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。

アプリケーション・ノート

78K0/Kx1+

8ビット・シングルチップ・マイクロコンピュータ

フラッシュ・メモリ・プログラミング（プログラマ編）

μPD78F0101H
μPD78F0102H
μPD78F0103H
μPD78F0112H
μPD78F0113H
μPD78F0114H
μPD78F0114HD
μPD78F0122H
μPD78F0123H
μPD78F0124H
μPD78F0124HD
μPD78F0132H
μPD78F0133H
μPD78F0134H
μPD78F0136H
μPD78F0138H
μPD78F0138HD
μPD78F0148H
μPD78F0148HD

資料番号　U17593JJ1V0AN00（第1版）
発行年月　September 2005 NS　CP(K)

〔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2005年9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，78K0/Kx1+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，78K0/Kx1+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
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式
- 3線式シリアルI/O（CSI）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
- 電気的特性（参考値）

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，マイクロコンピュータの一般知識を必要とします。

□78K0/Kx1+のハードウエア機能を知りたいとき
　→78K0/Kx1+ 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

関連資料

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

デバイスの関連資料

資　料　名	資料番号	
	和　　文	英　　文
78K0/KB1+　ユーザーズ・マニュアル	U16846J	U16846E
78K0/KC1+　ユーザーズ・マニュアル	U16961J	U16961E
78K0/KD1+　ユーザーズ・マニュアル	U16962J	U16962E
78K0/KE1+　ユーザーズ・マニュアル	U16899J	U16899E
78K0/KF1+　ユーザーズ・マニュアル	U16819J	U16819E
78K/0シリーズ　ユーザーズ・マニュアル　命令編	U12326J	U12326E

目　　次

第8章　3線式シリアルI/O（CSI）通信方式 … 163

第1章　フラッシュ・メモリ・プログラミング

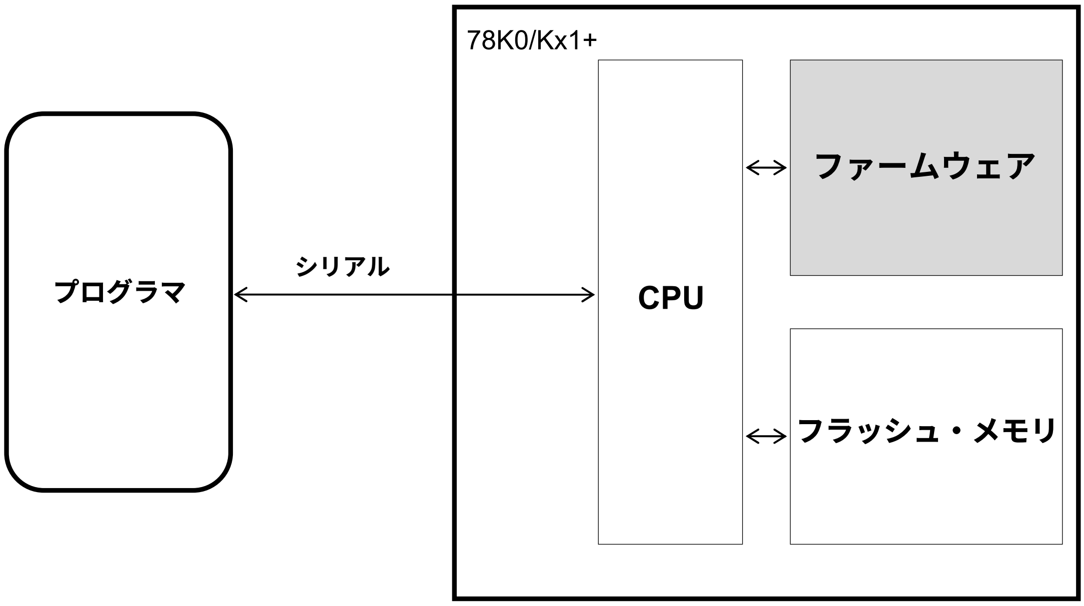
78K0/Kx1+に内蔵されるフラッシュ・メモリの書き換えを行うには，通常は専用のフラッシュ・メモリ・プログラマ（以降プログラマ）を使う必要があります。

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

78K0/Kx1+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78K0/Kx1+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　78K0/Kx1+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。

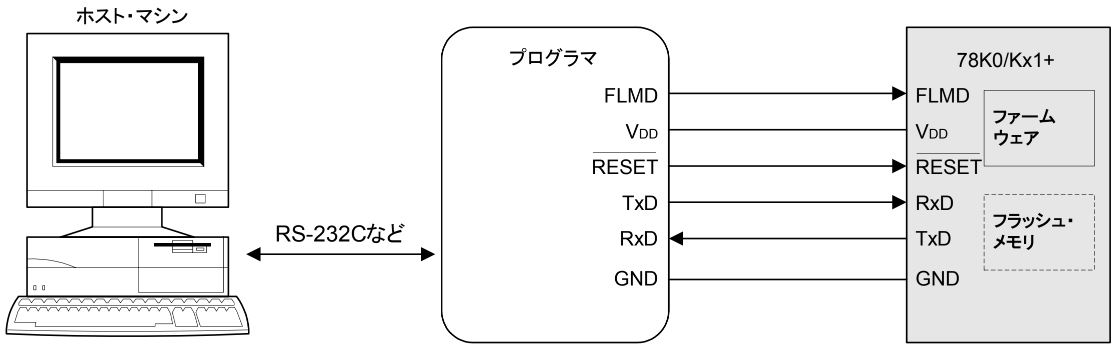
図1－2は，ホスト・マシンからの制御によりプログラマを使用するプログラミング方法を示しています。

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。

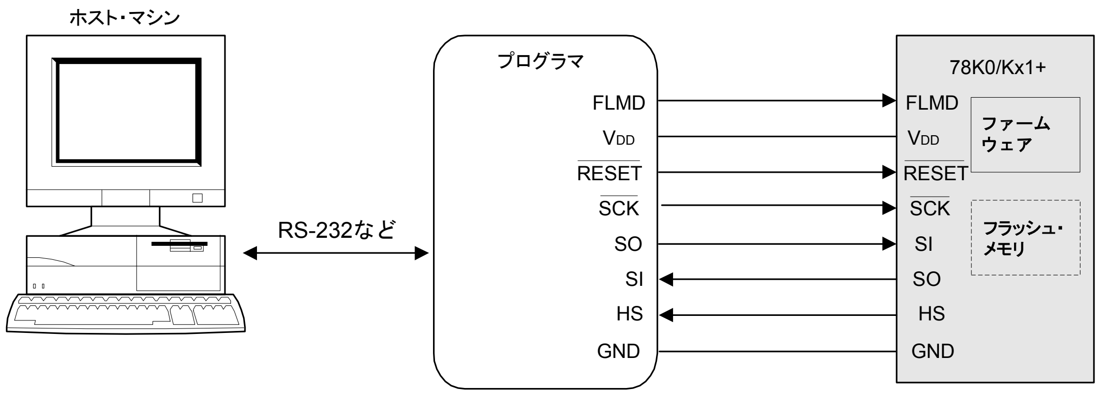
たとえば，NECエレクトロニクス製フラッシュ・メモリ・プログラマ PG-FP4は，ホスト・マシンを接続してGUIソフトウエアにより実行する方法と，スタンド・アローンで実行する方法のどちらでも動作可能です。

図1－2　システム構成

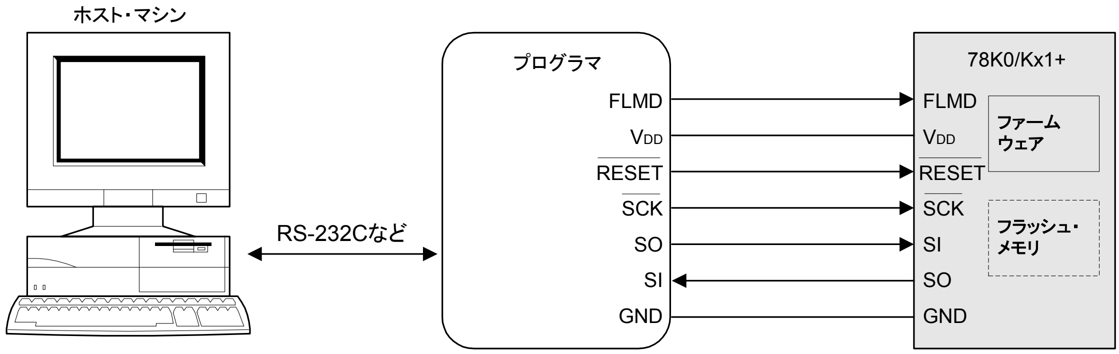
（1）UART通信方式（LSB先頭転送）



（2）3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式（MSB先頭転送）



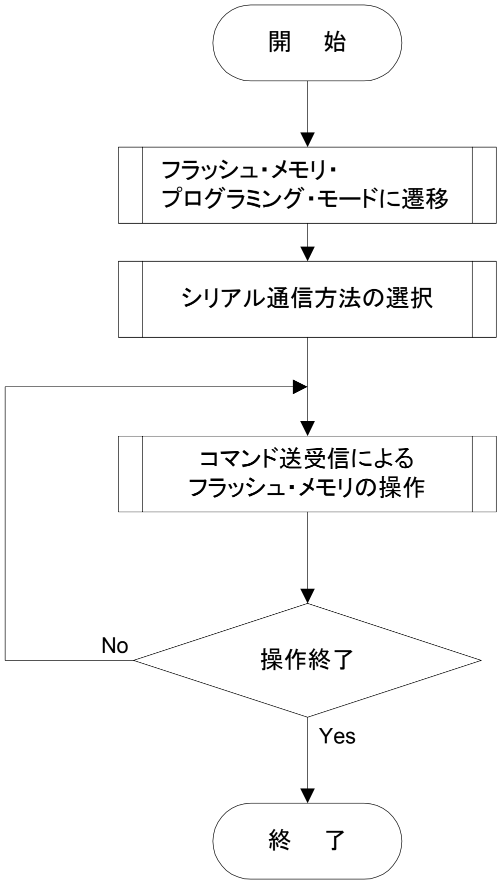
（3）3線式シリアルI/O（CSI）通信方式（MSB先頭転送）



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78K0/Kx1+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，プログラマと78K0/Kx1+間の通信方式を選択し，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

78K0/Kx1+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，FLMD1）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　シリアル通信方式の選択

フラッシュ・メモリ書き換えのためのシリアル通信の選択方法として，プログラミング・モード遷移後にフラッシュ・プログラミング・モード引き込み用端子0（FLMD0）をV_{DD}電圧とGND電圧の間で変化させてパルスを生成し，そのパルスの回数で通信方式を確定します。

1. 3. 3　コマンドの送受信によるフラッシュ・メモリの操作

78K0/Kx1+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78K0/Kx1+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78K0/Kx1+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。

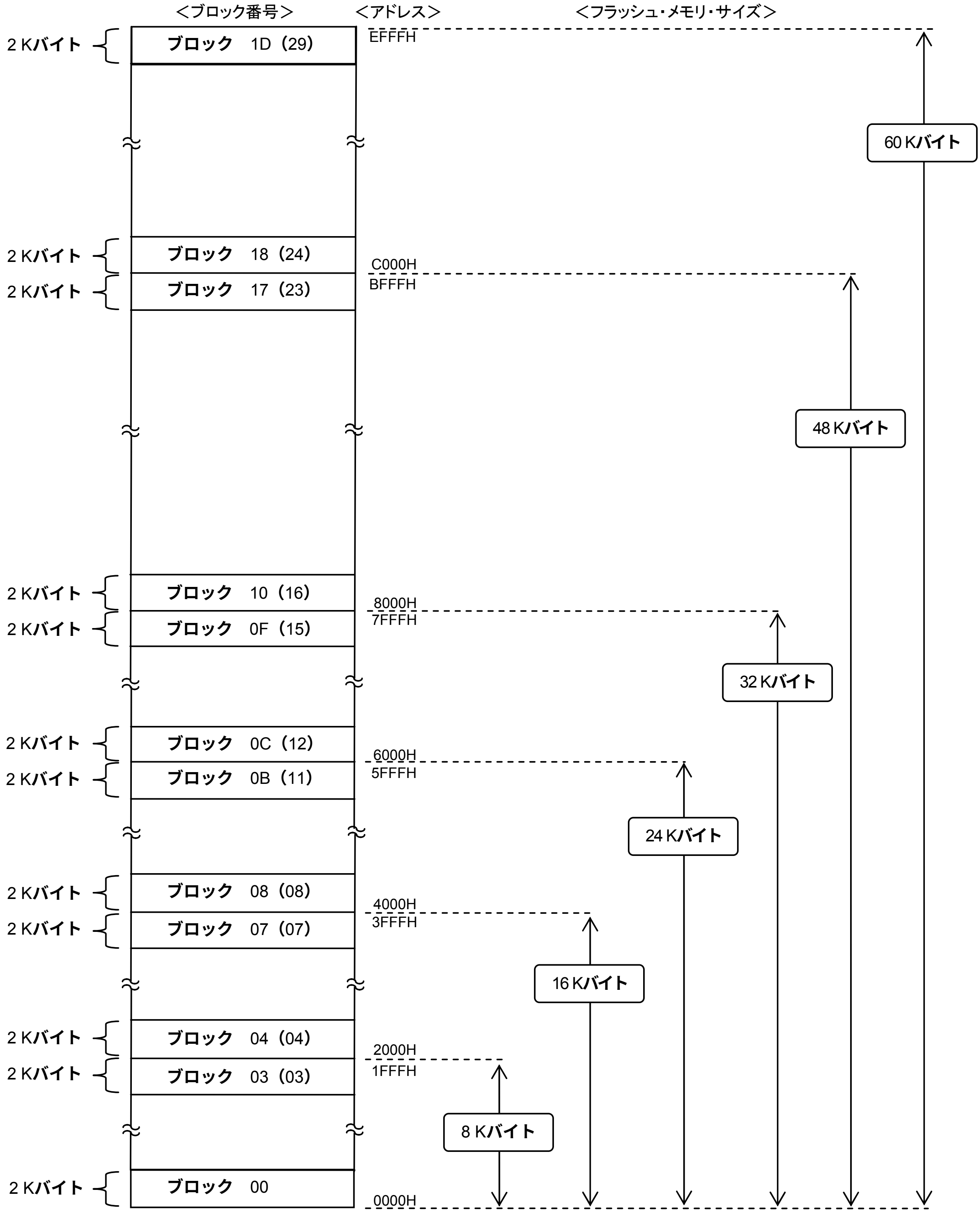
1. 4　78K0/Kx1+製品固有情報

78K0/Kx1+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78K0/Kx1+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　78K0/Kx1+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
78K0/KB1+	μPD78F0101H	8 KB
	μPD78F0102H	16 KB
	μPD78F0103H	24 KB
78K0/KC1+	μPD78F0112H	16 KB
	μPD78F0113H	24 KB
	μPD78F0114H	32 KB
	μPD78F0114HD	
78K0/KD1+	μPD78F0122H	16 KB
	μPD78F0123H	24 KB
	μPD78F0124H	32 KB
	μPD78F0124HD	
78K0/KE1+	μPD78F0132H	16 KB
	μPD78F0133H	24 KB
	μPD78F0134H	32 KB
	μPD78F0136H	48 KB
	μPD78F0138H	60 KB
	μPD78F0138HD	
78K0/KF1+	μPD78F0148H	60 KB
	μPD78F0148HD	

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			78K0/Kx1+	ターゲットとの通信方式		
信号名	入出力	端子機能	端子名	CSI	CSI＋HS	UART
FLMD0	出力	書き込みモードへの選択レベル出力および通信方式選択パルス出力	FLMD0	○	○	○
FLMD1	出力	書き込みモードへの選択レベル出力	FLMD1	○	○	○
V_{DD}	出力	V_{DD}電圧生成／電圧監視	V_{DD} EV_{DD} AV_{REF}	△	△	△
GND	－	グランド	V_{SS} EV_{SS} AV_{SS}	○	○	○
CLK	出力	78K0/Kx1+への動作クロック出力	X1, X2[注]	△	△	△
$\overline{\text{RESET}}$	出力	プログラミング・モード切り替えトリガ	$\overline{\text{RESET}}$	○	○	○
SO	出力	78K0/Kx1+へのコマンド送信	SI10	○	○	×
SI	入力	78K0/Kx1+からの応答ステータスおよび各種データの受信	SO10	○	○	×
SCK	出力	78K0/Kx1+へのシリアル・クロック供給	$\overline{\text{SCK10}}$	○	○	×
HS （ハンドシェーク）	入力	78K0/Kx1+とのシリアル通信用ハンドシェーク信号受信	HS	×	○	×
TxD	出力	78K0/Kx1+へのコマンド送信	RxD6	×	×	○
RxD	入力	78K0/Kx1+からの応答ステータスおよび各種データの受信	TxD6	×	×	○

注　X2にはX1の反転を接続してください。

備考　○：端子を使用します。
　　　×：端子を使用しません。
　　　△：ユーザ・システム側で供給されている場合は接続の必要はありません。

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, FLMD1）

FLMD0, FLMD1端子は78K0/Kx1+の動作モードを制御するための端子です。FLMD0, FLMD1端子に規定の電圧を加えリセットを解除すると，78K0/Kx1+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。

リセット解除後にFLMD0端子をV_{DD}とGND間の電圧で制御しパルスを出力することにより，プログラマと78K0/Kx1+間のシリアル通信方式を確定します。FLMD0パルス数と通信方式の関係は表2－3を参照してください。

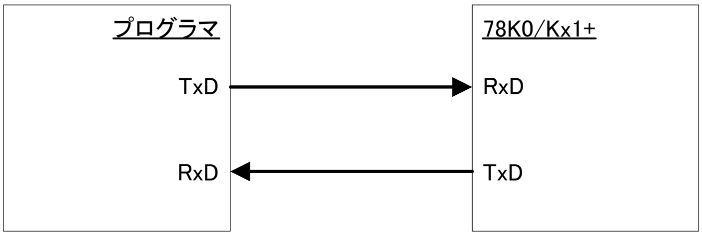
2. 2. 2　シリアル・インタフェース端子（SI, SO, SCK, TxD, RxD, HS）

シリアル・インタフェース端子は，プログラマと78K0/Kx1+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。

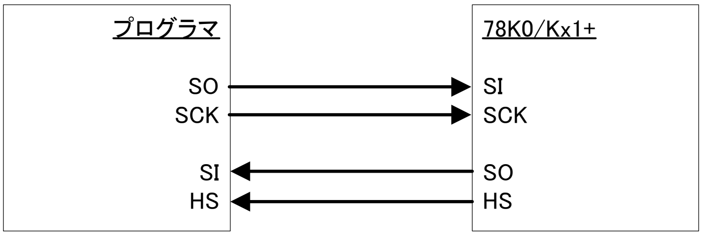
78K0/Kx1+の場合，通信方式としてUART，CSI＋HS，CSIの3種類の中から選択できます。通信方式と使用端子の接続図を次に示します。

図2－1　シリアル・インタフェース端子

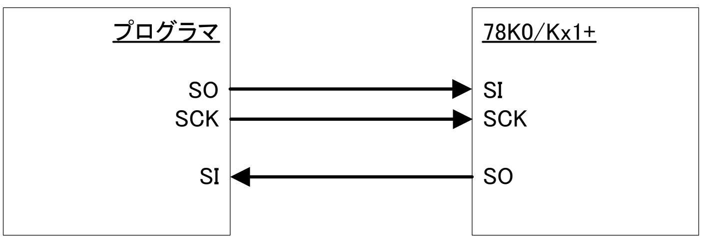
（1）UART通信方式



（2）3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式



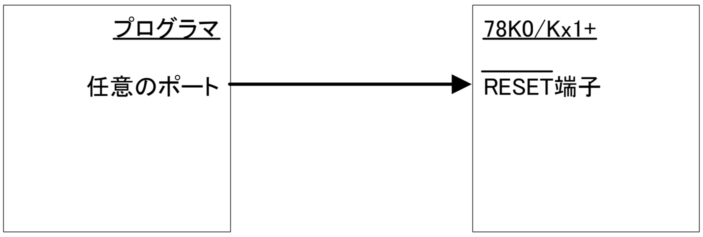
（3）3線式シリアルI/O（CSI）通信方式



2. 2. 3　リセット制御端子

リセット制御端子（$\overline{\text{RESET}}$端子）は，プログラマから78K0/Kx1+のシステム・リセットを制御するための端子です。FLMD0，FLMD1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。

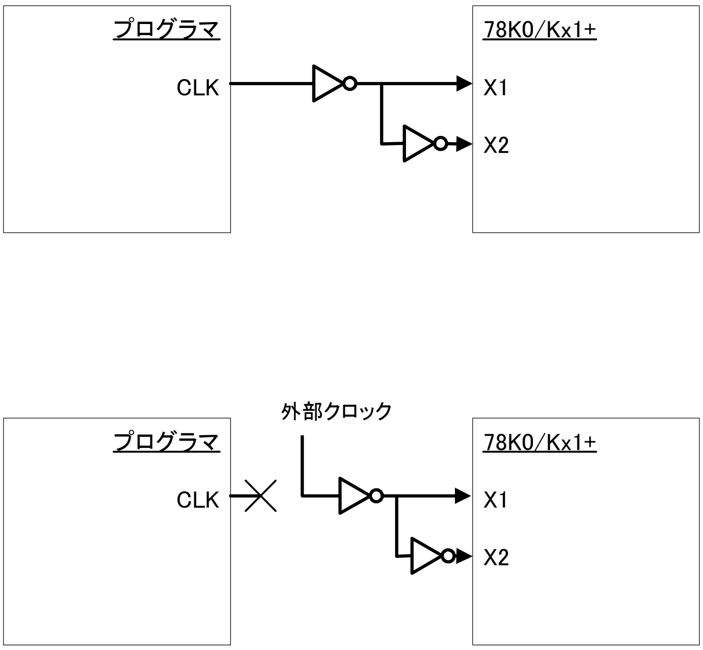
図2－2　$\overline{\text{RESET}}$端子



2. 2. 4　クロック制御端子

クロック制御端子は，プログラマが78K0/Kx1+にクロックを供給する場合のみ使用します。78K0/Kx1+の動作クロックをプログラマより供給する必要のないときは，CLK端子を接続する必要はありません。

図2－3　クロック端子

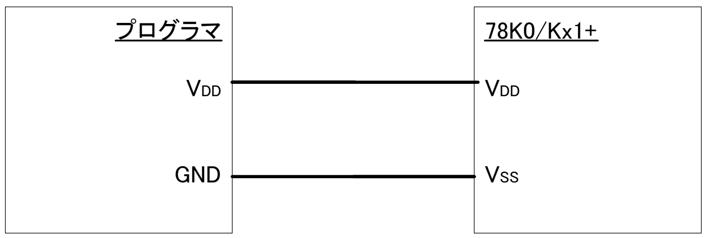


2.2.5　V_{DD}, GND制御端子

V_{DD}制御端子は，プログラマから78K0/Kx1+に電源を供給する場合に使用します。プログラマから78K0/Kx1+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78K0/Kx1+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78K0/Kx1+のV_{SS}と接続してください。

図2－4　V_{DD}/GND端子接続



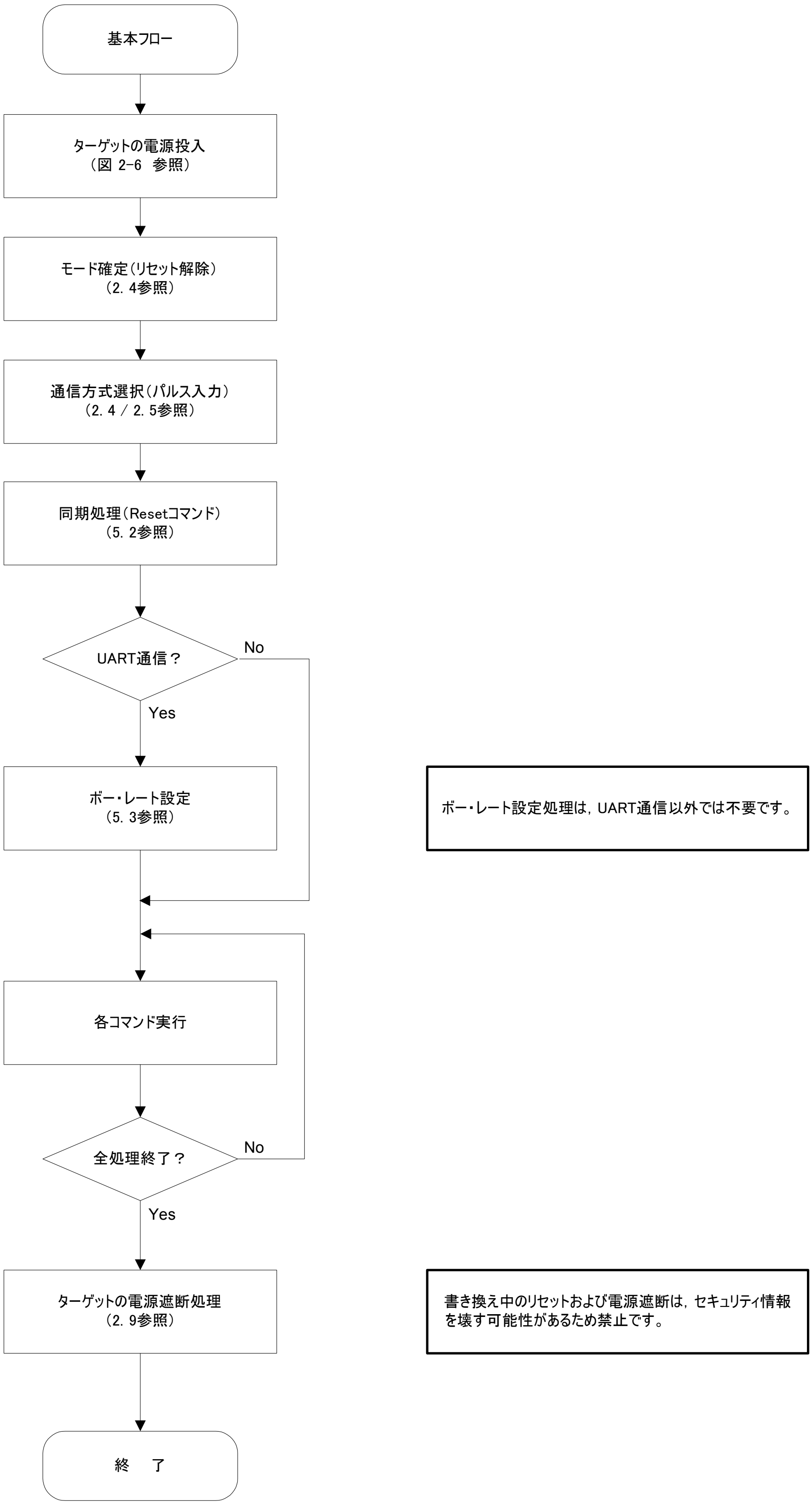
2.2.6　その他の端子

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2－5　フラッシュ処理の基本フロー・チャート



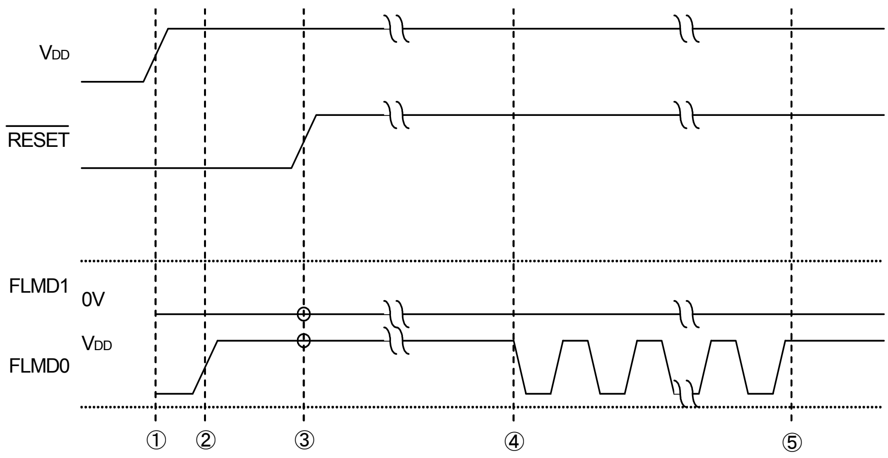
2.4　フラッシュ・メモリ・プログラミング・モードへの遷移方法

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78K0/Kx1+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，78K0/Kx1+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, FLMD1）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・モードへの遷移と通信方式の選択のタイミング図を次に示します。

図2－6　フラッシュ・メモリ・プログラミング・モードへの遷移および通信方式の選択



① ：電源（V_{DD}）投入
② ：FLMD0 = ハイ・レベル，FLMD1 = ロウ・レベル
③ ：リセット解除（モード確定）
④ ：パルス出力開始
⑤ ：パルス出力終了

リセット解除時のFLMD0, FLMD1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, FLMD1端子の設定と動作モード

FLMD0	FLMD1	動作モード
ロウ（GND）	任意	通常動作モード
ハイ（V_{DD}）	ロウ（GND）	フラッシュ・メモリ・プログラミング・モード
ハイ（V_{DD}）	ハイ（V_{DD}）	設定禁止

2. 5　シリアル通信方式の選択

フラッシュ・メモリ・プログラミング・モードへの遷移のためのリセット解除後，78K0/Kx1+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0（FLMD0）にパルスを入力することで通信方式を決定します。

FLMD0によるパルス制御に関して，FLMD0パルスのハイ／ロウ・レベルはそれぞれV_{DD}/GND電圧となります。

78K0/Kx1+にて選択できる通信方式とFLMD0端子へのパルス数および使用するポートを次に示します。

表2－3　78K0/Kx1+のFLMD0端子へのパルス数と通信方式の関係

通信方式	FLMD0 パルス数	使用通信ポート
UART（UART6）	0	TxD6（P13），RxD6（P14）
3 線式シリアル I/O（CSI10）	8	SO10（P12），SI10（P11），$\overline{\text{SCK10}}$（P10）
3 線式シリアル I/O ハンドシェーク対応（CSI10＋HS）	11	SO10（P12），SI10（P11），$\overline{\text{SCK10}}$（P10），HS（P15）
（設定禁止）	その他	－

2. 6　UART通信方式

UART通信は，RxD, TxD端子を使用します。通信条件は次のようになります。

表2－4　UART通信の通信条件

項　　目	内　　容
ボー・レート	9600 / 19200 / 31250 / 38400 / 76800 / 153600 bps のいずれかから選択（ディフォールトは 9600 bps）
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	1 ビット

CSI通信では，常にプログラマがマスタになるので，78K0/Kx1+での書き込みや消去に関しては，プログラマ側から処理が正常に終了したかどうかを確認する必要があります。しかし，UART通信では，マスタとスレーブの関係を入れ換えながら通信を行うため，CSI＋HS通信のように1端子を余分に使用せずに最適なタイミングでの通信が可能です。

注意　UART通信を行う場合は，マスタとスレーブともに同一のボー・レートにしてください。

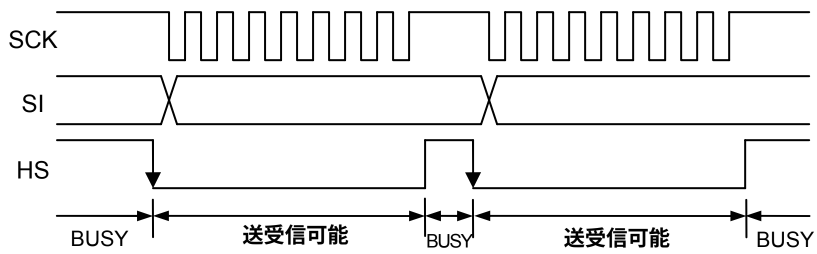
2.7　3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式

CSI＋HS通信は，コマンドやデータの通信タイミングを最適化するための通信方式です。SI, SO, SCK端子のほかにHS（ハンドシェーク）端子を使用し，効率的な通信を実現します。

HS端子は，78K0/Kx1+がデータ送受信可能な状態となったときに立ち下がります（ロウ・レベル）。プログラマは，HS端子の立ち下がり（ロウ・レベル）を確認してから，78K0/Kx1+に対してコマンドなどの送受信を開始してください。

通信のデータ形式は8ビット単位のMSB先頭です。また，クロック周波数は2.5 MHz以下にしてください。

図2－7　CSI＋HS通信のタイミング・チャート



2.8　3線式シリアルI/O（CSI）通信方式

CSI通信では，SCK, SO, SIの3端子を使用します。プログラマが常にマスタとなるため，78K0/Kx1+が送受信可能になっていない状態のときにSCK端子で送信した場合には，正常に通信できない場合があります。

通信のデータ形式は8ビット単位のMSB先頭です。また，クロック周波数は2.5 MHz以下にしてください。

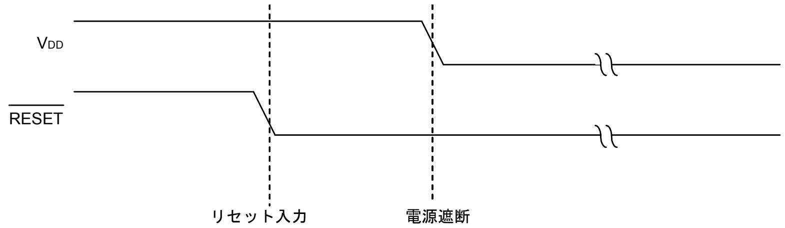
2.9　ターゲットの電源遮断処理

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2－8　フラッシュ・メモリ・プログラミングモードの終了手順



2.10　フラッシュ・メモリの操作方法

78K0/Kx1+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2－5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78K0/Kx1+に送信し，78K0/Kx1+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2－5　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 78K0/Kx1+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	78K0/Kx1+およびファームウエアのバージョンを取得します。
	ステータス取得	現在の動作状態を取得します。
	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. 11　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2－6　プログラマから78K0/Kx1+への送信コマンド一覧

コマンド番号	コマンド名	機　能
70H	Status	現在の動作状況（ステータス・データ）を取得します。
00H	Reset	通信同期検出に使用します。
90H	Oscillating Frequency Set	フラッシュ・メモリ内蔵 78K0/Kx1+の発振周波数を指定します。
9AH	Baud Rate Set	UART 選択時のボー・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	78K0/Kx1+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	78K0/Kx1+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. 12　ステータス一覧

プログラマが78K0/Kx1+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2－7　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
18H	FLMD error	書き込みエラーが発生した場合のエラー
1AH	MRG10 error	消去ベリファイ・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。

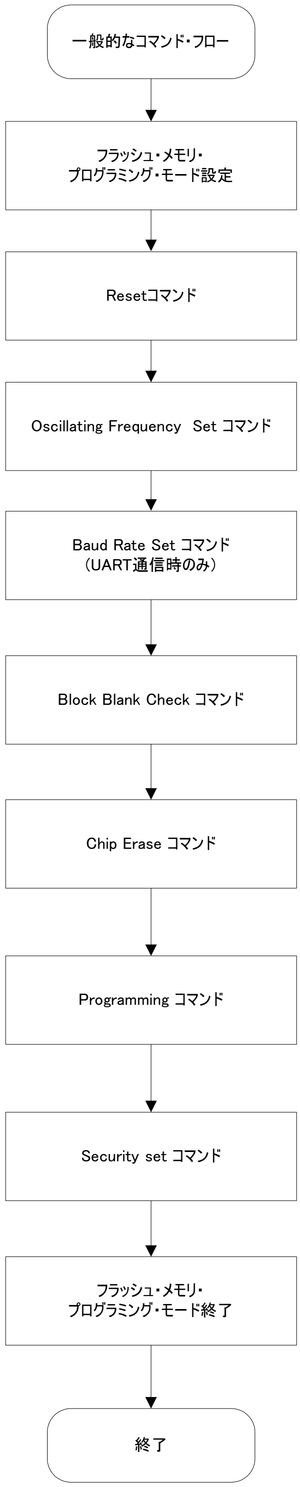
なお，このマニュアルではChecksum errorやNACKを受信した際は即時異常終了として扱っていますが，実際にプログラマを作る際は，Checksum errorやNACKが発生したコマンド送信直前のウエイトおよびHS端子のBUSYチェックからリトライしても構いません。ただし，無限にリトライを繰り返さないようにリトライの回数制限を設けることをお勧めします。

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，HS端子のタイムアウト，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78K0/Kx1+に対して電源遮断処理（2. 9　**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お勧めします。

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のフロー・チャートに示します。

図3－1　書き換え時の一般的なコマンド・フロー



備考　**その他に，**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78K0/Kx1+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。78K0/Kx1+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	LEN	COM	コマンド情報（可変長）	SUM	ETX
（1 バイト）	（1 バイト）	（1 バイト）	（最大 255 バイト）	（1 バイト）	（1 バイト）

図4－2　データ・フレームのフォーマット

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

通信モードごとの各コマンド処理において，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，次の節をお読みください。

- UART通信方式の場合は，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。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7.1　**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。
- 3線式シリアルI/O（CSI）通信方式の場合は，8.1　**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。

通信モードごとの各コマンド処理において，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，次の節をお読みください。

- UART通信方式の場合は，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。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7.2　**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。
- 3線式シリアルI/O（CSI）通信方式の場合は，8.2　**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。

通信モードごとの各コマンド処理において，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，次の節をお読みください。

- UART通信方式の場合は，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。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7.3　**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。
- 3線式シリアルI/O（CSI）通信方式の場合は，8.3　**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の78K0/Kx1+の動作状態を確認します。

Statusコマンド発行後，通信の問題などで78K0/Kx1+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，78K0/Kx1+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，Statusコマンドをリトライしてください。

5. 1. 2　コマンド・フレームとステータス・フレーム

Statusコマンドのコマンド・フレームは図5－1，そのコマンドに対するステータス・フレームは図5－2のようになります。

図5－1　Statusコマンド・フレーム（プログラマから78K0/Kx1+へ）

SOH	LEN	COM	SUM	ETX
01H	01H	70H（Status）	Checksum	03H

図5－2　Statusコマンドに対するステータス・フレーム（78K0/Kx1+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，78K0/Kx1+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。

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

・UART通信方式の場合は，Statusコマンドを使用しません。
・3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 4　Statusコマンドをお読みください。
・3線式シリアルI/O（CSI）通信方式の場合は，8. 4　Statusコマンドをお読みください。

注意　UART通信の場合は，書き込み／消去などの各コマンド送信後，一定時間内に78K0/Kx1+から自動的にステータス・フレームを返してきます。そのため，Statusコマンドは使用しません。
もしUART通信時にStatusコマンドを送信した場合はCommand Number Errorとなります。

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78K0/Kx1+間の通信が確立されたことを確認します。

78K0/Kx1+との通信方式にUART通信を選択した場合，プログラマと78K0/Kx1+は同じボー・レートである必要がありますが，78K0/Kx1+は自身の動作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，78K0/Kx1+はその“ 00H ”のロウ幅を測定し2回の平均値を計算することで初めて自身の動作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5－3，そのコマンドに対するステータス・フレームは図5－4のようになります。

図5－3　Resetコマンド・フレーム（プログラマから78K0/Kx1+へ）

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

図5－4　Resetコマンドに対するステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

・UART通信方式の場合は，6. 4　Resetコマンドをお読みください。
・3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 5　Resetコマンドをお読みください。
・3線式シリアルI/O（CSI）通信方式の場合は，8. 5　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。

Baud Rate Setコマンドは，UART通信時のみ有効で，ボー・レート設定データは1バイトの数値で表されます。

UART通信時以外で，Baud Rate Setコマンドを送信した場合，78K0/Kx1+は無視します。

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

図5－5　Baud Rate Setコマンド・フレーム（プログラマから78K0/Kx1+へ）

SOH	LEN	COM	コマンド情報	SUM	ETX
01H	02H	9AH （Baud Rate Set）	D01	Checksum	03H

備考　D01　：ボー・レート選択値

D01 値	03H	04H	05H	06H	07H	08H
ボー･レート （bps）	9600	19200	31250	38400	76800	153600

図5－6　Baud Rate Setコマンドに対するステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 5　Baud Rate Setコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，Baud Rate Setコマンドを使用しません。
- 3線式シリアルI/O（CSI）通信方式の場合は，Baud Rate Setコマンドを使用しません。

5. 4　Oscillating Frequency Setコマンド

5. 4. 1　説明

78K0/Kx1+の発振周波数のデータを設定します。

実際に78K0/Kx1+のX1端子に入力されているクロックの周波数を指定してください。

5. 4. 2　コマンド・フレームとステータス・フレーム

Oscillating Frequency Setコマンドのコマンド・フレームは図5－7，そのコマンドに対するステータス・フレームは図5－8のようになります。

図5－7　Oscillating Frequency Setコマンド・フレーム（プログラマから78K0/Kx1+へ）

SOH	LEN	COM	コマンド情報				SUM	ETX
01H	05H	90H（Oscillating Frequency Set）	D01	D02	D03	D04	Checksum	03H

備考　D01 - D04　：発振周波数　=（D01×0.1＋D02×0.01＋D03×0.001）×10^{D04}　（単位：kHz）
設定可能範囲は10 kHzから100 MHzですが，実際にコマンドを送信する際は各デバイスの仕様に合わせてください。
D01 - D03はアンパックドBCDで，D04は符号付き整数です。

設定例　：6 MHzの場合
D01 = 06H
D02 = 00H
D03 = 00H
D04 = 04H
発振周波数　= 0.1　×　6　×　10^4 = 6000 kHz = 6 MHz

設定例　：10 MHzの場合
D01 = 01H
D02 = 00H
D03 = 00H
D04 = 05H
発振周波数　= 1　×　0.1　×　10^5 = 10000 kHz = 10 MHz

図5－8　Oscillating Frequency Setコマンドに対するステータス・フレーム（78K0/Kx1+からプログラマへ）

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

備考　ST1　：発振周波数設定結果

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 6　Oscillating Frequency Setコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 6　Oscillating Frequency Setコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 6　Oscillating Frequency Setコマンドをお読みください。

5. 5　Chip Eraseコマンド

5. 5. 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. 13　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

Chip Eraseコマンドのコマンド・フレームは図5－9，そのコマンドに対するステータス・フレームは，図5－10のようになります。

図5－9　Chip Eraseコマンド・フレーム（プログラマから78K0/Kx1+へ）

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

図5－10　Chip Eraseコマンドに対するステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 7　Chip Eraseコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 7　Chip Eraseコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 7　Chip Eraseコマンドをお読みください。

5. 6　Block Eraseコマンド

5. 6. 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。

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. 13　Security Setコマンド参照）。

5. 6. 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5－11，そのコマンドに対するステータス・フレームは図5－12のようになります。

図5－11　Block Eraseコマンド・フレーム（プログラマから78K0/Kx1+へ）

SOH	LEN	COM	コマンド情報	SUM	ETX
01H	02H	22H （Block Erase）	BLK	Checksum	03H

備考　BLK　：消去するブロック番号

図5－12　Block Eraseコマンドに対するステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 8　Block Eraseコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 8　Block Eraseコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 8　Block Eraseコマンドをお読みください。

5. 7　Programmingコマンド

5. 7. 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，書き込みバイト数分のデータを送信します。それにより，ユーザ・プログラムをフラッシュ・メモリに書きこみ，内部ベリファイを行います。

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，78K0/Kx1+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Statusコマンドの送信が必要となります。

5. 7. 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5－13，そのコマンドに対するステータス・フレームは図5－14のようになります。

図5－13　Programmingコマンド・フレーム（プログラマから78K0/Kx1+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　：消去開始アドレス
　　　EAH - EAL　：消去終了アドレス

図5－14　Programmingコマンドに対するステータス・フレーム（78K0/Kx1+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

書き込みを行うデータのデータ・フレームは図5－15，そのデータ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　書き込みを行うデータ・フレーム（プログラマから78K0/Kx1+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5－16　データ・フレームに対するステータス・フレーム（78K0/Kx1+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. 7. 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5－17のようになります。

図5－17　全データ転送完了後のステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 9　Programmingコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 9　Programmingコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 9　Programmingコマンドをお読みください。

5. 8　Verifyコマンド

5. 8. 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78K0/Kx1+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. 8. 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5－18，そのコマンドに対するステータス・フレームは図5－19のようになります。

図5－18　Verifyコマンド・フレーム（プログラマから78K0/Kx1+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5－19　Verifyコマンドに対するステータス・フレーム（78K0/Kx1+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. 8. 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5－20，そのデータに対するステータス・フレームは図5－21のようになります。

図5－20　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78K0/Kx1+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5－21　データ・フレームに対するステータス・フレーム（78K0/Kx1+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。**

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART**通信方式の場合は，**6. 10　Verify**コマンドをお読みください。**
- 3**線式シリアル**I/O **ハンドシェーク対応**（CSI＋HS）**通信方式の場合は，**7. 10　Verify**コマンドをお読みください。**
- 3**線式シリアル**I/O（CSI）**通信方式の場合は，**8. 10　Verify**コマンドをお読みください。**

5. 9　Block Blank Checkコマンド

5. 9. 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。

5. 9. 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5－22，そのコマンドに対するステータス・フレームは図5－23のようになります。

図5－22　Block Blank Checkコマンド・フレーム（プログラマから78K0/Kx1+へ）

SOH	LEN	COM	コマンド情報	SUM	ETX
01H	02H	32H （Block Blank Check）	BLK	Checksum	03H

備考　BLK　：ブランク・チェックを行うブロック番号

図5－23　Block Blank Checkコマンドに対するステータス・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 11　Block Blank Checkコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 11　Block Blank Checkコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 11　Block Blank Checkコマンドをお読みください。

5. 10　Silicon Signatureコマンド

5. 10. 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）を読み出します。

たとえば，プログラマが78K0/Kx1+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バイト目と3バイト目の値に従い，適切なプロトコルを選択します。

5. 10. 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5－24，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　Silicon Signatureコマンド・フレームのフォーマット（プログラマから78K0/Kx1+へ）

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

図5－25　Silicon Signatureコマンドに対するステータス・フレーム（78K0/Kx1+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　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5－26のようになります。

図5－26　シリコン・シグネチャ・データ・フレーム（78K0/Kx1+からプログラマへ）

STX	LEN	Data				SUM	ETX
02H	n	VEN	EXT	FNC	INVALID DATA	Checksum	03H

備考1.　n（LEN）　：データ長
VEN　：ベンダー・コード（NEC：10H）
EXT　：拡張コード
FNC　：機能情報
INVALID DATA　：長さ90～198バイトの無効データです。

2.　上記ベンダー・コード（VEN），拡張コード（EXT），機能情報（FNC）は，上位1ビットを奇数パリティとして使用します。次に例を示します。

表5－1　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[注]	実際の値
VEN	ベンダー・コード（NEC）	1	10H　(00010000B)	10H
EXT	拡張情報　（固定値）	1	7FH　(01111111B)	7FH
FNC	機能情報　（固定値）	1	01H　(00000001B)	01H

注　0 は奇数パリティ（バイト中の1の数を奇数するための調整値）

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 12　Silicon Signatureコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 12　Silicon Signatureコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 12　Silicon Signatureコマンドをお読みください。

5. 11　Version Getコマンド

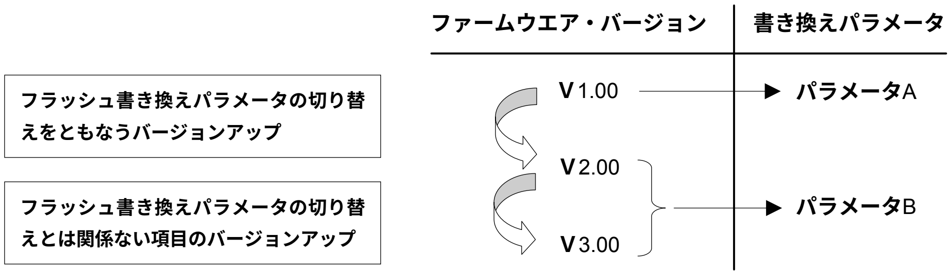
5. 11. 1　説明

78K0/Kx1+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。

書き換え用パラメータを78K0/Kx1+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. 11. 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5－28，そのコマンドに対するステータス・フレームは図5－29のようになります。

図5－28　Version Getコマンド・フレーム（プログラマから78K0/Kx1+へ）

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

図5－29　Version Getコマンドに対するステータス・フレーム（78K0/Kx1+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　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5－30のようになります。

図5－30　バージョン・データ・フレーム（78K0/Kx1+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
DV2　：デバイス・バージョン小数点第一位
DV3　：デバイス・バージョン小数点第二位
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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 13　Version Getコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 13　Version Getコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 13　Version Getコマンドをお読みください。

5. 12　Checksumコマンド

5. 12. 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. 12. 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5－31，そのコマンドに対するステータス・フレームは図5－32のようになります。

図5－31　Checksumコマンド・フレーム（プログラマから78K0/Kx1+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5－32　Checksumコマンドに対するステータス・フレーム（78K0/Kx1+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. 12. 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5－33のようになります。

図5－33　チェックサム・データ・フレーム（78K0/Kx1+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

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

- UART通信方式の場合は，6. 14　Checksumコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 14　Checksumコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 14　Checksumコマンドをお読みください。

5. 13　Security Setコマンド

5. 13. 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，チップ消去の許可／禁止）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ます。

注意　一度セキュリティ設定をした場合，セキュリティ・フラグの追加設定や禁止から許可への変更は不可能で，これらを行おうとした場合Write error（1CH）が発生します。セキュリティ・フラグの再設定を行う場合は，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）。ただし，チップ消去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お勧めします。

5. 13. 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5－34，そのコマンドに対するステータス・フレームは図5－35のようになります。

Security Setコマンド・フレームには，ブロック番号とページ番号のフィールドがありますが，特に意味を持ちませんので，両フィールドともに00Hを設定してください。

図5－34　Security Setコマンド・フレーム（プログラマから78K0/Kx1+へ）

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

備考　BLK, PAG　：00H固定

図5－35　Security Setコマンドに対するステータス・フレーム（78K0/Kx1+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. 13. 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5－36，そのデータに対するステータス・フレームは図5－37のようになります。

図5－36　セキュリティ・データ・フレーム（プログラマから78K0/Kx1+へ）

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

備考　FLG　：セキュリティ・フラグ

図5－37　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（78K0/Kx1+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. 13. 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5－38のようになります。

図5－38　内部ベリファイ確認に対するステータス・フレーム（78K0/Kx1+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5－2　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	
ビット3	
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5－3　セキュリティ・フラグ・フィールドと各動作の禁止／許可

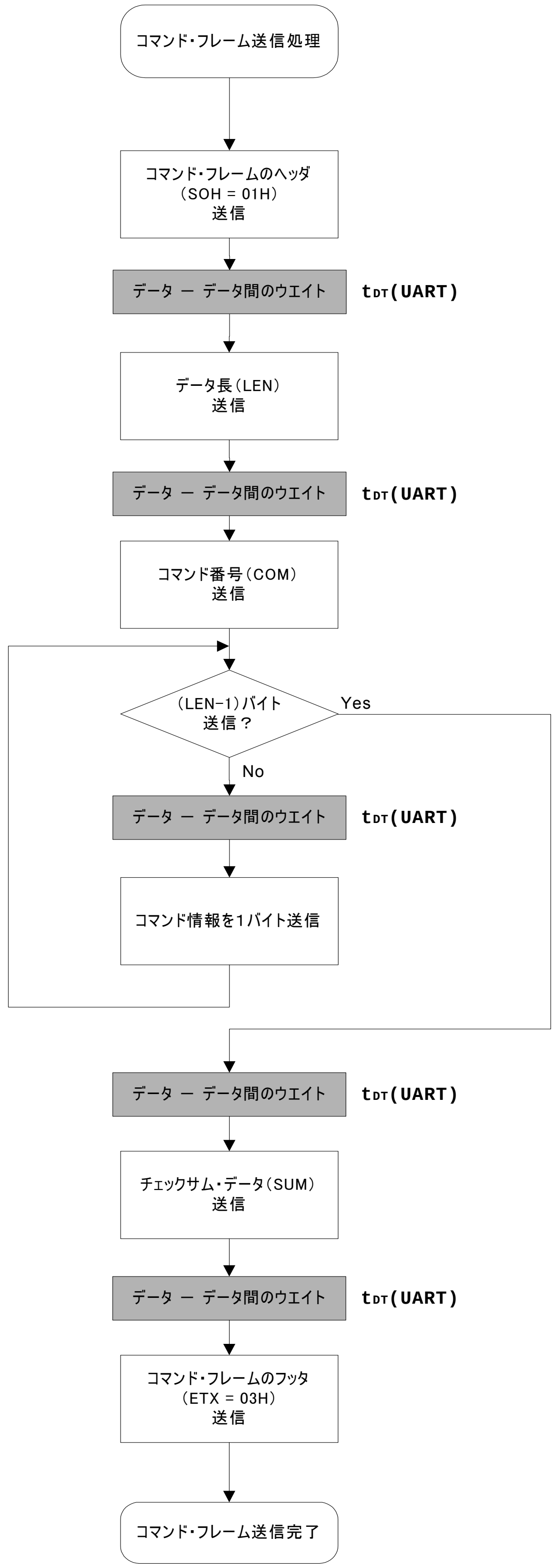
動作モード	フラッシュ・メモリ・プログラミング・モード			セルフ・プログラミング・モード
コマンド セキュリティ設定項目	セキュリティ設定後のコマンド動作 ○：実行可能　×：実行不可			・セキュリティ設定値にかかわらず，全コマンド実行可能 ・セキュリティ設定値の保持のみ可能
	Programming	Chip Erase	Block Erase	
書き込み禁止	×	○	×	
チップ消去禁止	○	×	×	
ブロック消去禁止	○	○	×	

通信方式ごとの，プログラマと78K0/Kx1+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，次の節をお読みください。

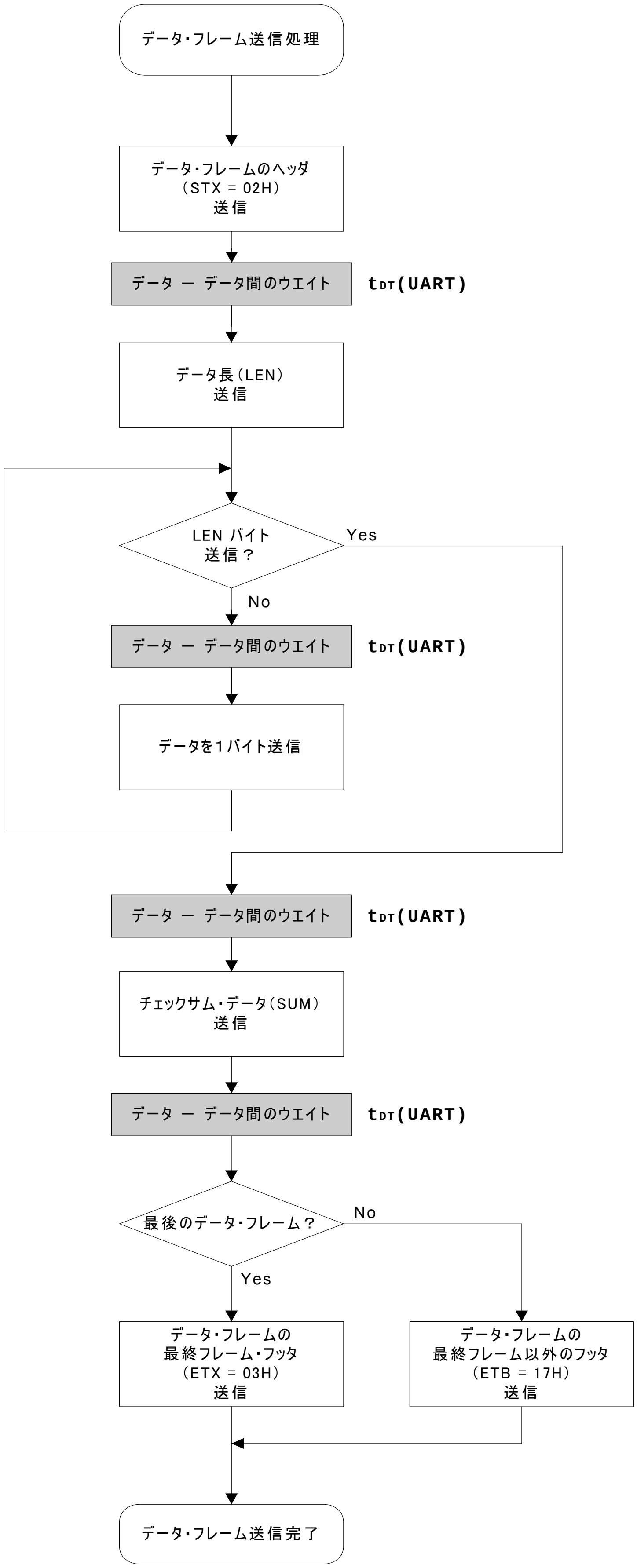
- UART通信方式の場合は，6. 15　Security Setコマンドをお読みください。
- 3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式の場合は，7. 15　Security Setコマンドをお読みください。
- 3線式シリアルI/O（CSI）通信方式の場合は，8. 15　Security Setコマンドをお読みください。

第6章　UART通信方式

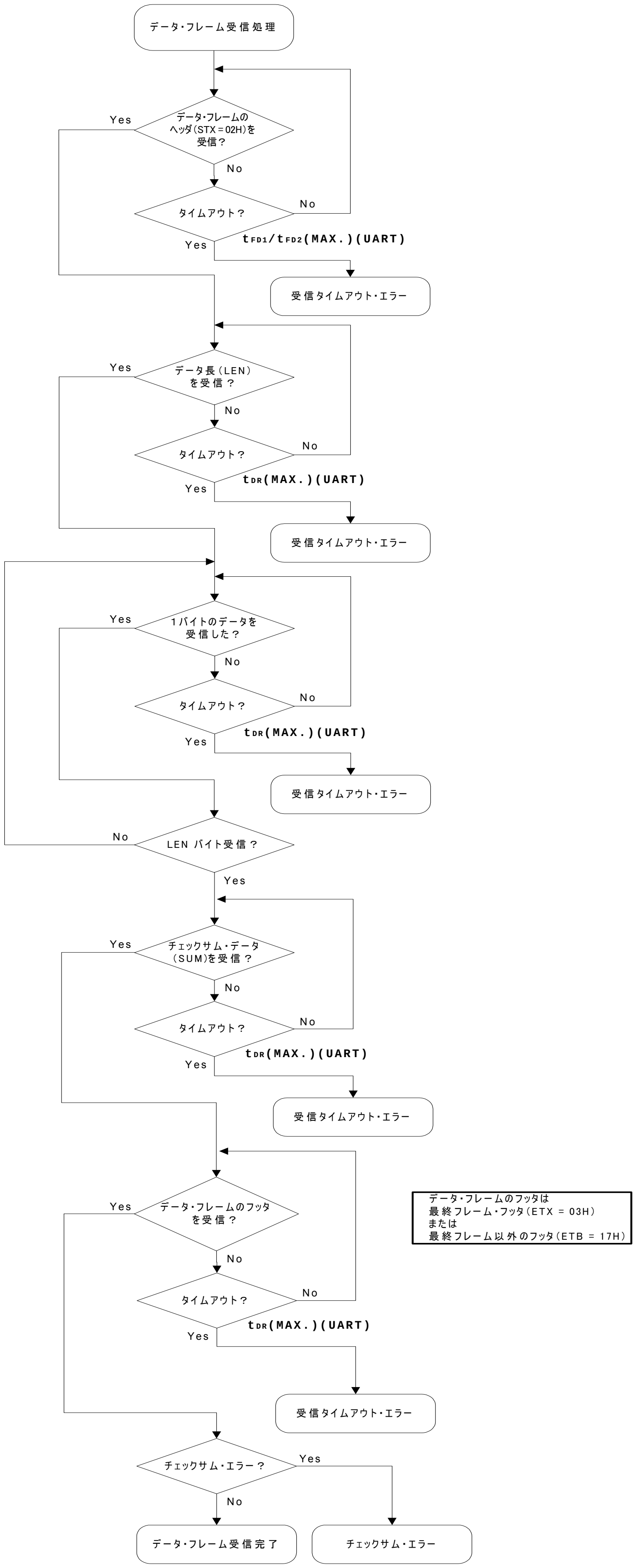
6. 1　コマンド・フレーム送信処理のフロー・チャート



6. 2　データ・フレーム送信処理のフロー・チャート



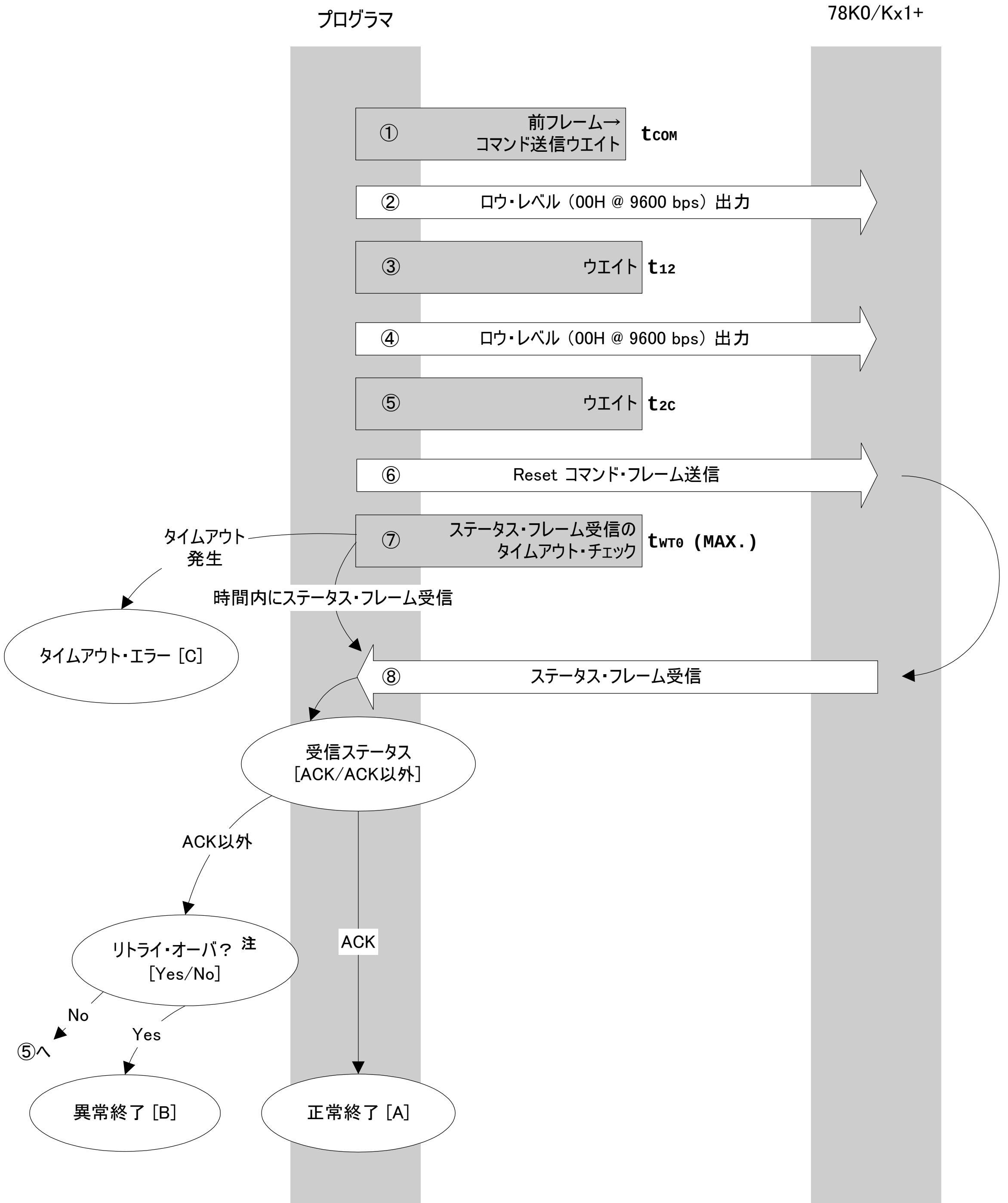
6. 3　データ・フレーム受信処理のフロー・チャート



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

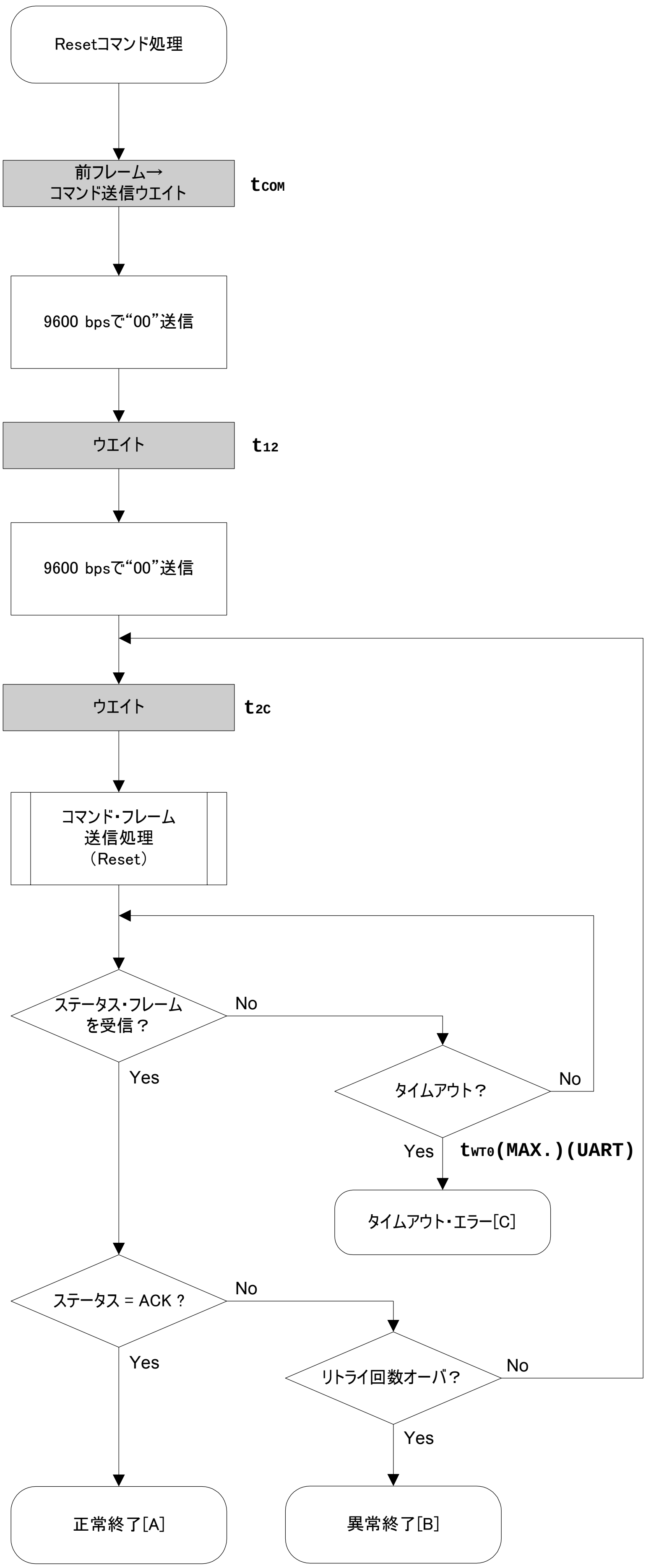
① 直前のフレームからコマンド処理開始前のウエイトをします（ウエイト時間t_{COM}）。
② ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
③ ウエイトします（ウエイト時間t_{12}）。
④ ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
⑤ ウエイトします（ウエイト時間t_{2C}）。
⑥ コマンド・フレーム送信処理にて Resetコマンド を送信します。
⑦ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
　タイムアウトが発生した場合は タイムアウト・エラー [C] となります
　（タイムアウト時間t_{WT0}（MAX.））。
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78K0/Kx1+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.)(UART)
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

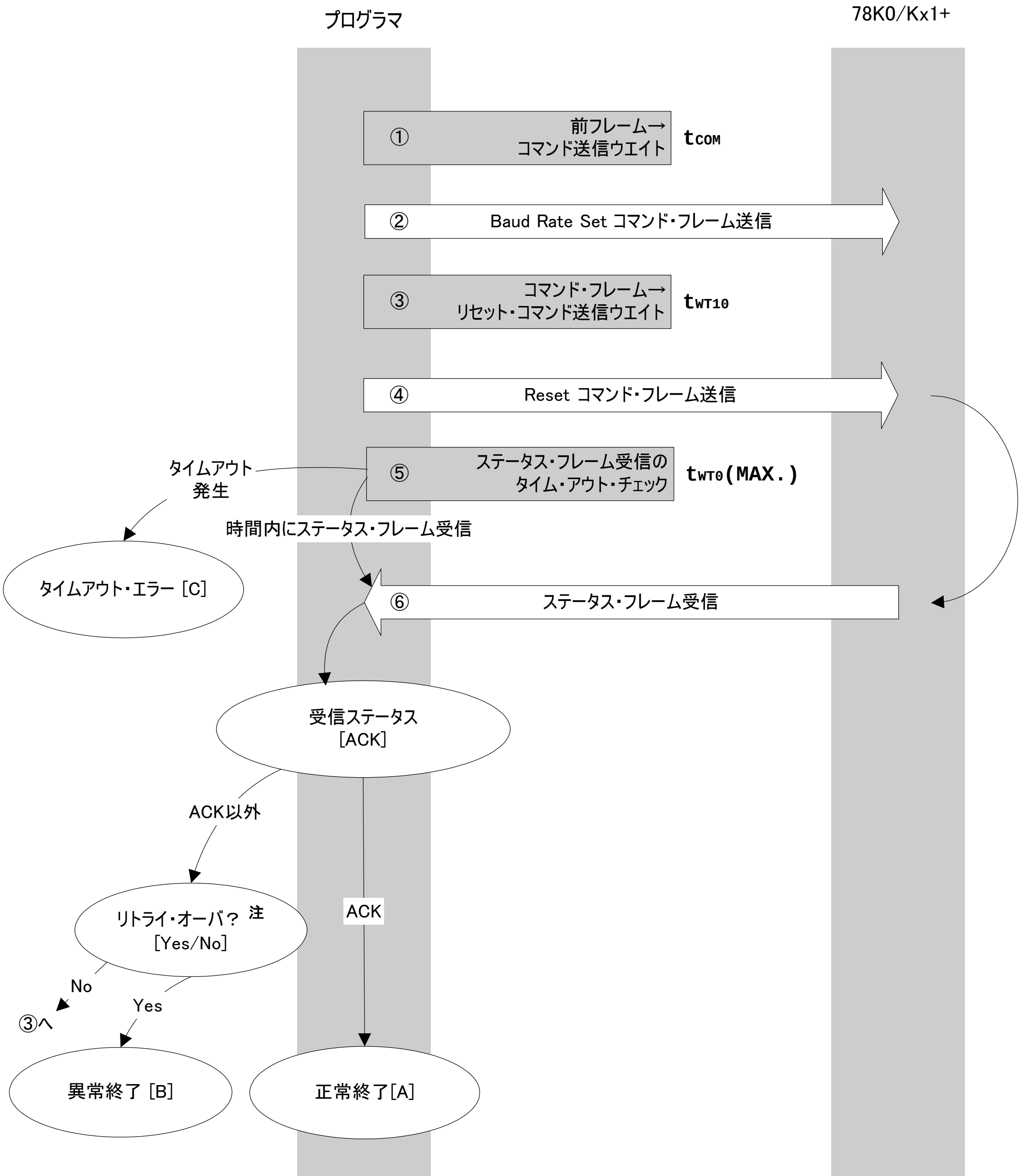
Resetコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* 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
   putc_ua(0x00);          // send 0x00 @ 9600bps

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

   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。

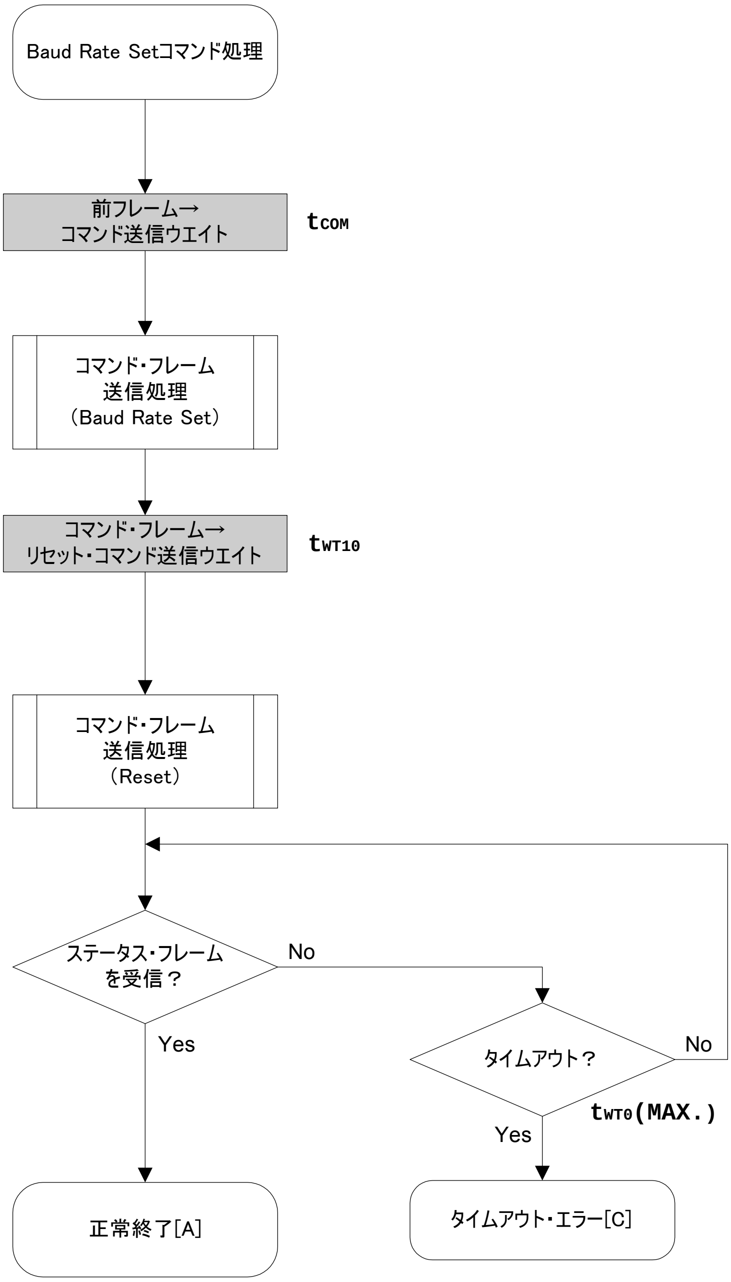
6. 5. 2　処理手順説明

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 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。
⑤ 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.)）。
⑥ 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 78K0/Kx1+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]		－	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 78K0/Kx1+では次の場合もこのエラーになります。 ・コマンド情報（パラメータ）に不正があった場合 ・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

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

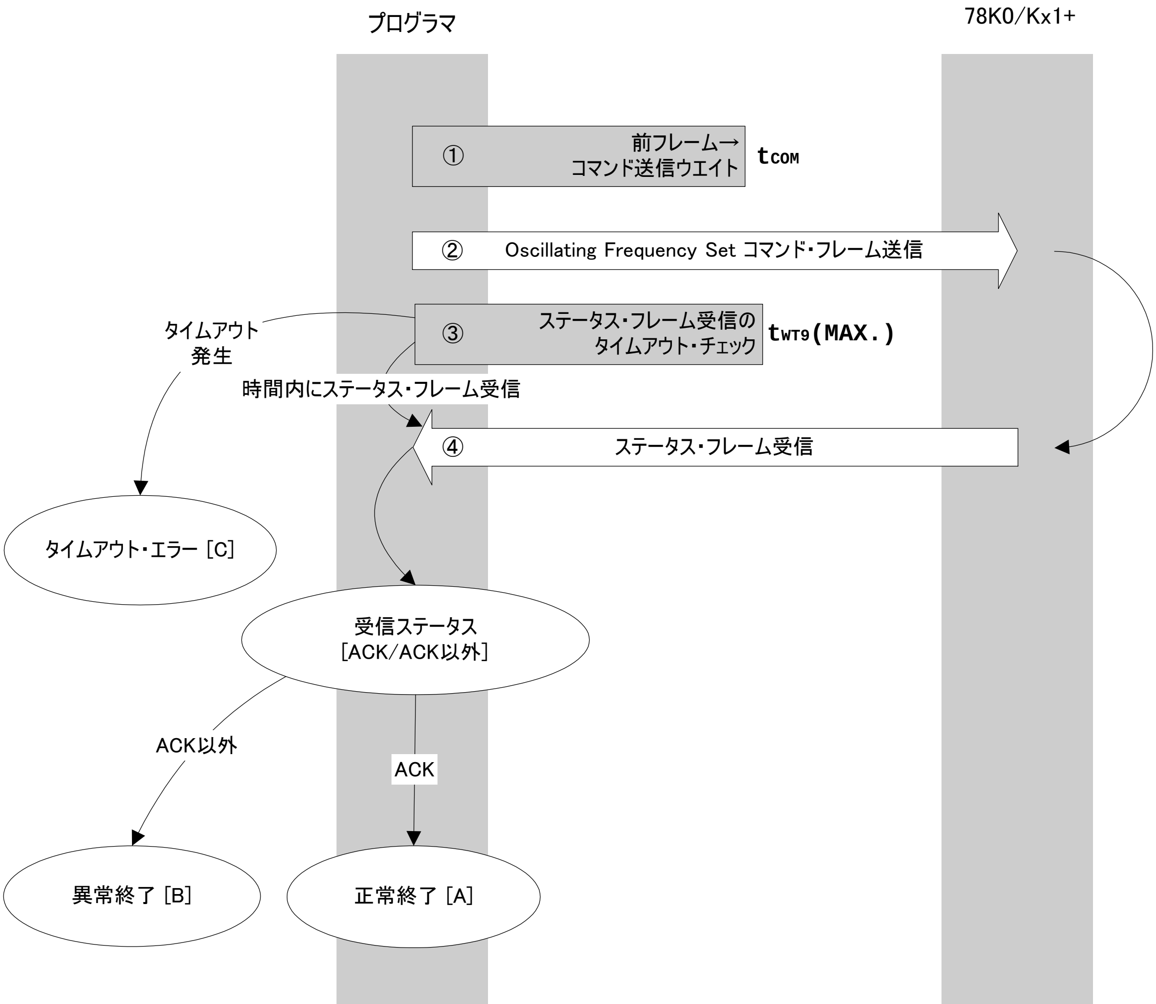
Baud Rate Setコマンド処理のサンプル・プログラムです。

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

   switch(brid){
           default:
           case    BR_9600:        br = 0x03;      break;
           case    BR_19200:       br = 0x04;      break;
           case    BR_31250:       br = 0x05;      break;
           case    BR_38400:       br = 0x06;      break;
           case    BR_76800:       br = 0x07;      break;
           case    BR_153600:      br = 0x08;      break;
   }
   fl_cmd_prm[0] = br;     // "D01"


   fl_wait(tCOM);          // wait before sending command
   put_cmd_ua(FL_COM_SET_BAUDRATE, 2, 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　Oscillating Frequency Setコマンド

6. 6. 1　処理手順チャート

Oscillating Frequency Setコマンド処理手順



6.6.2　処理手順説明

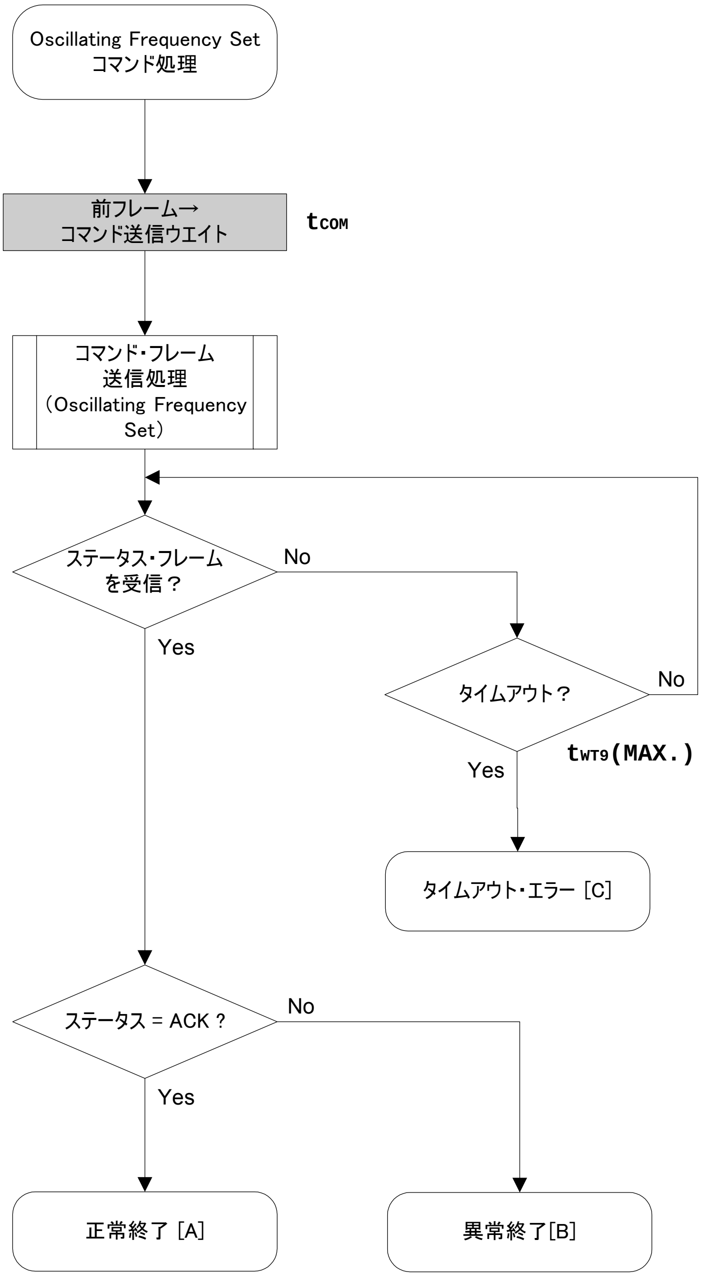
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Oscillating Frequency Setコマンドを送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C]となります（タイムアウト時間t_{WT9}(MAX.)）。
④ ステータス・コードをチェックします。

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，78K0/Kx1+に動作周波数を正しく設定できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	発振周波数値が範囲外です。
	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）。
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


Oscillating Frequency 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
（Oscillating Frequency
Set）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
Yes
tWT9(MAX.)
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

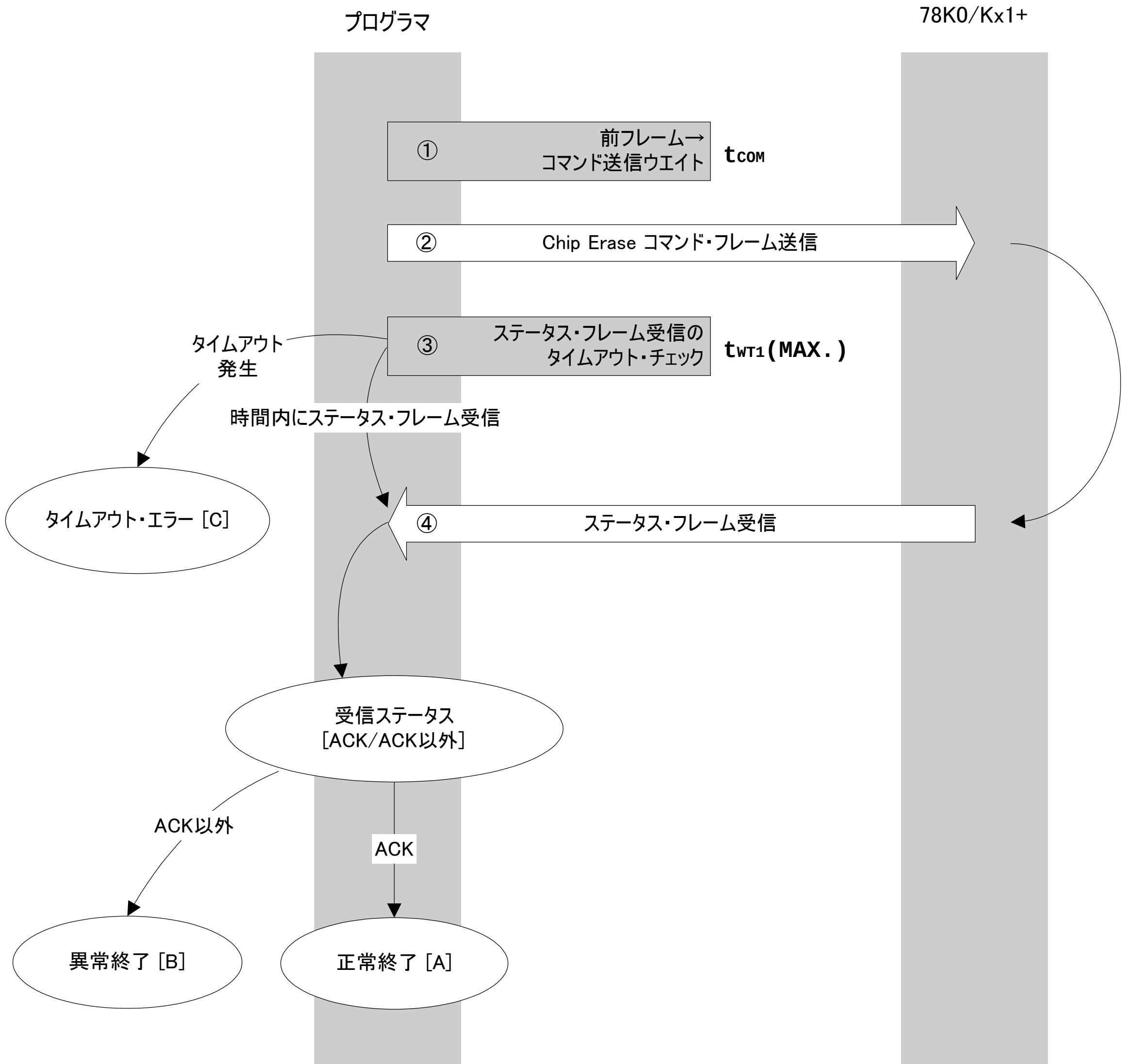
Oscillating Frequency Setコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Set Flash device clock value command                          */
/*                                                               */
/*****************************************************************/
/* [i] u8 clk[4]   ... frequency data(D1-D4)                     */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_ua_setclk(u8 clk[])
{
   u16     rc;

   fl_cmd_prm[0] = clk[0]; // "D01"
   fl_cmd_prm[1] = clk[1]; // "D02"
   fl_cmd_prm[2] = clk[2]; // "D03"
   fl_cmd_prm[3] = clk[3]; // "D04"


   fl_wait(tCOM);          // wait before sending command
   put_cmd_ua(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);

   rc = get_sfrm_ua(fl_ua_sfrm, tWT9_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　Chip Eraseコマンド

6. 7. 1　処理手順チャート

Chip Eraseコマンド処理手順



6. 7. 2　処理手順説明

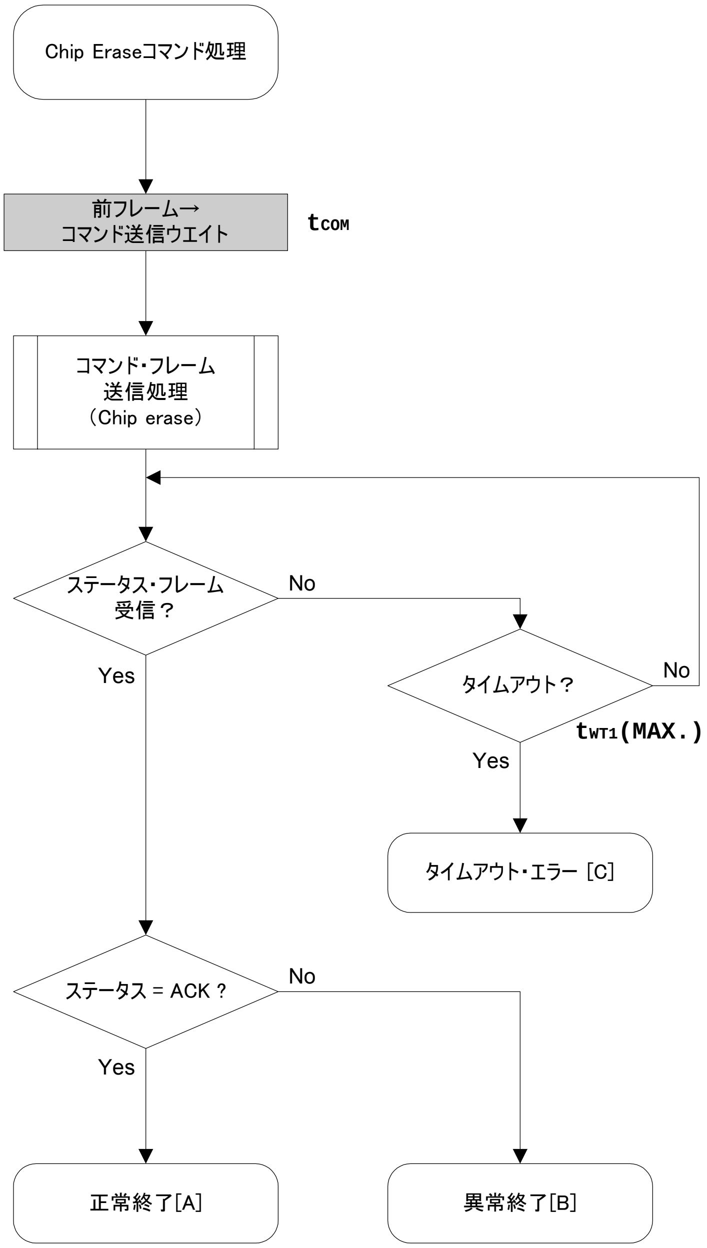
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Chip Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は， タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT1}(MAX.)）。
④ ステータス・コードをチェックします。

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. 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，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」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）。
	消去エラー	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



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. 7. 5　サンプル・プログラム

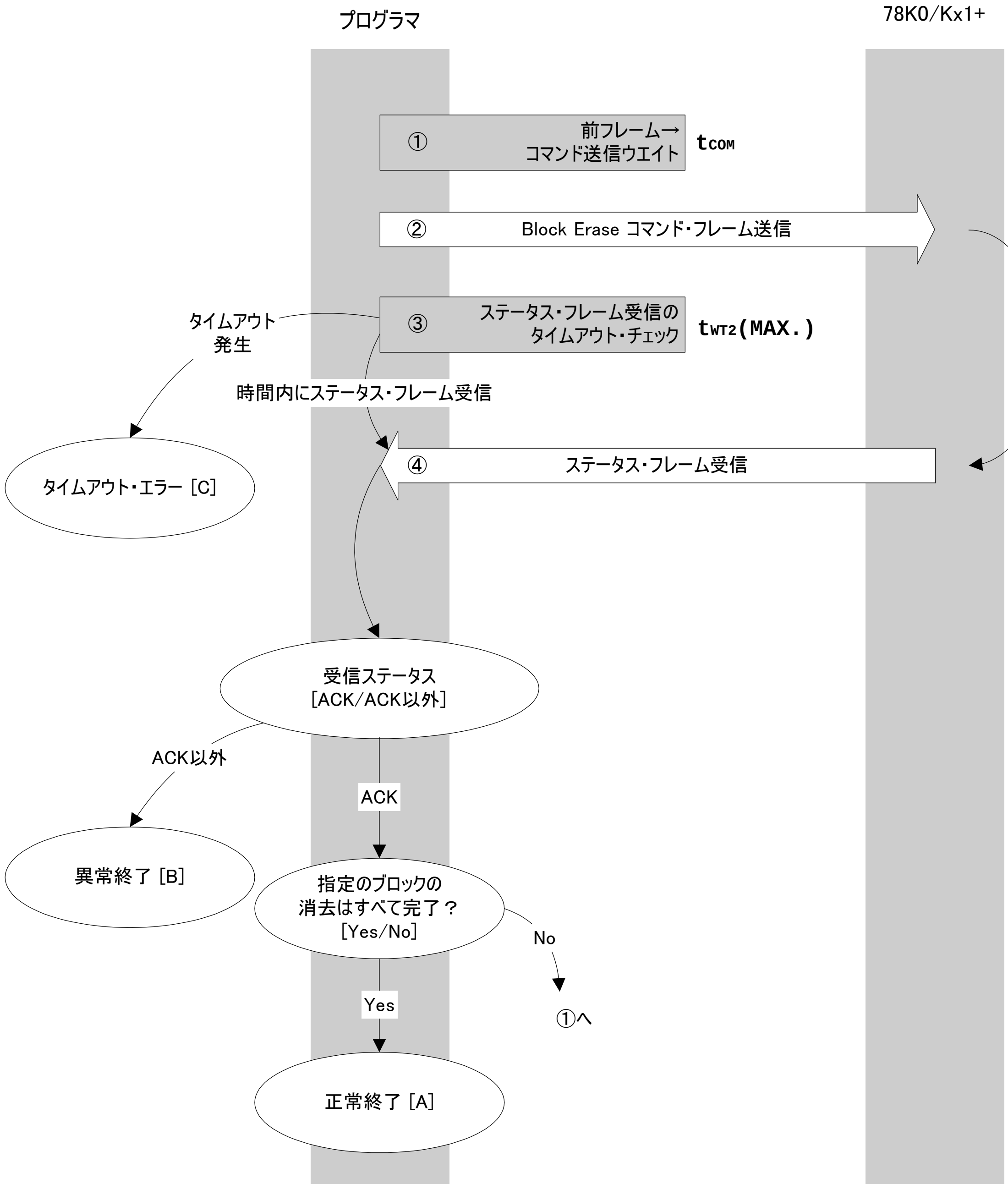
Chip Eraseコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* 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. 8　Block Eraseコマンド

6. 8. 1　処理手順チャート

Block Eraseコマンド処理手順



6. 8. 2　処理手順説明

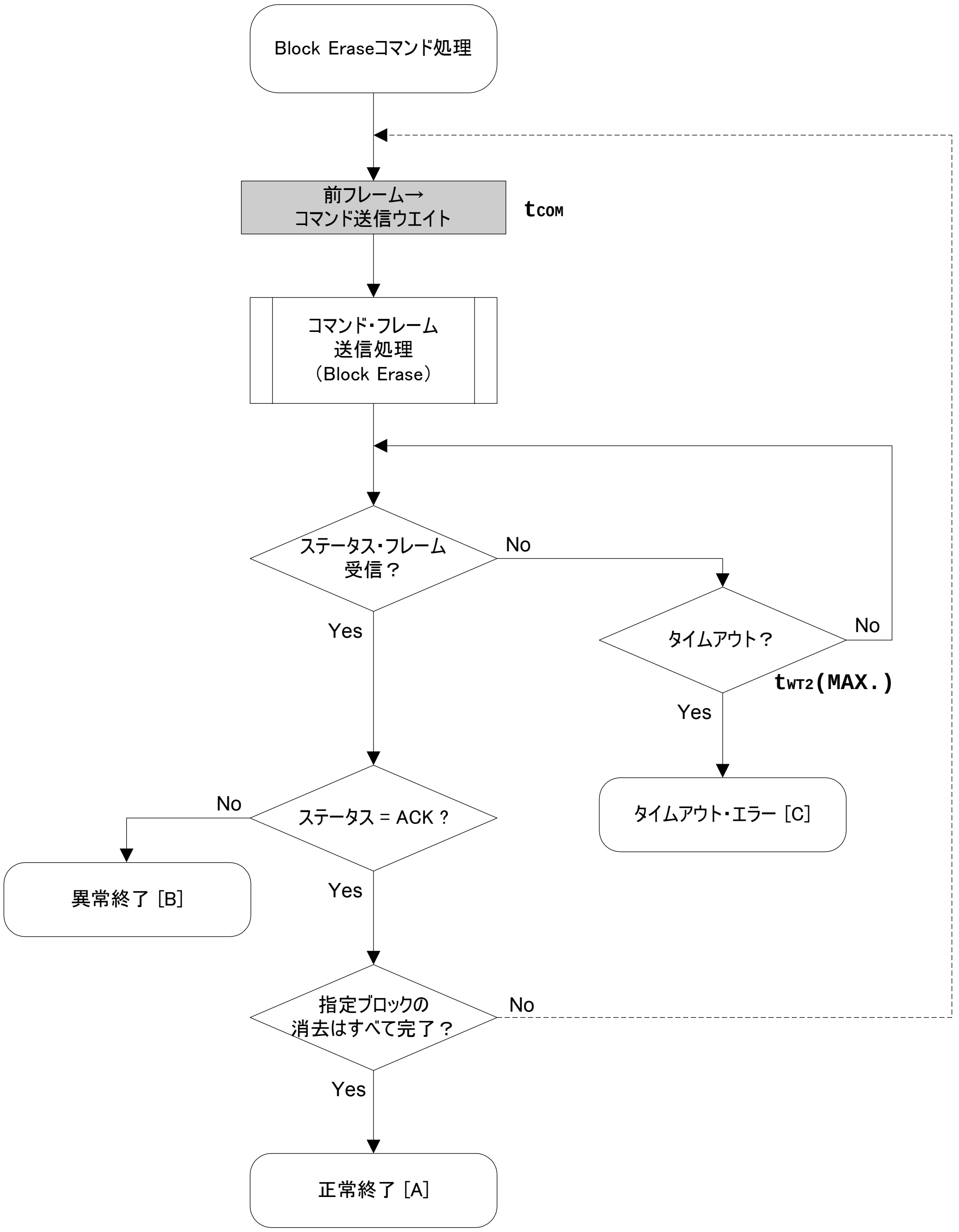
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Block Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C] となります（タイムアウト時間t_{WT2}(MAX.)）。
④ ステータス・コードをチェックします。

ST1 = ACKの場合　：指定したブロックの消去がすべて完了していない場合は，ブロック番号を変えて①より再実行します。
指定したすべてのブロックの消去が完了した場合は，正常終了[A] です。
ST1 = ACK以外の場合 ： 異常終了[B] 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，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	ブロック番号が範囲外です。
	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。
	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」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）。
	消去エラー	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. 8. 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
タイムアウト？
No
tWT2(MAX.)
Yes
タイムアウト・エラー [C]
Yes
No
ステータス = ACK ?
異常終了 [B]
Yes
指定ブロックの
消去はすべて完了？
No
Yes
正常終了 [A]

6. 8. 5　サンプル・プログラム

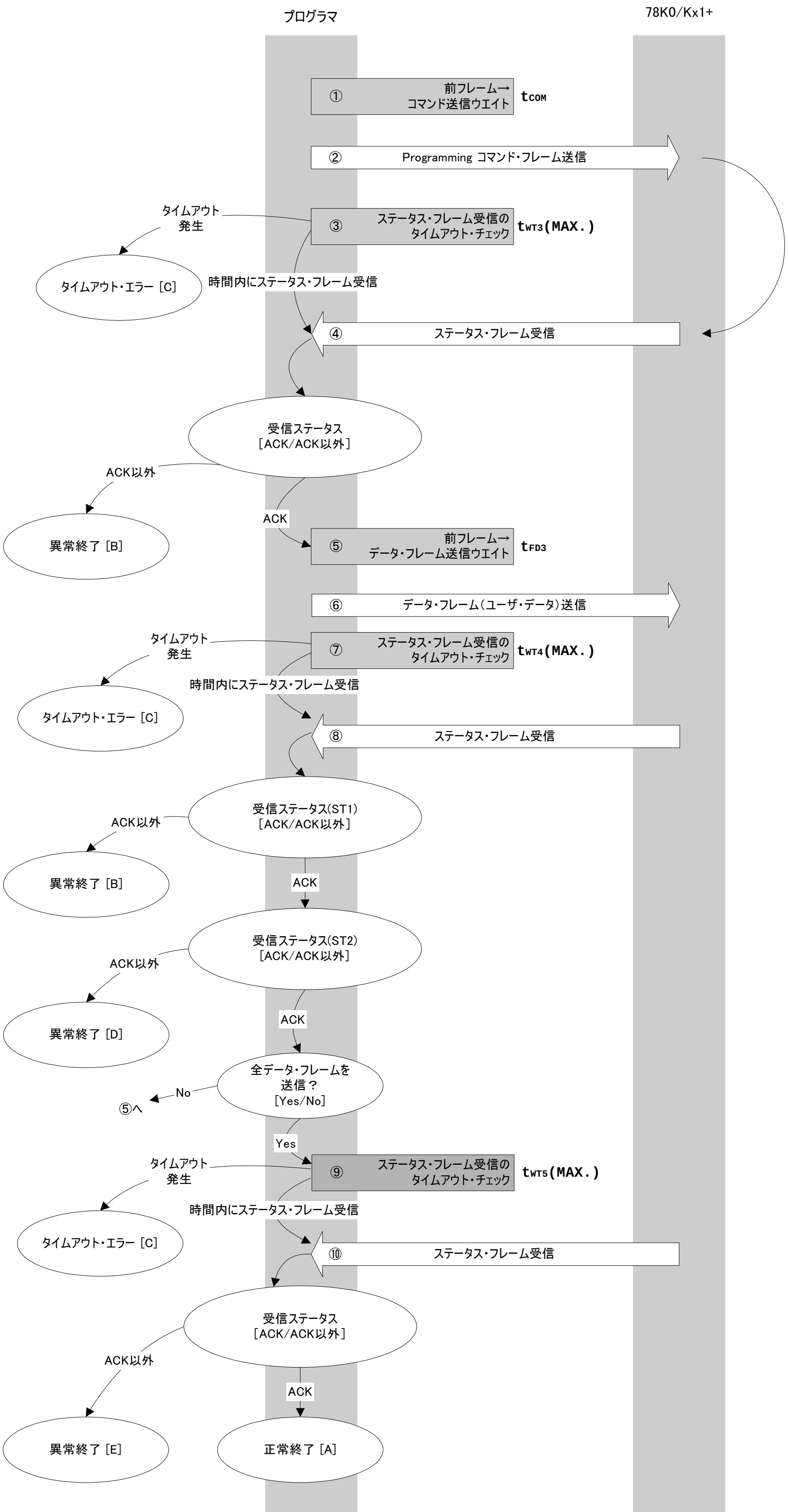
1ブロック分のBlock Eraseコマンド処理のサンプル・プログラムです。

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

   fl_cmd_prm[0] = block;  // BLK


   fl_wait(tCOM);          // wait before sending command

   put_cmd_ua(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm); // send ERASE CHIP command

   rc = get_sfrm_ua(fl_ua_sfrm, tWT2_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. 9　Programmingコマンド

6. 9. 1　処理手順チャート

Programmingコマンド処理手順



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，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。

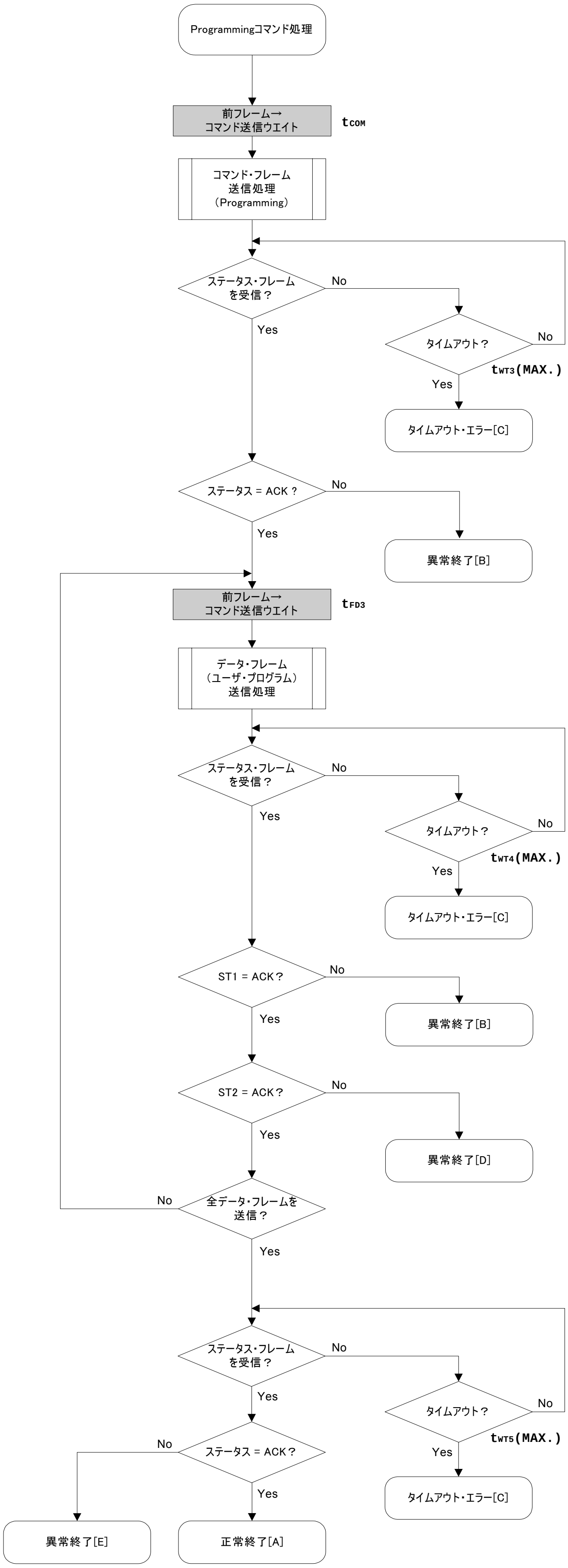
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合は タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT5}(MAX.)）。
⑩ ステータス・コードをチェックします。

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. 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，8 の倍数ではありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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。
	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]	FLMD エラー	18H（ST2）	書き込みエラーが発生しました。
	Write エラー	1CH（ST2）	書き込みエラーが発生しました。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

6. 9. 4　フロー・チャート


Programmingコマンド処理
前フレーム→
コマンド送信ウエイト
t_{COM}
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
t_{WT3}(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
t_{FD3}
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
t_{WT4}(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]
全データ・フレームを
送信？
No
Yes
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
t_{WT5}(MAX.)
Yes
タイムアウト・エラー[C]
ステータス = ACK？
No
Yes
異常終了[E]
正常終了[A]

6. 9. 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。

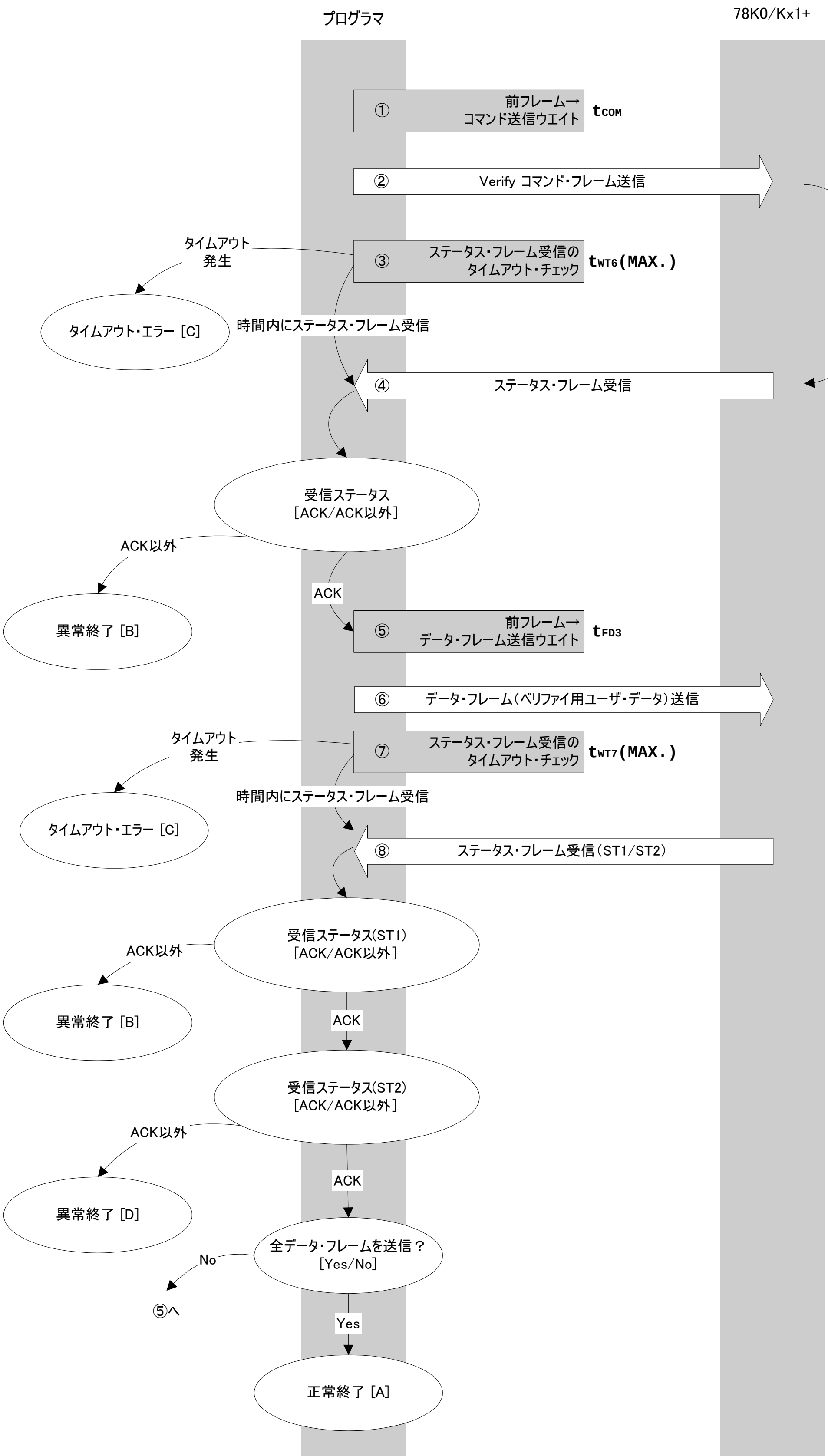
```
/*****************************************************************/
/*                                                               */
/* Write command                                                 */
/*                                                               */
/*****************************************************************/
/* [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;

      /************************************************/
      /*   set params                                 */
      /************************************************/
      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

      /************************************************/
      /*   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. 10　Verifyコマンド

6. 10. 1　処理手順チャート

Verify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。

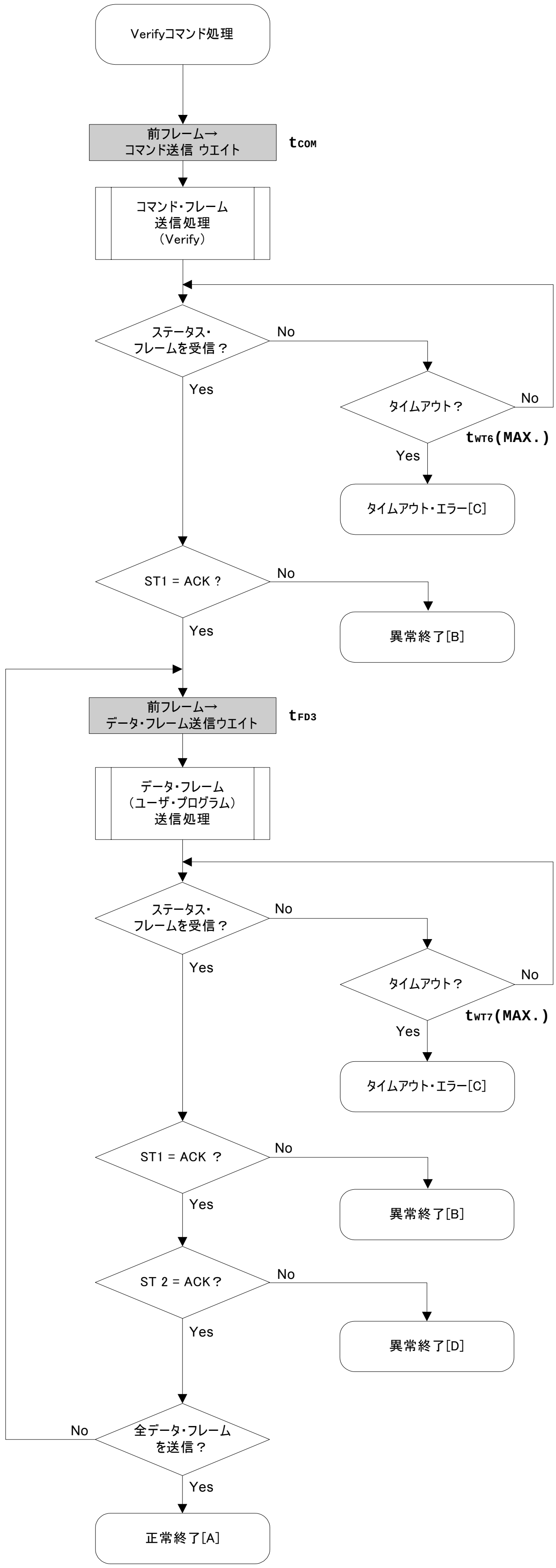
⑤ 直前のフレームからデータ・フレーム送信までのウエイトをします（ウエイト時間t_{FD3}）。
⑥ データ・フレーム送信処理により，ベリファイ用ユーザ・データを送信します。
⑦ ユーザ・データ送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT7}(MAX.)）。
⑧ ステータス・コード（ST1/ST2）をチェックします（処理手順チャートやフロー・チャートも参照してください）。

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. 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，ベリファイ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。2 Kバイトごとの固定アドレスでありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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）	ベリファイに失敗しました。または，その他のエラーが発生しました。

6. 10. 4　フロー・チャート



6. 10. 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。

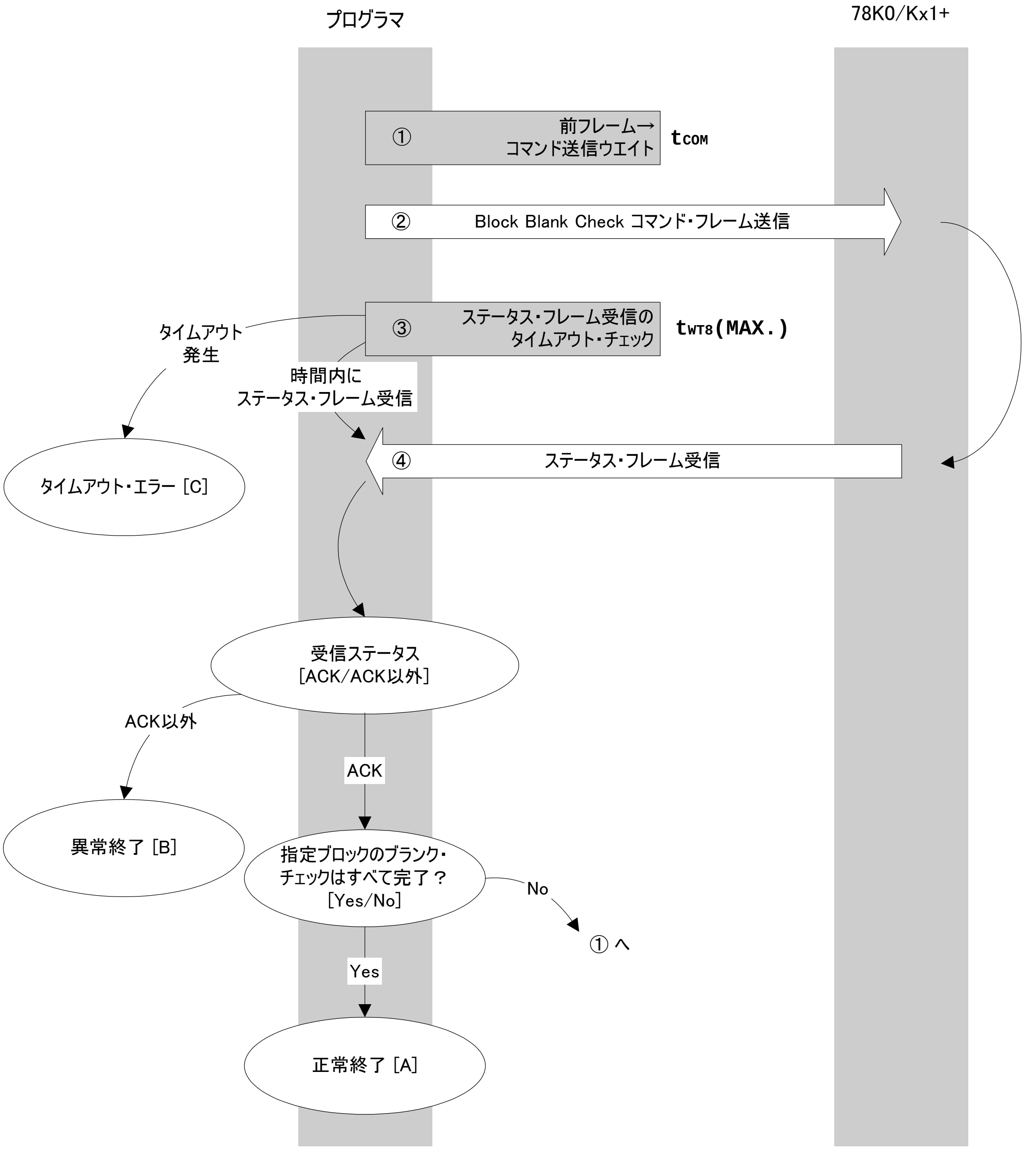
```
/*******************************************************************/
/*                                                                 */
/* Verify command                                                  */
/*                                                                 */
/*******************************************************************/
/* [i] u32 top     ... start address                               */
/* [i] u32 bottom  ... end address                                 */
/* [r] u16         ... error code                                  */
/*******************************************************************/
u16  fl_ua_verify(u32 top, u32 bottom)
{
   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, rom_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. 11　Block Blank Checkコマンド

6. 11. 1　処理手順チャート

Block Blank Check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_{WT8}(MAX.)）。
④ ステータス・コードをチェックします。

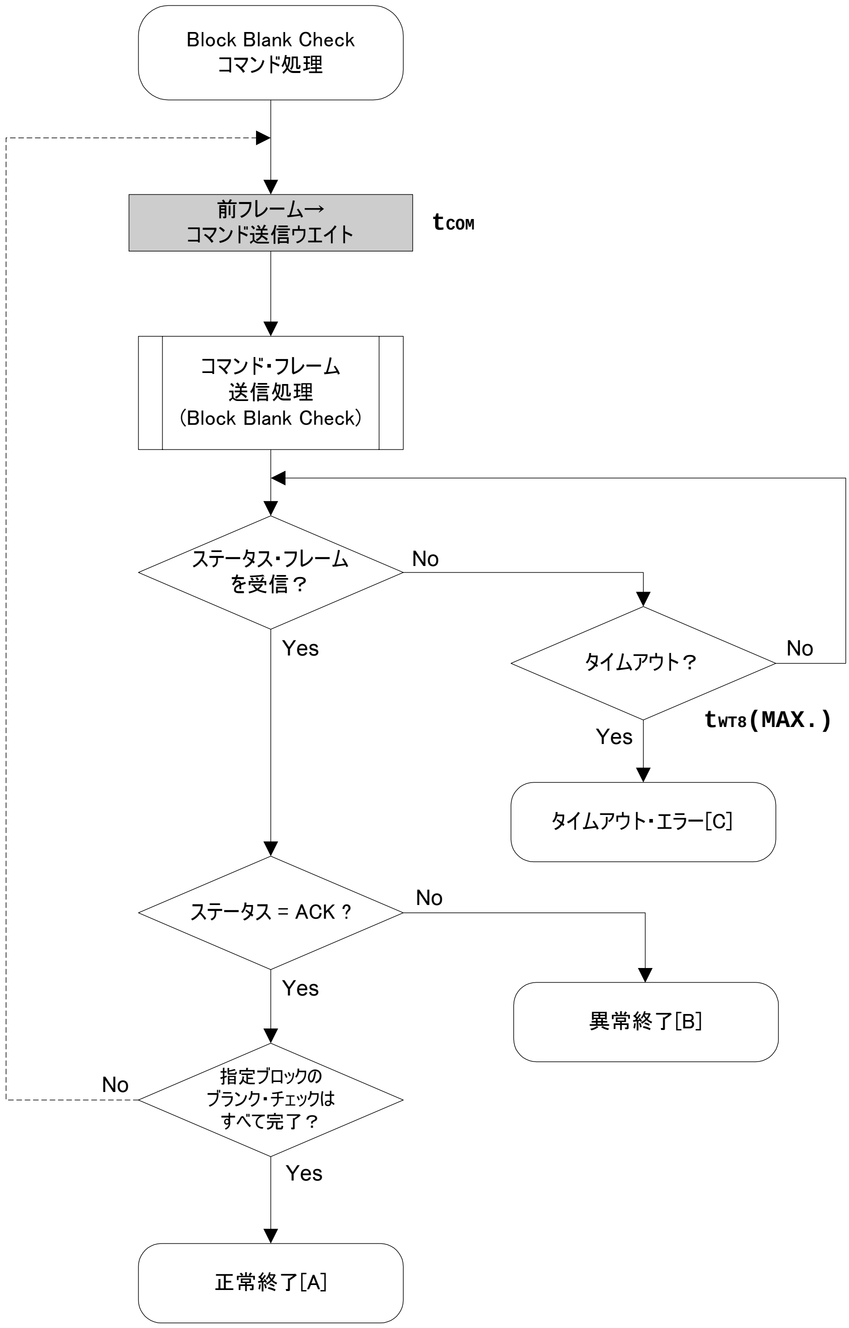
ST1 = ACKの場合　：指定したブロックのブランク・チェックがすべて完了していない場合は，ブロック番号を変えて①より再実行します。
指定したすべてのブロックのブランク・チェックが完了した場合は， 正常終了[A] です。

ST1 = ACK以外の場合　： 異常終了[B] 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，指定したブロックすべてがブランクであることを示します。
異常終了 [B]	パラメータ・エラー	05H	ブロック番号が範囲外です。
	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・データが異常の場合（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]		－	ステータス・フレーム受信のタイムアウト・エラーが発生しました。

6. 11. 4　フロー・チャート



6. 11. 5　サンプル・プログラム

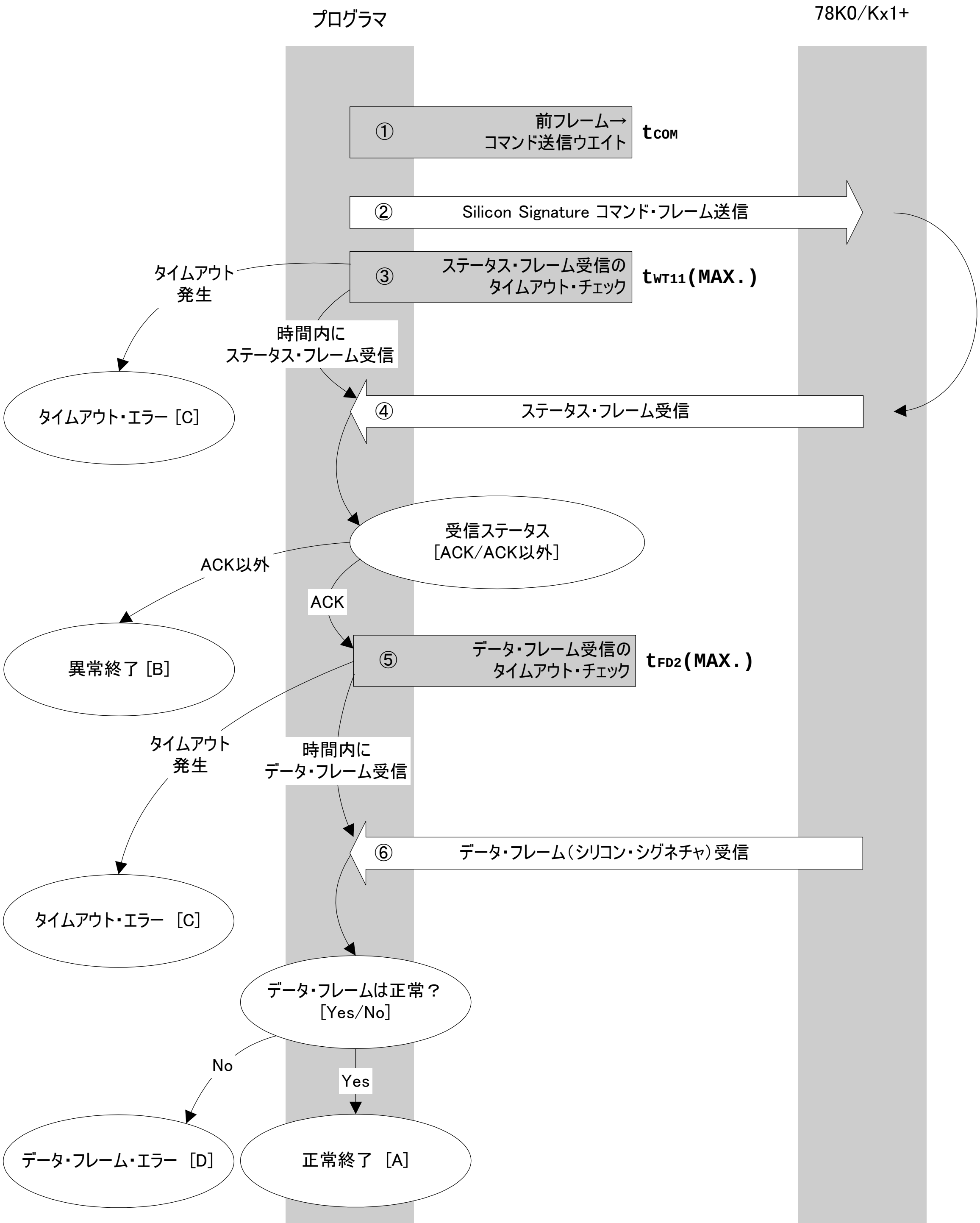
1ブロック分のBlock Blank Checkコマンド処理のサンプル・プログラムです。

```
/****************************************************************/
/*                                                              */
/* Block blank check command                                    */
/*                                                              */
/****************************************************************/
/* [i]  u8 block   ... block number                             */
/* [r] u16         ... error code                               */
/****************************************************************/
u16        fl_ua_blk_blank_chk(u8 block)
{
   u16     rc;

   fl_cmd_prm[0] = block;  // "BLK"

   fl_wait(tCOM);          // wait before sending command

   put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);


   rc = get_sfrm_ua(fl_ua_sfrm, tWT8_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. 12　Silicon Signatureコマンド

6. 12. 1　処理手順チャート

Silicon Signature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，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。

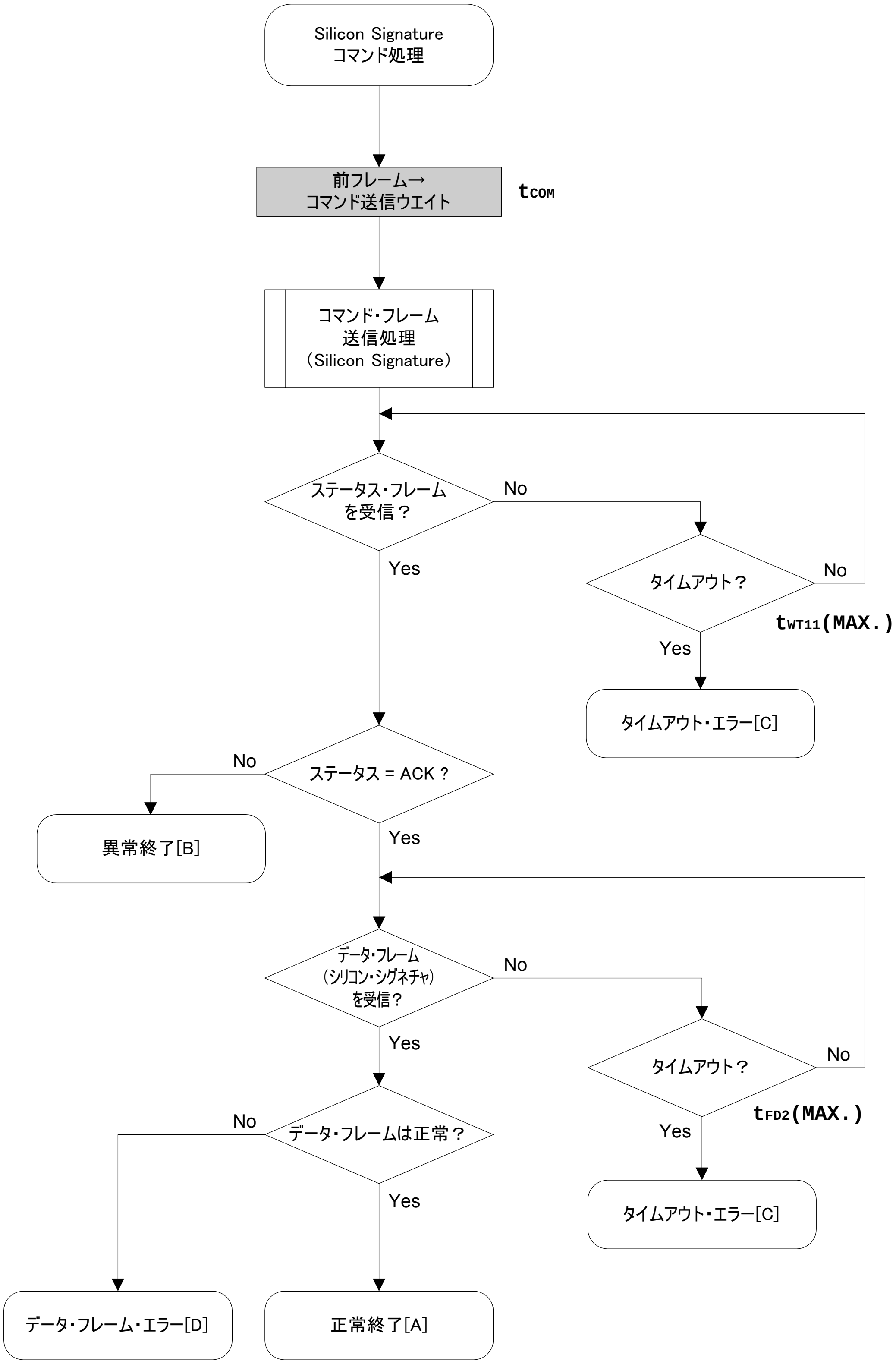
⑤ データ・フレーム（シリコン・シグネチャ・データ）受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}(MAX.)）。
⑥ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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　　：異常終了[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，シリコン・シグネチャ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。
異常終了 [D]	データ・フレーム・エラー	－	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. 12. 4　フロー・チャート



6. 12. 5　サンプル・プログラム

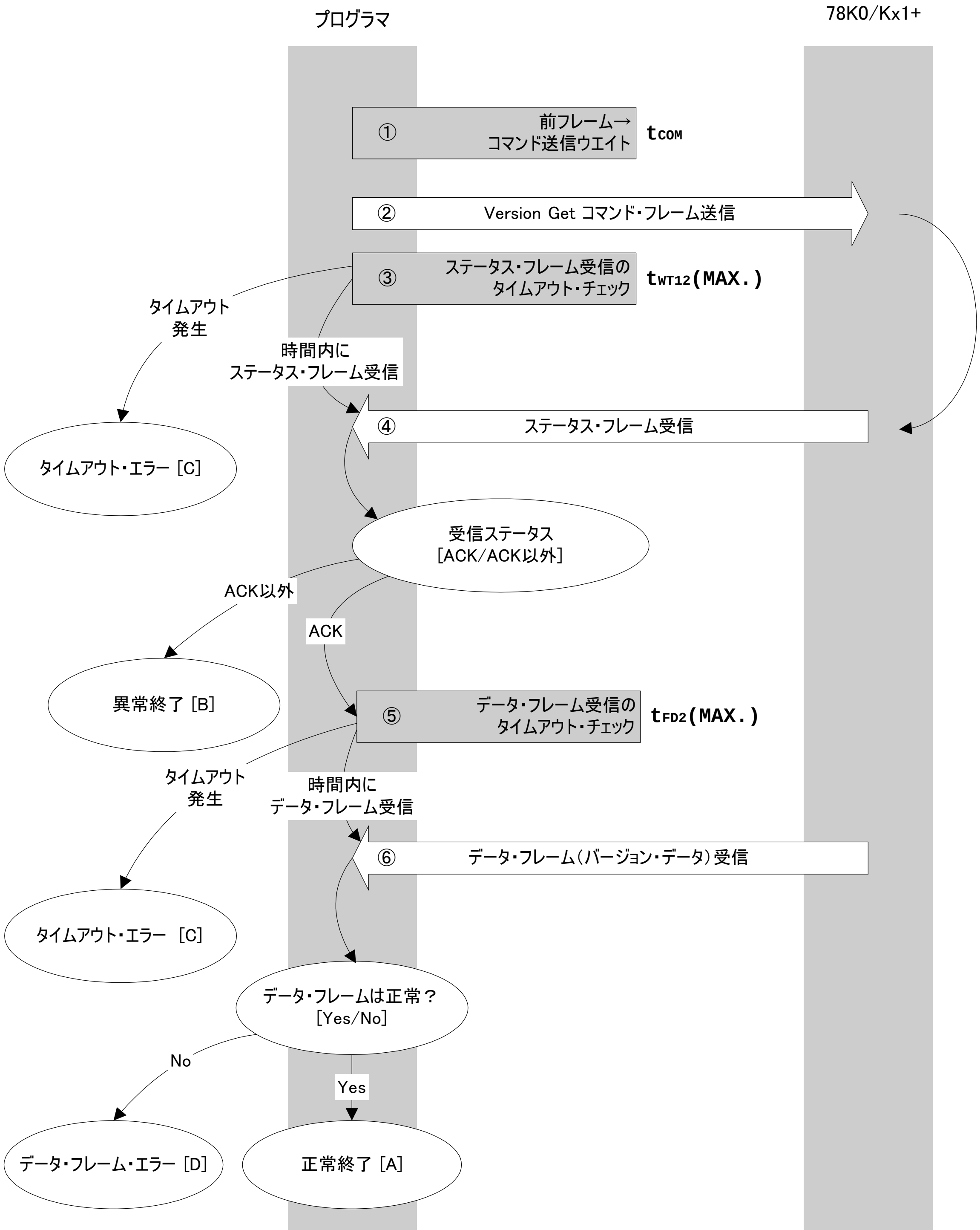
Silicon Signatureコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* 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. 13　Version Getコマンド

6. 13. 1　処理手順チャート

Version Get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，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。

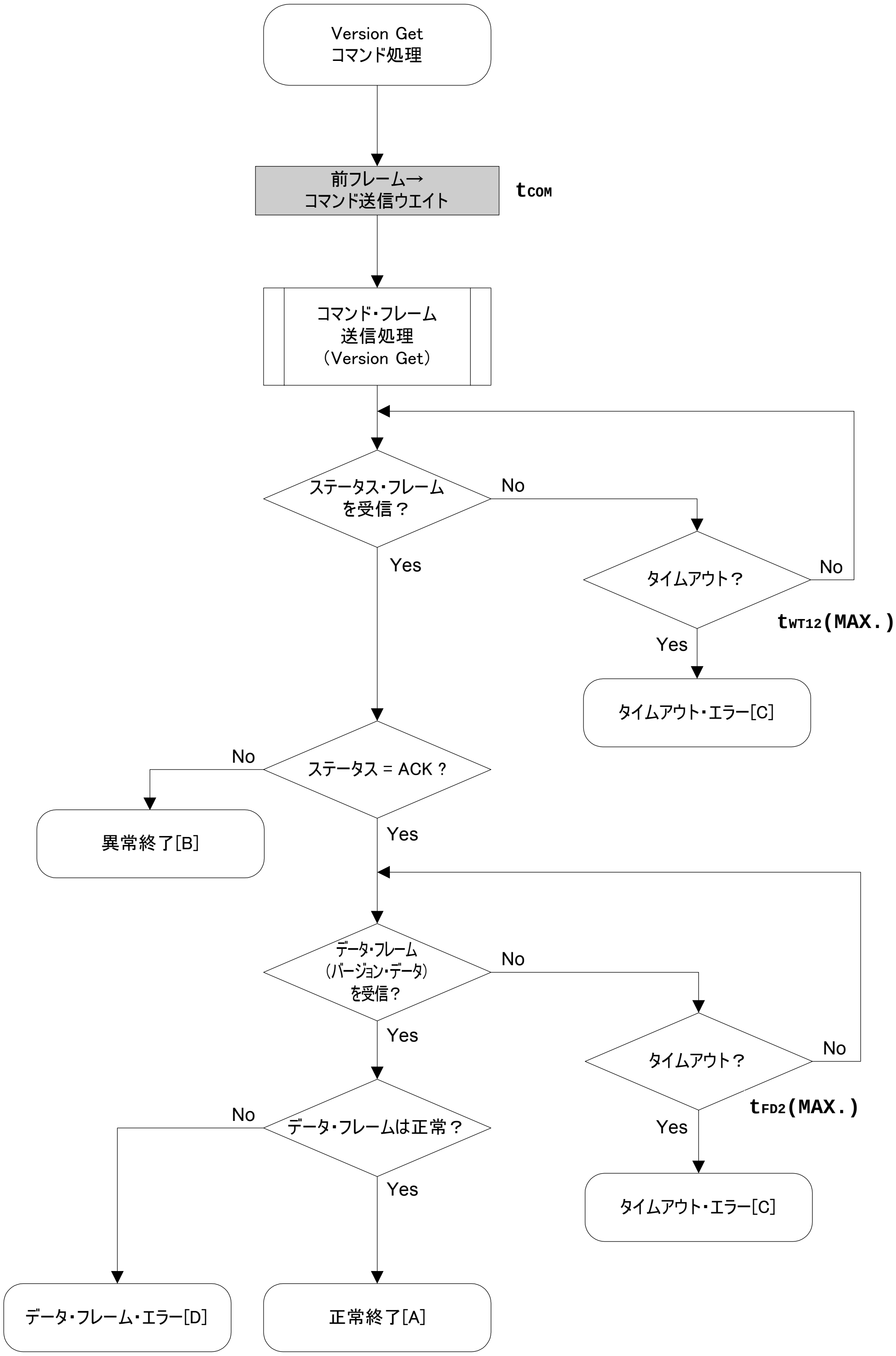
⑤ データ・フレーム（バージョン・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}(MAX.)）。
⑥ 受信したデータ・フレーム（バージョン・データ）をチェックします。

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　　：異常終了[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，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。
異常終了 [D]	データ・フレーム・エラー	－	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. 13. 4　フロー・チャート



6. 13. 5　サンプル・プログラム

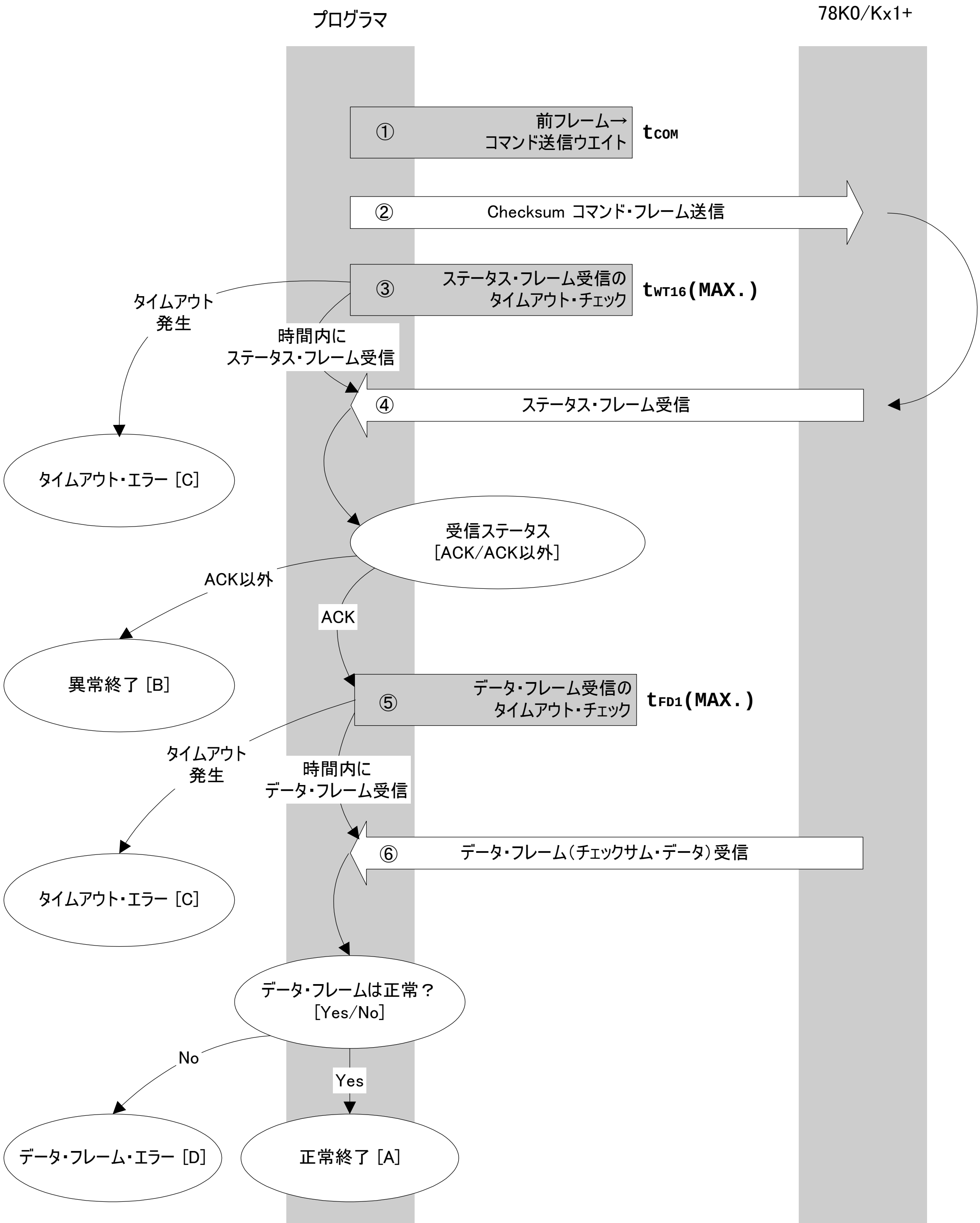
Version Getコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* 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. 14　Checksumコマンド

6. 14. 1　処理手順チャート

Checksumコマンド処理手順



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，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。

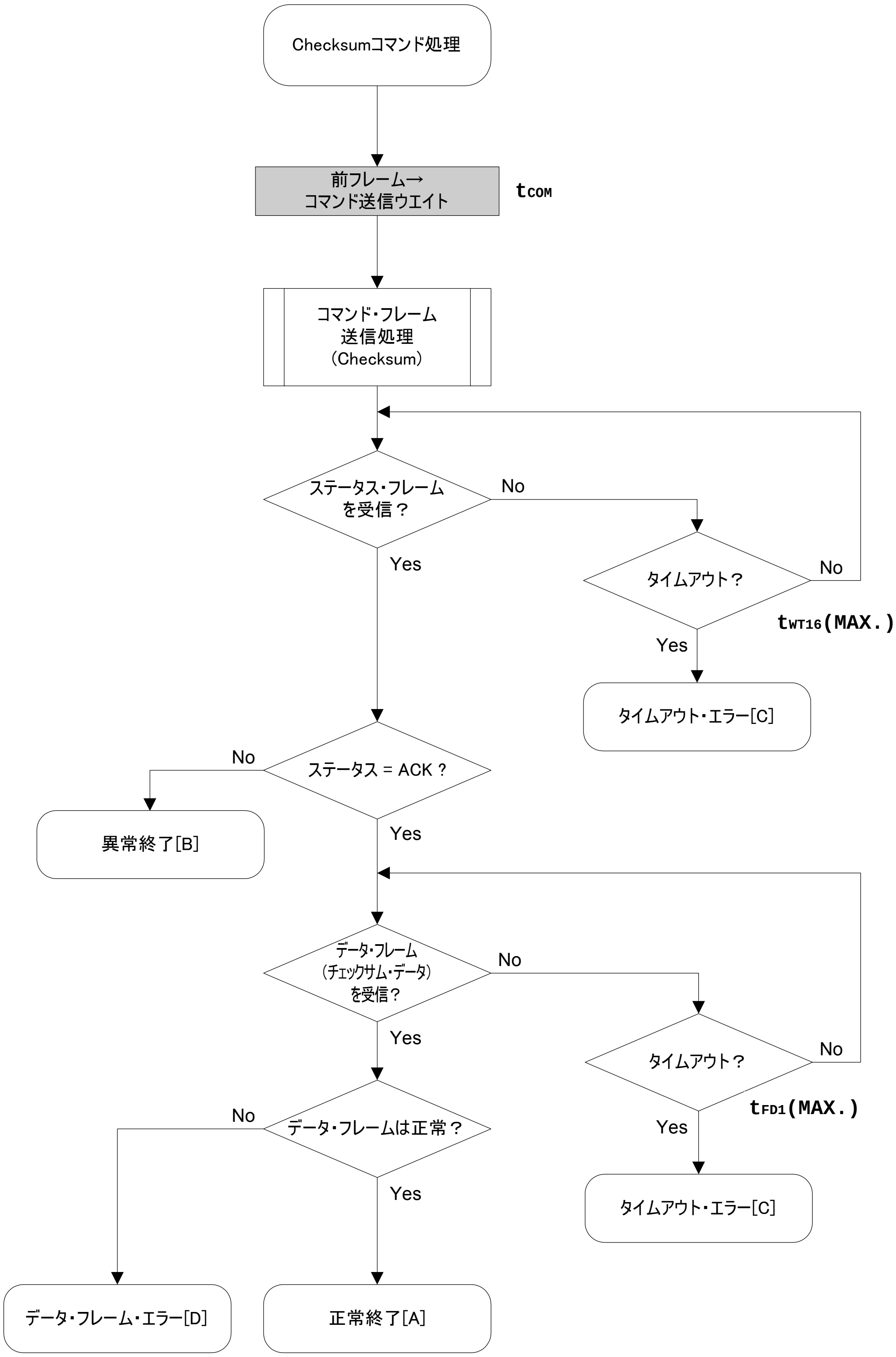
⑤ データ・フレーム（チェックサム・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD1}(MAX.)）。
⑥ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

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　：異常終了[D]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，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，2 K バイトごとの固定アドレスでありません。
	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。
異常終了 [D]	データ・フレーム・エラー	－	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. 14. 4　フロー・チャート



6. 14. 5　サンプル・プログラム

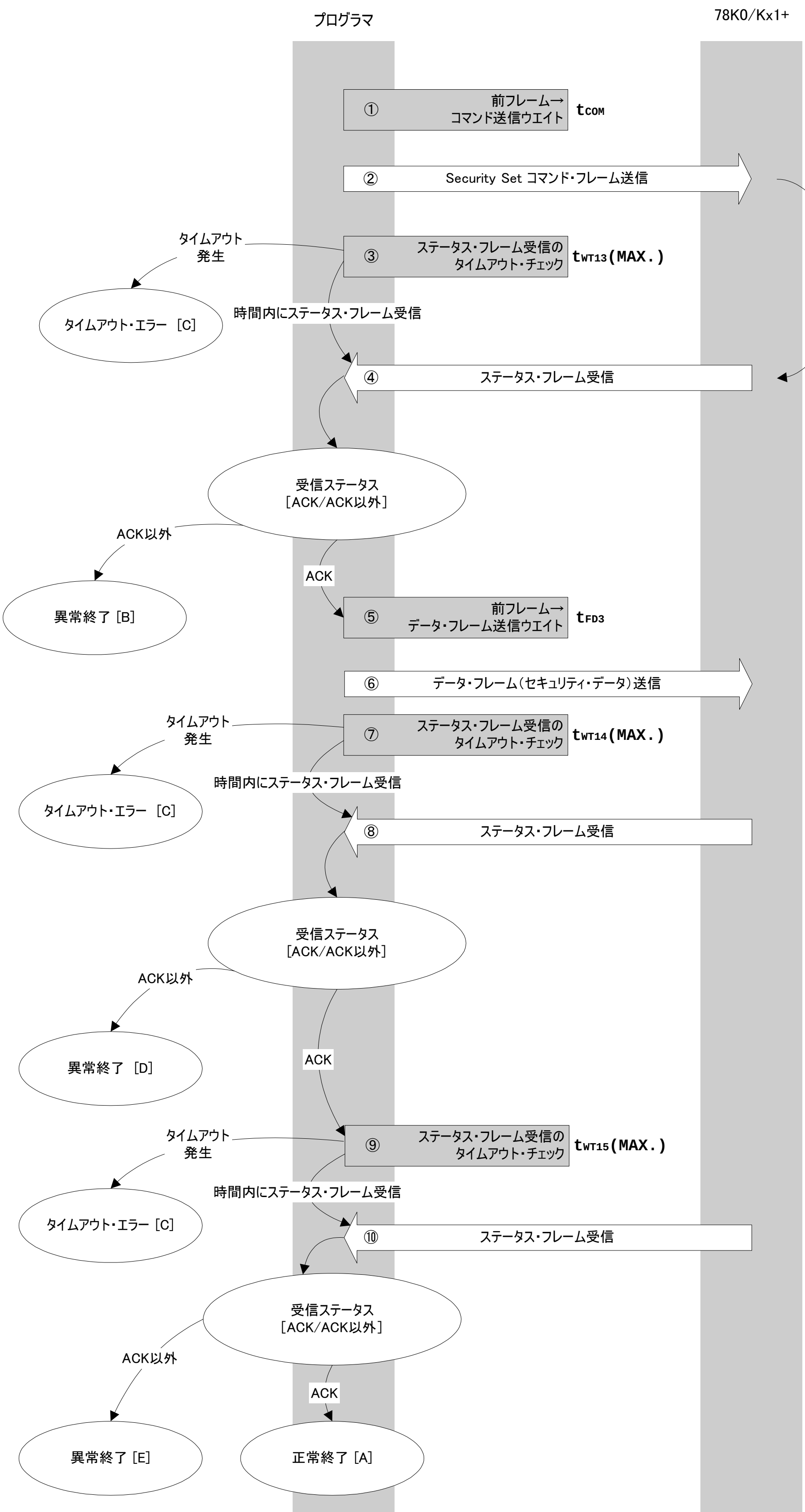
Checksumコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                          */
/* 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. 15　Security Setコマンド

6. 15. 1　処理手順チャート

Security Setコマンド処理手順



6. 15. 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。

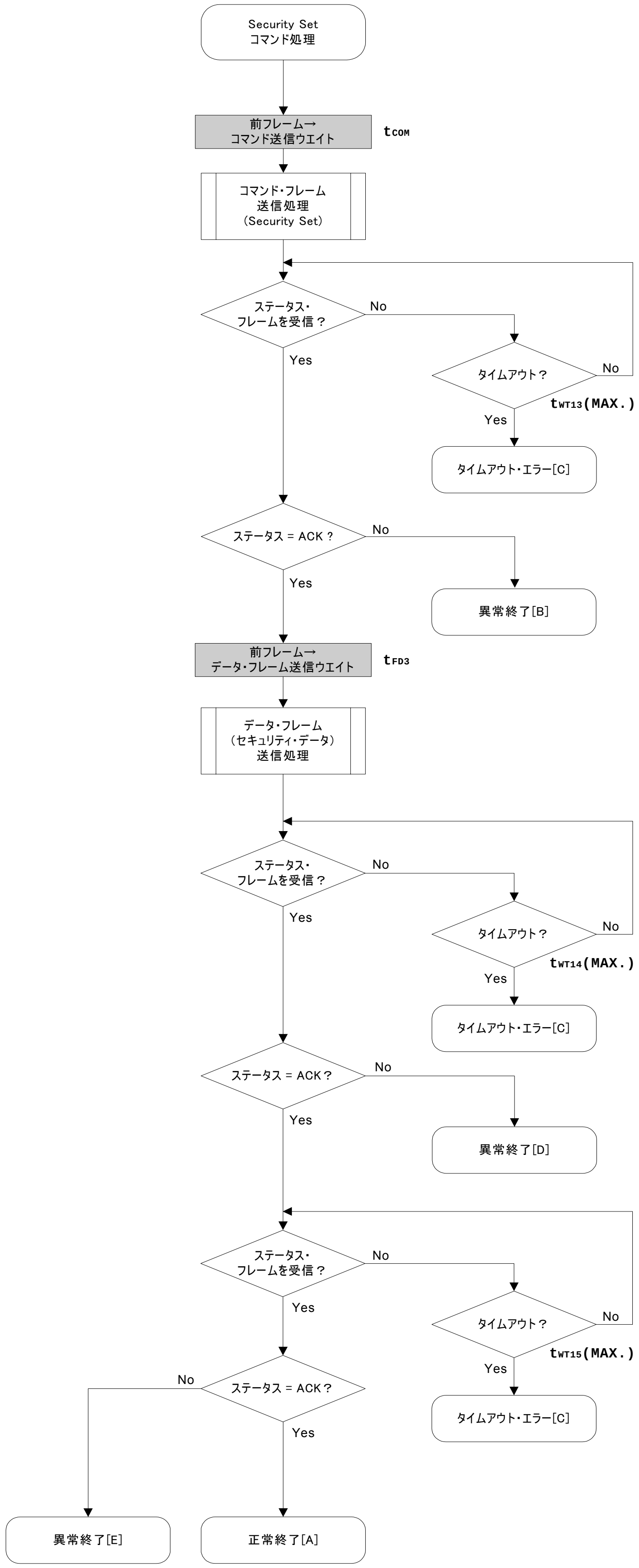
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT15}(MAX.)）。
⑩ ステータス・コードをチェックします。

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. 15. 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	BLK または PAG が 00H でありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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]		−	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]	FLMD エラー	18H	書き込みエラーが発生しました。
	Write エラー	1CH	書き込みエラーが発生しました（すでにセキュリティ・データが設定されている場合も含む）。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

6. 15. 4　フロー・チャート



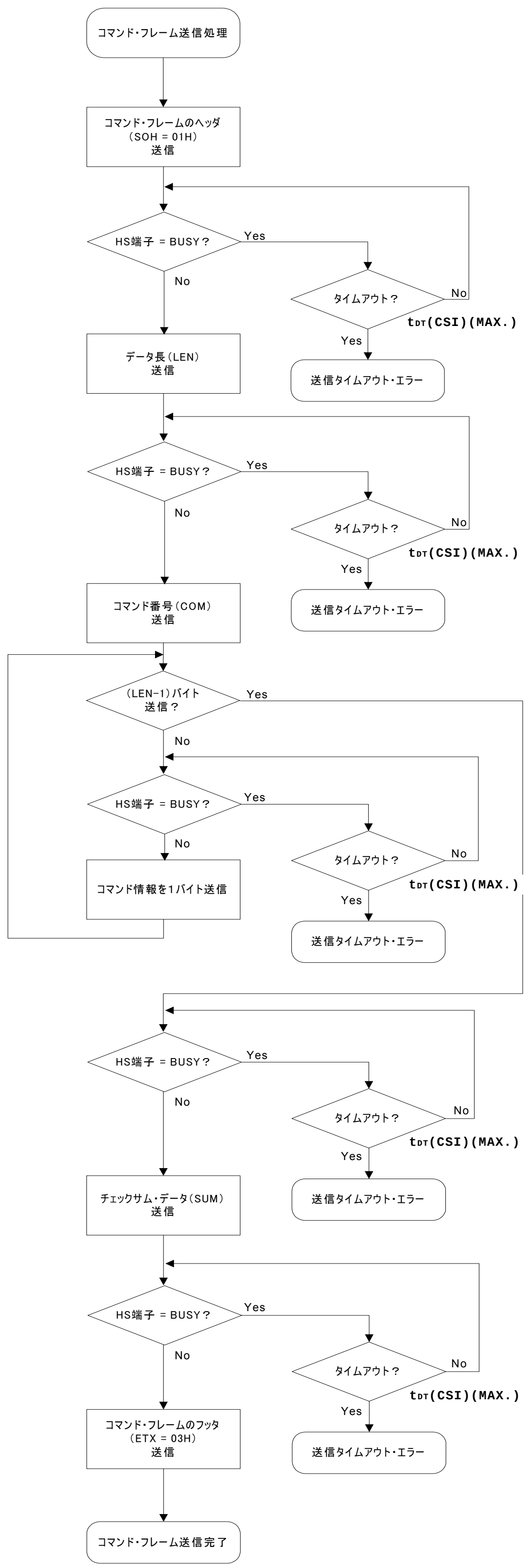
6. 15. 5　サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

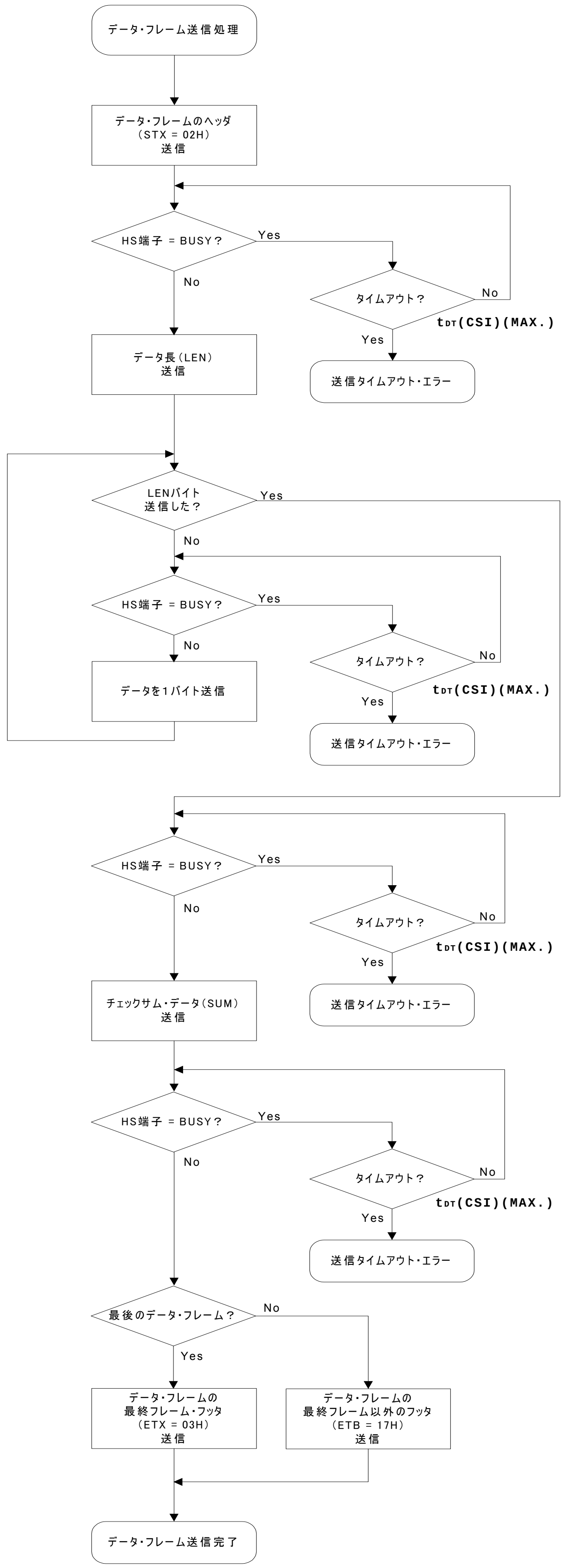
```
/******************************************************************/
/*                                                                */
/* Set security flag command                                      */
/*                                                                */
/******************************************************************/
/* [i] u8 scf      ... Security flag data                         */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16         fl_ua_setscf(u8 scf)
{
   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|= 0b11111000;
                                      // "FLG" (upper 5bits must be '1' (to make sure))


   /*************************************************/
   /*      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(tFD3);
   put_dfrm_ua(1, fl_txdata_frm, true);   // send securithi setting data

   rc = get_sfrm_ua(fl_ua_sfrm, tWT14_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]
   }

   /*************************************************/
   /*      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　3線式シリアルI/O ハンドシェーク対応（CSI＋HS）通信方式

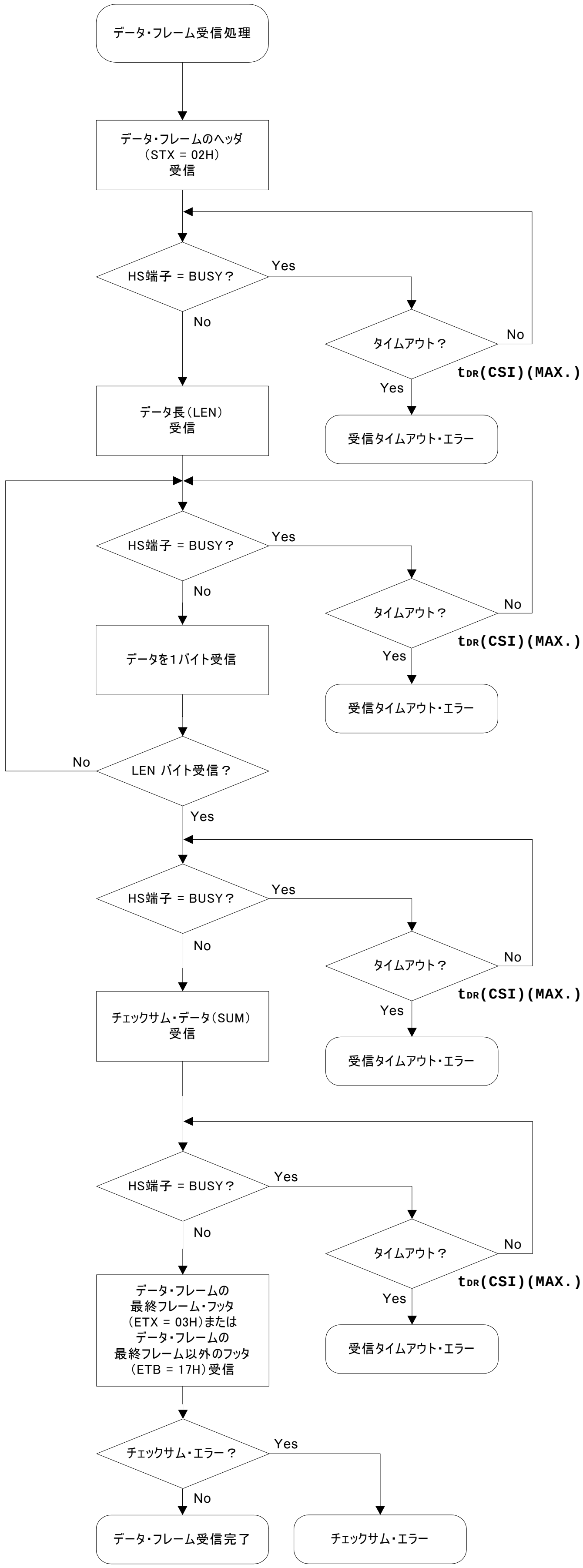
7. 1　コマンド・フレーム送信処理のフロー・チャート



7. 2　データ・フレーム送信処理のフロー・チャート



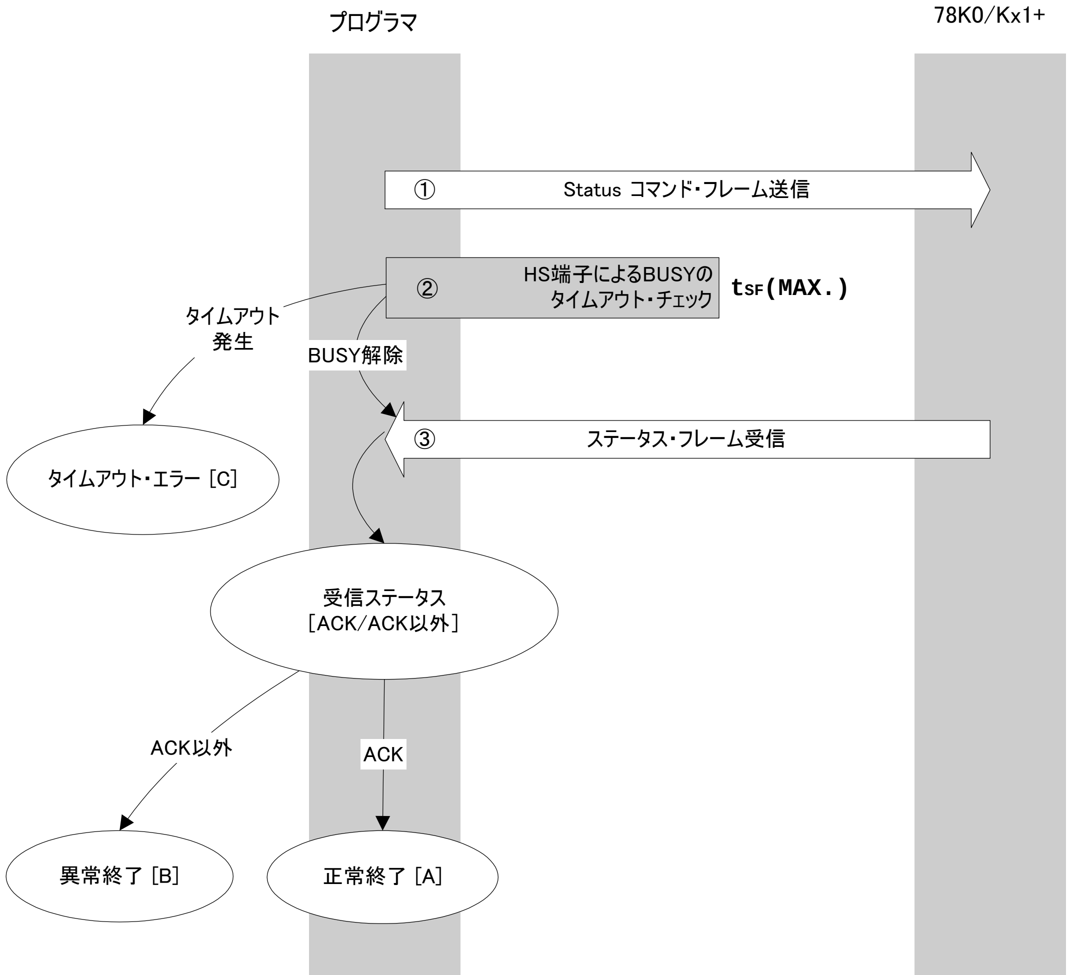
7.3　データ・フレーム受信処理のフロー・チャート



7. 4　Statusコマンド

7. 4. 1　処理手順チャート

Statusコマンド処理手順



7.4.2　処理手順説明

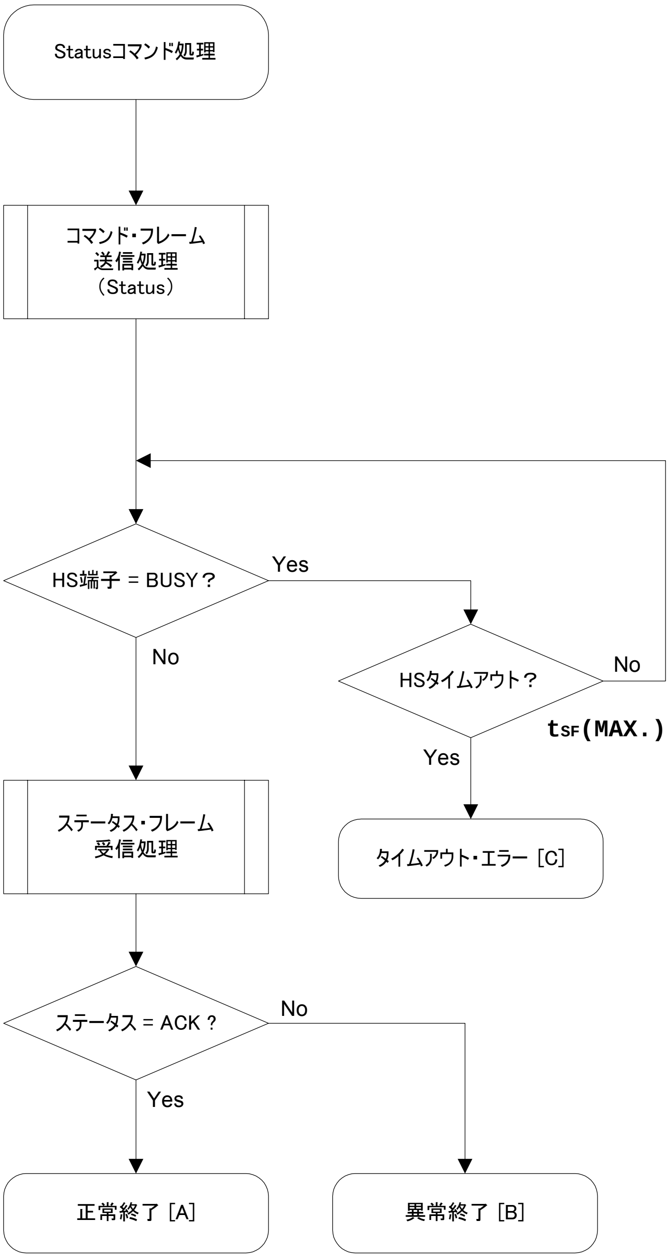
① コマンド・フレーム送信処理により，Statusコマンドを送信します。
② HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYのタイムアウトが発生した場合は，タイムアウト・エラー[C]となります（タイムアウト時間t_{SF}(MAX.)）。
③ ステータス・コードをチェックします。

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。

7.4.3　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	78K0/Kx1+からステータス・フレームを正常に受信しました。
異常終了 [B]	コマンド・エラー	04H	サポートされていないコマンド，または異常フレームを受信しました。
	パラメータ・エラー	05H	コマンド情報（パラメータ）が適切でありません。
	チェックサム・エラー	07H	プログラマから送信されたフレームのデータが異常です。
	ベリファイ・エラー	0FH	プログラマから送信されたデータとのベリファイでエラーが発生しました。
	プロテクト・エラー	10H	Security Set コマンドで禁止した処理を実行しようとしました。
	否定応答（NACK）	15H	否定応答
	FLMD エラー	18H	書き込みエラーが発生しました。
	MRG10 エラー	1AH	消去エラーが発生しました。
	MRG11 エラー	1BH	データ書き込み時に内部ベリファイ・エラーが発生，またはブランク・チェック・エラーが発生しました。
	Write エラー	1CH	書き込みエラーが発生しました。
タイムアウト・エラー [C]		－	HS のビジーにタイムアウトが発生しました。

7. 4. 4　フロー・チャート


Statusコマンド処理
コマンド･フレーム
送信処理
（Status）
HS端子 = BUSY？
Yes
No
HSタイムアウト？
No
tSF(MAX.)
Yes
ステータス･フレーム
受信処理
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]

7. 4. 5　サンプル・プログラム

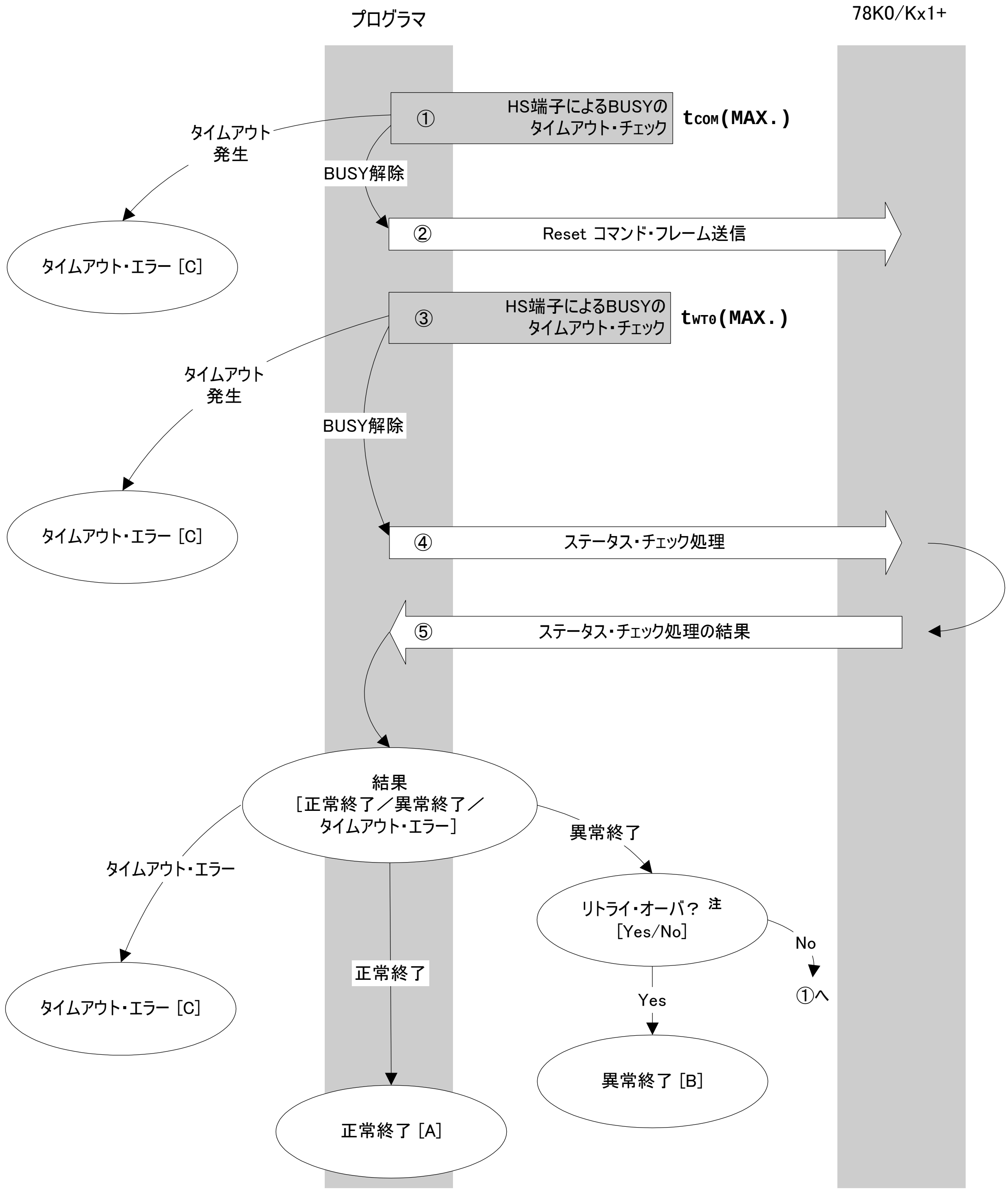
Statusコマンド処理のサンプル・プログラムです。

```
/****************************************************************/
/*                                                              */
/* Get status command (CSI-HS)                                  */
/*                                                              */
/****************************************************************/
/* [r] u16         ... decoded status or error code             */
/*                                                              */
/* (see fl.h/fl-proto.h &                                       */
/*         definition of decode_status() in fl.c)               */
/****************************************************************/
static u16 fl_hs_getstatus(void)
{
   u16     rc;
   u32     retry = 0;



   rc = put_cmd_hs(FL_COM_GET_STA, 1, fl_cmd_prm);// send "Status" command
   if (rc)
          return  rc;     // case [C]

   if (hs_busy_to(tSF_MAX))               // HS-Busy t.o. ?
          return  FLC_HSTO_ERR;           // t.o. detected  : case [C]

   if (rc = get_sfrm_hs(fl_rxdata_frm))
          return  rc;            // case [C] or [B(checksum error)]


   rc = decode_status(fl_st1);     // decode return code

   return rc;             // case [A] or [B]
}
```

7. 5　Resetコマンド

7. 5. 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。

7.5.2　処理手順説明

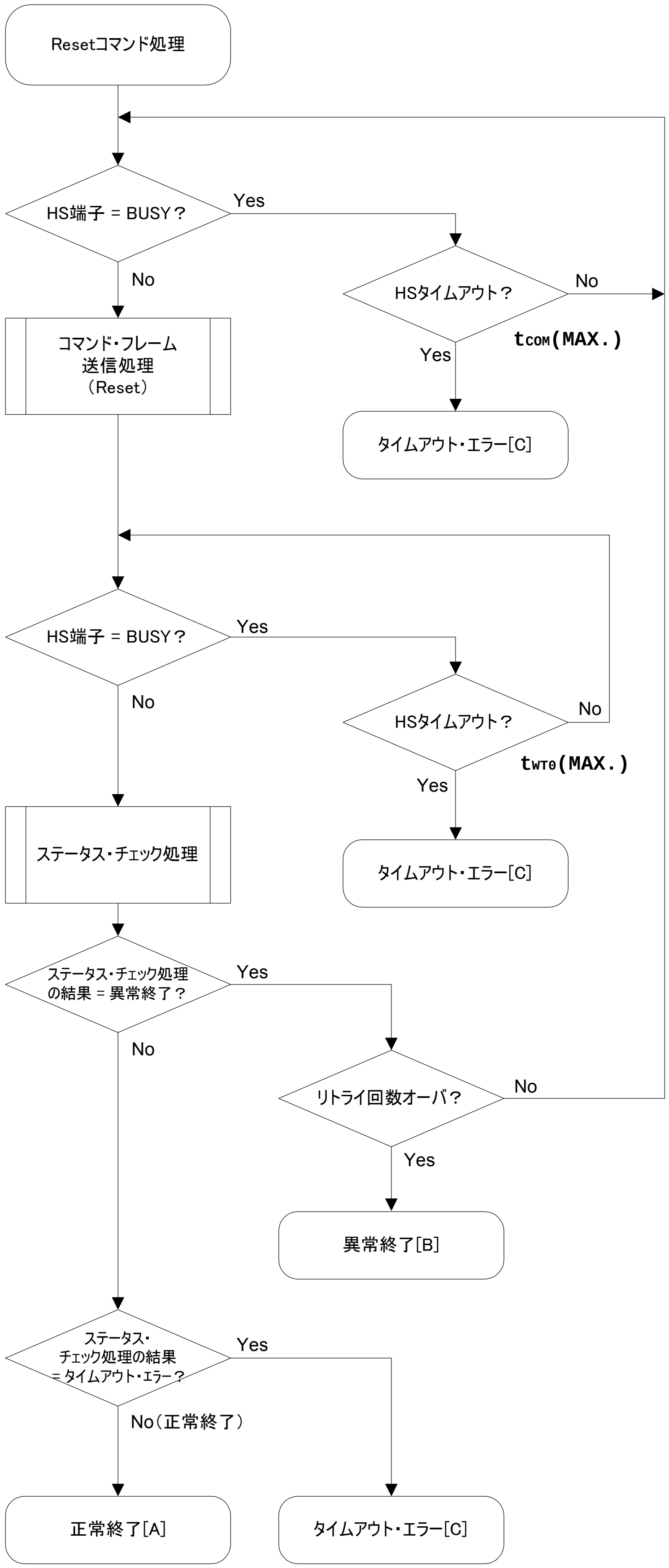
① HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYのタイムアウトが発生した場合は，タイムアウト・エラー[C] となります（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理により，Resetコマンド を送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYのタイムアウトが発生した場合は，タイムアウト・エラー[C] となります（タイムアウト時間t_{WT0}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：正常終了[A] です。
異常終了の場合　：リトライ・オーバでなければ①より再実行します。
リトライ・オーバであれば，異常終了[B] です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C] です。

7.5.3　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78K0/Kx1+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]		－	ステータス・チェック処理がタイムアウト・エラーで終了した，または HS 端子のビジーでタイムアウトが発生しました。

7. 5. 4　フロー・チャート


Resetコマンド処理
HS端子 = BUSY？
Yes
No
HSタイムアウト？
No
t_{COM}(MAX.)
Yes
コマンド・フレーム
送信処理
（Reset）
タイムアウト・エラー[C]
HS端子 = BUSY？
Yes
No
HSタイムアウト？
No
t_{WT0}(MAX.)
Yes
ステータス・チェック処理
タイムアウト・エラー[C]
ステータス・チェック処理
の結果 = 異常終了？
Yes
No
リトライ回数オーバ？
No
Yes
異常終了[B]
ステータス・
チェック処理の結果
= タイムアウト・エラー？
Yes
No（正常終了）
正常終了[A]
タイムアウト・エラー[C]

7. 5. 5　サンプル・プログラム

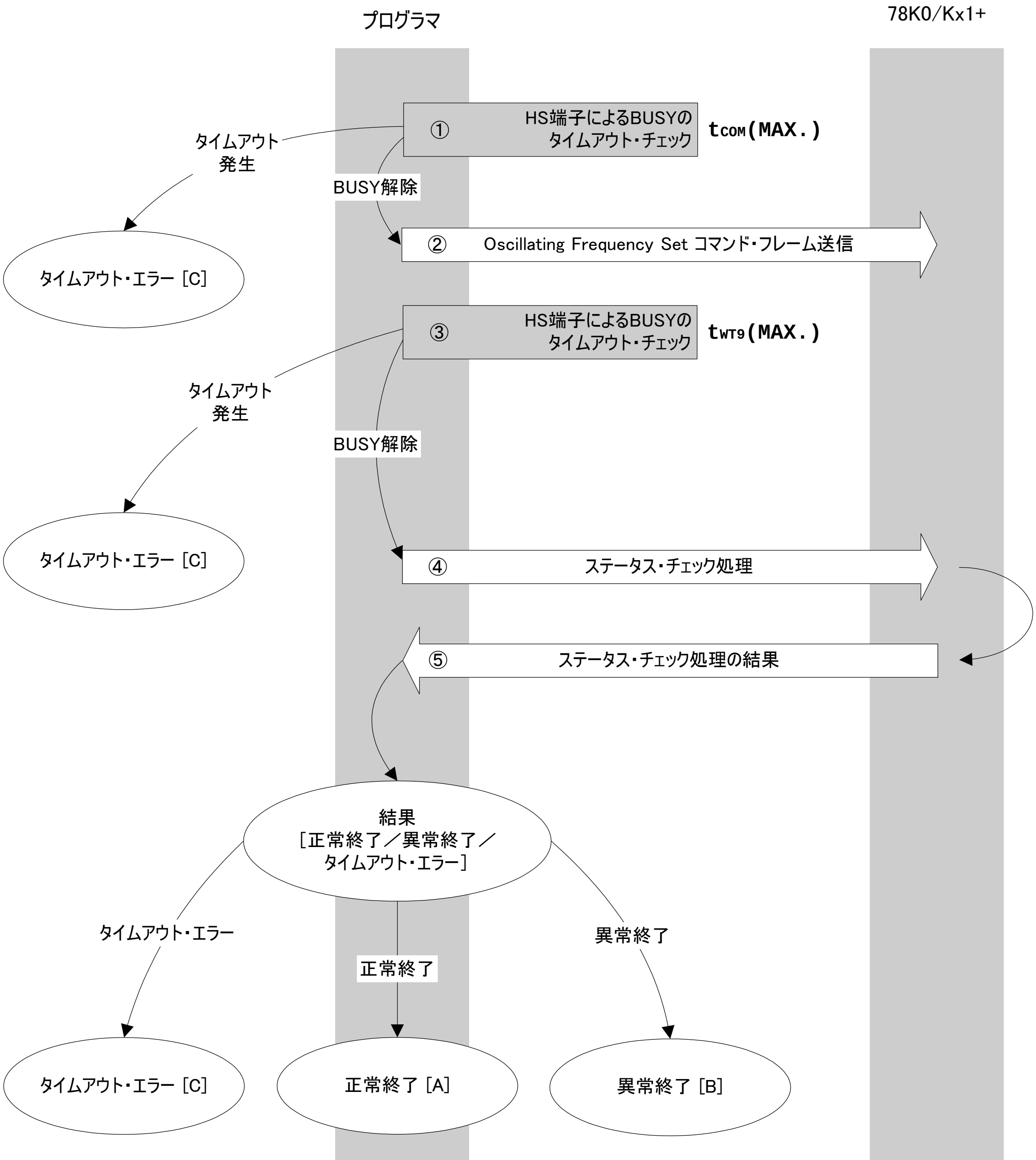
Resetコマンド処理のサンプル・プログラムです。

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

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

           if (hs_busy_to(tCOM_MAX))
                   return  FLC_HSTO_ERR;                  //t.o.detected:case[C]

           rc = put_cmd_hs(FL_COM_RESET, 1, fl_cmd_prm);  // send "Reset" command
           if (rc)
                   return  rc;     // case [C]

           if (hs_busy_to(tWT0_MAX))
                   return  FLC_HSTO_ERR;                  //t.o.detected:case[C]

           rc = fl_hs_getstatus();        // get status frame
           if (rc == FLC_ACK)             // ST1 = ACK ?
                   break;                 // case [A]
           //continue;                    // case [B] (if exit from loop)

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

7. 6　Oscillating Frequency Setコマンド

7. 6. 1　処理手順チャート

Oscillating Frequency Setコマンド処理手順



7.6.2　処理手順説明

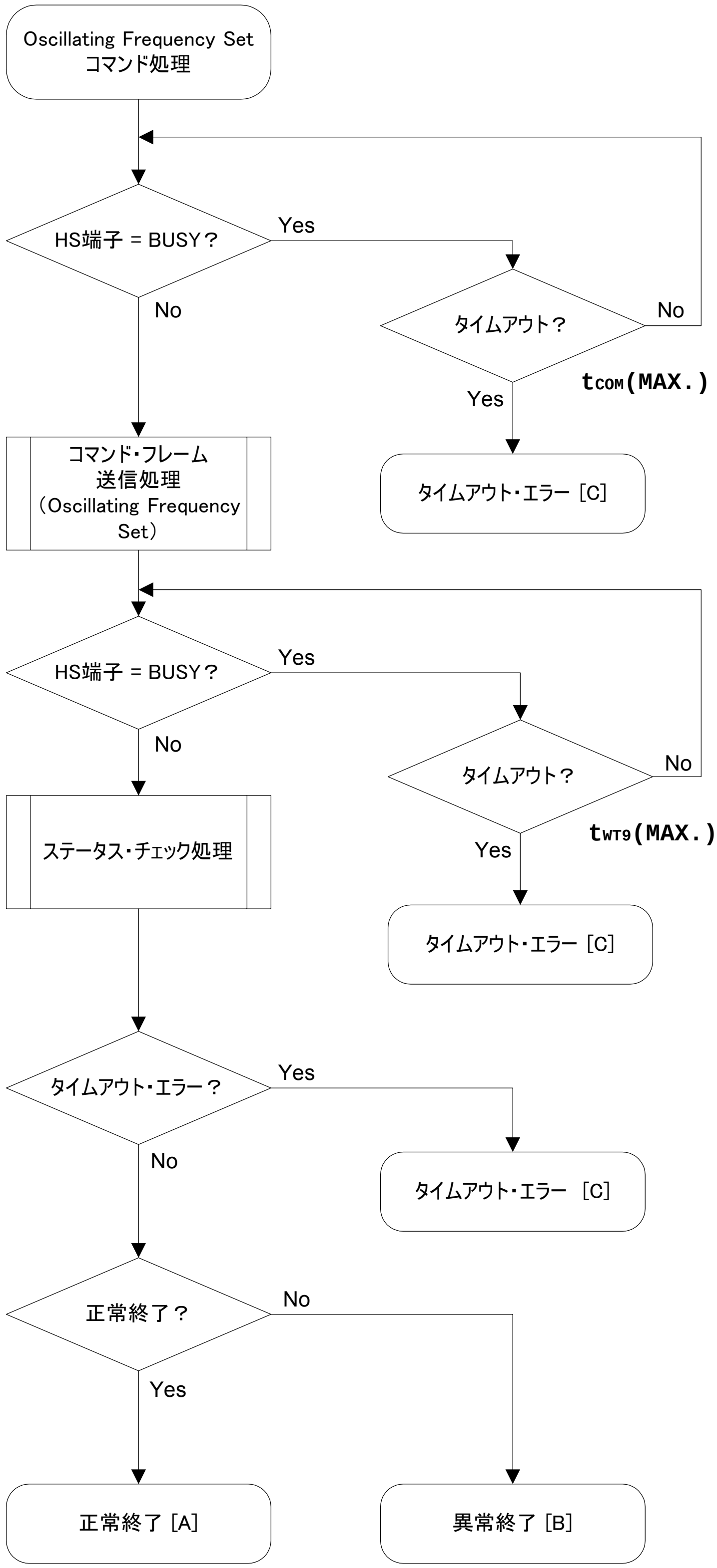
① HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー [C] となります
（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理により Oscillating Frequency Setコマンド を送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT9}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

<u>正常終了</u>の場合　：正常終了[A] です。
<u>異常終了</u>の場合　：異常終了[B] です。
<u>タイムアウト・エラー</u>の場合　：タイムアウト・エラー[C] です。

7.6.3　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，78K0/Kx1+に動作周波数を正しく設定できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	発振周波数値が範囲外です。
	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）。
タイムアウト・エラー [C]		−	HS 端子のビジーでタイムアウトしました。

7. 6. 4　フロー・チャート



Oscillating Frequency Set
コマンド処理
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tCOM(MAX.)
タイムアウト・エラー [C]
コマンド・フレーム
送信処理
（Oscillating Frequency
Set）
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tWT9(MAX.)
ステータス・チェック処理
タイムアウト・エラー [C]
タイムアウト・エラー？
Yes
No
タイムアウト・エラー [C]
正常終了？
No
Yes
正常終了 [A]
異常終了 [B]

7. 6. 5　サンプル・プログラム

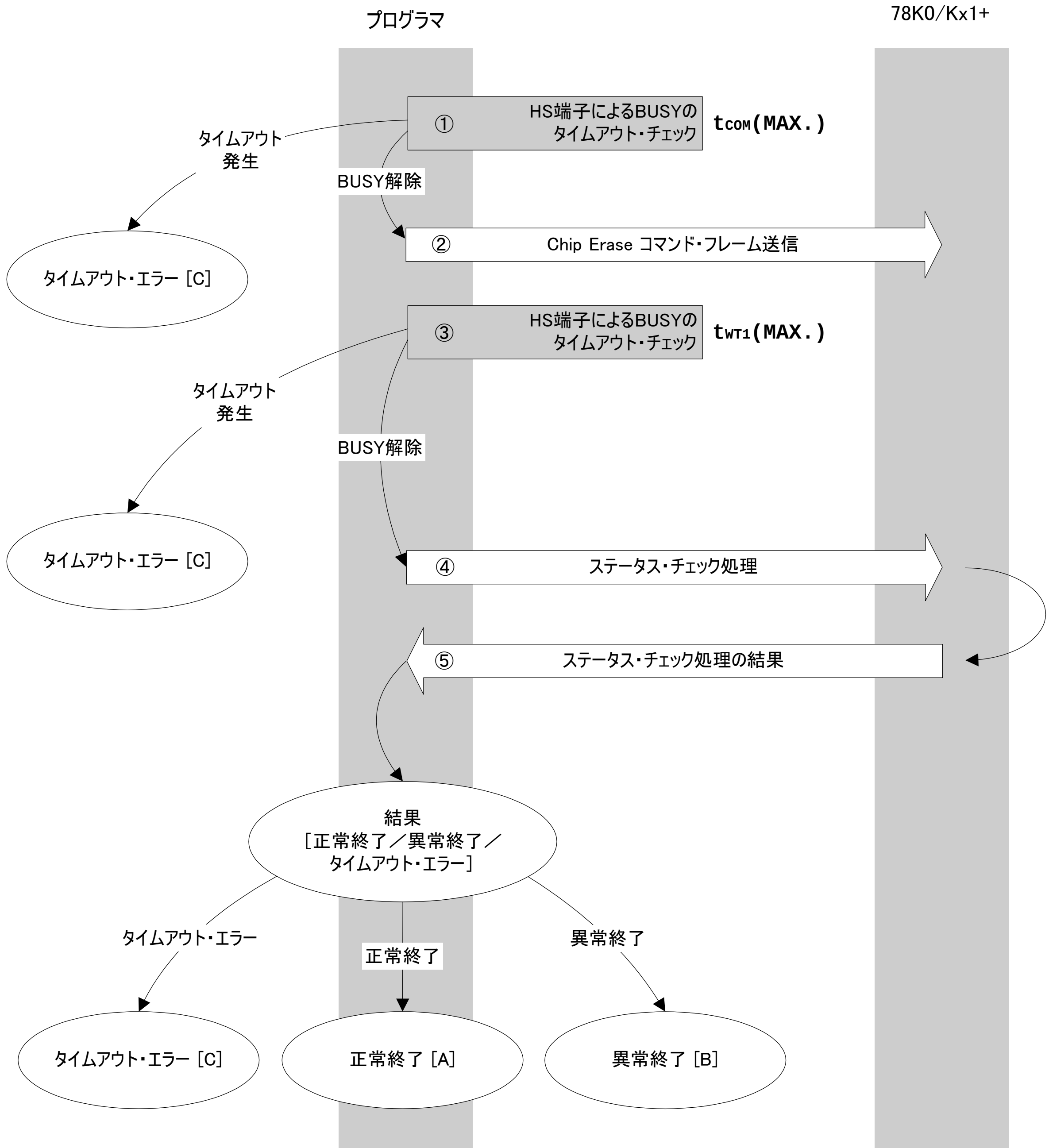
Oscillating Frequencyコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* Set Flash device clock value command (CSI-HS)                  */
/*                                                                */
/******************************************************************/
/* [i] u8 clk[4]   ... frequency data(D1-D4)                      */
/* [r] u16         ... error code                                 */
/******************************************************************/
/******************************************************************/
u16        fl_hs_setclk(u8 clk[])
{
   u16     rc;

   fl_cmd_prm[0] = clk[0]; // "D01"
   fl_cmd_prm[1] = clk[1]; // "D02"
   fl_cmd_prm[2] = clk[2]; // "D03"
   fl_cmd_prm[3] = clk[3]; // "D04"


   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm))
                                        // send "Oscilating Frequency Set" command
           return  rc;                     // case [C]

   if (hs_busy_to(tWT9_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]


   rc = fl_hs_getstatus();                 // get status frame
// switch(rc) {
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

7. 7　Chip Eraseコマンド

7. 7. 1　処理手順チャート

Chip Eraseコマンド処理手順



7.7.2　処理手順説明

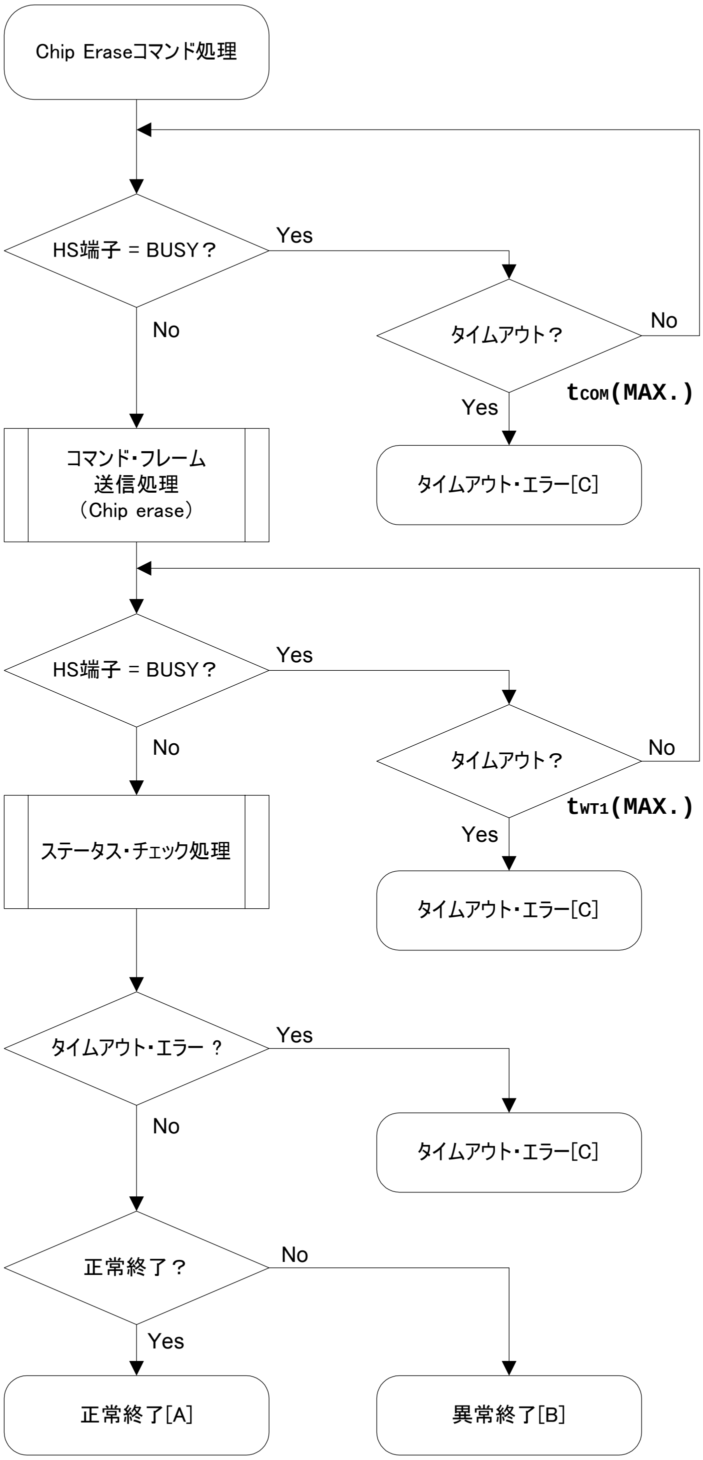
① HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー[C]となります
（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理にてChip Eraseコマンドを送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー[C]となります
（タイムアウト時間t_{WT1}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：正常終了[A]です。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

7.7.3　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，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」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）。
	消去エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]		－	HS 端子のビジーでタイムアウトしました。

7. 7. 4　フロー・チャート



Chip Eraseコマンド処理
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tCOM(MAX.)
タイムアウト・エラー[C]
コマンド・フレーム
送信処理
（Chip erase）
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tWT1(MAX.)
タイムアウト・エラー[C]
ステータス・チェック処理
タイムアウト・エラー ？
Yes
No
タイムアウト・エラー[C]
正常終了？
No
Yes
正常終了[A]
異常終了[B]

7. 7. 5　サンプル・プログラム

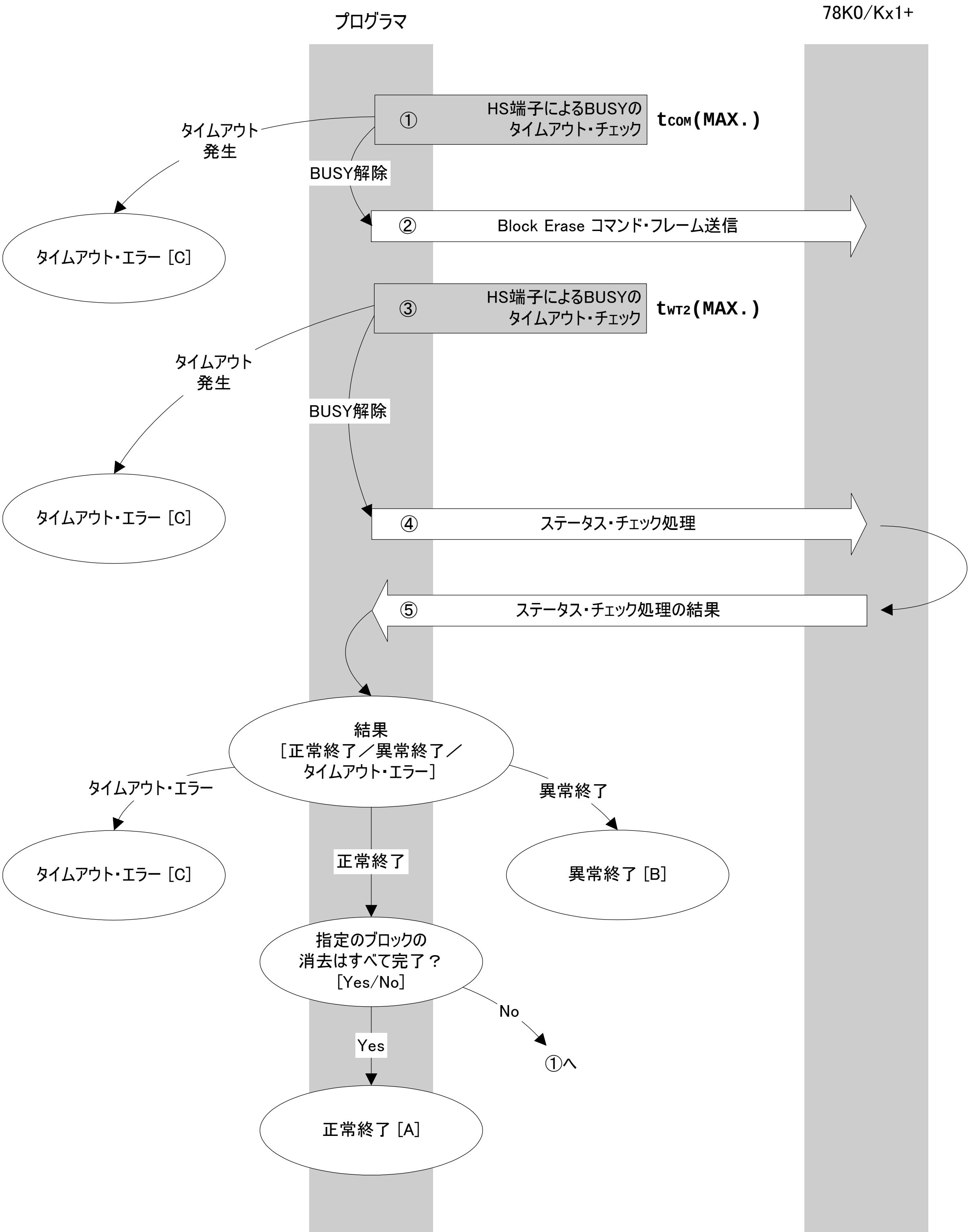
Chip Eraseコマンド処理のサンプル・プログラムです。

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

   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;                  // t.o. detected

   if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))
                                                       //send"ChipErase"command
           return  rc;                    // case [C]

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

   rc = fl_hs_getstatus();                // get status frame
// switch(rc) {
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

7. 8 Block Eraseコマンド

7. 8. 1 処理手順チャート

Block Eraseコマンド処理手順



7.8.2　処理手順説明

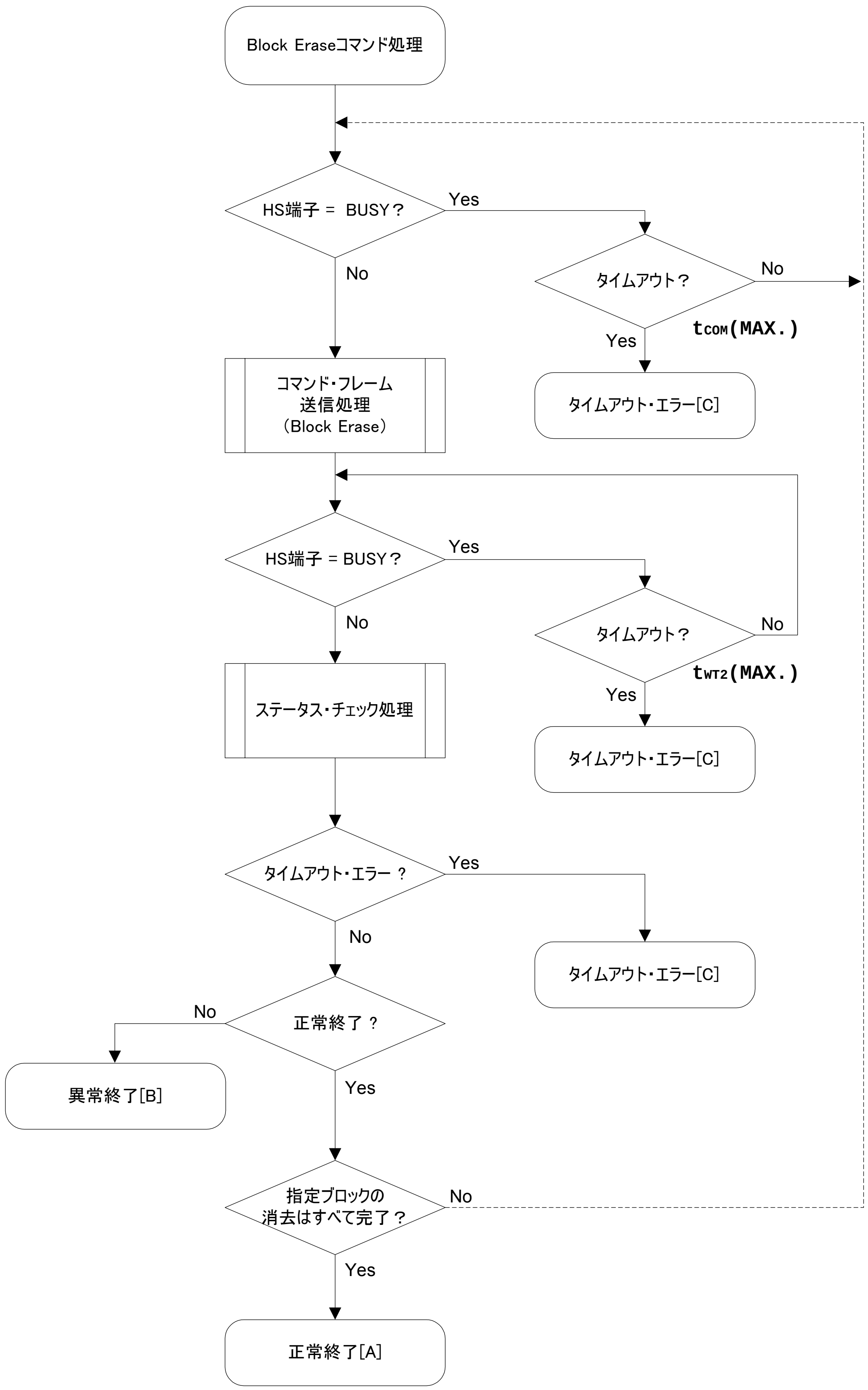
① HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー[C]となります（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理にてBlock Eraseコマンドを送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
BUSYがタイムアウトした場合は，タイムアウト・エラー[C]となります（タイムアウト時間t_{WT2}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合	：指定したブロックの消去がすべて完了していない場合は，ブロック番号を変えて①より再実行します。 指定したすべてのブロックの消去が完了した場合は，正常終了[A]です。
異常終了の場合	：異常終了[B]です。
タイムアウト・エラーの場合	：タイムアウト・エラー[C]です。

7.8.3　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，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	ブロック番号が範囲外です。
	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。
	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」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）。
	消去エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]		－	HS 端子のビジーでタイムアウトしました。

7. 8. 4　フロー・チャート


Block Eraseコマンド処理
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tCOM(MAX.)
タイムアウト・エラー[C]
コマンド・フレーム
送信処理
（Block Erase）
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
tWT2(MAX.)
タイムアウト・エラー[C]
ステータス・チェック処理
タイムアウト・エラー ？
Yes
No
タイムアウト・エラー[C]
正常終了 ？
No
Yes
異常終了[B]
指定ブロックの
消去はすべて完了？
No
Yes
正常終了[A]

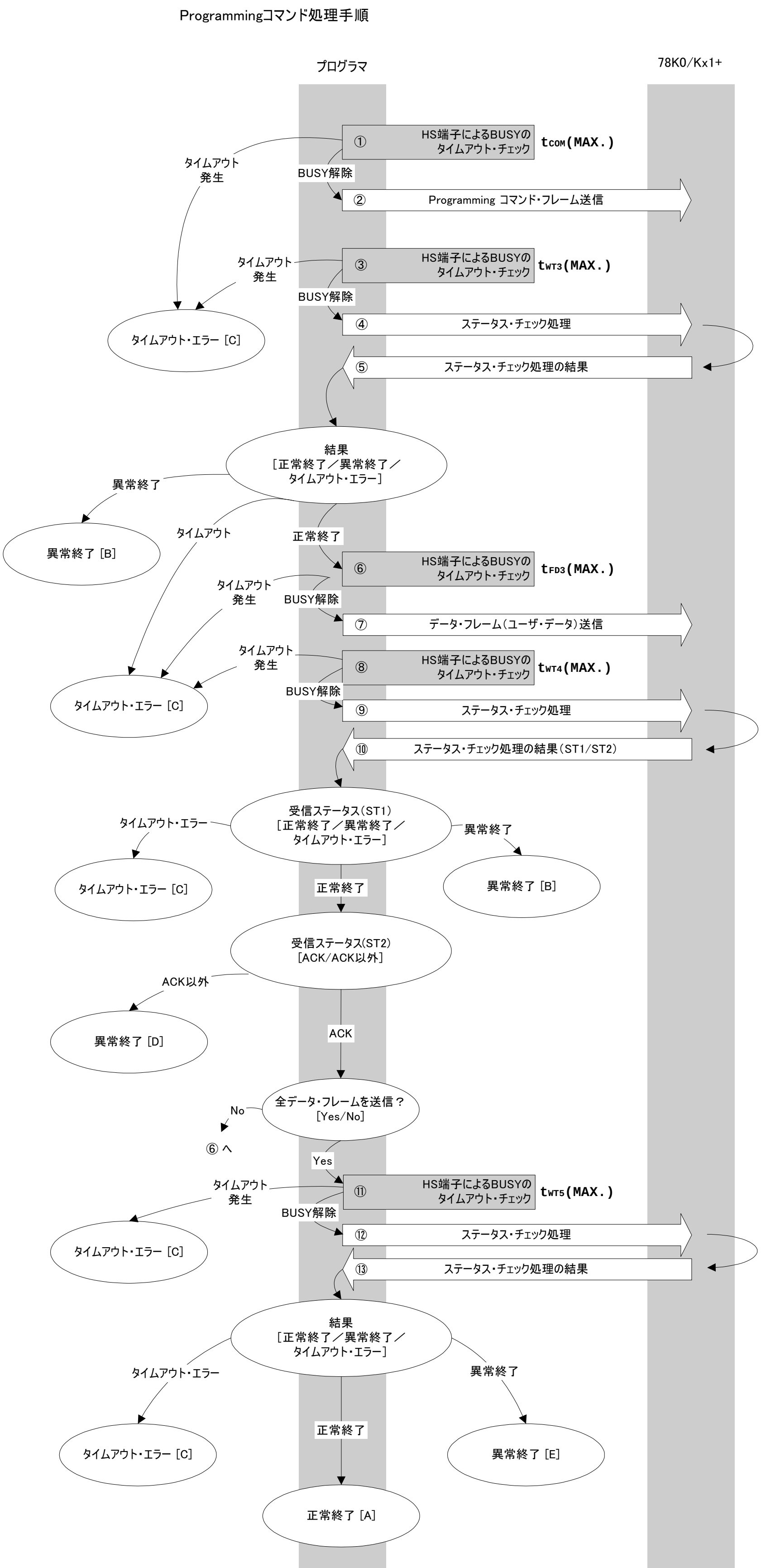
7.8.5　サンプル・プログラム

1ブロック分のBlock Eraseコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Erase block command (CSI-HS)                                  */
/*                                                               */
/*****************************************************************/
/* [i] u8 block    ... block number                              */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16       fl_hs_erase_blk(u8 block)
{
   u16     rc;

   fl_cmd_prm[0] = block;         // set param (BLK)

   if (hs_busy_to(tCOM_MAX))
          return  FLC_HSTO_ERR;                  // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm))
                                                     //send "Block Erase" command
          return  rc;                            // case [C]

   if (hs_busy_to(tWT2_MAX))
          return  FLC_HSTO_ERR;                  // t.o. detected :case [C]

   rc = fl_hs_getstatus();                // get status frame
// switch(rc) {
//        case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//        case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//        default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

7. 9　Programmingコマンド

7. 9. 1　処理手順チャート



7.9.2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
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 HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT3}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合	：⑥に進みます。
異常終了の場合	： 異常終了[B] です。
タイムアウト・エラーの場合	： タイムアウト・エラー[C] です。

⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{FD3}(MAX.)）。
⑦ 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。
⑧ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT4}(MAX.)）。
⑨ ステータス・チェック処理により，ステータス・フレームを取得します。
⑩ ステータス・チェック処理の結果（ステータス・コード（ST1/ST2））に応じて次の処理を行います（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 = 異常終了の場合	： 異常終了[B] です。
ST1 = タイムアウト・エラーの場合	： タイムアウト・エラー[C] です。
ST1 = 正常終了の場合	：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以外の場合	： 異常終了[D] です。
・ST2 = ACKの場合	：全ユーザ・データを送信した場合は⑪へ，まだ送信するユーザ・データがある場合は⑥から実行します。

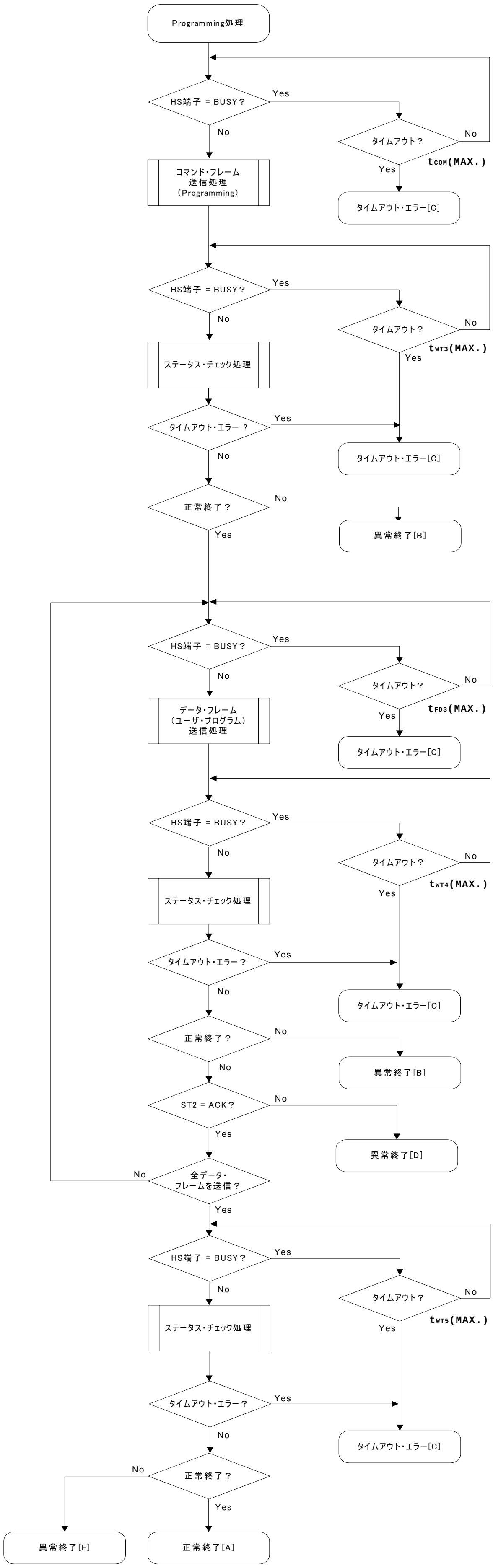
⑪ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT5}(MAX.)）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合	： 正常終了[A] です（書き込み完了後の内部ベリファイ・チェックが正常であったことを示します）。
異常終了の場合	： 異常終了[E] です（書き込み完了後の内部ベリファイ・チェックが異常であったことを示します）。
タイムアウト・エラーの場合	： タイムアウト・エラー[C] です。

7.9.3　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 8 の倍数でありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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。
	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]		－	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	FLMD エラー	18H（ST2）	書き込みエラーが発生しました。
	Write エラー	1CH（ST2）	書き込みエラーが発生しました。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

7. 9. 4　フロー・チャート


Programming処理
HS端子 = BUSY？
Yes
No
タイムアウト？
No
Yes
t_{COM}(MAX.)
タイムアウト・エラー[C]
コマンド・フレーム
送信処理
(Programming)
HS端子 = BUSY？
タイムアウト？
t_{WT3}(MAX.)
ステータス・チェック処理
タイムアウト・エラー？
タイムアウト・エラー[C]
正常終了？
異常終了[B]
HS端子 = BUSY？
タイムアウト？
t_{FD3}(MAX.)
データ・フレーム
(ユーザ・プログラム)
送信処理
タイムアウト・エラー[C]
HS端子 = BUSY？
タイムアウト？
t_{WT4}(MAX.)
ステータス・チェック処理
タイムアウト・エラー？
タイムアウト・エラー[C]
正常終了？
異常終了[B]
ST2 = ACK？
異常終了[D]
全データ・
フレームを送信？
HS端子 = BUSY？
タイムアウト？
t_{WT5}(MAX.)
ステータス・チェック処理
タイムアウト・エラー？
タイムアウト・エラー[C]
正常終了？
異常終了[E]
正常終了[A]

7. 9. 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。

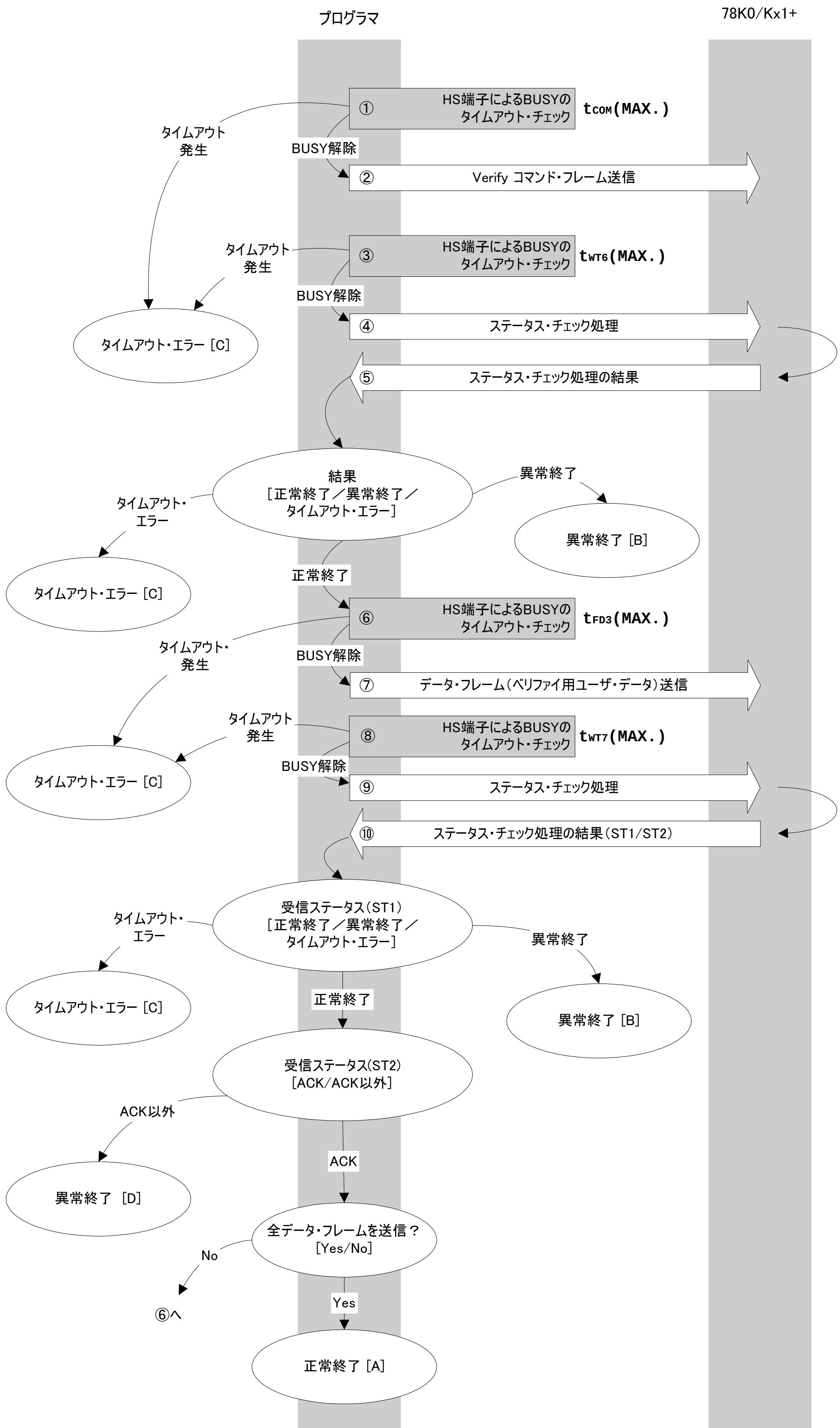
```
/******************************************************************/
/*                                                                */
/* Write command (CSI-HS)                                         */
/*                                                                */
/******************************************************************/
/* [i] u32 top     ... start address                              */
/* [i] u32 bottom  ... end address                                */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_hs_write(u32 top, u32 bottom)
{
   u16     rc;
   u32     send_head, send_size;
   bool    is_end;
   u16     block_num;


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

   block_num = get_block_num(top, bottom); // get block num

   /************************************************/
   /*      send command & check status             */
   /************************************************/
   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected

   if (rc = put_cmd_hs(FL_COM_WRITE, 7, fl_cmd_prm))  //send "Programming" command
           return  rc;                                // t.o. detected

   if (hs_busy_to(tWT3_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected

   rc = fl_hs_getstatus();                 // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                     break;  // continue
   //      case    FLC_HSTO_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 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;

                if (hs_busy_to(tFD3_MAX))            // t.o. check before sending data frame
                        return  FLC_HSTO_ERR;        // t.o. detected

                if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                                                                        // send user data
                        return  rc;                     // error detected


                if (hs_busy_to(tWT4_MAX))
                        return  FLC_HSTO_ERR;           // t.o. detected


                rc = fl_hs_getstatus();                 // get status frame
                switch(rc) {
                        case    FLC_NO_ERR:                     break;  // continue
                //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
                        default:                return  rc;     break;  // case [B]
                }
                if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
                        rc = decode_status(fl_st2);     // No
                        return  rc;                     // case [D]
                }
                if (is_end)                     // send all user data ?
                        break;                  // yes



        }
        /************************************************/
        /*      Check internally verify                 */
        /************************************************/
        if (hs_busy_to(tWT5_MAX * block_num))
                return  FLC_HSTO_ERR;           // t.o. detected

        rc = fl_hs_getstatus();                 // get status frame
//      switch(rc) {
//              case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//              case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//              default:                return  rc;     break;  // case [B]
//      }
        return rc;

}
```

7. 10　Verifyコマンド

7. 10. 1　処理手順チャート

Verifyコマンド処理手順



7. 10. 2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理にて Verifyコマンド を送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT6}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　　　　　　：⑥に進みます。
異常終了の場合　　　　　　： 異常終了[B] です。
タイムアウト・エラーの場合　： タイムアウト・エラー[C] です。

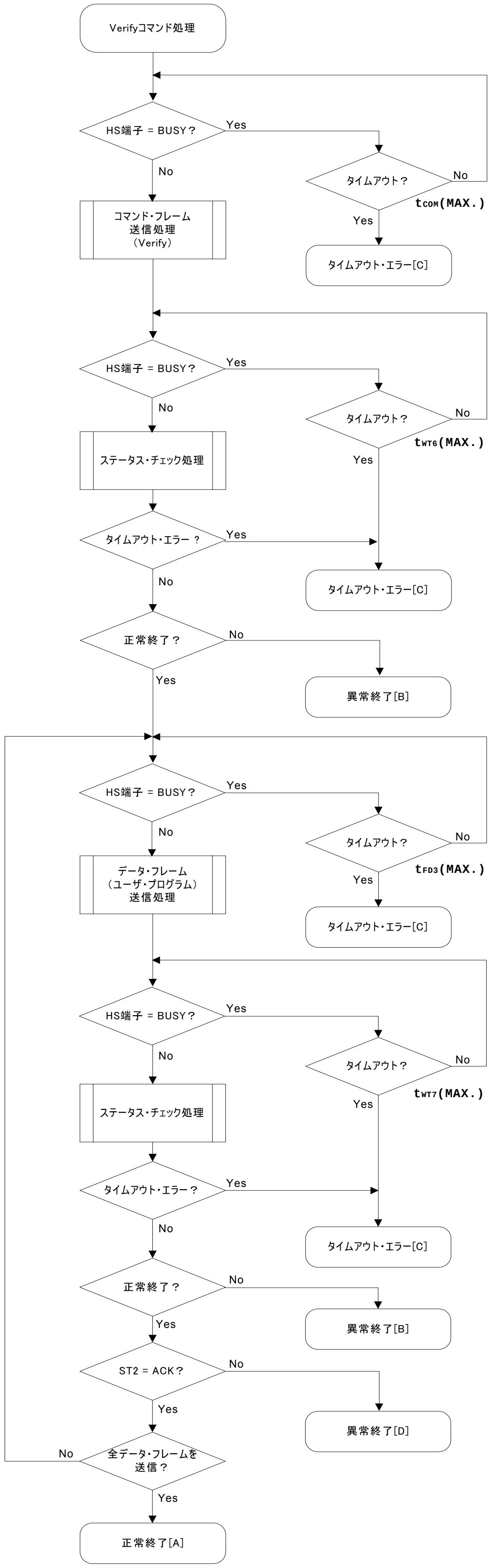
⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{FD3}(MAX.)）。
⑦ データ・フレーム送信処理により，ベリファイ用ユーザ・データを送信します。
⑧ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT7}(MAX.)）。
⑨ ステータス・チェック処理によりステータス・データを取得します。
⑩ ステータス処理の結果（ステータス・コード（ST1/ST2））に応じて次の処理を行います（処理手順チャートやフロー・チャートも参照してください）。

ST1 = 異常終了の場合　　　　　　： 異常終了[B] です。
ST1 = タイムアウト・エラーの場合　： タイムアウト・エラー[C] です。
ST1 = 正常終了の場合　　　　　　：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⑥から再実行します。
・ST2 = ACK以外の場合　　　： 異常終了[D] です。

7. 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，ベリファイ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。2 K バイトごとの固定アドレスでありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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]		－	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	ベリファイ・エラー	0FH（ST2）	ベリファイに失敗しました。または，その他のエラーが発生しました。

7. 10. 4　フロー・チャート



7. 10. 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。

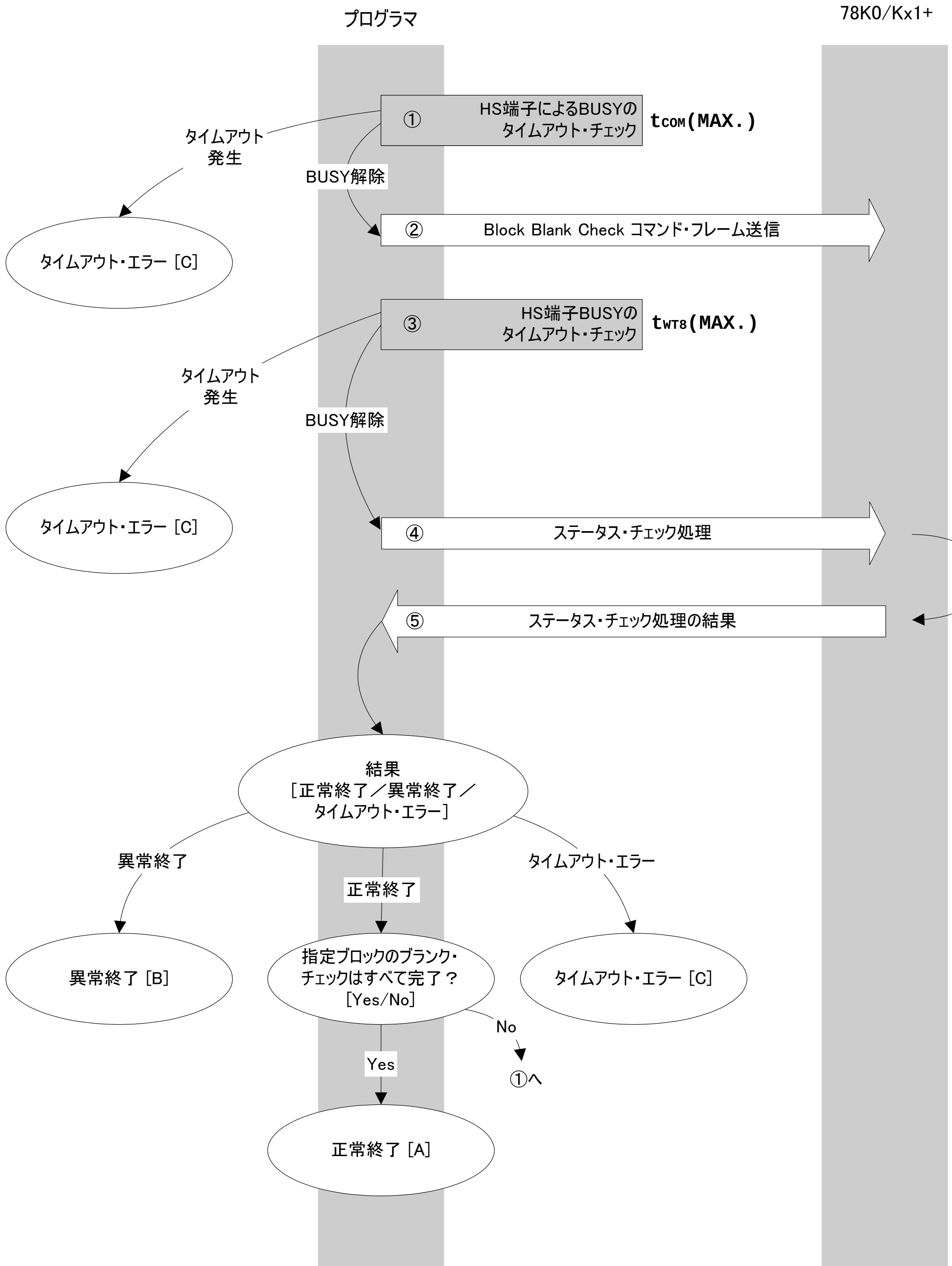
```
/******************************************************************/
/*                                                                */
/* Verify command (CSI-HS)                                        */
/*                                                                */
/******************************************************************/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

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


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

   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected

   if (rc = put_cmd_hs(FL_COM_VERIFY, 7, fl_cmd_prm))     // send "Verify" command
           return  rc;                    // error detected

   if (hs_busy_to(tWT6_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected

   rc = fl_hs_getstatus();                // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                    break;  // continue
   //      case    FLC_HSTO_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;

        if (hs_busy_to(tFD3_MAX))
                return  FLC_HSTO_ERR;           // t.o. detected

        if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                                                               // send user data
                return  rc;                     // error detected


        if (hs_busy_to(tWT7_MAX))
                return  FLC_HSTO_ERR;           // t.o. detected

        rc = fl_hs_getstatus();                 // get status frame
        switch(rc) {
                case    FLC_NO_ERR:                     break;  // continue
        //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
                default:                return  rc;     break;  // case [B]
        }
        if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
                rc = decode_status(fl_st2);     // No
                return  rc;                     // case [D]
        }
        if (is_end)                     // send all user data ?
                break;                  // yes
    }
    return  FLC_NO_ERR;     // case [A]
}
```

7. 11　Block Blank Checkコマンド

7. 11. 1　処理手順チャート

Block Blank Checkコマンド処理手順



7. 11. 2　処理手順説明

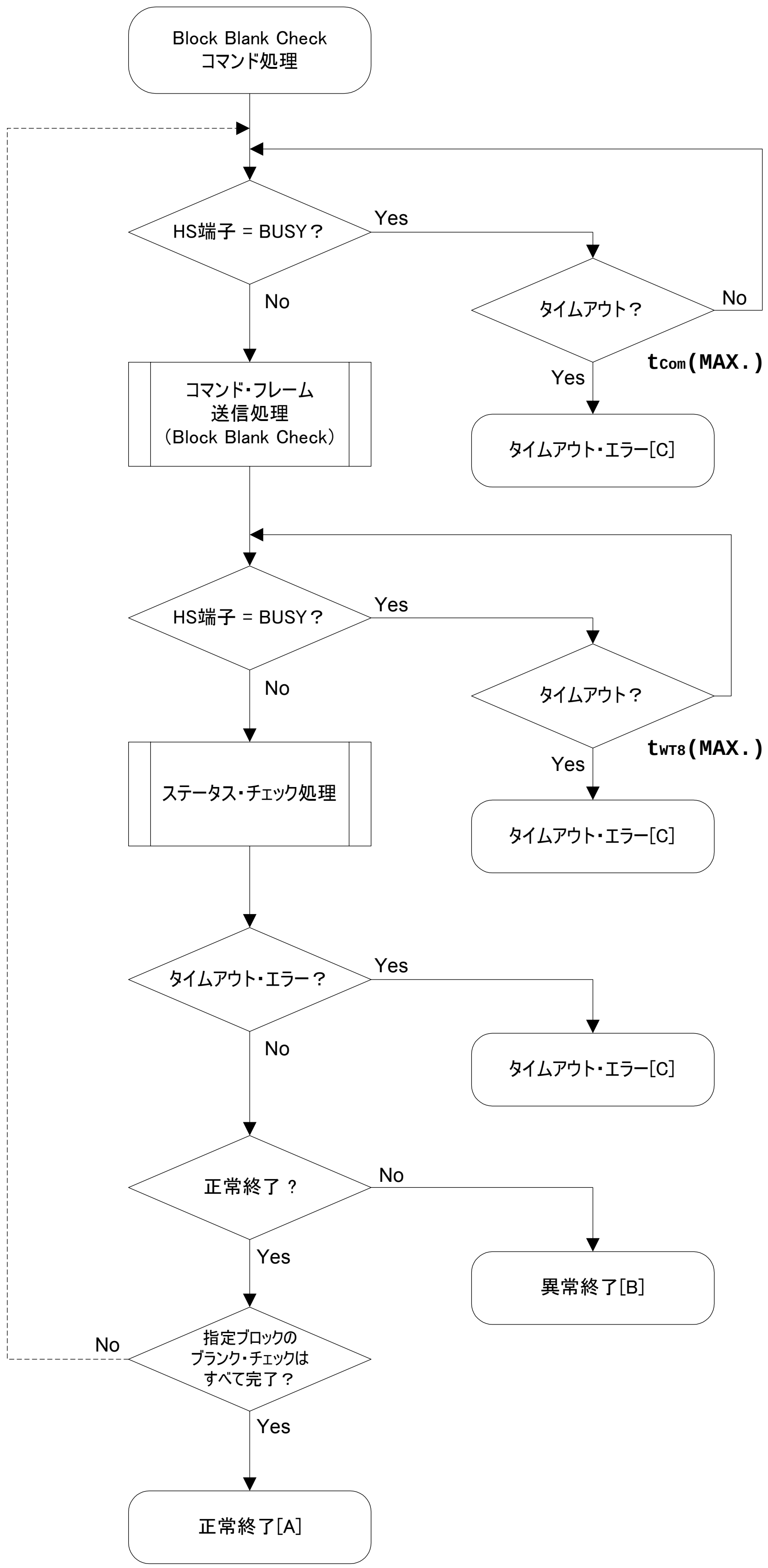
① HS端子による78K0/Kx1+のBUSYチェックを行います。
　タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
② コマンド・フレーム送信処理にて Block Blank Checkコマンド を送信します。
③ HS端子による78K0/Kx1+のBUSYチェックを行います。
　タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{WT8}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

<u>タイムアウト・エラー</u>の場合　：タイムアウト・エラー[C] です。
<u>異常終了</u>の場合　：異常終了[B] です。
<u>正常終了</u>の場合　：指定したブロックのブランク・チェックがすべて完了していない場合は，ブロック番号を変えて①より再実行します。
指定したすべてのブロックのブランク・チェックが完了した場合は， 正常終了[A] です。

7. 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，指定したブロックすべてがブランクであることを示します。
異常終了 [B]	パラメータ・エラー	05H	ブロック番号が範囲外です。
	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ありません。
タイムアウト・エラー [C]		－	HS 端子のビジーでタイムアウトしました。

7. 11. 4　フロー・チャート



7. 11. 5　サンプル・プログラム

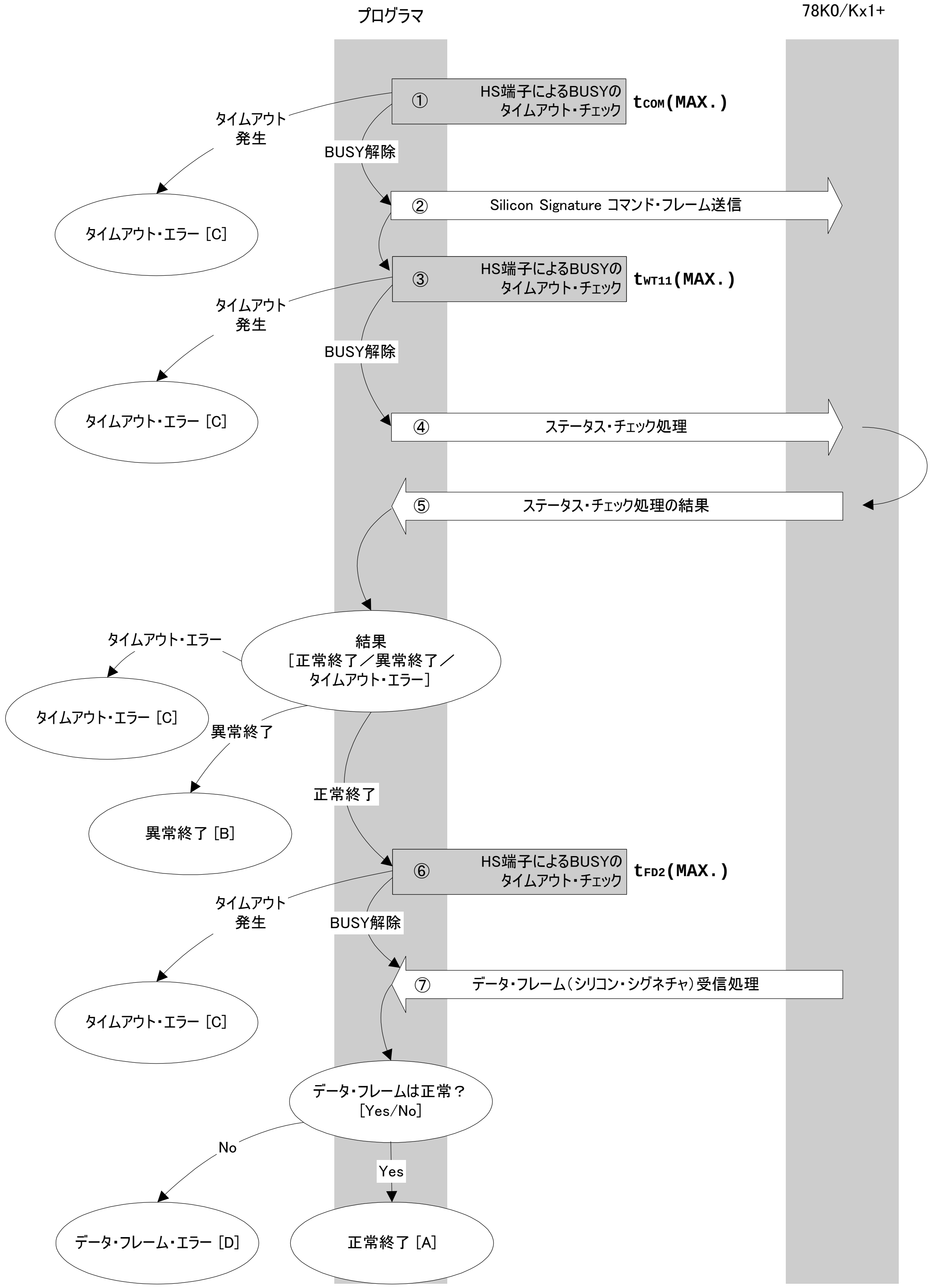
1ブロック分のBlock Blank Checkコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* Block blank check command (CSI-HS)                             */
/*                                                                */
/******************************************************************/
/* [i] u16 block   ... block number                               */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_hs_blk_blank_chk(u8 block)
{
   u16     rc;

   fl_cmd_prm[0] = block;  // "BLK"



   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm))
                                                 // send "Block Blank Check" command
           return  rc;                     // case [C]

   if (hs_busy_to(tWT8_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   rc = fl_hs_getstatus();                 // get status frame
// switch(rc) {
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

7. 12　Silicon Signatureコマンド

7. 12. 1　処理手順チャート

Silicon Signatureコマンド処理手順



7. 12. 2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
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 HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば 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・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B] です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C] です。

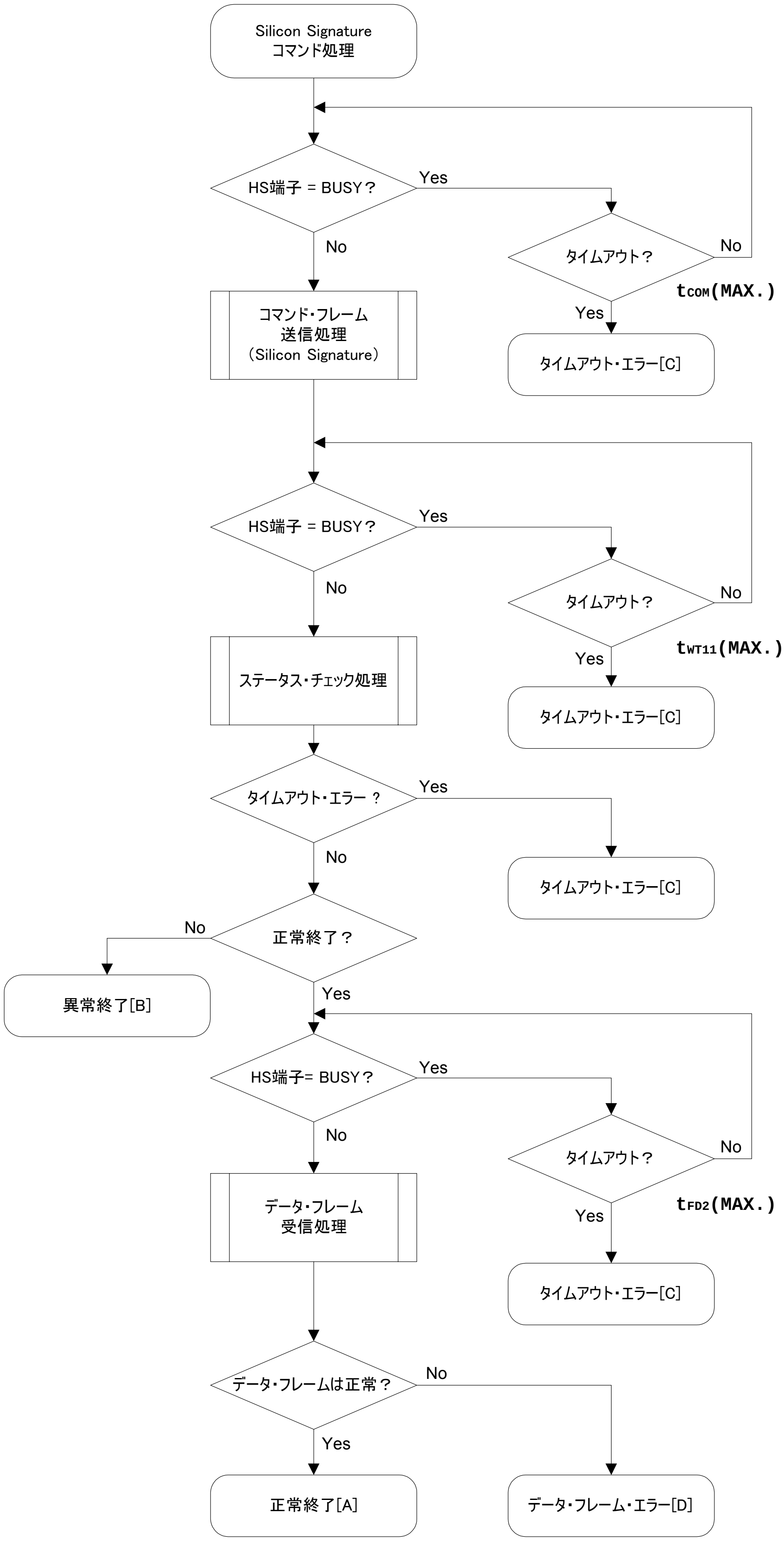
⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{FD2}(MAX.)）。
⑦ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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　：異常終了[D] です。

7. 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，シリコン・シグネチャ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]		－	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	データ・フレーム・エラー	－	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。

7. 12. 4　フロー・チャート



7. 12. 5　サンプル・プログラム

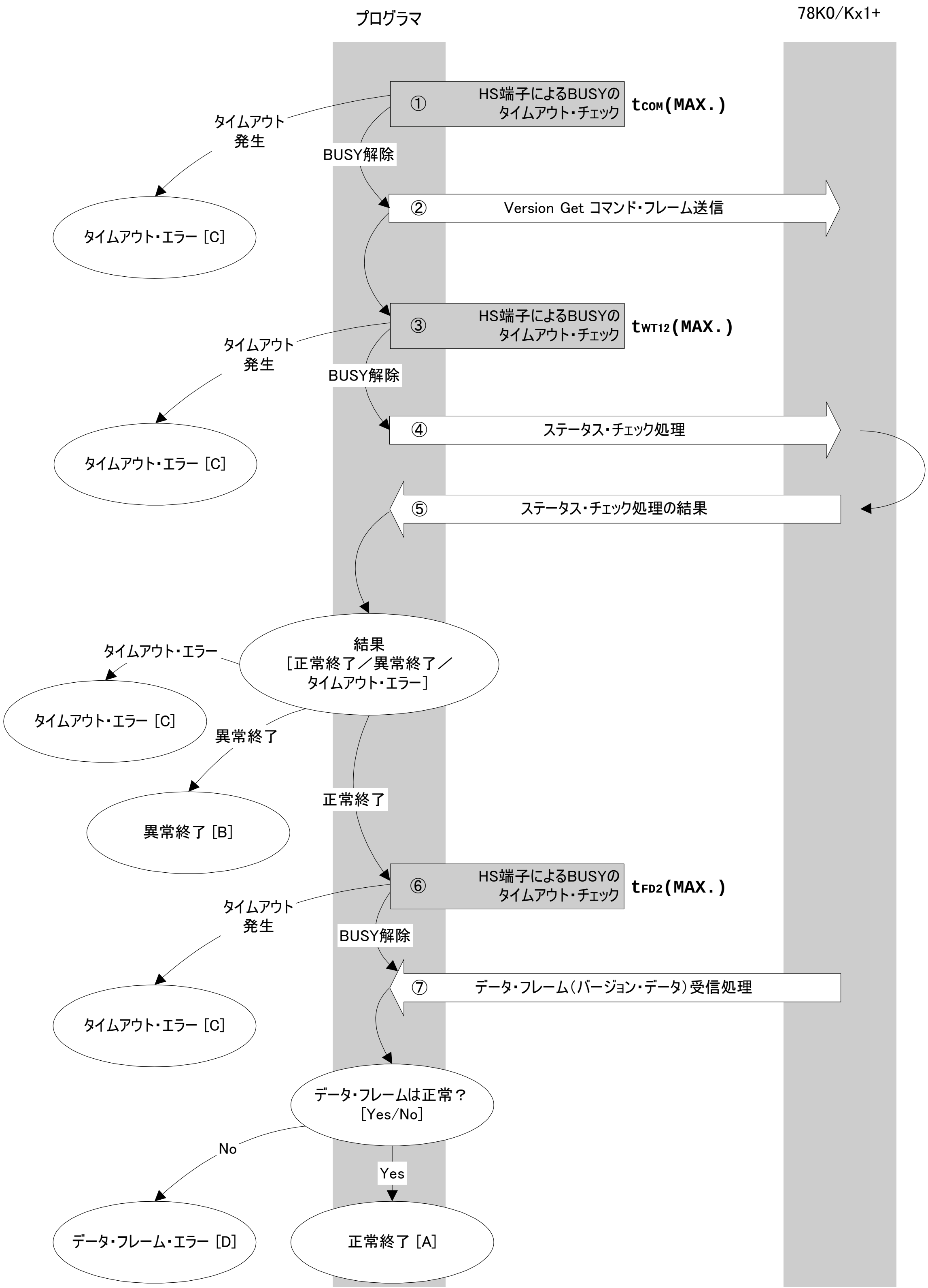
Silicon Signatureコマンド処理のサンプル・プログラムです。

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

   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm))
                                                //send "Silicon Signature" command
           return  rc;                    // error detected :case [C]

   if (hs_busy_to(tWT11_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]


   rc = fl_hs_getstatus();                // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                    break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [B]
   }


   if (hs_busy_to(tFD2_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]

   rc = get_dfrm_hs(fl_rxdata_frm);               // get signature data

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

   memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                          // copy Signature data

   return  rc;                            // case [A]

}
```

7. 13　Version Getコマンド

7. 13. 1　処理手順チャート

Version Getコマンド処理手順



7. 13. 2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
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 HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば 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・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B] です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C] です。

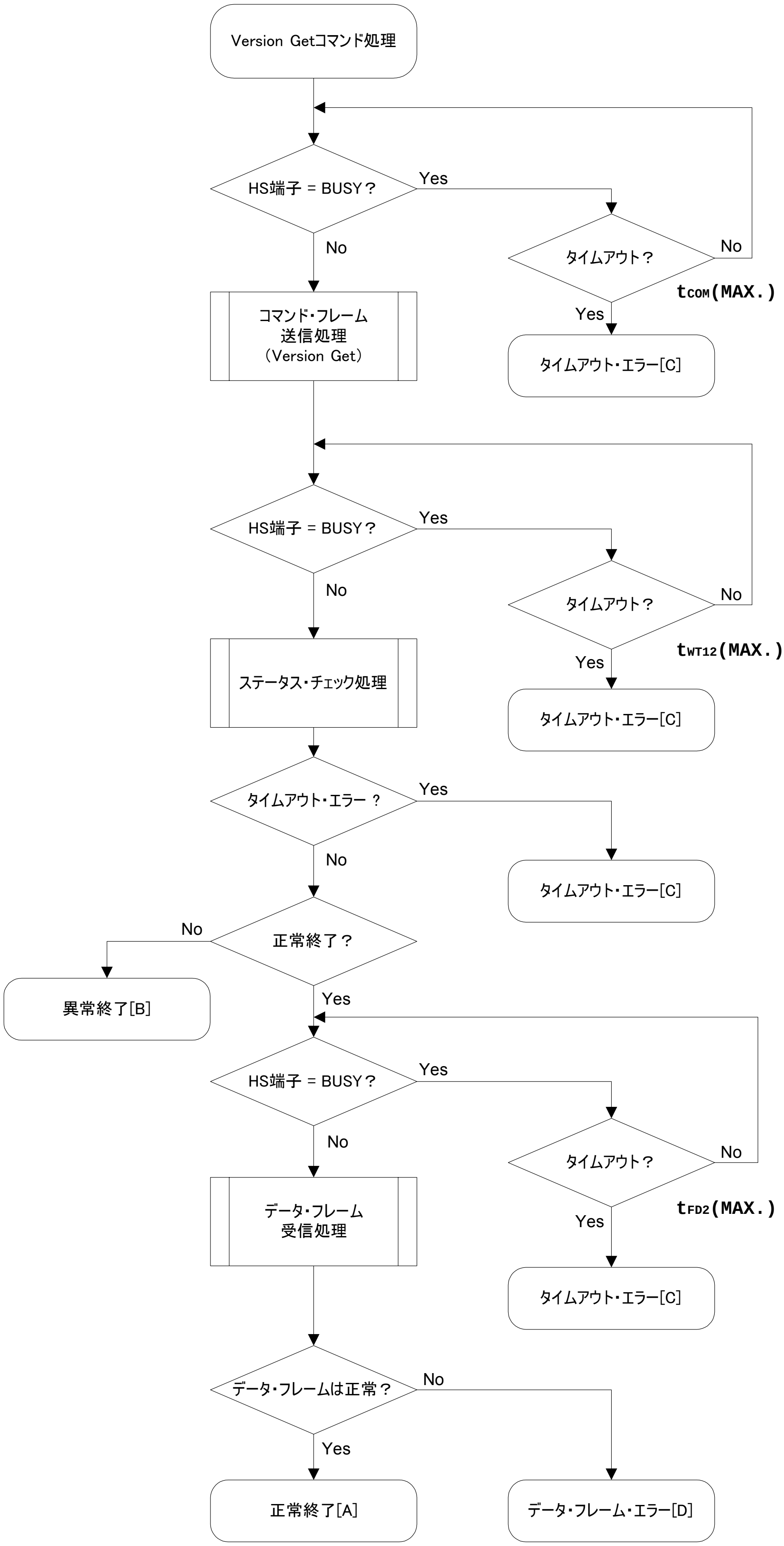
⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{FD2}(MAX.)）。
⑦ 受信したデータ・フレーム（バージョン・データ）をチェックします。

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　：異常終了[D] です。

7. 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，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]		－	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	データ・フレーム・エラー	－	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。

7. 13. 4　フロー・チャート



7. 13. 5　サンプル・プログラム

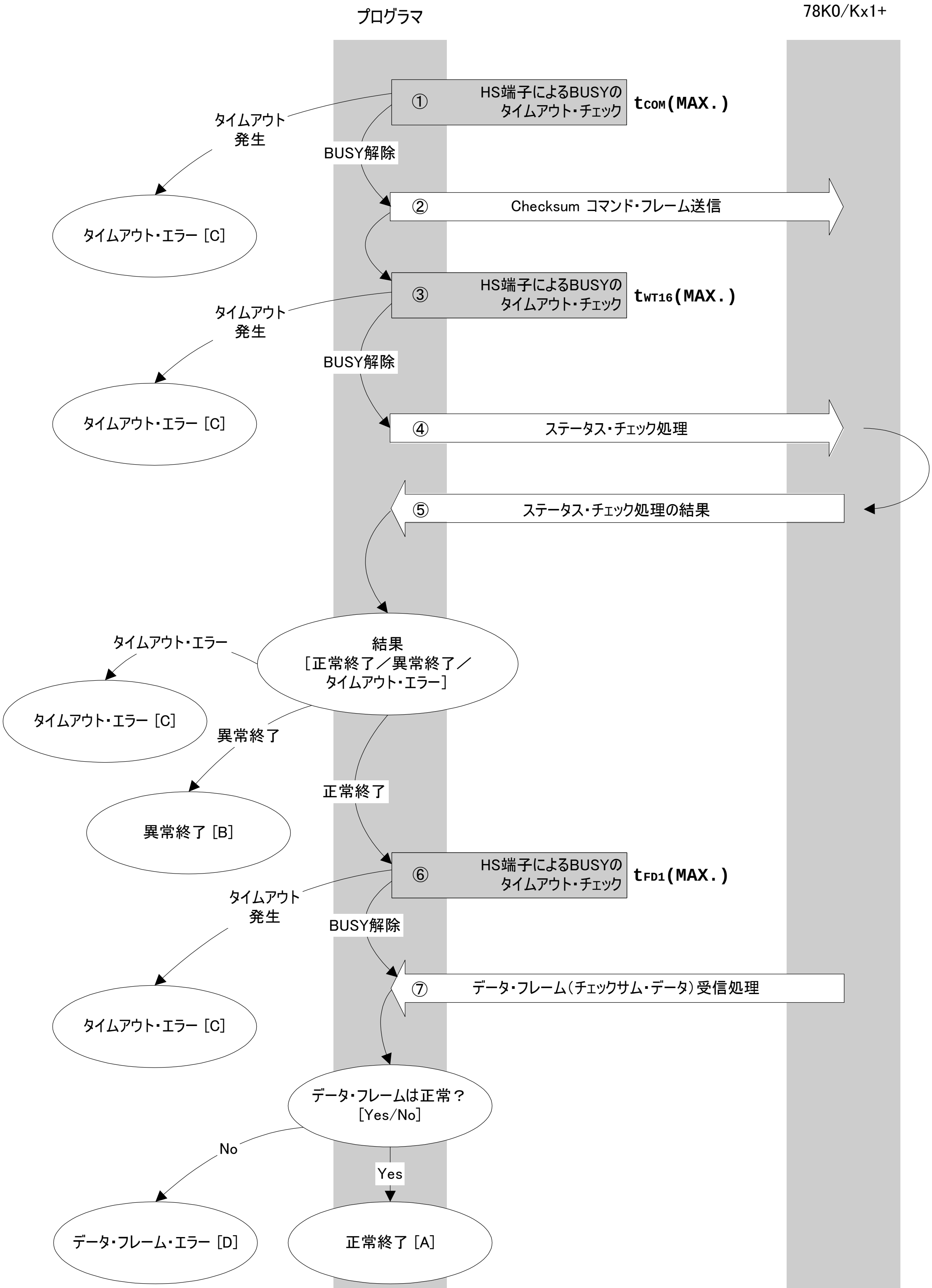
Version Getコマンド処理のサンプル・プログラムです。

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

   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_GET_VERSION, 1, fl_cmd_prm))
                                                   // send "Version Get" command
           return  rc;                     // error detected :case [C]

   if (hs_busy_to(tWT12_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   rc = fl_hs_getstatus();                 // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                     break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [B]
   }

   if (hs_busy_to(tFD2_MAX))
           return  FLC_HSTO_ERR;           // t.o. detected :case [C]

   rc = get_dfrm_hs(fl_rxdata_frm);                // get signature data
   switch(rc) {
           case    FLC_NO_ERR:                     break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [D]
   }
   memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// copy version data
   return  rc;                             // case [A]

}
```

7. 14　Checksumコマンド

7. 14. 1　処理手順チャート

Checksumコマンド処理手順



7. 14. 2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
　タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
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 HS端子による78K0/Kx1+のBUSYチェックを行います。
　タイムアウトであれば 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・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

　正常終了の場合　：⑥に進みます。
　異常終了の場合　：異常終了[B] です。
　タイムアウト・エラーの場合　：タイムアウト・エラー[C] です。

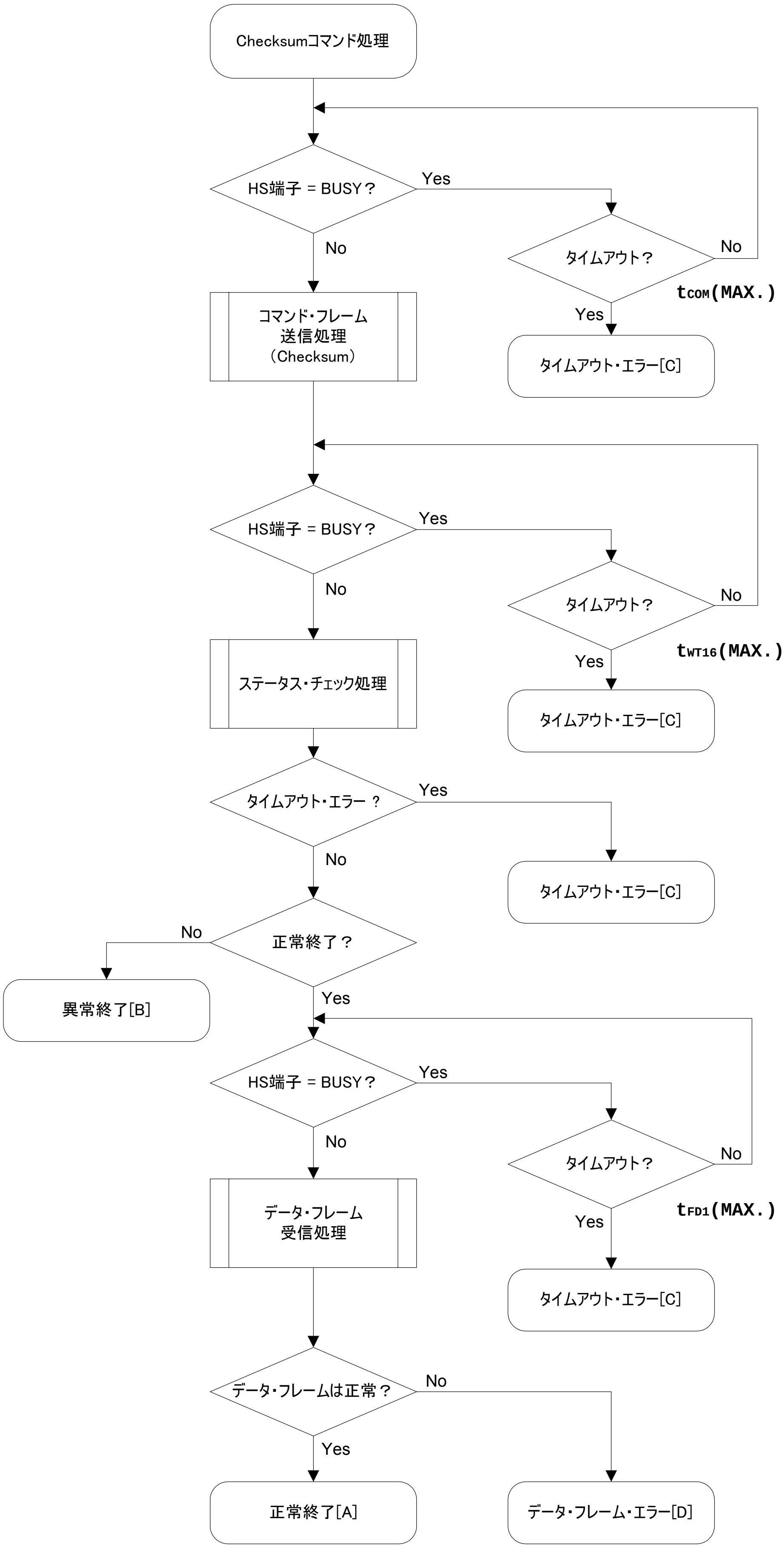
⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
　タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{FD1}(MAX.)）。
⑦ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

　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　：異常終了[D] です。

7. 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，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，2 K バイトごとの固定アドレスでありません。
	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]		−	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	データ・フレーム・エラー	−	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。

7. 14. 4　フロー・チャート



7. 14. 5　サンプル・プログラム

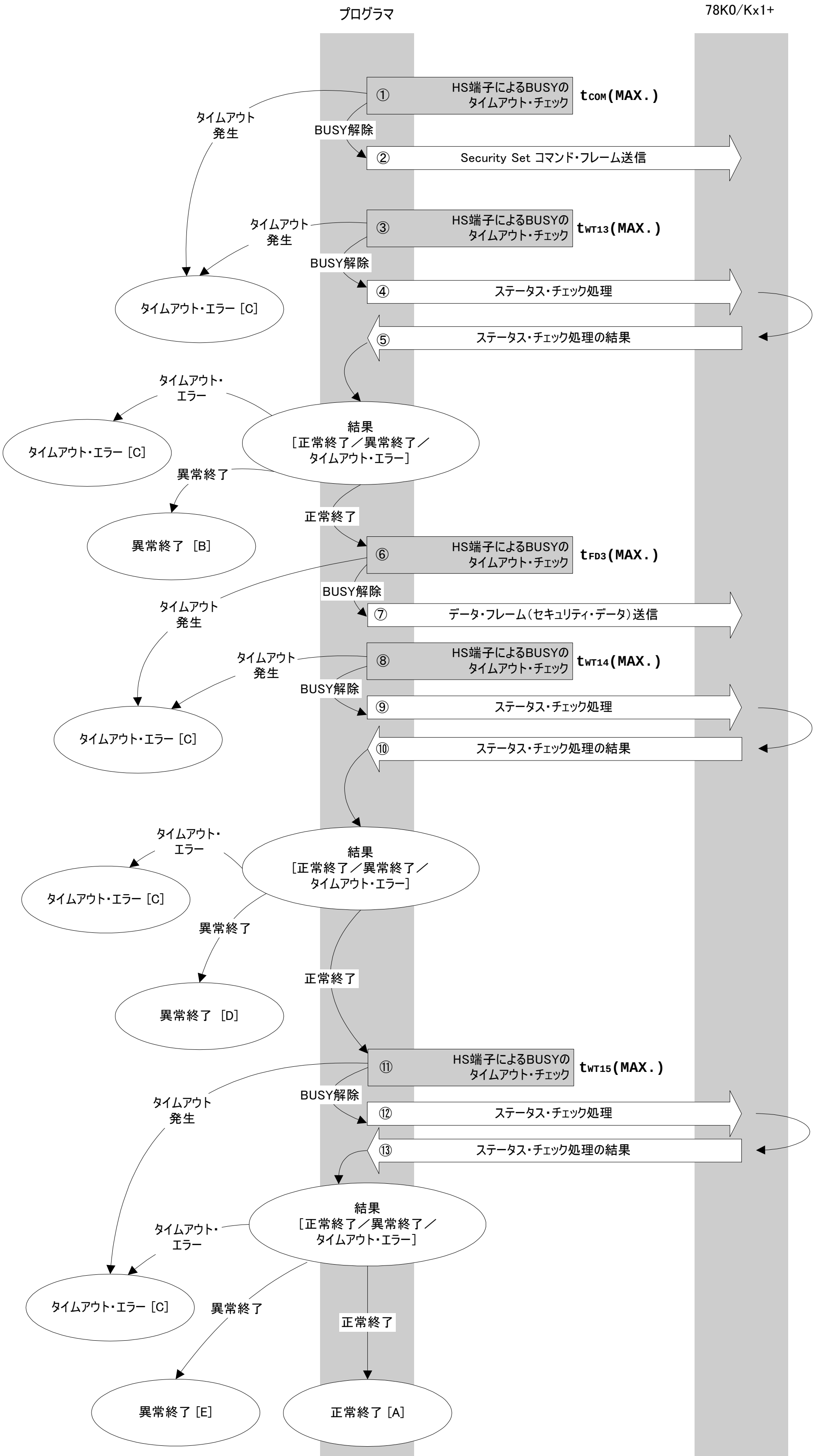
Checksumコマンド処理のサンプル・プログラムです。

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

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

   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm))
                                                 // send "Checksum" command
           return  rc;                    // error detected :case [C]

   if (hs_busy_to(tWT16_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]

   rc = fl_hs_getstatus();                // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                       break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;      break;  // case [C]
           default:                return  rc;      break;  // case [B]
   }


   if (hs_busy_to(tFD1_MAX))
           return  FLC_HSTO_ERR;          // t.o. detected :case [C]

   rc = get_dfrm_hs(fl_rxdata_frm);               // get signature data

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

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

7. 15　Security Setコマンド

7. 15. 1　処理手順チャート

Security Setコマンド処理手順



7. 15. 2　処理手順説明

① HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です（タイムアウト時間t_{COM}(MAX.)）。
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 HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です
（タイムアウト時間時間t_{WT13}(MAX.)）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

<u>正常終了</u>の場合　　　：⑥に進みます。
<u>異常終了</u>の場合　　　： 異常終了[B] です。
<u>タイムアウト・エラー</u>の場合　： タイムアウト・エラー[C] です。

⑥ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です
（タイムアウト時間時間t_{FD3}(MAX.)）。
⑦ 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。
⑧ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です
（タイムアウト時間時間t_{WT14}(MAX.)）。
⑨ ステータス・チェック処理により，ステータス・フレームを取得します。
⑩ ステータス・チェック処理の結果に応じて次の処理を行います。

<u>正常終了</u>の場合　　　：⑪に進みます。
<u>異常終了</u>の場合　　　： 異常終了[D] です。
<u>タイムアウト・エラー</u>の場合　： タイムアウト・エラー[C] です。

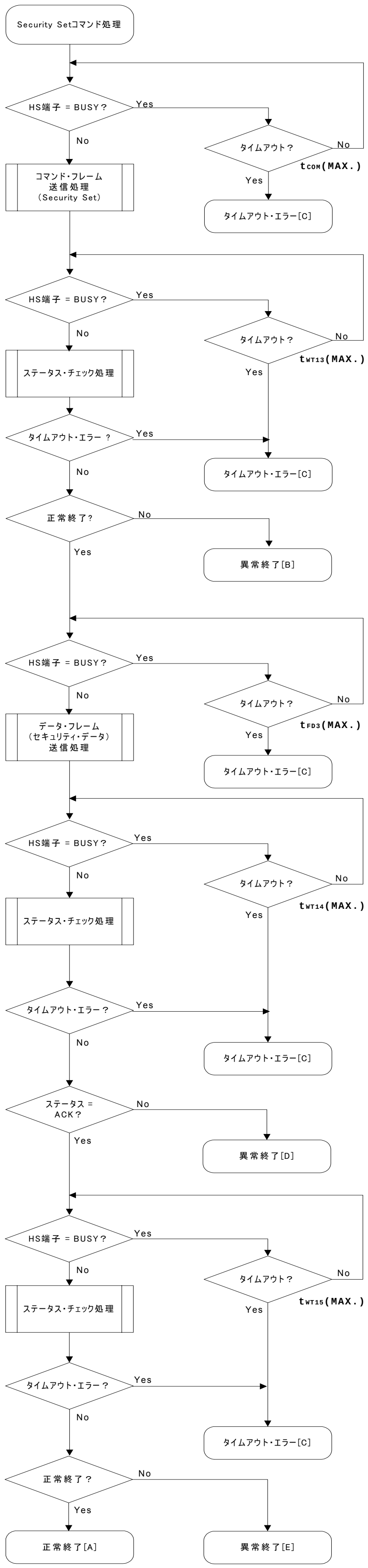
⑪ HS端子による78K0/Kx1+のBUSYチェックを行います。
タイムアウトであれば タイムアウト・エラー[C] です
（タイムアウト時間時間t_{WT15}(MAX.)）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

<u>正常終了</u>の場合　　　： 正常終了[A] です。
<u>異常終了</u>の場合　　　： 異常終了[E] です。
<u>タイムアウト・エラー</u>の場合　： タイムアウト・エラー[C] です。

7. 15. 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	BLK または PAG が 00H でありません。
	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]		－	HS 端子のビジーでタイムアウトしました。
異常終了 [D]	FLMD エラー	18H	書き込みエラーが発生しました。
	Write エラー	1CH	書き込みエラーが発生しました（すでにセキュリティ・データが設定されている場合も含む）。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

7. 15. 4　フロー・チャート



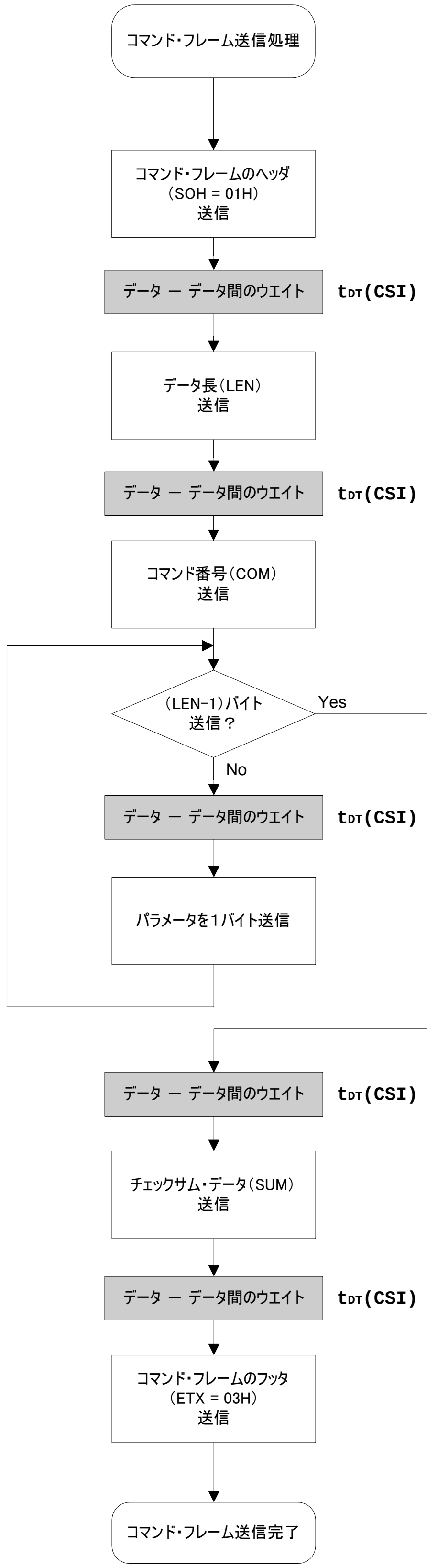
7. 15. 5　サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

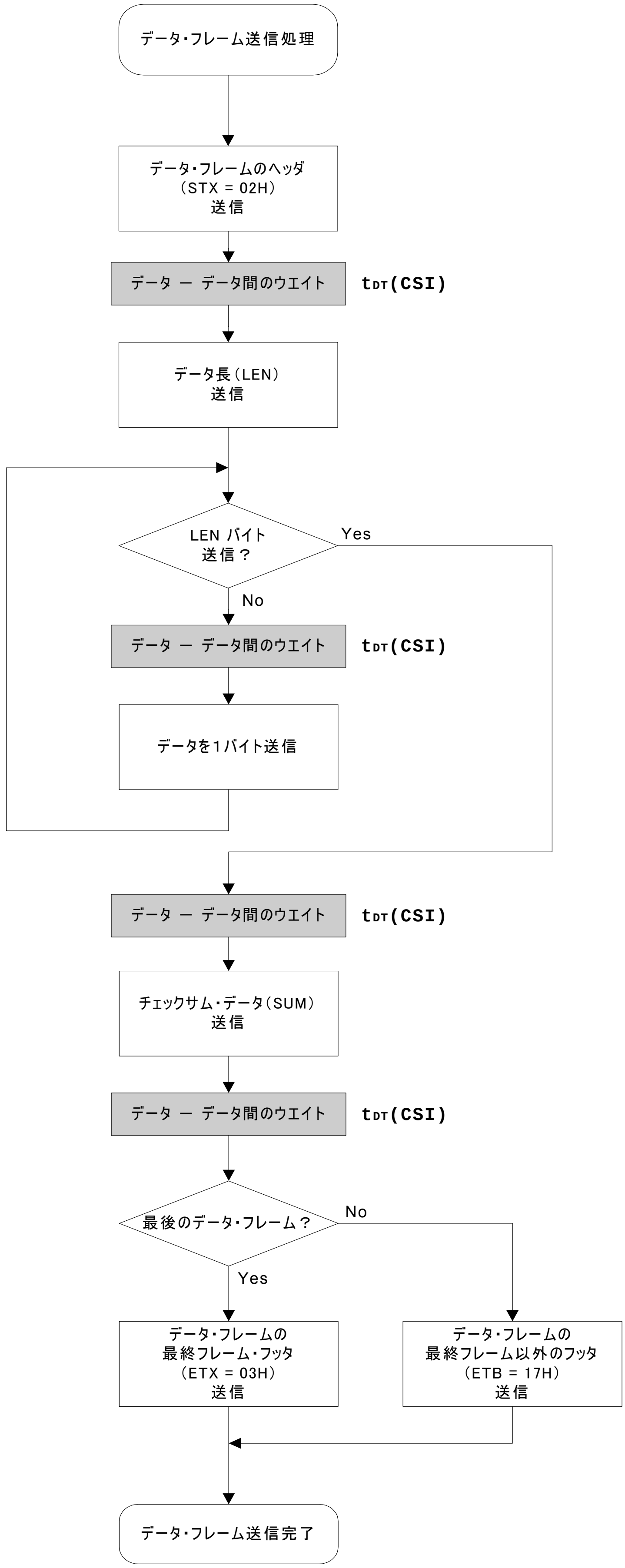
```
/******************************************************************/
/*                                                                */
/* Set security flag command (CSI-HS)                             */
/*                                                                */
/******************************************************************/
/* [i] u8 scf      ... Security flag data                         */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_hs_setscf(u8 scf)
{
   u16     rc;

   fl_cmd_prm[0] = 0x00;                 // "BLK" (must be 0x00)
   fl_cmd_prm[1] = 0x00;                 // "PAG" (must be 0x00)
   fl_txdata_frm[0] = scf|= 0b11111000;
                                     // "FLG" (upper 5bits must be '1' (to make sure))


   if (hs_busy_to(tCOM_MAX))
           return  FLC_HSTO_ERR;         // t.o. detected :case [C]

   if (rc = put_cmd_hs(FL_COM_SET_SECURITY, 3, fl_cmd_prm))
                                                  // send "Security Set" command
           return  rc;                   // error detected :case [C]

   if (hs_busy_to(tWT13_MAX))
           return  FLC_HSTO_ERR;         // t.o. detected :case [C]

   rc = fl_hs_getstatus();               // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                  break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [B]
   }


   if (hs_busy_to(tFD3_MAX))
           return  FLC_HSTO_ERR;         // t.o. detected :case [C]

   if (rc = put_dfrm_hs(1, fl_txdata_frm, true))  // send securithi setting data
           return  rc;                   // error detected :case [C]

   if (hs_busy_to(tWT14_MAX))
           return  FLC_HSTO_ERR;         // t.o. detected :case [C]

   rc = fl_hs_getstatus();               // get status frame
   switch(rc) {
           case    FLC_NO_ERR:                  break;  // continue
   //      case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
           default:                return  rc;     break;  // case [B]
   }


   if (hs_busy_to(tWT15_MAX))
           return  FLC_HSTO_ERR;         // t.o. detected

   rc = fl_hs_getstatus();               // get status frame again
// switch(rc) {
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_HSTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return  rc;
}
```

第8章　3線式シリアルI/O（CSI）通信方式

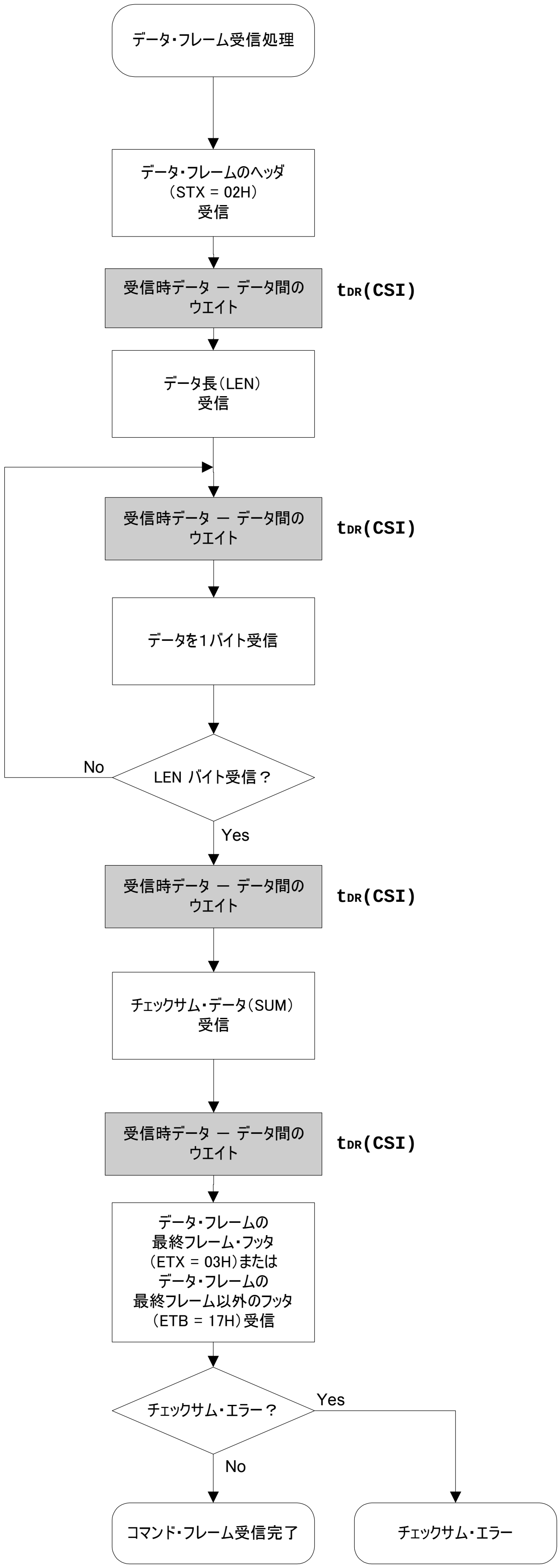
8. 1　コマンド・フレーム送信処理のフロー・チャート



8. 2　データ・フレーム送信処理のフロー・チャート



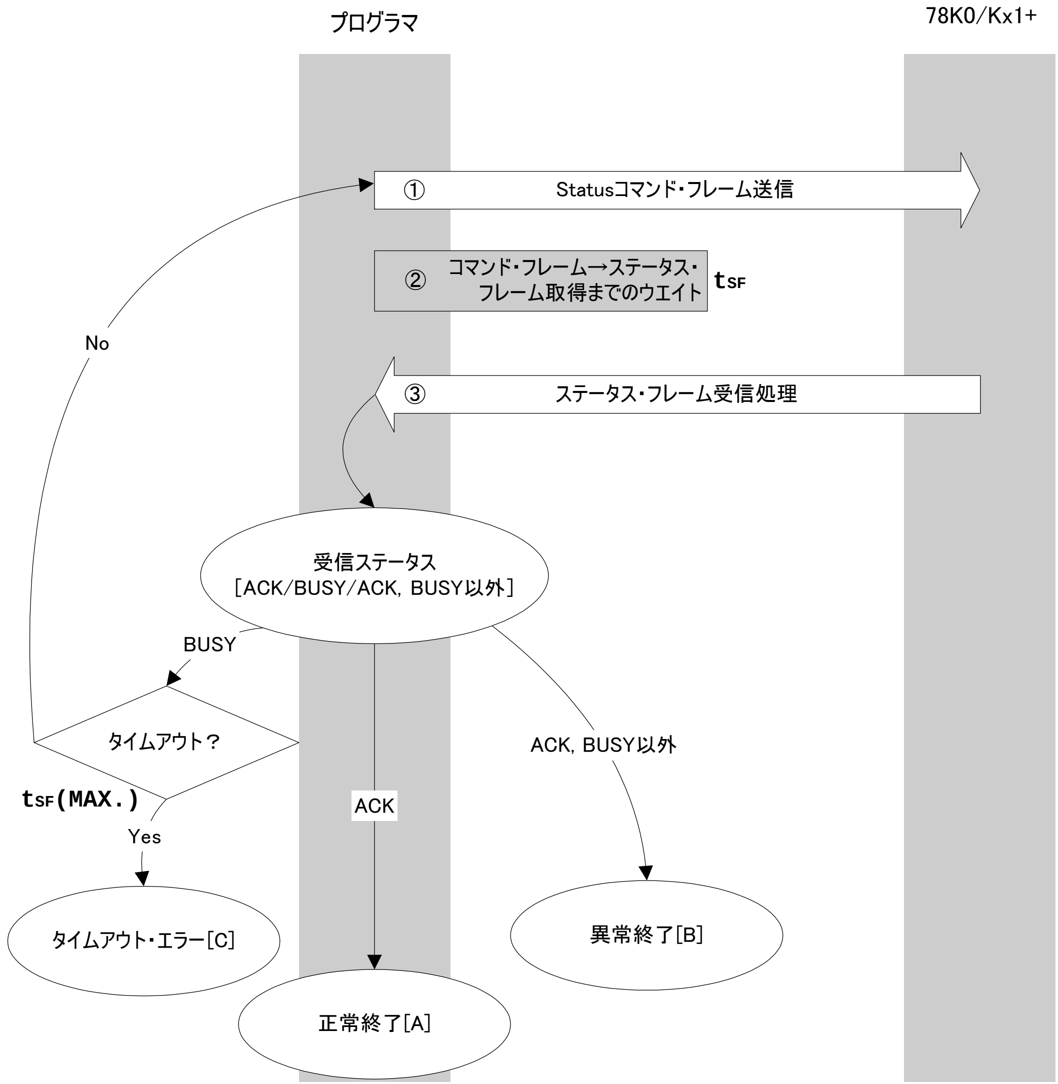
8. 3　データ・フレーム受信処理のフロー・チャート



8. 4　Statusコマンド

8. 4. 1　処理手順チャート

Statusコマンド処理手順



8. 4. 2　処理手順説明

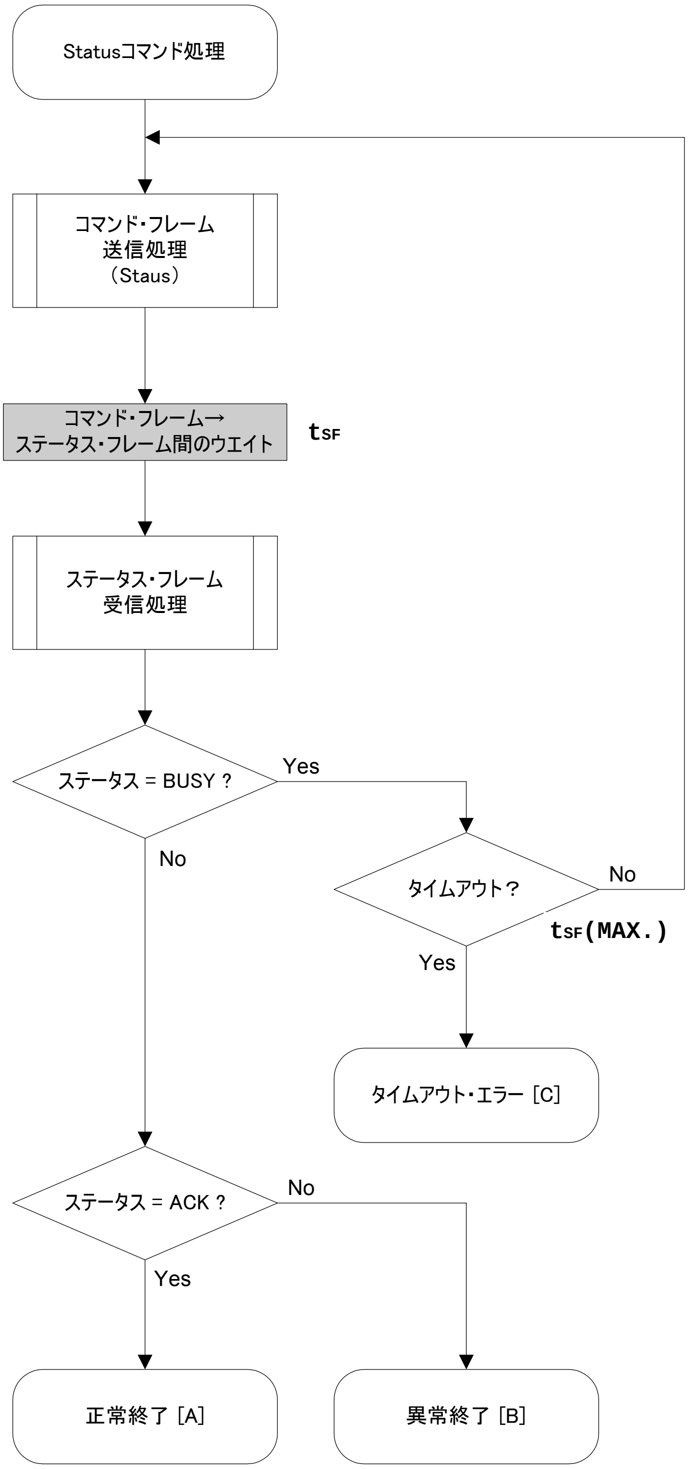
① コマンド・フレーム送信処理により，Statusコマンドを送信します。
② コマンド送信からステータス・フレーム受信までのウエイトをします（ウエイト時間t_{SF}）。
③ ステータス・コードをチェックします。

ST1 = ACKの場合　　：正常終了[A]です。
ST1 = BUSYの場合　：タイムアウト（t_{SF}(MAX.)）をチェックします。
→　タイムアウトでなければ①からやり直します。
→　タイムアウトであればタイムアウト・エラー[C]です。
ST1 = ACK, BUSY以外の場合　：異常終了[B]です。

8. 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	78K0/Kx1+からステータス・フレームを正常に受信しました。
異常終了 [B]	コマンド・エラー	04H	サポートされていないコマンド，または異常フレームを受信しました。
	パラメータ・エラー	05H	コマンド情報（パラメータ）が適切でありません。
	チェックサム・エラー	07H	プログラマから送信されたフレームのデータが異常です。
	ベリファイ・エラー	0FH	プログラマから送信されたデータとのベリファイでエラーが発生しました。
	プロテクト・エラー	10H	Security Set コマンドで禁止した処理を実行しようとしました。
	否定応答（NACK）	15H	否定応答
	FLMD エラー	18H	書き込みエラーが発生しました。
	MRG10 エラー	1AH	消去エラーが発生しました。
	MRG11 エラー	1BH	データ書き込み時に内部ベリファイ・エラーが発生，またはブランク・チェック・エラーが発生しました。
	Write エラー	1CH	書き込みエラーが発生しました。
タイムアウト・エラー [C]		－	コマンド送信後，規定の時間が経過してもビジー応答が返ってきました。

8. 4. 4　フロー・チャート


Statusコマンド処理
コマンド･フレーム
送信処理
（Staus）
コマンド･フレーム→
ステータス･フレーム間のウエイト
tSF
ステータス･フレーム
受信処理
ステータス = BUSY ?
Yes
No
タイムアウト？
No
tSF(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]

8. 4. 5　サンプル・プログラム

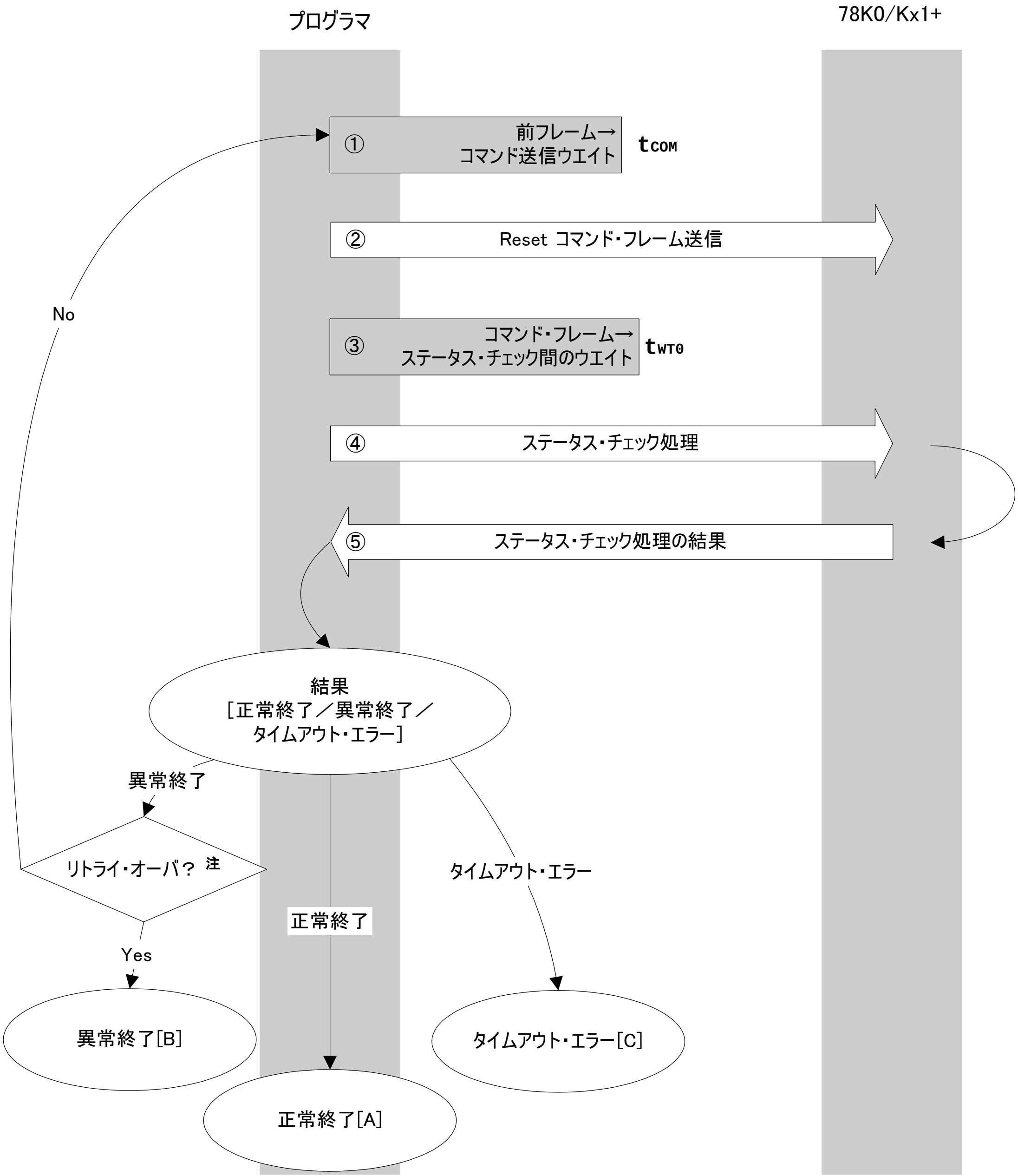
Statusコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* Get status command (CSI)                                       */
/*                                                                */
/******************************************************************/
/* [r] u16          ... decoded status or error code              */
/*                                                                */
/* (see fl.h/fl-proto.h &                                         */
/*          definition of decode_status() in fl.c)                */
/******************************************************************/
static u16 fl_csi_getstatus(void)
{
   u16     rc;

   start_flto(tSF_MAX);

   while(1){

            put_cmd_csi(FL_COM_GET_STA, 1, fl_cmd_prm);    // send "Status" command
 frame
            fl_wait(tSF);                                  // wait

            rc = get_sfrm_csi(fl_rxdata_frm);              // get status frame

            switch(rc){
                    case    FLC_BUSY:
                            if (check_flto())              // time out ?
                                    return  FLC_DFTO_ERR; //Yes,time-out//case[C]
                            continue;                      // No, retry

                    default:                               // checksum error
                            return  rc;

                    case    FLC_NO_ERR:                    // no error
                            break;

            }
            if (fl_st1 == FLST_BUSY){        // ST1 = BUSY
                    if (check_flto())                // time out ?
                            return  FLC_DFTO_ERR;    // Yes, time-out // case [C]
                    continue;                        // No, retry
            }
            break;                           // ACK or other error (but BUSY)
   }

   rc = decode_status(fl_st1);     // decode status to return code
// switch(rc) {
//
//          case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//          default:                return  rc;     break;  // case [B]
// }
   return rc;

}
```

8. 5　Resetコマンド

8. 5. 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。

8. 5. 2　処理手順説明

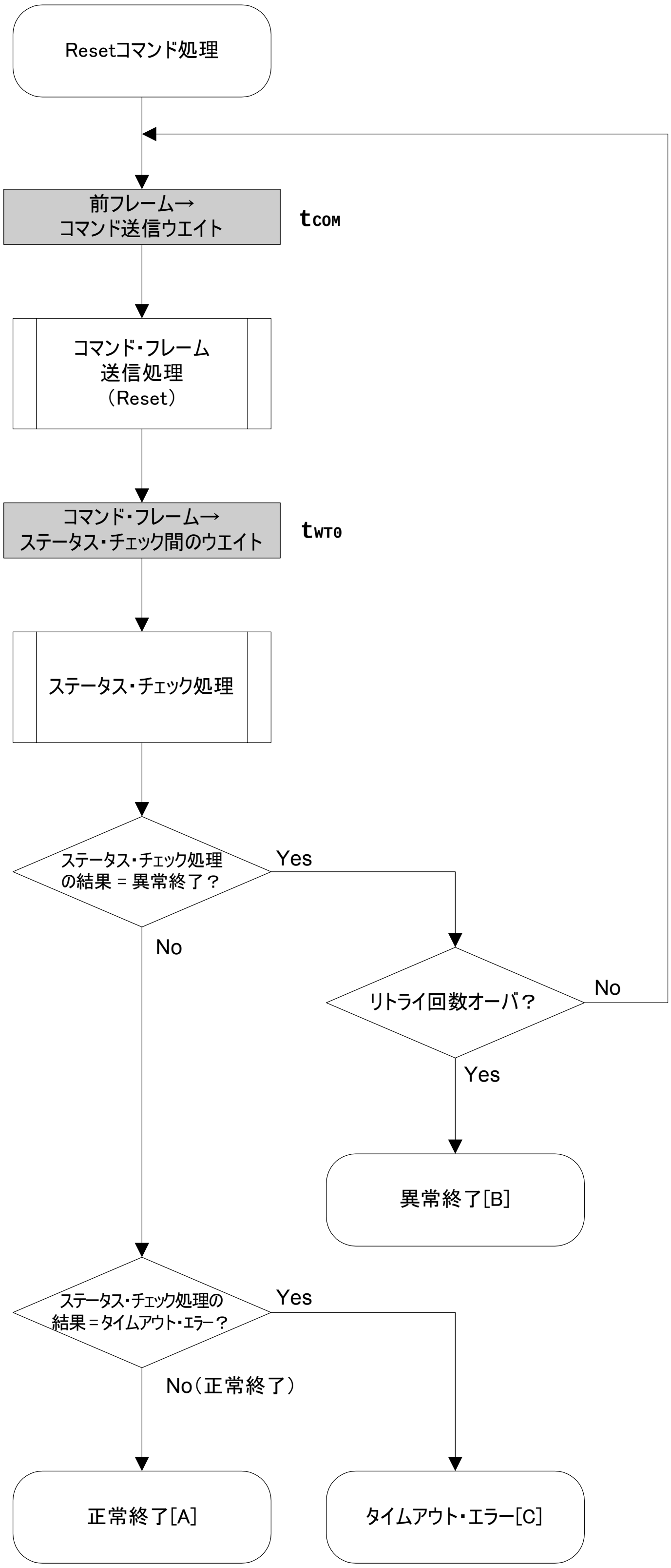
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Resetコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT0}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：正常終了[A]です。
異常終了の場合　：リトライ・オーバでなければ①より再実行します。
　　　　　　　　　リトライ・オーバであれば，異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

8. 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78K0/Kx1+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]		－	ステータス・チェック処理がタイムアウト・エラーで終了しました。

8. 5. 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
コマンド・フレーム
送信処理
（Reset）
コマンド・フレーム→
ステータス・チェック間のウエイト
tWT0
ステータス・チェック処理
ステータス・チェック処理
の結果 = 異常終了？
Yes
No
リトライ回数オーバ？
No
Yes
異常終了[B]
ステータス・チェック処理の
結果 = タイムアウト・エラー？
Yes
No（正常終了）
正常終了[A]
タイムアウト・エラー[C]

8. 5. 5　サンプル・プログラム

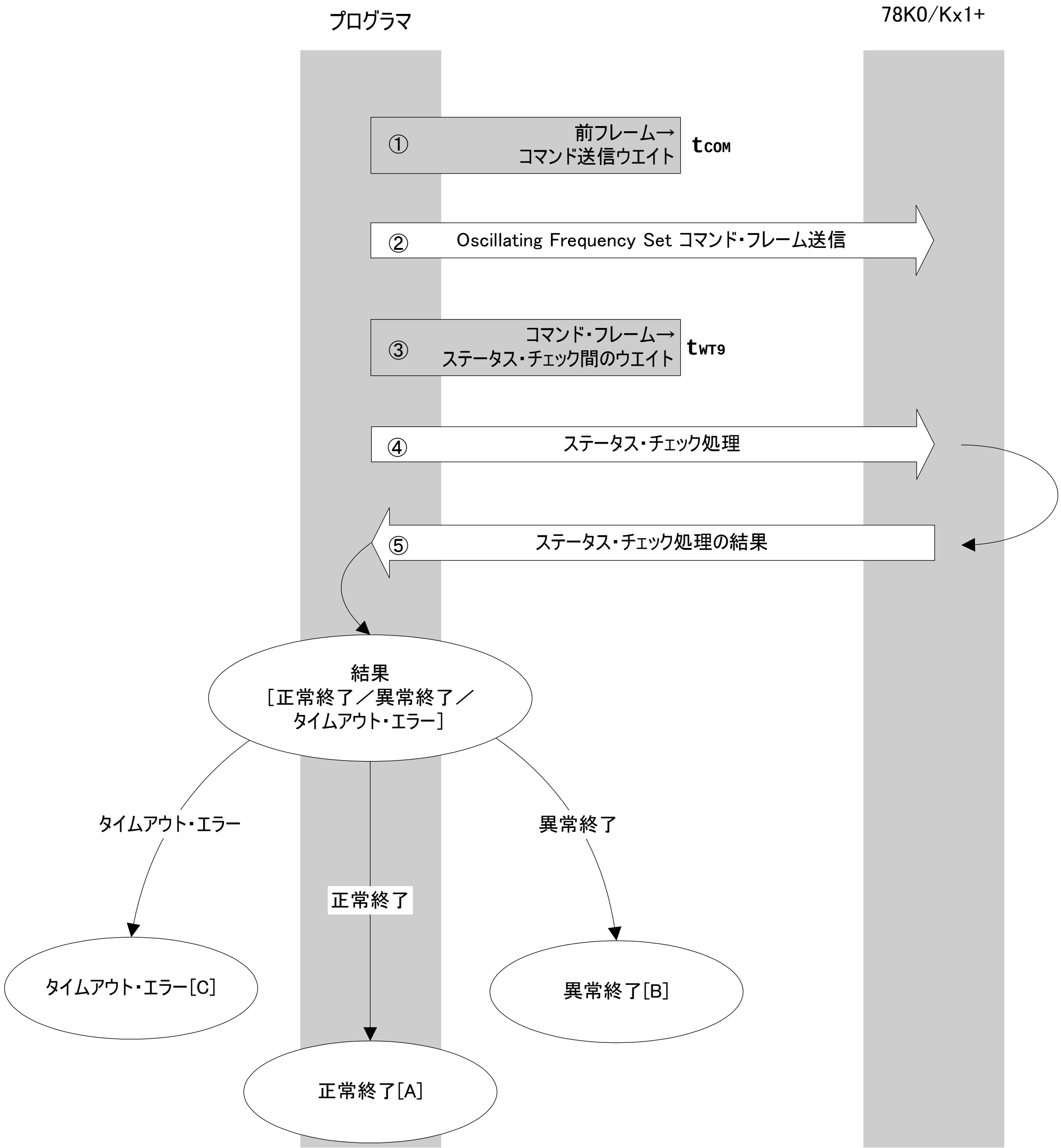
Resetコマンド処理のサンプル・プログラムです。

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

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

           fl_wait(tCOM);                      // wait before sending command frame

           put_cmd_csi(FL_COM_RESET, 1, fl_cmd_prm);   // send "Reset" command frame

           fl_wait(tWT0);

           rc = fl_csi_getstatus();        // get status

           if (rc == FLC_DFTO_ERR)         // timeout error ?
                   break;                  // yes  // case [C]
           if (rc == FLC_ACK)              // Ack ?
                   break;                  // yes  // case [A]
           //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;
}
```

8. 6　Oscillating Frequency Setコマンド

8. 6. 1　処理手順チャート

Oscillating Frequency Setコマンド処理手順



8. 6. 2　処理手順説明

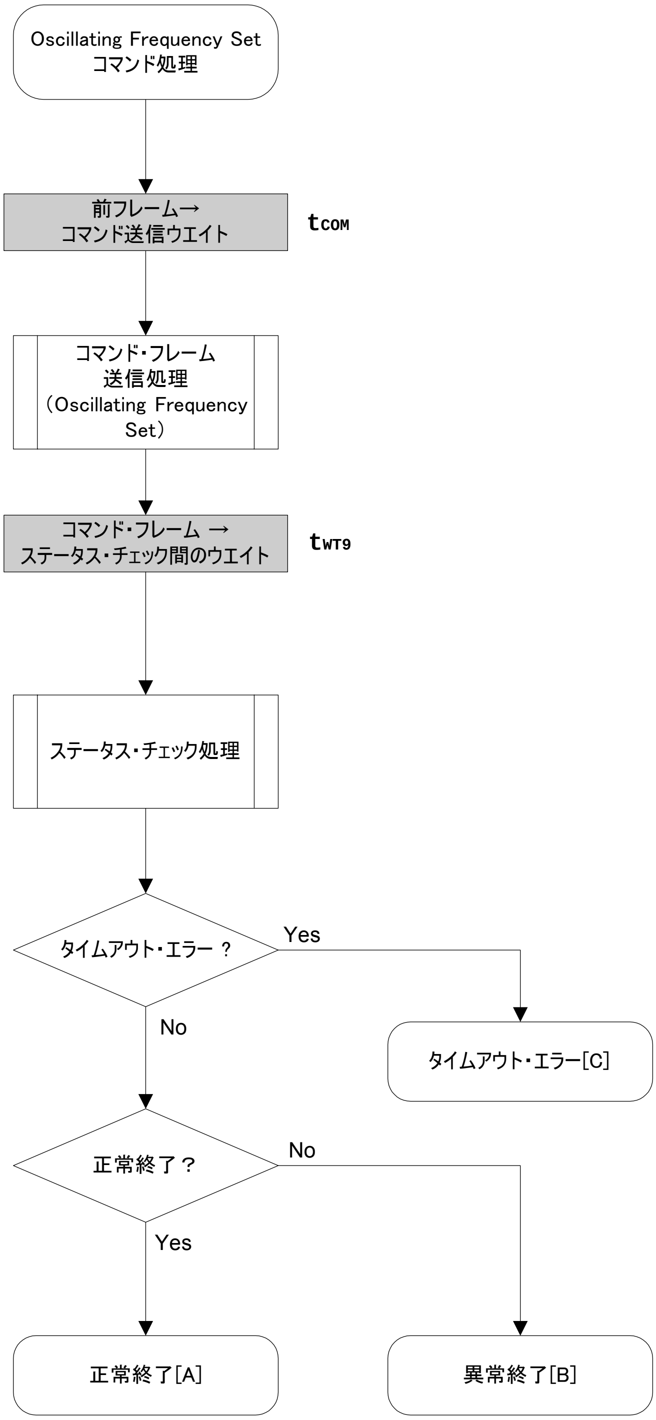
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Oscillating Frequency Setコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT9}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて以下の処理を行います。

正常終了の場合　：正常終了[A]です。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

8. 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，78K0/Kx1+に動作周波数を正しく設定できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	発振周波数値が範囲外です。
	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）。
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。

8. 6. 4　フロー・チャート


Oscillating Frequency 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
（Oscillating Frequency
Set）
コマンド・フレーム →
ステータス・チェック間のウエイト
tWT9
ステータス・チェック処理
タイムアウト・エラー ？
Yes
No
タイムアウト・エラー[C]
正常終了？
No
Yes
正常終了[A]
異常終了[B]

8.6.5　サンプル・プログラム

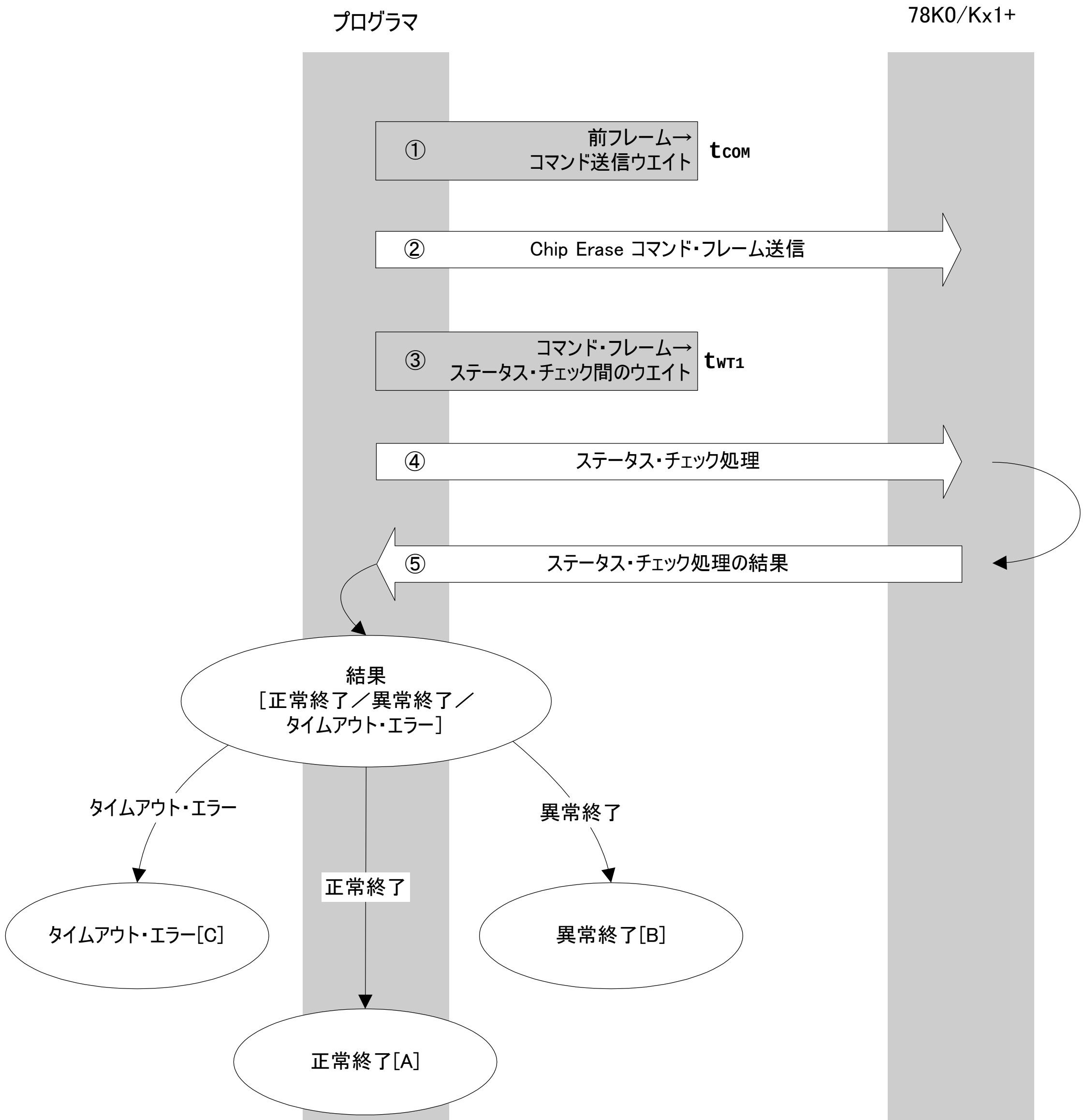
Oscillating Frequencyコマンド処理のサンプル・プログラムです。

```
/******************************************************************/
/*                                                                */
/* Set Flash device clock value command (CSI)                     */
/*                                                                */
/******************************************************************/
/* [i] u8 clk[4]   ... frequency data(D1-D4)                      */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_csi_setclk(u8 clk[])
{
   u16     rc;

   fl_cmd_prm[0] = clk[0]; // "D01"
   fl_cmd_prm[1] = clk[1]; // "D02"
   fl_cmd_prm[2] = clk[2]; // "D03"
   fl_cmd_prm[3] = clk[3]; // "D04"


   fl_wait(tCOM);                        // wait before sending command frame

   put_cmd_csi(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);
                                         // send "Oscillation Frequency Set" command

   fl_wait(tWT9);

   rc = fl_csi_getstatus();       // 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;
}
```

8. 7　Chip Eraseコマンド

8. 7. 1　処理手順チャート

Chip Eraseコマンド処理手順



8. 7. 2　処理手順説明

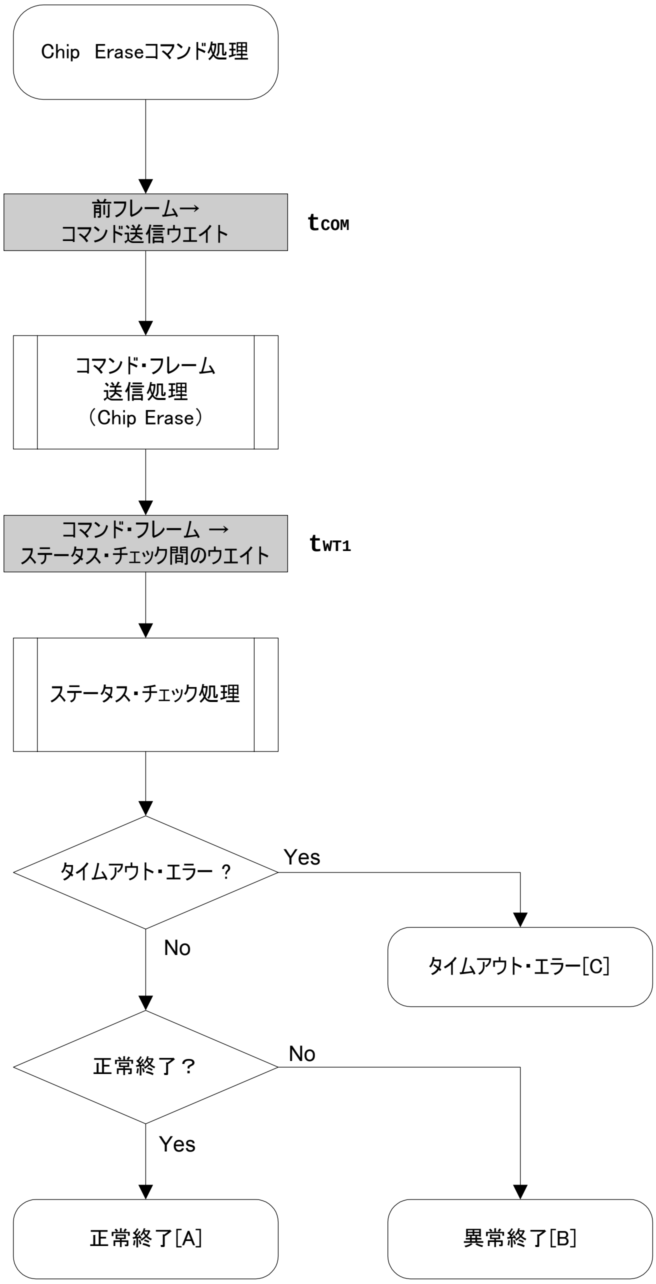
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Chip Eraseコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT1}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて以下の処理を行います。

正常終了の場合　：正常終了[A]です
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

8. 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，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」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）。
	消去エラー	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。

8. 7. 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）
コマンド・フレーム →
ステータス・チェック間のウエイト
tWT1
ステータス・チェック処理
タイムアウト・エラー ？
Yes
No
タイムアウト・エラー[C]
正常終了？
No
Yes
正常終了[A]
異常終了[B]

8.7.5　サンプル・プログラム

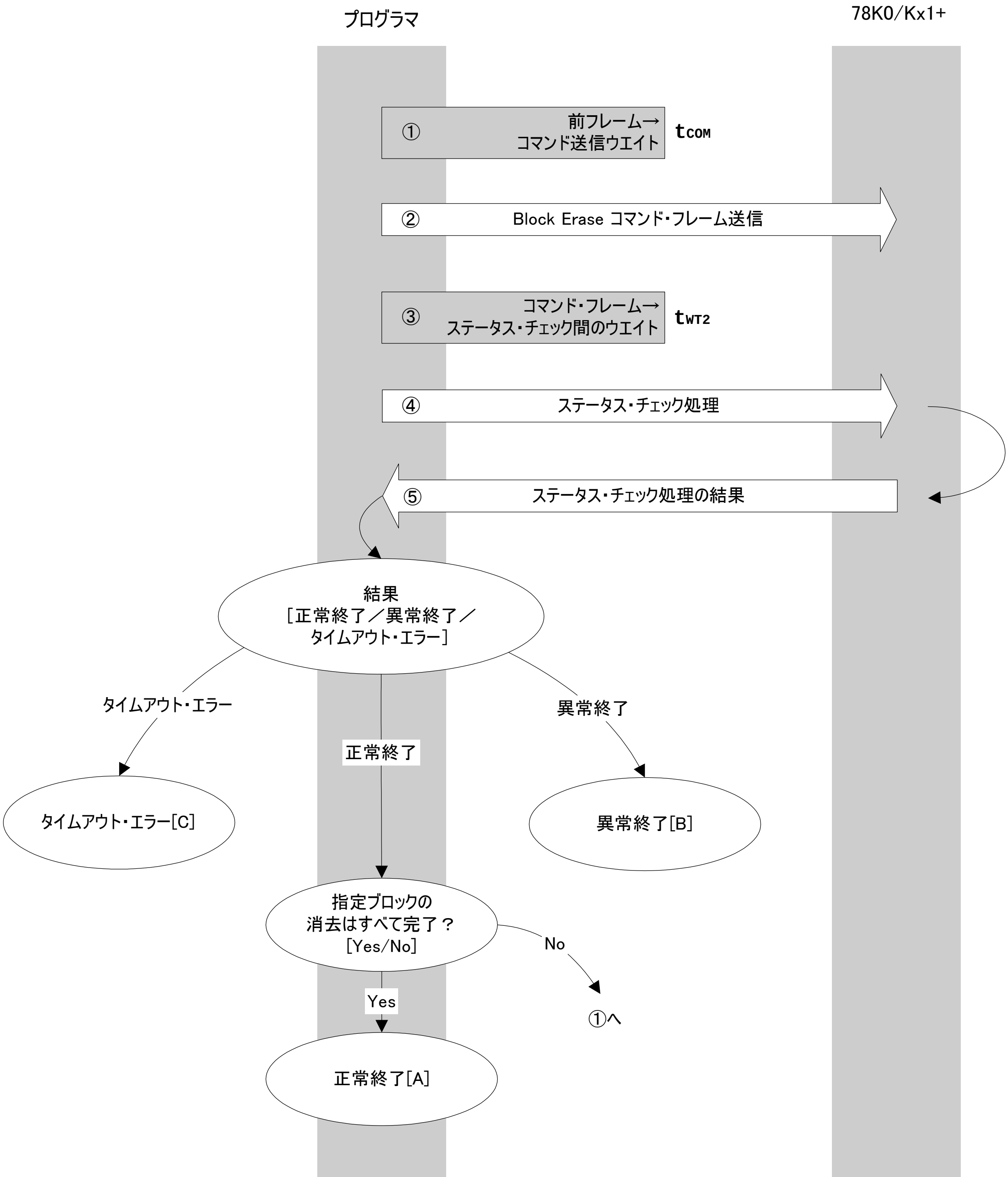
Chip Eraseコマンド処理のサンプル・プログラムです。

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


   fl_wait(tCOM);                          // wait before sending command frame

   put_cmd_csi(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send "Chip Erase" command

   fl_wait(tWT1);

   rc = fl_csi_getstatus();       // 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;

}
```

8. 8　Block Eraseコマンド

8. 8. 1　処理手順チャート

Block Eraseコマンド処理手順



8.8.2　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，Block Eraseコマンドを送信します。
③ ステータス・フレーム取得までのウエイトをします（ウエイト時間t_{WT2}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

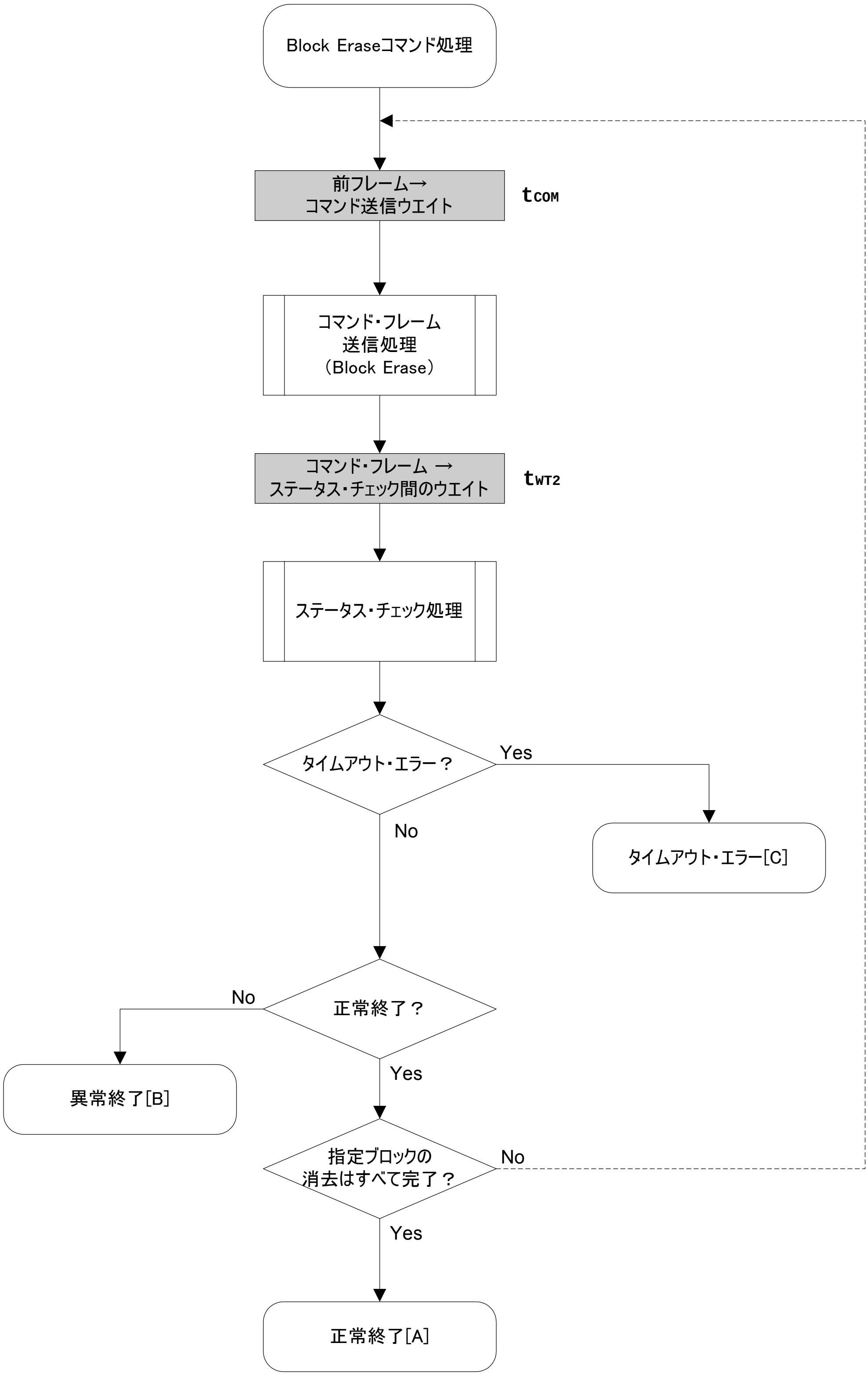
正常終了の場合：指定したブロックの消去がすべて完了していない場合は，ブロック番号を変えて①より再実行します。
指定したすべてのブロックの消去が完了した場合は，正常終了[A]です。

異常終了の場合：異常終了[B]です。
タイムアウト・エラーの場合：タイムアウト・エラー[C]です。

8.8.3　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，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	ブロック番号が範囲外です。
	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。
	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」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）。
	消去エラー	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。

8. 8. 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）
コマンド・フレーム →
ステータス・チェック間のウエイト
tWT2
ステータス・チェック処理
タイムアウト・エラー？
Yes
No
タイムアウト・エラー[C]
正常終了？
No
Yes
異常終了[B]
指定ブロックの
消去はすべて完了？
No
Yes
正常終了[A]

8.8.5　サンプル・プログラム

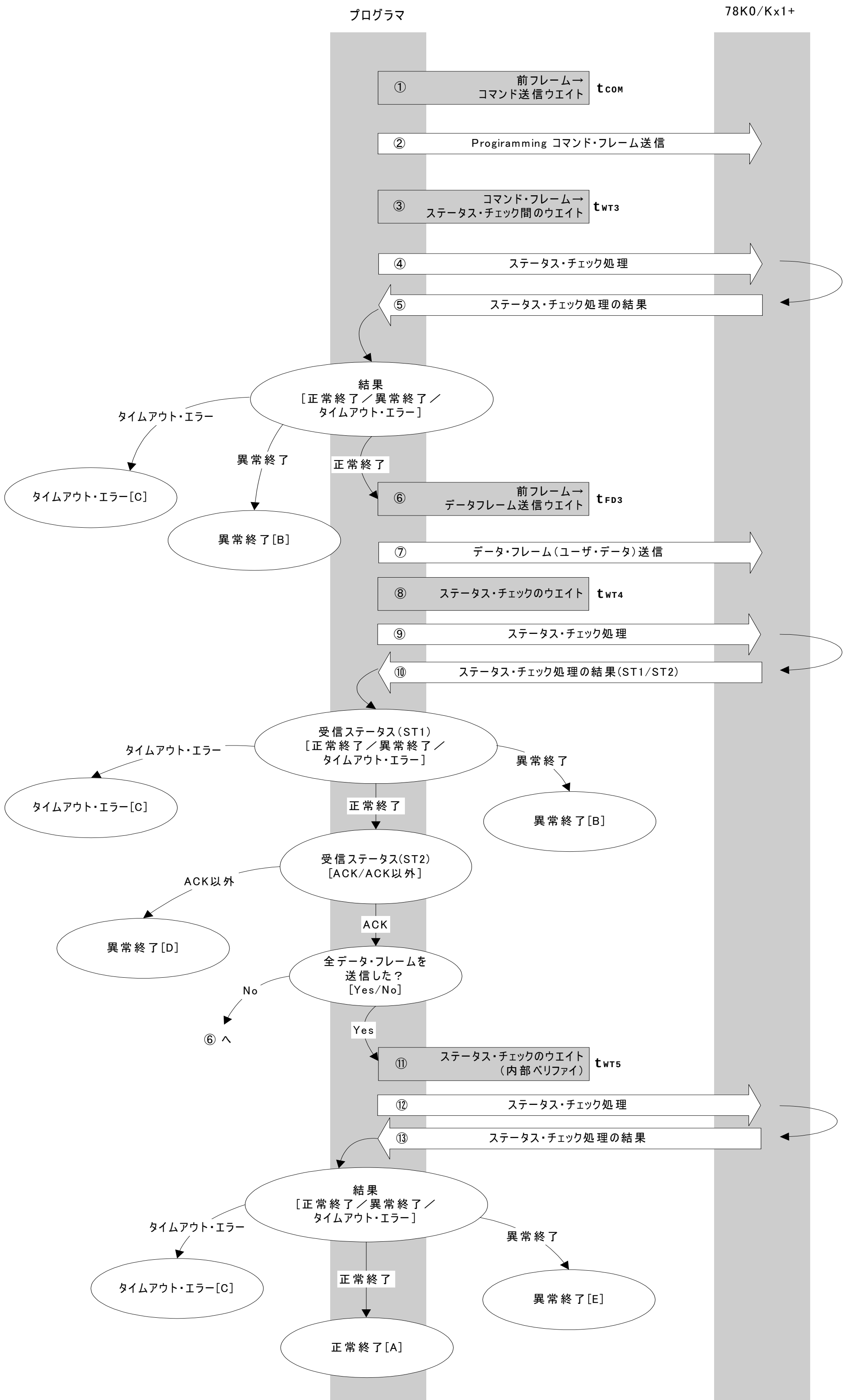
1ブロック分のBlock Eraseコマンド処理のサンプル・プログラムです。

```
/***************************************************************/
/*                                                             */
/* Erase block command (CSI)                                   */
/*                                                             */
/***************************************************************/
/* [i] u8 block    ... block number                            */
/* [r] u16         ... error code                              */
/***************************************************************/
u16        fl_csi_erase_blk(u8 block)
{
   u16     rc;

   fl_cmd_prm[0] = block;  // set params


   fl_wait(tCOM);                          // wait before sending command frame

   put_cmd_csi(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm);// send "Block Erase" command

   fl_wait(tWT2);


   rc = fl_csi_getstatus();       // 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;
}
```

8. 9　Programmingコマンド

8. 9. 1　処理手順チャート

Programmingコマンド処理手順



8.9.2　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，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・チェック処理までのウエイトをします（ウエイト時間t_{WT3}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B]です
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

⑥ データ・フレーム送信前のウエイトを行います（ウエイト時間t_{FD3}）。
⑦ データ・フレーム送信処理により，78K0/Kx1+のフラッシュROMに書き込むユーザ・データを送信します。
⑧ データ・フレーム（ユーザ・データ）送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT4}）。
⑨ ステータス・チェック処理により，ステータス・フレームを取得します。
⑩ ステータス・チェック処理の結果（ステータス・コード（ST1/ST2））に応じて次の処理を行います（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 = 異常終了の場合　：異常終了[B]です。
ST1 = タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。
ST1 = 正常終了の場合　：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以外の場合　：異常終了[D]です。
・ST2 = ACKの場合　：全ユーザ・データを送信した場合は⑪へ，まだ送信するユーザ・データがある場合は⑥から実行します。

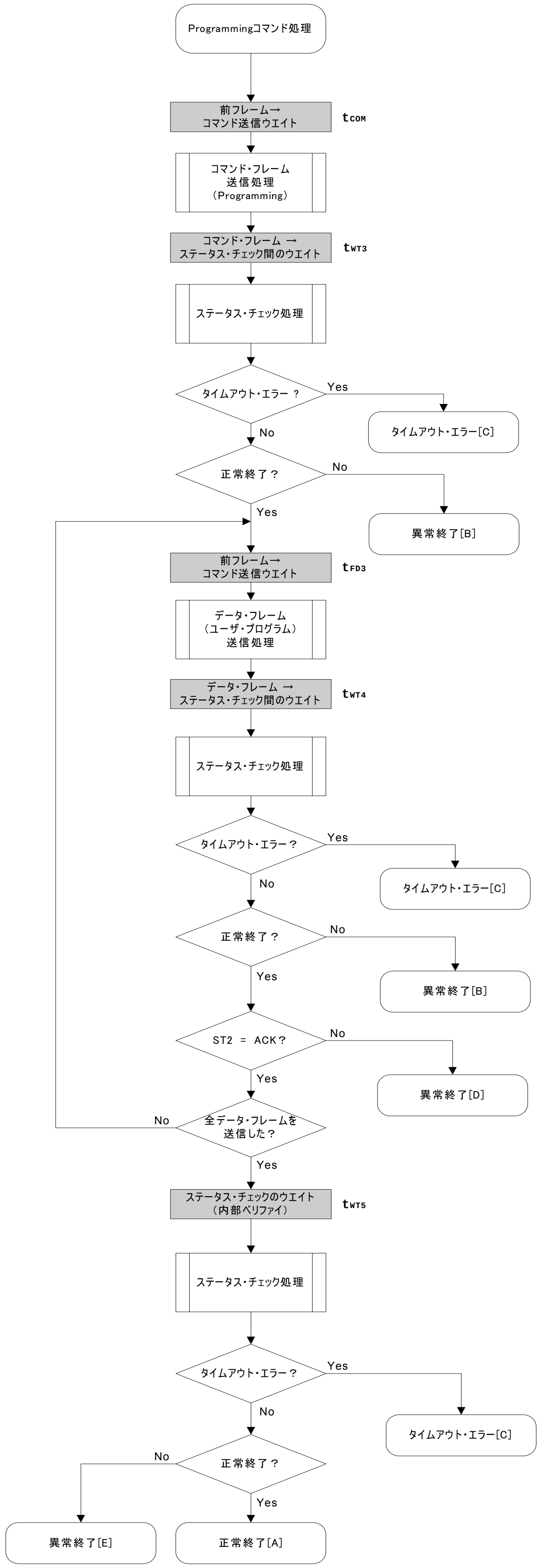
⑪ ステータス・チェック処理までのウエイトをします（ウエイト時間 t_{WT5}）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：正常終了[A]です（書き込み完了後の内部ベリファイ・チェックが正常であったことを示します）。
異常終了の場合　：異常終了[E]です（書き込み完了後の内部ベリファイ・チェックが異常であったことを示します）。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

8. 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8の倍数でありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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。
	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］	FLMDエラー	18H（ST2）	書き込みエラーが発生しました。
	Writeエラー	1CH（ST2）	書き込みエラーが発生しました。
異常終了［E］	MRG11エラー	1BH	内部ベリファイ・エラーが発生しました。

8. 9. 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
t_{COM}
コマンド・フレーム
送信処理
（Programming）
コマンド・フレーム →
ステータス・チェック間のウエイト
t_{WT3}
ステータス・チェック処理
タイムアウト・エラー ？
Yes
タイムアウト・エラー[C]
No
正常終了？
No
異常終了[B]
Yes
前フレーム→
コマンド送信ウエイト
t_{FD3}
データ・フレーム
（ユーザ・プログラム）
送信処理
データ・フレーム →
ステータス・チェック間のウエイト
t_{WT4}
ステータス・チェック処理
タイムアウト・エラー？
Yes
タイムアウト・エラー[C]
No
正常終了？
No
異常終了[B]
Yes
ST2 ＝ ACK？
No
異常終了[D]
Yes
全データ・フレームを
送信した？
No
Yes
ステータス・チェックのウエイト
（内部ベリファイ）
t_{WT5}
ステータス・チェック処理
タイムアウト・エラー？
Yes
タイムアウト・エラー[C]
No
正常終了？
No
異常終了[E]
Yes
正常終了[A]

8.9.5 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。

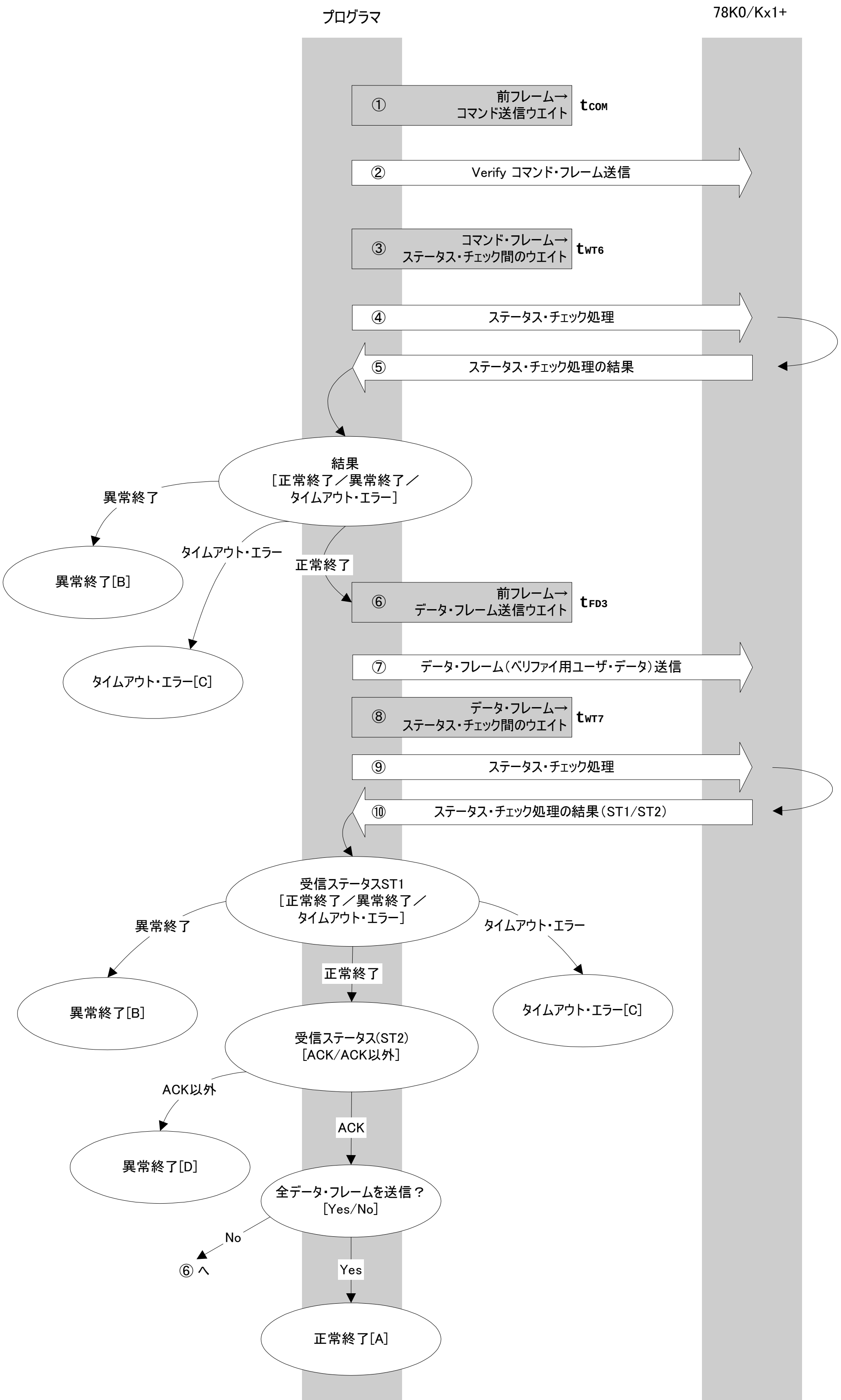
```
/******************************************************************/
/*                                                                */
/* Write command (CSI)                                            */
/*                                                                */
/******************************************************************/
/* [i] u32 top     ... start address                              */
/* [i] u32 bottom  ... end address                                */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_csi_write(u32 top, u32 bottom)
{
   u16     rc;
   u32     send_head, send_size;
   bool    is_end;
   u16     block_num;

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

   block_num = get_block_num(top, bottom); // get block num

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

   fl_wait(tCOM);
   put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm);      // send "Programming" command
   fl_wait(tWT3);

   rc = fl_csi_getstatus();                        // 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){

           if ((bottom - send_head) > 256){               // rest size > 256 ?
                   is_end = false;                         // yes, not 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_csi(send_size, fl_txdata_frm, is_end);
                                                // send data frame (user data)
        fl_wait(tWT4);                          // wait


        rc = fl_csi_getstatus();        // 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 != FLST_ACK){                // ST2 = ACK ?
                rc = decode_status(fl_st2);     // No
                return  rc;                     // case [D]
        }

        if (is_end)                     // send all user data ?
                break;                  // yes
        //continue;
   }
   /**************************************************/
   /*      Check internally verify                   */
   /**************************************************/

   fl_wait(tWT5 * block_num);                   // wait

   rc = fl_csi_getstatus();                     // 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 [E]
// }
   return  rc;

}
```

8. 10　Verifyコマンド

8. 10. 1　処理手順チャート

Verifyコマンド処理手順



8. 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･チェック処理までのウエイトをします（ウエイト時間t_{WT6}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

⑥ 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}）。
⑦ データ・フレーム送信処理により，ベリファイ用のユーザ・データを送信します。
⑧ データ・フレーム送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT7}）。
⑨ ステータス・チェック処理にてステータス・フレームを取得します。
⑩ ステータス・チェック処理の結果（受信ステータス（ST1/ST2））に応じて次の処理を行います（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）。

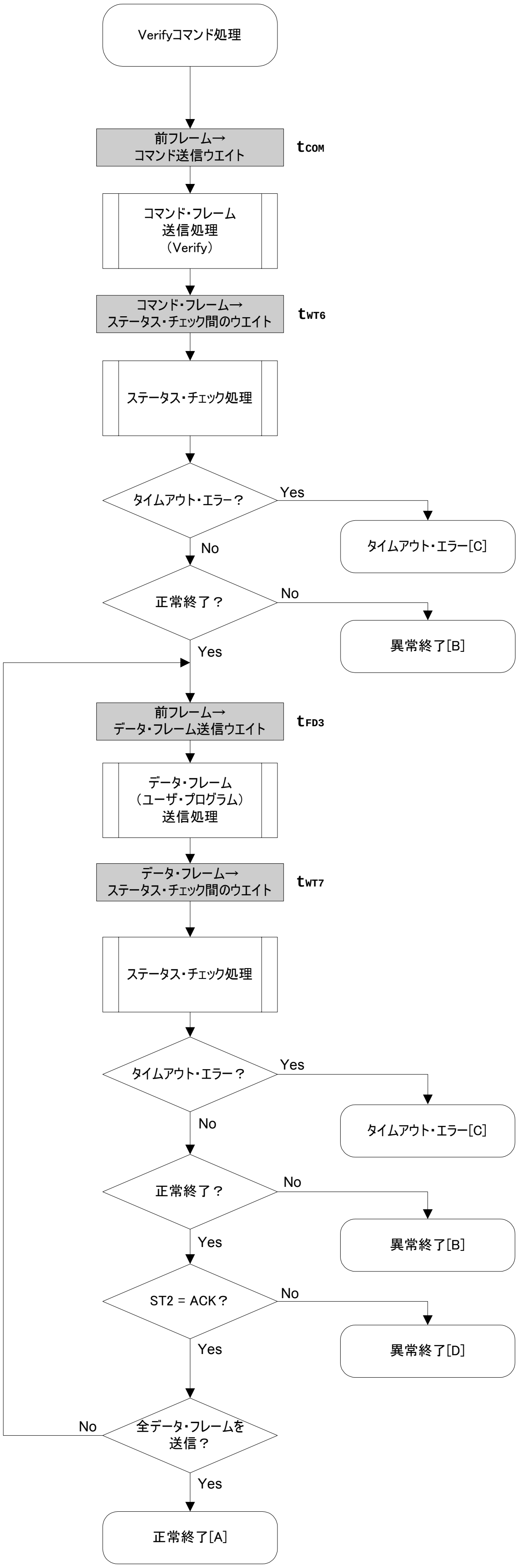
ST1 = 異常終了の場合　：異常終了[B]です。
ST1 = タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。
ST1 = 正常終了の場合　：受信ステータス（ST2）の値に応じて次の処理を行います。

- ST2 = ACK以外の場合　：異常終了[D]です。
- ST2 = ACKの場合　：全ユーザ・データを送信済みの場合は正常終了[A]です。まだ送信するユーザ・データがある場合は⑥から実行します。

8. 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，ベリファイ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。2 Kバイトごとの固定アドレスでありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム，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）	ベリファイに失敗しました。または，その他のエラーが発生しました。

8. 10. 4　フロー・チャート



8. 10. 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。

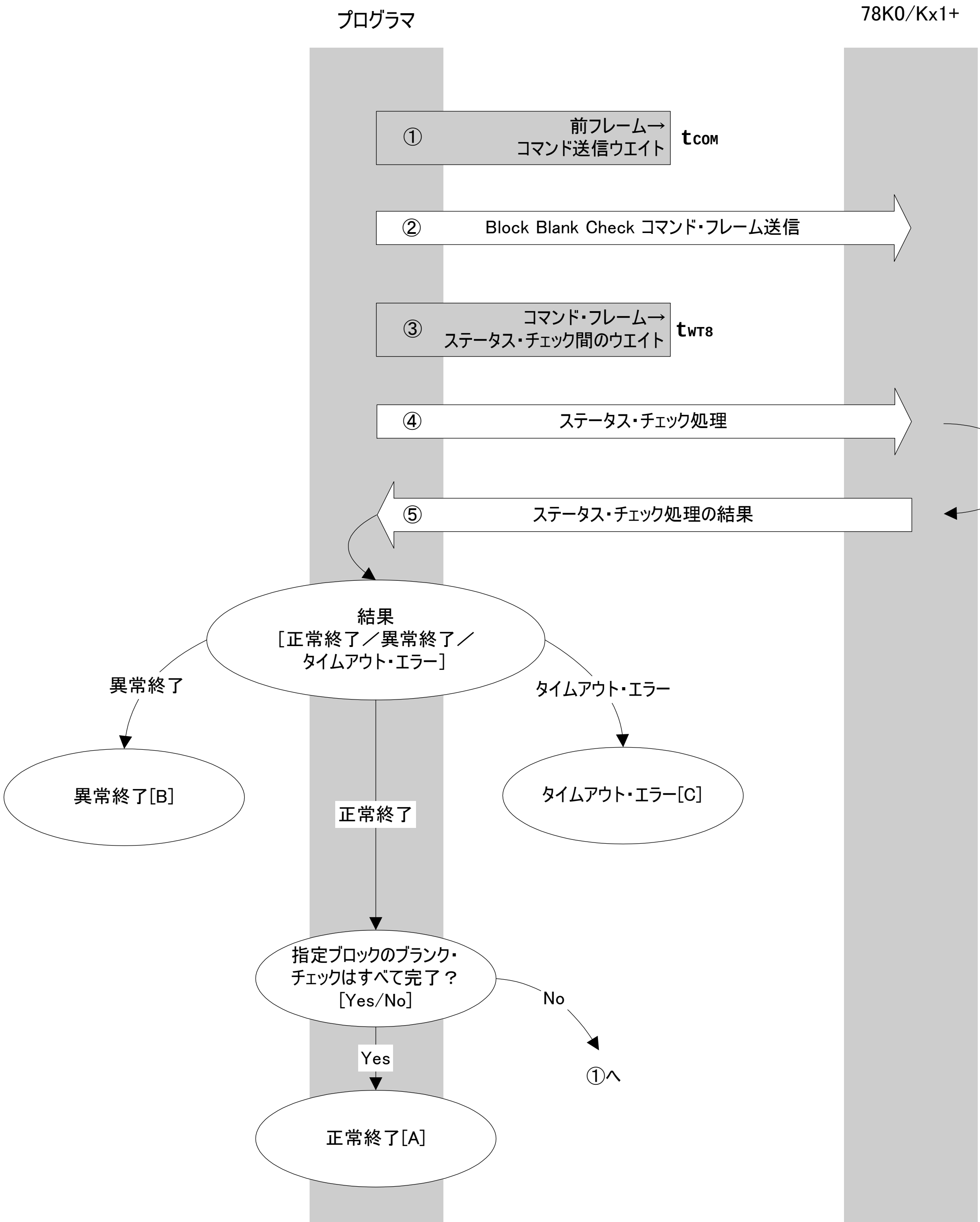
```
/******************************************************************/
/*                                                                */
/* Verify command (CSI)                                           */
/*                                                                */
/******************************************************************/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

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


   /**********************************************/
   /*      send command & check status           */
   /**********************************************/
   fl_wait(tCOM);
   put_cmd_csi(FL_COM_VERIFY, 7, fl_cmd_prm);     // send "Verify" command
   fl_wait(tWT6);

   rc = fl_csi_getstatus();                       // 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){

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

           }

           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_csi(send_size, fl_txdata_frm, is_end); // send data frame
        fl_wait(tWT7);                          // wait

        rc = fl_csi_getstatus();        // 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 != FLST_ACK){                // ST2 = ACK ?
                rc = decode_status(fl_st2);     // No
                return  rc;                     // case [D]
        }

        if (is_end)                     // send all user data ?
                break;                  // yes
        //continue;

    }
    return  FLC_NO_ERR;     // case [A]

}
```

8. 11　Block Blank Checkコマンド

8. 11. 1　処理手順チャート

Block Blank Checkコマンド処理手順



8. 11. 2　処理手順説明

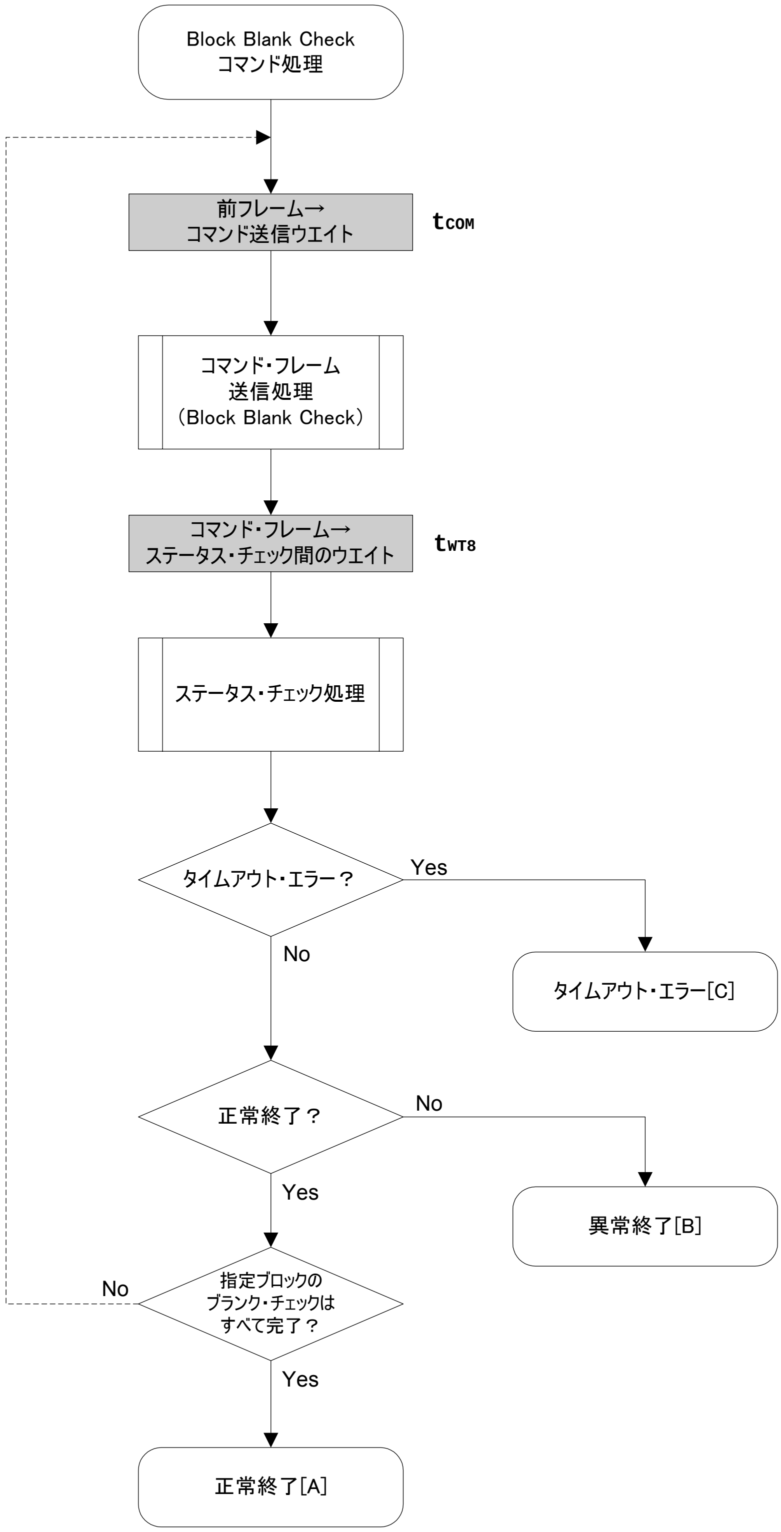
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Block Blank Checkコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT8}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。
異常終了の場合　：異常終了[B]です。
正常終了の場合　：指定したすべてのブロックのブロック・チェックが完了した場合は，正常終了[A]です。
指定したブロックのブランク・チェックがすべて完了していない場合は，ブロック番号を変えて①より再実行します。

8. 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，指定したブロックすべてがブランクであることを示します。
異常終了 [B]	パラメータ・エラー	05H	ブロック番号が範囲外です。
	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。

8. 11. 4　フロー・チャート



8. 11. 5　サンプル・プログラム

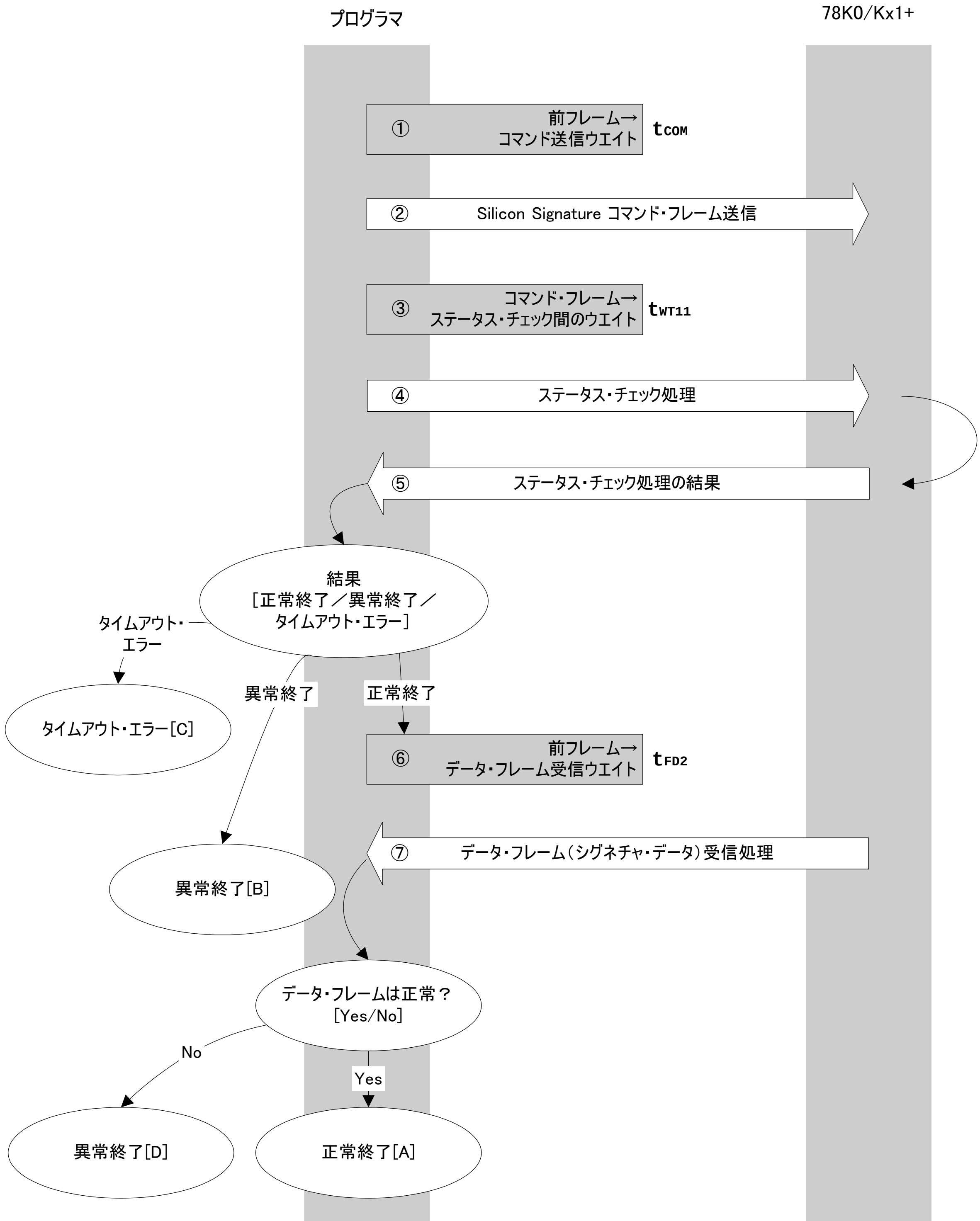
1ブロック分のBlock Blank Checkコマンド処理のサンプル・プログラムです。

```
/***************************************************************/
/*                                                             */
/* Block blank check command (CSI)                             */
/*                                                             */
/***************************************************************/
/* [i]  u8 block   ... block number                            */
/* [r] u16         ... error code                              */
/***************************************************************/
u16        fl_csi_blk_blank_chk(u8 block)
{
   u16     rc;

   fl_cmd_prm[0] = block;  // "BLK"



   fl_wait(tCOM);                         // wait before sending command frame

   put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);
                                                 // send "Block Blank Check" command

   fl_wait(tWT8);

   rc = fl_csi_getstatus();        // 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;
}
```

8. 12　Silicpn Signatureコマンド

8. 12. 1　処理手順チャート

Silicon Signatureコマンド処理手順



8. 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，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・チェック処理までのウエイトをします（ウエイト時間t_{WT11}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B] です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C] です。

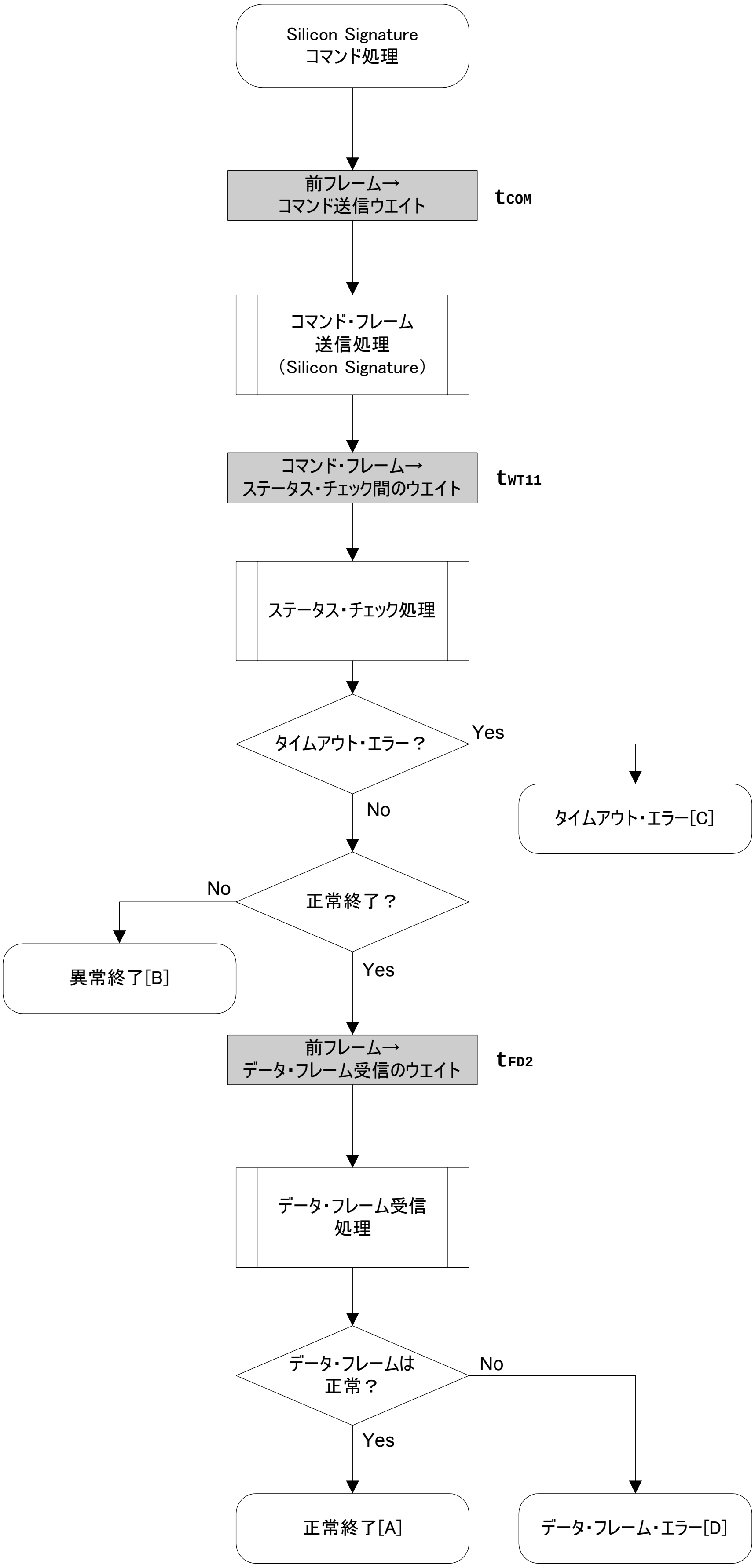
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD2}）。
⑦ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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　：異常終了[D] です。

8. 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，シリコン・シグネチャ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）。
異常終了 [C]	タイムアウト・エラー	－	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]	データ・フレーム・エラー	－	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。

8. 12. 4　フロー・チャート



8. 12. 5　サンプル・プログラム

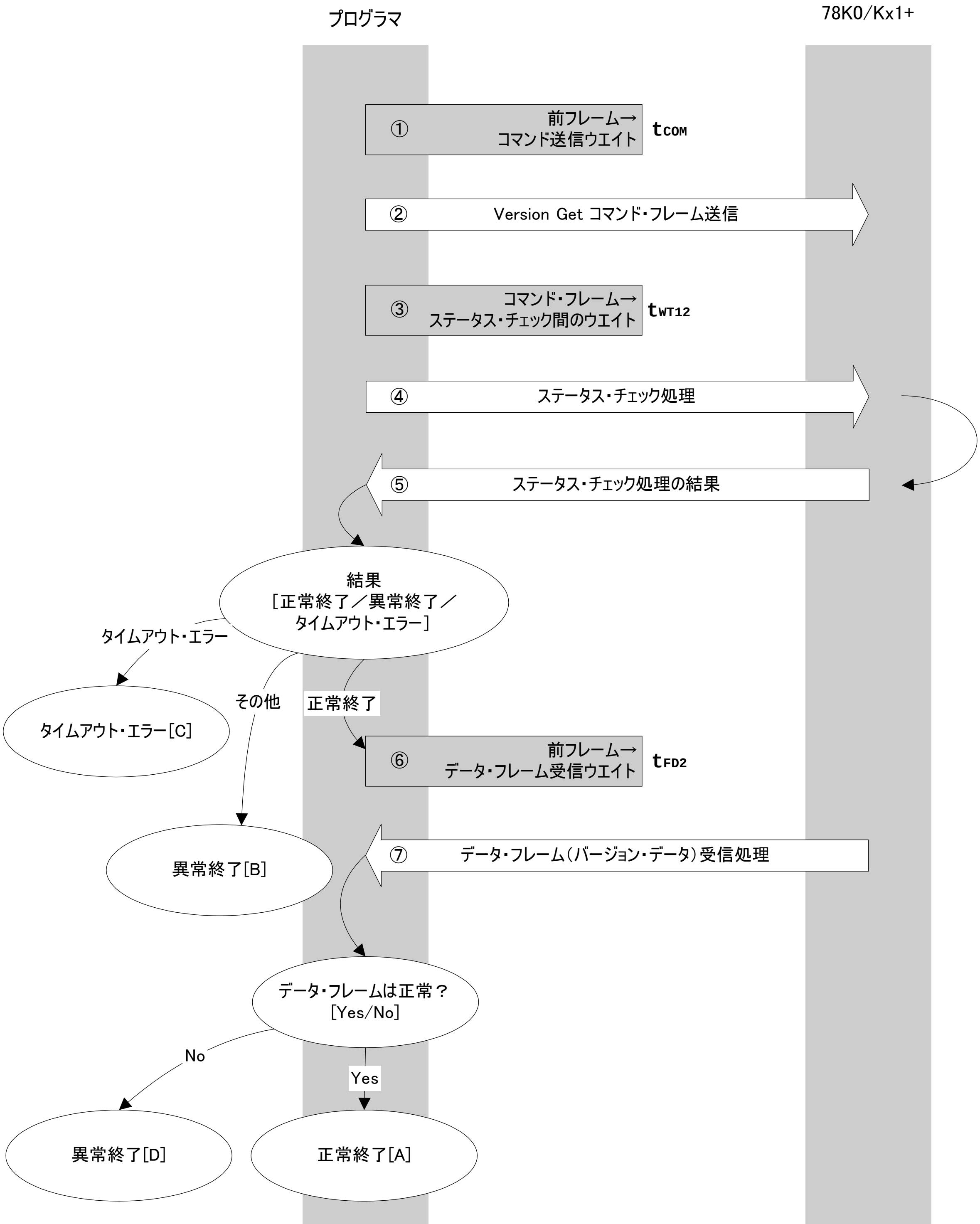
Silicon Signatureコマンド処理のサンプル・プログラムです。

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

   fl_wait(tCOM);                        // wait before sending command frame

   put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);
                                               // send "Silicon Signature" command

   fl_wait(tWT11);

   rc = fl_csi_getstatus();       // 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]
   }

   fl_wait(tFD2_SIG);                    // wait before getting data frame

   rc = get_dfrm_csi(fl_rxdata_frm);     // get data frame (signature data)

   if (rc){                                                // if no 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]
}
```

8. 13　Version Getコマンド

8. 13. 1　処理手順チャート

Version Getコマンド処理手順



8. 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，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・チェック処理までのウエイトをします（ウエイト時間t_{WT12}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果により，次の処理を行います。

正常終了の場合　　　　　：⑥に進みます。
異常終了の場合　　　　　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

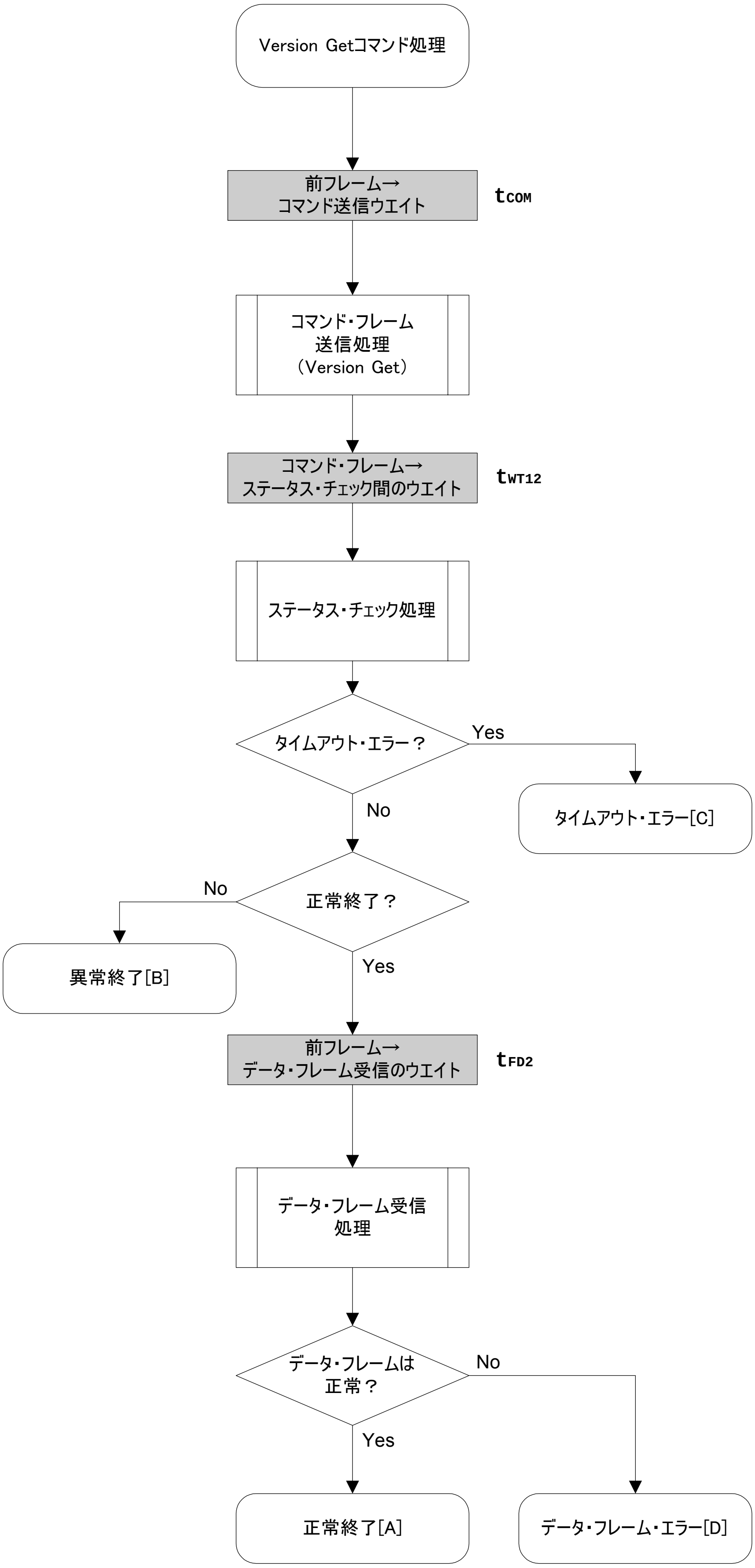
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD2}）。
⑦ 受信したデータ・フレーム（バージョン・データ）をチェックします。

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　：異常終了[D]です。

8. 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，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]		－	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]	データ・フレーム・エラー	－	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。

8. 13. 4　フロー・チャート



8. 13. 5　サンプル・プログラム

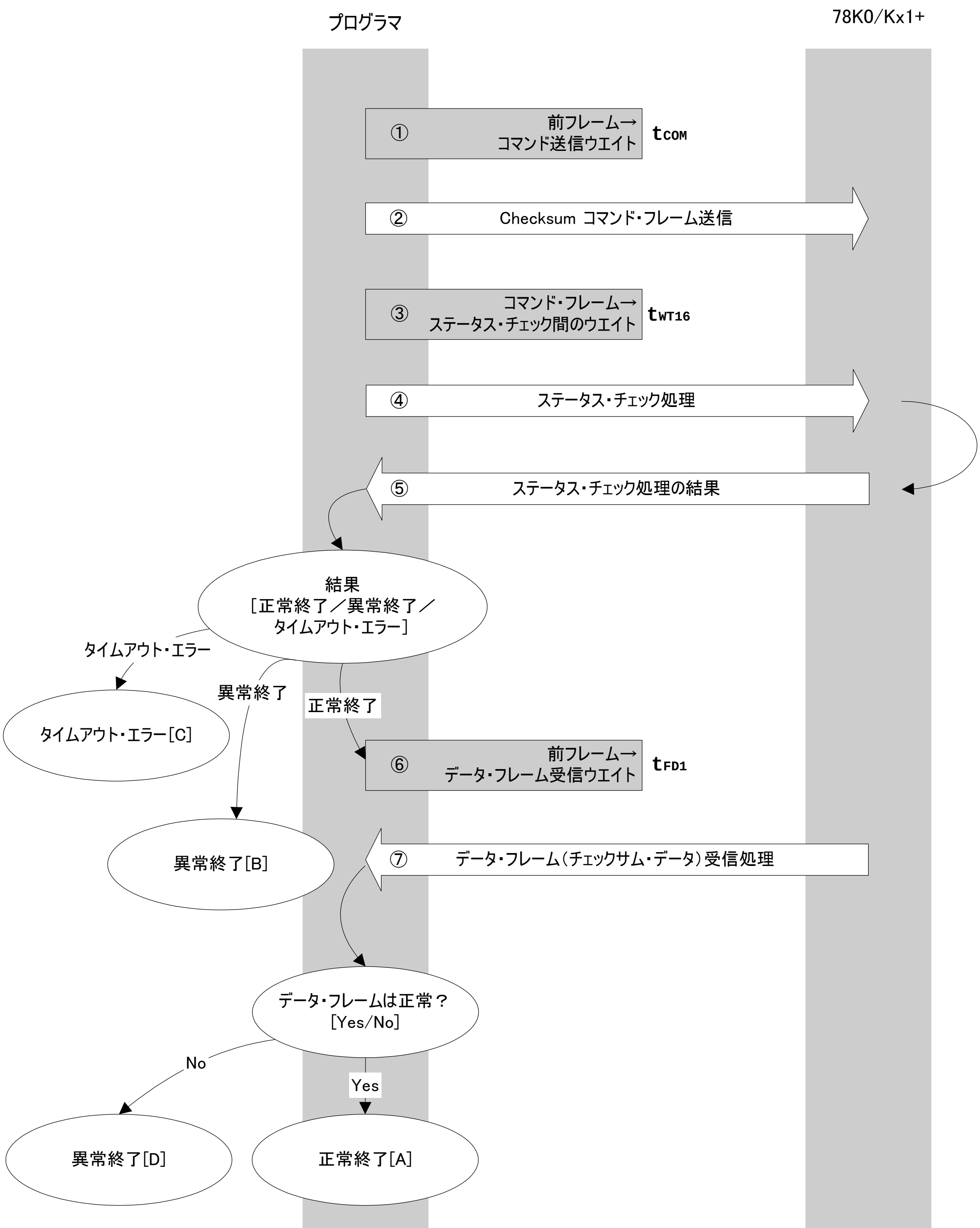
Version Getコマンド処理のサンプル・プログラムです。

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


   fl_wait(tCOM);                          // wait before sending command frame

   put_cmd_csi(FL_COM_GET_VERSION, 1, fl_cmd_prm);// send "Version Get" command

   fl_wait(tWT12);

   rc = fl_csi_getstatus();        // 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]
   }

   fl_wait(tFD2_VG);                       // wait before getting data frame

   rc = get_dfrm_csi(fl_rxdata_frm);       // get version data

   if (rc){                                                        // if no error,
           return rc;                      // case [D]
   }
   memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN);// copy version data
   return rc;                              // case [A]
}
```

8. 14　Checksumコマンド

8. 14. 1　処理手順チャート

Checksumコマンド処理手順



8. 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，Checksumコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT16}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果により，次の処理を行います。

正常終了の場合　　：⑥に進みます。
異常終了の場合　　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

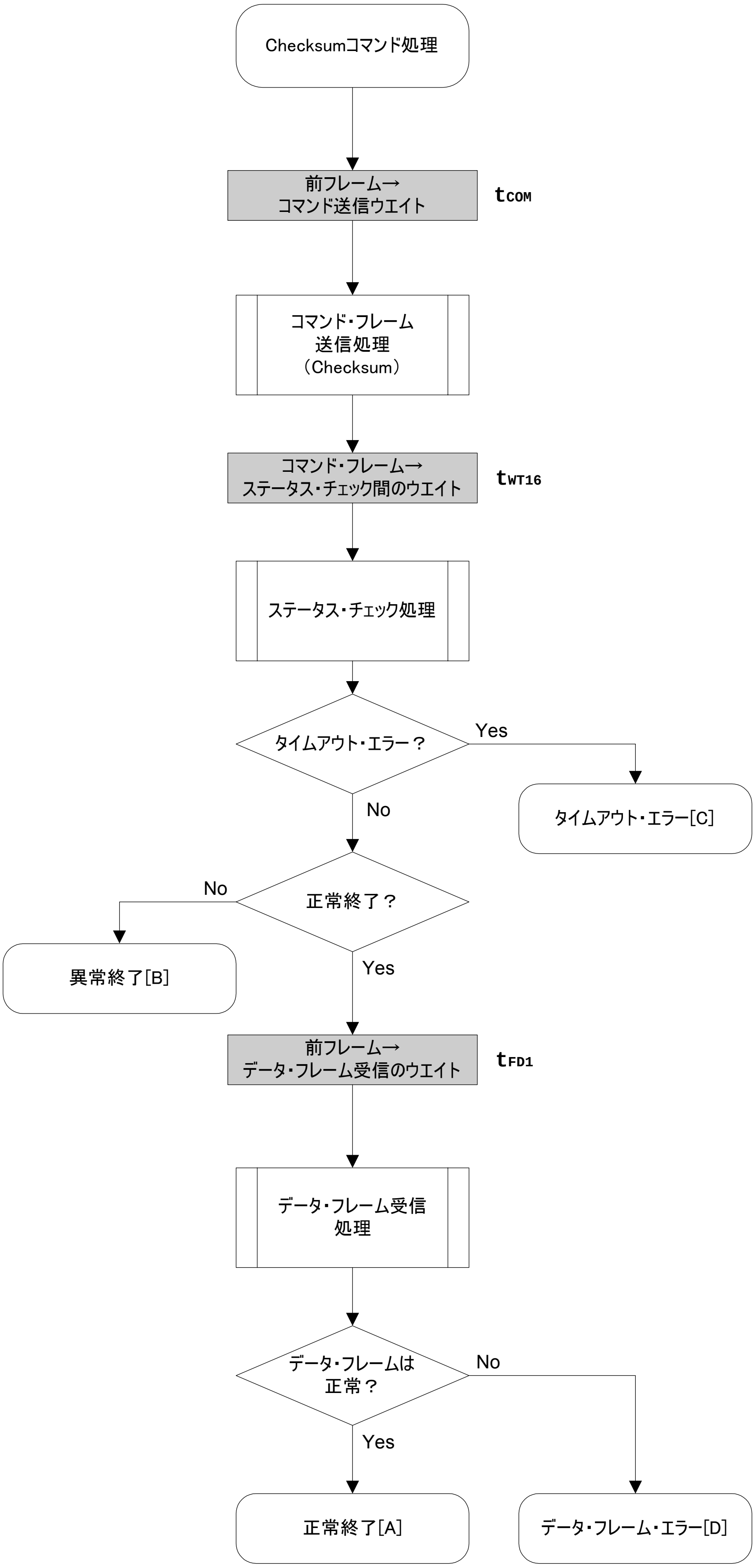
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD1}）。
⑦ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

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　：異常終了[D]です。

8. 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，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]	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２Ｋバイトごとの固定アドレスでありません。
	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]		−	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]	データ・フレーム・エラー	−	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。

8. 14. 4　フロー・チャート



8. 14. 5　サンプル・プログラム

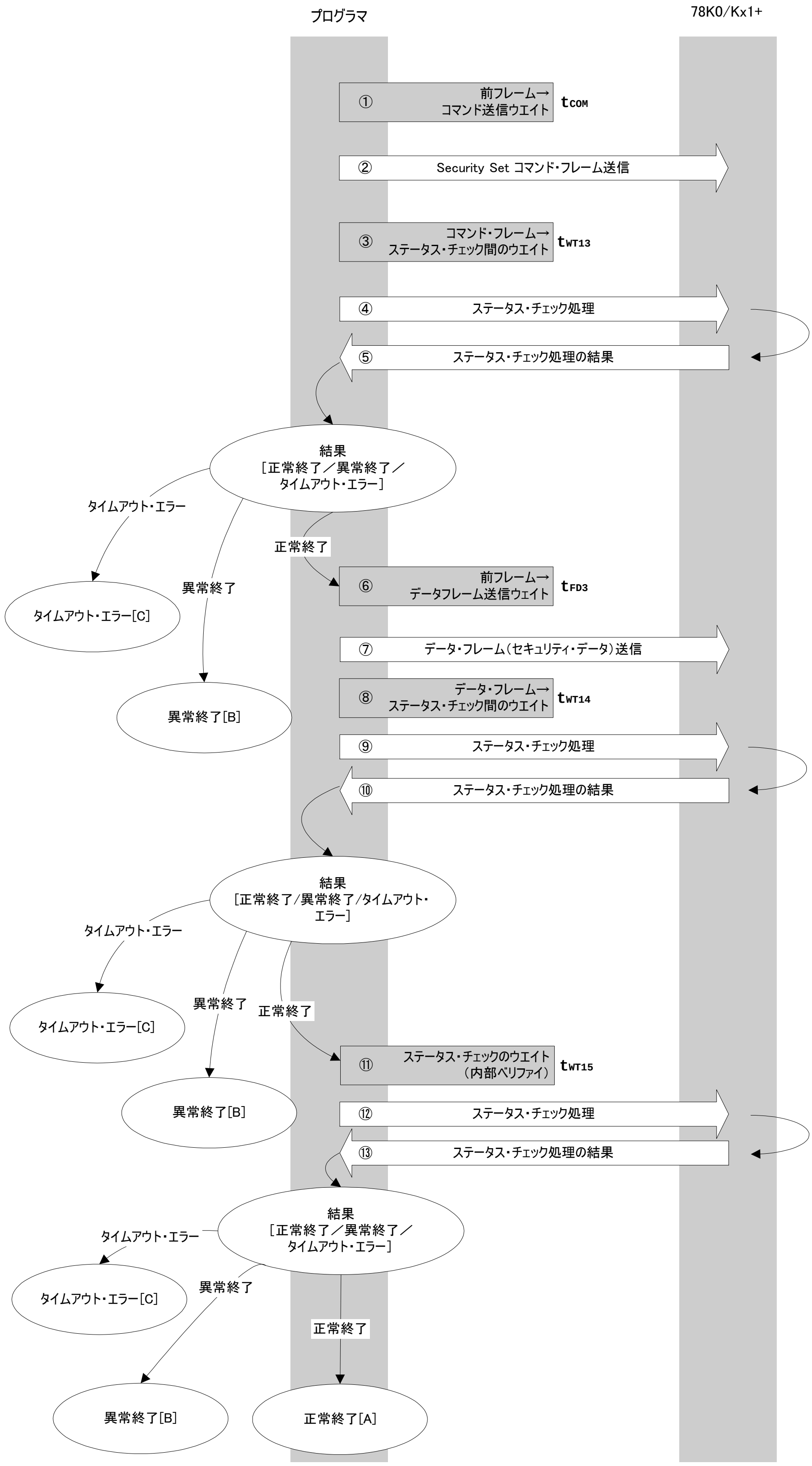
Checksumコマンド処理のサンプル・プログラムです。

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

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

   block_num = get_block_num(top, bottom); // get block num

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

   put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm);     // send "Checksum" command

   fl_wait(tWT16);

   rc = fl_csi_getstatus();        // 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)          */
   /************************************************/
   fl_wait(tFD1 * block_num);                          // wait before getting data frame

   rc = get_dfrm_csi(fl_rxdata_frm);       // get data frame(version data)

   if (rc){                                // if 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]
}
```

8. 15　Security Setコマンド

8. 15. 1　処理手順チャート

Security Setコマンド処理手順



8. 15. 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・チェック処理までのウエイトをします（ウエイト時間t_{WT13}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑥に進みます。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

⑥ 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}）。
⑦ 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。
⑧ データ送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT14}）。
⑨ ステータス・チェック処理により，ステータス・フレームを取得します。
⑩ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：⑪に進みます。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

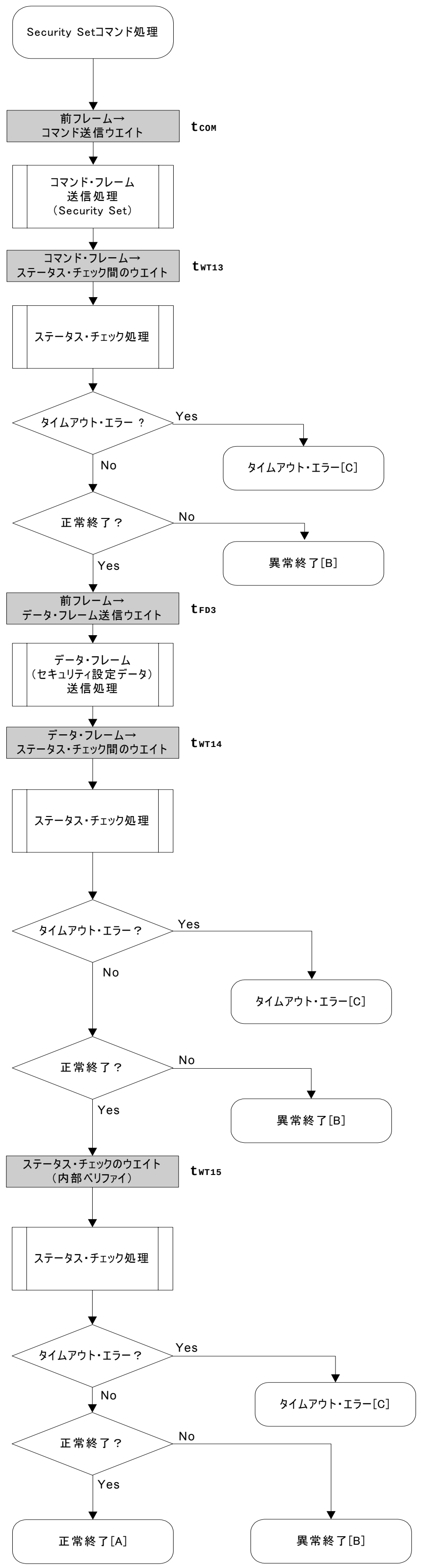
⑪ ステータス取得（内部ベリファイ完了）までのウエイトをします（ウエイト時間t_{WT15}）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：正常終了[A]です。
異常終了の場合　：異常終了[B]です。
タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。

8. 15. 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	BLK または PAG が 00H 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，またはデータ・フレームのチェックサムが異常です。
	Write エラー	1CH	すでにセキュリティ・データが設定されている，または書き込みエラーが発生しました。
	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。

8. 15. 4　フロー・チャート



8. 15. 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 (CSI)                                */
/*                                                                */
/******************************************************************/
/* [i] u8 scf      ... Security flag data                         */
/* [r] u16         ... error code                                 */
/******************************************************************/
u16        fl_csi_setscf(u8 scf)
{
   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|= 0b11111000;
                                      // "FLG" (upper 5bits must be '1' (to make sure))


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

   put_cmd_csi(FL_COM_SET_SECURITY, 3, fl_cmd_prm);   // send "Security Set" command

   fl_wait(tWT13);                // wait

   rc = fl_csi_getstatus();       // 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(tFD3);                 // wait before getting data frame

   put_dfrm_csi(1, fl_txdata_frm, true);  // send data frame(Security data)

   fl_wait(tWT14);

   rc = fl_csi_getstatus();       // 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]
   }
```

```
    /***********************************************/
    /*      Check internally verify                */
    /***********************************************/
    fl_wait(tWT15);

    rc = fl_csi_getstatus();        // 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9章　フラッシュ・メモリ・プログラミング・パラメータ特性

以降の表中のfxは78K0/Kx1+に入力されているクロック周波数（MHz）です。

9. 1　フラッシュ・メモリ・プログラミング・モード設定時間

（T_A = －10～＋65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

項　　目	略　号	MIN.	TYP.	MAX.
V_{DD}↑ to FLMD0 / 1↑	t_{DP}	10ms		
FLMD0 / 1↑ to RESET↑	t_{PR}	2ms		
Count start time from RESET↑ to FLMD0[注1]	t_{RP}	78704/fx		
Count finish time from RESET↑ to FLMD0[注1]	t_{RPE}			249952/fx
FLMD0 counter high level width / low level width	t_{PW}	10μs		100μs
Wait for Read command (CSI / CSI+HS)	t_{RC}	335934/fx		3s
Wait for low level data1 (UART)	t_{R1}			3s
Wait for low level data2 (UART)	t_{12}	30000/fx		3s
Wait for Read command (UART)	t_{2C}	30000/fx		3s
Low level data1 / data2 width[注2]	t_{L1}, t_{L2}		注2	
FLMD0 counter rise time / fall time	－			1μs

注1.　FLMD0パルスの入力タイミングは，（78704/fx＋249952/fx）/2を標準値として推奨します。

2.　ロウ・レベル幅は，9600 bps時の00Hデータ幅と同じです。

9.2　プログラミング特性

ウエイト	条件	略号	シリアルI/F	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}	CSI	176/fx	3s
			UART	176/fx	3s
	データ・フレーム送信	t_{DT}	CSI	304/fx	3s
			UART	0[注1]	3s
Status コマンド・フレーム受信～ステータス・フレーム送信	—	t_{SF}	CSI	569/fx	3s
ステータス・フレーム送信～データ・フレーム送信（1）	—	t_{FD1}[注2]	CSI	187642/fx ＋33μs	3s
			UART	0[注1]	3s
ステータス・フレーム送信～データ・フレーム送信（2）	シリコン・シグネチャ・データ	t_{FD2}	CSI	10016/fx	3s
	バージョン・データ			480/fx	3s
	—		UART	0[注1]	3s
ステータス・フレーム送信～データ・フレーム受信	—	t_{FD3}	CSI	192/fx	3s
			UART	192/fx	3s
ステータス・フレーム送信～コマンド・フレーム受信	—	t_{COM}	CSI	104/fx	3s
			UART	104/fx	3s

注1.　プログラマが連続受信可に設定されている場合

2.　1ブロックの場合の時間

備考　ウエイトには，次のような定義があります。

＜t_{DR}, t_{FD3}, t_{COM}＞

78K0/Kx1+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～MAX時間内に次データの送信を行ってください。

＜t_{DT}, t_{SF}, t_{FD1}, t_{FD2}＞

78K0/Kx1+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～MAX時間内に次のデータの受信を行ってください。

コマンド	略号	シリアル I/F	MIN.	MAX.
Reset	t_{WT0}	CSI	304/fx	3s
		UART	注 1	3s
Chip Erase	t_{WT1}	—	【78K0/KB1+】 1444656/fx＋12.1ms	【78K0/KB1+】 369712596/fx＋3089ms
			【78K0/KC1+】 1866544/fx＋12.1ms	【78K0/KC1+】 477715924/fx＋3089ms
			【78K0/KD1+, 78K0/KE1+, 78K0/KF1+】 3343152/fx＋12.1ms	【78K0/KD1+, 78K0/KE1+, 78K0/KF1+】 855727572/fx＋3089ms
Brock Erase	t_{WT2} 注 2	—	147184/fx＋12.1ms	32733379/fx＋3089ms
Programming	t_{WT3}	CSI	1488/fx＋27μs	3s
		UART	注 1	3s
	t_{WT4} 注 3	—	81600/fx＋25ms	674240/fx＋274ms
	t_{WT5} 注 2	CSI	363546/fx＋24579μs	436256/fx＋29495μs
		UART	注 1	436256/fx＋29495μs
Verify	t_{WT6}	CSI	1008/fx	3s
		UART	注 1	3s
	t_{WT7} 注 3	CSI	20368/fx＋27μs	3s
		UART	注 1	3s
Block Blank Check	t_{WT8} 注 2	CSI	132368/fx＋27μs	158842/fx＋33μs
		UART	注 1	158842/fx＋33μs
Oscillating Frequency Set	t_{WT9}	CSI	17984/fx	3s
		UART	注 1	3s
Baud Rate Set	t_{WT10}	UART	19200/fx	3s
Silicon Signature	t_{WT11}	CSI	576/fx	3s
		UART	注 1	3s
Version Get	t_{WT12}	CSI	544/fx	3s
		UART	注 1	3s
Security Set	t_{WT13}	CSI	784/fx＋27μs	3s
		UART	注 1	3s
	t_{WT14}	—	848/fx＋389μs	1018/fx＋467μs
	t_{WT15}	CSI	3248/fx＋195μs	3898/fx＋234μs
		UART	注 1	3898/fx＋234μs
Checksum	t_{WT16}	CSI	816/fx	3s
		UART	注 1	3s

注1.　コマンド送信前に，プログラマが受信可に設定されている必要があります。

2.　1ブロックの場合の時間

3.　256バイトの場合の時間

備考　ウエイトには，次のような定義があります。

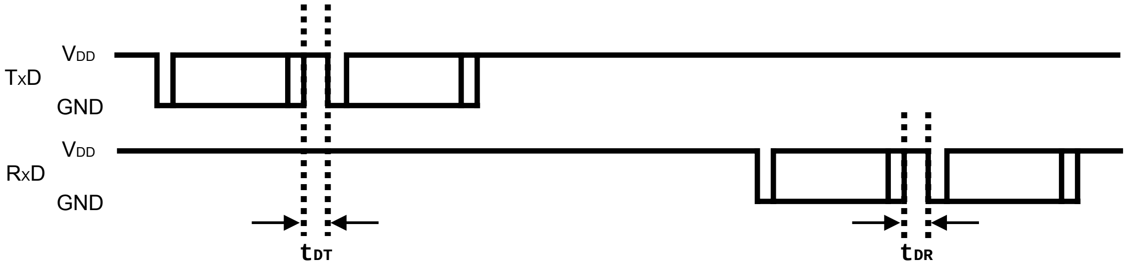
＜t_{WT0} - t_{WT16}＞

78K0/Kx1+は，MIN～MAX時間内に各コマンド処理を終了します。

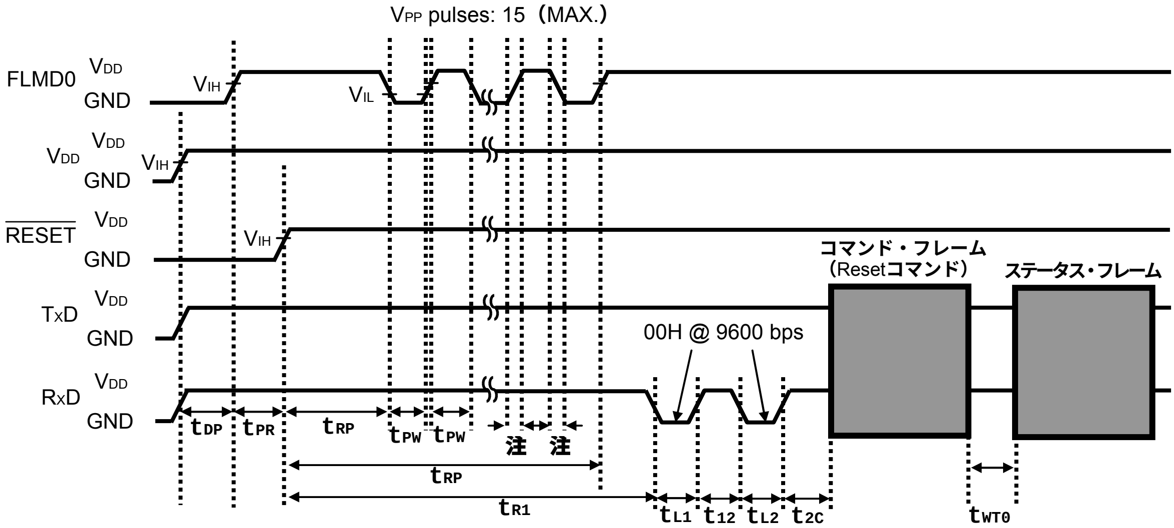
プログラマは，MAX時間まで，ステータス・チェックを繰り返す必要があります。

9. 3　UART通信方式

・データ・フレーム

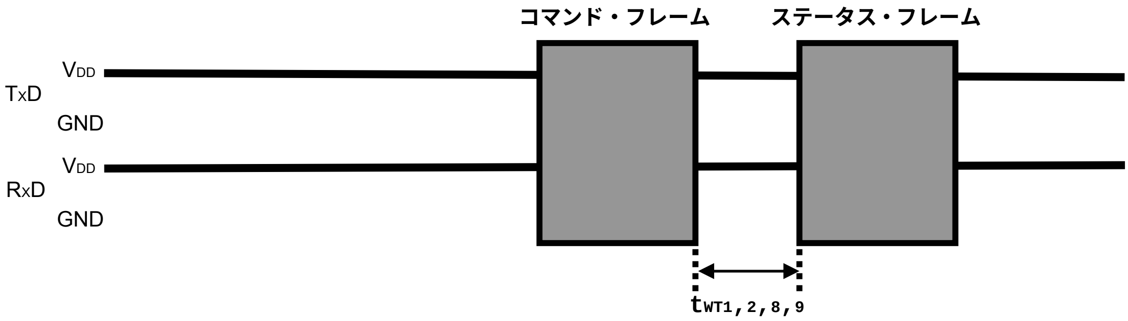


・プログラミング・モード設定／Resetコマンド

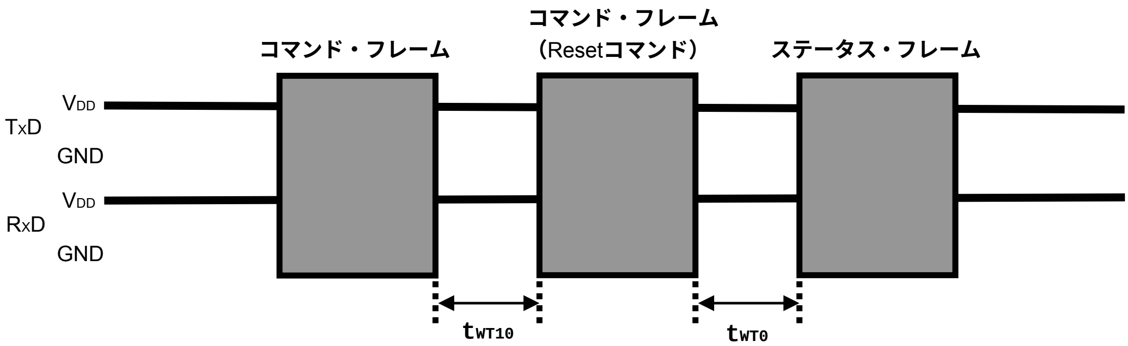


注　FLMD0 counter rise time / fall time

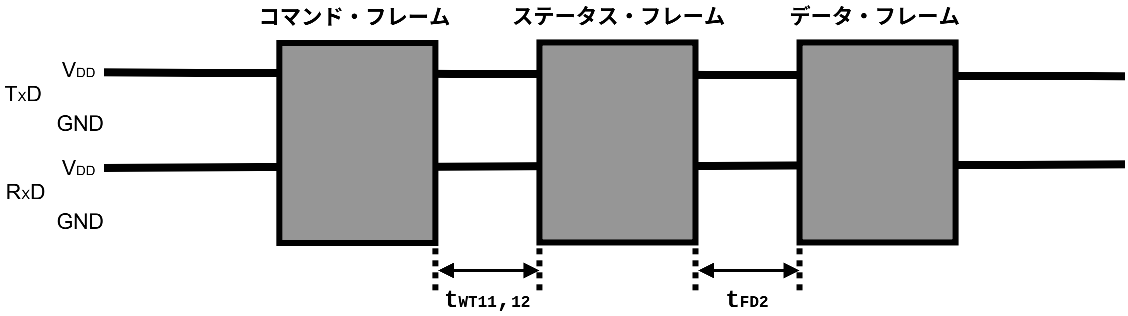
・Chip Eraseコマンド／Block Eraseコマンド／Block Blank Checkコマンド／Oscillating Frequency Setコマンド



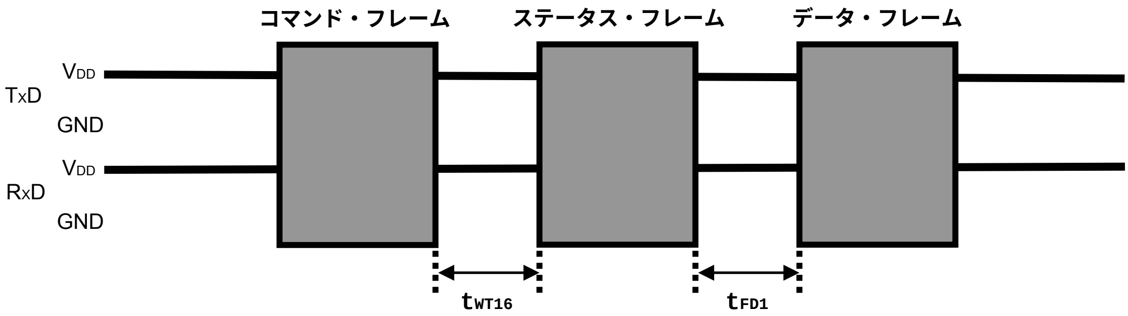
・Baud Rate Setコマンド



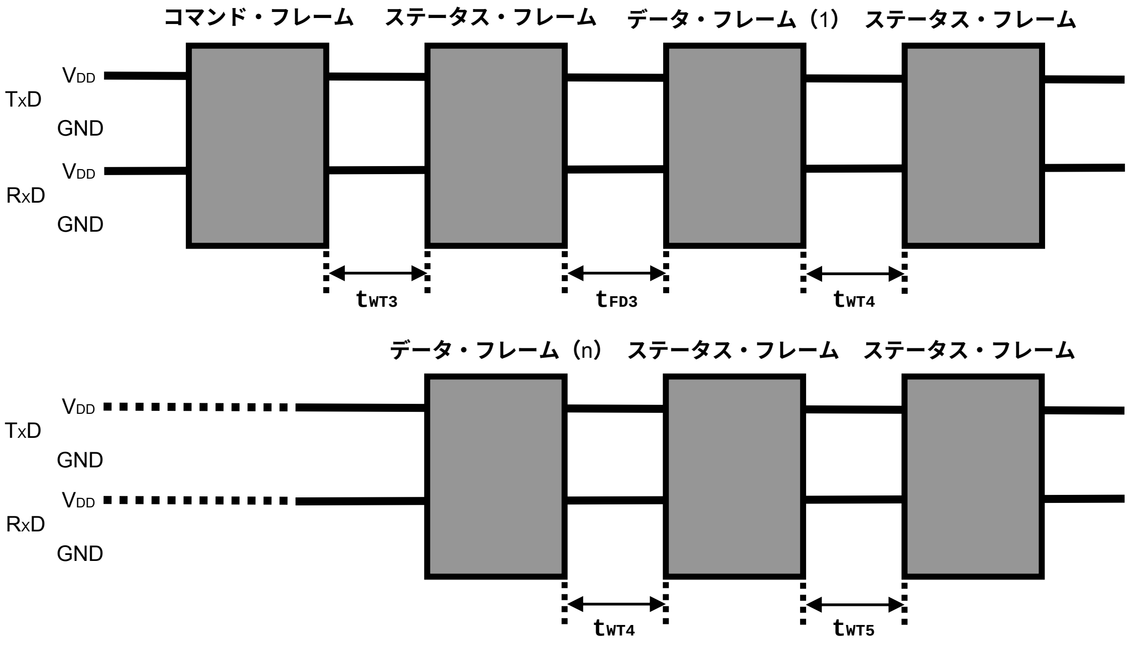
• Sillicon Signatureコマンド



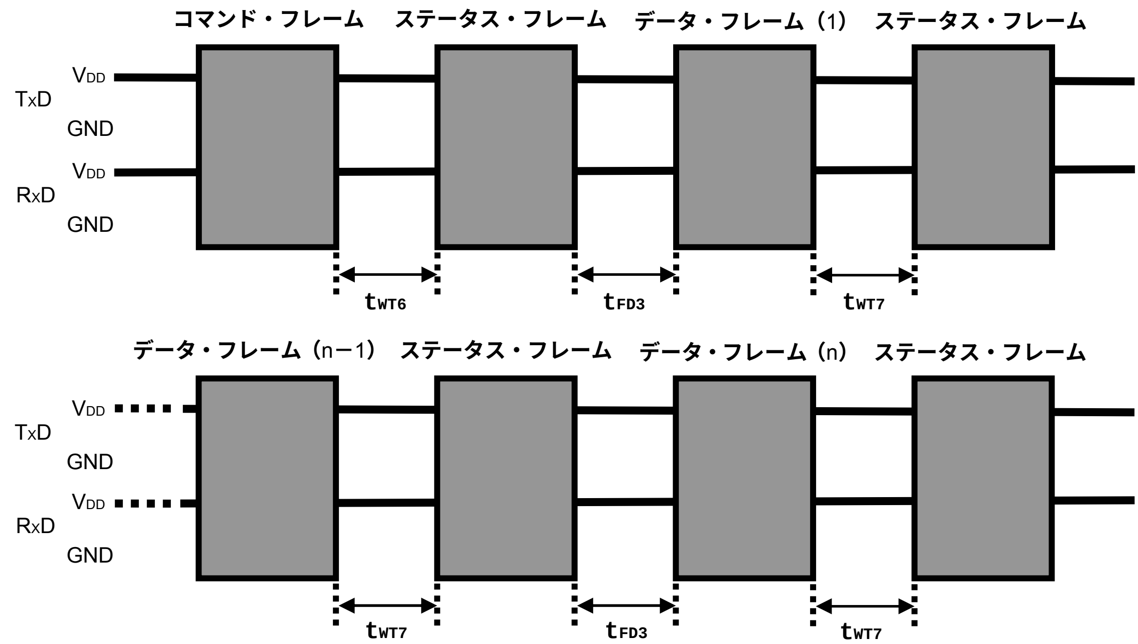
• Checksumコマンド



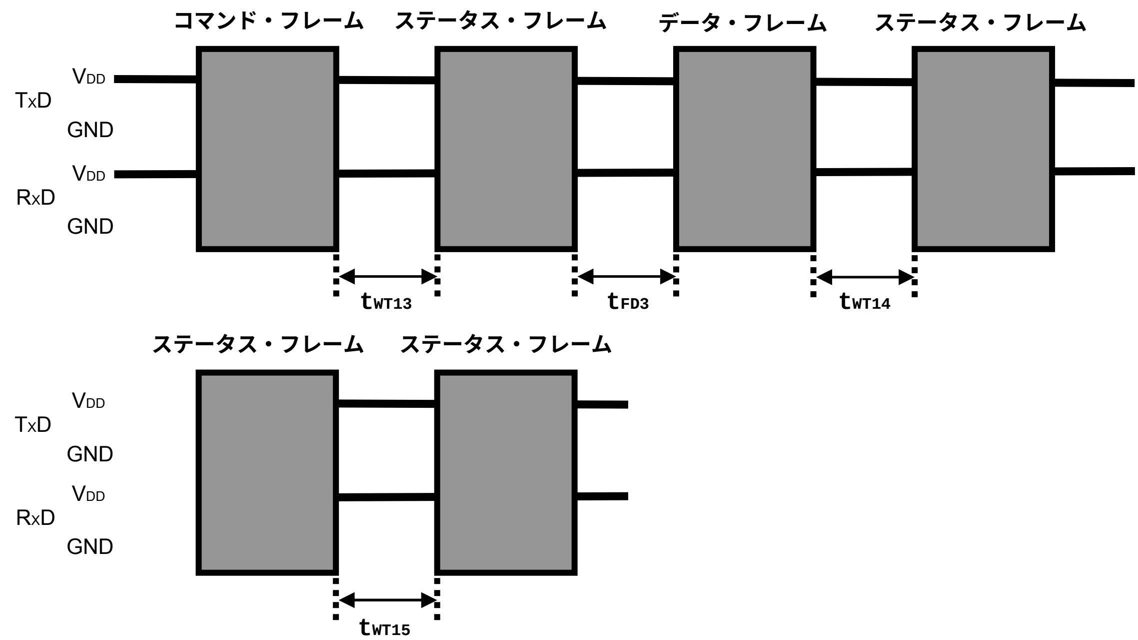
• Programmingコマンド



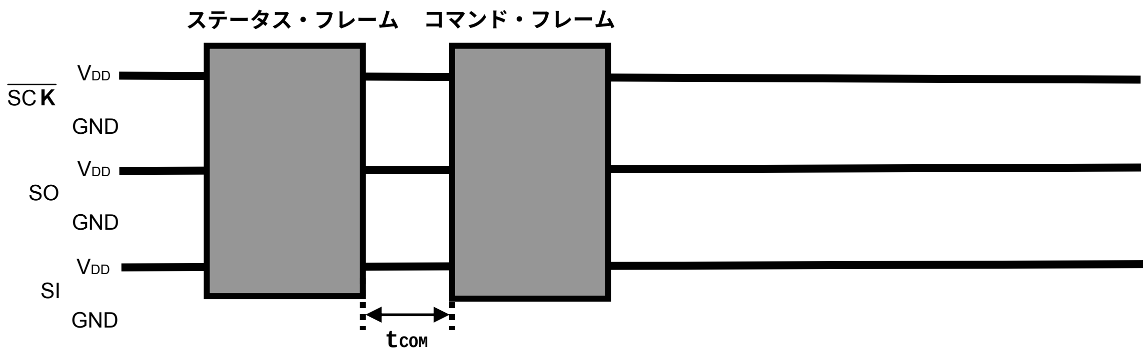
• Verifyコマンド



• Security Setコマンド

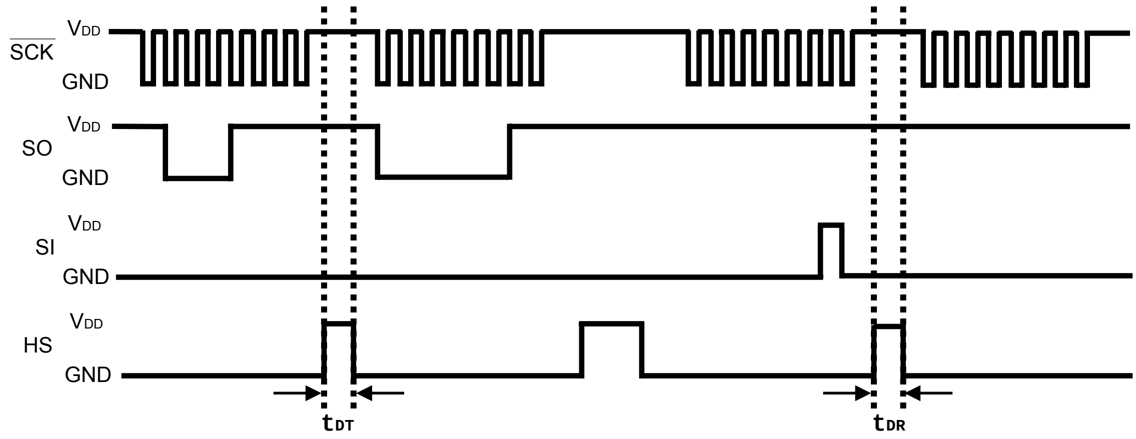


• コマンド・フレーム送信前のウエイト

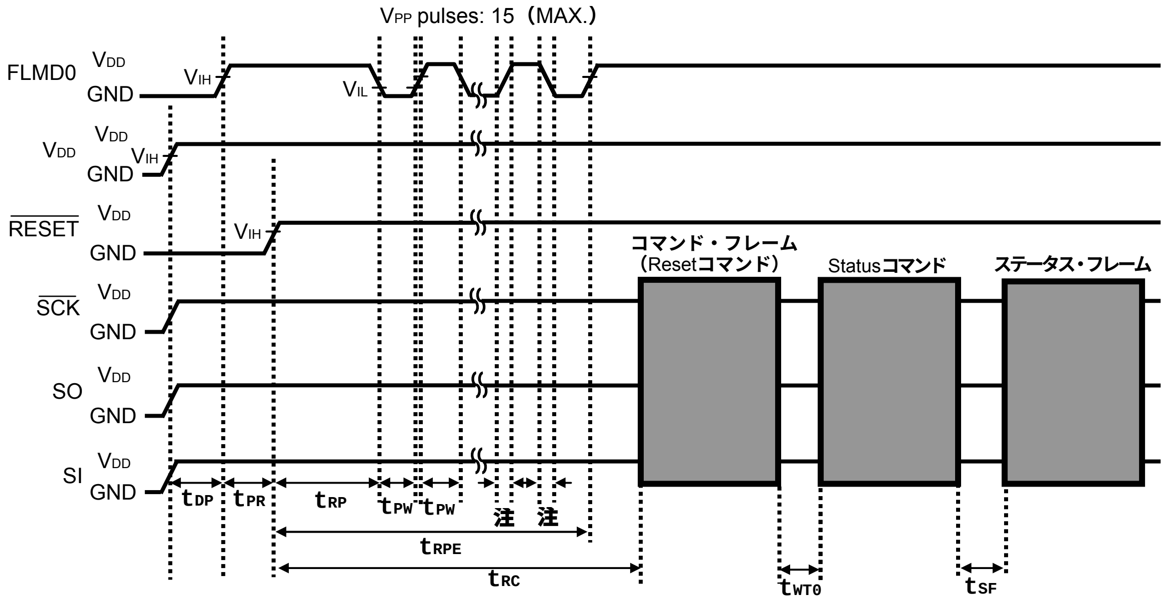


9．4　3線式シリアルI/O通信方式

・データ・フレーム

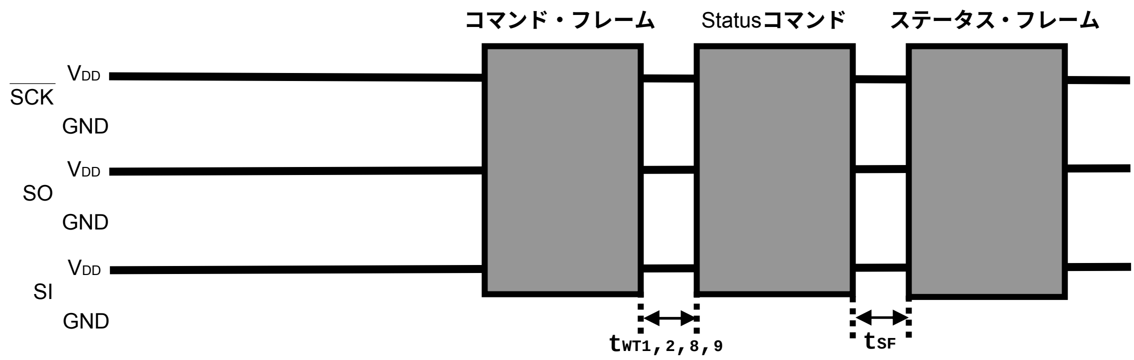


・プログラミング・モード設定／Resetコマンド

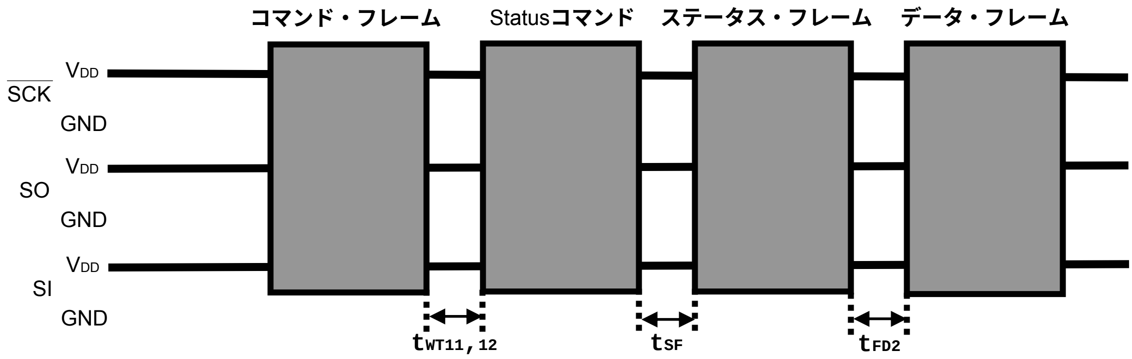


注　FLMD0 counter rise time / fall time

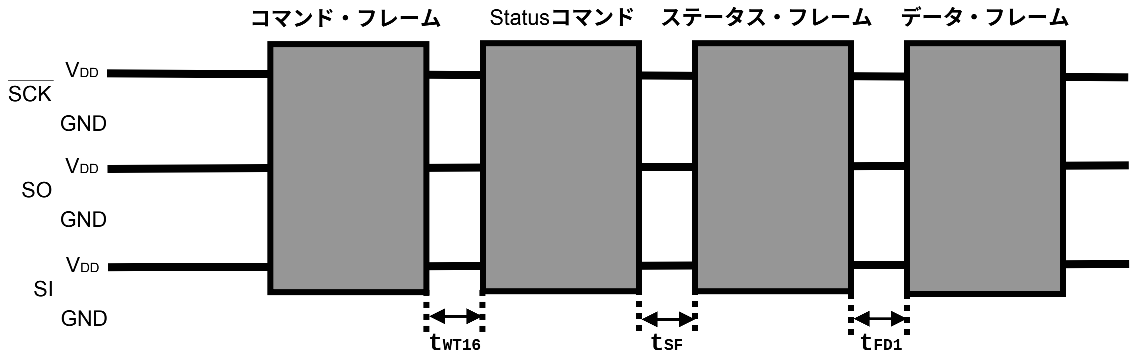
• Chip Eraseコマンド／Block Eraseコマンド／Block Blank Checkコマンド／Oscillating Frequency Setコマンド



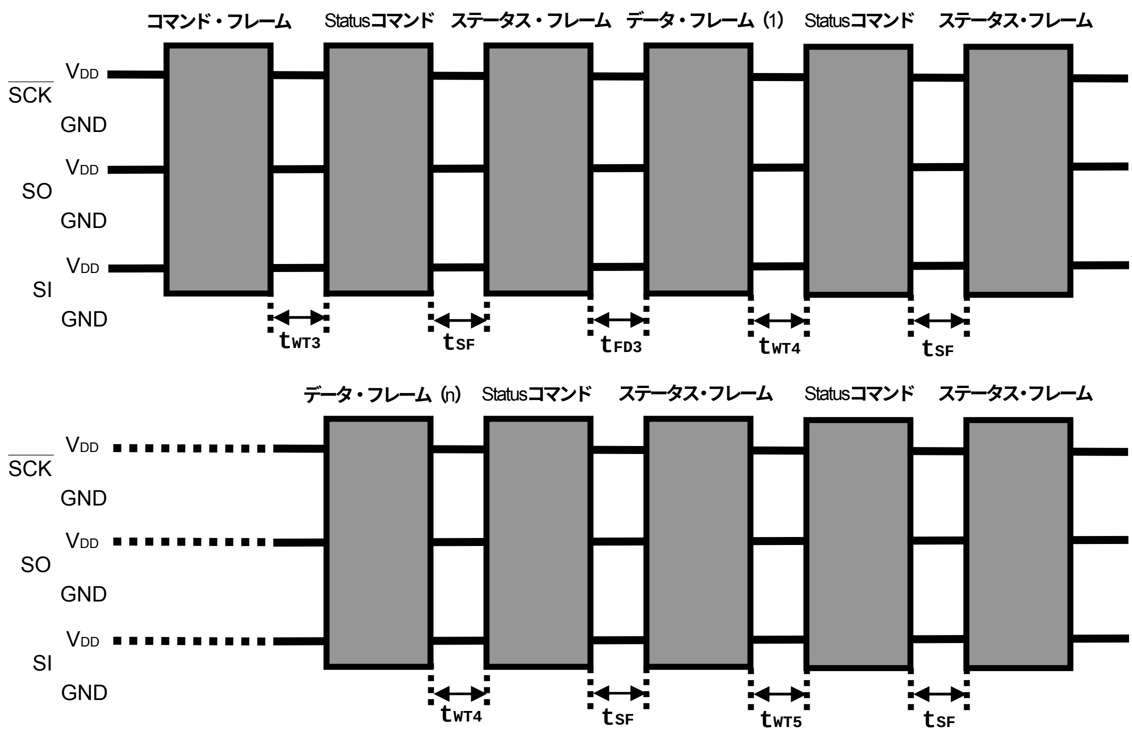
• Sillicon Signatureコマンド／Version Getコマンド



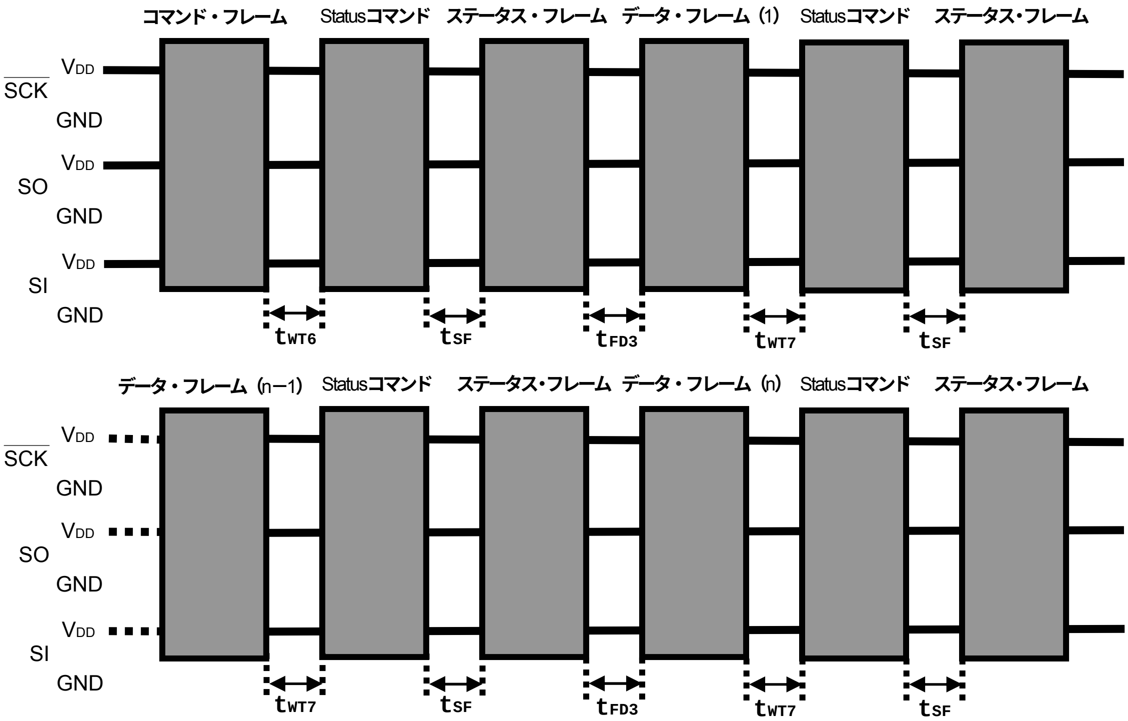
• Checksumコマンド



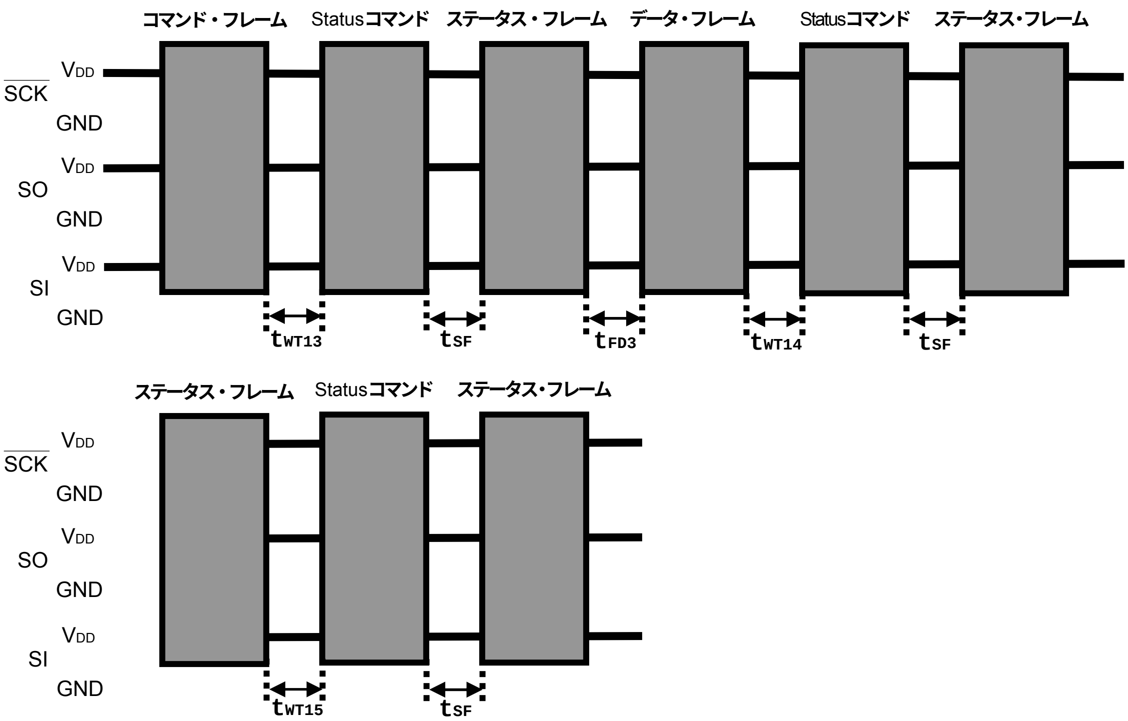
• Programmingコマンド



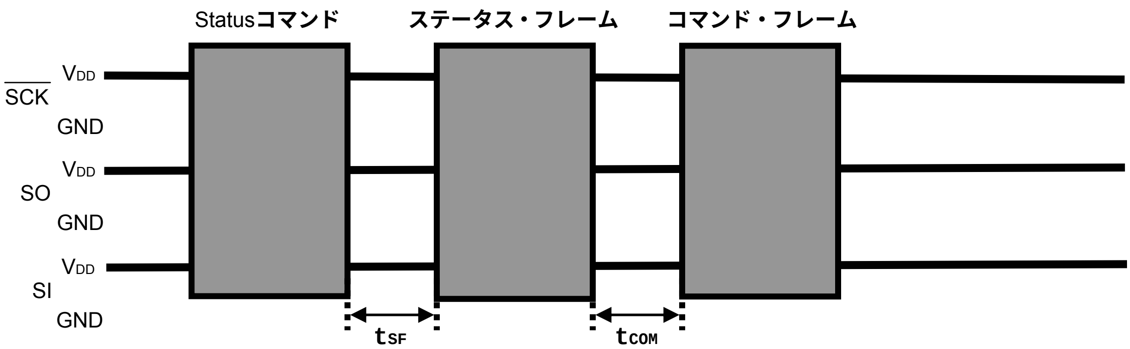
• Verifyコマンド



• Security Setコマンド



• コマンド・フレーム送信前のウエイト



第10章　電気的特性（参考値）

このスペックは参考値です。

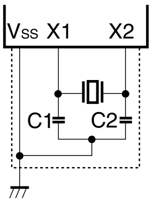
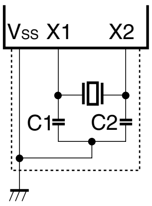
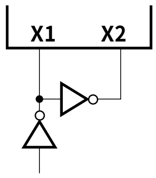
プログラマ設計の際は，必ず，各製品の最新ユーザーズ・マニュアルにて電気的特性をご確認ください。

フラッシュ・メモリ・プログラミング特性

（T_A = −10～＋65 ℃，2.7≦V_{DD}≦5.5 V, 2.7≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

項　目	略号	条　件		MIN.	TYP.	MAX.	単　位
電源電流	I_{DD}	f_{XP} = 16 MHz, V_{DD} = 5.5 V時	78K0/KB1+			30.5	mA
			78K0/KC1+, 78K0/KD1+			32	mA
			78K0/KE1+, 78K0/KF1+			35	mA

高速システム・クロック（水晶／セラミック）発振回路特性（T_A = －10～＋65 ℃，$2.7 \leq V_{DD} \leq 5.5$ V, $2.7 \leq AV_{REF} \leq V_{DD}$, V_{SS} = 0 V）

発振子	推奨回路	項　目	条　件	MIN.	TYP.	MAX.	単位
セラミック発振子		発振周波数（f_{XP}）注	$4.0\ V \leq V_{DD} \leq 5.5\ V$	2.0		16	MHz
			$3.5\ V \leq V_{DD} < 4.0\ V$	2.0		10	
			$3.0\ V \leq V_{DD} < 3.5\ V$	2.0		8.38	
			$2.7\ V \leq V_{DD} < 3.0\ V$	2.0		5.0	
水晶振動子		発振周波数（f_{XP}）注	$4.0\ V \leq V_{DD} \leq 5.5\ V$	2.0		16	MHz
			$3.5\ V \leq V_{DD} < 4.0\ V$	2.0		10	
			$3.0\ V \leq V_{DD} < 3.5\ V$	2.0		8.38	
			$2.7\ V \leq V_{DD} < 3.0\ V$	2.0		5.0	
外部クロック		X1入力周波数（f_{XP}）注	$4.0\ V \leq V_{DD} \leq 5.5\ V$	2.0		16	MHz
			$3.5\ V \leq V_{DD} < 4.0\ V$	2.0		10	
			$3.0\ V \leq V_{DD} < 3.5\ V$	2.0		8.38	
			$2.7\ V \leq V_{DD} < 3.0\ V$	2.0		5.0	
		X 1入力ハイ，ロウ・レベル幅（t_{XPH}, t_{XPL}）	$4.0\ V \leq V_{DD} \leq 5.5\ V$	30		250	ns
			$3.5\ V \leq V_{DD} < 4.0\ V$	46		250	
			$3.0\ V \leq V_{DD} < 3.5\ V$	56		250	
			$2.7\ V \leq V_{DD} < 3.0\ V$	96		250	

注意　高速システム・クロック発振回路を使用する場合は，配線容量などの影響を避けるために，図中の破線の部分を次のように配線してください。

- 配線は極力短くする。
- 他の信号線と交差させない。
- 変化する大電流が流れる線に接近させない。
- 発振回路のコンデンサの接地点は，常にV_{SS}と同電位になるようにする。
- 大電流が流れるグランド・パターンに接地しない。
- 発振回路から信号を取り出さない。

DC特性（1/3）

（T_A = −10～+65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

（1）78K0/KB1+, 78K0/KE1+, 78K0/KF1+

項　目	略　号	条　件		MIN.	TYP.	MAX.	単　位
ハイ・レベル出力電流	I_{OH}	1端子	4.0 V≦V_{DD}≦5.5 V			−5	mA
		$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1　合計	4.0 V≦V_{DD}≦5.5 V			−25	mA
		全端子合計	2.7 V≦V_{DD}<4.0 V			−10	mA
ロウ・レベル出力電流	I_{OL}	1端子	4.0 V≦V_{DD}≦5.5 V			10	mA
		$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1　合計	4.0 V≦V_{DD}≦5.5 V			30	mA
		全端子合計	2.7 V≦V_{DD}<4.0 V			10	mA
ハイ・レベル入力電圧	V_{IH1}	SO10, TxD6, HS		0.7 V_{DD}		V_{DD}	V
	V_{IH2}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\mathrm{RESET}}$		0.8 V_{DD}		V_{DD}	V
	V_{IH3}	X1, X2		V_{DD}−0.5		V_{DD}	V
ロウ・レベル入力電圧	V_{IL1}	SO10, TxD6, HS		0		0.3V_{DD}	V
	V_{IL2}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\mathrm{RESET}}$		0		0.2V_{DD}	V
	V_{IL3}	X1, X2		0		0.4	V
ハイ・レベル出力電圧	V_{OH}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1 端子合計I_{OH} = −25 mA	4.0 V≦V_{DD}≦5.5 V, I_{OH} = −5 mA	V_{DD}−1.0			V
		I_{OH} = −100 μA	2.7 V≦V_{DD}<4.0 V	V_{DD}−0.5			V
ロウ・レベル出力電圧	V_{OL}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1 端子合計I_{OL} = 30 mA	4.0 V≦V_{DD}≦5.5 V, I_{OL} = 10 mA			1.3	V
		I_{OL} = 400 μA	2.7 V≦V_{DD}<4.0 V			0.4	V
ハイ・レベル入力リーク電流	I_{LIH1}	V_I = V_{DD}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\mathrm{RESET}}$			3	μA
	I_{LIH2}	V_I = V_{DD}	X1, X2注			20	μA
ロウ・レベル入力リーク電流	I_{LIL1}	V_I = 0 V	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\mathrm{RESET}}$			−3	μA
	I_{LIL2}		X1, X2注			−20	μA
ハイ・レベル出力リーク電流	I_{LOH}	V_O = V_{DD}				3	μA
ロウ・レベル出力リーク電流	I_{LOL}	V_O = 0 V				−3	μA
プルアップ抵抗値	R_L	V_I = 0 V		10	30	100	kΩ
FLMD0電源電圧	Flmd	通常動作時		0		0.2V_{DD}	V

注　X2にX1の反転入力レベルを印加している場合。

DC特性（2/3）

（T_A = －10～＋65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

（2）78K0/KC1+

項　目	略　号	条　件		MIN.	TYP.	MAX.	単　位
ハイ・レベル出力電流	I_{OH}	1端子	4.0 V≦V_{DD}≦5.5 V			－5	mA
		$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS 合計	4.0 V≦V_{DD}≦5.5 V			－25	mA
		FLMD1	4.0 V≦V_{DD}≦5.5 V			－25	mA
		全端子合計	2.7 V≦V_{DD}＜4.0 V			－10	mA
ロウ・レベル出力電流	I_{OL}	1端子	4.0 V≦V_{DD}≦5.5 V			10	mA
		$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS 合計	4.0 V≦V_{DD}≦5.5 V			30	mA
		全端子合計	2.7 V≦V_{DD}＜4.0 V			10	mA
ハイ・レベル入力電圧	V_{IH1}	SO10, TxD6, HS		0.7V_{DD}		V_{DD}	V
	V_{IH2}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\mathrm{RESET}}$		0.8V_{DD}		V_{DD}	V
	V_{IH3}	X1, X2		V_{DD}－0.5		V_{DD}	V
ロウ・レベル入力電圧	V_{IL1}	SO10, TxD6, HS		0		0.3V_{DD}	V
	V_{IL2}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\mathrm{RESET}}$		0		0.2V_{DD}	V
	V_{IL3}	X1, X2		0		0.4	V
ハイ・レベル出力電圧	V_{OH}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS 端子合計I_{OH} = －25 mA	4.0 V≦V_{DD}≦5.5 V, I_{OH} = －5 mA	V_{DD}－1.0			V
		FLMD1 I_{OH} = －25 mA	4.0 V≦V_{DD}≦5.5 V, I_{OH} = －5 mA	V_{DD}－1.0			V
		I_{OH} = －100 μA	2.7 V≦V_{DD}＜4.0 V	V_{DD}－0.5			V
ロウ・レベル出力電圧	V_{OL}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS 端子合計I_{OL} = 30 mA	4.0 V≦V_{DD}≦5.5 V, I_{OL} = 10 mA			1.3	V
		FLMD1 I_{OL} = 30 mA	4.0 V≦V_{DD}≦5.5 V, I_{OL} = 10 mA			1.3	V
		I_{OL} = 400 μA	2.7 V≦V_{DD}＜4.0 V			0.4	V
ハイ・レベル入力リーク電流	I_{LIH1}	V_I = V_{DD}	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\mathrm{RESET}}$			3	μA
	I_{LIH2}	V_I = V_{DD}	X1, X2注			20	μA
ロウ・レベル入力リーク電流	I_{LIL1}	V_I = 0 V	$\overline{\mathrm{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\mathrm{RESET}}$			－3	μA
	I_{LIL2}		X1, X2注			－20	μA
ハイ・レベル出力リーク電流	I_{LOH}	V_O = V_{DD}				3	μA
ロウ・レベル出力リーク電流	I_{LOL}	V_O = 0 V				－3	μA
プルアップ抵抗値	R_L	V_I = 0 V		10	30	100	kΩ
FLMD0電源電圧	Flmd	通常動作時		0		0.2V_{DD}	V

注　X2にX1の反転入力レベルを印加している場合。

DC特性（3/3）

（T_A = －10～＋65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

（3）78K0/KD1+

項　目	略　号	条　件		MIN.	TYP.	MAX.	単　位
ハイ・レベル出力電流	I_{OH}	1端子	4.0 V≦V_{DD}≦5.5 V			－5	mA
		$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6　合計	4.0 V≦V_{DD}≦5.5 V			－25	mA
		HS, FLMD1　合計	4.0 V≦V_{DD}≦5.5 V			－25	mA
		全端子合計	2.7 V≦V_{DD}＜4.0 V			－10	mA
ロウ・レベル出力電流	I_{OL}	1端子	4.0 V≦V_{DD}≦5.5 V			10	mA
		$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6　合計	4.0 V≦V_{DD}≦5.5 V			30	mA
		HS, FLMD1　合計	4.0 V≦V_{DD}≦5.5 V			30	mA
		全端子合計	2.7 V≦V_{DD}＜4.0 V			10	mA
ハイ・レベル入力電圧	V_{IH1}	SO10, TxD6, HS		0.7V_{DD}		V_{DD}	V
	V_{IH2}	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\text{RESET}}$		0.8V_{DD}		V_{DD}	V
	V_{IH3}	X1, X2		V_{DD}－0.5		V_{DD}	V
ロウ・レベル入力電圧	V_{IL1}	SO10, TxD6, HS		0		0.3V_{DD}	V
	V_{IL2}	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, RxD6, FLMD1, $\overline{\text{RESET}}$		0		0.2V_{DD}	V
	V_{IL3}	X1, X2		0		0.4	V
ハイ・レベル出力電圧	V_{OH}	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6 端子合計I_{OH} = －25 mA	4.0 V≦V_{DD}≦5.5 V, I_{OH} = －5 mA	V_{DD}－1.0			V
		HS, FLMD1 端子合計I_{OH} = －25 mA	4.0 V≦V_{DD}≦5.5 V, I_{OH} = －5 mA	V_{DD}－1.0			V
		I_{OH} = －100 μA	2.7 V≦V_{DD}＜4.0 V	V_{DD}－0.5			V
ロウ・レベル出力電圧	V_{OL}	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6 端子合計I_{OL} = 30 mA	4.0 V≦V_{DD}≦5.5 V, I_{OL} = 10 mA			1.3	V
		HS, FLMD1 端子合計I_{OL} = 30 mA	4.0 V≦V_{DD}≦5.5 V, I_{OL} = 10 mA			1.3	V
		I_{OL} = 400 μA	2.7 V≦V_{DD}＜4.0 V			0.4	V
ハイ・レベル入力リーク電流	I_{LIH1}	V_I = V_{DD}	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\text{RESET}}$			3	μA
	I_{LIH2}	V_I = V_{DD}	X1, X2注			20	μA
ロウ・レベル入力リーク電流	I_{LIL1}	V_I = 0 V	$\overline{\text{SCK10}}$/TxD0, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\text{RESET}}$			－3	μA
	I_{LIL2}		X1, X2注			－20	μA
ハイ・レベル出力リーク電流	I_{LOH}	V_O = V_{DD}				3	μA
ロウ・レベル出力リーク電流	I_{LOL}	V_O = 0 V				－3	μA
プルアップ抵抗値	R_L	V_I = 0 V		10	30	100	kΩ
FLMD0電源電圧	Flmd	通常動作時		0		0.2V_{DD}	V

注　X2にX1の反転入力レベルを印加している場合。

AC特性

(1) 基本動作（T_A = −10～+65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

項　目	略 号	条　件	MIN.	TYP.	MAX.	単位
$\overline{\text{RESET}}$ 入力ロウ・レベル幅	t_{RSL}		10			μs

(2) シリアル・インタフェース

（T_A = −10～+65 ℃，2.7 V≦V_{DD}≦5.5 V, 2.7 V≦AV_{REF}≦V_{DD}, V_{SS} = 0 V）

(a) UARTモード（UART6，専用ボー・レート・ジェネレータ出力）

項　目	略 号	条　件	MIN.	TYP.	MAX.	単 位
転送レート					312.5	kbps

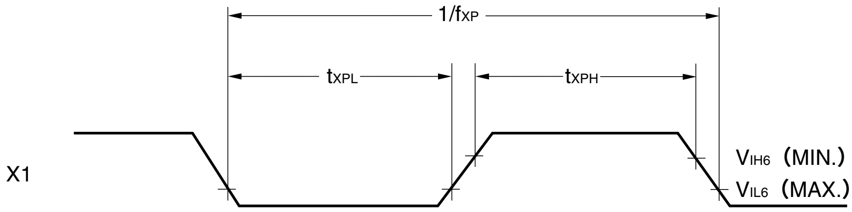
(b) 3線式シリアルI/Oモード（スレーブ・モード，$\overline{\text{SCK10}}$…外部クロック入力）

項　目	略 号	条　件	MIN.	TYP.	MAX.	単 位
$\overline{\text{SCK10}}$サイクル・タイム	t_{KCY2}		400			ns
$\overline{\text{SCK10}}$ハイ，ロウ・レベル幅	t_{KH2}, t_{KL2}		$t_{KCY2}/2$			ns
SI10セットアップ時間（対$\overline{\text{SCK10}}$↑）	t_{SIK2}		80			ns
SI1nホールド時間（対$\overline{\text{SCK10}}$↑）	t_{KSI2}		50			ns
$\overline{\text{SCK10}}$↓→SO10出力遅延時間	t_{KSO2}	C = 100 pF[注]			120	ns

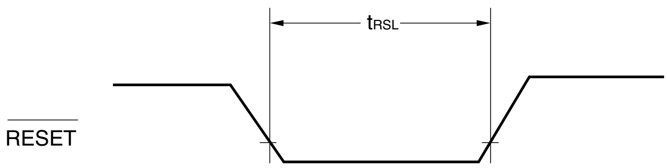
注　Cは，SO10出力ラインの負荷容量です。

ACタイミング測定点（X1を除く）

クロック・タイミング

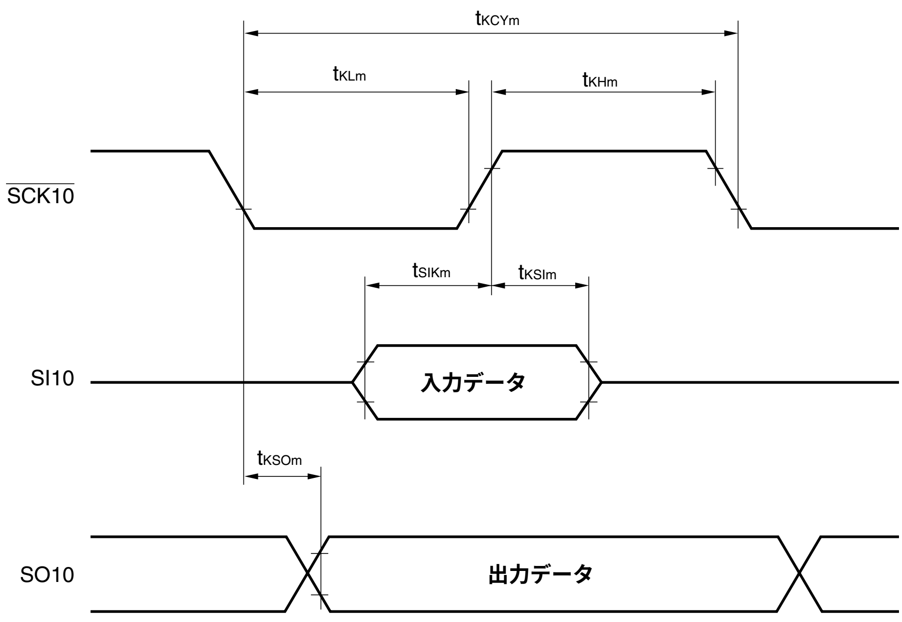


$\overline{\text{RESET}}$入力タイミング



シリアル転送タイミング

3線式シリアルI/Oモード：

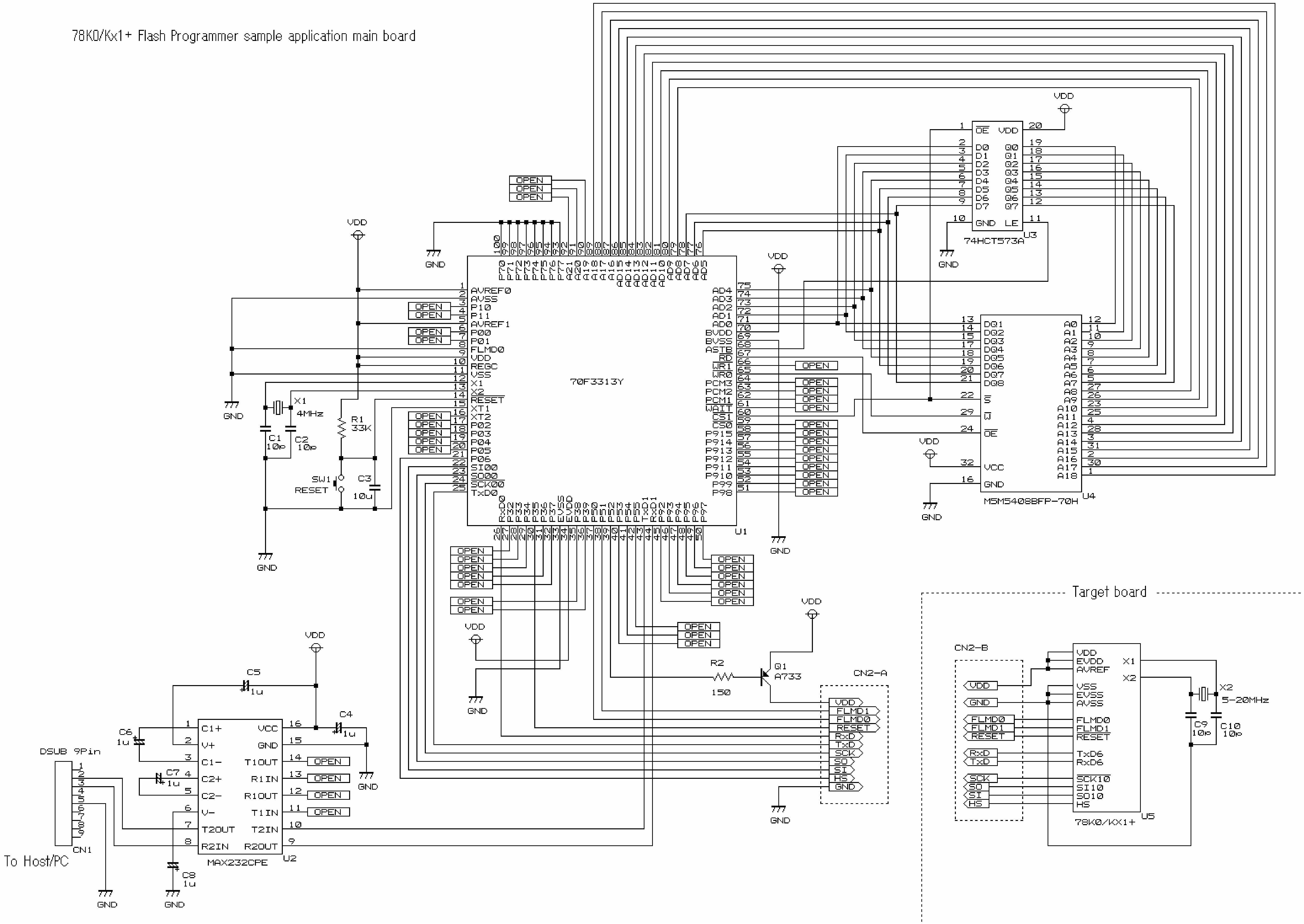


備考　m = 1, 2

付録A　参考回路図

プログラマと78K0/Kx1+の参考回路図を図A－1に示します。

図A－1　プログラマと78K0/Kx1+の参考回路図



〔メ　モ〕

【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　：044-435-9494
（電話：午前 9:00～12:00，午後 1:00～5:00）　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。

C04.2T