

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のですが、誤りが無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、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

アプリケーション・ノート

78K0/Lx3

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)

μPD78F0400	μPD78F0441	μPD78F0471
μPD78F0401	μPD78F0442	μPD78F0472
μPD78F0402	μPD78F0443	μPD78F0473
μPD78F0403	μPD78F0444	μPD78F0474
μPD78F0410	μPD78F0445	μPD78F0475
μPD78F0411	μPD78F0451	μPD78F0481
μPD78F0412	μPD78F0452	μPD78F0482
μPD78F0413	μPD78F0453	μPD78F0483
μPD78F0420	μPD78F0454	μPD78F0484
μPD78F0421	μPD78F0455	μPD78F0485
μPD78F0422	μPD78F0461	μPD78F0491
μPD78F0423	μPD78F0462	μPD78F0492
μPD78F0430	μPD78F0463	μPD78F0493
μPD78F0431	μPD78F0464	μPD78F0494
μPD78F0432	μPD78F0465	μPD78F0495
μPD78F0433		

〔メモ〕

目次要約

第1章	フラッシュ・メモリ・プログラミング	...	14
第2章	プログラマ動作環境	...	20
第3章	プログラマの基本動作	...	34
第4章	コマンド/データ・フレーム・フォーマット	...	35
第5章	コマンド処理説明	...	38
第6章	UART通信方式	...	73
第7章	3線式シリアルI/O (CSI) 通信方式	...	125
第8章	フラッシュ・メモリ・プログラミング・パラメータ特性	...	181
付 録 A	参考回路図	...	198

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。

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。

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イやマガジン・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。

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。

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。

電源OFF時における入力信号

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

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

- 本資料に記載されている内容は2007年11月現在のものです、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に、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。

はじめに

対 象 者 このアプリケーション・ノートは、78K0/Lx3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/Lx3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・プログラムを開発するための方法をユーザに理解していただくことを目的としています。
なお、掲載のプログラムおよび回路図は例示したものであり、量産設計を対象とするものではありません。
したがって、お客様の機器に使用される場合には、設計後、お客様の責任において十分な評価を行ってください。

構 成 このマニュアルは、大きく分けて次の内容で構成しています。

- ・フラッシュ・メモリ・プログラミング
- ・プログラマ動作環境
- ・プログラマの基本動作
- ・コマンド/データ・フレーム・フォーマット
- ・コマンド処理説明
- ・UART通信方式
- ・3線式シリアルI/O (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。
- 78K0/Lx3のハードウェア機能を知りたいとき
→ 78K0/Lx3 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... $\times \times \times \times$ または $\times \times \times \times B$ 10進数... $\times \times \times \times$ 16進数... $\times \times \times \times H$

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/LC3 ユーザーズ・マニュアル	U18698J	U18698E
78K0/LD3 ユーザーズ・マニュアル	U18697J	U18697E
78K0/LE3 ユーザーズ・マニュアル	U18696J	U18696E
78K0/LF3 ユーザーズ・マニュアル	U18329J	U18329E
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 ... 14

- 1.1 概 要 ... 14
- 1.2 システム構成 ... 15
- 1.3 プログラミング概要 ... 16
 - 1.3.1 プログラミング・モードへの遷移 ... 16
 - 1.3.2 シリアル通信方式の選択 ... 16
 - 1.3.3 コマンドの送受信によるフラッシュ・メモリの操作 ... 17
- 1.4 78K0/Lx3製品固有情報 ... 17

第2章 プログラム動作環境 ... 20

- 2.1 プログラム制御端子 ... 20
- 2.2 各制御端子の詳細 ... 21
 - 2.2.1 フラッシュ・メモリ・プログラミング・モード引き込み用端子 (FLMD0) ... 21
 - 2.2.2 シリアル・インタフェース端子 (TxD, RxD, SI, SO, $\overline{\text{SCK}}$) ... 22
 - 2.2.3 リセット制御端子 ($\overline{\text{RESET}}$) ... 23
 - 2.2.4 クロック制御端子 (CLK) ... 23
 - 2.2.5 VDD, GND制御端子 ... 24
 - 2.2.6 その他の端子 ... 24
- 2.3 基本フロー・チャート 25
- 2.4 フラッシュ・メモリ・プログラミング・モードへの遷移方法 ... 26
 - 2.4.1 モード引き込みのフロー・チャート ... 27
 - 2.4.2 サンプル・プログラム ... 28
- 2.5 シリアル通信方式の選択 ... 30
- 2.6 UART通信方式 ... 30
- 2.7 3線式シリアルI/O (CSI) 通信方式 ... 31
- 2.8 ターゲットの電源遮断処理 ... 31
- 2.9 フラッシュ・メモリの操作方法 ... 31
- 2.10 コマンド一覧 ... 32
- 2.11 ステータス一覧 ... 33

第3章 プログラムの基本動作 ... 34

第4章 コマンド/データ・フレーム・フォーマット ... 35

- 4.1 コマンド・フレーム送信処理 ... 37
- 4.2 データ・フレーム送信処理 ... 37
- 4.3 データ・フレーム受信処理 ... 37

第5章 コマンド処理説明 ... 38

- 5.1 Statusコマンド ... 38
 - 5.1.1 説明 ... 38
 - 5.1.2 コマンド・フレームとステータス・フレーム ... 38
- 5.2 Resetコマンド ... 39
 - 5.2.1 説明 ... 39
 - 5.2.2 コマンド・フレームとステータス・フレーム ... 39
- 5.3 Baud Rate Setコマンド ... 40
- 5.4 Oscillating Frequency Setコマンド ... 41
 - 5.4.1 説明 41
 - 5.4.2 コマンド・フレームとステータス・フレーム ... 41
- 5.5 Chip Eraseコマンド ... 43
 - 5.5.1 説明 ... 43
 - 5.5.2 コマンド・フレームとステータス・フレーム ... 43
- 5.6 Block Eraseコマンド ... 44
 - 5.6.1 説明 ... 44
 - 5.6.2 コマンド・フレームとステータス・フレーム ... 44
- 5.7 Programmingコマンド ... 45
 - 5.7.1 説明 ... 45
 - 5.7.2 コマンド・フレームとステータス・フレーム ... 45
 - 5.7.3 データ・フレームとステータス・フレーム ... 45
 - 5.7.4 全データ転送完了とステータス・フレーム ... 46
- 5.8 Verifyコマンド ... 47
 - 5.8.1 説明 ... 47
 - 5.8.2 コマンド・フレームとステータス・フレーム ... 47
 - 5.8.3 データ・フレームとステータス・フレーム ... 47
- 5.9 Block Blank Checkコマンド ... 49
 - 5.9.1 説明 ... 49
 - 5.9.2 コマンド・フレームとステータス・フレーム ... 49
- 5.10 Silicon Signatureコマンド ... 50
 - 5.10.1 説明 ... 50
 - 5.10.2 コマンド・フレームとステータス・フレーム ... 50
 - 5.10.3 シリコン・シグネチャ・データ・フレーム ... 51
 - 5.10.4 78K0/Lx3シリコン・シグネチャー一覧 ... 54
- 5.11 Version Getコマンド ... 67
 - 5.11.1 説明 ... 67
 - 5.11.2 コマンド・フレームとステータス・フレーム ... 67
 - 5.11.3 バージョン・データ・フレーム ... 68
- 5.12 Checksumコマンド ... 69
 - 5.12.1 説明 ... 69
 - 5.12.2 コマンド・フレームとステータス・フレーム ... 69
 - 5.12.3 チェックサム・データ・フレーム ... 69
- 5.13 Security Setコマンド ... 70
 - 5.13.1 説明 ... 70
 - 5.13.2 コマンド・フレームとステータス・フレーム ... 70
 - 5.13.3 データ・フレームとステータス・フレーム ... 71
 - 5.13.4 内部ベリファイ確認とステータス・フレーム ... 71

第6章 UART通信方式 ... 73

6.1	コマンド・フレーム送信処理のフロー・チャート ...	74
6.2	データ・フレーム送信処理のフロー・チャート ...	75
6.3	データ・フレーム受信処理のフロー・チャート ...	76
6.4	Resetコマンド ...	77
6.4.1	処理手順チャート ...	77
6.4.2	処理手順説明 ...	78
6.4.3	終了時の内容 ...	78
6.4.4	フロー・チャート ...	79
6.4.5	サンプル・プログラム ...	80
6.5	Oscillating Frequency Setコマンド ...	81
6.5.1	処理手順チャート ...	81
6.5.2	処理手順説明 ...	82
6.5.3	終了時の内容 ...	82
6.5.4	フロー・チャート ...	83
6.5.5	サンプル・プログラム ...	84
6.6	Chip Eraseコマンド ...	85
6.6.1	処理手順チャート ...	85
6.6.2	処理手順説明 ...	86
6.6.3	終了時の内容 ...	86
6.6.4	フロー・チャート ...	87
6.6.5	サンプル・プログラム ...	88
6.7	Block Eraseコマンド ...	89
6.7.1	処理手順チャート ...	89
6.7.2	処理手順説明 ...	90
6.7.3	終了時の内容 ...	90
6.7.4	フロー・チャート ...	91
6.7.5	サンプル・プログラム ...	92
6.8	Programmingコマンド ...	93
6.8.1	処理手順チャート ...	93
6.8.2	処理手順説明 ...	94
6.8.3	終了時の内容 ...	95
6.8.4	フロー・チャート ...	96
6.8.5	サンプル・プログラム ...	97
6.9	Verifyコマンド ...	99
6.9.1	処理手順チャート ...	99
6.9.2	処理手順説明 ...	100
6.9.3	終了時の内容 ...	100
6.9.4	フロー・チャート ...	101
6.9.5	サンプル・プログラム ...	102
6.10	Block Blank Checkコマンド ...	104
6.10.1	処理手順チャート ...	104
6.10.2	処理手順説明 ...	105
6.10.3	終了時の内容 ...	105
6.10.4	フロー・チャート ...	106
6.10.5	サンプル・プログラム ...	107
6.11	Silicon Signatureコマンド ...	108
6.11.1	処理手順チャート ...	108
6.11.2	処理手順説明 ...	109

6. 11. 3	終了時の内容	...	109
6. 11. 4	フロー・チャート	...	110
6. 11. 5	サンプル・プログラム	...	111
6. 12	Version Getコマンド	...	112
6. 12. 1	処理手順チャート	...	112
6. 12. 2	処理手順説明	...	113
6. 12. 3	終了時の内容	...	113
6. 12. 4	フロー・チャート	...	114
6. 12. 5	サンプル・プログラム	...	115
6. 13	Checksumコマンド	...	116
6. 13. 1	処理手順チャート	...	116
6. 13. 2	処理手順説明	...	117
6. 13. 3	終了時の内容	...	117
6. 13. 4	フロー・チャート	...	118
6. 13. 5	サンプル・プログラム	...	119
6. 14	Security Setコマンド	...	120
6. 14. 1	処理手順チャート	...	120
6. 14. 2	処理手順説明	...	121
6. 14. 3	終了時の内容	...	121
6. 14. 4	フロー・チャート	...	122
6. 14. 5	サンプル・プログラム	...	123

第7章 3線式シリアルI/O (CSI) 通信方式 ... 125

7. 1	コマンド・フレーム送信処理のフロー・チャート	...	126
7. 2	データ・フレーム送信処理のフロー・チャート	...	127
7. 3	データ・フレーム受信処理のフロー・チャート	...	128
7. 4	Statusコマンド	...	129
7. 4. 1	処理手順チャート	...	129
7. 4. 2	処理手順説明	...	130
7. 4. 3	終了時の内容	...	130
7. 4. 4	フロー・チャート	...	131
7. 4. 5	サンプル・プログラム	...	132
7. 5	Resetコマンド	...	133
7. 5. 1	処理手順チャート	...	133
7. 5. 2	処理手順説明	...	134
7. 5. 3	終了時の内容	...	134
7. 5. 4	フロー・チャート	...	135
7. 5. 5	サンプル・プログラム	...	136
7. 6	Oscillating Frequency Setコマンド	...	137
7. 6. 1	処理手順チャート	...	137
7. 6. 2	処理手順説明	...	138
7. 6. 3	終了時の内容	...	138
7. 6. 4	フロー・チャート	...	139
7. 6. 5	サンプル・プログラム	...	140
7. 7	Chip Eraseコマンド	...	141
7. 7. 1	処理手順チャート	...	141
7. 7. 2	処理手順説明	...	142
7. 7. 3	終了時の内容	...	142
7. 7. 4	フロー・チャート	...	143

7.7.5	サンプル・プログラム	...	144
7.8	Block Erase コマンド	...	145
7.8.1	処理手順チャート	...	145
7.8.2	処理手順説明	...	146
7.8.3	終了時の内容	...	146
7.8.4	フロー・チャート	...	147
7.8.5	サンプル・プログラム	...	148
7.9	Programming コマンド	...	149
7.9.1	処理手順チャート	...	149
7.9.2	処理手順説明	...	150
7.9.3	終了時の内容	...	151
7.9.4	フロー・チャート	...	152
7.9.5	サンプル・プログラム	...	153
7.10	Verify コマンド	...	155
7.10.1	処理手順チャート	...	155
7.10.2	処理手順説明	...	156
7.10.3	終了時の内容	...	156
7.10.4	フロー・チャート	...	157
7.10.5	サンプル・プログラム	...	158
7.11	Block Blank Check コマンド	...	160
7.11.1	処理手順チャート	...	160
7.11.2	処理手順説明	...	161
7.11.3	終了時の内容	...	161
7.11.4	フロー・チャート	...	162
7.11.5	サンプル・プログラム	...	163
7.12	Silicon Signature コマンド	...	164
7.12.1	処理手順チャート	...	164
7.12.2	処理手順説明	...	165
7.12.3	終了時の内容	...	165
7.12.4	フロー・チャート	...	166
7.12.5	サンプル・プログラム	...	167
7.13	Version Get コマンド	...	168
7.13.1	処理手順チャート	...	168
7.13.2	処理手順説明	...	169
7.13.3	終了時の内容	...	169
7.13.4	フロー・チャート	...	170
7.13.5	サンプル・プログラム	...	171
7.14	Checksum コマンド	...	172
7.14.1	処理手順チャート	...	172
7.14.2	処理手順説明	...	173
7.14.3	終了時の内容	...	173
7.14.4	フロー・チャート	...	174
7.14.5	サンプル・プログラム	...	175
7.15	Security Set コマンド	...	176
7.15.1	処理手順チャート	...	176
7.15.2	処理手順説明	...	177
7.15.3	終了時の内容	...	177
7.15.4	フロー・チャート	...	178
7.15.5	サンプル・プログラム	...	179

第8章	フラッシュ・メモリ・プログラミング・パラメータ特性	...	181
8.1	フラッシュ・メモリ・プログラミング・モード設定時間	...	181
8.2	プログラミング特性	...	182
8.3	UART通信方式	...	192
8.4	3線式シリアルI/O通信方式	...	195
付録A	参考回路図	...	198

第1章 フラッシュ・メモリ・プログラミング

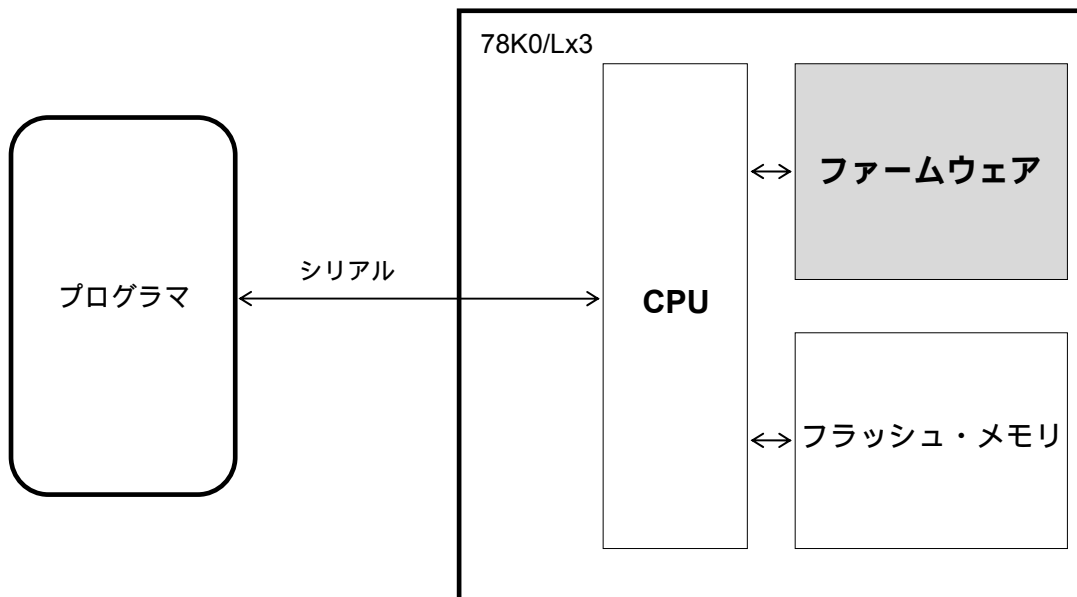
78K0/Lx3に内蔵されるフラッシュ・メモリの書き換えを行うには、通常は専用のフラッシュ・メモリ・プログラマ（以降プログラマ）を使う必要があります。

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/Lx3は、フラッシュ・メモリ書き換え制御を行うファームウェ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/Lx3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/Lx3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。

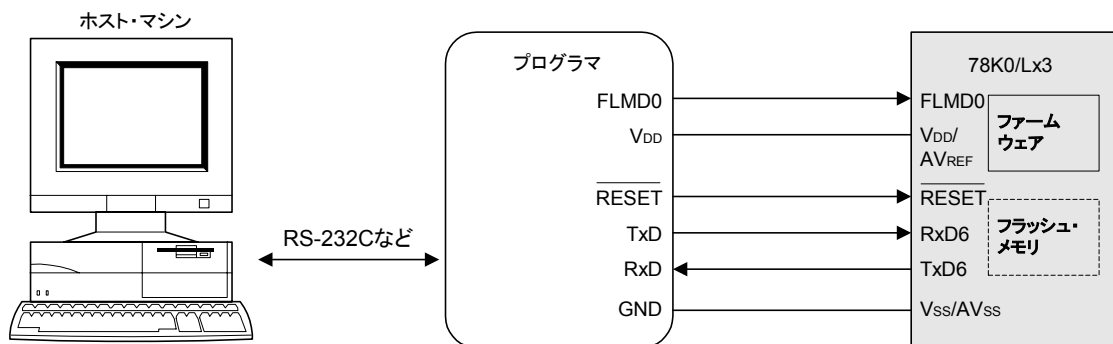
図1 - 2は、ホスト・マシンからの制御によりプログラマを使用するプログラミング方法を示しています。

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。

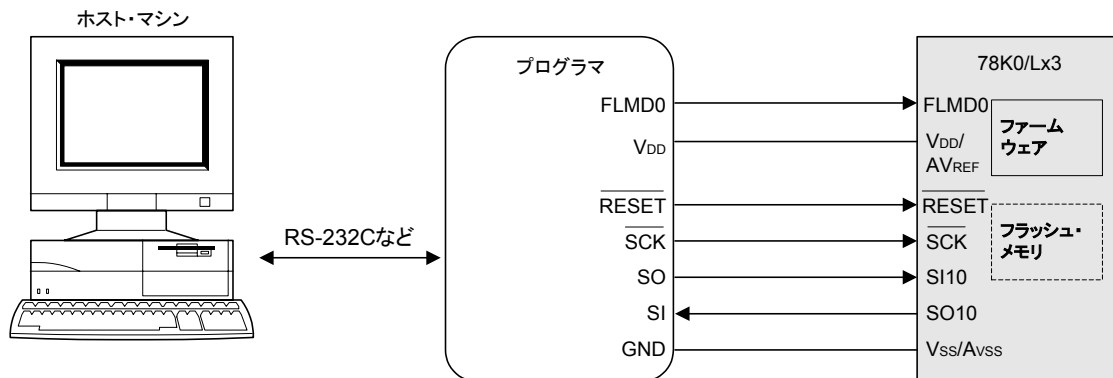
たとえば、NECエレクトロニクス製フラッシュ・メモリ・プログラマ PG-FP5は、ホスト・マシンを接続してGUIソフトウェアにより実行する方法と、スタンド・アローンで実行する方法のどちらでも動作可能です。

図1 - 2 システム構成

(1) UART通信方式 (LSB先頭転送)



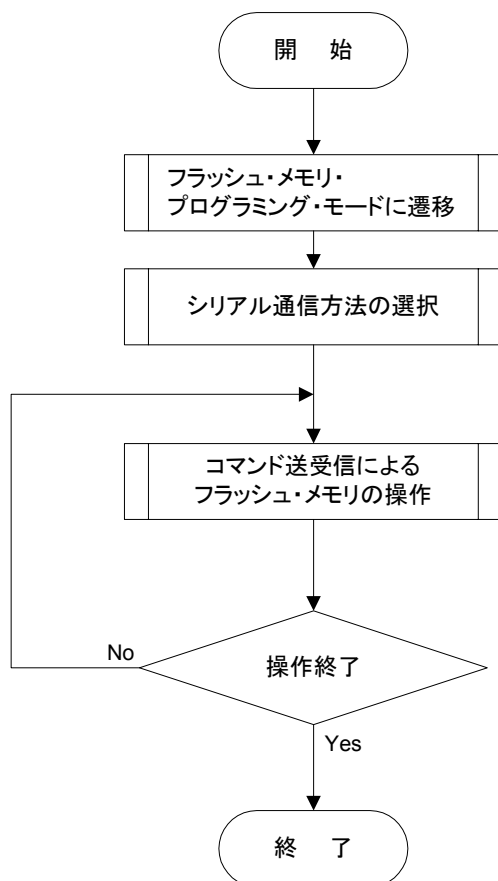
(2) 3線式シリアルI/O (CSI) 通信方式 (MSB先頭転送)



1.3 プログラミング概要

プログラマにてフラッシュ・メモリの書き換えを行うには、まず78K0/Lx3の動作モードをフラッシュ・メモリ・プログラミング・モードに遷移させる必要があります。その後、プログラマと78K0/Lx3間の通信方式を選択し、プログラマよりシリアル通信にてコマンドを送信し、フラッシュ・メモリの書き換えを行います。その流れを図1-3のフロー・チャートに示します。

図1-3 プログラミング概要図



1.3.1 プログラミング・モードへの遷移

78K0/Lx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子(FLMD0)に規定の電圧を供給し、その後リセットを解除することにより、フラッシュ・メモリ・プログラミング・モードに遷移させることができます。

1.3.2 シリアル通信方式の選択

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

1. 3. 3 コマンドの送受信によるフラッシュ・メモリの操作

78K0/Lx3が内蔵するフラッシュ・メモリには、フラッシュ・メモリを書き換えるための機能が内蔵されており、表1 - 1に示すようなフラッシュ・メモリ操作機能が使用できます。

表1 - 1 フラッシュ・メモリ機能概要

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

これらの機能を制御するためにプログラマから78K0/Lx3に対しシリアル通信でコマンドを送信します。また、そのコマンドに対し78K0/Lx3から応答ステータスが返信されます。これら一連のシリアル通信のやりとりを繰り返すことにより、フラッシュ・メモリの書き換えを行います。

1. 4 78K0/Lx3製品固有情報

78K0/Lx3は、プログラマ側で製品固有情報（デバイス名、メモリ情報）を管理しておく必要があります。

表1 - 2に78K0/Lx3のフラッシュ・メモリ・サイズ、図1 - 4にフラッシュ・メモリ構成を示します。

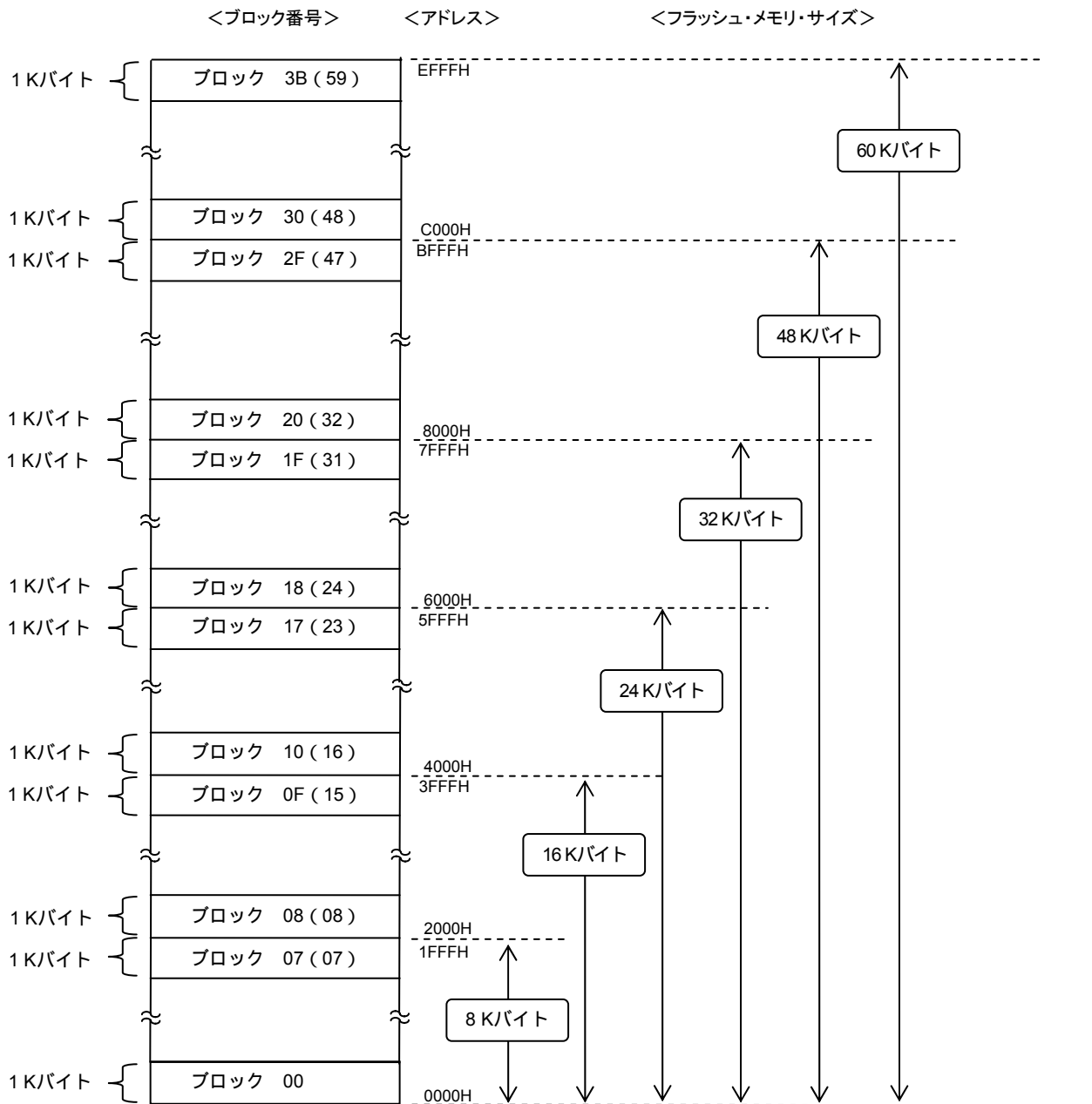
表1 - 2 78K0/Lx3のフラッシュ・メモリ・サイズ (1/2)

デバイス名		フラッシュ・メモリ・サイズ
78K0/LC3	μPD78F0400	8 KB
	μPD78F0401	16 KB
	μPD78F0402	24 KB
	μPD78F0403	32 KB
	μPD78F0410	8 KB
	μPD78F0411	16 KB
	μPD78F0412	24 KB
	μPD78F0413	32 KB
78K0/LD3	μPD78F0420	8 KB
	μPD78F0421	16 KB
	μPD78F0422	24 KB
	μPD78F0423	32 KB
	μPD78F0430	8 KB
	μPD78F0431	16 KB
	μPD78F0432	24 KB
	μPD78F0433	32 KB

表1 - 2 78K0/Lx3のフラッシュ・メモリ・サイズ (2/2)

デバイス名		フラッシュ・メモリ・サイズ
78K0/LE3	μPD78F0441	16 KB
	μPD78F0442	24 KB
	μPD78F0443	32 KB
	μPD78F0444	48 KB
	μPD78F0445	60 KB
	μPD78F0451	16 KB
	μPD78F0452	24 KB
	μPD78F0453	32 KB
	μPD78F0454	48 KB
	μPD78F0455	60 KB
	μPD78F0461	16 KB
	μPD78F0462	24 KB
	μPD78F0463	32 KB
	μPD78F0464	48 KB
	μPD78F0465	60 KB
78K0/LF3	μPD78F0471	16 KB
	μPD78F0472	24 KB
	μPD78F0473	32 KB
	μPD78F0474	48 KB
	μPD78F0475	60 KB
	μPD78F0481	16 KB
	μPD78F0482	24 KB
	μPD78F0483	32 KB
	μPD78F0484	48 KB
	μPD78F0485	60 KB
	μPD78F0491	16 KB
	μPD78F0492	24 KB
	μPD78F0493	32 KB
	μPD78F0494	48 KB
	μPD78F0495	60 KB

図1-4 フラッシュ・メモリ構成



備考 1ブロックは、すべて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）。

第2章 プログラマ動作環境

2.1 プログラマ制御端子

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

表2 - 1 端子説明

プログラマ			78K0/Lx3	ターゲットとの通信方式	
信号名	入出力	端子機能	端子名	CSI ^{注1}	UART
FLMD0	出力	書き込みモードへの選択レベル出力および通信方式選択パルス出力	FLMD0	○	○
V _{DD}	出力	V _{DD} 電圧生成 / 電圧監視	V _{DD} AV _{REF}		
GND	-	グランド	V _{SS} AV _{SS}	○	○
CLK	出力	78K0/Lx3への動作クロック出力	EXCLK	× ^{注2}	^{注3}
RESET	出力	プログラミング・モード切り替えトリガ	RESET	○	○
SO	出力	78K0/Lx3へのコマンド送信	SI10	○	×
SI	入力	78K0/Lx3からの応答ステータスおよび各種データの受信	SO10	○	×
SCK	出力	78K0/Lx3へのシリアル・クロック供給	SCK10	○	×
TxD	出力	78K0/Lx3へのコマンド送信	RxD6	×	○
RxD	入力	78K0/Lx3からの応答ステータスおよび各種データの受信	TxD6	×	○

注1. 78K0/LC3ではサポートされていません。

2. CSI10使用時は、高速内蔵発振クロック (f_{RH}) で動作します。

3. 高速内蔵発振クロック (f_{RH}) を使用したUART以外は、X1クロック (f_X) または外部メイン・システム・クロック (f_{EXCLK}) が必要です。

備考 ○：端子を使用します。

×：端子を使用しません。

：ユーザ・システム側で供給されている場合は接続の必要はありません。

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

2.2 各制御端子の詳細

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

FLMD0端子は78K0/Lx3の動作モードを制御するための端子です。FLMD0端子に規定の電圧を加えリセットを解除すると、78K0/Lx3はフラッシュ・メモリ・プログラミング・モードで動作します。

リセット解除後にFLMD0端子を V_{DD} とGND間の電圧で制御しパルスを出力することにより、プログラマと78K0/Lx3間のシリアル通信方式を確定します。FLMD0パルス数と通信方式の関係は表2 - 3を参照してください。

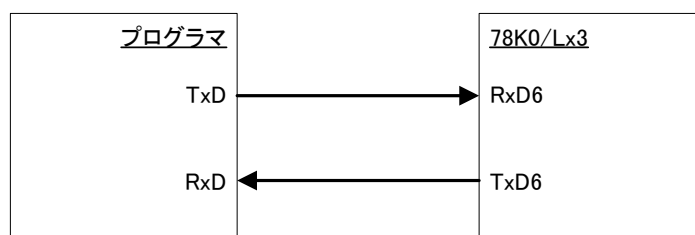
2.2.2 シリアル・インタフェース端子 (TxD, RxD, SI, SO, $\overline{\text{SCK}}$)

シリアル・インタフェース端子は、プログラマと78K0/Lx3間でフラッシュ・メモリ書き換えコマンドの受け渡しを行う端子です。

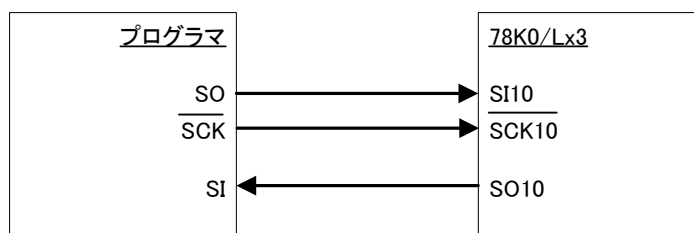
78K0/Lx3の場合、通信方式としてUART, CSIの2種類の中から選択できます。通信方式と使用端子の接続図を次に示します。

図2 - 1 シリアル・インタフェース端子

(1) UART通信方式



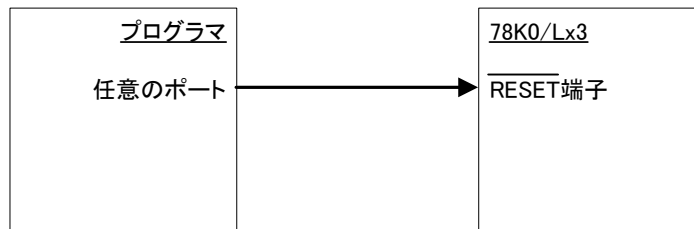
(2) 3線式シリアルI/O (CSI) 通信方式



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

リセット制御端子 ($\overline{\text{RESET}}$ 端子) は、プログラマから78K0/Lx3のシステム・リセットを制御するための端子です。FLMD0端子を規定の電圧に設定し、その後リセットを解除することにより、フラッシュ・メモリ・プログラミング・モードを選択できます。

図2 - 2 $\overline{\text{RESET}}$ 端子

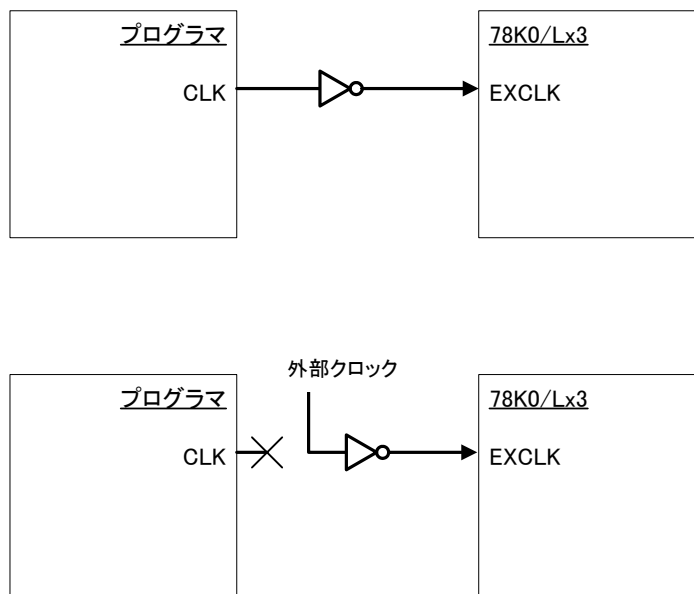


2.2.4 クロック制御端子 (CLK)

クロック制御端子は、プログラマが78K0/Lx3にクロックを供給する場合のみ使用します。78K0/Lx3の動作クロックをプログラマより供給する必要のないときは、CLK端子を接続する必要はありません。

(1) UART通信方式

図2 - 3 クロック端子



(2) 3線式シリアルI/O (CSI) 通信方式

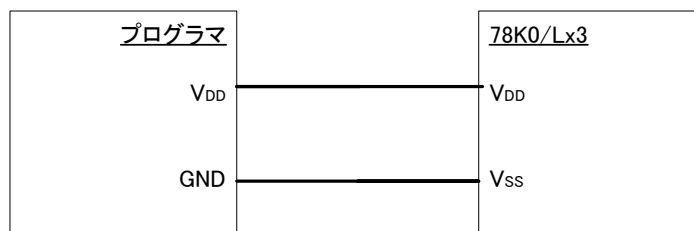
高速内蔵発振クロック (f_{RH}) で動作します。

2.2.5 V_{DD}, GND制御端子

V_{DD}制御端子は、プログラマから78K0/Lx3に電源を供給する場合に使用します。プログラマから78K0/Lx3に電源を供給する必要のないときは、V_{DD}制御端子を接続する必要はありません。ただし、専用プログラマは78K0/Lx3の電源状態の検出を行っているため、電源供給の有無にかかわらず接続する必要があります。

GND制御端子は、電源供給の有無に関係なく78K0/Lx3のV_{SS}と接続してください。

図2 - 4 V_{DD}/GND端子接続



2.2.6 その他の端子

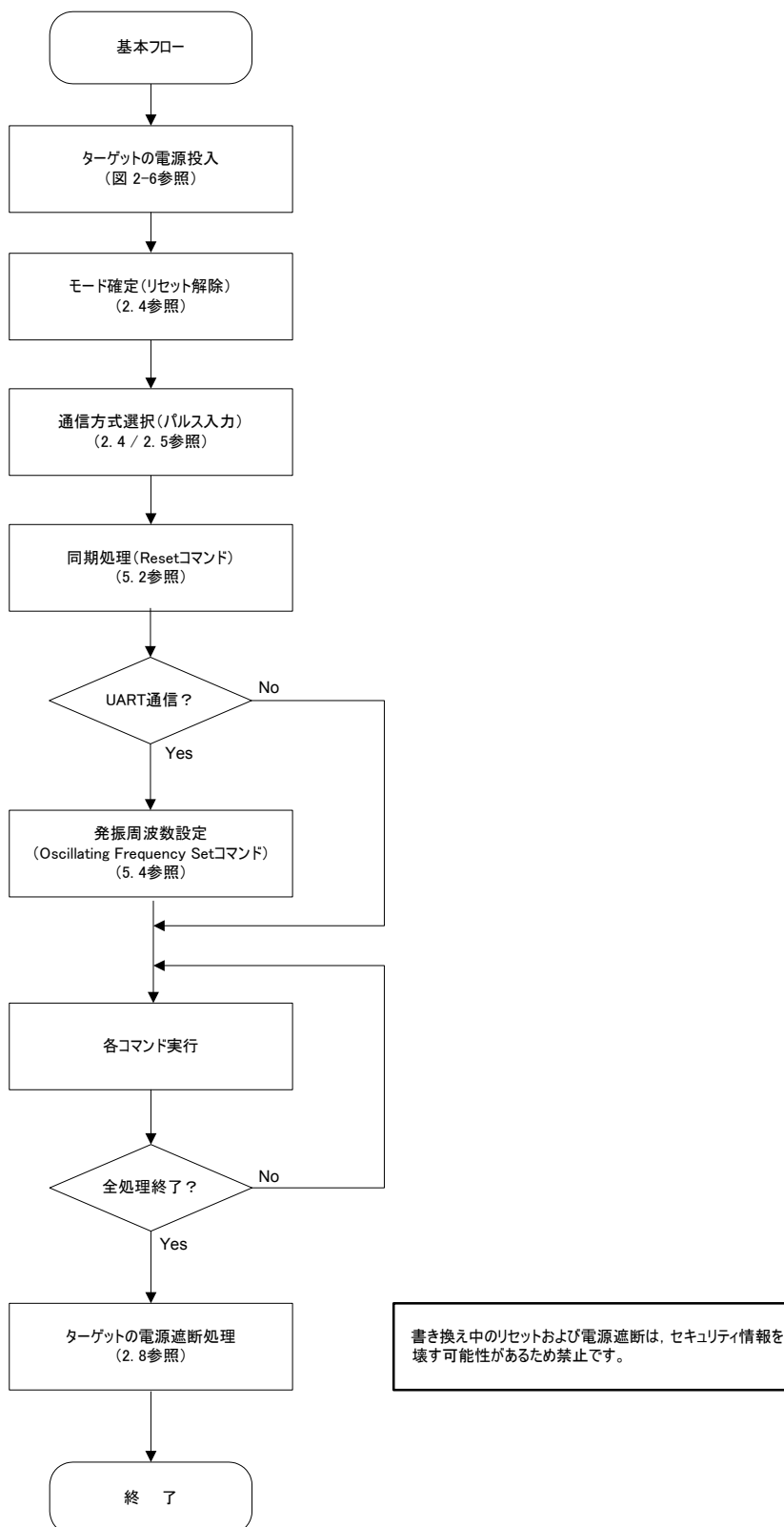
V_{DD}, V_{SS}以外の電源（AV_{REF}, AV_{SS}）も通常動作モード時と同じ電源を供給してください。

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

2.3 基本フロー・チャート

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

図2 - 5 フラッシュ処理の基本フロー・チャート



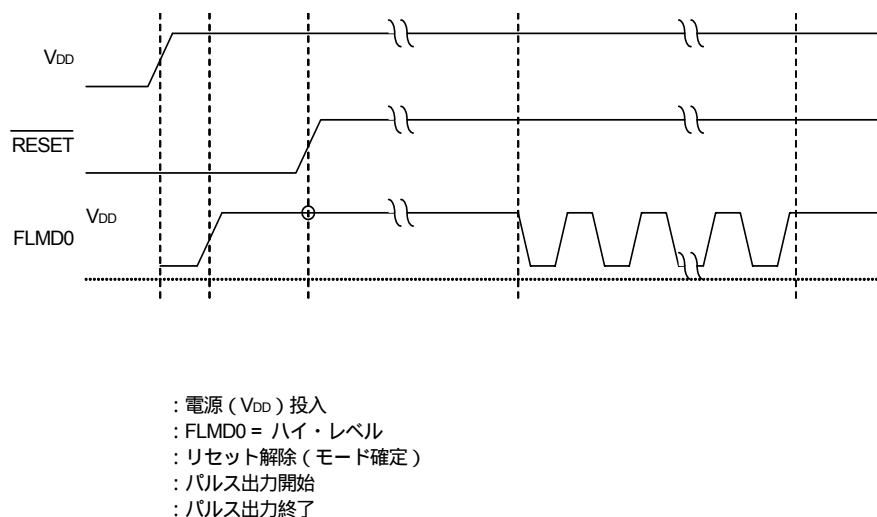
2.4 フラッシュ・メモリ・プログラミング・モードへの遷移方法

プログラマにてフラッシュ・メモリの書き換えを行うには、まず78K0/Lx3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/Lx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子(FLMD0)に規定の電圧を供給し、その後リセットを解除します。

フラッシュ・メモリ・プログラミング・モードへの遷移と通信方式の選択のタイミング図を次に示します。

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

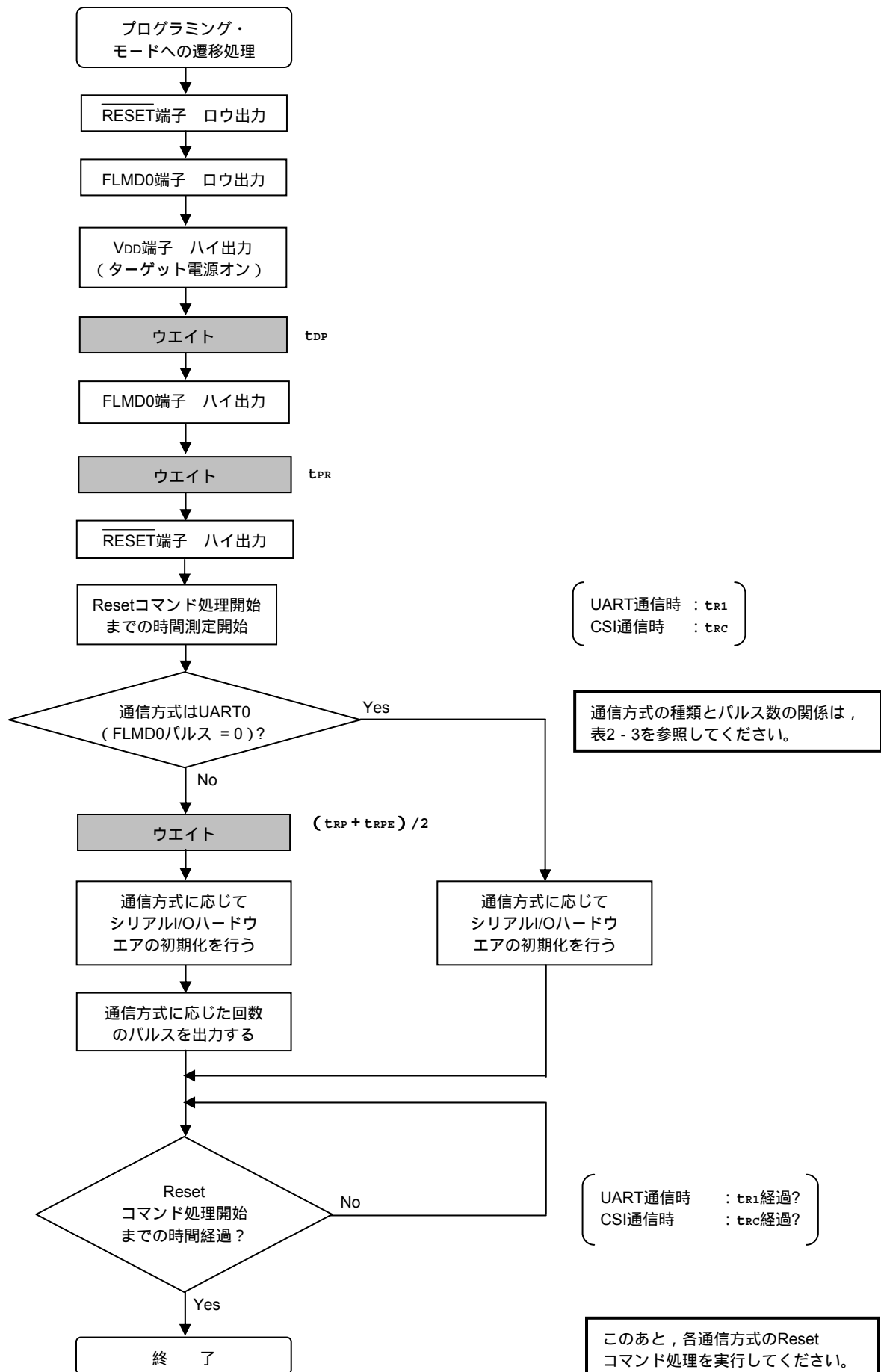


リセット解除時のFLMD0端子と動作モードの関係を次に示します。

表2 - 2 リセット時のFLMD0端子の設定と動作モード

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

2.4.1 モード引き込みのフロー・チャート



2.4.2 サンプル・プログラム

引き込み処理のサンプル・プログラムです。

```

/*****
/*
/* connect to Flash device
/*
/*
*****/
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){
        case FLIF_CSI:                pulse = PULSE_CSI;        break;
        case FLIF_UART:
            switch(UseTgCLK){
                case TGCLK_X1:        pulse = PULSE_UART;
break;
                case TGCLK_EXCLK:    pulse = PULSE_UART_EX;
break;
                case TGCLK_INT_OSC: pulse = PULSE_UART_INT_OSC;
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.

}
```

2.5 シリアル通信方式の選択

フラッシュ・メモリ・プログラミング・モードへの遷移のためのリセット解除後，78K0/Lx3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）にパルスを入力することで通信方式を決定します。

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

78K0/Lx3にて選択できる通信方式とFLMD0端子へのパルス数および使用するポートを次に示します。

表2 - 3 78K0/Lx3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 (P112), RxD6 (P113)
	3 (外部メイン・システム・クロック (f_{EXCLK}) 使用時)	
	5 (高速内蔵発振クロック (f_{RH}) 使用時)	
3 線式シリアル I/O (CSI10) 注	8	SO10 (P13), SI10 (P12), $\overline{SCK10}$ (P11)
設定禁止	その他	-

注 78K0/LC3ではサポートされていません。

2.6 UART通信方式

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

表2 - 4 UART通信の通信条件

項 目	内 容
ボー・レート	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/Lx3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。

2.7 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/Lx3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。

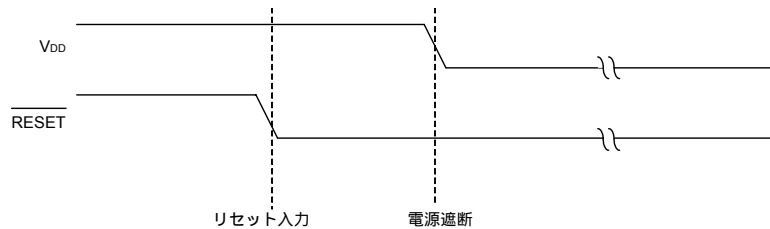
通信のデータ形式は8ビット単位のMSB先頭です。また、クロック周波数は2.5 MHz以下にしてください。

2.8 ターゲットの電源遮断処理

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

注意 コマンド処理中の電源遮断およびリセット入力は禁止です。

図2-8 フラッシュ・メモリ・プログラミングモードの終了手順



2.9 フラッシュ・メモリの操作方法

78K0/Lx3が内蔵するフラッシュ・メモリには、フラッシュ・メモリ書き換えのための機能が内蔵されており、表2-5に示すようなフラッシュ・メモリ操作機能があります。プログラマは、これらの機能を制御するコマンドを78K0/Lx3に送信し、78K0/Lx3からの応答ステータスを確認しながらフラッシュ・メモリを操作します。

表2-5 フラッシュ・メモリ操作機能概要一覧

分 類	機能名	概 要
消去	チップ消去	全フラッシュ・メモリを消去します。また、セキュリティ・フラグもクリアされます。
	ブロック消去	指定したブロックのフラッシュ・メモリを消去します。
書き込み	書き込み	指定したフラッシュ・メモリの領域にデータを書き込みます。
ベリファイ	ベリファイ	指定したフラッシュ・メモリのアドレスから取得したデータと、プログラマから送信されたデータを78K0/Lx3側で比較します。
ブランク・チェック	ブロック・ブランク・チェック	指定したフラッシュ・メモリの領域の消去状態を確認します。
情報取得	シリコン・シグネチャ取得	書き込みプロトコル情報を取得します。
	バージョン取得	78K0/Lx3 およびファームウェアのバージョンを取得します。
	ステータス取得	現在の動作状態を取得します。
	チェックサム取得	指定された領域のチェックサム・データを取得します。
セキュリティ	セキュリティ設定	セキュリティ情報を設定します。
その他	リセット	通信の同期検出に使用します。

2. 10 コマンド一覧

プログラマで使用されるコマンドの一覧と機能を次に示します。

表2 - 6 プログラマから78K0/Lx3への送信コマンド一覧

コマンド番号	コマンド名	機 能
70H	Status	現在の動作状況（ステータス・データ）を取得します。
00H	Reset	通信同期検出に使用します。
90H	Oscillating Frequency Set	フラッシュ・メモリ内蔵 78K0/Lx3 の発振周波数を指定します。
20H	Chip Erase	全フラッシュ・メモリを消去します。
22H	Block Erase	指定された領域のフラッシュ・メモリを消去します。
40H	Programming	フラッシュ・メモリの指定された領域にデータを書き込みます。
13H	Verify	フラッシュ・メモリの指定された領域の内容とプログラマから送信されたデータを比較します。
32H	Block Blank Check	指定されたブロックのフラッシュ・メモリの消去状態をチェックします。
C0H	Silicon Signature	78K0/Lx3 情報（品名，フラッシュ・メモリ構成など）を取得します。
C5H	Version Get	78K0/Lx3 バージョン，ファームウェア・バージョンを取得します。
B0H	Checksum	指定された領域のチェックサム・データを取得します。
A0H	Security Set	セキュリティ情報を設定します。

2. 11 ステータス一覧

プログラマが78K0/Lx3から受信するステータス・コードの一覧を次に示します。

表2 - 7 ステータス・コード一覧

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

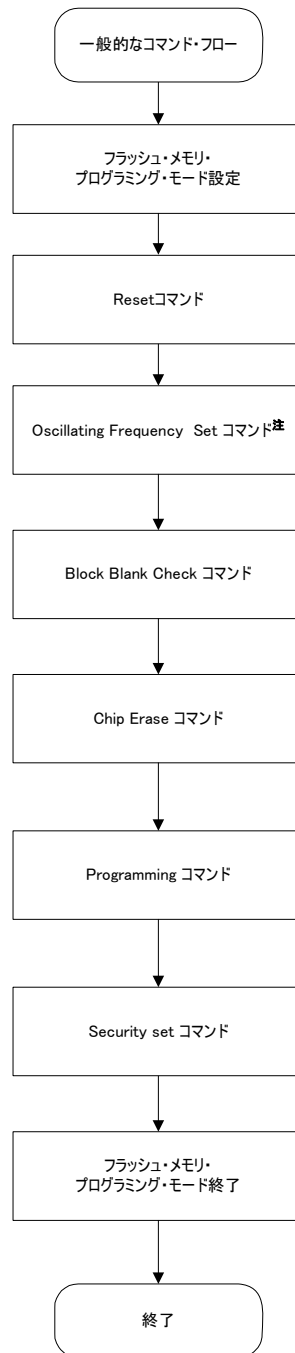
なお，このマニュアルではChecksum errorやNACKを受信した際は即時異常終了として扱っていますが，実際にプログラムを作る際は，Checksum errorやNACKが発生したコマンド送信直前のウエイトからリトライしても構いません。ただし，無限にリトライを繰り返さないようにリトライの回数制限を設けることを推奨します。

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/Lx3に対して電源遮断処理（2.8 ターゲットの電源遮断処理参照）を行ってから改めて接続することを推奨します。

第3章 プログラムの基本動作

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

図3 - 1 書き換え時の一般的なコマンド・フロー



注 78K0/Lx3では、CSI通信モードあるいは高速内蔵発振クロック（ f_{RH} ）を用いたUART通信モードを使用してフラッシュ・メモリの書き換えを行う場合、このコマンドの実行は必要ありません。

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

第4章 コマンド/データ・フレーム・フォーマット

プログラマと78K0/Lx3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/Lx3からプログラマに書き込みデータやベリファイ・データなどを送信する場合は、データ・フレームを使用します。これらのフレームには、転送データの信頼性を向上させるために、フレーム単位でヘッダ、フッタ、データ長情報、チェックサムを付けて送受信します。

次に両フレーム・フォーマットを示します。

図4 - 1 コマンド・フレームのフォーマット

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

図4 - 2 データ・フレームのフォーマット

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

表4 - 1 各フレームの記号説明

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

フレーム内のチェックサム (SUM) の計算例を次に示します。

【コマンド・フレームの場合】

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

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

この場合、チェックサム・データは次のように計算します。

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

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。

$$\begin{aligned} &00\text{H (初期値)} - 04\text{H (LEN)} - \text{FFH (D1)} - 80\text{H (D2)} - 40\text{H (D3)} - 22\text{H (D4)} \\ &= 1\text{BH (ボロー無視。下位8ビットのみ)} \end{aligned}$$

よって、このデータ・フレームは最終的に次のようになります。

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

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

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

↑ 本来なら 1BH

4.1 コマンド・フレーム送信処理

通信モードごとの各コマンド処理において、コマンド・フレームを送信する処理のフロー・チャートについては、次の節をお読みください。

- ・UART通信方式の場合は、6.1 コマンド・フレーム送信処理のフロー・チャートをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、7.1 コマンド・フレーム送信処理のフロー・チャートをお読みください。

4.2 データ・フレーム送信処理

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

通信モードごとの各コマンド処理において、データ・フレームを送信する処理のフロー・チャートについては、次の節をお読みください。

- ・UART通信方式の場合は、6.2 データ・フレーム送信処理のフロー・チャートをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、7.2 データ・フレーム送信処理のフロー・チャートをお読みください。

4.3 データ・フレーム受信処理

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

通信モードごとの各コマンド処理において、データ・フレームを受信する処理のフロー・チャートについては、次の節をお読みください。

- ・UART通信方式の場合は、6.3 データ・フレーム受信処理のフロー・チャートをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、7.3 データ・フレーム受信処理のフロー・チャートをお読みください。

第5章 コマンド処理説明

5.1 Statusコマンド

5.1.1 説明

書き込み / 消去などの各コマンド発行後の78K0/Lx3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/Lx3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/Lx3ではステータスの設定を行いません。よって、ステータス・フレームではなく、ビジー応答 (FFH) を受信する場合があります。この場合は、Statusコマンドをリトライしてください。

5.1.2 コマンド・フレームとステータス・フレーム

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

図5 - 1 Statusコマンド・フレーム (プログラマから78K0/Lx3へ)

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

図5 - 2 Statusコマンドに対するステータス・フレーム (78K0/Lx3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/Lx3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/Lx3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、7. 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/Lx3から自動的にステータス・フレームを返してきます。そのため、Statusコマンドは使用しません。

もしUART通信時にStatusコマンドを送信した場合はCommand Number Errorとなります。

5.2 Resetコマンド

5.2.1 説明

通信方式設定後に、プログラマと78K0/Lx3間の通信が確立されたことを確認します。

78K0/Lx3との通信方式にUART通信を選択した場合、プログラマと78K0/Lx3は同じボー・レートである必要がありますが、78K0/Lx3は自身のボー・レート生成クロック（fxまたはfEXCLK）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/Lx3はその“00H”のロウ幅を測定し2回の平均値を計算することで初めて自身のボー・レート生成クロック周波数を判別できます。それによって、ボー・レートの設定が可能になり同期検出が行えるようになります。

5.2.2 コマンド・フレームとステータス・フレーム

Resetコマンドのコマンド・フレームは図5 - 3、そのコマンドに対するステータス・フレームは図5 - 4のようになります。

図5 - 3 Resetコマンド・フレーム（プログラマから78K0/Lx3へ）

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

図5 - 4 Resetコマンドに対するステータス・フレーム（78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.4 Resetコマンドをお読みください。
- ・3線式シリアルI/O（CSI）通信方式の場合は、7.5 Resetコマンドをお読みください。

5.3 Baud Rate Setコマンド

78K0/Lx3は、Baud Rate Setコマンドをサポートしていません。

78K0/Lx3において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。

5. 4 Oscillating Frequency Setコマンド

5. 4. 1 説明

fxまたはfEXCLKで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/Lx3は、受信した周波数データを使用し、115200 bpsを生成します。

CSI通信あるいは高速内蔵発振クロック (f_{RH}) を用いたUART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(プログラムの仕様上、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)。

注意 78K0/Lx3において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。

5. 4. 2 コマンド・フレームとステータス・フレーム

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

図5 - 5 Oscillating Frequency Setコマンド・フレーム (プログラマから78K0/Lx3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はアンパッキング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 × 10⁴ = 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 × 10⁵ = 10000 kHz = 10 MHz

図5 - 6 Oscillating Frequency Setコマンドに対するステータス・フレーム (78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・ UART通信方式の場合は、6. 5 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、7. 6 Oscillating Frequency Set**コマンド**をお読みください。

5.5 Chip Eraseコマンド

5.5.1 説明

全フラッシュ・メモリの内容を消去します。また、チップ消去処理によりセキュリティ設定処理で設定されたすべての情報を初期化できます。ただし、セキュリティ設定により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（5.13 Security Setコマンド参照）。

5.5.2 コマンド・フレームとステータス・フレーム

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

図5-7 Chip Eraseコマンド・フレーム（プログラマから78K0/Lx3へ）

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

図5-8 Chip Eraseコマンドに対するステータス・フレーム（78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.6 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、7.7 Chip Eraseコマンドをお読みください。

5.6 Block Eraseコマンド

5.6.1 説明

指定したブロック番号のフラッシュ・メモリの内容を消去します。

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

ただし、セキュリティ設定により消去禁止となっている場合は消去できません（5.13 Security Setコマンド参照）。

5.6.2 コマンド・フレームとステータス・フレーム

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

図5-9 Block Eraseコマンド・フレーム（プログラマから78K0/Lx3へ）

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

備考 SAH - SAL : ブロック消去開始アドレス（任意のブロックの先頭アドレス）

SAH : 開始アドレスHigh（ビット23 - ビット16）

SAM : 開始アドレスMiddle（ビット15 - ビット8）

SAL : 開始アドレスLow（ビット7 - ビット0）

EAH - EAL : ブロック消去終了アドレス（任意のブロックの最終アドレス）

EAH : 最終アドレスHigh（ビット23 - ビット16）

EAM : 最終アドレスMiddle（ビット15 - ビット8）

EAL : 最終アドレスLow（ビット7 - ビット0）

図5-10 Block Eraseコマンドに対するステータス・フレーム（78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

・UART通信方式の場合は、6.7 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、7.8 Block Eraseコマンドをお読みください。

5.7 Programmingコマンド

5.7.1 説明

書き込み開始アドレス、書き込み終了アドレスを送信したあとに、書き込みバイト数分のデータを送信します。それにより、ユーザ・プログラムをフラッシュ・メモリに書きこみ、内部ペ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/Lx3のファームウェアは自動的に内部ペリファイを実行するので、さらにこの内部ペリファイに対するStatusコマンドの送信が必要となります。

5.7.2 コマンド・フレームとステータス・フレーム

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

図5 - 11 Programmingコマンド・フレーム（プログラマから78K0/Lx3へ）

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

備考 SAH - SAL : 書き込み開始アドレス

EAH - EAL : 書き込み終了アドレス

図5 - 12 Programmingコマンドに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1(a) : コマンド受信結果

5.7.3 データ・フレームとステータス・フレーム

書き込みを行うデータのデータ・フレームは図5 - 13、そのデータに対するステータス・フレームは図5 - 14のようになります。

図5 - 13 書き込みを行うデータ・フレーム（プログラマから78K0/Lx3へ）

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

備考 Write Data : 書き込むユーザ・プログラム

図5 - 14 データ・フレームに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1(b) : データ受信確認結果

ST2(b) : 書き込み結果

5.7.4 全データ転送完了とステータス・フレーム

全データ転送完了後のステータス・フレームは図5 - 15のようになります。

図5 - 15 全データ転送完了後のステータス・フレーム（78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.8 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、7.9 Programmingコマンドをお読みください。

5.8 Verifyコマンド

5.8.1 説明

指定したアドレス範囲のデータに対して、プログラマから送信したデータと78K0/Lx3から読み出したデータ（リード・レベル）を比較し、一致しているかを確認します。

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

5.8.2 コマンド・フレームとステータス・フレーム

Verifyコマンドのコマンド・フレームは図5 - 16，そのコマンドに対するステータス・フレームは図5 - 17のようになります。

図5 - 16 Verifyコマンド・フレーム（プログラマから78K0/Lx3へ）

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

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

図5 - 17 Verifyコマンドに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1(a) : コマンド受信結果

5.8.3 データ・フレームとステータス・フレーム

ベリファイを行うデータのデータ・フレームは図5 - 18，そのデータに対するステータス・フレームは図5 - 19のようになります。

図5 - 18 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/Lx3へ）

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

備考 Verify Data : ベリファイを行うユーザ・プログラム

図5 - 19 データ・フレームに対するステータス・フレーム（78K0/Lx3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 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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6. 9 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、7. 10 Verifyコマンドをお読みください。

5.9 Block Blank Checkコマンド

5.9.1 説明

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

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

5.9.2 コマンド・フレームとステータス・フレーム

Block Blank Checkコマンドのコマンド・フレームは図5 - 20、そのコマンドに対するステータス・フレームは図5 - 21のようになります。

図5 - 20 Block Blank Checkコマンド・フレーム（プログラマから78K0/Lx3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5 - 21 Block Blank Checkコマンドに対するステータス・フレーム（78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

・UART通信方式の場合は、6. 10 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、7. 11 Block Blank Checkコマンドをお読みください。

5. 10 Silicon Signatureコマンド

5. 10. 1 説明

デバイスの書き込みプロトコル情報（シリコン・シグネチャ）を読み出します。

たとえば，プログラマが78K0/Lx3と異なる書き込みプロトコルを同時にサポートする場合に，Silicon Signatureコマンドを実行し，2バイト目と3バイト目の値に従い，適切なプロトコルを選択します。

5. 10. 2 コマンド・フレームとステータス・フレーム

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

図5 - 22 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/Lx3へ）

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

図5 - 23 Silicon Signatureコマンドに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1 : コマンド受信結果

5. 10. 3 シリコン・シグネチャ・データ・フレーム

シリコン・シグネチャ・データのデータ・フレームは図5 - 24のようになります。

図5 - 24 シリコン・シグネチャ・データ・フレーム (78K0/Lx3からプログラマへ)

STX	LEN	Data								SUM	ETX
02H	n	VEN	MET	MSC	DEC	END	DEV	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
DEV : デバイス名 (μ PD番号)
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. 上記のうち、ブート・ブロック番号 (BOT) 以外は、下位7ビットをデータ本体、上位1ビットを奇数パリティとして使用します。次ページに例を示します。

表5 - 1 シリコン・シグネチャ・データの例 (μPD78F0482 (78K0/LF3) の場合)

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/Lx3 固定値)	1	7FH (01111111B)	7FH	あり
MSC	機能コード (78K0/Lx3 固定値)	1	04H (00000100B)	04H	あり
DEC	デバイス拡張コード (78K0/Lx3 固定値)	1	BCH (10111100B)	3CH	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)		
DEV	デバイス名	10	C4H (11000100B)	' D '	あり
			37H (00110111B)	' 7 '	
			38H (11010000B)	' 8 '	
			46H (01000110B)	' F '	
			B0H (11010000B)	' 0 '	
			34H (00110100B)	' 4 '	
			38H (00111000B)	' 8 '	
			32H (00110010B)	' 2 '	
			20H (00100000B)	' '	
			20H (00100000B)	' '	
SCF	セキュリティ・フラグ情報	1	任意	左欄に同じ	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)

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

```

奇数パリティ・ビットを最上位ビットに付加します

```

p0000001 p0111111 p1111111 (p = 奇数パリティ・ビット)
= 0000001 1011111 0111111
= 01 BF 7F

```

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。

下位バイトを先頭にします

7F BF 01

↓

01 BF 7F

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）

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

↓

00000001 01111111 11111111

↓

000 00000001 01111111 11111111

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/Lx3間の処理手順チャート，コマンド処理のフロー・チャート，サンプル・プログラムについては，次の節をお読みください。

- ・UART通信方式の場合は，6. 11 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，7. 12 Silicon Signatureコマンドをお読みください。

5. 10. 4 78K0/Lx3シリコン・シグネチャー一覧

表5-2 78K0/Lx3のシリコン・シグネチャ・データー一覧

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	BC
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1
デバイス名 (μPD)	78F0400/78F0401/78F0402/78F0403 78F0410/78F0411/78F0412/78F0413 78F0420/78F0421/78F0422/78F0423 78F0430/78F0431/78F0432/78F0433 78F0441/78F0442/78F0443/78F0444/78F0445 78F0451/78F0452/78F0453/78F0454/78F0455 78F0461/78F0462/78F0463/78F0464/78F0465 78F0471/78F0472/78F0473/78F0474/78F0475 78F0481/78F0482/78F0483/78F0484/78F0485 78F0491/78F0492/78F0493/78F0494/78F0495	10	注2
セキュリティ情報	セキュリティ情報	1	任意
ブート・ブロック番号	現在, 選択されているブート・クラスタの最終ブロック番号	1	03

注1. 内蔵フラッシュROMの最終アドレス・リスト

項 目	内 容	長さ (バイト)	データ (Hex)
内蔵フラッシュ ROM の最終アドレス	8 K バイト (1FFFH)	3	7FBF80
	16 K バイト (3FFFH)		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

(注2は次ページにあります。)

注2. デバイス名リストを次に示します。

デバイス名リスト (1/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0400	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B0 = '0' B0 = '0' 20 = ' ' 20 = ' '
	D78F0410		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 31 = '1' B0 = '0' 20 = ' ' 20 = ' '
	D78F0420		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 32 = '2' B0 = '0' 20 = ' ' 20 = ' '
	D78F0430		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B3 = '3' B0 = '0' 20 = ' ' 20 = ' '

デバイス名リスト (2/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0401	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B0 = '0' 31 = '1' 20 = ' ' 20 = ' '
	D78F0411		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 31 = '1' 31 = '1' 20 = ' ' 20 = ' '
	D78F0421		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 32 = '2' 31 = '1' 20 = ' ' 20 = ' '
	D78F0431		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B3 = '3' 31 = '1' 20 = ' ' 20 = ' '

デバイス名リスト (3/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0441	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 34 = '4' 31 = '1' 20 = '' 20 = ''
	D78F0451		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B5 = '5' 31 = '1' 20 = '' 20 = ''
	D78F0461		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B6 = '6' 31 = '1' 20 = '' 20 = ''
	D78F0471		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 37 = '7' 31 = '1' 20 = '' 20 = ''

デバイス名リスト (4/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0481	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 38 = '8' 31 = '1' 20 = '' 20 = ''
	D78F0491		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B9 = '9' 31 = '1' 20 = '' 20 = ''
	D78F0402		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B0 = '0' 32 = '2' 20 = '' 20 = ''
	D78F0412		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 31 = '1' 32 = '2' 20 = '' 20 = ''

デバイス名リスト (5/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0422	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 32 = '2' 32 = '2' 20 = ' ' 20 = ' '
	D78F0432		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B3 = '3' 32 = '2' 20 = ' ' 20 = ' '
	D78F0442		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 34 = '4' 32 = '2' 20 = ' ' 20 = ' '
	D78F0452		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B5 = '5' 32 = '2' 20 = ' ' 20 = ' '

デバイス名リスト (6/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0462	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B6 = '6' 32 = '2' 20 = ' ' 20 = ' '
	D78F0472		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 37 = '7' 32 = '2' 20 = ' ' 20 = ' '
	D78F0482		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 38 = '8' 32 = '2' 20 = ' ' 20 = ' '
	D78F0492		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B9 = '9' 32 = '2' 20 = ' ' 20 = ' '

デバイス名リスト (7/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0403	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B0 = '0' B3 = '3' 20 = '' 20 = ''
	D78F0413		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 31 = '1' B3 = '3' 20 = '' 20 = ''
	D78F0423		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 32 = '2' B3 = '3' 20 = '' 20 = ''
	D78F0433		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B3 = '3' B3 = '3' 20 = '' 20 = ''

デバイス名リスト (8/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0443	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 34 = '4' B3 = '3' 20 = '' 20 = ''
	D78F0453		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B5 = '5' B3 = '3' 20 = '' 20 = ''
	D78F0463		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B6 = '6' B3 = '3' 20 = '' 20 = ''
	D78F0473		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 37 = '7' B3 = '3' 20 = '' 20 = ''

デバイス名リスト (9/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0483	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 38 = '8' B3 = '3' 20 = '' 20 = ''
	D78F0493		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B9 = '9' B3 = '3' 20 = '' 20 = ''
	D78F0444		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 34 = '4' 34 = '4' 20 = '' 20 = ''
	D78F0454		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B5 = '5' 34 = '4' 20 = '' 20 = ''

デバイス名リスト (10/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0464	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B6 = '6' 34 = '4' 20 = '' 20 = ''
	D78F0474		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 37 = '7' 34 = '4' 20 = '' 20 = ''
	D78F0484		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 38 = '8' 34 = '4' 20 = '' 20 = ''
	D78F0494		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B9 = '9' 34 = '4' 20 = '' 20 = ''

デバイス名リスト (11/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0445	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 34 = '4' B5 = '5' 20 = '' 20 = ''
	D78F0455		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B5 = '5' B5 = '5' 20 = '' 20 = ''
	D78F0465		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B6 = '6' B5 = '5' 20 = '' 20 = ''
	D78F0475		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 37 = '7' B5 = '5' 20 = '' 20 = ''

デバイス名リスト (12/12)

項 目	内 容	長さ (バイト)	実際の値
デバイス名	D78F0485	10	C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' 38 = '8' B5 = '5' 20 = '' 20 = ''
	D78F0495		C4 = 'D' 37 = '7' 38 = '8' 46 = 'F' B0 = '0' 34 = '4' B9 = '9' B5 = '5' 20 = '' 20 = ''

5. 11 Version Getコマンド

5. 11. 1 説明

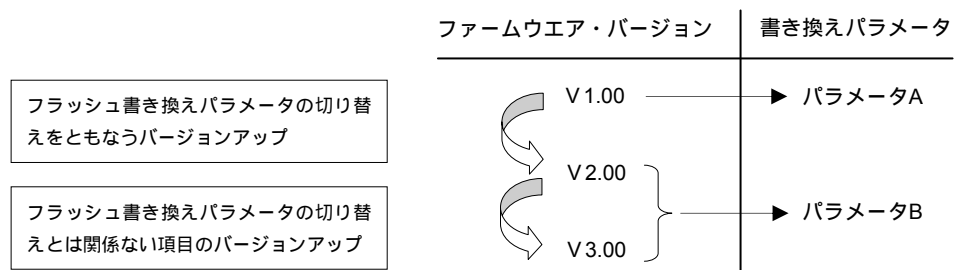
78K0/Lx3のデバイス・バージョン，ファームウェア・バージョン情報を取得します。

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/Lx3のファームウェア・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ェア改版時も，ファームウェア・バージョンが更新される場合があります（このとき，ファームウェア・バージョン更新の通知は行いません）。

例 ファームウェア・バージョンと書き換えパラメータ



5. 11. 2 コマンド・フレームとステータス・フレーム

Version Getコマンドのコマンド・フレームは図5 - 26，そのコマンドに対するステータス・フレームは図5 - 27のようになります。

図5 - 26 Version Getコマンド・フレーム（プログラマから78K0/Lx3へ）

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

図5 - 27 Version Getコマンドに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1 : コマンド受信結果

5. 11. 3 バージョン・データ・フレーム

バージョン・データのデータ・フレームは図5 - 28のようになります。

図5 - 28 バージョン・データ・フレーム (78K0/Lx3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 : ファームウェア・バージョン整数値
 FV2 : ファームウェア・バージョン小数点第一位
 FV3 : ファームウェア・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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6. 12 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、7. 13 Version Getコマンドをお読みください。

5. 12 Checksumコマンド

5. 12. 1 説明

指定された領域のデータのチェックサム・データを取得します。

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

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

5. 12. 2 コマンド・フレームとステータス・フレーム

Checksumコマンドのコマンド・フレームは図5 - 29、そのコマンドに対するステータス・フレームは図5 - 30のようになります。

図5 - 29 Checksumコマンド・フレーム (プログラマから78K0/Lx3へ)

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

備考 SAH-SAL : チェックサム計算開始アドレス

EAH-EAL : チェックサム計算終了アドレス

図5 - 30 Checksumコマンドに対するステータス・フレーム (78K0/Lx3からプログラマへ)

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

備考 ST1 : コマンド受信結果

5. 12. 3 チェックサム・データ・フレーム

チェックサム・データのデータ・フレームは図5 - 31のようになります。

図5 - 31 チェックサム・データ・フレーム (78K0/Lx3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/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

・UART通信方式の場合は、6. 13 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、7. 14 Checksumコマンドをお読みください。

5. 13 Security Setコマンド

5. 13. 1 説明

セキュリティに関する設定（書き込み，ブロック消去，チップ消去，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・ブロック書き換え禁止の設定をした場合，チップ消去自体が不可能になり，プログラマからは消去ができなくなります。プログラマの仕様としては，チップ消去禁止の設定を行う前に，設定実行の再確認をすることを推奨します。

5. 13. 2 コマンド・フレームとステータス・フレーム

Security Setコマンドのコマンド・フレームは図5 - 32，そのコマンドに対するステータス・フレームは図5 - 33のようになります。

図5 - 32 Security Setコマンド・フレーム（プログラマから78K0/Lx3へ）

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

図5 - 33 Security Setコマンドに対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1(a) : コマンド受信結果

5. 13. 3 データ・フレームとステータス・フレーム

セキュリティ・データのデータ・フレームは図5 - 34，そのデータに対するステータス・フレームは図5 - 35のようになります。

図5 - 34 セキュリティ・データ・フレーム（プログラマから78K0/Lx3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・クラスタの最終ブロック番号（03H固定）

図5 - 35 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/Lx3からプログラマへ）

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

備考 ST1(b) : セキュリティ・データ書き込み結果

5. 13. 4 内部ペリファイ確認とステータス・フレーム

内部ペリファイ確認に対するステータス・フレームは図5 - 36のようになります。

図5 - 36 内部ペリファイ確認に対するステータス・フレーム（78K0/Lx3からプログラマへ）

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

備考 ST1(c) : 内部ペリファイ結果

セキュリティ・フラグ・フィールドの内容を次に示します。

表5 - 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・ブロック書き換え禁止フラグ（1：許可，0：禁止）
ビット3	1 固定
ビット2	書き込み禁止フラグ（1：書き込み許可，0：書き込み禁止）
ビット1	ブロック消去禁止フラグ（1：ブロック消去許可，0：ブロック消去禁止）
ビット0	チップ消去禁止フラグ（1：チップ消去許可，0：チップ消去禁止）

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

表5 - 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・ブロック書き 換え禁止フラグ		×		フラッシュ・メモリ・プログラミング・モード (オンボード / オフボード・プログラミング) 時と同様

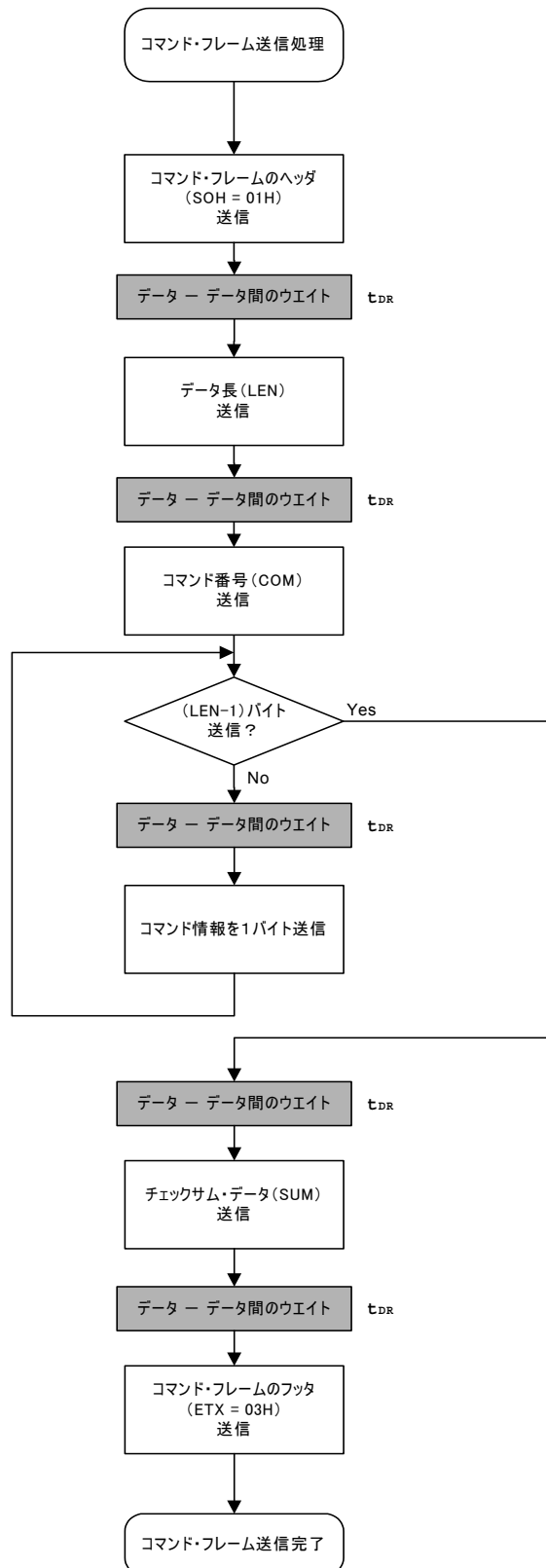
通信方式ごとの、プログラマと78K0/Lx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6. 14 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、7. 15 Security Setコマンドをお読みください。

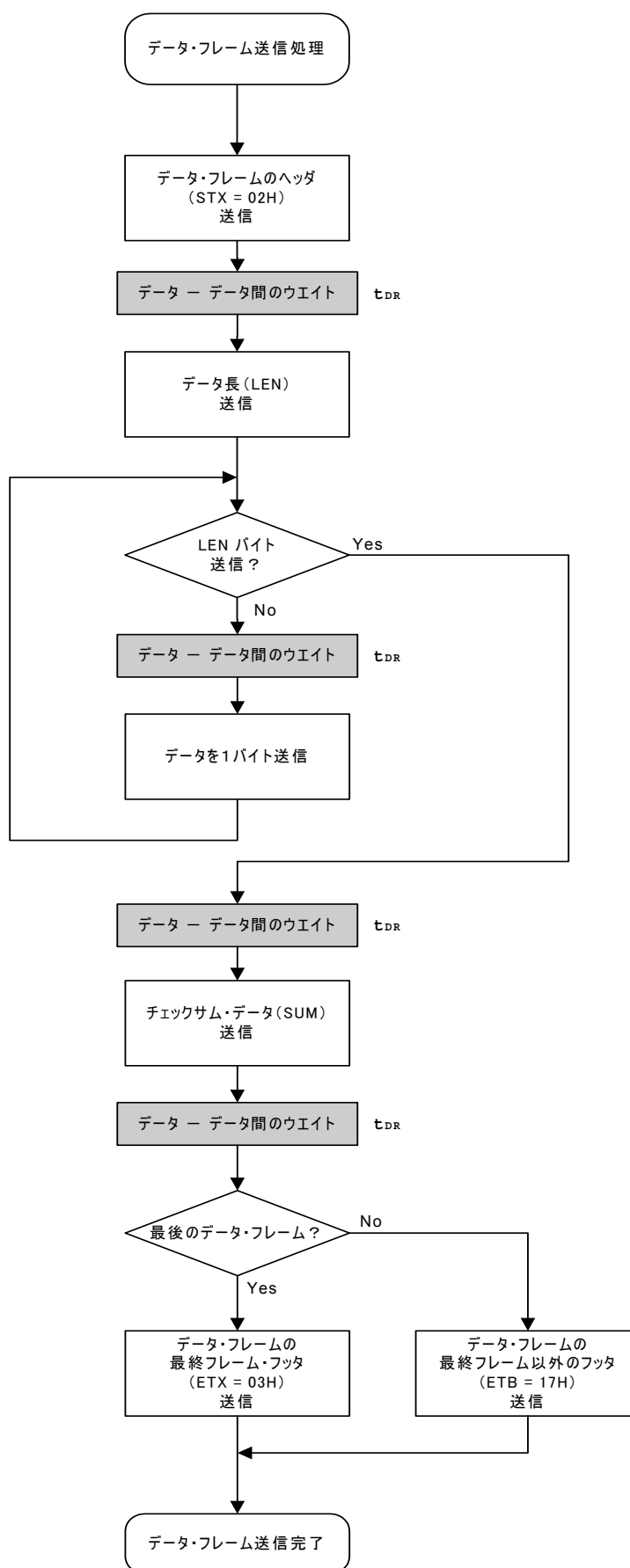
第6章 UART通信方式

この章のフロー・チャートで示した略号（ t_{xx} および $t_{W\text{TX}x}$ ）は、第8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8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。

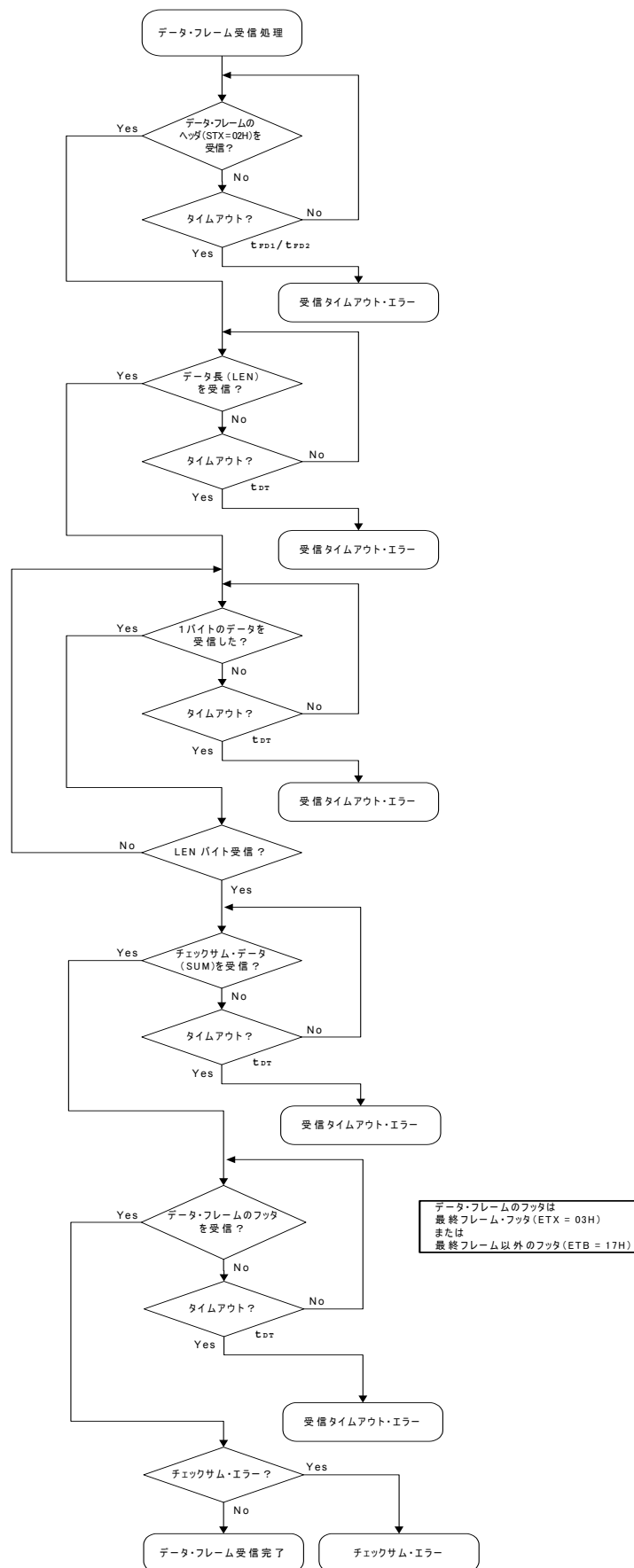
6.1 コマンド・フレーム送信処理のフロー・チャート



6.2 データ・フレーム送信処理のフロー・チャート



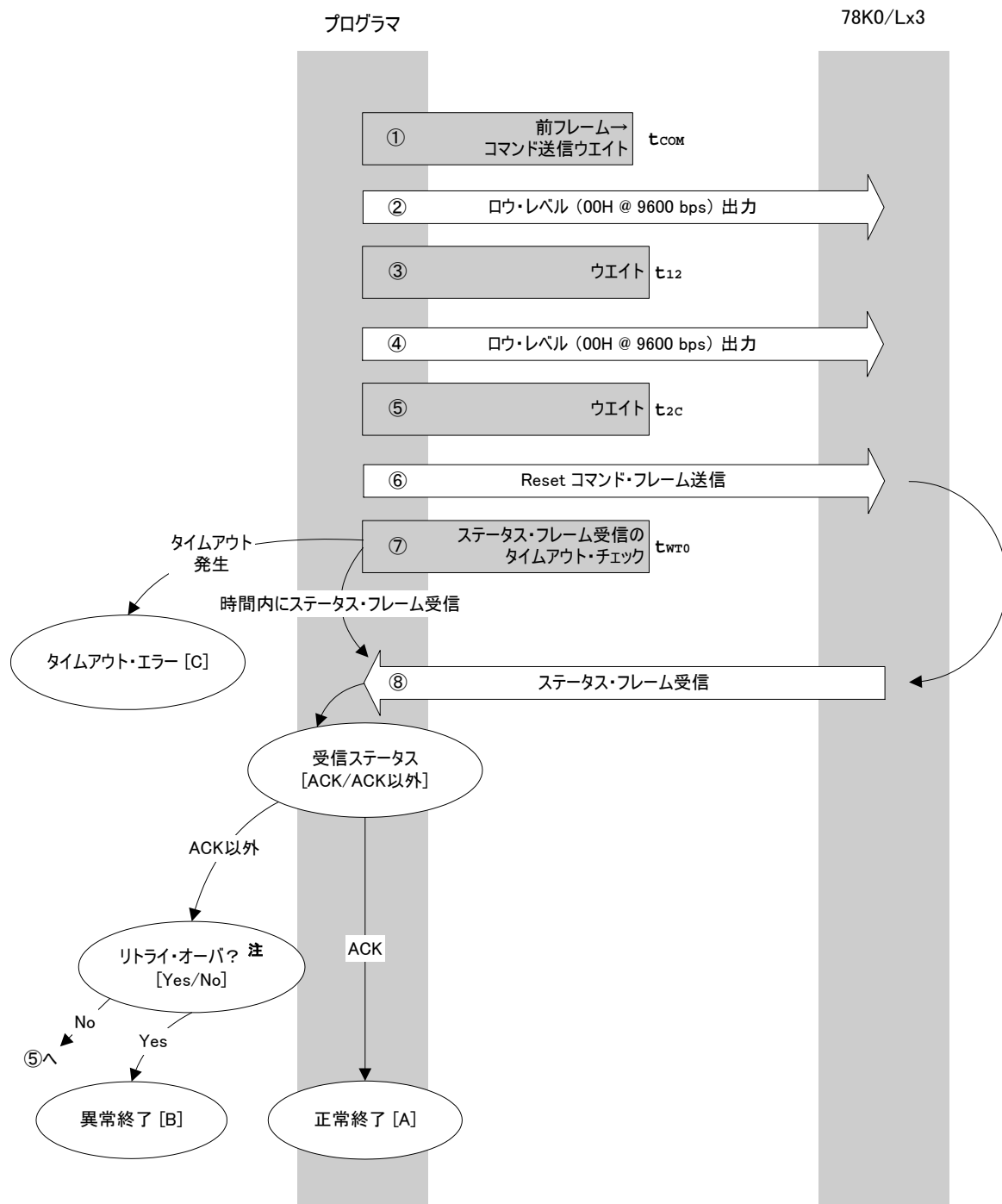
6.3 データ・フレーム受信処理のフロー・チャート



6.4 Resetコマンド

6.4.1 処理手順チャート

Resetコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

6.4.2 処理手順説明

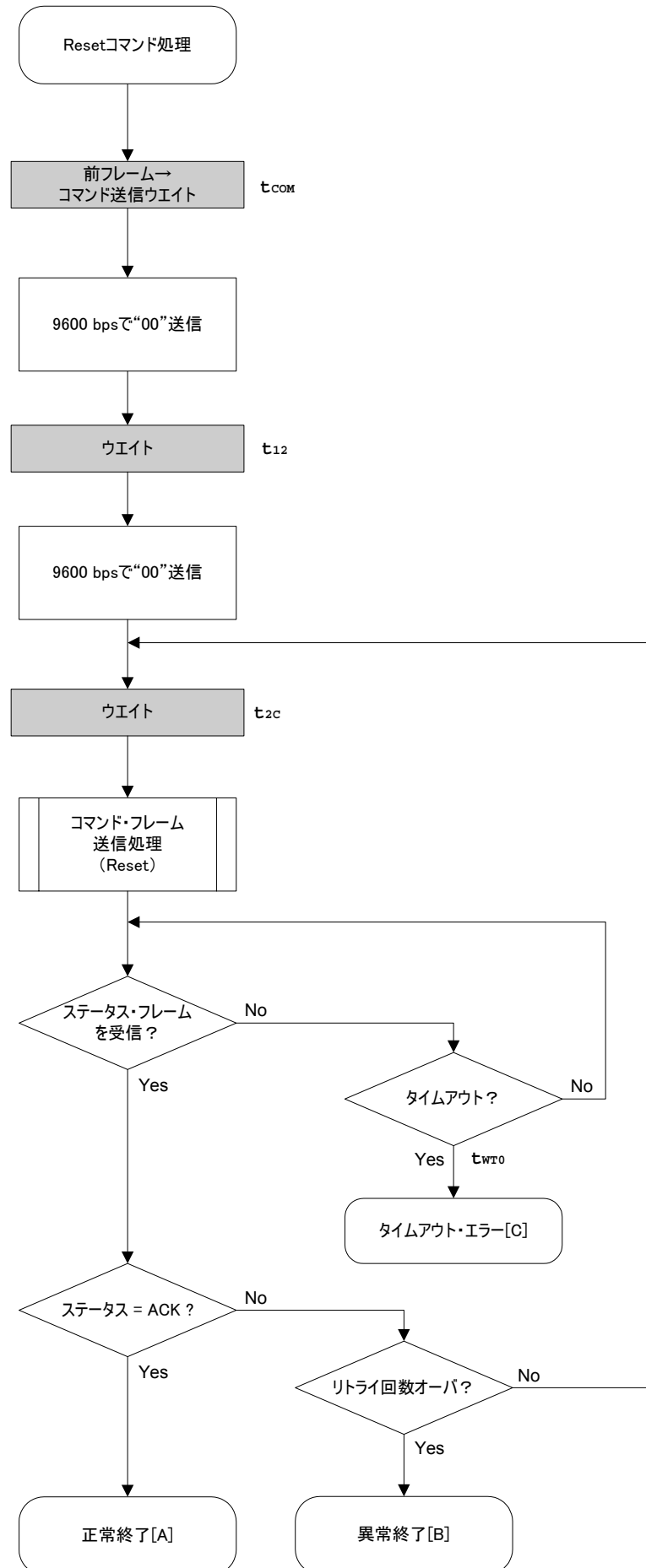
直前のフレームからコマンド処理開始前のウェイトをします（ウェイト時間 t_{COM} ）。
 ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
 ウェイトします（ウェイト時間 t_{12} ）。
 ロウ・レベル出力を行います（データ00Hを9600 bpsで送信）。
 ウェイトします（ウェイト時間 t_{2c} ）。
 コマンド・フレーム送信処理にて **Resetコマンド** を送信します。
 コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は **タイムアウト・エラー [C]** となります（タイムアウト時間 t_{WT0} ）。
 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 からやり直します。
 リトライ・オーバであれば **異常終了 [B]** です。

6.4.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され、プログラマと 78K0/Lx3 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）。または，処理中にステータス・コマンド以外を受信しました。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームが受信できませんでした。

6.4.4 フロー・チャート



6.4.5 サンプル・プログラム

Resetコマンド処理のサンプル・プログラムです。

```

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

    set_uart0_br(BR_9600); // change to 9600bps

    //fl_wait(3000000);

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

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

    // set_uart0_br(flbaud); // restore baud-rate

    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_DFEO_ERR)          // t.o. ?
            break;                        // yes // case [C]

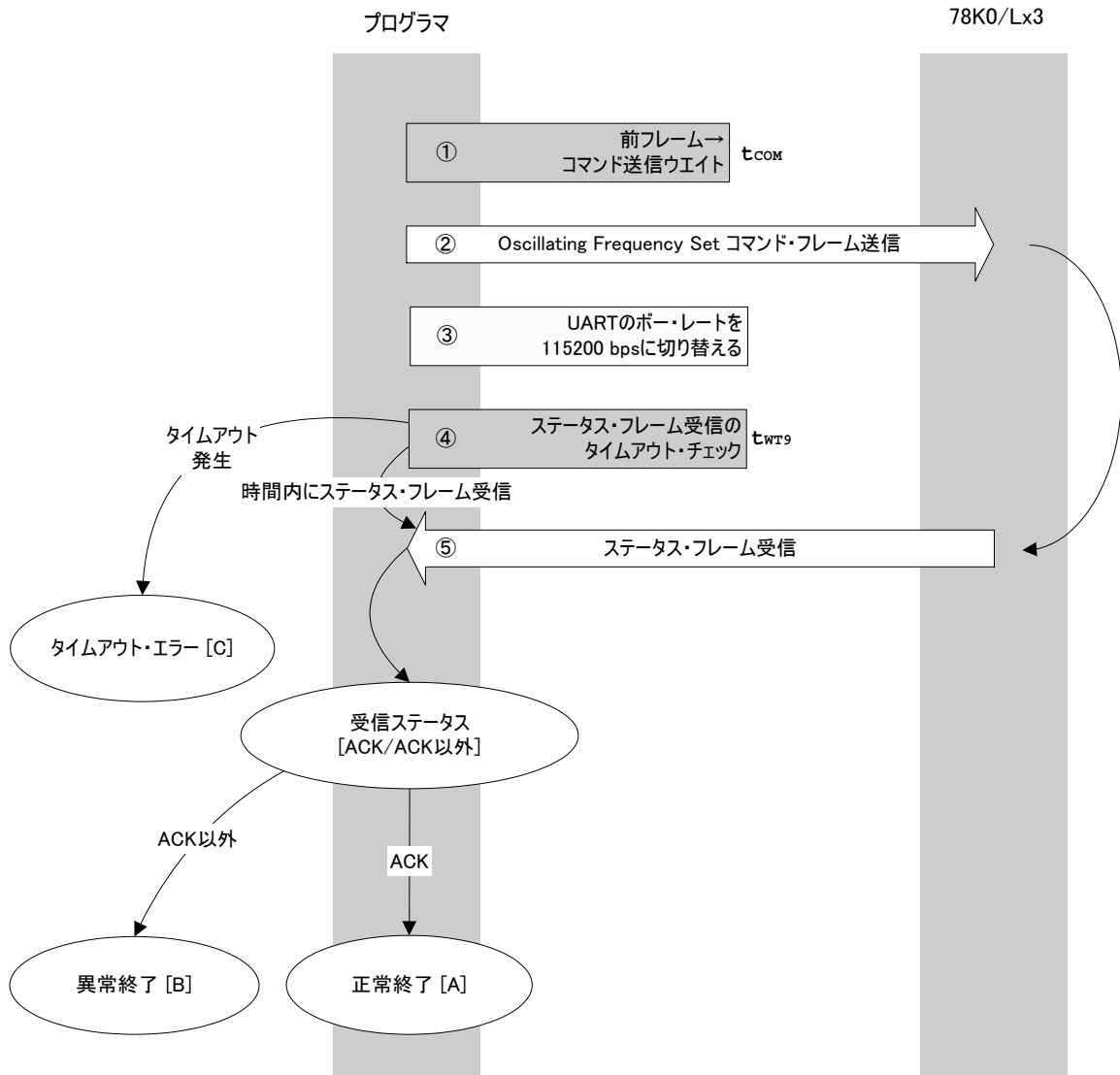
        if (rc == FLC_ACK){              // ACK ?
            break;                        // yes // case [A]
        }
        else{
            NOP();
        }
        //continue;                      // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //      case   FLC_NO_ERR:      return rc;      break; // case [A]
    //      case   FLC_DFEO_ERR:    return rc;      break; // case [C]
    //      default:                return rc;      break; // case [B]
    // }
    return rc;
}

```

6.5 Oscillating Frequency Setコマンド

6.5.1 処理手順チャート

Oscillating Frequency Setコマンド処理手順



6.5.2 処理手順説明

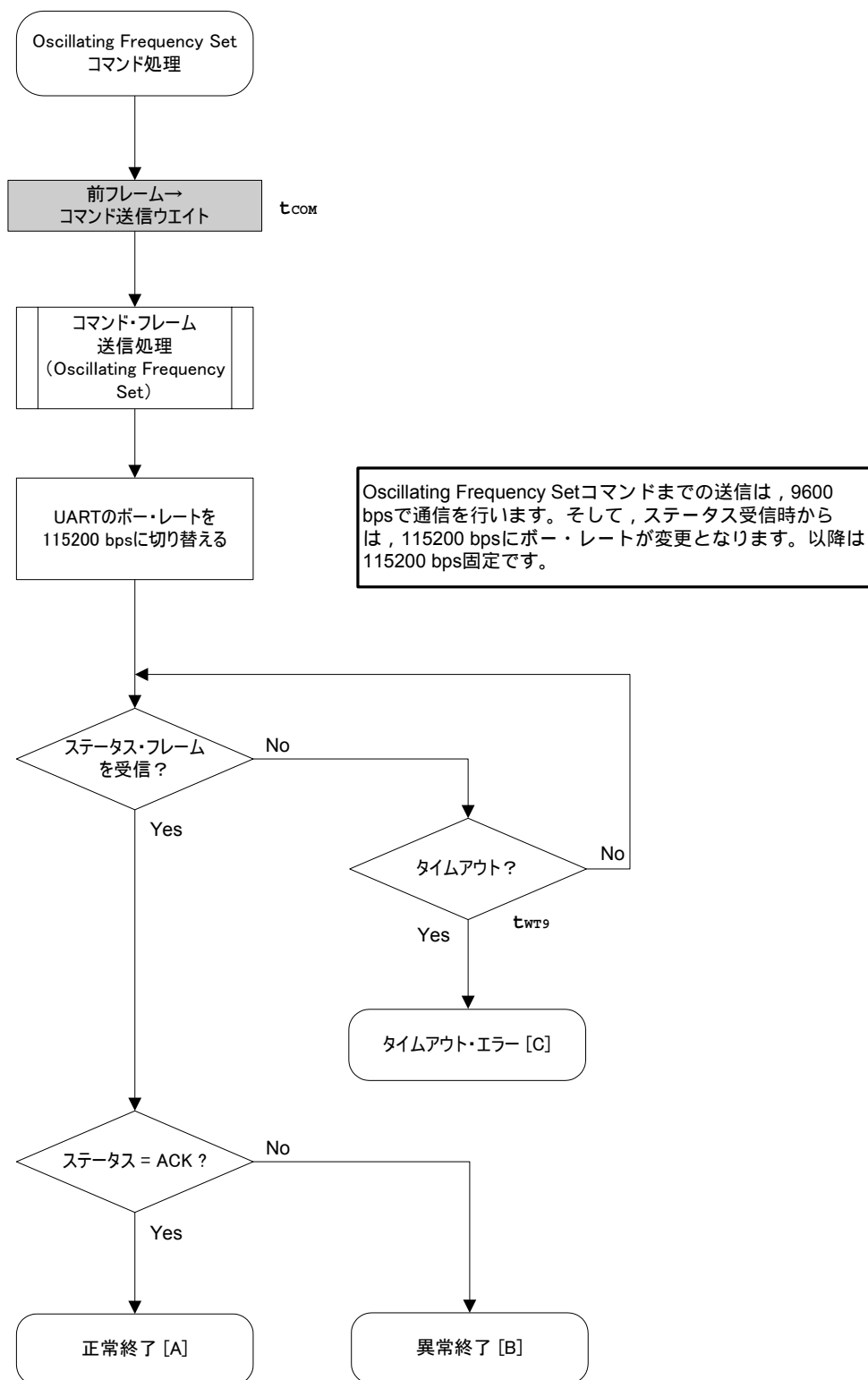
直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 コマンド・フレーム送信処理により、Oscillating Frequency Setコマンド^{*}を送信します。
 ステータス受信時から、UART通信のボー・レートを115200 bpsに切り替えます。以降は、115200 bps固定です。
 コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は、タイムアウト・エラー[C]となります（タイムアウト時間 t_{WT9} ）。ステータス・コードをチェックします。

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

6.5.3 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/Lx3 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）。または、処理中にステータス・コマンド以外を受信しました。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

6.5.4 フロー・チャート



6.5.5 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_UA);          // 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]
    // }

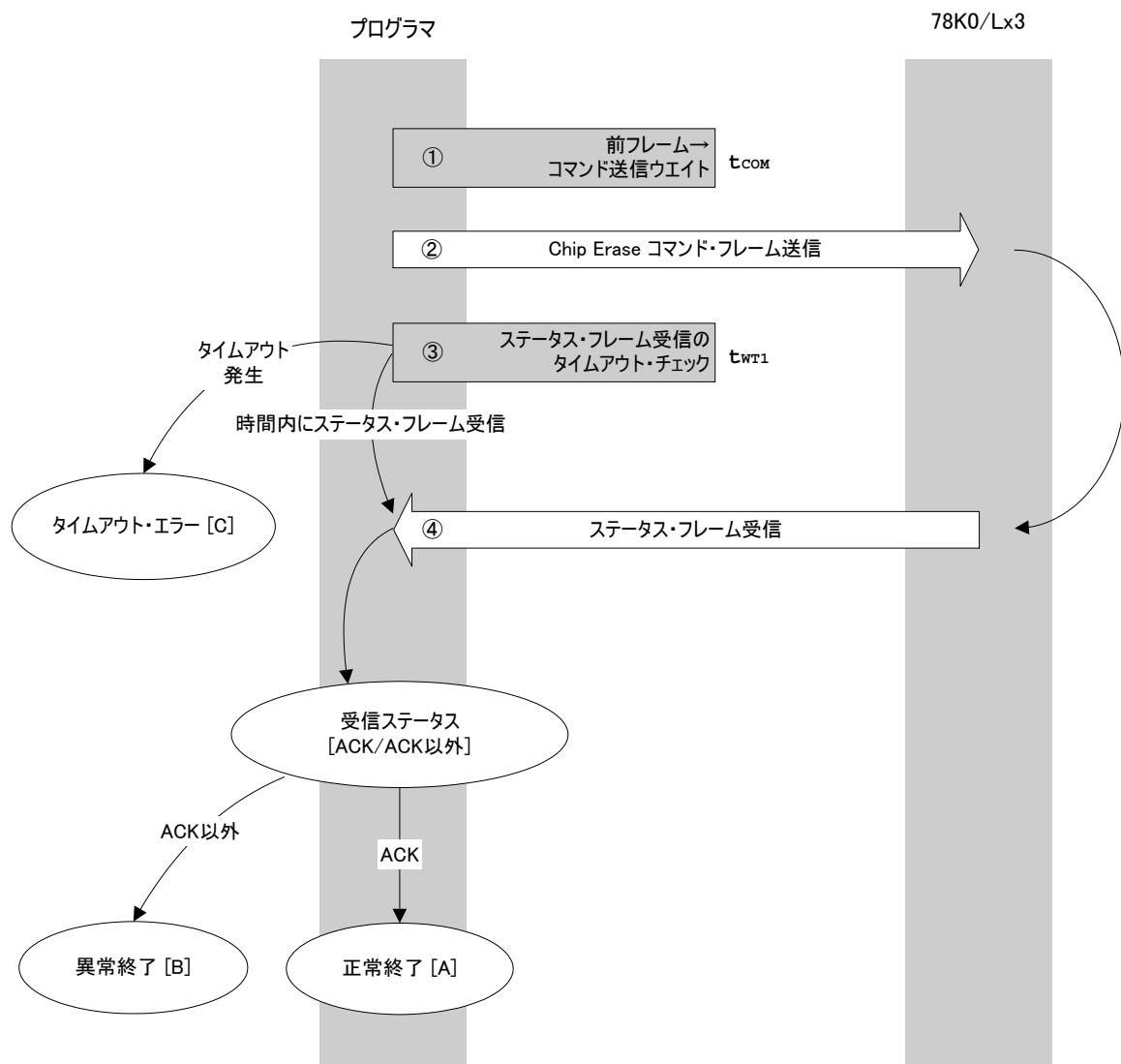
    return rc;
}

```

6.6 Chip Eraseコマンド

6.6.1 処理手順チャート

Chip Eraseコマンド処理手順



6.6.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

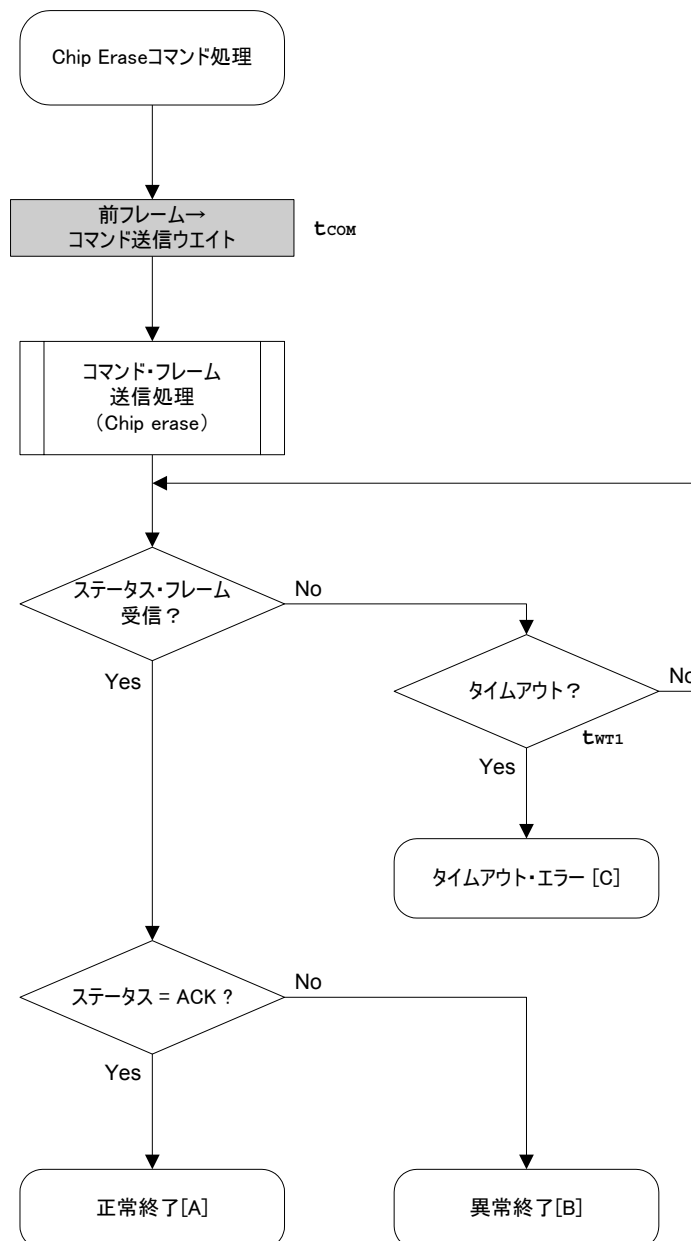
ST1 = ACKの場合 : **正常終了[A]** です。

ST1 = ACK以外の場合 : **異常終了[B]** です。

6.6.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され、チップ消去が正常に実行されたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、「チップ消去禁止」、「ブート・ブロック書き換え禁止」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）。または，処理中にステータス・コマンド以外を受信しました。
	MRG10 エラー	1AH	消去エラーが発生しました。
	MRG11 エラー	1BH	
	Write エラー	1CH	
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

6.6.4 フロー・チャート



6.6.5 サンプル・プログラム

Chip Eraseコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM_UA);          // wait before sending command

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

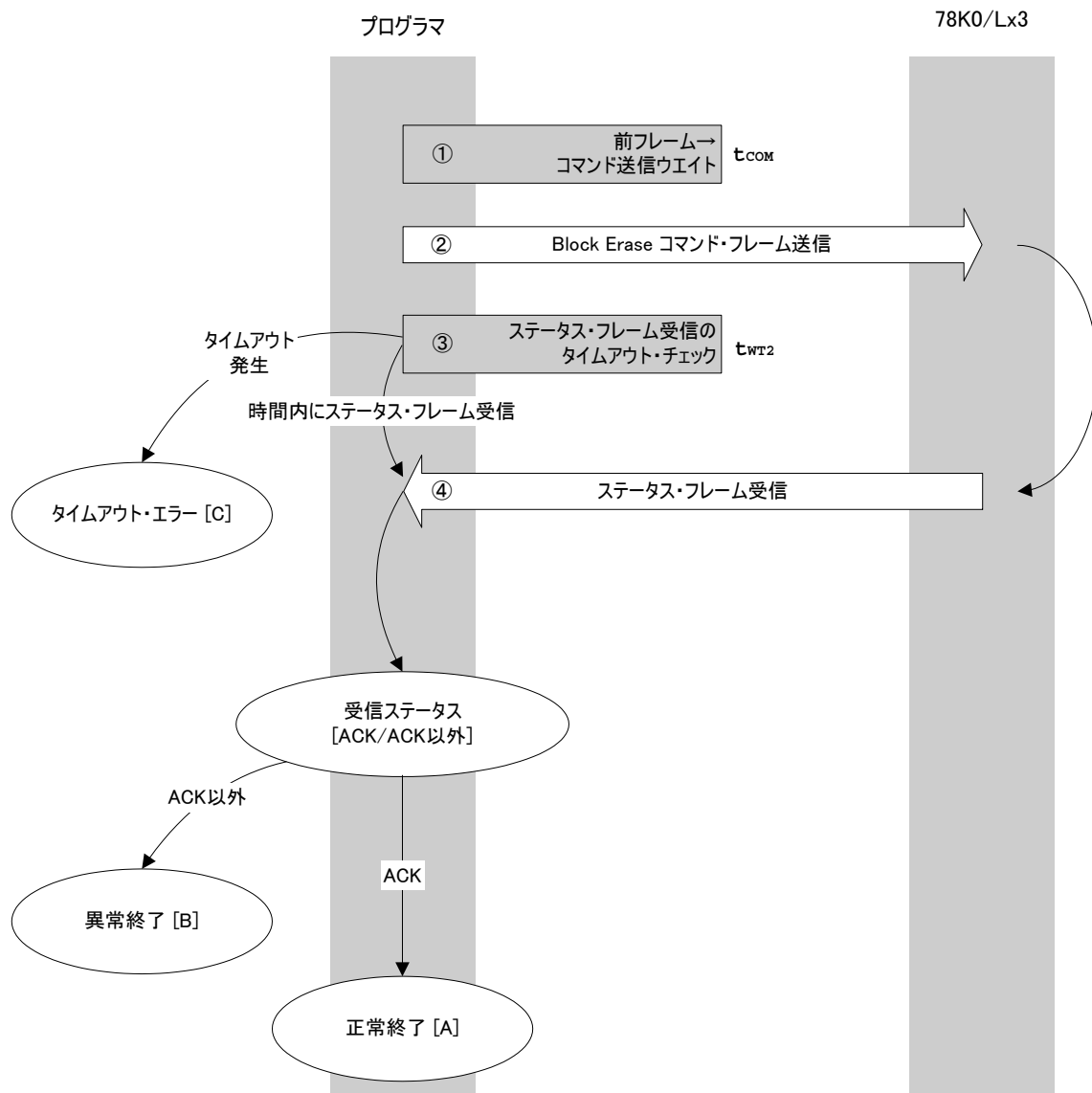
    rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX); // get status frame
    // switch(rc) {
    //
    //     case    FLC_NO_ERR:      return rc;      break; // case [A]
    //     case    FLC_DFTO_ERR:    return rc;      break; // case [C]
    //     default:                  return rc;      break; // case [B]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

```

6.7 Block Eraseコマンド

6.7.1 処理手順チャート

Block Eraseコマンド処理手順



6.7.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

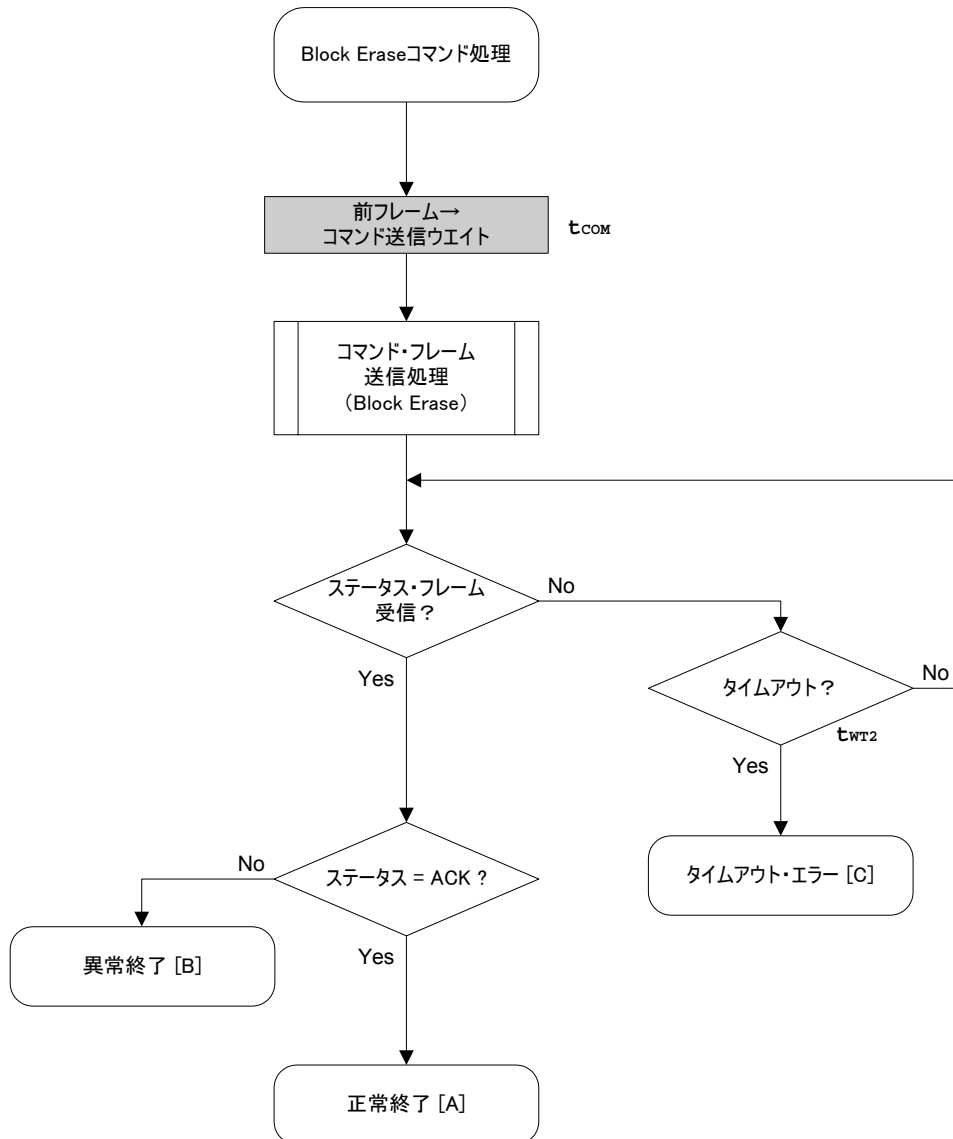
ST1 = ACKの場合 : **正常終了[A]** です。

ST1 = ACK以外の場合 : **異常終了[B]** です。

6.7.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ブロック消去が正常に実行されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。
	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、「書き込み禁止」, 「ブロック消去禁止」, 「チップ消去禁止」のいずれかになっています。または、指定範囲にブート領域が含まれており、セキュリティ設定で、「ブート・ブロック書き換え禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（データ長 (LEN) 不正, ETX なしなど）。
	消去エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

6.7.4 フロー・チャート



6.7.5 サンプル・プログラム

Block Eraseコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Erase block command
/*
/*****
/* [i] 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_UA);              // 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, tWT2_MAX); // get status frame
    rc = get_sfrm_ua(fl_ua_sfrm, wt2_max); // get status frame

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

    return rc;

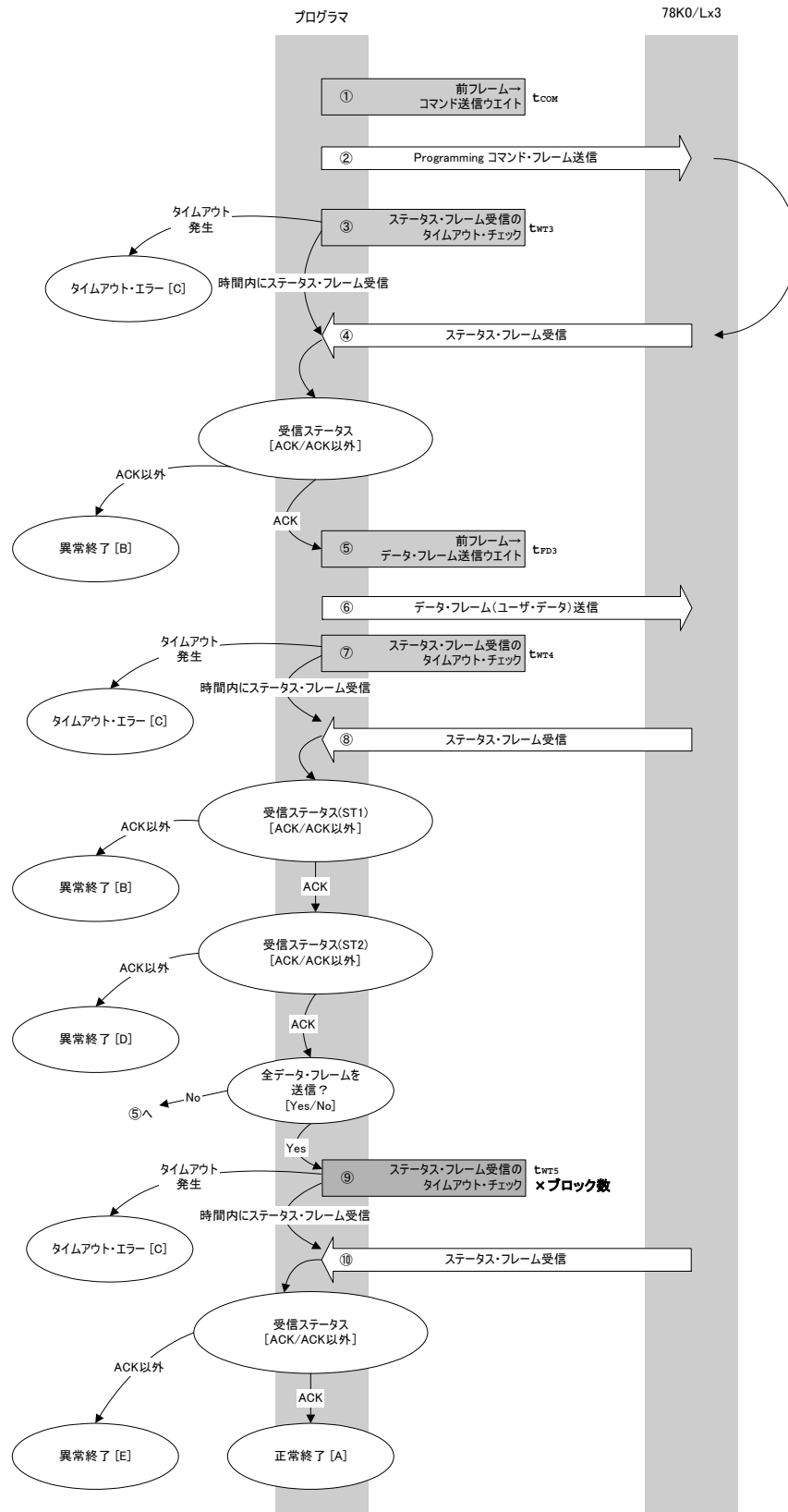
}

```

6.8 Programmingコマンド

6.8.1 処理手順チャート

Programmingコマンド処理手順



6.8.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

直前のフレームからデータ・フレーム送信までのウェイトをします（ウェイト時間 t_{FD3} ）。
 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。
 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} ）。
 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、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、より再実行します。
 ・ ST2 = ACK以外の場合 : **異常終了[D]**です。

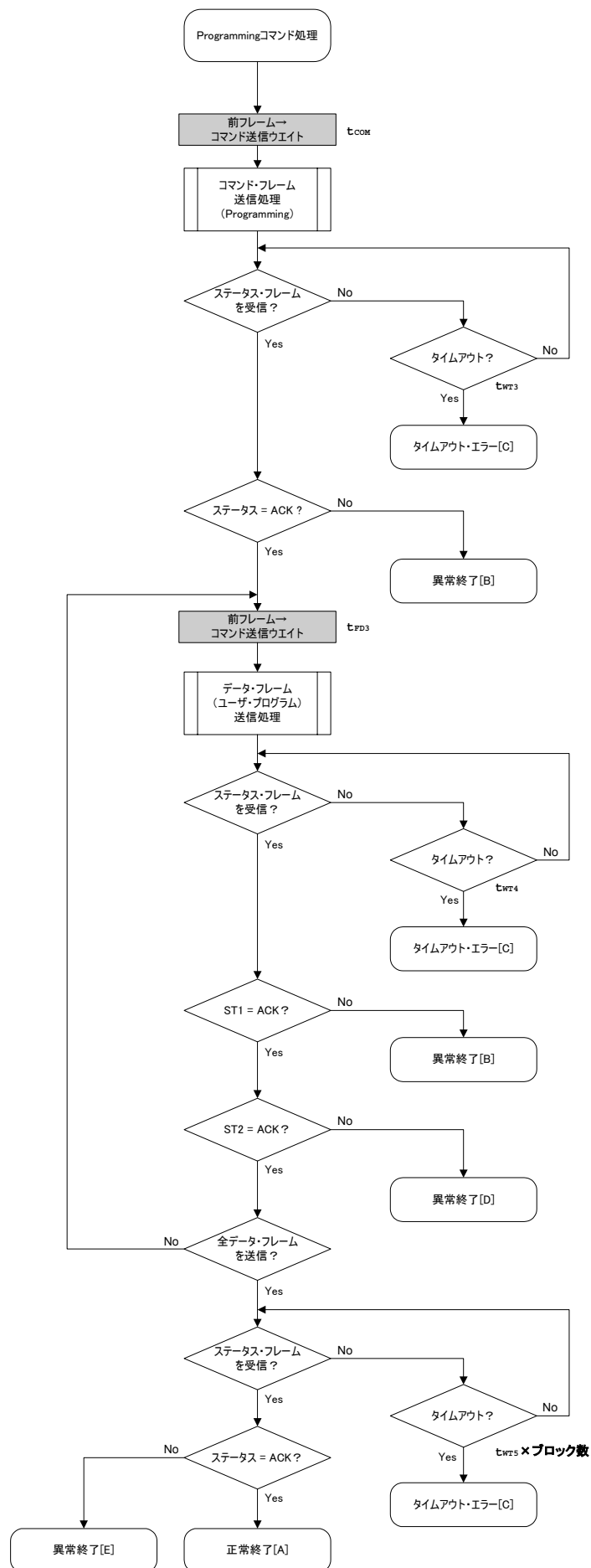
ステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は**タイムアウト・エラー[C]**となります
 （タイムアウト時間 $t_{WT5} \times$ ブロック数）。
 ステータス・コードをチェックします。

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

6.8.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または 1 K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム、またはデータ・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、「書き込み禁止」になっています。または、指定範囲にブート領域が含まれており、セキュリティ設定で、「ブート・ブロック書き換え禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。または、処理中にステータス・コマンド以外を受信しました。
	MR10 エラー	1AH	書き込み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。
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。

6.8.4 フロー・チャート



6.8.5 サンプル・プログラム

Programmingコマンド処理のサンプル・プログラムです。

```

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

#define          fl_st2_ua          (fl_ua_sfrm[OFS_STA_PLD+1])

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

    /*****
    /*      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_UA);          // 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_UA_MAX * block_num)); // get status frame
again
    // rc = get_sfrm_ua(fl_ua_sfrm, 5000000);          // get status frame again
    switch(rc) {
        //      case FLC_NO_ERR:        return rc;    break; // case [A]
        //      case FLC_DFTO_ERR:      return rc;    break; // case [C]
        //      default:                return rc;    break; // case [E]
    }

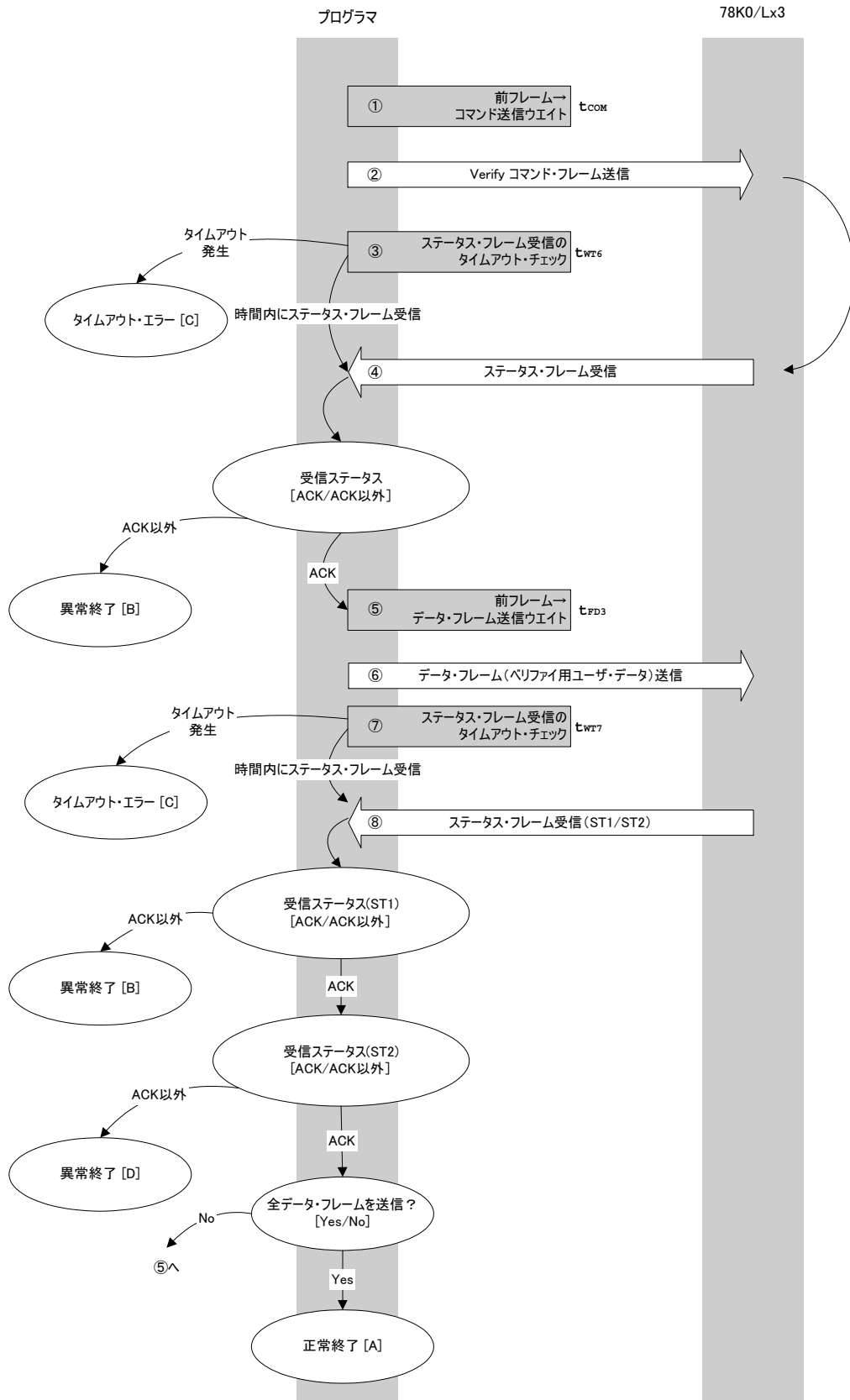
    return rc;
}

```

6.9 Verifyコマンド

6.9.1 処理手順チャート

Verifyコマンド処理手順



6.9.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

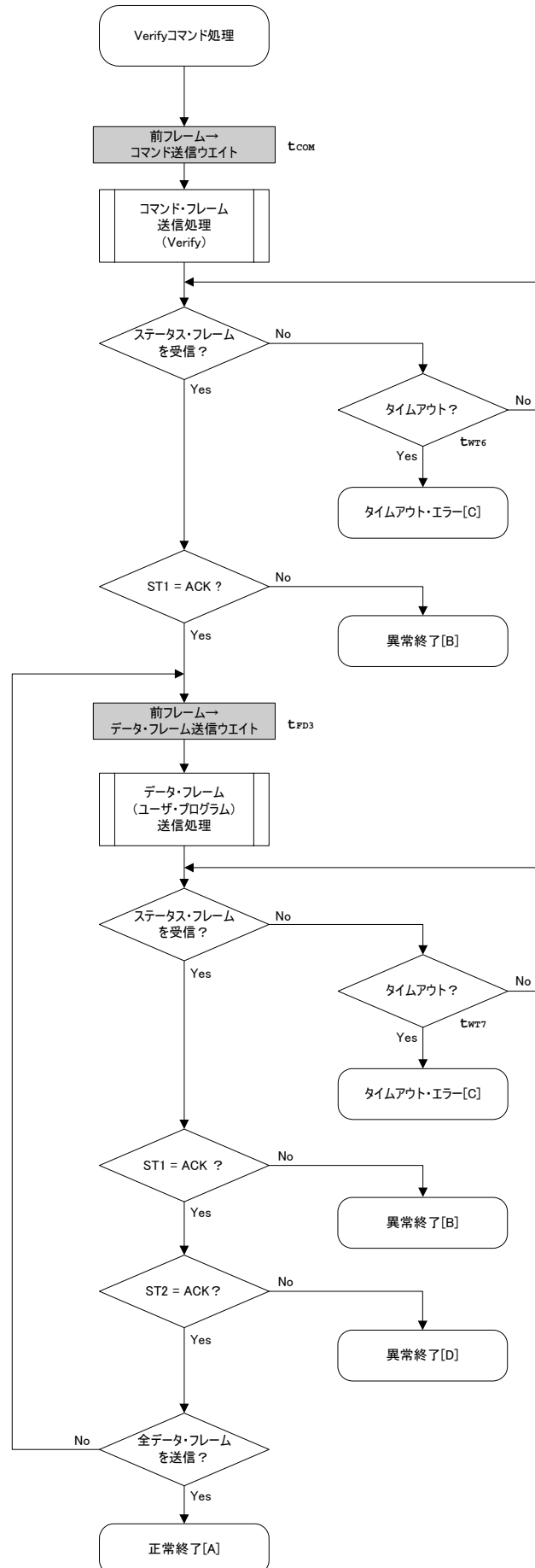
直前のフレームからデータ・フレーム送信までのウェイトをします（ウェイト時間 t_{FD3} ）。
 データ・フレーム送信処理により、ペリファイ用ユーザ・データを送信します。
 ユーザ・データ送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合はタイムアウト・エラー[C]となります（タイムアウト時間 t_{WT7} ）。
 ステータス・コード（ST1/ST2）をチェックします（処理手順チャートやフロー・チャートも参照してください）。

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 から再実行します。
 ・ ST2 = ACK以外の場合 : 異常終了[D]です。

6.9.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され、ペ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 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または1 K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム、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）。または、処理中にステータス・コマンド以外を受信しました。
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。
異常終了 [D]	ペリファイ・エラー	0FH（ST2）	ペリファイに失敗しました。または、その他のエラーが発生しました。

6.9.4 フロー・チャート



6.9.5 サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Verify command
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom   ... end address
/* [r] u16          ... error code
/*****
u16      fl_ua_verify(u32 top, u32 bottom)
{
    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_UA);          // 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

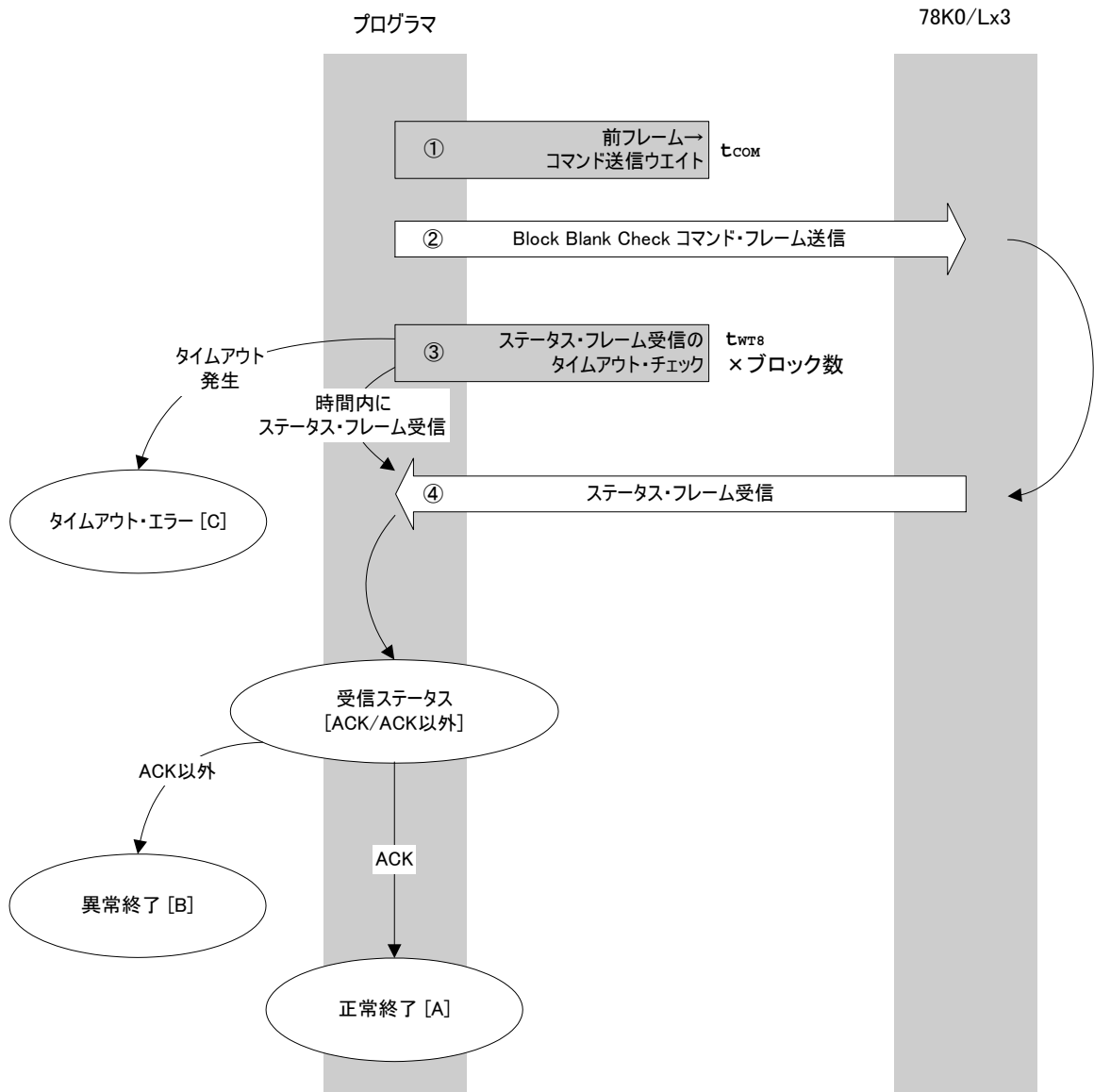
```

```
rc = get_sfrm_ua(fl_ua_sfrm, tWT7_TO);          // get status frame
switch(rc) {
    case FLC_NO_ERR:                             break; // continue
    // case FLC_DFEO_ERR: return rc; break; // case [C]
    default: return rc; break; // case [B]
}
if (fl_st2_ua != FLST_ACK) {                    // ST2 = ACK ?
    rc = decode_status(fl_st2_ua); // No
    return rc;                             // case [D]
}
if (is_end)                                    // send all user data ?
    break;                                    // yes
//continue;
}
return FLC_NO_ERR;    // case [A]
}
```

6. 10 Block Blank Checkコマンド

6. 10. 1 処理手順チャート

Block Blank Checkコマンド処理手順



6.10.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{com} ）。
 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。
 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 $t_{WT8} \times$ ブロック数）。
 ステータス・コードをチェックします。

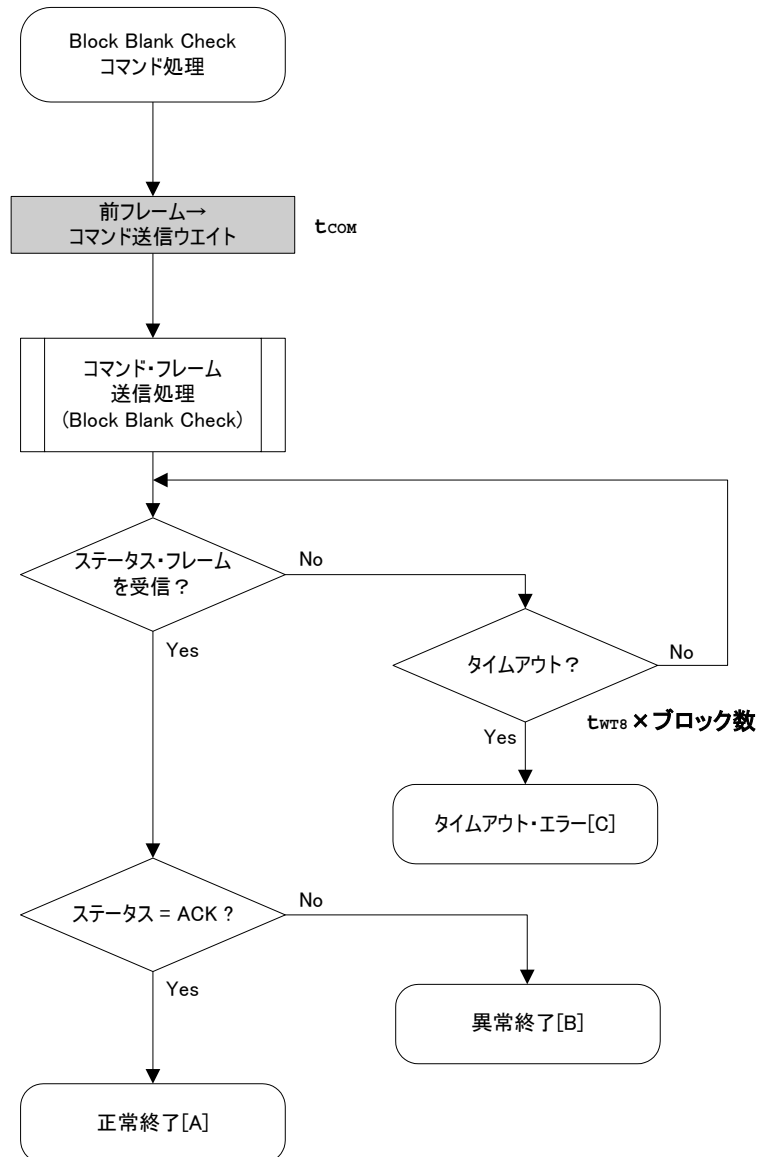
ST1 = ACKの場合 : **正常終了[A]** です。

ST1 = ACK以外の場合 : **異常終了[B]** です。

6.10.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、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	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または1 K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（データ長 (LEN) 不正, ETX なしなど）。または、処理中にステータス・コマンド以外を受信しました。
	MRG11 エラー	1BH	指定したブロックのフラッシュ・メモリがブランクではありません。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

6.10.4 フロー・チャート



6.10.5 サンプル・プログラム

Block Blank Checkコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Block blank check command
/*
/*
/*****
/* [i] u32 top      ... 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_UA);                // wait before sending command

    put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);

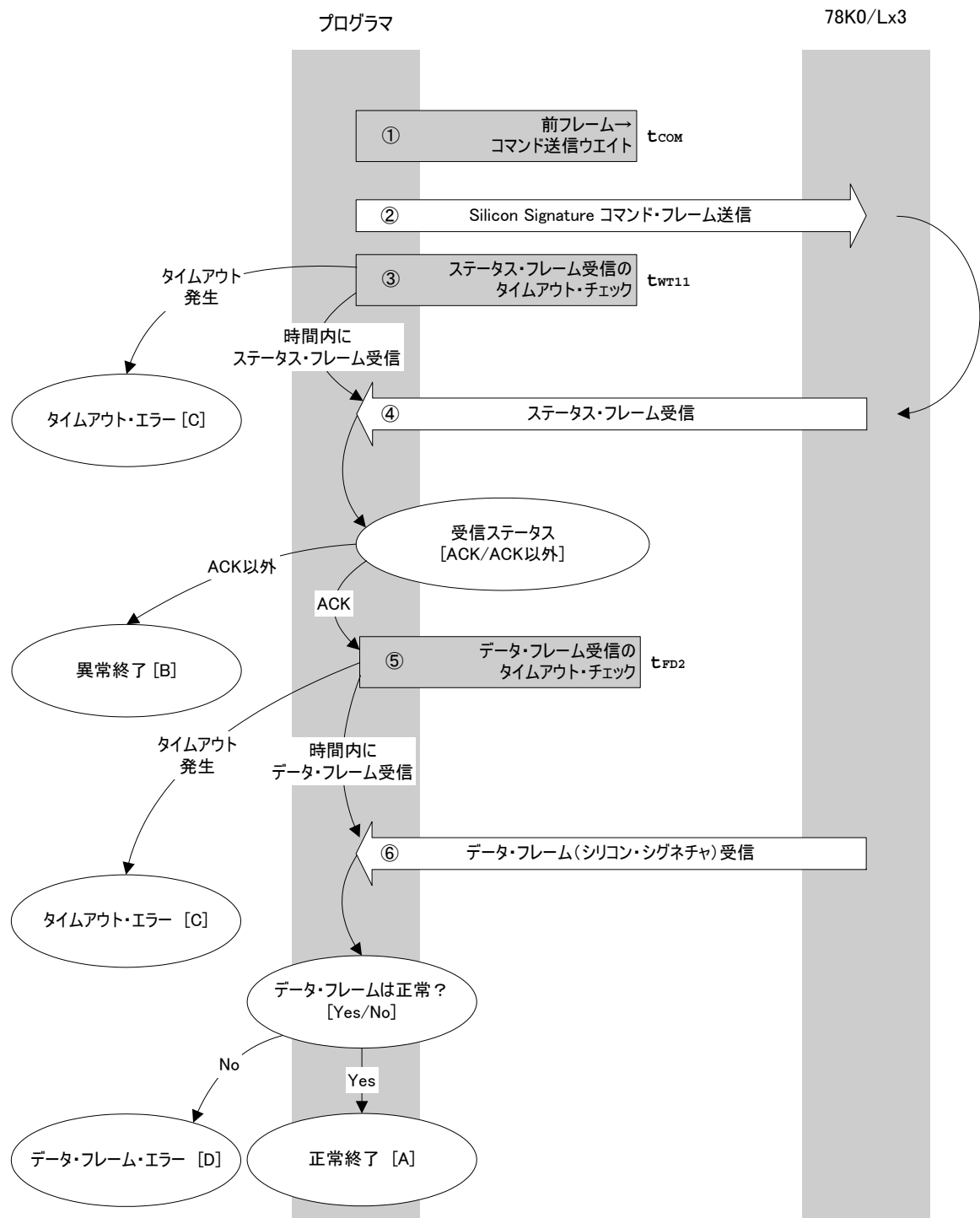
    rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX * block_num);    // get status frame
    // switch(rc) {
    //
    //      case    FLC_NO_ERR:      return rc;      break; // case [A]
    //      case    FLC_DFTO_ERR:    return rc;      break; // case [C]
    //      default:                  return rc;      break;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

```

6. 11 Silicon Signatureコマンド

6. 11. 1 処理手順チャート

Silicon Signatureコマンド処理手順



6. 11. 2 処理手順説明

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} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

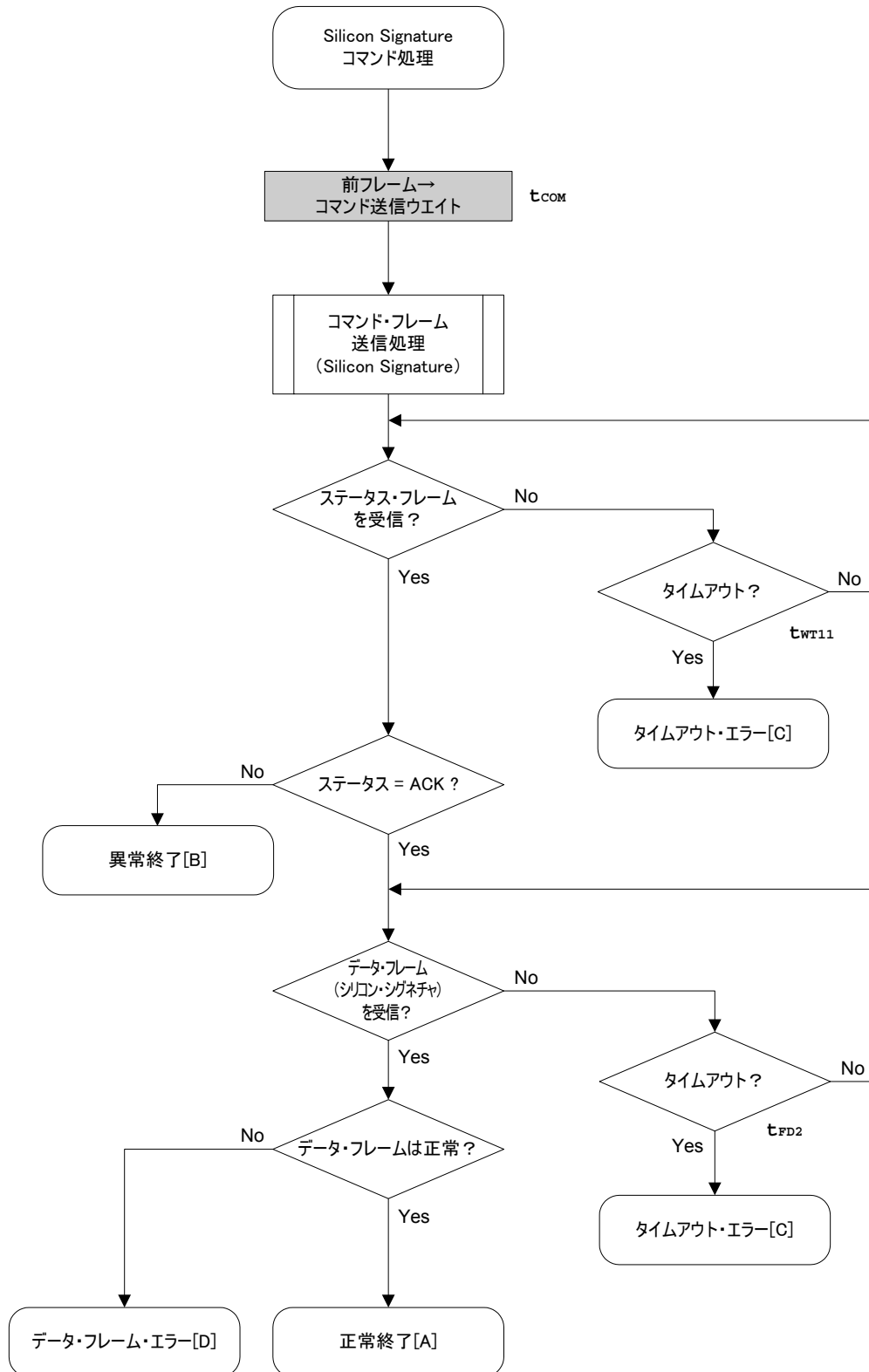
データ・フレーム（シリコン・シグネチャ・データ）受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は**タイムアウト・エラー[C]**です（タイムアウト時間 t_{FD2} ）。
 受信したデータ・フレーム（シリコン・シグネチャ・データ）をチェックします。

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

6. 11. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、シリコン・シグネチャ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）。または、処理中にステータス・コマンド以外を受信しました。
	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]		-	規定の時間内にステータス・フレーム、またはデータ・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。

6.11.4 フロー・チャート



6.11.5 サンプル・プログラム

Silicon Signatureコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM_UA);          // 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]
    }

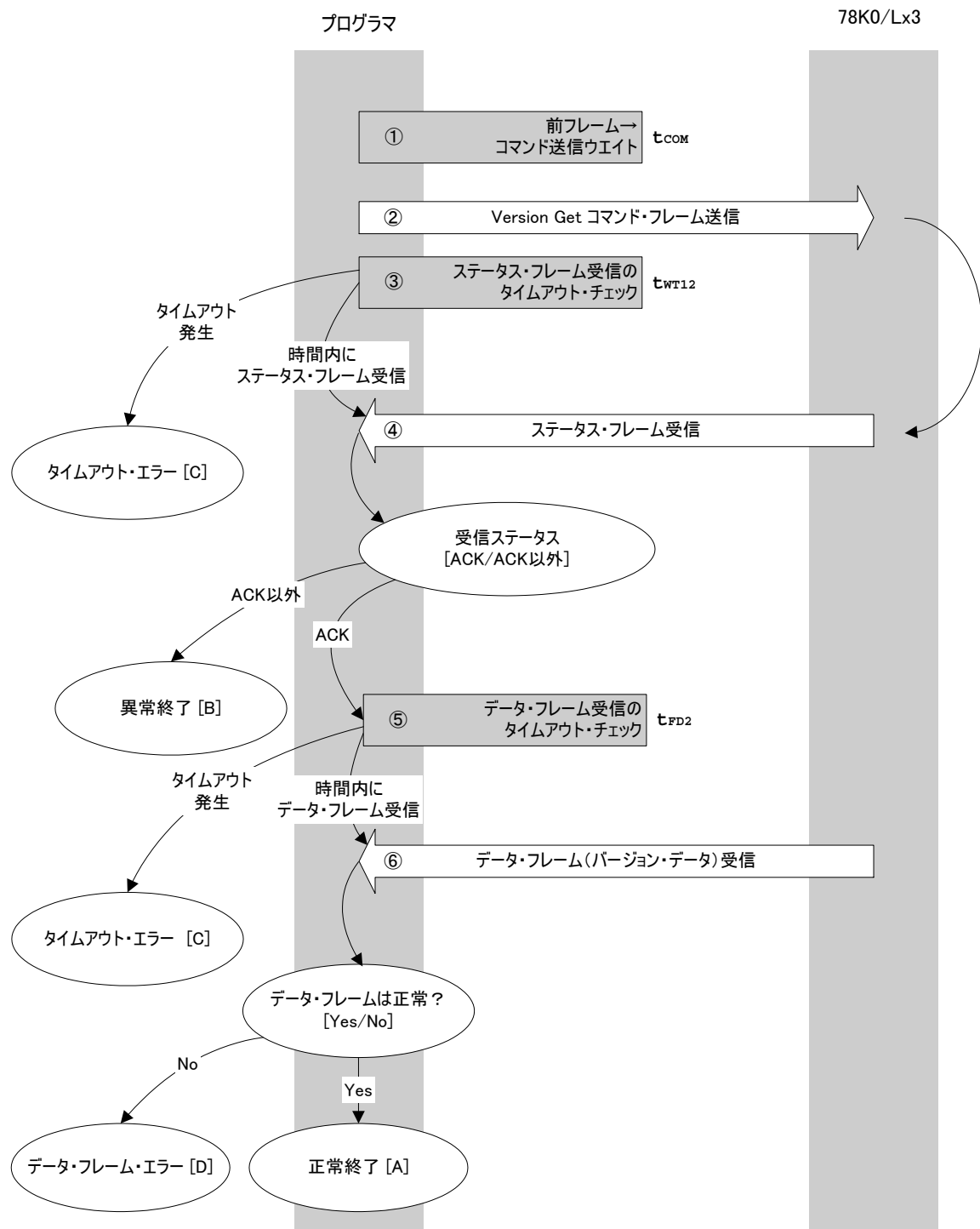
    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_TO); // get status frame
    if (rc){
        return rc;                // if error
        return rc;                // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]); // copy Signature
data
    return rc;                // case [A]
}

```

6. 12 Version Getコマンド

6. 12. 1 処理手順チャート

Version Getコマンド処理手順



6. 12. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

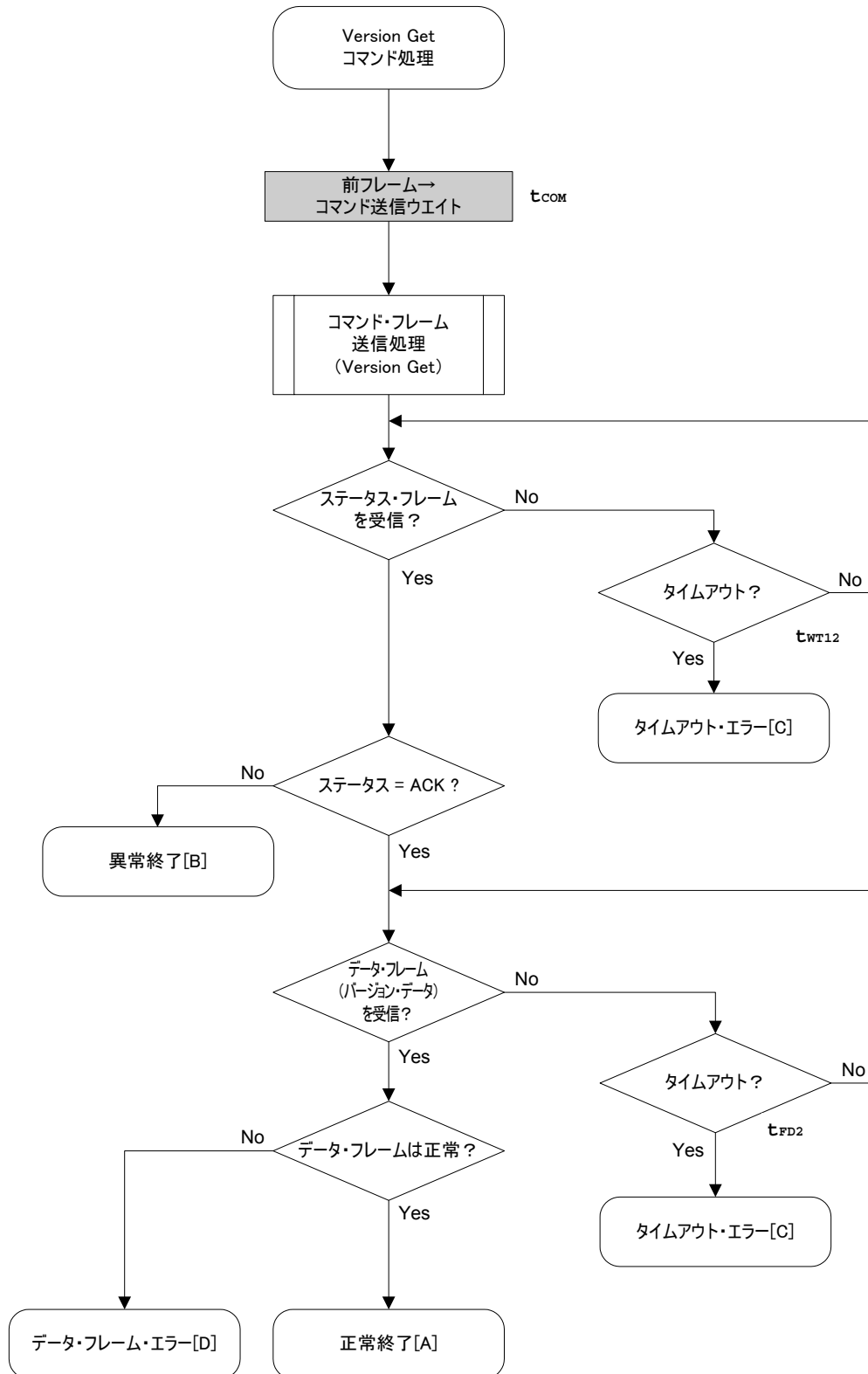
データ・フレーム（バージョン・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は**タイムアウト・エラー[C]**です（タイムアウト時間 t_{FD2} ）。
 受信したデータ・フレーム（バージョン・データ）をチェックします。

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

6. 12. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、バージョン・データを取得できたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（データ長（LEN）不正、ETX なしなど）。または、処理中にステータス・コマンド以外を受信しました。
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、またはデータ・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。

6.12.4 フロー・チャート



6. 12. 5 サンプル・プログラム

Version Getコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM_UA);          // 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]
    }

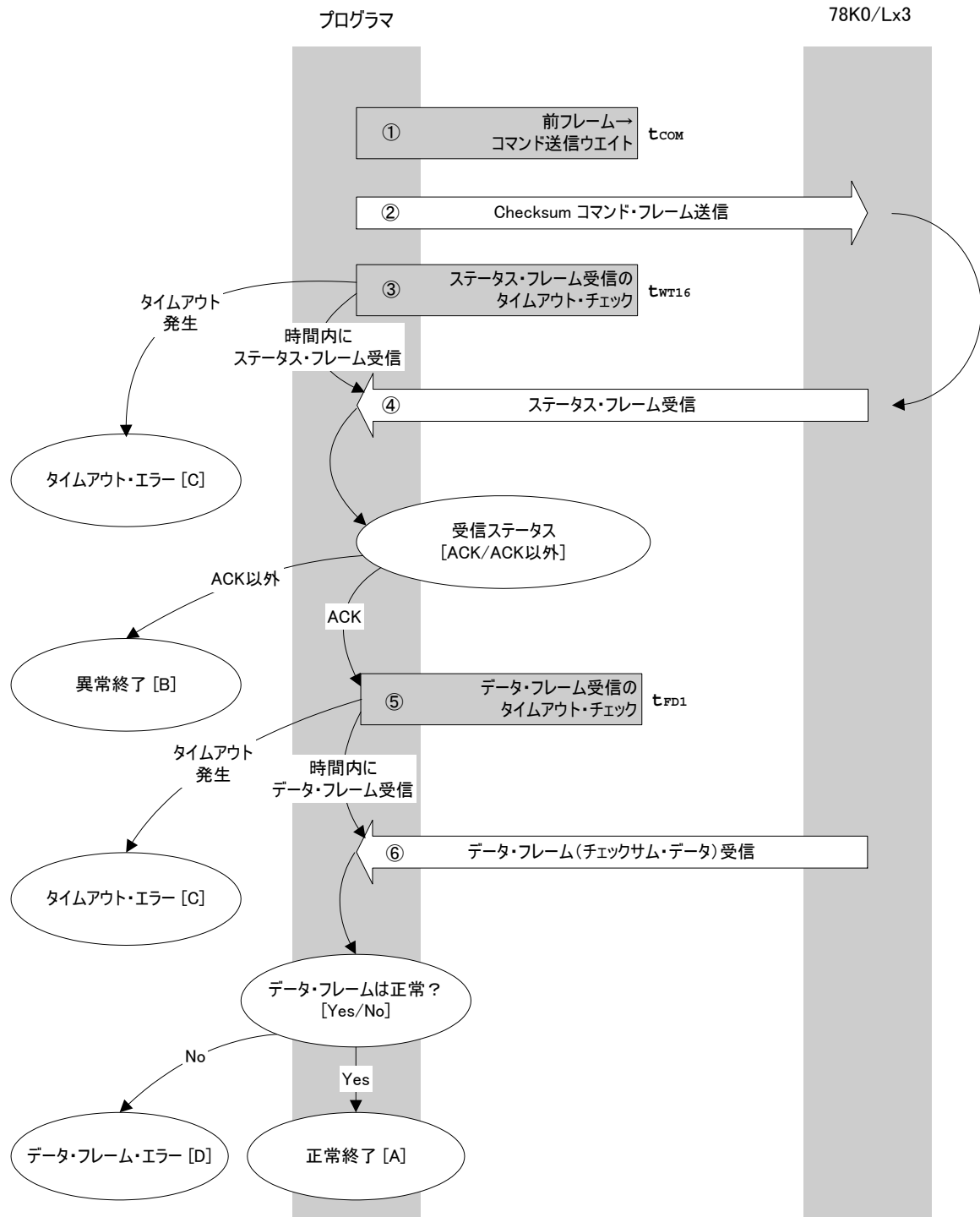
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                // case [A]
}

```

6. 13 Checksumコマンド

6. 13. 1 処理手順チャート

Checksumコマンド処理手順



6.13.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{com} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

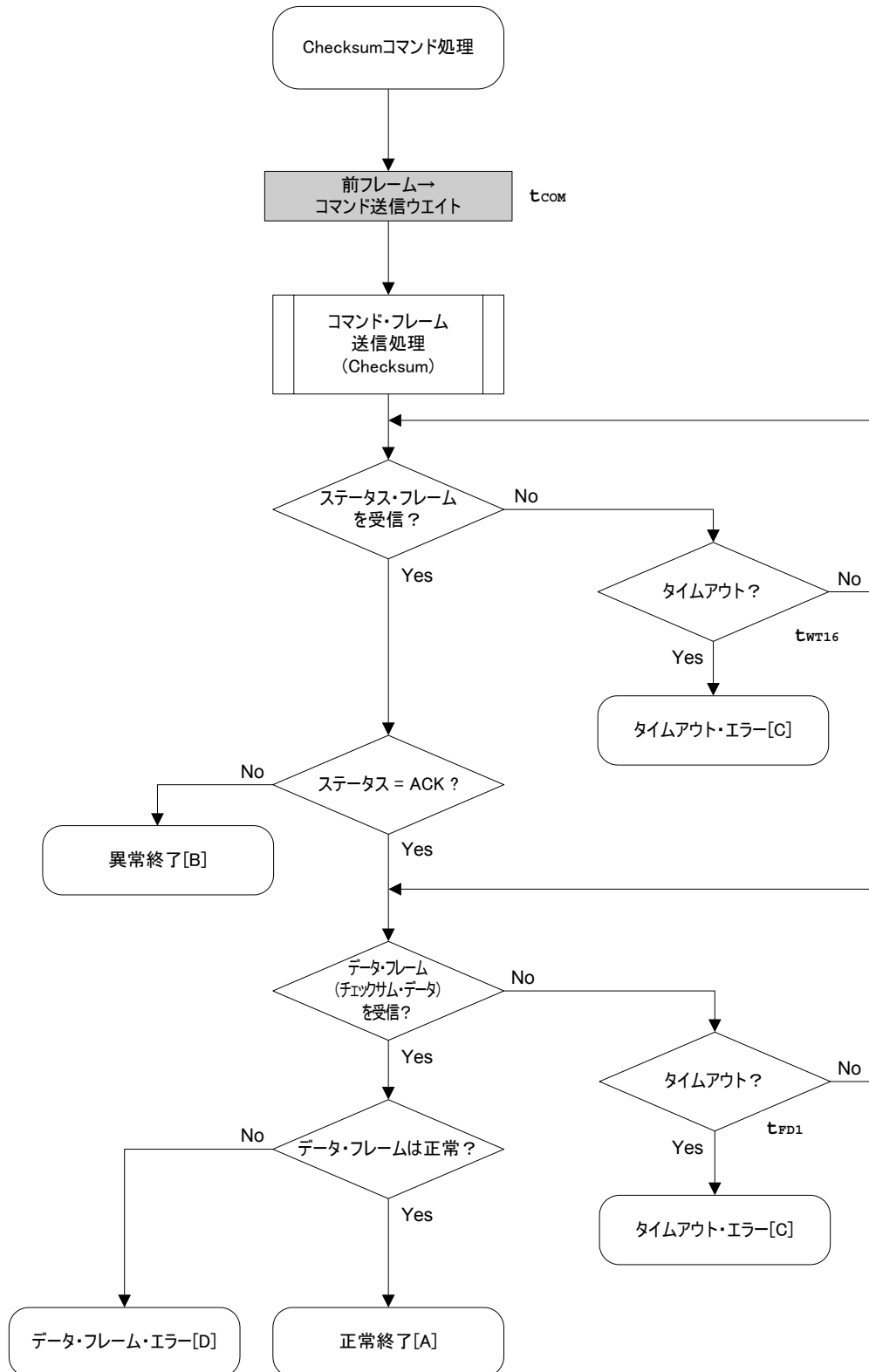
データ・フレーム（チェックサム・データ）受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です（タイムアウト時間 t_{FD1} ）。
 受信したデータ・フレーム（チェックサム・データ）をチェックします。

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

6.13.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、チェックサム・データを取得できたことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または、1 K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（データ長 (LEN) 不正, ETX なしなど）。または、処理中にステータス・コマンド以外を受信しました。
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、またはデータ・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。

6.13.4 フロー・チャート



6. 13. 5 サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

```

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

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

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

    fl_wait(tCOM_UA);          // 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){
        return rc;                // case [D]
    }

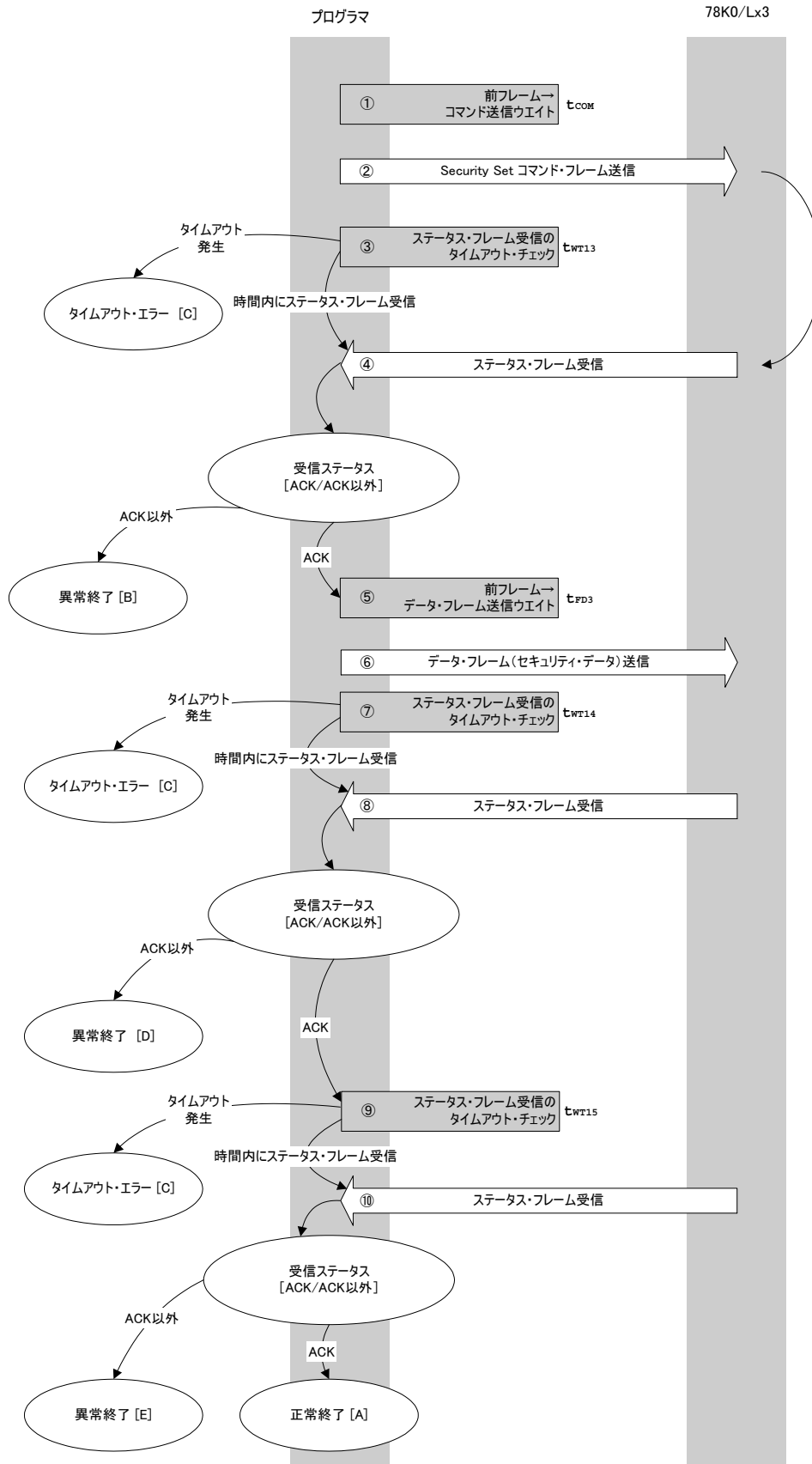
    *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1]; // set SUM
data
    return rc;                // case [A]
}

```

6. 14 Security Setコマンド

6. 14. 1 処理手順チャート

Security Setコマンド処理手順



6. 14. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

直前のフレームからデータ・フレーム送信までのウェイトをします（ウェイト時間 t_{FD3} ）。
 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。
 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} ）。
 ステータス・コードをチェックします。

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

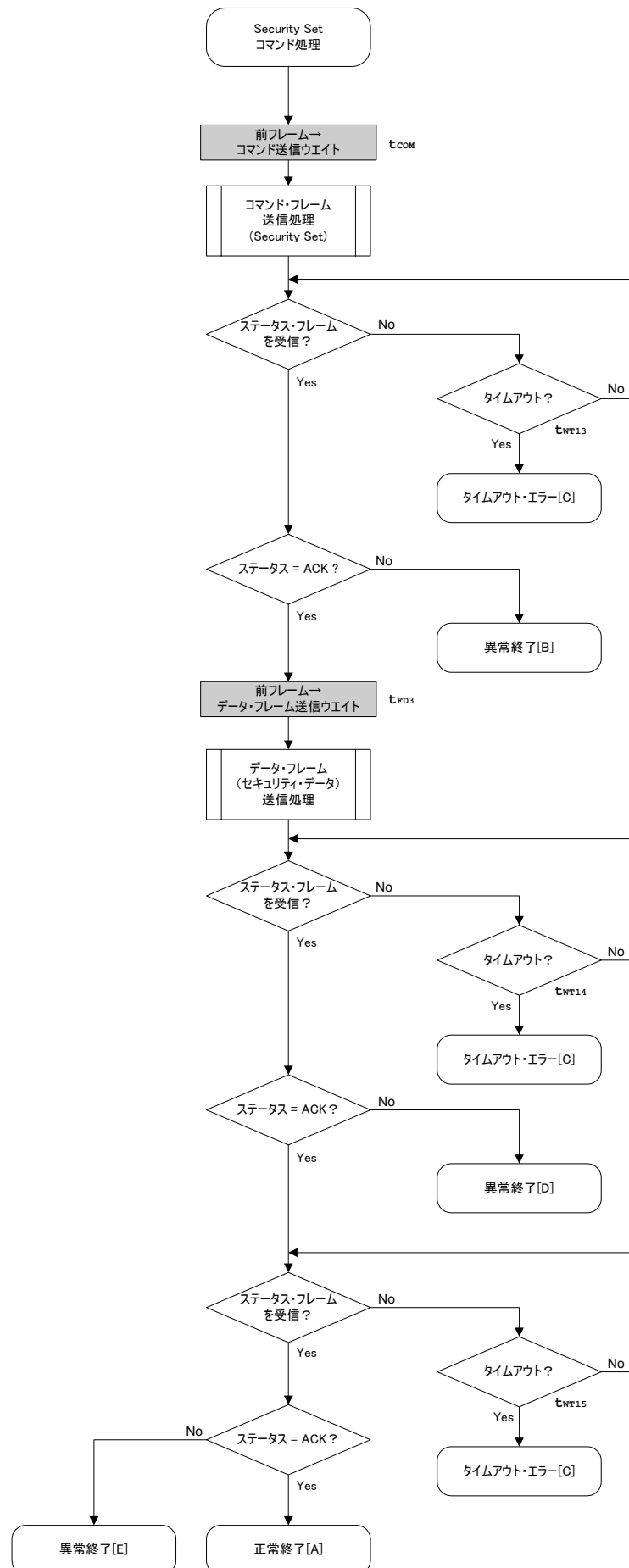
ステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は**タイムアウト・エラー[C]**となります（タイムアウト時間 t_{WT15} ）。
 ステータス・コードをチェックします。

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

6. 14. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、セキュリティ設定データが正しく設定されたことを示します。
異常終了 [B]	パラメータ・エラー	05H	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。
	MRG10 エラー	1AH	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）。または、処理中にステータス・コマンド以外を受信しました。
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、またはデータ・フレームの受信ができませんでした。
異常終了 [D]	プロテクト・エラー	10H	すでに禁止されているセキュリティ・フラグを許可しようとした。
	Write エラー	1CH	書き込みエラーが発生しました。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

6. 14. 4 フロー・チャート



6. 14. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```

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

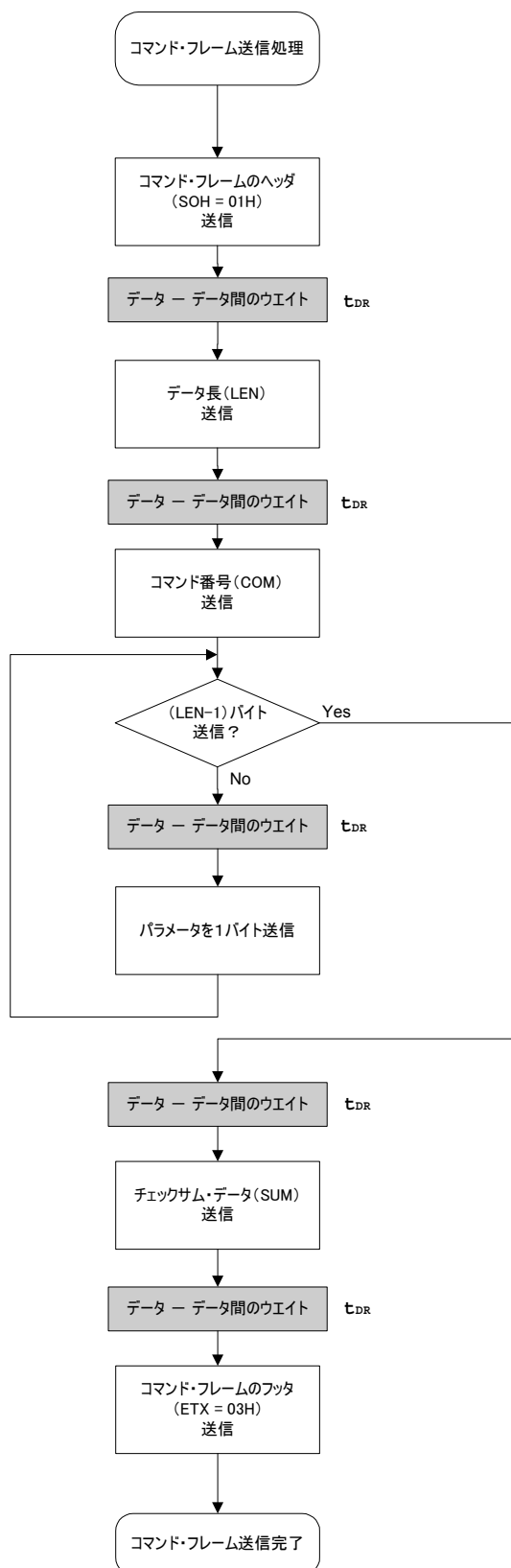
```

```
// 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;  
}
```

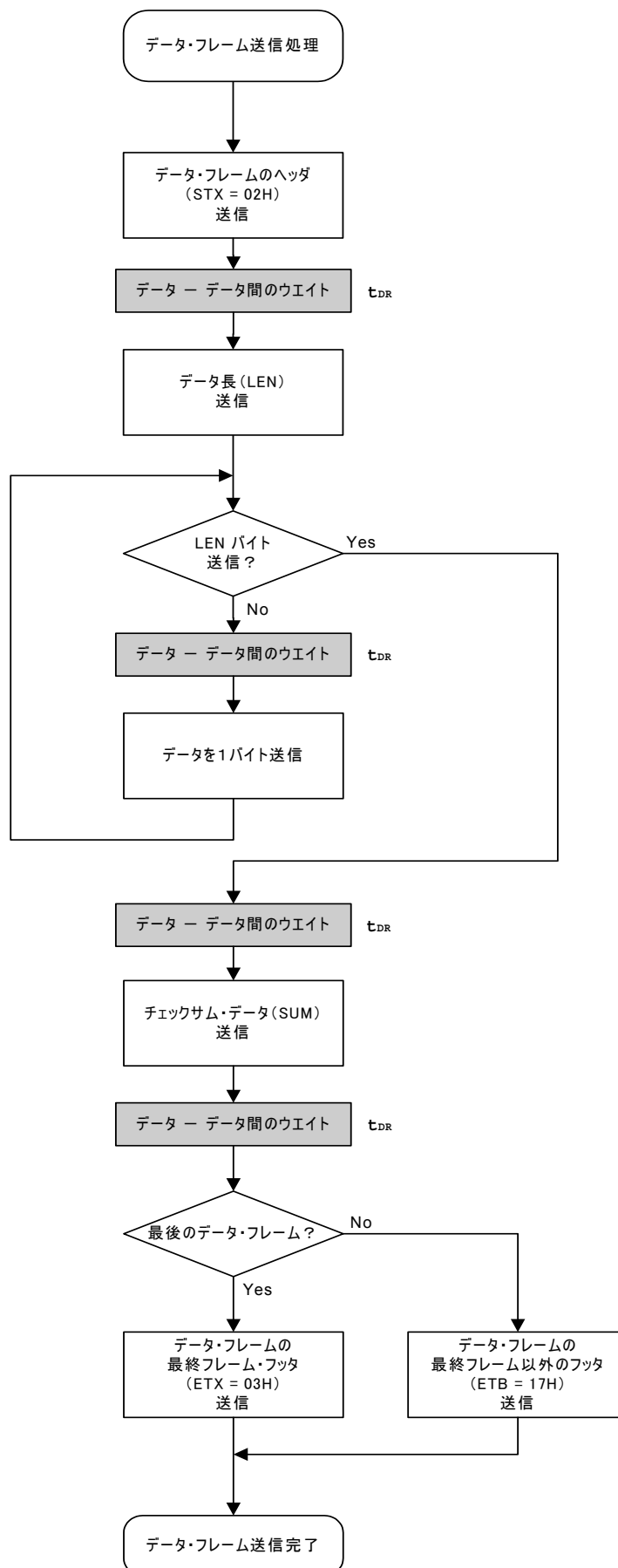
第7章 3線式シリアルI/O (CSI) 通信方式

この章のフロー・チャートで示した略号 (txxおよび t_{WTXX}) は、第8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8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。

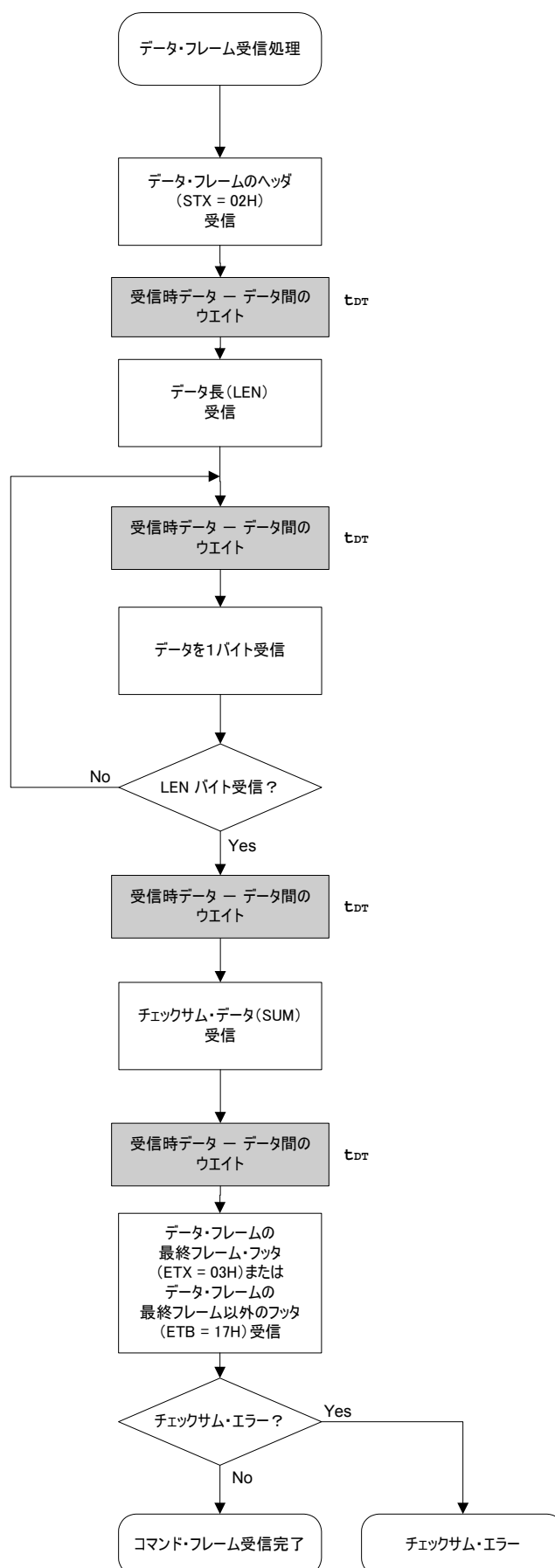
7.1 コマンド・フレーム送信処理のフロー・チャート



7.2 データ・フレーム送信処理のフロー・チャート



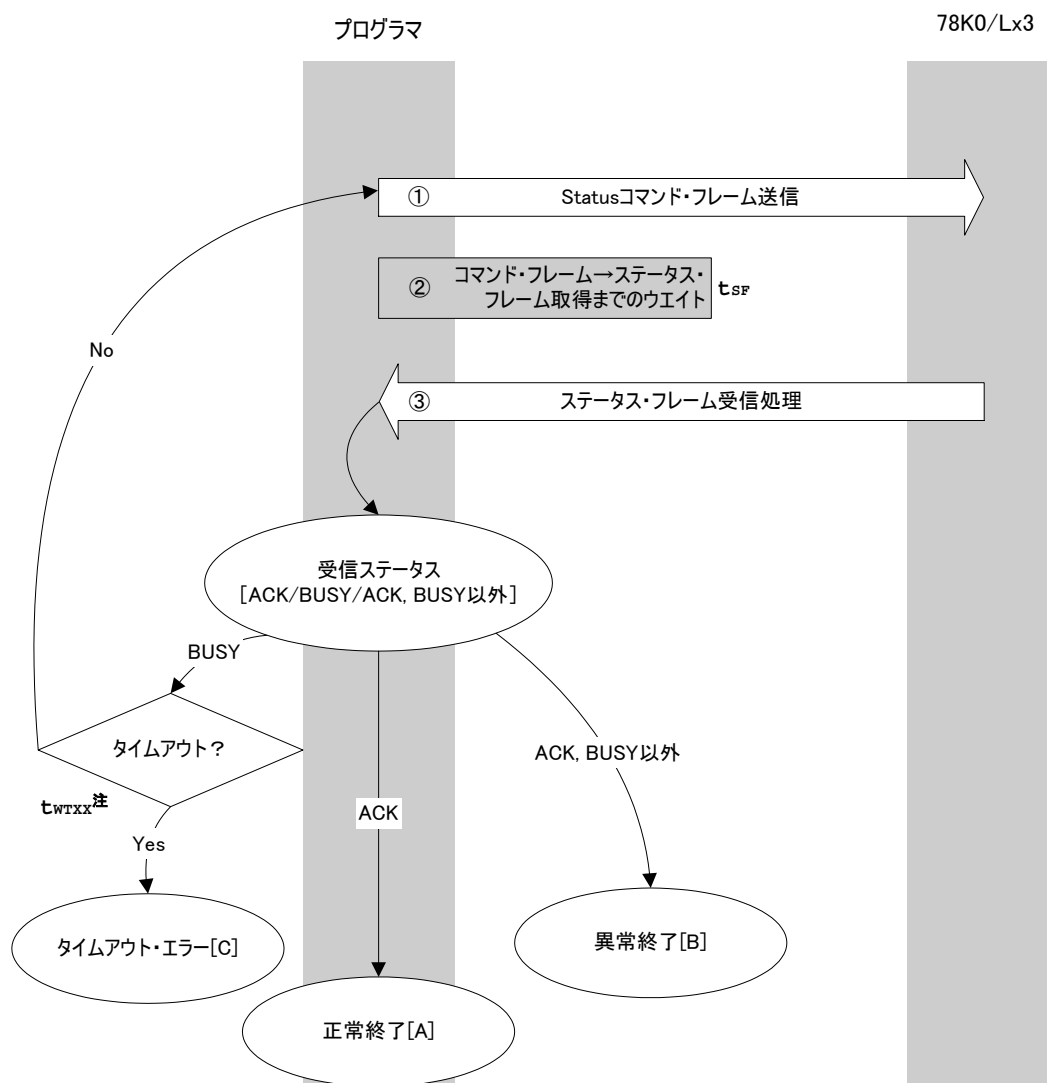
7.3 データ・フレーム受信処理のフロー・チャート



7.4 Statusコマンド

7.4.1 処理手順チャート

Statusコマンド処理手順



注 実行コマンドにより、適用スペックが異なります。

7.4.2 処理手順説明

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。
 コマンド送信からステータス・フレーム受信までのウェイトをします(ウェイト時間 t_{SF})。
 ステータス・コードをチェックします。

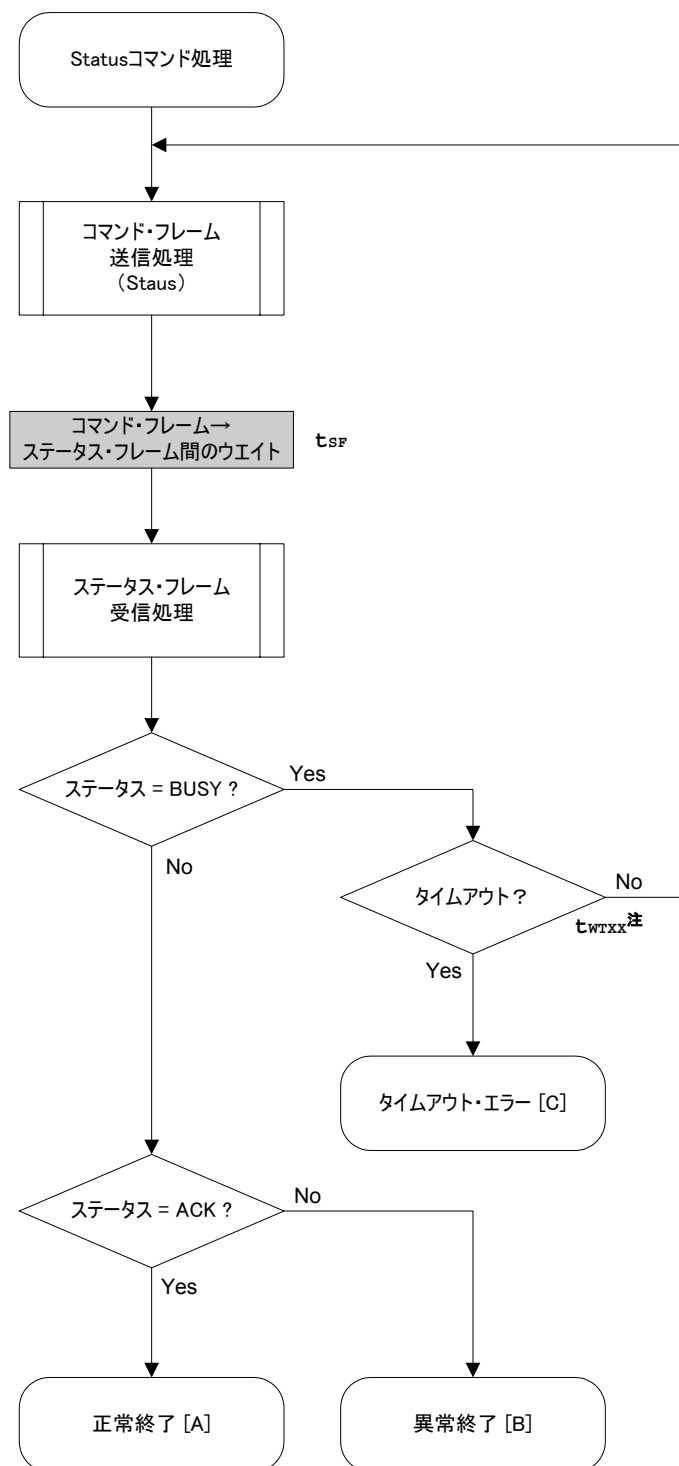
ST1 = ACKの場合 : **正常終了[A]**です。
 ST1 = BUSYの場合 : タイムアウト (t_{WXXX}) ^注をチェックします。
 → タイムアウトでなければ からやり直します。
 → タイムアウトであれば **タイムアウト・エラー[C]**です。
 ST1 = ACK, BUSY以外の場合 : **異常終了[B]**です。

注 実行コマンドにより、適用スペックが異なります。

7.4.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	78K0/Lx3 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)が適切ではありません。
	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。
	ペリファイ・エラー	0FH	プログラマから送信されたデータとのペ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しました。
	否定応答 (NACK)	15H	否定応答
	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ペリファイ・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。

7.4.4 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。

7.4.5 サンプル・プログラム

Statusコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Get status command (CSI)
/*
/*
*****/
/* [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:
                return  rc;                          // checksum error

            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)
    }

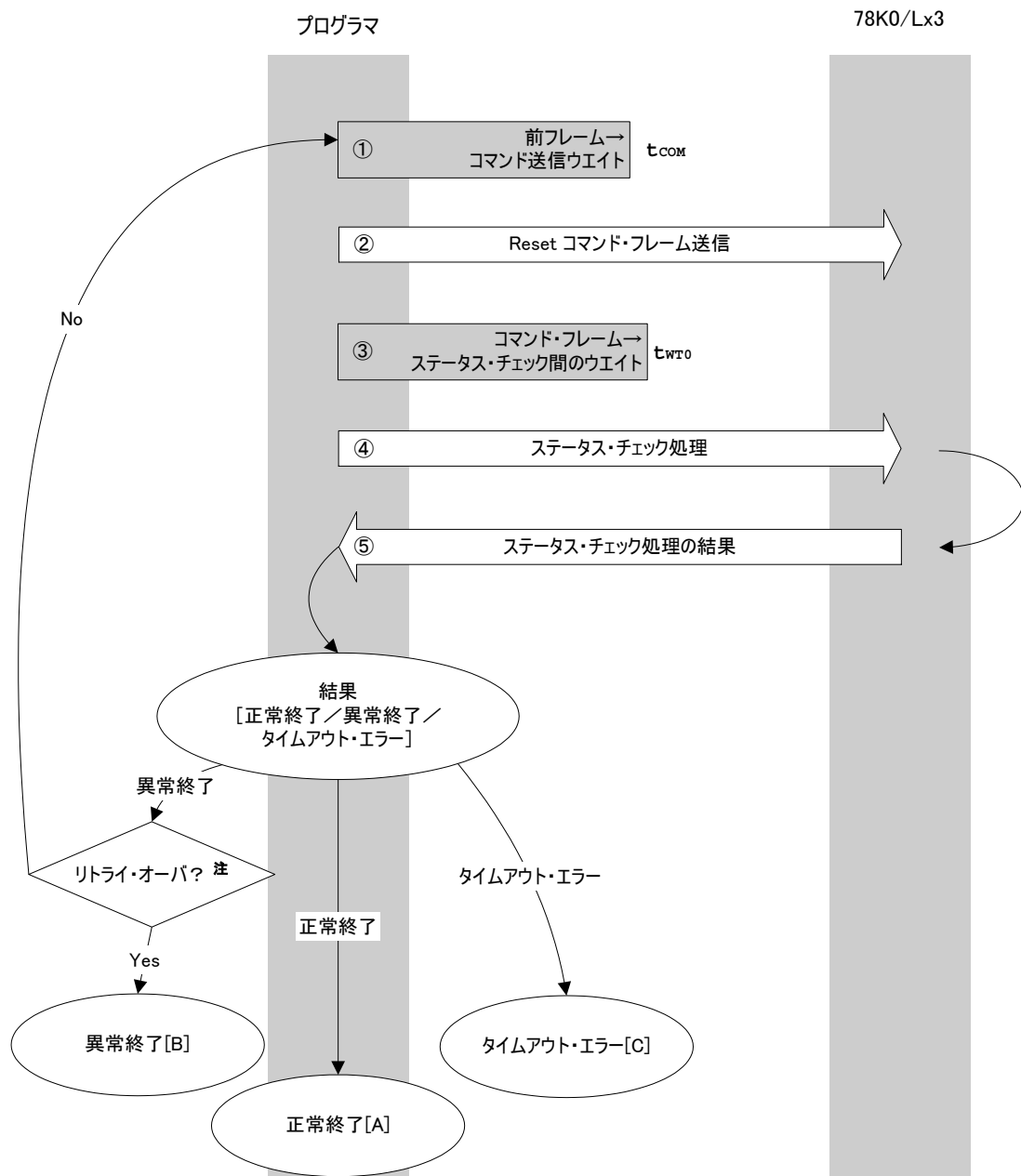
    rc = decode_status(fl_st1); // decode status to return code
    // switch(rc) {
    //
    //      case   FLC_NO_ERR:    return rc;    break; // case [A]
    //      default:             return rc;    break; // case [B]
    // }
    return rc;
}

```

7.5 Resetコマンド

7.5.1 処理手順チャート

Resetコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

7.5.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします（ウェイト時間 t_{WT0} ）。
 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。
 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。

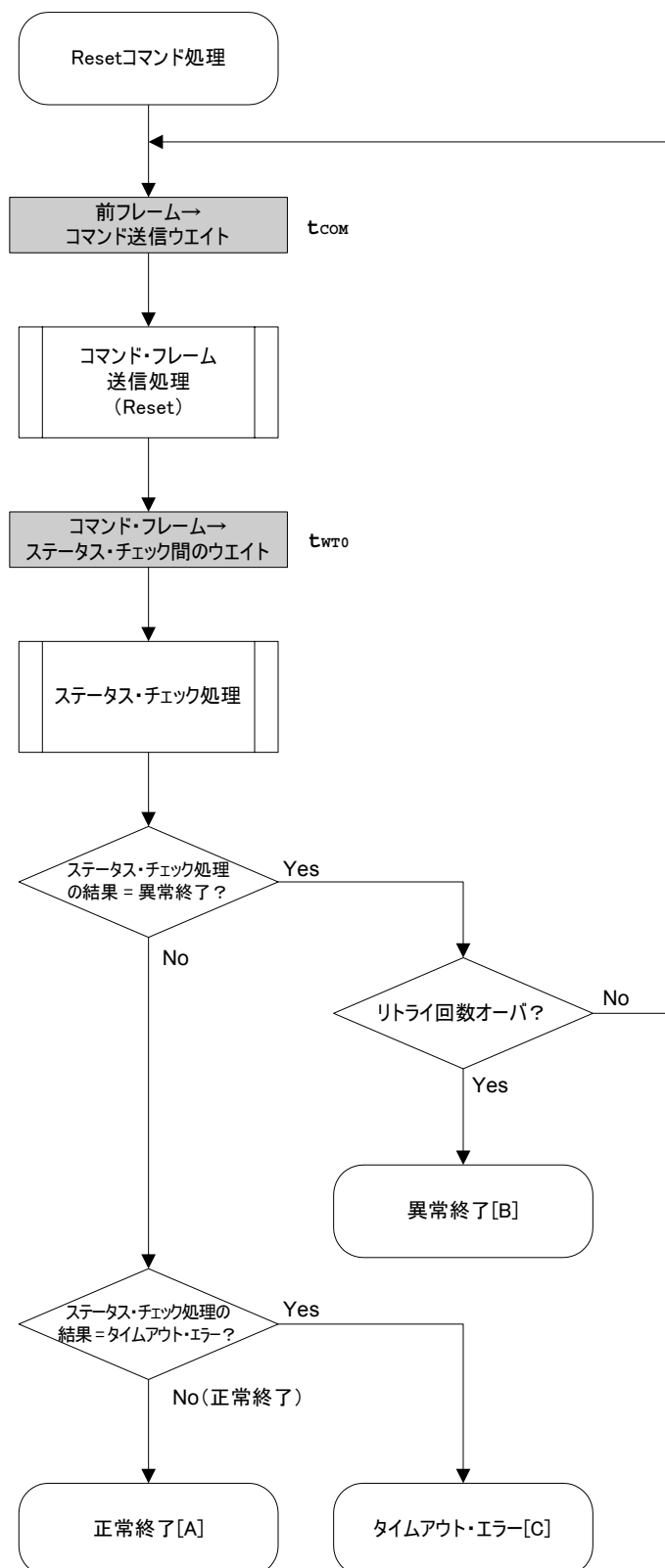
異常終了の場合 : リトライ・オーバでなければ、より再実行します。
リトライ・オーバであれば、異常終了[B] です。

タイムアウト・エラーの場合 : タイムアウト・エラー[C] です。

7.5.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、プログラマと 78K0/Lx3 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)。または、処理中にステータス・コマンド以外を受信しました。
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。

7.5.4 フロー・チャート



7.5.5 サンプル・プログラム

Resetコマンド処理のサンプル・プログラムです。

```

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

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

        fl_wait(tCOM_CSI);                // 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;
}

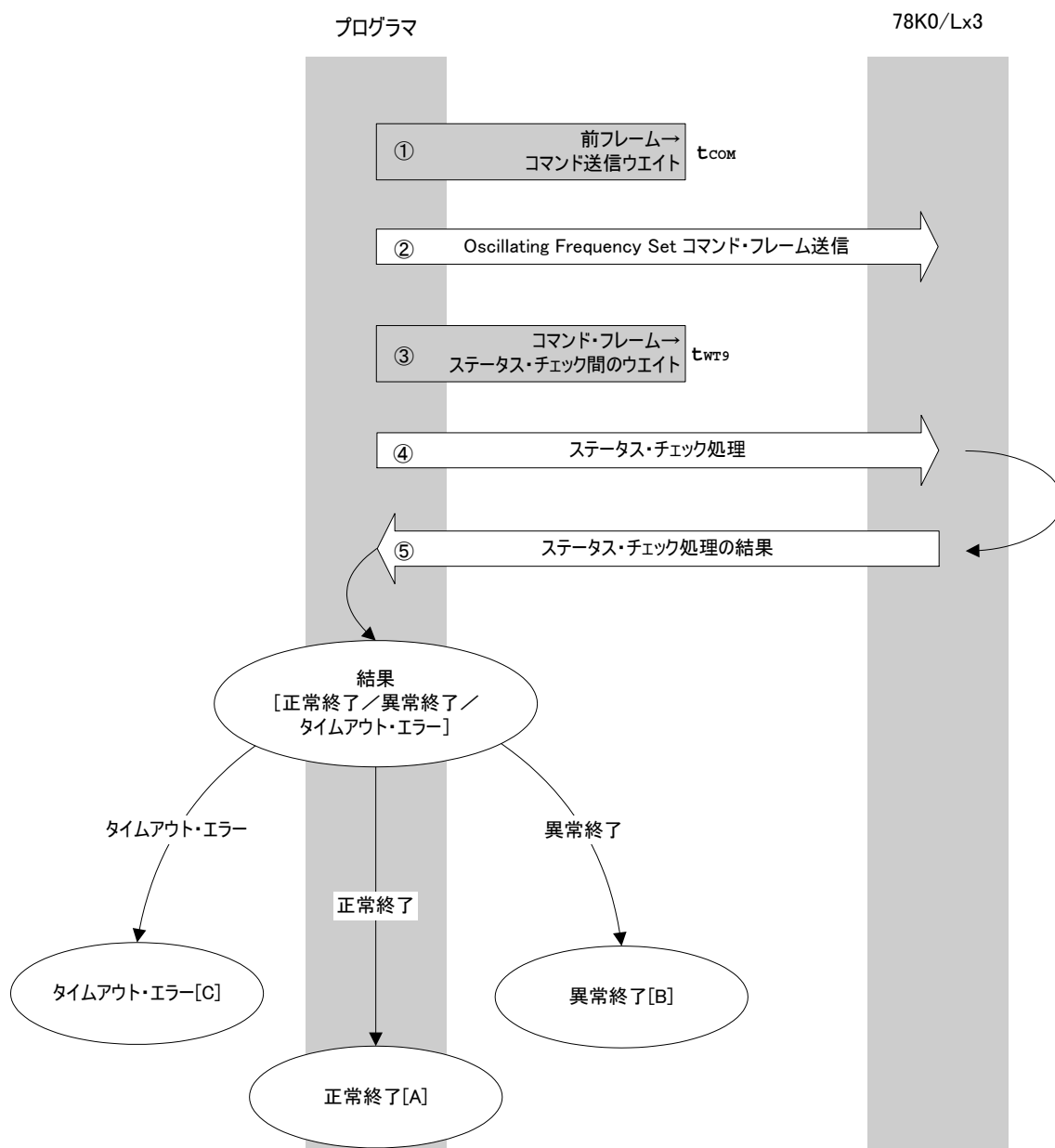
```

7.6 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通信時で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）。

7.6.1 処理手順チャート

Oscillating Frequency Setコマンド処理手順



7.6.2 処理手順説明

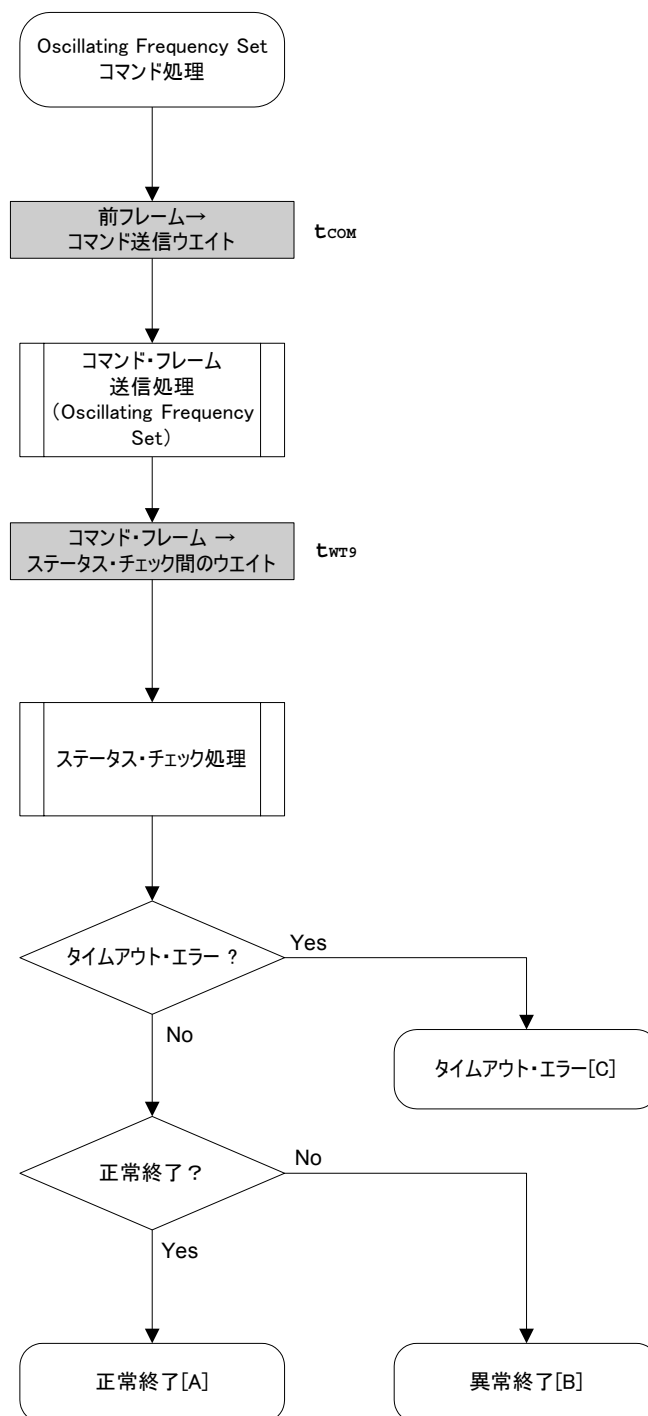
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 コマンド・フレーム送信処理により, Oscillating Frequency Setコマンド^{*}を送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT9})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

7.6.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され, 78K0/Lx3 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)。または, 処理中にステータス・コマンド以外を受信しました。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームが受信できませんでした。

7.6.4 フロー・チャート



7.6.5 サンプル・プログラム

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

```

/*****
/*
/* Set Flash device clock value command (CSI)
/*
/*****
/* [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_CSI);                // wait before sending command frame

    put_cmd_csi(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);
                                    // send "OscilationFrequencySet" command

    fl_wait(tWT9);

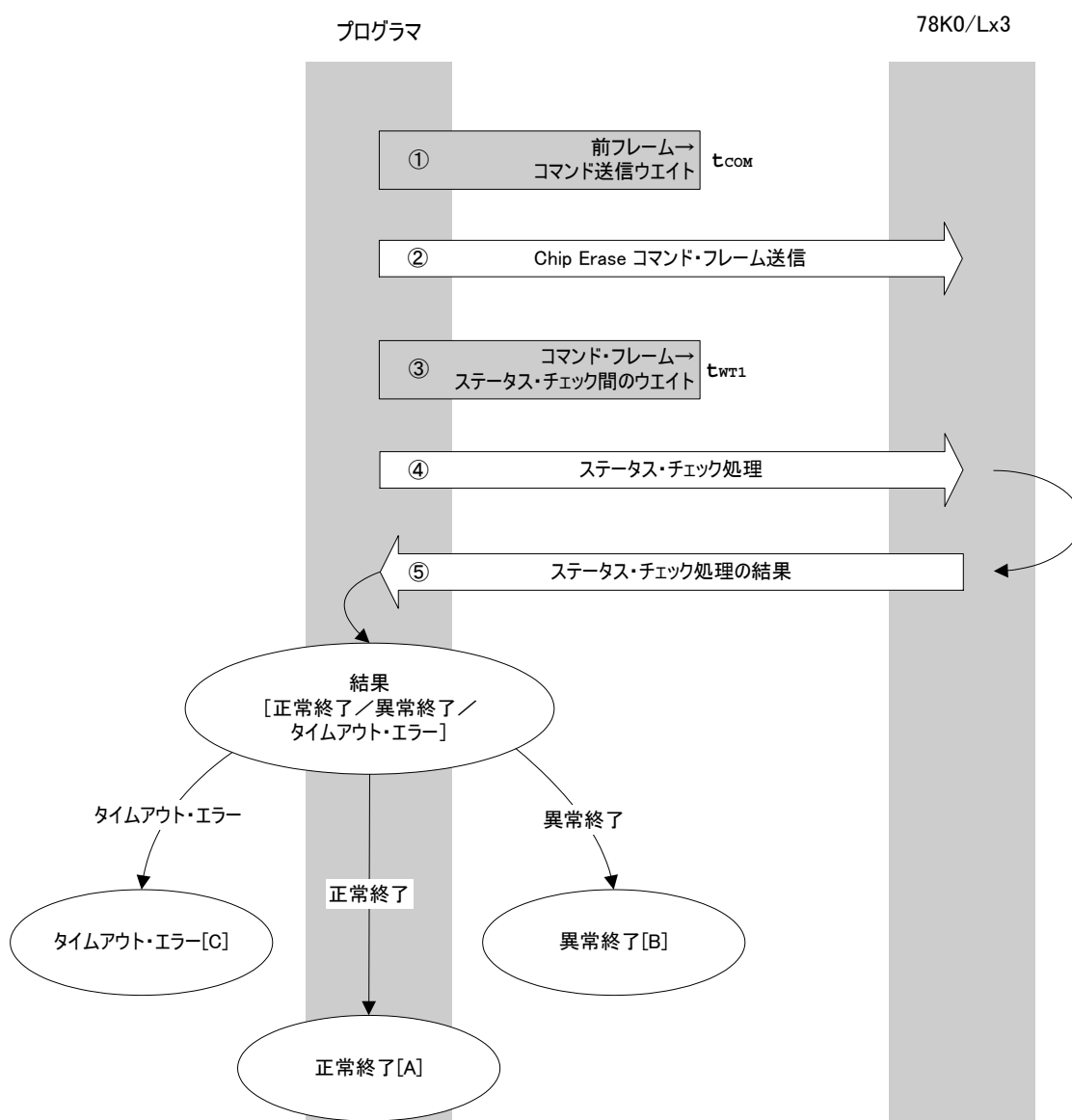
    rc = fl_csi_getstatus(tWT9_TO); // get status frame
    // switch(rc) {
    //
    //      case    FLC_NO_ERR:      return rc;      break; // case [A]
    //      case    FLC_DFTO_ERR:    return rc;      break; // case [C]
    //      default:                  return rc;      break; // case [B]
    // }
    return rc;
}

```

7.7 Chip Eraseコマンド

7.7.1 処理手順チャート

Chip Eraseコマンド処理手順



7.7.2 処理手順説明

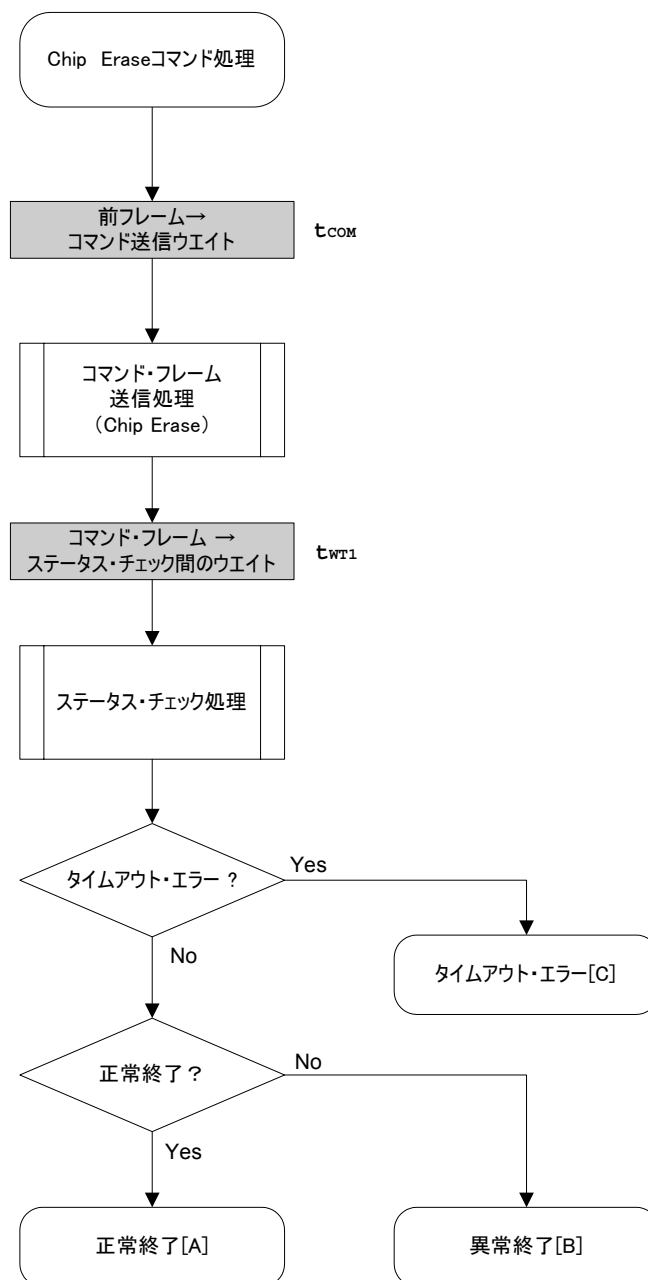
直前のフレームからコマンド送信までのウェイトをします(ウェイト時間 t_{COM})。
 コマンド・フレーム送信処理により、Chip Eraseコマンドを送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします(ウェイト時間 t_{WT1})。
 ステータス・チェック処理により、ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

7.7.3 終了時の内容

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

7.7.4 フロー・チャート



7.7.5 サンプル・プログラム

Chip Eraseコマンド処理のサンプル・プログラムです。

```

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

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

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

    fl_wait(tWT1);

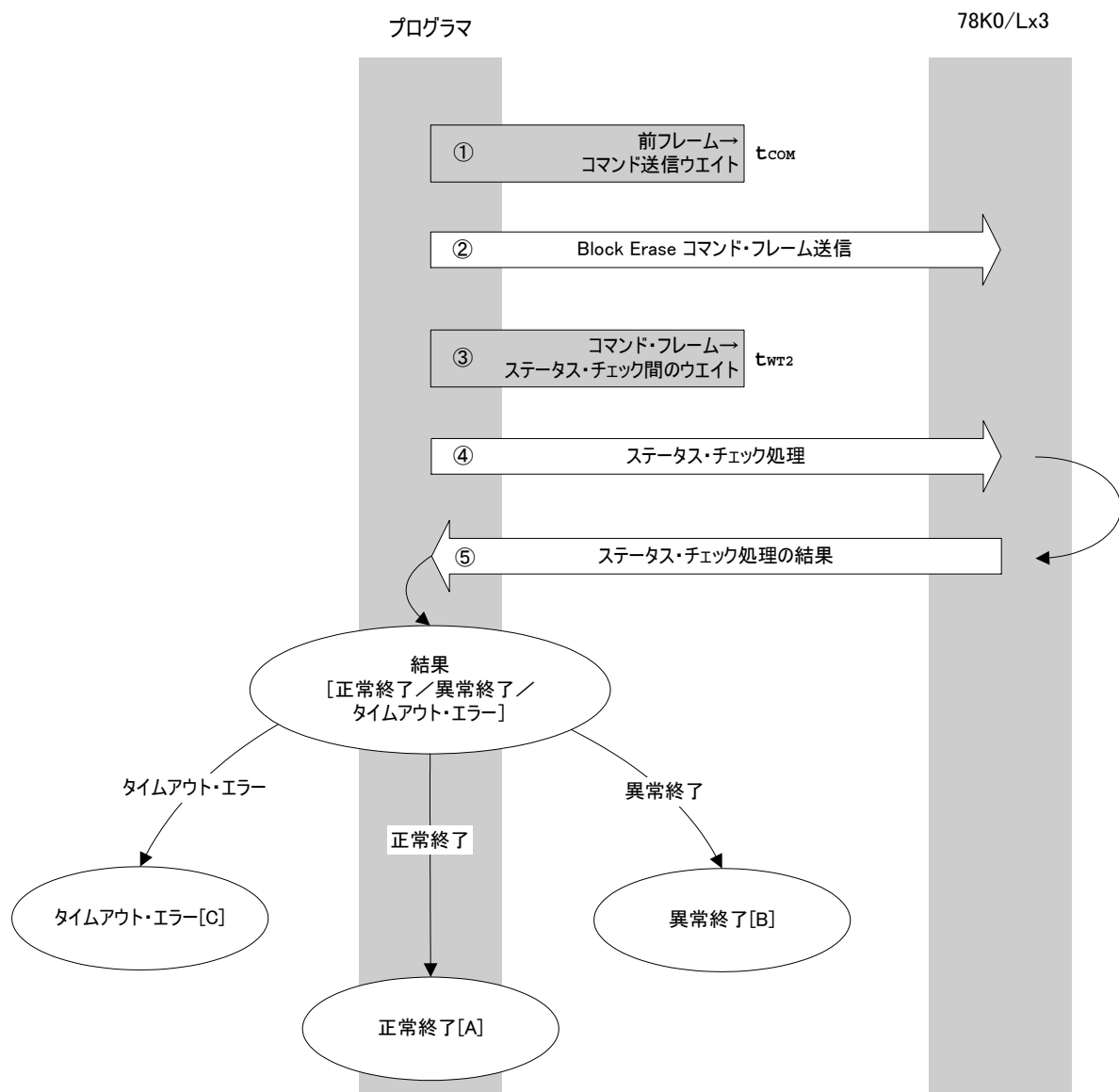
    rc = fl_csi_getstatus(tWT1_MAX);          // get status frame
    // switch(rc) {
    //
    //      case    FLC_NO_ERR:      return rc;      break; // case [A]
    //      case    FLC_DFTO_ERR:    return rc;      break; // case [C]
    //      default:                  return rc;      break; // case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

```

7.8 Block Eraseコマンド

7.8.1 処理手順チャート

Block Eraseコマンド処理手順



7.8.2 処理手順説明

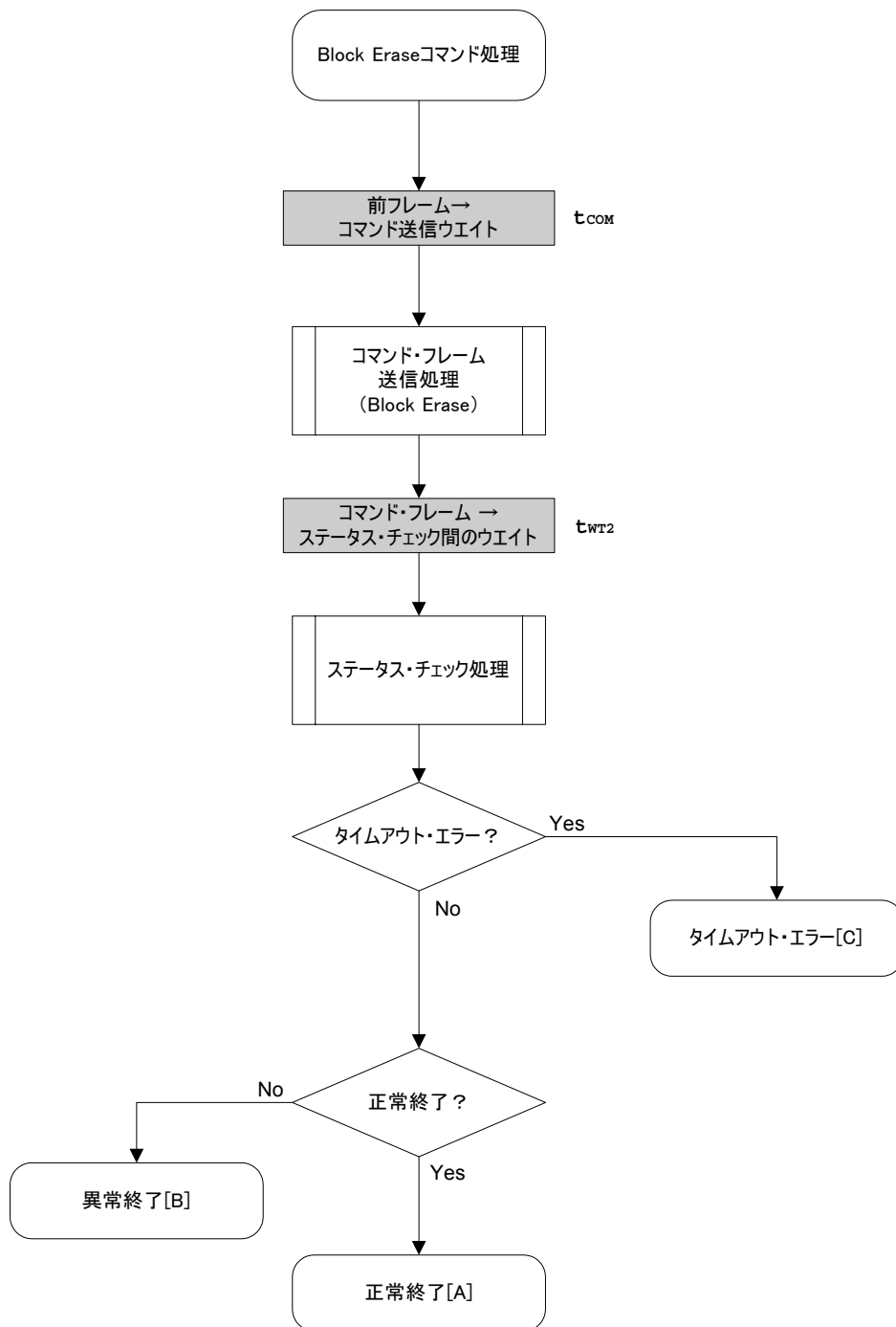
直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 コマンド・フレーム送信処理により，Block Eraseコマンドを送信します。
 ステータス・フレーム取得までのウェイトをします（ウェイト時間 t_{WT2} ）。
 ステータス・チェック処理により，ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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

7.8.3 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	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。
	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，「書き込み禁止」，「ブロック消去禁止」，「チップ消去禁止」のいずれかになっています。または，指定範囲にブート領域が含まれており，セキュリティ設定で，「ブート・ブロック書き換え禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（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。

7.8.4 フロー・チャート



7.8.5 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_CSI);              // wait before sending command frame

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

    fl_wait(wt2);

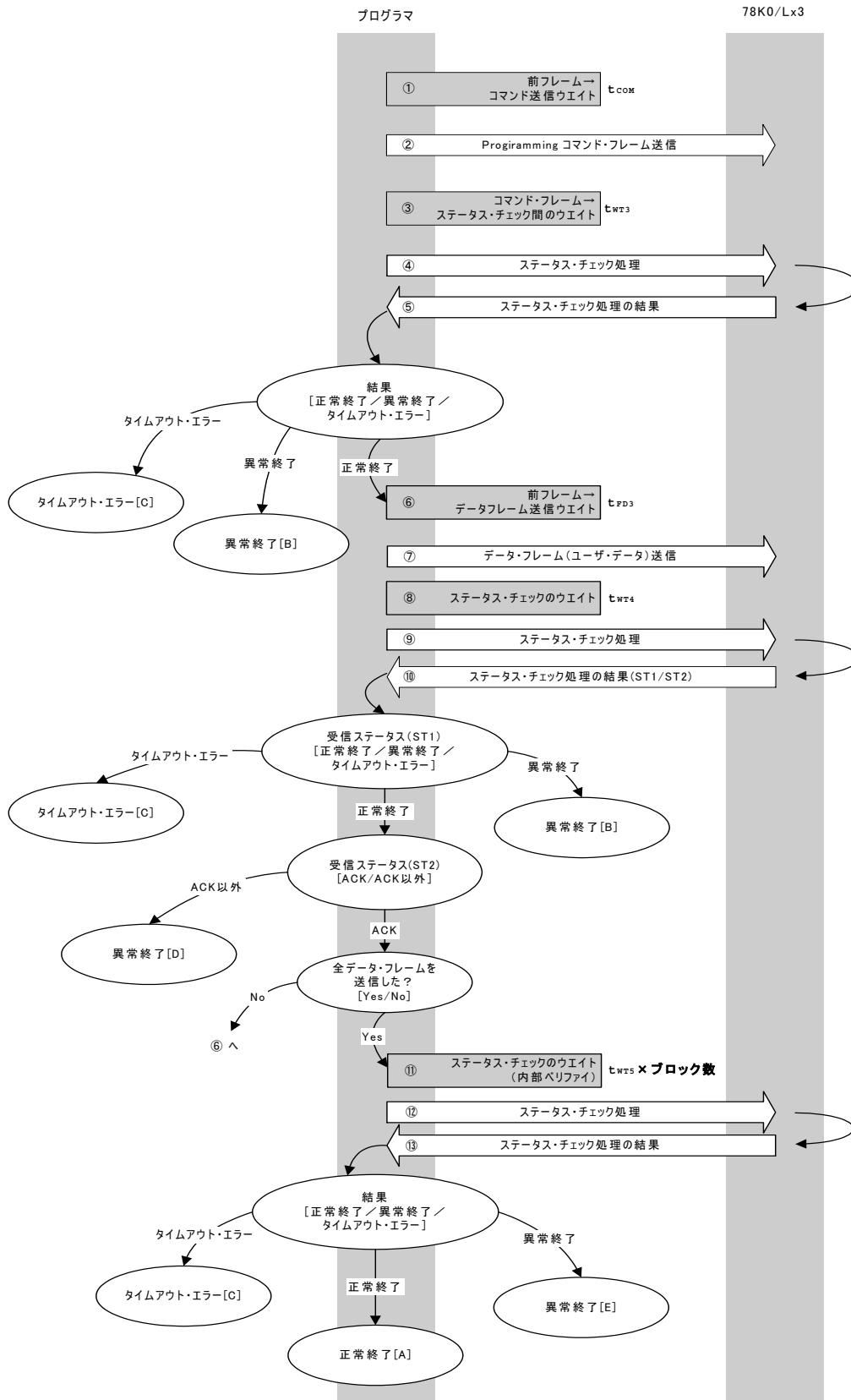
    rc = fl_csi_getstatus(wt2_max); // get status frame
    // switch(rc) {
    //
    //      case    FLC_NO_ERR:    return rc;    break; // case [A]
    //      case    FLC_DFTO_ERR:  return rc;    break; // case [C]
    //      default:                return rc;    break; // case [B]
    // }
    return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

```

7.9 Programmingコマンド

7.9.1 処理手順チャート

Programmingコマンド処理手順



7.9.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT3})。
 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。
 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 : 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。

データ・フレーム送信前のウェイトを行います (ウェイト時間 t_{FD3})。
 データ・フレーム送信処理により, 78K0/Lx3のフラッシュROMに書き込むユーザ・データを送信します。
 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ェイトをします (ウェイト時間 t_{WT4})。
 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。
 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 へ, まだ送信するユーザ・データがある場合はから実行します。

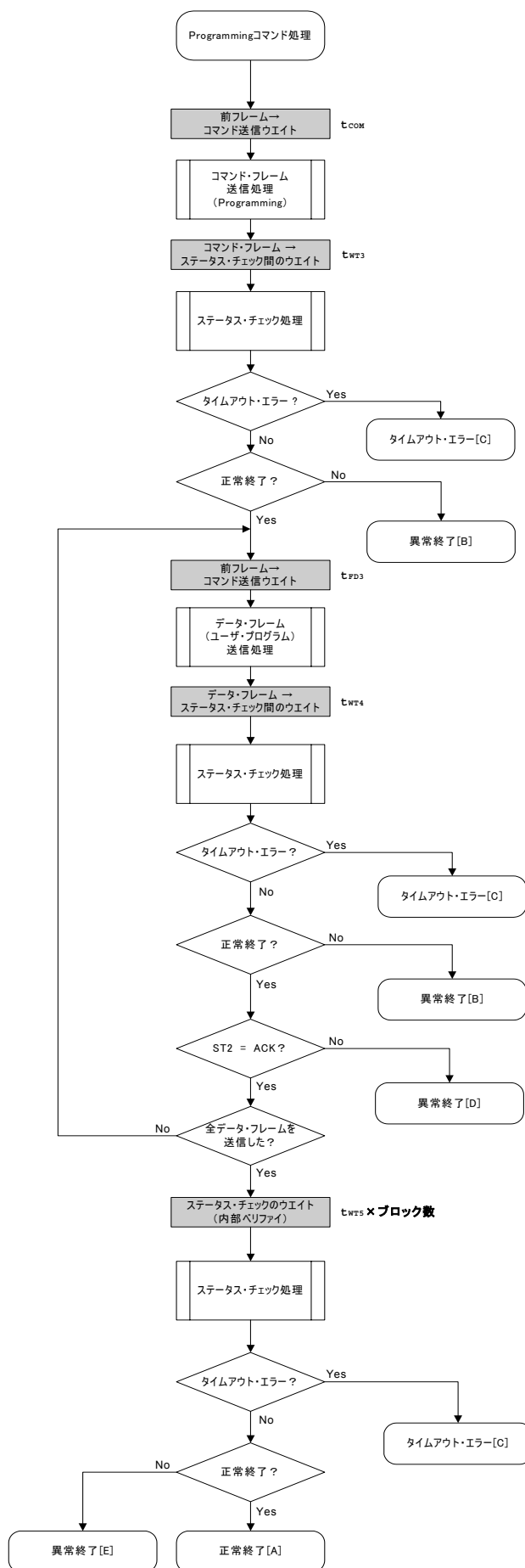
ステータス・チェック処理までのウェイトをします (ウェイト時間 $t_{WT5} \times \text{ブロック数}$)。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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

7.9.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または 1K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム、またはデータ・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、「書き込み禁止」になっています。または、指定範囲にブート領域が含まれており、セキュリティ設定で、「ブート・ブロック書き換え禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。または、処理中にステータス・コマンド以外を受信しました。
	MR10 エラー	1AH	書き込み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。
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。

7.9.4 フロー・チャート



7.9.5 サンプル・プログラム

Programmingコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Write command (CSI)
/*
/*
/* [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_CSI);
    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_CSI_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;

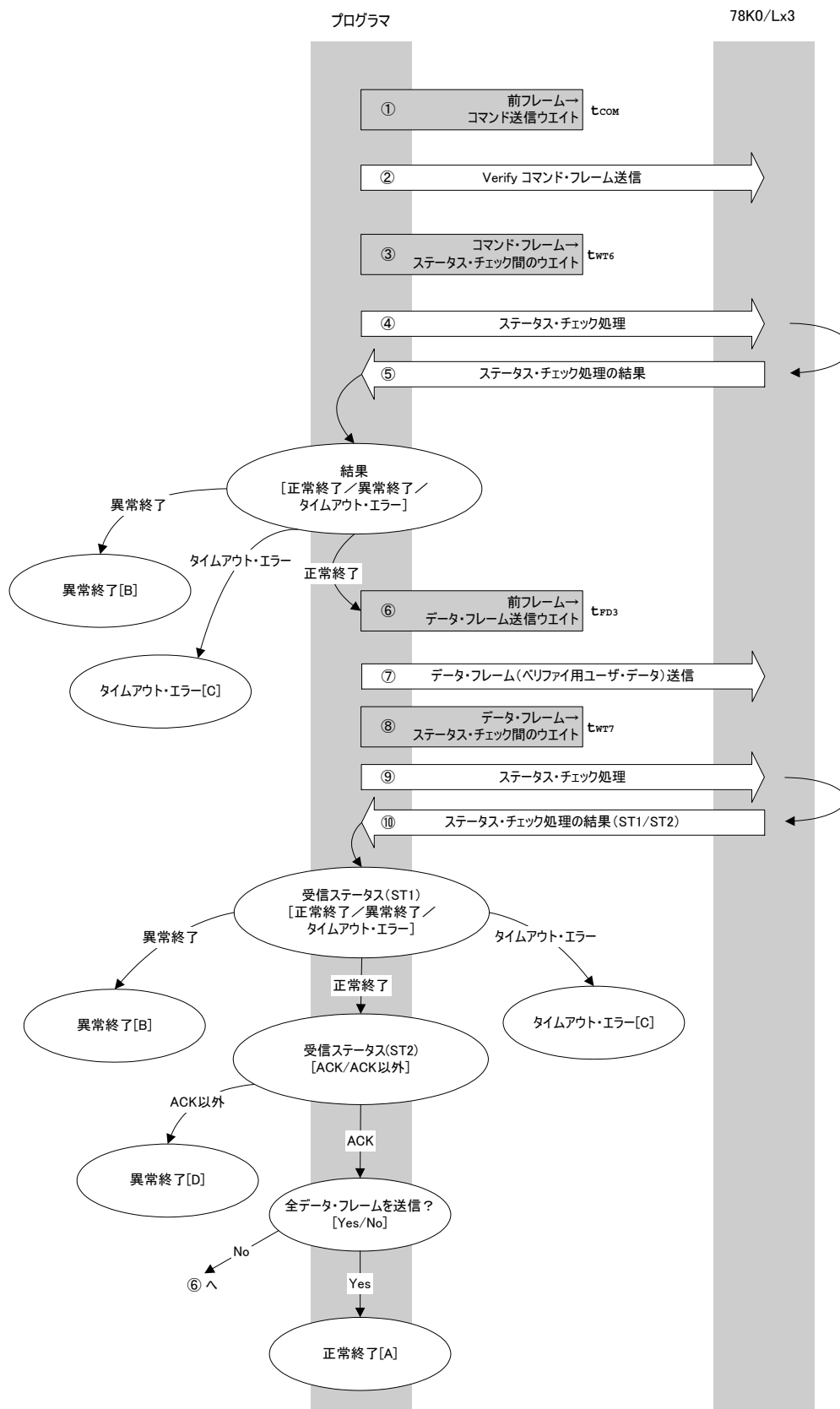
}

```

7. 10 Verifyコマンド

7. 10. 1 処理手順チャート

Verifyコマンド処理手順



7.10.2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{COM} ）。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします（ウェイト時間 t_{WT6} ）。
 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。
 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 : 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。

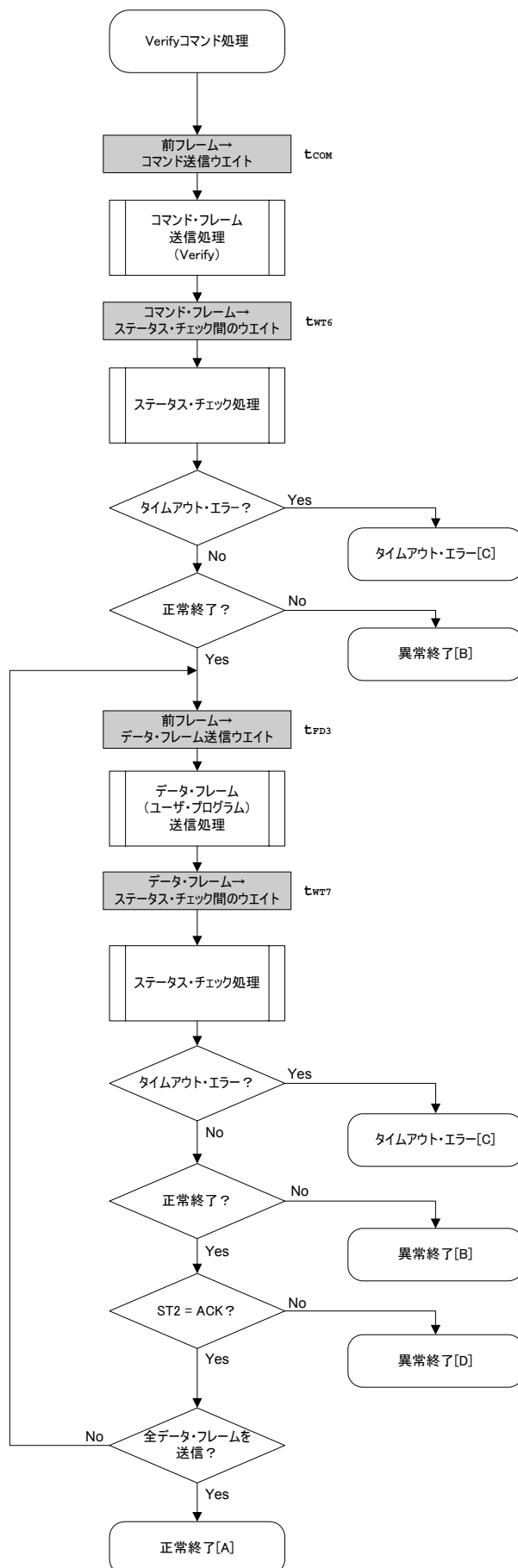
直前のフレームからデータ・フレーム送信までのウェイトをします
 （ウェイト時間 t_{FD3} （CSI））。
 データ・フレーム送信処理により、ペリファイ用のユーザ・データを送信します。
 データ・フレーム送信からステータス・チェック処理までのウェイトをします
 （ウェイト時間 t_{WT7} ）。
 ステータス・チェック処理にてステータス・フレームを取得します。
 ステータス・チェック処理の結果（受信ステータス（ST1/ST2））に応じて次の処理を行います（処理手順チャートやフロー・チャートも参照してください）。

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
 から実行します。

7.10.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答（ACK）	06H	コマンドが正常に実行され、ペ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 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または 1K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレーム、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）。または、処理中にステータス・コマンド以外を受信しました。
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。
異常終了 [D]	ペリファイ・エラー	0FH	ペリファイに失敗しました。または、その他のエラーが発生しました。

7.10.4 フロー・チャート



7.10.5 サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Verify command (CSI)
/*
/*
/*****
/* [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_CSI);
    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]
}

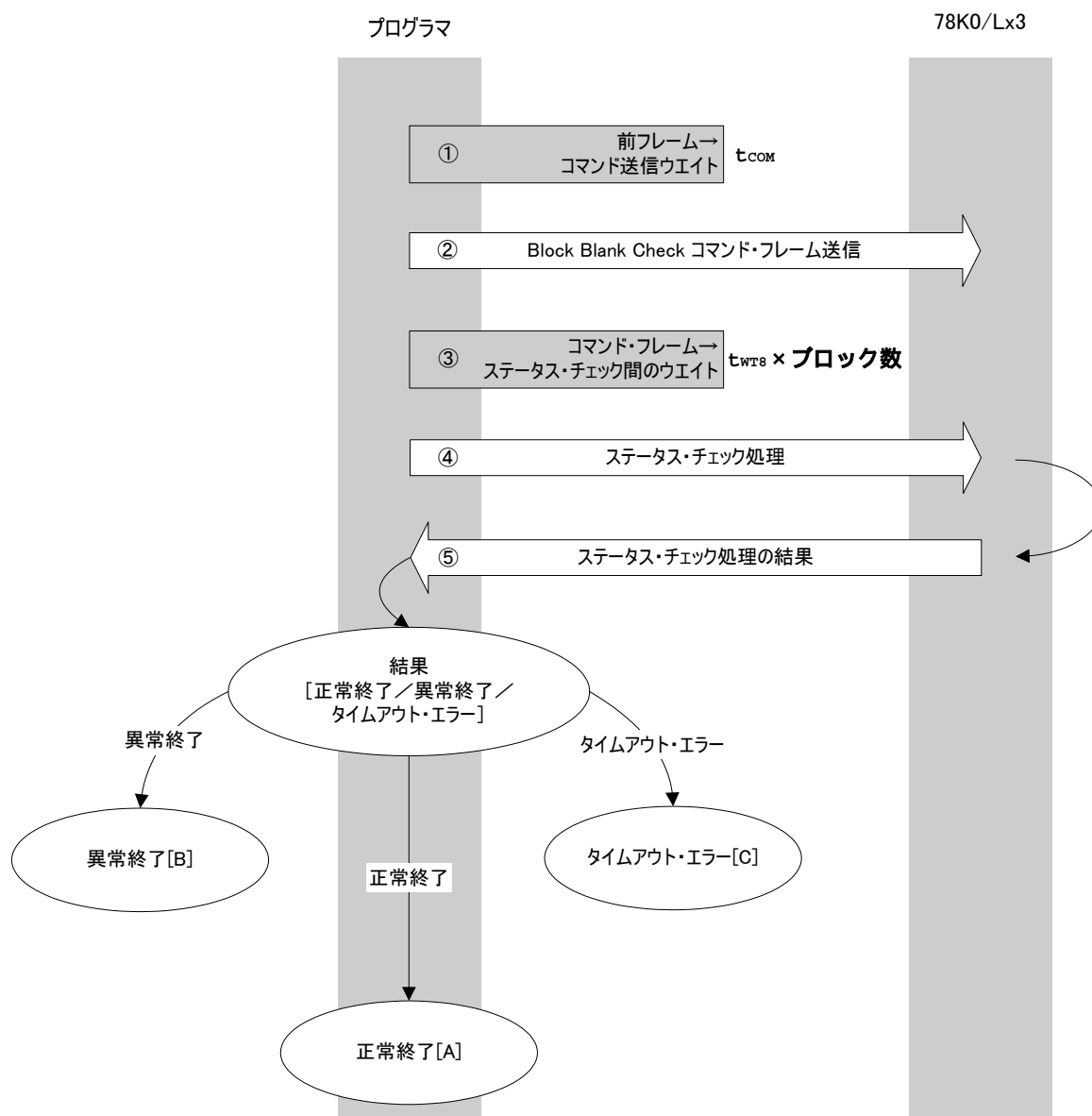
if (is_end)                        // send all user data ?
    break;                        // yes
//continue;

}
return FLC_NO_ERR;                // case [A]
}
```

7. 11 Block Blank Checkコマンド

7. 11. 1 処理手順チャート

Block Blank Checkコマンド処理手順



7. 11. 2 処理手順説明

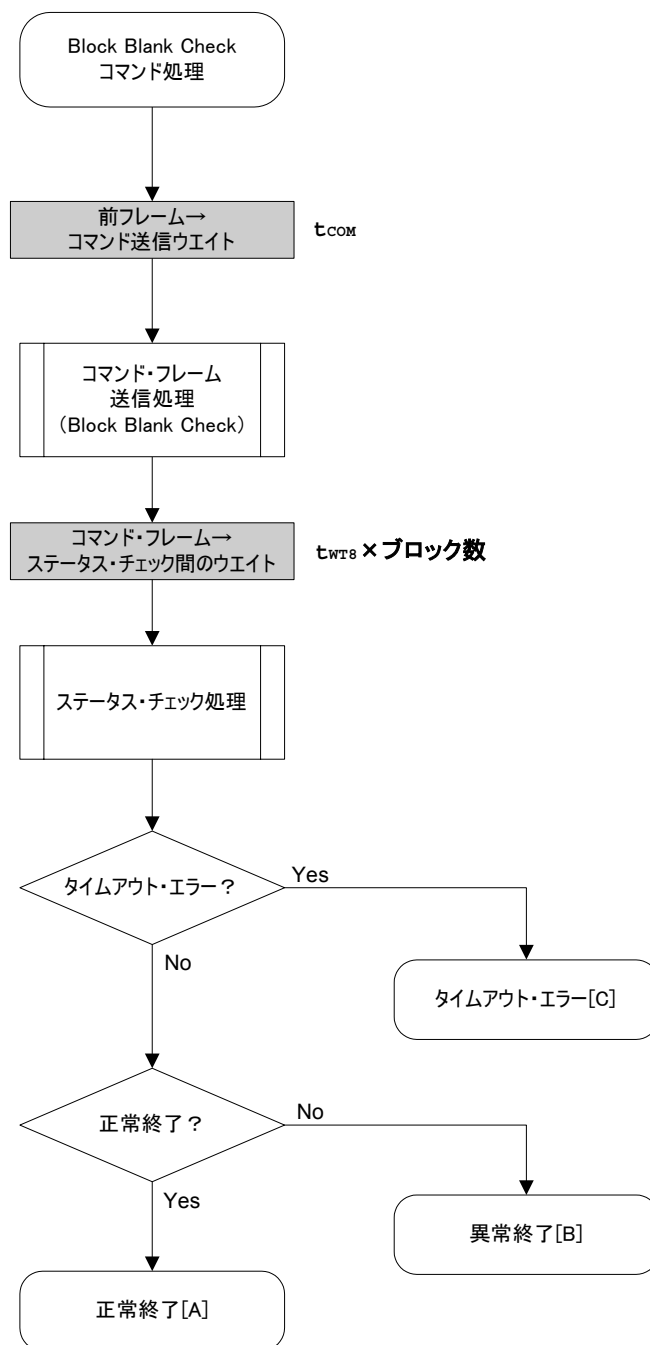
直前のフレームからコマンド送信までのウェイトをします（ウェイト時間 t_{com} ）。
 コマンド・フレーム送信処理により，Block Blank Checkコマンドを送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします
 （ウェイト時間 $t_{WT8} \times \text{ブロック数}$ ）。
 ステータス・チェック処理により，ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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。

7. 11. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され，指定したブロックすべてがブランクであることを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または1 K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です（データ長 (LEN) 不正，ETX なしなど）。または，処理中にステータス・コマンド以外を受信しました。
	MRG11 エラー	1BH	指定したブロックのフラッシュ・メモリがブランクではありません。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

7.11.4 フロー・チャート



7.11.5 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_CSI);                      // 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);

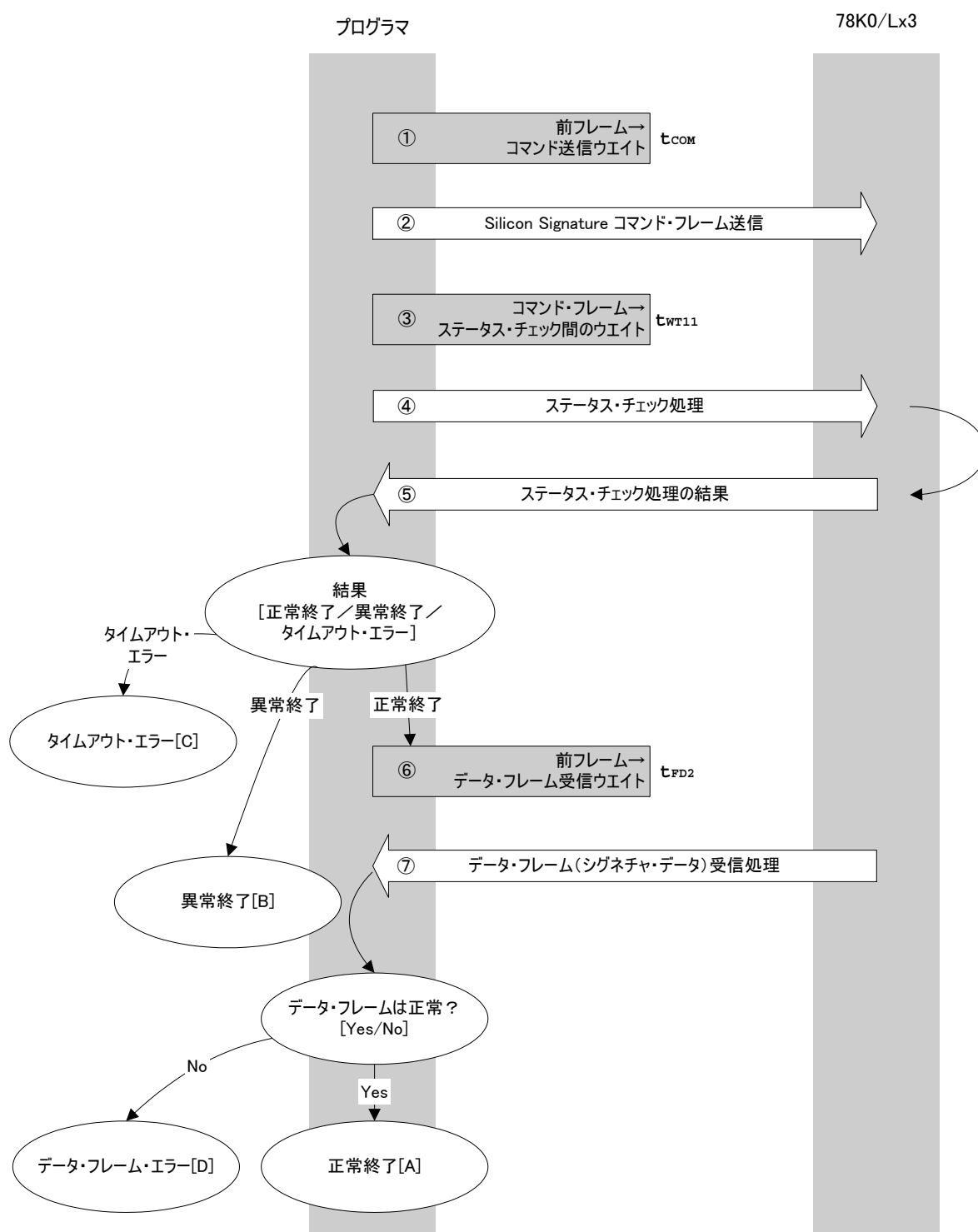
    rc = fl_csi_getstatus(tWT8_MAX * block_num); // get status frame
// switch(rc) {
//
//      case    FLC_NO_ERR:      return rc;      break; // case [A]
//      case    FLC_DFTO_ERR:    return rc;      break; // case [C]
//      default:                  return rc;      break; // case [B]
// }
    return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

```

7. 12 Silicon Signatureコマンド

7. 12. 1 処理手順チャート

Silicon Signatureコマンド処理手順



7. 12. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT11})。
 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。
 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 : 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。

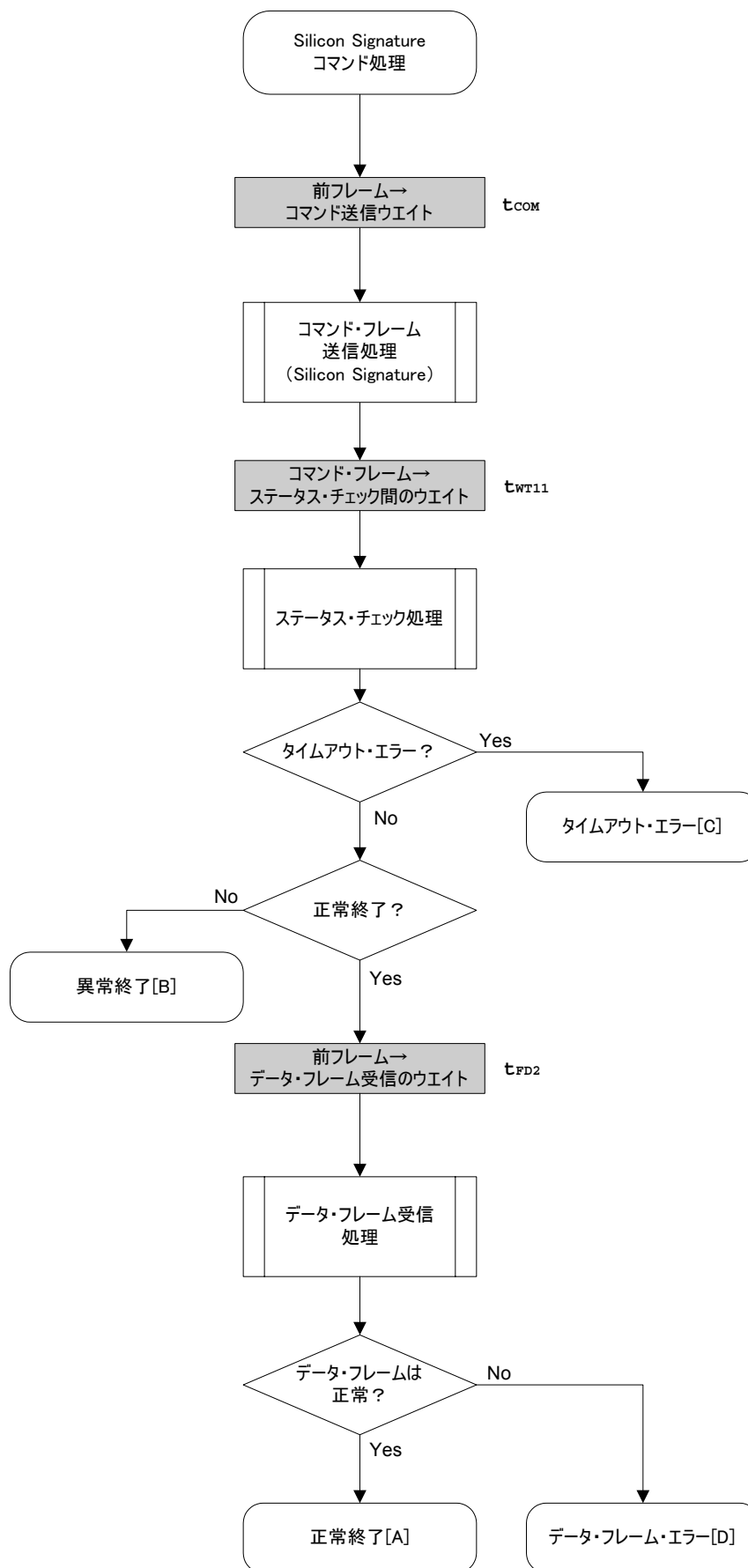
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{FD2})。
 受信したデータ・フレーム (シリコン・シグネチャ・データ) をチェックします。

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。

7. 12. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され, シリコン・シグネチャを取得できたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。または, 処理中にステータス・コマンド以外を受信しました。
	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]		-	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。

7.12.4 フロー・チャート



7. 12. 5 サンプル・プログラム

Silicon Signatureコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM_CSI);                // 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)

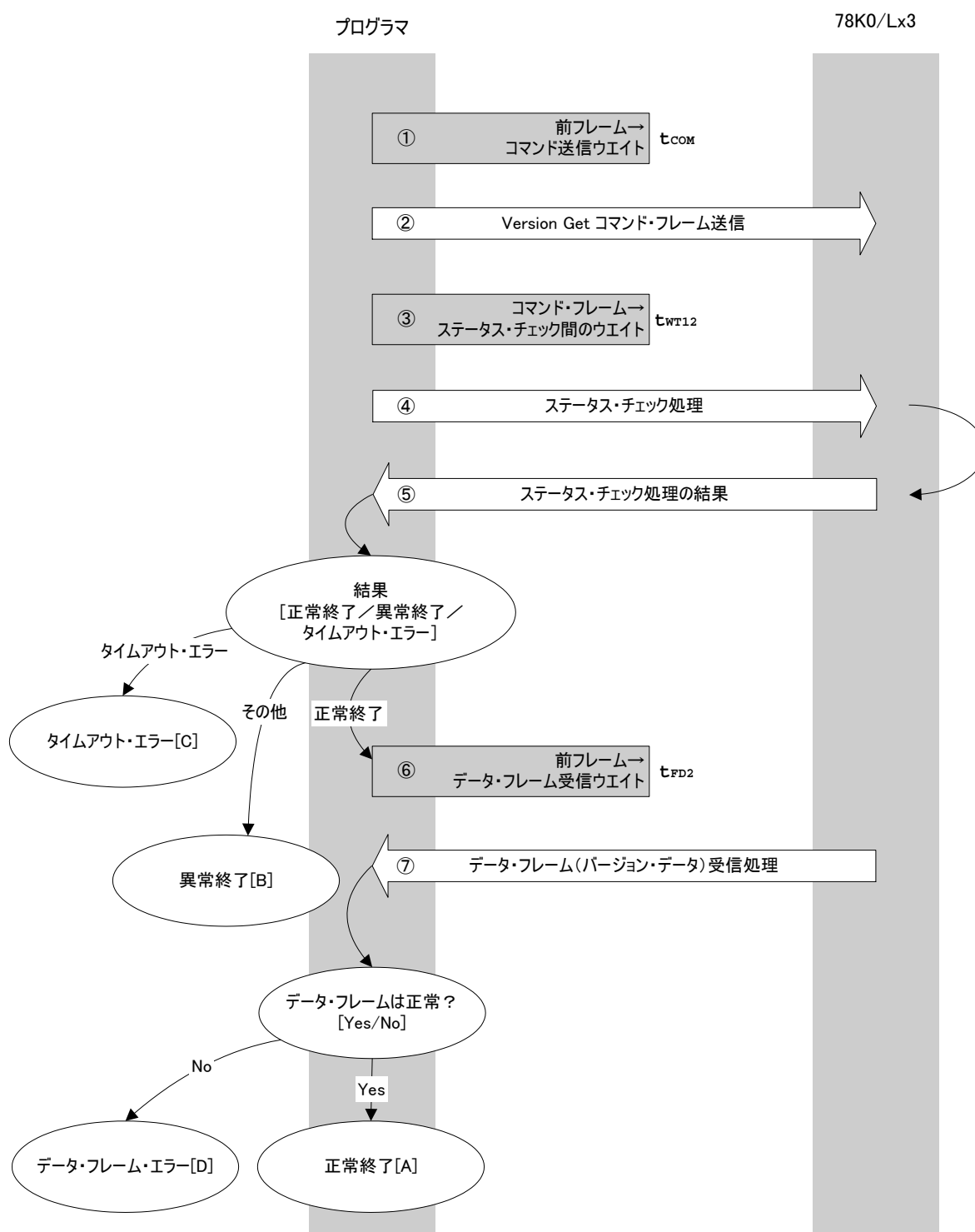
    if (rc){                            // if no error,
        return rc;                        // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                    // copy Signature data
    return rc;                            // case [A]
}

```

7. 13 Version Getコマンド

7. 13. 1 処理手順チャート

Version Getコマンド処理手順



7. 13. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT12})。
 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。
 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 : 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。

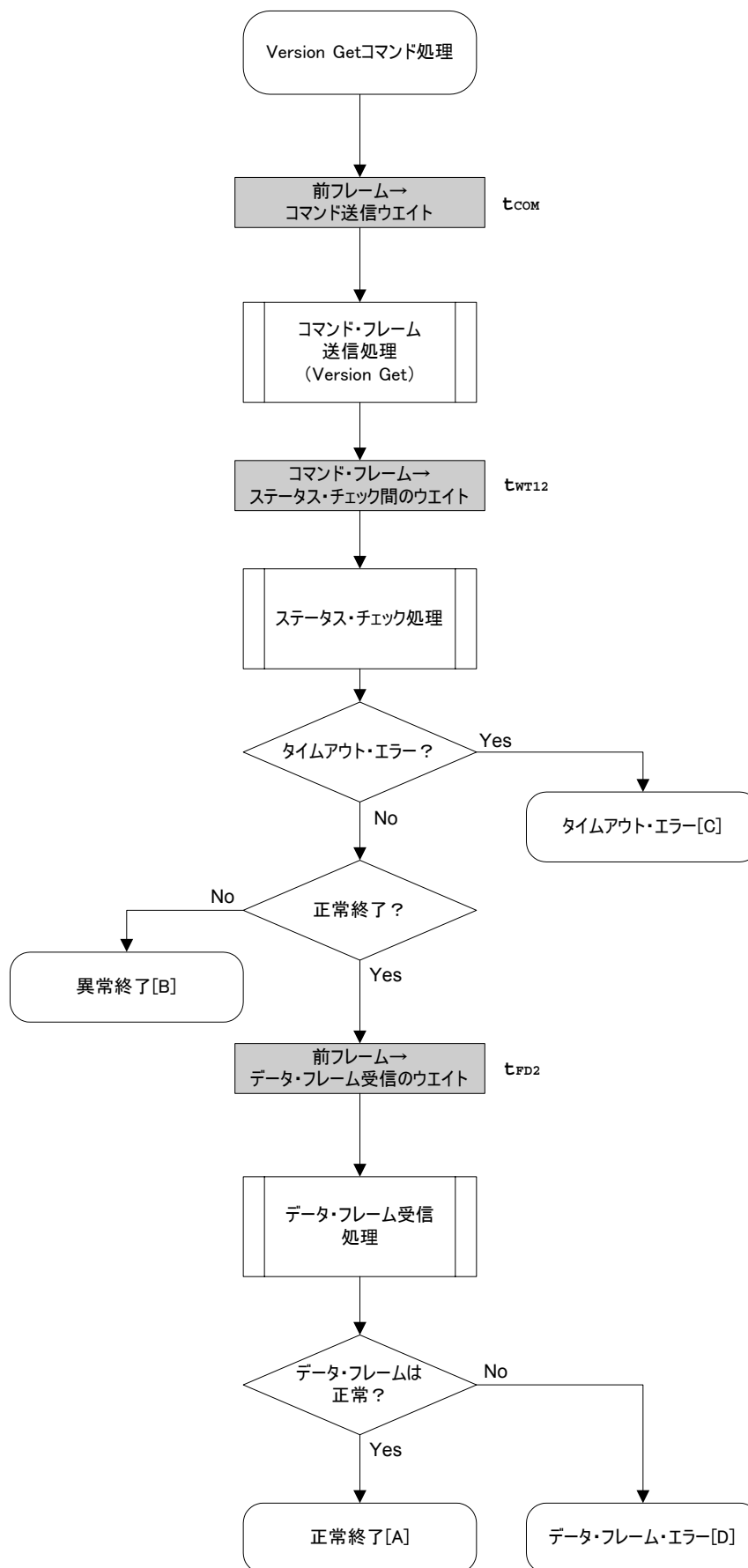
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{FD2})。
 受信したデータ・フレーム (バージョン・データ) をチェックします。

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。

7. 13. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、バージョン・データを取得できたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。または、処理中にステータス・コマンド以外を受信しました。
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。
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。

7.13.4 フロー・チャート



7.13.5 サンプル・プログラム

Version Getコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM_CSI);                // 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

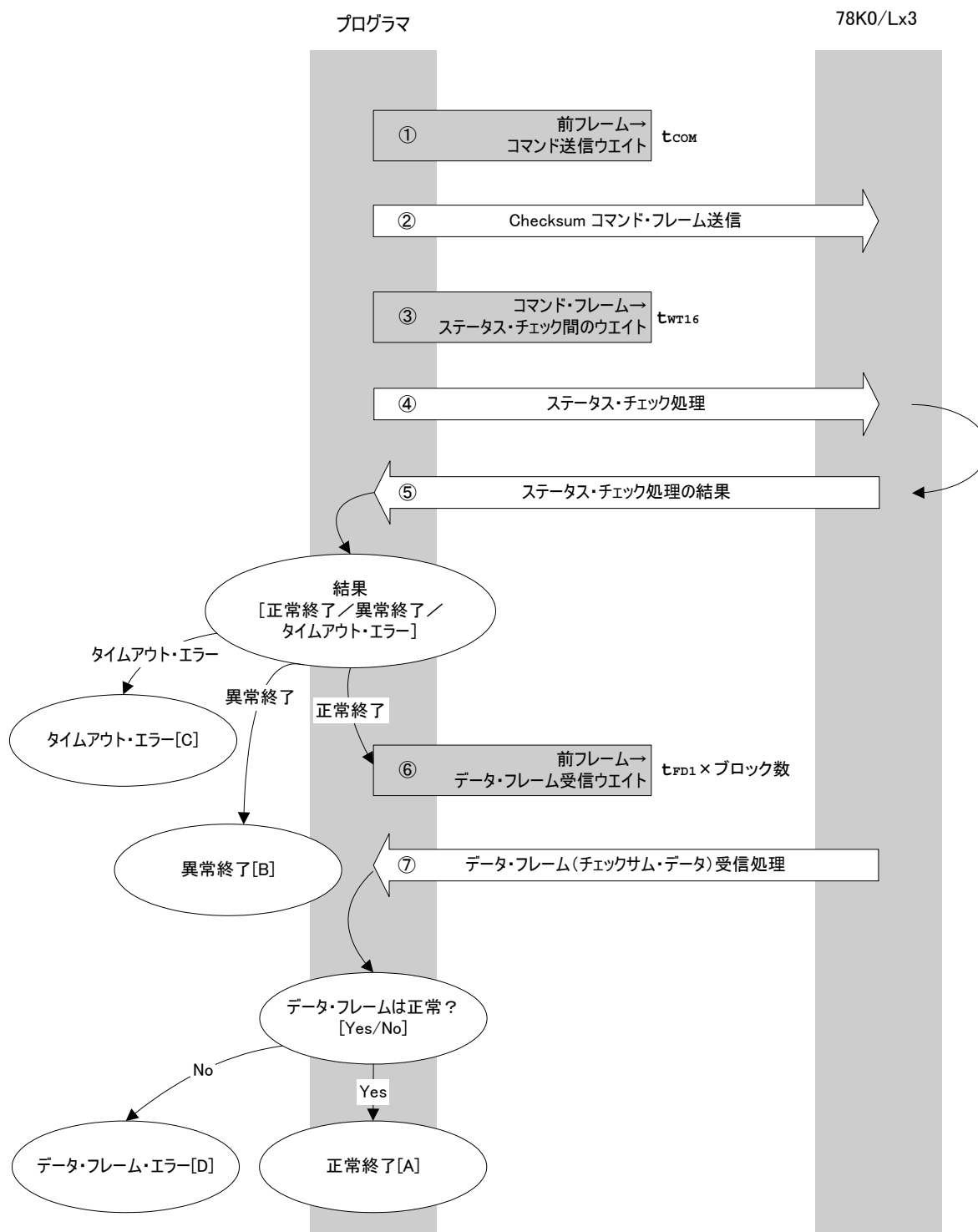
    if (rc){                            // if no error,
        return rc;                      // case [D]
    }
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                          // case [A]
}

```

7. 14 Checksumコマンド

7. 14. 1 処理手順チャート

Checksumコマンド処理手順



7. 14. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT16})。
 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。
 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 : 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。

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 $t_{FD1} \times \text{ブロック数}$)。

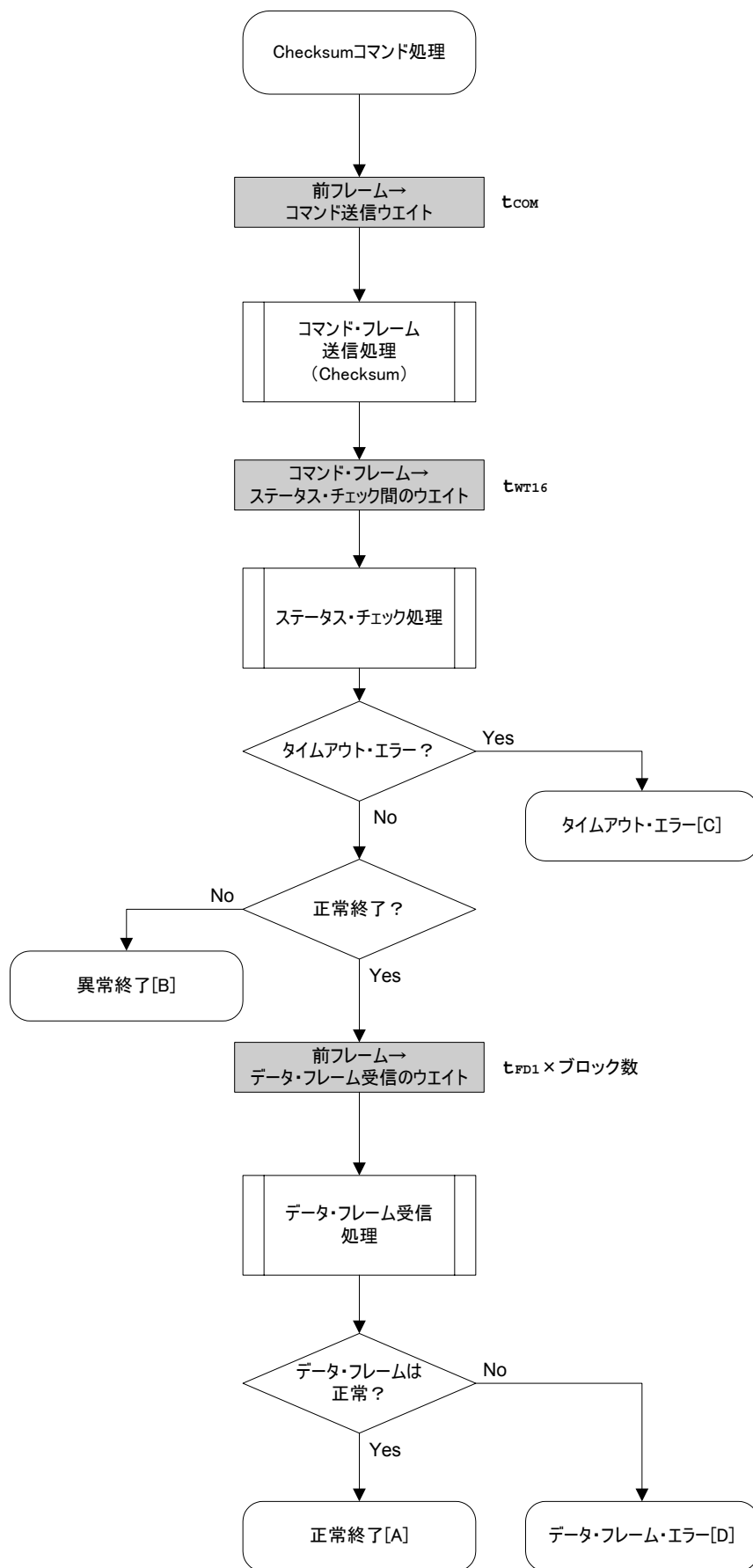
受信したデータ・フレーム (チェックサム・データ) をチェックします。

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。

7. 14. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、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 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。または 1K バイトごとの固定アドレスではありません。
	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。または、処理中にステータス・コマンド以外を受信しました。
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。
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。

7.14.4 フロー・チャート



7.14.5 サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Get checksum command (CSI)
/*
/*
/*****
/* [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_CSI); // 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]
    }

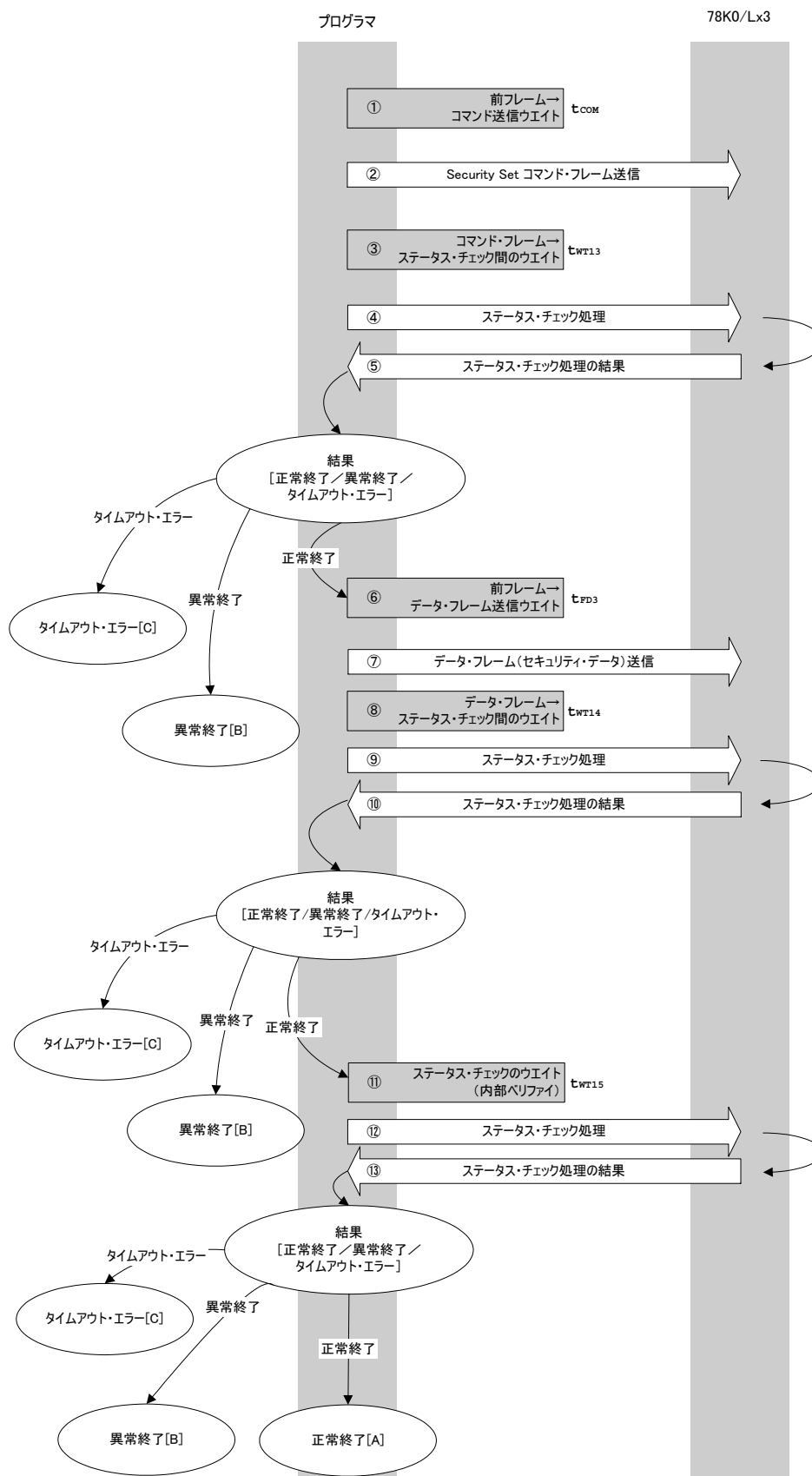
    *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1]; // set SUM data
    return rc; // case [A]
}

```

7. 15 Security Setコマンド

7. 15. 1 処理手順チャート

Security Setコマンド処理手順



7. 15. 2 処理手順説明

直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 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。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT13})。
 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。
 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 : 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。

直前のフレームからデータ・フレーム送信までのウェイトをします (ウェイト時間 t_{FD3})。
 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。
 データ送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT14})。
 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。
 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 : 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。

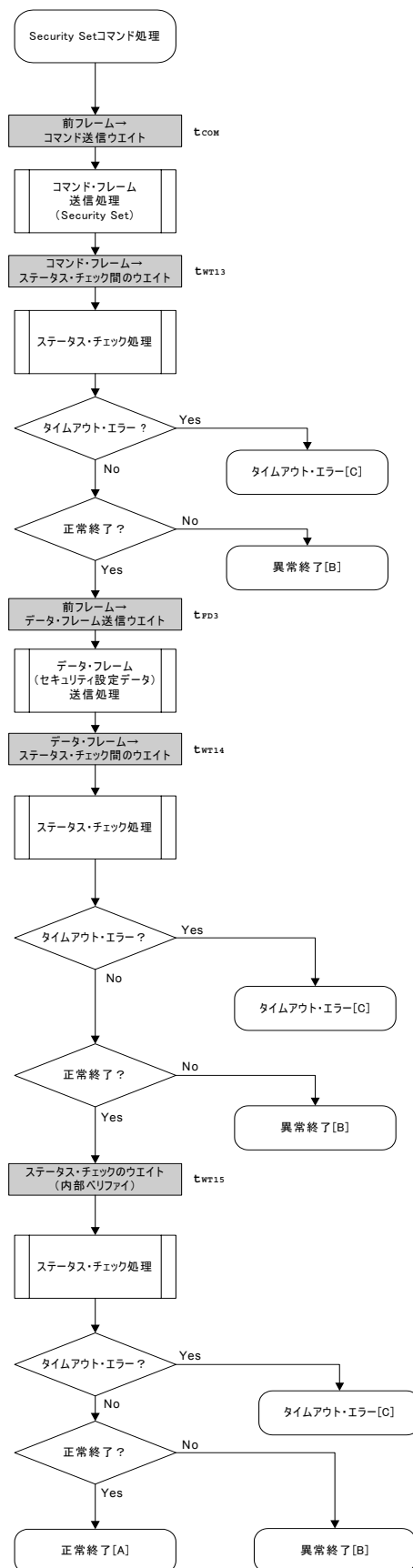
ステータス取得 (内部ベリファイ完了) までのウェイトをします (ウェイト時間 t_{WT15})。
 ステータス・チェック処理により、ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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

7. 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。
	プロテクト・エラー	10H	すでに禁止されているセキュリティ・フラグを許可しようとした。
	MRG10 エラー	1AH	書き込みエラーが発生しました。
	MRG11 エラー	1BH	内部ベリファイ・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。
	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）。または，処理中にステータス・コマンド以外を受信しました。
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。

7.15.4 フロー・チャート



7. 15. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Set security flag command (CSI)
/*
/*****
/* [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_CSI);              // 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8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/Lx3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/Lx3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。

8.1 フラッシュ・メモリ・プログラミング・モード設定時間

($T_A = -40 \sim +85$, $V_{DD} = AV_{REF}$, $V_{SS} = AV_{SS} = 0\text{ V}$)

項 目	略 号	MIN.	TYP.	MAX.
$V_{DD} \uparrow$ to FLMD0 $\uparrow$	t_{DP}	1 ms		
FLMD0 $\uparrow$ to $\overline{\text{RESET}}$ $\uparrow$	t_{PR}	2 ms		
Count start time from $\overline{\text{RESET}}$ $\uparrow$ to FLMD0 ^{注1}	t_{RP}	7.42 ms		
Count finish time from $\overline{\text{RESET}}$ $\uparrow$ to FLMD0 ^{注1}	t_{RPE}			33.8 ms
FLMD0 counter high level width / low level width	t_{PW}	10 μs		100 μs
Wait for Reset command (CSI) ^{注2}	t_{RC}	55.56 ms		
Wait for low level data1 (UART) ^{注2}	X1 clock	t_{R1}	$55.62\text{ ms} + 2^{16}/f_x$	
	External main system clock, high speed internal oscillator		55.62 ms	
Wait for low level data2 (UART) ^{注2}	t_{12}	3.75 ms		
Wait for Read command (UART) ^{注2}	t_{2C}	3.75 ms		
Low level data1 / data2 width (UART)	t_{L1} , t_{L2}		注3	
FLMD0 counter rise time / fall time	t_R , t_F			1 μs

注1. FLMD0パルスの入力タイミングは、(7.42 ms + 33.8 ms) / 2を標準値として推奨します。

2. タイムアウト時間は3秒以上になるようプログラマを設定してください。
3. ロウ・レベル幅は、9600 bps時の00Hデータ幅と同じです。

備考 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。

8.2 プログラミング特性

($T_A = -40 \sim +85$, $V_{DD} = AV_{REF}$, $V_{SS} = AV_{SS} = 0\text{ V}$)

ウェイト	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	$29.63\ \mu\text{s}$	
			UART	$26.25\ \mu\text{s}$	
	データ・フレーム送信	t_{DT}	CSI	$23.13\ \mu\text{s}$	
			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	$75.63\ \mu\text{s}$	
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	$13502.75\ \mu\text{s}$	
			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	$86.63\ \mu\text{s}$	
	バージョン・データ			$61.00\ \mu\text{s}$	
	—		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	$30.75\ \mu\text{s}$	
			UART	$29.63\ \mu\text{s}$	
ステータス・フレーム送信～コマンド・フレーム受信	—	t_{COM}	CSI	$36.00\ \mu\text{s}$	
			UART	$34.88\ \mu\text{s}$ 注1	

注1. プログラマは連続受信可にしておいてください。また、プログラマのタイムアウト時間は3 s以上にしてください。

2. 1ブロックの場合の時間

備考 ウェイトには、次のような定義があります。

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

78K0/Lx3は、直前の通信完了後、MIN.後から次の通信が可能となります。

プログラマは、直前の通信完了後、MIN.～MAX.時間内に次のデータの送信を行ってください。

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

78K0/Lx3は、直前の通信完了後、MIN.後から次の通信が可能となります。

プログラマは、直前の通信完了後、MIN.～MAX.時間内に次のデータの受信を行ってください。

CSI通信の場合、プログラマは、MIN.時間経過後、Statusコマンドを発行してください。ACKが返ってこない場合はステータス・チェックを繰り返さず、エラー処理(タイムアウト処理など)を行ってください。

コマンド	略号	シリアル I/F	MIN. (μ s)	MAX. (μ s)
Reset	t_{WT0}	CSI	55.68	
		UART	注1	
Chip Erase	t_{WT1}	—	92770.88 + 11788.875 × 全ブロック数	945798.50 + 165043.25 × 全ブロック数
Block Erase	t_{WT2} 注2	—	316.75 + 13522 × 同時選択消去の実行回数 + 11788.875 × 消去するブロック数	316.75 + 190196 × 同時選択消去の実行回数 + 164444.5 × 消去するブロック数
Programming	t_{WT3}	CSI	283.38	
		UART	注1	
	t_{WT4} 注3	—	12781.25	140019.13
	t_{WT5} 注4, 7	CSI	24286.88	24393.63
		UART	注5	24393.50
	t_{WT5} 注4, 8	CSI	103518.00	776321.25
		UART	注5	776321.25
Verify	t_{WT6}	CSI	184.50	
		UART	注1	
	t_{WT7} 注3	CSI	2848.88	
		UART	注6	
Block Blank Check	t_{WT8} 注4	CSI	11455.50	13746.63
		UART	注1	13746.63
Oscillating Frequency Set	t_{WT9}	CSI	313.38	
		UART	注1	
Silicon Signature	t_{WT11}	CSI	253.38	
		UART	注1	
Version Get	t_{WT12}	CSI	82.38	
		UART	注1	
Security Set	t_{WT13}	CSI	165.88	
		UART	注1	
	t_{WT14}	—	27858.38	375030.00
	t_{WT15}	CSI	51405.50	382672.38
		UART	注1	382672.38
Checksum	t_{WT16}	CSI	163.25	
		UART	注1	

注1. プログラマはコマンド・フレーム送信前に受信可に設定しておいてください。また、プログラマのタイムアウト時間は3 s以上にしてください。

2. 同時選択消去の実行回数の求め方に関しては、次ページ以降の補足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

5. プログラマは連続受信可にしておいてください。また、プログラマのタイムアウト時間は3 s以上にしてください。

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

7. ブロック0以外

8. ブロック0

(備考は次ページにあります)

備考 ウェイトには、次のような定義があります。

< tWT0 – tWT9, tWT11 – tWT16 >

- ・ MIN.とMAX.の規定がある項目の場合

78K0/Lx3はMIN.～MAX.時間内に各コマンド処理を終了します。MAX.時間経過しても78K0/Lx3の処理が終了しない場合は、エラー処理（タイムアウト処理など）を行ってください。

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

- ・ MIN.の規定のみの項目の場合

CSI通信の場合、プログラマは、MIN.時間経過後、Statusコマンドを発行してください。ACKが返ってこない場合はステータス・チェックを繰り返さず、エラー処理（タイムアウト処理など）を行ってください。

補足 Block Eraseコマンドにおける同時選択消去について

78K0/Lx3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ェイト時間は、“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）を算出する必要があります。

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】

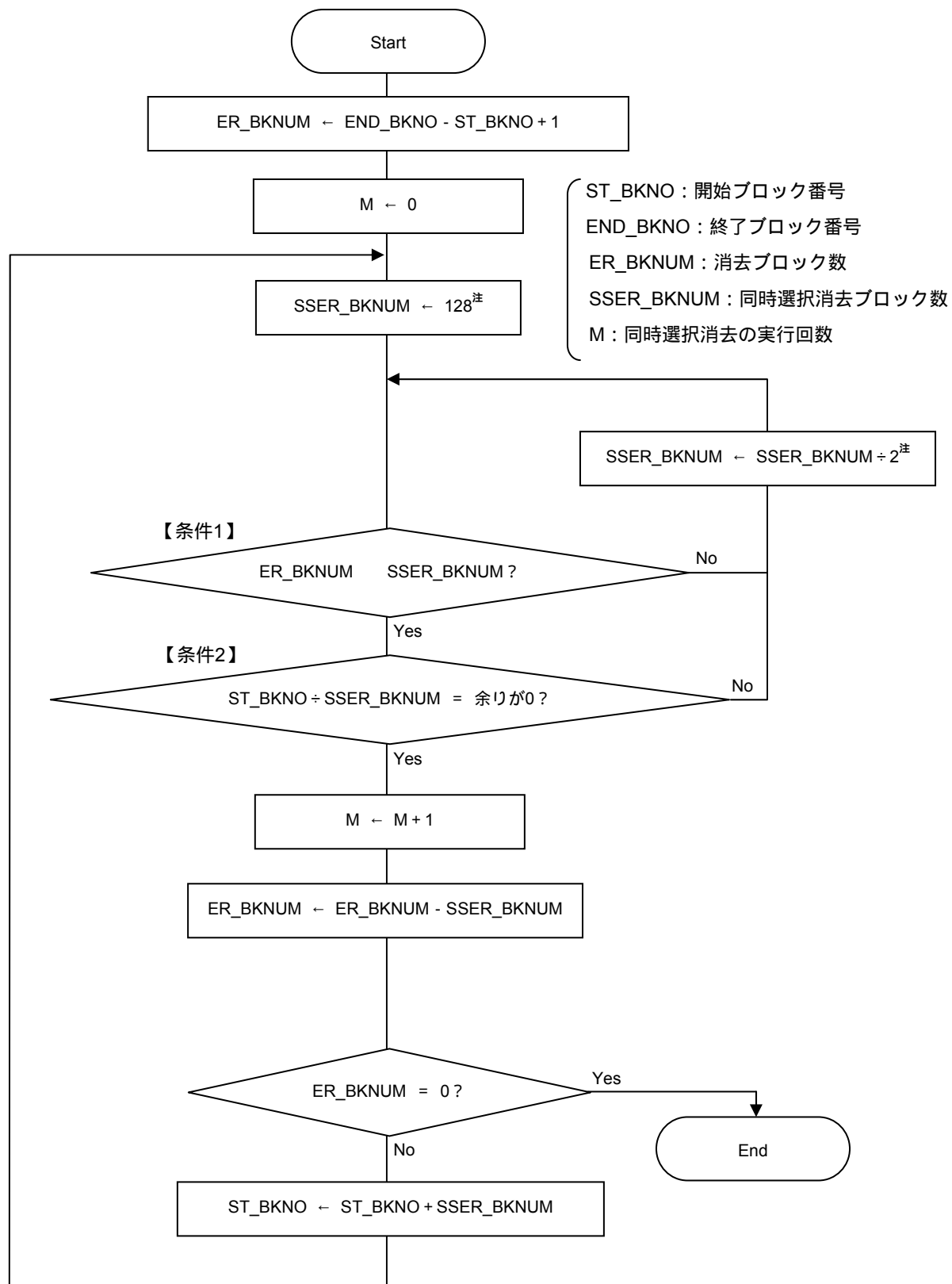
開始ブロック番号 ÷ 同時選択消去ブロック数 = 余りが0

【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）

最初の開始ブロック番号は1で、消去ブロック数が127であることから、【条件1】を満たす数値は、1, 2, 4, 8, 16, 32, 64, 128となります。

さらに【条件2】を満たす数値は、1となり、【条件3】を満たす数値は“1”であるため、同時選択消去ブロック数は“1”となることから、ブロック1のみ消去します。

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。

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。

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。

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。

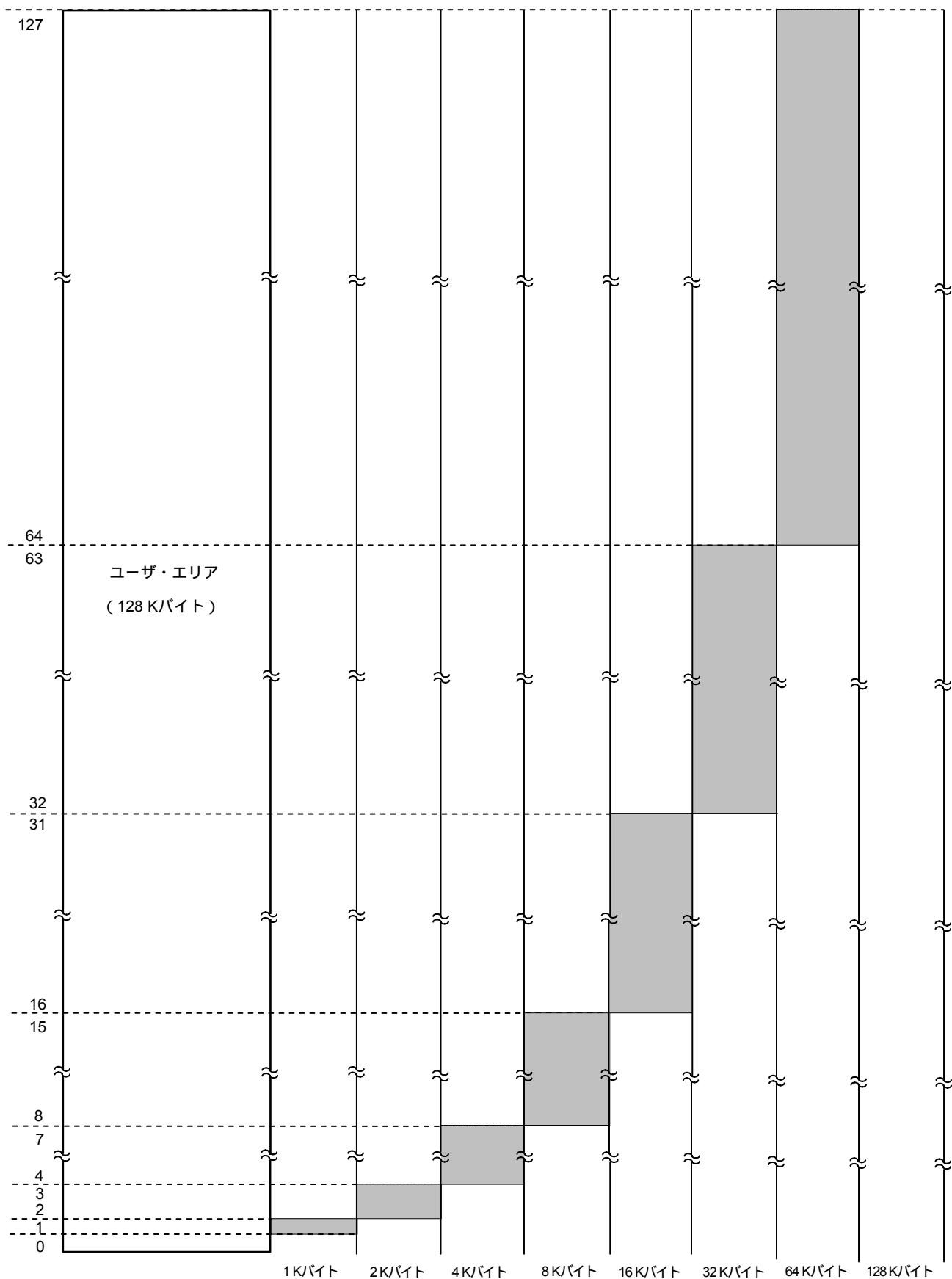
ブロック32～63を消去すると、次の開始ブロック番号は64で、消去ブロック数が64であることから、【条件1】を満たす数値は、1, 2, 4, 8, 16, 32, 64となります。

さらに【条件2】を満たす数値は、1, 2, 4, 8, 16, 32, 64となり、【条件3】を満たす数値は“64”であるため、同時選択消去ブロック数は“64”となることから、ブロック64～127を消去します。

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

同時選択消去実行時のブロック構成（ブロック1～127を消去する場合）

<ブロック番号>



<同時選択消去可能なブロックの範囲>

例2 ブロック5～10を消去する場合（N（消去するブロック数）= 6）

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。

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。

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。

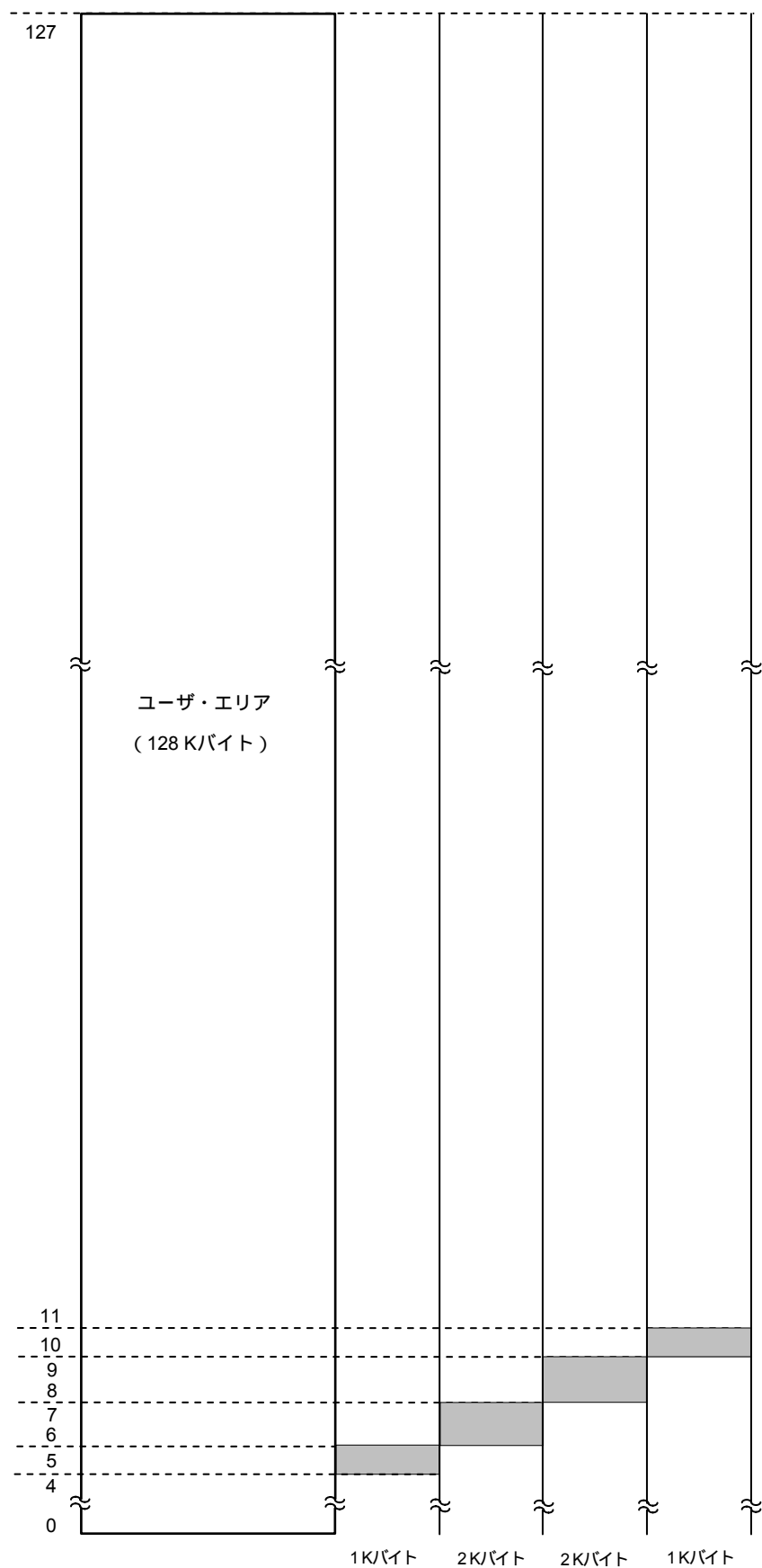
さらに【条件2】を満たす数値は、1, 2となり、【条件3】を満たす数値は“2”であるため、同時選択消去ブロック数は“2”となることから、ブロック8～9を消去する。

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）

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。

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。

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。

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。

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。

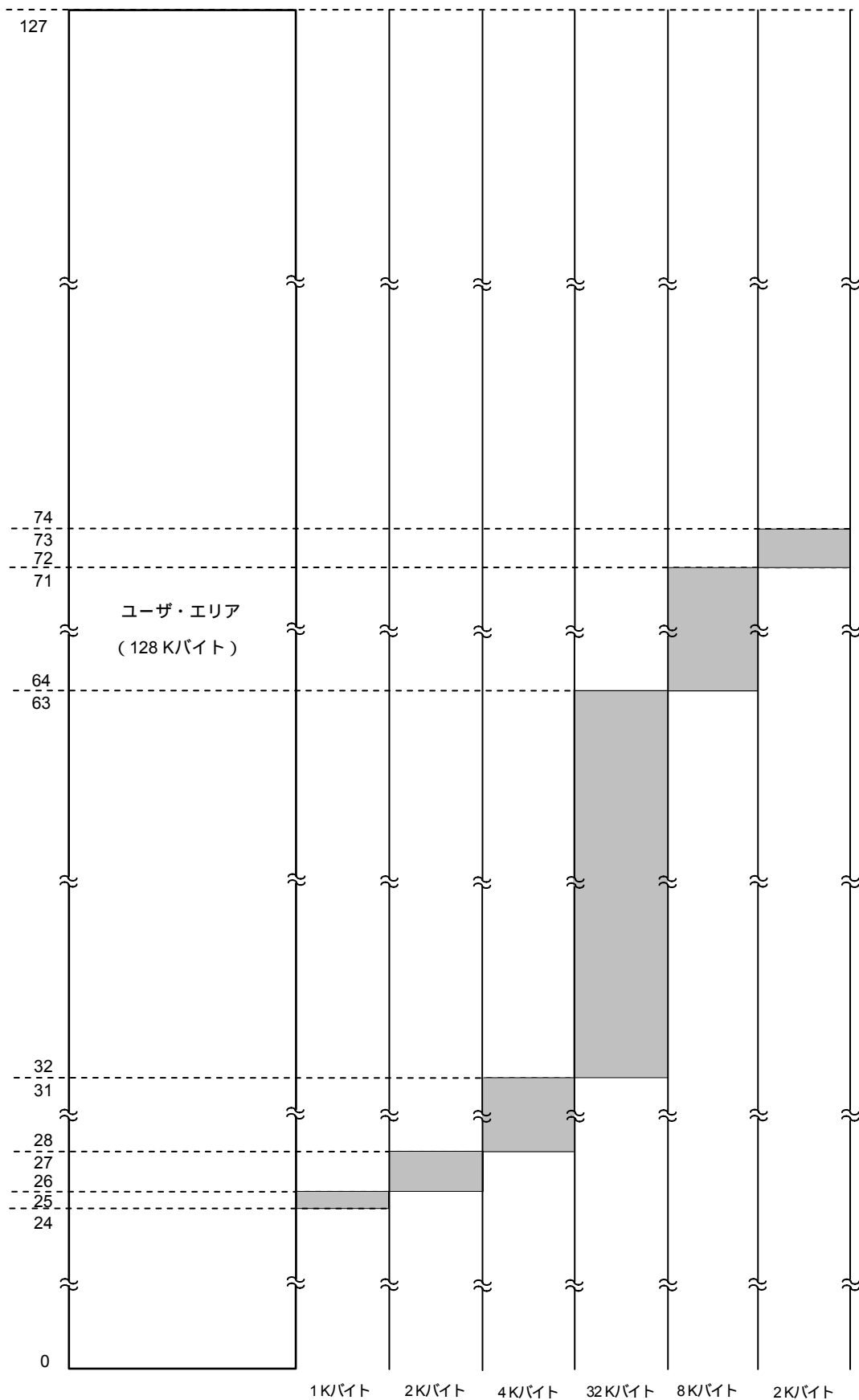
ブロック 64～71 を消去すると、次の開始ブロック番号は 72 で、消去ブロック数が 2 であることから、【条件 1】を満たす数値は、1, 2 となります。

さらに【条件 2】を満たす数値は、1, 2 となり、【条件 3】を満たす数値は“2”であるため、同時選択消去ブロック数は“2”となることから、ブロック 72～73 を消去します。

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

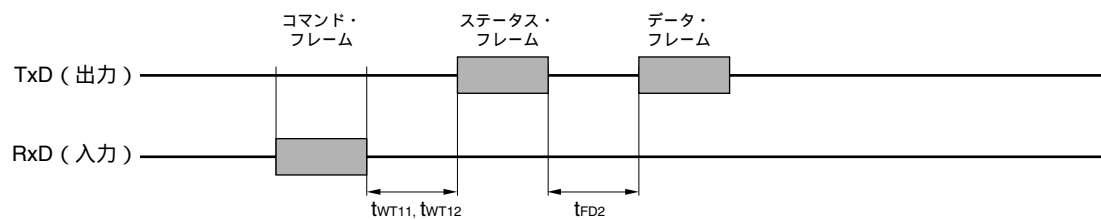
同時選択消去実行時のブロック構成（ブロック25～73を消去する場合）

<ブロック番号>

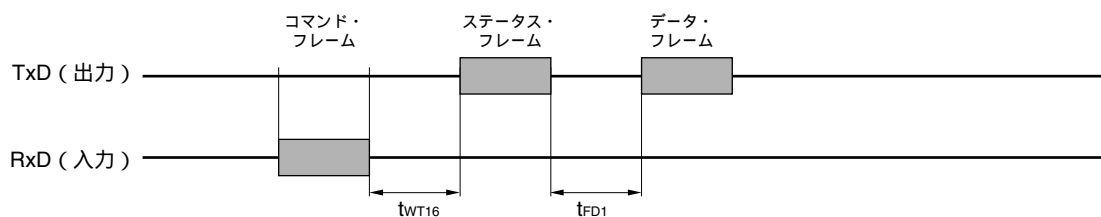


<同時選択消去可能なブロックの範囲>

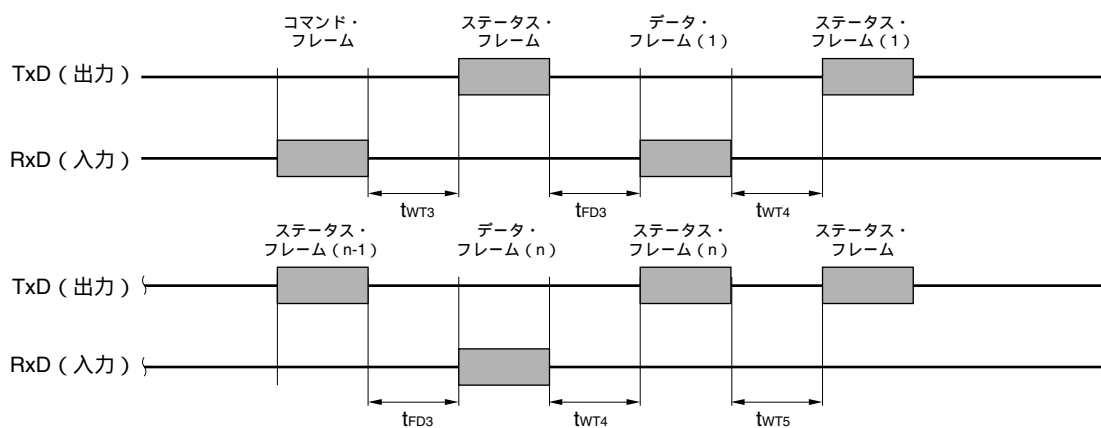
(e) Silicon Signatureコマンド / Version Getコマンド



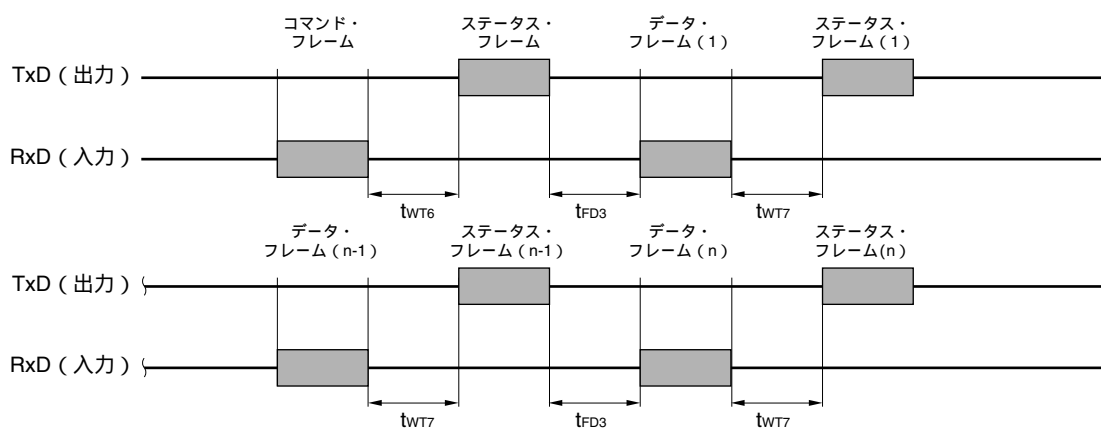
(f) Checksumコマンド



(g) Programmingコマンド



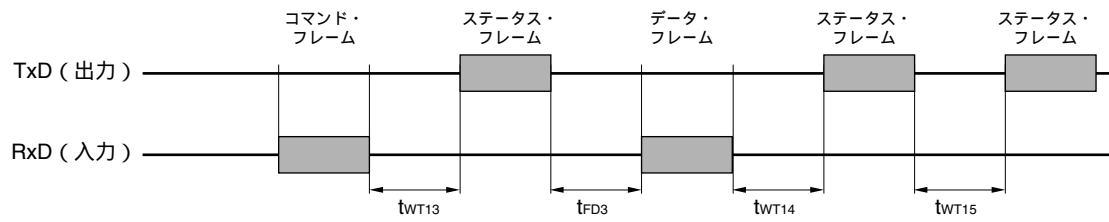
(h) Verifyコマンド



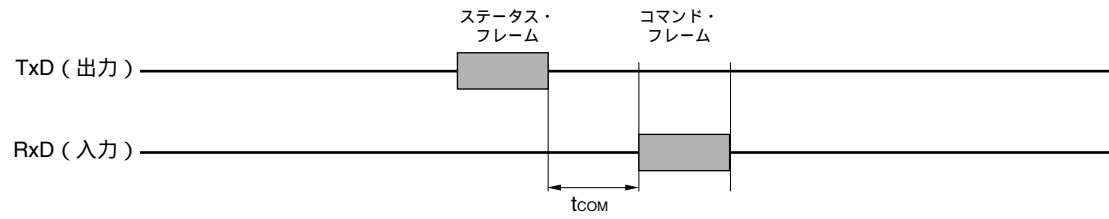
備考 TxD: TxD6

RxD: RxD6

(i) Security Settingコマンド



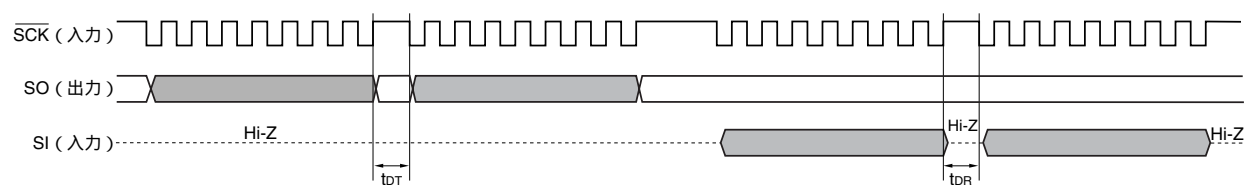
(j) コマンド・フレーム送信前のウェイト



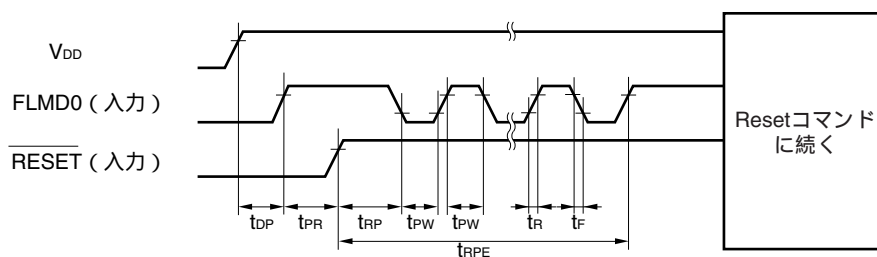
備考 TxD: TxD6
RxD: RxD6

8.4 3線式シリアルI/O通信方式

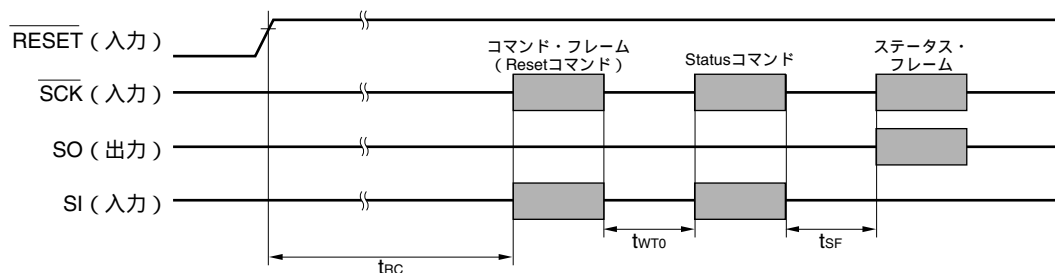
(a) データ・フレーム



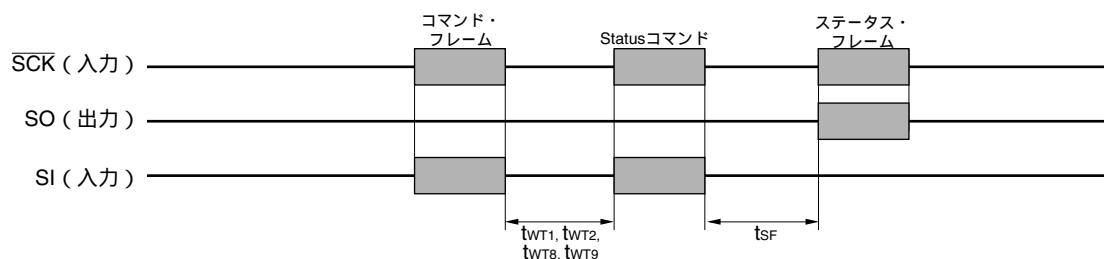
(b) プログラミング・モード設定



(c) Resetコマンド



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

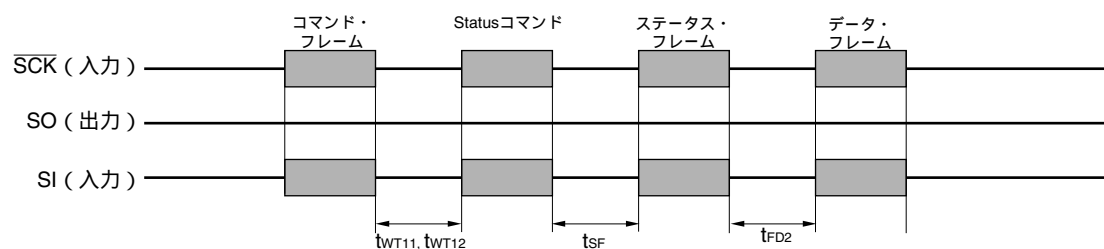


備考 SCK: SCK10

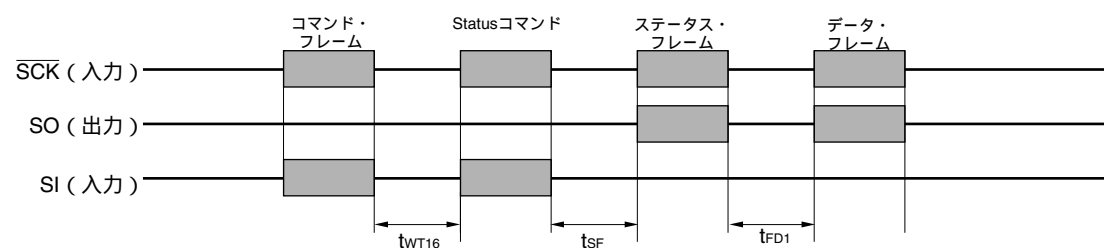
SO: SO10

SI: SI10

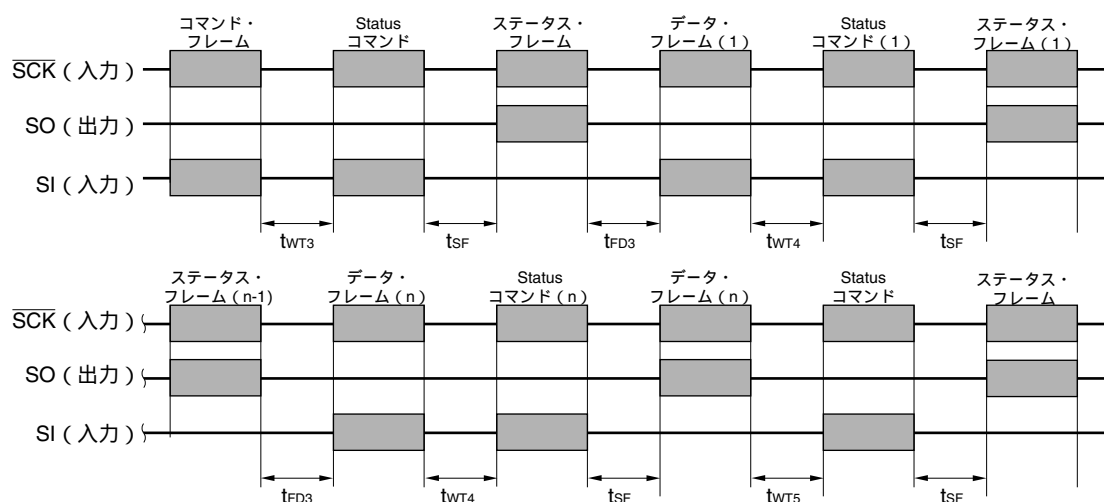
(e) Silicon Signatureコマンド / Version Getコマンド



(f) Checksumコマンド



(g) Programmingコマンド

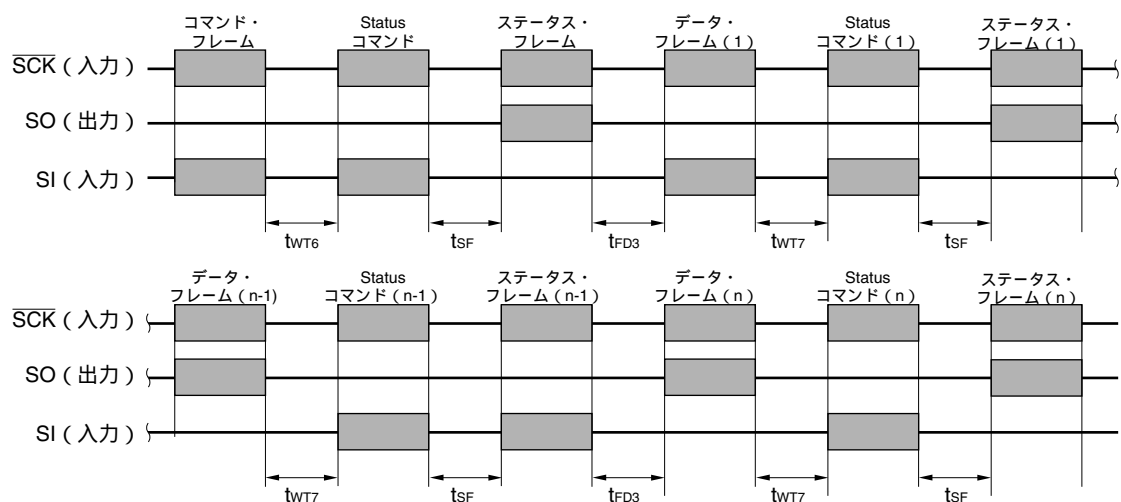


備考 SCK: SCK10

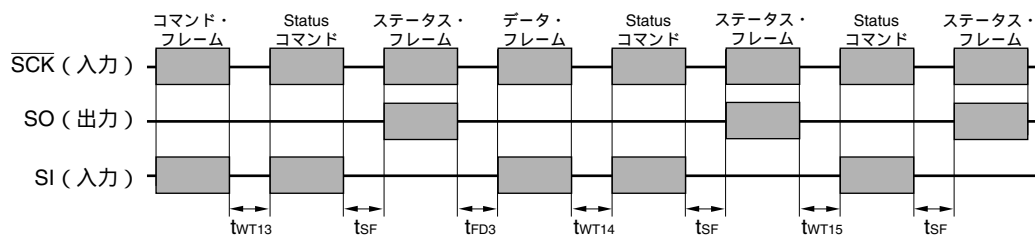
SO: SO10

SI: SI10

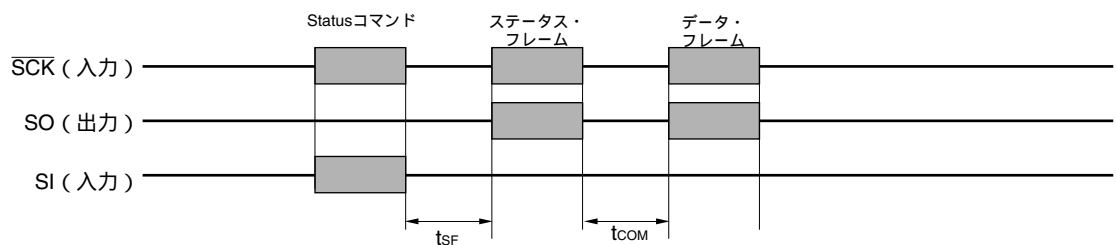
(h) Verifyコマンド



(i) Security Settingコマンド



(j) コマンド・フレーム送信前のウェイト



備考 SCK: SCK10

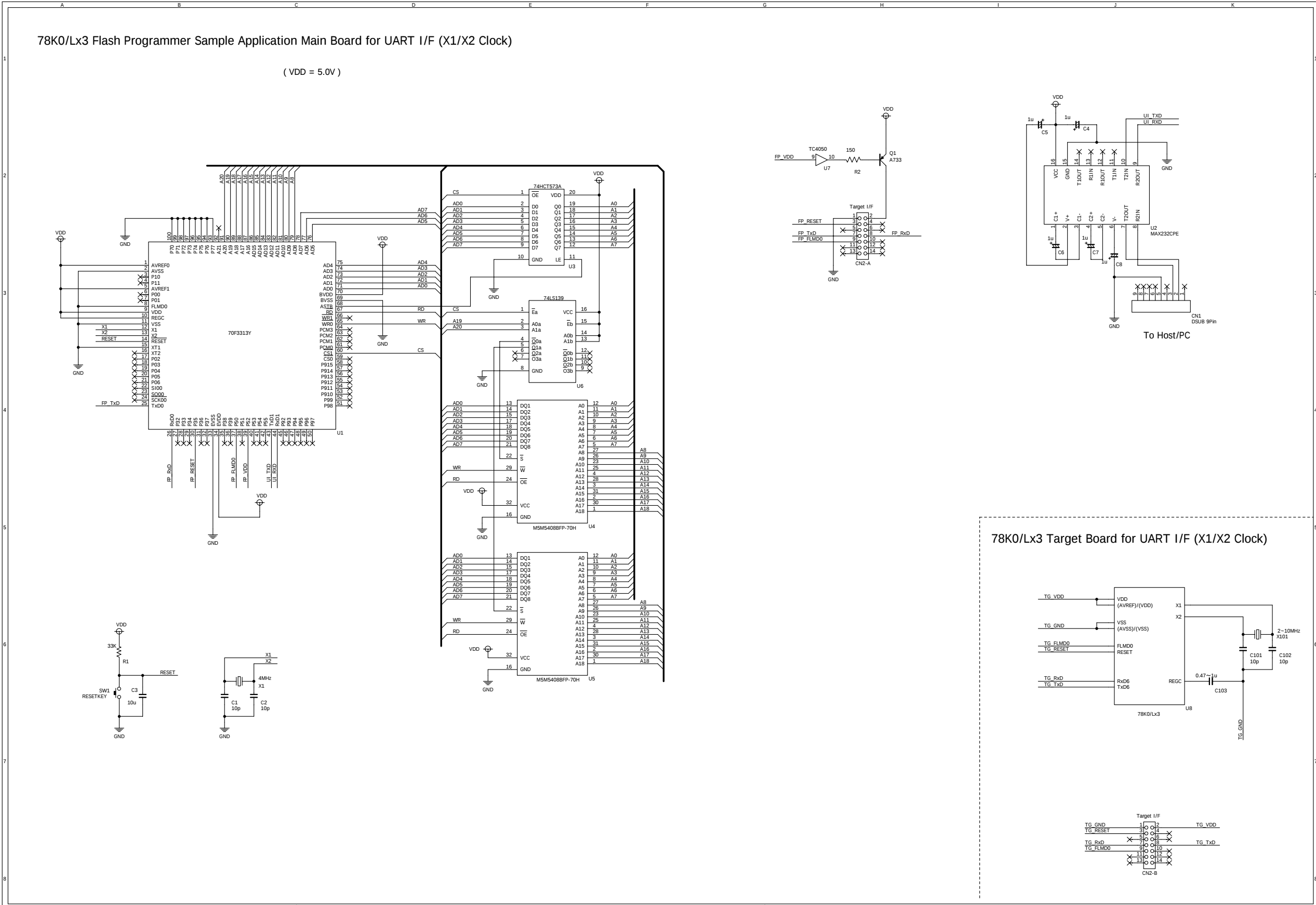
SO: SO10

SI: SI10

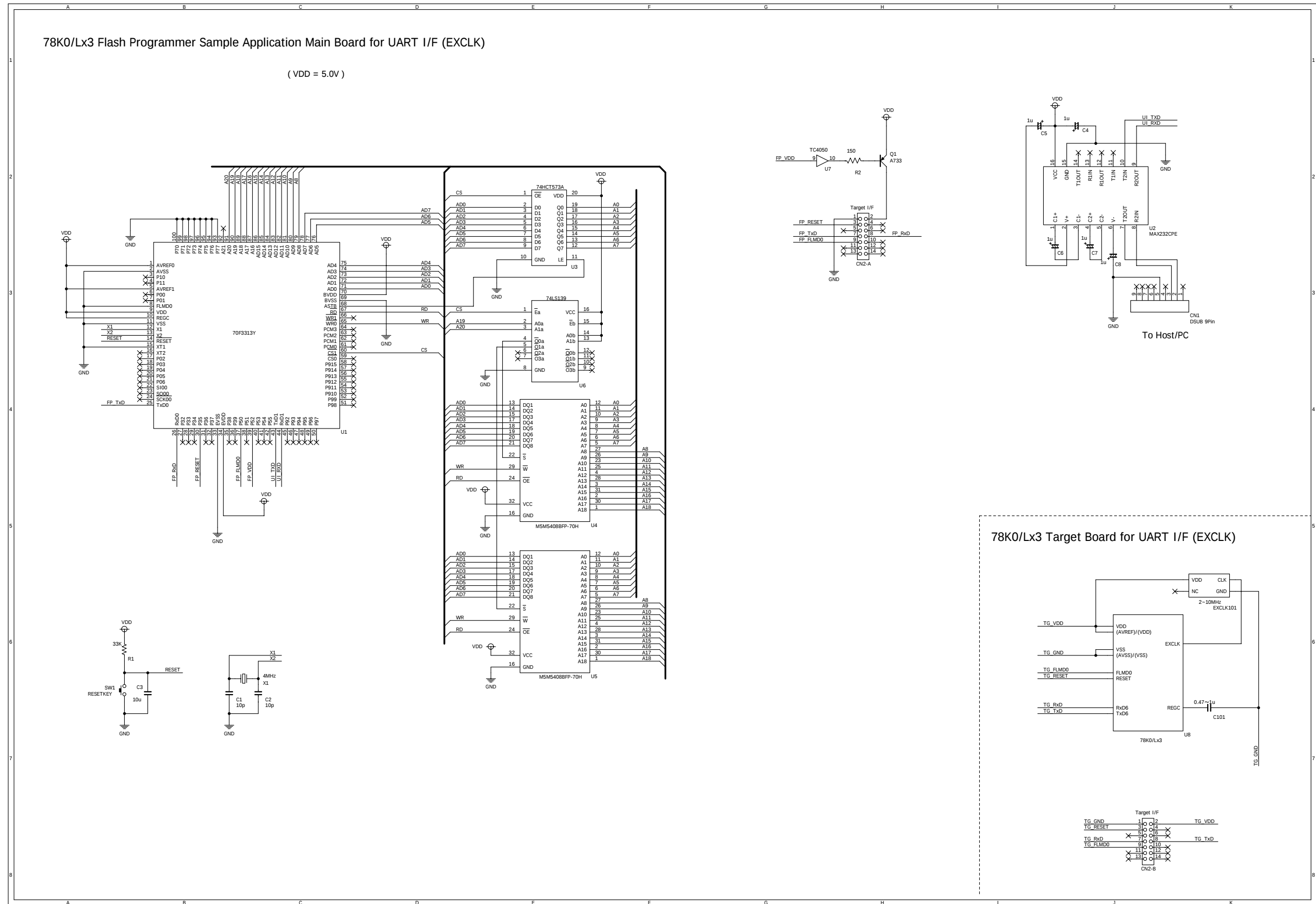
付録A 参考回路図

プログラマと78K0/Lx3の参考回路図を図A - 1～図A - 4に示します。

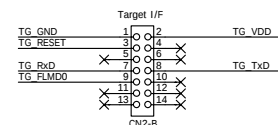
図A - 1 プログラマと78K0/Lx3の参考回路図 (UART通信 : X1クロック使用時)



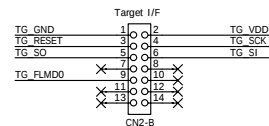
図A - 2 プログラマと78K0/Lx3の参考回路図 (UART通信：外部クロック使用時)



図A - 3 プログラマと78K0/Lx3の参考回路図（UART通信：内蔵発振器使用時）



図A - 4 プログラマと78K0/Lx3の参考回路図（CSI通信時）



〔メ モ〕

【発 行】

NECエレクトロニクス株式会社

〒211-8668 神奈川県川崎市中原区下沼部1753

電話（代表）：044(435)5111

—— お問い合わせ先 ——

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。

URL(アドレス) <http://www.necel.co.jp/>

【営業関係，技術関係お問い合わせ先】

半導体ホットライン

（電話：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。