お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

アプリケーション・ノート

78K0R/Kx3

16ビット・シングルチップ・マイクロコントローラ

フラッシュ・メモリ・プログラミング（プログラマ編）

μPD78F1142
μPD78F1143
μPD78F1144
μPD78F1145
μPD78F1146
μPD78F1152
μPD78F1153
μPD78F1154
μPD78F1155
μPD78F1156
μPD78F1162
μPD78F1163
μPD78F1164
μPD78F1165
μPD78F1166
μPD78F1167
μPD78F1168

資料番号　U18433JJ2V0AN00（第2版）
発行年月　April 2007 NS　CP(N)

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

① 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力がノイズなどに起因して，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，V_{IL}（MAX.）からV_{IH}（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

② 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③ 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

④ 初期化以前の状態

電源投入時，MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

⑤ 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

⑥ 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

- 本資料に記載されている内容は2007年4月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対 象 者　このアプリケーション・ノートは，78K0R/Kx3の機能を理解し，それを用いたアプリケーション・システムを設計するユーザを対象としています。

目　　的　このアプリケーション・ノートは，78K0R/Kx3内蔵のフラッシュ・メモリの書き換えを行うのに，ユーザ専用のフラッシュ・メモリ・プログラマを開発するための方法をユーザに理解していただくことを目的としています。
なお，掲載のプログラムおよび回路図は例示したものであり，量産設計を対象とするものではありません。
したがって，お客様の機器に使用される場合には，設計後，お客様の責任において十分な評価を行ってください。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

- フラッシュ・メモリ・プログラミング
- プログラマ動作環境
- プログラマの基本動作
- コマンド／データ・フレーム・フォーマット
- コマンド処理説明
- UART通信方式
- フラッシュ・メモリ・プログラミング・パラメータ特性

読 み 方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコントローラの一般知識を必要とします。

□一通りの機能を理解しようとするとき

→目次に従ってお読みください。本文欄外の★印は，本版で改訂された主な箇所を示しています。この"★"をPDF上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

□78K0R/Kx3のハードウエア機能を知りたいとき

→78K0R/Kx3 各製品のユーザーズ・マニュアルを参照してください。

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称に上線）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2進数…××××または××××B 10進数…×××× 16進数…××××H

目　　次

第1章　フラッシュ・メモリ・プログラミング

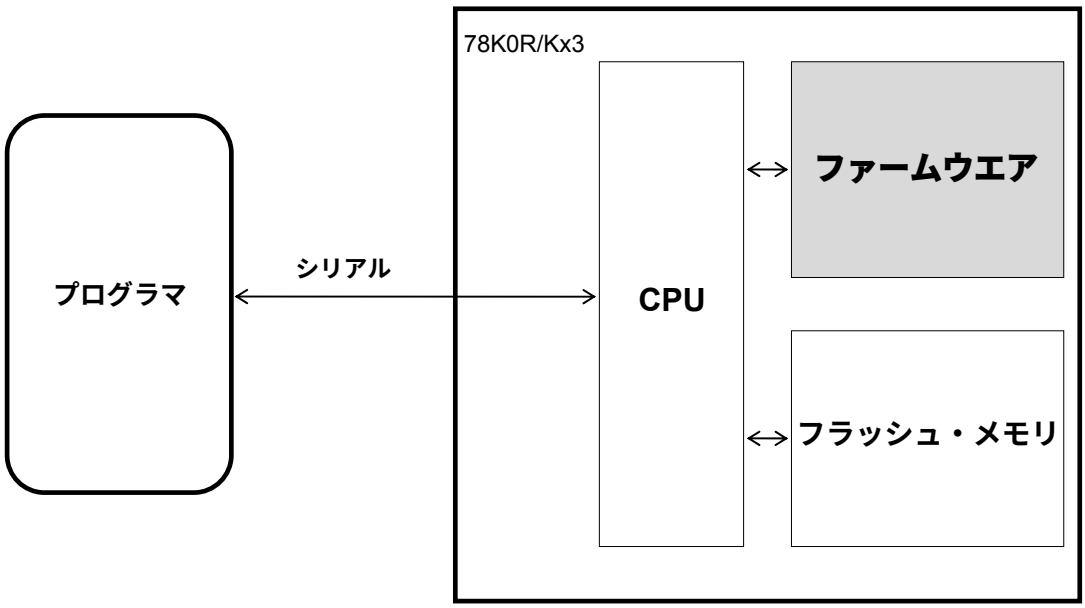
78K0R/Kx3に内蔵されるフラッシュ・メモリの書き換えを行うには，通常は専用のフラッシュ・メモリ・プログラマ（以降プログラマ）を使う必要があります。

このアプリケーション・ノートでは，ユーザが専用のプログラマを開発するための方法を説明します。

1.1　概　　要

78K0R/Kx3は，フラッシュ・メモリ書き換え制御を行うファームウエアを内蔵しています。シリアル通信により，プログラマと78K0R/Kx3間でコマンドを送受信し，内蔵フラッシュ・メモリの書き換えを行います。

図1－1　78K0R/Kx3のフラッシュ・メモリ・プログラミングのシステム概略



1. 2　システム構成

フラッシュ・メモリ・プログラミング時のシステム構成例を次に示します。

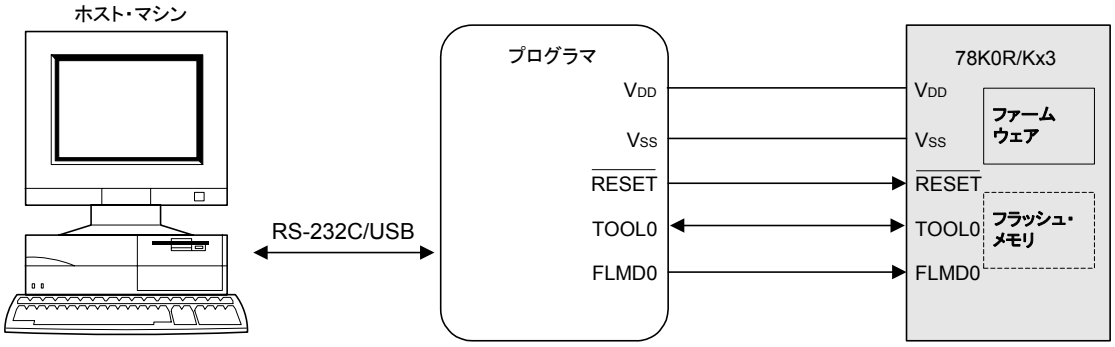
図1－2は，ホスト・マシンからの制御によりプログラマを使用するプログラミング方法を示しています。

プログラマの実装方法によって，あらかじめユーザ・プログラムがプログラマにダウンロードされている場合には，ホスト・マシンを使用せずにスタンド・アローンでもプログラマを動作させることができます。

たとえば，NECエレクトロニクス製フラッシュ・メモリ・プログラマ PG-FP4は，ホスト・マシンを接続してGUIソフトウエアにより実行する方法と，スタンド・アローンで実行する方法のどちらでも動作可能です。

図1－2　システム構成

単線UART通信方式（LSB先頭転送）

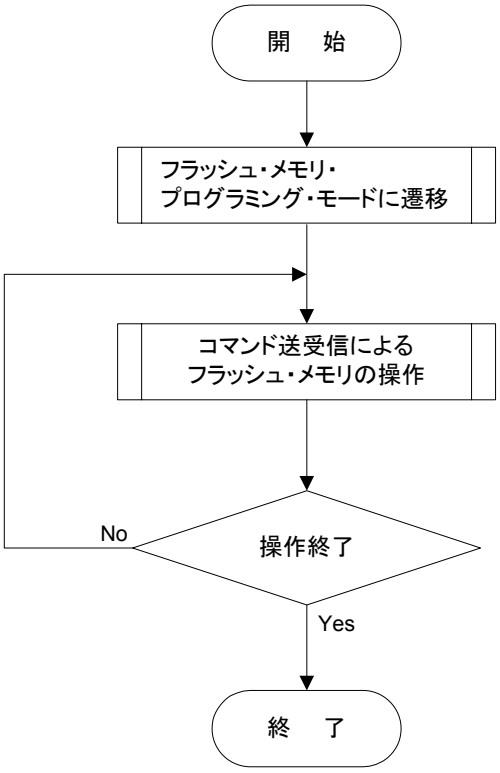


備考　78K0R/KX3の通信方式は，単線UART通信方式のみ使用可です。

1. 3　プログラミング概要

プログラマにてフラッシュ・メモリの書き換えを行うには，まず78K0R/Kx3の動作モードをフラッシュ・メモリ・プログラミング・モードに遷移させる必要があります。その後，プログラマよりシリアル通信にてコマンドを送信し，フラッシュ・メモリの書き換えを行います。その流れを図1－3のフロー・チャートに示します。

図1－3　プログラミング概要図



1. 3. 1　プログラミング・モードへの遷移

78K0R/Kx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子（FLMD0）に規定の電圧を供給し，その後リセットを解除することにより，フラッシュ・メモリ・プログラミング・モードに遷移させることができます。

1. 3. 2　コマンドの送受信によるフラッシュ・メモリの操作

78K0R/Kx3が内蔵するフラッシュ・メモリには，フラッシュ・メモリを書き換えるための機能が内蔵されており，表1－1に示すようなフラッシュ・メモリ操作機能が使用できます。

表1－1　フラッシュ・メモリ機能概要

機　　能	概　　要
消去	フラッシュ・メモリの内容を消去します。
書き込み	フラッシュ・メモリへデータを書き込みます。
ベリファイ	フラッシュ・メモリとベリファイ用データの比較を行います。
情報取得	フラッシュ・メモリに関する情報を読み出します。

これらの機能を制御するためにプログラマから78K0R/Kx3に対しシリアル通信でコマンドを送信します。また，そのコマンドに対し78K0R/Kx3から応答ステータスが返信されます。これら一連のシリアル通信のやりとりを繰り返すことにより，フラッシュ・メモリの書き換えを行います。

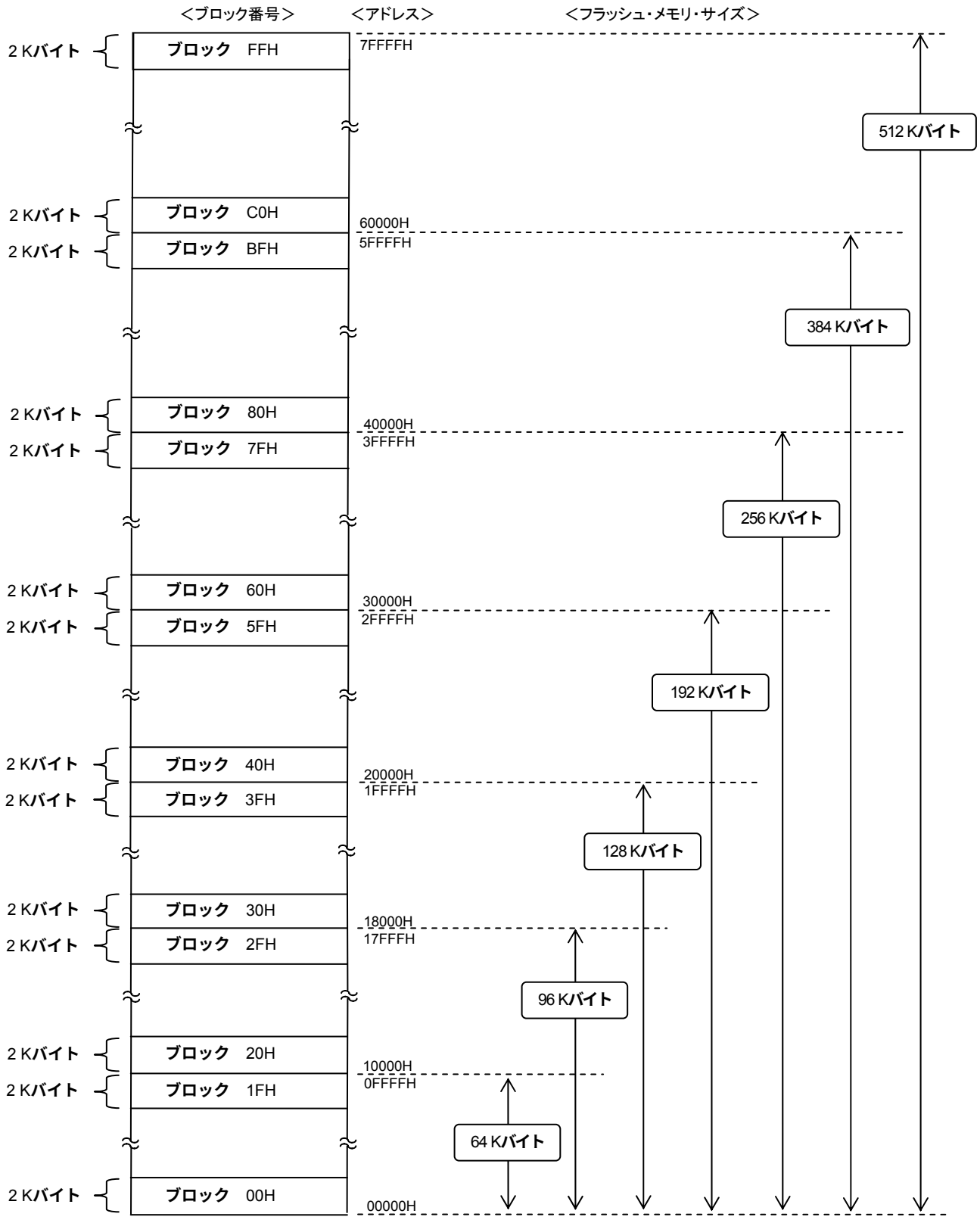
1. 4　78K0R/Kx3製品固有情報

78K0R/Kx3は，プログラマ側で製品固有情報（デバイス名，メモリ情報）を管理しておく必要があります。
表1－2に78K0R/Kx3のフラッシュ・メモリ・サイズ，図1－4にフラッシュ・メモリ構成を示します。

表1－2　78K0R/Kx3のフラッシュ・メモリ・サイズ

デバイス名		フラッシュ・メモリ・サイズ
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

図1－4　フラッシュ・メモリ構成



備考　1ブロックは，すべて2 Kバイトです（この図では，ブロックは一部分しか表記しておりません）。

第2章　プログラマ動作環境

2. 1　プログラマ制御端子

ユーザ・システムにてプログラマ機能を実現するために，プログラマが制御する必要のある端子を表2－1に示します。各端子の詳細は次ページ以降を参照してください。

表2－1　端子説明

プログラマ			78K0R/Kx3	接続時の処置
信号名	入出力	端子機能	端子名	
FLMD0	出力	モード信号	FLMD0	○
★ V_{DD}	入出力	V_{DD}電圧生成／電圧監視	V_{DD} $EV_{DD (0/1)}$ $AV_{REF (0/1)}$注	○
GND	－	グランド	V_{SS} $EV_{SS (0/1)}$ AV_{SS}	○
CLK	出力	クロック出力	－	×
/RESET	出力	リセット信号	$\overline{\text{RESET}}$	○
SI/R_XD	入力	受信信号	TOOL0	○
SO/T_XD	出力	送信信号		
SCK	出力	転送クロック	－	×

★ 注　オフボードで書き込みを行う場合は，V_{DD}と接続してください。
オンボードで書き込みを行う場合は，通常動作モードと同じ電源供給を行ってください（このとき，V_{DD}は必ず$V_{DD} \geqq AV_{REF(0,1)}$となるように設定してください）。

備考　○：端子を使用します。
×：端子を使用しません。

プログラマが制御する各端子の電圧レベルは，フラッシュ・メモリの書き換え対象デバイスのユーザーズ・マニュアルを参照してください。

2. 2　各制御端子の詳細

2. 2. 1　フラッシュ・メモリ・プログラミング・モード引き込み用端子（FLMD0）

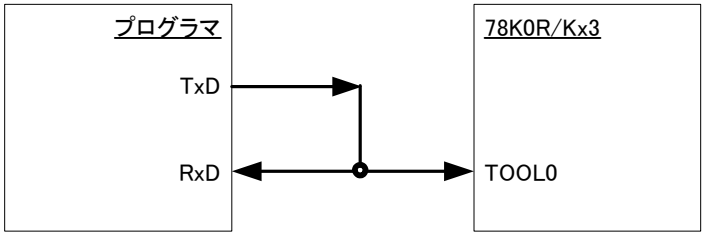
FLMD0端子は78K0R/Kx3の動作モードを制御するための端子です。FLMD0端子に規定の電圧を加えリセットを解除すると，78K0R/Kx3はフラッシュ・メモリ・プログラミング・モードで動作します。

2. 2. 2　シリアル・インタフェース端子（TOOL0）

シリアル・インタフェース端子は，プログラマと78K0R/Kx3間でフラッシュ・メモリ書き換えコマンドの受け渡しを行う端子です。

使用端子の接続図を次に示します。

図2－1　シリアル・インタフェース端子

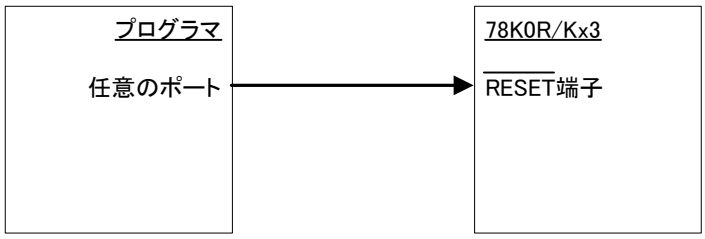


単線UART通信

2. 2. 3　リセット制御端子（$\overline{\text{RESET}}$）

リセット制御端子（$\overline{\text{RESET}}$端子）は，プログラマから78K0R/Kx3のシステム・リセットを制御するための端子です。FLMD0端子を規定の電圧に設定し，その後リセットを解除することにより，フラッシュ・メモリ・プログラミング・モードを選択できます。

図2－2　$\overline{\text{RESET}}$端子

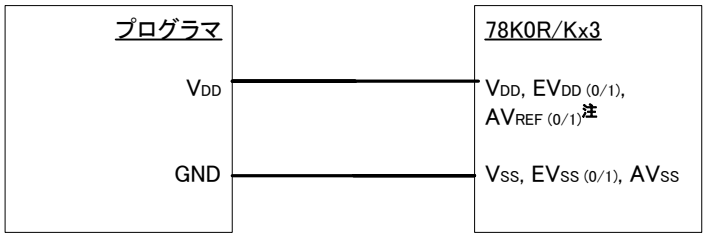


2. 2. 4　V_{DD}, GND制御端子

V_{DD}制御端子は，プログラマから78K0R/Kx3に電源を供給する場合に使用します。プログラマから78K0R/Kx3に電源を供給する必要のないときは，V_{DD}制御端子を接続する必要はありません。ただし，専用プログラマは78K0R/Kx3の電源状態の検出を行っているため，電源供給の有無にかかわらず接続する必要があります。

GND制御端子は，電源供給の有無に関係なく78K0R/Kx3のV_{SS}と接続してください。

図2－3　V_{DD}/GND端子接続



★

★　注　オフボードで書き込みを行う場合は，V_{DD}と接続してください。
オンボードで書き込みを行う場合は，通常動作モードと同じ電源供給を行ってください（このとき，V_{DD}は必ず$V_{DD} \geqq AV_{REF(0,1)}$となるように設定してください）。

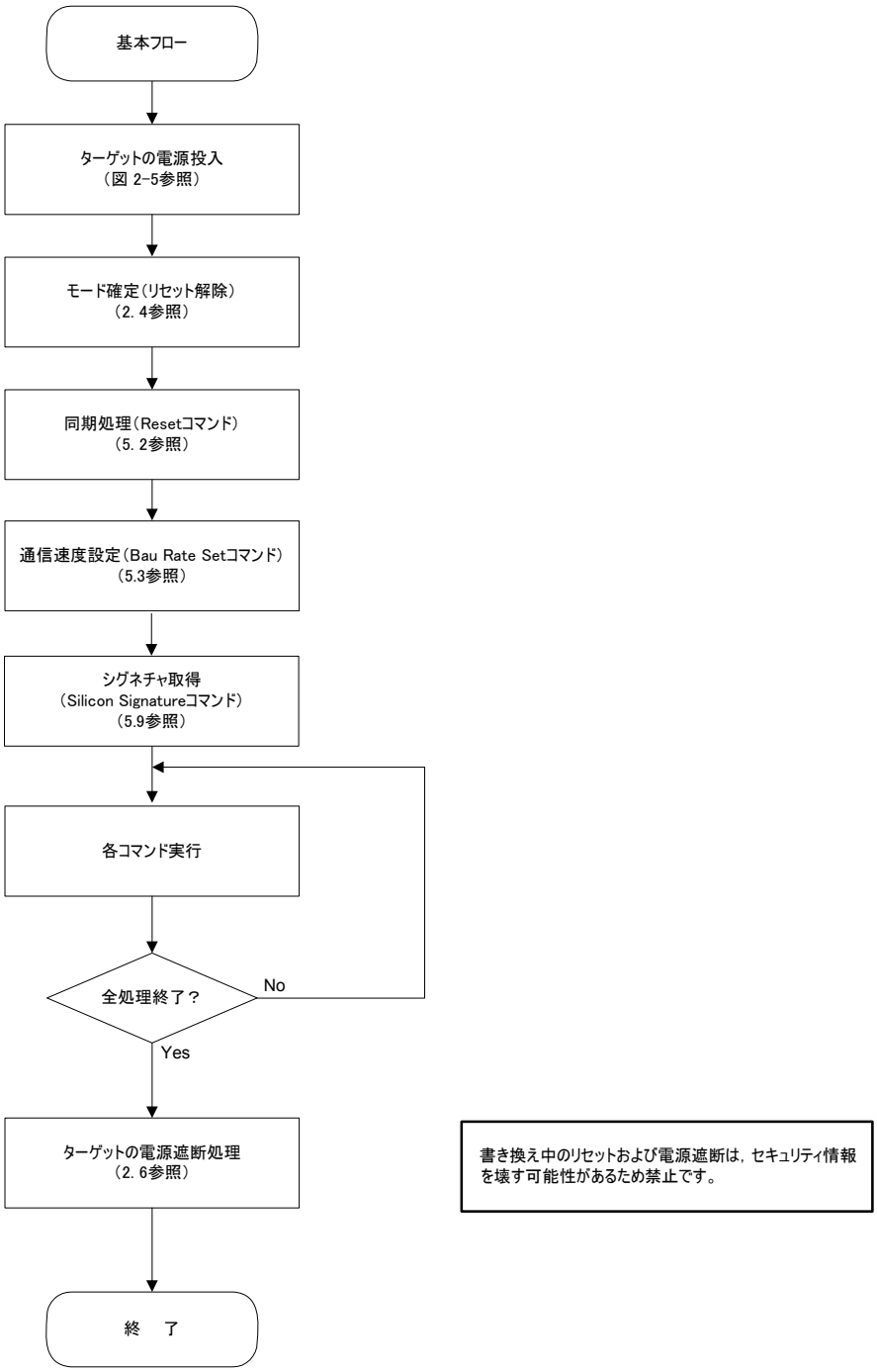
2. 2. 5　その他の端子

プログラマと接続されていない，その他の端子の端子処理は，デバイスのユーザーズ・マニュアルのフラッシュ・メモリの章を参照してください。

2. 3　基本フロー・チャート

プログラマにてフラッシュ・メモリの書き換えを行う際の基本フロー・チャートを次に示します。

図2－4　フラッシュ処理の基本フロー・チャート



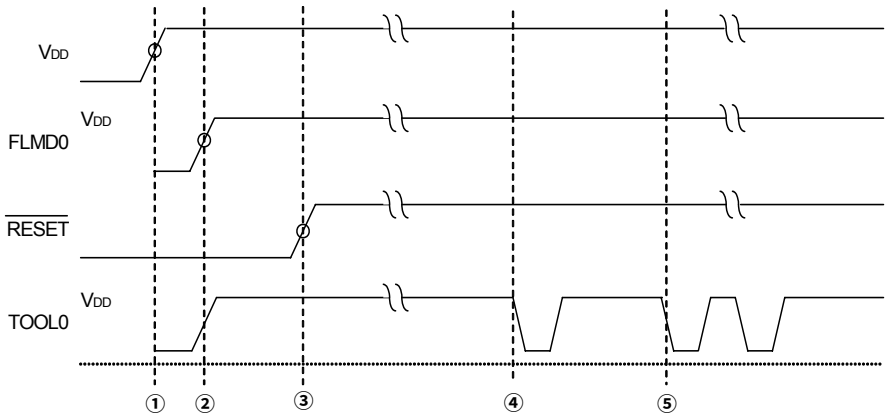
2. 4　フラッシュ・メモリ・プログラミング・モードへの遷移方法

プログラマにてフラッシュ・メモリの書き換えを行うには，まず78K0R/Kx3の動作モードをフラッシュ・メモリ・プログラミング・モードに遷移させる必要があります。

このモードに遷移するには，78K0R/Kx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子（FLMD0）に規定の電圧を供給し，その後リセットを解除します。

フラッシュ・メモリ・プログラミング・モードへの遷移のタイミング図を次に示します。

図2－5　フラッシュ・メモリ・プログラミング・モードへの遷移



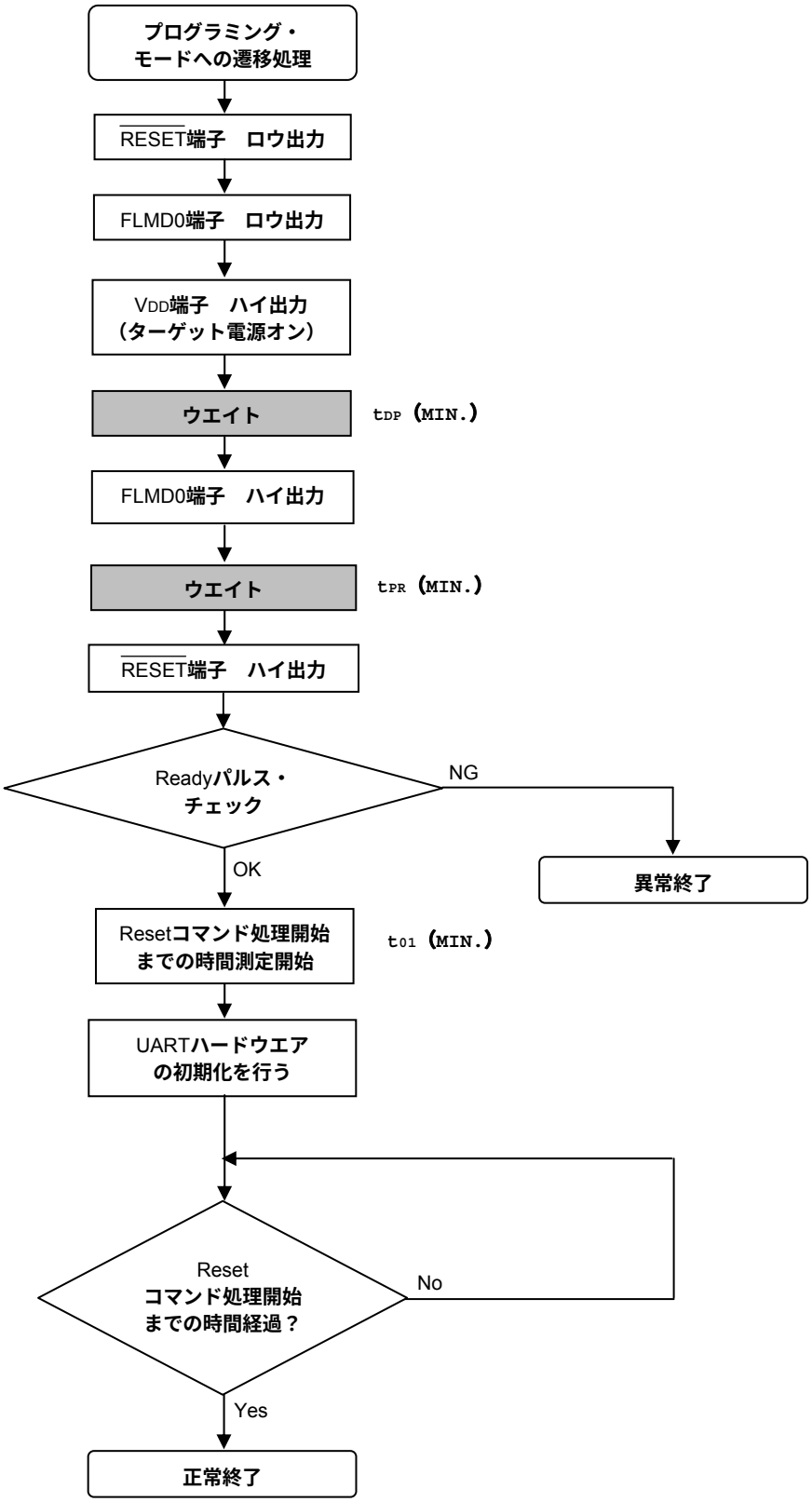
① ：電源（V_{DD}）投入
② ：FLMD0 = ハイ・レベル
③ ：リセット解除（シリアル・プログラミング・モード確定）
④ ：READY パルス（”00”@9600 bps）入力開始（78K0R/Kx3→プログラマ）
⑤ ：LOW パルス（”00”@9600 bps）出力開始（プログラマ→78K0R/Kx3）

リセット解除時のFLMD0端子と動作モードの関係を次に示します。

表2－2　リセット時のFLMD0端子の設定と動作モード

FLMD0	動作モード
ロウ（GND）	通常動作モード
ハイ（V_{DD}）	フラッシュ・メモリ・プログラミング・モード

2. 4. 1　モード引き込みのフロー・チャート



2.4.2　サンプル・プログラム

引き込み処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* connect to Flash device                                   */
/*                                                           */
/*************************************************************/
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;                    // disable UART Rx INT.
   UARTE0 = false;                  // disable UART H.W.
   stop_UART0();                    // TxD/RxD = Hi-Z

   pFL_RES          = low;          // RESET = low
   pmFL_FLMD0       = PM_OUT;       // FLMD0 = Low output
   pFL_FLMD0        = low;
   FL_VDD_HI();                     // VDD = high

   fl_wait(tDP);                    // wait

   pFL_FLMD0        = hi;           // FLMD0 = high
   fl_wait(tPR);                    // wait

   pFL_RES          = hi;           // RESET = high

   rc = check_ready_pulse();        // check "READY PULSE" from target device
   if (rc){
           return  rc;              // pulse width/timing error
   }
   start_flto(t01);                 // start "t01" wait timer

   init_fl_uart();                  // Initialize UART h.w.(for Flash device control)
   UARTE0 = true;                   // enable UART h.w.
   SRIF0 = false;                   // clear UART Rx IRQ flag
   SRMK0 = false;                   // enable UART Rx INT.

   while(!check_flto())             // timeout "t01" ?
           ;                        // no

   return  rc;
   // start RESET command proc.
}
```

2. 5　単線UART通信方式

単線UART通信は，78K0R/Kx3のTOOL0端子を使用します。通信条件は次のようになります。

表2－3　単線UART通信の通信条件

項　　目	内　　容
ボー・レート	ボー・レート設定コマンド処理の Baud Rate Set コマンドの送信までは 9600 bps で通信を行います。そして，ボー・レート・コマンド処理の Reset コマンドの送信から Baud Rate Set コマンドで設定したボー・レートに通信レートが変更になります。設定可能なボー・レートに関しては 5. 3　Baud Rate Set コマンドを参照してください。
パリティ・ビット	なし
データ長	8 ビット（LSB 先頭）
ストップ・ビット	2 ビット（プログラマ→78K0R/Kx3）/1 ビット（78K0R/Kx3→プログラマ）

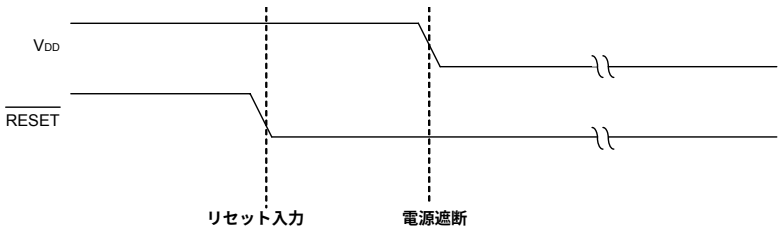
注意　プログラマと78K0R/Kx3ともに同一のボー・レートにしてください。

2. 6　ターゲットの電源遮断処理

各コマンド実行の終了後に，下記のように$\overline{\text{RESET}}$端子をロウ・レベルにしてから電源を遮断してください。
また他の端子は，電源遮断時はHi-Zにしてください。

注意　コマンド処理中の電源遮断およびリセット入力は禁止です。

図2－6　フラッシュ・メモリ・プログラミングモードの終了手順



2. 7　フラッシュ・メモリの操作方法

78K0R/Kx3が内蔵するフラッシュ・メモリには，フラッシュ・メモリ書き換えのための機能が内蔵されており，表2－4に示すようなフラッシュ・メモリ操作機能があります。プログラマは，これらの機能を制御するコマンドを78K0R/Kx3に送信し，78K0R/Kx3からの応答ステータスを確認しながらフラッシュ・メモリを操作します。

表2－4　フラッシュ・メモリ操作機能概要一覧

分　類	機能名	概　要
消去	チップ消去	全フラッシュ・メモリを消去します。また，セキュリティ・フラグもクリアされます。
	ブロック消去	指定したブロックのフラッシュ・メモリを消去します。
書き込み	書き込み	指定したフラッシュ・メモリの領域にデータを書き込みます。
ベリファイ	ベリファイ	指定したフラッシュ・メモリのアドレスから取得したデータと，プログラマから送信されたデータを 78K0R/Kx3 側で比較します。
ブランク・チェック	ブロック・ブランク・チェック	指定したフラッシュ・メモリの領域の消去状態を確認します。
情報取得	シリコン・シグネチャ取得	書き込みプロトコル情報を取得します。
	バージョン取得	78K0R/Kx3 およびファームウエアのバージョンを取得します。
	チェックサム取得	指定された領域のチェックサム・データを取得します。
セキュリティ	セキュリティ設定	セキュリティ情報を設定します。
その他	リセット	通信の同期検出に使用します。

2. 8　コマンド一覧

プログラマで使用されるコマンドの一覧と機能を次に示します。

表2－5　プログラマから78K0R/Kx3への送信コマンド一覧

コマンド番号	コマンド名	機　　能
00H	Reset	通信同期検出に使用します。
9AH	Baud Rate Set	単線 UART のボー・レートを設定します。
20H	Chip Erase	全フラッシュ・メモリを消去します。
22H	Block Erase	指定された領域のフラッシュ・メモリを消去します。
40H	Programming	フラッシュ・メモリの指定された領域にデータを書き込みます。
13H	Verify	フラッシュ・メモリの指定された領域の内容とプログラマから送信されたデータを比較します。
32H	Block Blank Check	指定されたブロックのフラッシュ・メモリの消去状態をチェックします。
C0H	Silicon Signature	78K0R/Kx3 情報（品名，フラッシュ・メモリ構成など）を取得します。
C5H	Version Get	78K0R/Kx3 バージョン，ファームウエア・バージョンを取得します。
B0H	Checksum	指定された領域のチェックサム・データを取得します。
A0H	Security Set	セキュリティ情報を設定します。

2.9　ステータス一覧

プログラマが78K0R/Kx3から受信するステータス・コードの一覧を次に示します。

表2－6　ステータス・コード一覧

ステータス・コード	ステータス	内　容
04H	Command number error	サポートされていないコマンドを受信した場合のエラー
05H	Parameter error	コマンド情報（パラメータ）が適切でない場合のエラー
06H	正常応答（ACK）	正常応答
07H	Checksum error	プログラマから送信されたフレームのデータが異常の場合のエラー
0FH	Verify error	プログラマから送信されたデータとのベリファイ・エラー
10H	Protect error	Security Setコマンドで禁止した処理を実行しようとした場合のエラー
15H	否定応答（NACK）	否定応答
1AH	MRG10 error	消去エラー
1BH	MRG11 error	データ書き込み時に内部ベリファイ・エラー，またはブランク・チェック・エラー
1CH	Write error	書き込みエラー
FFH	処理中（BUSY）	ビジー応答[注]

注　CSI通信の場合，データ・フレーム形式での“FFH”のほかに，1バイトの“FFH”が送信される場合があります。

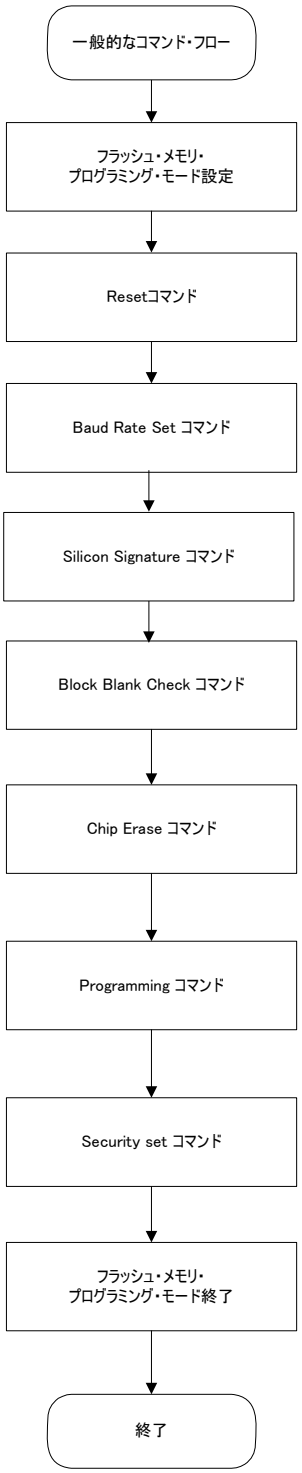
なお，このマニュアルではChecksum errorやNACKを受信した際は即時異常終了として扱っていますが，実際にプログラマを設計する際は，Checksum errorやNACKが発生したコマンド送信直前のウエイトからリトライしても構いません。ただし，無限にリトライを繰り返さないようにリトライの回数制限を設けることを推奨します。

また，上記ステータス・コード一覧には出てきませんが，各種タイムアウト・エラー（BUSYのタイムアウト，UART通信時のデータ・フレーム受信のタイムアウトなど）が発生した場合は，一度78K0R/Kx3に対して電源遮断処理（2.6　**ターゲットの電源遮断処理**参照）を行ってから改めて接続することを推奨します。

第3章　プログラマの基本動作

プログラマによるフラッシュ・メモリ書き換えの一般的なコマンド・フローを図3－1に示します。

図3－1　書き換え時の一般的なコマンド・フロー



備考　そのほかに Verify**コマンドや**Checksum**コマンドのサポートが可能です。**

第4章　コマンド／データ・フレーム・フォーマット

プログラマと78K0R/Kx3間でデータを送受信する際，プログラマがコマンドを送信する場合は，コマンド・フレームを使用します。78K0R/Kx3からプログラマに書き込みデータやベリファイ・データなどを送信する場合は，データ・フレームを使用します。これらのフレームには，転送データの信頼性を向上させるために，フレーム単位でヘッダ，フッタ，データ長情報，チェックサムを付けて送受信します。

次に両フレーム・フォーマットを示します。

図4－1　コマンド・フレームのフォーマット

SOH （1 バイト）	LEN （1 バイト）	COM （1 バイト）	コマンド情報（可変長） （最大 255 バイト）	SUM （1 バイト）	ETX （1 バイト）

図4－2　データ・フレームのフォーマット

STX （1 バイト）	LEN （1 バイト）	データ（可変長） （最大 256 バイト）	SUM （1 バイト）	ETX or ETB （1 バイト）

表4－1　各フレームの記号説明

記号	値	内　　容
SOH	01H	コマンド・フレームのヘッダ
STX	02H	データ・フレームのヘッダ
LEN	－	データ長情報（00H = 256 を示します）。 コマンド・フレームの場合　：COM＋コマンド情報の長さ データ・フレームの場合　　：データ・フィールドの長さ
COM	－	コマンド番号
SUM	－	フレーム内のチェックサム・データ。 初期値 00H から計算対象すべてのデータを１バイトごとに減算した値（ボローは無視）。計算対象を次に示します。 コマンド・フレームの場合　：LEN＋COM＋コマンド情報すべて データ・フレームの場合　　：LEN＋データすべて
ETB	17H	データ・フレームの最終フレーム以外のフッタ
ETX	03H	コマンド・フレームのフッタ，またはデータ・フレームの最終フレームのフッタ

フレーム内のチェックサム（SUM）の計算例を次に示します。

【コマンド・フレームの場合】

Statusコマンド・フレームは次のようになります。この場合，コマンド情報がないので，チェックサム計算の対象になるのはLENとCOMです。

SOH	LEN	COM	SUM	ETX
01H	01H	70H	Checksum	03H
	チェックサム計算対象			

この場合，チェックサム・データは次のように計算します。

00H（初期値）－ 01H（LEN） － 70H（COM）= 8FH （ボロー無視。下位8ビットのみ）

よって， Statusコマンド・フレームは最終的に次のようになります。

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

【データ・フレームの場合】

たとえば，次のようなデータ・フレームを送信する場合，チェックサム計算の対象はLENからD4までです。

STX	LEN	D1	D2	D3	D4	SUM	ETX
02H	04H	FFH	80H	40H	22H	Checksum	03H
	チェックサム計算対象						

この場合，チェックサム・データは次のように計算します。

00H（初期値）－ 04H（LEN）－ FFH（D1）－ 80H（D2）－ 40H（D3）－ 22H（D4）
= 1BH （ボロー無視。下位8ビットのみ）

よって，このデータ・フレームは最終的に次のようになります。

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

データ・フレームを受信した場合も同様にチェックサム・データを計算して，その値が受信したSUMフィールドの値と同じであるか否かでチェックサム・エラーを検出できます。たとえば，次のようなデータ・フレームを受信した場合は，チェックサム・エラーと見なすことができます。

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

↑本来なら 1BH

4.1　コマンド・フレーム送信処理

コマンド・フレームを送信する処理のフロー・チャートについては，6.1　**コマンド・フレーム送信処理のフロー・チャート**をお読みください。

4.2　データ・フレーム送信処理

データ・フレームとして送信するものは，書き込みデータ・フレーム（ユーザ・プログラム），ベリファイ・データ・フレーム（ユーザ・プログラム），セキュリティ・データ・フレーム（セキュリティ・フラグ）があります。

データ・フレームを送信する処理のフロー・チャートについては，6.2　**データ・フレーム送信処理のフロー・チャート**をお読みください。

4.3　データ・フレーム受信処理

データ・フレームとして受信するものは，ステータス・フレーム，シリコン・シグネチャ・データ・フレーム，バージョン・データ・フレーム，チェックサム・データ・フレームがあります。

データ・フレームを受信する処理のフロー・チャートについては，6.3　**データ・フレーム受信処理のフロー・チャート**をお読みください。

第5章　コマンド処理説明

5. 1　Statusコマンド

5. 1. 1　説　　明

書き込み／消去などの各コマンド発行後，一定時間内に78K0R/Kx3は動作状態を通知するために自動的にステータス・フレームを送信します。

プログラマが各コマンド発行後，通信の問題などで78K0R/Kx3でStatusコマンド・フレームを正しく受信できなかった場合などは，78K0R/Kx3ではステータスの設定を行いません。よって，ステータス・フレームではなく，ビジー応答（FFH）を受信する場合があります。この場合は，各コマンドをリトライしてください。

5. 1. 2　ステータス・フレーム

各コマンドに対するステータス・フレームは図5－1のようになります。

図5－1　Statusコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data			SUM	ETX
02H	n	ST1	・・・	STn	Checksum	03H

備考1. ST1 - STn　：ステータス#1 - ステータス#n

2. ステータス・フレームの長さは，78K0R/Kx3に送信される書き込み／消去などの各コマンドによって異なります。

5. 2　Resetコマンド

5. 2. 1　説　　明

通信方式設定後に，プログラマと78K0R/Kx3間の通信が確立されたことを確認します。

プログラマと78K0R/Kx3は同じボー・レートである必要がありますが，78K0R/Kx3は自身のボー・レート生成クロック周波数が判別できないためにボー・レートが設定できません。よって，プログラマから9600 bpsでの“00H”を2回送信し，78K0R/Kx3はその“00H”のロウ・レベル幅を測定し2回の平均値を計算することで初めて自身のボー・レート生成クロック周波数を判別できます。それによって，ボー・レートの設定が可能になり同期検出が行えるようになります。

5. 2. 2　コマンド・フレームとステータス・フレーム

Resetコマンドのコマンド・フレームは図5－2，そのコマンドに対するステータス・フレームは図5－3のようになります。

図5－2　Resetコマンド・フレーム（プログラマから78K0R/Kx3へ）

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

図5－3　Resetコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	1	ST1	Checksum	03H

備考　ST1　：同期検出結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 4　Resetコマンドをお読みください。

5. 3　Baud Rate Setコマンド

5. 3. 1　説　　明

UART通信でのボー・レートの変更を行います（初期値9600 bps）。

Baud Rate Setコマンドのあとには，変更したボー・レートでの同期確認のためにResetコマンドを実行する必要があります。

ボー・レート設定データは1バイトの数値で表されます。

5. 3. 2　コマンド・フレームとステータス・フレーム

Baud Rate Setコマンドのコマンド・フレームは図5－4，そのコマンドに対するステータス・フレームは図5－5のようになります。

図5－4　Baud Rate Setコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報[注]				SUM	ETX
01H	05H	9AH	D01	D02H	D02L	D03	sum	03H

注　コマンド情報の設定の詳細は表5－1を参照してください。表5－1以外のデータを設定した場合，タイムアウト・エラーとなります。

タイムアウト・エラーが発生した場合は，ハードウエア・リセットを実行し，再度フラッシュ・メモリ・プログラミング・モードに設定してください。

備考　D01　　：同期補正モード
D02H, D02L　　：ボー・レート設定
D03　　：ノイズ・フィルター設定

表5－1　コマンド情報の設定

同期補正モード	D01	D02H	D02L	D03
マイコン補正モード	00H	00H 固定	0AH 固定（115200 bps）	ノイズ・フィルター 00H ：オフ
プログラマ補正モード	01H	注	注	01H ：オン

注　以下の計算式で求められるk値をhexでD02H/D02Lに代入してください。k値は0003H以上にしてください。

k =（8×10^6×E）/BAUD RATE

E：フラッシュ引き込み時の78K0RのREADYパルス（9600 bps）誤差

例1：READYパルス（ロウ・レベル 9 bit@9600 bps）長に対し，0%誤差の場合

（READYパルス= 937.5 μsの場合）

250000 bps設定時

E=1.00

k=0020H

D02H=00H

D02L=20H

例2：READYパルス（ロウ・レベル 9 bit@9600 bps）長に対し，＋5%誤差の場合

（READYパルス= 984.375 μsの場合）

250000 bps設定時

E=1.05

k=0021H

D02H=00H

D02L=21H

例3：READYパルス（ロウ・レベル 9 bit@9600 bps）長に対し，－5%誤差の場合

（READYパルス= 890.625 μsの場合）

250000 bps設定時

E=0.95

k=001EH

D02H=00H

D02L=1EH

図5－5　Baud Rate Setコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	checksum	03H

備考　ST1 ：同期検出結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては， 6. 5　Baud Rate Setコマンドをお読みください。

5. 4　Chip Eraseコマンド

5. 4. 1　説　　明

全フラッシュ・メモリの内容を消去します。また，チップ消去処理によりセキュリティ設定処理で設定されたすべての情報を初期化できます。ただし，セキュリティ設定により消去禁止となっている場合は消去できません（5. 12　Security Setコマンド参照）。

5. 4. 2　コマンド・フレームとステータス・フレーム

Chip Eraseコマンドのコマンド・フレームは図5－6，そのコマンドに対するステータス・フレームは，図5－7のようになります。

図5－6　Chip Eraseコマンド・フレーム（プログラマから78K0R/Kx3へ）

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

図5－7　Chip Eraseコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　：チップ消去結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 6　Chip Eraseコマンドをお読みください。

5. 5　Block Eraseコマンド

5. 5. 1　説　　明

指定したブロック番号のフラッシュ・メモリの内容を消去します。

ブロックの指定は，消去開始ブロックの先頭アドレスから，消去終了ブロックの最終アドレスの指定で行い，連続した複数のブロックの設定が可能です。

ただし，セキュリティ設定により消去禁止となっている場合は消去できません（5. 12　Security Setコマンド参照）。

5. 5. 2　コマンド・フレームとステータス・フレーム

Block Eraseコマンドのコマンド・フレームは図5－8，そのコマンドに対するステータス・フレームは図5－9のようになります。

図5－8　Block Eraseコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	22H （Block Erase）	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考　SAH - SAL　：ブロック消去開始アドレス（任意のブロックの先頭アドレス）
SAH　：開始アドレスHigh（ビット23 － ビット16）
SAM　：開始アドレスMiddle（ビット15 － ビット8）
SAL　：開始アドレスLow（ビット7 － ビット0）
EAH - EAL　：ブロック消去終了アドレス（任意のブロックの最終アドレス）
EAH　：最終アドレスHigh（ビット23 － ビット16）
EAM　：最終アドレスMiddle（ビット15 － ビット8）
EAL　：最終アドレスLow（ビット7 － ビット0）

図5－9　Block Eraseコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　：ブロック消去結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 7　Block Eraseコマンドをお読みください。

5. 6　Programmingコマンド

5. 6. 1　説　　明

書き込み開始アドレス，書き込み終了アドレスを送信したあとに，書き込みデータを送信しユーザ・プログラムをフラッシュ・メモリに書き込みます。最終データ送信後，書き込みが終了すると内部ベリファイを実行します。

書き込み開始／終了アドレスは，ブロックの開始／終了アドレス単位でのみ設定できます。

最終データ送信後のステータス・フレーム（ST1，ST2）が両方ともACKであれば，78K0R/Kx3のファームウエアは自動的に内部ベリファイを実行するので，さらにこの内部ベリファイに対するステータスの確認が必要となります。

5. 6. 2　コマンド・フレームとステータス・フレーム

Programmingコマンドのコマンド・フレームは図5－10，そのコマンドに対するステータス・フレームは図5－11のようになります。

図5－10　Programmingコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	40H （Programming）	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考　SAH - SAL　：書き込み開始アドレス
　　EAH - EAL　：書き込み終了アドレス

図5－11　Programmingコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(a)　：コマンド受信結果

5. 6. 3　データ・フレームとステータス・フレーム

書き込みを行うデータのデータ・フレームは図5－12，そのデータに対するステータス・フレームは図5－13のようになります。

図5－12　書き込みを行うデータ・フレーム（プログラマから78K0R/Kx3へ）

STX	LEN	Data	SUM	ETX/ETB
02H	00H-FFH （00H＝256）	Write Data	Checksum	03H/17H

備考　Write Data　：書き込むユーザ・プログラム

図5－13　データ・フレームに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(b)　：データ受信確認結果
　　ST2(b)　：書き込み結果

5. 6. 4　全データ転送完了とステータス・フレーム

全データ転送完了後のステータス・フレームは図5－14のようになります。

図5－14　全データ転送完了後のステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(c)　：内部ベリファイ結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 8　Programmingコマンドをお読みください。

5. 7　Verifyコマンド

5. 7. 1　説　　明

指定したアドレス範囲のデータに対して，プログラマから送信したデータと78K0R/Kx3から読み出したデータ（リード・レベル）を比較し，一致しているかを確認します。

ベリファイ開始アドレス／ベリファイ終了アドレスは，ブロックの開始アドレス／終了アドレス単位でのみ設定できます。

5. 7. 2　コマンド・フレームとステータス・フレーム

Verifyコマンドのコマンド・フレームは図5－15，そのコマンドに対するステータス・フレームは図5－16のようになります。

図5－15　Verifyコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	13H （Verify）	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考　SAH - SAL　：ベリファイ開始アドレス
　　EAH - EAL　：ベリファイ終了アドレス

図5－16　Verifyコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(a)　：コマンド受信結果

5. 7. 3　データ・フレームとステータス・フレーム

ベリファイを行うデータのデータ・フレームは図5－17，そのデータに対するステータス・フレームは図5－18のようになります。

図5－17　ベリファイを行うデータのデータ・フレーム（プログラマから78K0R/Kx3へ）

STX	LEN	Data	SUM	ETX/ETB
02H	00H-FFH （00H＝256）	Verify Data	Checksum	03H/17H

備考　Verify Data　：ベリファイを行うユーザ・プログラム

図5－18　データ・フレームに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(b)　　：データ受信確認結果
　　ST2(b)　　：ベリファイ結果[注]

注　ベリファイ結果は指定したアドレス範囲の途中でベリファイ・エラーが発生しても，ステータスとしては必ずACKを返し，最終データのベリファイ結果にすべてのエラーが反映されます。したがって，指定したアドレス範囲すべてのベリファイが終了した時点でのみ，ベリファイ・エラーが発生したかどうかを確認できます。

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 9　Verifyコマンドをお読みください。

5. 8　Block Blank Checkコマンド

5. 8. 1　説　　明

指定したブロック番号のフラッシュ・メモリのデータがブランク（消去状態）であるかを確認します。

ブロックの指定は，ブランク・チェック開始ブロックの先頭アドレスから，ブランク・チェック終了ブロックの最終アドレスの指定で行い，連続した複数のブロックの設定が可能です。

5. 8. 2　コマンド・フレームとステータス・フレーム

Block Blank Checkコマンドのコマンド・フレームは図5－19，そのコマンドに対するステータス・フレームは図5－20のようになります。

図5－19　Block Blank Checkコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報							SUM	ETX
01H	08H	32H （Block Blank Check）	SAH	SAM	SAL	EAH	EAM	EAL	D01	Checksum	03H

備考　SAH - SAL　：ブロック・ブランク・チェック開始アドレス（任意のブロックの先頭アドレス）
SAH　：開始アドレスHigh（ビット23 － ビット16）
SAM　：開始アドレスMiddle（ビット15 － ビット8）
SAL　：開始アドレスLow（ビット7 － ビット0）
EAH － EAL　：ブロック・ブランク・チェック終了アドレス（任意のブロックの最終アドレス）
EAH　：最終アドレスHigh（ビット23 － ビット16）
EAM　：最終アドレスMiddle（ビット15 － ビット8）
EAL　：最終アドレスLow（ビット7 － ビット0）
D01　：00H：単独でブロック・ブランク・チェックを行う場合。
01H：チップ消去前に全領域のブランク・チェックをする場合

図5－20　Block Blank Checkコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　：ブロック・ブランク・チェック結果

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 10　Block Blank Checkコマンドをお読みください。

5. 9　Silicon Signatureコマンド

5. 9. 1　説　　明

デバイスの書き込みプロトコル情報（シリコン・シグネチャ）やセキュリティ・フラグ情報を読み出します。

たとえば，プログラマが78K0R/Kx3と異なる書き込みプロトコルを同時にサポートする場合に，Silicon Signatureコマンドを実行し，2バイト目から3バイト目までの値に従い，適切なプロトコルを選択します。

5. 9. 2　コマンド・フレームとステータス・フレーム

Silicon Signatureコマンドのコマンド・フレームは図5－21，そのコマンドに対するステータス・フレームは図5－22のようになります。

図5－21　Silicon Signatureコマンド・フレームのフォーマット（プログラマから78K0R/Kx3へ）

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

図5－22　Silicon Signatureコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　：コマンド受信結果

5.9.3　シリコン・シグネチャ・データ・フレーム

シリコン・シグネチャ・データのデータ・フレームは図5－23のようになります。

図5－23　シリコン・シグネチャ・データ・フレーム（78K0R/Kx3からプログラマへ）

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

Data（続き）						SUM	ETX
SCF	BOT	FSWSH	FSWSL	FSWEH	FSWEL	checksum	03H

備考1.　n（LEN）　：データ長
VEN　：ベンダー・コード（NEC：10H）
MET　：マクロ拡張コード
MSC　：マクロ機能コード
DEC1　：デバイス拡張コード1
DEC2　：デバイス拡張コード2
UAE　：ユーザ・フラッシュROM最終アドレス（3バイト）
DEV　：デバイス名（10バイト）
SCF　：セキュリティ・フラグ情報
BOT　：ブート・ブロック番号
FSWSH　：フラッシュ・シールド・ウインドウ(FSW)開始ブロック上位8ビット側
FSWSL　：フラッシュ・シールド・ウインドウ(FSW)開始ブロック下位8ビット側
FSWEH　：フラッシュ・シールド・ウインドウ(FSW)終了ブロック上位8ビット側
FSWEL　：フラッシュ・シールド・ウインドウ(FSW)終了ブロック下位8ビット側

2.　上記ベンダー・コード（VEN），拡張コード（MET），機能コード（MSC），デバイス拡張コード1（DEC1），デバイス拡張コード2（DEC2）は，下位7ビットをデータ本体，上位1ビットを奇数パリティとして使用します。次ページに例を示します。

表5－2　シリコン・シグネチャ・データの例

フィールド名	内容	長さ（バイト）	シグネチャデータの例	実際の値	パリティ付加
VEN	ベンダー・コード（NEC）	1	10H (00010000B)	10H	あり
MET	マクロ拡張コード	1	7FH (01111111B)	7FH	あり
MSC	マクロ機能コード	1	04H (01000000B)	04H	あり
DEC1	デバイス拡張コード 1	1	DCH (11011100B)	DCH	あり
DEC2	デバイス拡張コード 2	1	FDH (11111101B)	FDH	あり
UAE	ユーザ・フラッシュ ROM 最終アドレス	3	FFH (11111111B)	00FFFFH	なし
			FFH (11111111B)		
			00H (00000000B)		
DEV	デバイス名	10	44H (01000100B) = ‘D’	‘D’	なし
			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	セキュリティ・フラグ情報	1	任意	左欄に同じ	なし
BOT	ブート・ブロック・番号（固定値）	1	01H (00000001B)	01H	なし
FSWS(H)	フラッシュ・シールド・ウインドウ開始ブロック　上位 8 ビット	1	任意	左欄に同じ	なし
FSWS(L)	フラッシュ・シールド・ウインドウ開始ブロック　下位 8 ビット	1	任意	左欄に同じ	なし
FSWE(H)	フラッシュ・シールド・ウインドウ終了ブロック　上位 8 ビット	1	任意	左欄に同じ	なし
FSWE(L)	フラッシュ・シールド・ウインドウ終了ブロック　下位 8 ビット	1	任意	左欄に同じ	なし

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 11　Silicon Signature**コマンドをお読みください。**

5. 9. 4　78K0R/Kx3シリコン・シグネチャー覧

表5－3　78K0R/Kx3のシリコン・シグネチャ・データー覧

項　　目	内　　容	長さ（バイト）	データ（Hex）
ベンダー・コード	NEC	1	10
拡張コード	拡張コード	1	7F
機能情報	機能情報	1	04
デバイス情報	デバイス情報	2	DC
			FD
内蔵フラッシュ ROM の最終アドレス	アドレスの下位バイトから送信	3	注 1
デバイス名（μPD）	78F1142 / 78F1152 / 78F1162 78F1143 / 78F1153 / 78F1163 78F1144 / 78F1154 / 78F1164 78F1145 / 78F1155 / 78F1165 78F1146 / 78F1156 / 78F1166 78F1167 / 78F1168	10	注 2
セキュリティ情報	セキュリティ情報	1	任意
ブート・ブロック番号	現在，選択されているブート・クラスタの最終ブロック番号	1	01
FSW ブロック番号	FSW 情報	4	任意

注1.　内蔵フラッシュROMの最終アドレス・リスト

項　　目	内　　容	長さ（バイト）	データ（Hex）
内蔵フラッシュ ROM の最終アドレス	64 K バイト（00FFFFH）	3	FFFF00
	96 K バイト（017FFFH）		FF7F01
	128 K バイト（01FFFFH）		FFFF01
	192 K バイト（02FFFFH）		FFFF02
	256 K バイト（03FFFFH）		FFFF03
	384 K バイト（05FFFFH）		FFFF05
	512 K バイト（07FFFFH）		FFFF07

（注2は次ページにあります。）

注2．　デバイス名リストを次に示します。

デバイス名リスト（1/4）

項　　目	内　　容	長さ（バイト）	実際の値
デバイス名	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 = ‘ ‘

デバイス名リスト（2/4）

項　　目	内　　容	長さ（バイト）	実際の値
デバイス名	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 = ‘ ‘

デバイス名リスト（3/4）

項　　目	内　　容	長さ（バイト）	実際の値
デバイス名	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 = ‘ ‘

デバイス名リスト（4/4）

項　　目	内　　容	長さ（バイト）	実際の値
デバイス名	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コマンド

5. 10. 1　説　　明

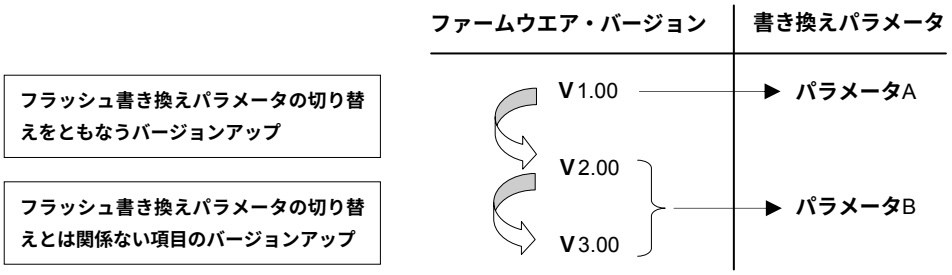
78K0R/Kx3のデバイス・バージョン，ファームウエア・バージョン情報を取得します。

デバイス・バージョンは，00H固定です。

書き換え用パラメータを78K0R/Kx3のファームウエア・バージョンに従い，切り替える必要がある場合に，このコマンドを使用します。

注意　フラッシュ書き換え用パラメータの変更とは関係ないファームウエア改版時も，ファームウエア・バージョンが更新される場合があります（このとき，ファームウエア・バージョン更新の通知は行いません）。

例　ファームウエア・バージョンと書き換えパラメータ



5. 10. 2　コマンド・フレームとステータス・フレーム

Version Getコマンドのコマンド・フレームは図5－24，そのコマンドに対するステータス・フレームは図5－25のようになります。

図5－24　Version Getコマンド・フレーム（プログラマから78K0R/Kx3へ）

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

図5－25　Version Getコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　：コマンド受信結果

5. 10. 3　バージョン・データ・フレーム

バージョン・データのデータ・フレームは図5－26のようになります。

図5－26　バージョン・データ・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data						SUM	ETX
02H	06H	DV1	DV2	DV3	FV1	FV2	FV3	Checksum	03H

備考　DV1　：デバイス・バージョン整数値（00H固定）
DV2　：デバイス・バージョン小数点第一位（00H固定）
DV3　：デバイス・バージョン小数点第二位（00H固定）
FV1　：ファームウエア・バージョン整数値
FV2　：ファームウエア・バージョン小数点第一位
FV3　：ファームウエア・バージョン小数点第二位

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 12　Version Getコマンドをお読みください。

5. 11　Checksumコマンド

5. 11. 1　説　　明

指定された領域のデータのチェックサム・データを取得します。

チェックサム計算の開始／終了アドレスは，フラッシュ・メモリの先頭からブロック単位（2 Kバイト）ごとの固定アドレスを指定してください。

チェックサム・データは，指定されたアドレス範囲のデータを1バイト単位で順次初期値0000Hから引き算したものです。

5. 11. 2　コマンド・フレームとステータス・フレーム

Checksumコマンドのコマンド・フレームは図5－27，そのコマンドに対するステータス・フレームは図5－28のようになります。

図5－27　Checksumコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	B0H （Checksum）	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考　SAH-SAL　：チェックサム計算開始アドレス
EAH-EAL　：チェックサム計算終了アドレス

図5－28　Checksumコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

備考　ST1　　：コマンド受信結果

5. 11. 3　チェックサム・データ・フレーム

チェックサム・データのデータ・フレームは図5－29のようになります。

図5－29　チェックサム・データ・フレーム（78K0R/Kx3からプログラマへ）

STX	LEN	Data		SUM	ETX
02H	02H	CK1	CK2	Checksum	03H

備考　CK1　　：チェックサム・データの上位8ビット
CK2　　：チェックサム・データの下位8ビット

プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 13　Checksumコマンドをお読みください。

5. 12　Security Setコマンド

5. 12. 1　説　　明

セキュリティに関する設定（書き込み，ブロック消去，チップ消去，ブート・ブロック書き換えの許可／禁止，フラッシュ・シールド・ウインドウの開始・終了ブロック番号の設定）を行います。Security Setコマンドでこれらの設定を行うことで，第三者からのフラッシュの書き換えを制限したり，セルフ・プログラミング時の書き換え領域を指定します。

注意　セキュリティ設定後も許可から禁止への追加設定が可能です。ただし禁止から許可への変更は行えず，実行しようとした場合，Protect error（10H）が発生します。禁止から許可への設定変更が必要な場合は，Chip Eraseコマンドの実行によって全セキュリティ・フラグの初期化をする必要があります（Block Eraseコマンドでは，セキュリティ・フラグの初期化はできません）。そのあとに設定変更を行ってください。

ただし，チップ消去禁止，またはブート・ブロック書き換え禁止の設定をした場合，チップ消去自体が不可能になり，プログラマからは消去ができなくなります。プログラマの仕様としては，チップ消去禁止の設定を行う前に，設定実行の再確認をすることを推奨します。

5. 12. 2　コマンド・フレームとステータス・フレーム

Security Setコマンドのコマンド・フレームは図5－30，そのコマンドに対するステータス・フレームは図5－31のようになります。

図5－30　Security Setコマンド・フレーム（プログラマから78K0R/Kx3へ）

SOH	LEN	COM	コマンド情報		SUM	ETX
01H	03H	A0H （Security Set）	00H （固定）	00H （固定）	Checksum	03H

図5－31　Security Setコマンドに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(a)　：コマンド受信結果

5. 12. 3　データ・フレームとステータス・フレーム

セキュリティ・データのデータ・フレームは図5－32，そのデータに対するステータス・フレームは図5－33のようになります。

図5－32　セキュリティ・データ・フレーム（プログラマから78K0R/Kx3へ）

STX	LEN	Data						SUM	ETX
02H	06H	FLG	BOT	FSWS(H)	FSWS(L)	FSWE(H)	FSWE(L)	Checksum	03H

備考1.　FLG　　：セキュリティ・フラグ
　　BOT　　：ブート・クラスタの最終ブロック番号（01H固定）
　　FSWS(H)：フラッシュ・シールド・ウインドウの開始ブロック番号上位8ビット（00H固定）
　　FSWS(L)：フラッシュ・シールド・ウインドウの開始ブロック番号下位8ビット
　　FSWE(H)：フラッシュ・シールド・ウインドウの終了ブロック番号上位8ビット（00H固定）
　　FSWE(L)：フラッシュ・シールド・ウインドウの終了ブロック番号下位8ビット
　2.　フラッシュ・シールド・ウインドウを設定しない場合は，FSWSは0000H，終了ブロックはターゲット・デバイスのエンド・ブロック番号を設定します。

図5－33　セキュリティ・データ書き込みに対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(b)　：セキュリティ・データ書き込み結果

5. 12. 4　内部ベリファイ確認とステータス・フレーム

内部ベリファイ確認に対するステータス・フレームは図5－34のようになります。

図5－34　内部ベリファイ確認に対するステータス・フレーム（78K0R/Kx3からプログラマへ）

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

備考　ST1(c)　：内部ベリファイ結果

セキュリティ・フラグ・フィールドの内容を次に示します。

表5－4　セキュリティ・フラグ・フィールドの内容

項　目	内　容
ビット7	1固定
ビット6	
ビット5	
ビット4	ブート・ブロック書き換え禁止フラグ（1：許可，0：禁止）
ビット3	1固定
ビット2	書き込み禁止フラグ（1：書き込み許可，0：書き込み禁止）
ビット1	ブロック消去禁止フラグ（1：ブロック消去許可，0：ブロック消去禁止）
ビット0	チップ消去禁止フラグ（1：チップ消去許可，0：チップ消去禁止）

セキュリティ・フラグ・フィールドの設定と，各動作の禁止／許可の関係を次に示します。

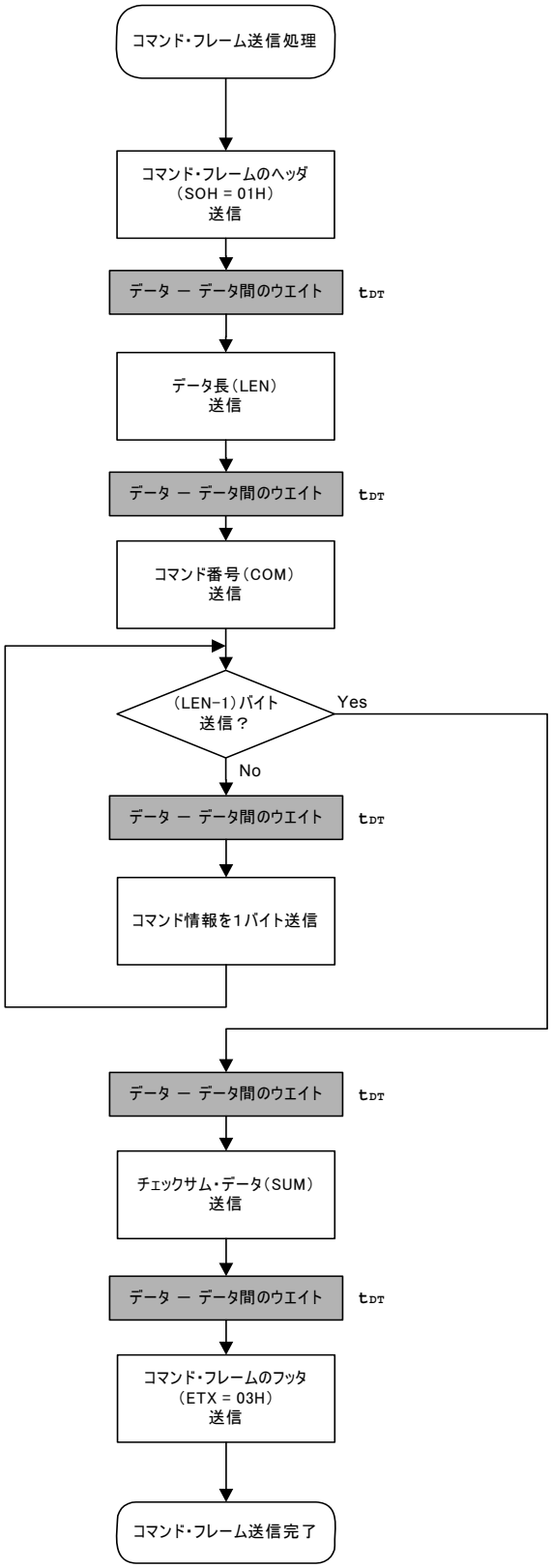
表5－5　セキュリティ・フラグ・フィールドと各動作の禁止／許可

動作モード	フラッシュ・メモリ・プログラミング・モード			セルフ・プログラミング・モード
コマンド / セキュリティ設定項目	セキュリティ設定後のコマンド動作 ○：実行可能　×：実行不可 △：ブート領域への書き込み，またはブロック消去は不可 ブート領域以外への書き込み，またはブロック消去は可能			・セキュリティ設定値にかかわらず，全コマンド実行可能 ・セキュリティ設定値の保持のみ可能
	Programming	Chip Erase	Block Erase	
書き込み禁止	×	○	×	
チップ消去禁止	○	×	×	
ブロック消去禁止	○	○	×	
ブート・ブロック書き換え禁止フラグ	△	×	△	フラッシュ・メモリ・プログラミング・モード（オンボード／オフボード・プログラミング）時と同様

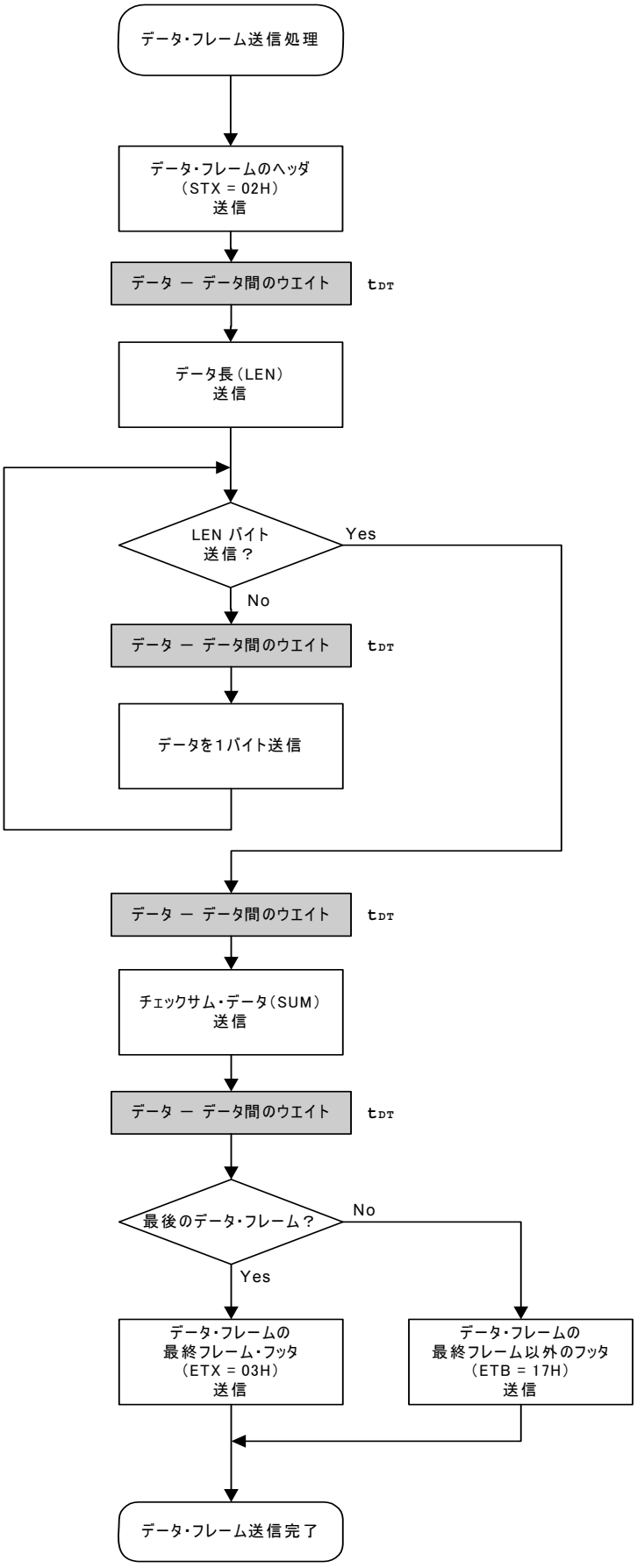
プログラマと78K0R/Kx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，6. 14　Security Setコマンドをお読みください。

第6章　UART通信方式

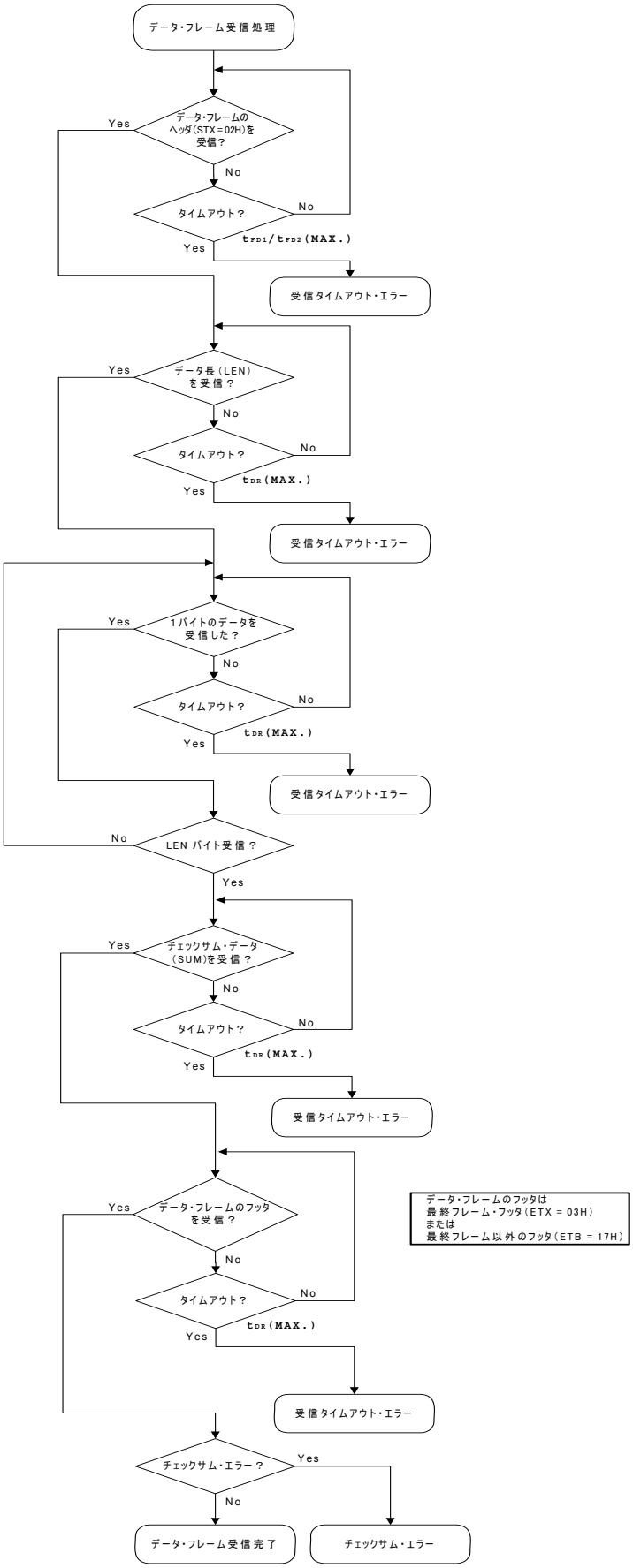
6. 1　コマンド・フレーム送信処理のフロー・チャート



6. 2　データ・フレーム送信処理のフロー・チャート



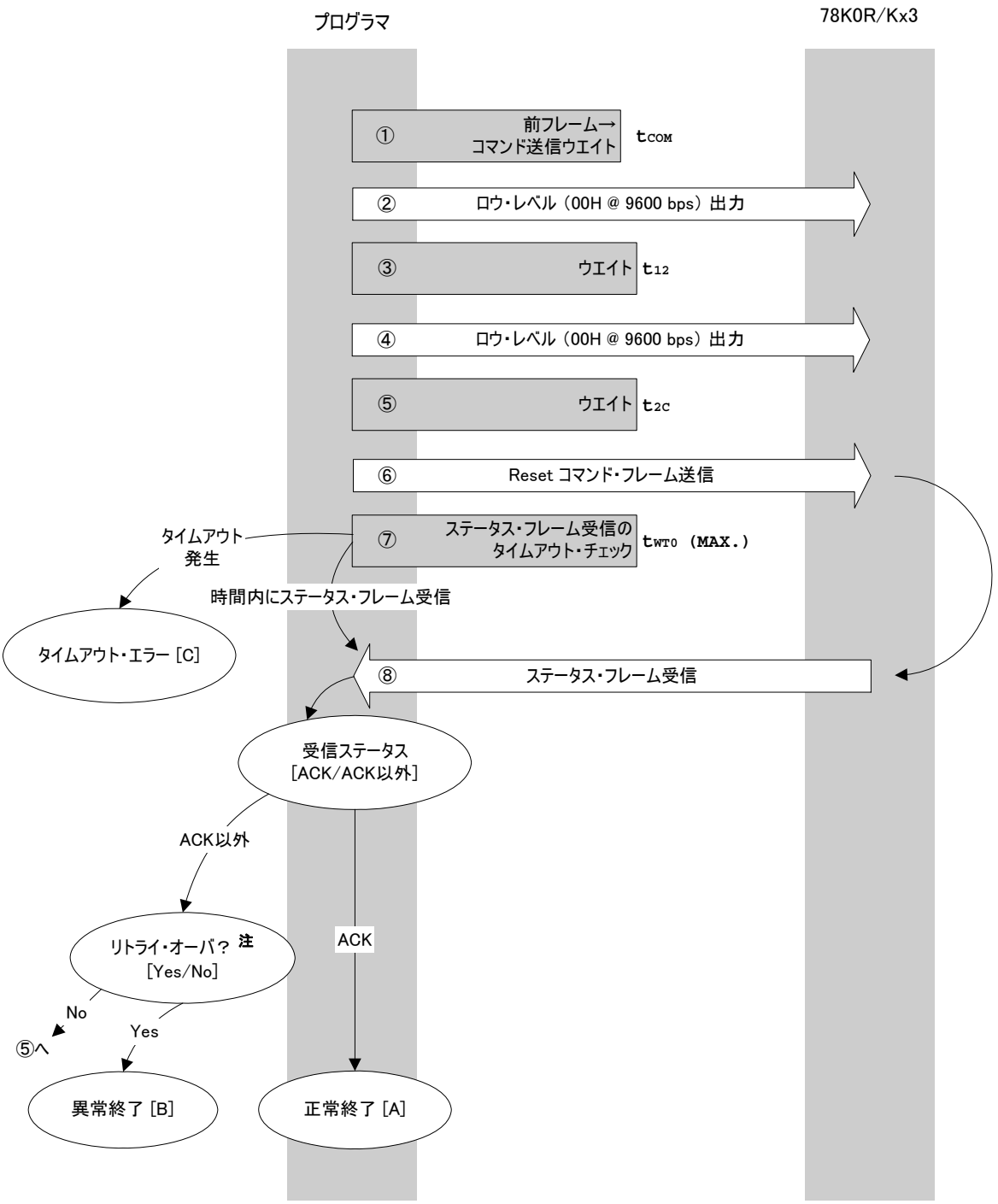
6. 3　データ・フレーム受信処理のフロー・チャート



6. 4　Reset**コマンド**

6. 4. 1　**処理手順チャート**

Resetコマンド処理手順



注　リセット・コマンドの送信は16回（MAX.）としてください。

6.4.2　処理手順説明

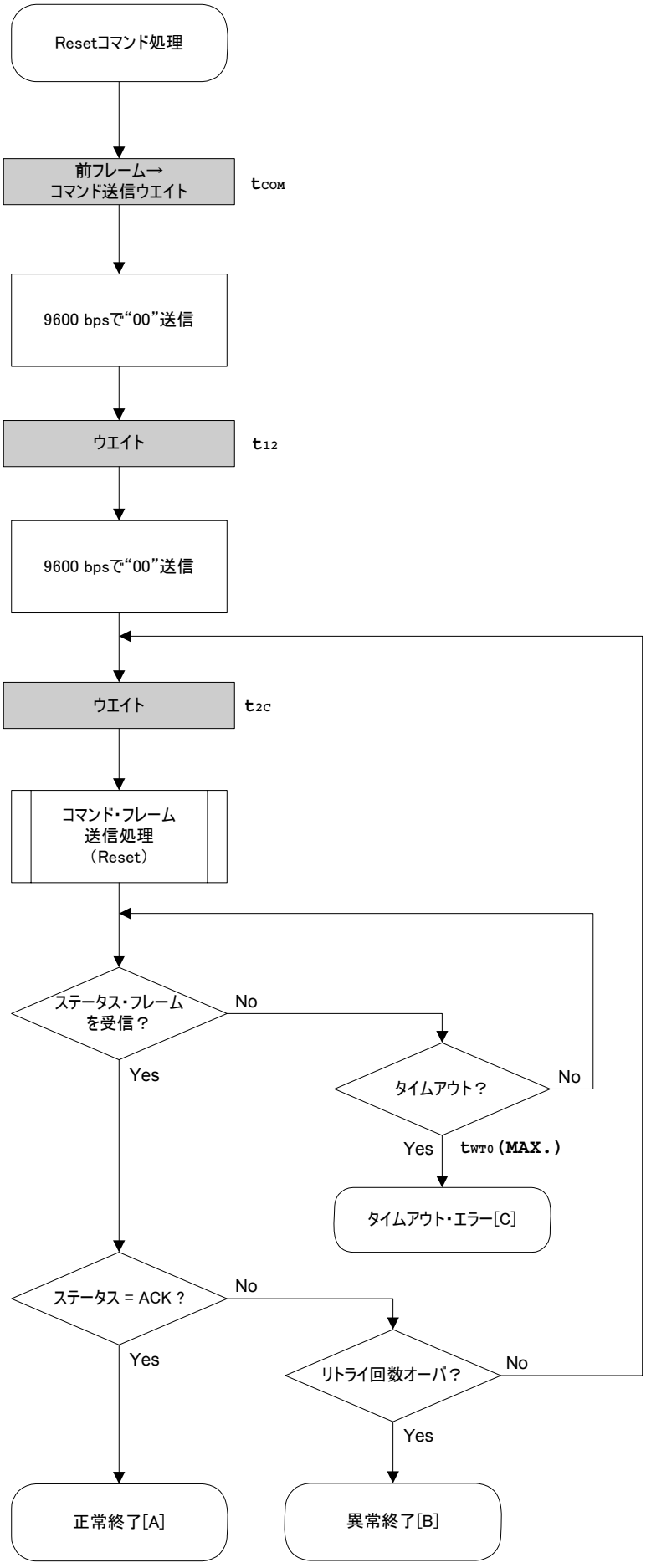
① 直前のフレームからコマンド処理開始前のウエイトをします（ウエイト時間t_{COM}）。
② ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
③ ウエイトします（ウエイト時間t_{12}）。
④ ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
⑤ ウエイトします（ウエイト時間t_{2C}）。
⑥ コマンド・フレーム送信処理にて Resetコマンド を送信します。
⑦ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合は タイムアウト・エラー [C] となります
（タイムアウト時間t_{WT0}（MAX.））。
⑧ ステータス・コードをチェックします。

ST1 = ACKの場合　：正常終了 [A] です。
ST1 = ACK以外の場合　：リトライ回数（t_{RS}）をチェックします。
リトライ・オーバでなければ⑤からやり直します。
リトライ・オーバであれば 異常終了 [B] です。

6.4.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，プログラマと78K0R/Kx3 間で同期が取れたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		－	規定の時間内にステータス・フレームが受信できませんでした。

6.4.4　フロー・チャート


Resetコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
9600 bpsで“00”送信
ウエイト
t12
9600 bpsで“00”送信
ウエイト
t2C
コマンド・フレーム
送信処理
（Reset）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
Yes
tWT0（MAX.）
タイムアウト・エラー[C]
ステータス = ACK ?
No
Yes
リトライ回数オーバ？
No
Yes
正常終了[A]
異常終了[B]

6.4.5　サンプル・プログラム

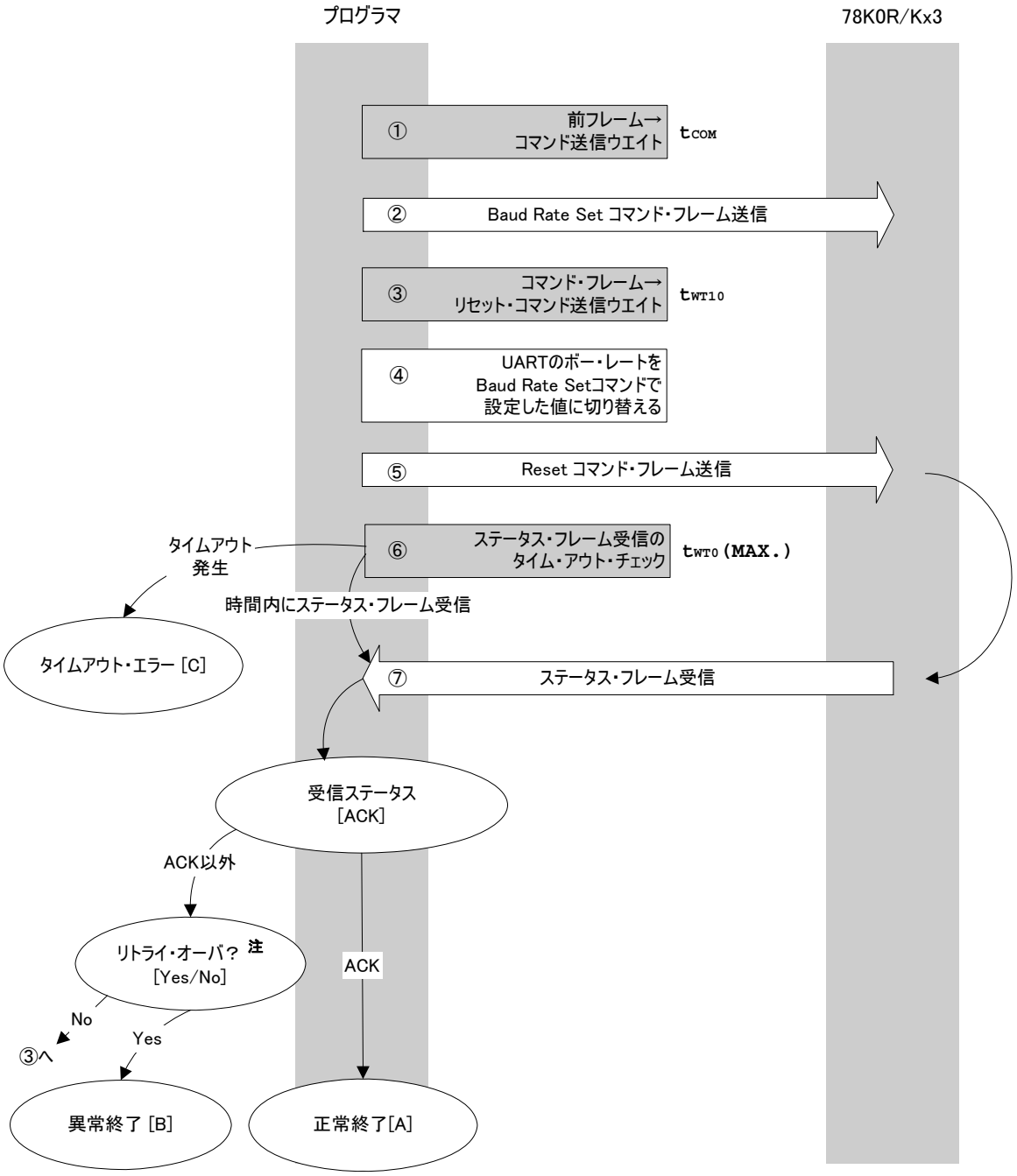
Resetコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Reset command                                                 */
/*                                                               */
/*****************************************************************/
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_reset(void)
{
   u16     rc;
   u32     retry;

   set_uart0_br(BR_9600);  // change to 9600bps

   fl_wait(tCOM);          // wait

   set_ua_dir_tx();        // Change Mono-wire UART transmit mode
   putc_ua(0x00);          // send 0x00 @ 9600bps

   fl_wait(t12);           // wait

   putc_ua(0x00);          // send 0x00 @ 9600bps
   set_ua_dir_rx();        // Change Mono-wire UART receive mode

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

           fl_wait(t2C);   // wait

           put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm); // send RESET command

           rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
           if (rc == FLC_DFTO_ERR)         // t.o. ?
                   break;                  // yes  // case [C]
           if (rc == FLC_ACK){             // ACK ?
                   break;                  // yes  // case [A]
           }
           else{
                   NOP();
           }
           //continue;                     // case [B] (if exit from loop)
   }
// 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　Baud Rate Setコマンド

6. 5. 1　処理手順チャート

Baud Rate Setコマンド処理手順



注　リセット・コマンドの送信は16回（MAX.）としてください。

6.5.2　処理手順説明

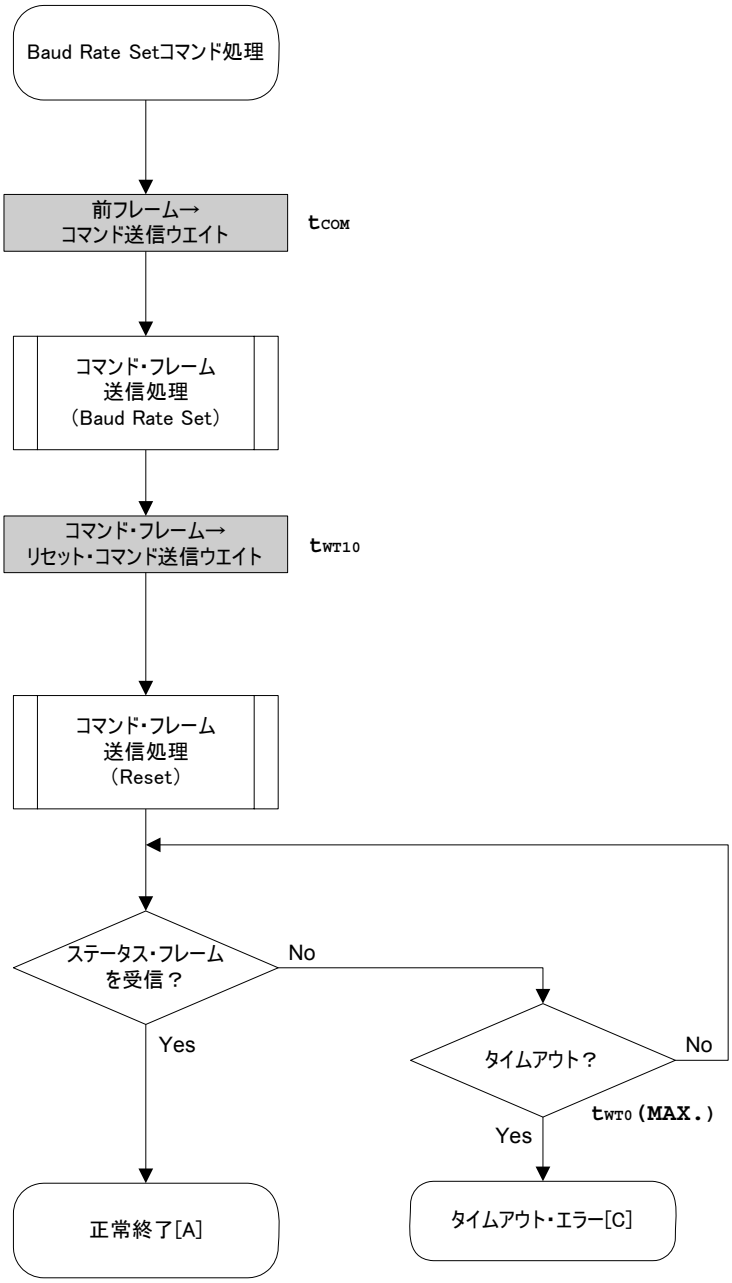
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Baud Rate Setコマンド を送信します。
③ コマンド送信からリセット・コマンド送信までのウエイトをします（ウエイト時間t_{WT10}）。
④ UART通信のボー・レートをBaud Rate Setコマンドで設定した値に切り替えます。
⑤ コマンド・フレーム送信処理にて Resetコマンド を送信します。
⑥ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウト発生であれば タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT0}(MAX.)）。
⑦ ステータス・コードはACKのはずですので， 正常終了[A] となります。

6.5.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，プログラマと 78K0R/Kx3 間で UART 通信速度の同期が取れたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C] 注		−	データ・フレームの受信でタイムアウトが発生しました。 ただし，このコマンドにおいて 78K0R/Kx3 では次の場合もこのエラーになります。 ・コマンド情報（D01, D02H, D02L, D03）が不正な場合 ・コマンド・フレームにチェックサム・エラーがあった場合 ・コマンド・フレームのデータ長（LEN）が不正の場合 ・コマンド・フレームのフッタ（ETX）がない場合 ・ボー・レート設定後，16 回分のコマンド・フレーム・データを受信しても Reset コマンドが検出できなかった場合

注　タイムアウト・エラーが発生した場合は，ハードウェア・リセットを実行し，再度フラッシュ・メモリ・プログラミング・モードに設定してください。

6.5.4　フロー・チャート


Baud Rate Setコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Baud Rate Set）
コマンド・フレーム→
リセット・コマンド送信ウエイト
tWT10
コマンド・フレーム
送信処理
（Reset）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT0 (MAX.)
Yes
正常終了[A]
タイムアウト・エラー[C]

6. 5. 5　サンプル・プログラム

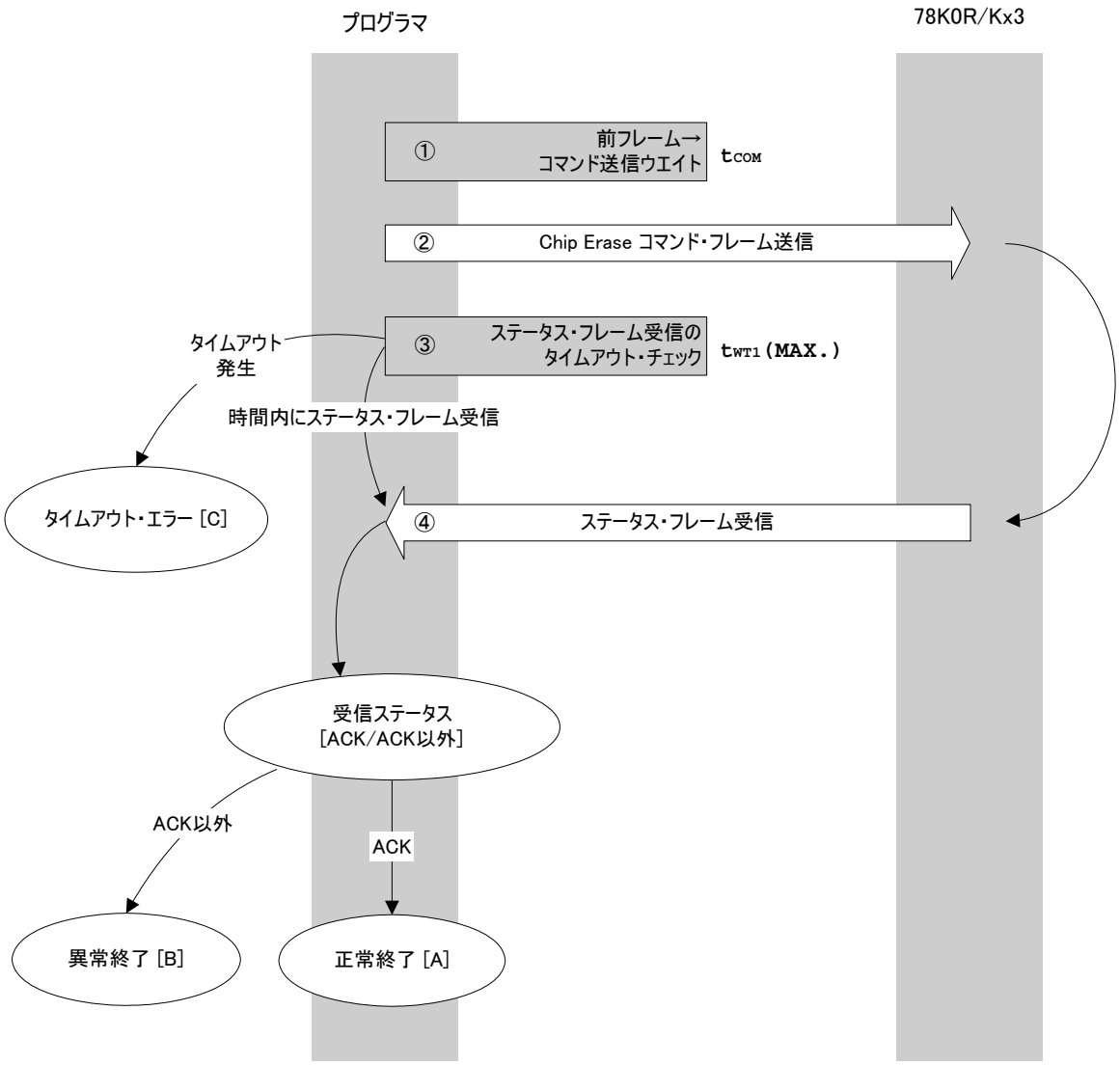
Baud Rate Setコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Set baudrate command                                          */
/*                                                               */
/*****************************************************************/
/* [i]  u8 brid     ... baudrate ID                              */
/* [r] u16          ... error code                               */
/*****************************************************************/
u16        fl_ua_setbaud(u8 brid)
{
   u16     rc;
   u8      br;
   u32     retry;

   fl_cmd_prm[0] = 0x00;   // "D01" : adjust by target device (115200bps)
   fl_cmd_prm[1] = 0x00;   // "D02" : adjust by target device (115200bps)
   fl_cmd_prm[2] = 0x0a;   // "D03" : (fixed value)
   fl_cmd_prm[3] = 0x01;   // "D04" : noise filter on

   fl_wait(tCOM);          // wait before sending command
   put_cmd_ua(FL_COM_SET_BAUDRATE, 1+4, fl_cmd_prm);// send "Baudrate Set" command
   set_flbaud(brid);       // change baud-rate
   set_uart0_br(brid);     // change baud-rate (h.w.)

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

           put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);// send RESET command
           rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX); // get status frame
           if (rc){
                   if (retry--)
                           continue;
                   else
                           return  rc;
           }
           break;          // got ACK !!

   }
// 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　Chip Erase**コマンド**

6. 6. 1　**処理手順チャート**

Chip Eraseコマンド処理手順



6.6.2　処理手順説明

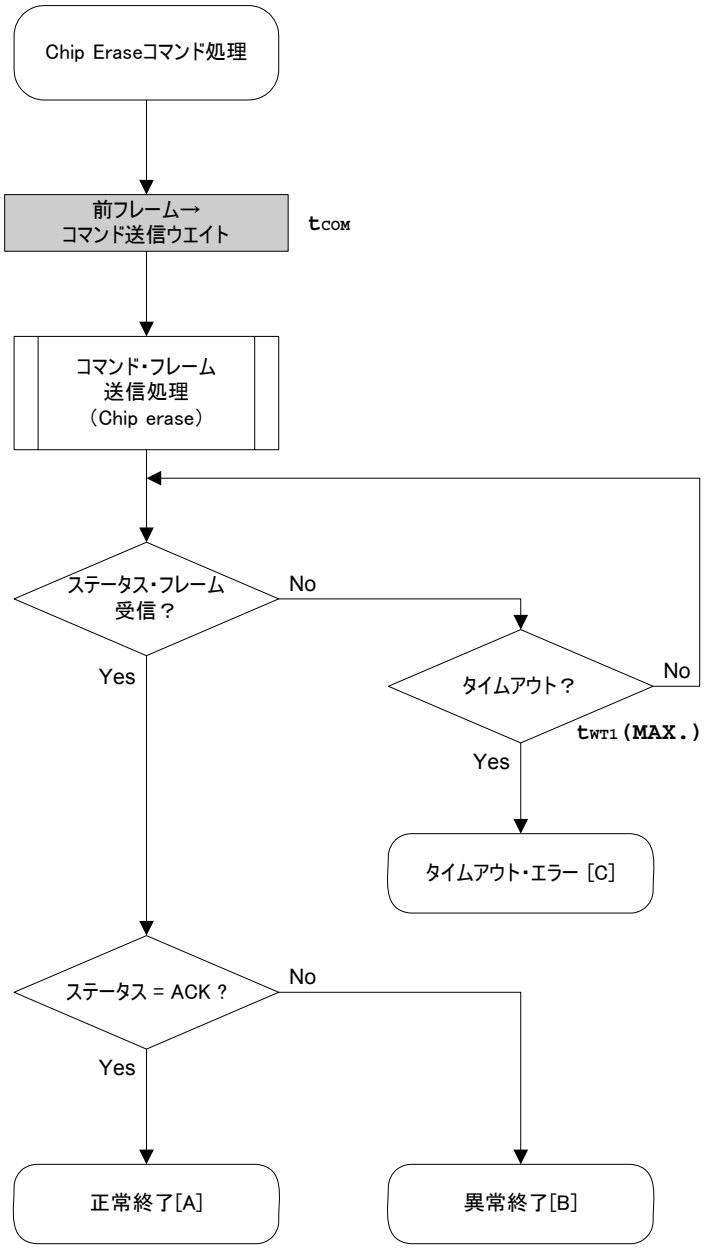
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Chip Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT1}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　：正常終了[A] です。
ST1 = ACK以外の場合　：異常終了[B] です。

6.6.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，チップ消去が正常に実行されたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で「チップ消去禁止」，「ブート・ブロック書き換え禁止」のいずれかの状態になっています。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
	MRG10 エラー	1AH	消去エラーが発生しました。
	MRG11 エラー	1BH	
	Write エラー	1CH	
タイムアウト・エラー [C]		−	規定の時間内にステータス・フレームの受信ができませんでした。

6.6.4　フロー・チャート


Chip Eraseコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Chip erase）
ステータス・フレーム
受信？
No
Yes
タイムアウト？
No
tWT1 (MAX.)
Yes
タイムアウト・エラー [C]
ステータス = ACK ?
No
Yes
正常終了[A]
異常終了[B]

6.6.5　サンプル・プログラム

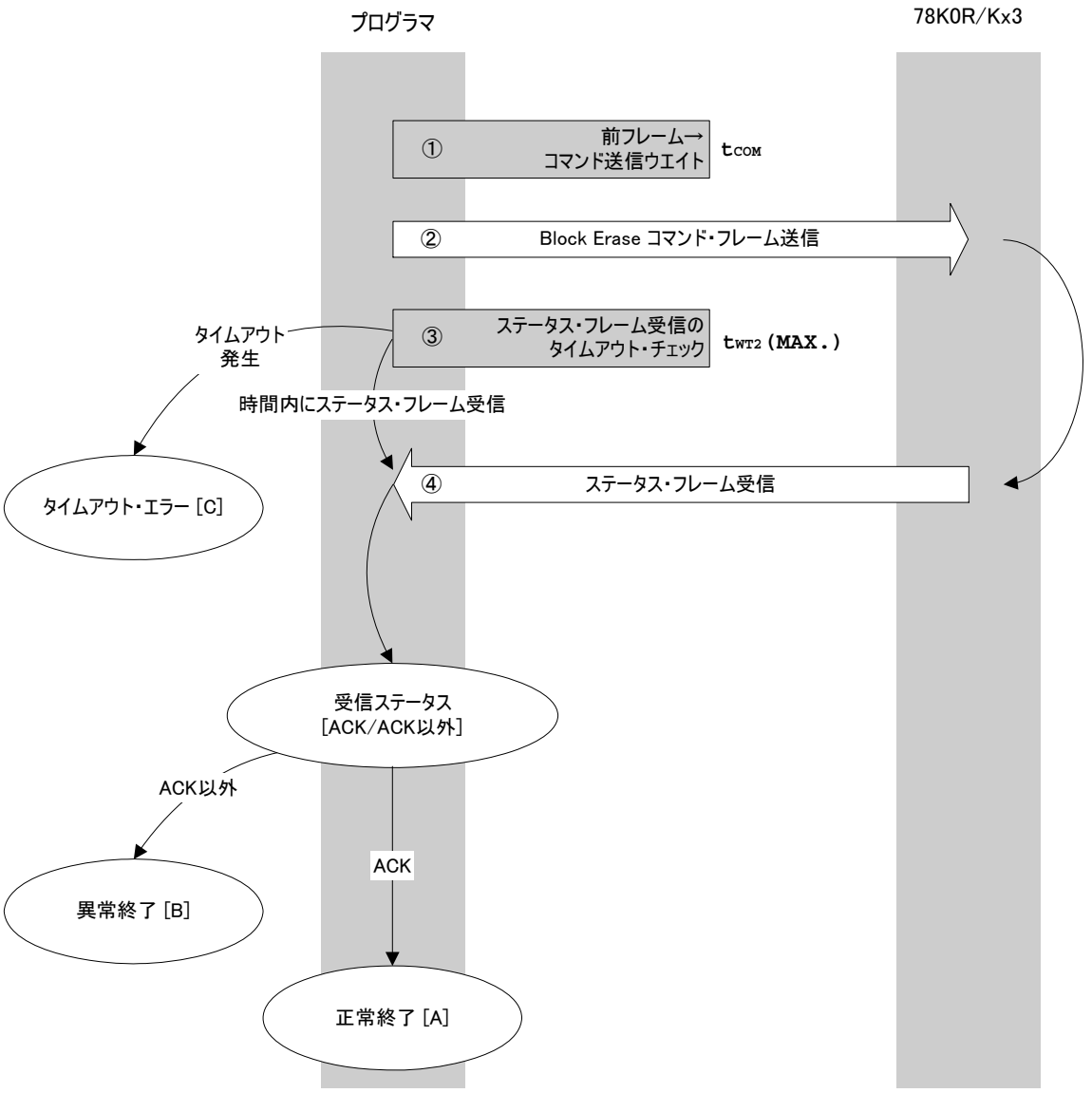
Chip Eraseコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Erase all(chip) command                                       */
/*                                                               */
/*****************************************************************/
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_erase_all(void)
{
   u16     rc;


   fl_wait(tCOM);          // wait before sending command

   put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send ERASE CHIP command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX);// get status frame
// 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 Eraseコマンド

6. 7. 1　処理手順チャート

Block Eraseコマンド処理手順



6.7.2　処理手順説明

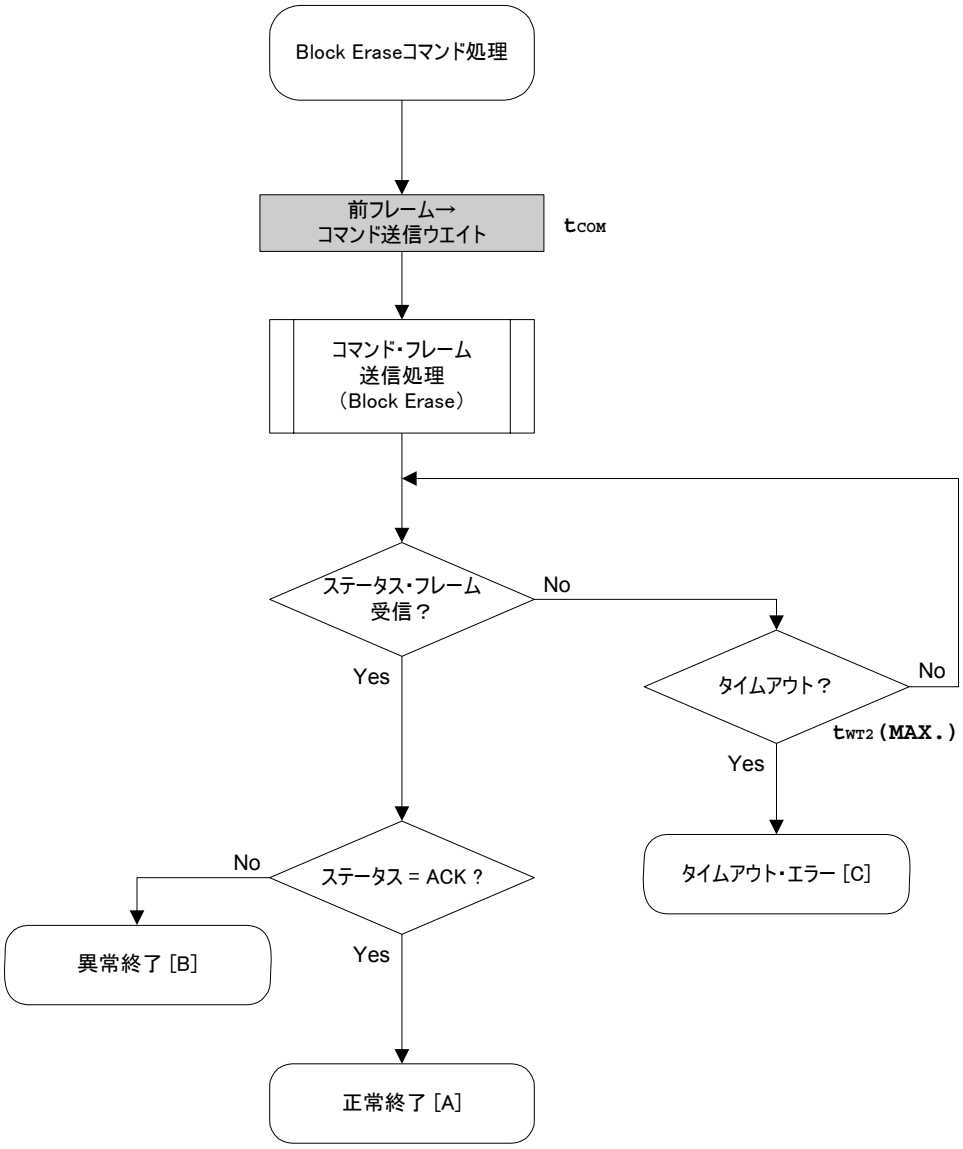
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Block Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT2}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　　：正常終了[A] です。
ST1 = ACK以外の場合：異常終了[B] です。

6.7.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ブロック消去が正常に実行されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	指定した終了アドレスがフラッシュ・メモリの範囲外です。または指定した開始・終了アドレスが，ブロックの先頭・終了アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で「書込み禁止」，「ブロック消去禁止」，「チップ消去禁止」のいずれかの状態になっています。指定範囲にブート・ブロックが含まれており「ブート・ブロック書き換え禁止」が設定されています。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
	MRG10 エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]		－	規定の時間内にステータス・フレームの受信ができませんでした。

6.7.4　フロー・チャート


Block Eraseコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Block Erase）
ステータス・フレーム
受信？
No
Yes
タイムアウト？
No
tWT2 (MAX.)
Yes
タイムアウト・エラー [C]
No
ステータス = ACK ?
異常終了 [B]
Yes
正常終了 [A]

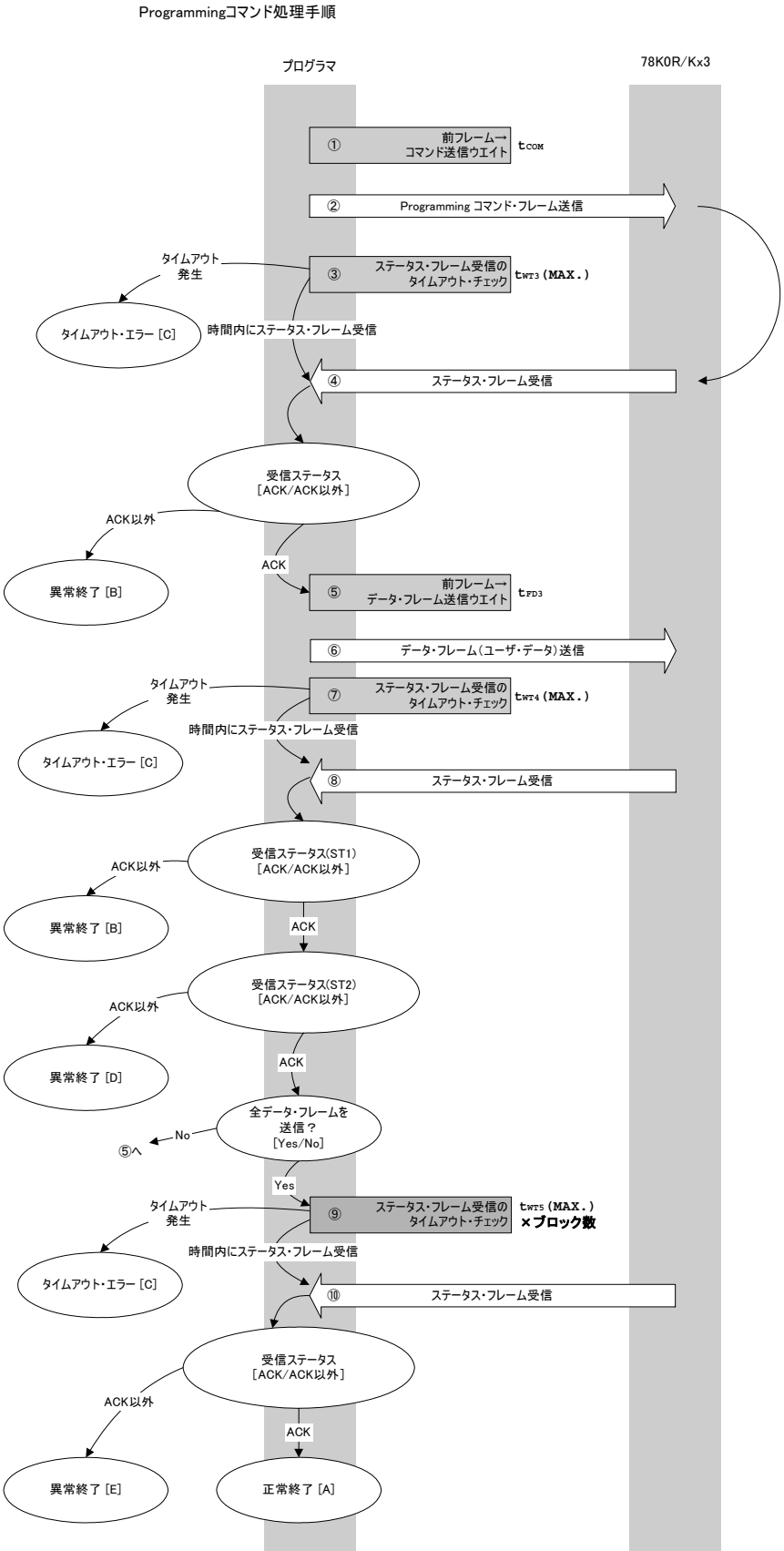
6.7.5　サンプル・プログラム

Block Eraseコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Erase block command                                           */
/*                                                               */
/*****************************************************************/
/* [i] u8 block    ... block number                              */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_erase_blk(u16 sblk, u16 eblk)
{

   u16     rc;
   u32     wt2_max;
   u32     top, bottom;

   top = get_top_addr(sblk);              // get start address of start block
   bottom = get_bottom_addr(eblk);        // get end address of end block

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

   wt2_max = make_wt2_max(sblk, eblk);

   fl_wait(tCOM);                         // wait before sending command

   put_cmd_ua(FL_COM_ERASE_BLOCK, 1+6, fl_cmd_prm); // send ERASE CHIP command

   rc = get_sfrm_ua(fl_ua_sfrm, wt2_max); // get status frame
// 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　Programmingコマンド

6. 8. 1　処理手順チャート



6.8.2 処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Programmingコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT3}（MAX.））。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　：⑤に進みます。
ST1 = ACK以外の場合　：異常終了[B]です。

⑤ 直前のフレームからデータ・フレーム送信までのウエイトをします（ウエイト時間t_{FD3}）。
⑥ データ・フレーム送信処理により，ユーザ・データを送信します。
⑦ ユーザ・データ送信からデータ・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT4}（MAX.））。
⑧ ステータス・コード（ST1/ST2）をチェックします（処理手順チャートやフロー・チャートも参照してください）。

ST1 = ACK以外の場合　：異常終了[B]です。
ST1 = ACKの場合　：ST2の値に応じて以下の処理を行います。
・ST2 = ACKの場合　：全データ・フレームの送信が完了した場合は，⑨に進みます。
まだ送信するデータ・フレームが残っている場合は，⑤より再実行します。
・ST2 = ACK以外の場合　：異常終了[D]です。

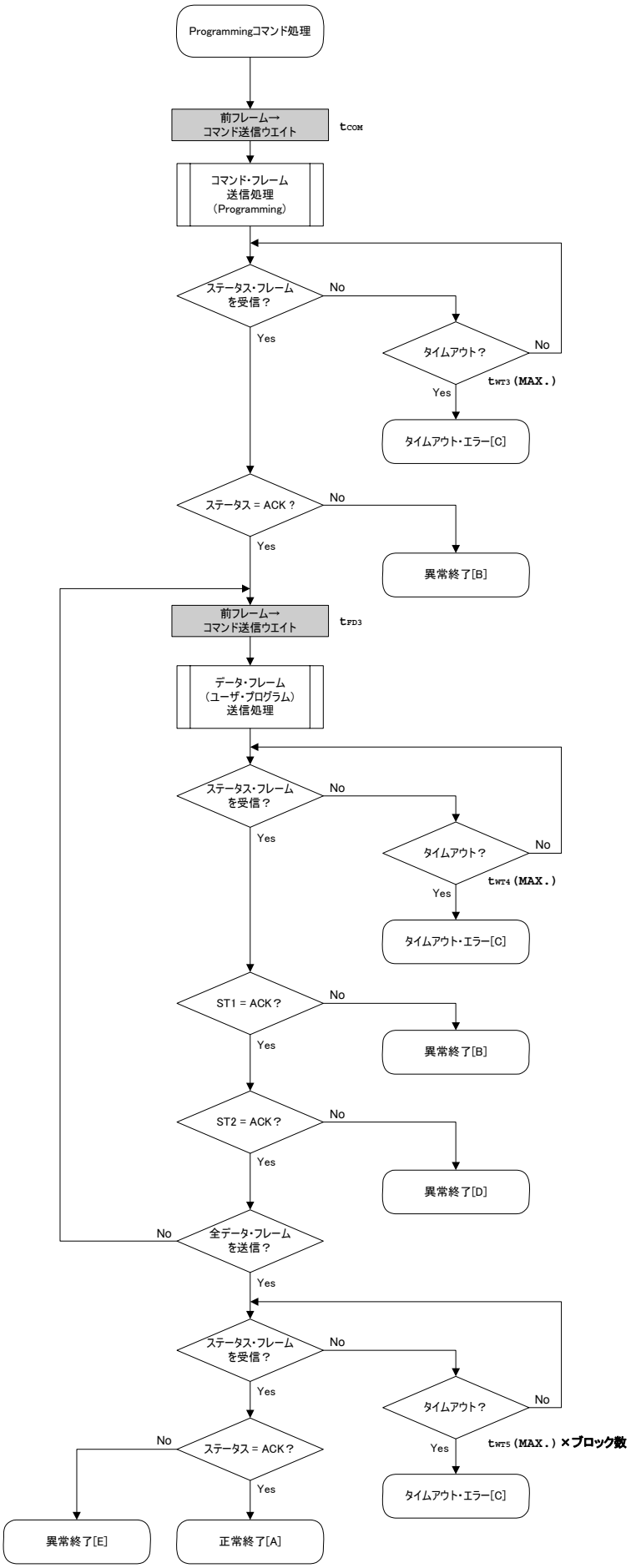
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT5}（MAX.）×ブロック数）。
⑩ ステータス・コードをチェックします。

ST1 = ACKの場合　：正常終了[A]です。
ST1 = ACK以外の場合　：異常終了[E]です。

6.8.3　終了時の内容

終了内容		ステータス・コード	内　　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始／終了アドレスがフラッシュ・メモリの範囲外です。または，指定した開始／終了アドレスがブロックの先頭／終了アドレスではありません。または書込み開始アドレスが終了アドレスよりも大きいです。
	チェックサム・エラー	07H	送信したコマンド・フレーム，またはデータ・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で，「書き込み禁止」になっています。また書込み指定範囲にブート・ブロックが含まれ，「ブート・ブロック書換え禁止」になっています。
	否定応答（NACK）	15H	コマンド・フレーム・データまたはデータ・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D], [E]	MRG10 エラー	1AH	書き込みエラーが発生しました。
	MRG11 エラー	1BH	
	Write エラー	1CH	

6. 8. 4　フロー・チャート


Programmingコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Programming）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT3（MAX.）
Yes
タイムアウト・エラー[C]
ステータス = ACK？
No
Yes
異常終了[B]
前フレーム→
コマンド送信ウエイト
tFD3
データ・フレーム
（ユーザ・プログラム）
送信処理
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT4（MAX.）
Yes
タイムアウト・エラー[C]
ST1 = ACK？
No
Yes
異常終了[B]
ST2 = ACK？
No
Yes
異常終了[D]
No
全データ・フレーム
を送信？
Yes
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT5（MAX.）×ブロック数
Yes
No
ステータス = ACK？
Yes
タイムアウト・エラー[C]
異常終了[E]
正常終了[A]

6.8.5 サンプル・プログラム

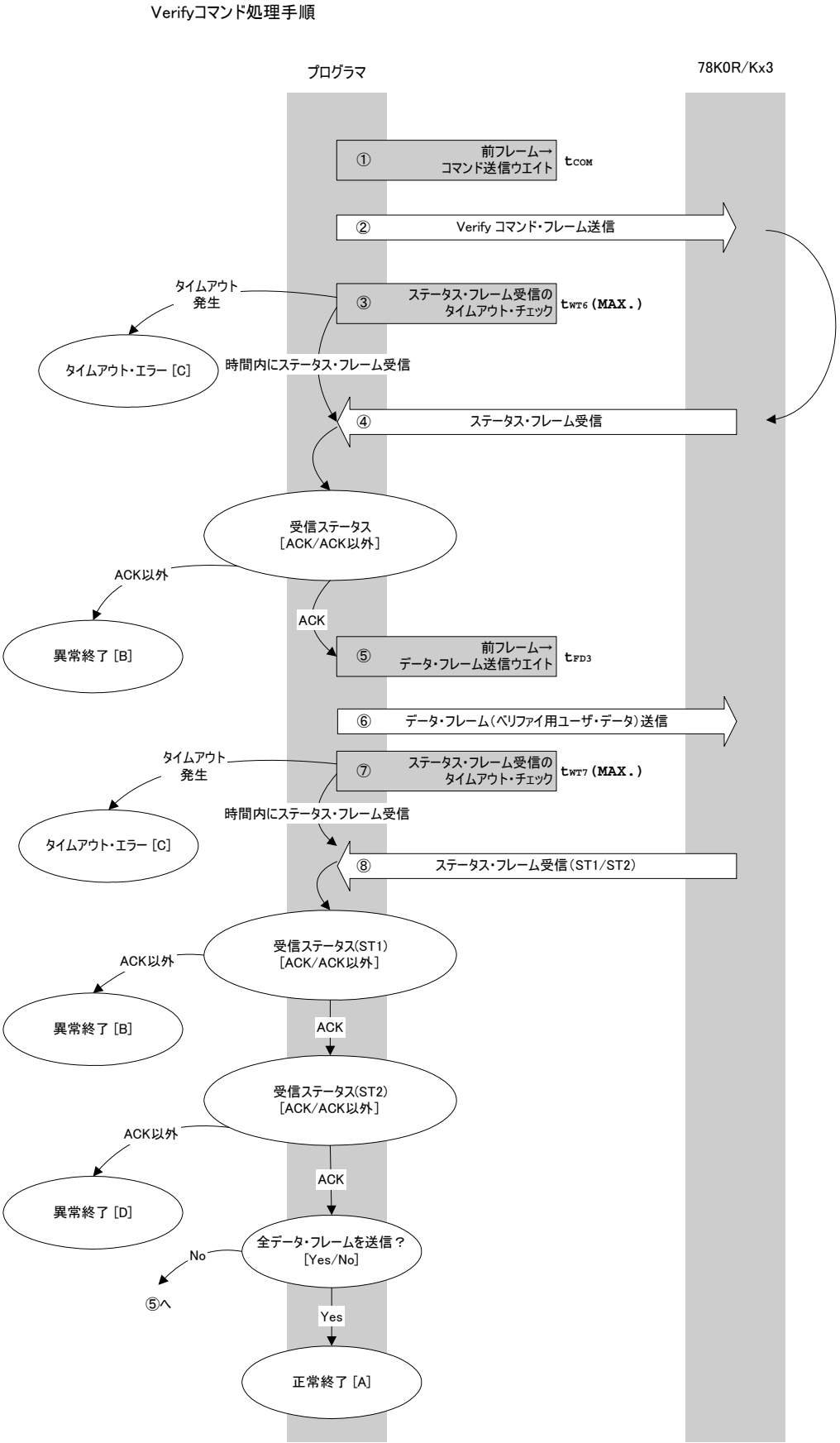
Programmingコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Write command                                                 */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... start address                             */
/* [i] u32 bottom  ... end address                               */
/* [r] u16         ... error code                                */
/*****************************************************************/

#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); // get block num

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


   /*********************************************/
   /*      send command & check status          */
   /*********************************************/
   fl_wait(tCOM);                          // wait before sending command

   put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm);// send "Programming" command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT3_MAX); // get status frame
   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){

           // make send data frame
           if ((bottom - send_head) > 256){        // rest size > 256 ?
                   is_end = false;                 // yes, not is_end frame
                   send_size  = 256;               // transmit size = 256 byte
           }
           else{
```

```
                    is_end = true;
                    send_size = bottom - send_head + 1;     // transmit size = (bottom
- send_head)+1 byte

            }
            memcpy(fl_txdata_frm, rom_buf+send_head, send_size);   // set data frame
payload
            send_head += send_size;

            fl_wait(tFD3);                          // wait before sending data frame

            put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

            rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX);        // get status frame
            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); // No
                    return  rc;                    // case [D]
            }
            if (is_end)
                    break;

    }
    /*********************************************/
    /*      Check internally verify                */
    /*********************************************/
    rc = get_sfrm_ua(fl_ua_sfrm, tWT5_MAX*block_num);      // get status frame again

    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　Verifyコマンド

6. 9. 1　処理手順チャート



6.9.2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Verifyコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT6}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　　：⑤に進みます。
ST1 = ACK以外の場合　　：異常終了[B]です。

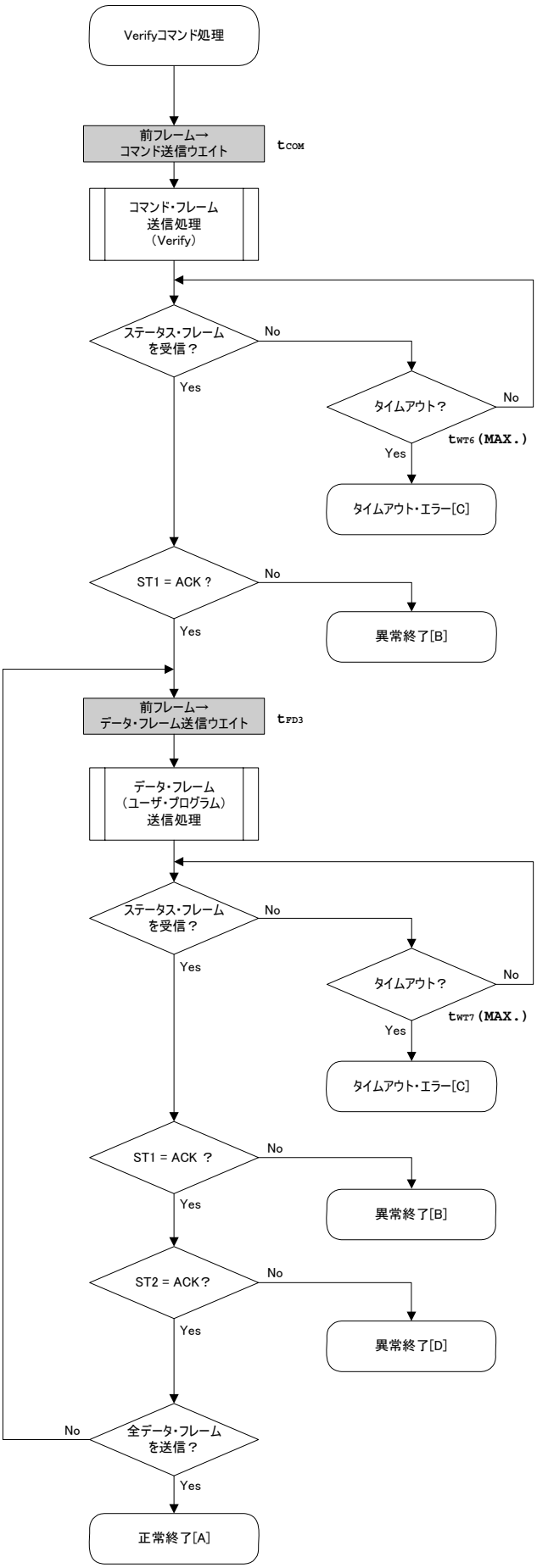
⑤ 直前のフレームからデータ・フレーム送信までのウエイトをします（ウエイト時間t_{FD3}）。
⑥ データ・フレーム送信処理により，ベリファイ用ユーザ・データを送信します。
⑦ ユーザ・データ送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT7}(MAX.)）。
⑧ ステータス・コード（ST1/ST2）をチェックします（処理手順チャートやフロー・チャートも参照してください）。

ST1 = ACK以外の場合　　：異常終了[B]です。
ST1 = ACKの場合　　：受信ステータス（ST2）の値に応じて次の処理を行います。
・ST2 = ACKの場合　　：全データ・フレームを送信済みの場合は正常終了[A]です。
まだ送信すべきデータ・フレームがある場合は⑤から再実行します。
・ST2 = ACK以外の場合　：異常終了[D]です。

6.9.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ユーザ・データのベリファイが正常に実行されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始／終了アドレスがフラッシュ・メモリの範囲外です。または，開始／終了アドレスがブロックの開始／終了アドレスではありません。または書込み開始アドレスが終了アドレスよりも大きいです。
	チェックサム・エラー	07H	送信したコマンド・フレーム，またはデータ・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D]	ベリファイ・エラー	0FH（ST2）	ベリファイ・エラーが発生しました。

6.9.4　フロー・チャート


Verifyコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Verify）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT6 (MAX.)
Yes
タイムアウト・エラー[C]
ST1 = ACK ?
No
異常終了[B]
Yes
前フレーム→
データ・フレーム送信ウエイト
tFD3
データ・フレーム
（ユーザ・プログラム）
送信処理
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT7 (MAX.)
Yes
タイムアウト・エラー[C]
ST1 = ACK ?
No
異常終了[B]
Yes
ST2 = ACK?
No
異常終了[D]
Yes
No
全データ・フレーム
を送信？
Yes
正常終了[A]

6.9.5　サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

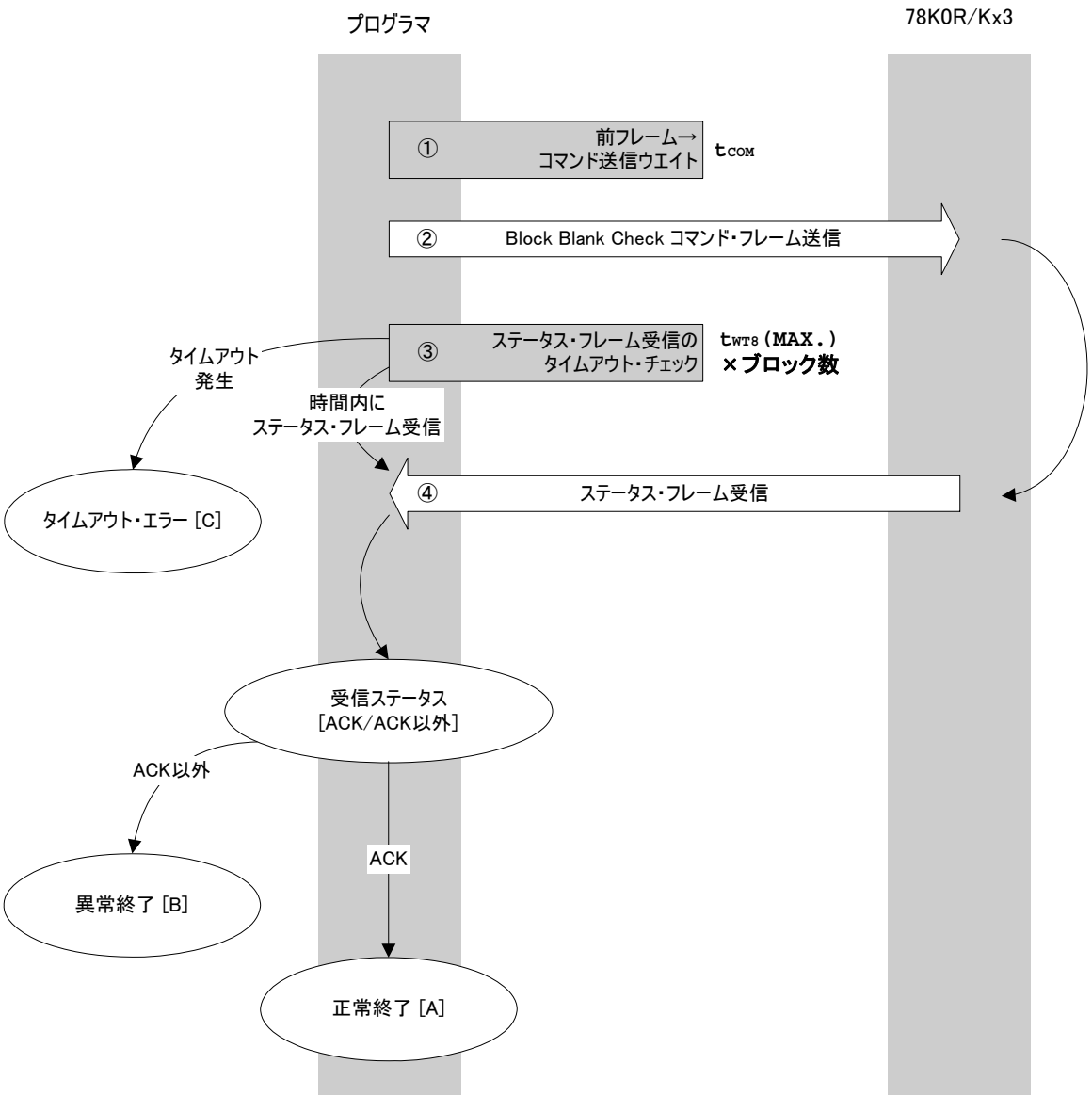
```
/*************************************************************/
/*                                                           */
/* Verify command                                            */
/*                                                           */
/*************************************************************/
/* [i] u32 top     ... start address                         */
/* [i] u32 bottom  ... end address                           */
/* [r] u16         ... error code                            */
/*************************************************************/
u16        fl_ua_verify(u32 top, u32 bottom, u8 *buf)
{
   u16     rc;
   u32     send_head, send_size;
   bool    is_end;

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


   /*********************************************/
   /*      send command & check status          */
   /*********************************************/

   fl_wait(tCOM);                                  // wait before sending command

   put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm);       // send VERIFY command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT6_MAX);         // get status frame
   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){

           // make send data frame
           if ((bottom - send_head) > 256){        // rest size > 256 ?
                   is_end = false;                 // yes, not is_end frame
                   send_size  = 256;               // transmit size = 256 byte
           }
           else{
                   is_end = true;
                   send_size = bottom - send_head + 1;// transmit size = (bottom -
send_head)+1 byte

           }
           memcpy(fl_txdata_frm, buf+send_head, send_size);// set data frame payload
```

```
            send_head += send_size;

            fl_wait(tFD3);
            put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

            rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX);        // get status frame
            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);  // No
                    return  rc;                     // case [D]
            }
            if (is_end)                     // send all user data ?
                    break;                  // yes
            //continue;
    }
    return  FLC_NO_ERR;     // case [A]
}
```

6. 10　Block Blank Checkコマンド

6. 10. 1　処理手順チャート

Block Blank Checkコマンド処理手順



6. 10. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Block Blank Checkコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は タイムアウト・エラー[C] となります
（タイムアウト時間t_{WTS}(MAX.)×ブロック数）。
④ ステータス・コードをチェックします。

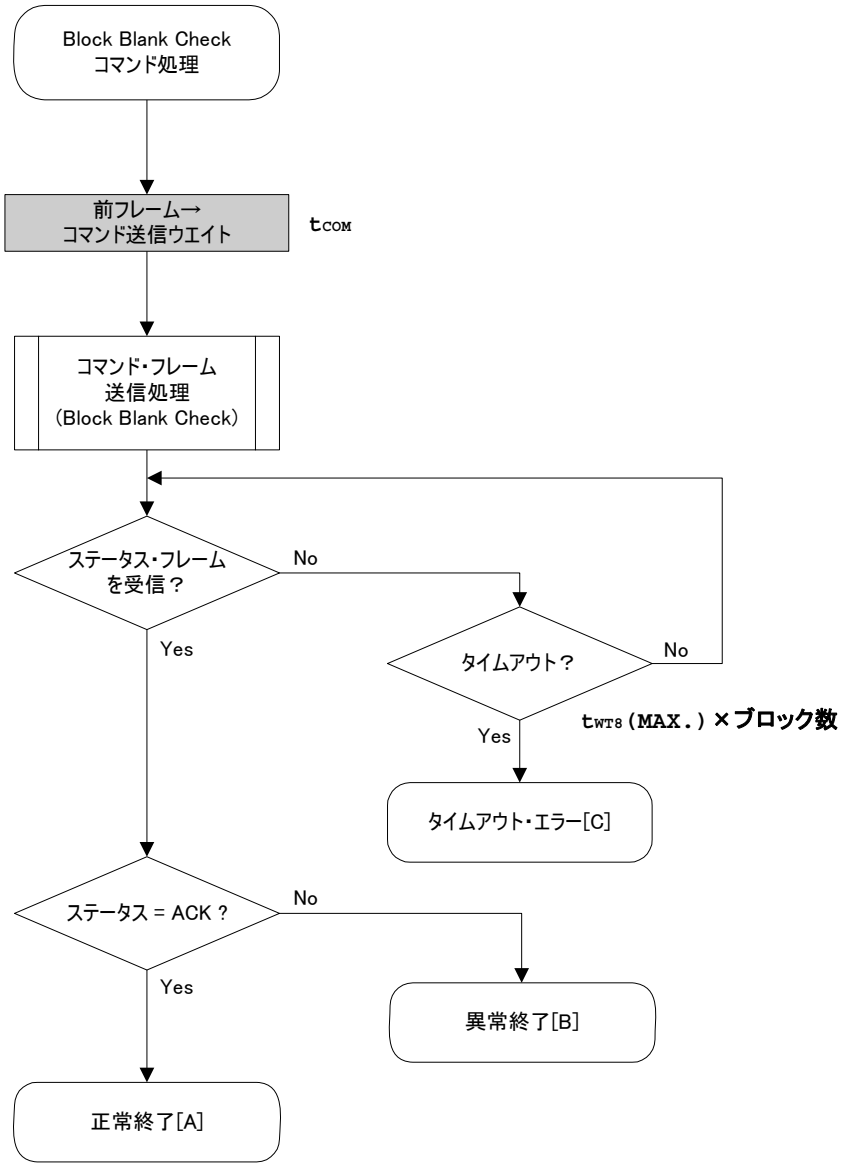
ST1 = ACKの場合　：正常終了[A]です。

ST1 = ACK以外の場合　：異常終了[B]です。

6. 10. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ブロック・ブランク・チェックが正常に実行されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	終了アドレスがフラッシュ・メモリの範囲外です。または，開始／終了アドレスがブロックの先頭・終了アドレスではありません。またはパラメータ D01 が，00H, 01H 以外の値です。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
	MRG11 エラー	1BH	指定したブロックのフラッシュ・メモリがブランクではありません。
タイムアウト・エラー [C]		−	規定の時間内にステータス・フレームの受信ができませんでした。

6. 10. 4　フロー・チャート



6. 10. 5　サンプル・プログラム

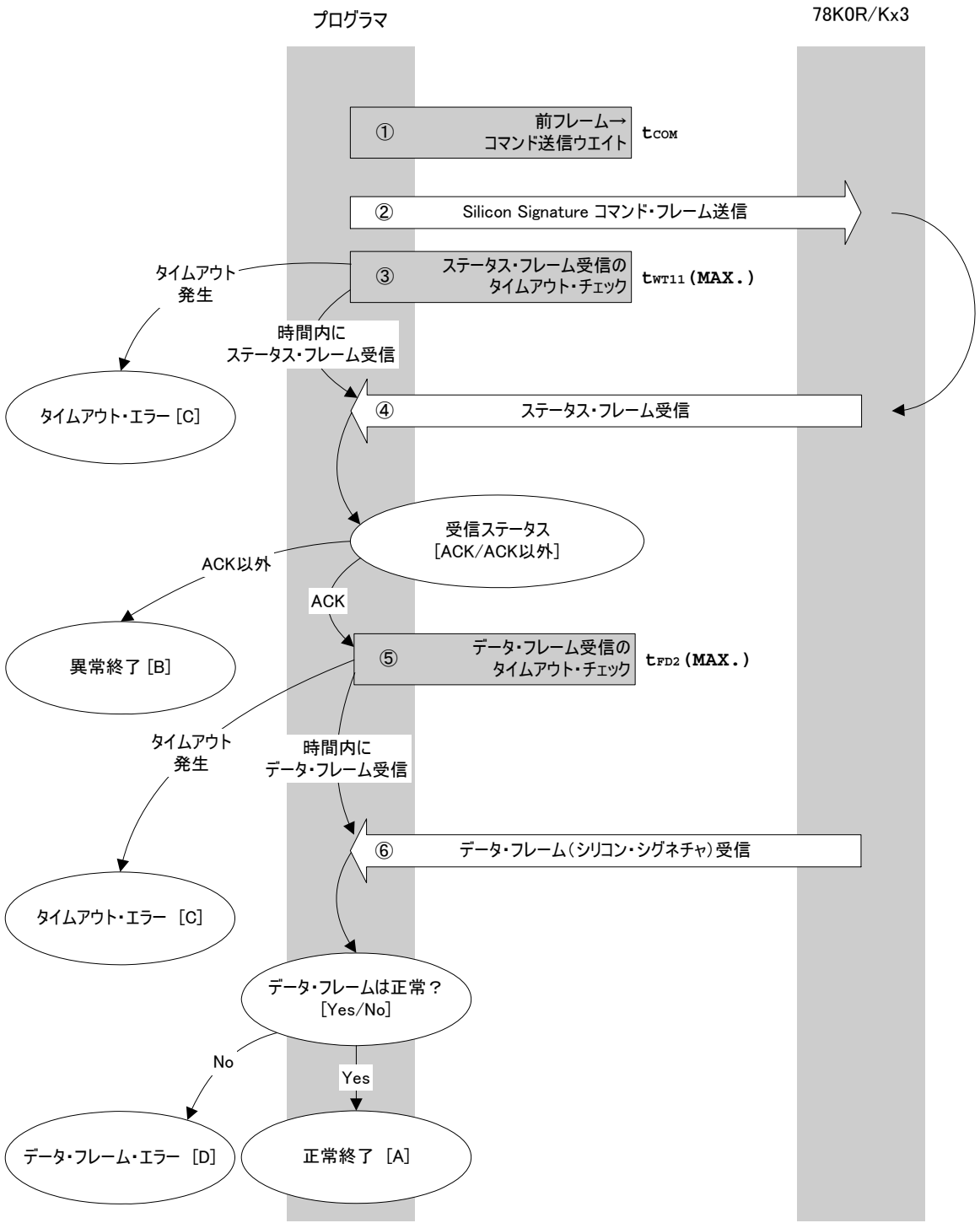
Block Blank Checkコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Block blank check command                                     */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... top address of blank check                */
/* [i] u32 bottom  ... bottom address of blank check             */
/* [i]  u8 whole   ... <1>check w/NON user flash                 */
/*                     <0>chek only user flash                   */
/* [r] u16         ... error code                                */
/*****************************************************************/
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); // set SAH/SAM/SAL, EAH/EAM/EAL
   block_num = get_block_num(top, bottom); // get block num
   fl_cmd_prm[6] = whole;                  // check only user area or not

   fl_wait(tCOM);                          // wait before sending command

   put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7+1, fl_cmd_prm);


   rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num);    // get status frame
// 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　Silicon Signatureコマンド

6. 11. 1　処理手順チャート

Silicon Signatureコマンド処理手順



6. 11. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Silicon Signatureコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{WT11}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　　：⑤に進みます。
ST1 = ACK以外の場合　　：異常終了[B]です。

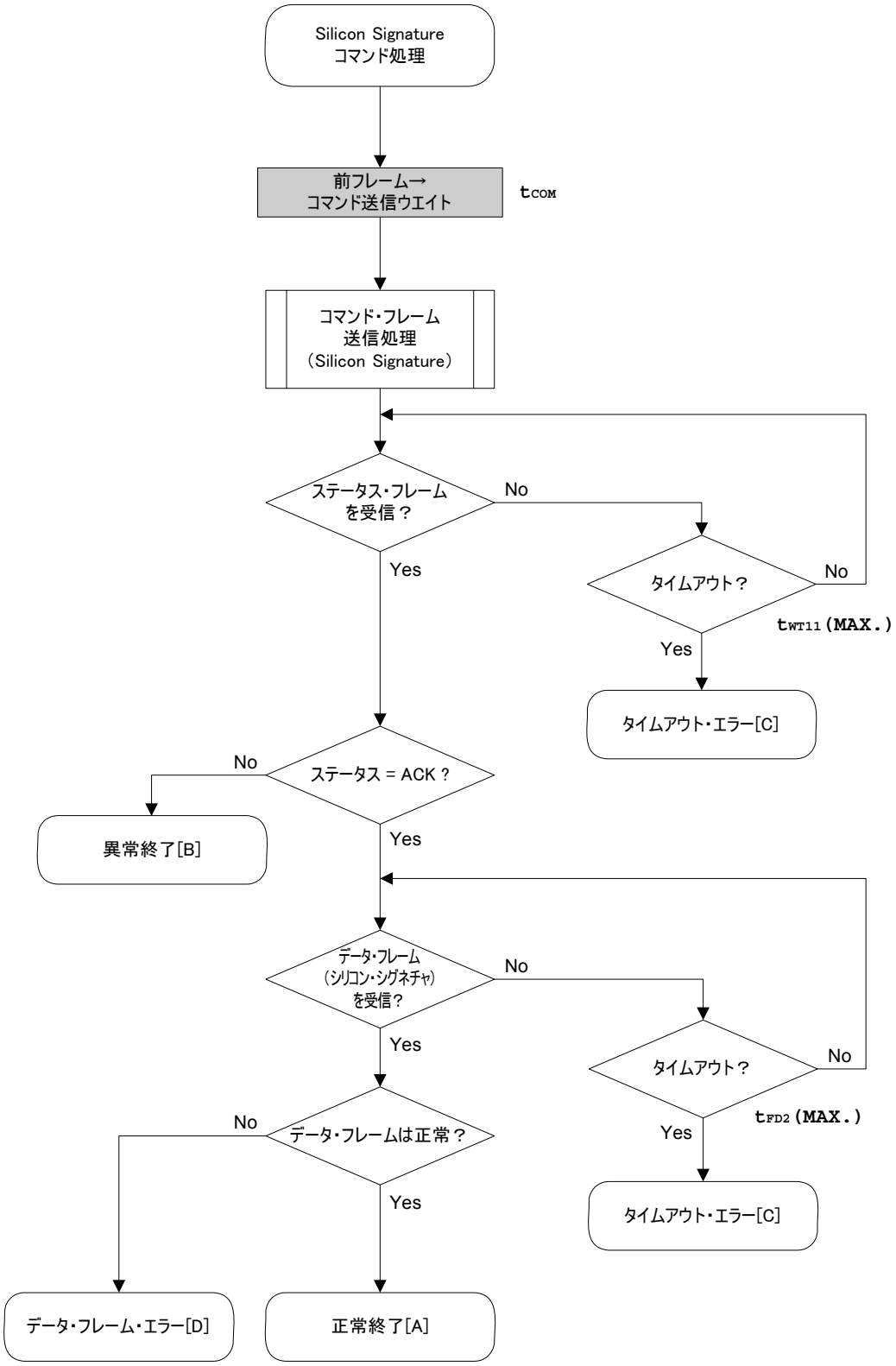
⑤ データ・フレーム（シリコン・シグネチャ・データ）受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}(MAX.)）。
⑥ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

データ・フレームが正常の場合　　：正常終了[A]です。
データ・フレームが異常の場合　　：データ・フレーム・エラー[D]です。

6. 11. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，シリコン・シグネチャ・データの取得が正常に実行されたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	ステータス・フレーム，またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー[D]		−	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

6. 11. 4　フロー・チャート



6. 11. 5　サンプル・プログラム

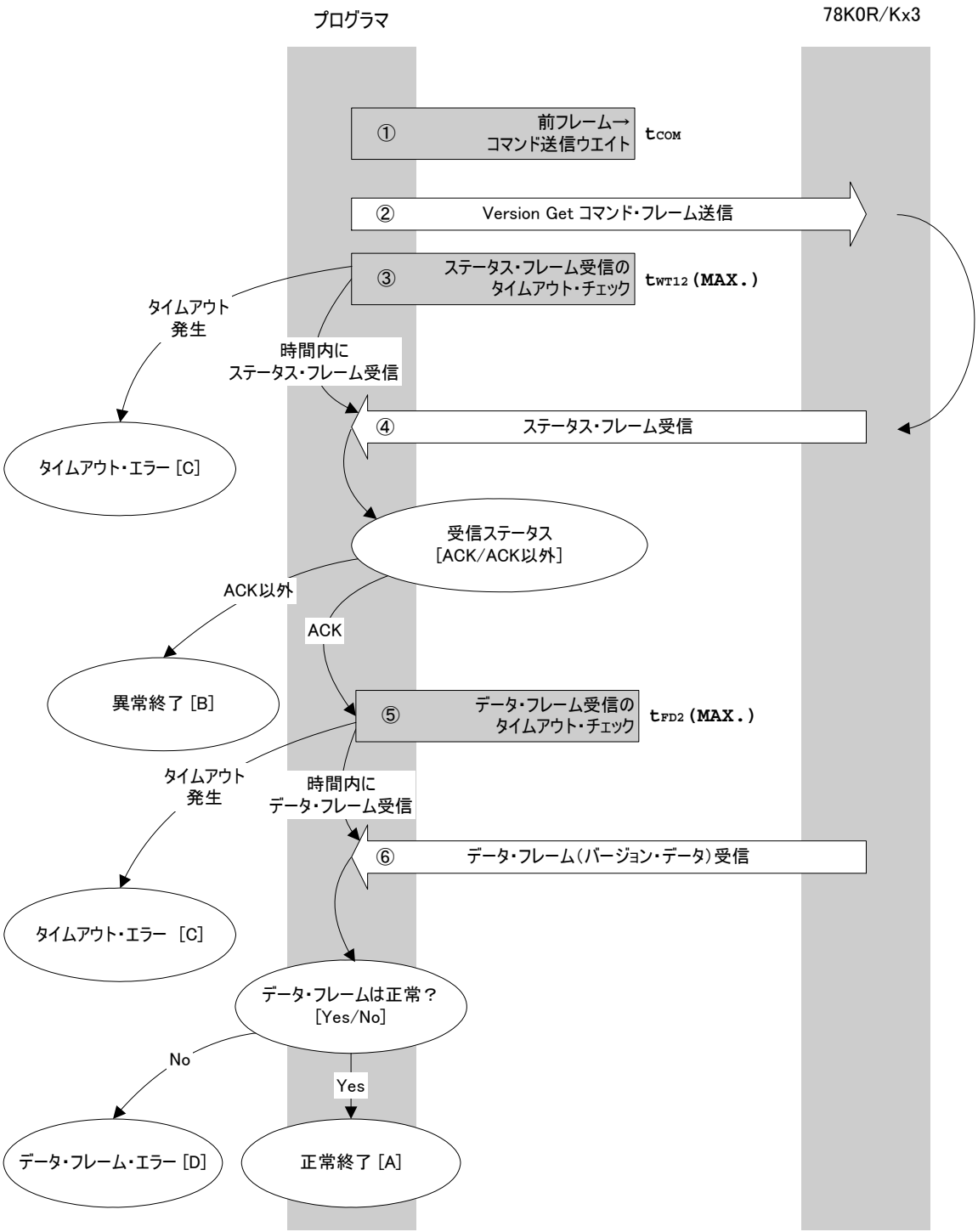
Silicon Signatureコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Get silicon signature command                                 */
/*                                                               */
/*****************************************************************/
/* [i] u8 *sig     ... pointer to signature save area            */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_getsig(u8 *sig)
{
   u16     rc;


   fl_wait(tCOM);                                    // wait before sending command

   put_cmd_ua(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm); // send GET SIGNATURE command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT11_MAX);       // get status frame
   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);     // get status frame
   if (rc){                                          // if error
           return rc;                                // case [D]
   }
   memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);// copy Signature
data
   return  rc;                                       // case [A]
 }
```

6. 12　Version Getコマンド

6. 12. 1　処理手順チャート

Version Getコマンド処理手順



6. 12. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Version Getコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{WT12}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　　：⑤に進みます。
ST1 = ACK以外の場合　　：異常終了[B]です。

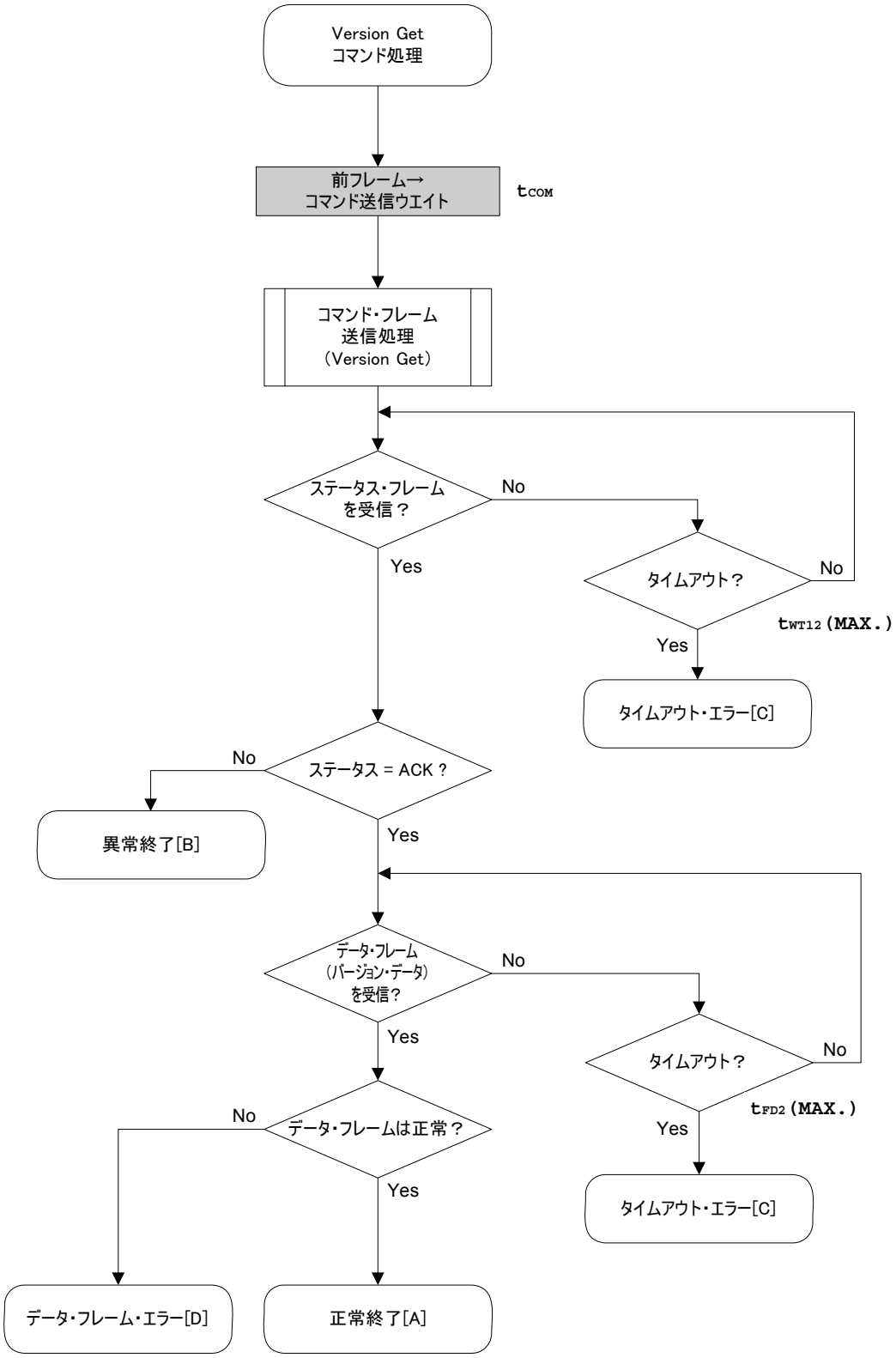
⑤ データ・フレーム（バージョン・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}(MAX.)）。
⑥ 受信したデータ・フレーム（バージョン・データ）をチェックします。

データ・フレームが正常の場合　　：正常終了[A]です。
データ・フレームが異常の場合　　：データ・フレーム・エラー[D]です。

6. 12. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，バージョン・データを取得できたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	ステータス・フレーム，またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]		−	バージョン・データとして受信したデータ・フレームのチェックサムが異常です。

6. 12. 4　フロー・チャート



6. 12. 5　サンプル・プログラム

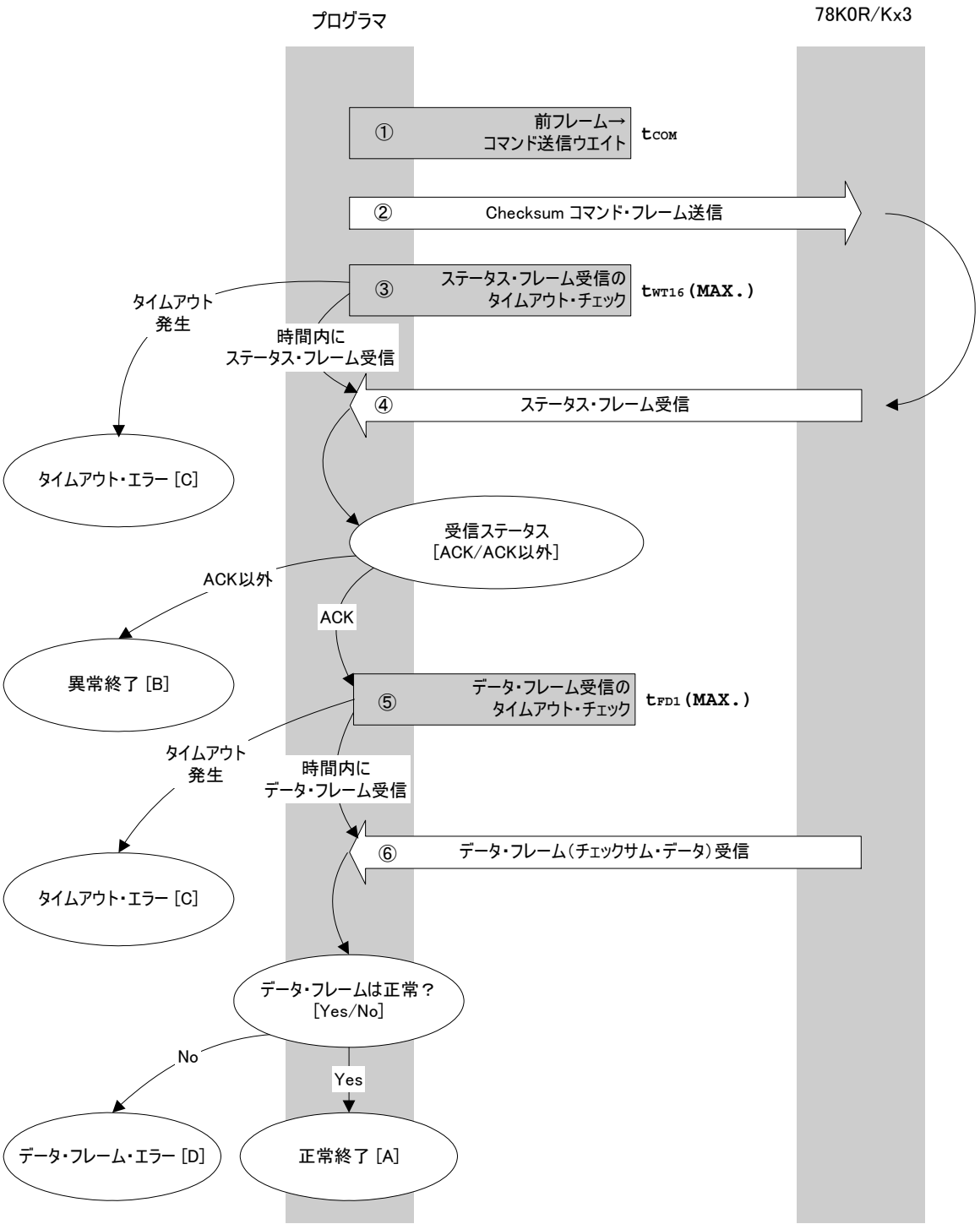
Version Getコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Get device/firmware version command                           */
/*                                                               */
/*****************************************************************/
/* [i] u8 *buf     ... pointer to version date save area         */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_getver(u8 *buf)
{
   u16     rc;


   fl_wait(tCOM);                                    // wait before sending command


   put_cmd_ua(FL_COM_GET_VERSION, 1, fl_cmd_prm); // send GET VERSION command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT12_MAX);       // get status frame
   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);     // get data frame
   if (rc){
           return rc;                             // case [D]
   }

   memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// copy version data
   return rc;                                     // case [A]
}
```

6. 13　Checksumコマンド

6. 13. 1　処理手順チャート

Checksumコマンド処理手順



6. 13. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Checksumコマンドを送信します。
③ コマンドの送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{WT16}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　：⑤に進みます。
ST1 = ACK以外の場合　：異常終了[B]です。

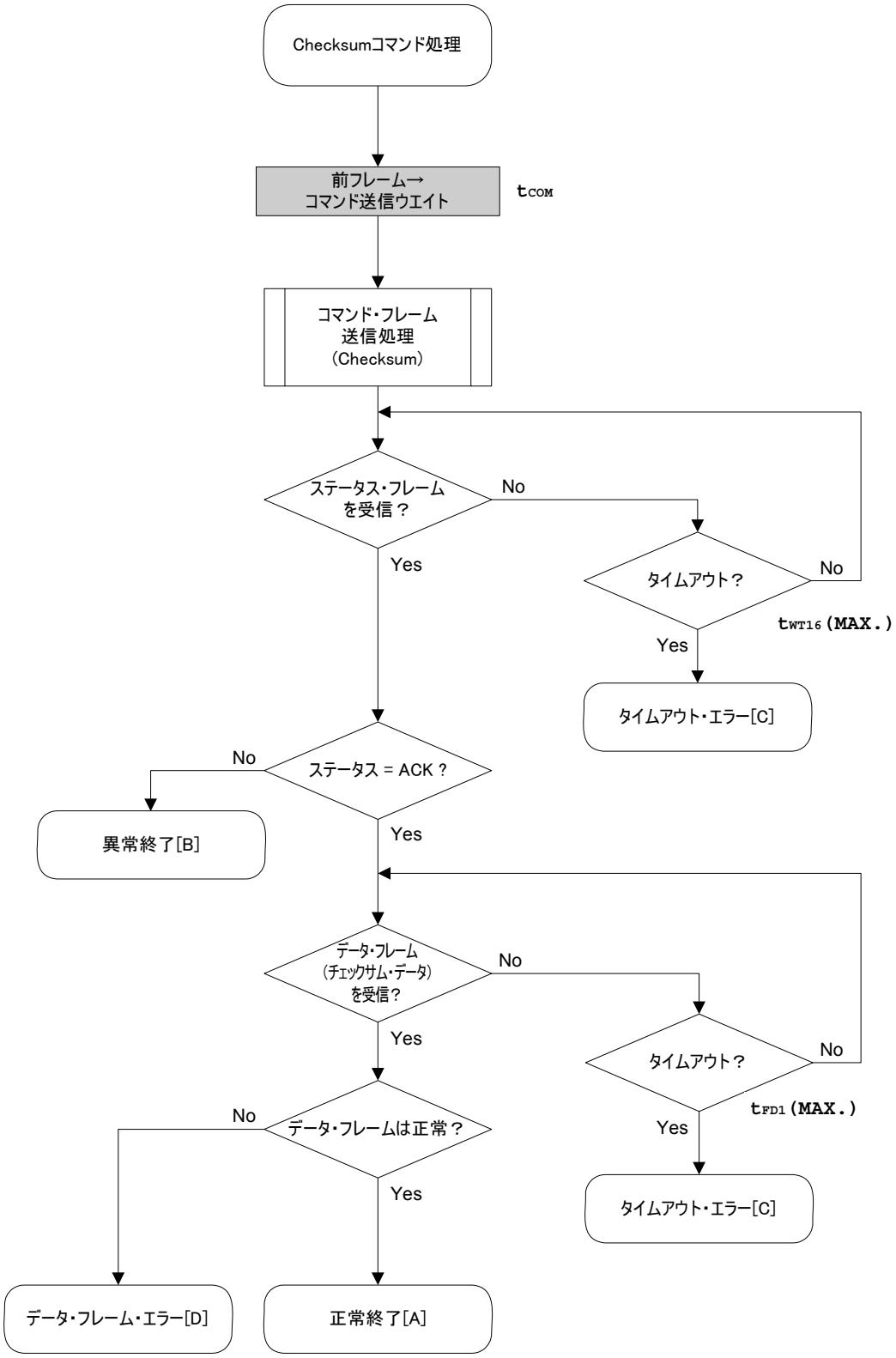
⑤ データ・フレーム（チェックサム・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD1}(MAX.)）。
⑥ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

データ・フレームが正常の場合　：正常終了[A]です。
データ・フレームが異常の場合　：データ・フレーム・エラー[D]です。

6. 13. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，チェックサム・データを取得できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始／終了アドレスがフラッシュ・メモリの範囲外です。または，開始／終了アドレスがブロックの開始／終了アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	ステータス・フレーム，またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]		−	チェックサム・データとして受信したデータ・フレームのチェックサムが異常です。

6. 13. 4　フロー・チャート



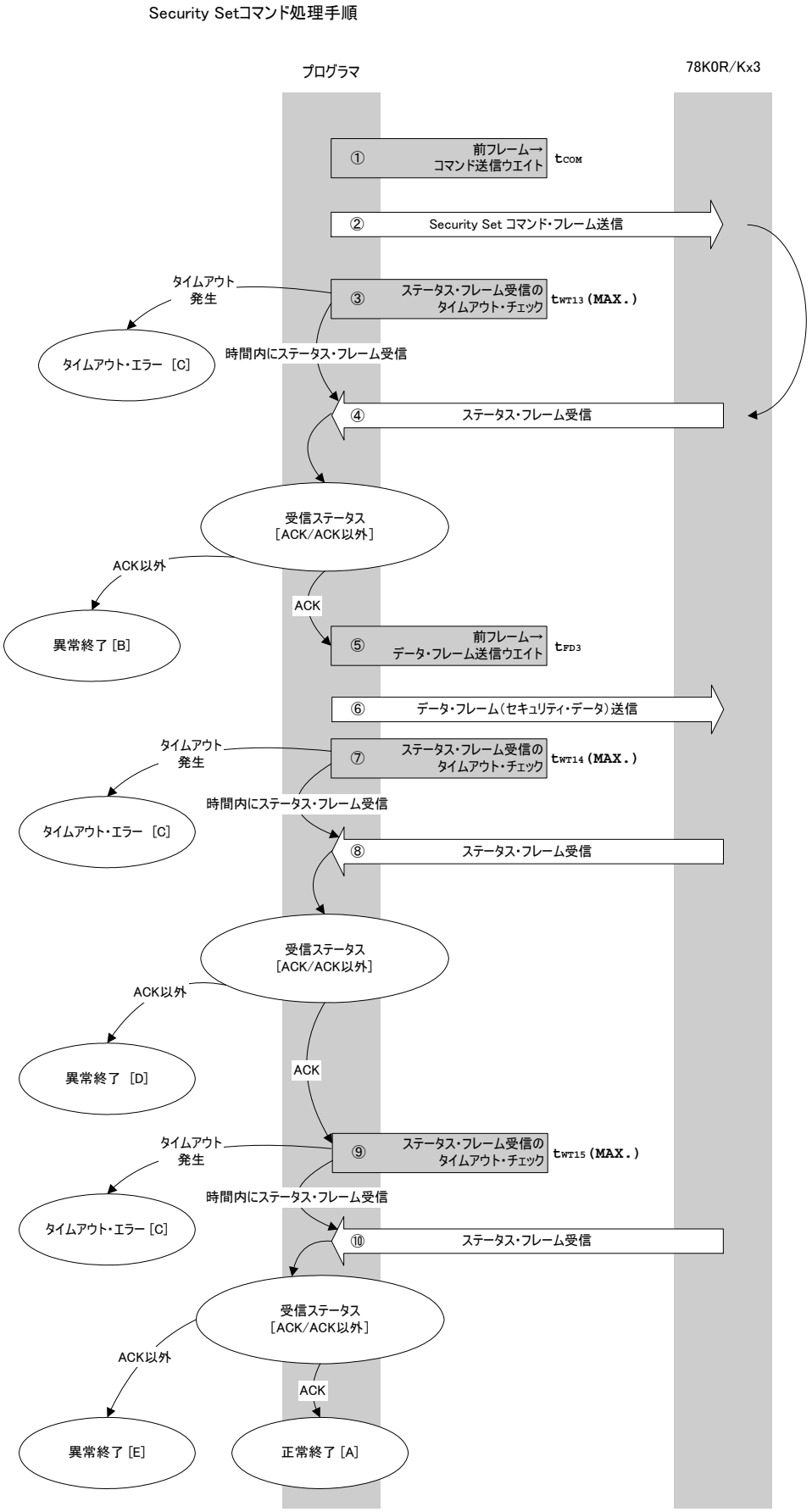
6. 13. 5　サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* Get checksum command                                      */
/*                                                           */
/*************************************************************/
/* [i] u16 *sum    ... pointer to checksum save area         */
/* [i] u32 top     ... start address                         */
/* [i] u32 bottom  ... end address                           */
/* [r] u16         ... error code                            */
/*************************************************************/
u16        fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
   u16     rc;

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


   /*********************************************/
   /*      send command                         */
   /*********************************************/

   fl_wait(tCOM);                                  // wait before sending command

   put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send GET VERSION command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT16_MAX);        // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                     break;  // continue
   //      case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [B]
   }

   /*********************************************/
   /*      get data frame (Checksum data)       */
   /*********************************************/
   rc = get_dfrm_ua(fl_rxdata_frm, tFD1_MAX);      // get status frame
   if (rc){                                        // if no error,
           return rc;                              // case [D]
   }

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

 }
```

6. 14　Security Setコマンド

6. 14. 1　処理手順チャート



6. 14. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Security Setコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT13}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　：⑤に進みます。
ST1 = ACK以外の場合　：異常終了[B]です。

⑤ 直前のフレームからデータ・フレーム送信までのウエイトをします（ウエイト時間t_{FD3}）。
⑥ データ・フレーム送信処理によりデータ・フレーム（セキュリティ設定データ）を送信します。
⑦ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT14}(MAX.)）。
⑧ ステータス・コードをチェックします。

ST1 = ACKの場合　：⑨に進みます。
ST1 = ACK以外の場合　：異常終了[D]です。

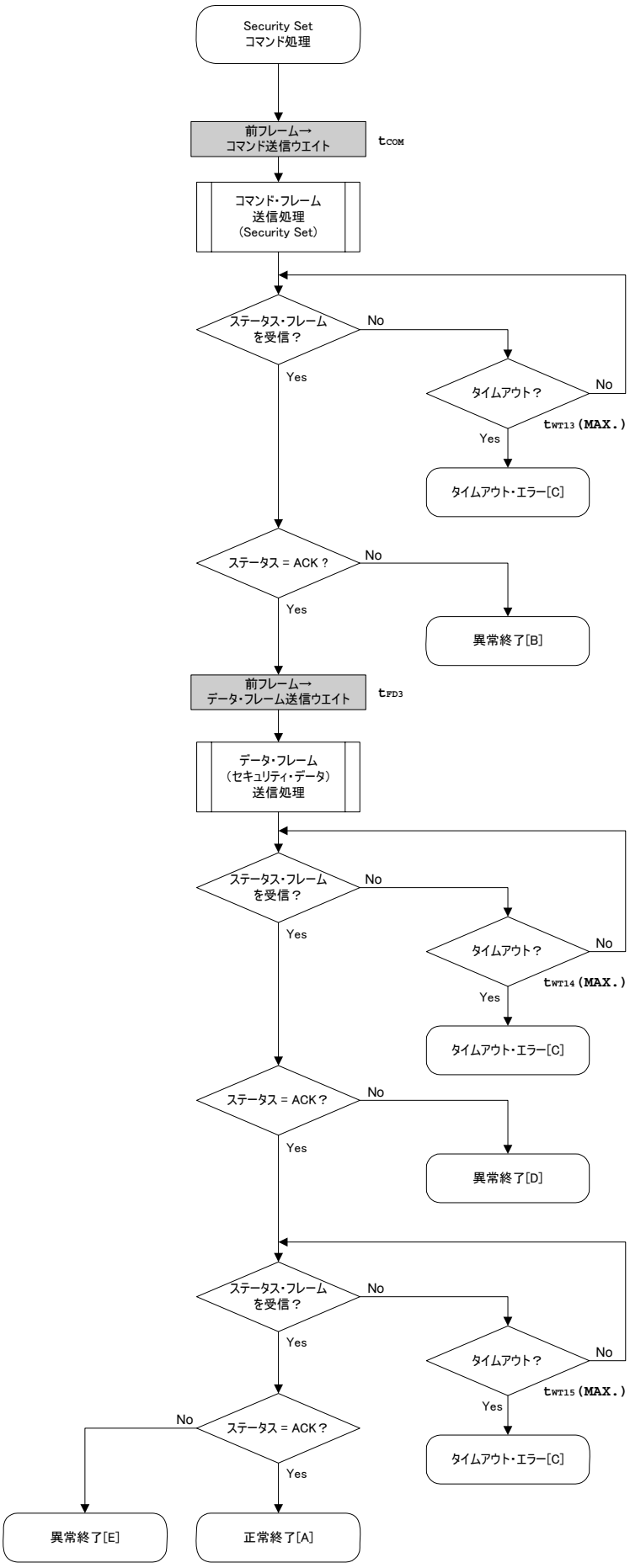
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT15}(MAX.)）。
⑩ ステータス・コードをチェックします。

ST1 = ACKの場合　：正常終了[A]です。
ST1 = ACK以外の場合　：異常終了[E]です。

6. 14. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，セキュリティ設定データが正しく設定されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	パラメータ BOT が 01H ではありません。または FSW 設定ブロック番号が，スタート・ブロック番号＞エンド・ブロック番号になっています。または FSW のエンド・ブロック番号が最終ブロック番号よりも大きいです。
	チェックサム・エラー	07H	送信したコマンド・フレーム，またはデータ・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	すでに禁止されているフラグを許可にしようとしています。
	否定応答（NACK）	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正，ETX なしなど）。
タイムアウト・エラー [C]		−	ステータス・フレーム，またはデータ・フレームの受信でタイムアウトが発生しました。
異常終了 [D], [E]	MRG10 エラー	1AH	セキュリティ・データの書き込みに失敗しました。
	MRG11 エラー	1BH	
	Write エラー	1CH	

6. 14. 4　フロー・チャート



6. 14. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Set security flag command                                     */
/*                                                               */
/*****************************************************************/
/* [i] u8 scf      ... Security flag data                        */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_setscf(u8 scf, u8 bot, u8 fsws, u8 fswe)
{
   u16     rc;

   /***********************************************/
   /*      set params                              */
   /***********************************************/
   fl_cmd_prm[0] = 0x00;                  // "BLK" (must be 0x00)
   fl_cmd_prm[1] = 0x00;                  // "PAG" (must be 0x00)

   fl_txdata_frm[0] = scf|= 0b11101000;   // "FLG" (bit 7,6,5,3 must be '1')
   fl_txdata_frm[1] = bot;                // "BOT"
   fl_txdata_frm[2] = 0x00;               // "FSWS High"
   fl_txdata_frm[3] = fsws;               // "FSWS Low"
   fl_txdata_frm[4] = 0x00;               // "FSWE High"
   fl_txdata_frm[5] = fswe;               // "FSWE Low"

   /***********************************************/
   /*      send command                            */
   /***********************************************/
   fl_wait(tCOM);                                  // wait before sending command

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


   /***********************************************/
   /*      send data frame (security setting data) */
   /***********************************************/

   fl_wait(tFD4);
   put_dfrm_ua(6, fl_txdata_frm, true);           // send securithi setting data

  // rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX);       // get status frame
   rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX+100);   // get status frame (+100us is
overhead)
   switch(rc) {
          case    FLC_NO_ERR:                     break;  // continue
   //     case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
          default:                return  rc;     break;  // case [B]
   }
```

```
   /*********************************************/
   /*      Check internally verify               */
   /*********************************************/
   rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);       // get status frame
// 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7章　フラッシュ・メモリ・プログラミング・パラメータ特性

この章では，フラッシュ・メモリ・プログラミング・モード時のプログラマと78K0R/Kx3の間のパラメータ特性を記載しています。その他の電気的特性は，78K0R/Kx3各製品のユーザーズ・マニュアルを参照のうえ，設計してください。

(1) フラッシュ・メモリ・パラメータ特性

(a) フラッシュ・メモリ・プログラミング・モード・セット時間

項　　目	略　号	MIN.	TYP.	MAX.
V_{DD}↑ to FLMD0↑	t_{DP}	0		
FLMD0↑ to $\overline{\text{RESET}}$↑	t_{PR}	2 ms		
Ready start time from $\overline{\text{RESET}}$↑	t_{R0}	3 ms		100 ms
Low level data0 (Ready) width[注]	t_{L0}	892 μs	937.5 μs	987 μs
Wait for low level data1	t_{01}	120 μs		
Wait for low level data2	t_{02}	10 μs		
Wait for Read command	t_{2C}	300 μs		
Low level data1 / data2 width[注]	t_{L1}, t_{L2}		937.5 μs	

注　ロウ・レベル幅は，9600 bps時の00Hデータ幅と同じです（スタート・ビットを含むため，9ビットの“0”データとなります）。

t_{L0}は，78K0R/Kx3のファームウエアより送信されたデータのロウ・レベル幅です。t_{L1}とt_{L2}は，フラッシュ・プログラマより送信されたデータのロウ・レベル幅です。

(b) プログラミング特性

	ウエイト	条件	略号	MIN.	MAX.
	データ・フレーム～データ・フレーム	データ・フレーム受信	t_{DR}	8.0 μs	
★		データ・フレーム送信	t_{DT}	注	
★	ステータス・フレーム送信～データ・フレーム送信	－	t_{FD1}	注	
	ステータス・フレーム送信～データ・フレーム受信 (1)	プログラム・コマンド	t_{FD2}	8.7 μs	
	ステータス・フレーム送信～データ・フレーム受信 (2)	ベリファイ・コマンド	t_{FD3}	145 μs	
	ステータス・フレーム送信～データ・フレーム受信 (3)	セキュリティ設定コマンド	t_{FD4}	120 μs	
	ステータス・フレーム送信～コマンド・フレーム受信	－	t_{COM}	595 μs	

★ 注　プログラマは連続受信許可にしておいてください。また，プログラマのタイムアウト時間は，3 s以上にしてください。

備考　ウエイトには，次のような定義があります。

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

78K0R/Kx3は，直前の通信完了後，MIN.後から次の通信が可能となります。

＜t_{DT}, t_{FD1}＞

78K0R/Kx3は，直前の通信完了後，MIN.後から次の通信が可能となります。

(c) コマンド特性

	コマンド	略号	条件	MIN.	MAX.
★	Reset	t_{WT0}	－	注1	
★	Chip Erase	t_{WT1}	製品グループA 注2	(60.6＋5.7×全ブロック数) ms	(1112＋140.9×全ブロック数) ms
★			製品グループB 注3	(812.9＋5.7×(全ブロック数－128)) ms	(19403.5＋140.9×(全ブロック数－128)) ms
★	Block Erase	t_{WT2} 注4	－	17.5 ms	(1.1＋275.5×同時選択消去の実行回数＋137.9×消去するブロック数) ms
★	Programming	t_{WT3}	－	注1	
		t_{WT4} 注5	－	2.8 ms	47.2 ms
★		t_{WT5} 注6	ブロック0	13.3 ms	860.0 ms
★			ブロック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. **プログラマは，コマンド・フレーム送信前に受信許可にしておいてください。また，プログラマのタイムアウト時間は，**3 s**以上にしてください。**

2. **製品グループ**A**：フラッシュ・サイズ≦**256 KB**（ブロック数≦**128**）**
3. **製品グループ**B**：フラッシュ・サイズ＞**256 KB**（ブロック数＞**128**）**
4. **同時選択消去の実行回数の求め方に関しては，（**2**）**Block Erase**コマンドにおける同時選択消去について**を参照してください。**
5. 256**バイトの場合の時間**
6. 1**ブロックの場合の時間**

★ 7. **プログラマは，データ・フレーム送信前に受信許可にしておいてください。また，プログラマのタイムアウト時間は，**3 s**以上にしてください。**

★ 8. **プログラマは，連続受信許可にしておいてください。また，プログラマのタイムアウト時間は，**3 s**以上にしてください。**

（次ページに備考を示します。）

★ **備考　ウエイトには，次のような定義があります。**

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

78K0R/Kx3**は，**MIN.～MAX.**時間内に各コマンド処理を終了し，ステータス・フレームを送出します。**

MAX.**時間の規定があるコマンドについては，プログラマは，受信フレームのスタート・ビットを**MAX.**時間までは待ち，**MAX.**時間経過後にタイムアウト処理を行ってください。**

MAX.**時間の規定がないコマンドについては，各注を参照してください。**

＜t_{WT10}＞

78K0R/Kx3**は，直前の通信完了後，**MIN.**後から次の通信が可能となります。**

プログラマは，直前の通信完了後，MIN.**時間経過後に次データの送信を行ってください。**

（2）Block Eraseコマンドにおける同時選択消去について

78K0R/Kx3のBlock Eraseコマンド実行は，複数ブロックを同時に消去する“同時選択消去”を繰り返し実行することにより実現しています。
したがって，ブロック消去コマンド実行時のウエイト時間は，“同時選択消去”の実行時間の総和となります。
“同時選択消去の実行時間の総和”を算出するためには，同時選択消去の実行回数（M）を算出することが必要です。
Mは，同時に消去するブロック数（同時選択消去ブロック数）を求めながら，算出します。

次に同時選択消去ブロック数とMの求め方を記載します。

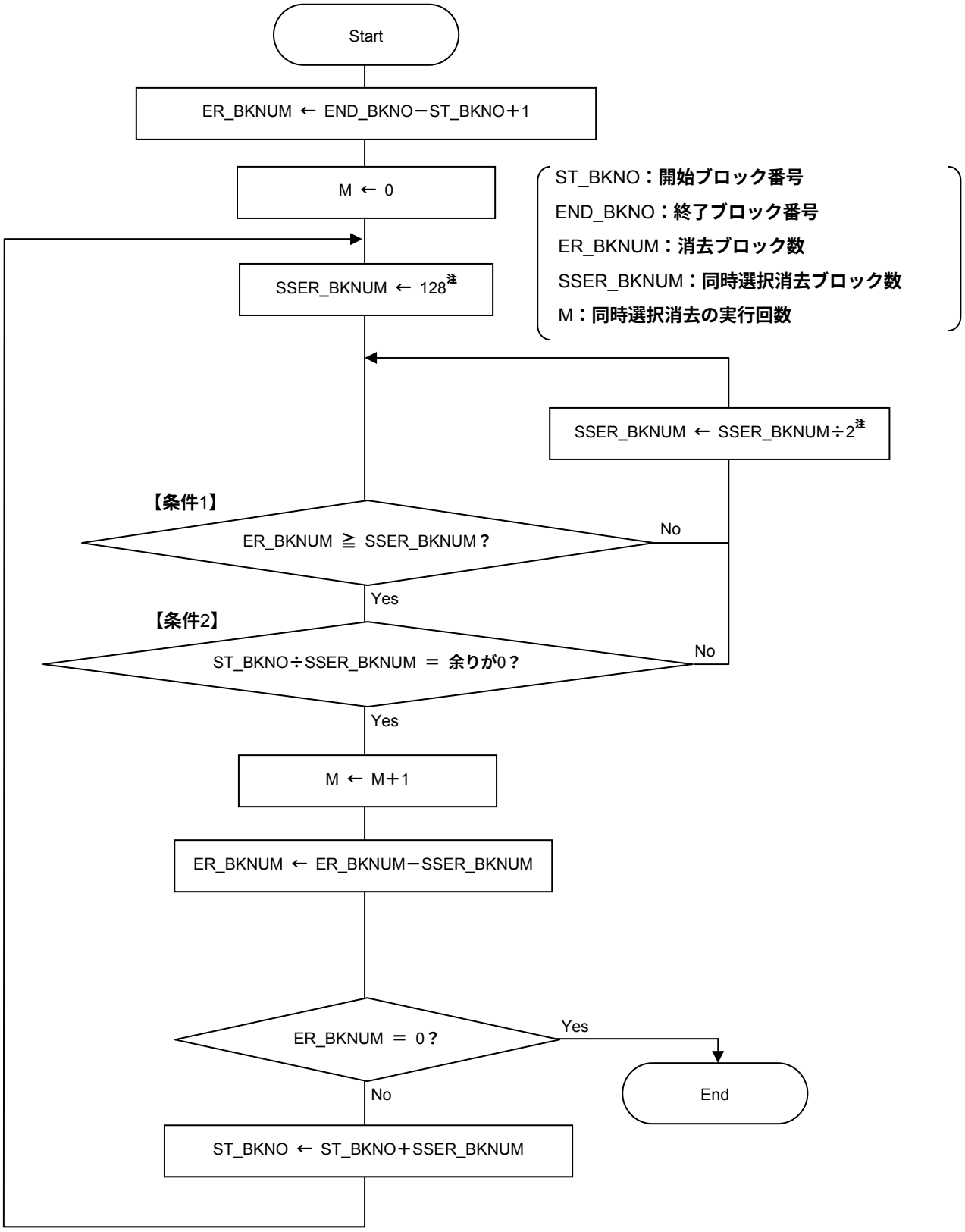
（a）同時選択消去ブロック数の求め方

選択消去ブロック数は次の条件をすべて満たす，“1, 2, 4, 8, 16, 32, 64, 128”のいずれかの数値になります。

【条件1】
消去ブロック数≧同時選択消去ブロック数
【条件2】
開始ブロック番号÷同時選択消去ブロック数 ＝ 余りが0
【条件3】
【条件1】と【条件2】を満たす最も大きな数値

(b) 同時選択消去の実行回数（M）の求め方

Mの算出方法をフローで表現すると次のようになります。



注　SSER_BKNUMの最大値（128）から，【条件1】と【条件2】に当てはまる数値を，SSER_BKNUM÷2しながら算出することで，【条件3】は満たされます。

例1　**ブロック**1～127**を消去する場合（**N（**消去するブロック数**）= 127）

① **最初の開始ブロック番号は** 1 **で，消去ブロック数が** 127 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64, 128 **となります。**
さらに【条件2**】を満たす数値は，**1**となり，【条件**3**】を満たす数値は“** 1 **”であるため，同時選択消去ブロック数は“** 1**”となることから，ブロック**1**のみ消去します。**

② **ブロック** 1 **を消去すると,次の開始ブロック番号は** 2 **で,消去ブロック数が** 126 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2**となり，【条件**3**】を満たす数値は“** 2 **”であるため，同時選択消去ブロック数は“** 2 **”となることから，ブロック**2～3**を消去します。**

③ **ブロック** 2～3 **を消去すると，次の開始ブロック番号は** 4 **で，消去ブロック数が** 124 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4**となり，【条件**3**】を満たす数値は“** 4 **”であるため，同時選択消去ブロック数は“** 4 **”となることから，ブロック**4～7**を消去します。**

④ **ブロック** 4～7 **を消去すると，次の開始ブロック番号は** 8 **で，消去ブロック数が** 120 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8**となり，【条件**3**】を満たす数値は“** 8 **”であるため，同時選択消去ブロック数は“** 8 **”となることから，ブロック**8～15**を消去します。**

⑤ **ブロック** 8～15 **を消去すると，次の開始ブロック番号は** 16 **で，消去ブロック数が** 112 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16**となり，【条件**3**】を満たす数値は“** 16 **”であるため，同時選択消去ブロック数は“** 16 **”となることから，ブロック**16～31**を消去します。**

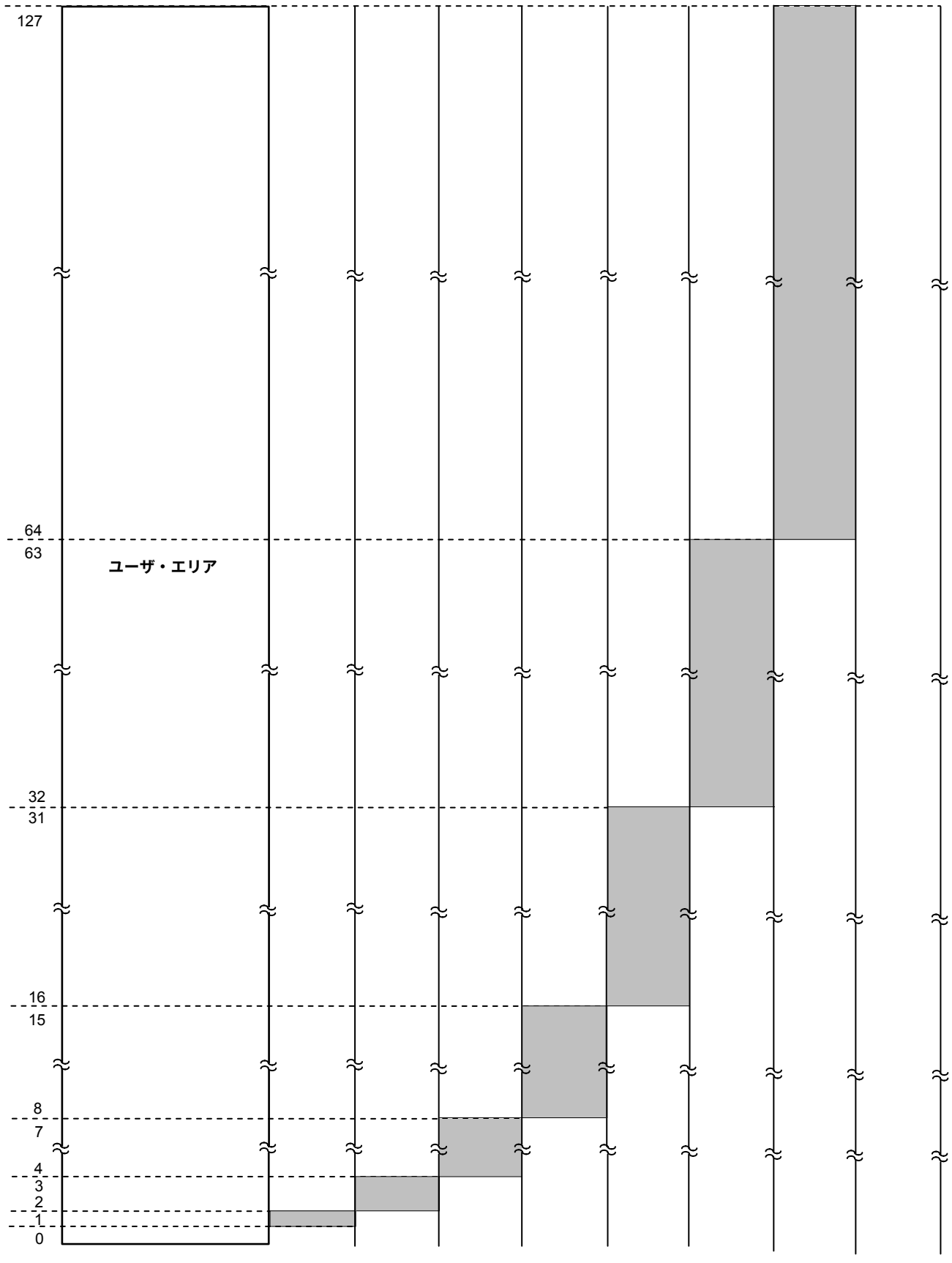
⑥ **ブロック** 16～31 **を消去すると，次の開始ブロック番号は** 32 **で，消去ブロック数が** 96 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16, 32**となり，【条件**3**】を満たす数値は“** 32 **”であるため，同時選択消去ブロック数は“** 32 **”となることから，ブロック**32～63**を消去します。**

⑦ **ブロック** 32～63 **を消去すると，次の開始ブロック番号は** 64 **で，消去ブロック数が** 64 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64**となり，【条件**3**】を満たす数値は“** 64 **”であるため，同時選択消去ブロック数は“** 64 **”となることから，ブロック**64～127**を消去します。**

以上より，ブロック1～127**を消去する場合，**1**，**2～3**，**4～7**，**8～15**，**16～31**，**32～63**，**64～127**の**7**回，同時選択消去を実行するため，**M = 7**となります。**

同時選択消去実行時のブロック構成（ブロック1～127を消去する場合）

＜ブロック番号＞



＜同時選択消去可能なブロックの範囲＞

例2　ブロック5～10を消去する場合（N（消去するブロック数）= 6）

① **最初の開始ブロック番号は 5 で，消去ブロック数が 6 であることから，【条件 1】を満たす数値は，1, 2, 4 となります。**
さらに【条件2】を満たす数値は，1となり，【条件3】を満たす数値は“ 1 ”であるため，同時選択消去ブロック数は“ 1 ”となることから，ブロック5のみ消去します。

② **ブロック 5 を消去すると，次の開始ブロック番号は 6 で，消去ブロック数が 5 であることから，【条件 1】を満たす数値は，1, 2, 4 となります。**
さらに【条件2】を満たす数値は，1, 2となり，【条件3】を満たす数値は“ 2 ”であるため，同時選択消去ブロック数は“ 2 ”となることから，ブロック6～7を消去します。

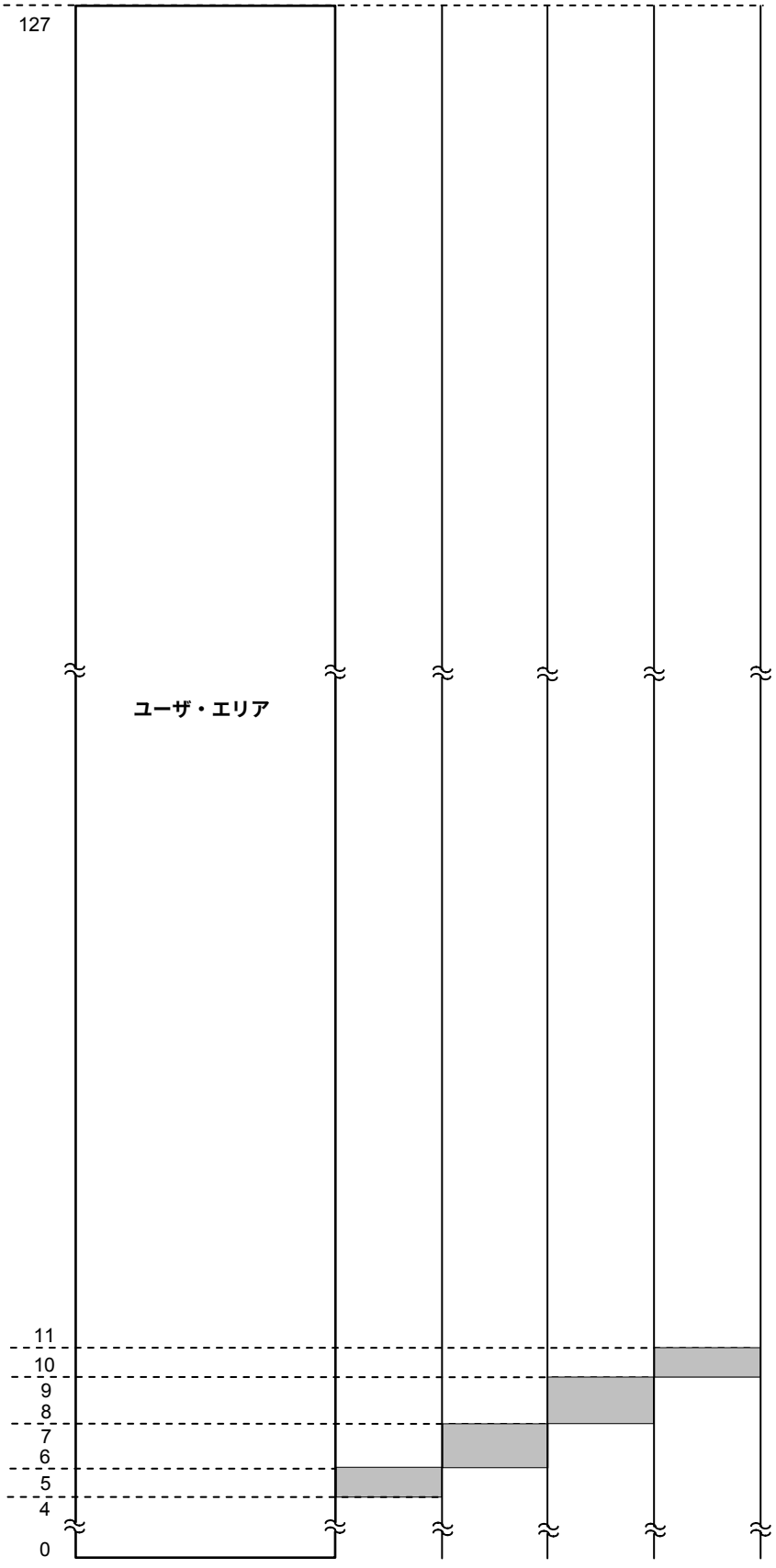
③ **ブロック 6～7 を消去すると，次の開始ブロック番号は 8 で，消去ブロック数が 3 であることから，【条件 1】を満たす数値は，1, 2 となります。**
さらに【条件2】を満たす数値は，1, 2となり，【条件3】を満たす数値は“ 2 ”であるため，同時選択消去ブロック数は“ 2 ”となることから，ブロック8～9を消去する。

④ **ブロック 8～9 を消去すると，次の開始ブロック番号は 10 で，消去ブロック数が 1 であることから，【条件 1】を満たす数値は，1 となり，これは，【条件 2】と【条件 3】も満たしているため，同時選択消去ブロック数は”1”となることから，ブロック 10 を消去する。**

以上より，ブロック5～10を消去する場合，5，6～7，8～9，10の4回，同時選択消去を実行するため，M = 4となります。

同時選択消去実行時のブロック構成（ブロック5～10を消去する場合）

＜ブロック番号＞



＜同時選択消去可能なブロックの範囲＞

例3　**ブロック**25～73**を消去する場合（**N（**消去するブロック数**）= 49）

① **最初の開始ブロック番号は** 25 **で，消去ブロック数が** 49 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32 **となります。**
さらに【条件2**】を満たす数値は，**1**となり，【条件**3**】を満たす数値は“** 1 **”であるため，同時選択消去ブロック数は“** 1 **”となることから，ブロック**25**のみ消去します。**

② **ブロック** 25 **を消去すると，次の開始ブロック番号は** 26 **で，消去ブロック数が** 48 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32 **となります。**
さらに【条件2**】を満たす数値は，**1, 2**となり，【条件**3**】を満たす数値は“** 2 **”であるため，同時選択消去ブロック数は“** 2 **”となることから，ブロック**26～27**を消去します。**

③ **ブロック** 26～27 **を消去すると，次の開始ブロック番号は** 28 **で，消去ブロック数が** 46 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4**となり，【条件**3**】を満たす数値は“** 4 **”であるため，同時選択消去ブロック数は“** 4 **”となることから，ブロック**28～31**を消去します。**

④ **ブロック** 28～31 **を消去すると，次の開始ブロック番号は** 32 **で，消去ブロック数が** 42 **であることから，【条件**1**】を満たす数値は，**1, 2, 4, 8, 16, 32**となります。**
さらに【条件 2**】を満たす数値は，**1, 2, 4, 8, 32 **となり，【条件** 3**】を満たす数値は“** 32 **”であるため，同時選択消去ブロック数は“** 32 **”となることから，ブロック** 32～63 **を消去します。**

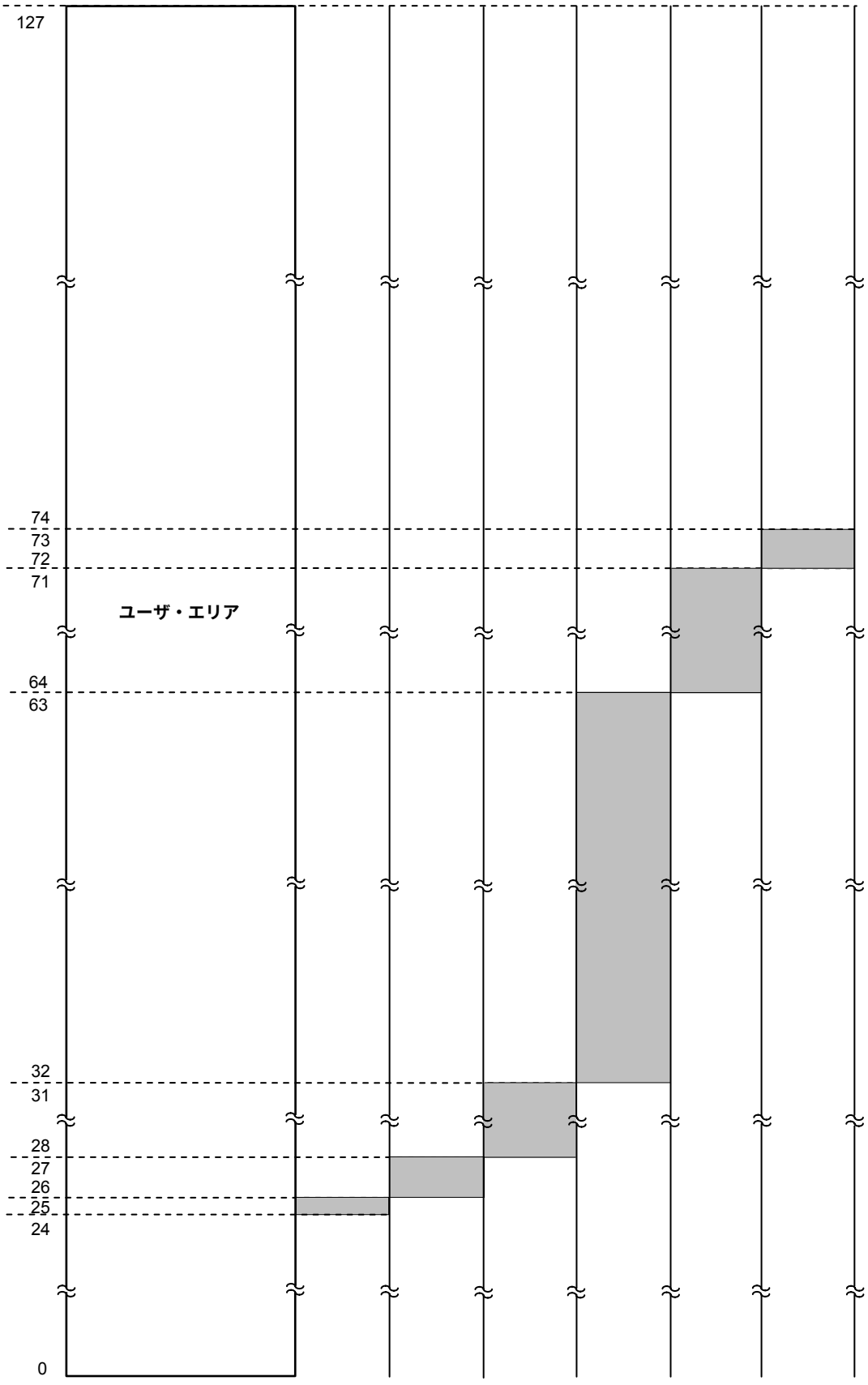
⑤ **ブロック** 32～63 **を消去すると，次の開始ブロック番号は** 64 **で，消去ブロック数が** 10 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8 **となります。**
さらに【条件 2**】を満たす数値は，**1, 2, 4, 8 **となり，【条件** 3**】を満たす数値は“** 8 **”であるため，同時選択消去ブロック数は“** 8 **”となることから，ブロック** 64～71 **を消去します。**

⑥ **ブロック** 64～71 **を消去すると，次の開始ブロック番号は** 72 **で，消去ブロック数が** 2 **であることから，【条件** 1**】を満たす数値は，**1, 2 **となります。**
さらに【条件 2**】を満たす数値は，**1, 2 **となり，【条件** 3**】を満たす数値は“** 2 **”であるため，同時選択消去ブロック数は“** 2 **”となることから，ブロック** 72～73 **を消去します。**

以上より，ブロック25～73**を消去する場合，**25**，**26～27**，**28～31**，**32～63**，**64～71**，**72～73**の**6**回消去されるため，**M = 6**となります。**

同時選択消去実行時のブロック構成（ブロック25～73を消去する場合）

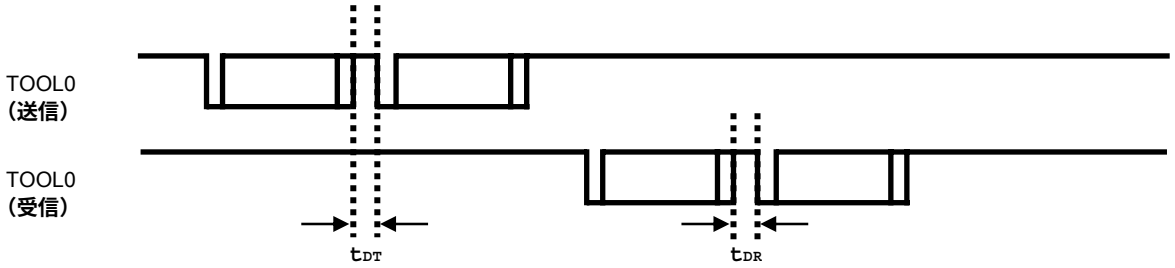
＜ブロック番号＞



＜同時選択消去可能なブロックの範囲＞

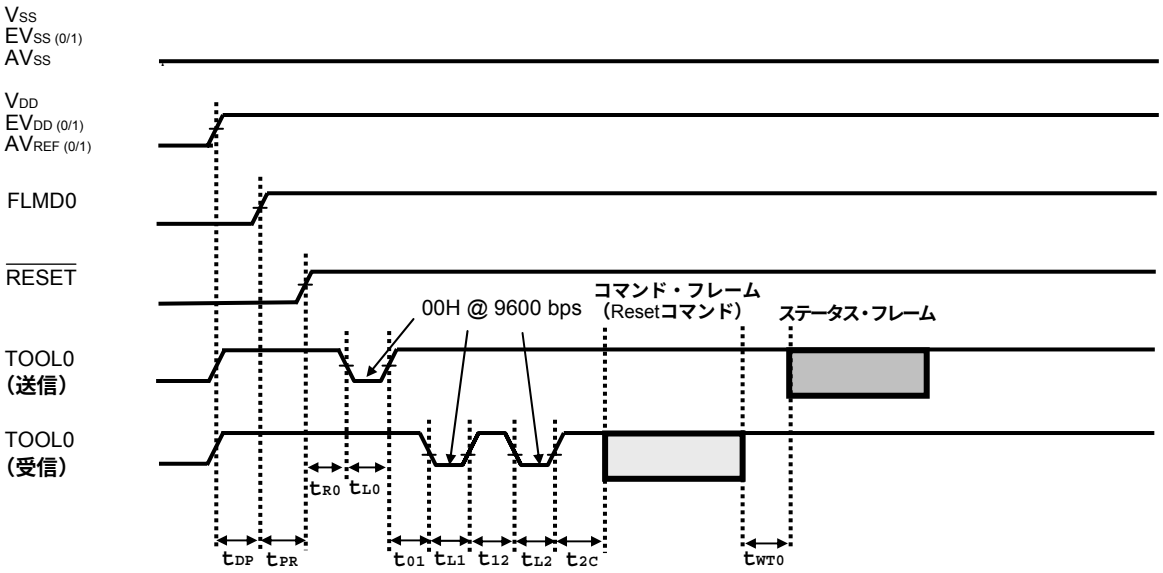
★（3）UART通信方式

（a）データ・フレーム



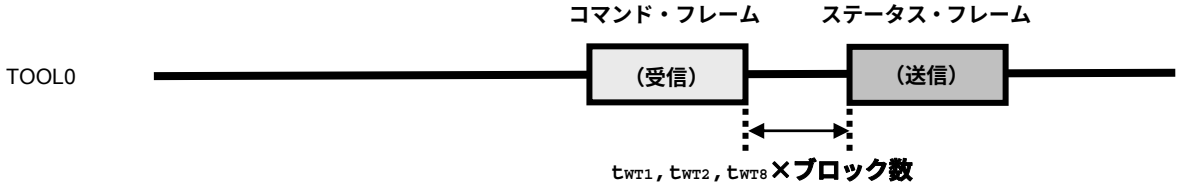
備考　上図では説明のため，TOOL0を2本に分けて記載していますが，実際は単線です。TOOL0のVDDレベルは，プルアップ抵抗によって実現します（端子はHi-Z）。

（b）プログラミング・モード設定／Resetコマンド



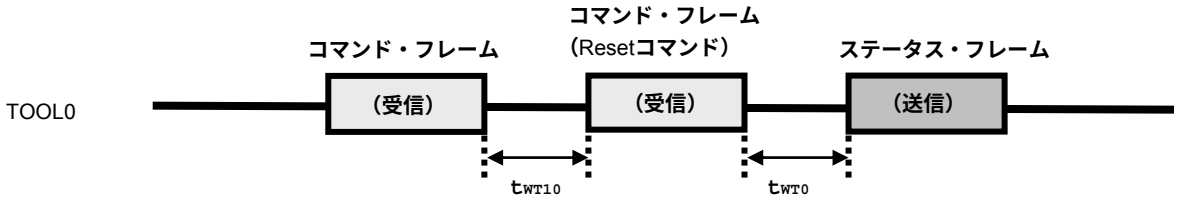
備考　上図では説明のため，TOOL0を2本に分けて記載していますが，実際は単線です。TOOL0のVDDレベルは，プルアップ抵抗によって実現します（端子はHi-Z）。

（c）Chip Eraseコマンド／Block Eraseコマンド／Block Blank Checkコマンド／Oscillating Frequency Setコマンド



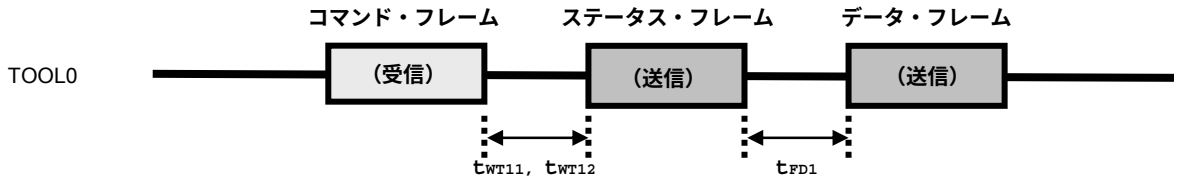
備考　上図の（）内は78K0R/Kx3の動作です。

（d）Baud Rate Setコマンド



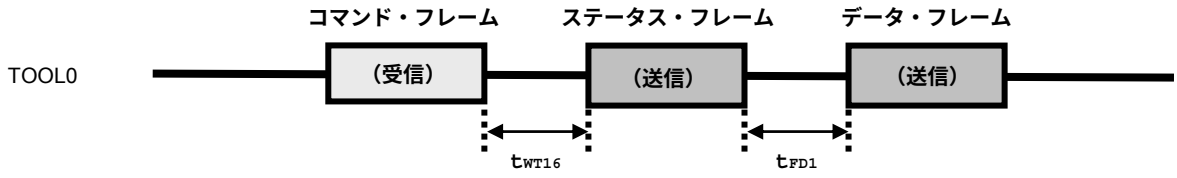
備考　上図の（）内は78K0R/Kx3**の動作です。**

（e）Sillicon Signatureコマンド／Version Getコマンド



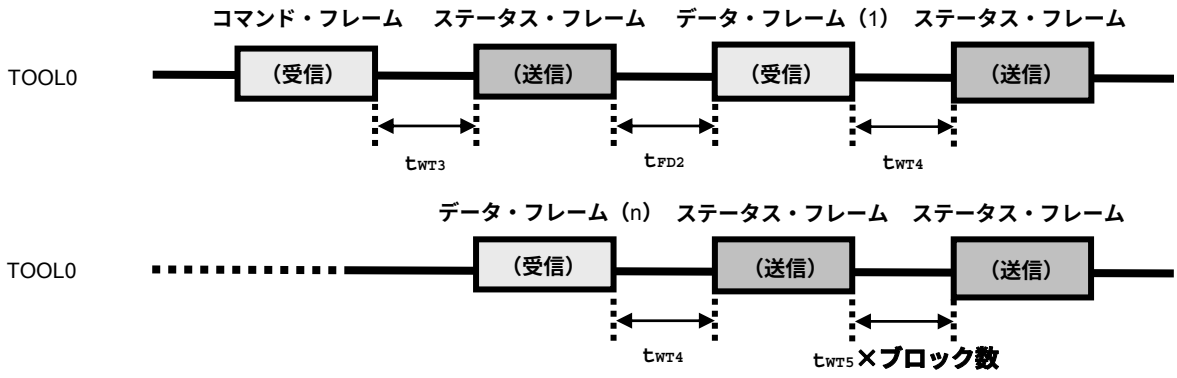
備考　上図の（）内は78K0R/Kx3**の動作です。**

（f）Checksumコマンド



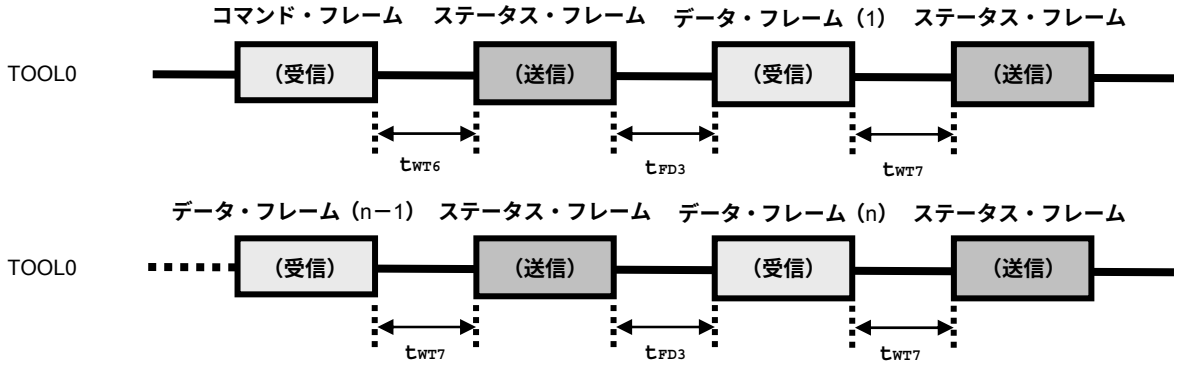
備考　上図の（）内は78K0R/Kx3**の動作です。**

（g）Programmingコマンド



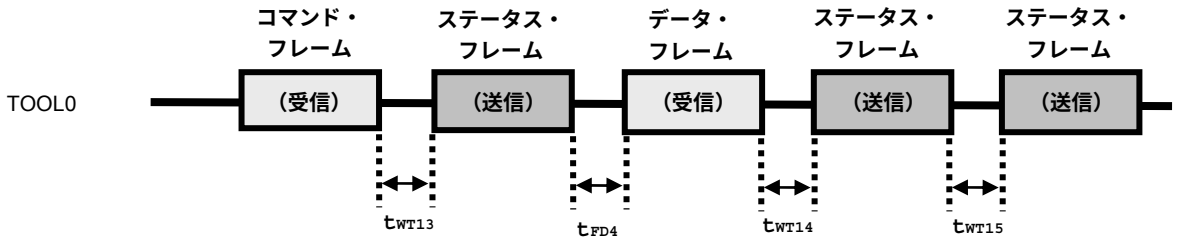
備考　上図の（）内は78K0R/Kx3**の動作です。**

(h) Verifyコマンド



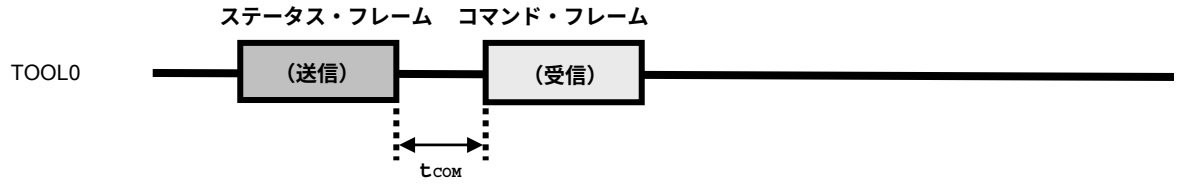
備考　上図の（）内は78K0R/Kx3の動作です。

(i) Security Setコマンド



備考　上図の（）内は78K0R/Kx3の動作です。

(j) コマンド・フレーム送信前のウエイト



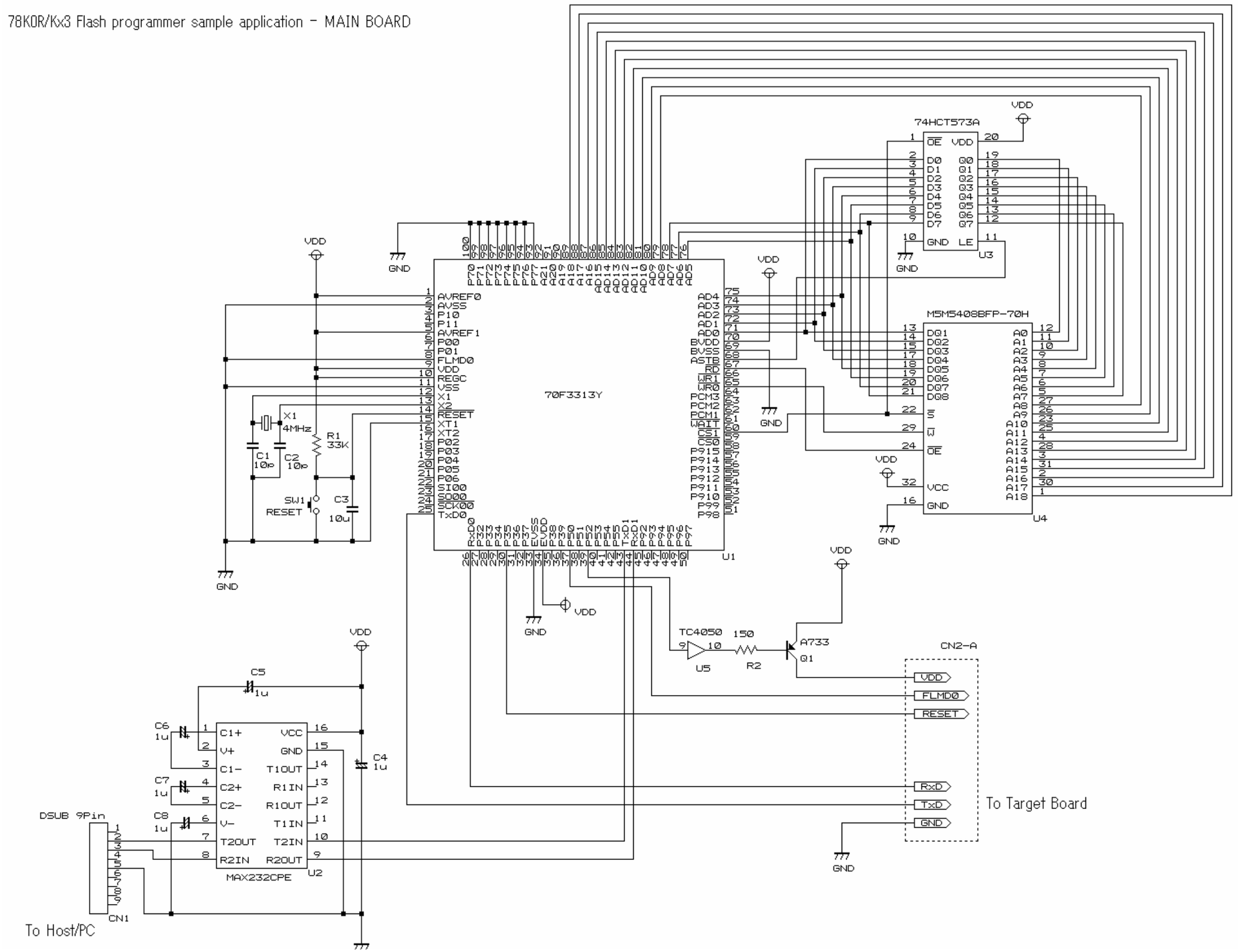
備考　上図の（）内は78K0R/Kx3の動作です。

〔メ　モ〕

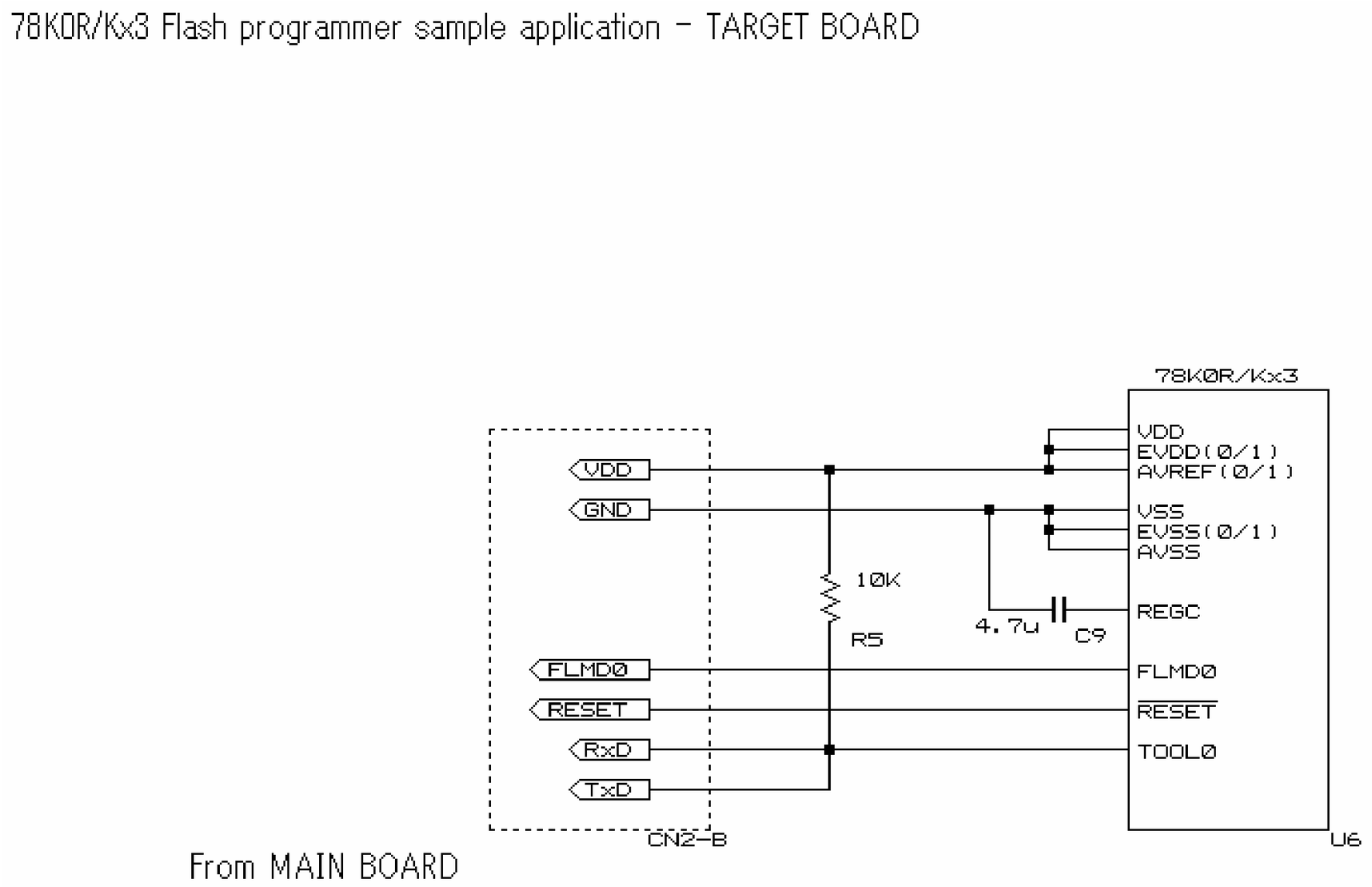
付録A　参考回路図

プログラマと78K0R/Kx3**の参考回路図を図**A－1，A－2**に示します。**

図A－1　プログラマと78K0R/Kx3の参考回路図（メイン・ボード）



図A－2　プログラマと78K0R/Kx3の参考回路図（ターゲット・ボード）



★

付録B　改版履歴

B. 1　本版で改訂された主な箇所

箇　所	内　容
第2章　プログラマ動作環境	
p.17	**表**2－1**に注を追加**
p.19	**図**2－3**に注を追加**
第7章　フラッシュ・メモリ・プログラミング・パラメータ特性	
p.109	**(1)　(b) プログラミング特性に注を追加**
p.110	**(1)　(c) コマンド特性に注**1,7,8**を追加、各コマンドの特性値を更新**
p.111	**(1)　(c) コマンド特性の備考を変更**
pp.120-122	**(3)** UART**通信方式を大幅に変更**
付録B　**改版履歴**	
p.129	**付録**B　**改版履歴を追加**

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。