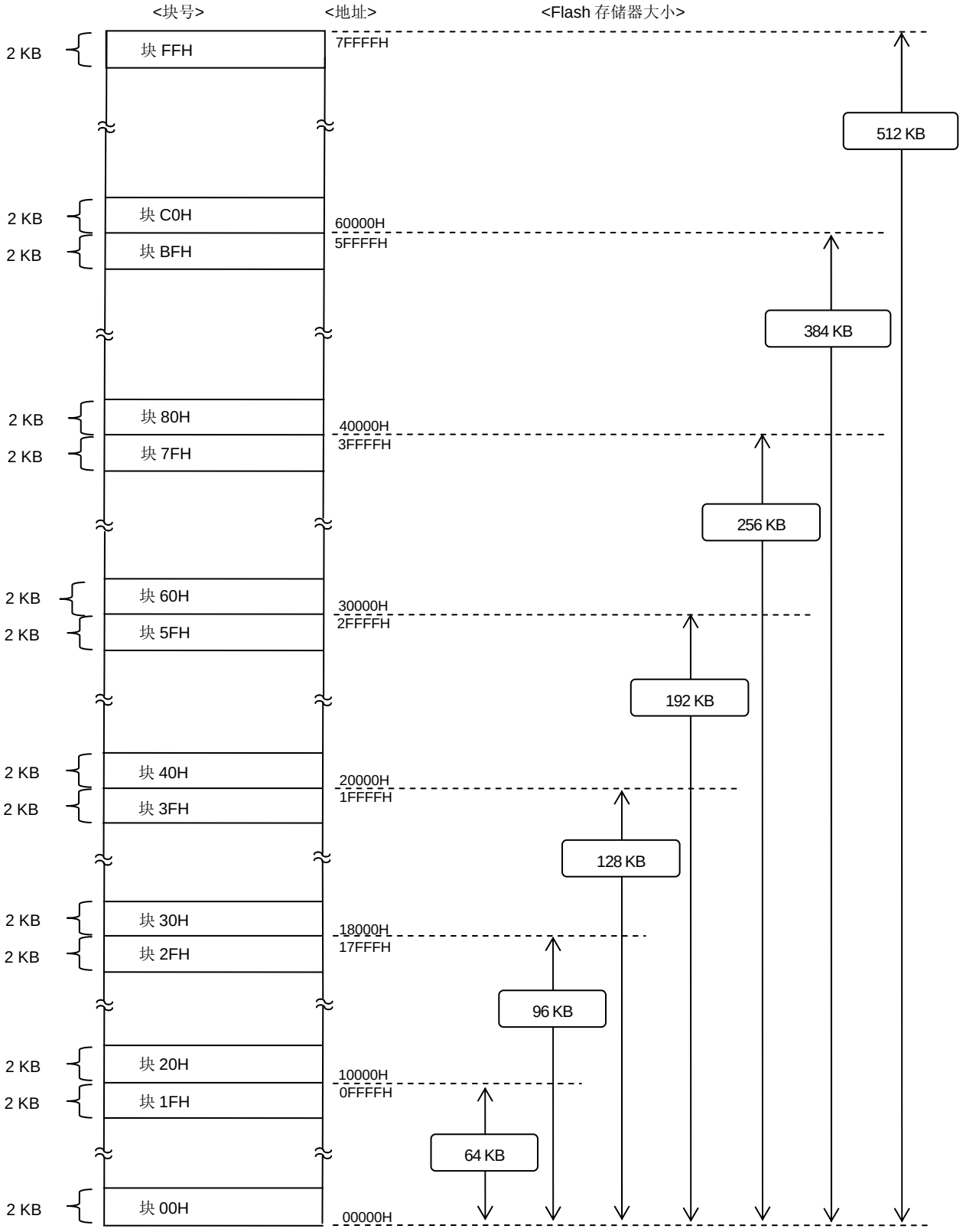
1.4 78K0R/Kx3 的特有信息

编程器必须管理产品特有信息（例如设备名和存储器信息）。
表 1-2 表示 78K0R/Kx3 的 flash 存储器大小，图 1-4 表示 flash 存储器的结构。

表 1-2. 78K0R/Kx3 的 Flash 存储器大小

设备名		Flash存储器大小
78K0R/KE3	μPD78F1142	64 KB
	μPD78F1143	96 KB
	μPD78F1144	128 KB
	μPD78F1145	192 KB
	μPD78F1146	256 KB
78K0R/KF3	μPD78F1152	64 KB
	μPD78F1153	96 KB
	μPD78F1154	128 KB
	μPD78F1155	192 KB
	μPD78F1156	256 KB
78K0R/KG3	μPD78F1162	64 KB
	μPD78F1163	96 KB
	μPD78F1164	128 KB
	μPD78F1165	192 KB
	μPD78F1166	256 KB
	μPD78F1167	384 KB
	μPD78F1168	512 KB

图 1-4. Flash 存储器结构



备注 每个块由 2 KB 组成（这个图只表示 flash 存储器中整个块的一些部分）。

第 2 章 编程器工作环境

2.1 编程器控制管脚

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

表 2-1. 管脚说明

编程器			78K0R/Kx3	连接时的过程
信号名	输入/输出	管脚功能	管脚名	
FLMD0	输出	模式信号	FLMD0	√
V_{DD}	输入/输出	V_{DD} 电压产生/监视	V_{DD} $EV_{DD\ (0/1)}$ $AV_{REF\ (0/1)}$注	√
GND	–	地	V_{SS} $EV_{SS\ (0/1)}$ AV_{SS}	√
CLK	输出	时钟输出	–	×
/RESET	输出	复位信号	$\overline{RESET}$	√
SI/RxD	输入	接收信号	TOOL0	√
SO/TxD	输出	发送信号		
SCK	输出	发送时钟	–	×

注 当执行板外写入操作时，连接这个管脚到 V_{DD}。
当执行板上写入操作时，提供与正常工作模式相同的电源。（这时，确认设置 $V_{DD} \geq AV_{REF(0/1)}$。）

备注 √：确认连接这个管脚。
×：这个管脚不必连接。

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

2.2 控制管脚的细节

2.2.1 Flash 存储器编程模式设置管脚（FLMD0）

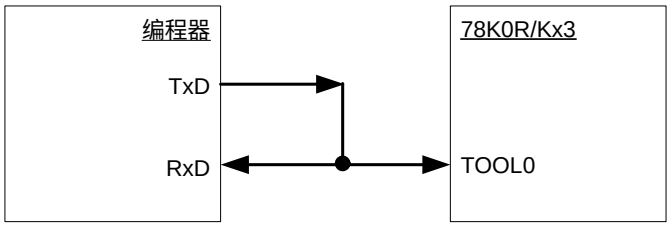
FLMD0 管脚被用于控制 78K0R/Kx3 的工作模式。当特定的电压被提供到这个管脚并且复位被释放时，78K0R/Kx3 工作于 flash 存储器编程模式。

2.2.2 串行接口管脚（TOOL0）

串行接口管脚被用于在编程器和 78K0R/Kx3 之间发送 flash 存储器写入命令。

以下图表示使用的管脚的连接。

图 2-1. 串行接口管脚

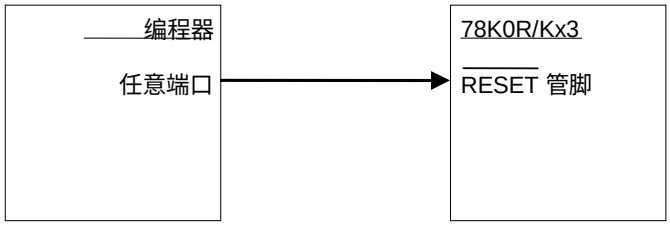


单线 UART通信

2.2.3 复位控制管脚 （RESET）

复位控制管脚（RESET 管脚）被用于从编程器控制 78K0R/Kx3 的系统复位。当特定的电压被提供到 FLMD0 管脚并且复位被释放时，flash 存储器编程模式可以被选择。

图 2-2. RESET 管脚

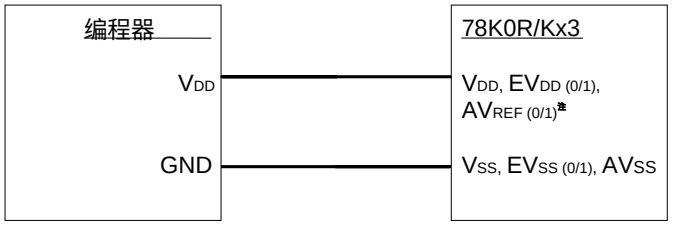


2.2.4 VDD/GND 控制管脚

V_{DD} 控制管脚被用于从编程器向 78K0R/Kx3 提供电源。当不需要从编程器向 78K0R/Kx3 提供电源时，这个管脚不需要连接。然而，当专用的编程器被使用时，不论是否从编程器提供电源，这个管脚必须被连接，因为专用的编程器监视 78K0R/Kx3 的电源状态。

不论是否从编程器提供电源，GND 控制管脚必须被连接到 78K0R/Kx3 的 V_{SS}。

图 2-3. V_{DD}/地 控制管脚



注 当执行板外写入操作时，连接这个管脚到 V_{DD}。
当执行板上写入操作时，提供与正常工作模式相同的电源。（这时，确认设置 $V_{DD} \geq AV_{REF(0/1)}$。）

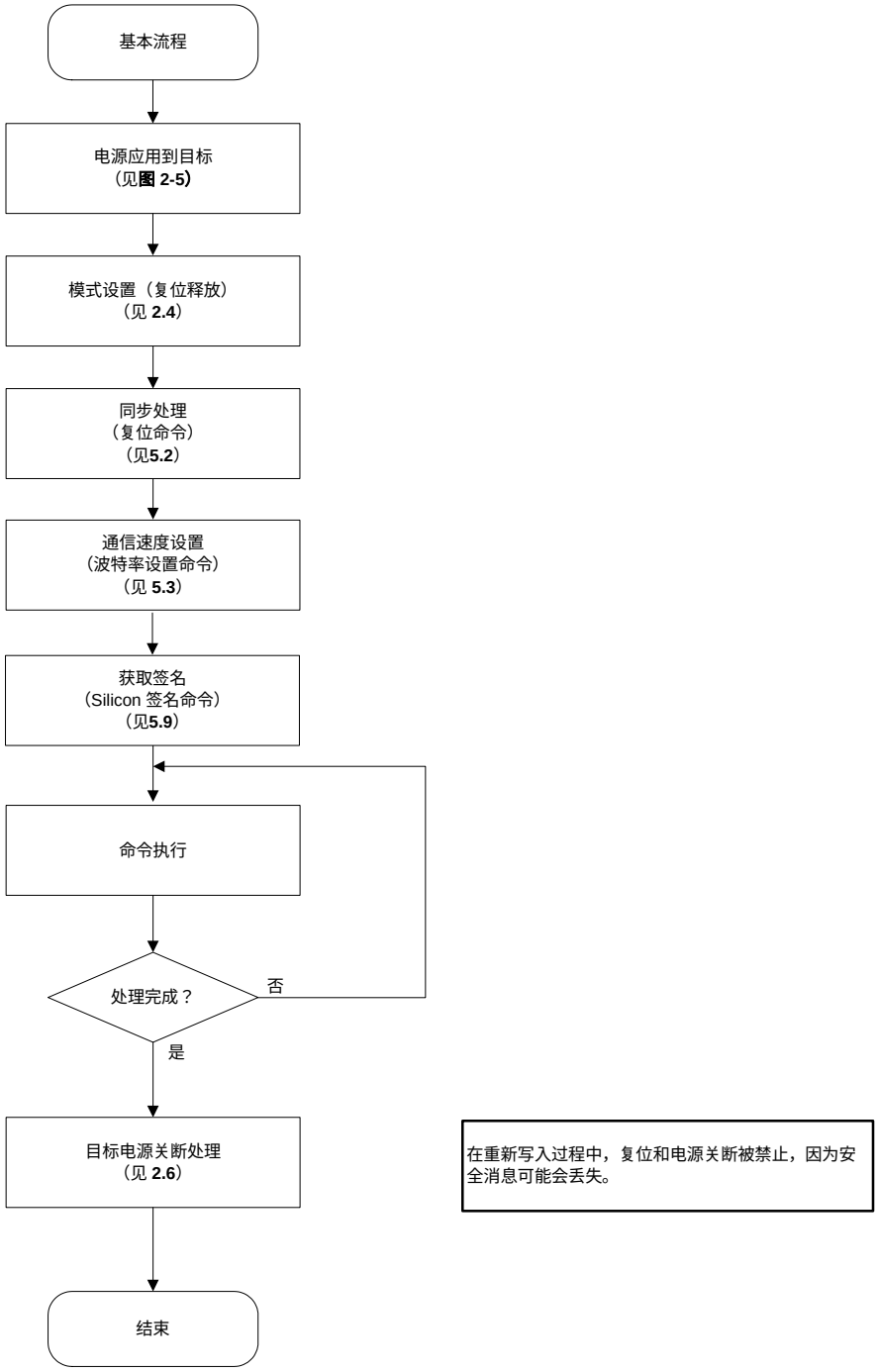
2.2.5 其它管脚

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

2.3 基本流程图

以下显示使用编程器执行 flash 存储器重新写入的基本流程图。

图 2-4. Flash 存储器重新写入处理的基本流程图

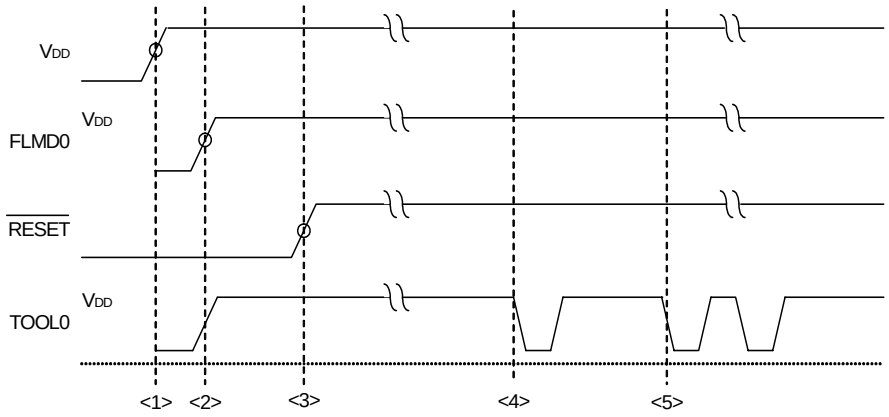


2.4 设置 Flash 存储器编程模式

要使用编程器重新写入 flash 存储器内容，必须首先向 78K0R/Kx3 中的 flash 存储器编程模式设置管脚（FLMD0）提供一个特定的电源并释放复位，将 78K0R/Kx3 设置为 flash 存储器编程模式。

以下表示设置 flash 存储器编程模式的时序图。

图 2-5. 设置 Flash 存储器编程模式



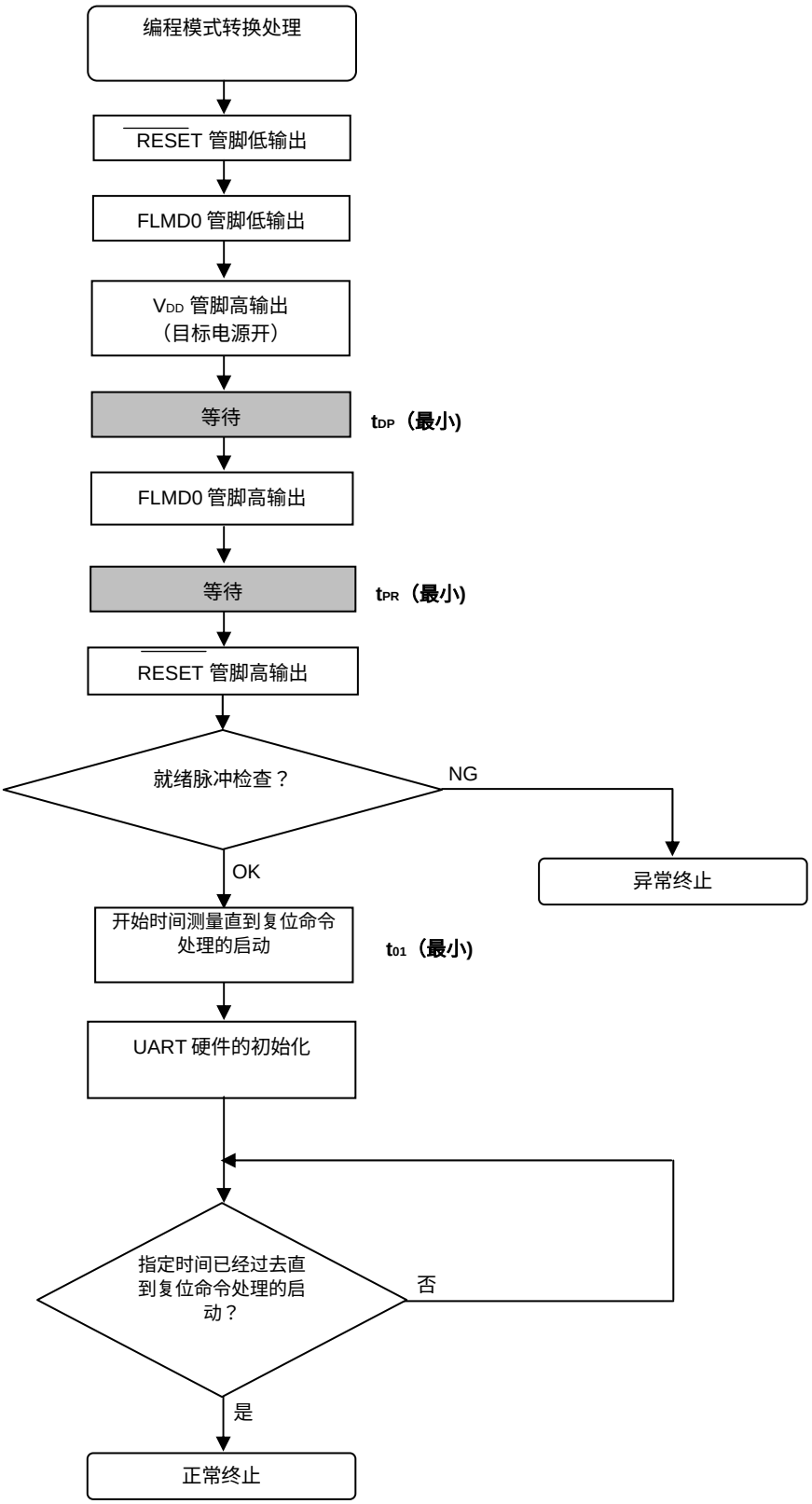
<1>：电源应用（V_{DD}）
<2>：FLMD0 = 高电平
<3>：复位释放（串行编程模式设置）
<4>：READY 脉冲（“ 00 ” @9600 bps）输入启动（78K0R/Kx3 → 编程器）
<5>：低脉冲（“ 00 ” @9600 bps）输出启动（编程器 → 78K0R/Kx3）

复位释放后 FLMD0 管脚的设置和工作模式之间的关系如下所示。

表 2-2. 复位释放后 FLMD0 管脚的设置和工作模式之间的关系

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

2.4.1 模式设置流程图


编程模式转换处理
RESET 管脚低输出
FLMD0 管脚低输出
VDD 管脚高输出
（目标电源开）
等待
tDP（最小）
FLMD0 管脚高输出
等待
tPR（最小）
RESET 管脚高输出
就绪脉冲检查？
NG
异常终止
OK
开始时间测量直到复位命令处理的启动
t01（最小）
UART 硬件的初始化
指定时间已经过去直到复位命令处理的启动？
否
是
正常终止

2.4.2 样本程序

以下表示模式设置处理的一个样本程序。

```
/*****************************************************************/
/*                                                               */
/* 连接到Flash设备                                               */
/*                                                               */
/*****************************************************************/
u16         fl_con_dev(void)
{
extern      void      init_fl_uart(void);
extern      void      init_fl_csi(void);
extern      void      stop_UART0(void);

   u16      rc = NO_ERROR;

   SRMK0 = true;                    // 使 UART Rx INT.无效
   UARTE0 = false;            // 使 UART 硬件无效
   stop_UART0();                    // TxD/RxD = 高阻

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

   fl_wait(tDP);                    // 等待

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

   pFL_RES          = hi;           // RESET = 高

   rc = check_ready_pulse();  // 检查从目标设备输出的 “ READY PULSE ”
   if (rc){
            return    rc;           // 脉冲宽度/时序错误
   }
   start_flto(t01);           // 启动 “ t01 ” 等待定时器

   init_fl_uart();            // 初始化UART硬件（用于Flash设备控制）
   UARTE0 = true;             // 使能 UART硬件
   SRIF0 = false;             // 清除 UART Rx IRQ 标志
   SRMK0 = false;             // 使能 UART Rx INT

   while(!check_flto())       // 超时 “ t01 ” ？
            ;                 // 否

   return    rc;
   // 启动 RESET 命令处理
}
```

2.5 单线 UART 通信模式

78K0R/Kx3 的 TOOL0 管脚被用于单线 UART 通信。通信条件如下所示。

表 2-3. 单线 UART 通信条件

项目	说明
波特率	在用于波特率设置命令处理的 Baud Rate Set 命令被发送前，通信以 9,600 bps 波特率被执行。发送速率从 Reset 命令的发送被更改到 Baud Rate Set 命令设置的波特率。关于可以设置的波特率的细节，参阅 **5.3 Baud Rate Set 命令**。
奇偶位	无
时间长度	8 位 （LSB 在前）
停止位	2 位（编程器 → 78K0R/Kx3）/1 位（78K0R/Kx3 → 编程器）

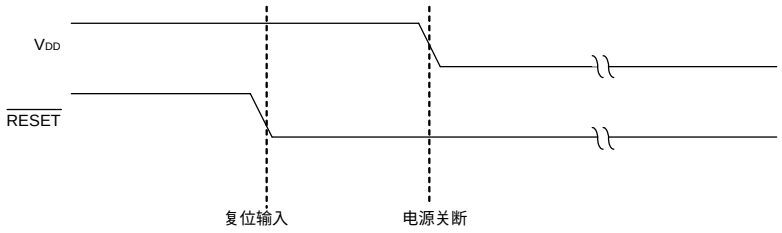
注意事项 对编程器和 78K0R/Kx3 设置相同的波特率。

2.6 关断目标电源

在每个命令执行完成后，在设置 $\overline{\text{RESET}}$ 管脚到低电平后，关断目标的电源，如下所示。
当关断目标的电源时，设置其它管脚为高阻。

注意事项 在命令处理过程中，关断电源和输入复位被禁止。

图 2-6. 终止 Flash 存储器编程模式的时序



2.7 Flash 存储器的操作

78K0R/Kx3 中集成的 flash 存储器有操作 flash 存储器的功能，如表 2-4 所示。编程器发送命令到 78K0R/Kx3 来控制这些功能，并且检查从 78K0R/Kx3 发出的响应状态，来操作 flash 存储器。

表 2-4. Flash 存储器操作功能列表

分类	功能名	说明
擦除	片擦除	擦除整个 flash 存储器区域。清除安全标志。
	块擦除	擦除 flash 存储器中的指定块。
写入	写入	写入数据到 flash 存储器中的指定区域。
校验	校验	在 78K0R/Kx3 端，比较从 flash 存储器中指定地址获取的数据和从编程器发送的数据。
空白检查	块空白检查	检查 flash 存储器中指定区域的擦除状态。
信息获取	Silicon 签名获取	获取写入协议信息。
	版本获取	获取 78K0R/Kx3 和固件的版本信息。
	校验和获取	获取指定区域的校验和数据。
安全	安全设置	设置安全信息。
其它	复位	删除通信中的同步。

2.8 命令列表

编程器使用的命令和它们的功能被列表如下。

表 2-5. 从编程器发送到 78K0R/Kx3 的命令列表

命令号	命令名	功能
00H	Reset	删除通信中的同步。
9AH	Baud Rate Set	设置单线 UART 的波特率。
20H	Chip Erase	擦除整个 flash 存储器区域。
22H	Block Erase	擦除 flash 存储器中的指定区域。
40H	Programming	写入数据到 flash 存储器中的指定区域。
13H	Verify	比较从 flash 存储器中指定区域的内容和从编程器发送的数据。
32H	Block Blank Check	检查 flash 存储器中指定区域的擦除状态。
C0H	Silicon Signature	获取 78K0R/Kx3 信息（部件号，flash 存储器配置，等等）。
C5H	Version Get	获取 78K0R/Kx3 和固件的版本信息。
B0H	Checksum	获取指定区域的校验和数据。
A0H	Security Set	设置安全信息。

2.9 状态列表

下面的表列表编程器从 78K0R/Kx3 接收的状态码。

表 2-6. 状态码列表

状态码	状态	说明
04H	命令号错误	如果一个不支持的命令被接收返回的错误
05H	参数错误	如果命令信息（参数）无效返回的错误
06H	正常响应（ACK）	正常响应
07H	校验和错误	如果从编程器发送的一帧中的数据异常返回的错误
0FH	校验错误	校验从编程器发送的数据时如果校验错误发生返回的错误
10H	保护错误	试图执行被 Security Set 命令禁止的处理时返回的错误
15H	消极响应（NACK）	消极响应
1AH	MRG10 错误	擦除校验错误
1BH	MRG11 错误	数据写入过程中的内部校验错误或空白检查错误
1CH	写入错误	写入错误
FFH	处理进行中（BUSY）	忙响应[注]

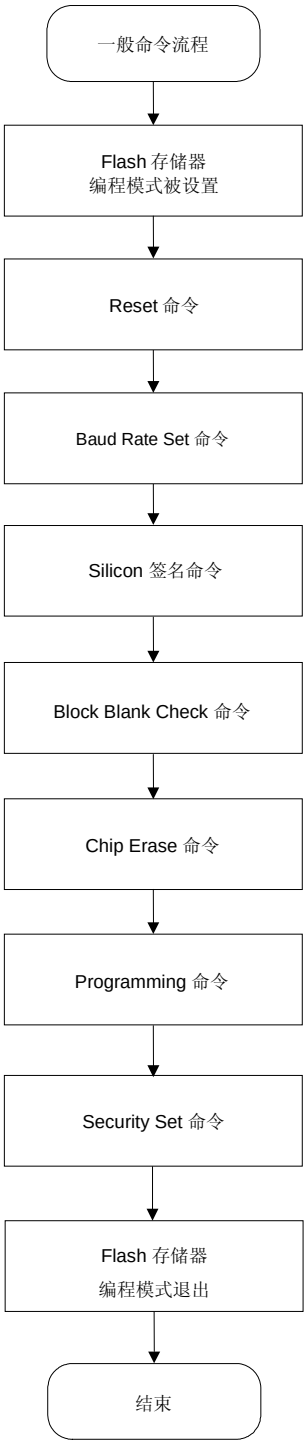
注 在 CSI 通信过程中，1 字节 “ FFH ” 可能被发送，“ FFH ” 也作为数据帧格式。

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

尽管没有在上表列出，如果一个超时错误（在 UART 通信过程中的 BUSY 超时或数据帧接收超时）发生，建议关断 78K0R/Kx3 的电源（参阅 **2.6 关断目标电源**）并且然后再次连接电源。

图 3-1 显示当使用编程器执行 flash 存储器重新写入时的一般的命令执行流程。

图 3-1. Flash 存储器重新写入时的一般命令执行流程



备注 Verify 命令和 Checksum 命令也可以被支持。

第 4 章 命令/数据帧格式

编程器使用命令帧来发送命令到 78K0R/Kx3。78K0R/Kx3 使用数据帧来发送写入数据或校验数据到编程器。头、脚、数据长度信息和校验和被附加到每帧来提高发送数据的可靠性。

以下表示命令帧和数据帧的格式。

图 4-1. 命令帧格式

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

图 4-2. 数据帧格式

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

表 4-1. 每帧中符号的说明

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

以下表示对一帧计算校验和（SUM）的例子。

[命令帧]

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

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

对这个命令帧，校验和数据按照下面被获得。

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

最终发送的命令帧如下所示。

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

[数据帧]

要发送如下所示的数据帧，LEN 和 D1 到 D4 是校验和计算目标。

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

对这个数据帧，校验和按照下面被获得。

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

最终发送的数据帧如下所示。

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

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

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

↑如果正常，应该是 1BH

4.1 命令帧发送处理

关于发送命令帧的处理的流程图的细节，请读 **6.1 命令帧发送处理流程图**。

4.2 数据帧发送处理

写入数据帧（用户程序）、校验数据帧（用户程序）和安全数据帧（安全标志）被作为数据帧发送。
关于发送数据帧的处理的流程图的细节，请读 **6.2 数据帧发送处理流程图**。

4.3 数据帧接收处理

状态帧、silicon 签名数据帧、版本数据帧和校验和数据帧被作为数据帧接收。
关于接收数据帧的处理的流程图的细节，请读 **6.3 数据帧接收处理流程图**。

第 5 章 命令处理说明

5.1 Status 命令

5.1.1 说明

78K0R/Kx3 在发布各种命令后的给定时间周期内自动发送一个状态帧来报告它的工作状态，例如写入或擦除。

在编程器执行每个命令后，如果由于通信或类似的问题 Status 命令帧不能被 78K0R/Kx3 正常接收，状态设置不能被 78K0R/Kx3 执行。因此，一个忙响应（FFH）而不是状态帧可能被接收。在这种情况下，重试每个命令。

5.1.2 状态帧

图 5-1 表示对应每个命令的状态帧。

图 5-1. 对于 Status 命令的状态帧（从 78K0R/Kx3 到编程器）

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

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

2. 状态帧的长度根据要发送到 78K0R/Kx3 的每个命令（例如写入或擦除）而改变。

5.2 Reset 命令

5.2.1 说明

这个命令被用来在通信模式被设置后检查编程器和 78K0R/Kx3 之间的通信建立。

必须对编程器和 78K0R/Kx3 设置同样的波特率，然而，78K0R/Kx3 不能检测它自己的波特率产生时钟频率，所以波特率不能被设置。当“00H”从编程器以 9,600 bps 被发送两次，并且 78K0R/Kx3 测量“00H”的低电平宽度并计算两次发送信号的平均值，78K0R/Kx3 被使能来自己检测波特率产生时钟频率。因此，波特率可以被设置，从而使能通信中的同步检测。

5.2.2 命令帧和状态帧

图 5-2 表示 Reset 命令的命令帧的格式，图 5-3 表示这个命令的状态帧。

图 5-2. Reset 命令帧（从编程器到 78K0R/Kx3）

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

图 5-3. Reset 命令的状态帧（从 78K0R/Kx3 到编程器）

STX	LEN	Data	SUM	ETX
02H	1	ST1	校验和	03H

备注 ST1：同步检测结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.4 Reset 命令**。

5.3 Baud Rate Set 命令

5.3.1 说明

这个命令被用来更改 UART 命令的波特率（默认 9,600 bps）。

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

波特率设置数据由 1 字节值表示。

5.3.2 命令帧和状态帧

图 5-4 表示 Baud Rate Set 命令的命令帧格式，图 5-5 表示这个命令的状态帧。

图 5-4. Baud Rate Set 命令帧（从编程器到 78K0R/Kx3）

SOH	LEN	COM	命令信息[注]				SUM	ETX
01H	05H	9AH	D01	D02H	D02L	D03	校验和	03H

注 关于命令信息的细节，参阅**表 5-1**。如果表 5-1 中以外的数据被设置，一个超时错误将发生。
如果一个超时错误发生，执行硬件复位并重新设置 flash 存储器编程模式。

备注 D01： 同步修正模式
D02H，D02L： 波特率设置
D03： 噪声滤波器设置

表 5-1. 命令信息设置

同步修正模式	D01	D02H	D02L	D03
微控制器修正模式	00H	固定为 00H	固定为 0AH （115,200 bps）	噪声滤波器 00H：关 01H：开
编程器修正模式	01H	**注**	**注**	

注 将 D02H/D02L 以十六进制替换为下面表达式计算的 k 值。确认 k 值比 0003H 大。

$k = （8 \times 10^6 \times E）/BAUD\ RATE$

E：在 flash 导入过程中 78K0R 的 READY 脉冲（9,600 bps）误差

例 1： 0% READY 脉冲（低电平 9 位@ 9,600 bps）长度
（READY 脉冲 = 937.5 μs）
当设置为 250,000 bps 时
E = 1.00
k = 0020H
D02H = 00H
D02L = 20H

例 2： +5% READY 脉冲（低电平 9 位@ 9,600 bps）长度
（READY 脉冲 = 984.375 μs）
当设置为 250,000 bps 时
E = 1.05
k = 0021H
D02H = 00H
D02L = 21H

例 3: –5% READY 脉冲（低电平 9 位@ 9,600 bps）长度
（READY 脉冲 = 890.625 μs）
当设置为 250,000 bps 时
E = 0.95
k = 001EH
D02H = 00H
D02L = 1EH

图 5-5. Baud Rate Set 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：同步检测结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.5 Baud Rate Set 命令**。

5.4 Chip Erase 命令

5.4.1 说明

这个命令被用来擦除 flash 存储器的整个内容。此外，只要擦除没有被安全设置禁止，由安全设置处理设置的所有信息都会被片擦除处理初始化（见 **5.12 Security Set 命令**）。

5.4.2 命令帧和状态帧

图 5-6 表示 Chip Erase 命令的命令帧的格式，图 5-7 表示这个命令的状态帧。

图 5-6. Chip Erase 命令帧（从编程器到 78K0R/Kx3）

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

图 5-7. Chip Erase 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：片擦除结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.6 Chip Erase 命令**。

5.5 Block Erase命令

5.5.1 说明

这个命令被用来擦除 flash 存储器中指定块的内容。

块可以通过擦除起始的块的第一个地址和擦除结束的最后一个地址来指定。可以指定连续多个块。

然而，如果擦除被安全设置禁止，擦除不能被执行（见 **5.12 Security Set 命令**）。

5.5.2 命令帧和状态帧

图 5-8 表示 Block Erase 命令的命令帧格式，图 5-9 表示这个命令的状态帧。

图 5-8. Block Erase 命令帧（从编程器到 78K0R/Kx3）

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

备注 SAH，SAM，SAL：块擦除起始地址（任意块的起始地址）

SAH：起始地址，高（位 23 到 16）

SAM：起始地址，中（位 15 到 8）

SAL：起始地址，低（位 7 到 0）

EAH，EAM，EAL：块擦除结束地址（任意块的最后一个地址）

EAH：结束地址，高（位 23 到 16）

EAM：结束地址，中（位 15 到 8）

EAL：结束地址，低（位 7 到 0）

图 5-9. Block Erase 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：块擦除结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.7 Block Erase 命令**。

5.6 Programming命令

5.6.1 说明

通过在发送写起始地址和写结束地址后写入数据，这个命令被用来写入用户程序到 flash 存储器中。在最后一个数据已经被发送并且写入完成后，内部校验被执行。

写起始/结束 地址只能被设置为块起始/结束地址。

如果在最后一个数据被发送后两个状态帧（ST1 和 ST2）都表示 ACK，78K0R/Kx3 固件自动执行内部校验。因此，这个内部校验的 Status 命令必须被发送。

5.6.2 命令帧和状态帧

图 5-10 表示 Programming 命令的命令帧格式，图 5-11 表示这个命令的状态帧。

图 5-10. Programming 命令帧（从编程器到 78K0R/Kx3）

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

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

图 5-11. Programming 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1（a）：命令接收结果

5.6.3 数据帧和状态帧

图 5-12 表示包含要被写入的数据的帧的格式，图 5-13 表示数据的状态帧。

图 5-12. 要被写入的数据帧（从编程器到 78K0R/Kx3）

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

备注 写入数据：要被写入的用户程序

图 5-13. 数据帧的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1（b）：数据接收检查结果
ST2（b）：写入结果

5.6.4 发送所有数据的完成和状态帧

图 5-14 表示在所有数据发送完成后的状态帧。

图 5-14. 发送所有数据完成后的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1 （c）：内部校验结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.8 Programming 命令**。

5.7 Verify命令

5.7.1 说明

这个命令被用来比较从编程器发送的数据和从 78K0R/Kx3（读取级别）指定地址范围读取的数据，并且检查它们是否匹配。

校验起始/结束地址只能被设置为块起始/结束地址。

5.7.2 命令帧和状态帧

图 5-15 表示 Verify 命令的命令帧格式，图 5-16 表示这个命令的状态帧。

图 5-15. Verify 命令帧（从编程器到 78K0R/Kx3）

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

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

图 5-16. Verify 命令的状态帧（从 78K0R/Kx3 到编程器）

STX	LEN	Data	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 5-17 表示包含要被校验的数据的帧的格式，图 5-18 表示数据的状态帧。

图 5-17. 要被校验的数据的数据帧（从编程器到 78K0R/Kx3）

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

备注 校验数据：要被校验的用户程序

图 5-18. 数据帧的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1（b）：数据接收检查结果
ST2（b）：校验结果[注]

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

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.9 Verify 命令**。

5.8 Block Blank Check命令

5.8.1 说明

这个命令被用来检查 flash 存储器中指定块号是否为空白（擦除状态）。

通过空白检查起始块的起始地址和空白检查结束块的最后一个地址，一个块可以被指定。连续多个块可以被指定。

5.8.2 命令帧和状态帧

图 5-19 表示 Block Blank Check 命令的命令帧格式，图 5-20 表示这个命令的状态帧。

图 5-19. Block Blank Check 命令帧（从编程器到 78K0R/Kx3）

SOH	LEN	COM	命令信息							SUM	ETX
01H	08H	32H （Block Blank Check）	SAH	SAM	SAL	EAH	EAM	EAL	D01	校验和	03H

备注 SAH，SAM，SAL：块空白检查起始地址（任意块的起始地址）

SAH：起始地址，高（位 23 到 16）

SAM：起始地址，中（位 15 到 8）

SAL：起始地址，低（位 7 到 0）

EAH，EAM，EAL：块空白检查结束地址（任意块的最后一个地址）

EAH：结束地址，高（位 23 到 16）

EAM：结束地址，中（位 15 到 8）

EAL：结束地址，中（位 15 到 8）

D01： 00H：当对一个单独块执行块空白检查时

01H：当对擦除芯片前的全部区域执行空白检查时

图 5-20. Block Blank Check 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：块空白检查结果

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.10 Block Blank Check 命令**。

5.9 Silicon Signature命令

5.9.1 说明

这个命令被用来读取信息，例如设备的写入协议信息（silicon 签名）和安全标志信息。

例如，如果编程器支持 78K0R/Kx3 不支持的编程协议，执行这个命令根据第二个和第三个字节的值来选择合适的协议。

5.9.2 命令帧和状态帧

图 5-21 表示 Silicon Signature 命令的命令帧格式，图 5-22 表示这个命令的状态帧。

图 5-21. Silicon Signature 命令帧（从编程器到 78K0R/Kx3）

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

图 5-22. Silicon Signature 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：命令接收结果

5.9.3 Silicon 签名数据帧

图 5-23 表示包含 silicon 签名数据的帧的格式

图 5-23. Silicon 签名数据帧（从 78K0R/Kx3 到编程器）

STX	LEN	数据						
02H	n	VEN	MET	MSC	DEC1	DEC2	UAE（3）	DEV（10）

数据 （续）						SUM	ETX
SCF	BOT	FSWSH	FSWSL	FSWEH	FSWEL	校验和	03H

备注 **1.** n （LEN）： 数据长度

VEN： 提供商码（NEC：10H）

MET： 宏扩展码

MSC： 宏功能码

DEC1： 设备扩展码 1

DEC2： 设备扩展码 2

UAE： 用户 flash ROM 最后一个地址（3 字节）

DEV： 设备名（10 字节）

SCF： 安全标志信息

BOT： 引导块号

FSWSH： flash 保护窗口（FSW）起始块的高 8 位端

FSWSL： flash 保护窗口（FSW）起始块的低 8 位端

FSWEH： flash 保护窗口（FSW）结束块的高 8 位端

FSWEL： flash 保护窗口（FSW）结束块的低 8 位端

2. 对于提供商码（VEN）、扩展码（MET）、功能码（MSC）、设备扩展码 1（DEC1）和设备扩展码 2（DEC2），低 7 位被用作数据实体，并且最高位被用作奇数奇偶。下面表示一个例子。

表 5-2. Silicon 签名数据的例子

区域	内容	长度（字节）	Silicon 签名数据的例子	实际值	奇偶
VEN	提供商码（NEC）	1	10H （00010000B ）	10H	附加
MET	宏扩展码	1	7FH （01111111B ）	7FH	附加
MSC	宏功能码	1	04H （01000000B ）	04H	附加
DEC1	设备扩展码 1	1	DCH （11011100B ）	DCH	附加
DEC2	设备扩展码 2	1	FDH （11111101B ）	FDH	附加
UAE	用户 flash ROM 最后一个地址	3	FFH （11111111B ）	00FFFFH	不附加
			FFH （11111111B ）		
			00H （00000000B ）		
DEV	设备名	10	44H （01000100B） = ‘D’	‘D’	不附加
			37H （00110111B） = ‘7’	‘7’	
			38H （00111000B） = ‘8’	‘8’	
			46H （01001111B） = ‘F’	‘F’	
			31H （00110001B） = ‘1’	‘1’	
			31H （00110001B） = ‘1’	‘1’	
			34H （00110100B） = ‘4’	‘4’	
			32H （00110010B） = ‘2’	‘2’	
			20H （00100000B） = ‘ ’	‘ ’	
			20H （00100000B） = ‘ ’	‘ ’	
SCF	安全标志信息	1	任意	与左边栏相同	不附加
BOT	引导块号（固定）	1	01H （00000001B）	01H	不附加
FSWS（H）	flash 保护窗口起始块的高 8 位端	1	任意	与左边栏相同	不附加
FSWS（L）	flash 保护窗口起始块的低 8 位端	1	任意	与左边栏相同	不附加
FSWE（H）	flash 保护窗口结束块的高 8 位端	1	任意	与左边栏相同	不附加
FSWE（L）	flash 保护窗口结束块的低 8 位端	1	任意	与左边栏相同	不附加

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.11 Silicon Signature 命令**。

5.9.4 78K0R/Kx3 silicon 签名列表

表 5-3. 78K0R/Kx3 Silicon 签名列表

项目	说明	长度（字节）	数据（十六进制）
提供商码	NEC	1	10
扩展码	扩展码	1	7F
功能码	功能码	1	04
设备信息	设备信息	2	DC
			FD
内部 flash ROM 最后一个地址	从地址的低字节发送	3	**注 1**
设备名（μPD）	78F1142/78F1152/78F1162 78F1143/78F1153/78F1163 78F1144/78F1154/78F1164 78F1145/78F1155/78F1165 78F1146/78F1156/78F1166 78F1167/78F1168	10	**注 2**
安全信息	安全信息	1	任意
引导块号	当前选择的引导串的最后一个块号	1	01
FSW 块号	FSW 信息	4	任意

注 1. 内部 flash ROM 最后一个地址的列表

项目	说明	长度（字节）	数据（十六进制）
内部 flash ROM 最后一个地址	64 KB （00FFFFH）	3	FFFF00
	96 KB （017FFFH）		FF7F01
	128 KB （01FFFFH）		FFFF01
	192 KB （02FFFFH）		FFFF02
	256 KB （03FFFFH）		FFFF03
	384 KB （05FFFFH）		FFFF05
	512 KB （07FFFFH）		FFFF07

（注 2 在下一页。）

注 2. 设备名列表如下。

设备名列表（1/4）

项目	说明	长度（字节）	实际值
设备名	D78F1142	10	44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 34 = ‘4’ 32 = ‘2’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1143		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 34 = ‘4’ 33 = ‘3’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1144		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 34 = ‘4’ 34 = ‘4’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1145		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 34 = ‘4’ 35 = ‘5’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1146		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 34 = ‘4’ 36 = ‘6’ 20 = ‘ ‘ 20 = ‘ ‘

设备名列表（2/4）

项目	说明	长度（字节）	实际值
设备名	D78F1152	10	44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 35 = ‘5’ 32 = ‘2’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1153		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 35 = ‘5’ 33 = ‘3’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1154		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 35 = ‘5’ 34 = ‘4’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1155		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 35 = ‘5’’ 35 = ‘5’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1156		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 35 = ‘5’ 36 = ‘6’ 20 = ‘ ‘ 20 = ‘ ‘

设备名列表（3/4）

项目	说明	长度（字节）	实际值
设备名	D78F1162	10	44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 32 = ‘2’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1163		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 33 = ‘3’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1164		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 34 = ‘4’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1165		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 35 = ‘5’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1166		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 36 = ‘6’ 20 = ‘ ‘ 20 = ‘ ‘

设备名列表（4/4）

项目	说明	长度（字节）	实际值
设备名	D78F1167	10	44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 37 = ‘7’ 20 = ‘ ‘ 20 = ‘ ‘
	D78F1168		44 = ‘D’ 37 = ‘7’ 38 = ‘8’ 46 = ‘F’ 31 = ‘1’ 31 = ‘1’ 36 = ‘6’ 38 = ‘8’ 20 = ‘ ‘ 20 = ‘ ‘

5.10 Version Get命令

5.10.1 说明

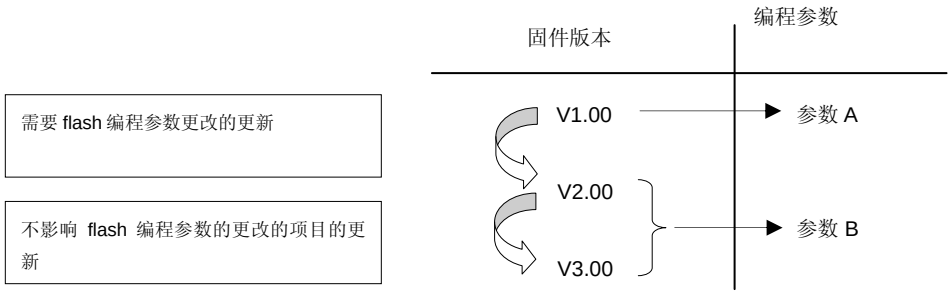
这个命令被用来获取 78K0R/Kx3 设备版本和固件版本信息。

设备版本值固定为 00H。

当编程参数必须根据 78K0R/Kx3 固件版本被更改时，使用这个命令。

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

例 固件版本和程序编程参数



5.10.2 命令帧和状态帧

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

图 5-24. Version Get 命令帧（从编程器到 78K0R/Kx3）

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

图 5-25. Version Get 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：命令接收结果

5.10.3 版本数据帧

图 5-26 表示版本数据的数据帧。

图 5-26. 版本数据帧（从 78K0R/Kx3 到编程器）

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

备注 DV1：设备版本的整数（固定为 00H）
DV2：设备版本的第一个小数位置（固定为 00H）
DV3：设备版本的第二个小数位置（固定为 00H）
FV1：固件版本的整数
FV2：固件版本的第一个小数位置
FV3：固件版本的第二个小数位置

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.12 Version Get 命令**。

5.11 Checksum命令

5.11.1 说明

这个命令被用来获取指定区域中的校验和数据。

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

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

5.11.2 命令帧和状态帧

图 5-27 表示 Checksum 命令的命令帧格式，图 5-28 表示这个命令的状态帧。

图 5-27. Checksum 命令帧（从编程器到 78K0R/Kx3）

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

备注 SAH，SAM，SAL：校验和计算起始地址

EAH，EAM，EAL：校验和计算结束地址

图 5-28. Checksum 命令的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1：命令接收结果

5.11.3 校验和数据帧

图 5-29 表示包含校验和数据的帧的格式。

图 5-29. 校验和数据帧（从 78K0R/Kx3 到编程器）

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

备注 CK1：校验和的高 8 位

CK2：校验和的低 8 位

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.13 Checksum 命令**。

5.12 Security Set命令

5.12.1 说明

这个命令被用来执行安全设置（写入的使能/使无效，块擦除，芯片擦除和引导块重写，以及 flash 保护窗口起始/结束块号的设置）。通过使用这个命令执行这些设置，未经授权的人对 flash 存储器的重新写入被限制，并且自编程的重写区域可以被指定。

注意事项 即使在安全设置后，其它的从使能到无效的设置也可以被执行。然而，从无效到使能的更改不可能。如果试图执行这样的设置，一个保护错误（10H）将发生。如果需要这样的设置，必须首先通过执行 Chip Erase 命令（Block Erase 命令不能被用来初始化安全标志）初始化所有安全标志。
然而，如果芯片擦除或引导块重写被使无效，芯片擦除本身不可能，所以设置不能从编程器擦除。由于这个编程器规范，建议在使芯片擦除无效前，再次确认安全设置的执行。

5.12.2 命令帧和状态帧

图 5-30 表示 Security Set 命令的命令帧格式，图 5-31 表示这个命令的状态帧。

图 5-30. Security Set 命令帧（从编程器到 78K0R/Kx3）

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

图 5-31. Security Set 命令的状态帧（从 78K0R/Kx3 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

5.12.3 数据帧和状态帧

图 5-32 表示安全数据帧的格式，图 5-33 表示数据的状态帧。

图 5-32. 安全数据帧（从编程器到 78K0R/Kx3）

STX	LEN	数据						SUM	ETX
02H	06H	FLG	BOT	FSWS（H）	FSWS（L）	FSWE（H）	FSWE（L）	校验和	03H

备注 **1.** FLG： 安全标志
BOT： 引导串最后一个块号（固定为 01H）
FSWS（H）： flash 保护窗口起始块号的高 8 位（固定为 00H）
FSWS（L）： flash 保护窗口起始块号的低 8 位
FSWE（H）： flash 保护窗口结束块号的高 8 位（固定为 00H）
FSWE（L）： flash 保护窗口结束块号的低 8 位

2. 如果 flash 保护窗口没有被设置，设置 FSWS 为 0000H,并且设置结束块为目标设备的结束块号。

图 5-33. 安全数据写入的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1 （b）：安全数据写入结果

5.12.4 内部校验检查和状态帧

图 5-34 表示内部校验检查的状态帧。

图 5-34. 内部校验检查的状态帧（从 78K0R/Kx3 到编程器）

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

备注 ST1 （c）：内部校验结果

下表表示安全标志区域的内容。

表 5-4. 安全标志区域的内容

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

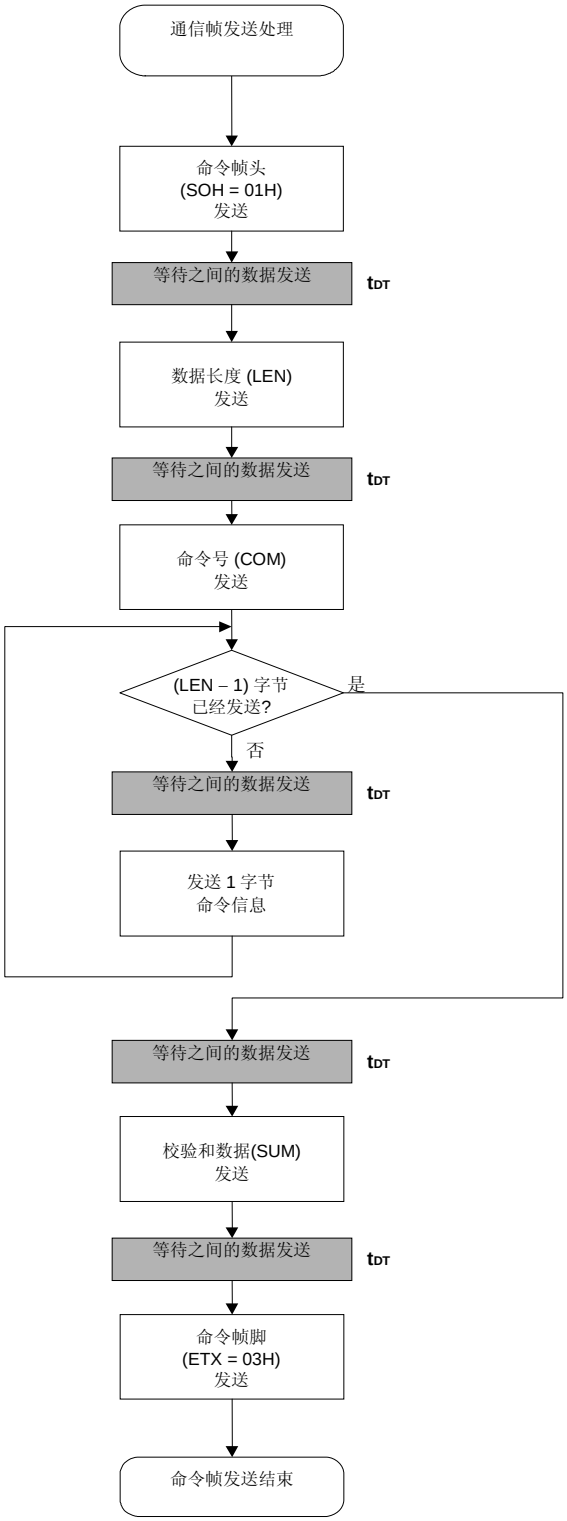
下表表示安全标志设置和每个操作的使能/使无效状态之间的关系。

表 5-5. 安全标志区域和每个操作的使能/使无效状态

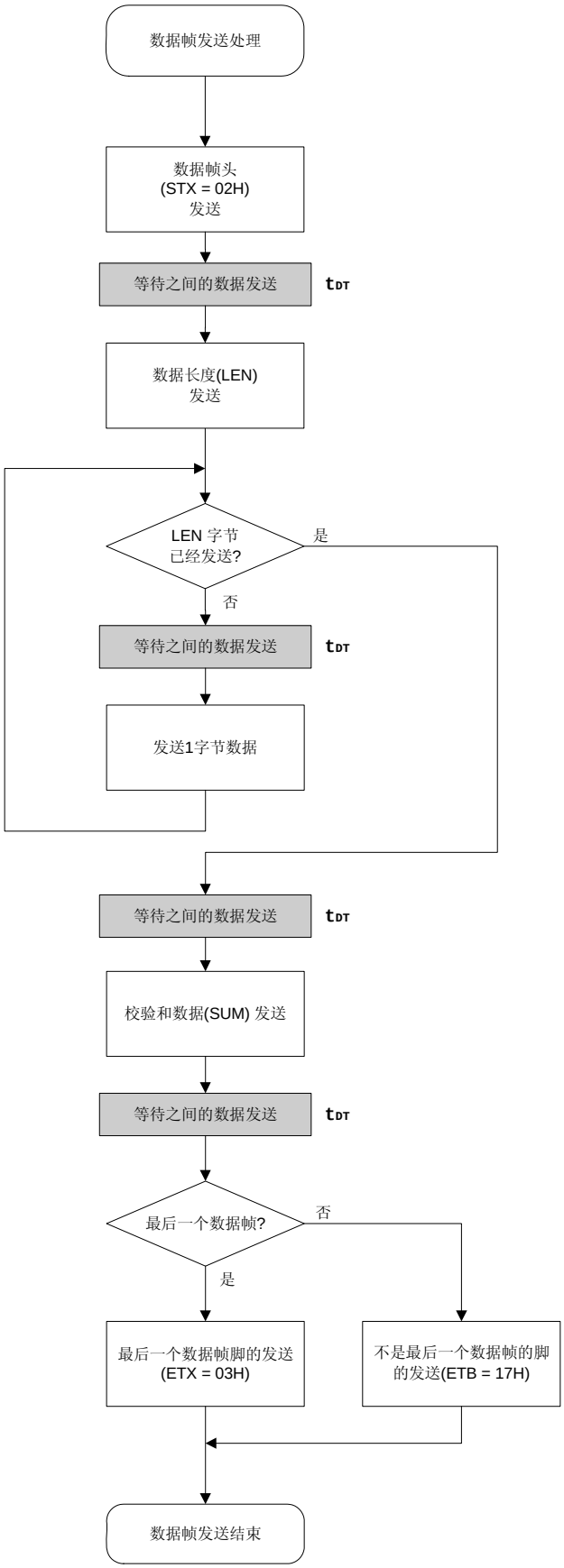
工作模式	Flash 存储器编程模式			自编程模式
命令 / 安全设置项目	安全设置后的命令操作 √：可以执行，×：不能执行 △：在引导区域写入和块擦除不可能			• 所有命令都可以被执行，而不管安全设置值 • 只有与安全设置值有关的值可能
	Programming	Chip Erase	Block Erase	
使编程无效	×	√	×	
使芯片擦除无效	√	×	×	
使块擦除无效	√	√	×	
引导块重写无效标志	△	×	△	与 flash 存储器编程模式的情况相同（板上/板外编程）

关于编程器和 78K0R/Kx3 之间的处理顺序的流程图、命令处理的流程图和样本程序的细节，请读 **6.14 Security Set 命令**。

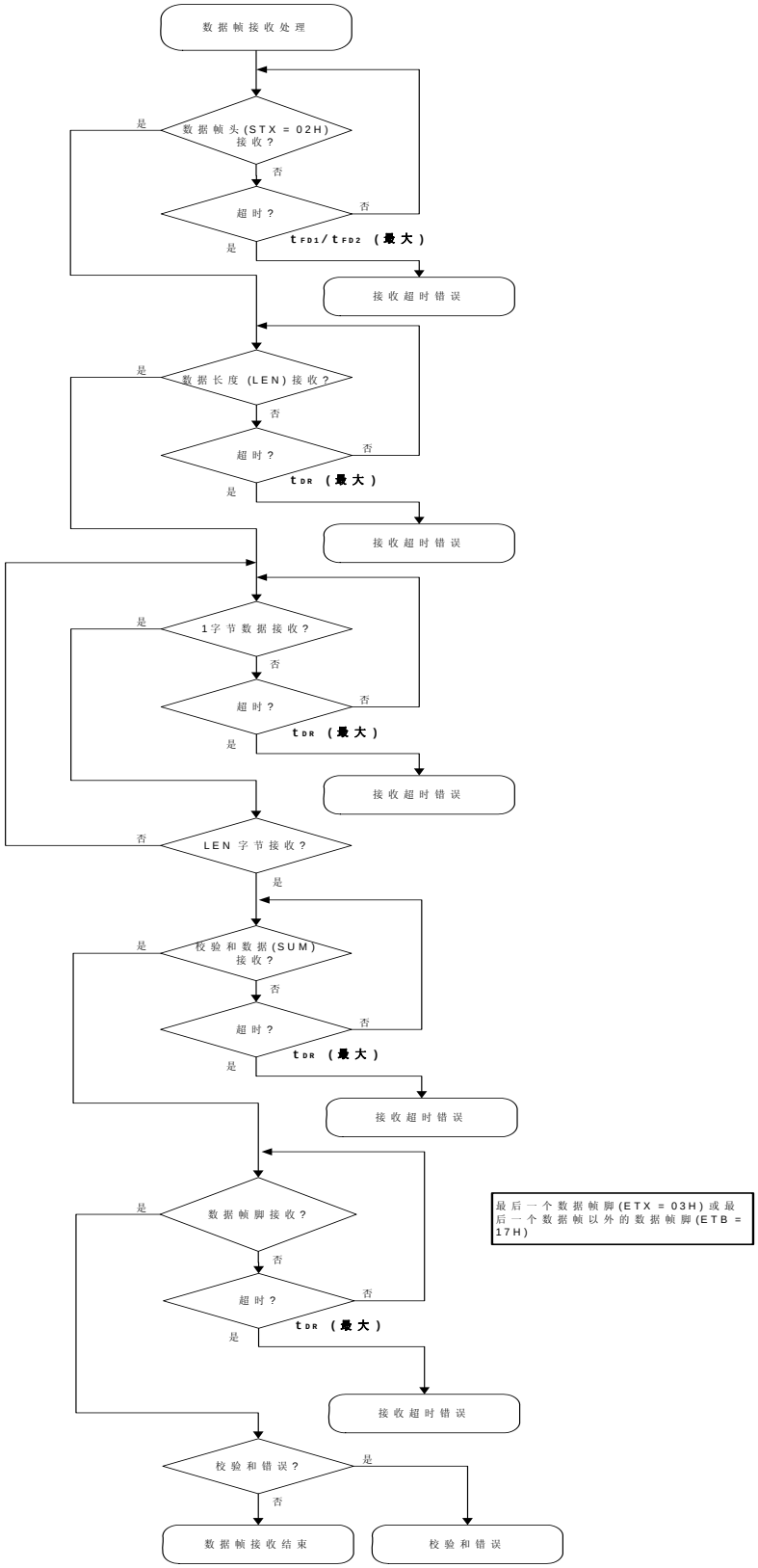
6.1 命令帧发送处理流程图



6.2 数据帧发送处理流程图



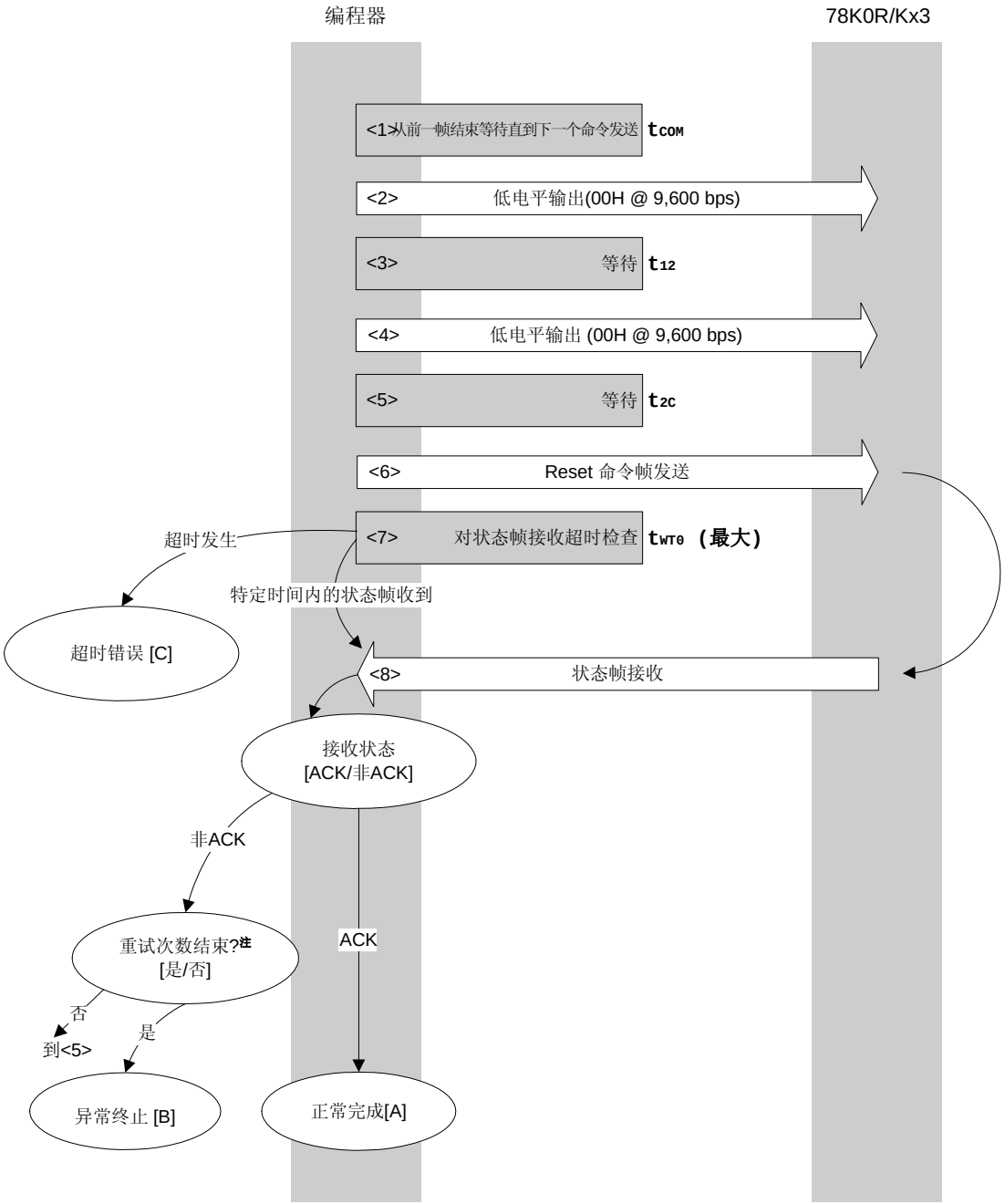
6.3 数据帧接收处理流程图



6.4 Reset 命令

6.4.1 处理顺序图

Reset 命令处理顺序



注 不要超过复位命令发送的重试次数（最多 16 次）。

6.4.2 处理顺序的说明

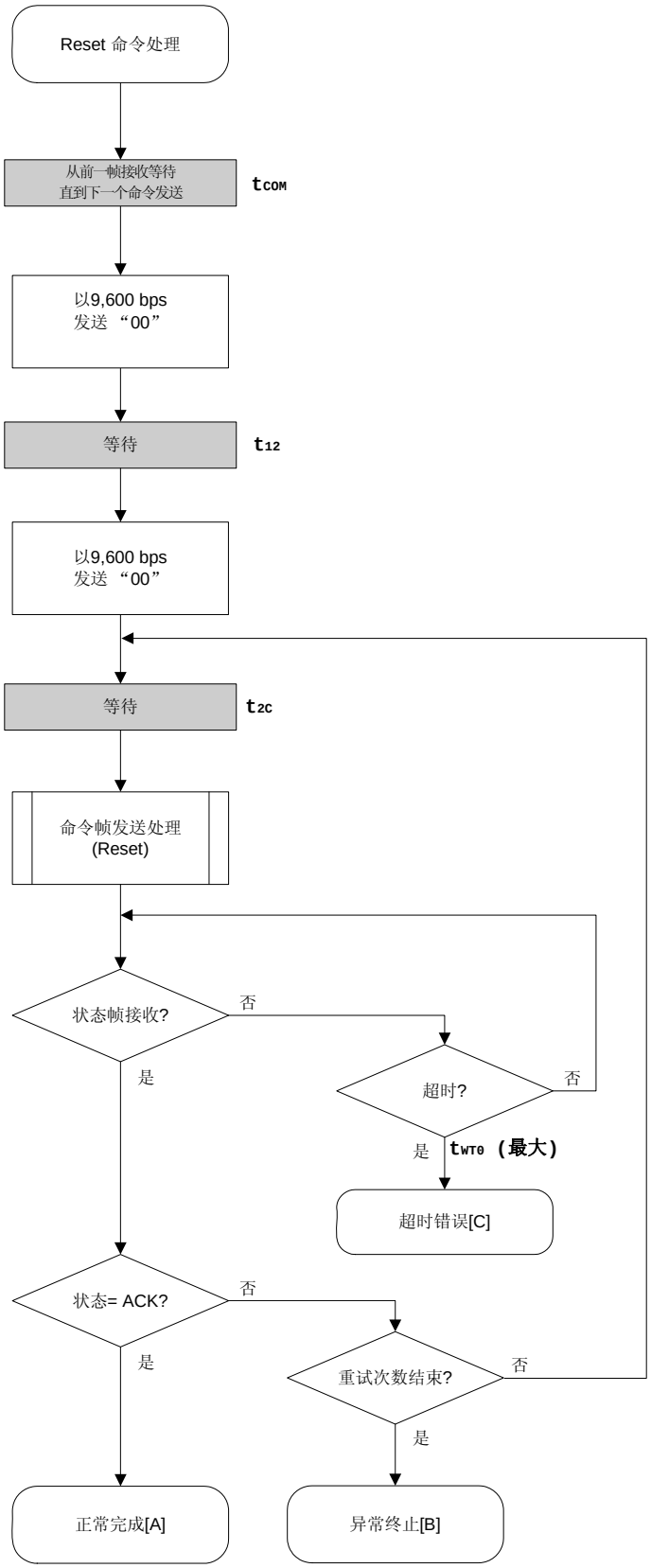
<1> 从前一帧接收等待直到下一个命令处理开始（等待时间 t_{COM}）。
<2> 低电平被输出（数据 00H 以 9,600 bps 被发送）。
<3> 等待状态（等待时间 t_{12}）。
<4> 低电平被输出（数据 00H 以 9,600 bps 被发送）。
<5> 等待状态（等待时间 t_{2C}）。
<6> Reset 命令通过命令帧发送处理被发送。
<7> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT0}（最大））。
<8> 检查状态码。

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

6.4.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且编程器和 78K0R/Kx3 之间的同步已经被建立。
异常终止[B]	校验和错误	07H	发送命令帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]		–	状态帧在特定时间内没有被接收。

6.4.4 流程图


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

6.4.5 样本程序

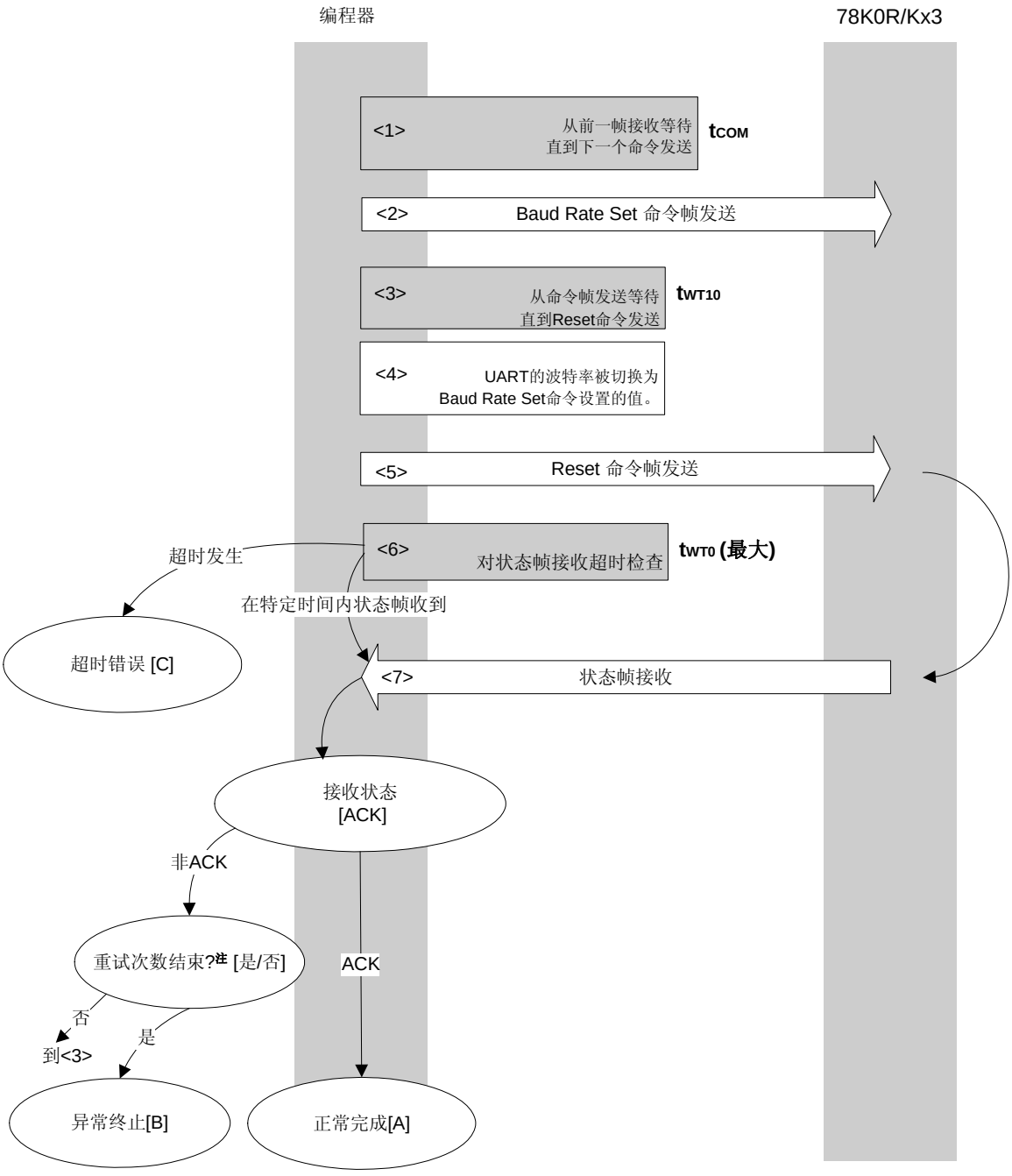
以下表示 Reset 命令处理的一个样本程序。

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

        set_uart0_br(BR_9600);    // 更改为9600bps

        fl_wait(tCOM);            // 等待

        set_ua_dir_tx();          // 更改单线UART发送模式
        putc_ua(0x00);            // 发送 0x00 @ 9600bps

        fl_wait(t12);                     // 等待

        putc_ua(0x00);            // 发送 0x00 @ 9600bps
        set_ua_dir_rx();          // 更改单线UART接收模式

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

                fl_wait(t2C);             // 等待

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

                rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
                if (rc == FLC_DFTO_ERR)     // 超时. ?
                        break;                    // 是   // 如果 [C]
                if (rc == FLC_ACK){               // ACK ?
                        break;                    // 是   // 如果 [A]
                }
                else{
                        NOP();
                }
                //continue;                       // 如果 [B] （如果从循环退出)
        }
//      switch(rc) {
//
//              case    FLC_NO_ERR:     return rc;          break;    // 如果 [A]
//              case    FLC_DFTO_ERR:           return rc;          break;    // 如果 [C]
//              default:            return rc;          break;    // 如果 [B]
//      }
        return rc;
}
```

6.5 Baud Rate Set 命令

6.5.1 处理顺序图

Baud Rate Set 命令处理顺序



注 不要超过复位命令发送的重试次数（最多 16 次）。

6.5.2 处理顺序的说明

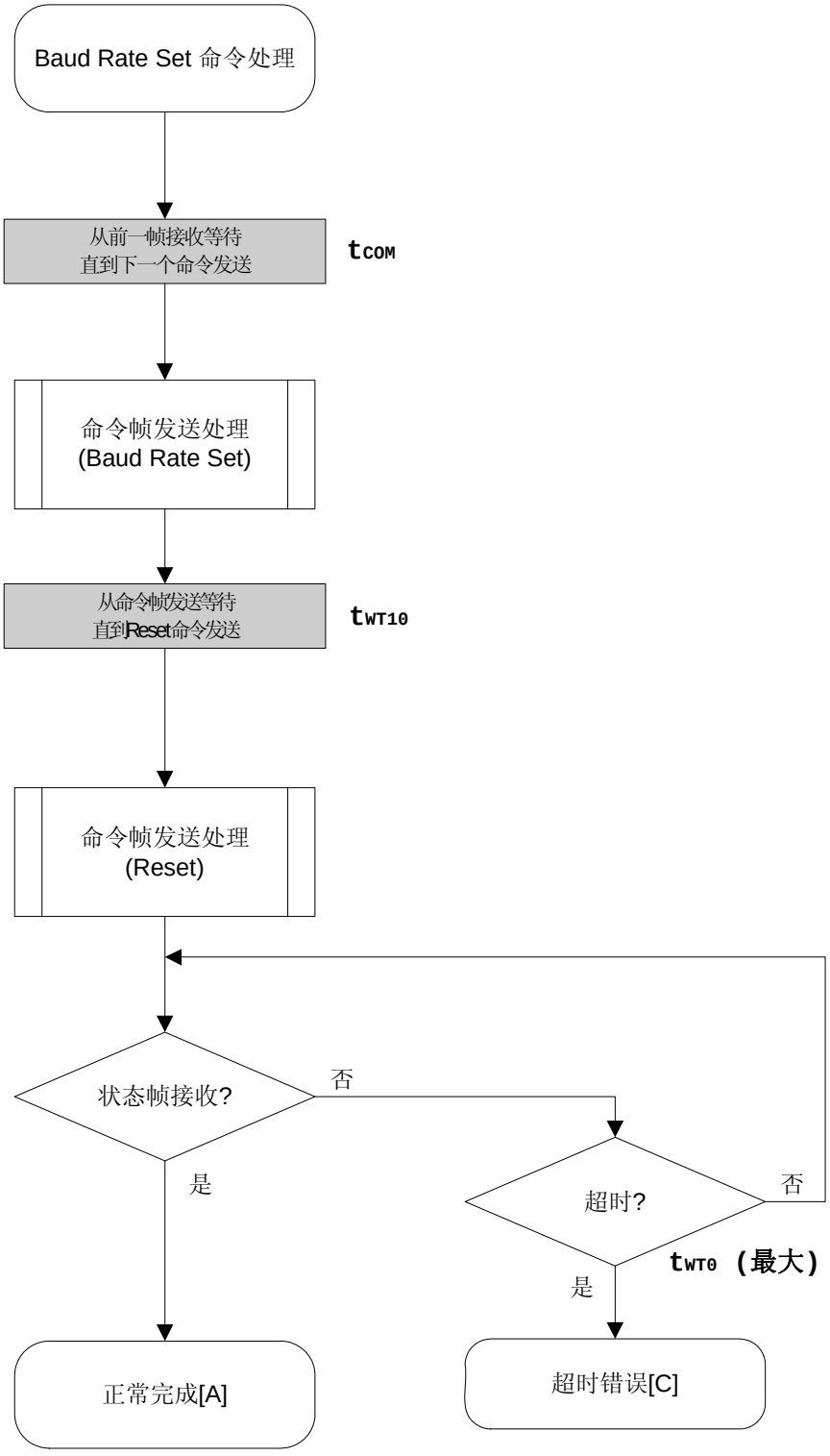
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> Baud Rate Set 命令通过命令帧发送处理被发送。
<3> 从命令发送等待直到 Reset 命令发送（等待时间 t_{WT10}）。
<4> UART 通信的波特率被切换为 Baud Rate Set 命令设置的值。
<5> Reset 命令通过命令帧发送处理被发送。
<6> 从命令发送直到状态帧接收，执行超时检查。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT0}（最大））。
<7> 因为状态码应该是 ACK，处理正常结束[A]。

6.5.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且编程器和 78K0R/Kx3 之间的 UART 通信速度的同步已经被建立。
异常终止[B]	校验和错误	07H	发送命令帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]注		–	数据帧接收超时。对于 78K0R/Kx3，在以下情况下，这个命令也会导致错误。 • 命令信息（D01，D02H，D02L，D03）无效 • 命令帧包含校验和错误 • 命令帧的数据长度（LEN）无效 • 命令帧的脚（ETX）丢失 • 在设置波特率并且接收命令帧数据 16 次后，Reset 命令没有被检测到。

注 如果应该超时错误已经发生，执行硬件复位并重新设置为 flash 存储器编程模式。

6.5.4 流程图


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

6.5.5 样本程序

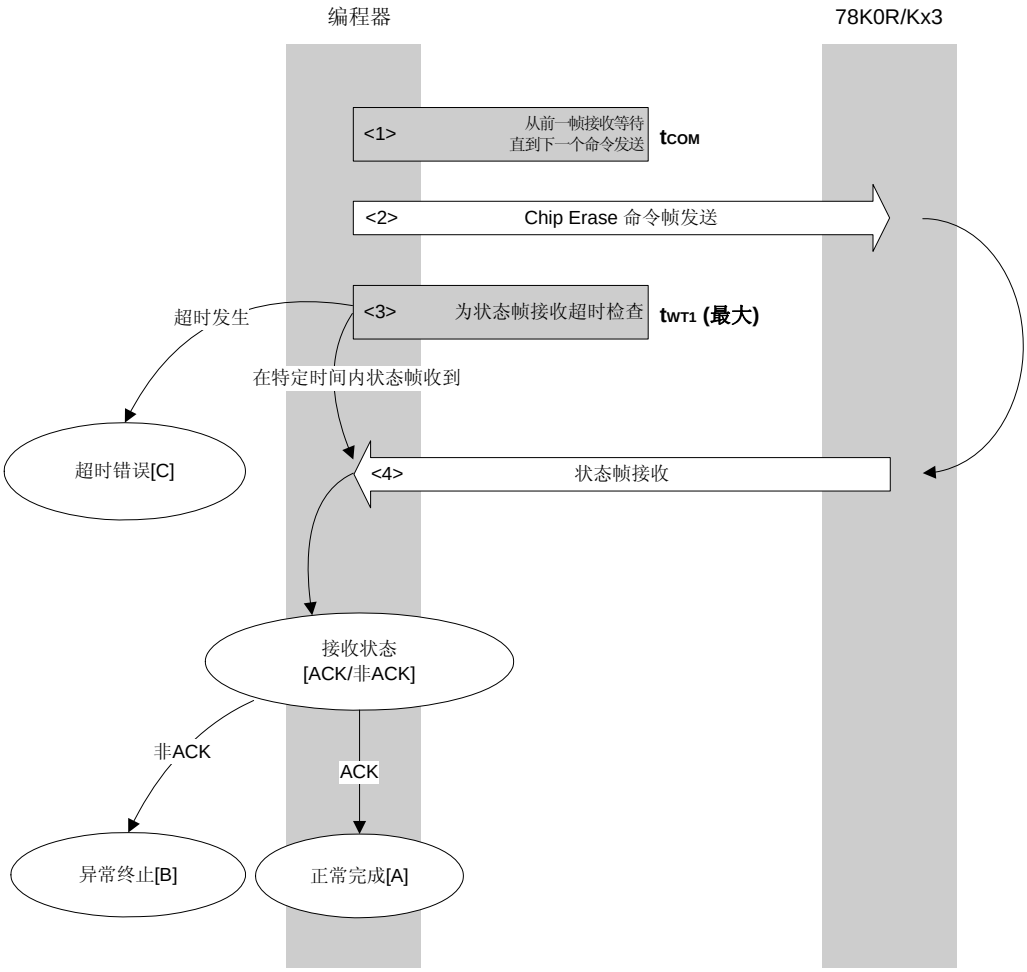
以下表示 Baud Rate Set 命令处理的一个样本程序。

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

        fl_cmd_prm[0] = 0x00;       // “D01”：由目标设备调整（115200bps）
        fl_cmd_prm[1] = 0x00;       // “D02”：由目标设备调整（115200bps）
        fl_cmd_prm[2] = 0x0a;       // “D03”：（固定值）
        fl_cmd_prm[3] = 0x01;       // “D04”：噪声滤波器开

        fl_wait(tCOM);              // 在发送命令前等待
        put_cmd_ua(FL_COM_SET_BAUDRATE, 1+4, fl_cmd_prm);// 发送“Baudrate Set”命令
        set_flbaud(brid);           // 更改波特率
        set_uart0_br(brid);                 // 更改波特率（硬件）

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

                put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);// 发送 RESET 命令
                rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);    // 获取状态帧
                if (rc){
                        if (retry--)
                                continue;
                        else
                                return  rc;
                }
                break;              // 到 ACK !!

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

        return rc;
}
```

6.6 Chip Erase 命令

6.6.1 处理顺序图

Chip Erase 命令处理顺序



6.6.2 处理顺序的说明

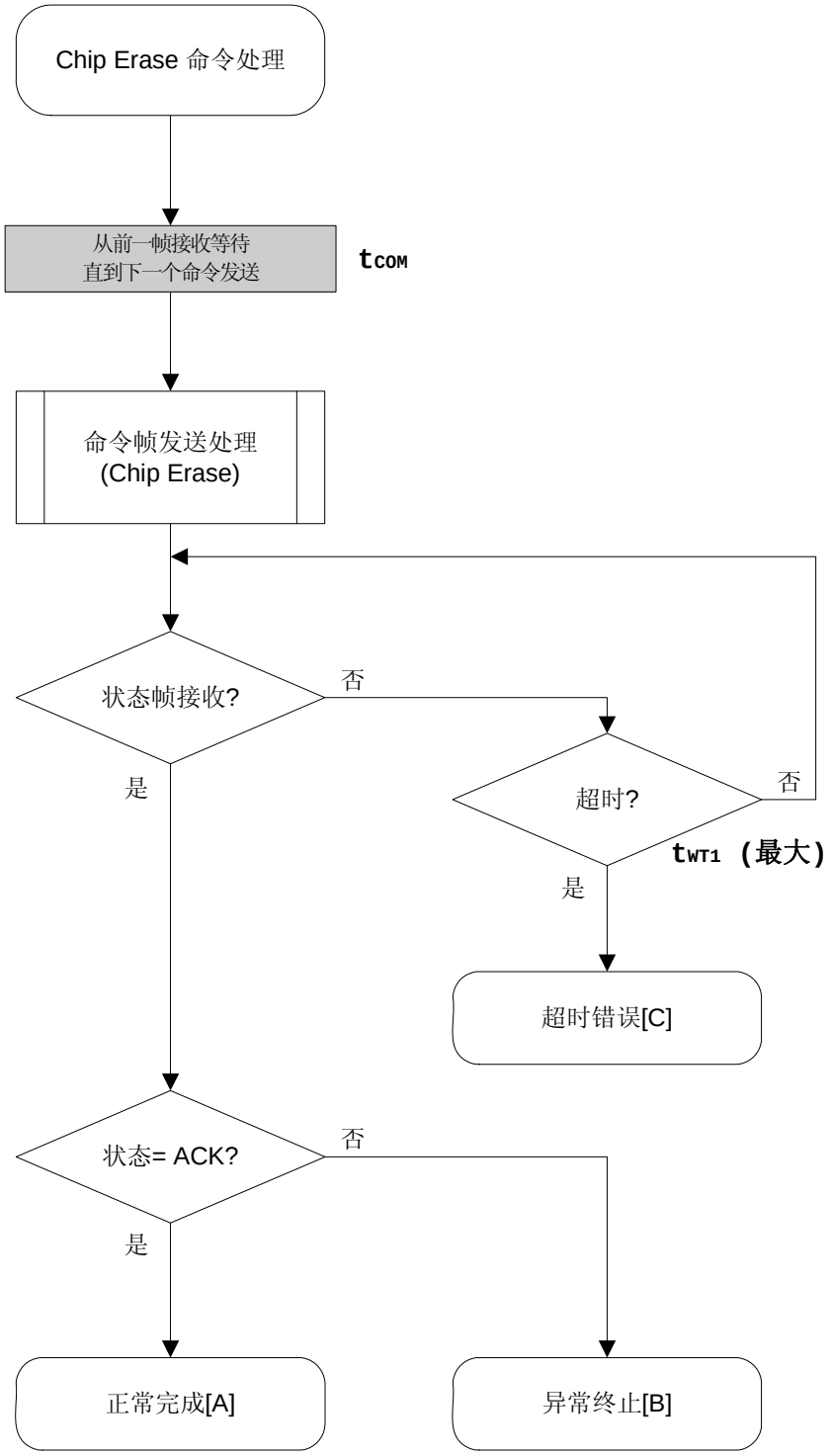
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> Chip Erase 命令通过命令帧发送处理被发送。
<3> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT1}（最大））。
<4> 检查状态码。

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

6.6.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且正常完成芯片擦除。
异常终止[B]	校验和错误	07H	发送命令帧的校验和不匹配。
	产品错误	10H	芯片擦除或引导块重写在安全设置中被禁止。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	MRG10 错误	1AH	发生擦除错误。
	MRG11 错误	1BH	
	写入错误	1CH	
超时错误[C]		–	状态帧在特定时间内没有被接收。

6.6.4 流程图


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

6.6.5 样本程序

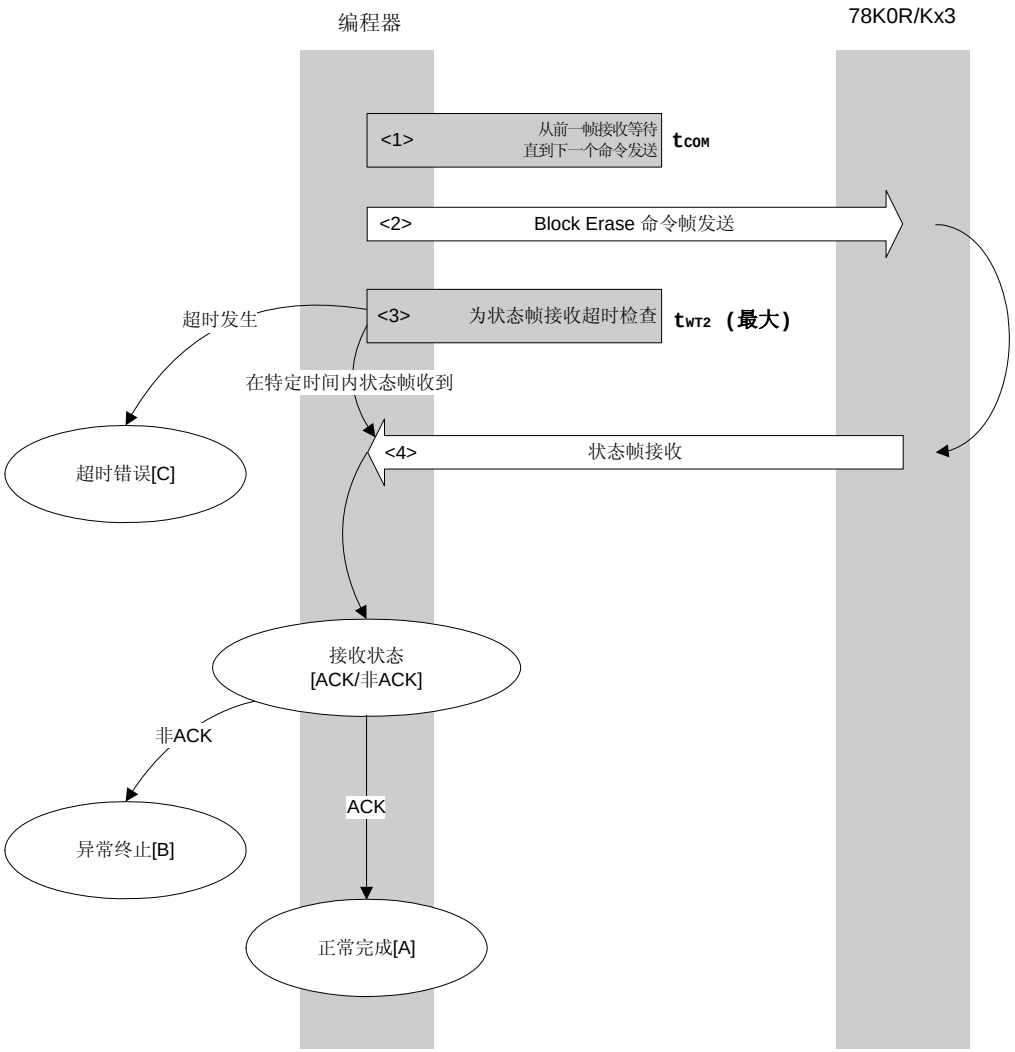
以下表示 Chip Erase 命令处理的一个样本程序。

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


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

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

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

}
```

6.7 Block Erase 命令

6.7.1 处理顺序图

Block Erase 命令处理顺序



6.7.2 处理顺序的说明

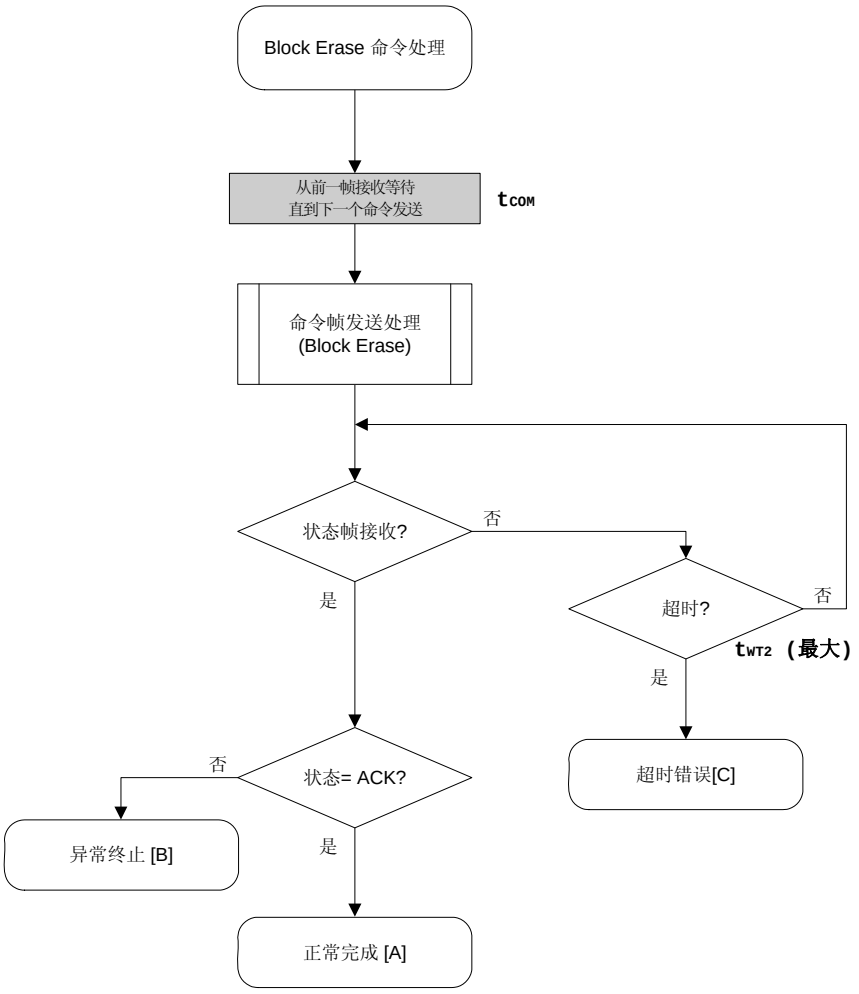
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> Block Erase 命令通过命令帧发送处理被发送。
<3> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT2}（最大））。
<4> 状态码被检查。

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

6.7.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且块擦除被正常完成。
异常终止[B]	参数错误	05H	指定的结束地址超出 flash 存储器范围，或者指定的起始/结束地址不是块的第一个/结束地址。
	校验和错误	07H	发送命令帧的校验和不匹配。
	产品错误	10H	写入、块擦除或芯片擦除在安全设置中被禁止。一个引导块被包含在指定范围内，并且引导块重写被禁止。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	MRG10 错误	1AH	一个擦除错误发生。
超时错误[C]		–	状态帧在特定时间内没有被接收。

6.7.4 流程图


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

6.7.5 样本程序

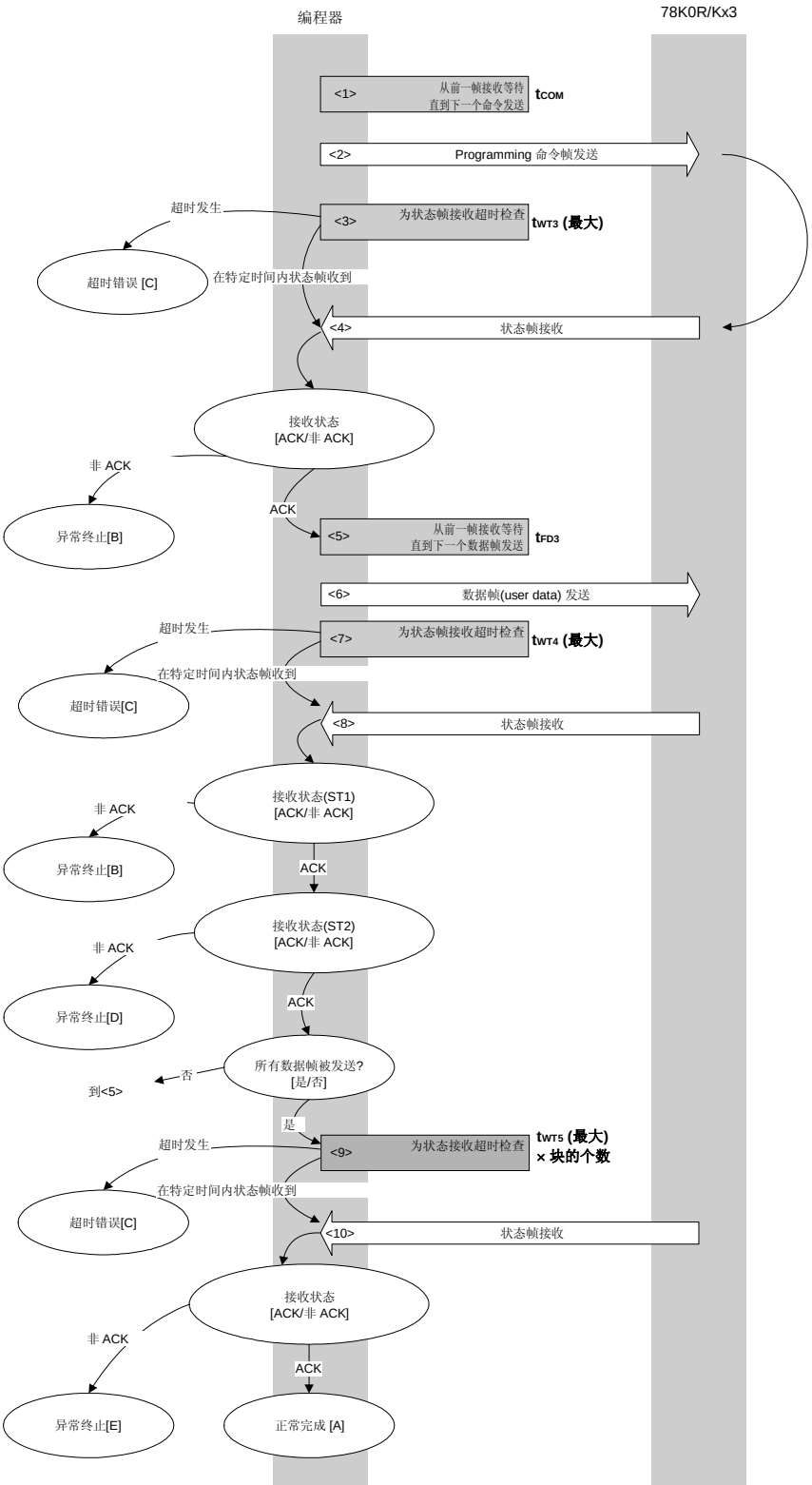
以下表示 Block Erase 命令处理的一个样本程序。

```
/*****************************************************************/
/*                                                               */
/*      块擦除命令                                               */
/*                                                               */
/*****************************************************************/
/*      [i] u8 block                ... 块号                     */
/*      [r] u16             ... 错误码                           */
/*****************************************************************/
u16             fl_ua_erase_blk(u16 sblk, u16 eblk)
{

        u16     rc;
        u32     wt2_max;
        u32     top, bottom;

        top = get_top_addr(sblk);                       // 获得起始块的起始地址
        bottom = get_bottom_addr(eblk);                 //获得结束块的结束地址

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

        wt2_max = make_wt2_max(sblk, eblk);

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

        put_cmd_ua(FL_COM_ERASE_BLOCK, 1+6, fl_cmd_prm); // 发送 ERASE CHIP 命令

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

        return rc;
}
```

6.8 Programming 命令

6.8.1 处理顺序图

Programming 命令处理顺序



6.8.2 处理顺序的说明

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

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

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

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

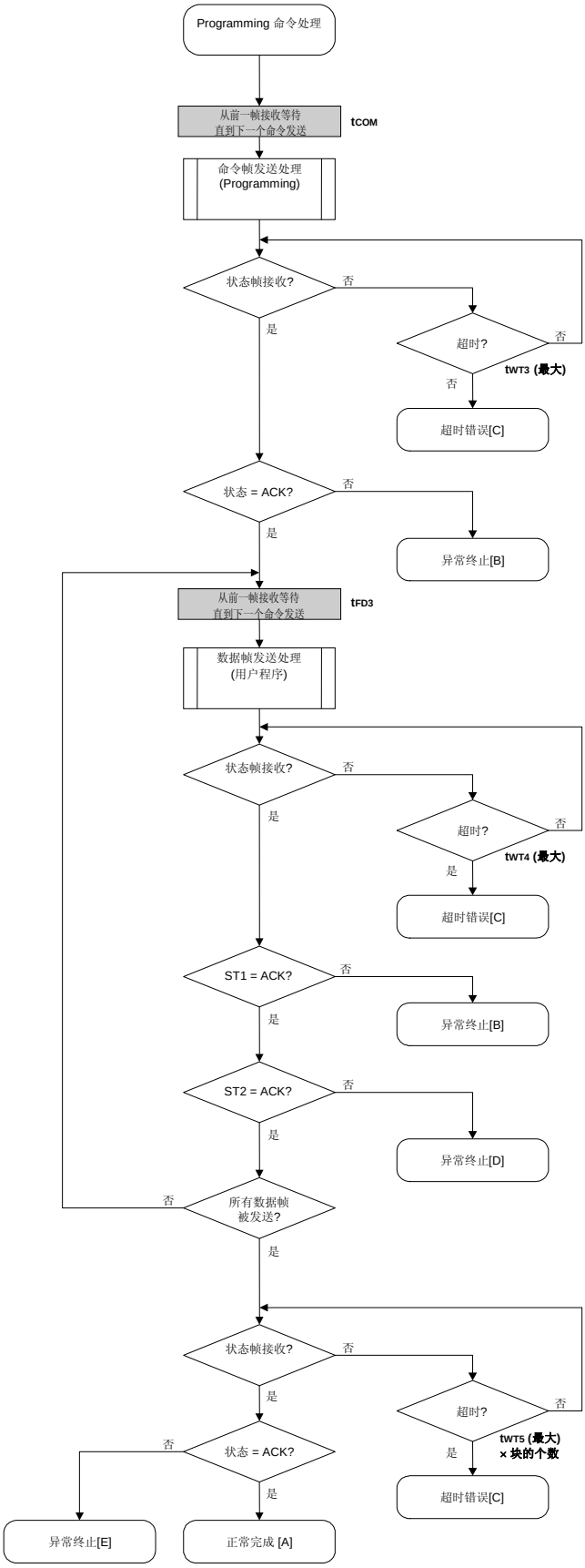
<9> 一个超时检查被执行直到状态帧接收。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT5}（最大）× 块的个数）。
<10> 状态码被检查。

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

6.8.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且用户数据被正常写入。
异常终止[B]	参数错误	05H	起始/结束地址超出 flash 存储器范围，指定的起始/结束地址不是块的第一个/结束地址，或者写入起始地址比结束地址大。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	写入在安全设置中被禁止。一个引导块被包含在指定范围内，并且引导块重写被禁止。
	消极响应（NACK）	15H	命令帧数据或数据帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]		–	状态帧在特定时间内没有被接收。
异常终止[D]，[E]	MRG10 错误	1AH	一个写入错误发生。
	MRG11 错误	1BH	
	写入错误	1CH	

6.8.4 流程图


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

6.8.5 样本程序

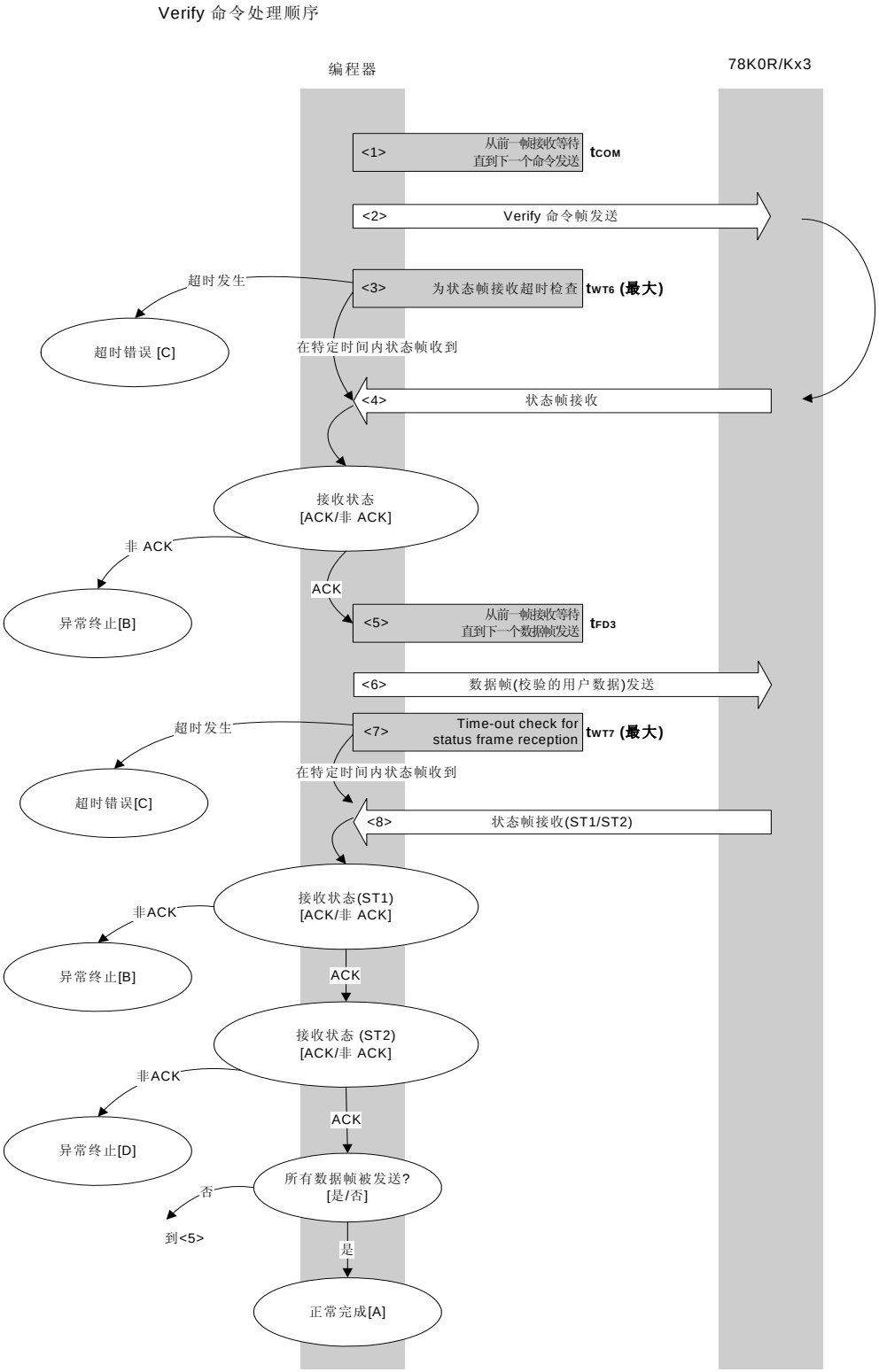
以下表示 Programming 命令处理的一个样本程序。

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

#define                 fl_st2_ua           (fl_ua_sfrm[OFS_STA_PLD+1])

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


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

        /*************************************************/
        /*          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);                             // 在发送命令前等待

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

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

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

        while(1){

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

```
                }
                else{
                        is_end = true;
                        send_size = bottom - send_head + 1;          // 发送大小 = (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);                                      // 在发送数据帧前等待

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

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

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

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

        return rc;
}
```

6.9 Verify 命令

6.9.1 处理顺序图



6.9.2 处理顺序的说明

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

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

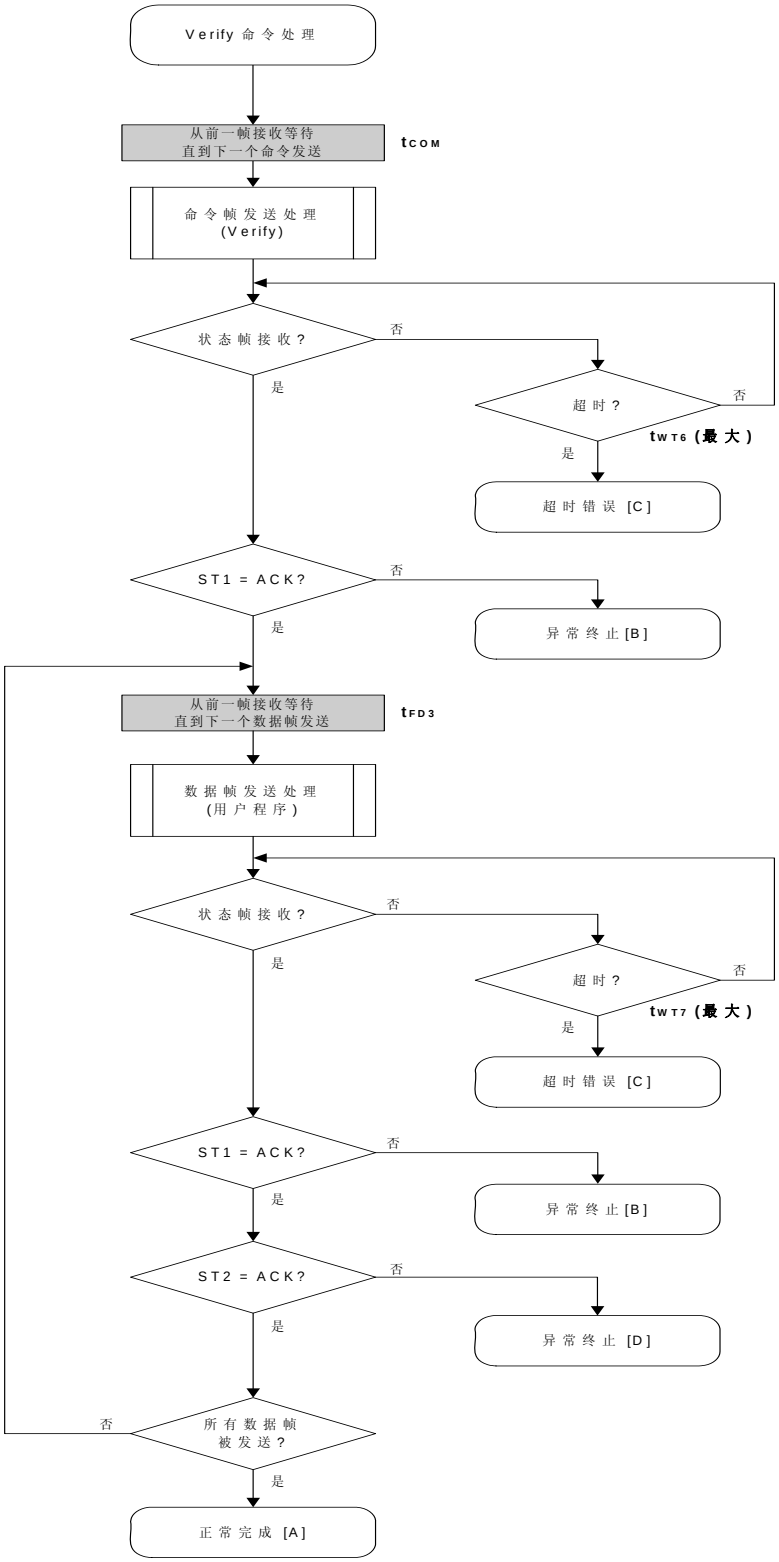
<5> 从前一帧接收等待直到下一个数据帧发送（等待时间 t_{FD3}）。
<6> 校验用的用户数据通过数据帧发送处理被发送。
<7> 从用户数据发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT7}（最大））。
<8> 状态码（ST1/ST2）被检查（参考处理顺序和流程图）。

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

6.9.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且校验被正常完成。
异常终止[B]	参数错误	05H	起始/结束地址超出 flash 存储器范围，指定的起始/结束地址不是块的第一个/结束地址，或者写入起始地址比结束地址大。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据或数据帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]		–	状态帧在特定时间内没有被接收。
异常终止[D]	校验错误	0FH (ST2)	一个校验错误发生。

6.9.4 流程图


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

6.9.5 样本程序

以下表示 Verify 命令处理的一个样本程序。

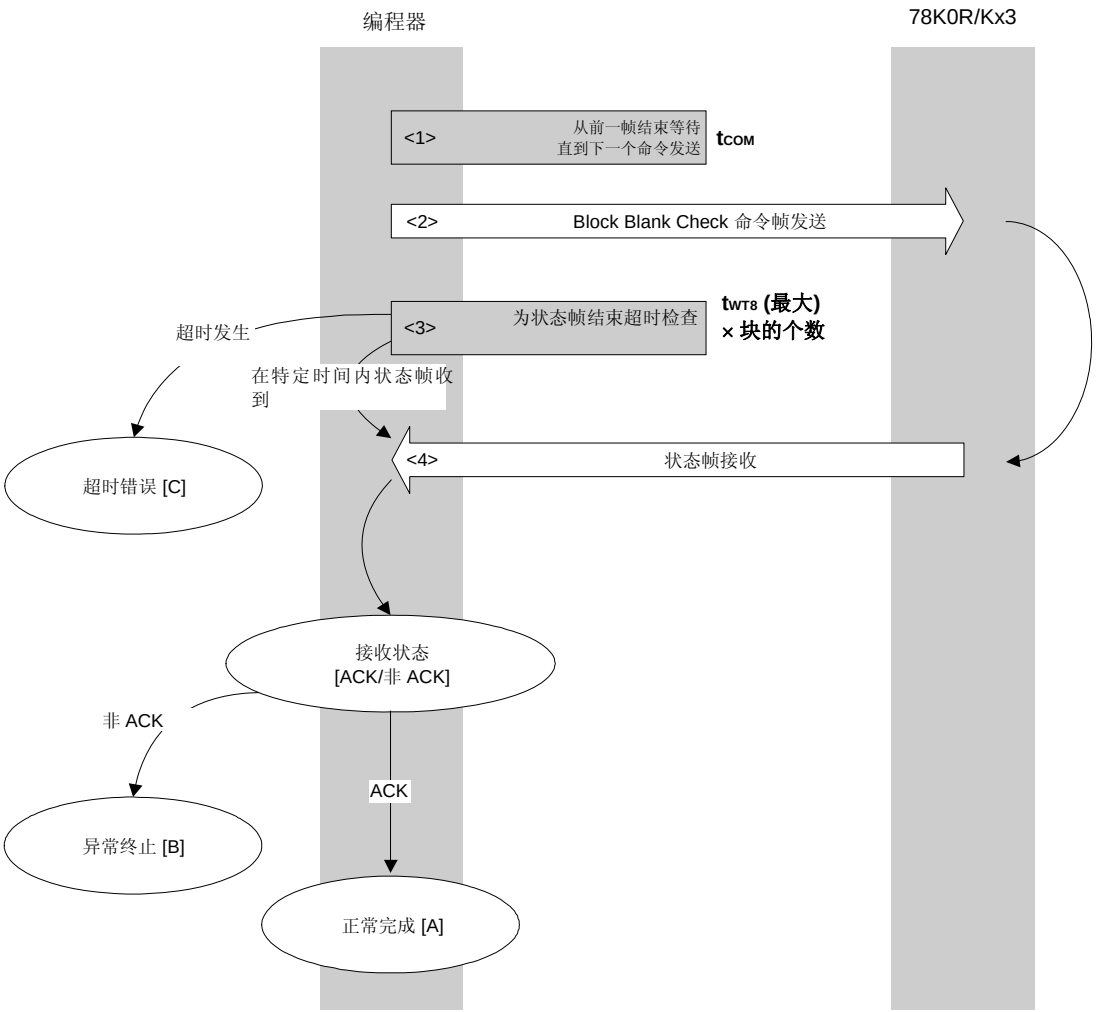
```
/*****************************************************************/
/*                                                               */
/*      校验命令                                                 */
/*                                                               */
/*****************************************************************/
/*      [i] u32 top             ... 起始地址                     */
/*      [i] u32 bottom      ... 结束地址                         */
/*      [r] u16             ... 错误码                           */
/*****************************************************************/
u16             fl_ua_verify(u32 top, u32 bottom, u8 *buf)
{
        u16     rc;
        u32     send_head, send_size;
        bool    is_end;

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


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

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

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

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

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

        while(1){

                // 发送数据帧
                if ((bottom - send_head) > 256){        // 剩余大小 > 256 ?
                        is_end = false;                 // 是，不是最后一帧
                        send_size  = 256;               // 发送大小 = 256 字节
                }
                else{
                        is_end = true;
                        send_size = bottom - send_head + 1;// 发送大小 = (bottom - send_head)+1 字节

                }
```

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

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

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

6.10 Block Blank Check 命令

6.10.1 处理顺序图

Block Blank Check 命令处理顺序



6.10.2 处理顺序的说明

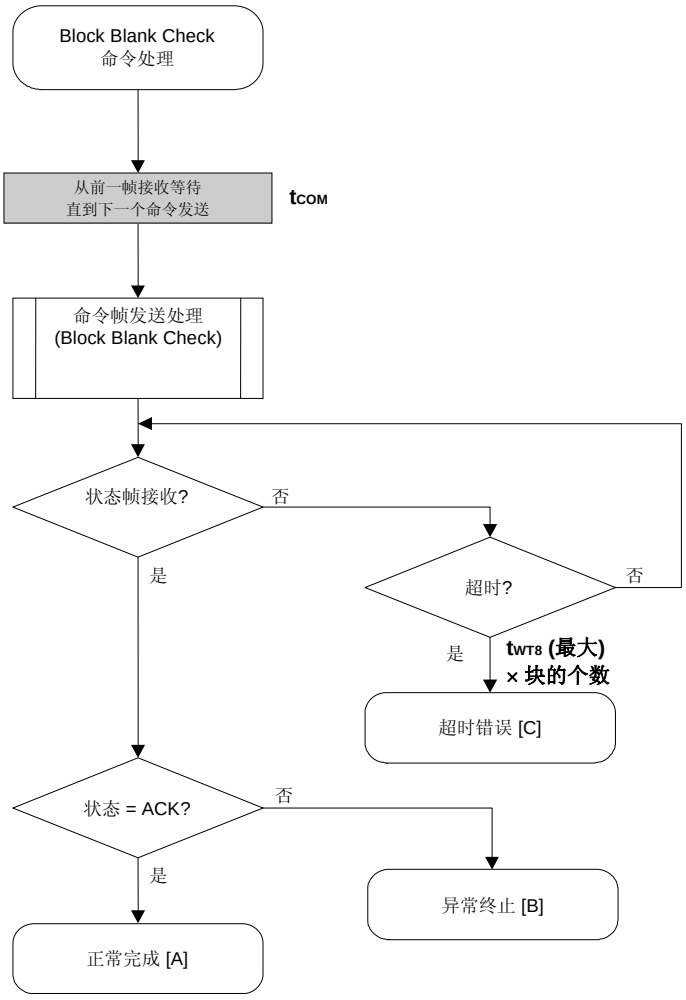
<1> 从前一帧接收等待直到下一个命令发送（等待时间 t_{COM}）。
<2> Block Blank Check 命令通过命令帧发送处理被发送。
<3> 从命令发送直到状态帧接收，超时检查被执行。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT8}（最大）× 块的个数）。
<4> 状态码被检查。

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

6.10.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且块空白检查被正常执行。
异常终止[B]	参数错误	05H	结束地址超出 flash 存储器范围，指定的起始/结束地址不是块的第一个/结束地址，或者参数 D01 的值不是 00H 或 01H。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
	MRG11 错误	1BH	指定块的 flash 存储器不是空白。
超时错误[C]		–	状态帧在特定时间内没有被接收。

6.10.4 流程图


Block Blank Check
命令处理
从前一帧接收等待
直到下一个命令发送
tCOM
命令帧发送处理
(Block Blank Check)
状态帧接收?
否
是
超时?
否
是
tWT8 (最大)
× 块的个数
超时错误 [C]
状态 = ACK?
否
是
异常终止 [B]
正常完成 [A]

6.10.5 样本程序

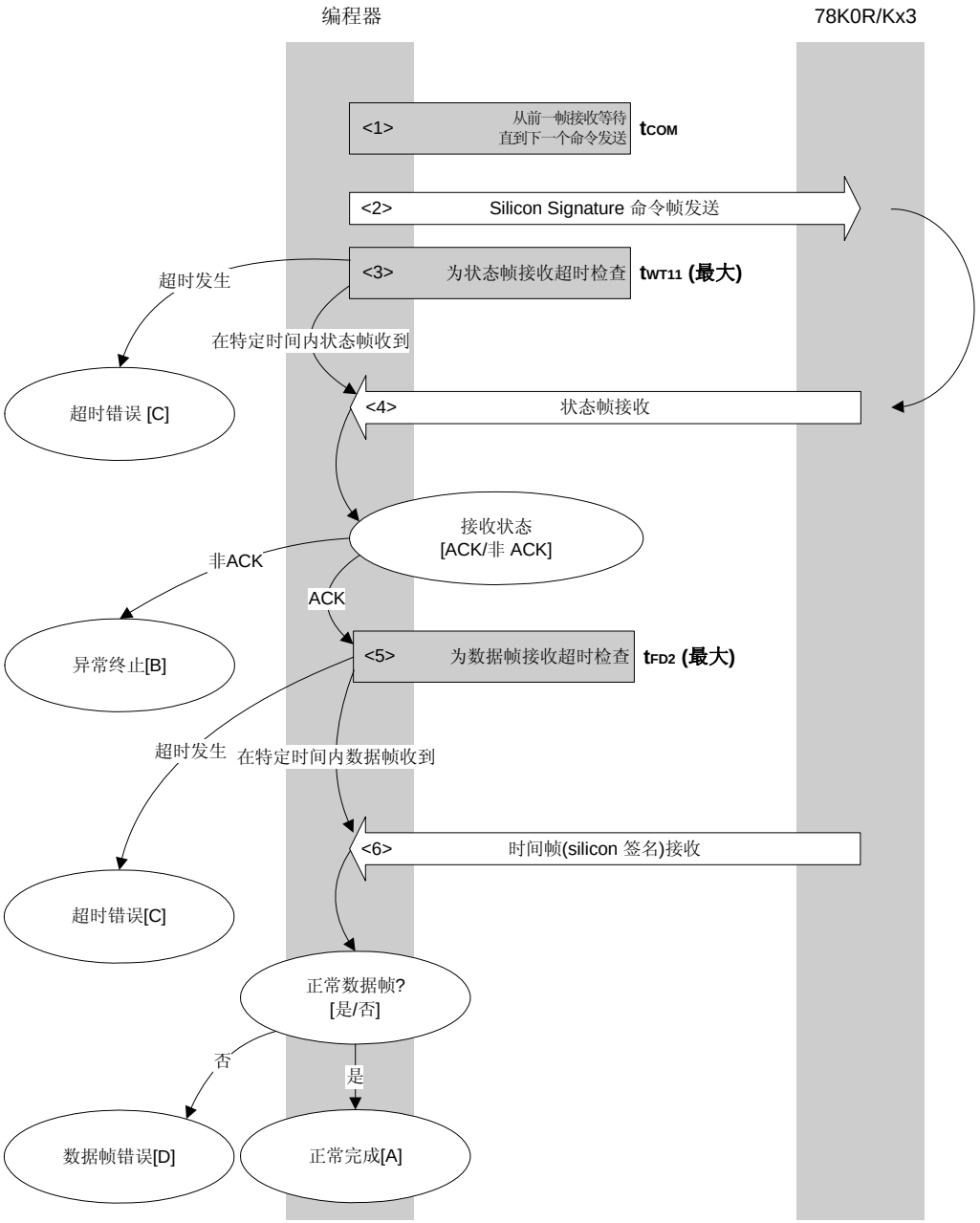
以下表示 Block Blank Check 命令处理的一个样本程序。

```
/*****************************************************************/
/*                                                               */
/* 块空白检查命令                                                */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... 空白检查的顶端地址                        */
/* [i] u32 bottom          ... 空白检查的底部地址                */
/* [i]  u8 whole  ... <1>检查 w/NON 用户 flash                   */
/*                        <0>只检查用户 flash                    */
/* [r] u16                 ... 错误码                                    */
/*****************************************************************/
u16             fl_ua_blk_blank_chk(u32 top, u32 bottom, u8 whole)
{
        u16     rc;
        u16     block_num;

        set_range_prm(fl_cmd_prm, top, bottom);     // 设置 SAH/SAM/SAL，EAH/EAM/EAL
        block_num = get_block_num(top, bottom);     // 获得块号
        fl_cmd_prm[6] = whole;                      // 是否只检查用户区域

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

        put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7+1, fl_cmd_prm);


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

6.11 Silicon Signature 命令

6.11.1 处理顺序图

Silicon Signature 命令处理顺序



6.11.2 处理顺序的说明

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

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

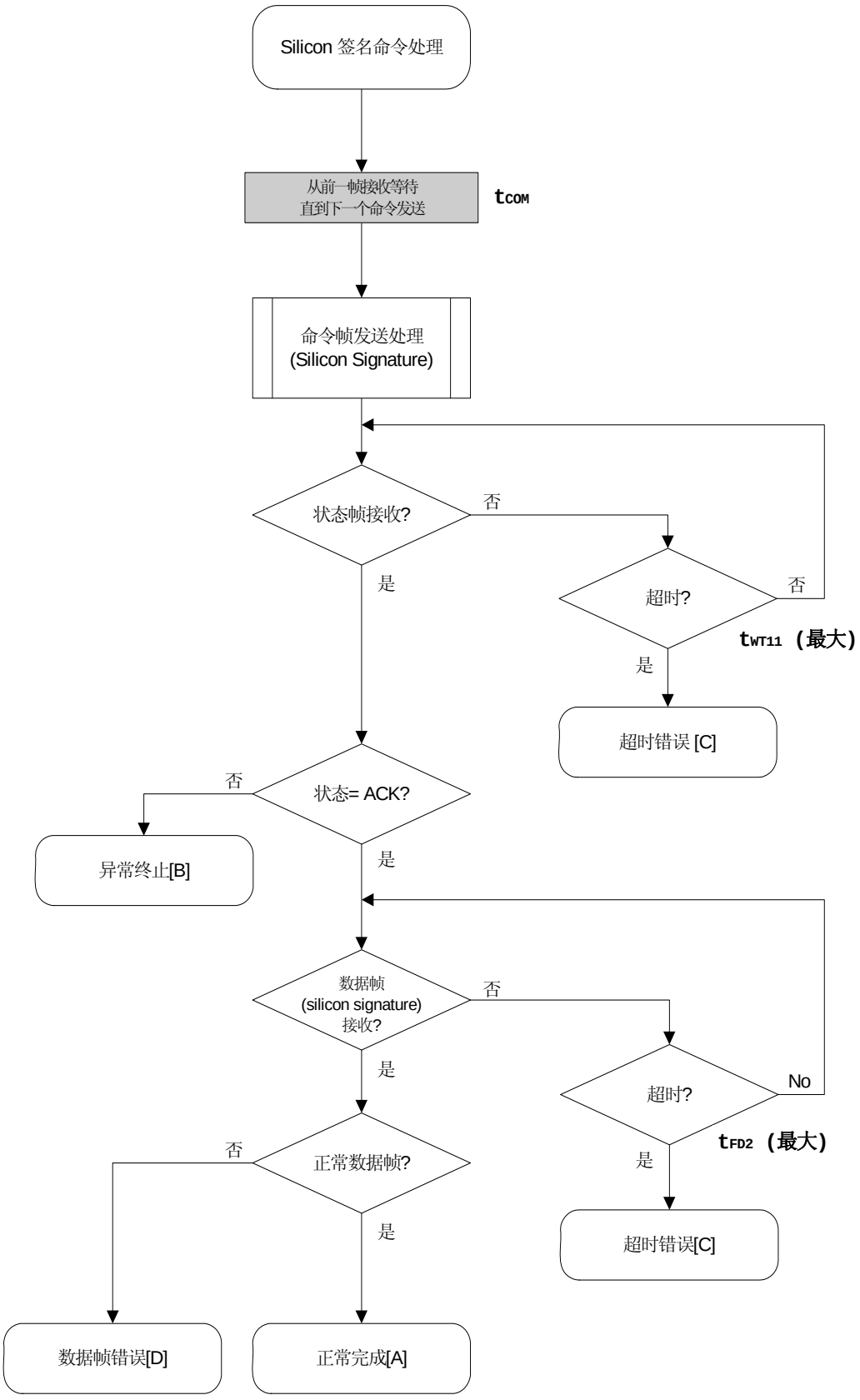
<5> 超时检查被执行直到数据帧（silicon 签名数据）接收。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{FD2}（最大））。
<6> 接收的数据帧（silicon 签名数据）被检查。

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

6.11.4 流程图


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

6.11.5 样本程序

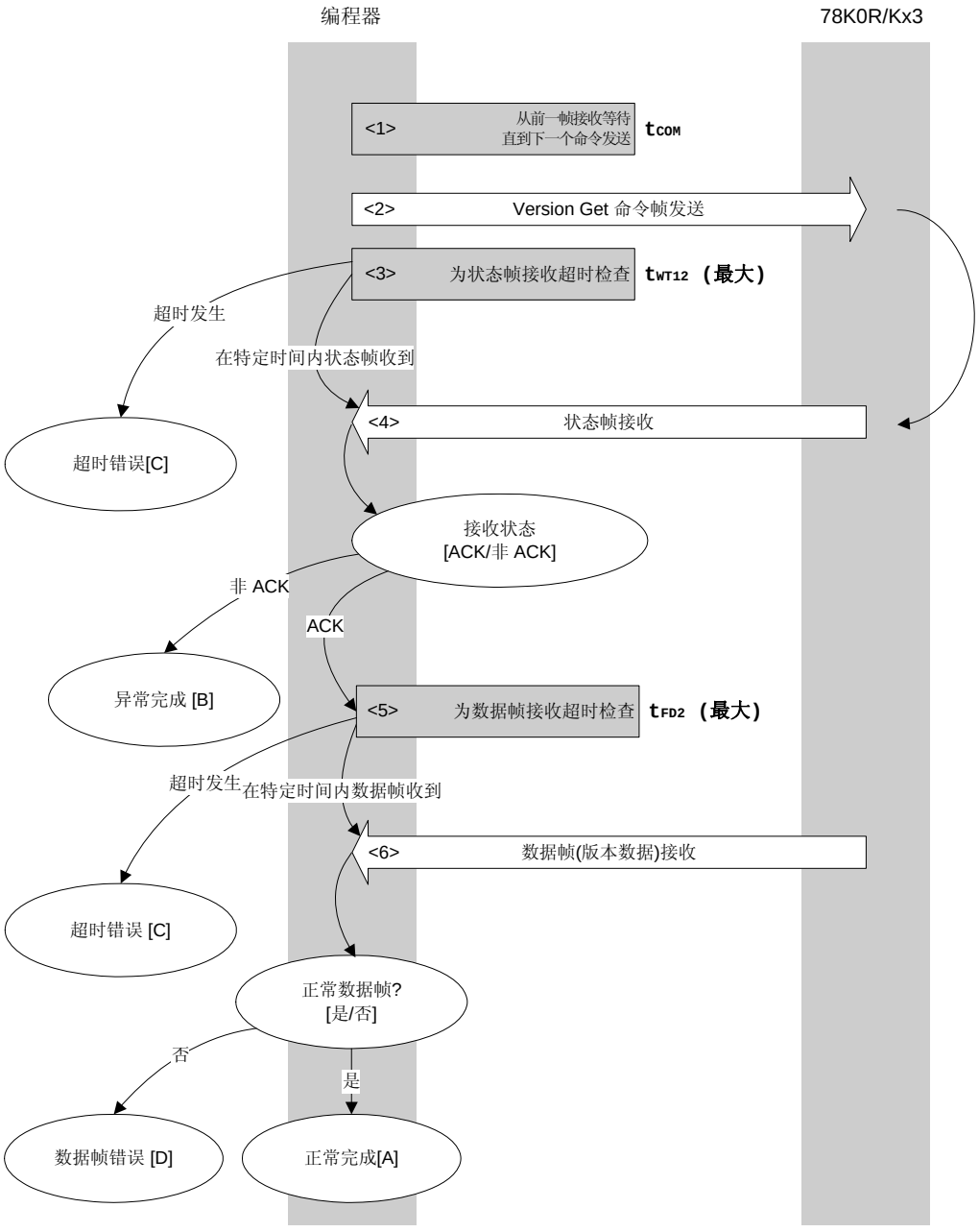
以下表示 Silicon Signature 命令处理的一个样本程序。

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


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

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

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


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

6.12 Version Get 命令

6.12.1 处理顺序图

Version Get 命令处理顺序



6.12.2 处理顺序的说明

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

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

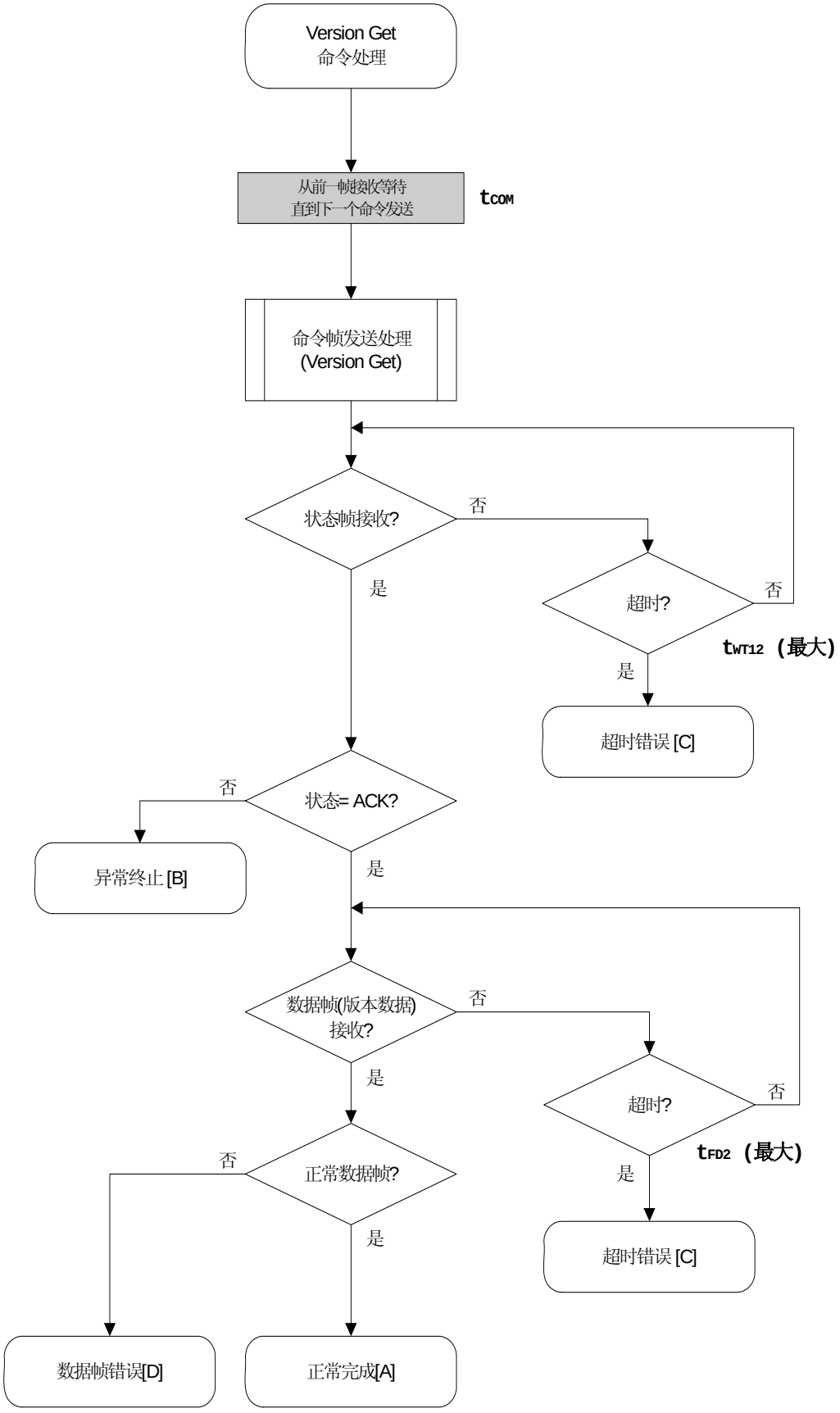
<5> 一个超时检查被执行直到数据帧（版本数据）接收。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{FD2}（最大））。
<6> 接收到的数据帧（版本数据）被检查。

如果数据帧正常： 正常完成[A]
如果数据帧异常： 数据帧错误 [D]

6.12.3 处理完成时的状态

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

6.12.4 流程图


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

6.12.5 样本程序

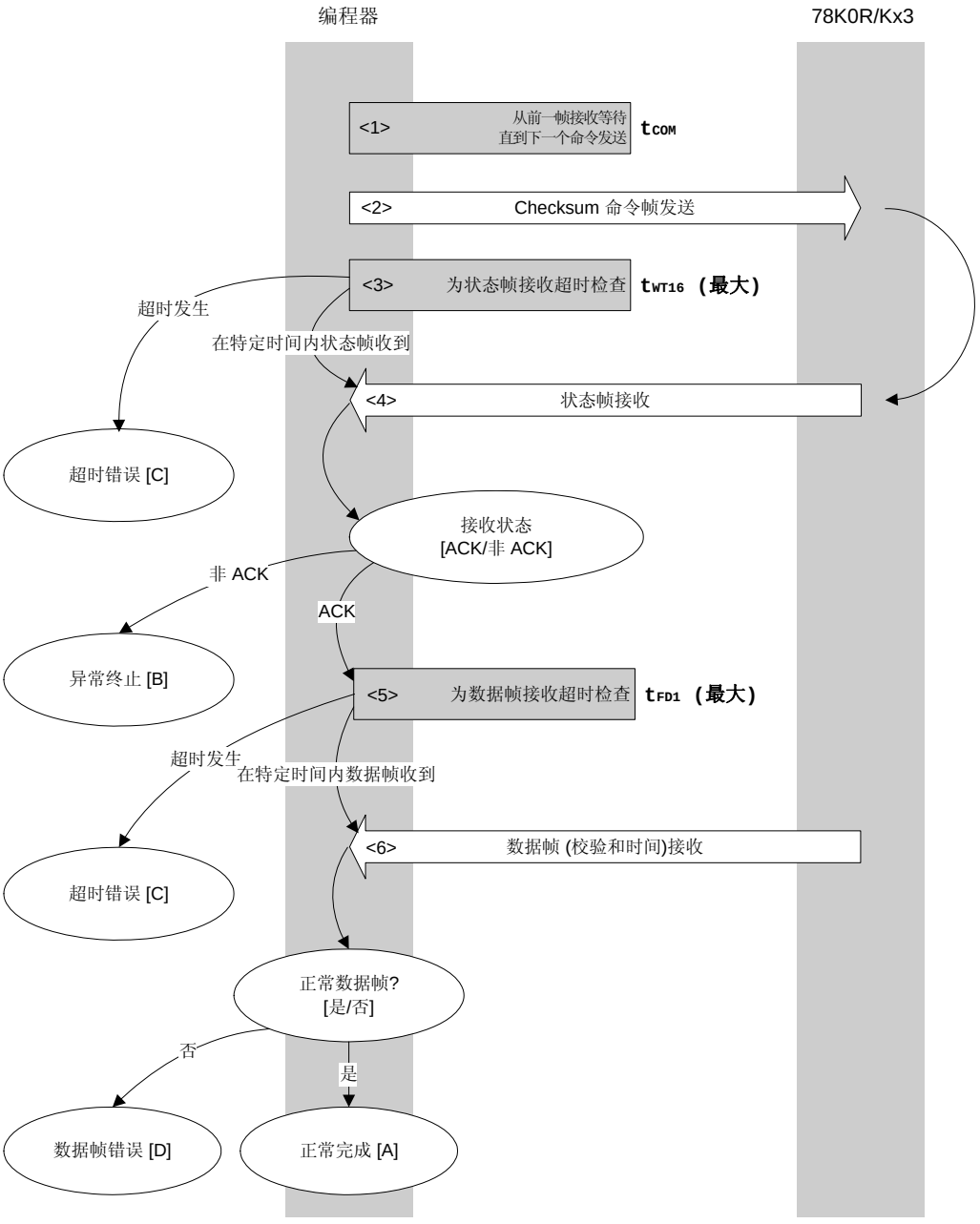
以下表示 Version Get 命令处理的一个样本程序。

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


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

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

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

        rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX); // get data frame
        if (rc){
                return rc;                                      // 如果 [D]
        }

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

6.13 Checksum 命令

6.13.1 处理顺序图

Checksum 命令处理顺序



6.13.2 处理顺序的说明

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

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

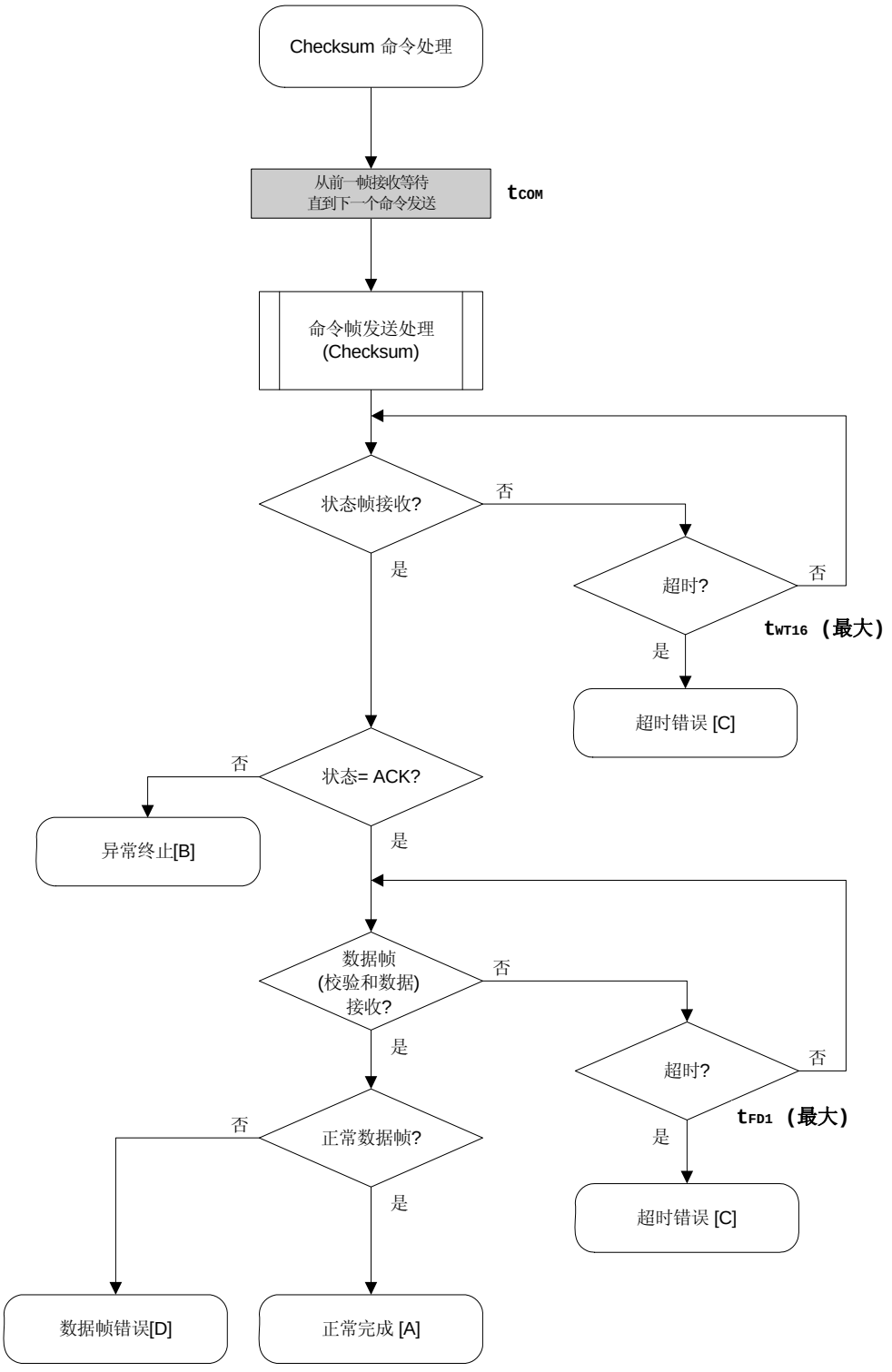
<5> 一个超时检查被执行直到数据帧（校验和数据）接收。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{FD1}（最大））。
<6> 接收的数据帧（校验和数据）被检查。

如果数据帧正常： 正常完成[A]
如果数据帧异常： 数据帧错误[D]

6.13.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且校验和数据被正常获取。
异常终止[B]	参数错误	05H	指定的起始/结束地址超出 flash 存储器范围，或者指定的起始/结束地址不是块的第一个/结束地址。
	校验和错误	07H	发送的命令帧的校验和不匹配。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]		–	状态帧在特定时间内没有被接收。
数据帧错误 [D]		–	作为校验和数据接收到的数据帧的校验和不匹配。

6.13.4 流程图


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

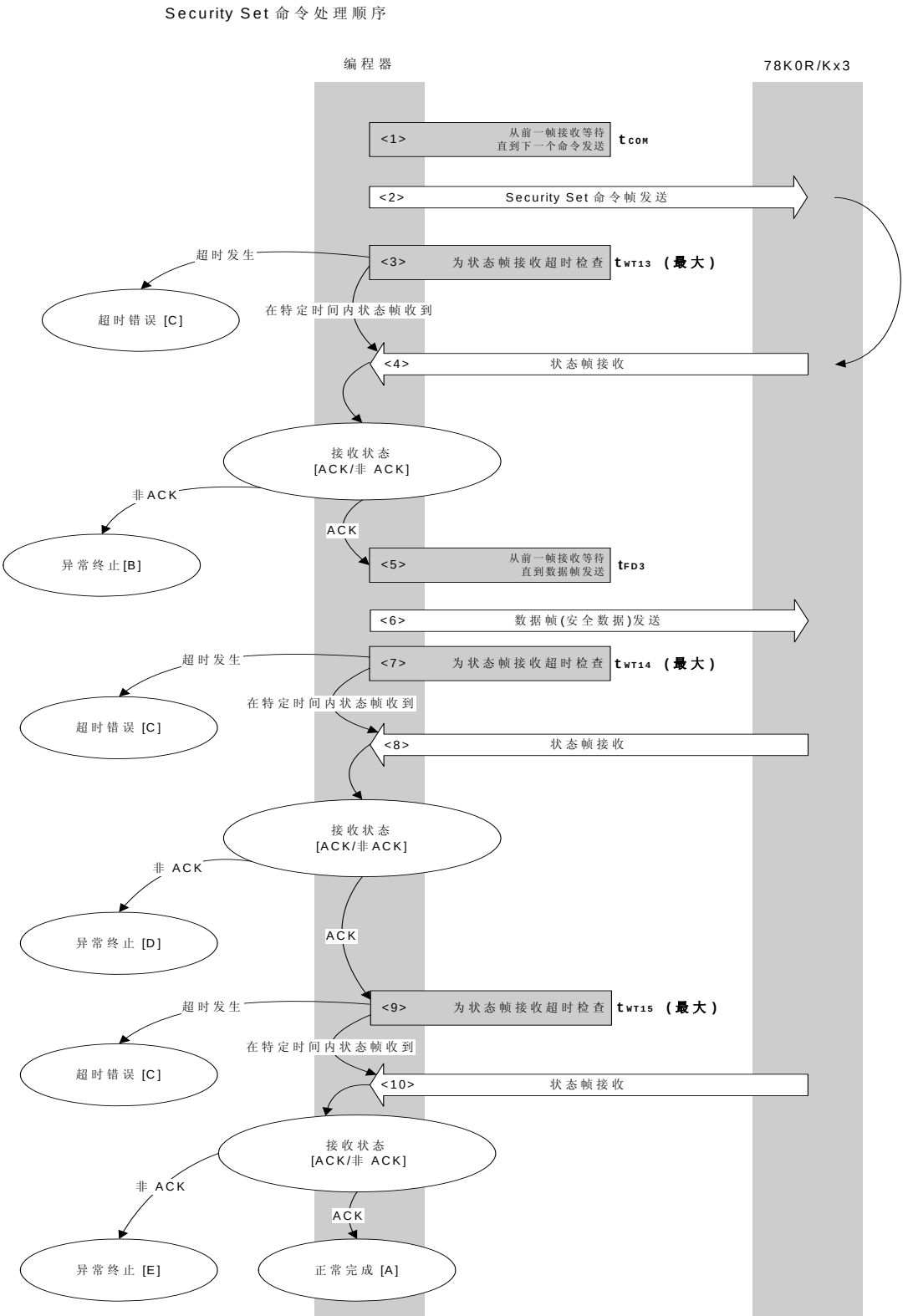
6.13.5 样本程序

以下表示 Checksum 命令处理的一个样本程序。

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

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


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

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

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

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

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

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

}
```

6.14 Security Set 命令

6.14.1 处理顺序图



6.14.2 处理顺序的说明

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

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

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

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

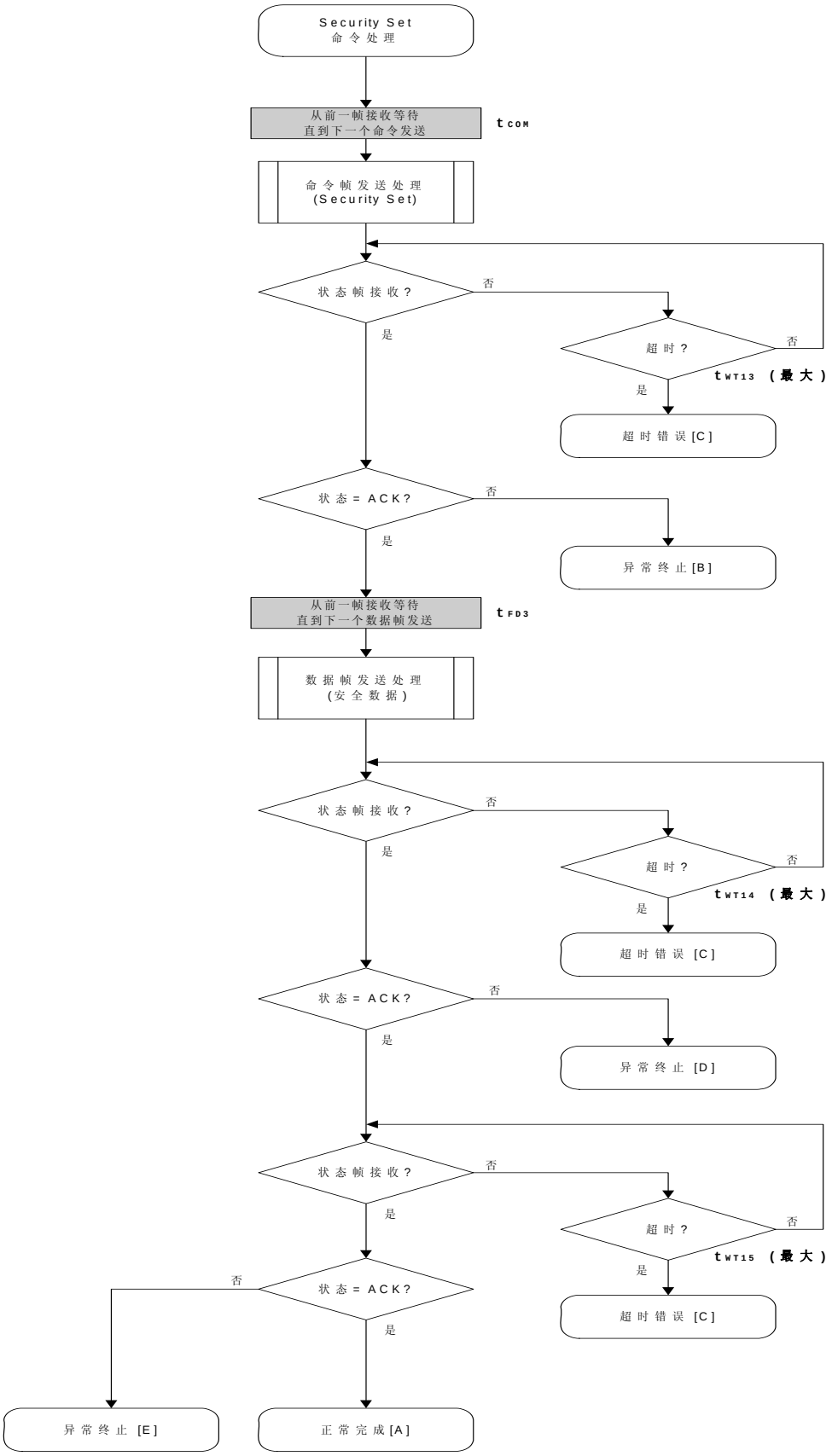
<9> 一个超时检查被执行直到状态帧接收。
如果一个超时发生，超时错误[C]被返回（超时时间 t_{WT15}（最大））。
<10> 状态码被检查。

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

6.14.3 处理完成时的状态

处理完成时的状态		状态码	说明
正常完成[A]	正常响应（ACK）	06H	命令被正常执行，并且安全设置数据被正常设置。
异常终止[B]	参数错误	05H	参数 BOT 不是 01H，FSW 设置块没有设置，所以起始块号比结束块号大，或者 FSW 结束块号比最后一个块号大。
	校验和错误	07H	发送的命令帧或数据帧的校验和不匹配。
	产品错误	10H	一个已经禁止的标志被使能。
	消极响应（NACK）	15H	命令帧数据异常（例如无效数据长度（LEN）或者无 ETX）。
超时错误[C]		–	状态帧或数据帧在特定时间内没有被接收。
异常终止[D]，[E]	MRG10 错误	1AH	写入安全数据失败。
	MRG11 错误	1BH	
	写入错误	1CH	

6.14.4 流程图


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

6.14.5 样本程序

以下表示 Security Set 命令处理的一个样本程序。

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

        /*************************************************/
        /*          设置参数                             */
        /*************************************************/
        fl_cmd_prm[0] = 0x00;                   // “BLK”（必须为0x00）
        fl_cmd_prm[1] = 0x00;                   // “PAG”（必须为0x00）

        fl_txdata_frm[0] = scf|= 0b11101000;    // “FLG”（位7、6、5、3必须为“1”）
        fl_txdata_frm[1] = bot;                 // “BOT”
        fl_txdata_frm[2] = 0x00;                // “FSWS 高”
        fl_txdata_frm[3] = fsws;                // “FSWS 低”
        fl_txdata_frm[4] = 0x00;                // “FSWE 高”
        fl_txdata_frm[5] = fswe;                // “FSWE 低”

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

        put_cmd_ua(FL_COM_SET_SECURITY, 3, fl_cmd_prm);

        rc = get_sfrm_ua(fl_ua_sfrm, tWT13_MAX);  // get status frame
        switch(rc) {
                case    FLC_NO_ERR:                             break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;             break;  // 如果 [C]
                default:        return  rc;             break;  // 如果 [B]
        }


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

        fl_wait(tFD4);
        put_dfrm_ua(6, fl_txdata_frm, true);            // 发送安全设置数据

//      rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX);  // 获取状态帧
        rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX+100);    // 获取状态帧（+100us被加上）
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // 继续
        //      case    FLC_DFTO_ERR:   return  rc;     break;  // 如果 [C]
                default:        return  rc;     break;  // 如果 [B]
```

```
    }

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

第 7 章　FLASH存储器编程参数特性

本章描述在 flash 存储器编程模式下编程器和 78K0R/Kx3 之间的参数特性。
当设计一个编程器时，对于电气规范，确认参阅 78K0R/Kx3 的用户手册。

(1) Flash 存储器参数特性

(a) Flash 存储器编程模式设置时间

参数	符号	最小	典型	最大
V_{DD}↑ 到 FLMD0↑	t_{DP}	0		
FLMD0↑ 到 $\overline{\text{RESET}}$↑	t_{PR}	2 ms		
从 RESET↑的就绪起始时间	t_{R0}	3 ms		100 ms
低电平data0 （就绪）宽度[注]	t_{L0}	892 μs	937.5 μs	987 μs
等待低电平data1	t_{01}	120 μs		
等待低电平data2	t_{02}	10 μs		
等待读取命令	t_{2C}	300 μs		
低电平data1/data2 宽度[注]	t_{L1}，t_{L2}		937.5 μs	

注　低电平宽度与 9,600 bps 下的 00H 数据宽度一样。（它包含起始位，因此它是 9 位的“0”数据）
t_{L0} 是从 78K0R/Kx3 固件发送的数据的低电平宽度。t_{L1} 和 t_{L2} 是从 flash 编程器发送的数据的低电平宽度。

(b) 编程特性

等待	条件	符号	最小	最大
数据帧发送之间	数据帧接收	t_{DR}	8.0 μs	
	数据帧发送	t_{DT}	**注**	
从状态帧发送直到数据帧发送	–	t_{FD1}	**注**	
从状态帧发送直到数据帧接收（1）	程序命令	t_{FD2}	8.7 μs	
从状态帧发送直到数据帧接收（2）	校验命令	t_{FD3}	145 μs	
从状态帧发送直到数据帧接收（3）	安全设置命令	t_{FD4}	120 μs	
从状态帧发送直到命令帧接收	–	t_{COM}	595 μs	

注 对编程器使能连续接收。同时，对编程器设置超时时间为 3 秒或更多。

备注 等待被定义如下。

<t_{DR}，t_{FD2}，t_{FD3}，t_{FD4}，t_{COM}>

在前一个通信完成后的最小时间后，78K0R/Kx3 准备下一个通信。

<t_{DT}，t_{FD1}>

在前一个通信完成后的最小时间后，78K0R/Kx3 准备下一个通信。

(c) 命令特性

命令	符号	条件	最小	最大
Reset	t_{WT0}	–	**注 1**	
Chip Erase	t_{WT1}	产品组 A 注 2	（60.6 + 5.7 × 块的全部个数） ms	（1112 + 140.9 ×块的全部个数) ms
		产品组 B 注 3	（812.9 + 5.7 ×（块的全部个数– 128）） ms	（19403.5 + 140.9 ×（块的全部个数– 128)) ms
Block Erase	t_{WT2} 注 4	–	17.5 ms	（1.1 + 275.5 × 同时选择和擦除的执行次数 + 137.9 × 要被擦除的块的个数) ms
Programming	t_{WT3}	–	**注 1**	
	t_{WT4} 注 5	–	2.8 ms	47.2 ms
	t_{WT5} 注 6	块 0	13.3 ms	860.0 ms
		块 0 以外	13.3 ms	16.3 ms
Verify	t_{WT6}	–	**注 1**	
	t_{WT7} 注 5	–	**注 7**	
Block Blank Check	t_{WT8} 注 6	–	5.7 ms	7.7 ms
Baud Rate Set	t_{WT10}	–	66.0 μs	
Silicon Signature	t_{WT11}	–	**注 1**	
Version Get	t_{WT12}	–	**注 1**	
Security Set	t_{WT13}	–	**注 1**	
	t_{WT14}	–	**注 1**	20.0 μs
	t_{WT15}	–	**注 8**	843.7 ms
Checksum	t_{WT16}	–	**注 1**	

注 **1.** 在命令帧发送前，编程器的接收必须被使能。同时，对编程器设置超时时间为 3 秒或更多。

2. 产品组 A： Flash 大小 ≤ 256 KB （块的个数 ≤ 128）

3. 产品组 B： Flash 大小 > 256 KB （块的个数 > 128）

4. 关于同时选择和擦除的执行次数的计算方法，见（2）由 Block Erase 命令执行的同时选择和擦除。

5. 256 字节数据发送的时间

6. 一个块发送的时间

7. 在数据帧发送前，编程器的接收必须被使能。同时，对编程器设置超时时间为 3 秒或更多。

8. 对编程器使能连续接收。同时，对编程器设置超时时间为 3 秒或更多。

（备注在下一页。）

备注　等待被定义如下。

<t_{WT0} 到 t_{WT8}，t_{WT11} 到 t_{WT16}>

78K0R/Kx3 在最小和最大时间之间完成命令处理，并且发送一个状态帧。

对于指定最大时间的命令，编程器必须等待接收帧的起始位直到最大时间过去，然后执行超时处理。

对于没有指定最大时间的命令，见对应注。

<t_{WT10}>

78K0R/Kx3 在刚刚完成的通信后最小时间过去后可以执行下一个通信。

编程器必须在刚刚完成的通信后最小时间过去后发送下一个数据。

(2) Block Erase 命令执行的同时选择和擦除

通过重复“同时选择和擦除”，78K0R/Kx3 的 Block Erase 命令被执行，它同时擦除多个块。
因此，Block Erase 命令执行过程中插入的等待时间等于“同时选择和擦除”的全部执行时间。
要计算“同时选择和擦除的全部执行时间”，必须首先计算同时选择和擦除的执行次数（M）。
“M”通过获得要被同时擦除的块的个数（同时要被选择和擦除的块的个数）来被计算。

以下描述计算要被同时选择和擦除的块的个数和执行次数（M）的方法。

(a) 要被同时选择和擦除的块的个数的计算

要被同时选择和擦除的块的个数应该为 1、2、4、8、16、32、64 或 128，它依赖于哪一个满足下面的条件。

[条件 1]

（要被擦除的块的个数）≥（要被同时选择和擦除的块的个数）

[条件 2]

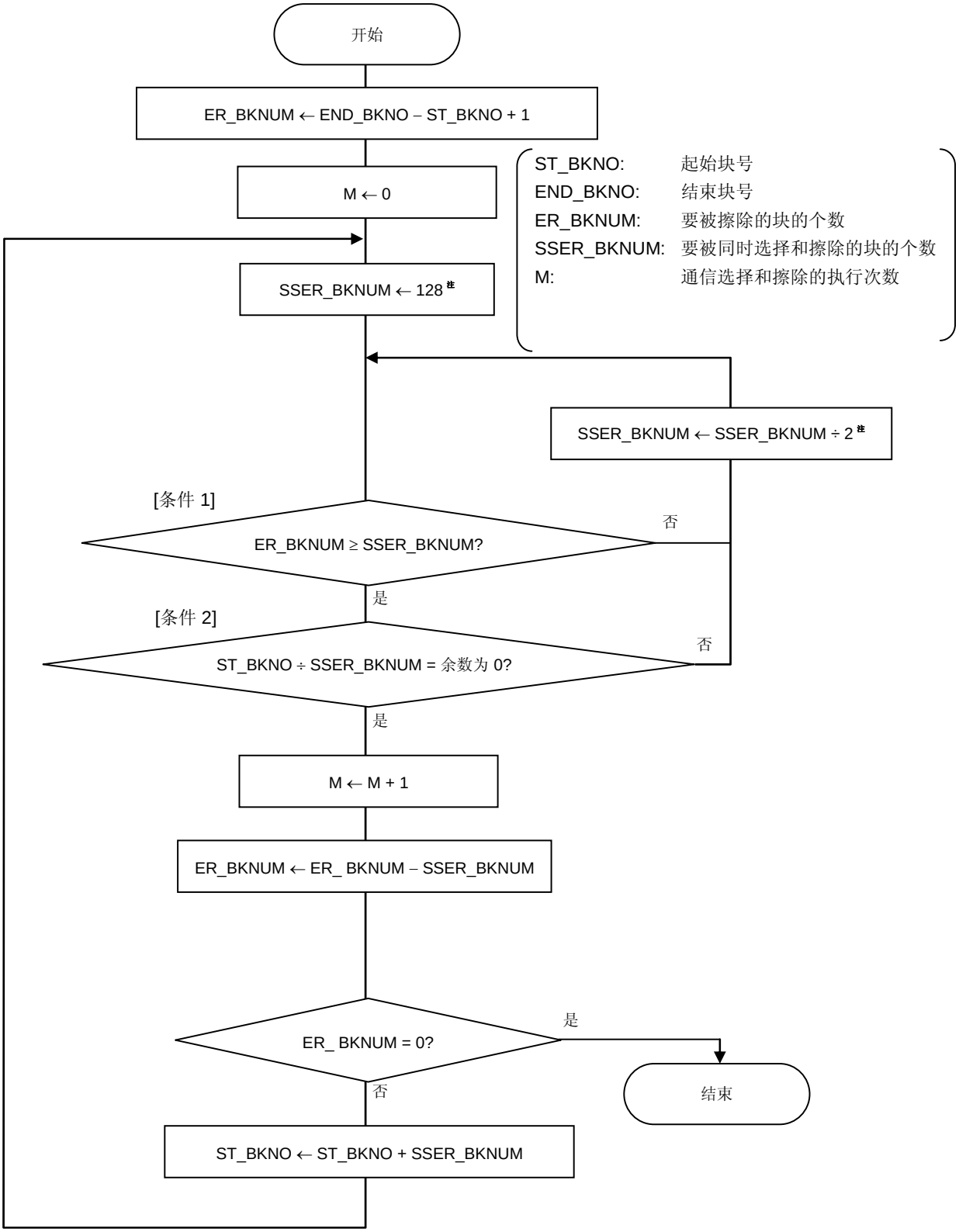
（起始块号）÷（要被同时选择和擦除的块的个数）= 余数为 0

[条件 3]

满足条件 1 和 2 的值中的最大值

(b)　同时选择和擦除的执行次数（M）的计算

在下面流程图中显示了执行次数（M）的计算。



注　基于 SSER_BKNUM （128）的最大值，通过执行 SSER_BKNUM ÷ 2 来获得满足条件 1 和 2 的值，然后条件 3 被满足。

例 1　　擦除块 1 到 127 （N （要被擦除的块的个数） = 127）

<1>　第一个起始块号是 1，要被擦除的块的个数是 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，并且满足条件 3 的值是 1，所以要被同时选择和擦除的块的个数是 1。然后，只有块 1 被擦除。

<2>　在块 1 被擦除后，下一个起始块号是 2，要被擦除的块的个数是 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，并且满足条件 3 的值是 2，所以要被同时选择和擦除的块的个数是 2。然后，块 2 和 3 被擦除。

<3>　在块 2 和 3 被擦除后，下一个起始块号是 4，要被擦除的块的个数是 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，并且满足条件 3 的值是 4，所以要被同时选择和擦除的块的个数是 4。然后，块 4 到 7 被擦除。

<4>　在块 4 到 7 被擦除后，下一个起始块号是 8，要被擦除的块的个数是 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，并且满足条件 3 的值是 8，所以要被同时选择和擦除的块的个数是 8。然后，块 8 到 15 被擦除。

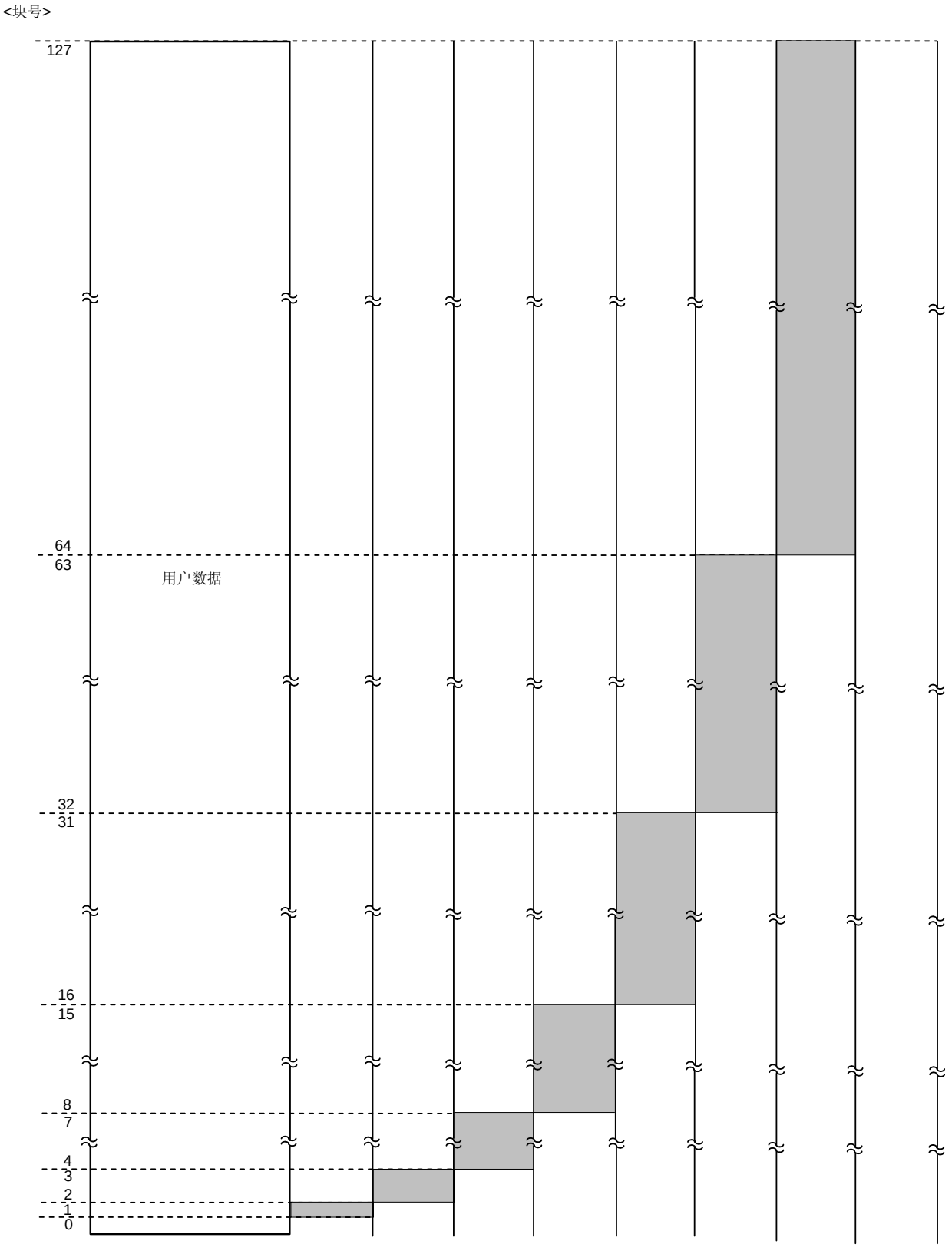
<5>　在块 8 到 15 被擦除后，下一个起始块号是 16，要被擦除的块的个数是 112。因此，满足条件 1 的值是 1、2、4、8、16、32 和 64。
此外，满足条件 2 的值是 1、2、4、8 和 16，并且满足条件 3 的值是 16，所以要被同时选择和擦除的块的个数是 16。然后，块 16 到 31 被擦除。

<6>　在块 16 到 31 被擦除后，下一个起始块号是 32，要被擦除的块的个数是 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，并且满足条件 3 的值是 32，所以要被同时选择和擦除的块的个数是 32。然后，块 32 到 63 被擦除。

<7>　在块 32 到 63 被擦除后，下一个起始块号是 64，要被擦除的块的个数是 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，并且满足条件 3 的值是 64，所以要被同时选择和擦除的块的个数是 64。然后，块 64 到 127 被擦除。

因此，同时选择和擦除被执行 7 次（1，2 和 3，4 到 7，8 到 15，16 到 31，32 到 63 以及 64 到 127）来擦除块 1 到 127，所以 M=7 被获得。

当执行同时选择和擦除时的块配置（当擦除块 1 到 127 时）



<可以被同时选择和擦除的块的范围>

例 2　　擦除块 5 到 10 （N （要被擦除的块的个数）= 6）

<1>　第一个起始块号是 5，要被擦除的块的个数是 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，并且满足条件 3 的值是 1，所以要被同时选择和擦除的块的个数是 1。然后，只有块 5 被擦除。

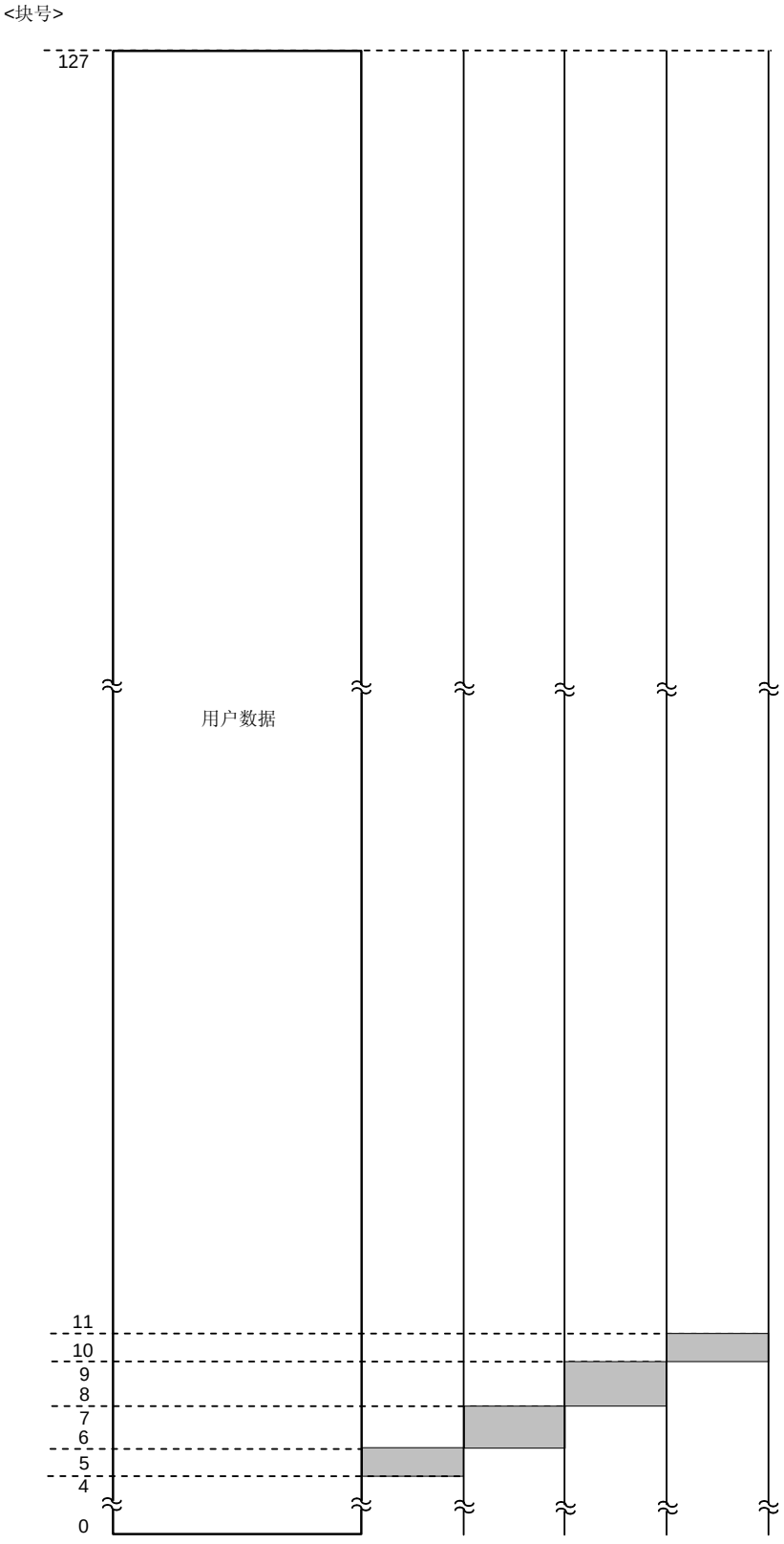
<2>　在块 1 被擦除后，下一个起始块号是 6，要被擦除的块的个数是 5。因此，满足条件 1 的值是 1、2 和 4。
此外，满足条件 2 的值是 1 和 2，并且满足条件 3 的值是 2，所以要被同时选择和擦除的块的个数是 2。然后，块 6 和 7 被擦除。

<3>　在块 6 和 7 被擦除后，下一个起始块号是 8，要被擦除的块的个数是 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，并且满足条件 3 的值是 2，所以要被同时选择和擦除的块的个数是 2。然后，块 8 和 9 被擦除。

<4>　在块 8 和 9 被擦除后，下一个起始块号是 10，要被擦除的块的个数是 1。因此，满足条件 1 的值是 1。这也满足条件 2 和 3，所以要被同时选择和擦除的块的个数是 1。然后，块 10 被擦除。

因此，同时选择和擦除被执行 4 次（5，6 和 7，8 和 9 以及 10）来擦除块 5 到 10，所以 M=4 被获得。

当执行同时选择和擦除时的块配置（当擦除块 5 到 10 时）



<可以被同时选择和擦除的块的范围>

例 3　　擦除块 25 到 73 （N （要被擦除的块的个数） = 49）

<1>　第一个起始块号是 25，要被擦除的块的个数是 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，并且满足条件 3 的值是 1，所以要被同时选择和擦除的块的个数是 1。然后，只有块 25 被擦除。

<2>　在块 25 被擦除后，下一个起始块号是 26，要被擦除的块的个数是 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，并且满足条件 3 的值是 2，所以要被同时选择和擦除的块的个数是 2。然后，块 26 和 27 被擦除。

<3>　在块 26 和 27 被擦除后，下一个起始块号是 28，要被擦除的块的个数是 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，并且满足条件 3 的值是 4，所以要被同时选择和擦除的块的个数是 4。然后，块 28 到 31 被擦除。

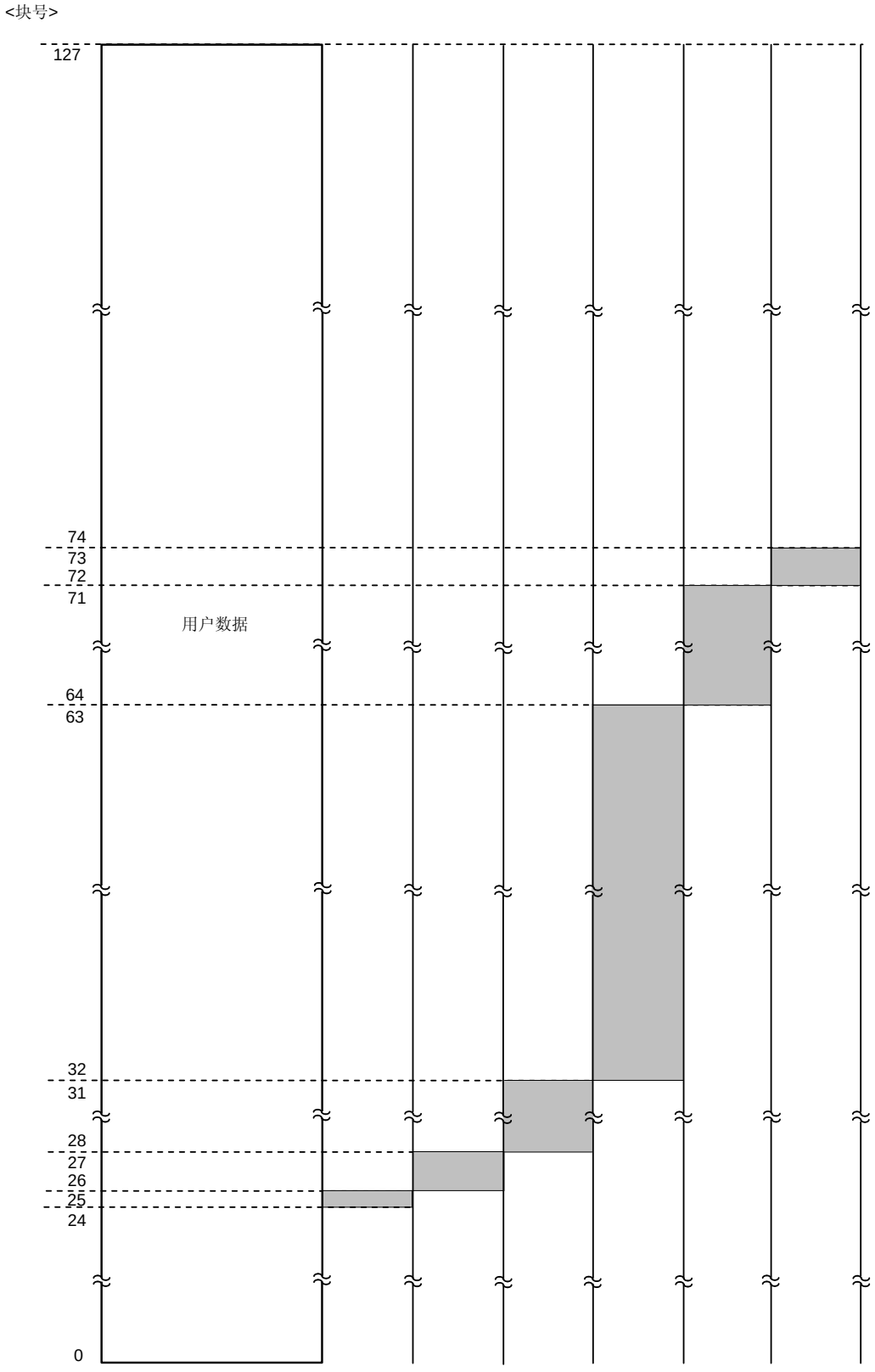
<4>　在块 28 到 31 被擦除后，下一个起始块号是 32，要被擦除的块的个数是 42。因此，满足条件 1 的值是 1、2、4、8、16 和 32。
此外，满足条件 2 的值是 1、2、4、8 和 32，并且满足条件 3 的值是 32，所以要被同时选择和擦除的块的个数是 32。然后，块 32 到 63 被擦除。

<5>　在块 32 到 63 被擦除后，下一个起始块号是 64，要被擦除的块的个数是 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，并且满足条件 3 的值是 8，所以要被同时选择和擦除的块的个数是 8。然后，块 64 到 71 被擦除。

<6>　在块 64 到 71 被擦除后，下一个起始块号 72，要被擦除的块的个数是 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，并且满足条件 3 的值是 2，所以要被同时选择和擦除的块的个数是 2。然后，块 72 和 73 被擦除。

因此，同时选择和擦除被执行 6 次（25，26 和 27，28 到 31，32 到 63，64 到 71 以及 72 和 73）来擦除块 25 到 73，所以 M=6 被获得。

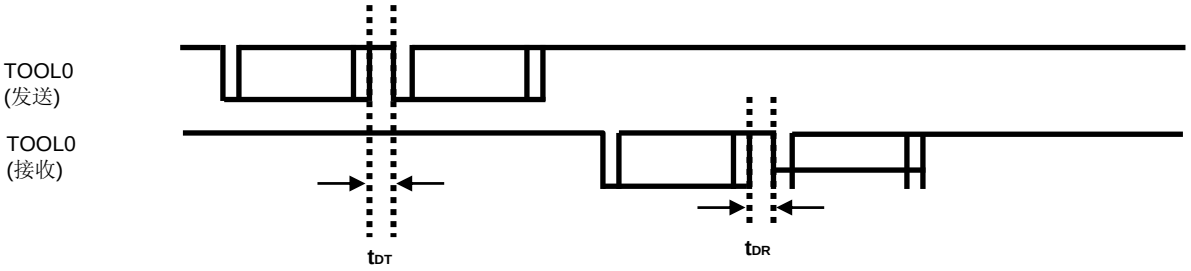
当执行同时选择和擦除时的块配置（当擦除块 25 到 73 时）



<可以被同时选择和擦除的块的范围>

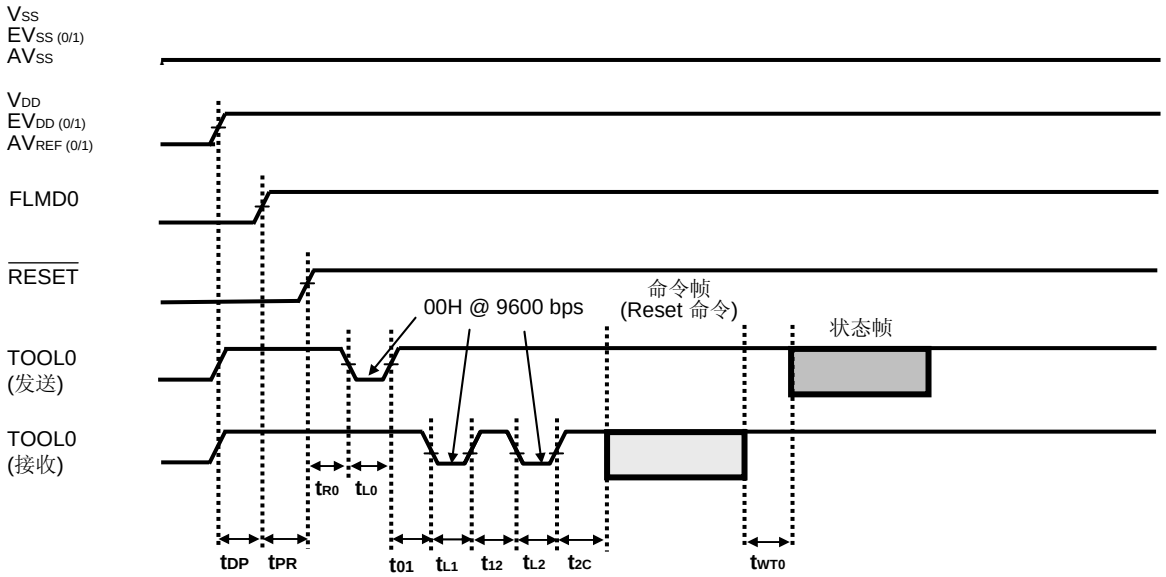
(3) UART 通信模式

(a) 数据帧



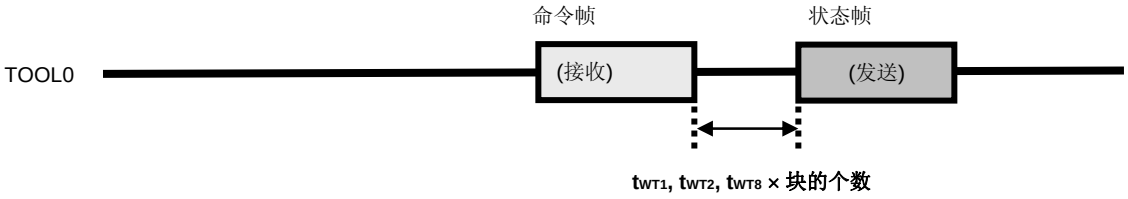
备注　在上面的图中，TOOL0 为了描述的方便被显示为两条独立的线，但是它实际上是一条单独的线。TOOL0 的 V_{DD} 电平可以通过使用一个上拉电阻来达到（管脚为高阻）。

(b) 编程模式设置/Reset 命令



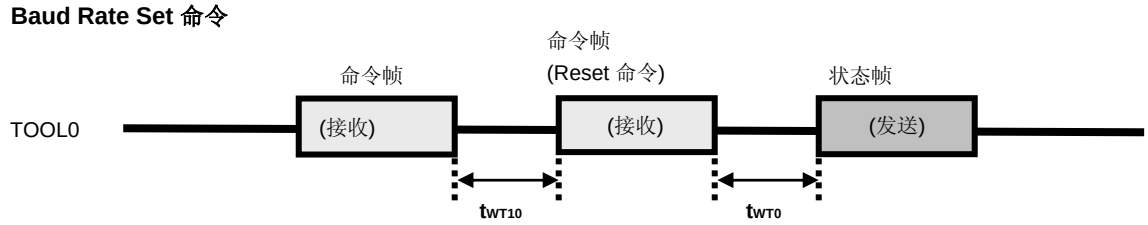
备注　在上面的图中，TOOL0 为了描述的方便被显示为两条独立的线，但是它实际上是一条单独的线。TOOL0 的 V_{DD} 电平可以通过使用一个上拉电阻来达到（管脚为高阻）。

(c) Chip Erase 命令/Block Erase 命令/Block Blank Check 命令/Oscillating Frequency Set 命令



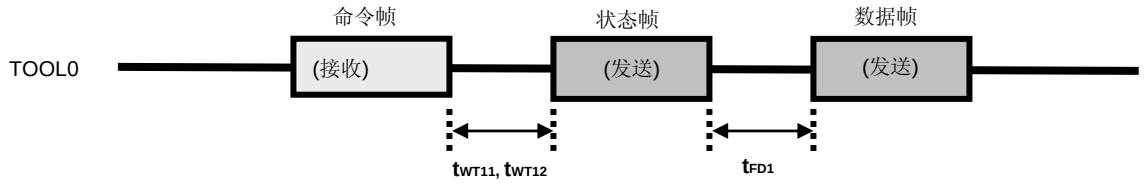
备注　括号中的说明表示 78K0R/Kx3 的操作。

(d) Baud Rate Set 命令



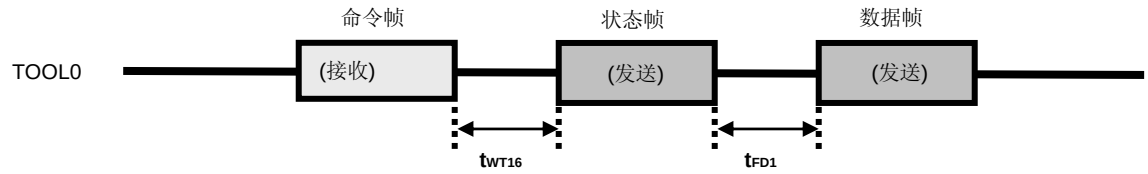
备注　括号中的说明表示 78K0R/Kx3 的操作。

(e) Silicon Signature 命令/Version Get 命令



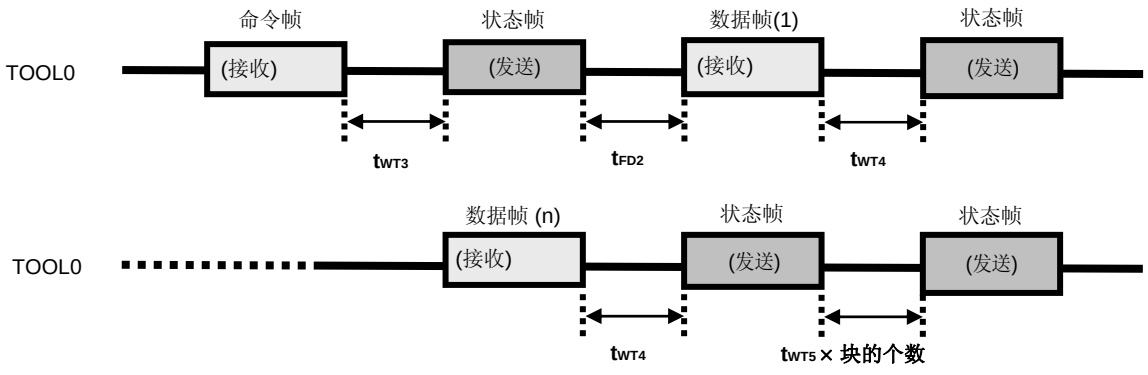
备注　括号中的说明表示 78K0R/Kx3 的操作。

(f) Checksum 命令



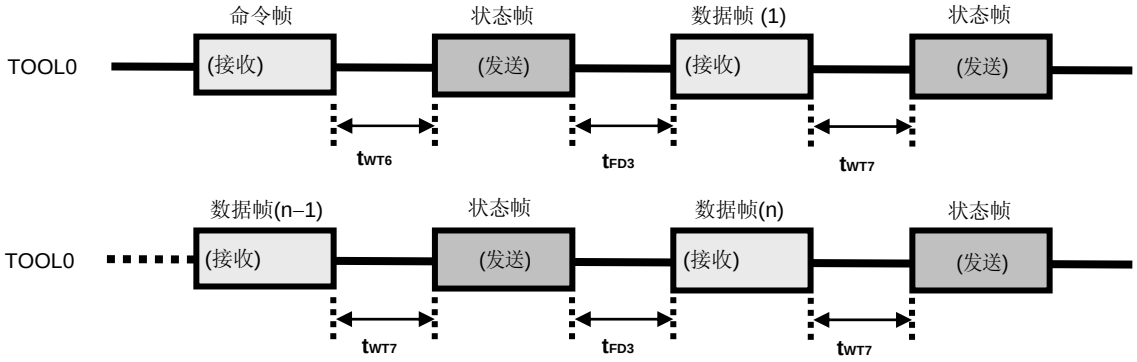
备注　括号中的说明表示 78K0R/Kx3 的操作。

(g) Programming 命令



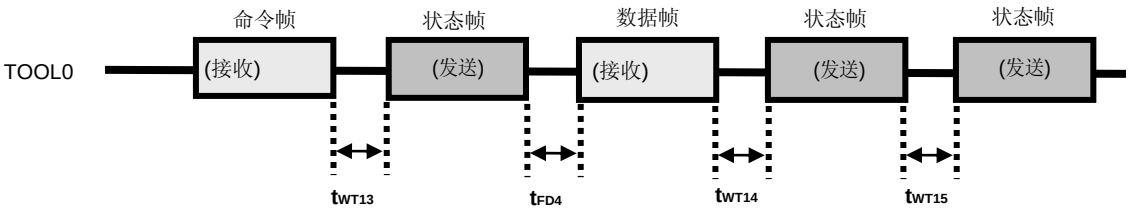
备注　括号中的说明表示 78K0R/Kx3 的操作。

(h) Verify 命令



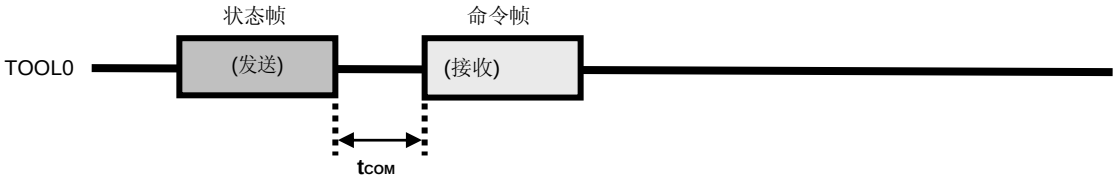
备注　　括号中的说明表示 78K0R/Kx3 的操作。

(i) Security Set 命令



备注　　括号中的说明表示 78K0R/Kx3 的操作。

(j) 命令帧发送前的等待



备注　　括号中的说明表示 78K0R/Kx3 的操作。

[备忘录]

附录 A 电路图（参考）

图 A-1 和 A-2 表示编程器和 78K0R/Kx3 的电路图，供参考。

图 A-1. 编程器和 78K0R/Kx3 的参考电路图（母板）

78K0R/Kx3 Flash 编程器应用举例 - 母板

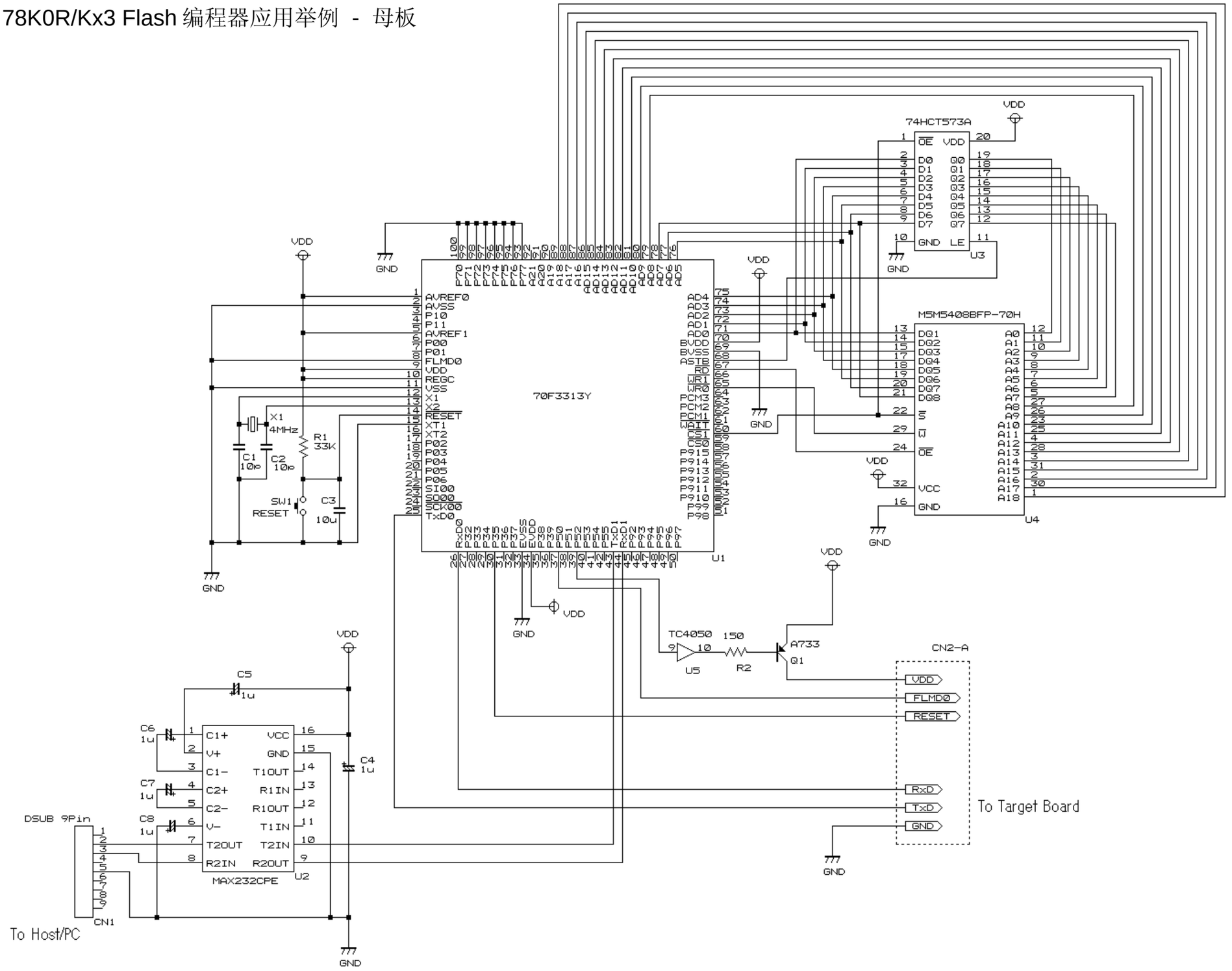
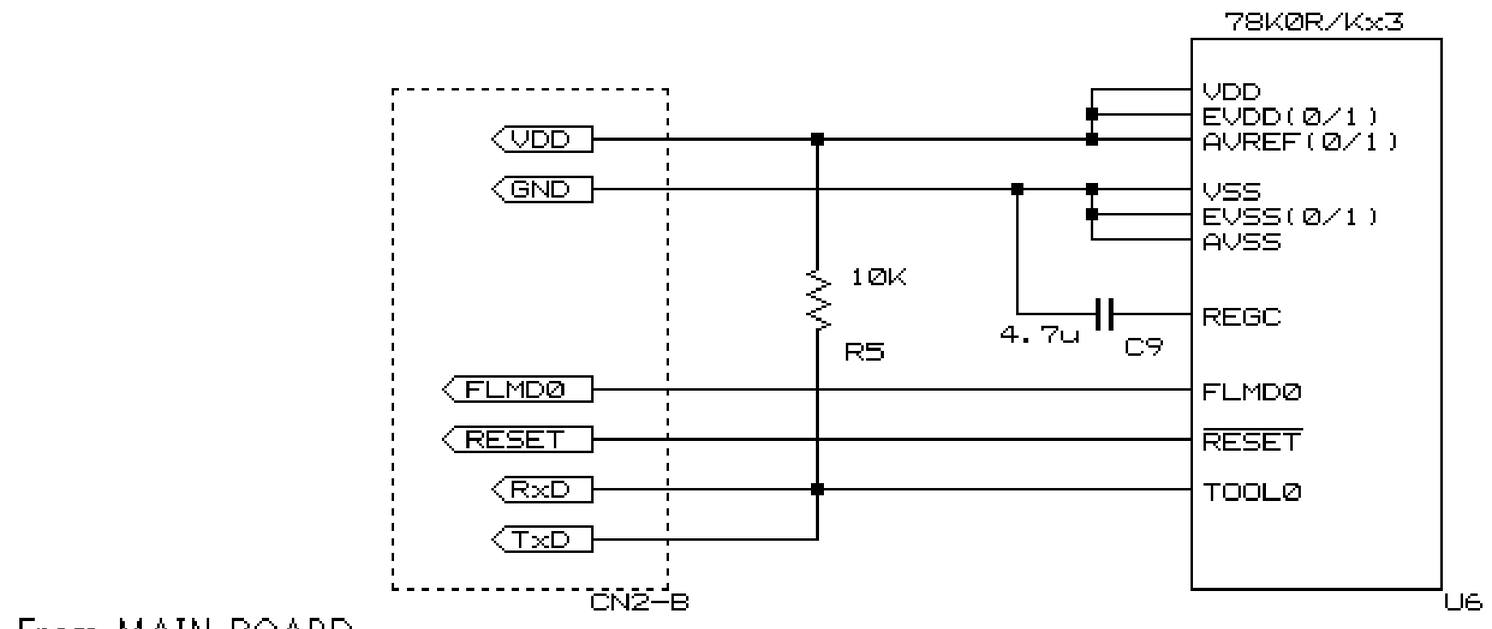


图 A-2. 编程器和 78K0R/Kx3 的参考电路图（目标板

78K0R/Kx3 Flash 编程器应用举例 - 目标板



区域信息

本文档中的某些信息可能因国家不同而有所差异。用户在使用任何一种 NEC 产品之前，请与当地的 NEC 办事处联系，以获取权威的代理商和发行商信息。请验证以下内容：

- 设备的可用性
- 定货信息
- 产品发布进度表
- 相关技术资料的可用性
- 开发环境要求（例如：要求第三方工具和组件，主计算机，电源插头，AC 供电电源等）
- 网络要求

此外，对于商标、注册商标、出口限制条款和其他法律规定，不同的国家也有不同的要求。

详细信息请联系：

（中国区）

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

[北京]

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

[深圳]

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

[上海]

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

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

[香港]

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