お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

RENESAS

アプリケーション・ノート

78K0/Lx2

8ビット・シングルチップ・マイクロコントローラ

フラッシュ・メモリ・プログラミング（プログラマ編）

78K0/LE2: μPD78F0361, 78F0362, 78F0363, 78F0363D
78K0/LF2: μPD78F0372, 78F0373, 78F0374, 78F0375, 78F0376, 78F0376D, 78F0382, 78F0383, 78F0384, 78F0385, 78F0386, 78F0386D
78K0/LG2: μPD78F0393, 78F0394, 78F0395, 78F0396, 78F0397, 78F0397D

資料番号 U18204JJ2V0AN00（第2版）
発行年月 July 2008 NS

〔メ　モ〕

目次要約

第1章　フラッシュ・メモリ・プログラミング　…　13

第2章　コマンド／データ・フレーム・フォーマット　…　27

第3章　コマンド処理説明　…　30

第4章　UART**通信方式　…**　51

第5章　3線式シリアルI/O**（**CSI**）通信方式　…**　103

第6章　フラッシュ・メモリ・プログラミング・パラメータ特性　…　159

付 録 A　**参考回路図　…**　176

付 録 B　**改版履歴　…**　183

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2008年7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，ソフトウエ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。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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。

M8E0710J

はじめに

対 象 者　このアプリケーション・ノートは，78K0/Lx2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/Lx2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
- 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（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

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

□一通りの機能を理解しようとするとき

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

□78K0/Lx2のハードウエア機能を知りたいとき

→78K0/Lx2 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/LE2　ユーザーズ・マニュアル	U17734J	U17734E
78K0/LF2　ユーザーズ・マニュアル	U17504J	U17504E
78K0/LG2　ユーザーズ・マニュアル	U17473J	U17473E
78K/0シリーズ　ユーザーズ・マニュアル　命令編	U12326J	U12326E

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　　次

第1章　フラッシュ・メモリ・プログラミング … 13

1. 1　**概　　要** … 13
1. 2　**システム構成** … 14
1. 3　**フラッシュ・メモリ構成** … 15
1. 4　**コマンド／ステータス一覧** … 17
    1. 4. 1　**コマンド一覧** … 17
    1. 4. 2　**ステータス一覧** … 18
1. 5　**電源投入とプログラミング・モードへの遷移** … 19
    1. 5. 1　**モード引き込みのフロー・チャート** … 21
    1. 5. 2　**サンプル・プログラム** … 22
1. 6　**ターゲットの電源遮断処理** … 24
1. 7　**フラッシュ・メモリ書き換えコマンド・フロー**… 24

第2章　コマンド／データ・フレーム・フォーマット … 27

2. 1　**コマンド・フレーム送信処理** … 29
2. 2　**データ・フレーム送信処理** … 29
2. 3　**データ・フレーム受信処理** … 29

第3章　コマンド処理説明 … 30

3. 1　Status**コマンド** … 30
    3. 1. 1　**説明** … 30
    3. 1. 2　**コマンド・フレームとステータス・フレーム** … 30
3. 2　Reset**コマンド** … 31
    3. 2. 1　**説明** … 31
    3. 2. 2　**コマンド・フレームとステータス・フレーム** … 31
3. 3　Baud Rate Set**コマンド** … 32
3. 4　Oscillating Frequency Set**コマンド** … 32
    3. 4. 1　**説明** … 32
    3. 4. 2　**コマンド・フレームとステータス・フレーム** … 32
3. 5　Chip Erase**コマンド** … 33
    3. 5. 1　**説明** … 33
    3. 5. 2　**コマンド・フレームとステータス・フレーム** … 33
3. 6　Block Erase**コマンド** … 34
    3. 6. 1　**説明** … 34
    3. 6. 2　**コマンド・フレームとステータス・フレーム** … 34
3. 7　Programming**コマンド** … 35
    3. 7. 1　**説明** … 35
    3. 7. 2　**コマンド・フレームとステータス・フレーム** … 35
    3. 7. 3　**データ・フレームとステータス・フレーム** … 35
    3. 7. 4　**全データ転送完了とステータス・フレーム** … 36

3. 8　Verify**コマンド** … 37
3. 8. 1　**説明** … 37
3. 8. 2　**コマンド・フレームとステータス・フレーム** … 37
3. 8. 3　**データ・フレームとステータス・フレーム** … 37
3. 9　Block Blank Check**コマンド** … 39
3. 9. 1　**説明** … 39
3. 9. 2　**コマンド・フレームとステータス・フレーム** … 39
3. 10　Silicon Signature**コマンド** … 40
3. 10. 1　**説明** … 40
3. 10. 2　**コマンド・フレームとステータス・フレーム** … 40
3. 10. 3　**シリコン・シグネチャ・データ・フレーム** … 41
3. 10. 4　78K0/Lx2**シリコン・シグネチャ一覧** … 44
3. 11　Version Get**コマンド** … 45
3. 11. 1　**説明** … 45
3. 11. 2　**コマンド・フレームとステータス・フレーム** … 45
3. 11. 3　**バージョン・データ・フレーム** … 46
3. 12　Checksum**コマンド** … 47
3. 12. 1　**説明** … 47
3. 12. 2　**コマンド・フレームとステータス・フレーム** … 47
3. 12. 3　**チェックサム・データ・フレーム** … 47
3. 13　Security Set**コマンド** … 48
3. 13. 1　**説明** … 48
3. 13. 2　**コマンド・フレームとステータス・フレーム** … 48
3. 13. 3　**データ・フレームとステータス・フレーム** … 49
3. 13. 4　**内部ベリファイ確認とステータス・フレーム** … 49

第4章　UART**通信方式** … 51

4. 1　**コマンド・フレーム送信処理のフロー・チャート** … 52
4. 2　**データ・フレーム送信処理のフロー・チャート** … 53
4. 3　**データ・フレーム受信処理のフロー・チャート** … 54
4. 4　Reset**コマンド** … 55
4. 4. 1　**処理手順チャート** … 55
4. 4. 2　**処理手順説明** … 56
4. 4. 3　**終了時の内容** … 56
4. 4. 4　**フロー・チャート** … 57
4. 4. 5　**サンプル・プログラム** … 58
4. 5　Oscillating Frequency Set**コマンド** … 59
4. 5. 1　**処理手順チャート** … 59
4. 5. 2　**処理手順説明** … 60
4. 5. 3　**終了時の内容** … 60
4. 5. 4　**フロー・チャート** … 61
4. 5. 5　**サンプル・プログラム** … 62
4. 6　Chip Erase**コマンド** … 63
4. 6. 1　**処理手順チャート** … 63
4. 6. 2　**処理手順説明** … 64
4. 6. 3　**終了時の内容** … 64
4. 6. 4　**フロー・チャート** … 65
4. 6. 5　**サンプル・プログラム** … 66

4. 7　Block Erase**コマンド …** 67
4. 7. 1　**処理手順チャート …** 67
4. 7. 2　**処理手順説明 …** 68
4. 7. 3　**終了時の内容 …** 68
4. 7. 4　**フロー・チャート …** 69
4. 7. 5　**サンプル・プログラム …** 70
4. 8　Programming**コマンド …** 71
4. 8. 1　**処理手順チャート …** 71
4. 8. 2　**処理手順説明 …** 72
4. 8. 3　**終了時の内容 …** 73
4. 8. 4　**フロー・チャート …** 74
4. 8. 5　**サンプル・プログラム …** 75
4. 9　Verify**コマンド …** 77
4. 9. 1　**処理手順チャート …** 77
4. 9. 2　**処理手順説明 …** 78
4. 9. 3　**終了時の内容 …** 78
4. 9. 4　**フロー・チャート …** 79
4. 9. 5　**サンプル・プログラム …** 80
4. 10　Block Blank Check**コマンド …** 82
4. 10. 1　**処理手順チャート …** 82
4. 10. 2　**処理手順説明 …** 83
4. 10. 3　**終了時の内容 …** 83
4. 10. 4　**フロー・チャート …** 84
4. 10. 5　**サンプル・プログラム …** 85
4. 11　Silicon Signature**コマンド …** 86
4. 11. 1　**処理手順チャート …** 86
4. 11. 2　**処理手順説明 …** 87
4. 11. 3　**終了時の内容 …** 87
4. 11. 4　**フロー・チャート …** 88
4. 11. 5　**サンプル・プログラム …** 89
4. 12　Version Get**コマンド …** 90
4. 12. 1　**処理手順チャート …** 90
4. 12. 2　**処理手順説明 …** 91
4. 12. 3　**終了時の内容 …** 91
4. 12. 4　**フロー・チャート …** 92
4. 12. 5　**サンプル・プログラム …** 93
4. 13　Checksum**コマンド …** 94
4. 13. 1　**処理手順チャート …** 94
4. 13. 2　**処理手順説明 …** 95
4. 13. 3　**終了時の内容 …** 95
4. 13. 4　**フロー・チャート …** 96
4. 13. 5　**サンプル・プログラム …** 97
4. 14　Security Set**コマンド …** 98
4. 14. 1　**処理手順チャート …** 98
4. 14. 2　**処理手順説明 …** 99
4. 14. 3　**終了時の内容 …** 99
4. 14. 4　**フロー・チャート …** 100
4. 14. 5　**サンプル・プログラム …** 101

第5章　3線式シリアルI/O（CSI）通信方式 … 103

5. 1　**コマンド・フレーム送信処理のフロー・チャート …** 104
5. 2　**データ・フレーム送信処理のフロー・チャート …** 105
5. 3　**データ・フレーム受信処理のフロー・チャート …** 106
5. 4　Status**コマンド …** 107
5. 4. 1　**処理手順チャート …** 107
5. 4. 2　**処理手順説明 …** 108
5. 4. 3　**終了時の内容 …** 108
5. 4. 4　**フロー・チャート …** 109
5. 4. 5　**サンプル・プログラム …** 110
5. 5　Reset**コマンド …** 111
5. 5. 1　**処理手順チャート …** 111
5. 5. 2　**処理手順説明 …** 112
5. 5. 3　**終了時の内容 …** 112
5. 5. 4　**フロー・チャート …** 113
5. 5. 5　**サンプル・プログラム …** 114
5. 6　Oscillating Frequency Set**コマンド …** 115
5. 6. 1　**処理手順チャート …** 115
5. 6. 2　**処理手順説明 …** 116
5. 6. 3　**終了時の内容 …** 116
5. 6. 4　**フロー・チャート …** 117
5. 6. 5　**サンプル・プログラム …** 118
5. 7　Chip Erase**コマンド …** 119
5. 7. 1　**処理手順チャート …** 119
5. 7. 2　**処理手順説明 …** 120
5. 7. 3　**終了時の内容 …** 120
5. 7. 4　**フロー・チャート …** 121
5. 7. 5　**サンプル・プログラム …** 122
5. 8　Block Erase**コマンド …** 123
5. 8. 1　**処理手順チャート …** 123
5. 8. 2　**処理手順説明 …** 124
5. 8. 3　**終了時の内容 …** 124
5. 8. 4　**フロー・チャート …** 125
5. 8. 5　**サンプル・プログラム …** 126
5. 9　Programming**コマンド …** 127
5. 9. 1　**処理手順チャート …** 127
5. 9. 2　**処理手順説明 …** 128
5. 9. 3　**終了時の内容 …** 129
5. 9. 4　**フロー・チャート …** 130
5. 9. 5　**サンプル・プログラム …** 131
5. 10　Verify**コマンド …** 133
5. 10. 1　**処理手順チャート …** 133
5. 10. 2　**処理手順説明 …** 134
5. 10. 3　**終了時の内容 …** 134
5. 10. 4　**フロー・チャート …** 135
5. 10. 5　**サンプル・プログラム …** 136
5. 11　Block Blank Check**コマンド …** 138
5. 11. 1　**処理手順チャート …** 138
5. 11. 2　**処理手順説明 …** 139

5. 11. 3　**終了時の内容** … 139
5. 11. 4　**フロー・チャート** … 140
5. 11. 5　**サンプル・プログラム** … 141
5. 12　Silicon Signature**コマンド** … 142
5. 12. 1　**処理手順チャート** … 142
5. 12. 2　**処理手順説明** … 143
5. 12. 3　**終了時の内容** … 143
5. 12. 4　**フロー・チャート** … 144
5. 12. 5　**サンプル・プログラム** … 145
5. 13　Version Get**コマンド** … 146
5. 13. 1　**処理手順チャート** … 146
5. 13. 2　**処理手順説明** … 147
5. 13. 3　**終了時の内容** … 147
5. 13. 4　**フロー・チャート** … 148
5. 13. 5　**サンプル・プログラム** … 149
5. 14　Checksum**コマンド** … 150
5. 14. 1　**処理手順チャート** … 150
5. 14. 2　**処理手順説明** … 151
5. 14. 3　**終了時の内容** … 151
5. 14. 4　**フロー・チャート** … 152
5. 14. 5　**サンプル・プログラム** … 153
5. 15　Security Set**コマンド** … 154
5. 15. 1　**処理手順チャート** … 154
5. 15. 2　**処理手順説明** … 155
5. 15. 3　**終了時の内容** … 155
5. 15. 4　**フロー・チャート** … 156
5. 15. 5　**サンプル・プログラム** … 157

第6章　フラッシュ・メモリ・プログラミング・パラメータ特性 … 159

6. 1　**基本特性** … 159
6. 2　**フラッシュ・メモリ・プログラミング・モード設定時間** … 159
6. 3　**プログラミング特性** … 160
6. 4　UART**通信方式** … 170
6. 5　3**線式シリアル**I/O**通信方式** … 173

付録A　**参考回路図** … 176

付録B　**改版履歴** … 183

B. 1　**本版で改訂された主な箇所** 183

第1章　フラッシュ・メモリ・プログラミング

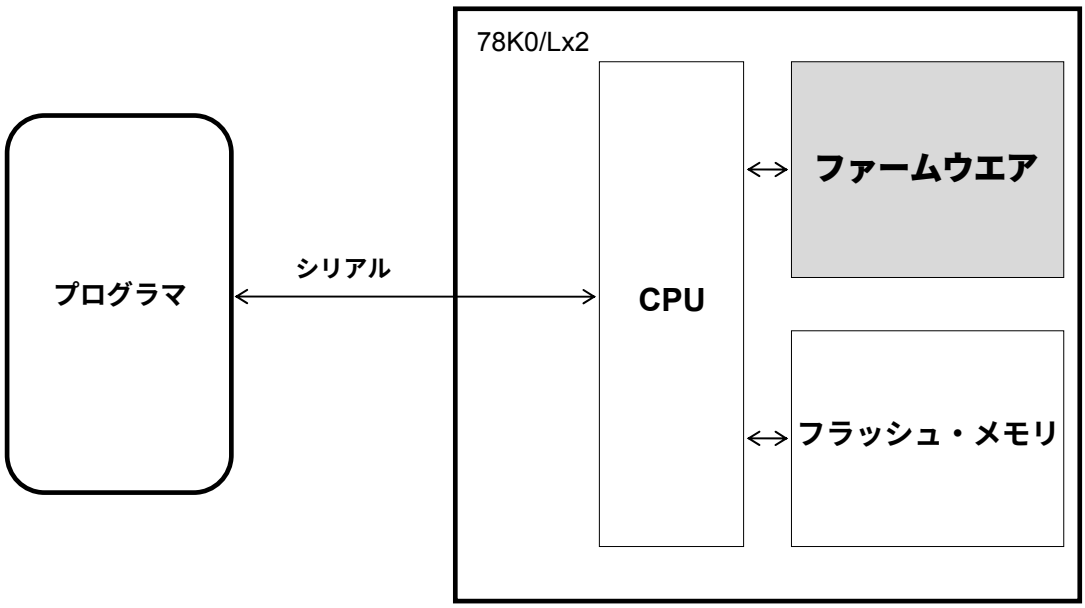
78K0/Lx2に内蔵されるフラッシュ・メモリの書き換えを行うには，通常は専用のフラッシュ・メモリ・プログラマ（以降プログラマ）を使う必要があります。

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/Lx2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/Lx2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/Lx2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。

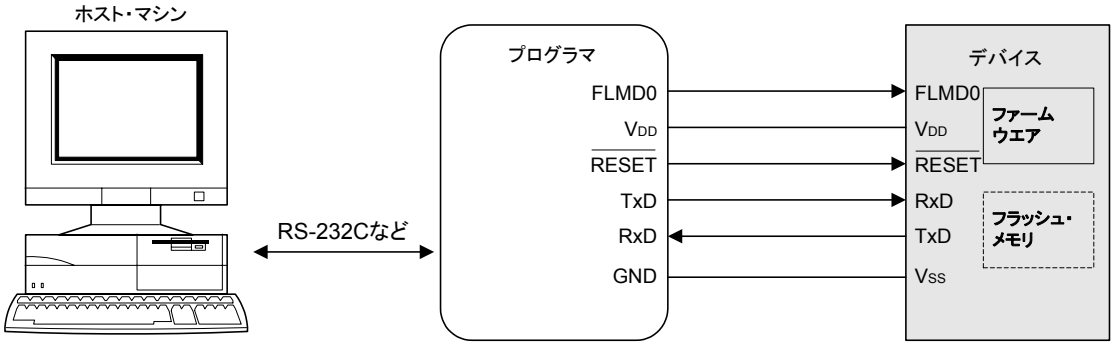
図1－2は，ホスト・マシンからの制御によりプログラマを使用するプログラミング方法を示しています。

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。

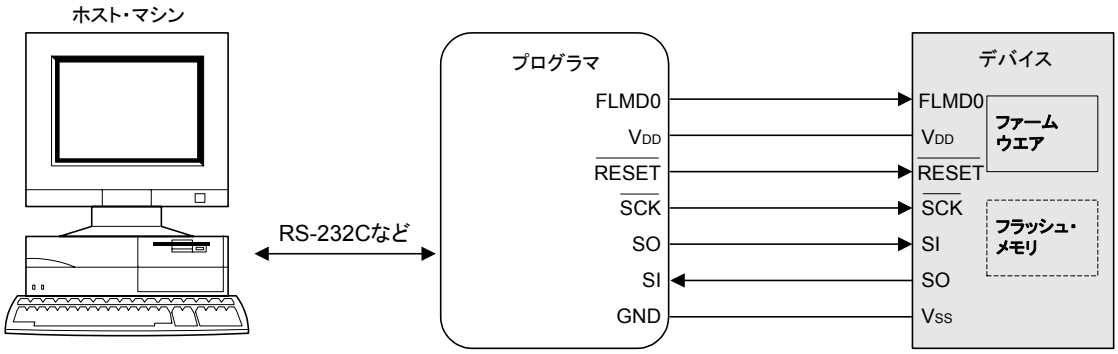
たとえば，NECエレクトロニクス製フラッシュ・メモリ・プログラマ PG-FP5は，ホスト・マシンを接続してGUIソフトウエアにより実行する方法と，スタンド・アローンで実行する方法のどちらでも動作可能です。

図1－2　システム構成例

・UART**通信方式（**LSB**先頭転送）**



・3**線式シリアル**I/O（CSI）**通信方式（**MSB**先頭転送）**



備考　プログラミング時に使用する端子名および未使用端子の処理に関しては各製品のユーザーズ・マニュアルを参照してください。

1. 3　フラッシュ・メモリ構成

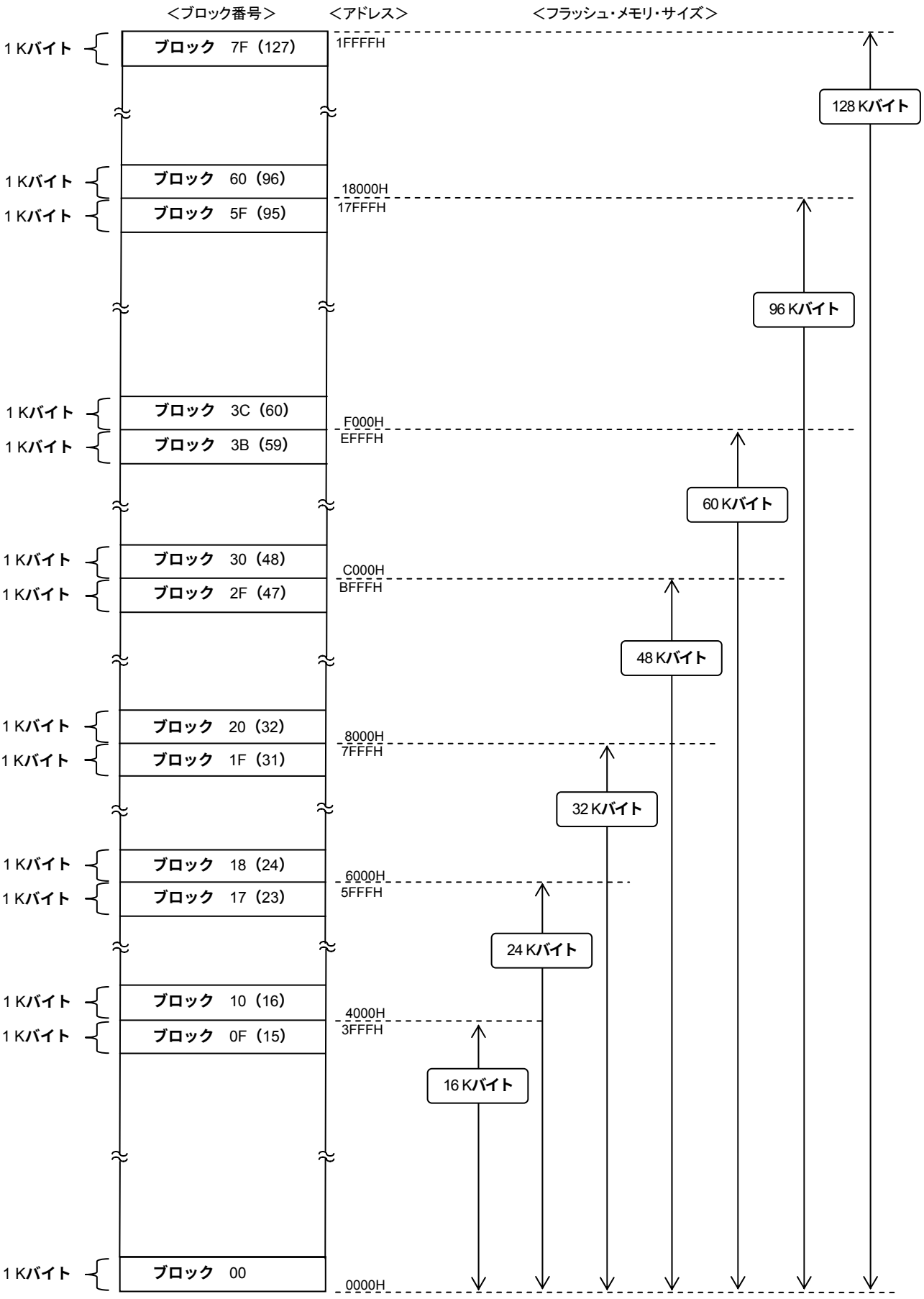
78K0/Lx2は，プログラマ側で製品固有情報（デバイス名，メモリ情報）を管理しておく必要があります。

表1－1に78K0/Lx2のフラッシュ・メモリ・サイズ，図1－3にフラッシュ・メモリ構成を示します。

表1－1　78K0/Lx2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/LE2	μPD78F0361	16 KB
	μPD78F0362	24 KB
	μPD78F0363, 78F0363D	32 KB
78K0/LF2	μPD78F0372	24 KB
	μPD78F0373	32 KB
	μPD78F0374	48 KB
	μPD78F0375	60 KB
	μPD78F0376, 78F0376D	96 KB
	μPD78F0382	24 KB
	μPD78F0383	32 KB
	μPD78F0384	48 KB
	μPD78F0385	60 KB
	μPD78F0386, 78F0386D	96 KB
78K0/LG2	μPD78F0393	32 KB
	μPD78F0394	48 KB
	μPD78F0395	60 KB
	μPD78F0396	96 KB
	μPD78F0397, 78F0397D	128 KB

図1－3　フラッシュ・メモリ構成



備考　1ブロックは，すべて1 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）。

1.4　コマンド／ステータス一覧

78K0/Lx2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1－2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/Lx2に送信し，78K0/Lx2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。

1.4.1　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1－2　プログラマから78K0/Lx2への送信コマンド一覧

コマンド番号	コマンド名	機能名	機　　能
20H	Chip Erase	消去	全フラッシュ・メモリを消去します。
22H	Block Erase		指定したブロックのフラッシュ・メモリを消去します。
40H	Programming	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。
13H	Verify	ベリファイ	指定したフラッシュ・メモリの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	ブランク・チェック	指定したブロックのフラッシュ・メモリの消去状態を確認します。
70H	Status	情報取得	現在の動作状況（ステータス・データ）を取得します。
C0H	Silicon Signature		78K0/Lx2情報（書き込みプロトコル情報）を取得します。
C5H	Version Get		78K0/Lx2のバージョン，ファームウエア・バージョンを取得します。
B0H	Checksum		指定し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	セキュリティ情報を設定します。
00H	Reset	その他	通信同期検出に使用します。
90H	Oscillating Frequency Set		78K0/Lx2の発振周波数を指定します。

1.4.2　ステータス一覧

プログラマが78K0/Lx2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1－3　ステータス・コード一覧

ステータス・コード	ステータス	内　　容
04H	Command number error	サポートされていないコマンドを受信した場合のエラー
05H	Parameter error	コマンド情報（パラメータ）が適切でない場合のエラー
06H	正常応答（ACK）	正常応答
07H	Checksum error	プログラマから送信されたフレームのデータが異常の場合のエラー
0FH	Verify error	プログラマから送信されたデータとのベリファイ・エラー
10H	Protect error	Security Setコマンドで禁止した処理を実行しようとした場合のエラー
15H	否定応答（NACK）	否定応答
1AH	MRG10 error	消去エラー
1BH	MRG11 error	データ書き込み時の内部ベリファイ・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
20H	Read error	セキュリティ情報の読み出しに失敗した場合のエラー
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からリトライしても構いません。ただし，無限にリトライを繰り返さないようにリトライの回数制限を設けることを推奨します。

また，上記ステータス・コード一覧には出てきませんが，各種タイムアウト・エラー（BUSYのタイムアウト，UART通信時のデータ・フレーム受信のタイムアウトなど）が発生した場合は，一度78K0/Lx2に対して電源遮断処理（1.6　**ターゲットの電源遮断処理**参照）を行ってから改めて接続することを推奨します。

1.5　電源投入とプログラミング・モードへの遷移

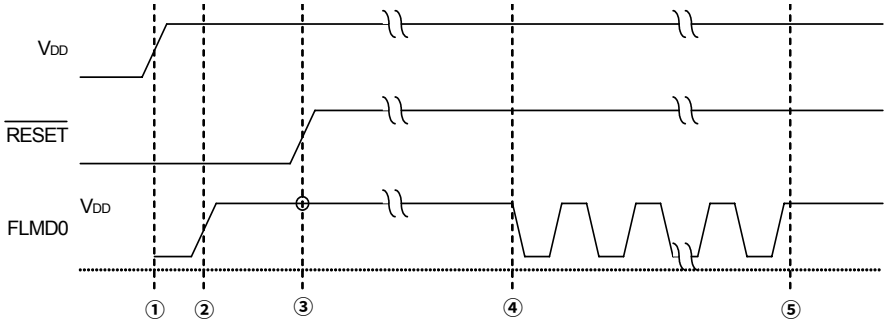
プログラマにてフラッシュ・メモリの書き換えを行うには，まず78K0/Lx2の動作モードをフラッシュ・メモリ・プログラミング・モードに遷移させる必要があります。

このモードに遷移するには，78K0/Lx2のフラッシュ・メモリ・プログラミング・モード引き込み用端子（FLMD0）に規定の電圧を供給し，その後リセットを解除します。

また，プログラミング・モード遷移後，FLMD0端子はフラッシュ・メモリ書き換えのための通信方式を選択するためのパルス入力を行います。

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1－4　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
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端子と動作モードの関係を次に示します。

表1－4　リセット時のFLMD0端子の設定と動作モード

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

78K0/Lx2で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1－5　78K0/Lx2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（X1 クロック（f_X）使用時）	TxD6（P13），RxD6（P14）
	3（外部メイン・システム・クロック（f_{EXCLK}）使用時）	
3 線式シリアル I/O（CSI10）	8	SO10（P12），SI10（P11），$\overline{SCK10}$（P10）

• 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1－6　UART通信の通信条件

項　　目	内　　容
ボー・レート	Oscillating Frequency Set コマンドまでの送信は，9600 bps で通信を行います。 そして，ステータス受信時からは，115200 bps に通信レートが変更となります。 以降は 115200 bps 固定です。
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/Lx2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，最適なタイミングでの通信が可能です。

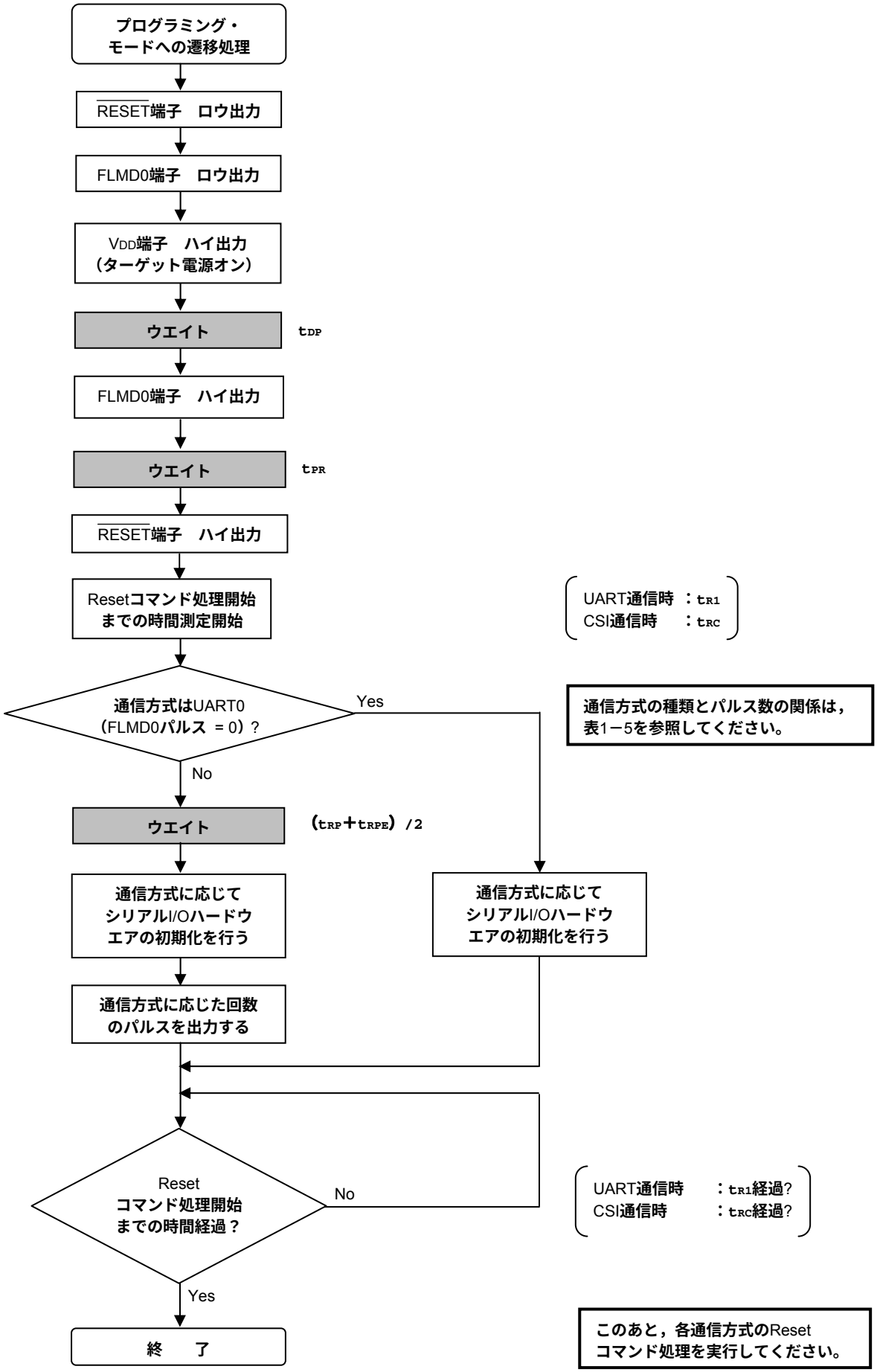
注意　UART通信を行う場合は，マスタとスレーブともに同一のボー・レートにしてください。

• 3線式シリアルI/O（CSI）通信方式

CSI通信では，$\overline{\text{SCK}}$, 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/Lx2が送受信可能になっていない状態のときに$\overline{\text{SCK}}$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。

1.5.1　モード引き込みのフロー・チャート



1.5.2　サンプル・プログラム

引き込み処理のサンプル・プログラムです。

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

        int   n;
        int   pulse;

        SRMK0 = true;
        UARTE0 = false;

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

        }

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

        fl_wait(tDP);               // wait

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

        pFL_RES    = hi;            // RESET = high
        start_flto(fl_if == FLIF_CSI ? tRC : tR1);     // start "tRC" wait timer
        fl_wait((tRP+tRPE)/2);

        if (fl_if == FLIF_UART){
          init_fl_uart();           // Initialize UART h.w. (for Flash device control)
          UARTE0 = true;
          SRIF0 = false;
          SRMK0 = false;
        }
        else{
          init_fl_csi();            // Initialize CSI h.w.
        }

        for (n = 0; n < pulse; n++){// pulse output

          pFL_FLMD0 = low;
          fl_wait(tPW);
          pFL_FLMD0 = hi;
```

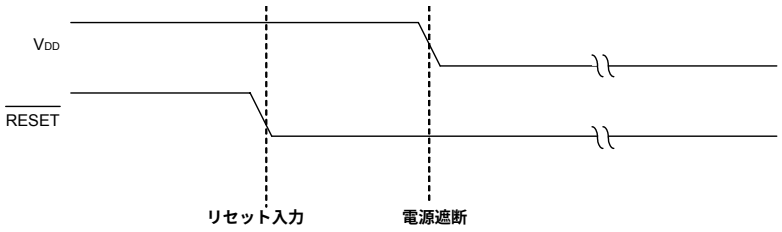
```
            fl_wait(tPW);
          }
          while(!check_flto())        // timeout tRC ?
                 ;                    // no

          // start RESET command proc.

}
```

1.6　ターゲットの電源遮断処理

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

注意　コマンド処理中の電源遮断およびリセット入力は禁止です。

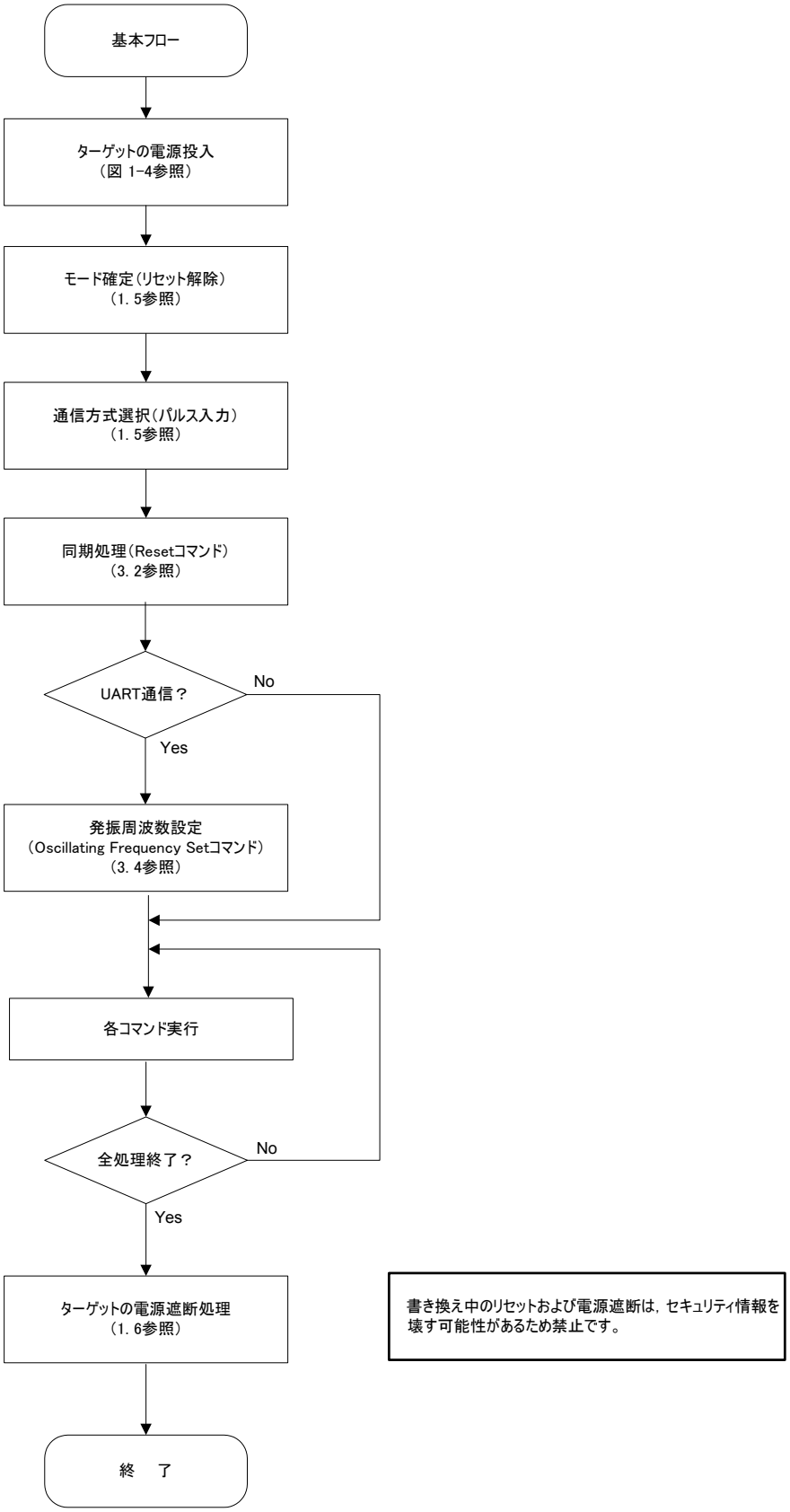
図1－5　フラッシュ・メモリ・プログラミングモードの終了手順



1.7　フラッシュ・メモリ書き換えコマンド・フロー

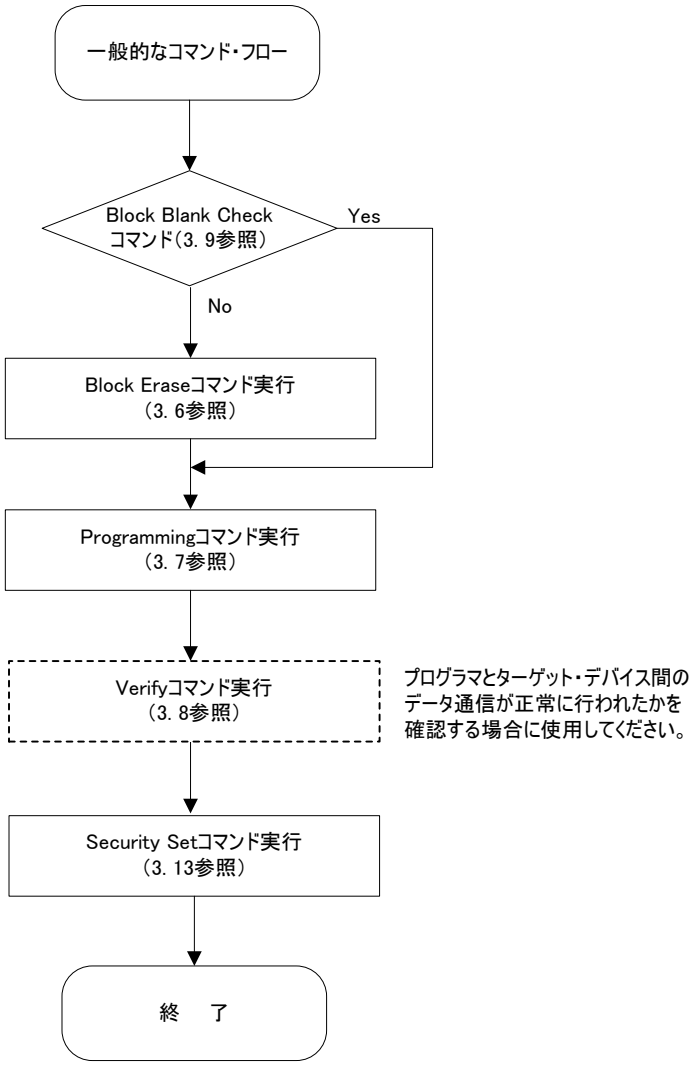
プログラマにてフラッシュ・メモリの書き換えを行う際の基本フロー・チャートを次に示します。
下記の基本フローで示したコマンドのほかにVerifyコマンドやChecksumコマンドをサポートしています。

図1－6　フラッシュ処理の基本フロー・チャート



備考　各コマンド実行の例は図1－7に示します。

図1－7　フラッシュ・メモリ書き換え時の一般的なコマンド・フロー



第2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/Lx2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/Lx2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2－1　コマンド・フレームのフォーマット

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

図2－2　データ・フレームのフォーマット

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

表2－1　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	

2. 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，4. 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，5. 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。

2. 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，4. 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，5. 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。

2. 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，4. 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，5. 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3章　コマンド処理説明

3. 1　Statusコマンド

3. 1. 1　説明

書き込み／消去などの各コマンド発行後の78K0/Lx2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/Lx2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/Lx2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。

3. 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3－1，そのコマンドに対するステータス・フレームは図3－2のようになります。

図3－1　Statusコマンド・フレーム（プログラマから78K0/Lx2へ）

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

図3－2　Statusコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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/Lx2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（CSI）通信方式の場合は，5. 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/Lx2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。

3. 2　Resetコマンド

3. 2. 1　説明

通信方式設定後に，プログラマと78K0/Lx2間の通信が確立されたことを確認します。

78K0/Lx2との通信方式にUART通信を選択した場合，プログラマと78K0/Lx2は同じボー・レートである必要がありますが，78K0/Lx2は自身のボー・レート生成クロック（f_Xまたはf_{EXCLK}）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/Lx2はその“ 00H ”のロウ・レベル幅を測定し2回の平均値を計算することで初めて自身のボー・レート生成クロック周波数を判別できます。それによって，ボー・レートの設定が可能になり同期検出が行えるようになります。

3. 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3－3，そのコマンドに対するステータス・フレームは図3－4のようになります。

図3－3　Resetコマンド・フレーム（プログラマから78K0/Lx2へ）

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

図3－4　Resetコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 4　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，5. 5　Resetコマンドをお読みください。

3. 3　Baud Rate Setコマンド

78K0/Lx2は，Baud Rate Setコマンドをサポートしていません。

78K0/Lx2でUART通信を行う場合，Oscillating Frequency Setコマンドまでの送信は，9600 bpsで通信を行います。そして，ステータス受信時からは，115200 bpsに通信レートが変更となります。以降は115200 bps固定です。

3. 4　Oscillating Frequency Setコマンド

3. 4. 1　説明

UART通信を使用する場合に，fxまたはfEXCLKの周波数を指定します。

78K0/Lx2は，受信した周波数データを使用し，115200 bpsを生成します。

注意　78K0/Lx2でUART通信を行う場合，Oscillating Frequency Setコマンドまでの送信は，9600 bpsで送信を行います。そして，ステータス受信時からは，115200 bpsに通信レートが変更となります。以降は115200 bps固定です。

3. 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3－5，そのコマンドに対するステータス・フレームは図3－6のようになります。

図3－5　Oscillating Frequency Setコマンド・フレーム（プログラマから78K0/Lx2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 = 6 × 0.1 × 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3－6　Oscillating Frequency Setコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 5　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，5. 6　Oscillating Frequency Setコマンドをお読みください。

3. 5　Chip Eraseコマンド

3. 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Chip Eraseコマンド実行不可となっている場合は消去できません（3. 13　Security Setコマンド参照）。

3. 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3－7，そのコマンドに対するステータス・フレームは，図3－8のようになります。

図3－7　Chip Eraseコマンド・フレーム（プログラマから78K0/Lx2へ）

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

図3－8　Chip Eraseコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 6　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，5. 7　Chip Eraseコマンドをお読みください。

3. 6　Block Eraseコマンド

3. 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。

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

ただし，セキュリティ設定により消去禁止となっている場合は消去できません（3. 13　Security Setコマンド参照）。

3. 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3－9，そのコマンドに対するステータス・フレームは図3－10のようになります。

図3－9　Block Eraseコマンド・フレーム（プログラマから78K0/Lx2へ）

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

備考　SAH - SAL　：ブロック消去開始アドレス
SAH　：開始アドレスHigh（ビット23 － ビット16）（00H固定）
SAM　：開始アドレスMiddle（ビット15 － ビット8）（00H固定）
SAL　：開始アドレスLow（ビット7 － ビット0）（00H固定）
EAH - EAL　：ブロック消去終了アドレス（内蔵フラッシュ・メモリの最終アドレス）
EAH　：最終アドレスHigh（ビット23 － ビット16）
EAM　：最終アドレスMiddle（ビット15 － ビット8）
EAL　：最終アドレスLow（ビット7 － ビット0）

図3－10　Block Eraseコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 7　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，5. 8　Block Eraseコマンドをお読みください。

3. 7　Programmingコマンド

3. 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・メモリに書き込み，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/Lx2のファームウエアは自動的に内部ベリファイを実行するので，さらにこの内部ベリファイに対するステータス・コードの確認が必要となります。

3. 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3－11，そのコマンドに対するステータス・フレームは図3－12のようになります。

図3－11　Programmingコマンド・フレーム（プログラマから78K0/Lx2へ）

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

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

図3－12　Programmingコマンドに対するステータス・フレーム（78K0/Lx2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

3. 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3－13，そのデータに対するステータス・フレームは図3－14のようになります。

図3－13　書き込みを行うデータ・フレーム（プログラマから78K0/Lx2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3－14　データ・フレームに対するステータス・フレーム（78K0/Lx2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

3. 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3－15のようになります。

図3－15　全データ転送完了後のステータス・フレーム（78K0/Lx2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/Lx2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，4. 8　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，5. 9　Programmingコマンドをお読みください。

3. 8　Verifyコマンド

3. 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/Lx2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。

3. 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3－16，そのコマンドに対するステータス・フレームは図3－17のようになります。

図3－16　Verifyコマンド・フレーム（プログラマから78K0/Lx2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3－17　Verifyコマンドに対するステータス・フレーム（78K0/Lx2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

3. 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3－18，そのデータに対するステータス・フレームは図3－19のようになります。

図3－18　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/Lx2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3－19　データ・フレームに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 9　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，5. 10　Verifyコマンドをお読みください。

3. 9　Block Blank Checkコマンド

3. 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。

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

3. 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3－20，そのコマンドに対するステータス・フレームは図3－21のようになります。

図3－20　Block Blank Checkコマンド・フレーム（プログラマから78K0/Lx2へ）

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	32H （Block Blank Check）	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考　SAH - SAL　：ブロック・ブランク・チェック開始アドレス（任意のブロックの先頭アドレス）
SAH　：開始アドレスHigh（ビット23 － ビット16）
SAM　：開始アドレスMiddle（ビット15 － ビット8）
SAL　：開始アドレスLow（ビット7 － ビット0）
EAH － EAL　：ブロック・ブランク・チェック終了アドレス（任意のブロックの最終アドレス）
EAH　：最終アドレスHigh（ビット23 － ビット16）
EAM　：最終アドレスMiddle（ビット15 － ビット8）
EAL　：最終アドレスLow（ビット7 － ビット0）

図3－21　Block Blank Checkコマンドに対するステータス・フレーム（78K0/Lx2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/Lx2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，4. 10　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，5. 11　Block Blank Checkコマンドをお読みください。

3. 10　Silicon Signatureコマンド

3. 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/Lx2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。

3. 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3－22，そのコマンドに対するステータス・フレームは図3－23のようになります。

図3ー22　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/Lx2へ）

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

図3ー23　Silicon Signatureコマンドに対するステータス・フレーム（78K0/Lx2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

3. 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3－24のようになります。

図3－24　シリコン・シグネチャ・データ・フレーム（78K0/Lx2からプログラマへ）

STX	LEN	Data								SUM	ETX
02H	n	VEN	MET	MSC	DEC	END	INVALID DATA	SCF	BOT	Checksum	03H

備考1.　n（LEN）　：データ長
VEN　：ベンダー・コード（NEC：10H）
MET　：マクロ拡張コード
MSC　：マクロ機能コード
DEC　：デバイス拡張コード
END　：内蔵フラッシュROM最終アドレス
INVALID DATA　：長さ10バイトの無効データです。
SCF　：セキュリティ・フラグ情報
BOT　：ブート・ブロック番号（03H固定）

2. 上記ベンダー・コード（VEN），拡張コード（MET），機能コード（MSC），デバイス拡張コード（DEC），内蔵フラッシュROM最終アドレス（END），セキュリティ・フラグ情報（SCF）は，下位7ビットをデータ本体，上位1ビットを奇数パリティとして使用します。次ページに例を示します。

表3－1　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1]	実際の値	パリティ付加
	VEN	ベンダー・コード（NEC）	1	10H　(00010000B)	10H	あり
	MET	拡張コード（78K0/Lx2 固定値）	1	7FH　(01111111B)	7FH	あり
	MSC	機能コード（78K0/Lx2 固定値）	1	04H　(00000100B)	04H	あり
★	DEC	デバイス拡張コード（78K0/Lx2 固定値）	1	7CH　(01111100B)	7CH	あり
	END	フラッシュ ROM 最終アドレス（下位バイトから抽出されます）	3	7FH　(01111111B)	005FFFH	あり[注2]
				BFH　(11011111B)		
				01H　(00000001B)		
	INVALID DATA	無効データ	10	–	–	–
	SCF	セキュリティ・フラグ情報	1	任意	左欄に同じ	あり[注3]
★	BOT	ブート・ブロック・クラスタ最終ブロック番号（固定値）	1	03H　(00000011B)	03H	なし

注1.　1や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）

2. ENDフィールドの場合は次のようにパリティ計算を行います。
（最終アドレス = 005FFFHの場合）

① 下位から7ビットごとに分割します（上位3ビットは捨てる）

```
  0   0          5   F          F   F
00000000      01011111      11111111
                  ↓
  ~~000~~  0000001  0111111  1111111
```

② 奇数パリティ・ビットを最上位ビットに付加します

```
  p0000001  p01111111  p1111111（p = 奇数パリティ・ビット）
= 0000001  10111111  01111111
= 01 BF 7F
```

③ 下位バイトを先頭にします

```
7F BF 01
```

マイコンから送信されたENDフィールドを実際のアドレスに戻す手順を次に示します。

① 下位バイトを先頭にします

7F BF 01
↓
01 BF 7F

② 各バイトにおいて“1”の数が奇数であることを確認します（他のタイミングでも可）

③ パリティ・ビットを外し，最上位に3ビットの“0”を付加します

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

④ 下位より8ビットごとの数字に変換します

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

よって，ENDフィールドに“7F BF 01”が与えられた場合，実際の最終アドレスは，005FFFHとなります。

注3. Security Setコマンドにてセキュリティ・フラグ情報を設定する際は，最上位ビットは1固定となりますが，Silicon Signatureコマンドにて読み出されたセキュリティ・フラグ情報は，最上位ビットが奇数パリティとなっています。

通信方式ごとの，プログラマと78K0/Lx2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，4. 11　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，5. 12　Silicon Signatureコマンドをお読みください。

3. 10. 4　78K0/Lx2シリコン・シグネチャー覧

表3－2　78K0/Lx2のシリコン・シグネチャ・データー覧

項　　目	内　　容	長さ（バイト）	データ（Hex）
ベンダー・コード	NEC	1	10
拡張コード	拡張コード	1	7F
機能情報	機能情報	1	04
デバイス情報	デバイス情報	1	7C
内蔵フラッシュ ROM の最終アドレス	（7 ビット・データ＋奇数パリティ・ビット）×3	3	注
無効データ	–	10	–
セキュリティ情報	セキュリティ情報	1	任意
ブート・ブロック番号	現在，選択されているブート・クラスタの最終ブロック番号	1	03

注　内蔵フラッシュROMの最終アドレス・リスト

項　　目	内　　容	長さ（バイト）	データ（Hex）
内蔵フラッシュ ROM の最終アドレス	16 K バイト（3FFFH）	3	7F7F80
	24 K バイト（5FFFH）		7FBF01
	32 K バイト（7FFFH）		7F7F01
	48 K バイト（BFFFH）		7F7F02
	60 K バイト（EFFFH）		7FDF83
	96 K バイト（17FFFH）		7F7F85
	128 K バイト（1FFFFH）		7F7F07

3. 11　Version Getコマンド

3. 11. 1　説明

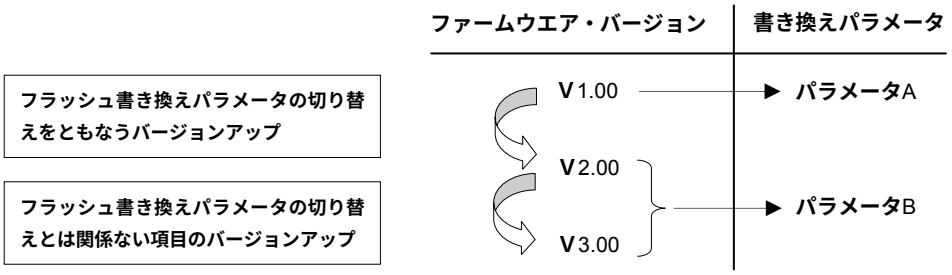
78K0/Lx2のデバイス・バージョン，ファームウエア・バージョン情報を取得します。

デバイス・バージョンは，00H固定です。

書き換え用パラメータを78K0/Lx2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



3. 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3－25，そのコマンドに対するステータス・フレームは図3－26のようになります。

図3－25　Version Getコマンド・フレーム（プログラマから78K0/Lx2へ）

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

図3－26　Version Getコマンドに対するステータス・フレーム（78K0/Lx2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

3. 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3－27のようになります。

図3－27　バージョン・データ・フレーム（78K0/Lx2からプログラマへ）

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

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

通信方式ごとの，プログラマと78K0/Lx2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，4. 12　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，5. 13　Version Getコマンドをお読みください。

3. 12　Checksumコマンド

3. 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。

3. 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3－28，そのコマンドに対するステータス・フレームは図3－29のようになります。

図3－28　Checksumコマンド・フレーム（プログラマから78K0/Lx2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3－29　Checksumコマンドに対するステータス・フレーム（78K0/Lx2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

3. 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3－30のようになります。

図3－30　チェックサム・データ・フレーム（78K0/Lx2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/Lx2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，4. 13　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，5. 14　Checksumコマンドをお読みください。

3. 13　Security Setコマンド

3. 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，ブート・ブロック書き換えの許可／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　セキュリティ設定後も許可から禁止への追加設定が可能です。ただし禁止から許可への変更は行えず，実行しようとした場合，Protect error（10H）**が発生します。禁止から許可への設定変更が必要な場合は，**Chip Erase**コマンドの実行によって全セキュリティ・フラグの初期化をする必要があります（**Block Erase**コマンドでは，セキュリティ・フラグの初期化はできません）。そのあとに設定変更を行ってください。**
ただし，チップ消去禁止，またはブート・ブロック・クラスタ書き換え禁止の設定をした場合，チップ消去自体が不可能になり，プログラマからは消去ができなくなります。プログラマの仕様としては，チップ消去禁止の設定を行う前に，設定実行の再確認をすることを推奨します。

3. 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3－31，そのコマンドに対するステータス・フレームは図3－32のようになります。

図3－31　Security Set**コマンド・フレーム（プログラマから**78K0/Lx2**へ）**

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

図3－32　Security Set**コマンドに対するステータス・フレーム（**78K0/Lx2**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**

3. 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3－33，そのデータに対するステータス・フレームは図3－34のようになります。

図3－33　セキュリティ・データ・フレーム（プログラマから78K0/Lx2へ）

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

備考　FLG　：セキュリティ・フラグ
　　BOT　：ブート・ブロック・クラスタ最終ブロック番号（03H固定）

図3－34　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/Lx2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

3. 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3－35のようになります。

図3－35　内部ベリファイ確認に対するステータス・フレーム（78K0/Lx2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3－3　セキュリティ・フラグ・フィールドの内容

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

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

表3－4　セキュリティ・フラグ・フィールドと各動作の禁止／許可

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

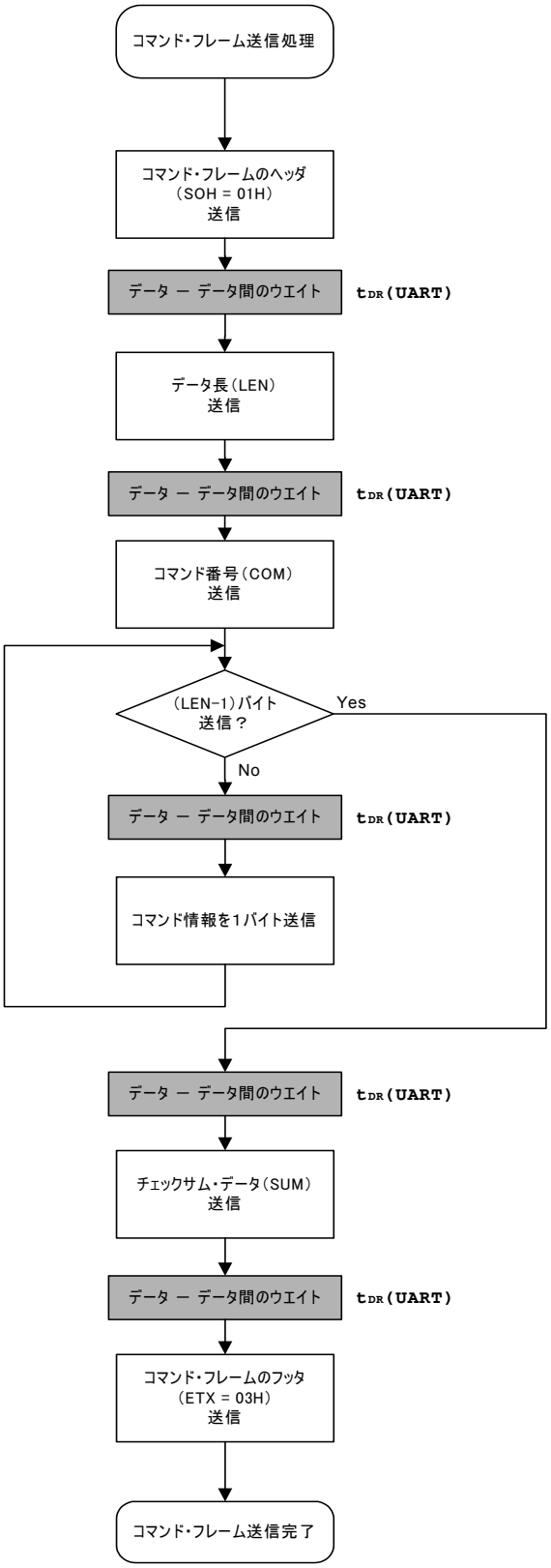
通信方式ごとの，プログラマと78K0/Lx2間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，次の節をお読みください。

- UART通信方式の場合は，4. 14　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，5. 15　Security Setコマンドをお読みください。

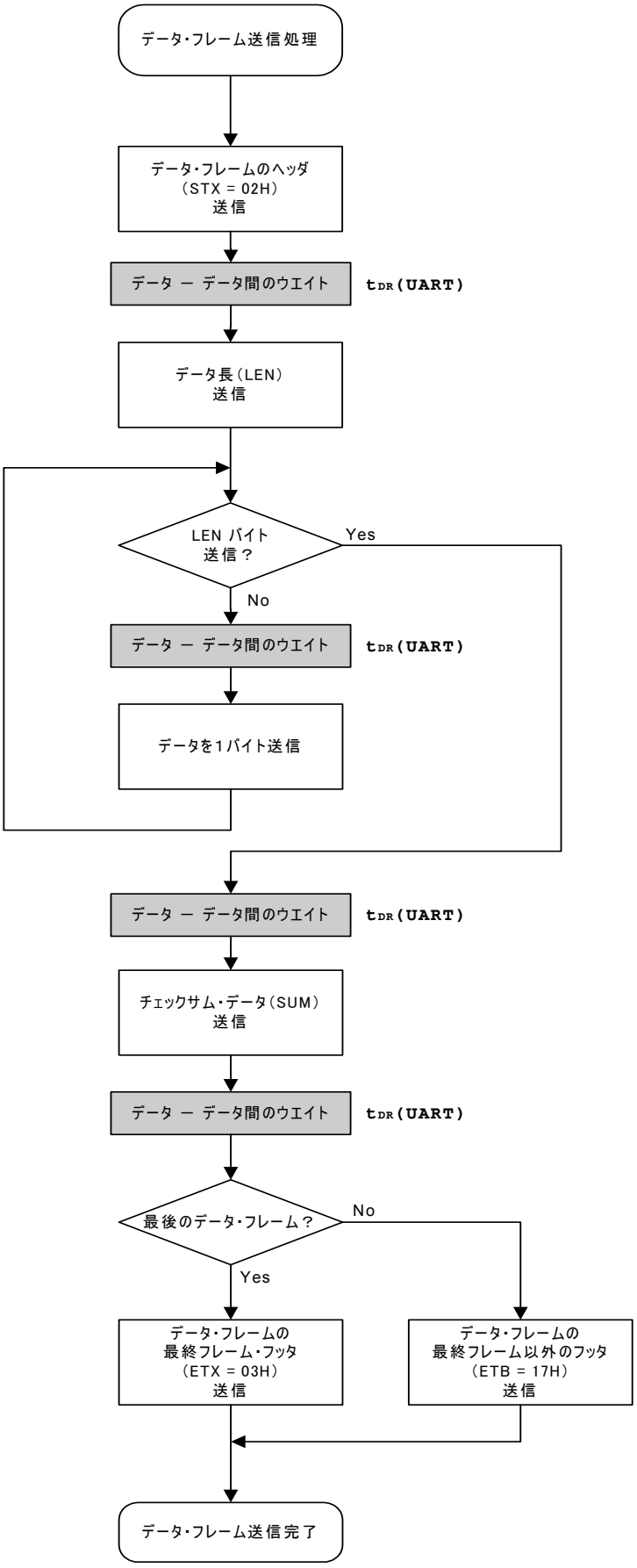
第4章　UART通信方式

この章のフロー・チャートで示した略号（t_{XX}およびt_{WTXX}）は，第6章　フラッシュ・メモリ・プログラミング・パラメータ特性における規格項目の略号です。各規格値については第6章　フラッシュ・メモリ・プログラミング・パラメータ特性を参照してください。

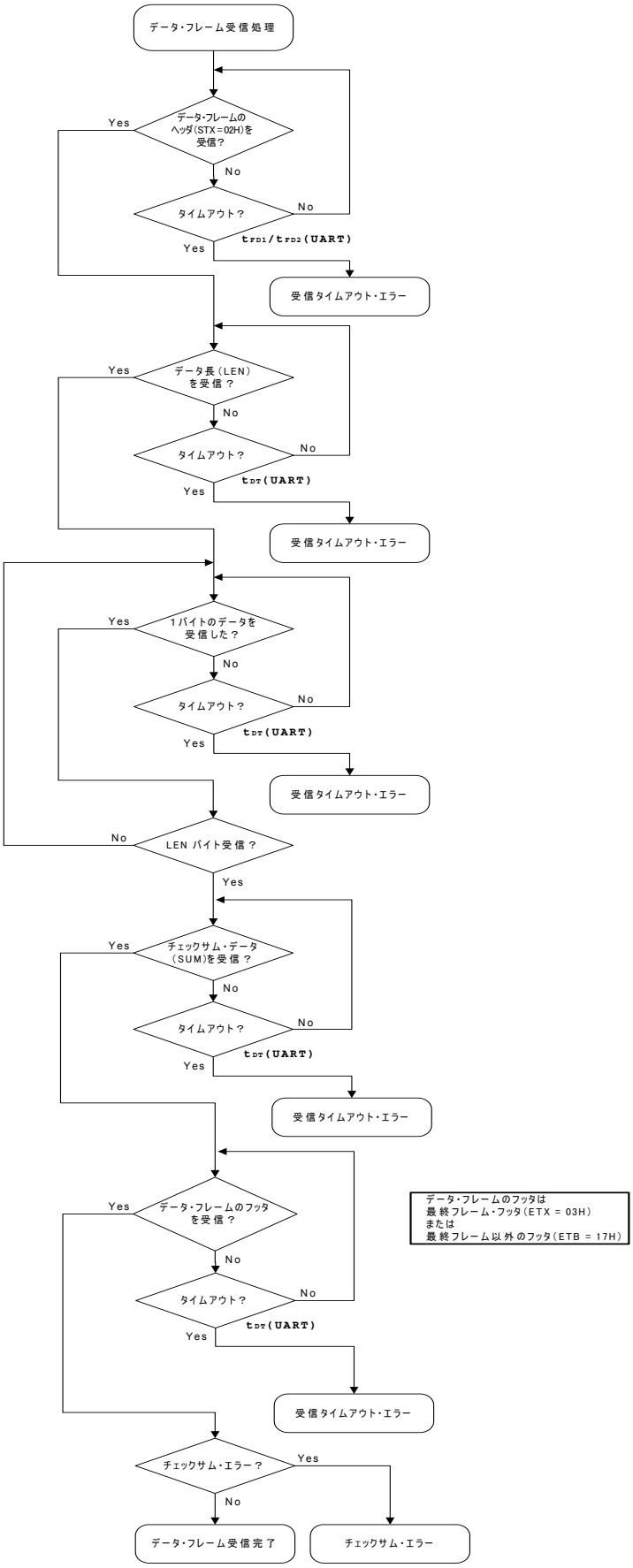
★ 4. 1　コマンド・フレーム送信処理のフロー・チャート



★ 4.2　データ・フレーム送信処理のフロー・チャート



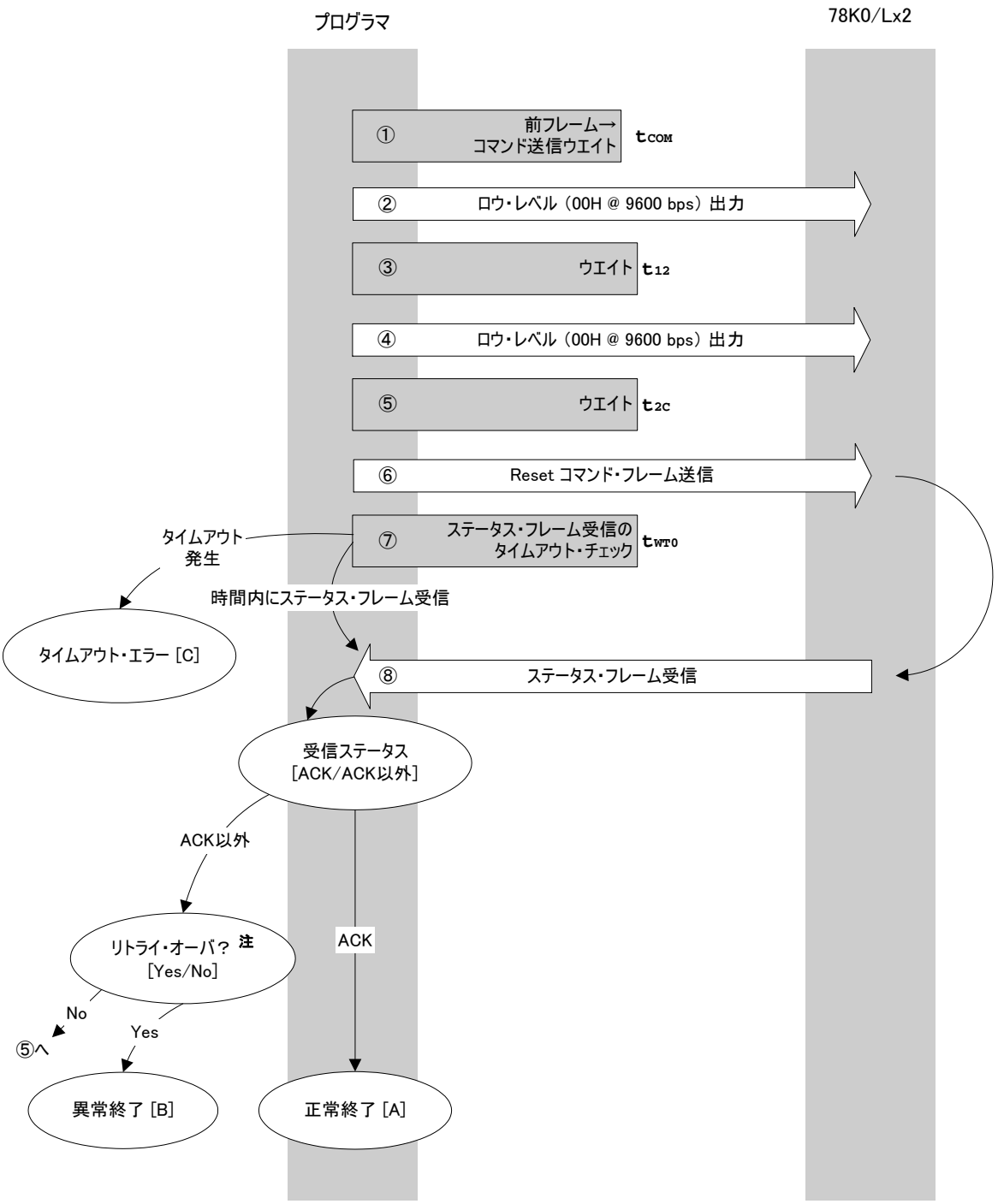
★ 4. 3　データ・フレーム受信処理のフロー・チャート



4. 4　Resetコマンド

4. 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。

4.4.2　処理手順説明

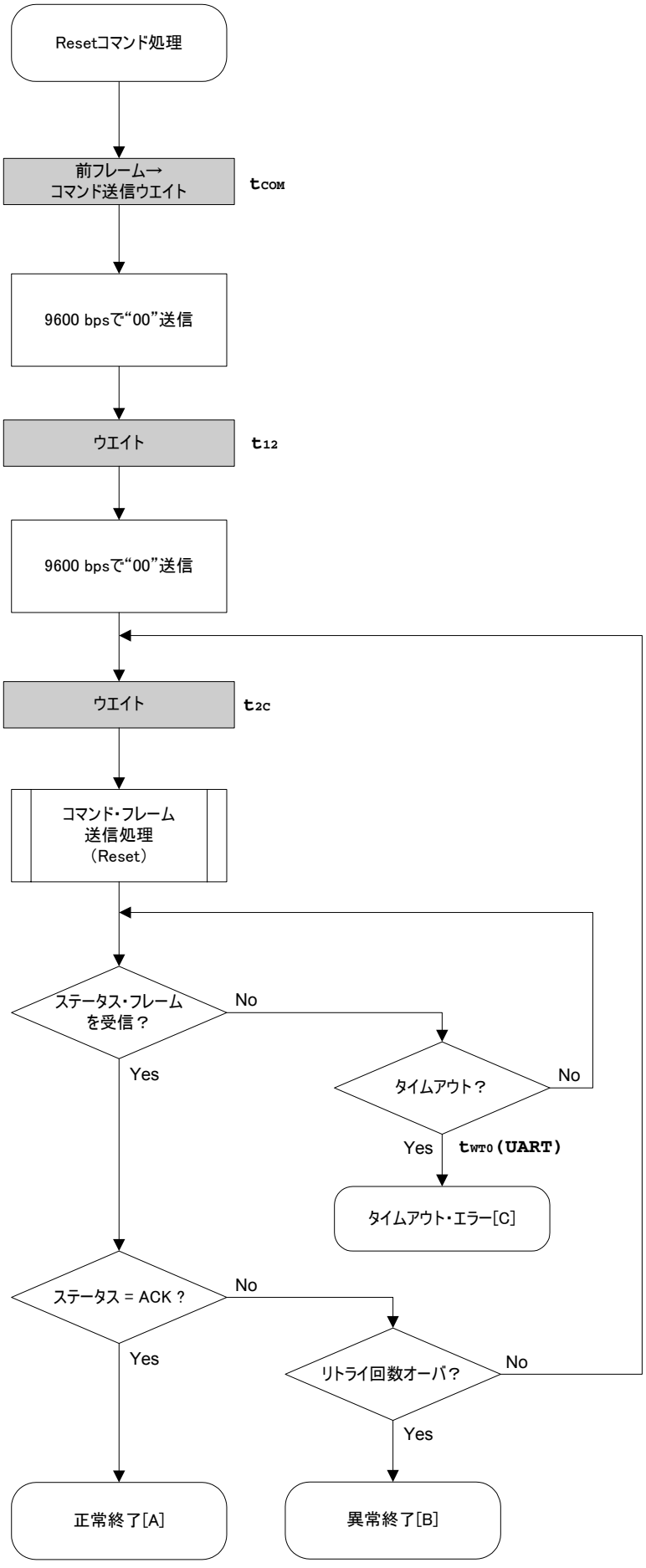
① 直前のフレームからコマンド処理開始前のウエイトをします（ウエイト時間t_{COM}）。
② ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
③ ウエイトします（ウエイト時間t_{12}）。
④ ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
⑤ ウエイトします（ウエイト時間t_{2C}）。
⑥ コマンド・フレーム送信処理にて Resetコマンド を送信します。
⑦ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合は タイムアウト・エラー [C] となります
（タイムアウト時間t_{WT0}）。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

4.4.3　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/Lx2 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。

4.4.4　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（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]

4.4.5　サンプル・プログラム

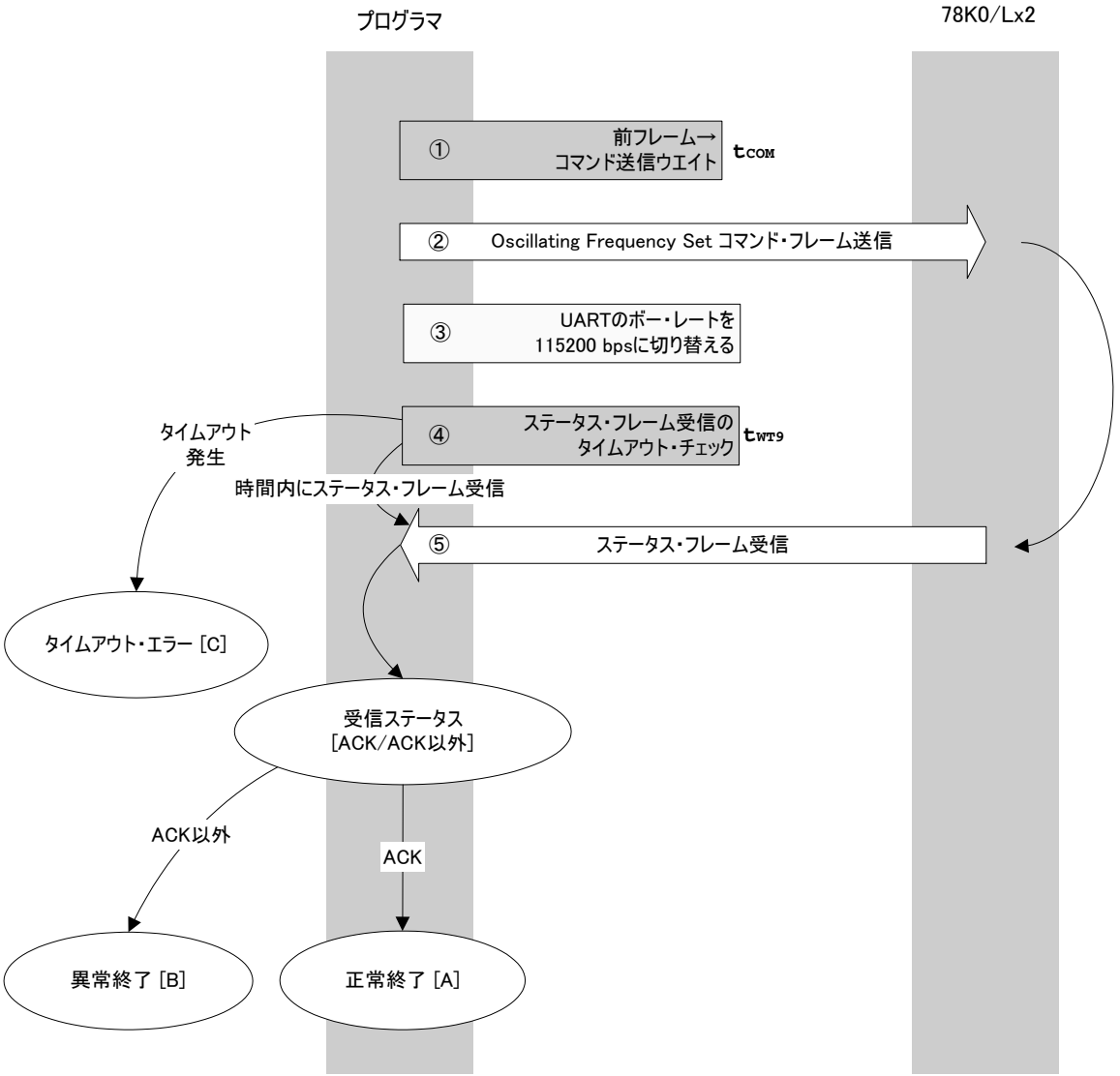
Resetコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* 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_TO);
           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;
}
```

4. 5　Oscillating Frequency Setコマンド

4. 5. 1　処理手順チャート

Oscillating Frequency Setコマンド処理手順



4.5.2　処理手順説明

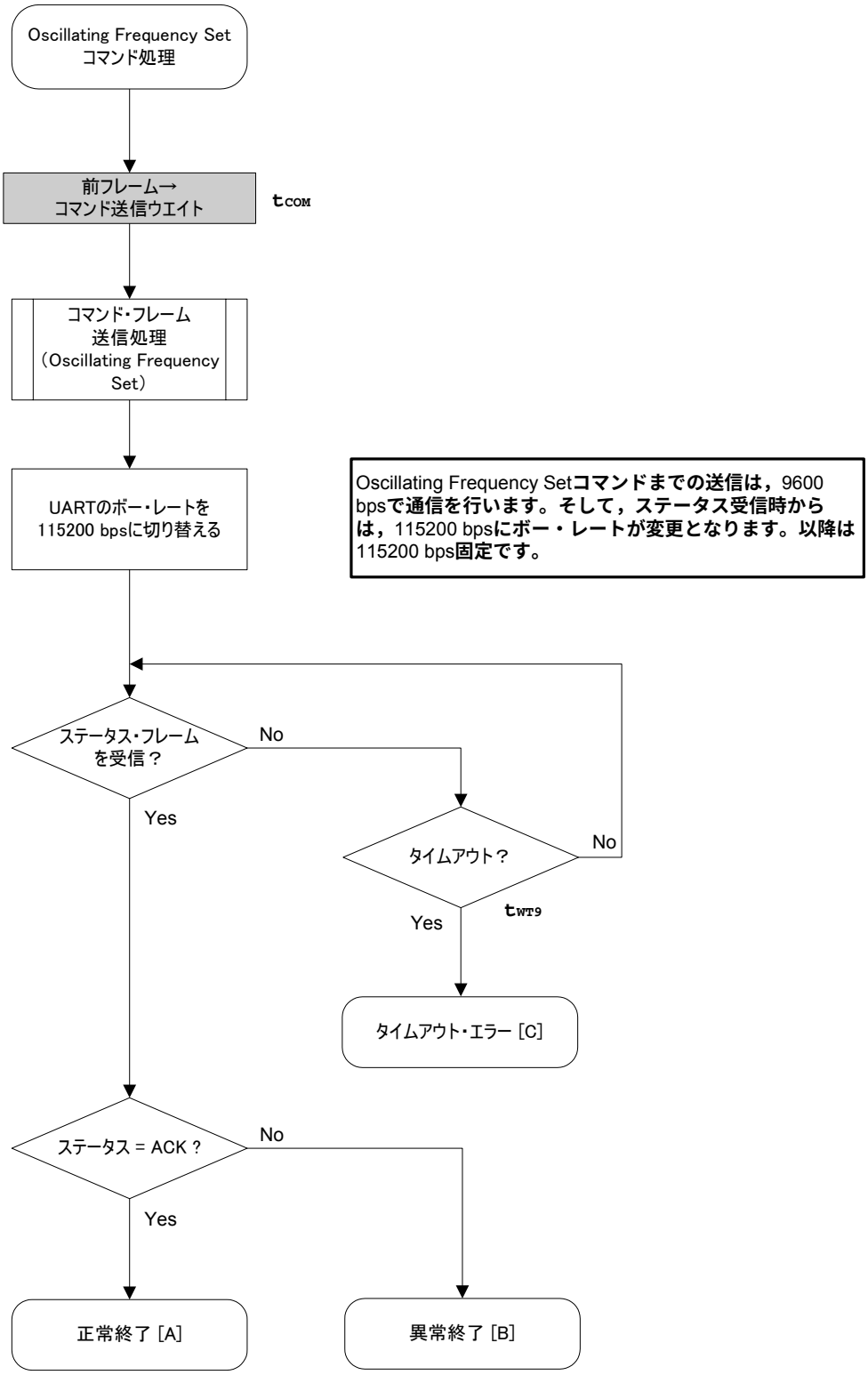
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Oscillating Frequency Setコマンドを送信します。
③ ステータス受信時からは，UART通信のボー・レートを115200 bpsに切り替えます。以降は，115200 bps固定です。
④ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C]となります（タイムアウト時間t_{WT9}）。
⑤ ステータス・コードをチェックします。

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。

4.5.3　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，78K0/Lx2 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。

4.5.4　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）
UARTのボー・レートを
115200 bpsに切り替える
Oscillating Frequency Setコマンドまでの送信は，9600 bpsで通信を行います。そして，ステータス受信時からは，115200 bpsにボー・レートが変更となります。以降は115200 bps固定です。
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT9
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]

4.5.5　サンプル・プログラム

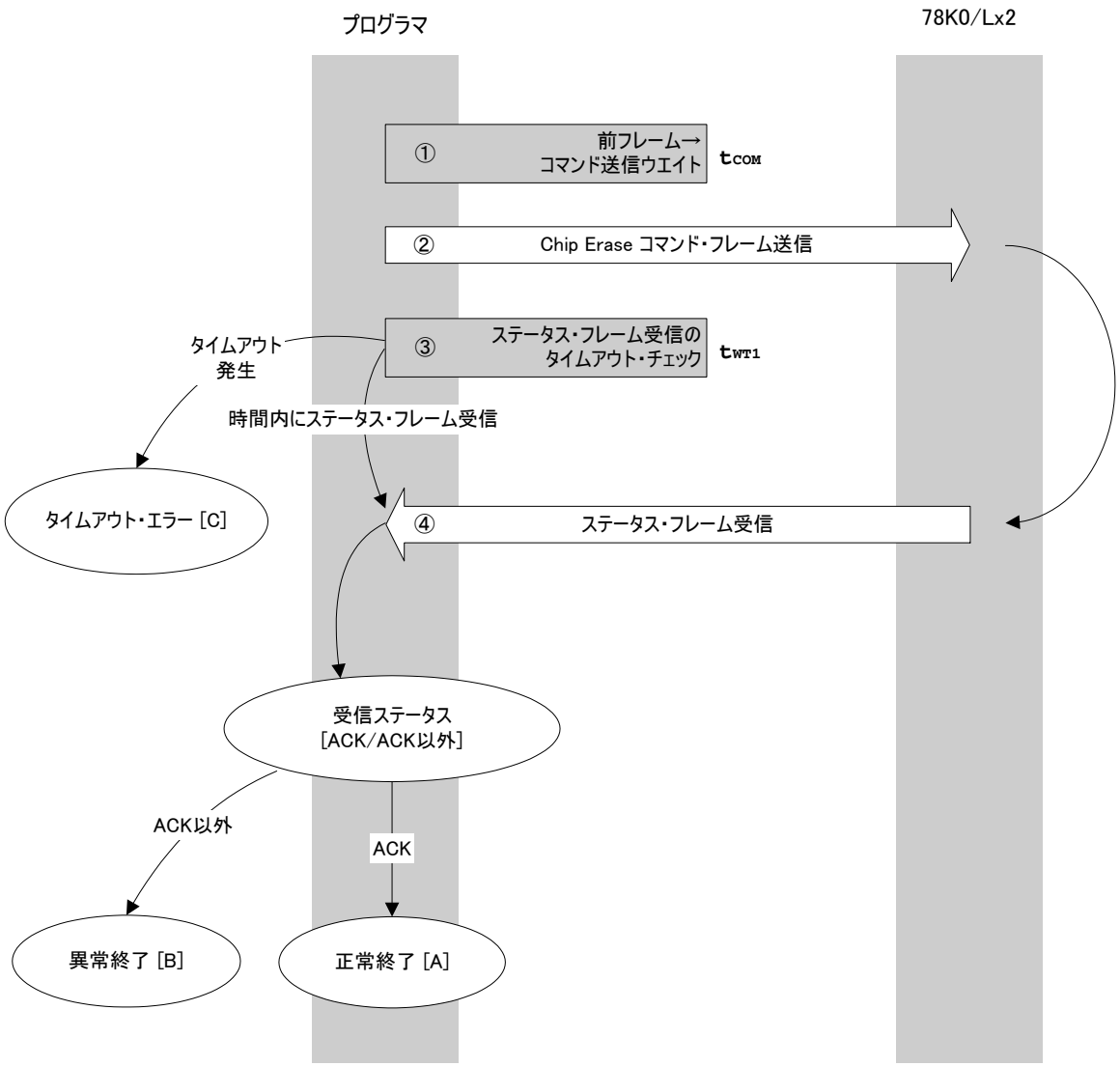
Oscillating Frequency Setコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* 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);

   set_flbaud(BR_115200);          // change baud-rate
   set_uart0_br(BR_115200);        // change baud-rate (h.w.)

   rc = get_sfrm_ua(fl_ua_sfrm, tWT9_TO); // 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;
}
```

4. 6　Chip Erase**コマンド**

4. 6. 1　**処理手順チャート**

Chip Eraseコマンド処理手順



4.6.2　処理手順説明

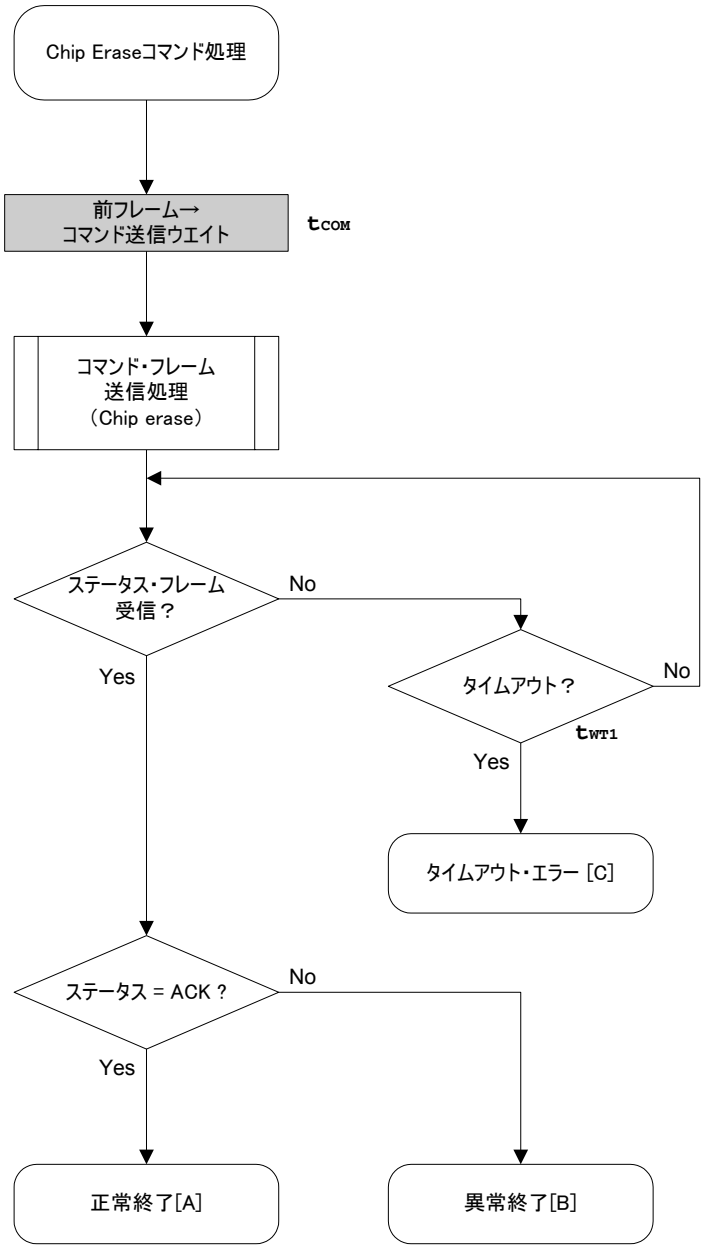
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Chip Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C] となります（タイムアウト時間t_{WT1}）。
④ ステータス・コードをチェックします。

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。

4.6.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，チップ消去が正常に実行されたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で，Chip Erase コマンドが禁止となる設定になっています。
	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。

4.6.4　フロー・チャート


Chip Eraseコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Chip erase）
ステータス・フレーム
受信？
No
Yes
タイムアウト？
No
tWT1
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]

4.6.5　サンプル・プログラム

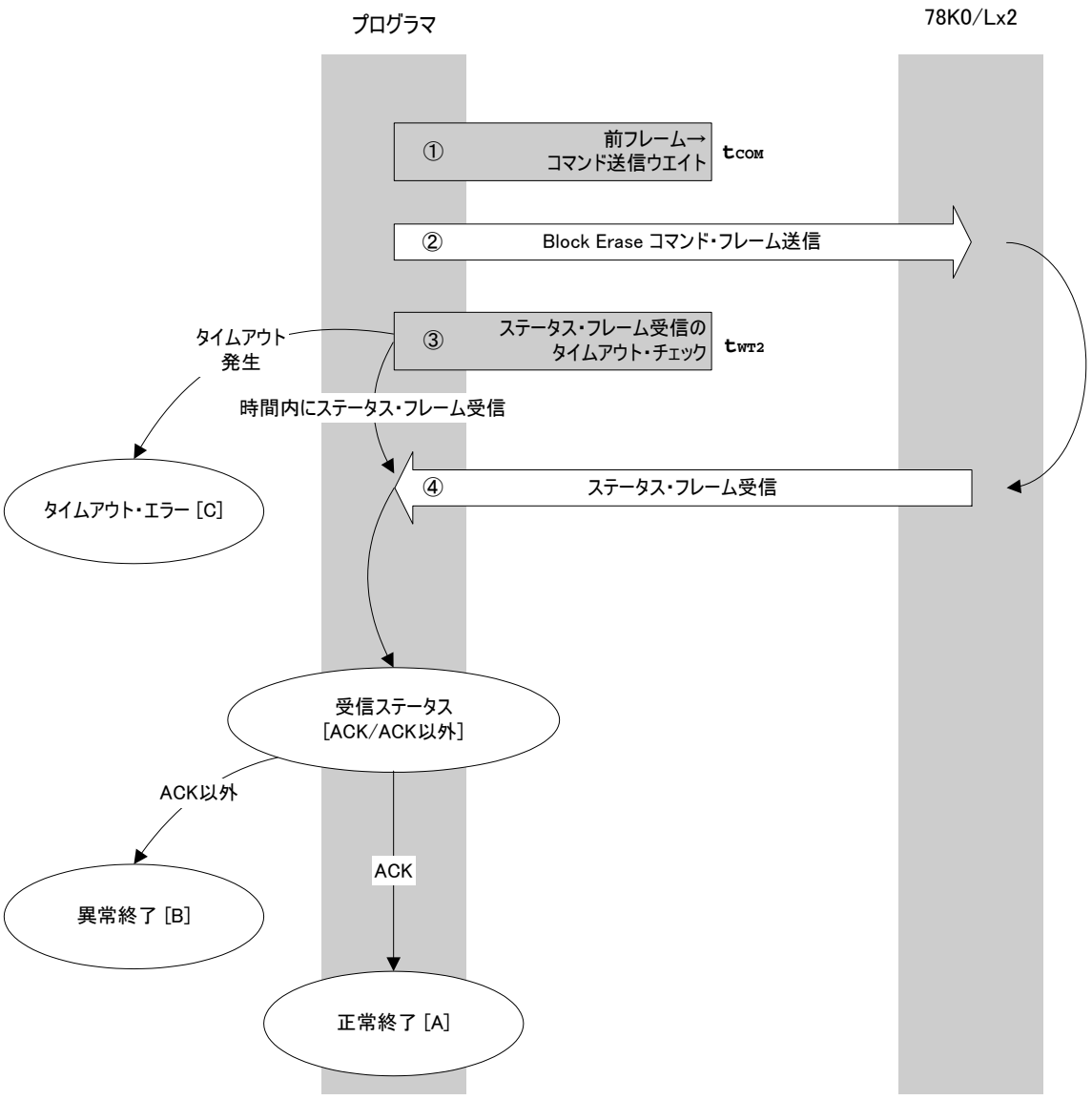
Chip Eraseコマンド処理のサンプル・プログラムです。

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


   fl_wait(tCOM);          // wait before sending command

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

   rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX); // get status frame
// switch(rc) {
//
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return  rc;

}
```

4. 7　Block Eraseコマンド

4. 7. 1　処理手順チャート

Block Eraseコマンド処理手順



4.7.2　処理手順説明

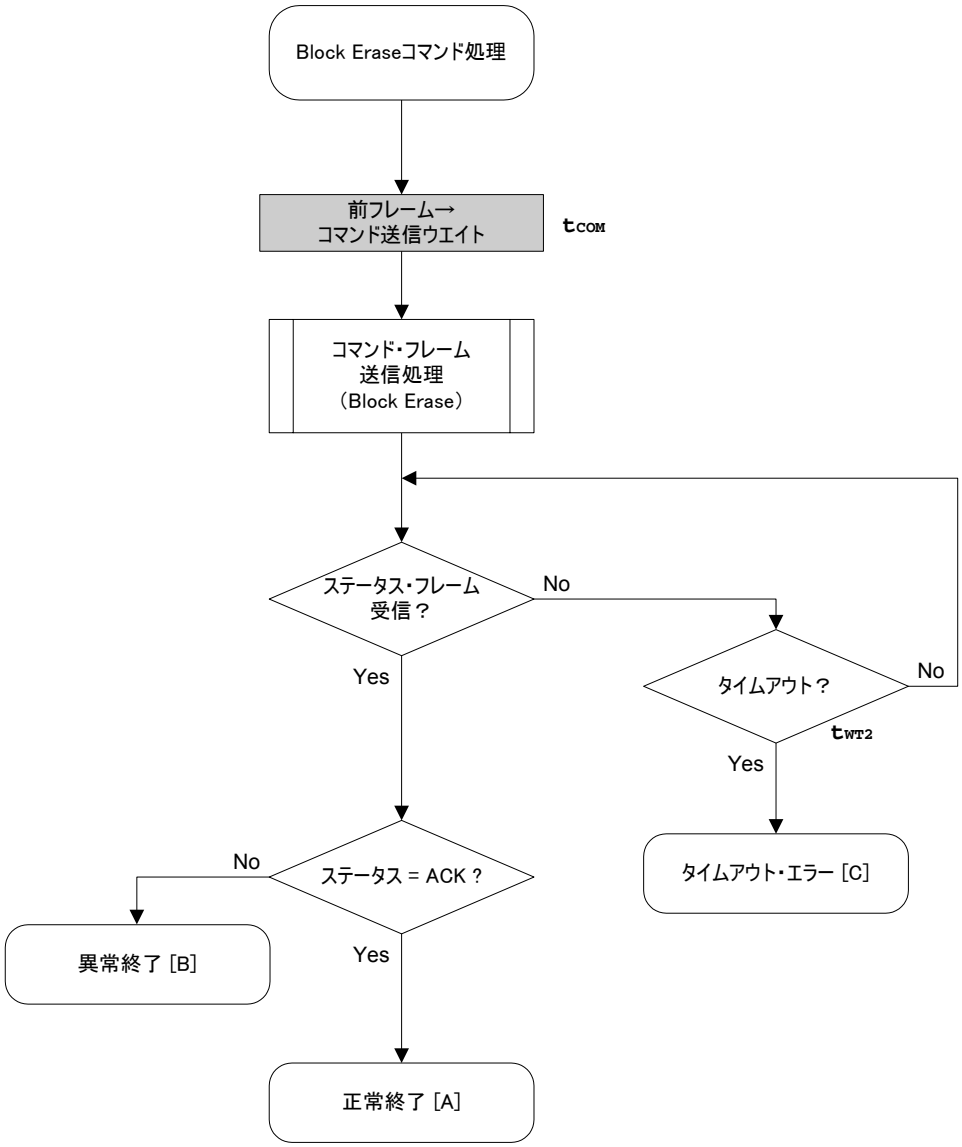
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Block Eraseコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は，タイムアウト・エラー[C] となります（タイムアウト時間t_{WT2}）。
④ ステータス・コードをチェックします。

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。

4.7.3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ブロック消去が正常に実行されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	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，Block Erase コマンドが禁止となる設定になっています。
	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。

4.7.4　フロー・チャート


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

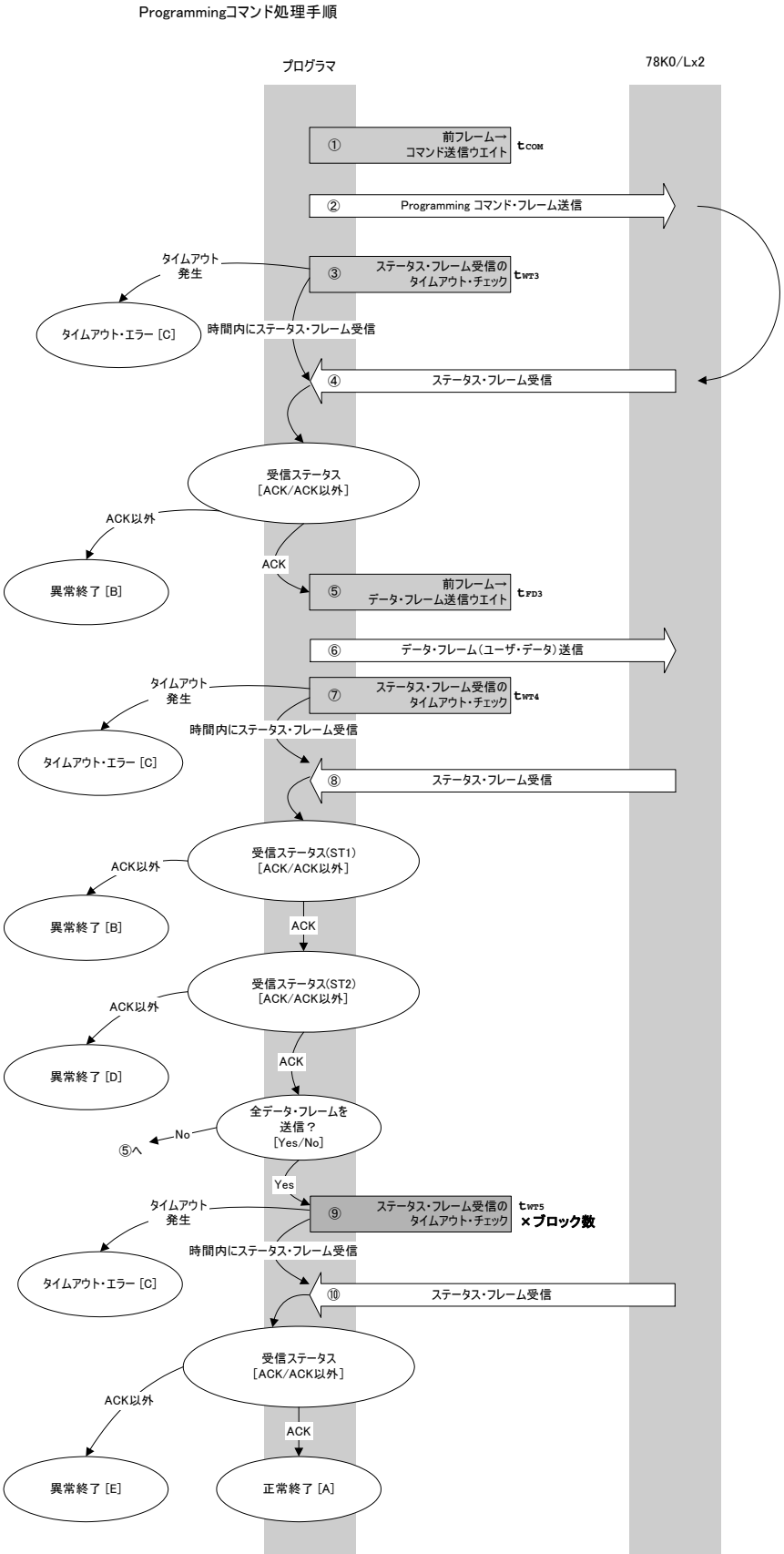
4.7.5　サンプル・プログラム

Block Eraseコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* Erase block command                                       */
/*                                                           */
/*************************************************************/
/* [i] u16 sblk    ... start block to erase (0...255)        */
/* [i] u16 eblk    ... end block to erase   (0...255)        */
/* [r] u16         ... error code                            */
/*************************************************************/
u16        fl_ua_erase_blk(u16 sblk, u16 eblk)
{

   u16     rc;
   u32     wt2_max;
   u32     top, bottom;

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

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

   wt2_max = make_wt2_max(sblk, eblk);

   fl_wait(tCOM);          // wait before sending command

   put_cmd_ua(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm); // send ERASE CHIP command

   rc = get_sfrm_ua(fl_ua_sfrm, wt2_max);  // get status frame


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

   return rc;

}
```

4. 8　Programmingコマンド

4. 8. 1　処理手順チャート



4.8.2 処理手順説明

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

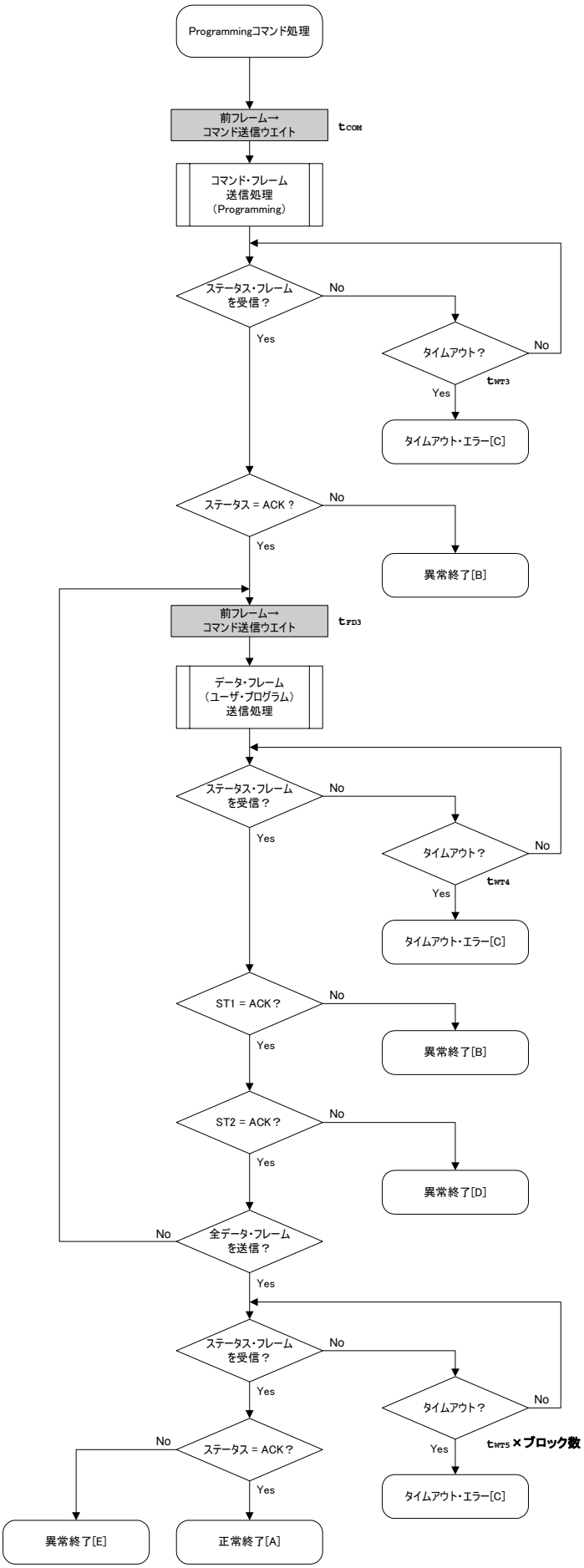
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT5}×ブロック数）。
⑩ ステータス・コードをチェックします。

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

4.8.3　終了時の内容

終了内容		ステータス・コード	内　　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始／終了アドレスがフラッシュ・メモリの範囲外です。または，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，Programming コマンドが禁止となる設定になっています。
	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]	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。

4.8.4　フロー・チャート


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

4.8.5 サンプル・プログラム

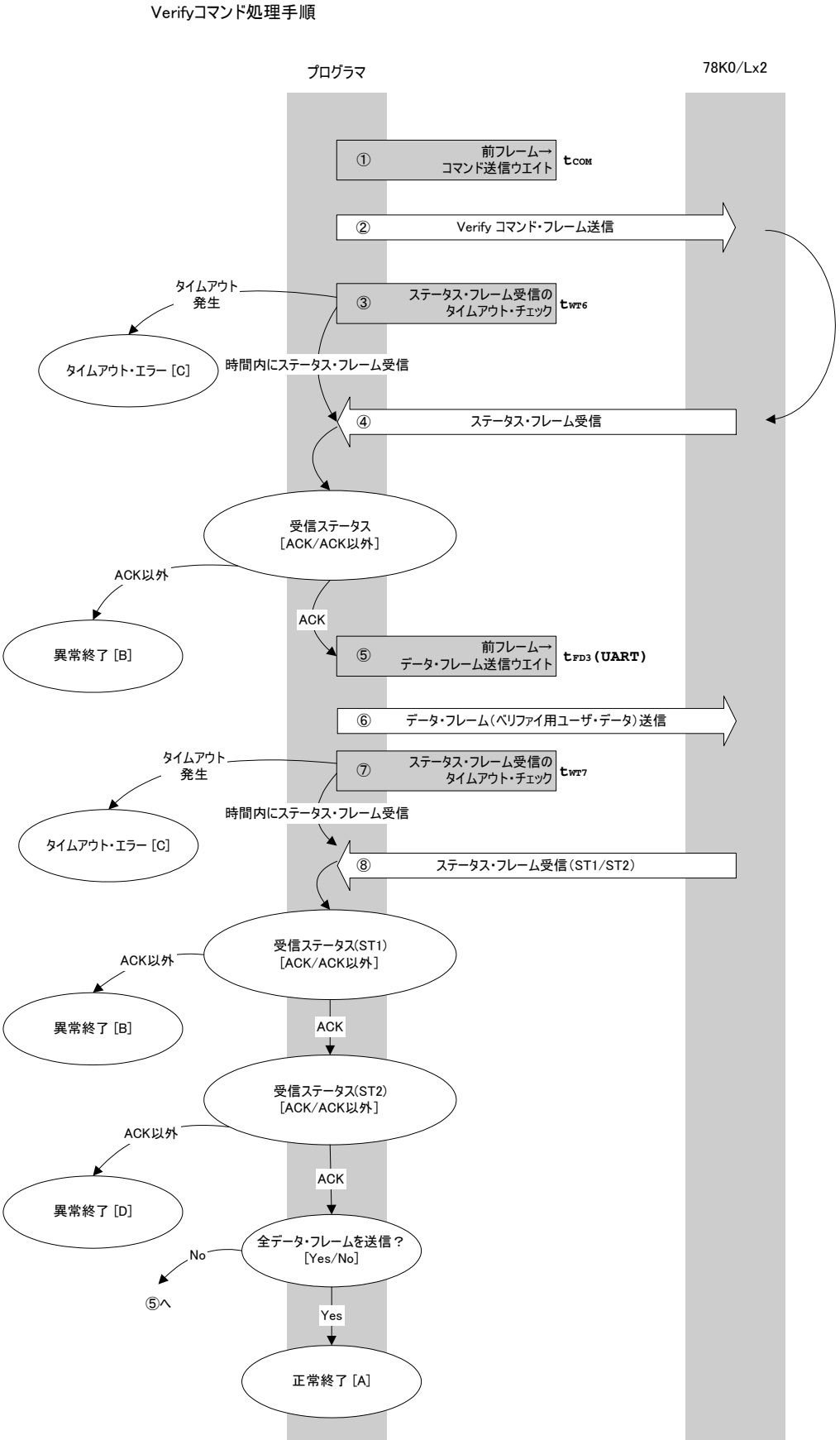
Programmingコマンド処理のサンプル・プログラムです。

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

#define            fl_st2_ua       (fl_ua_sfrm[OFS_STA_PLD+1])

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

   /*********************************************/
   /*      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_TO);  // 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_UA);                       // 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;
}
```

4. 9　Verifyコマンド

4. 9. 1　処理手順チャート



4.9.2　処理手順説明

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

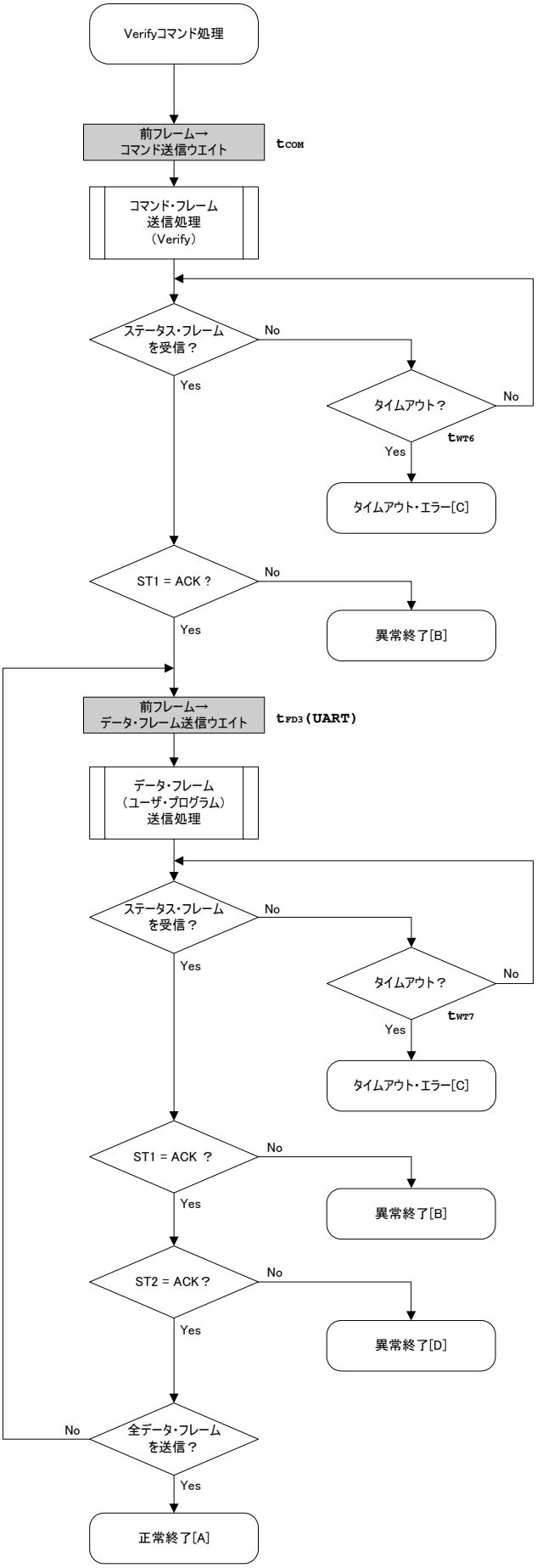
⑤ 直前のフレームからデータ・フレーム送信までのウエイトをします（ウエイト時間t_{FD3}（UART））。
⑥ データ・フレーム送信処理により，ベリファイ用ユーザ・データを送信します。
⑦ ユーザ・データ送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT7}）。
⑧ ステータス・コード（ST1/ST2）をチェックします（処理手順チャートやフロー・チャートも参照してください）。

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。

4.9.3　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，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。
	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。

4.9.4　フロー・チャート


Verifyコマンド処理
前フレーム→
コマンド送信ウエイト
tCOM
コマンド・フレーム
送信処理
（Verify）
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT6
Yes
タイムアウト・エラー[C]
ST1 = ACK ?
No
Yes
異常終了[B]
前フレーム→
データ・フレーム送信ウエイト
tPD3（UART）
データ・フレーム
（ユーザ・プログラム）
送信処理
ステータス・フレーム
を受信？
No
Yes
タイムアウト？
No
tWT7
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
正常終了[A]

4.9.5　サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

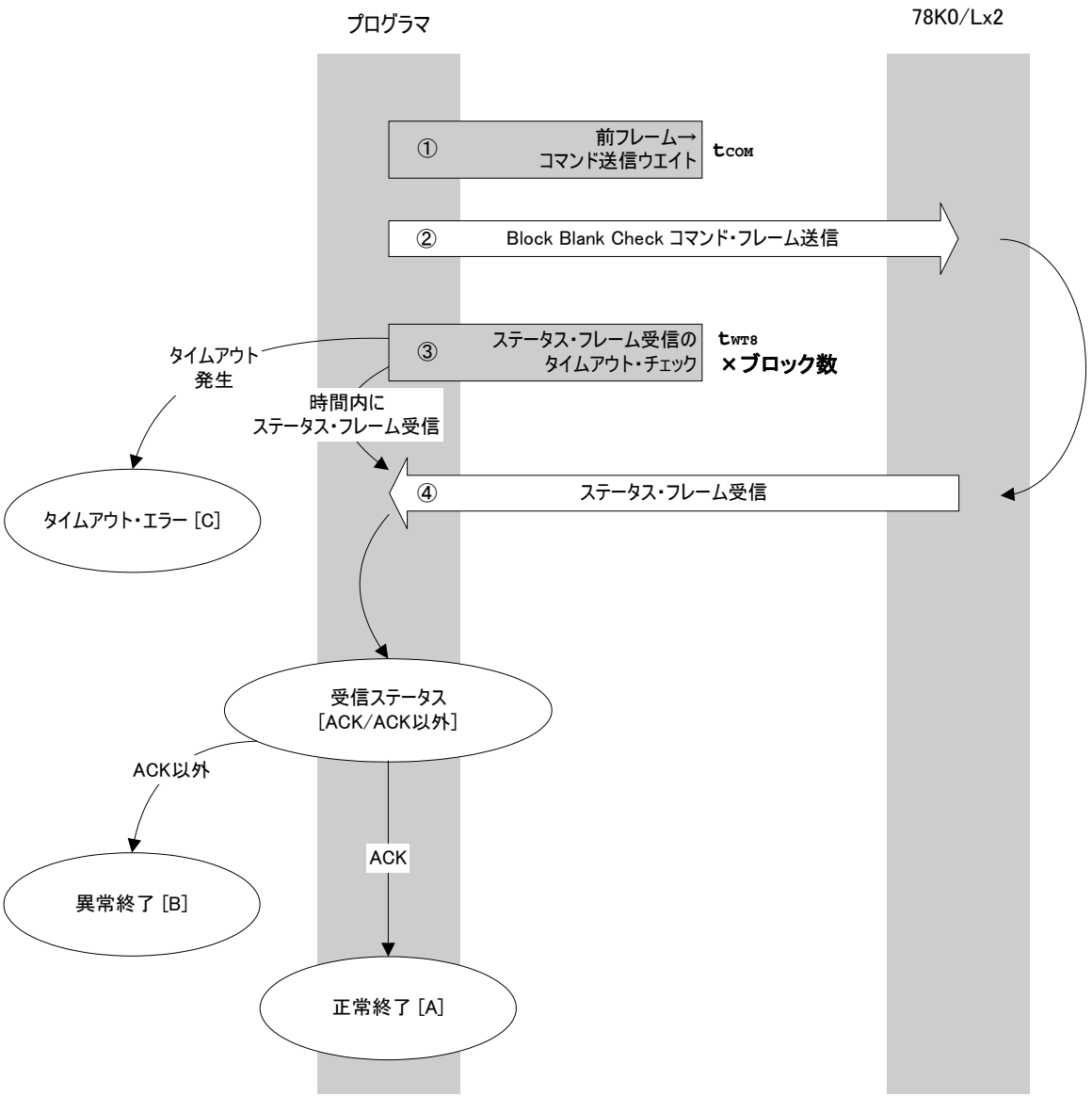
```
/*************************************************************/
/*                                                           */
/* Verify command                                            */
/*                                                           */
/*************************************************************/
/* [i] u32 top     ... start address                         */
/* [i] u32 bottom  ... end address                           */
/* [r] u16         ... error code                            */
/*************************************************************/
u16        fl_ua_verify(u32 top, u32 bottom)
{
   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_TO);         // 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_UA);
          put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data
```

```
            rc = get_sfrm_ua(fl_ua_sfrm, tWT7_TO);          // 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]
}
```

4. 10　Block Blank Checkコマンド

4. 10. 1　処理手順チャート

Block Blank Checkコマンド処理手順



4. 10. 2　処理手順説明

① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理にて Block Blank Checkコマンド を送信します。
③ コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は タイムアウト・エラー[C] となります
（タイムアウト時間t_{WT8}×ブロック数）。
④ ステータス・コードをチェックします。

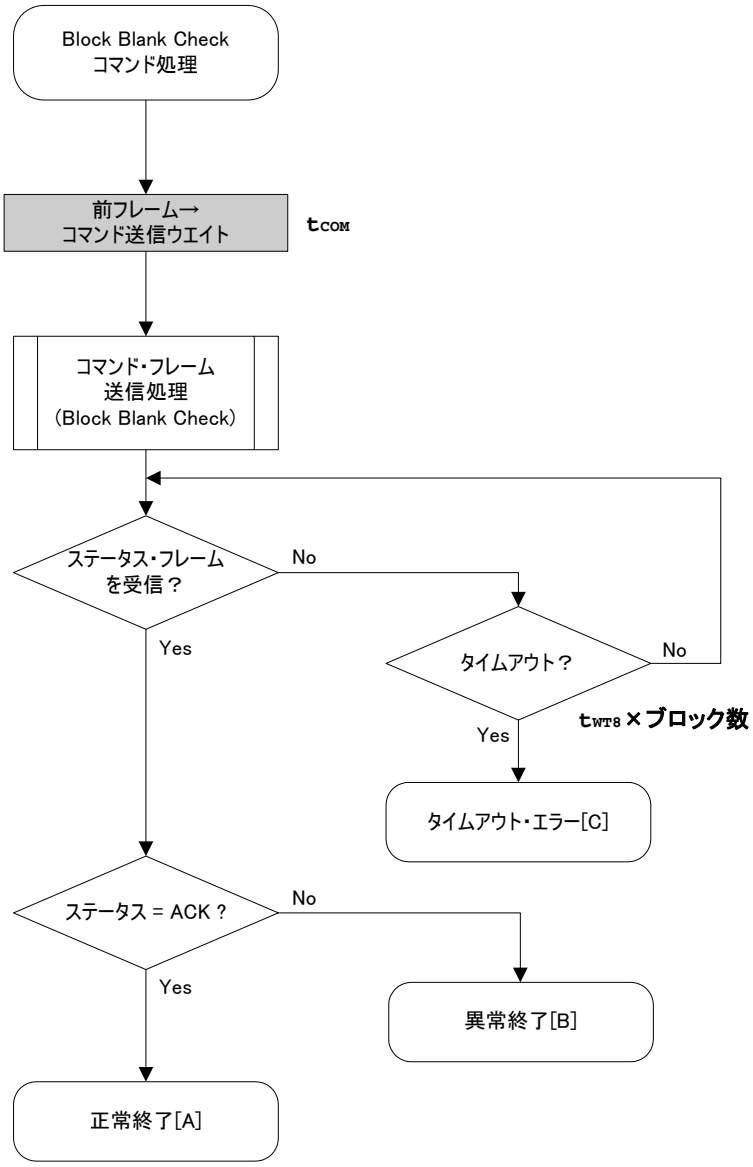
ST1 = ACKの場合　　：正常終了[A]です。

ST1 = ACK以外の場合　：異常終了[B]です。

4. 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，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。

4. 10. 4　フロー・チャート



4. 10. 5　サンプル・プログラム

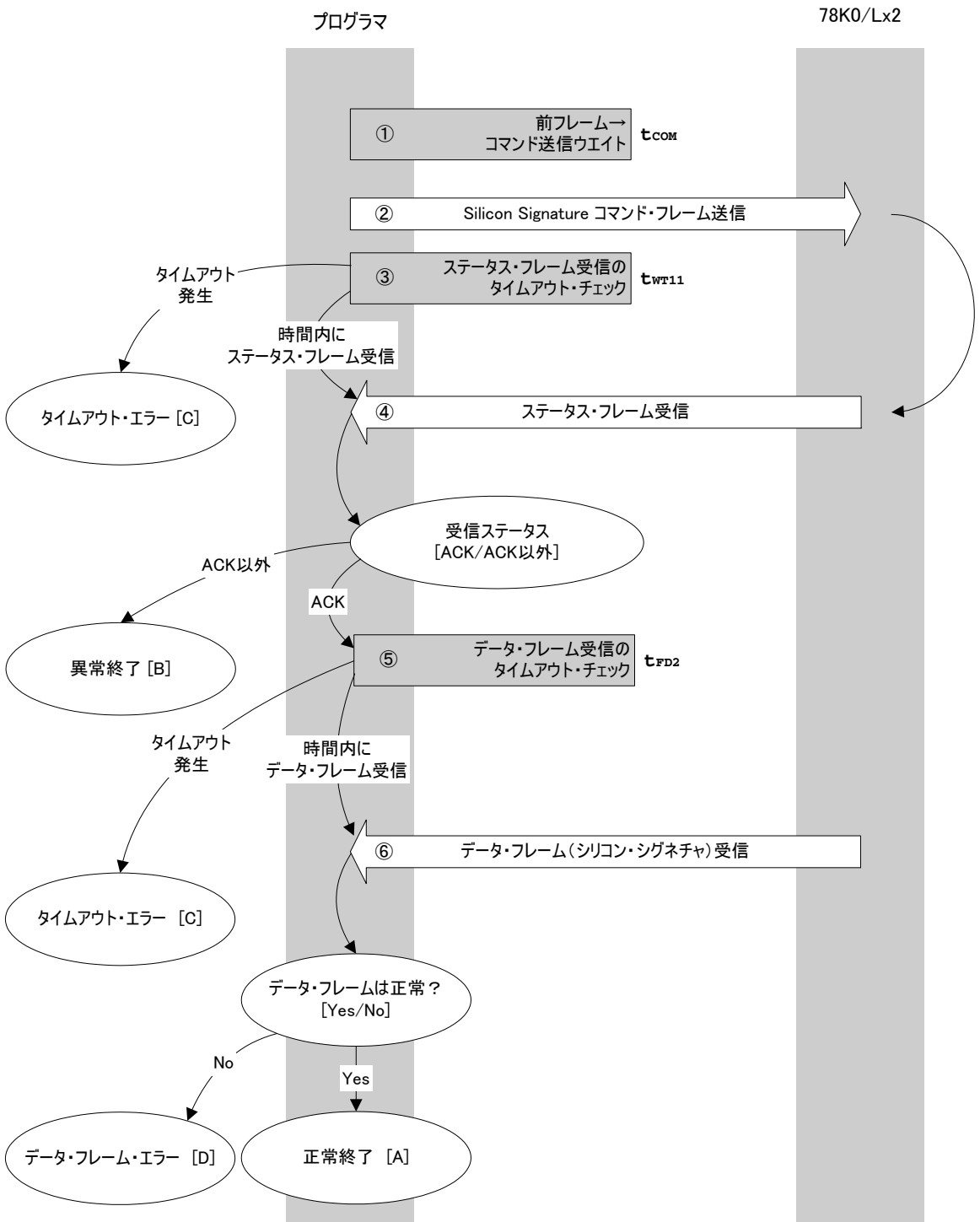
Block Blank Checkコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* Block blank check command                                 */
/*                                                           */
/*************************************************************/
/* [i] u32 top     ... start address                         */
/* [i] u32 bottom  ... end address                           */
/* [r] u16         ... error code                            */
/*************************************************************/
u16        fl_ua_blk_blank_chk(u32 top, u32 bottom)
{
   u16     rc;
   u16     block_num;

   set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
   block_num = get_block_num(top, bottom); // get block num

   fl_wait(tCOM);          // wait before sending command

   put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);


   rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num);    // get status frame
// switch(rc) {
//
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;

}
```

4. 11　Silicon Signatureコマンド

4. 11. 1　処理手順チャート

Silicon Signatureコマンド処理手順



4. 11. 2　処理手順説明

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

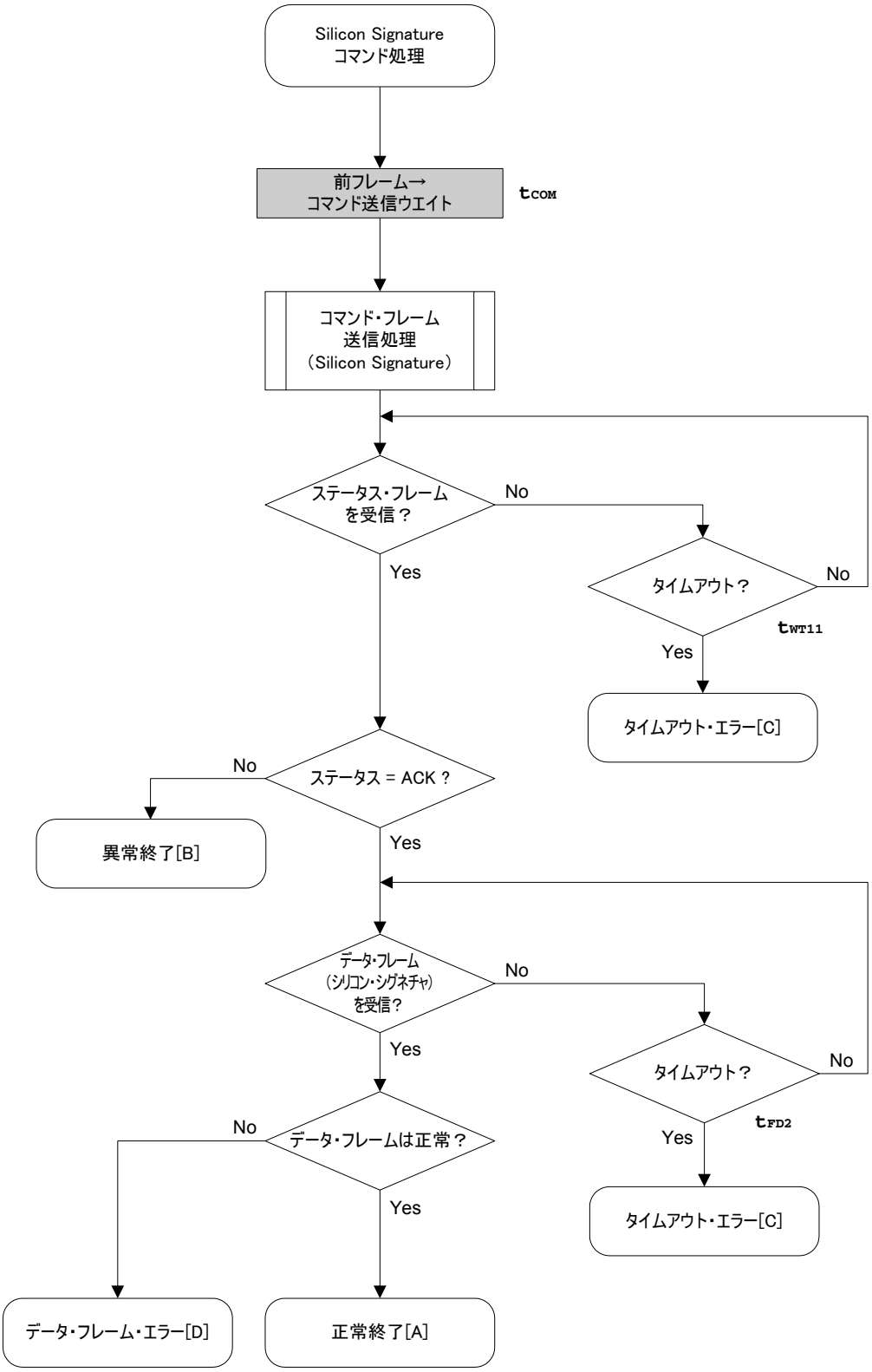
⑤ データ・フレーム（シリコン・シグネチャ・データ）受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}）。
⑥ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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

4. 11. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，シリコン・シグネチャ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）。
	リード・エラー	20H	セキュリティ情報の読み出しに失敗しました。
タイムアウト・エラー [C]		－	ステータス・フレーム，またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー[D]		－	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

4. 11. 4　フロー・チャート



4. 11. 5　サンプル・プログラム

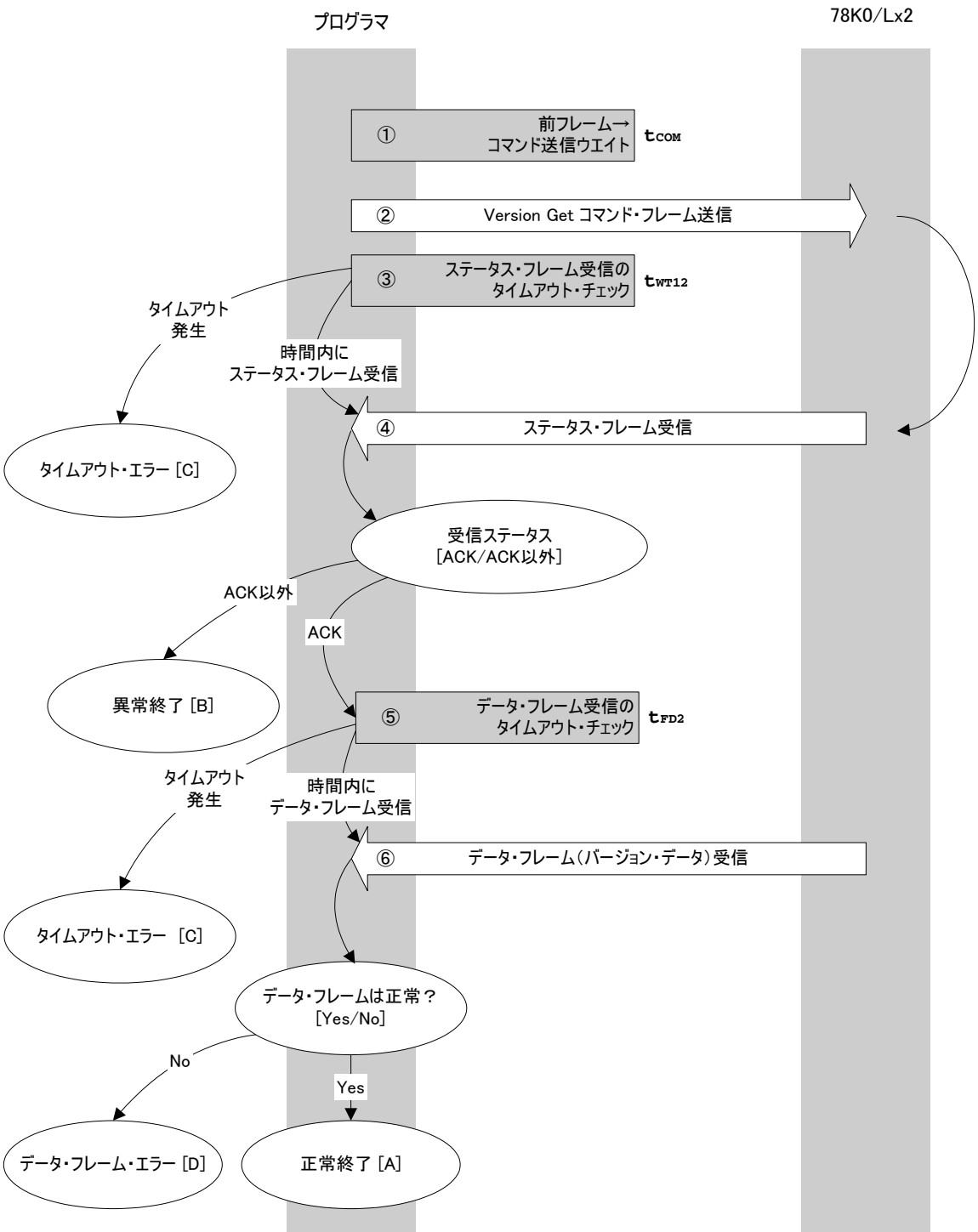
Silicon Signatureコマンド処理のサンプル・プログラムです。

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


   fl_wait(tCOM);          // wait before sending command

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

   rc = get_sfrm_ua(fl_ua_sfrm, tWT11_TO); // 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_TO);       // 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]
}
```

4. 12　Version Getコマンド

4. 12. 1　処理手順チャート

Version Getコマンド処理手順



4. 12. 2　処理手順説明

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

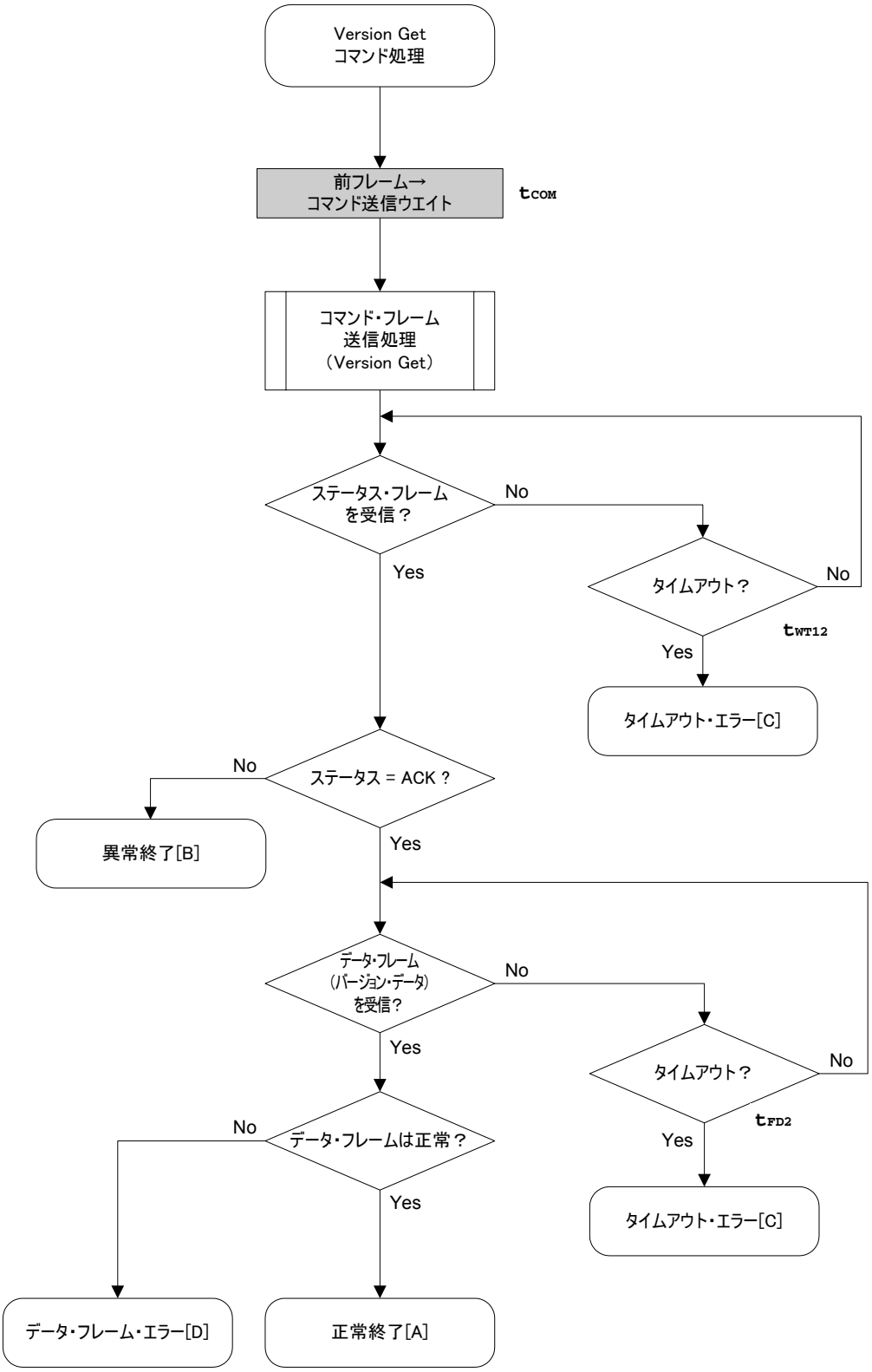
⑤ データ・フレーム（バージョン・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD2}）。
⑥ 受信したデータ・フレーム（バージョン・データ）をチェックします。

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

4. 12. 3　終了時の内容

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

4. 12. 4　フロー・チャート



4. 12. 5　サンプル・プログラム

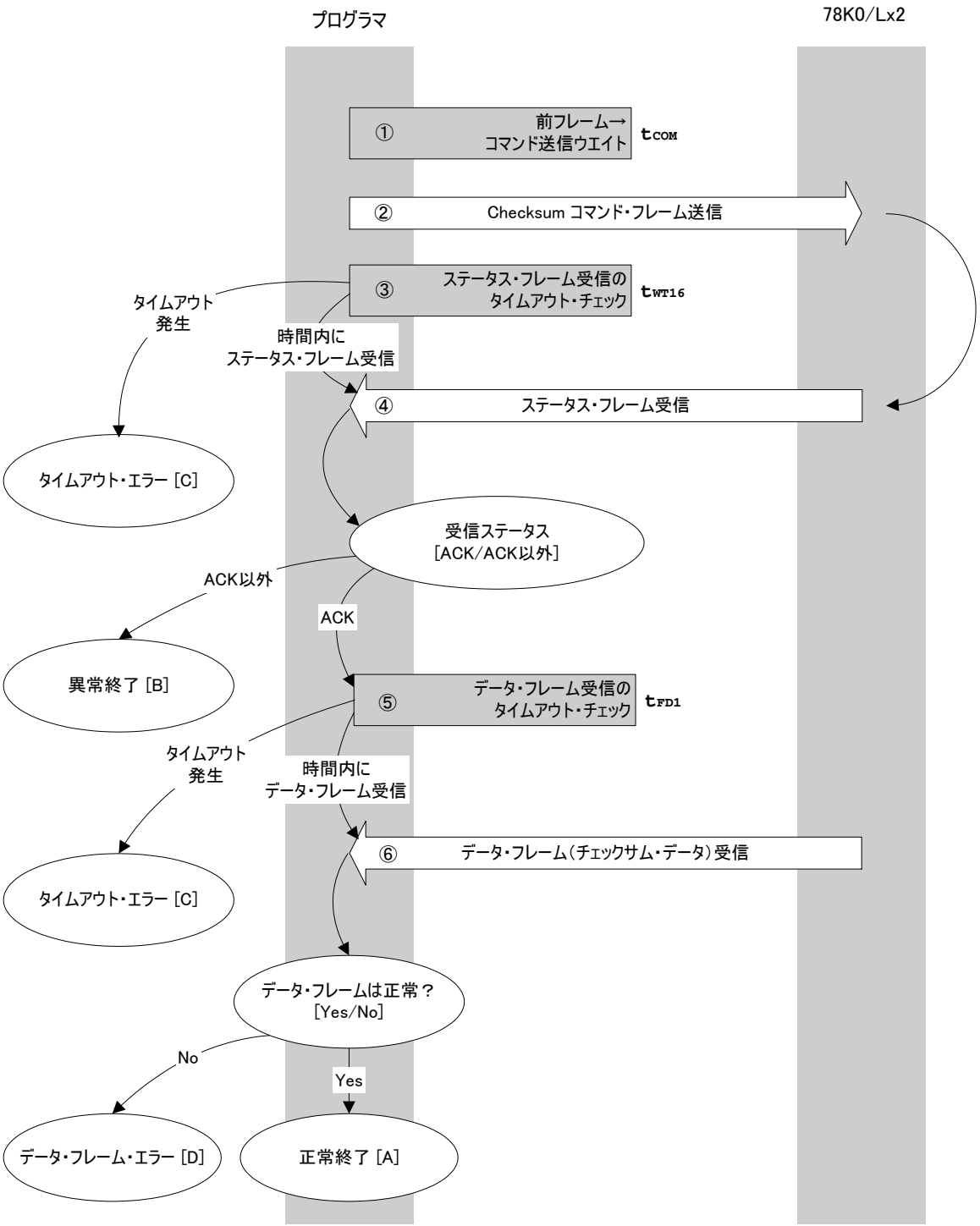
Version Getコマンド処理のサンプル・プログラムです。

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


   fl_wait(tCOM);          // wait before sending command


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

   rc = get_sfrm_ua(fl_ua_sfrm, tWT12_TO); // 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_TO);       // 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]
}
```

4. 13　Checksumコマンド

4. 13. 1　処理手順チャート

Checksumコマンド処理手順



4. 13. 2　処理手順説明

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

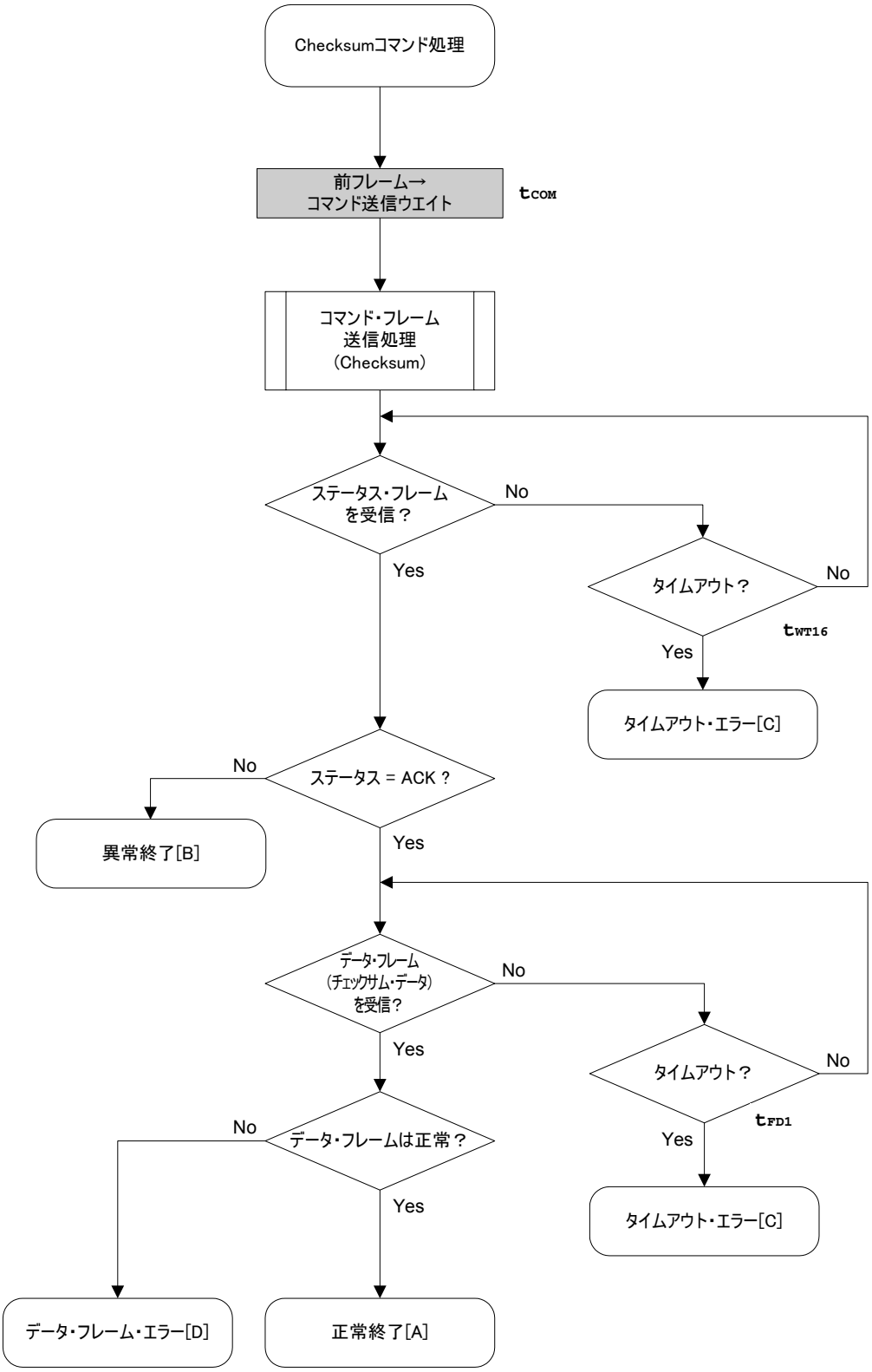
⑤ データ・フレーム（チェックサム・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
（タイムアウト時間t_{FD1}）。
⑥ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

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

4. 13. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，チェックサム・データを取得できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始／終了アドレスがフラッシュ・メモリの範囲外です。または，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。
データ・フレーム・エラー [D]		−	チェックサム・データとして受信したデータ・フレームのチェックサムが異常です。

4. 13. 4　フロー・チャート



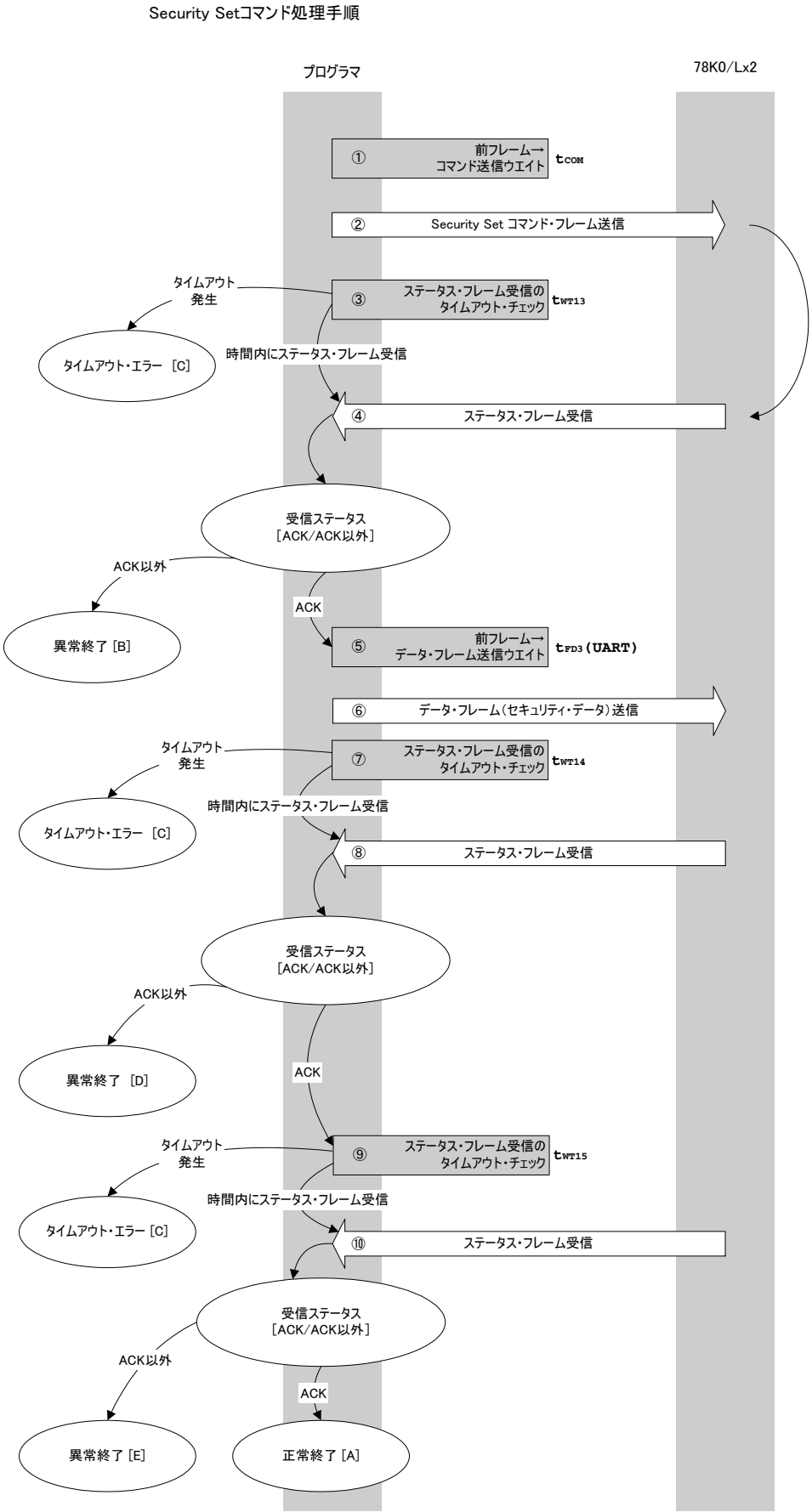
4. 13. 5　サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

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

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


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

   fl_wait(tCOM);          // wait before sending command

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

   rc = get_sfrm_ua(fl_ua_sfrm, tWT16_TO); // 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_TO);      // 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]

}
```

4. 14　Security Setコマンド

4. 14. 1　処理手順チャート



4. 14. 2　処理手順説明

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

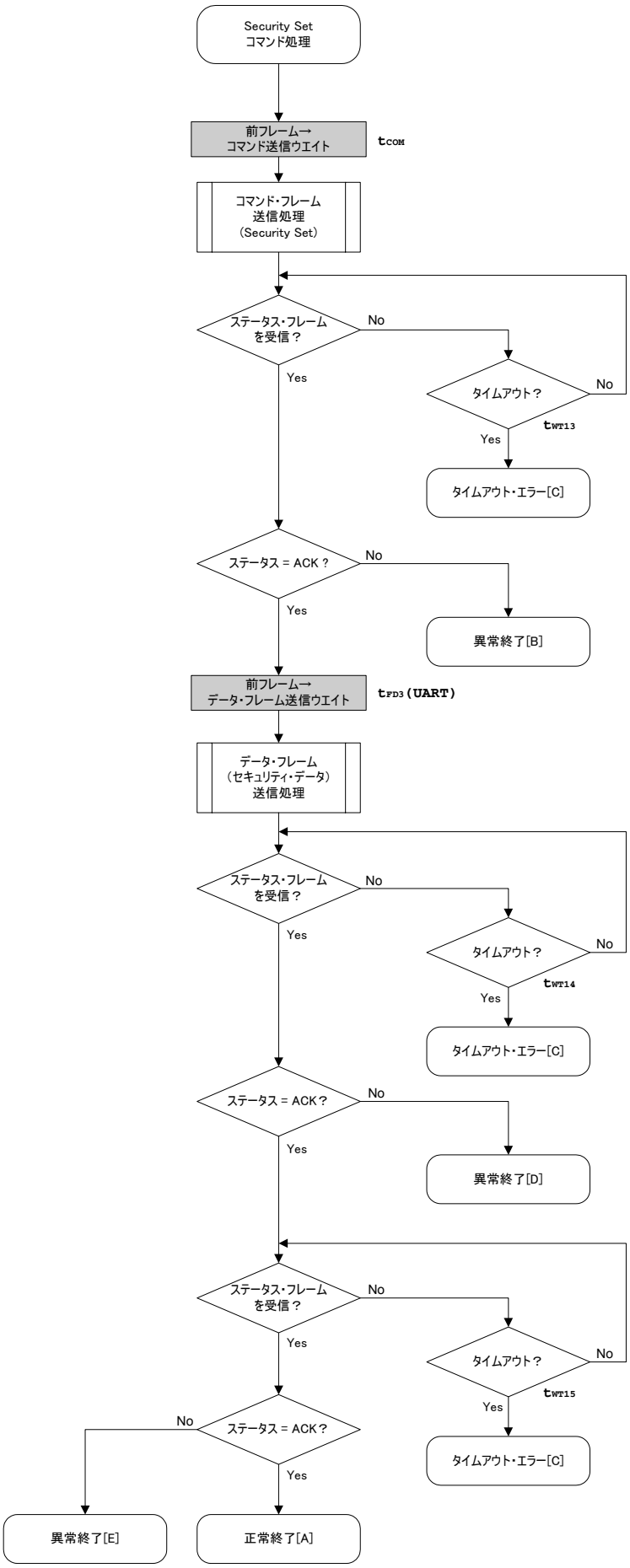
⑨ ステータス・フレーム受信までのタイムアウト・チェックを行います。
タイムアウトが発生した場合はタイムアウト・エラー[C]となります
（タイムアウト時間t_{WT15}）。
⑩ ステータス・コードをチェックします。

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。

4. 14. 3　終了時の内容

終了内容		ステータス・コード	内　容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され，セキュリティ設定データが正しく設定されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	コマンド情報（パラメータ）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。

4. 14. 4　フロー・チャート



4. 14. 5　サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

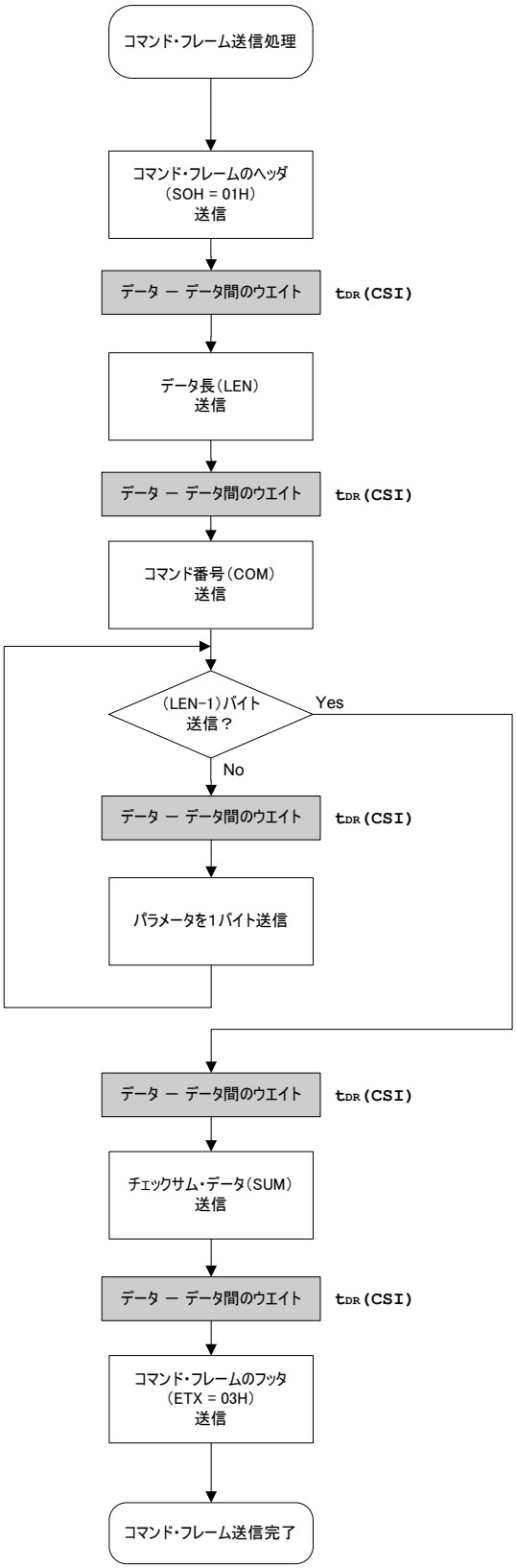
```
/*************************************************************/
/*                                                           */
/* Set security flag command                                 */
/*                                                           */
/*************************************************************/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

   /*********************************************/
   /*      set params                           */
   /*********************************************/
   fl_cmd_prm[0] = 0x00;                  // "BLK" (must be 0x00)
   fl_cmd_prm[1] = 0x00;                  // "PAG" (must be 0x00)
   fl_txdata_frm[0] = (scf |= 0b11101000);
                                    // "FLG" (bit7, 6, 5, 3 must be '1' (to make sure))

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


   /*********************************************/
   /*      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_TO); // 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_UA);


   put_dfrm_ua(2, fl_txdata_frm, true);    // send security setting(FLAG) & BOT 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;
}
```

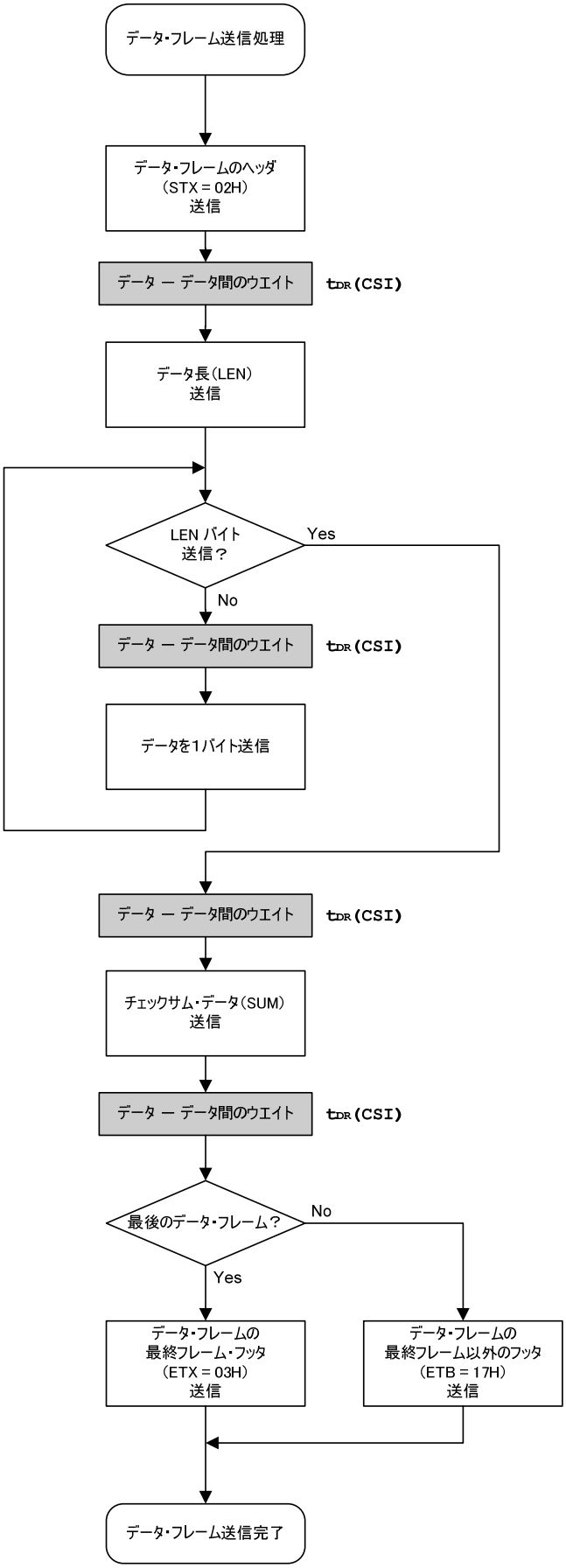
第5章　3線式シリアルI/O（CSI）通信方式

この章のフロー・チャートで示した略号（t_{XX}およびt_{WTXX}）は，第6章　フラッシュ・メモリ・プログラミング・パラメータ特性における規格項目の略号です。各規格値については第6章　フラッシュ・メモリ・プログラミング・パラメータ特性を参照してください。

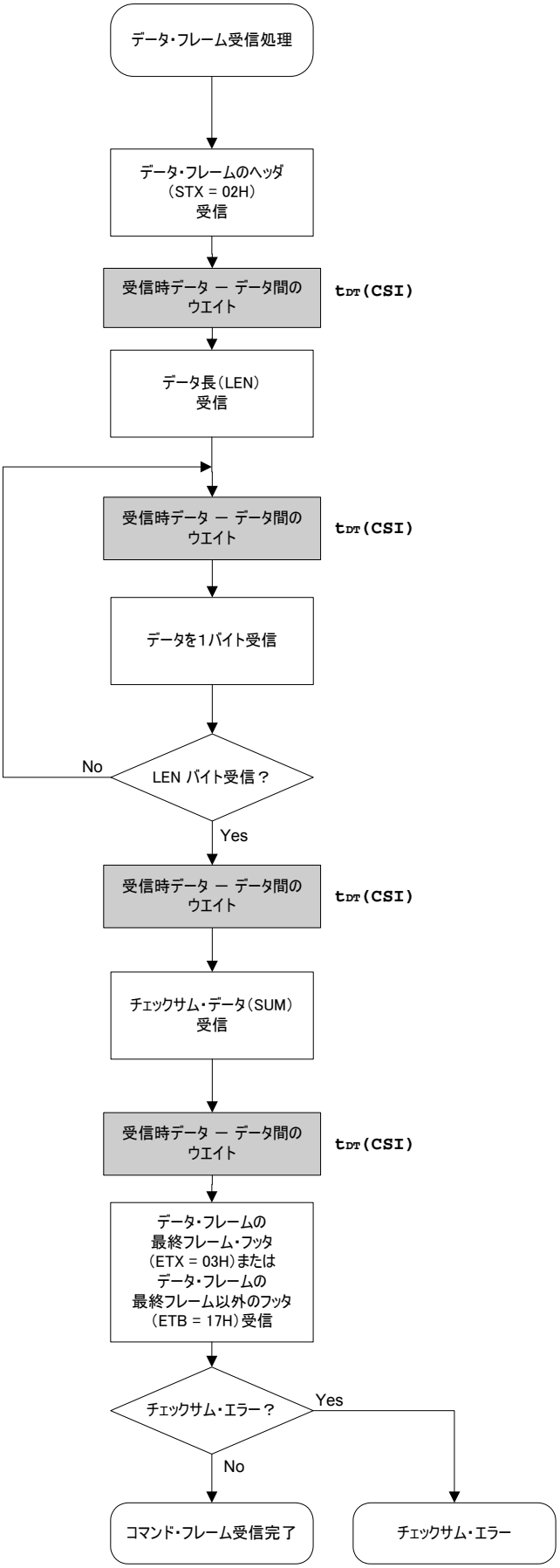
★ 5. 1　コマンド・フレーム送信処理のフロー・チャート



★　5. 2　データ・フレーム送信処理のフロー・チャート



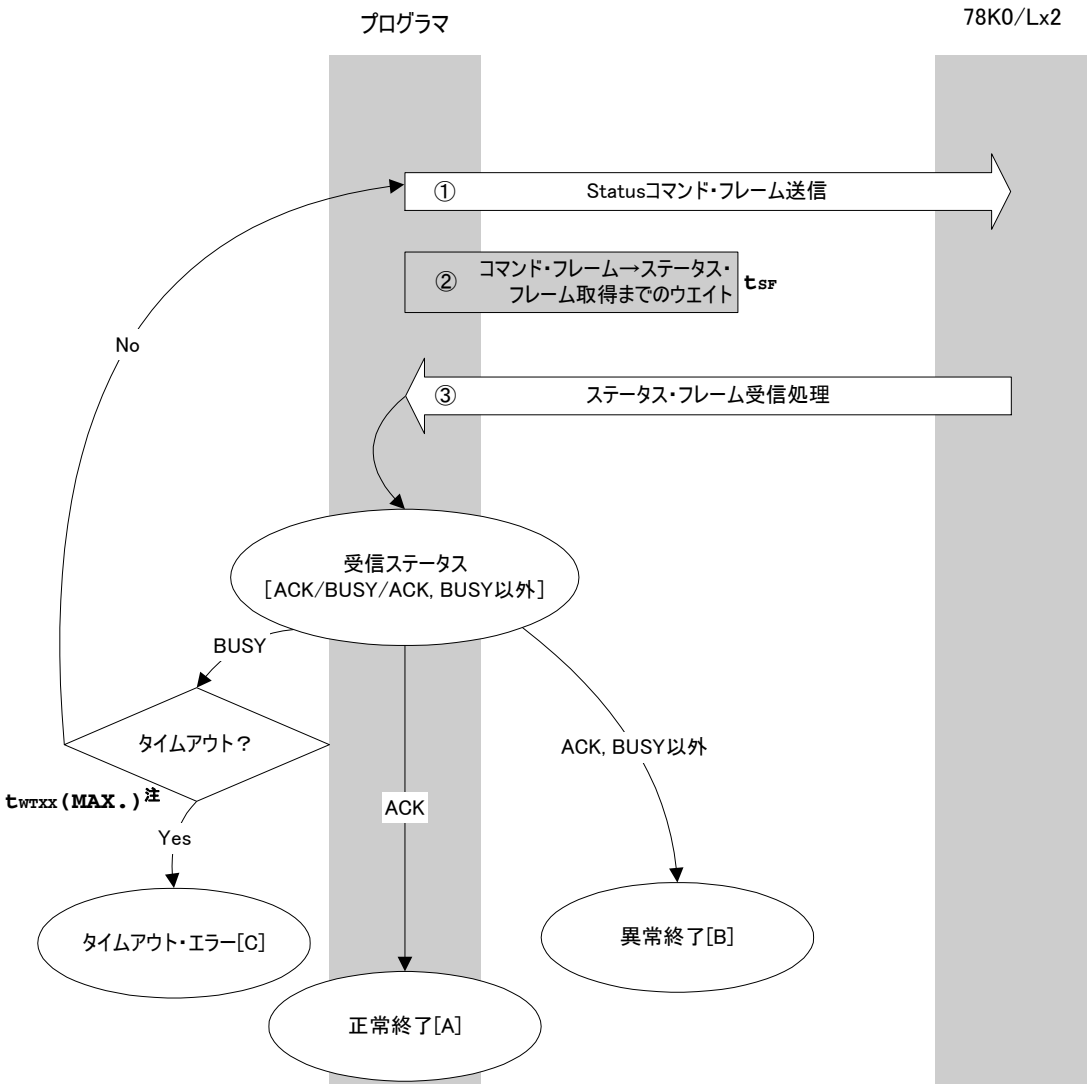
★ 5. 3　データ・フレーム受信処理のフロー・チャート



5. 4　Statusコマンド

5. 4. 1　処理手順チャート

Statusコマンド処理手順



注　実行コマンドにより，適用スペックが異なります。

5.4.2　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。

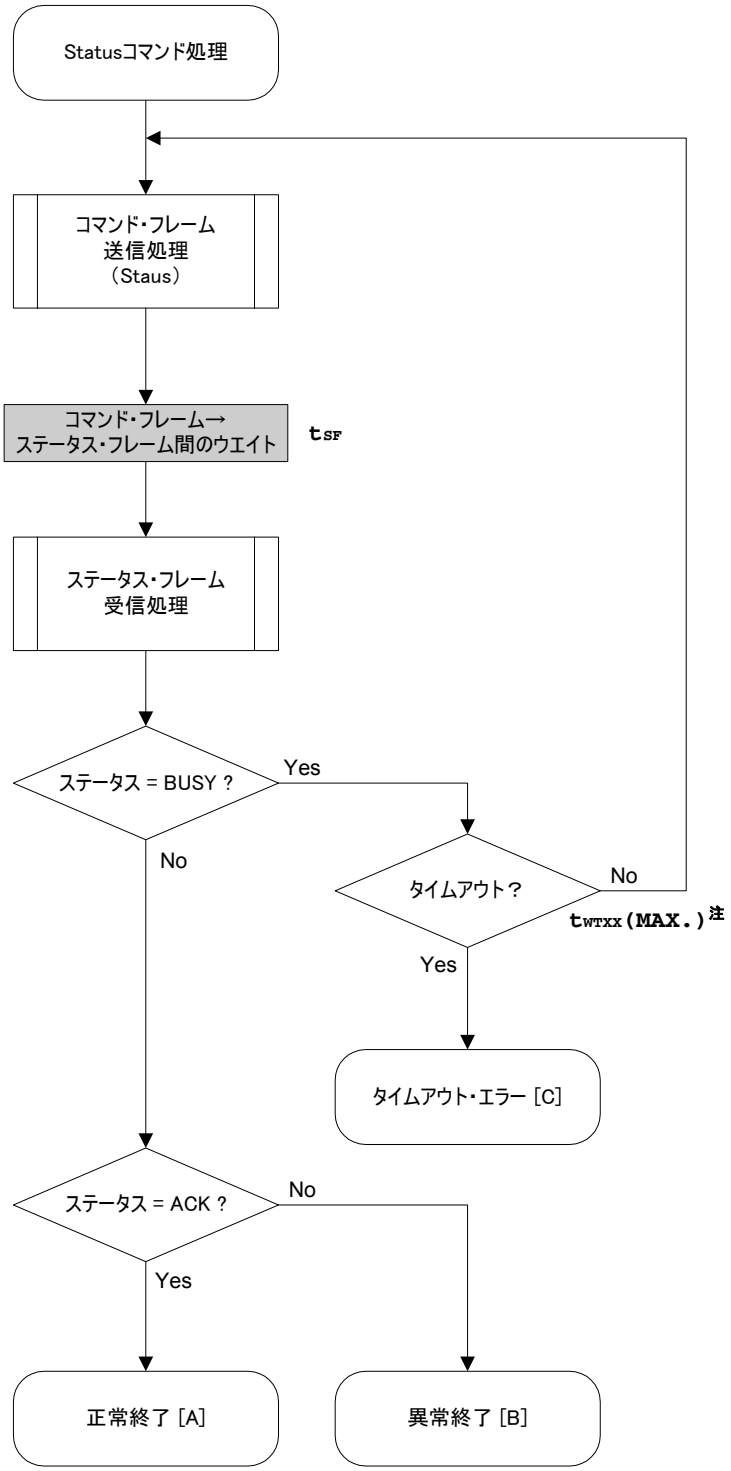
ST1 = ACKの場合　：正常終了[A]です。
ST1 = BUSYの場合　：タイムアウト（$t_{WTXX(MAX.)}$）[注]をチェックします。
→　タイムアウトでなければ①からやり直します。
→　タイムアウトであれば タイムアウト・エラー[C] です。
ST1 = ACK, BUSY以外の場合　：異常終了[B]です。

注　実行コマンドにより，適用スペックが異なります。

5.4.3　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/Lx2 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
★	Read エラー	20H	セキュリティ情報の読み出しに失敗しました。
	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。

5.4.4　フロー・チャート



注　実行コマンドにより，適用スペックが異なります。

5.4.5　サンプル・プログラム

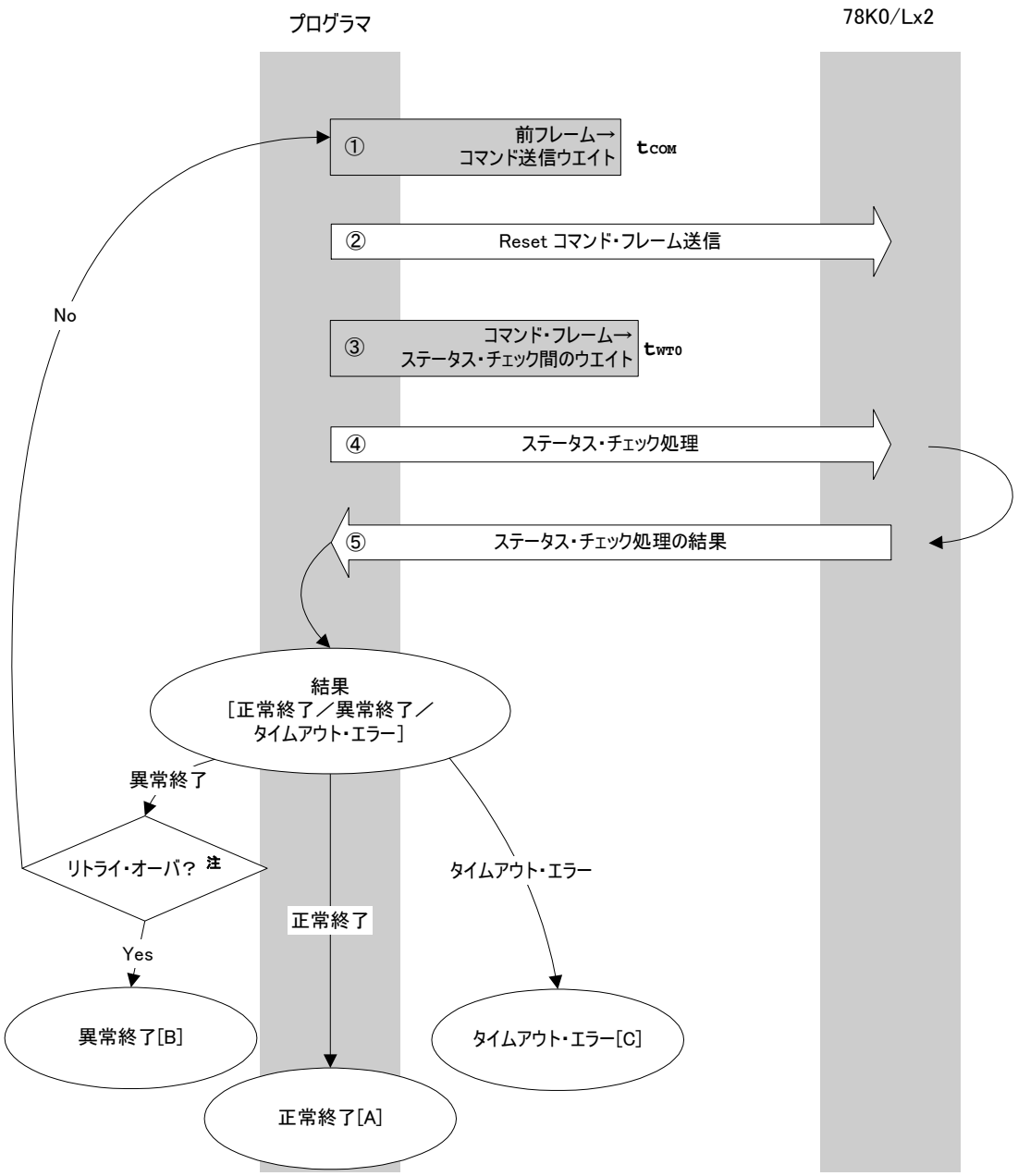
Statusコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* 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(u32 limit)
{
   u16     rc;

   start_flto(limit);

   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;

}
```

5. 5　Resetコマンド

5. 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。

5. 5. 2　処理手順説明

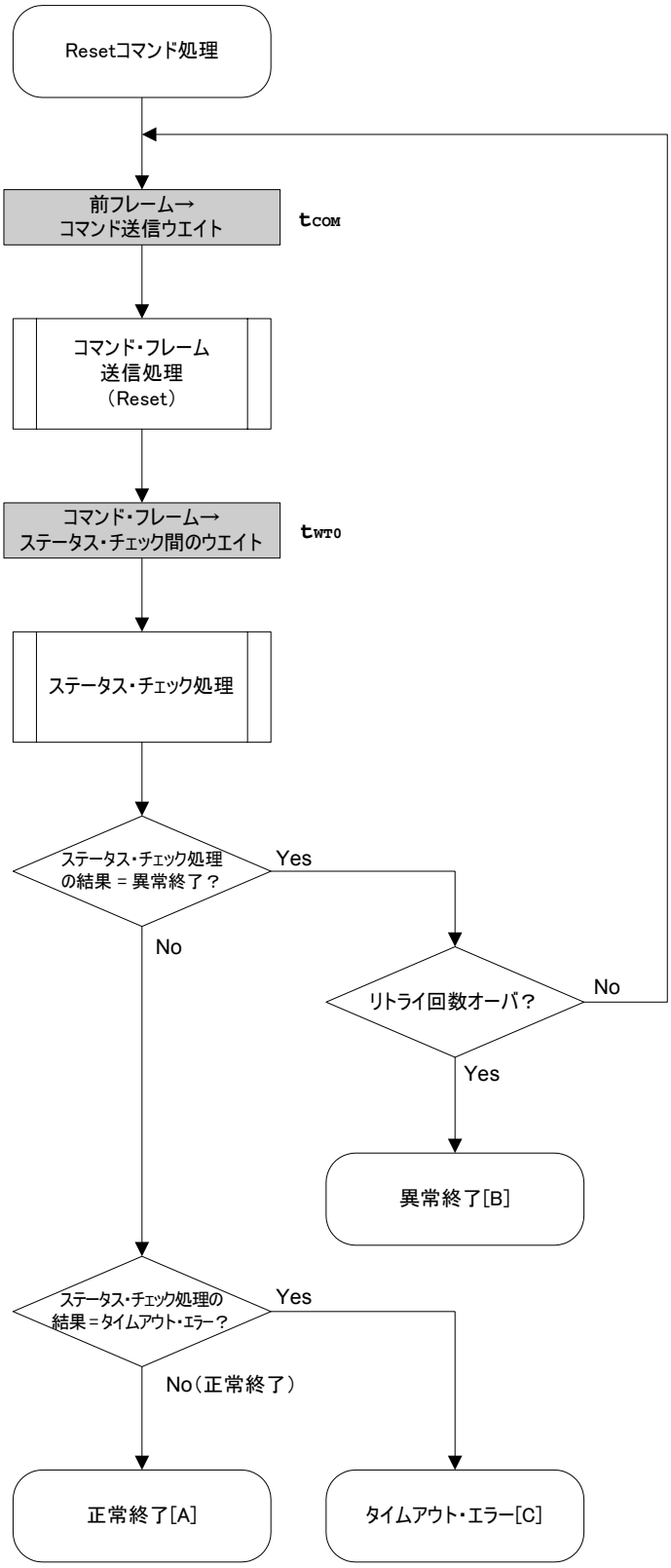
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Resetコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT0}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

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。

5. 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/Lx2 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。

5. 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]

5. 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。

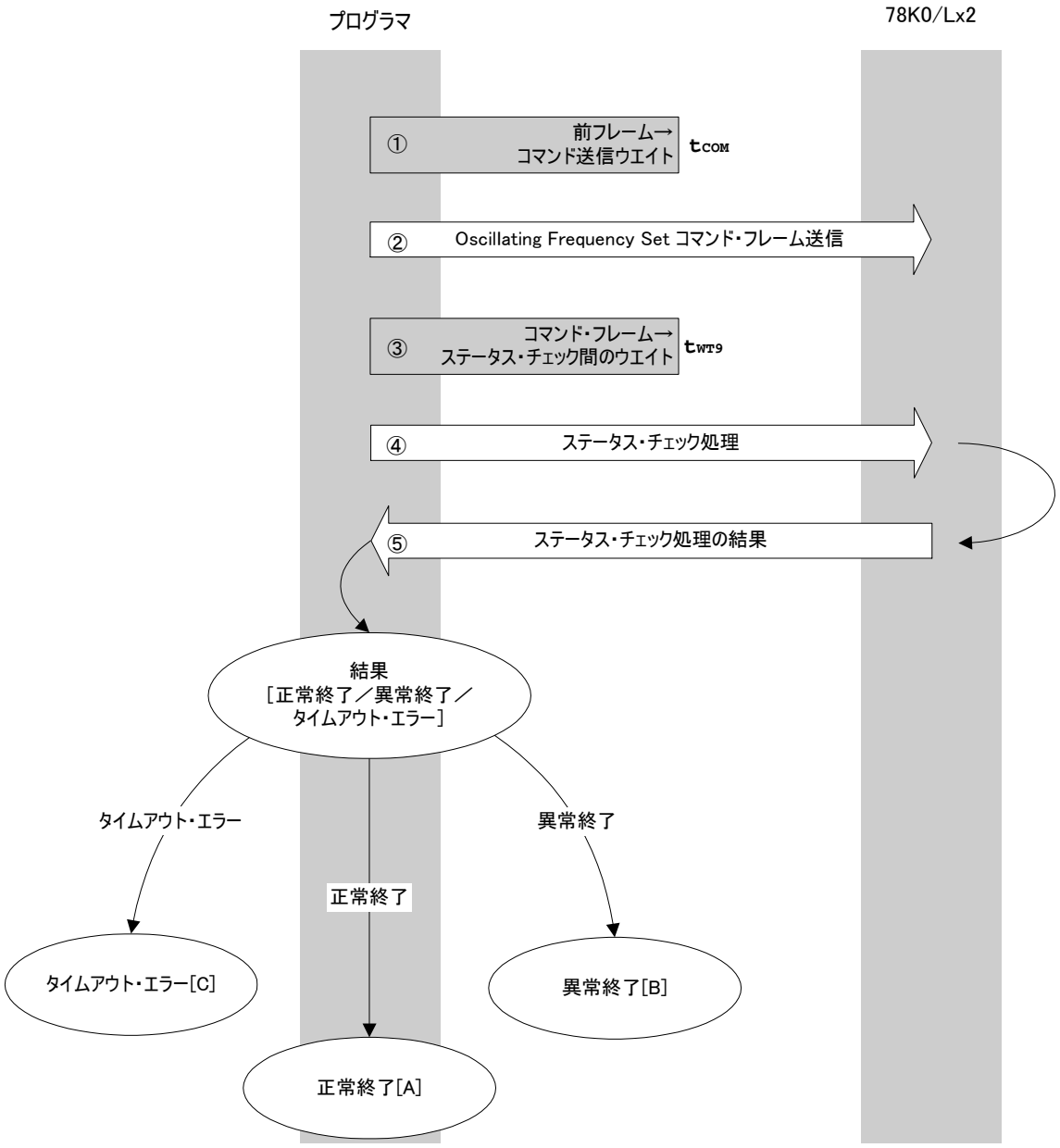
```
/*****************************************************************/
/*                                                               */
/* Reset command (CSI)                                           */
/*                                                               */
/*****************************************************************/
/* [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(tWT0_TO); // 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;
}
```

5. 6　Oscillating Frequency Setコマンド

CSI通信を使用する場合は，このコマンドを実行する必要はありません（プログラマの仕様上，CSI通信時もこのコマンドの実行が必要な場合は，8 MHzを設定してください）。

5. 6. 1　処理手順チャート

Oscillating Frequency Setコマンド処理手順



5.6.2　処理手順説明

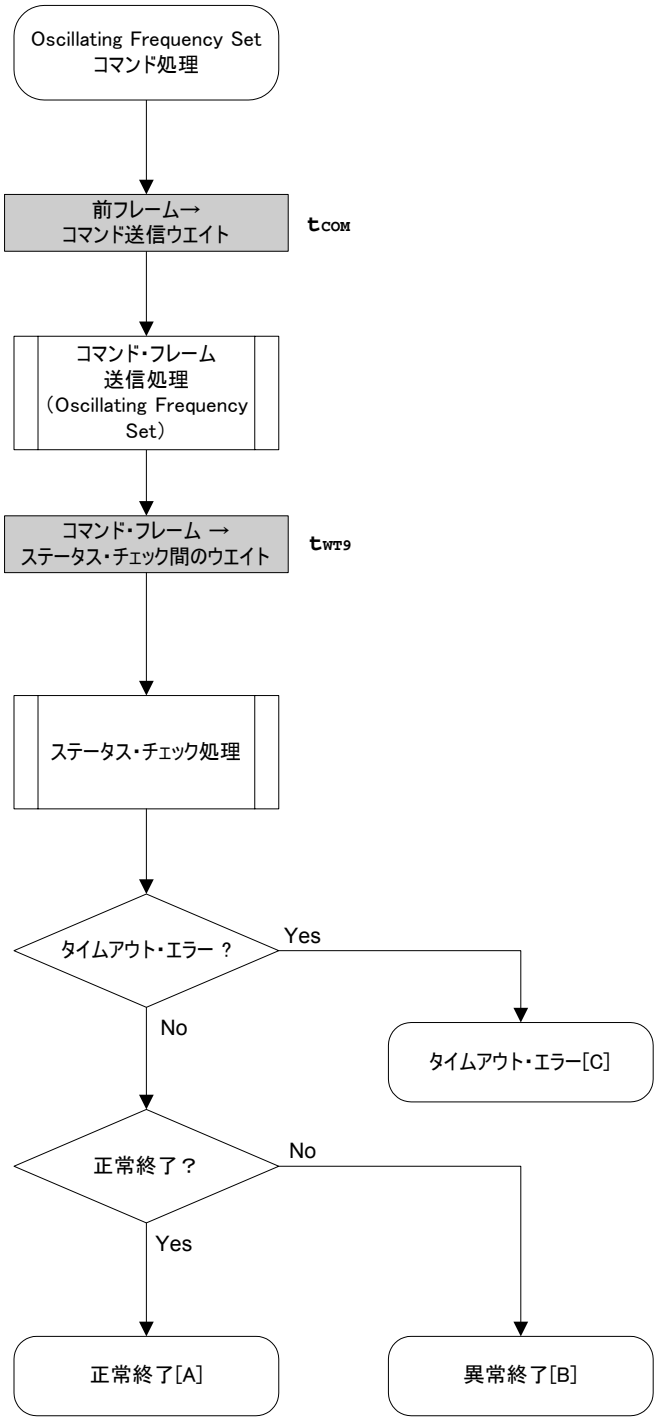
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Oscillating Frequency Setコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT9}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

5.6.3　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/Lx2 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。

5. 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]

5. 6. 5　サンプル・プログラム

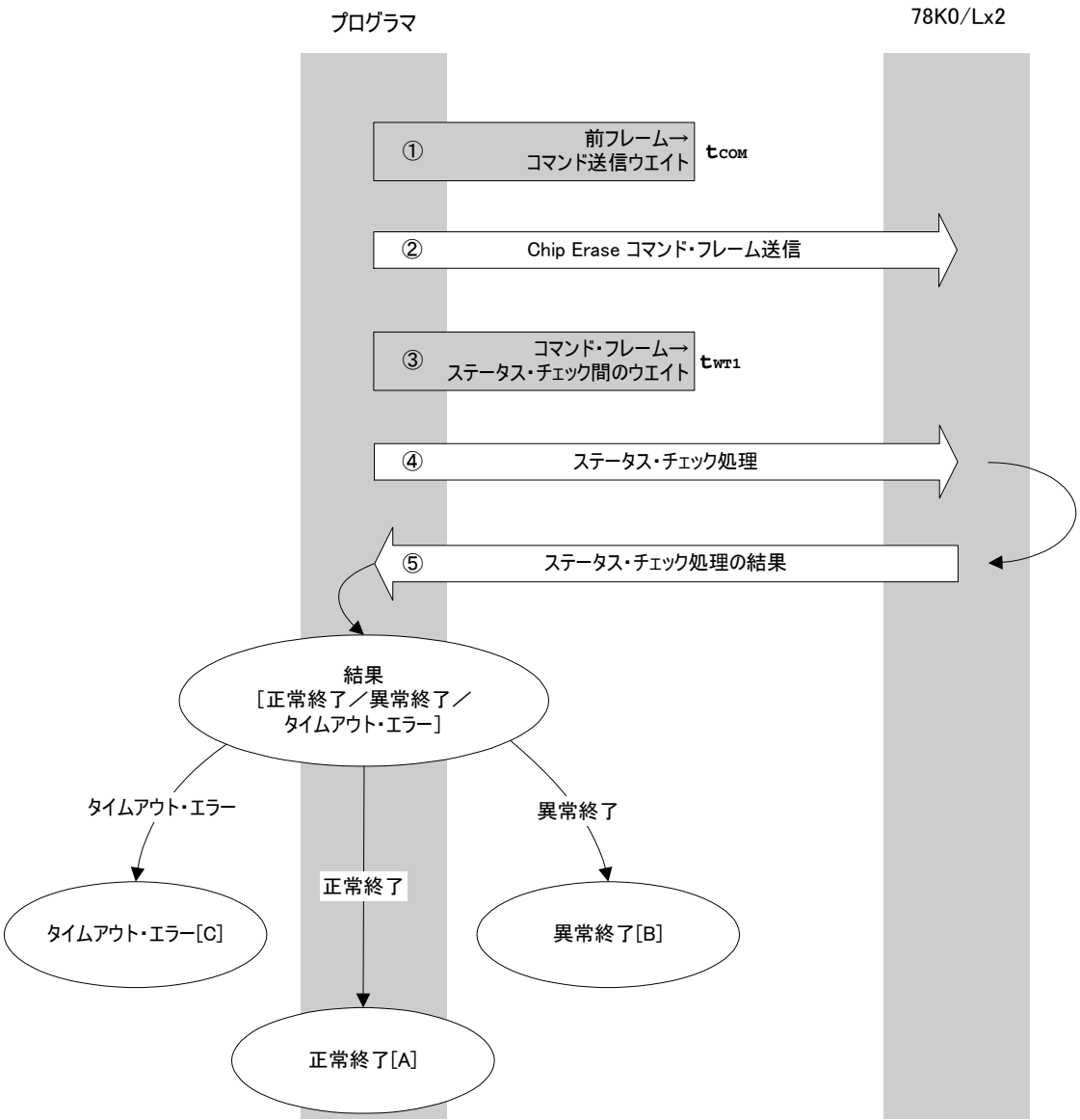
Oscillating Frequencyコマンド処理のサンプル・プログラムです。

```
/*************************************************************/
/*                                                           */
/* 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(tWT9_TO); // 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;
}
```

5. 7　Chip Eraseコマンド

5. 7. 1　処理手順チャート

Chip Eraseコマンド処理手順



5.7.2　処理手順説明

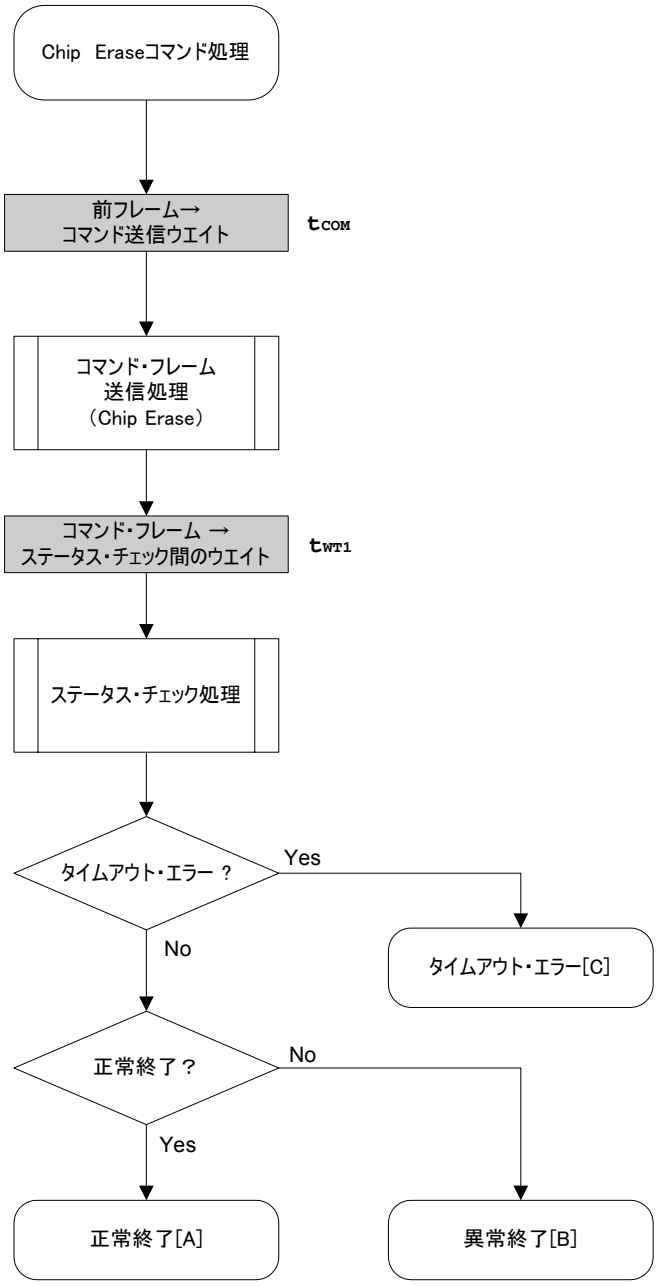
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Chip Eraseコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします（ウエイト時間t_{WT1}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

5.7.3　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，Chip Erase コマンドが禁止となる設定になっています。
	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。

5.7.4　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]

5. 7. 5　サンプル・プログラム

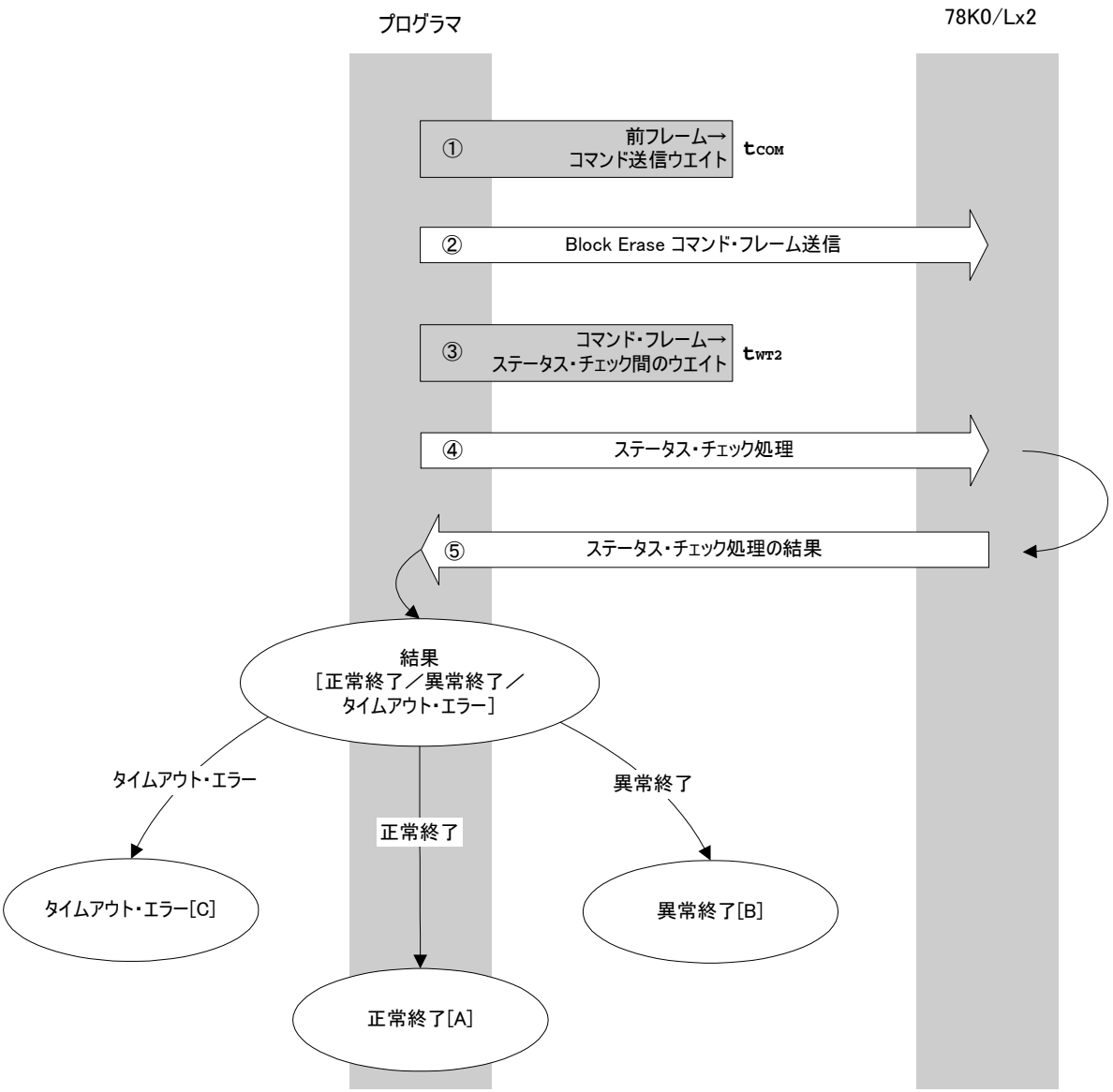
Chip Eraseコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* 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(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;

}
```

5. 8　Block Eraseコマンド

5. 8. 1　処理手順チャート

Block Eraseコマンド処理手順



5. 8. 2　処理手順説明

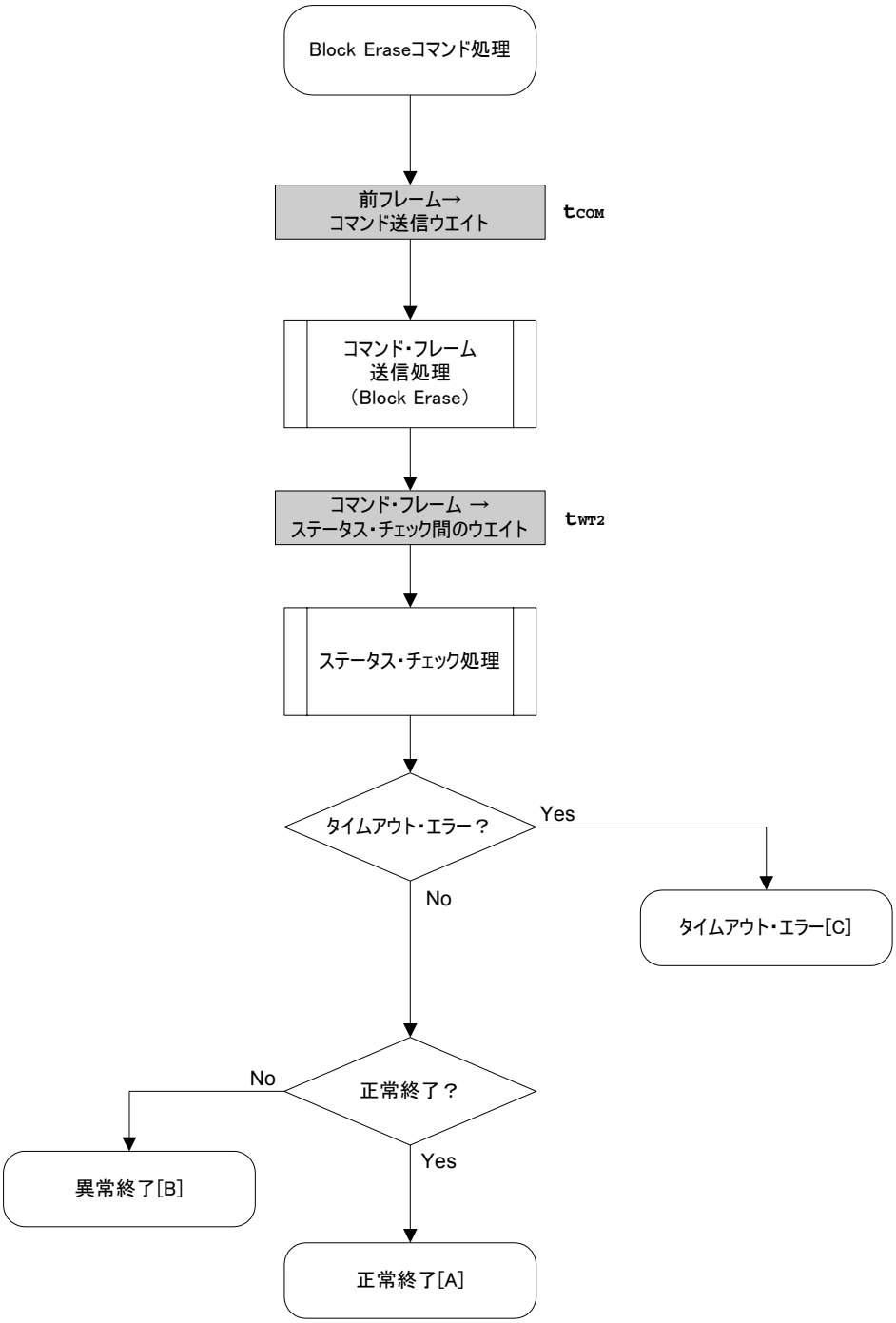
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Block Eraseコマンドを送信します。
③ ステータス・フレーム取得までのウエイトをします（ウエイト時間t_{WT2}）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

正常終了の場合　：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。

5. 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，Block Erase コマンドが禁止となる設定になっています。
	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。

5. 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]
No
正常終了？
Yes
異常終了[B]
正常終了[A]

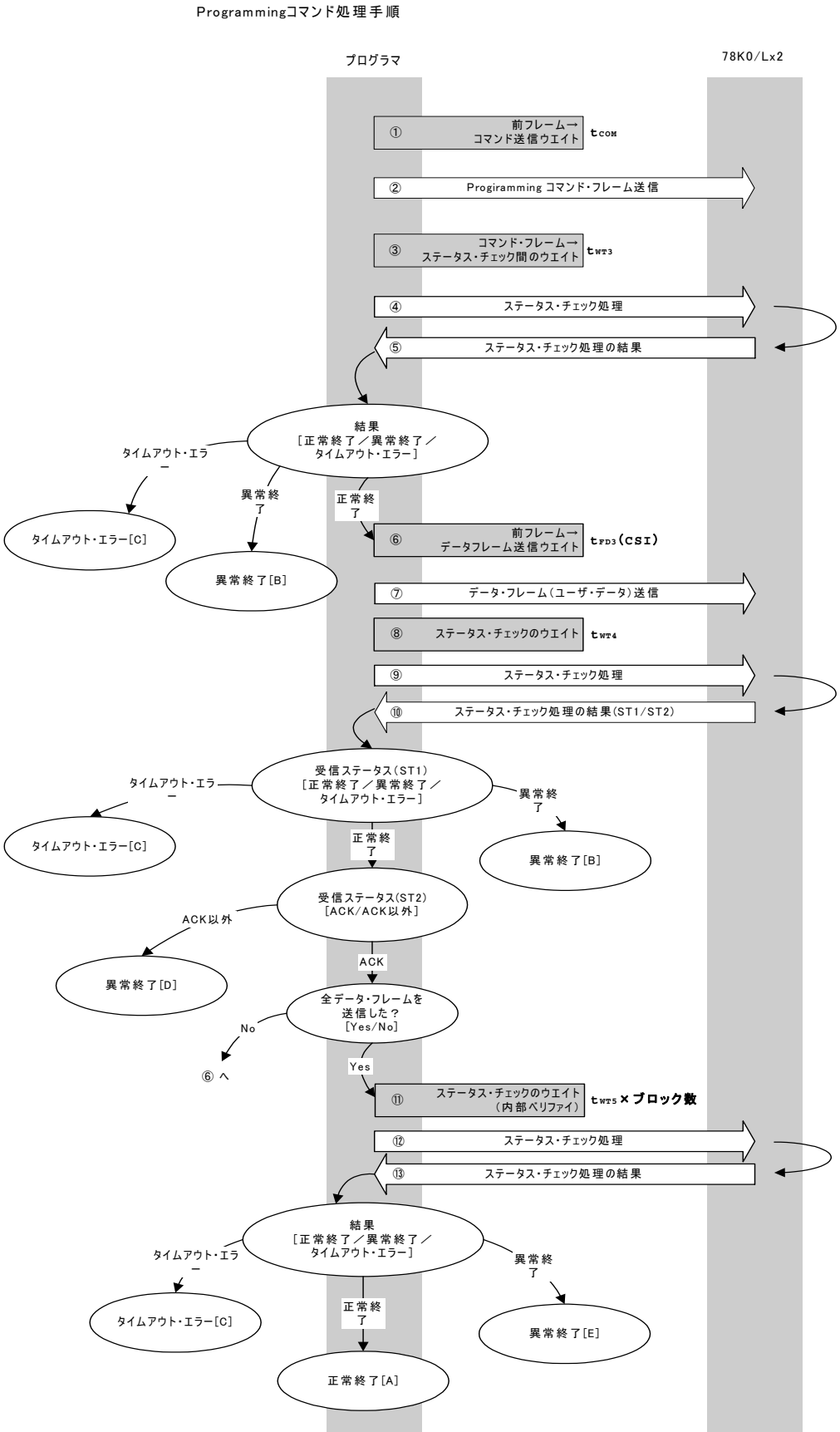
5. 8. 5　サンプル・プログラム

Block Eraseコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Erase block command (CSI)                                     */
/*                                                               */
/*****************************************************************/
/* [i] u16 sblk    ... start block to erase (0...255)            */
/* [i] u16 eblk    ... end block to erase   (0...255)            */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_csi_erase_blk(u16 sblk, u16 eblk)
{

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

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

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

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

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

   put_cmd_csi(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm);// send "Block Erase" command

   fl_wait(wt2);


   rc = fl_csi_getstatus(wt2_max); // get status frame
// switch(rc) {
//
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

5. 9　Programmingコマンド

5. 9. 1　処理手順チャート



5. 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}（CSI））。
⑦ データ・フレーム送信処理により，78K0/Lx2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。

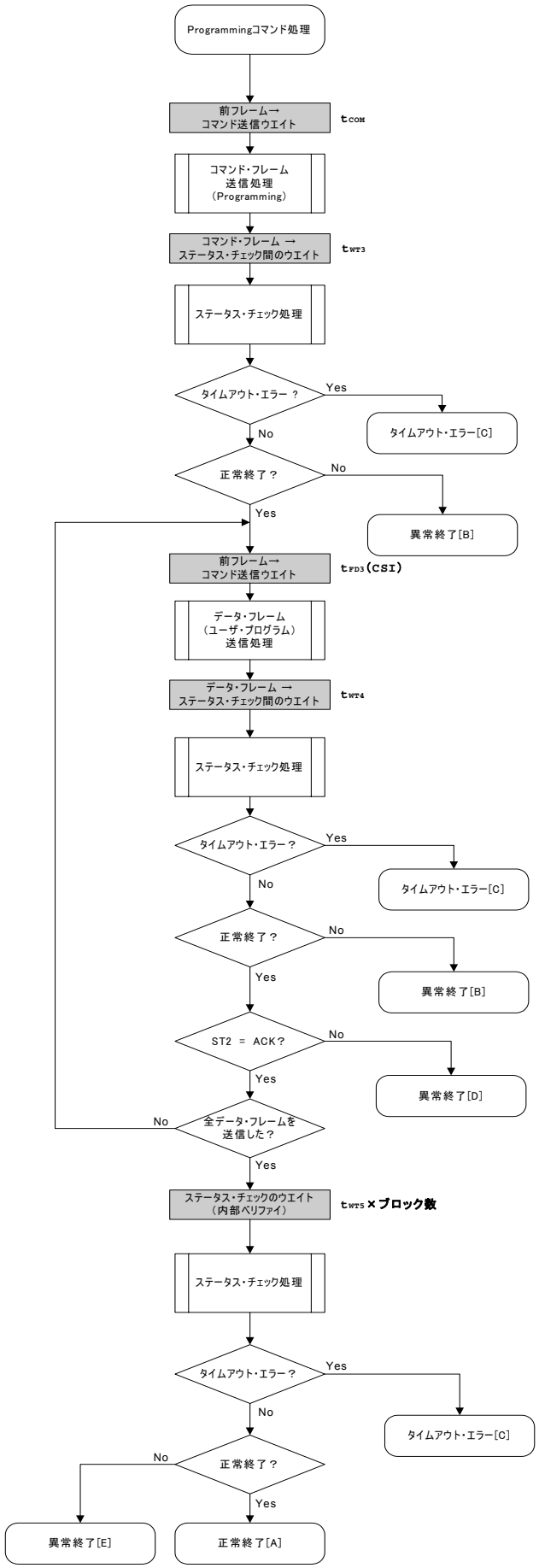
⑪ ステータス・チェック処理までのウエイトをします（ウエイト時間 t_{WT5}×ブロック数）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

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。

5. 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，Programming コマンドが禁止となる設定になっています。
	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]	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。

5. 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
tCOM
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
tWT3
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
tFD3（CSI）
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
tWT4
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
No
全データ・フレームを
送信した？
Yes
ステータス・チェックのウエイト
（内部ベリファイ）
tWT5 × ブロック数
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]

5. 9. 5　サンプル・プログラム

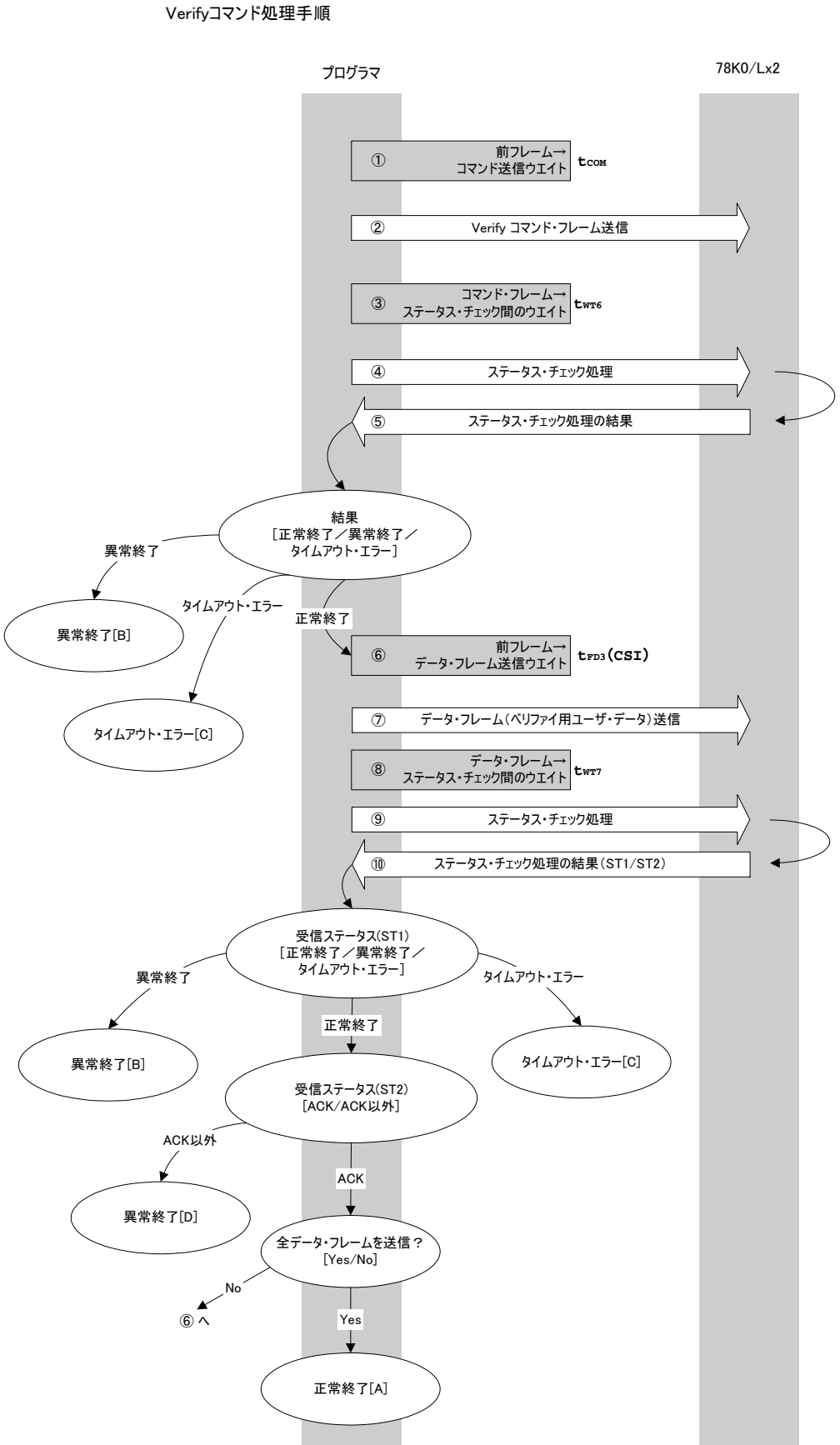
Programmingコマンド処理のサンプル・プログラムです。

```
/************************************************************/
/*                                                          */
/* 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(tWT3_TO);         // 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_CSI);                          //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(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 != 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(tWT5_MAX * block_num);    // get status frame
// switch(rc) {
//          case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//          case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
//          default:                return  rc;     break;  // case [E]
// }
    return  rc;

}
```

5. 10　Verifyコマンド

5. 10. 1　処理手順チャート



5. 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}（CSI））。
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）。

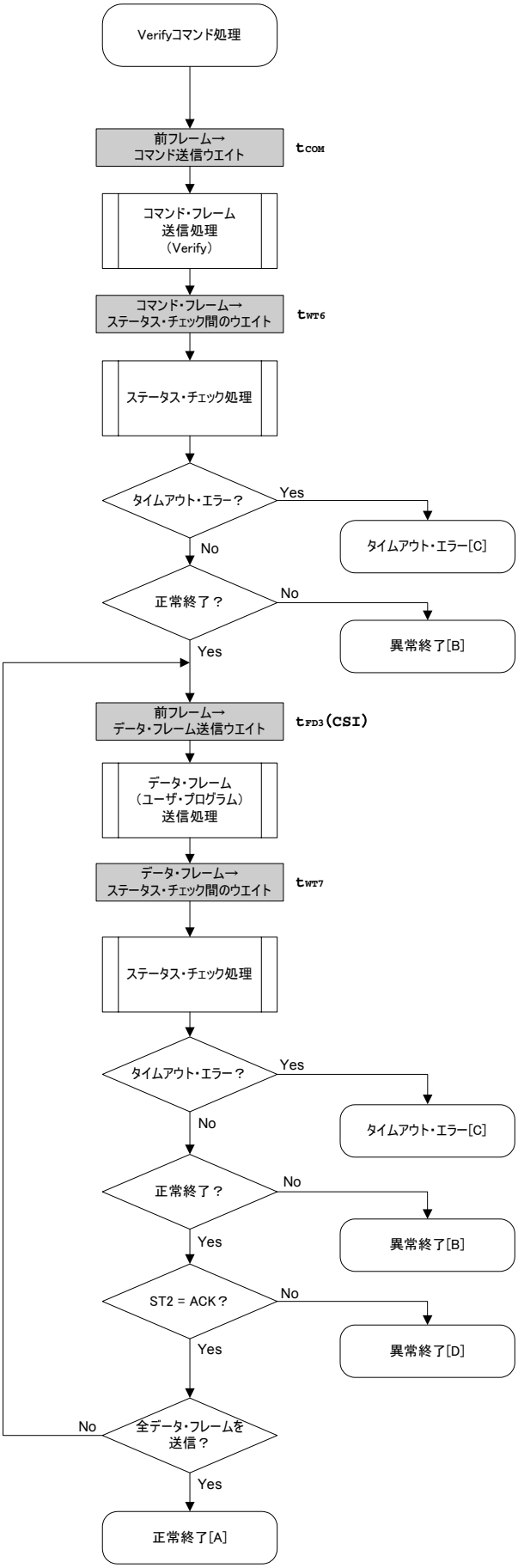
ST1 = 異常終了の場合　：異常終了[B]です。
ST1 = タイムアウト・エラーの場合　：タイムアウト・エラー[C]です。
ST1 = 正常終了の場合　：受信ステータス（ST2）の値に応じて次の処理を行います。

- 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。

5. 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。

5. 10. 4　フロー・チャート



5. 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。

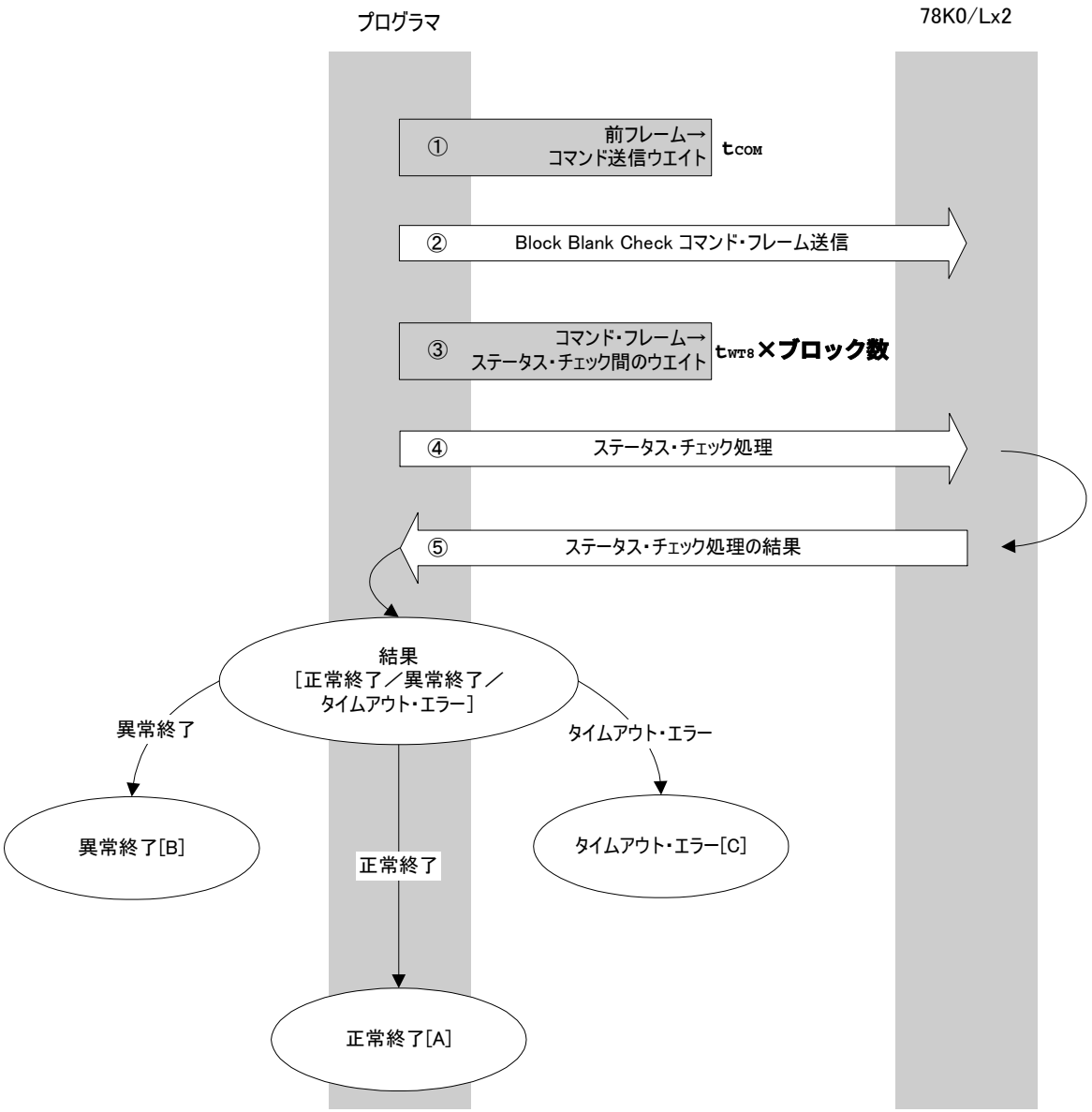
```
/*************************************************************/
/*                                                           */
/* Verify command (CSI)                                      */
/*                                                           */
/*************************************************************/
/* [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(tWT6_TO);         // 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_CSI);                          // 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(tWT7_TO); // 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]

}
```

5. 11　Block Blank Checkコマンド

5. 11. 1　処理手順チャート

Block Blank Checkコマンド処理手順



5. 11. 2　処理手順説明

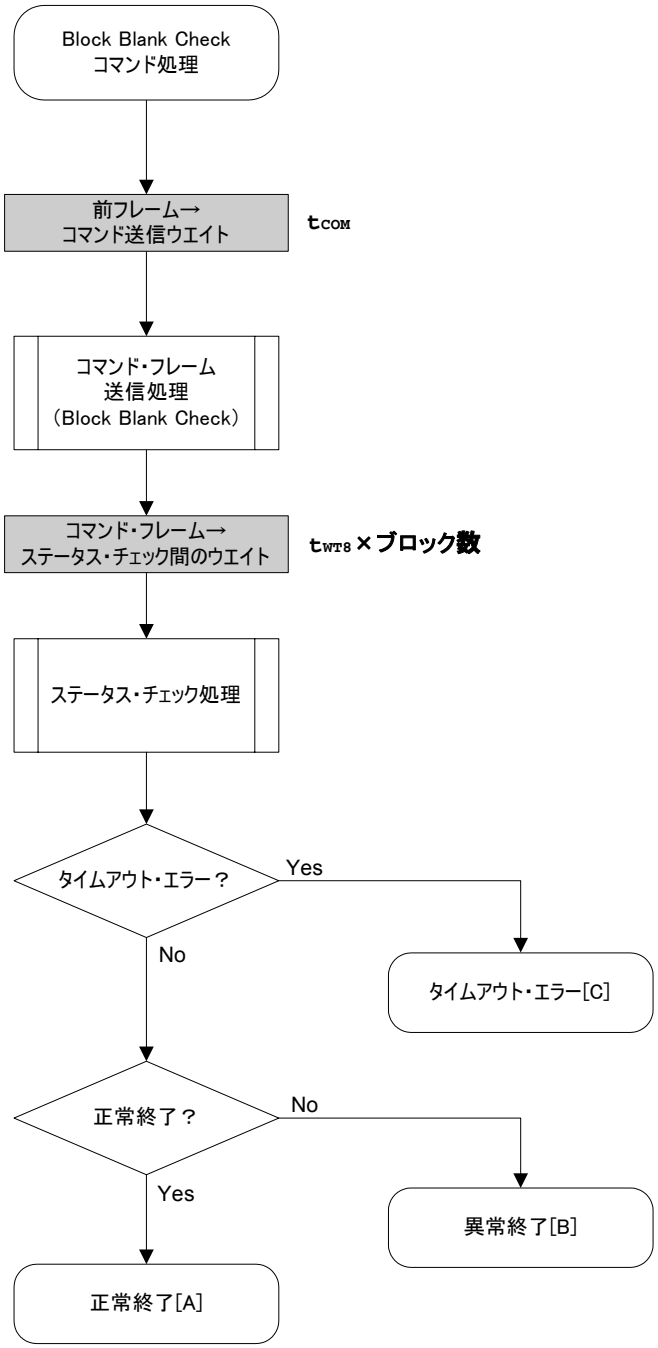
① 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{COM}）。
② コマンド・フレーム送信処理により，Block Blank Checkコマンドを送信します。
③ コマンド送信からステータス・チェック処理までのウエイトをします
（ウエイト時間t_{WT8}×ブロック数）。
④ ステータス・チェック処理により，ステータス・フレームを取得します。
⑤ ステータス・チェック処理の結果に応じて次の処理を行います。

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　：正常終了[A]です。

5. 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。

5. 11. 4　フロー・チャート



5. 11. 5　サンプル・プログラム

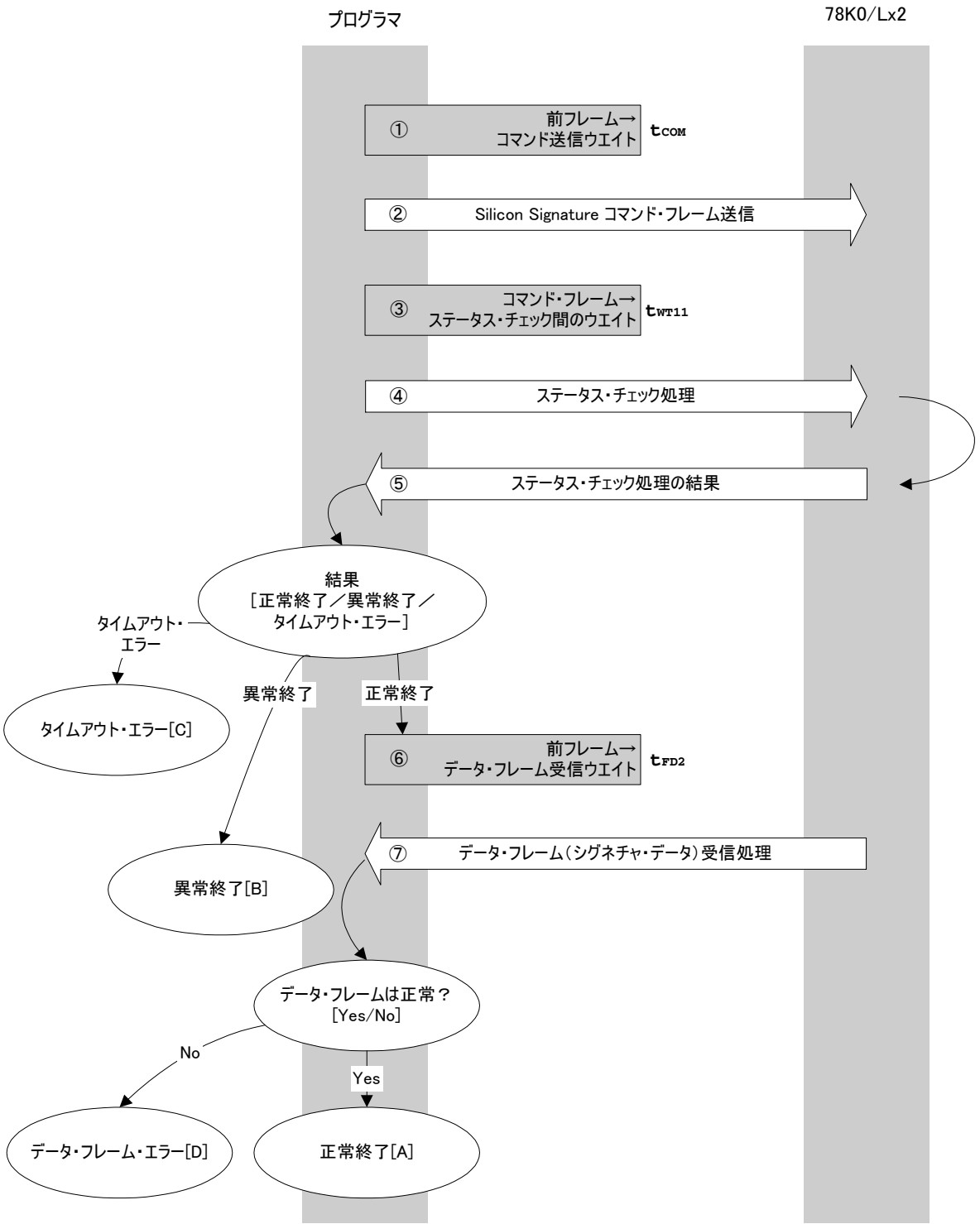
Block Blank Checkコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* Block blank check command (CSI)                               */
/*                                                               */
/*****************************************************************/
/* [i] u32 top     ... start address                             */
/* [i] u32 bottom  ... end address                               */
/* [r] u16         ... error code                                */
/*****************************************************************/
u16        fl_csi_blk_blank_chk(u32 top, u32 bottom)
{
   u16     rc;
   u16     block_num;

   set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
   block_num = get_block_num(top, bottom); // get block num

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

   put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);
                                           // send "Block Blank Check" command

   fl_wait(tWT8 * block_num);

   rc = fl_csi_getstatus(tWT8_MAX * block_num);    // get status frame
// switch(rc) {
//
//         case    FLC_NO_ERR:     return  rc;     break;  // case [A]
//         case    FLC_DFTO_ERR:   return  rc;     break;  // case [C]
//         default:                return  rc;     break;  // case [B]
// }
   return rc;
}
```

5. 12　Silicon Signatureコマンド

5. 12. 1　処理手順チャート

Silicon Signatureコマンド処理手順



5. 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。

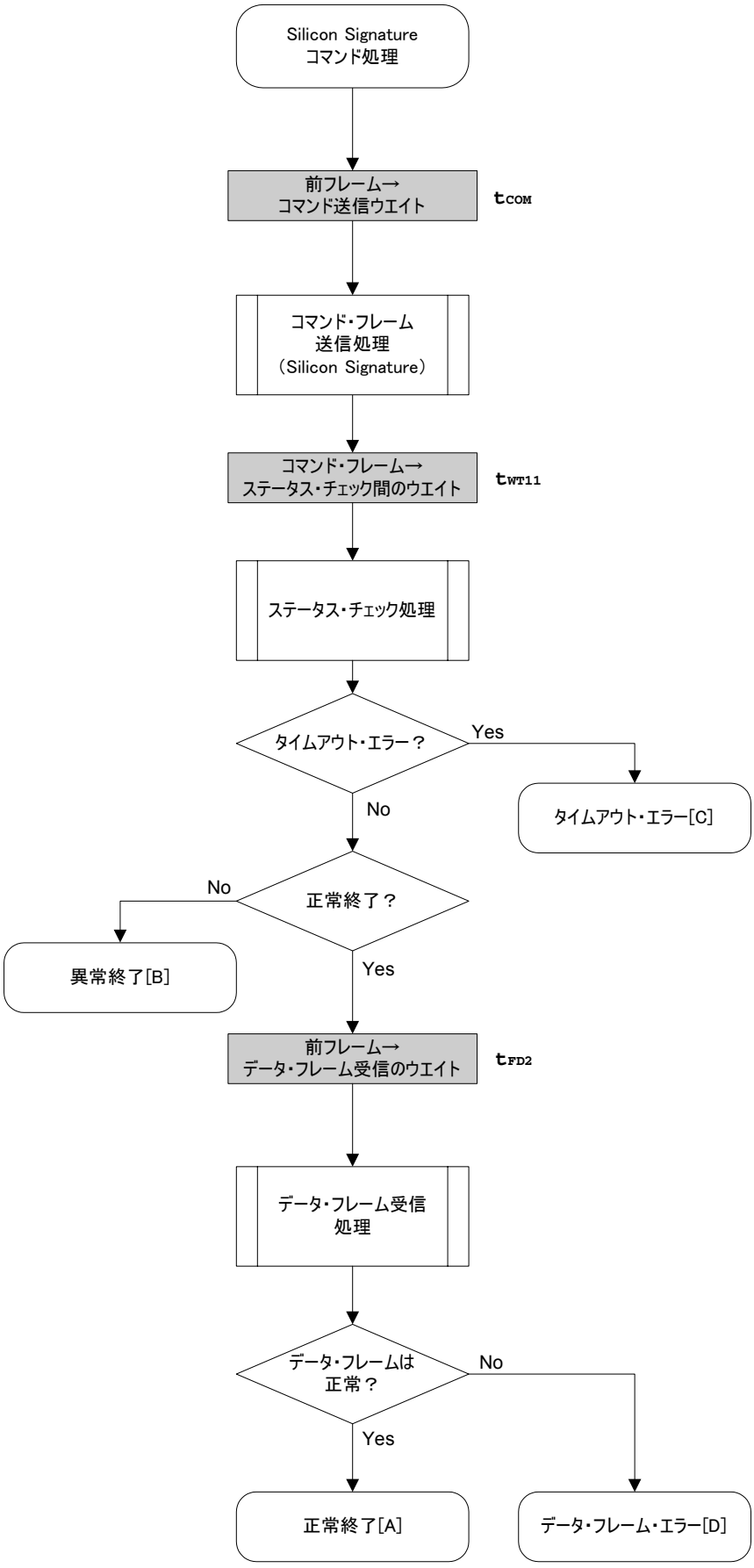
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD2}）。
⑦ 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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　：データ・フレーム・エラー[D]です。

5. 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）。
	リード・エラー	20H	セキュリティ情報の読み出しに失敗しました。
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。
データ・フレーム・エラー [D]		−	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

5. 12. 4　フロー・チャート



5. 12. 5　サンプル・プログラム

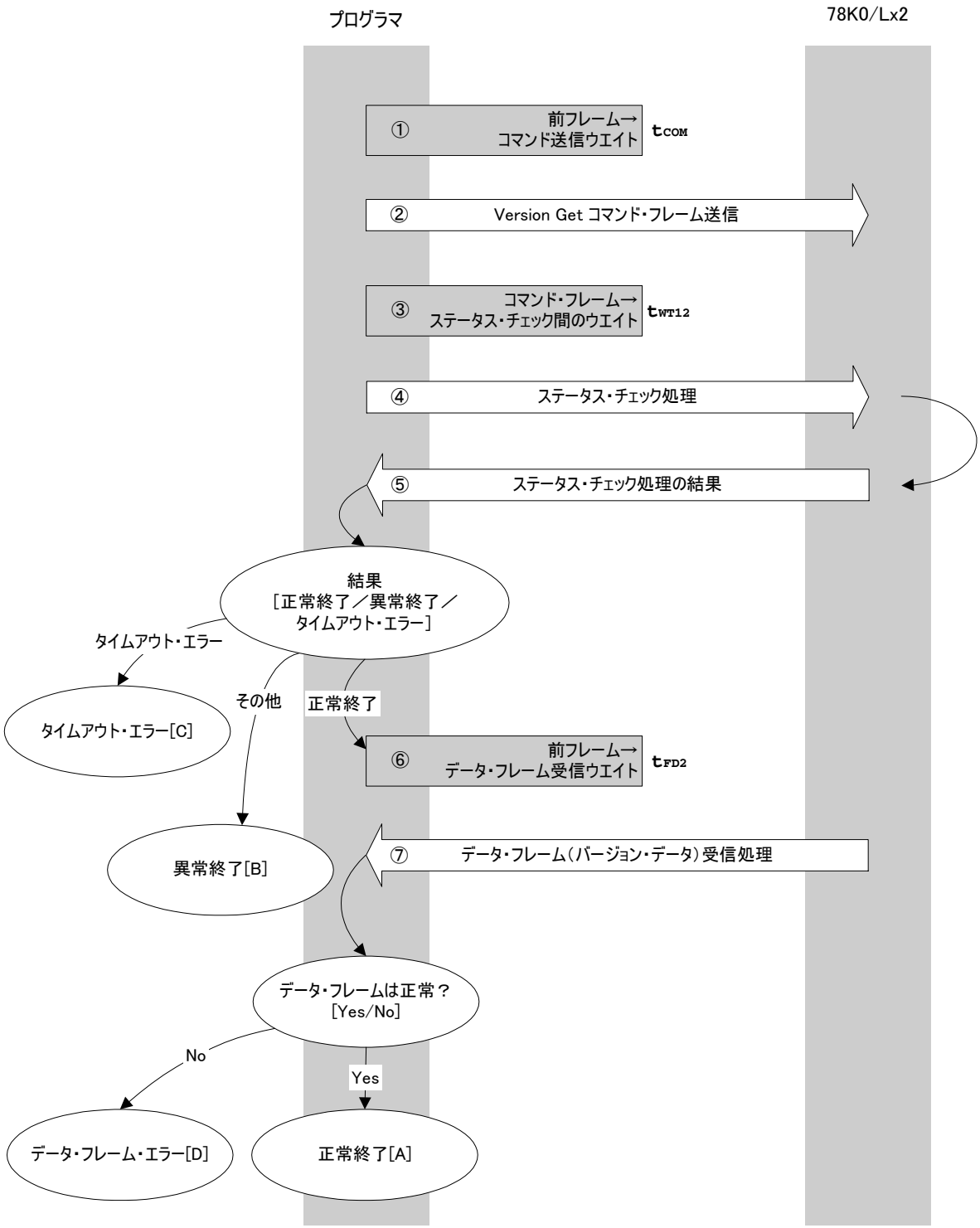
Silicon Signatureコマンド処理のサンプル・プログラムです。

```
/************************************************************/
/*                                                          */
/* 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(tWT11_TO);        // 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]
}
```

5. 13　Version Getコマンド

5. 13. 1　処理手順チャート

Version Getコマンド処理手順



5. 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。

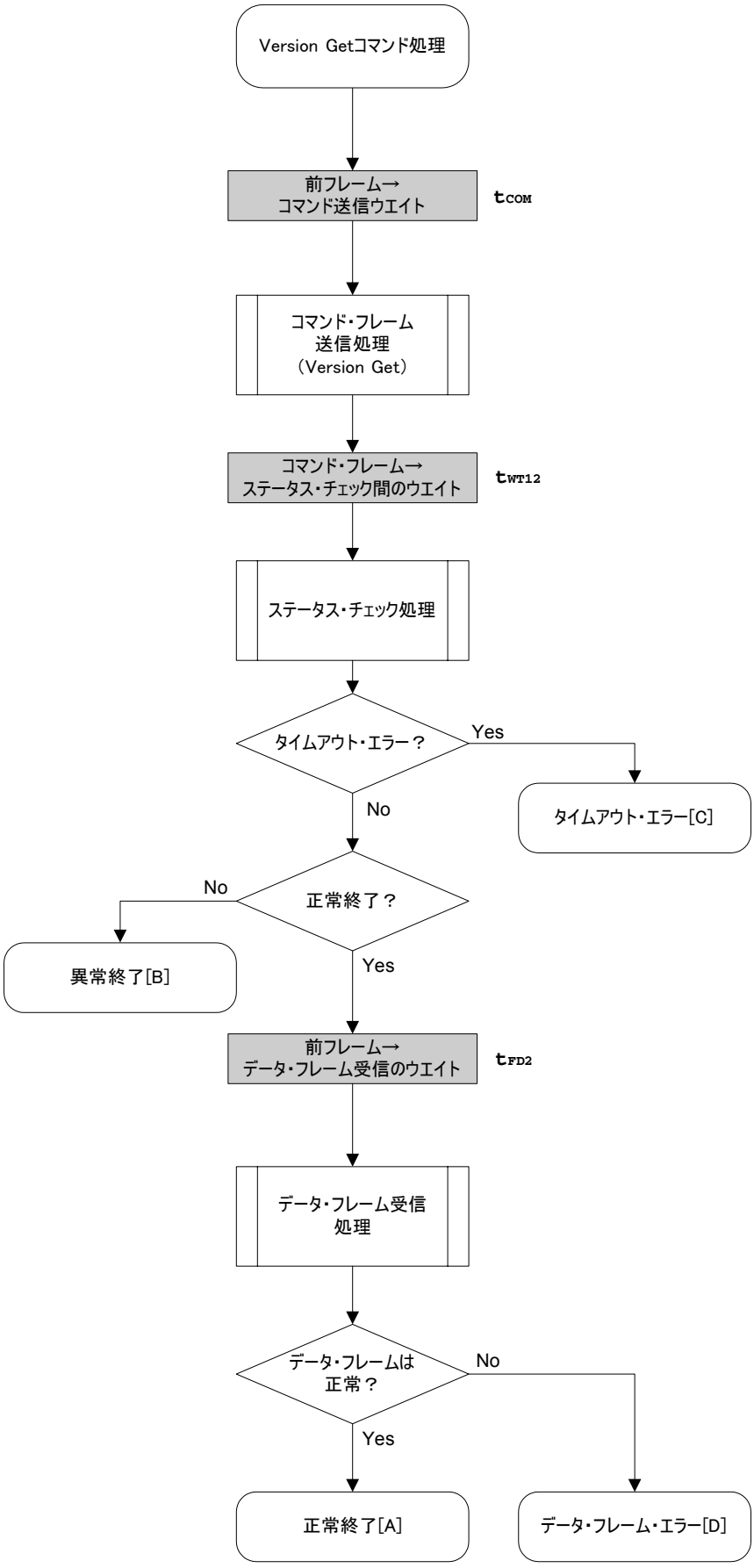
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD2}）。
⑦ 受信したデータ・フレーム（バージョン・データ）をチェックします。

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　：データ・フレーム・エラー[D]です。

5. 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。
データ・フレーム・エラー[D]		−	バージョン・データとして受信したデータ・フレームのチェックサムが異常です。

5. 13. 4　フロー・チャート



5. 13. 5　サンプル・プログラム

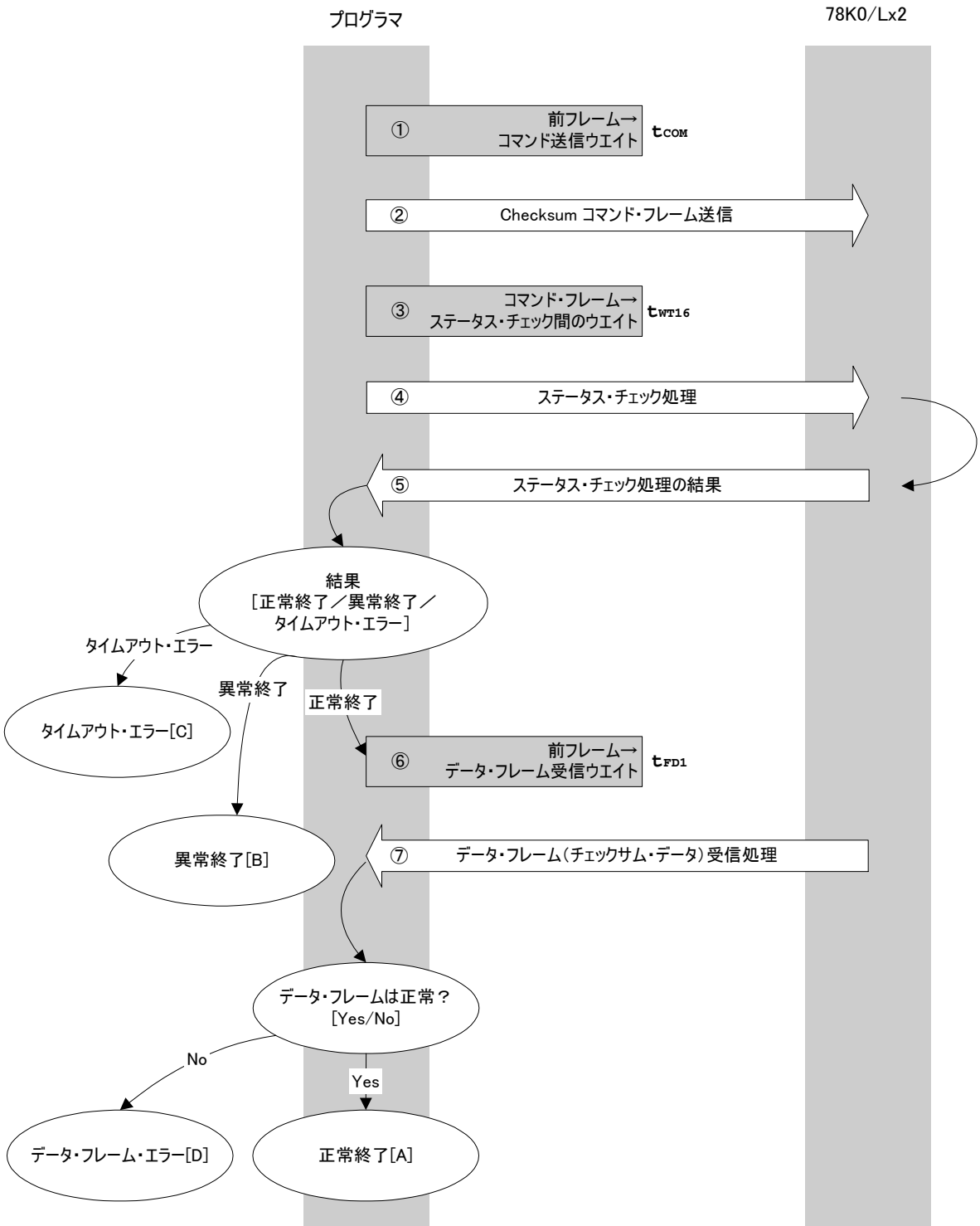
Version Getコマンド処理のサンプル・プログラムです。

```
/*****************************************************************/
/*                                                               */
/* 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(tWT12_TO);        // 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]
}
```

5. 14　Checksumコマンド

5. 14. 1　処理手順チャート

Checksumコマンド処理手順



5. 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。

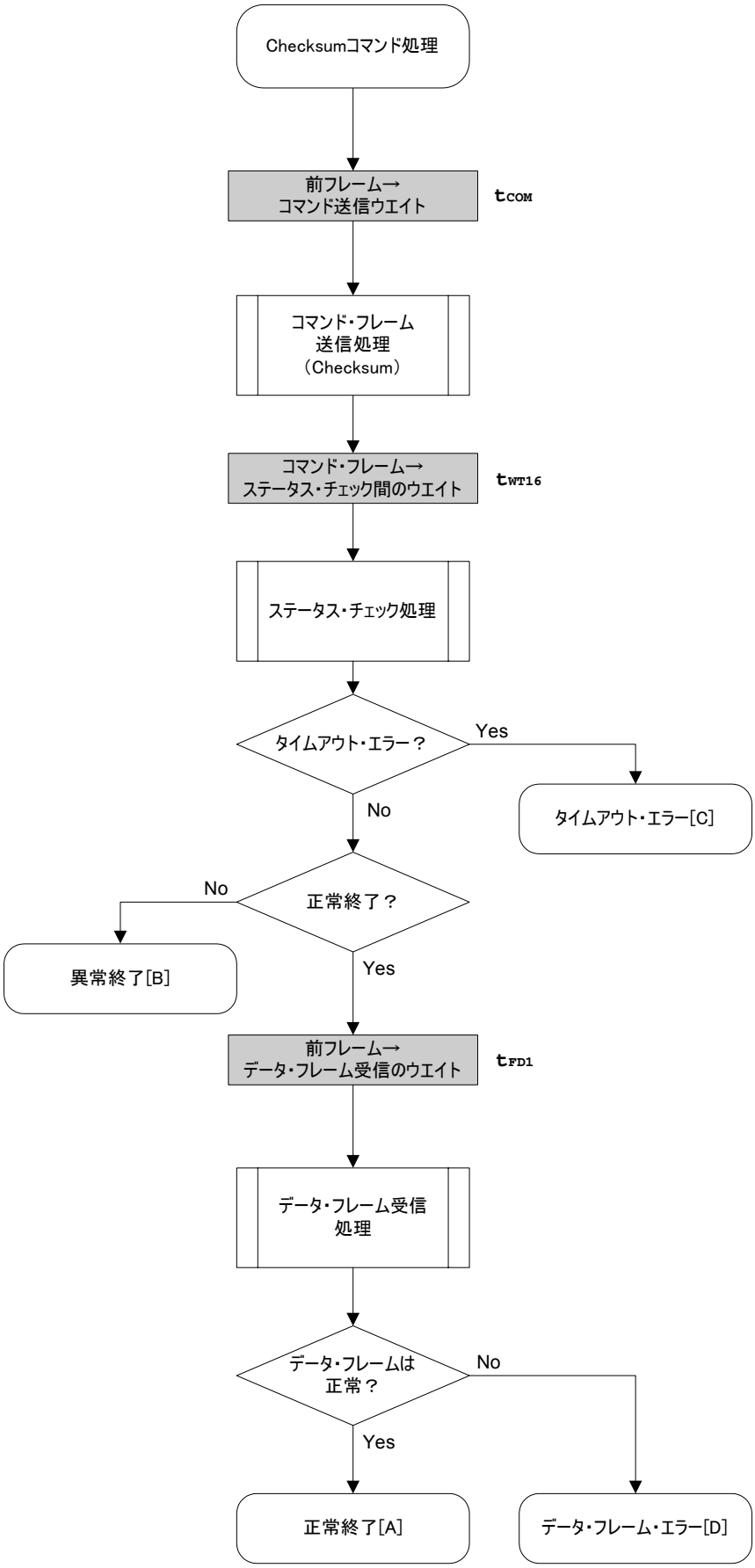
⑥ 直前のフレームからコマンド送信までのウエイトをします（ウエイト時間t_{FD1}）。
⑦ 受信したデータ・フレーム（チェックサム・データ）をチェックします。

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：データ・フレーム・エラー[D]です。

5. 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。
データ・フレーム・エラー [D]		－	チェックサム・データとして受信したデータ・フレームのチェックサムが異常です。

5. 14. 4　フロー・チャート



5. 14. 5　サンプル・プログラム

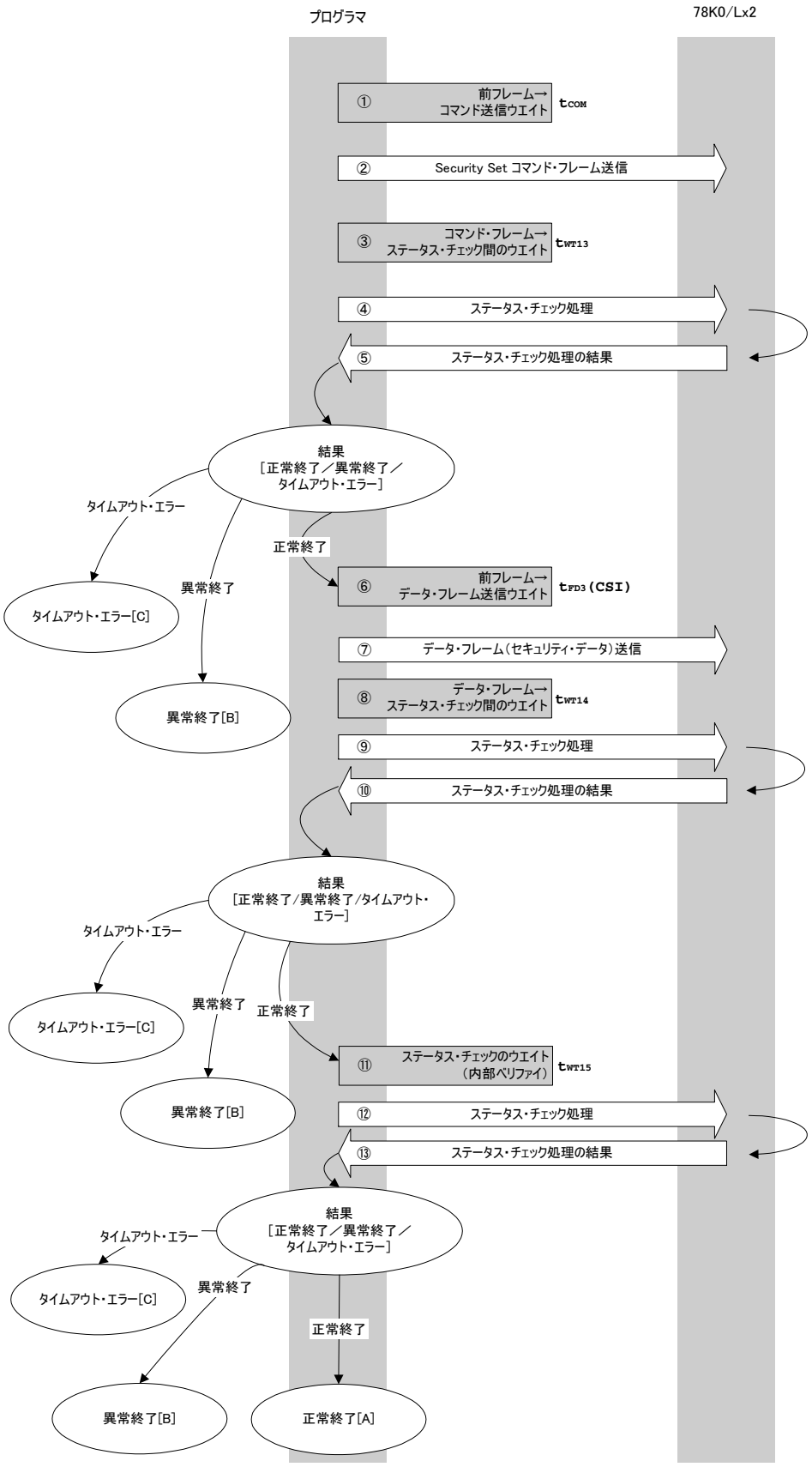
Checksumコマンド処理のサンプル・プログラムです。

```
/************************************************************/
/*                                                          */
/* 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(tWT16_TO);        // 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]
}
```

5. 15　Security Setコマンド

5. 15. 1　処理手順チャート

Security Setコマンド処理手順



5. 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$t_{FD3(CSI)}$）。
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。

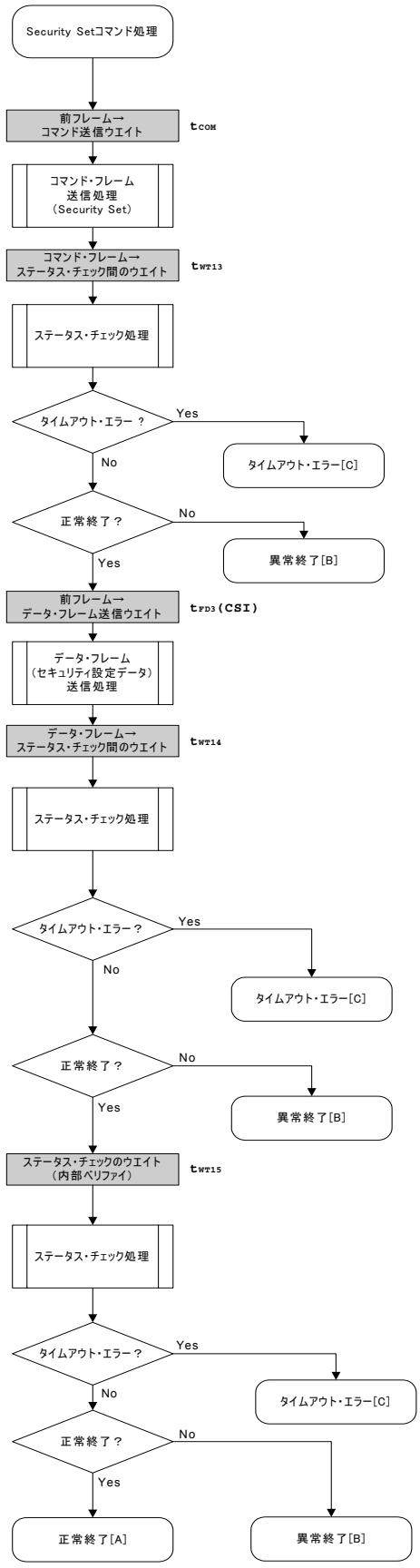
⑪ ステータス取得（内部ベリファイ完了）までのウエイトをします（ウエイト時間t_{WT15}）。
⑫ ステータス・チェック処理により，ステータス・フレームを取得します。
⑬ ステータス・チェック処理の結果に応じて次の処理を行います。

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。

5. 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	コマンド情報（パラメータ）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。

5. 15. 4　フロー・チャート



5. 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 |= 0b11101000);
                                  // "FLG" (upper 5bits must be '1' (to make sure))

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



   /***********************************************/
   /*   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(tWT13_TO);       // 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_CSI);                     // wait before getting data frame



   put_dfrm_csi(2, fl_txdata_frm, true);  // send data frame(Security data)


   fl_wait(tWT14);

   rc = fl_csi_getstatus(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               */
   /*********************************************/
   fl_wait(tWT15);

   rc = fl_csi_getstatus(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6章　フラッシュ・メモリ・プログラミング・パラメータ特性

この章では，フラッシュ・メモリ・プログラミング・モード時のプログラマと78K0/Lx2の間のパラメータ特性を記載しています。その他の電気的特性は，78K0/Lx2各製品のユーザーズ・マニュアルを参照のうえ，設計してください。

6.1　基本特性

項　　目	条　　件	略号	MIN.	TYP.	MAX.	単位
フラッシュ・メモリ・プログラミング・モード時の78K0/Lx2の動作クロック	高速内蔵発振クロック	f_{RH}	7.6	8	8.4	MHz
X1クロック	UART通信時	f_X	2		20	
外部メイン・システム・クロック		f_{EXCLK}	2		20	

★ 6.2　フラッシュ・メモリ・プログラミング・モード設定時間

項　　目		略　号	MIN.	TYP.	MAX.
V_{DD}↑ to FLMD0↑		t_{DP}	1 ms		
FLMD0↑ to $\overline{\text{RESET}}$↑		t_{PR}	2 ms		
Count start time from $\overline{\text{RESET}}$↑ to FLMD0[注1]		t_{RP}	59327/f_{RH}		
Count finish time from $\overline{\text{RESET}}$↑ to FLMD0[注1]		t_{RPE}			238414/ f_{RH}
FLMD0 counter high level width / low level width		t_{PW}	10 μs		100 μs
Wait for Reset command (CSI)		t_{RC}	444463/f_{RH}		
Wait for low level data1 (UART)	X1 clock	t_{R1}	444463/f_{RH}+ 2^{16}/f_X		
	External main system clock		444463/f_{RH}		
Wait for low level data2 (UART)		t_{12}	15000/f_{RH}		
Wait for Read command (UART)		t_{2C}	15000/f_{RH}		
Low level data1 / data2 width[注2]		t_{L1}, t_{L2}		注2	
FLMD0 counter rise time / fall time		−			1 μs
Reset low level width ($\overline{\text{RESET}}$↓ to $\overline{\text{RESET}}$↑) [注3]		t_{RST}	1950 ms		

注1. FLMD0パルスの入力タイミングは，（59327/f_{RH}＋238414/f_{RH}）/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。

3. マイコンの電源立ち上げ後（リセット解除後）に，通常動作モードからフラッシュ・メモリ・プログラミング・モードへ引き込む場合は，モード引き込み時のリセット期間をこの時間以上，確保してください。

備考1. f_{RH} = 8 MHzで計算してください。

2. ウエイトには，次のような定義があります。

＜t_{R1}（MIN.）＞

UARTのボー・レートは，外部クロックで生成しています。

このスペックと，使用する外部クロックの発振安定時間を考慮して，パルス入力を行ってください。

★ 6.3 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	64/f_{RH}	
			UART	74/f_{RH}	
	データ・フレーム送信	t_{DT}	CSI	88/f_{RH}	
			UART	0 注1	
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	166/f_{RH}	
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	54368/f_{RH}	
			UART	0 注1	
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	321/f_{RH}	
	バージョン・データ			136/f_{RH}	
	－		UART	0 注1	
ステータス・フレーム送信～データ・フレーム受信	－	t_{FD3}	CSI	163/f_{RH}	
			UART	101/f_{RH}	
ステータス・フレーム送信～コマンド・フレーム受信	－	t_{COM}	CSI	64/f_{RH}	
			UART	71/f_{RH}	

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1. f_{RH} = 8 MHz**で計算してください。**

2. **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/Lx2**は，直前の通信完了後，**MIN.**後から次の通信が可能となります。**

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

MAX.**時間の規定はありませんが，**3 s**程度を目安に送信を行ってください。**

＜t_{DT}, t_{SF}, t_{FD1}, t_{FD2}＞

78K0/Lx2**は，直前の通信完了後，**MIN.**後から次の通信が可能となります。**

プログラマは，直前の通信完了後，MIN.**時間経過前に次のデータの受信準備を完了してください。**

MAX.**時間の規定はありませんが，データが受信されるまで，**3 s**程度を目安に受信ポーリングを継続してください。**

コマンド	略号	シリアル I/F	MIN.		MAX.
Reset	t_{WT0}	CSI	172/f_{RH}		
		UART	注 1		
Chip Erase	t_{WT1}	―	857883/f_{RH}＋44160×全ブロック数/f_{RH}		186444400/f_{RH}＋11304960×全ブロック数/f_{RH}
Block Erase	t_{WT2} 注2	―	214714/f_{RH}×同時選択消去の実行回数＋44160/f_{RH}×消去するブロック数		54582372/f_{RH}×同時選択消去の実行回数＋11304960/f_{RH}×消去するブロック数
Programming	t_{WT3}	CSI	1348/f_{RH}		
		UART	注 1		
	t_{WT4} 注3	―	68118/f_{RH}		397587/f_{RH}
	t_{WT5} 注4	CSI	ブロック 0	100407/f_{RH}	132144427/f_{RH}
			ブロック 1-127		102178/f_{RH}
		UART	ブロック 0	注 1	132144427/f_{RH}
			ブロック 1-127		102178/f_{RH}
Verify	t_{WT6}	CSI	686/f_{RH}		
		UART	注 1		
	t_{WT7} 注3	CSI	12827/f_{RH}		
		UART	注 1		
Block Blank Check	t_{WT8} 注4	CSI	45835/f_{RH}		55004/f_{RH}
		UART	注 1		55004/f_{RH}
Oscillating Frequency Set	t_{WT9}	CSI	1127/f_{RH}		
		UART	注 1		
Silicon Signature	t_{WT11}	CSI	1233/f_{RH}		
		UART	注 1		
Version Get	t_{WT12}	CSI	242/f_{RH}		
		UART	注 1		
Security Set	t_{WT13}	CSI	923/f_{RH}		
		UART	注 1		
	t_{WT14}	―	275518/f_{RH}		66005812/f_{RH}
	t_{WT15}	CSI	368277/f_{RH}		66018156/f_{RH}
		UART	注 1		66018156/f_{RH}
Checksum	t_{WT16}	CSI	583/f_{RH}		
		UART	注 1		

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. 同時選択消去の実行回数の求め方に関しては，**補足**　Block Eraseコマンドにおける同時選択消去についてを参照してください。

3. 256バイトの場合の時間

4. 1ブロックの場合の時間

備考1. f_{RH} = 8 MHzで計算してください。

2. ウエイトには，次のような定義があります。

＜t_{WT0} - t_{WT16}＞

78K0/Lx2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（MAX.時間の規定のないものは3 s程度を目安に），ステータス・チェックを繰り返す必要があります。

補足　Block Eraseコマンドにおける同時選択消去について

78K0/Lx2のBlock Eraseコマンド実行は，複数ブロックを同時に消去する“同時選択消去”を繰り返し実行することにより実現しています。

したがって，ブロック消去コマンド実行時のウエイト時間は，“同時選択消去”の実行時間の総和となります。

“同時選択消去の実行時間の総和”を算出するためには，同時選択消去の実行回数（M）を算出することが必要です。

Mは，同時に消去するブロック数（同時選択消去ブロック数）を求めながら，算出します。

次に同時選択消去ブロック数とMの求め方を記載します。

（1）同時選択消去ブロック数の求め方

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

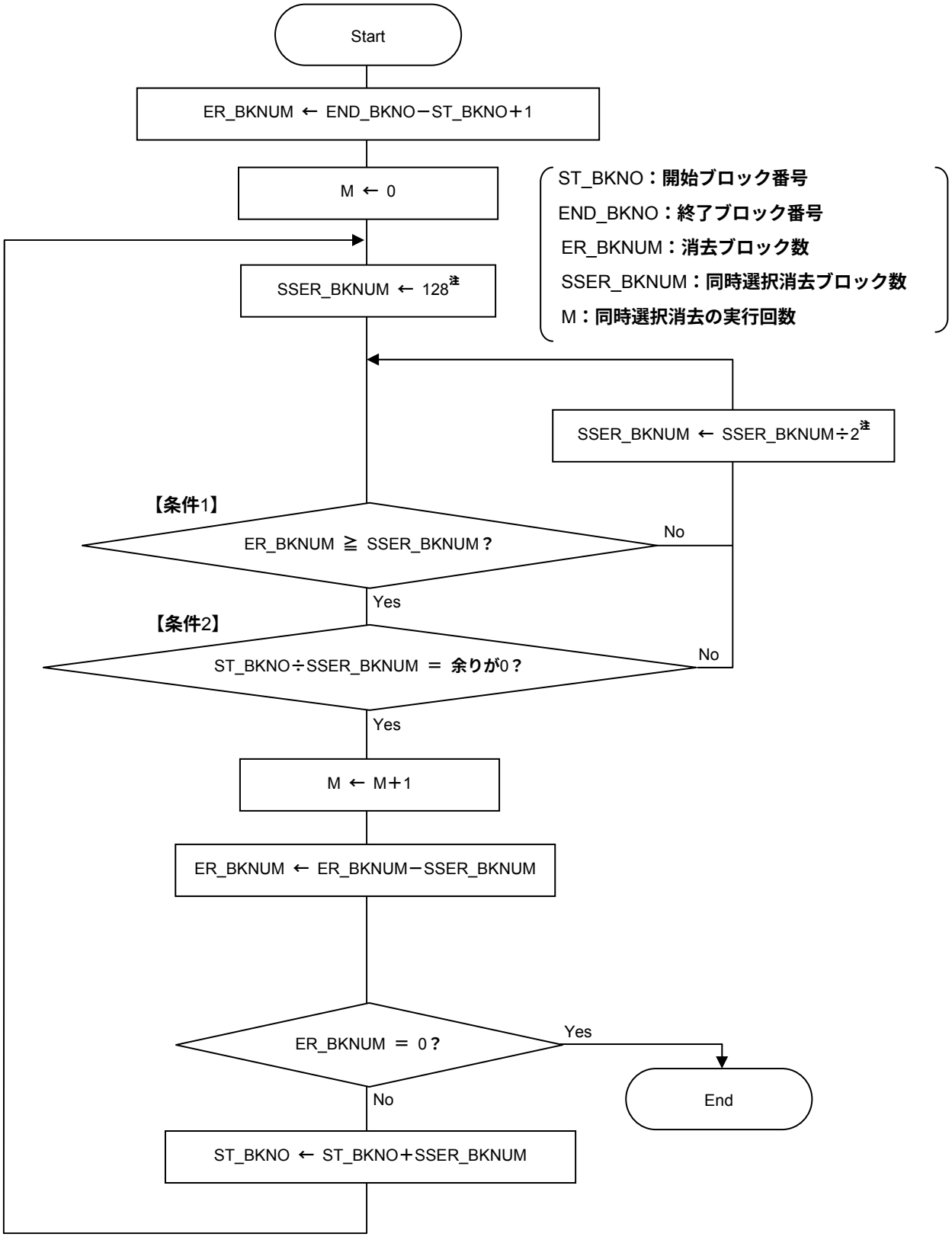
【条件1】
消去ブロック数≧同時選択消去ブロック数

【条件2】
開始ブロック番号÷同時選択消去ブロック数 ＝ 余りが0

【条件3】
【条件1】と【条件2】を満たす最も大きな数値

(2) 同時選択消去の実行回数（M）の求め方

Mの算出方法をフローで表現すると次のようになります。



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

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

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

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

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

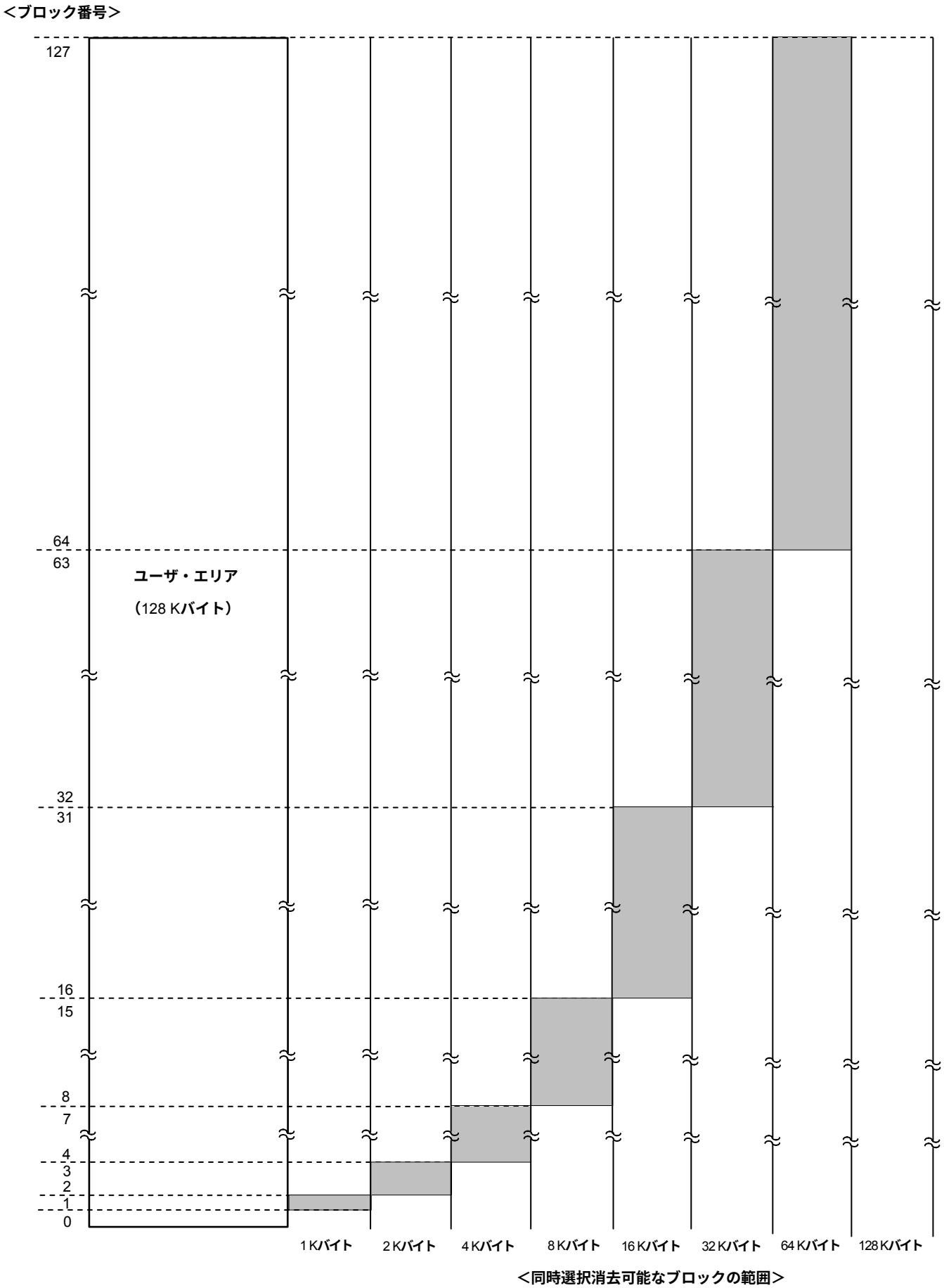
④ **ブロック** 4～7 **を消去すると，次の開始ブロック番号は** 8 **で，消去ブロック数が** 120 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8**となり，【条件**3**】を満たす数値は“** 8 **”であるため，同時選択消去ブロック数は“** 8 **”となることから，ブロック**8～15**を消去します。**

⑤ **ブロック** 8～15 **を消去すると，次の開始ブロック番号は** 16 **で，消去ブロック数が** 112 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16**となり，【条件**3**】を満たす数値は“** 16 **”であるため，同時選択消去ブロック数は“** 16 **”となることから，ブロック**16～31**を消去します。**
ブロック 16～31 **を消去すると，次の開始ブロック番号は** 32 **で，消去ブロック数が** 96 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16, 32**となり，【条件**3**】を満たす数値は“** 32 **”であるため，同時選択消去ブロック数は“** 32 **”となることから，ブロック**32～63**を消去します。**

⑥ **ブロック** 32～63 **を消去すると，次の開始ブロック番号は** 64 **で，消去ブロック数が** 64 **であることから，【条件** 1**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64 **となります。**
さらに【条件2**】を満たす数値は，**1, 2, 4, 8, 16, 32, 64**となり，【条件**3**】を満たす数値は“** 64 **”であるため，同時選択消去ブロック数は“** 64 **”となることから，ブロック**64～127**を消去します。**

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

同時選択消去実行時のブロック構成（ブロック1～127を消去する場合）



例2　ブロック5～10を消去する場合（N（消去するブロック数）= 6）

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

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

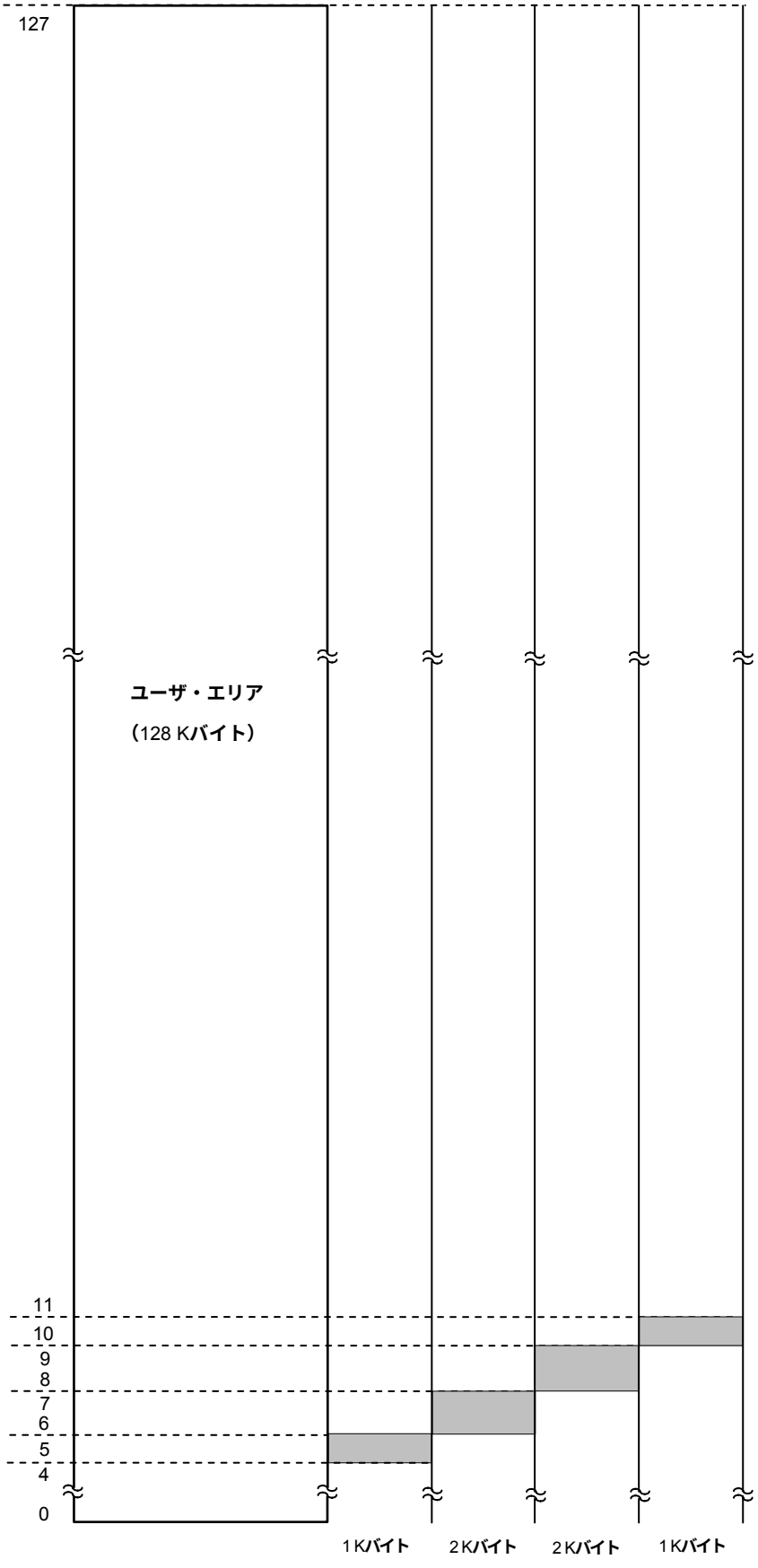
③ **ブロック 6～7 を消去すると，次の開始ブロック番号は 8 で，消去ブロック数が 3 であることから，【条件 1】を満たす数値は，1, 2 となります。**
さらに【条件2】を満たす数値は，1, 2となり，【条件3】を満たす数値は“ 2 ”であるため，同時選択消去ブロック数は“ 2 ”となることから，ブロック8～9を消去する。

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

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

同時選択消去実行時のブロック構成（ブロック5～10を消去する場合）

＜ブロック番号＞



＜同時選択消去可能なブロックの範囲＞

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

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

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

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

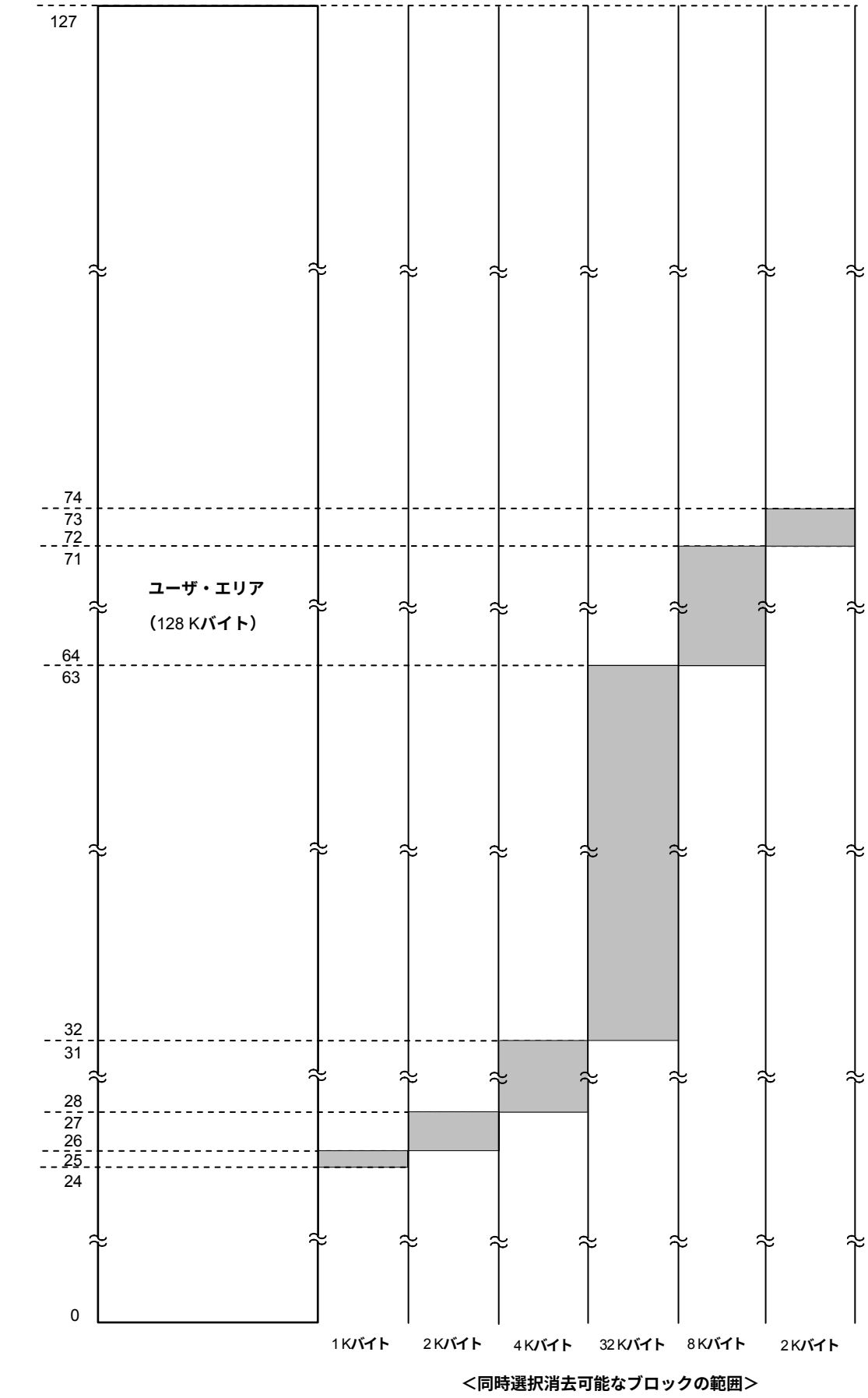
④ **ブロック** 28～31 **を消去すると，次の開始ブロック番号は** 32 **で，消去ブロック数が** 42 **であることから，【条件**1**】を満たす数値は，**1, 2, 4, 8, 16, 32**となります。**
さらに【条件 2**】を満たす数値は，**1, 2, 4, 8, 32 **となり，【条件** 3**】を満たす数値は“** 32 **”であるため，同時選択消去ブロック数は“** 32 **”となることから，ブロック** 32～63 **を消去します。**

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

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

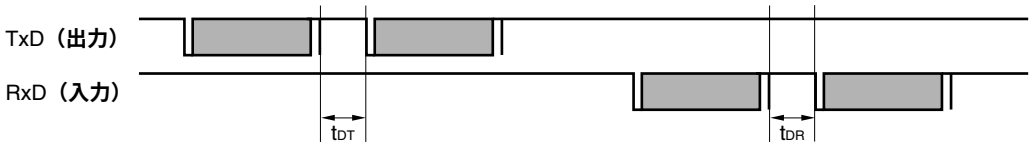
以上より，ブロック25～73**を消去する場合，**25**，**26～27**，**28～31**，**32～63**，**64～71**，**72～73**の**6**回消去されるため，**M = 6**となります。**

同時選択消去実行時のブロック構成（ブロック25～73を消去する場合）

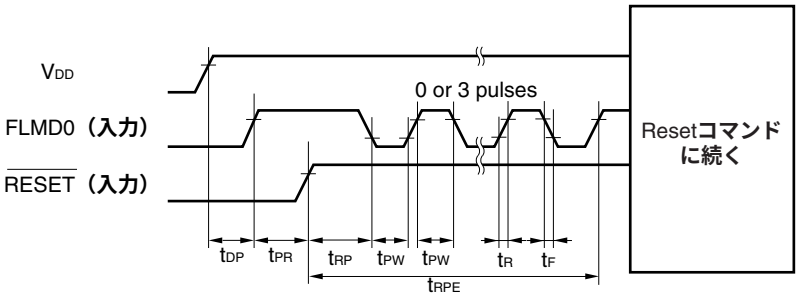


6. 4　UART通信方式

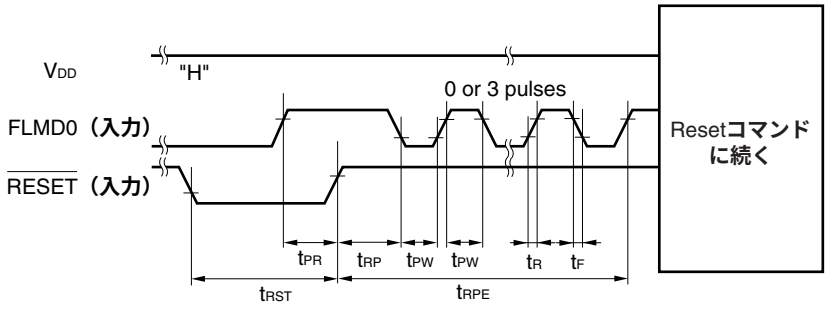
(a) データ・フレーム



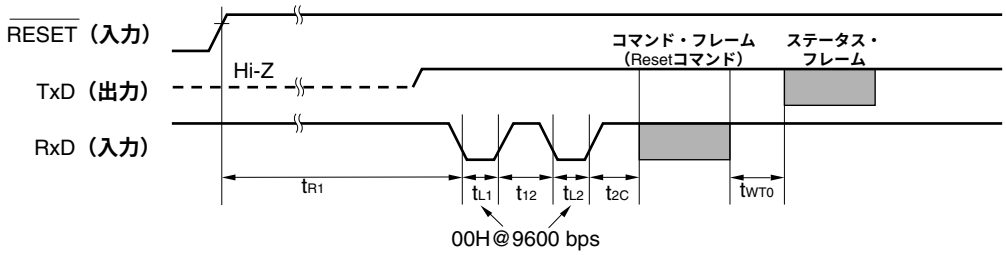
(b) プログラミング・モード設定（電源立ち上げ時）



★ **(c) プログラミング・モード設定（電源立ち上げ後）**



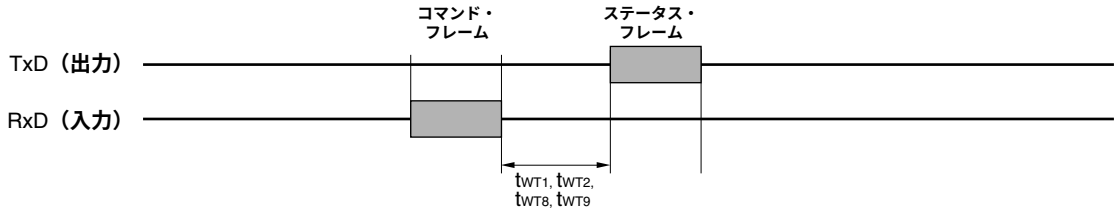
(d) Resetコマンド



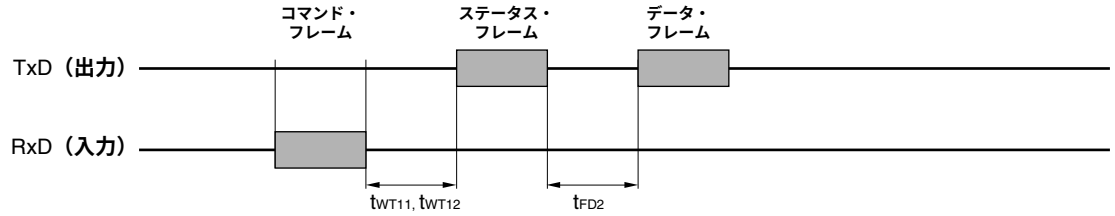
備考　TxD: TxD6

RxD: RxD6

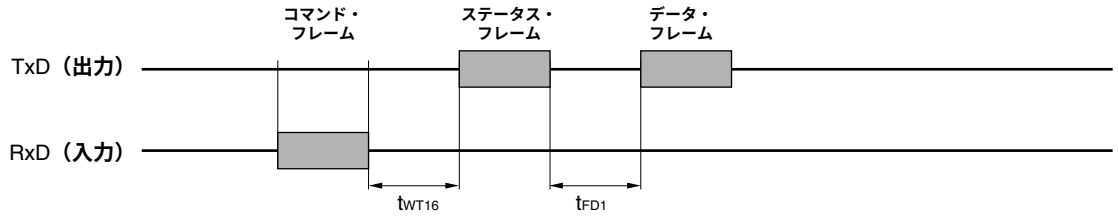
（e）Chip Erase**コマンド**／Block Erase**コマンド**／Block Blank Check**コマンド**／Oscillating Frequency Set**コマンド**



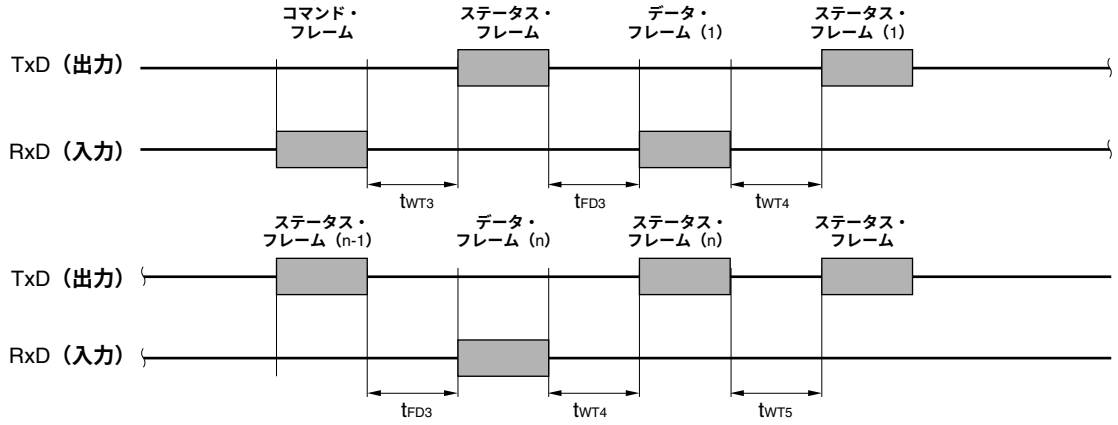
（f）Silicon Signature**コマンド**／Version Get**コマンド**



（g）Checksum**コマンド**



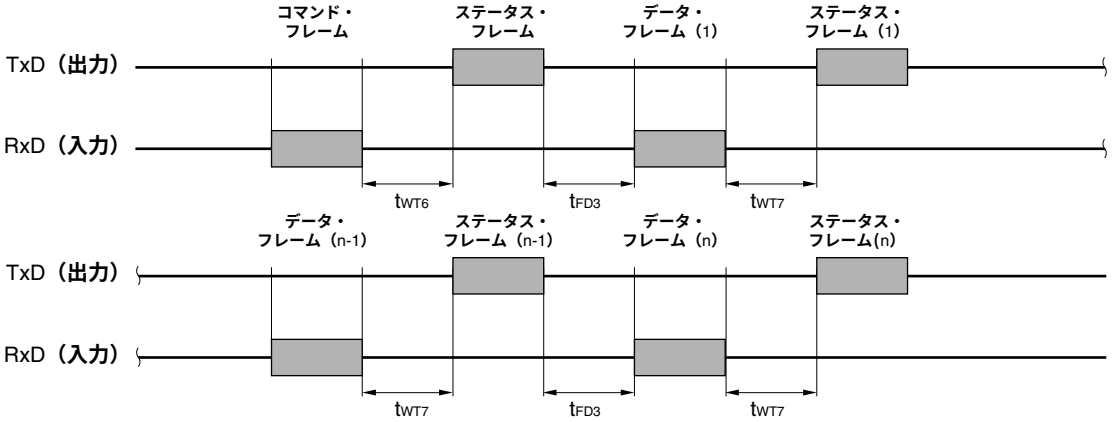
（h）Programming**コマンド**



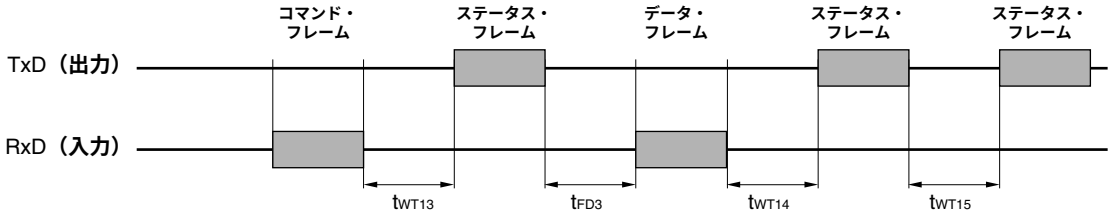
備考　TxD: TxD6

RxD: RxD6

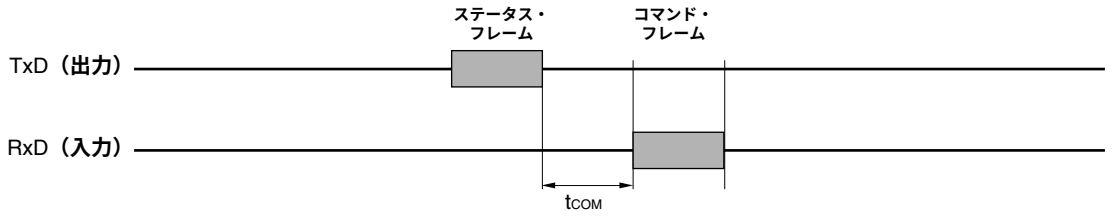
(i) Verifyコマンド



(j) Security Setコマンド



(k) コマンド・フレーム送信前のウエイト

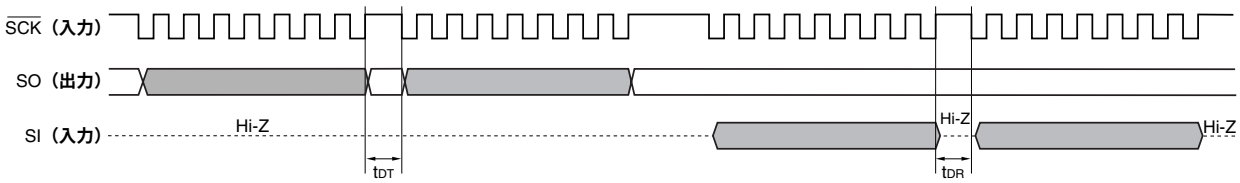


備考　TxD: TxD6

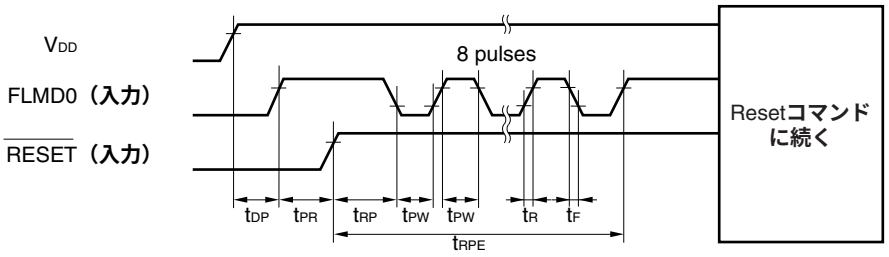
RxD: RxD6

6. 5　3線式シリアルI/O通信方式

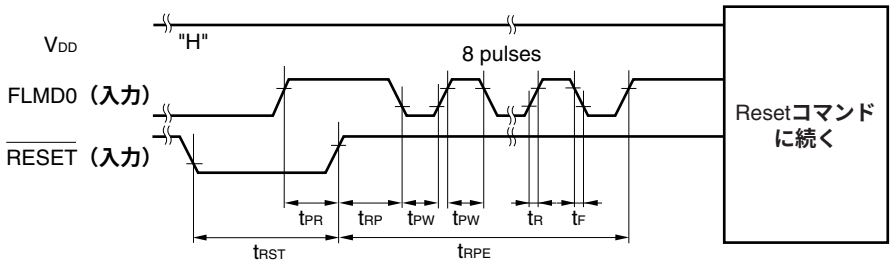
(a) データ・フレーム



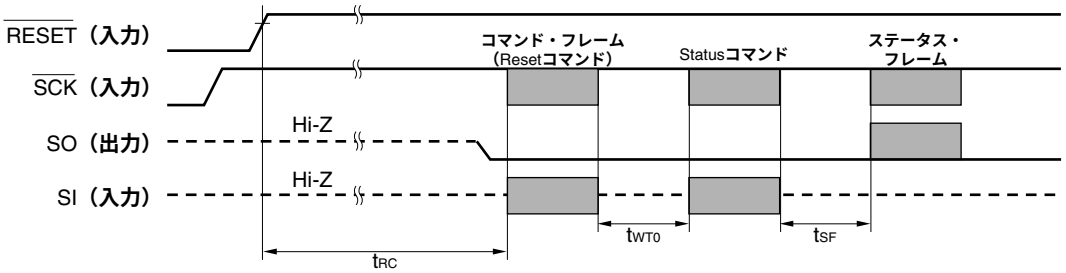
(b) プログラミング・モード設定（電源立ち上げ時）



★ **(c) プログラミング・モード設定（電源立ち上げ後）**



(d) Resetコマンド



★

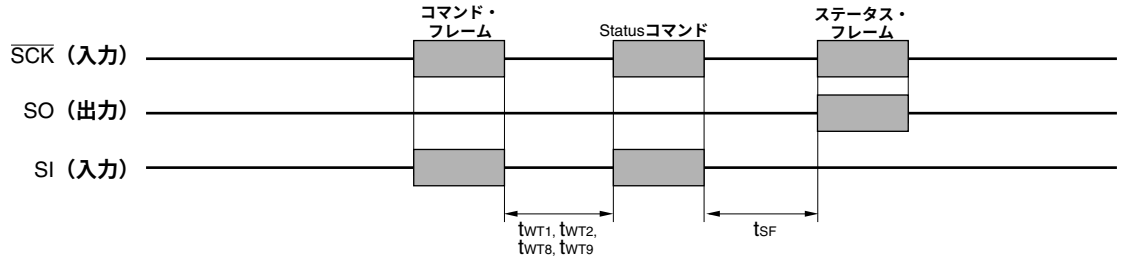
★

備考　SCK: SCK10

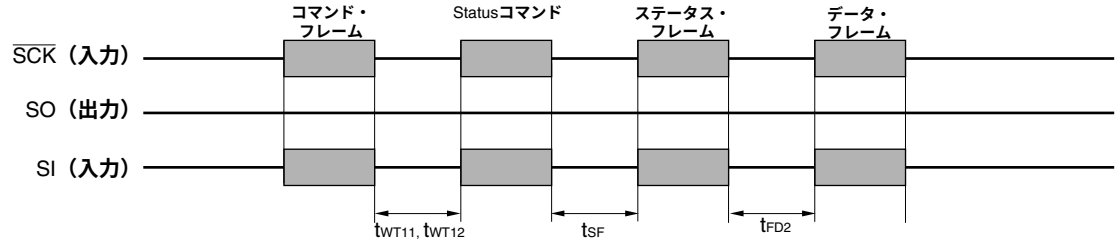
SO:　SO10

SI:　SI10

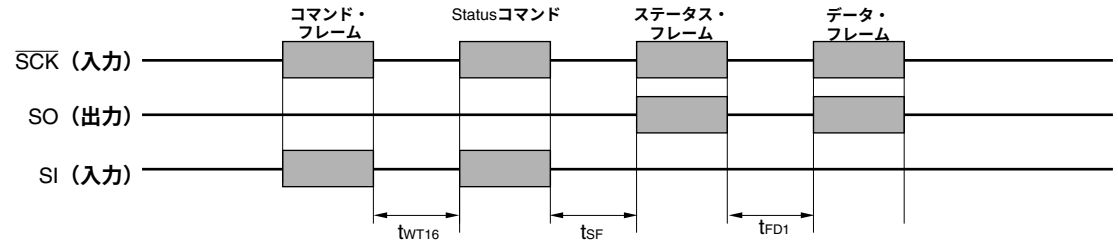
（e）Chip Erase**コマンド**／Block Erase**コマンド**／Block Blank Check**コマンド**／Oscillating Frequency Set**コマンド**



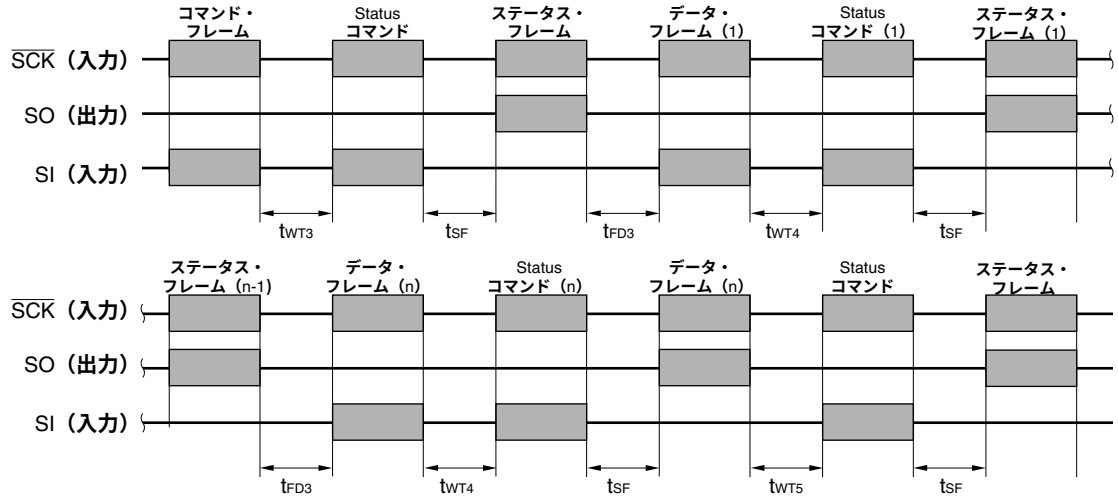
（f）Silicon Signature**コマンド**／Version Get**コマンド**



（g）Checksum**コマンド**



（h）Programming**コマンド**

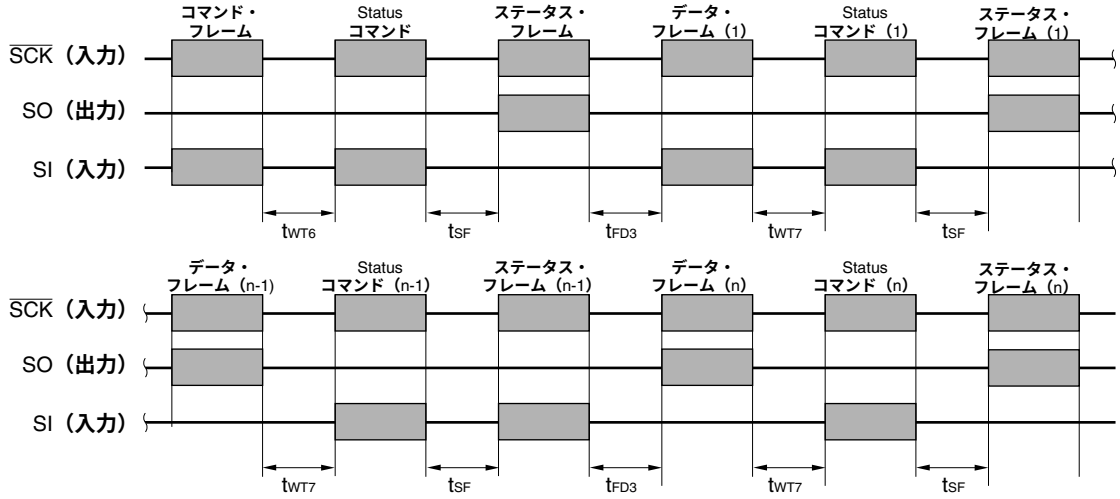


備考　$\overline{\text{SCK}}$: $\overline{\text{SCK10}}$

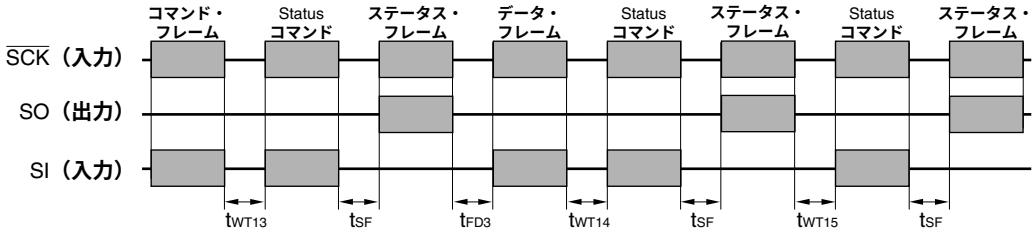
SO:　SO10

SI:　SI10

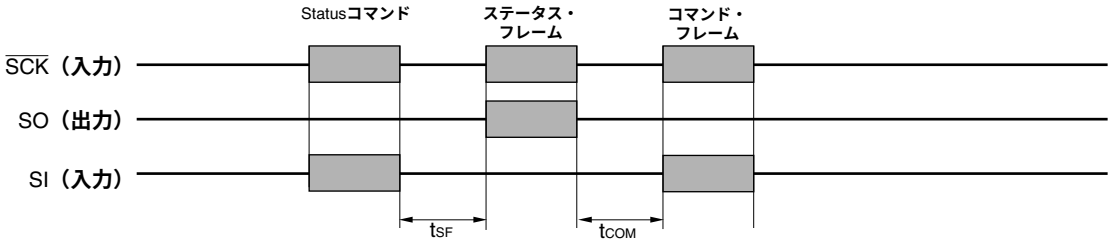
(i) Verifyコマンド



(j) Security Settingコマンド



(k) コマンド・フレーム送信前のウエイト



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

SO:　SO10

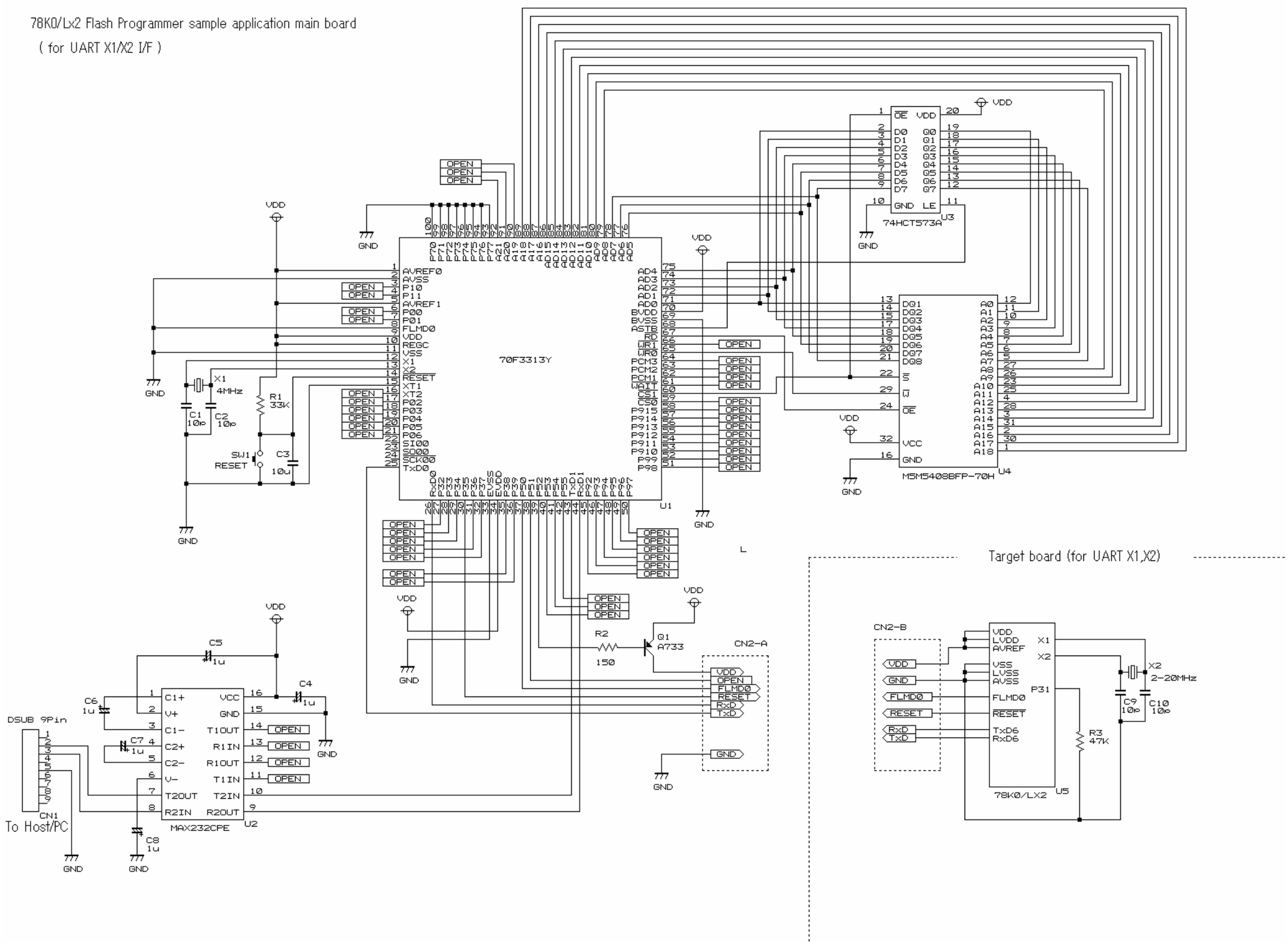
SI:　SI10

付録A　参考回路図

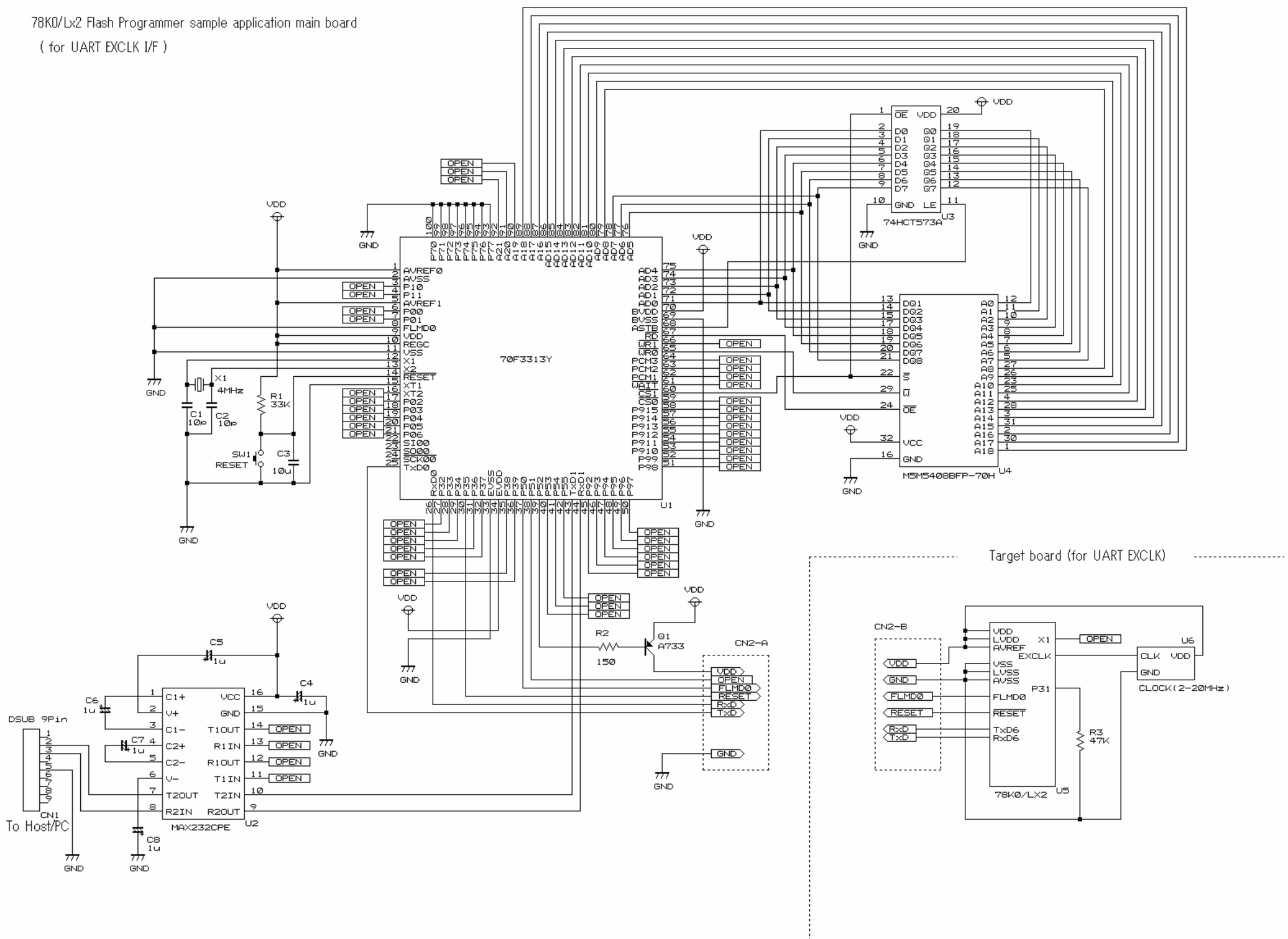
プログラマと78K0/Lx2の参考回路図を図A－1～図A－3に示します。

回路図に記載されていない端子に関しては，各製品のユーザーズ・マニュアルに従い，端子処理を行ってください。

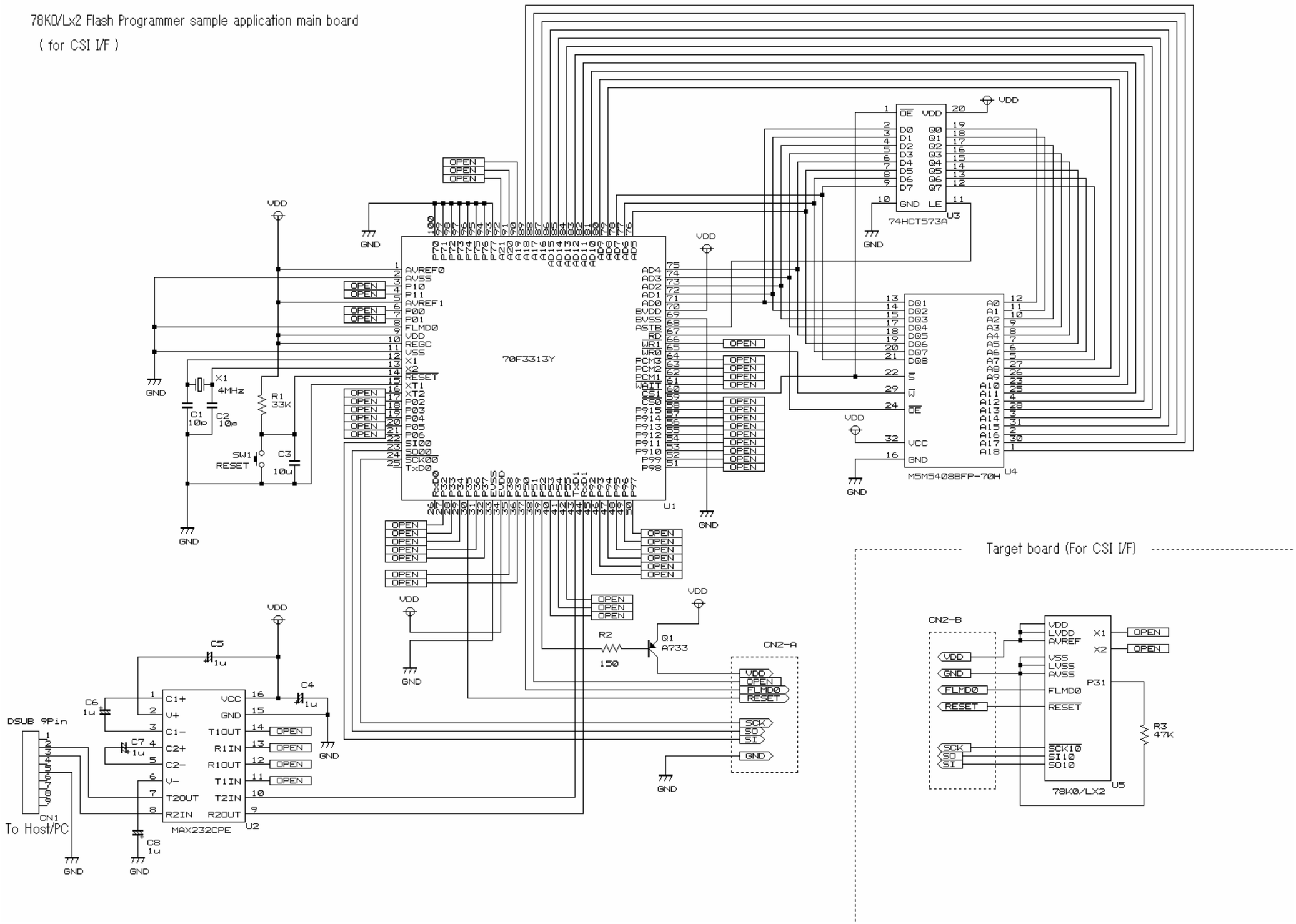
図A－1　プログラマと78K0/Lx2の参考回路図（UART通信：X1クロック使用時）



図A－2　プログラマと78K0/Lx2の参考回路図（UART通信：外部クロック使用時）



図A－3　プログラマと78K0/Lx2の参考回路図（CSI通信時）



★

付録B　改版履歴

B. 1　本版で改訂された主な箇所

箇　所	内　　容
全般	旧版　2. 1　プログラマ制御端子，2. 2　各制御端子の詳細を削除
	旧版　2. 3　基本フロー・チャート～2. 11　ステータス一覧を第1章　フラッシュ・メモリ・プログラミングに移動
p.33	3. 5　Chip Eraseコマンドの記述を変更
p.42	表3－1　シリコン・シグネチャ・データの例を変更
pp.52-54	4. 1　コマンド・フレーム送信処理のフロー・チャート～4. 3　データ・フレーム受信処理のフロー・チャート・フロー・チャート中の略号を変更
pp.104-106	5. 1　コマンド・フレーム送信処理のフロー・チャート～5. 3　データ・フレーム受信処理のフロー・チャート・フロー・チャート中の略号を変更
p.108	5. 4. 3　終了時の内容を修正
p.159	6. 2　フラッシュ・メモリ・プログラミング・モード設定時間を変更
pp.160, 161	6. 3　プログラミング特性を変更
p.170	6. 4　UART通信方式を変更
p.173	6. 5　3線式シリアルI/O通信方式を変更
p.183	付録B　改版履歴を追加

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

お問い合わせ先

【ホームページ】

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

【営業関係，技術関係お問い合わせ先】

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

【資料請求先】

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