

V850ES/SG3, V850ES/SJ3 マイクロコントローラ

V850ES/SG3, V850ES/SJ3 マイクロコントローラ

R01AN0930JJ0200

Rev.2.00

フラッシュ・メモリ・プログラミング (プログラム編)

2011.12.28

要旨

このアプリケーション・ノートは、V850ES/SG3, V850ES/SJ3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。

このアプリケーション・ノートは、V850ES/SG3, V850ES/SJ3内蔵のフラッシュ・メモリの書き換えを行うのに、ユーザ専用のフラッシュ・メモリ・プログラマを開発するための方法をユーザに理解していただくことを目的としています。

なお、掲載のプログラムおよび回路図は例示したものであり、量産設計を対象とするものではありません。

したがって、お客様の機器に使用される場合には、設計後、お客様の責任において十分な評価を行ってください。

対象デバイス

V850ES/SG3

μPD70F3333, μPD70F3334

μPD70F3335, μPD70F3336

μPD70F3340, μPD70F3341

μPD70F3342, μPD70F3343

μPD70F3350, μPD70F3351

μPD70F3352, μPD70F3353

V850ES/SJ3

μPD70F3344, μPD70F3345

μPD70F3346, μPD70F3347

μPD70F3348, μPD70F3354

μPD70F3355, μPD70F3356

μPD70F3357, μPD70F3358

μPD70F3364, μPD70F3365

μPD70F3366, μPD70F3367

μPD70F3368

本文欄外の 印は、本版で改訂された主な箇所を示しています。

この“ ”をPDF上でコピーして「検索する文字列」に指定することによって、改版箇所を容易に検索できます。

目 次

第1章	フラッシュ・メモリ・プログラミング	...	10
1.1	概 要	...	10
1.2	システム構成	...	11
1.3	プログラミング概要	...	12
1.3.1	プログラミング・モードへの遷移	...	12
1.3.2	シリアル通信方式の選択	...	12
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	...	13
1.4	V850ES/Sx3製品固有情報	...	13
第2章	プログラマ動作環境	...	17
2.1	プログラマ制御端子	...	17
2.2	各制御端子の詳細	...	18
2.2.1	フラッシュ・メモリ・プログラミング・モード引き込み用端子 (FLMD0, FLMD1)	...	18
2.2.2	シリアル・インタフェース端子 (TxD, RxD, SI, SO, SCK, HS)	...	19
2.2.3	リセット制御端子 (RESET)	...	20
2.2.4	クロック制御端子 (CLK)	...	20
2.2.5	VDD, GND制御端子	...	21
2.2.6	その他の端子	...	21
2.3	基本フロー・チャート	...	22
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	...	24
2.4.1	モード引き込みのフロー・チャート	...	25
2.4.2	サンプル・プログラム	...	26
2.5	シリアル通信方式の選択	...	28
2.6	UART通信方式	...	28
2.7	3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式	...	29
2.8	3線式シリアルI/O (CSI) 通信方式	...	29
2.9	ターゲットの電源遮断処理	...	29
2.10	フラッシュ・メモリの操作方法	...	29
2.11	コマンド一覧	...	30
2.12	ステータス一覧	...	31
第3章	プログラマの基本動作	...	32
第4章	コマンド/データ・フレーム・フォーマット	...	33
4.1	コマンド・フレーム送信処理	...	35
4.2	データ・フレーム送信処理	...	35
4.3	データ・フレーム受信処理	...	35

第5章 コマンド処理説明 ... 36

- 5.1 Statusコマンド ... 36
 - 5.1.1 説明 ... 36
 - 5.1.2 コマンド・フレームとステータス・フレーム ... 36
- 5.2 Resetコマンド ... 37
 - 5.2.1 説明 ... 37
 - 5.2.2 コマンド・フレームとステータス・フレーム ... 37
- 5.3 Baud Rate Setコマンド ... 38
 - 5.3.1 説明 ... 38
 - 5.3.2 コマンド・フレームとステータス・フレーム ... 38
- 5.4 Oscillating Frequency Setコマンド ... 39
 - 5.4.1 説明 ... 39
 - 5.4.2 コマンド・フレームとステータス・フレーム ... 39
- 5.5 Chip Eraseコマンド ... 41
 - 5.5.1 説明 ... 41
 - 5.5.2 コマンド・フレームとステータス・フレーム ... 41
- 5.6 Block Eraseコマンド ... 42
 - 5.6.1 説明 ... 42
 - 5.6.2 コマンド・フレームとステータス・フレーム ... 42
- 5.7 Programmingコマンド ... 43
 - 5.7.1 説明 ... 43
 - 5.7.2 コマンド・フレームとステータス・フレーム ... 43
 - 5.7.3 データ・フレームとステータス・フレーム ... 43
 - 5.7.4 全データ転送完了とステータス・フレーム ... 44
- 5.8 Verifyコマンド ... 45
 - 5.8.1 説明 ... 45
 - 5.8.2 コマンド・フレームとステータス・フレーム ... 45
 - 5.8.3 データ・フレームとステータス・フレーム ... 45
- 5.9 Block Blank Checkコマンド ... 47
 - 5.9.1 説明 ... 47
 - 5.9.2 コマンド・フレームとステータス・フレーム ... 47
- 5.10 Silicon Signatureコマンド ... 48
 - 5.10.1 説明 ... 48
 - 5.10.2 コマンド・フレームとステータス・フレーム ... 48
 - 5.10.3 シリコン・シグネチャ・データ・フレーム ... 48
 - 5.10.4 V850ES/Sx3シリコン・シグネチャー一覧 ... 52
- 5.11 Version Getコマンド ... 55
 - 5.11.1 説明 ... 55
 - 5.11.2 コマンド・フレームとステータス・フレーム ... 55
 - 5.11.3 バージョン・データ・フレーム ... 56
- 5.12 Checksumコマンド ... 57
 - 5.12.1 説明 ... 57
 - 5.12.2 コマンド・フレームとステータス・フレーム ... 57
 - 5.12.3 チェックサム・データ・フレーム ... 57
- 5.13 Security Setコマンド ... 58
 - 5.13.1 説明 ... 58
 - 5.13.2 コマンド・フレームとステータス・フレーム ... 58
 - 5.13.3 データ・フレームとステータス・フレーム ... 59
 - 5.13.4 内部ベリファイ確認とステータス・フレーム ... 59

- 5.14 Readコマンド ... 61
 - 5.14.1 説明 ... 61
 - 5.14.2 コマンド・フレームとステータス・フレーム ... 61
 - 5.14.3 データ・フレームとステータス・フレーム ... 61

第6章 UART通信方式 ... 63

- 6.1 コマンド・フレーム送信処理のフロー・チャート ... 63
- 6.2 データ・フレーム送信処理のフロー・チャート ... 64
- 6.3 データ・フレーム受信処理のフロー・チャート ... 65
- 6.4 Resetコマンド ... 66
 - 6.4.1 処理手順チャート ... 66
 - 6.4.2 処理手順説明 ... 67
 - 6.4.3 終了時の内容 ... 67
 - 6.4.4 フロー・チャート ... 68
 - 6.4.5 サンプル・プログラム ... 69
- 6.5 Baud Rate Setコマンド ... 70
 - 6.5.1 処理手順チャート ... 70
 - 6.5.2 処理手順説明 ... 71
 - 6.5.3 終了時の内容 ... 71
 - 6.5.4 フロー・チャート ... 72
 - 6.5.5 サンプル・プログラム ... 73
- 6.6 Oscillating Frequency Setコマンド ... 74
 - 6.6.1 処理手順チャート ... 74
 - 6.6.2 処理手順説明 ... 75
 - 6.6.3 終了時の内容 ... 75
 - 6.6.4 フロー・チャート ... 76
 - 6.6.5 サンプル・プログラム ... 77
- 6.7 Chip Eraseコマンド ... 78
 - 6.7.1 処理手順チャート ... 78
 - 6.7.2 処理手順説明 ... 79
 - 6.7.3 終了時の内容 ... 79
 - 6.7.4 フロー・チャート ... 80
 - 6.7.5 サンプル・プログラム ... 81
- 6.8 Block Eraseコマンド ... 82
 - 6.8.1 処理手順チャート ... 82
 - 6.8.2 処理手順説明 ... 83
 - 6.8.3 終了時の内容 ... 83
 - 6.8.4 フロー・チャート ... 84
 - 6.8.5 サンプル・プログラム ... 85
- 6.9 Programmingコマンド ... 86
 - 6.9.1 処理手順チャート ... 86
 - 6.9.2 処理手順説明 ... 87
 - 6.9.3 終了時の内容 ... 88
 - 6.9.4 フロー・チャート ... 89
 - 6.9.5 サンプル・プログラム ... 90
- 6.10 Verifyコマンド ... 92
 - 6.10.1 処理手順チャート ... 92
 - 6.10.2 処理手順説明 ... 93
 - 6.10.3 終了時の内容 ... 93

6. 10. 4	フロー・チャート	...	94
6. 10. 5	サンプル・プログラム	...	95
6. 11	Block Blank Checkコマンド	...	97
6. 11. 1	処理手順チャート	...	97
6. 11. 2	処理手順説明	...	98
6. 11. 3	終了時の内容	...	98
6. 11. 4	フロー・チャート	...	99
6. 11. 5	サンプル・プログラム	...	100
6. 12	Silicon Signatureコマンド	...	101
6. 12. 1	処理手順チャート	...	101
6. 12. 2	処理手順説明	...	102
6. 12. 3	終了時の内容	...	102
6. 12. 4	フロー・チャート	...	103
6. 12. 5	サンプル・プログラム	...	104
6. 13	Version Getコマンド	...	105
6. 13. 1	処理手順チャート	...	105
6. 13. 2	処理手順説明	...	106
6. 13. 3	終了時の内容	...	106
6. 13. 4	フロー・チャート	...	107
6. 13. 5	サンプル・プログラム	...	108
6. 14	Checksumコマンド	...	109
6. 14. 1	処理手順チャート	...	109
6. 14. 2	処理手順説明	...	110
6. 14. 3	終了時の内容	...	110
6. 14. 4	フロー・チャート	...	111
6. 14. 5	サンプル・プログラム	...	112
6. 15	Security Setコマンド	...	113
6. 15. 1	処理手順チャート	...	113
6. 15. 2	処理手順説明	...	114
6. 15. 3	終了時の内容	...	115
6. 15. 4	フロー・チャート	...	116
6. 15. 5	サンプル・プログラム	...	117
6. 16	Readコマンド	...	119
6. 16. 1	処理手順チャート	...	119
6. 16. 2	処理手順説明	...	120
6. 16. 3	終了時の内容	...	120
6. 16. 4	フロー・チャート	...	121
6. 16. 5	サンプル・プログラム	...	122
第7章	3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式	...	124
7. 1	コマンド・フレーム送信処理のフロー・チャート	...	124
7. 2	データ・フレーム送信処理のフロー・チャート	...	125
7. 3	データ・フレーム受信処理のフロー・チャート	...	126
7. 4	Statusコマンド	...	127
7. 4. 1	処理手順チャート	...	127
7. 4. 2	処理手順説明	...	128
7. 4. 3	終了時の内容	...	128
7. 4. 4	フロー・チャート	...	129

7.4.5	サンプル・プログラム	...	130
7.5	Reset コマンド	...	131
7.5.1	処理手順チャート	...	131
7.5.2	処理手順説明	...	132
7.5.3	終了時の内容	...	132
7.5.4	フロー・チャート	...	133
7.5.5	サンプル・プログラム	...	134
7.6	Oscillating Frequency Set コマンド	...	135
7.6.1	処理手順チャート	...	135
7.6.2	処理手順説明	...	136
7.6.3	終了時の内容	...	136
7.6.4	フロー・チャート	...	137
7.6.5	サンプル・プログラム	...	138
7.7	Chip Erase コマンド	...	139
7.7.1	処理手順チャート	...	139
7.7.2	処理手順説明	...	140
7.7.3	終了時の内容	...	140
7.7.4	フロー・チャート	...	141
7.7.5	サンプル・プログラム	...	142
7.8	Block Erase コマンド	...	143
7.8.1	処理手順チャート	...	143
7.8.2	処理手順説明	...	144
7.8.3	終了時の内容	...	144
7.8.4	フロー・チャート	...	145
7.8.5	サンプル・プログラム	...	146
7.9	Programming コマンド	...	147
7.9.1	処理手順チャート	...	147
7.9.2	処理手順説明	...	148
7.9.3	終了時の内容	...	149
7.9.4	フロー・チャート	...	150
7.9.5	サンプル・プログラム	...	151
7.10	Verify コマンド	...	153
7.10.1	処理手順チャート	...	153
7.10.2	処理手順説明	...	154
7.10.3	終了時の内容	...	155
7.10.4	フロー・チャート	...	156
7.10.5	サンプル・プログラム	...	157
7.11	Block Blank Check コマンド	...	159
7.11.1	処理手順チャート	...	159
7.11.2	処理手順説明	...	160
7.11.3	終了時の内容	...	160
7.11.4	フロー・チャート	...	161
7.11.5	サンプル・プログラム	...	162
7.12	Silicon Signature コマンド	...	163
7.12.1	処理手順チャート	...	163
7.12.2	処理手順説明	...	164
7.12.3	終了時の内容	...	164
7.12.4	フロー・チャート	...	165
7.12.5	サンプル・プログラム	...	166

7.13	Version Getコマンド	...	167
7.13.1	処理手順チャート	...	167
7.13.2	処理手順説明	...	168
7.13.3	終了時の内容	...	168
7.13.4	フロー・チャート	...	169
7.13.5	サンプル・プログラム	...	170
7.14	Checksumコマンド	...	171
7.14.1	処理手順チャート	...	171
7.14.2	処理手順説明	...	172
7.14.3	終了時の内容	...	172
7.14.4	フロー・チャート	...	173
7.14.5	サンプル・プログラム	...	174
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	...	178
7.15.4	フロー・チャート	...	179
7.15.5	サンプル・プログラム	...	180
7.16	Readコマンド	...	182
7.16.1	処理手順チャート	...	182
7.16.2	処理手順説明	...	183
7.16.3	終了時の内容	...	184
7.16.4	フロー・チャート	...	185
7.16.5	サンプル・プログラム	...	186
第8章	3線式シリアルI/O (CSI) 通信方式	...	188
8.1	コマンド・フレーム送信処理のフロー・チャート	...	188
8.2	データ・フレーム送信処理のフロー・チャート	...	189
8.3	データ・フレーム受信処理のフロー・チャート	...	190
8.4	Statusコマンド	...	191
8.4.1	処理手順チャート	...	191
8.4.2	処理手順説明	...	192
8.4.3	終了時の内容	...	192
8.4.4	フロー・チャート	...	193
8.4.5	サンプル・プログラム	...	194
8.5	Resetコマンド	...	196
8.5.1	処理手順チャート	...	196
8.5.2	処理手順説明	...	197
8.5.3	終了時の内容	...	197
8.5.4	フロー・チャート	...	198
8.5.5	サンプル・プログラム	...	199
8.6	Oscillating Frequency Setコマンド	...	200
8.6.1	処理手順チャート	...	200
8.6.2	処理手順説明	...	201
8.6.3	終了時の内容	...	201
8.6.4	フロー・チャート	...	202
8.6.5	サンプル・プログラム	...	203
8.7	Chip Eraseコマンド	...	204
8.7.1	処理手順チャート	...	204

8.7.2	処理手順説明	...	205
8.7.3	終了時の内容	...	205
8.7.4	フロー・チャート	...	206
8.7.5	サンプル・プログラム	...	207
8.8	Block Erase コマンド	...	208
8.8.1	処理手順チャート	...	208
8.8.2	処理手順説明	...	209
8.8.3	終了時の内容	...	209
8.8.4	フロー・チャート	...	210
8.8.5	サンプル・プログラム	...	211
8.9	Programming コマンド	...	212
8.9.1	処理手順チャート	...	212
8.9.2	処理手順説明	...	213
8.9.3	終了時の内容	...	214
8.9.4	フロー・チャート	...	215
8.9.5	サンプル・プログラム	...	216
8.10	Verify コマンド	...	218
8.10.1	処理手順チャート	...	218
8.10.2	処理手順説明	...	219
8.10.3	終了時の内容	...	219
8.10.4	フロー・チャート	...	220
8.10.5	サンプル・プログラム	...	221
8.11	Block Blank Check コマンド	...	223
8.11.1	処理手順チャート	...	223
8.11.2	処理手順説明	...	224
8.11.3	終了時の内容	...	224
8.11.4	フロー・チャート	...	225
8.11.5	サンプル・プログラム	...	226
8.12	Silicon Signature コマンド	...	227
8.12.1	処理手順チャート	...	227
8.12.2	処理手順説明	...	228
8.12.3	終了時の内容	...	228
8.12.4	フロー・チャート	...	229
8.12.5	サンプル・プログラム	...	230
8.13	Version Get コマンド	...	231
8.13.1	処理手順チャート	...	231
8.13.2	処理手順説明	...	232
8.13.3	終了時の内容	...	232
8.13.4	フロー・チャート	...	233
8.13.5	サンプル・プログラム	...	234
8.14	Checksum コマンド	...	235
8.14.1	処理手順チャート	...	235
8.14.2	処理手順説明	...	236
8.14.3	終了時の内容	...	236
8.14.4	フロー・チャート	...	237
8.14.5	サンプル・プログラム	...	238
8.15	Security Set コマンド	...	239
8.15.1	処理手順チャート	...	239
8.15.2	処理手順説明	...	240

8. 15. 3	終了時の内容	...	241
8. 15. 4	フロー・チャート	...	242
8. 15. 5	サンプル・プログラム	...	243
8. 16	Readコマンド	...	245
8. 16. 1	処理手順チャート	...	245
8. 16. 2	処理手順説明	...	246
8. 16. 3	終了時の内容	...	246
8. 16. 4	フロー・チャート	...	247
8. 16. 5	サンプル・プログラム	...	248
第9章	フラッシュ・メモリ・プログラミング・パラメータ特性	...	250
付録A	参考回路図	...	266

第1章 フラッシュ・メモリ・プログラミング

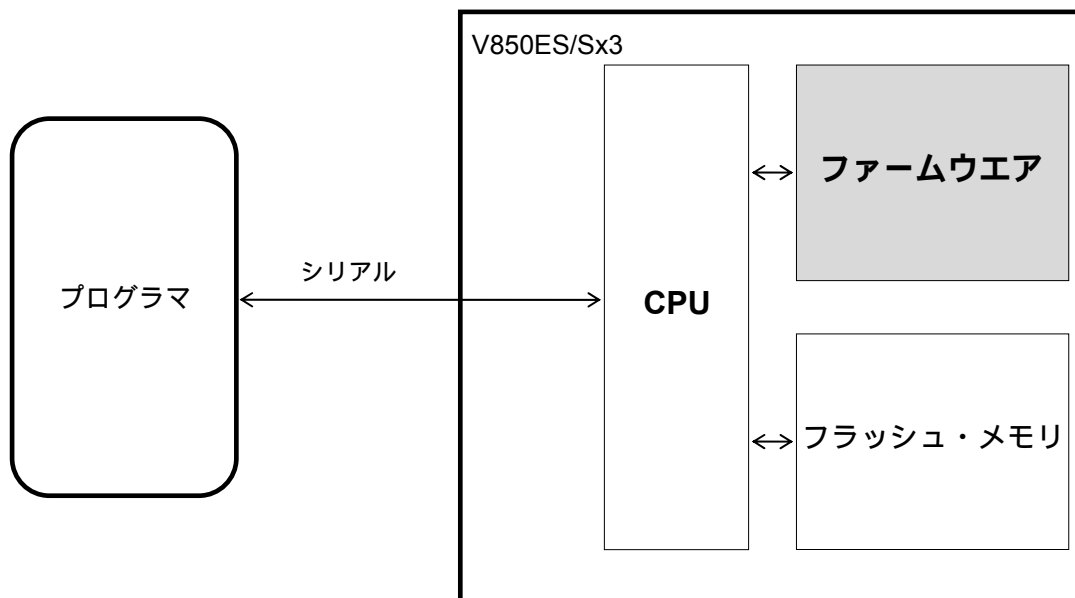
V850ES/Sx3に内蔵されるフラッシュ・メモリの書き換えを行うには、通常は専用のフラッシュ・メモリ・プログラマ (以降プログラマ) を使う必要があります。

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

V850ES/Sx3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V850ES/Sx3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 V850ES/Sx3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。

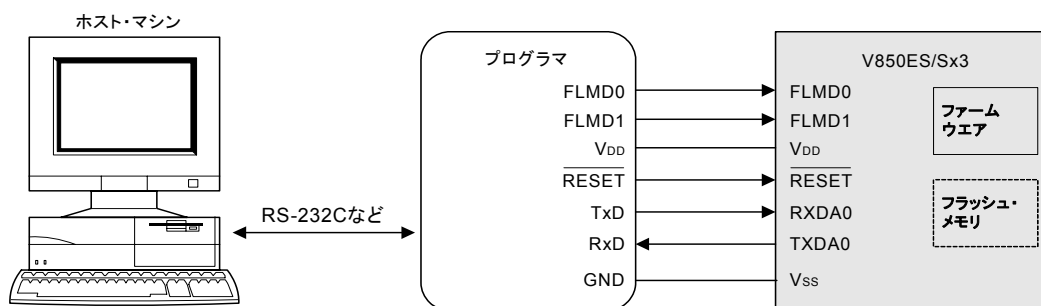
図1-2は、ホスト・マシンからの制御によりプログラマを使用するプログラミング方法を示しています。

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。

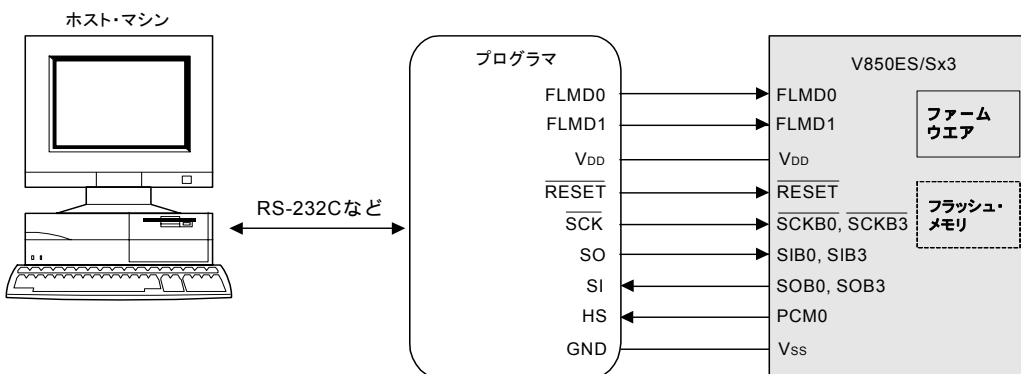
たとえば、ルネサス エレクトロニクス製フラッシュ・メモリ・プログラマ PG-FP4は、ホスト・マシンを接続してGUIソフトウェアにより実行する方法と、スタンド・アローンで実行する方法のどちらでも動作可能です。

図1-2 システム構成

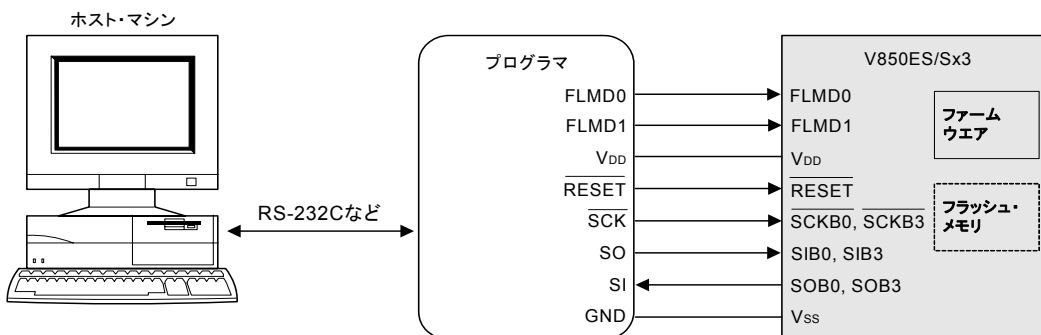
(1) UART通信方式 (LSB先頭転送)



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



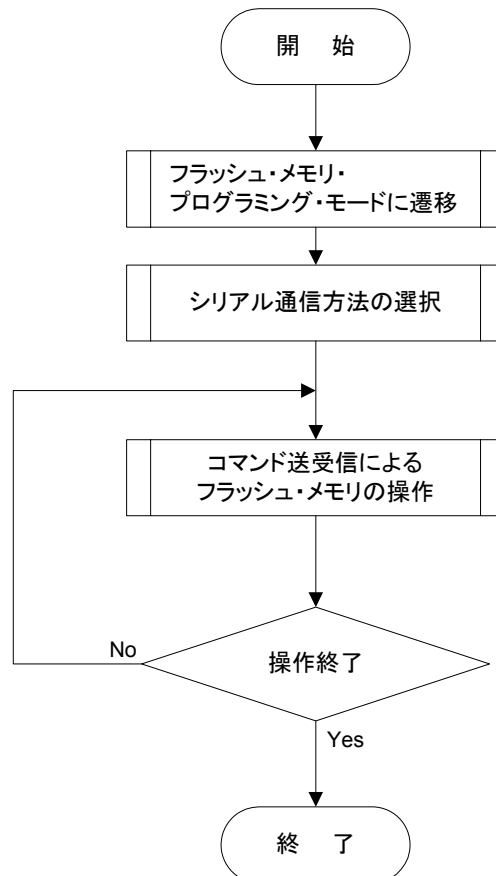
(3) 3線式シリアルI/O (CSI) 通信方式 (MSB先頭転送)



1.3 プログラミング概要

プログラマにてフラッシュ・メモリの書き換えを行うには、まずV850ES/Sx3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V850ES/Sx3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

V850ES/Sx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子 (FLMD0, FLMD1) に規定の電圧を供給し、その後リセットを解除することにより、フラッシュ・メモリ・プログラミング・モードに遷移させることができます。

1.3.2 シリアル通信方式の選択

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

1.3.3 コマンドの送受信によるフラッシュ・メモリの操作

V850ES/Sx3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V850ES/Sx3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V850ES/Sx3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 V850ES/Sx3製品固有情報

V850ES/Sx3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V850ES/Sx3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 V850ES/Sx3のフラッシュ・メモリ・サイズ (1/2)

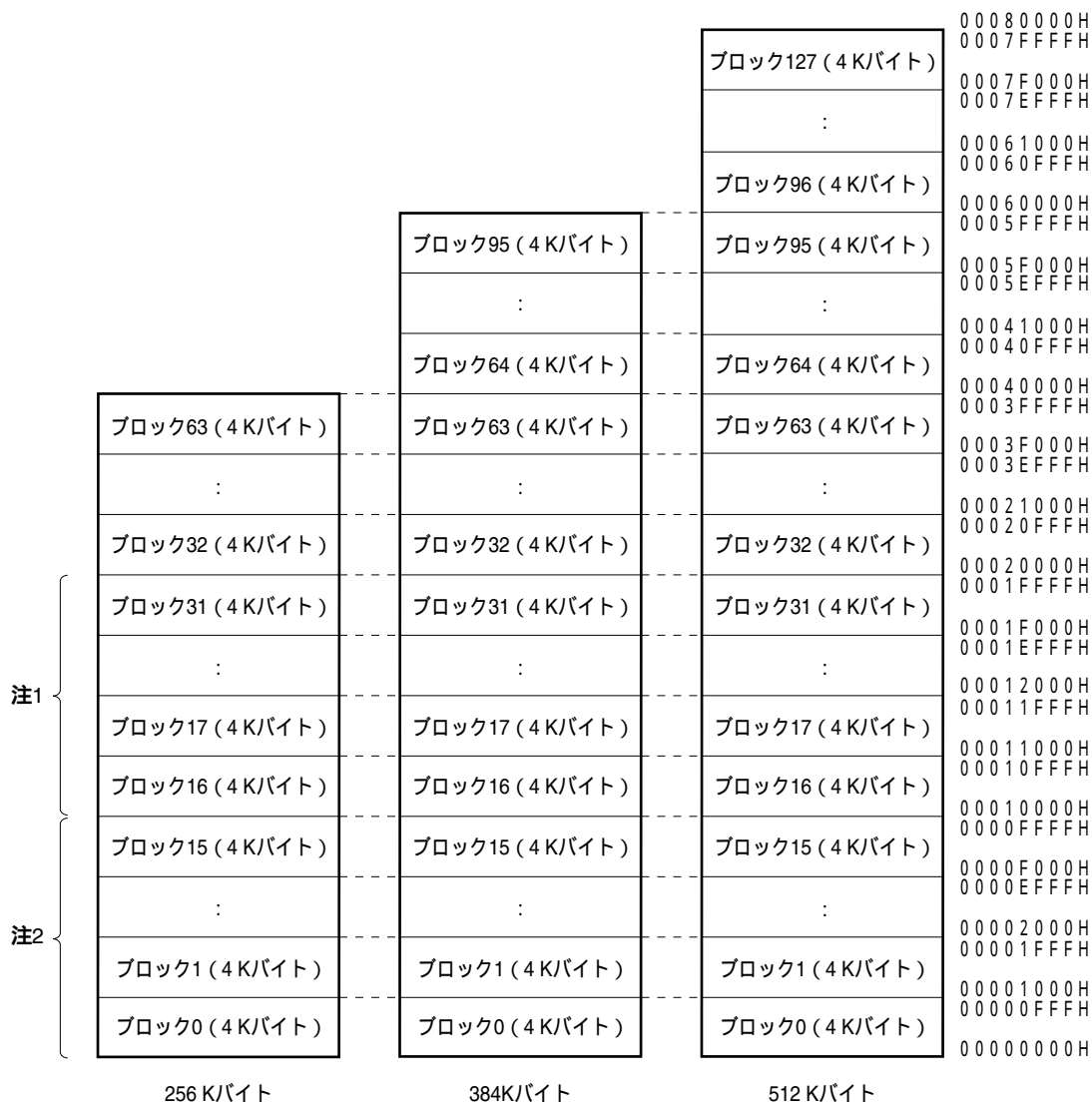
デバイス名		フラッシュ・メモリ・サイズ
V850ES/SG3	μ PD70F3333	256 KB
	μ PD70F3334	384 KB
	μ PD70F3335	256 KB
	μ PD70F3336	384 KB
	μ PD70F3340	512 KB
	μ PD70F3341	640 KB
	μ PD70F3342	768 KB
	μ PD70F3343	1024 KB
	μ PD70F3350	512 KB
	μ PD70F3351	640 KB
	μ PD70F3352	768 KB
	μ PD70F3353	1024 KB

表1 - 2 V850ES/Sx3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	
V850ES/SJ3	μ PD70F3344	384 KB
	μ PD70F3345	512 KB
	μ PD70F3346	640 KB
	μ PD70F3347	768 KB
	μ PD70F3348	1024 KB
	μ PD70F3354	384 KB
	μ PD70F3355	512 KB
	μ PD70F3356	640 KB
	μ PD70F3357	768 KB
	μ PD70F3358	1024 KB
	μ PD70F3364	384 KB
	μ PD70F3365	512 KB
	μ PD70F3366	640 KB
	μ PD70F3367	768 KB
	μ PD70F3368	1024 KB

図1 - 4 フラッシュ・メモリ構成 (1/2)

(a) 256 Kバイト / 384 Kバイト / 512 Kバイトの場合



注1. ブロック16-31 : ブート・スワップによりブート領域と入れ替わる領域

2. ブロック0-15 : ブート領域

図1 - 4 フラッシュ・メモリ構成 (2/2)

(b) 640 Kバイト / 768 Kバイト / 1024 Kバイトの場合

			ブロック255 (4 Kバイト)	000FFFFFFH
			:	000FF000H 000FFFFFFH
			ブロック192 (4 Kバイト)	000C1000H 000C0FFFFH
		ブロック191 (4 Kバイト)	ブロック191 (4 Kバイト)	000C0000H 000BFFFFFFH
		:	:	000BF000H 000BEFFFFH
		ブロック160 (4 Kバイト)	ブロック160 (4 Kバイト)	000A1000H 000A0FFFFH
	ブロック159 (4 Kバイト)	ブロック159 (4 Kバイト)	ブロック159 (4 Kバイト)	000A0000H 0009FFFFFFH
	:	:	:	0009F000H 0009EFFFFH
	ブロック128 (4 Kバイト)	ブロック128 (4 Kバイト)	ブロック128 (4 Kバイト)	00081000H 00080FFFFH
	ブロック127 (4 Kバイト)	ブロック127 (4 Kバイト)	ブロック127 (4 Kバイト)	00080000H 0007FFFFFFH
	:	:	:	0007F000H 0007EFFFFH
	ブロック96 (4 Kバイト)	ブロック96 (4 Kバイト)	ブロック96 (4 Kバイト)	00061000H 00060FFFFH
	ブロック95 (4 Kバイト)	ブロック95 (4 Kバイト)	ブロック95 (4 Kバイト)	00060000H 0005FFFFFFH
	:	:	:	0005F000H 0005EFFFFH
	ブロック64 (4 Kバイト)	ブロック64 (4 Kバイト)	ブロック64 (4 Kバイト)	00041000H 00040FFFFH
	ブロック63 (4 Kバイト)	ブロック63 (4 Kバイト)	ブロック63 (4 Kバイト)	00040000H 0003FFFFFFH
	:	:	:	0003F000H 0003EFFFFH
	ブロック32 (4 Kバイト)	ブロック32 (4 Kバイト)	ブロック32 (4 Kバイト)	00021000H 00020FFFFH
	ブロック31 (4 Kバイト)	ブロック31 (4 Kバイト)	ブロック31 (4 Kバイト)	00020000H 0001FFFFFFH
	:	:	:	0001F000H 0001EFFFFH
注1	ブロック17 (4 Kバイト)	ブロック17 (4 Kバイト)	ブロック17 (4 Kバイト)	00012000H 00011FFFFH
	ブロック16 (4 Kバイト)	ブロック16 (4 Kバイト)	ブロック16 (4 Kバイト)	00011000H 00010FFFFH
	ブロック15 (4 Kバイト)	ブロック15 (4 Kバイト)	ブロック15 (4 Kバイト)	00010000H 0000FFFFFFH
	:	:	:	0000F000H 0000EFFFFH
注2	ブロック1 (4 Kバイト)	ブロック1 (4 Kバイト)	ブロック1 (4 Kバイト)	00002000H 00001FFFFH
	ブロック0 (4 Kバイト)	ブロック0 (4 Kバイト)	ブロック0 (4 Kバイト)	00001000H 00000FFFFH
				00000000H
	640 Kバイト	768 Kバイト	1024 Kバイト	

- 注1. ブロック16-31 : ブート・スワップによりブート領域と入れ替わる領域
- 2. ブロック0-15 : ブート領域

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			V850ES/Sx3	ターゲットとの通信方式		
信号名	入出力	端子機能	端子名	CSI	CSI + HS	UART
FLMD0	出力	書き込みモードへの選択レベル出力および通信方式選択パルス出力	FLMD0			
FLMD1	出力	書き込みモードへの選択レベル出力	FLMD1			
V _{DD}	出力	V _{DD} 電圧生成 / 電圧監視	V _{DD}			
GND	-	グラウンド	V _{SS}			
CLK	出力	V850ES/Sx3への動作クロック出力	-	× ^注	× ^注	× ^注
RESET	出力	プログラミング・モード切り替えトリガ	RESET			
SO	出力	V850ES/Sx3へのコマンド送信	SIB0, SIB3			×
SI	入力	V850ES/Sx3からの応答ステータスおよび各種データの受信	SOB0, SOB3			×
SCK	出力	V850ES/Sx3へのシリアル・クロック供給	SCKB0, SCKB3			×
HS (ハンドシェーク)	入力	V850ES/Sx3とのシリアル通信用ハンドシェーク信号受信	PCM0	×		×
TxD	出力	V850ES/Sx3へのコマンド送信	RXDA0	×	×	
RxD	入力	V850ES/Sx3からの応答ステータスおよび各種データの受信	TXDA0	×	×	

注 プログラマのCLK端子からのクロック供給はできません。ターゲット上に発振回路を作成してクロックを供給してください。

備考 : 端子を使用します。

× : 端子を使用しません。

: ユーザ・システム側で供給されている場合は接続の必要はありません。

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

2.2 各制御端子の詳細

2.2.1 フラッシュ・メモリ・プログラミング・モード引き込み用端子 (FLMD0, FLMD1)

FLMD0, FLMD1端子はV850ES/Sx3の動作モードを制御するための端子です。FLMD0, FLMD1端子に規定の電圧を加えリセットを解除すると, V850ES/Sx3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V850ES/Sx3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2.5 シリアル通信方式の選択の表2 - 3を参照してください。

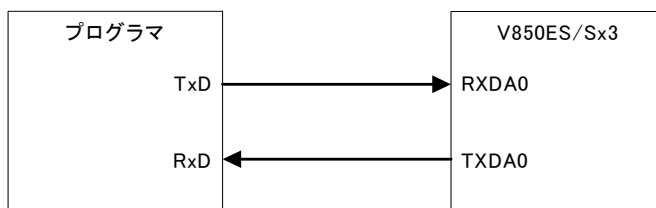
2.2.2 シリアル・インタフェース端子 (TxD, RxD, SI, SO, $\overline{\text{SCK}}$, HS)

シリアル・インタフェース端子は、プログラマとV850ES/Sx3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。

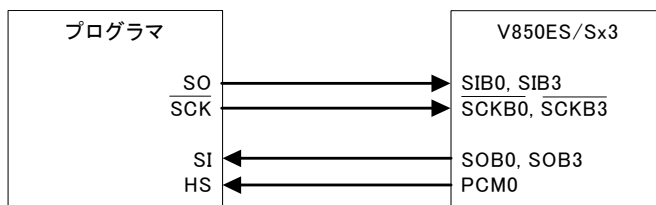
V850ES/Sx3の場合、通信方式としてUART, CSI + HS, CSIの3種類の中から選択できます。通信方式と使用端子の接続図を次に示します。

図2 - 1 シリアル・インタフェース端子

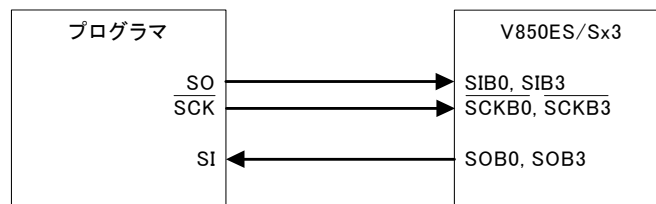
(1) UART通信方式



(2) 3線式シリアルI/O ハンドシェイク対応 (CSI + HS) 通信方式



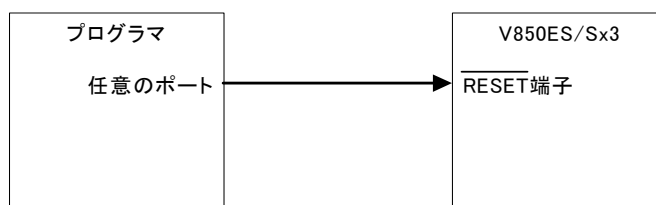
(3) 3線式シリアルI/O (CSI) 通信方式



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

リセット制御端子 ($\overline{\text{RESET}}$ 端子) は、プログラマからV850ES/Sx3のシステム・リセットを制御するための端子です。FLMD0, FLMD1端子を規定の電圧に設定し、その後リセットを解除することにより、フラッシュ・メモリ・プログラミング・モードを選択できます。

図2 - 2 $\overline{\text{RESET}}$ 端子



2.2.4 クロック制御端子 (CLK)

クロック制御端子は、使用いたしません。

プログラマのCLK端子からのクロック供給はできません。ターゲット上に発振回路を作成してクロックを供給してください。

2.2.5 V_{DD}, GND制御端子

V_{DD}制御端子は、プログラマからV850ES/Sx3に電源を供給する場合に使用します。プログラマからV850ES/Sx3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V850ES/Sx3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V850ES/Sx3のV_{SS}と接続してください。

図2 - 3 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 (EV_{DD}, EV_{SS}, BV_{DD}, BV_{SS}, AV_{REF0}, AV_{REF1}, 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2-4 フラッシュ処理の基本フロー・チャート

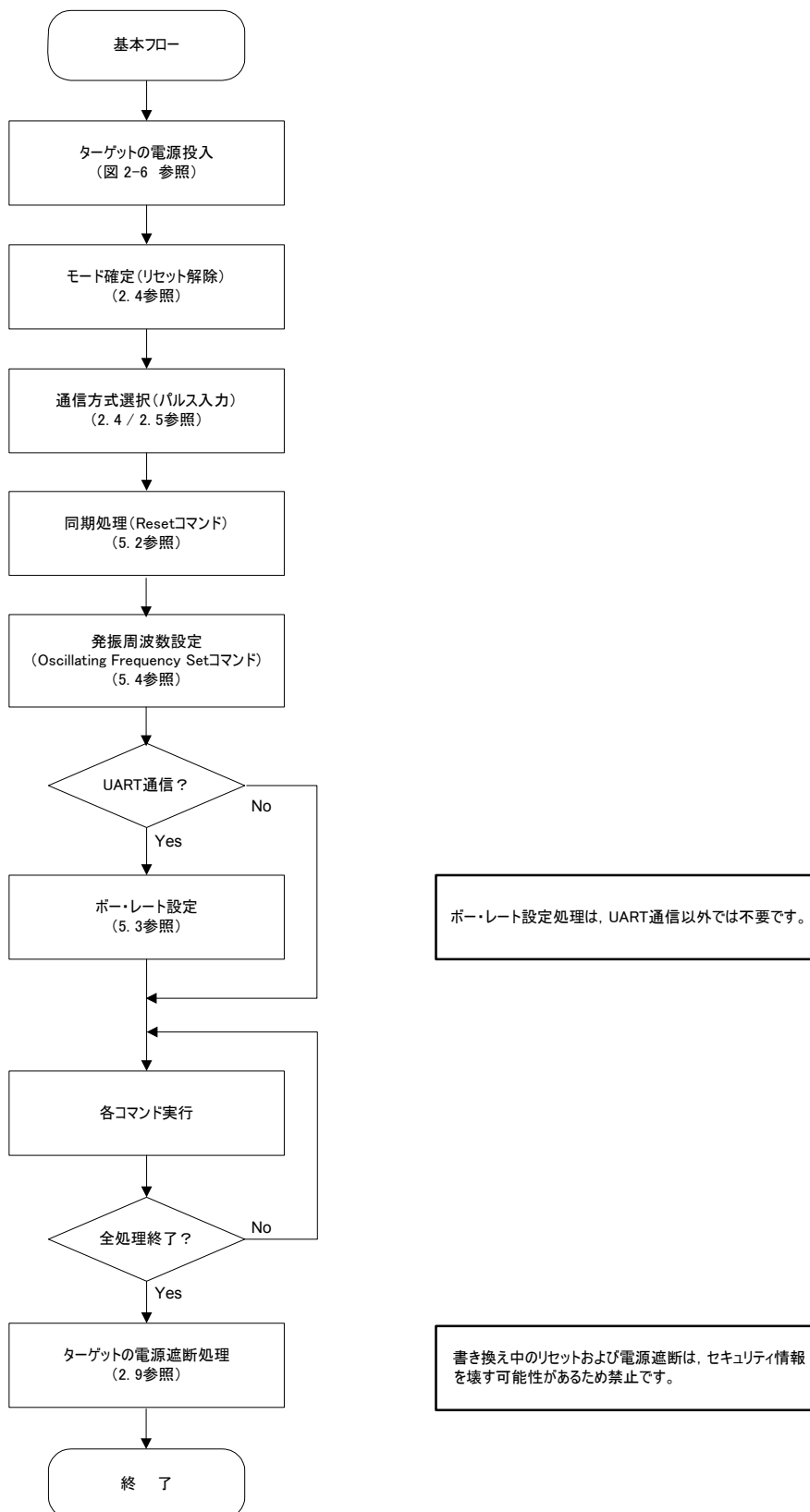
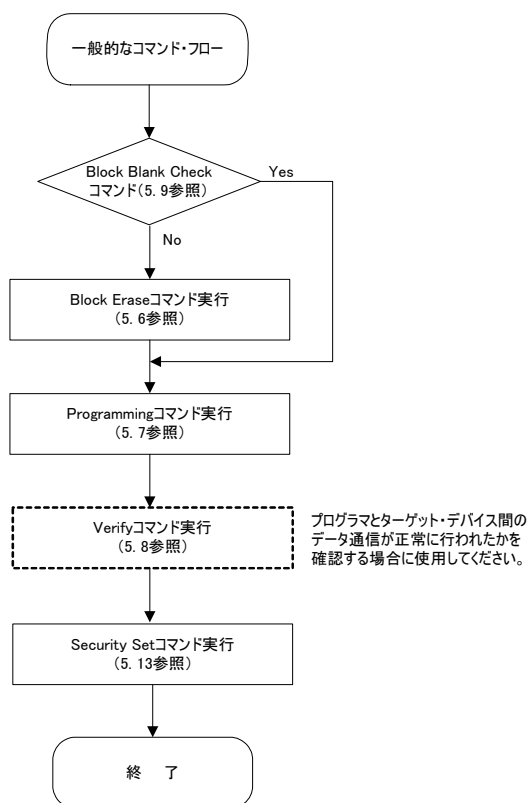


図2 - 5 フラッシュ・メモリ書き換え時の一般的なコマンド・フロー



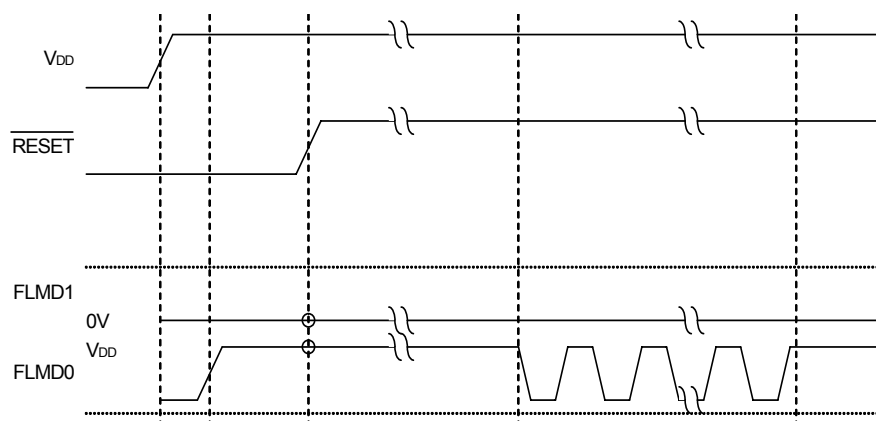
2.4 フラッシュ・メモリ・プログラミング・モードへの遷移方法

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V850ES/Sx3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、V850ES/Sx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子 (FLMD0, FLMD1) に規定の電圧を供給し、その後リセットを解除します。

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

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



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

リセット解除時のFLMD0, FLMD1端子と動作モードの関係を次に示します。

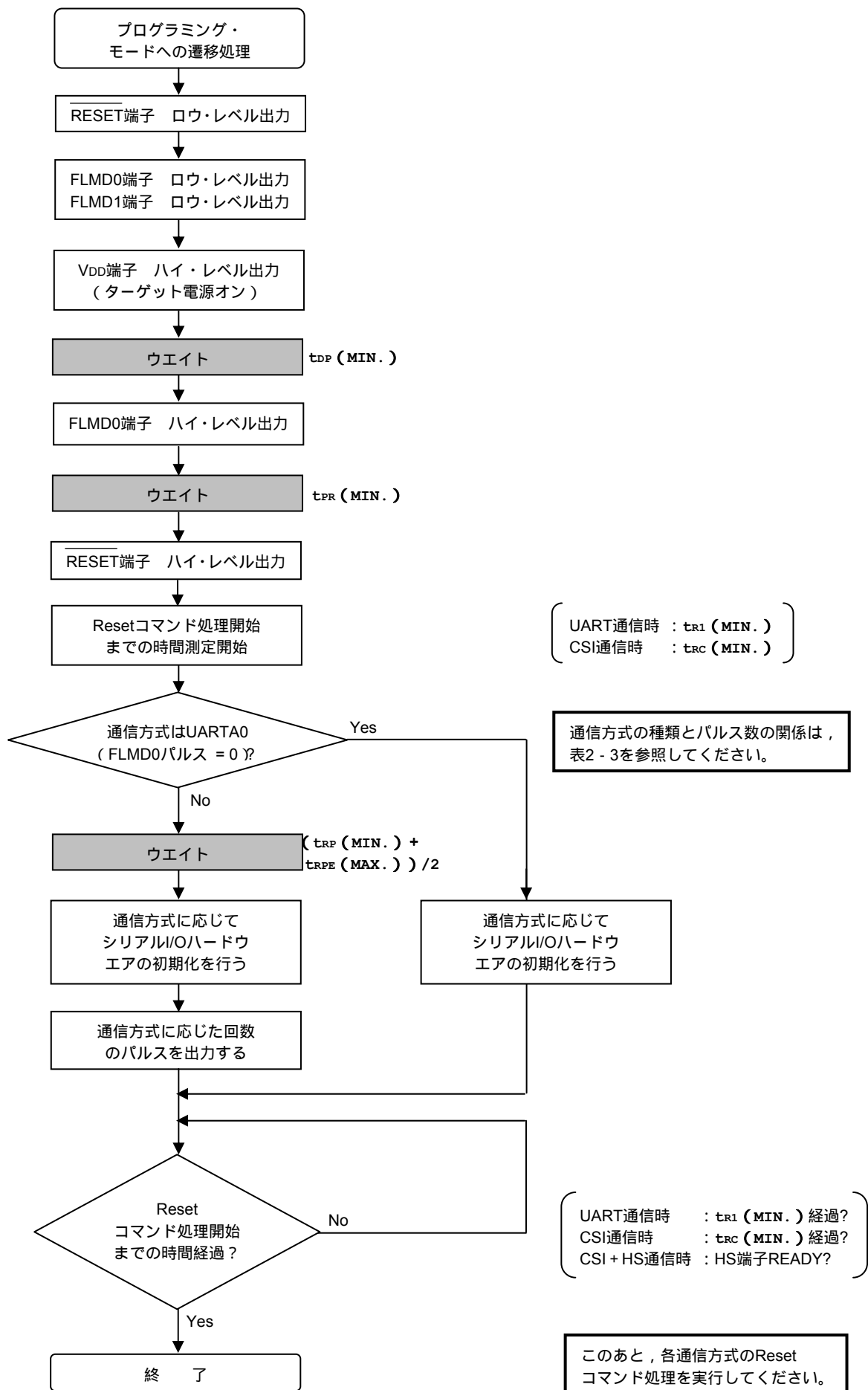
表2-2 リセット時のFLMD0, FLMD1端子の設定と動作モード

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

備考 L: ロウ・レベル入力

H: ハイ・レベル入力

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){
default:
case FLIF_UART: pulse = PULSE_UART; break;
case FLIF_CSI: pulse = UseCSIB3 ? PULSE_CSIB3 : PULSE_CSI; break;
case FLIF_CSI_HS:pulse = UseCSIB3 ? PULSE_CSIB3HS:PULSE_CSIHS; break;

}

pFL_RES = low; // RESET/FLMD0 = low
pmFL_FLMD0 = PM_OUT; // FLMD0 = Low output //[v1.01]
pFL_FLMD0 = low;
pmFL_FLMD1 = PM_OUT; // FLMD1 = Low output //[v1.01]
pFL_FLMD1 = 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(tRC); // start "tRC" wait timer
fl_wait((tRP+tRPE)/2); // wait

if (fl_if == FLIF_UART){
init_fl_uart(); // Initialize UARTh.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、V850ES/Sx3のフラッシュ・メモリ・プログラミング・モード引き込み用端子0 (FLMD0) にパルスを入力することで通信方式を決定します。FLMD0によるパルス制御に関して、FLMD0パルスのハイ / ロー・レベルはそれぞれV_{DD}/GND電圧となります。V850ES/Sx3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 V850ES/Sx3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 (UARTA0)	0	TXDA0 (P30), RXDA0 (P31)
3 線式シリアル I/O (CSIB0)	8	SOB0 (P41), SIB0 (P40), $\overline{\text{SCKB0}}$ (P42)
3 線式シリアル I/O (CSIB3)	9	SOB3 (P911), SIB3 (P910), $\overline{\text{SCKB3}}$ (P912)
3 線式シリアル I/O ハンドシェイク対応 (CSIB0 + HS)	11	SOB0 (P41), SIB0 (P40), $\overline{\text{SCKB0}}$ (P42), HS (PCM0)
3 線式シリアル I/O ハンドシェイク対応 (CSIB3 + HS)	12	SOB3 (P911), SIB3 (P910), $\overline{\text{SCKB3}}$ (P912), HS (PCM0)
(設定禁止)	その他	-

2.6 UART通信方式

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

表2 - 4 UART通信の通信条件

項目	内容
ボー・レート	9600 / 19200 / 31250 / 38400 / 57600 / 76800 / 115200 / 128000 / 153600 bps のいずれかから選択 (デフォルトは 9600 bps)
パリティ・ビット	なし
データ長	8 ビット (LSB 先頭)
ストップ・ビット	1 ビット

CSI通信では、常にプログラマがマスタになるので、V850ES/Sx3での書き込みや消去に関しては、プログラマ側から処理が正常に終了したかどうかを確認する必要があります。しかし、UART通信では、マスタとスレーブの関係を入れ換えながら通信を行うため、CSI + HS通信のように1端子を余分に使用せずに最適なタイミングでの通信が可能です。

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

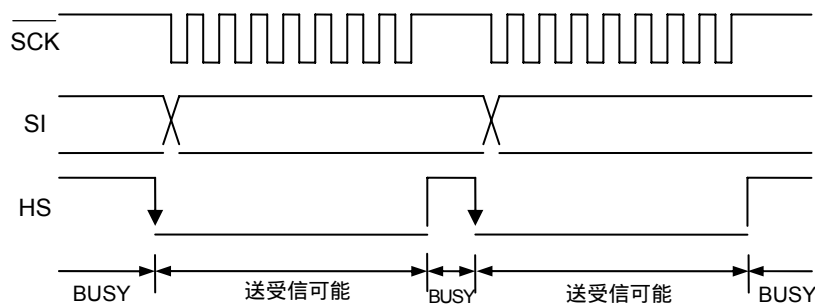
2.7 3線式シリアルI/O ハンドシェイク対応 (CSI + HS) 通信方式

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

HS端子は、V850ES/Sx3がデータ送受信可能な状態となったときに立ち下がります (ロウ・レベル)。プログラマは、HS端子の立ち下がり (ロウ・レベル) を確認してから、V850ES/Sx3に対してコマンドなどの送受信を開始してください。

通信のデータ形式は8ビット単位のMSB先頭です。また、クロック周波数は5 MHz以下にしてください。

図2 - 7 CSI + HS通信のタイミング・チャート



2.8 3線式シリアルI/O (CSI) 通信方式

CSI通信では、SCK, SO, SIの3端子を使用します。プログラマが常にマスタとなるため、V850ES/Sx3が送受信可能になっていない状態のときにSCK端子で送信した場合には、正常に通信できない場合があります。

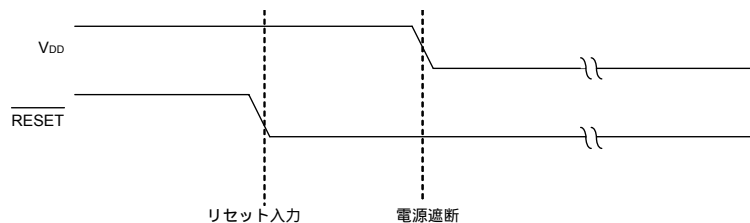
通信のデータ形式は8ビット単位のMSB先頭です。また、クロック周波数は5 MHz以下にしてください。

2.9 ターゲットの電源遮断処理

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

注意 コマンド処理中の電源遮断およびリセット入力は禁止です。

図2 - 8 フラッシュ・メモリ・プログラミングモードの終了手順



2.10 フラッシュ・メモリの操作方法

V850ES/Sx3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V850ES/Sx3に送信し、V850ES/Sx3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 V850ES/Sx3 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。
読み出し	読み出し	指定したフラッシュ・メモリの領域のデータを読み出します。
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	V850ES/Sx3 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。

2.11 コマンド一覧

プログラマで使用されるコマンドの一覧と機能を次に示します。

表2 - 6 プログラマからV850ES/Sx3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	V850ES/Sx3 の発振周波数を指定します。
9AH	Baud Rate Set	UART 選択時のボー・レートを設定します。
20H	Chip Erase	全フラッシュ・メモリを消去します。
22H	Block Erase	指定された領域のフラッシュ・メモリを消去します。
40H	Programming	フラッシュ・メモリの指定された領域にデータを書き込みます。
13H	Verify	フラッシュ・メモリの指定された領域の内容とプログラマから送信されたデータを比較します。
32H	Block Blank Check	指定されたブロックのフラッシュ・メモリの消去状態をチェックします。
C0H	Silicon Signature	V850ES/Sx3 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	V850ES/Sx3 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。
50H	Read	指定したフラッシュ・メモリの領域のデータを読み出します。

2. 12 ステータス一覧

プログラマがV850ES/Sx3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でエラー発生
10H	Protect error	Security Setコマンドで禁止した処理を実行しようとした場合のエラー
15H	否定応答 (NACK)	否定応答
1AH	MRG10 error	消去エラー
1BH	MRG11 error	データ書き込み時の内部ペリファイ・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
FFH	処理中 (BUSY)	ビジー応答 ^註

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

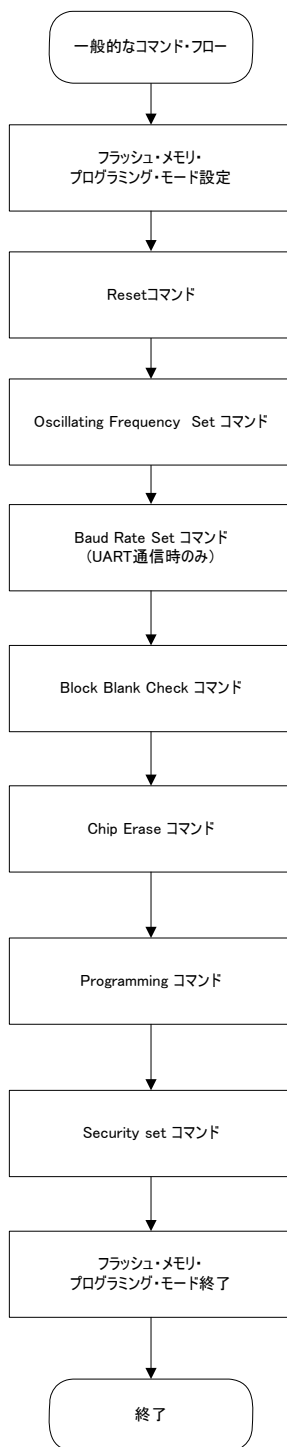
なお, このマニュアルではChecksum errorやNACKを受信した際は即時異常終了として扱っていますが, 実際にプログラマを作る際は, Checksum errorやNACKが発生したコマンド送信直前のウェイトおよびHS端子のBUSYチェックからリトライしても構いません。ただし, 無限にリトライを繰り返さないようにリトライの回数制限を設けることを推奨します。

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, HS端子のタイムアウト, UART通信時のデータ・フレーム受信のタイムアウトなど) が発生した場合は, 一度V850ES/Sx3に対して電源遮断処理 (2.9 ターゲットの電源遮断処理参照) を行ってから改めて接続することを推奨します。

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



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

第4章 コマンド/データ・フレーム・フォーマット

プログラマとV850ES/Sx3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。プログラマからV850ES/Sx3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 (ポローは無視)。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。

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

よって、このデータ・フレームは最終的に次のようになります。

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

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

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

本来なら 1BH

4.1 コマンド・フレーム送信処理

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

- ・UART通信方式の場合は、6.1 **コマンド・フレーム送信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.1 **コマンド・フレーム送信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.1 **コマンド・フレーム送信処理のフロー・チャート**をお読みください。

4.2 データ・フレーム送信処理

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

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

- ・UART通信方式の場合は、6.2 **データ・フレーム送信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.2 **データ・フレーム送信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.2 **データ・フレーム送信処理のフロー・チャート**をお読みください。

4.3 データ・フレーム受信処理

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

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

- ・UART通信方式の場合は、6.3 **データ・フレーム受信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.3 **データ・フレーム受信処理のフロー・チャート**をお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.3 **データ・フレーム受信処理のフロー・チャート**をお読みください。

第5章 コマンド処理説明

5.1 Statusコマンド

5.1.1 説明

書き込み / 消去などの各コマンド発行後のV850ES/Sx3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V850ES/Sx3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、V850ES/Sx3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V850ES/Sx3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 (V850ES/Sx3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、V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、Statusコマンドを使用しません。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.4 Statusコマンドをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.4 Statusコマンドをお読みください。

注意 UART通信の場合は、書き込み / 消去などの各コマンド送信後、一定時間内にV850ES/Sx3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V850ES/Sx3間の通信が確立されたことを確認します。

V850ES/Sx3との通信方式にUART通信を選択した場合、プログラマとV850ES/Sx3は同じボー・レートである必要がありますが、V850ES/Sx3は自身の動作周波数が判別できないためにボー・レートが設定できません。よって、プログラマから9600 bpsでの“00H”を2回送信し、V850ES/Sx3はその“00H”のロウ・レベル幅を測定し2回の平均値を計算することで初めて自身の動作周波数を判別できます。それによって、ボー・レートの設定が可能になり同期検出が行えるようになります。

5.2.2 コマンド・フレームとステータス・フレーム

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

図5-3 Resetコマンド・フレーム (プログラマからV850ES/Sx3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 (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・ UART通信方式の場合は、6.4 Resetコマンドをお読みください。
- ・ 3線式シリアルI/O ハンドシェーク対応 (CSI+HS) 通信方式の場合は、7.5 Resetコマンドをお読みください。
- ・ 3線式シリアルI/O (CSI) 通信方式の場合は、8.5 Resetコマンドをお読みください。

5.3 Baud Rate Setコマンド

5.3.1 説明

UART通信でのボー・レートの変更を行います (初期値9600 bps)。

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

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

UART通信時以外で、Baud Rate Setコマンドを送信した場合、V850ES/Sx3は無視します。

5.3.2 コマンド・フレームとステータス・フレーム

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

図5-5 Baud Rate Setコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

備考 D01 : ボー・レート選択値

D01 値	03H	04H	05H	06H	07H	08H	09H	0AH	0BH
ボー・レート (bps)	9600	19200	31250	38400	76800	153600	57600	115200	128000

図5-6 Baud Rate Setコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1 : 同期検出結果

通信方式ごとの、プログラマとV850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.5 Baud Rate Setコマンドをお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI + HS) 通信方式の場合は、Baud Rate Setコマンドを使用しません。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、Baud Rate Setコマンドを使用しません。

5.4 Oscillating Frequency Setコマンド

5.4.1 説明

V850ES/Sx3の発振周波数のデータを設定します。

実際にV850ES/Sx3のX1端子に入力されているクロックの周波数を指定してください。

V850ES/Sx3はこのコマンドで指定したクロックの周波数によりCPU動作クロックのてい倍率を自動設定します。

したがって、Oscillating Frequency Setコマンド実行の前後で、ウェイト算出用の基準クロックが異なりますのでご注意ください。

5.4.2 コマンド・フレームとステータス・フレーム

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

図5-7 Oscillating Frequency Setコマンド・フレーム (プログラマからV850ES/Sx3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 : 4 MHzの場合

D01 = 04H

D02 = 00H

D03 = 00H

D04 = 04H

発振周波数 = 4 × 0.1 × 10⁴ = 4000 kHz = 4 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5-8 Oscillating Frequency Setコマンドに対するステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・ UART通信方式の場合は、6.6 Oscillating Frequency Set**コマンド**をお読みください。
- ・ 3線式シリアルI/O ハンドシェーク対応 (CSI + HS) 通信方式の場合は、7.6 Oscillating Frequency Set**コマンド**をお読みください。
- ・ 3線式シリアルI/O (CSI) 通信方式の場合は、8.6 Oscillating Frequency Set**コマンド**をお読みください。

5.5 Chip Eraseコマンド

5.5.1 説明

全フラッシュ・メモリの内容を消去します。また、チップ消去処理によりセキュリティ設定処理で設定されたすべての情報を初期化できます。ただし、セキュリティ設定により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 - 9, そのコマンドに対するステータス・フレームは, 図5 - 10のようになります。

図5 - 9 Chip Eraseコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

図5 - 10 Chip Eraseコマンドに対するステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート, コマンド処理のフロー・チャート, サンプル・プログラムについては, 次の節をお読みください。

- ・UART通信方式の場合は, 6.7 Chip Eraseコマンドをお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は, 7.7 Chip Eraseコマンドをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は, 8.7 Chip Eraseコマンドをお読みください。

5.6 Block Eraseコマンド

5.6.1 説明

指定したブロックのフラッシュ・メモリの内容を消去します。

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

5.6.2 コマンド・フレームとステータス・フレーム

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

図5-11 Block Eraseコマンド・フレーム (プログラマからV850ES/Sx3へ)

SOH	LEN	COM	Data						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5-12 Block Eraseコマンドに対するステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.8 Block Eraseコマンドをお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI + HS) 通信方式の場合は、7.8 Block Eraseコマンドをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.8 Block Eraseコマンドをお読みください。

5.7 Programmingコマンド

5.7.1 説明

書き込み開始アドレス, 書き込み終了アドレスを送信したあとに, 書き込みバイト数分のデータを送信します。それにより, ユーザ・プログラムをフラッシュ・メモリに書きこみ, 内部ベリファイを行います。

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

最終データ送信後のステータス・フレーム (ST1, ST2) が両方ともACKであれば, V850ES/Sx3のファームウェアは自動的に内部ベリファイを実行するので, さらにこの内部ベリファイに対するStatusコマンドの送信が必要となります。

5.7.2 コマンド・フレームとステータス・フレーム

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

図5 - 13 Programmingコマンド・フレーム (プログラマからV850ES/Sx3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 - 14 Programmingコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(a) : コマンド受信結果

5.7.3 データ・フレームとステータス・フレーム

書き込みを行うデータのデータ・フレームは図5 - 15, そのデータに対するステータス・フレームは図5 - 16のようになります。

図5 - 15 書き込みを行うデータ・フレーム (プログラマからV850ES/Sx3へ)

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

備考 Write Data : 書き込むユーザ・プログラム

図5 - 16 データ・フレームに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(b) : データ受信確認結果
ST2(b) : 書き込み結果

5.7.4 全データ転送完了とステータス・フレーム

全データ転送完了後のステータス・フレームは図5-17のようになります。

図5-17 全データ転送完了後のステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.9 Programmingコマンドをお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.9 Programmingコマンドをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.9 Programmingコマンドをお読みください。

5.8 Verifyコマンド

5.8.1 説明

指定したアドレス範囲のデータに対して、プログラマから送信したデータとV850ES/Sx3から読み出したデータ (リード・レベル) を比較し、一致しているかを確認します。

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

5.8.2 コマンド・フレームとステータス・フレーム

Verifyコマンドのコマンド・フレームは図5 - 18, そのコマンドに対するステータス・フレームは図5 - 19のようになります。

図5 - 18 Verifyコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

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

図5 - 19 Verifyコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(a) : コマンド受信結果

5.8.3 データ・フレームとステータス・フレーム

ベリファイを行うデータのデータ・フレームは図5 - 20, そのデータに対するステータス・フレームは図5 - 21のようになります。

図5 - 20 ベリファイを行うデータのデータ・フレーム (プログラマからV850ES/Sx3へ)

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

備考 Verify Data : ベリファイを行うユーザ・プログラム

図5 - 21 データ・フレームに対するステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

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

5.9 Block Blank Checkコマンド

5.9.1 説明

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

5.9.2 コマンド・フレームとステータス・フレーム

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

図5 - 22 Block Blank Checkコマンド・フレーム (プログラマからV850ES/Sx3へ)

SOH	LEN	COM	Data						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 - 23 Block Blank Checkコマンドに対するステータス・フレーム (V850ES/Sx3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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

- ・UART通信方式の場合は、6.11 Block Blank Checkコマンドをお読みください。
- ・3線式シリアルI/O ハンドシェイク対応 (CSI+HS) 通信方式の場合は、7.11 Block Blank Checkコマンドをお読みください。
- ・3線式シリアルI/O (CSI) 通信方式の場合は、8.11 Block Blank Checkコマンドをお読みください。

5. 10 Silicon Signatureコマンド

5. 10. 1 説 明

デバイスの書き込みプロトコル情報 (シリコン・シグネチャ) を読み出します。

たとえば、プログラマがV850ES/Sx3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、適切なプロトコルを選択します。

5. 10. 2 コマンド・フレームとステータス・フレーム

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

図5 - 24 Silicon Signatureコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

図5 - 25 Silicon Signatureコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1 : コマンド受信結果

5. 10. 3 シリコン・シグネチャ・データ・フレーム

シリコン・シグネチャ・データのデータ・フレームは図5 - 26のようになります。

図5 - 26 シリコン・シグネチャ・データ・フレーム (V850ES/Sx3からプログラマへ)

STX	LEN	Data													SUM	ETX
02H	20H	VEN	MET	MSC	DEC1	DEC2	UAE	INVALID DATA	DEV	SCF	BOT	RVAL	RVAM	RVAH	Checksum	03H

- 備考1.
- VEN : ベンダー・コード (10H)
 - MET : マクロ拡張コード
 - MSC : マクロ機能コード
 - DEC1 : デバイス拡張コード1
 - DEC2 : デバイス拡張コード2
 - UAE : ユーザ・フラッシュROM最終アドレス
 - INVALID DATA : 長さ8バイトの無効データです。
 - 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
 - RVAL : リセット・ベクタ・アドレスL (ビット7-0) (00H固定)
 - RVAM : リセット・ベクタ・アドレスM (ビット15-8) (00H固定)
 - RVAH : リセット・ベクタ・アドレスH (ビット23-16) (00H固定)

備考2. 図5 - 26のうち、ブート・ブロック番号 (BOT), リセット・ベクタ・アドレス (RVAL-RVAH) 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。次に例を示します。

表5 - 1 シリコン・シグネチャ・データの例 (μ PD70F3368 (V850ES/SJ3) の場合)

フィールド名	内 容	長さ (バイト)	シグネチャ・データ ^{※1}	実際の値	パリティ付加
VEN	ベンダー・コード	1	10H (00010000B)	10H	あり
MET	マクロ拡張コード (固定値)	1	7FH (01111111B)	7FH	あり
MSC	マクロ機能 (固定値)	1	04H (00000100B)	04H	あり
DEC1	デバイス拡張コード1 (固定値)	1	ECH (11101100B)	6CH	あり
DEC2	デバイス拡張コード2 (固定値)	1	7FH (01111111B)	7FH	あり
UAE	ユーザ・フラッシュROM最終アドレス	4	7FH (01111111B) 7FH (01111111B) BFH (10111111B) 80H (10000000B)	FFFFFFH ^{※2}	あり
INVALID DATA	無効データ	8	-	-	-
DEV	デバイス名	10	C4H (11000100B = 'D') 37H (00110111B = '7') B0H (10110000B = '0') 46H (01000110B = 'F') B3H (10110011B = '3') B3H (10110011B = '3') B6H (10110110B = '6') 38H (00111000B = '8') 20H (00100000B = ' ') 20H (00100000B = ' ')	'D' '7' '0' 'F' '3' '3' '6' '8' ' ' ' '	あり
SCF	セキュリティ・フラグ情報	1	任意	任意	あり ^{※3}
BOT	ブート・ブロック・クラスタ 最終ブロック番号	1	任意	任意	なし
RVAL	リセット・ベクタ・アドレスL (ビット7-ビット0)	1	00H固定	00H固定	なし
RVAM	リセット・ベクタ・アドレスM (ビット15-ビット8)	1	00H固定	00H固定	なし
RVAH	リセット・ベクタ・アドレスH (ビット23-ビット16)	1	00H固定	00H固定	なし

注1. 0や1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)

(注2, 3は次ページ以降にあります。)

よって、ENDフィールドに “ 7F 7F BF 80 ” が与えられた場合、実際の最終アドレスは、000FFFFFH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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

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

5. 10. 4 V850ES/Sx3シリコン・シグネチャー一覧

表5 - 2 V850ES/Sx3のシリコン・シグネチャ・データ一覧

項目	内容	長さ (バイト)	データ (Hex)
ベンダー・コード	ベンダー・コード	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1	デバイス情報1	1	EC
デバイス情報2	デバイス情報2	1	7F
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) × 4	4	注1
デバイス名 (μPD)	70F3333, 70F3334, 70F3335, 70F3336, 70F3340, 70F3341, 70F3342, 70F3343, 70F3344, 70F3345, 70F3346, 70F3347, 70F3348, 70F3350, 70F3351, 70F3352, 70F3353, 70F3354, 70F3355, 70F3356, 70F3357, 70F3358, 70F3364, 70F3365, 70F3366, 70F3367, 70F3368	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・ブロック・クラスタの最終ブロック 番号	1	任意
リセット・ベクタ・ア ドレス	リセット・ベクタ・アドレス	3	任意

注1. 内蔵フラッシュROMの最終アドレス・リスト

項目	内容	長さ (バイト)	データ (Hex)
内蔵フラッシュROM の最終アドレス	256 Kバイト (3FFFFH)	4	7F7F8F80
	384 Kバイト (5FFFFH)		7F7F9780
	512 Kバイト (7FFFFH)		7F7F1F80
	640 Kバイト (9FFFFH)		7F7FA780
	768 Kバイト (BFFFFH)		7F7F2F80
	1024 Kバイト (FFFFFFH)		7F7FBF80

(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 (1/2)

愛 称	デバイス名	長さ (バイト)	実際の値									
			〔 上段 : シグネチャ・データの低位7ビット 〕							下段 : 文字コード		
V850ES/SG3	D70F3333	10	C4	37	B0	46	B3	B3	B3	B3	20	20
			D	7	0	F	3	3	3	3	-	-
	D70F3334		C4	37	B0	46	B3	B3	B3	34	20	20
			D	7	0	F	3	3	3	4	-	-
	D70F3335		C4	37	B0	46	B3	B3	B3	B5	20	20
			D	7	0	F	3	3	3	5	-	-
	D70F3336		C4	37	B0	46	B3	B3	B3	B6	20	20
			D	7	0	F	3	3	3	6	-	-
	D70F3340		C4	37	B0	46	B3	B3	34	B0	20	20
			D	7	0	F	3	3	4	0	-	-
	D70F3341		C4	37	B0	46	B3	B3	34	31	20	20
			D	7	0	F	3	3	4	1	-	-
	D70F3342		C4	37	B0	46	B3	B3	34	32	20	20
			D	7	0	F	3	3	4	2	-	-
	D70F3343		C4	37	B0	46	B3	B3	34	B3	20	20
			D	7	0	F	3	3	4	3	-	-
	D70F3350		C4	37	B0	46	B3	B3	B5	B0	20	20
			D	7	0	F	3	3	5	0	-	-
	D70F3351		C4	37	B0	46	B3	B3	B5	31	20	20
			D	7	0	F	3	3	5	1	-	-
D70F3352	C4	37	B0	46	B3	B3	B5	32	20	20		
	D	7	0	F	3	3	5	2	-	-		
D70F3353	C4	37	B0	46	B3	B3	B5	B3	20	20		
	D	7	0	F	3	3	5	3	-	-		
V850ES/SJ3	D70F3344	C4	37	B0	46	B3	B3	34	34	20	20	
		D	7	0	F	3	3	4	4	-	-	
	D70F3345	C4	37	B0	46	B3	B3	34	B5	20	20	
		D	7	0	F	3	3	4	5	-	-	
	D70F3346	C4	37	B0	46	B3	B3	34	B6	20	20	
		D	7	0	F	3	3	4	6	-	-	
	D70F3347	C4	37	B0	46	B3	B3	34	37	20	20	
		D	7	0	F	3	3	4	7	-	-	
D70F3348	C4	37	B0	46	B3	B3	34	38	20	20		
	D	7	0	F	3	3	4	8	-	-		

デバイス名リスト (2/2)

愛 称	デバイス名	長さ (バイト)	実際の値									
			上段 : シグネチャ・データの低位7ビット							下段 : 文字コード		
V850ES/SJ3	D70F3354	10	C4	37	B0	46	B3	B3	B5	34	20	20
			D	7	0	F	3	3	5	4	-	-
	D70F3355		C4	37	B0	46	B3	B3	B5	B5	20	20
			D	7	0	F	3	3	5	5	-	-
	D70F3356		C4	37	B0	46	B3	B3	B5	B6	20	20
			D	7	0	F	3	3	5	6	-	-
	D70F3357		C4	37	B0	46	B3	B3	B5	37	20	20
			D	7	0	F	3	3	5	7	-	-
	D70F3358		C4	37	B0	46	B3	B3	B5	38	20	20
			D	7	0	F	3	3	5	8	-	-
	D70F3364		C4	37	B0	46	B3	B3	B6	34	20	20
			D	7	0	F	3	3	6	4	-	-
	D70F3365		C4	37	B0	46	B3	B3	B6	B5	20	20
			D	7	0	F	3	3	6	5	-	-
	D70F3366		C4	37	B0	46	B3	B3	B6	B6	20	20
			D	7	0	F	3	3	6	6	-	-
	D70F3367		C4	37	B0	46	B3	B3	B6	37	20	20
			D	7	0	F	3	3	6	7	-	-
	D70F3368		C4	37	B0	46	B3	B3	B6	38	20	20
			D	7	0	F	3	3	6	8	-	-

5. 11 Version Getコマンド

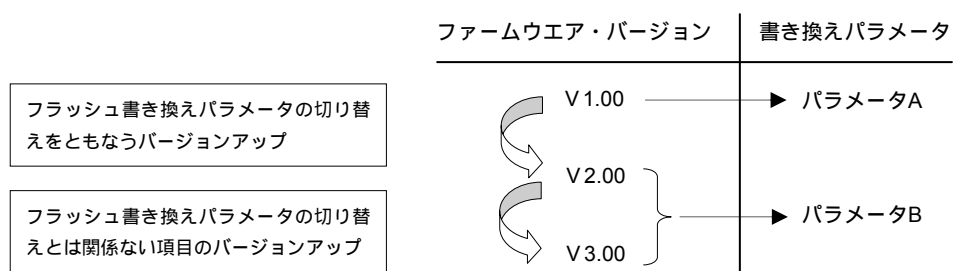
5. 11. 1 説 明

V850ES/Sx3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。

書き換え用パラメータをV850ES/Sx3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 - 28, そのコマンドに対するステータス・フレームは図5 - 29のようになります。

図5 - 28 Version Getコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

図5 - 29 Version Getコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1 : コマンド受信結果

5. 11. 3 バージョン・データ・フレーム

バージョン・データのデータ・フレームは図5 - 30のようになります。

図5 - 30 バージョン・データ・フレーム (V850ES/Sx3からプログラマへ)

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

備考 DV1 : デバイス・バージョン整数値
 DV2 : デバイス・バージョン小数点第一位
 DV3 : デバイス・バージョン小数点第二位
 FV1 : ファームウェア・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V850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

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

5. 12 Checksumコマンド

5. 12. 1 説 明

指定された領域のデータのチェックサム・データを取得します。

チェックサム計算の開始/終了アドレスは、ブロックの開始/終了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00Hから引き算したものです。

5. 12. 2 コマンド・フレームとステータス・フレーム

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

図5 - 31 Checksumコマンド・フレーム (プログラマからV850ES/Sx3へ)

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

備考 SAH-SAL : チェックサム計算開始アドレス

EAH-EAL : チェックサム計算終了アドレス

図5 - 32 Checksumコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1 : コマンド受信結果

5. 12. 3 チェックサム・データ・フレーム

チェックサム・データのデータ・フレームは図5 - 33のようになります。

図5 - 33 チェックサム・データ・フレーム (V850ES/Sx3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・データの下部8ビット

通信方式ごとの、プログラマとV850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

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

5.13 Security Setコマンド

5.13.1 説明

セキュリティに関する設定 (書き込み, ブロック消去, チップ消去などの許可/禁止) を行います。Security Setコマンドで, これらの設定を行うことで第三者からのフラッシュの書き換えを制限します。

注意 一度セキュリティ設定をした場合, セキュリティ・フラグの禁止から許可への変更はできません。セキュリティ・フラグの再設定を行う場合は, Chip Eraseコマンドの実行によって全セキュリティ・フラグの初期化をする必要があります (Block Eraseコマンドでは, セキュリティ・フラグの初期化はできません)。ただし, チップ消去禁止の設定をした場合, チップ消去自体が不可能になり, プログラマからは消去ができなくなります。プログラマの仕様としては, チップ消去禁止の設定を行う前に, 設定実行の再確認をすることを推奨します。

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

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

図5 - 34 Security Setコマンド・フレーム (プログラマからV850ES/Sx3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 - 35 Security Setコマンドに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(a) : コマンド受信結果

5. 13. 3 データ・フレームとステータス・フレーム

セキュリティ・データのデータ・フレームは図5 - 36, そのデータに対するステータス・フレームは図5 - 37 のようになります。

図5 - 36 セキュリティ・データ・フレーム (プログラマからV850ES/Sx3へ)

STX	LEN	Data					SUM	ETX
02H	05H	FLG	BOT	ADH	ADM	ADL	Checksum	03H

備考 FLG : セキュリティ・フラグ
 BOT : ブート・ブロック・クラスタ最終ブロック番号 (00H-7FH)
 ADH : リセット・ベクタ・ハンドラ・アドレス (ビット23 - ビット16) ^注
 ADM : リセット・ベクタ・ハンドラ・アドレス (ビット15 - ビット8) ^注
 ADL : リセット・ベクタ・ハンドラ・アドレス (ビット7 - ビット0) ^注

注 00Hを設定してください。

図5 - 37 セキュリティ・データ書き込みに対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(b) : セキュリティ・データ書き込み結果

5. 13. 4 内部ベリファイ確認とステータス・フレーム

内部ベリファイ確認に対するステータス・フレームは図5 - 38のようになります。

図5 - 38 内部ベリファイ確認に対するステータス・フレーム (V850ES/Sx3からプログラマへ)

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

備考 ST1(c) : 内部ベリファイ結果

セキュリティ・フラグ・フィールドの内容を次に示します。

表5 - 3 セキュリティ・フラグ・フィールドの内容

項 目	内 容
ビット7	1 固定
ビット6	
ビット5	
ビット4	ブート・ブロック・クラスタ書き換え禁止フラグ (1: 許可, 0: 禁止)
ビット3	リード禁止フラグ (1: 許可, 0: 禁止)
ビット2	書き込み禁止フラグ (1: 許可, 0: 禁止)
ビット1	ブロック消去禁止フラグ (1: 許可, 0: 禁止)
ビット0	チップ消去禁止フラグ (1: 許可, 0: 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・フィールドと各コマンドの禁止 / 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 : ブート指定以外の書き込みまたはブロック消去が可能			
	Programming	Chip Erase	Block Erase	Read
書き込み禁止	×		×	
チップ消去禁止		×	×	
ブロック消去禁止			×	
リード禁止				×
ブート・ブロック・ク ラスト書き換え禁止		×		

備考 セルフ・プログラミング・モードの場合については、各デバイスのユーザーズ・マニュアルを参照してください。

通信方式ごとの、プログラマとV850ES/Sx3間の処理手順チャート、コマンド処理のフロー・チャート、サンプル・プログラムについては、次の節をお読みください。

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

5. 14 Readコマンド

5. 14. 1 説 明

V850ES/Sx3のフラッシュROMからデータを読み出します。

読み出し開始 / 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. 14. 2 コマンド・フレームとステータス・フレーム

Readコマンドのコマンド・フレームは図5 - 39 , そのコマンドに対するステータス・フレームは図5 - 40のようになります。

図5 - 39 Readコマンド・フレーム (プログラムからV850ES/Sx3へ)

SOH	LEN	COM	コマンド情報						SUM	ETX
01H	07H	50H (Read)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

備考 SAH - SAL : 読み出し開始アドレス (ブロックの開始アドレス)
EAH - EAL : 読み出し終了アドレス (ブロックの終了アドレス)

図5 - 40 Readコマンドに対するステータス・フレーム (V850ES/Sx3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. 14. 3 データ・フレームとステータス・フレーム

読み出すデータのデータ・フレームは図5 - 41 , そのデータに対するステータス・フレームは図5 - 42のようになります。

図5 - 41 読み出すデータのデータ・フレーム (V850ES/Sx3からプログラマへ)

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

備考 Read Data : V850ES/Sx3から読み出したデータ

図5 - 42 読み出したデータに対するのステータス・フレーム (プログラムからV850ES/Sx3へ)

STX	LEN	Data	SUM	ETX
02H	01H	ST1(b)	Checksum	03H/17H

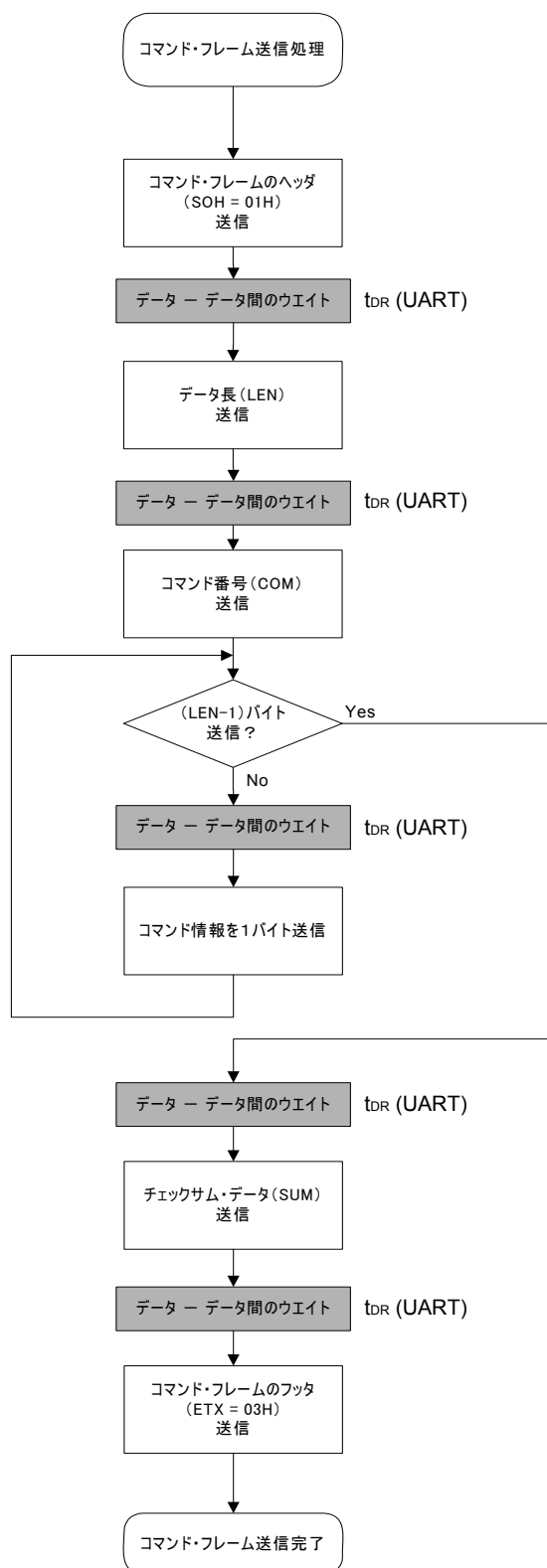
備考 ST1(b) : 読み出したデータに対するプログラマからのACK (06H) または
NACK (15H)

通信方式ごとの、プログラマとV850ES/Sx3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。

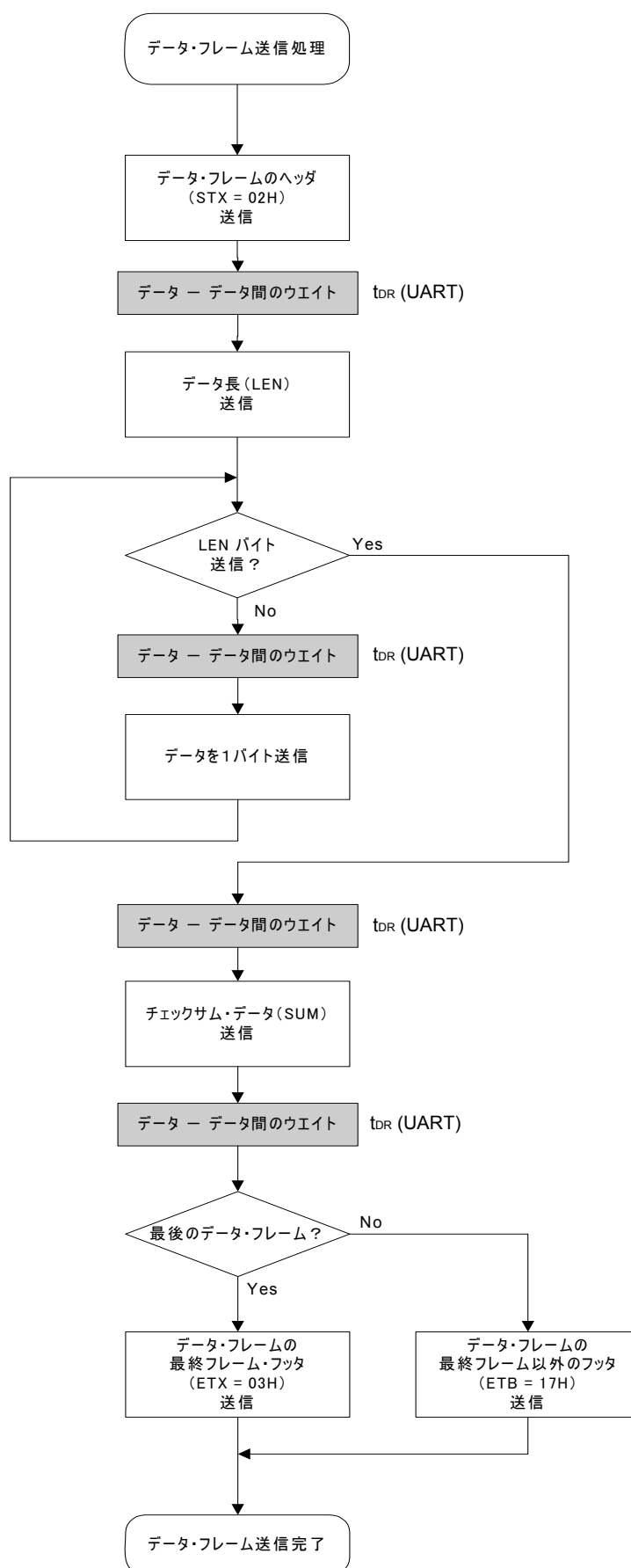
- ・ UART通信方式の場合は、6. 16 Readコマンドをお読みください。
- ・ 3線式シリアルI/O ハンドシェーク対応 (CSI + HS) 通信方式の場合は、7. 16 Readコマンドをお読みください。
- ・ 3線式シリアルI/O (CSI) 通信方式の場合は、8. 16 Readコマンドをお読みください。

第6章 UART通信方式

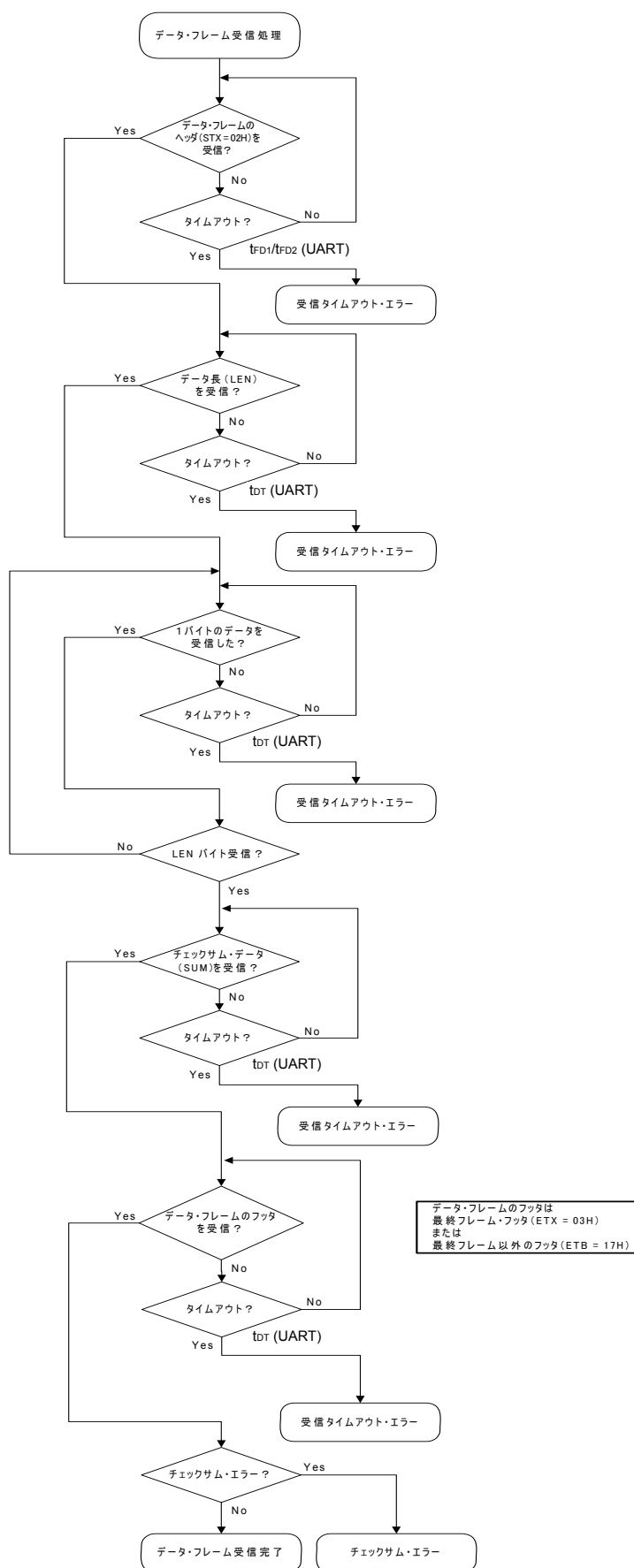
6.1 コマンド・フレーム送信処理のフロー・チャート



6.2 データ・フレーム送信処理のフロー・チャート



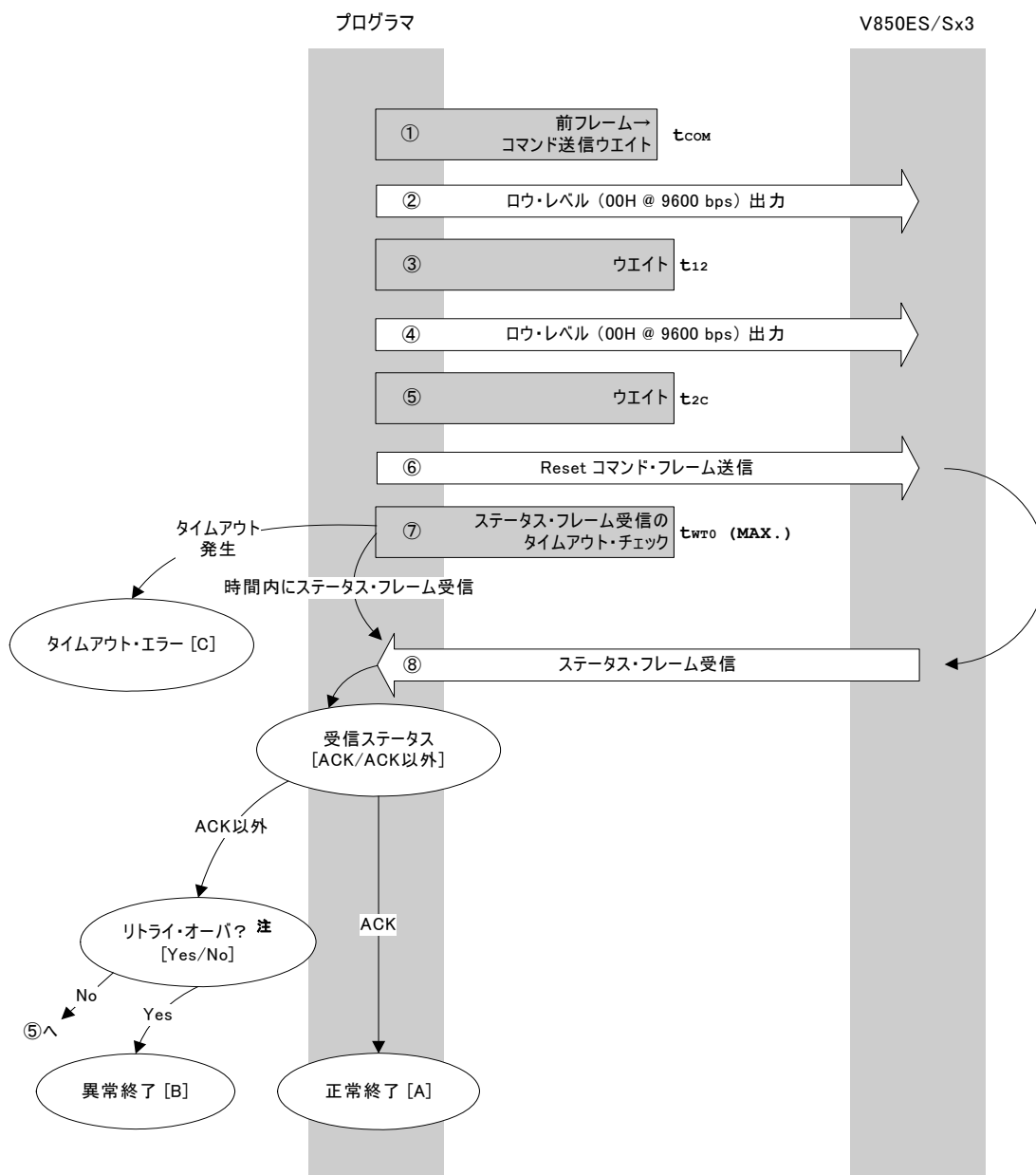
6.3 データ・フレーム受信処理のフロー・チャート



6.4 Resetコマンド

6.4.1 処理手順チャート

Resetコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

6.4.2 処理手順説明

直前のフレームからコマンド処理開始前のウエイトをします (ウエイト時間 t_{COM})。
 ロウ・レベル出力を行います (データ00Hを9600 bpsで送信)。
 ウエイトをします (ウエイト時間 t_{12})。
 ロウ・レベル出力を行います (データ00Hを9600 bpsで送信)。
 ウエイトをします (ウエイト時間 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_{WTO} (MAX.))。
 ステータス・コードをチェックします。

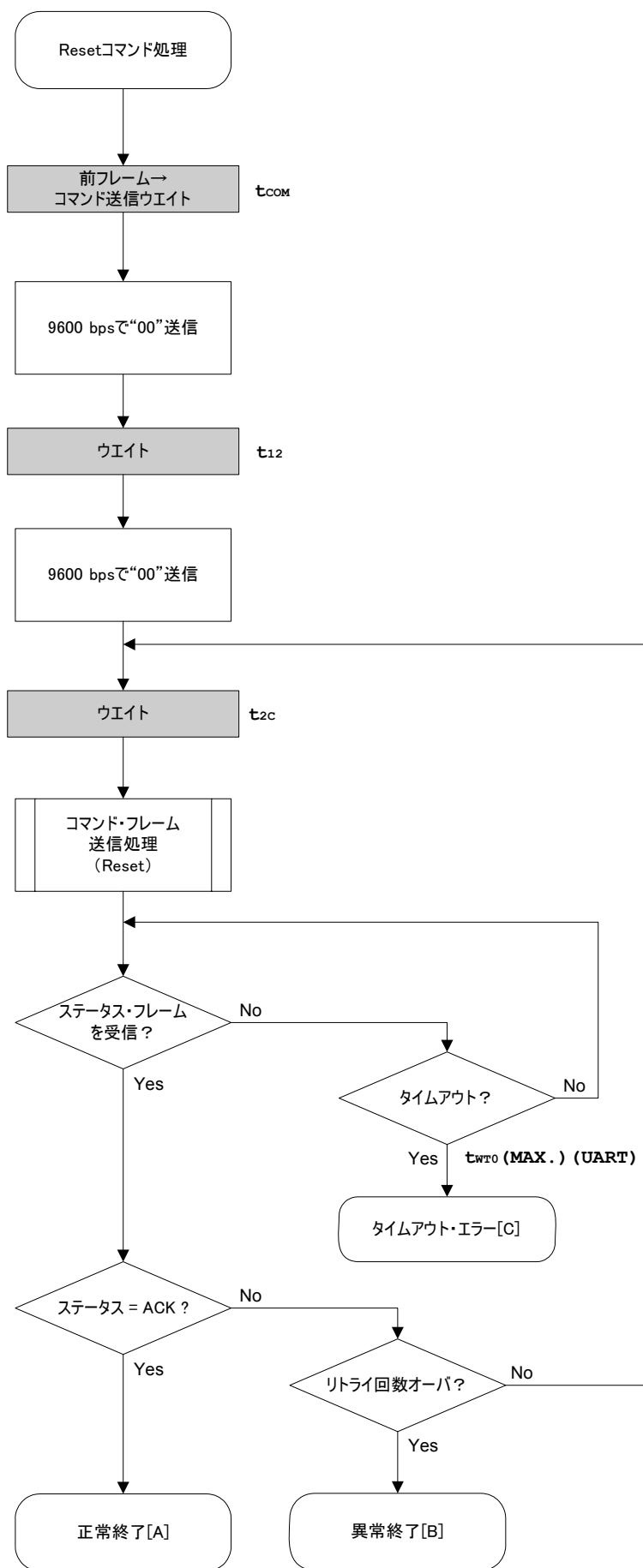
ST1 = ACKの場合 : **正常終了 [A]** です。

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V850ES/Sx3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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームが受信できませんでした。

6.4.4 フロー・チャート



6.4.5 サンプル・プログラム

Resetコマンド処理のサンプル・プログラムです。

```

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

    set_uart0_br(BR_9600); // change to 9600bps

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

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

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

        fl_wait(t2C);     // wait

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

        rc = get_sfrm_ua(fl_ua_sfrm, tWTO_MAX);
        if (rc == FLC_DFTO_ERR) // t.o. ?
            break; // yes // case [C]

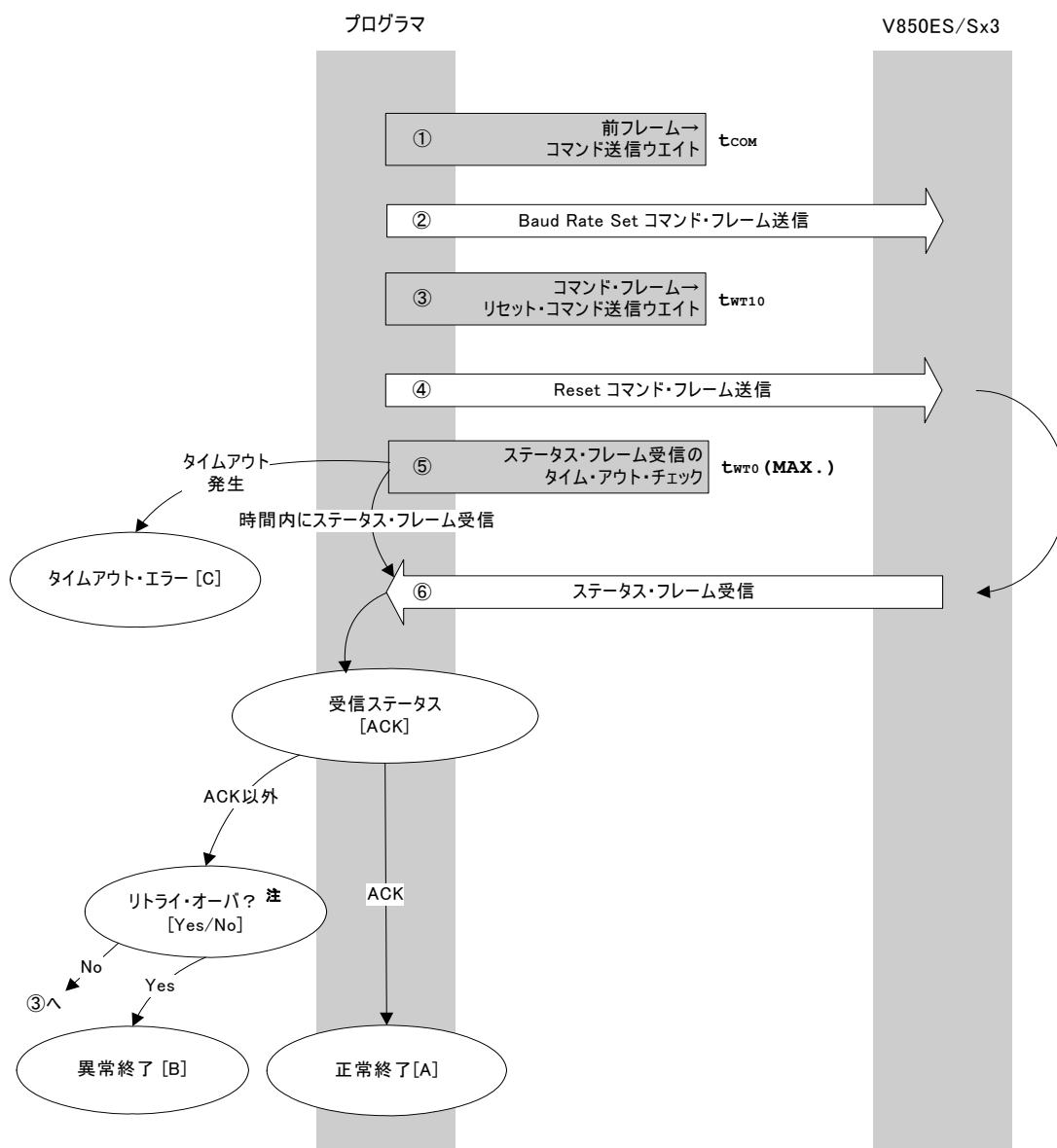
        if (rc == FLC_ACK){ // ACK ?
            break; // yes // case [A]
        }
        else{
            NOP();
        }
        //continue; // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //     case    FLC_NO_ERR:    return rc;    break; // case [A]
    //     case    FLC_DFTO_ERR: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

```

6.5 Baud Rate Setコマンド

6.5.1 処理手順チャート

Baud Rate Setコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

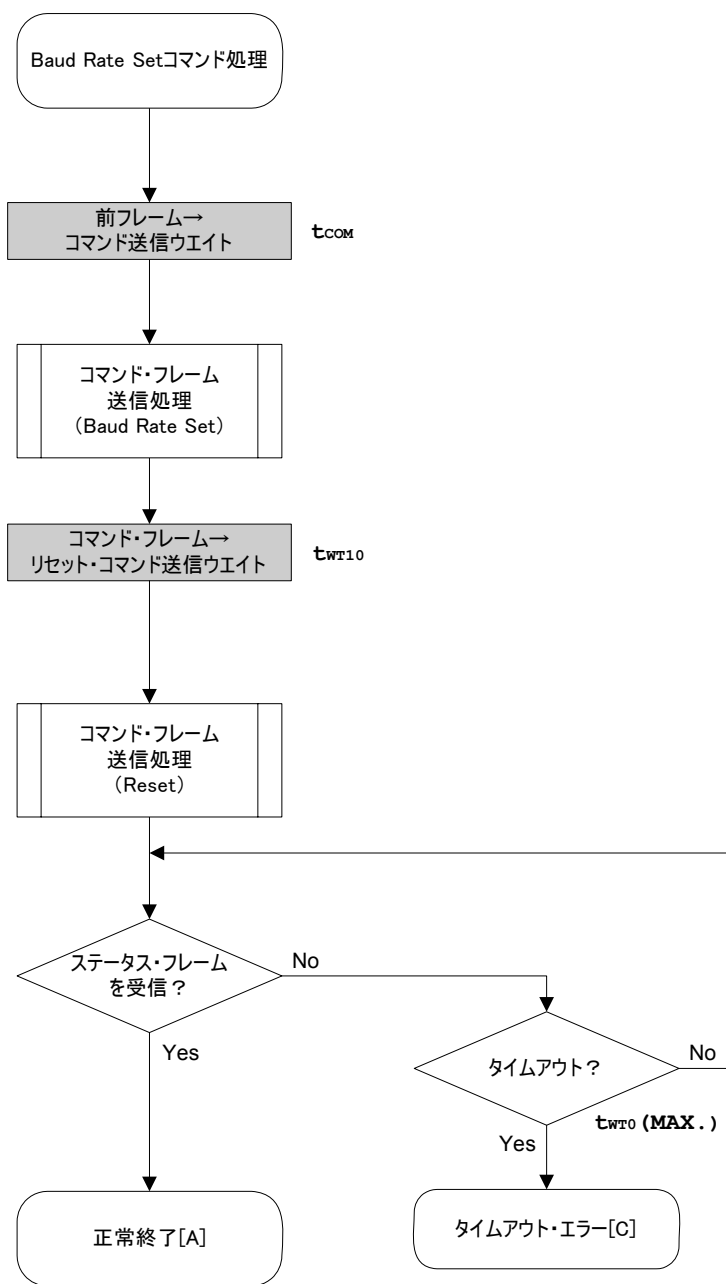
6.5.2 処理手順説明

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 **Baud Rate Setコマンド** を送信します。
 コマンド送信からリセット・コマンド送信までのウェイトをします (ウェイト時間 t_{WR10})。
 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。
 タイムアウト発生であれば **タイムアウト・エラー[C]** となります
 (タイムアウト時間 t_{WTO} (MAX.))。
 ステータス・コードはACKのはずですので、 **正常終了[A]** となります。

6.5.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、プログラマと V850ES/Sx3 間で UART 通信速度の同期が取れたことを示します。
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	<p>データ・フレームの受信でタイムアウトが発生しました。</p> <p>ただし、このコマンドにおいて V850ES/Sx3 では次の場合もこのエラーになります。</p> <ul style="list-style-type: none"> ・コマンド情報 (パラメータ) に不正があった場合 ・コマンド・フレームにチェックサム・エラーがあった場合 ・コマンド・フレームのデータ長 (LEN) が不正の場合 ・コマンド・フレームのフッタ (ETX) がない場合 ・ボー・レート設定後、16 回分のコマンド・フレーム・データを受信しても Reset コマンドが検出できなかった場合

6.5.4 フロー・チャート



6.5.5 サンプル・プログラム

Baud Rate Setコマンド処理のサンプル・プログラムです。

```

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

    switch(brid){
        default:
        case BR_9600:      br = 0x03;      break;
        case BR_19200:    br = 0x04;      break;
        case BR_31250:    br = 0x05;      break;
        case BR_38400:    br = 0x06;      break;
        case BR_76800:    br = 0x07;      break;
        case BR_153600:   br = 0x08;      break;

        case BR_57600:    br = 0x09;      break;
        case BR_115200:   br = 0x0a;      break;
        case BR_128000:   br = 0x0b;      break;

    }

    fl_cmd_prm[0] = br;          // "D01"

    fl_wait(tCOM);              // wait before sending command
    put_cmd_ua(FL_COM_SET_BAUDRATE, 2, fl_cmd_prm);
                                // send "Baudrate Set" command

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

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

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

        rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);          // get status frame
        if (rc){
            if (retry--){
                continue;
            }
            else{
                return rc;
            }
        }
        break;          // got ACK !!
    }

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

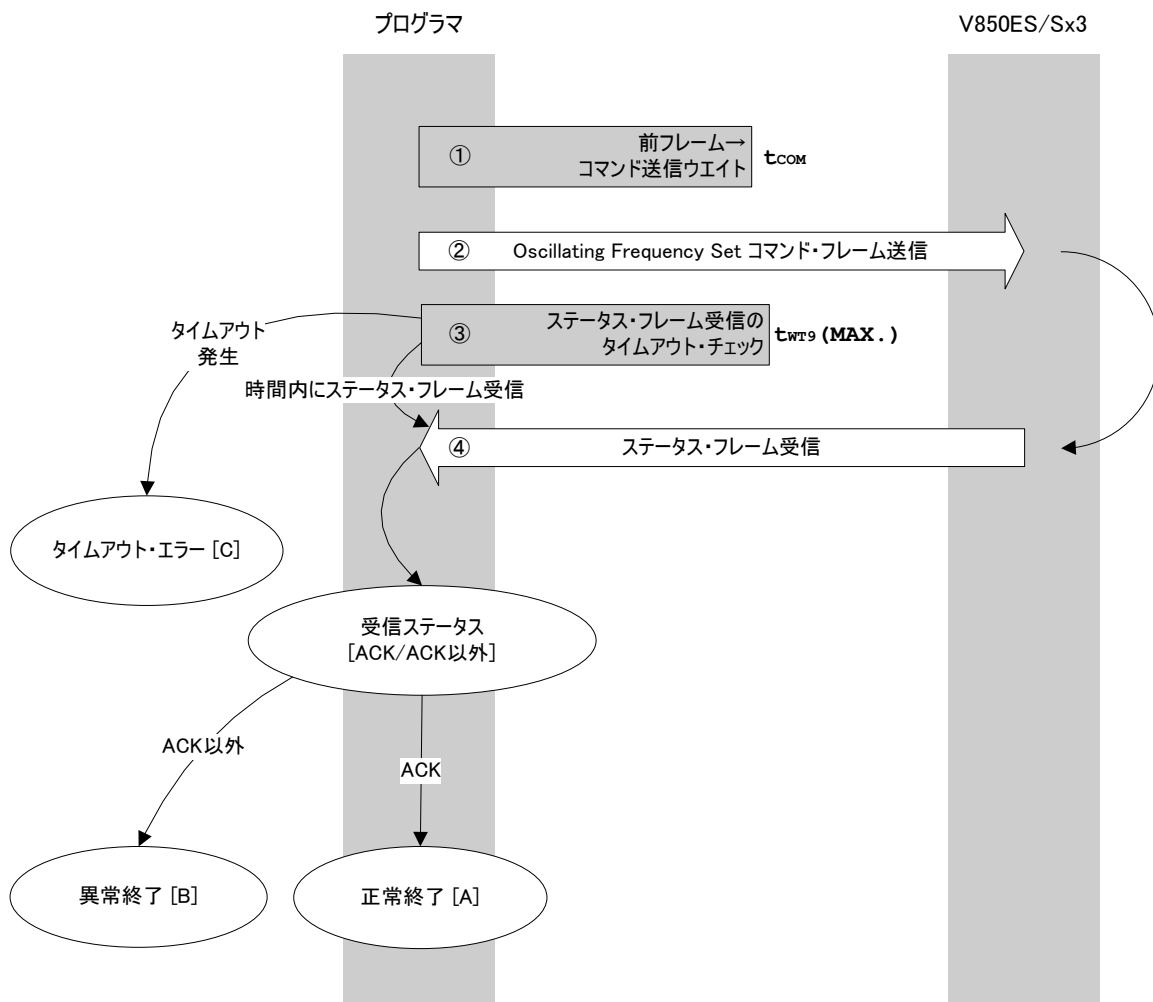
    return rc;
}

```

6.6 Oscillating Frequency Setコマンド

6.6.1 処理手順チャート

Oscillating Frequency Setコマンド処理手順



6.6.2 処理手順説明

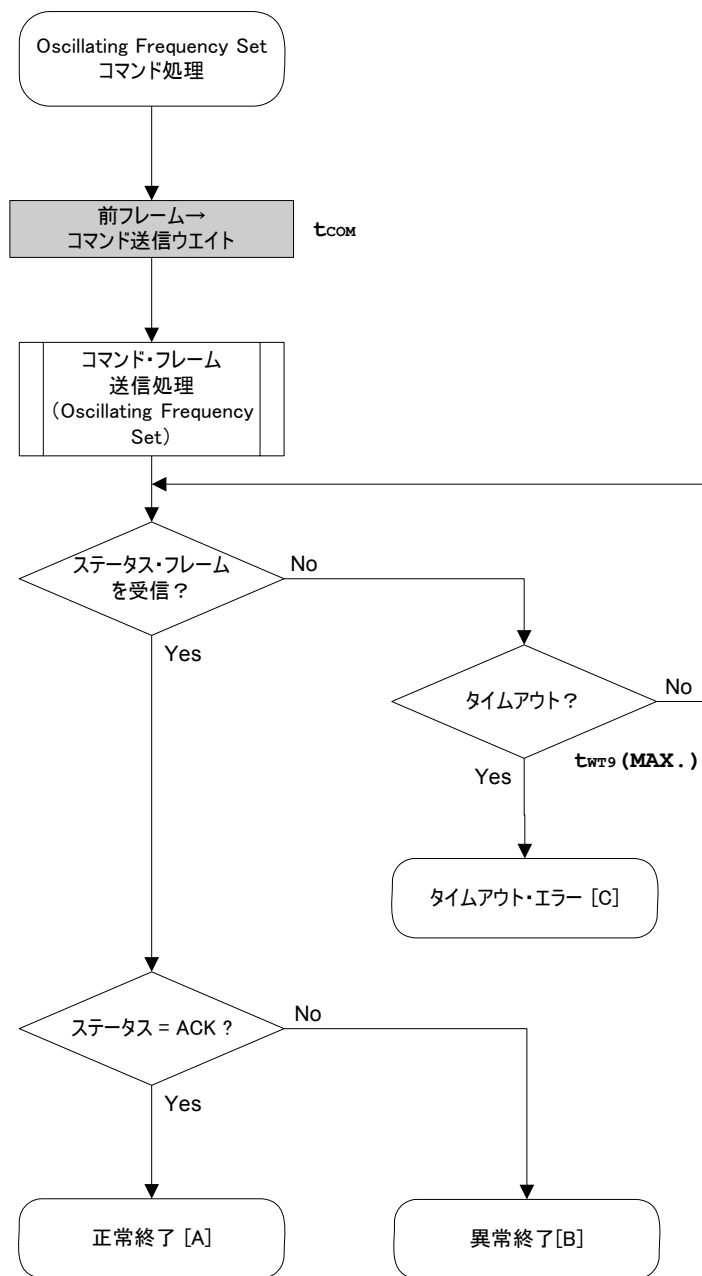
直前のフレームからコマンド送信までのウエイトをします (ウエイト時間 t_{COM})。コマンド・フレーム送信処理により、**Oscillating Frequency Setコマンド**を送信します。コマンド送信からステータス・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合は、**タイムアウト・エラー[C]**となります (タイムアウト時間 $t_{WT9} (MAX.)$)。ステータス・コードをチェックします。

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、V850ES/Sx3 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。

6.6.4 フロー・チャート



6.6.5 サンプル・プログラム

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

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

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

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

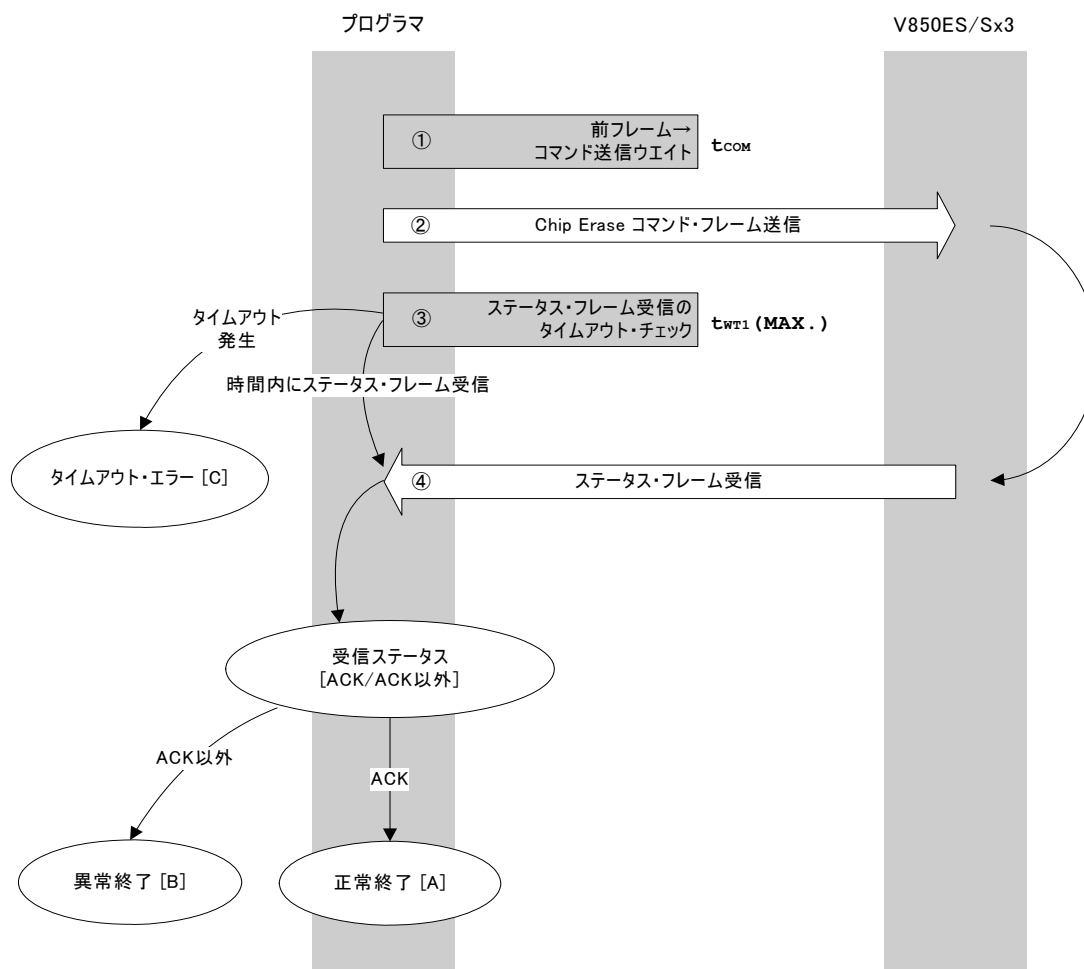
    rc = get_sfrm_ua(fl_ua_sfrm, tWT9_MAX); // get status frame
    // switch(rc) {
    //
    // case FLC_NO_ERR: return rc; break; // case [A]
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    // default: return rc; break; // case [B]
    // }

    return rc;
}
```

6.7 Chip Eraseコマンド

6.7.1 処理手順チャート

Chip Eraseコマンド処理手順



6.7.2 処理手順説明

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 $t_{WT1 (MAX.)}$)。
 ステータス・コードをチェックします。

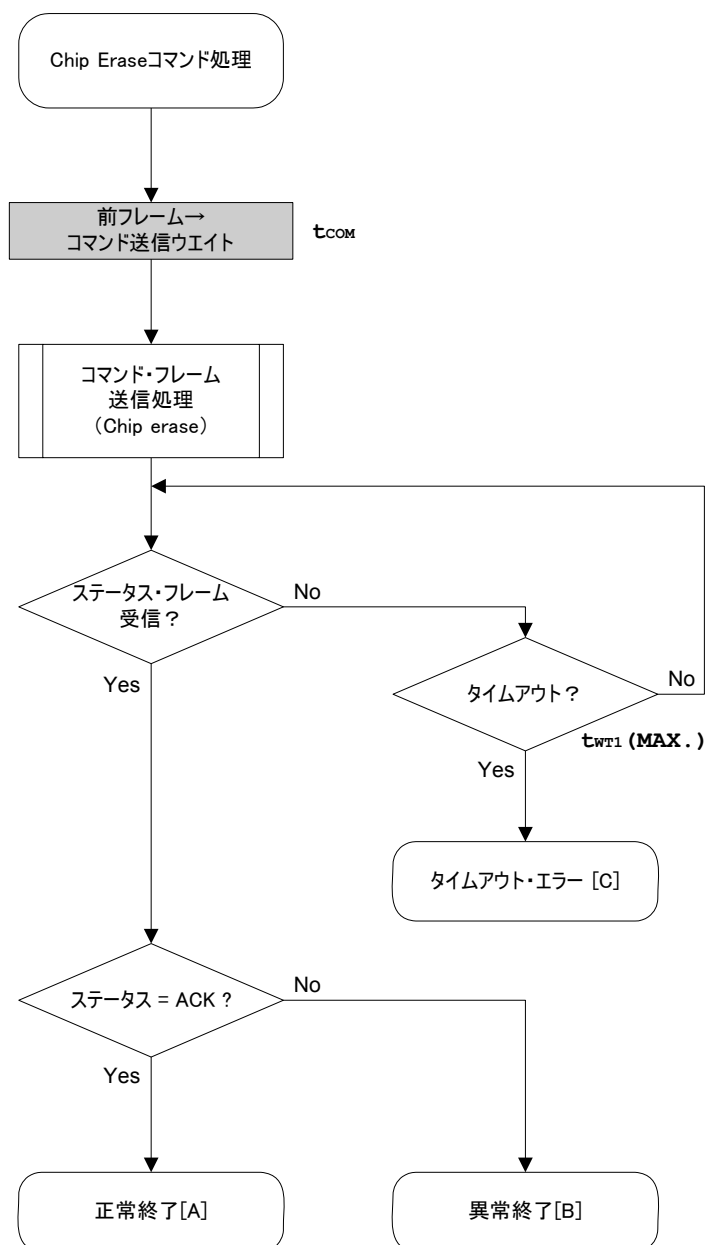
ST1 = ACKの場合 : **正常終了[A]** です。

ST1 = ACK以外の場合 : **異常終了[B]** です。

6.7.3 終了時の内容

終了内容	ステータス・コード	内 容	
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、チップ消去が正常に実行されたことを示します。	
異常終了 [B] チェックサム・エラー	プロテクト・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、Chip Erase コマンドが禁止となる設定になっています。
	否定応答 (NACK)	15H	・処理中にステータス・コマンド以外を受信しました。 ・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)。
	WRITE エラー	1CH	消去エラーが発生しました。
	MRG10 エラー	1AH	
MRG11 エラー	1BH		
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

Chip Eraseコマンド処理のサンプル・プログラムです。

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

    fl_wait(tCOM); // wait before sending command

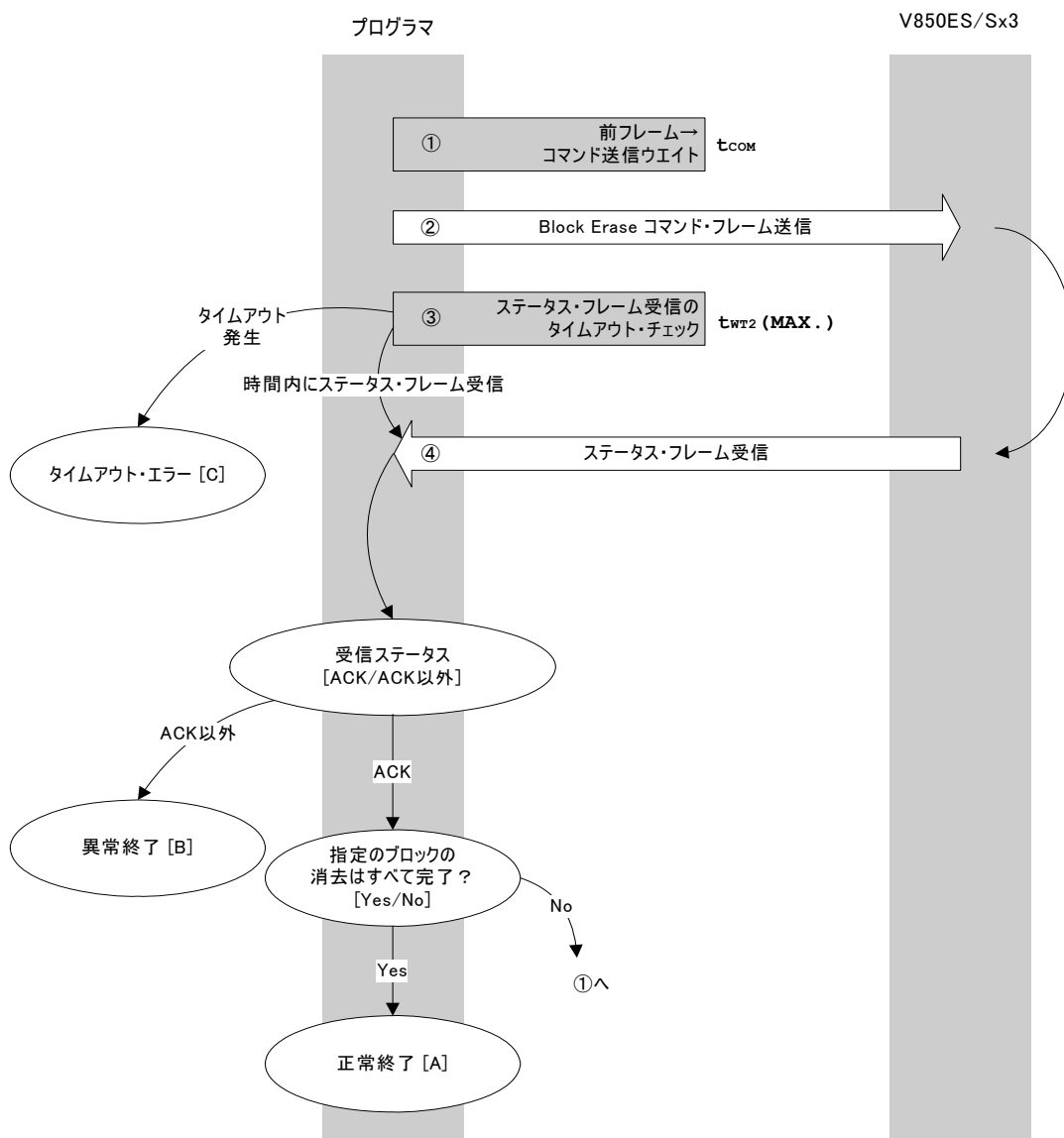
    put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send ERASE CHIP command

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

6.8 Block Eraseコマンド

6.8.1 処理手順チャート

Block Eraseコマンド処理手順



6.8.2 処理手順説明

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} (MAX.))。
 ステータス・コードをチェックします。

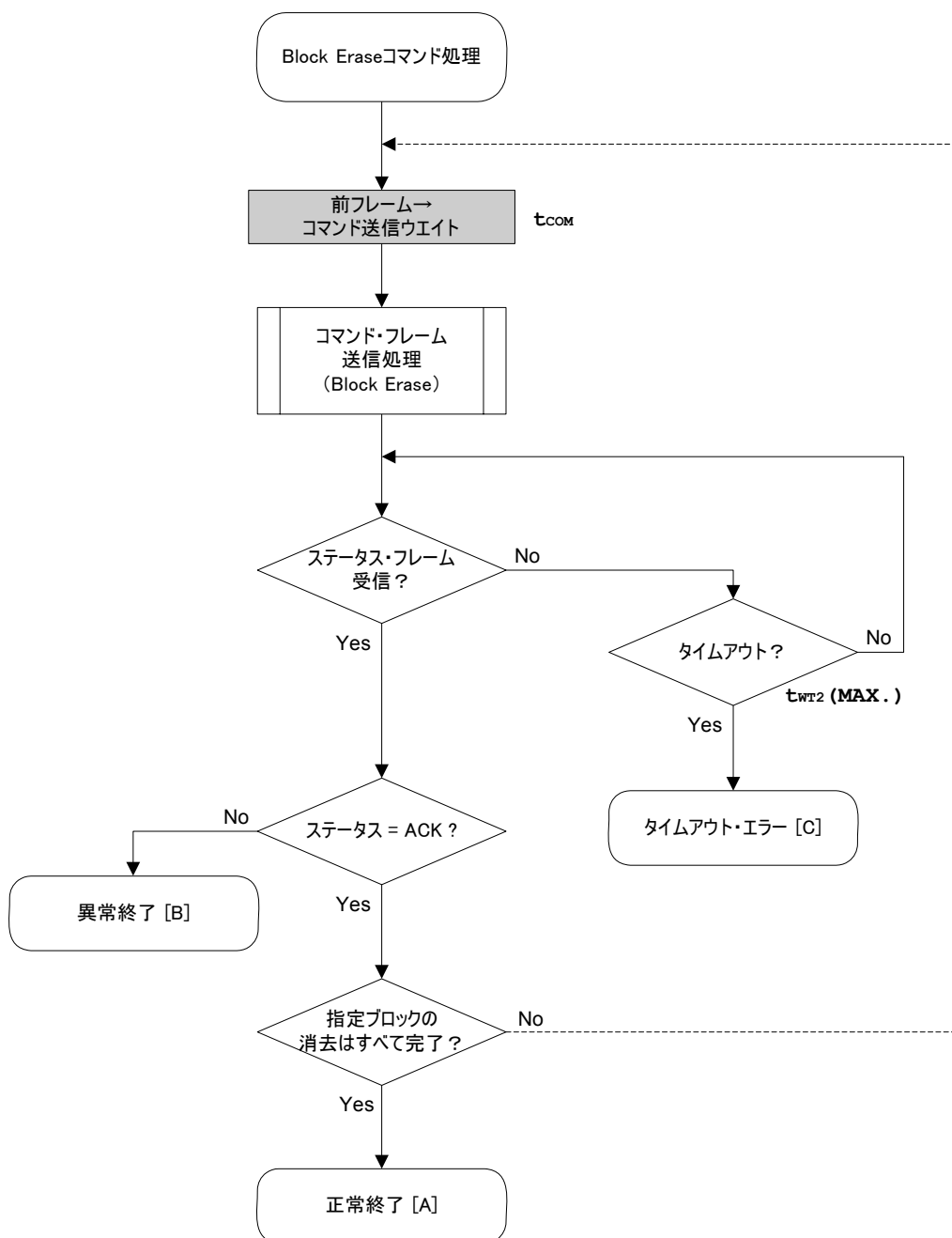
ST1 = ACK の場合 : 指定したブロックの消去がすべて完了していない場合は、ブロック番号を変えて より再実行します。
 指定したすべてのブロックの消去が完了した場合は、**正常終了[A]** です。

ST1 = ACK以外 の場合 : **異常終了[B]** です。

6.8.3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、ブロック消去が正常に実行されたことを示します。
異常終了 [B] パラメータ・エラー	05H	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。
	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。
	10H	セキュリティ設定で、Block Erase コマンドが禁止となる設定になっています。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど) 。
MRG10 エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。

6.8.4 フロー・チャート



6.8.5 サンプル・プログラム

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

```

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

    u16    rc;
    u32    wt2_max;
    u32    top, bottom;

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

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

    wt2_max = make_wt2_max(sblk, eblk); // get tWT2(Max)

    fl_wait(tCOM);                    // wait before sending command

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

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

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

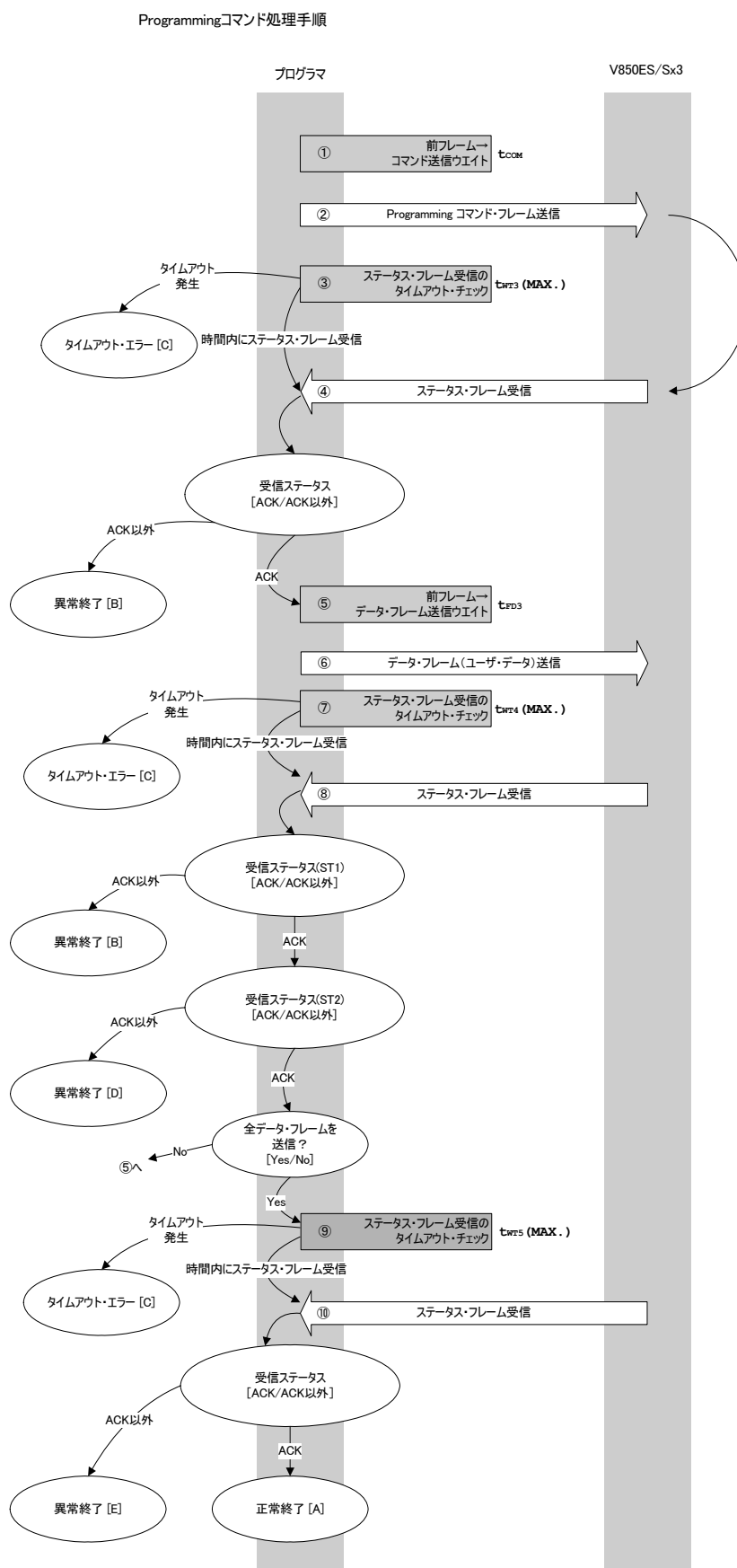
    return rc;

}

```

6.9 Programmingコマンド

6.9.1 処理手順チャート



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、**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} (MAX.))。
 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} (MAX.))。
 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。

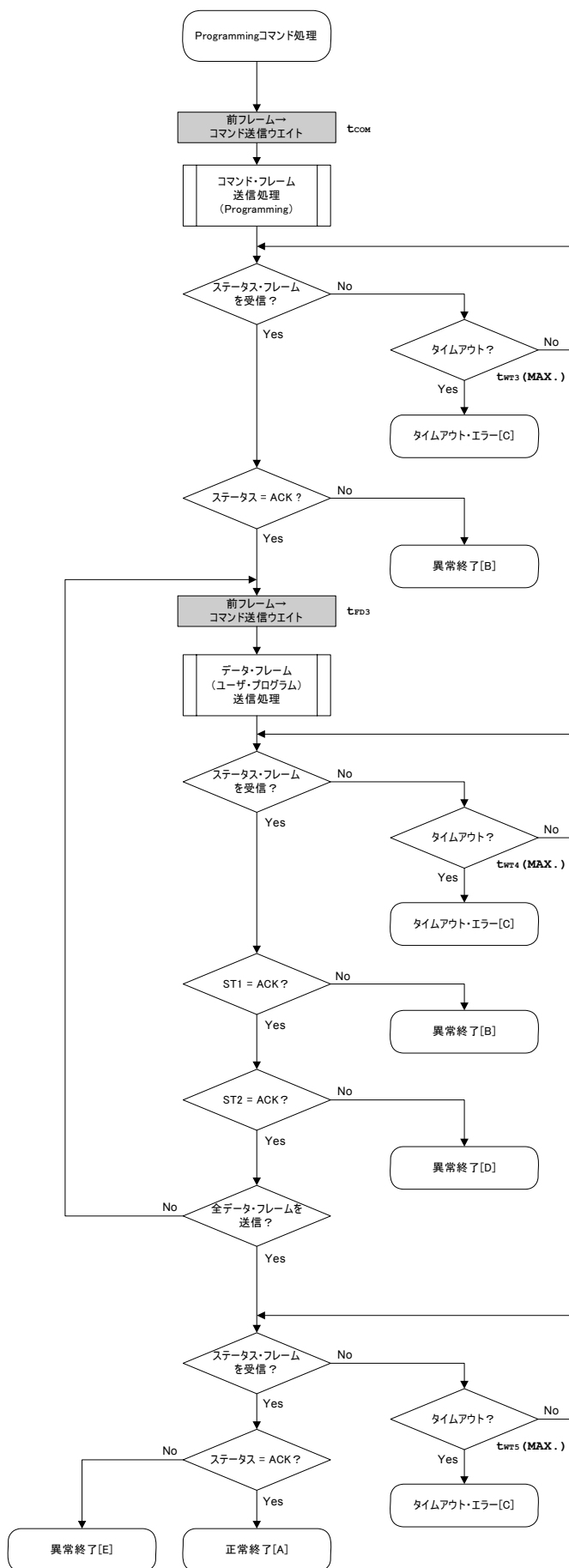
ステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は**タイムアウト・エラー[C]**となります
 (タイムアウト時間 t_{WT5} (MAX.))。
 ステータス・コードをチェックします。

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

6.9.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 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、Programming コマンドが禁止となる設定になっています。
	否定応答 (NACK)	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D]	WRITE エラー	1CH	書き込みエラーが発生しました。
異常終了 [E]	MRG11 エラー	1BH	内部ベリファイ・エラーが発生しました。

6.9.4 フロー・チャート



6.9.5 サンプル・プログラム

Programmingコマンド処理のサンプル・プログラムです。

```

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

#define          fl_st2_ua          (fl_ua_sfrm[OFS_STA_PLD+1])

u16          fl_ua_write(u32 top, u32 bottom)
{
    u16      rc;
    u32      send_head, send_size;
    bool     is_end;
    u32      wt5_max;

    /*****
    /*      set params
    /*****
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    wt5_max = make_wt5_max(get_block_num(top, bottom));

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

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

    rc = get_sfrm_ua(fl_ua_sfrm, tWT3_MAX); // get status frame
    switch(rc) {
        case    FLC_NO_ERR:                break; // continue
        // case    FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

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

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

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

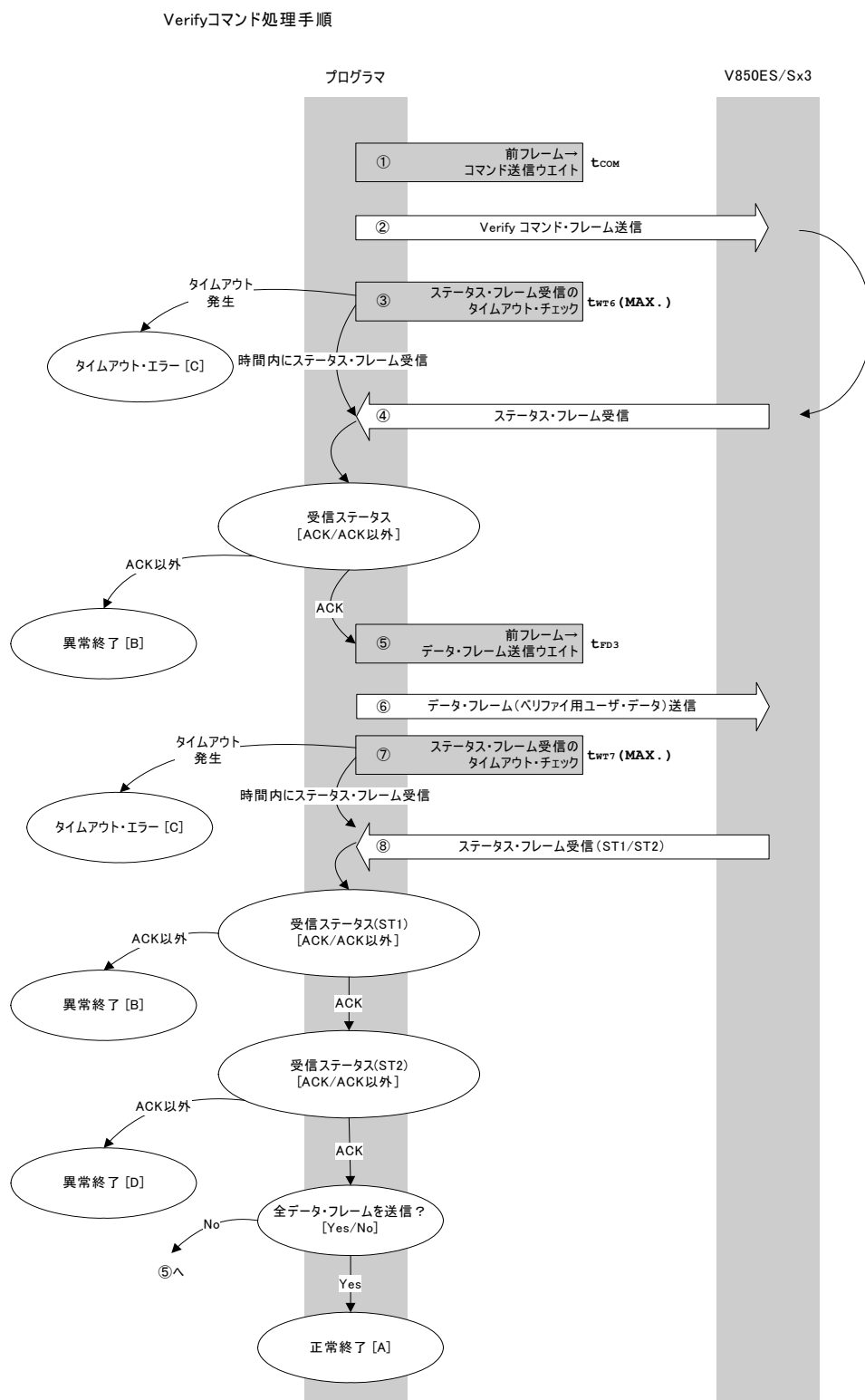
```

```
rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX);          // get status frame
switch(rc) {
    case FLC_NO_ERR:                            break; // continue
    case FLC_DFTO_ERR: return rc;               break; // case [C]
    default: return rc;                         break; // case [B]
}
if (fl_st2_ua != FLST_ACK){                    // ST2 = ACK ?
    rc = decode_status(fl_st2_ua);             // No
    return rc;                                  // case [D]
}
if (is_end)
    break;

}
/*****
/*      Check internally verify                */
*****/
rc = get_sfrm_ua(fl_ua_sfrm, wt5_max); // get status frame again
// switch(rc) {
//     case FLC_NO_ERR: return rc; break; // case [A]
//     case FLC_DFTO_ERR: return rc; break; // case [C]
//     default: return rc; break; // case [E]
// }
return rc;
}
```

6.10 Verifyコマンド

6.10.1 処理手順チャート



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、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} (MAX.))。
 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。

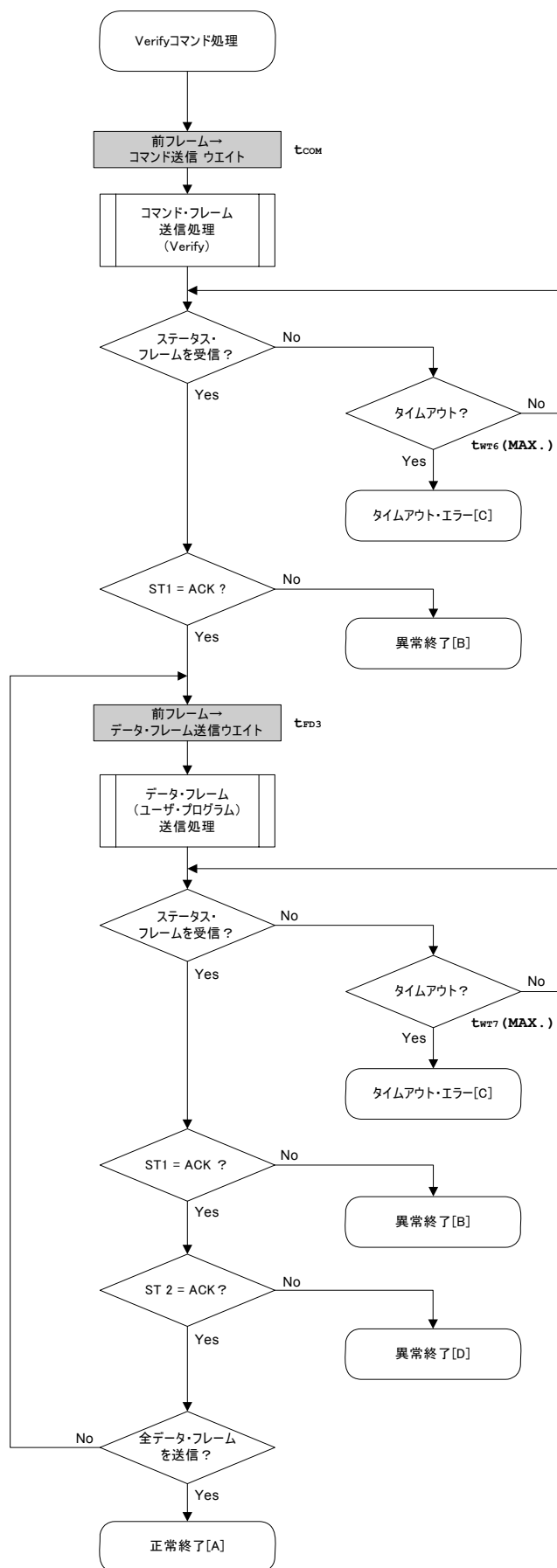
直前のフレームからデータ・フレーム送信までのウエイトをします (ウエイト時間 t_{FD3})。
 データ・フレーム送信処理により、ペリファイ用ユーザ・データを送信します。
 ユーザ・データ送信からステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合はタイムアウト・エラー[C]となります
 (タイムアウト時間 t_{WT7} (MAX.))。
 ステータス・コード (ST1/ST2) をチェックします (処理手順チャートやフロー・チャートも参照してください)。

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. 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。
	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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D] 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。

6. 10. 4 フロー・チャート



6. 10. 5 サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

```

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

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

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

    fl_wait(tCOM);          // wait before sending command

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

    rc = get_sfrm_ua(fl_ua_sfrm, tWT6_MAX);          // get status frame
    switch(rc) {
        case    FLC_NO_ERR:                break; // continue
        // case    FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

    while(1){

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

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

        fl_wait(tFD3);

```

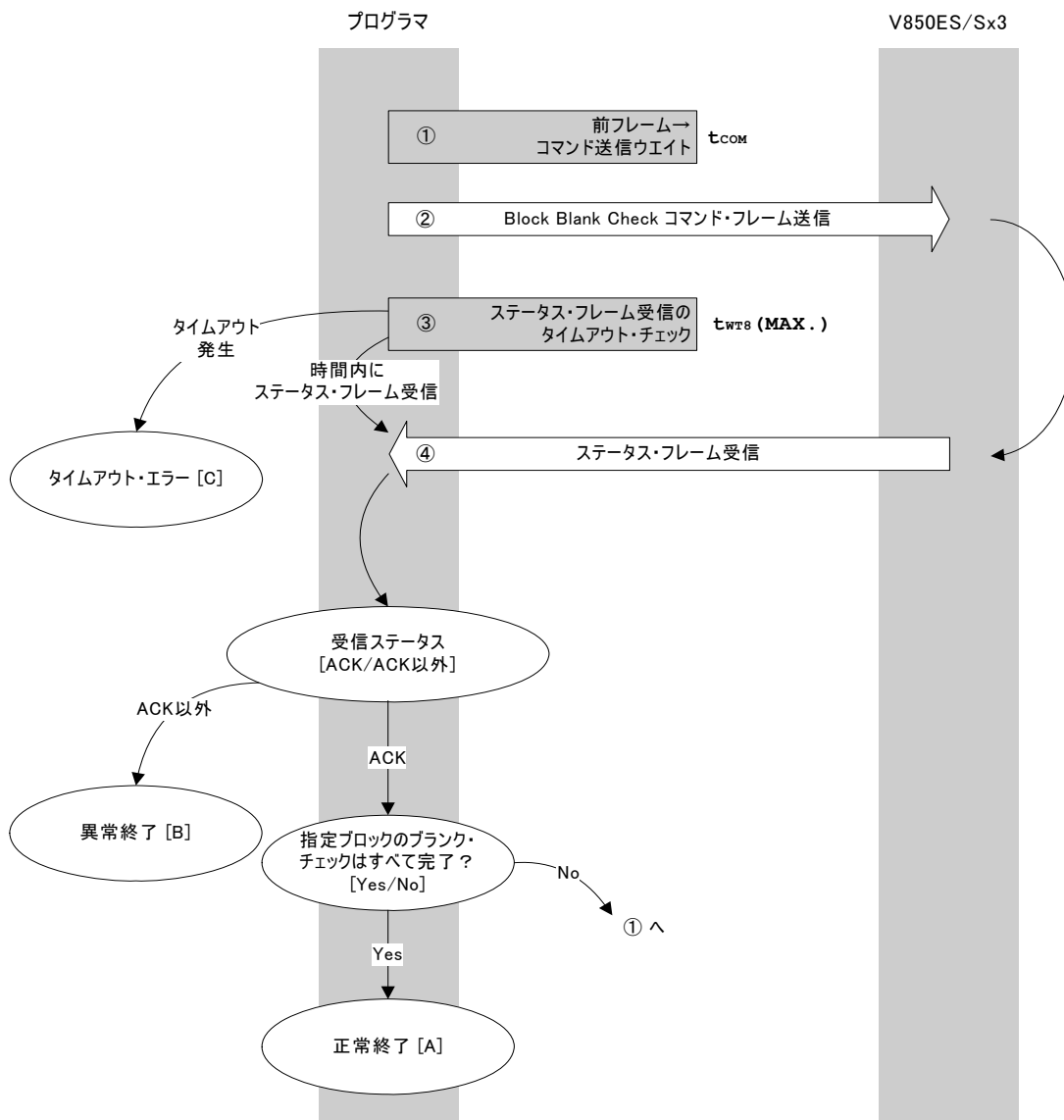
```
put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX); // get status frame
switch(rc) {
    case FLC_NO_ERR:                break; // continue
    // case FLC_DFTO_ERR: return rc; break; // case [C]
    default:                        return rc; break; // case [B]
}
if (fl_st2_ua != FLST_ACK){        // ST2 = ACK ?
    rc = decode_status(fl_st2_ua); // No
    return rc;                    // case [D]
}
if (is_end)                        // send all user data ?
    break;                        // yes
//continue;
}
return FLC_NO_ERR;                // case [A]
}
```


6. 11 Block Blank Checkコマンド

6. 11. 1 処理手順チャート

Block Blank Checkコマンド処理手順



6. 11. 2 処理手順説明

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_{WTB} (MAX.))。
 ステータス・コードをチェックします。

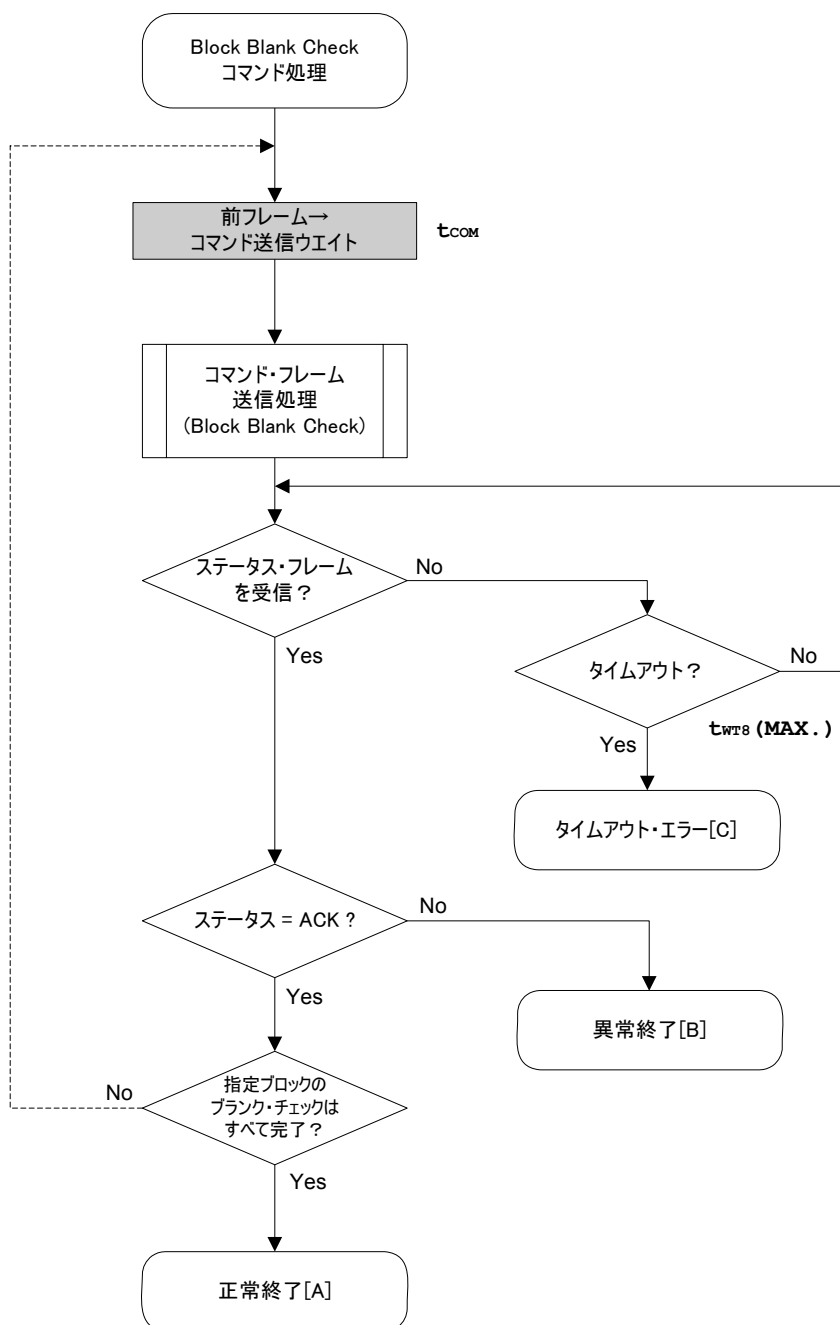
ST1 = ACKの場合 : 指定したブロックのブランク・チェックがすべて完了していない場合は、ブロック番号を変えて より再実行します。
 指定したすべてのブロックのブランク・チェックが完了した場合は、 **正常終了[A]** です。

ST1 = ACK以外の場合 : **異常終了[B]** です。

6. 11. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、指定したブロックすべてがブランクであることを示します。
異常終了 [B] パラメータ・エラー	05H	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。
	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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常の場合 (データ長 (LEN) 不正, ETX なしなど)
	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. 11. 4 フロー・チャート



6. 11. 5 サンプル・プログラム

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

```

/*****/
/*
/* Block blank check command
/*
/*****/
/* [i] u16 sblk    ... start block number
/* [i] u16 eblk    ... end block number
/* [r] u16         ... error code
/*****/
u16    fl_ua_blk_blank_chk(u16 sblk, u16 eblk)
{
    u16    rc;
    u32    wt8_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

    wt8_max = make_wt8_max(sblk, eblk); // get tWT8(Max)

    fl_wait(tCOM); // wait before sending command

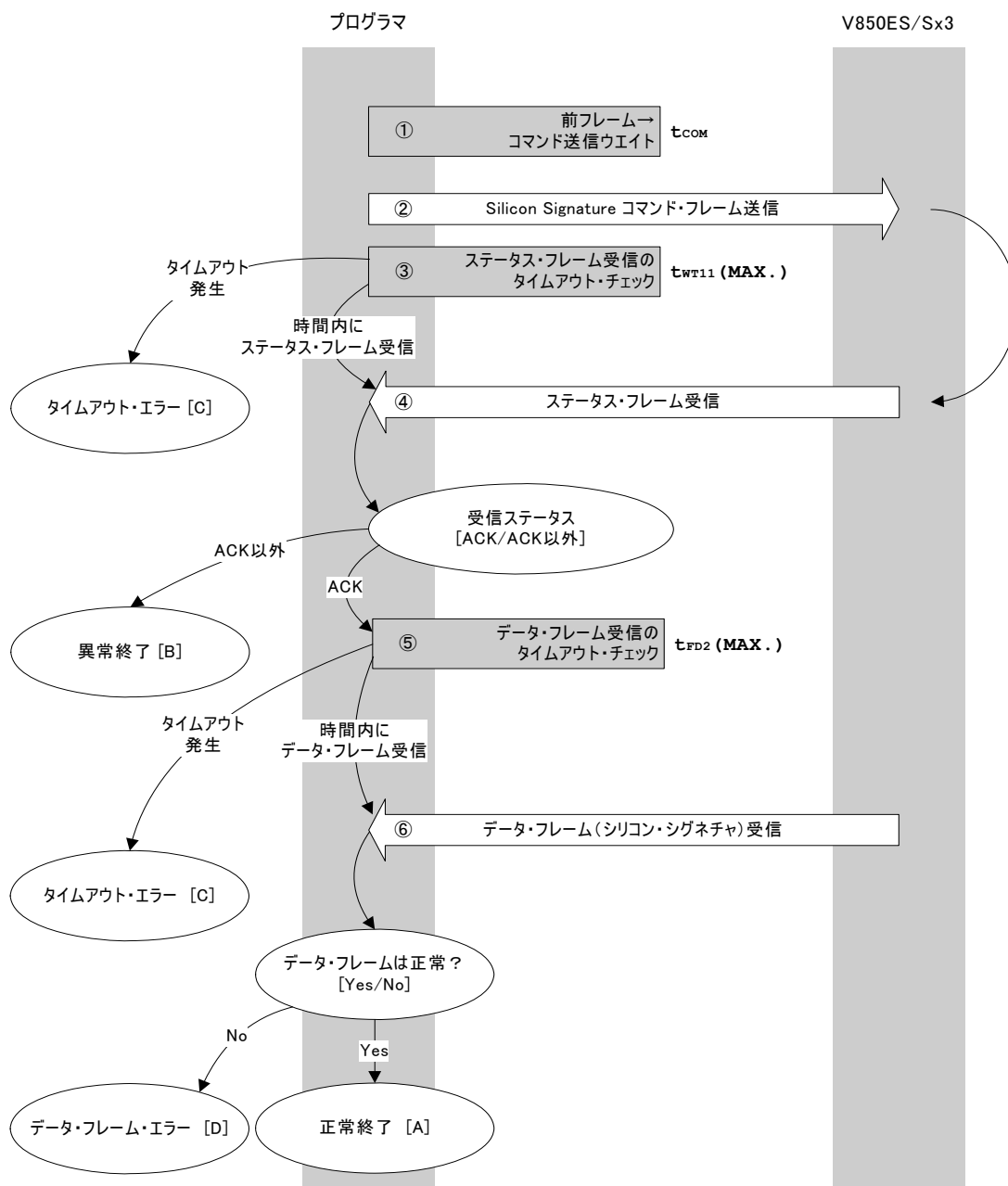
    put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm);
    rc = get_sfrm_ua(fl_ua_sfrm, wt8_max); // get status frame
    // switch(rc) {
    //
    //     case    FLC_NO_ERR:    return rc;    break; // case [A]
    //     case    FLC_DFTO_ERR:  return rc;    break; // case [C]
    //     default:                return rc;    break; // case [B]
    // }
    return rc;
}

```

6. 12 Silicon Signatureコマンド

6. 12. 1 処理手順チャート

Silicon Signatureコマンド処理手順



6. 12. 2 処理手順説明

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・フレーム受信までのタイムアウト・チェックを行います。タイムアウトが発生した場合はタイムアウト・エラー[C]です
 (タイムアウト時間 t_{WT11} (MAX.))。
 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。

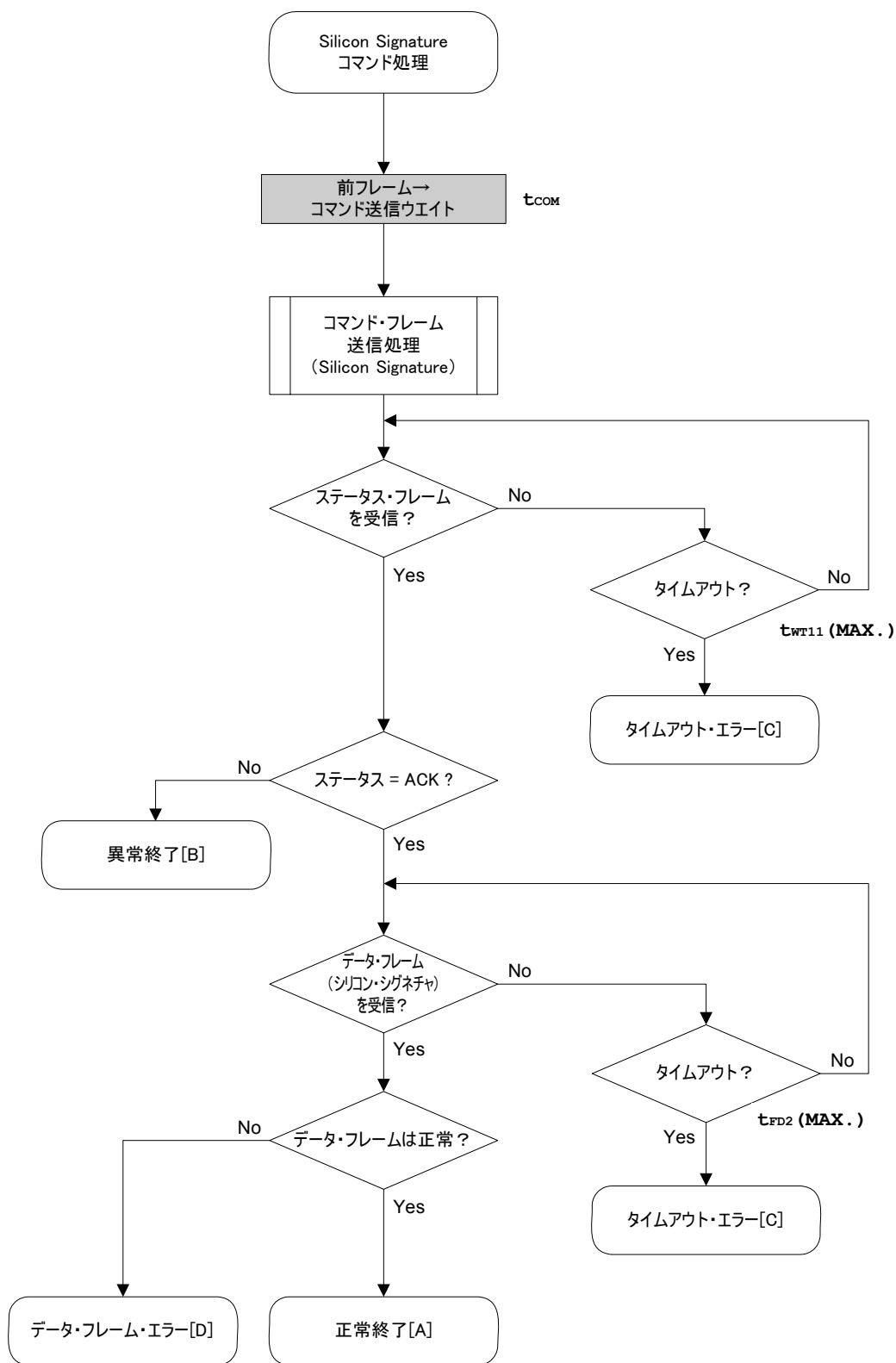
データ・フレーム (シリコン・シグネチャ・データ) 受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合はタイムアウト・エラー[C]です
 (タイムアウト時間 t_{FD2} (MAX.))。
 受信したデータ・フレーム (シリコン・シグネチャ・データ) をチェックします。

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. 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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	ステータス・フレーム, またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]	-	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

6. 12. 4 フロー・チャート



6. 12. 5 サンプル・プログラム

Silicon Signatureコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM);          // wait before sending command

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

    rc = get_sfrm_ua(fl_ua_sfrm, tWT11_MAX);          // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

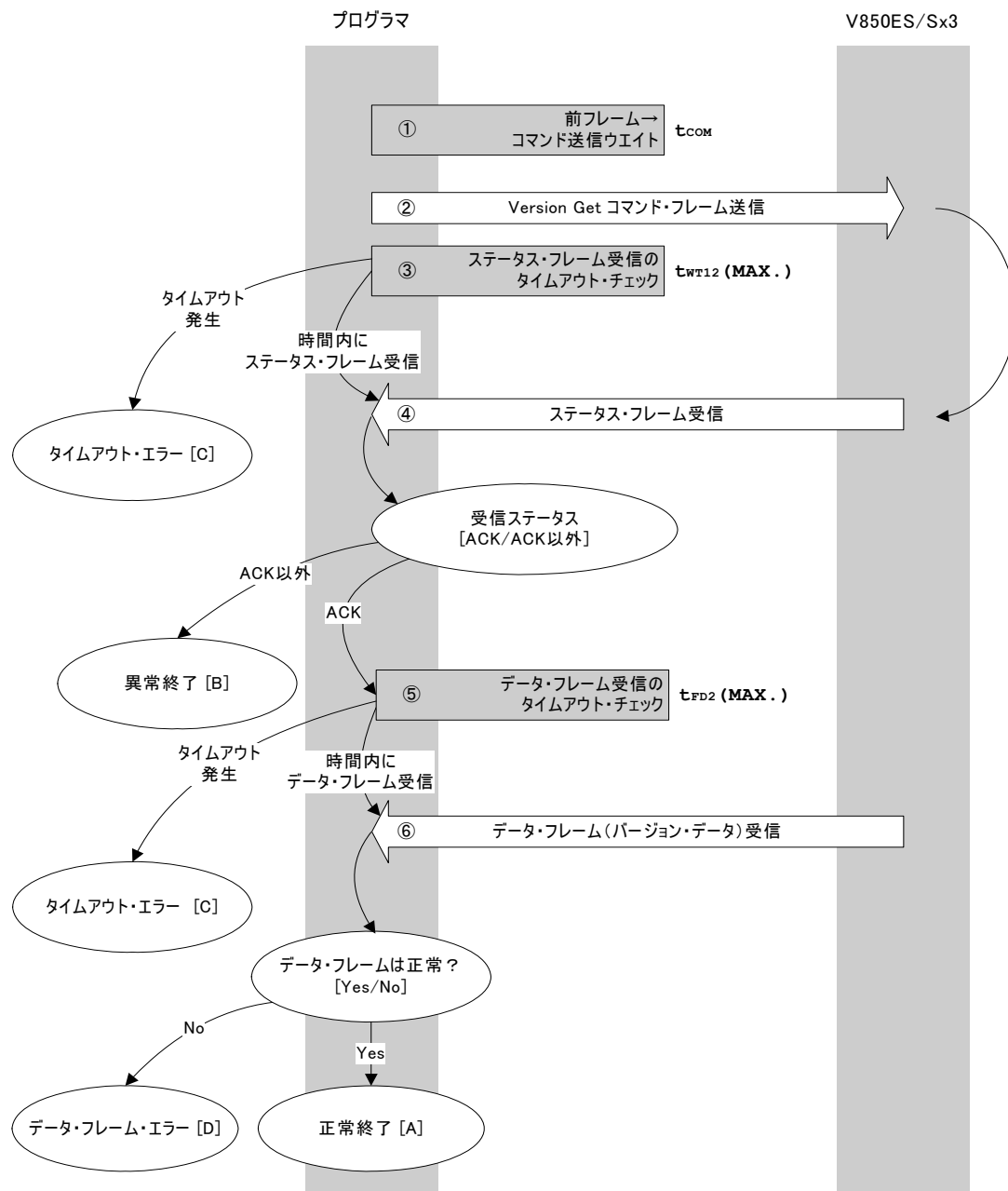
    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX);        // get status frame
    if (rc){
        return rc;                // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                // copy Signature data
    return rc;                // case [A]
}

```


6. 13 Version Getコマンド

6. 13. 1 処理手順チャート

Version Getコマンド処理手順



6. 13. 2 処理手順説明

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} (MAX.))。
 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。

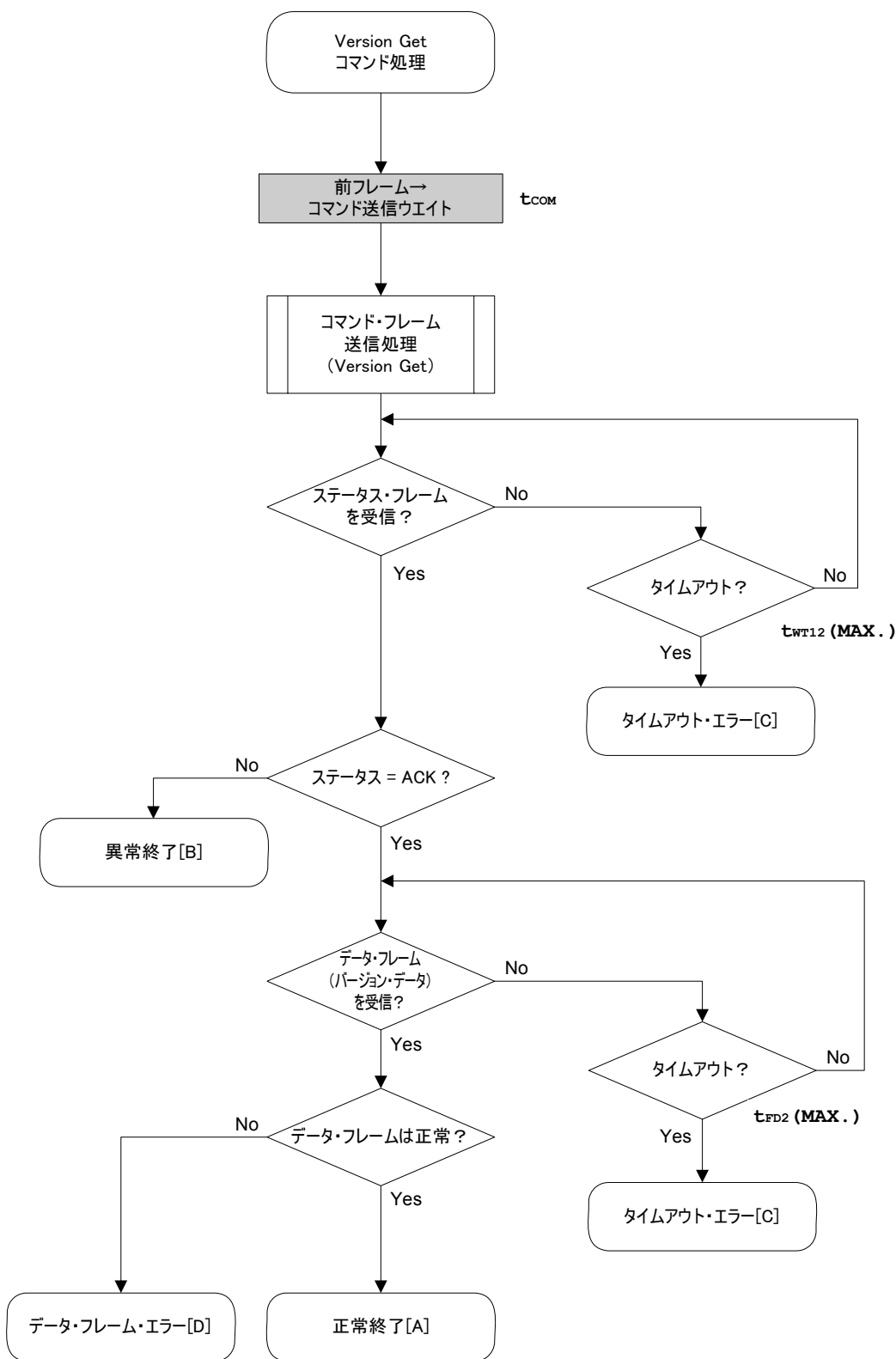
データ・フレーム (バージョン・データ) 受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は**タイムアウト・エラー[C]**です
 (タイムアウト時間 t_{FD2} (MAX.))。
 受信したデータ・フレーム (バージョン・データ) をチェックします。

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. 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	ステータス・フレーム、またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]		-	バージョン・データとして受信したデータ・フレームのチェックサムが異常です。

6. 13. 4 フロー・チャート



6. 13. 5 サンプル・プログラム

Version Getコマンド処理のサンプル・プログラムです。

```

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

    fl_wait(tCOM);          // wait before sending command

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

    rc = get_sfrm_ua(fl_ua_sfrm, tWT12_MAX);        // get status frame
    switch(rc) {
        case FLC_NO_ERR:                          break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

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

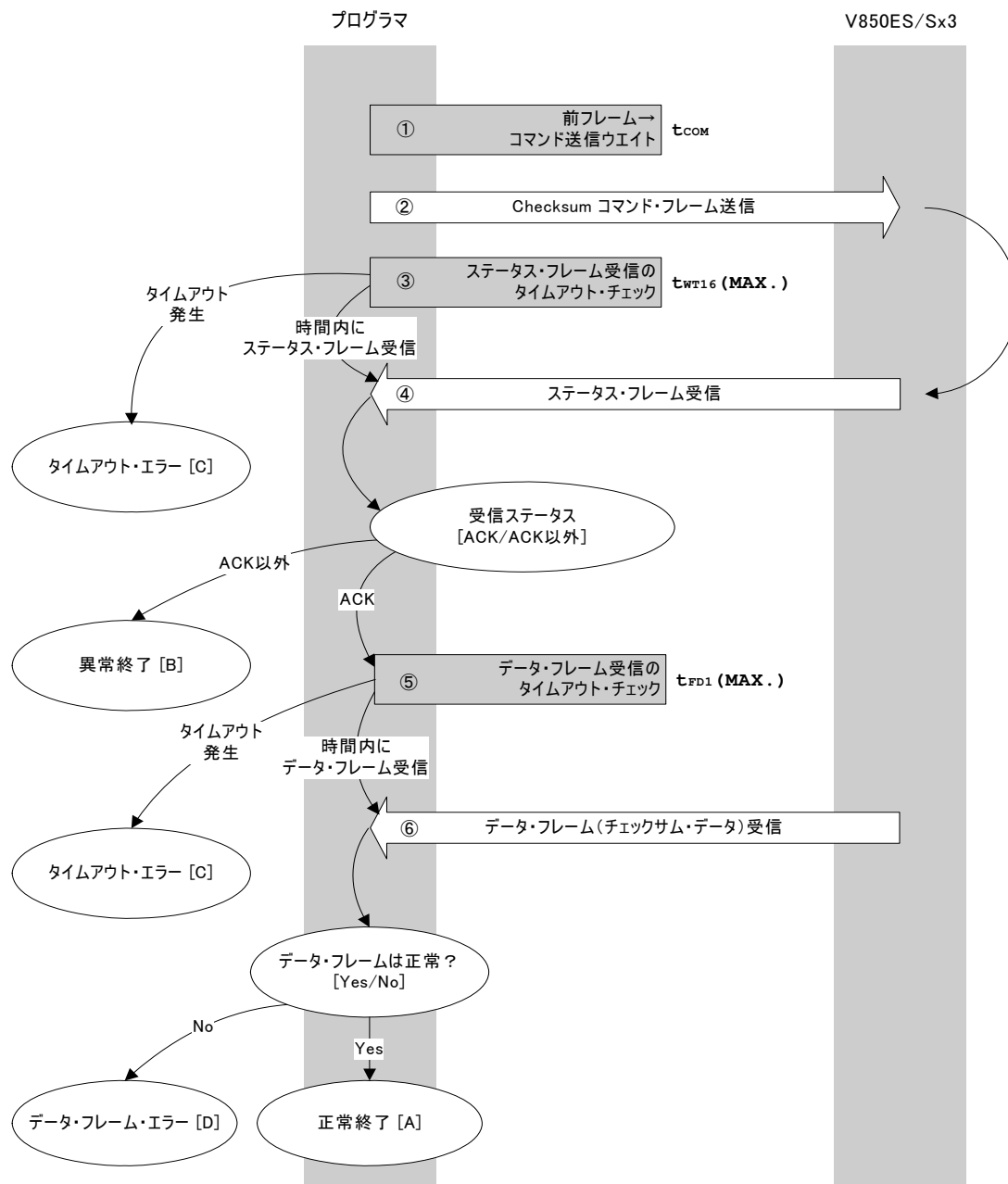
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                                    // case [A]
}

```

6. 14 Checksumコマンド

6. 14. 1 処理手順チャート

Checksumコマンド処理手順



6. 14. 2 処理手順説明

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} (MAX.))。
 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。

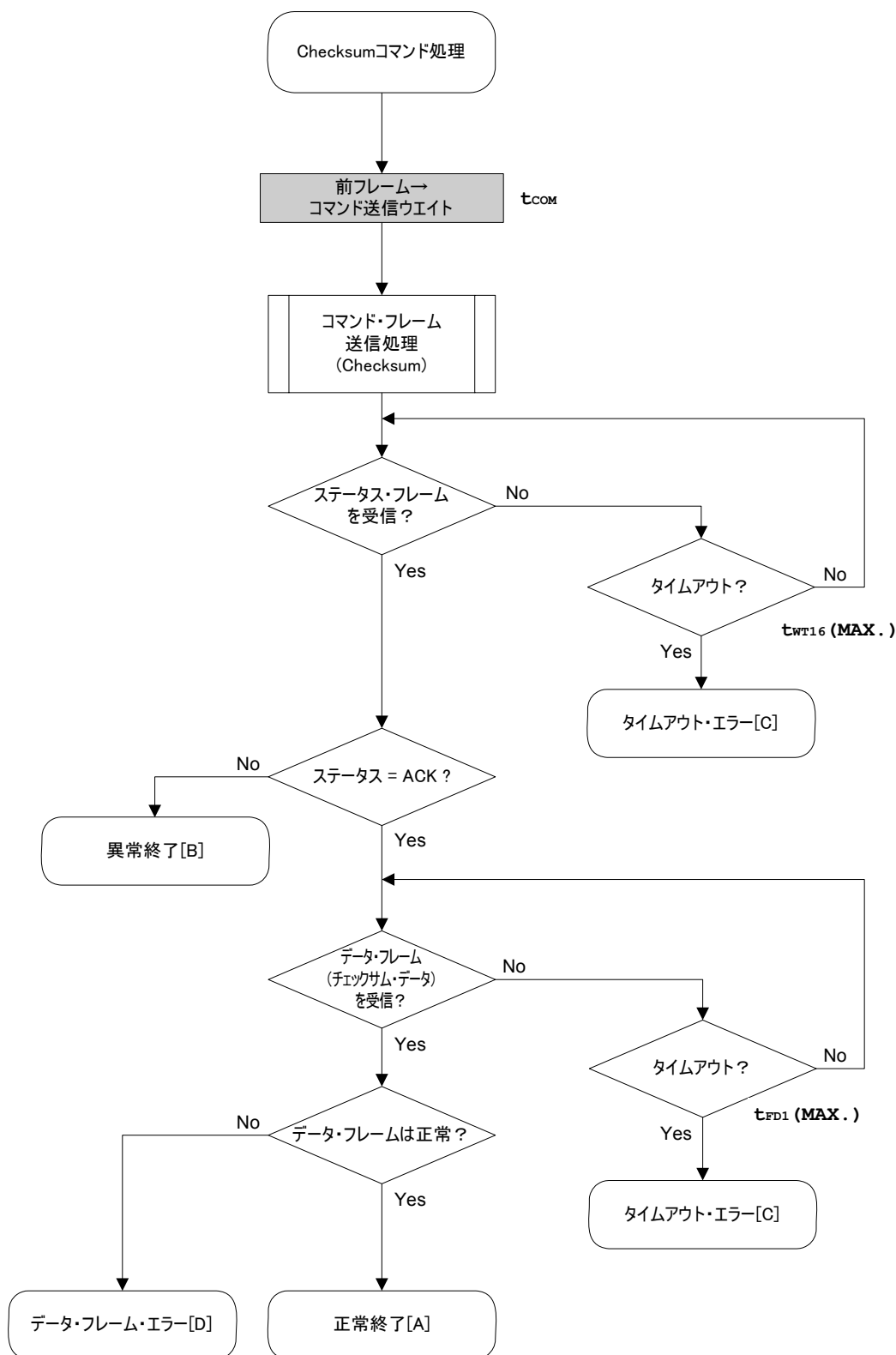
データ・フレーム(チェックサム・データ)受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は **タイムアウト・エラー[C]** です
 (タイムアウト時間 t_{FD1} (MAX.))。
 受信したデータ・フレーム (チェックサム・データ) をチェックします。

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. 14. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され, チェックサム・データ を取得できたことを示します。
異常終了 [B] パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アド レス以外で指定されています。
	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。
	15H	・処理中にステータス・コマンド以外を受信しまし た。 ・コマンド・フレーム・データが異常です (データ 長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	ステータス・フレーム, またはデータ・フレームの 受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]	-	チェックサム・データとして受信したデータ・フレ ームのチェックサムが異常です。

6. 14. 4 フロー・チャート



6. 14. 5 サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

```

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

    /*****
    /*      set params
    /*****
    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    fdl_max = get_fdl_max(get_block_num(top, bottom)); // get tFD1 (MAX)

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

    fl_wait(tCOM); // wait before sending command

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

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

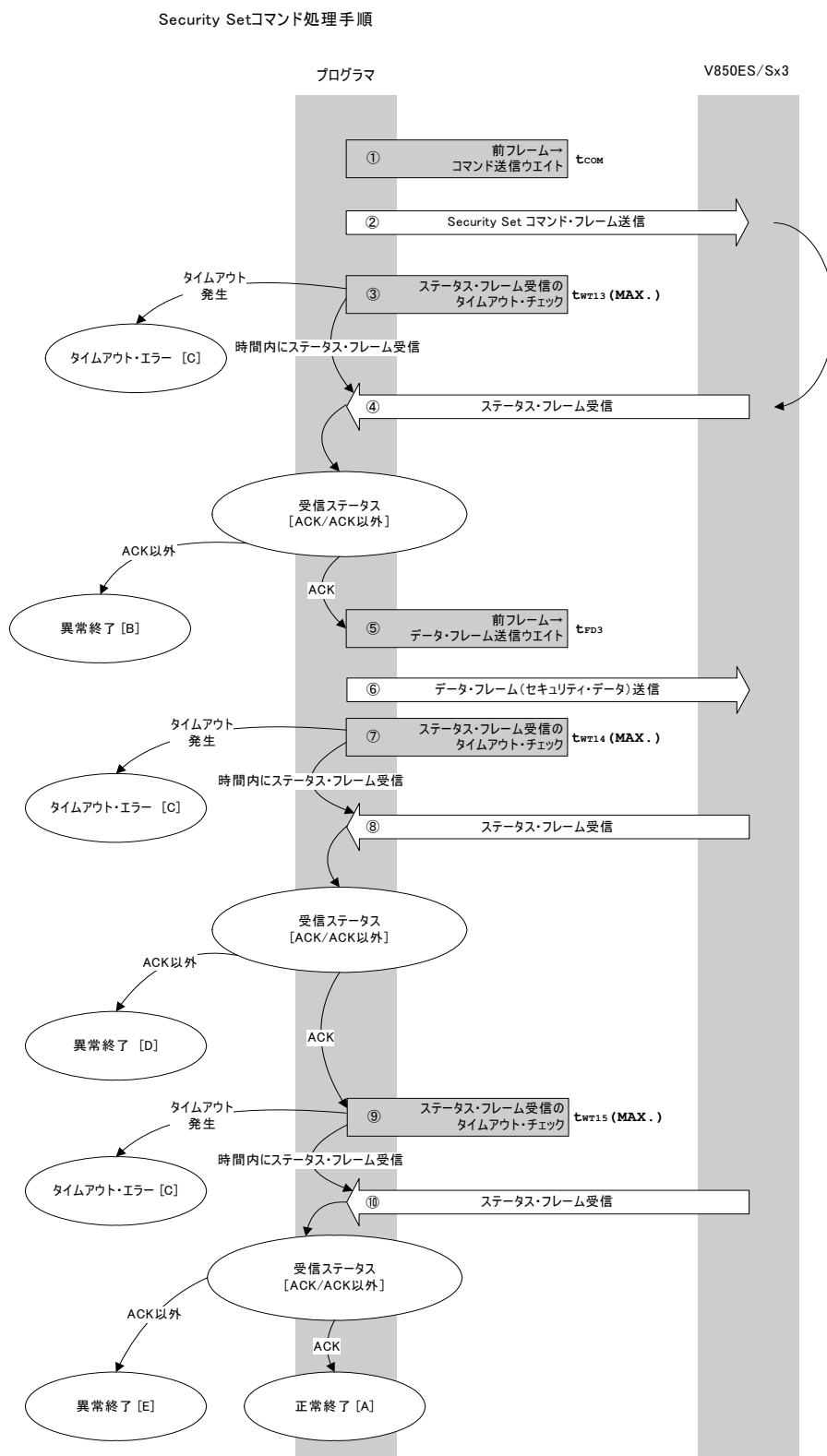
    /*****
    /*      get data frame (Checksum data)
    /*****
    rc = get_dfrm_ua(fl_rxdata_frm, fdl_max); // get status frame
    if (rc){
        return rc; // case [D]
    }

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

```


6. 15 Security Setコマンド

6. 15. 1 処理手順チャート



6. 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, **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} (MAX.))。
 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エイトをします (ウエイト時間 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} (MAX.))。
 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。

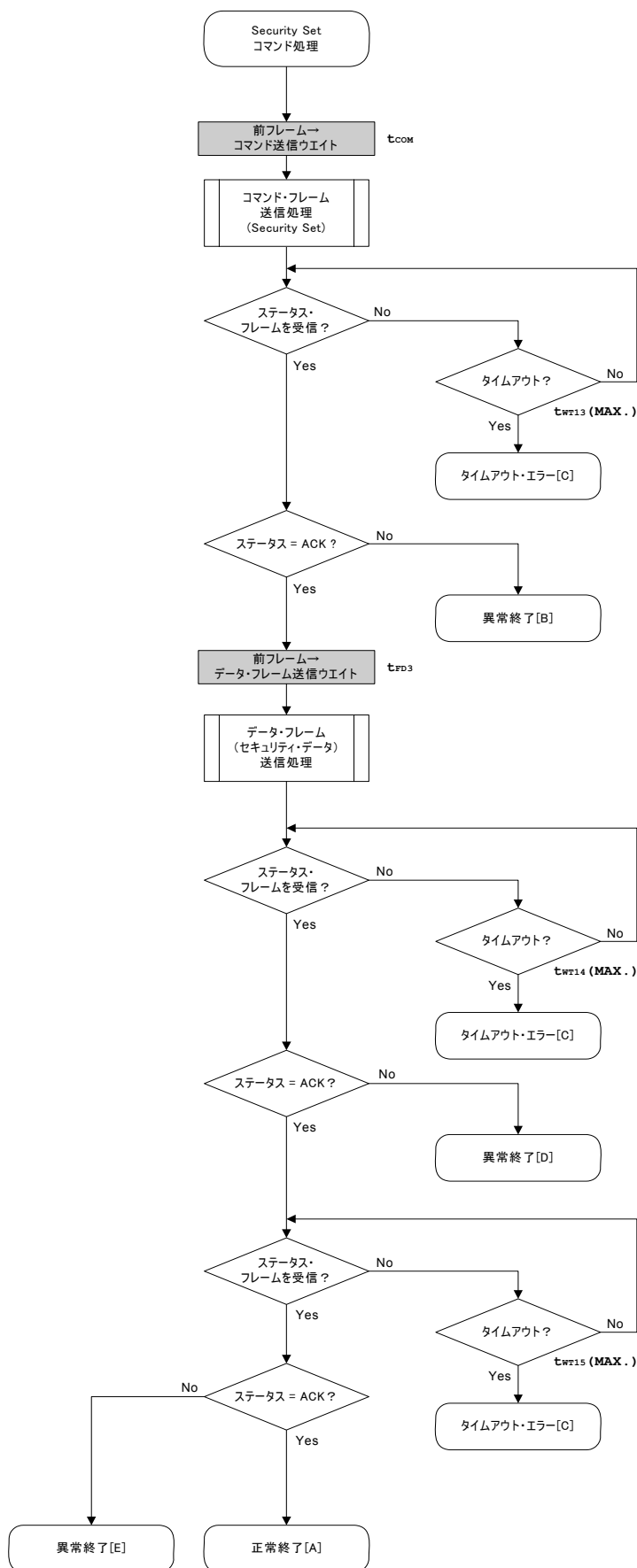
ステータス・フレーム受信までのタイムアウト・チェックを行います。
 タイムアウトが発生した場合は **タイムアウト・エラー[C]** となります
 (タイムアウト時間 t_{WT15} (MAX.))。
 ステータス・コードをチェックします。

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

6. 15. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、セキュリティ設定データが正しく設定されたことを示します。
異常終了 [B]	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	<ul style="list-style-type: none"> ・ 処理中にステータス・コマンド以外を受信しました。 ・ コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D]	否定応答 (NACK)	15H	セキュリティ・データ・フレームが異常です。
	チェックサム・エラー	07H	送信したセキュリティ・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・データが次の場合のエラーです。 <ul style="list-style-type: none"> ・ セキュリティを禁止から許可に変更 ・ ブート・ブロック・クラスタ書き換え禁止設定状態でブート・ブロック・クラスタの最終ブロック番号の値を変更
	パラメータ・エラー	05H	セキュリティ・データが次の場合のエラーです。 <ul style="list-style-type: none"> ・ ブート・ブロック・クラスタの最終ブロック番号がデバイスの最終ブロック番号より大きい ・ リセット・ベクタ・ハンドラ・アドレスの値が "00000000H" 以外
異常終了 [E]	MRG10 エラー	1AH	書き込みエラーが発生しました。
	MRG11 エラー	1BH	
	WRITE エラー	1CH	

6. 15. 4 フロー・チャート



6. 15. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```

/*****/
/*
/* Set security flag command
/*
/*****/
/* [i] u8 scf      ... Security flag data
/* [r] u16        ... error code
/*****/
u16      fl_ua_setscf(u8 scf, u8 bot, u32 vect)
{
    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|= 0b11100000; // "FLG" (bit 7,6,5 must be '1')
fl_txdata_frm[1] = bot;          // "BOT"

fl_txdata_frm[2] = (u8)(vect >> 16); // "ADH"
fl_txdata_frm[3] = (u8)(vect >> 8); // "ADM"
fl_txdata_frm[4] = (u8) vect;      // "ADL"

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

put_cmd_ua(FL_COM_SET_SECURITY, 3, fl_cmd_prm);

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

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

fl_wait(tFD3);
put_dfrm_ua(5, fl_txdata_frm, true); // send securithi setting data

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

/*****/

```

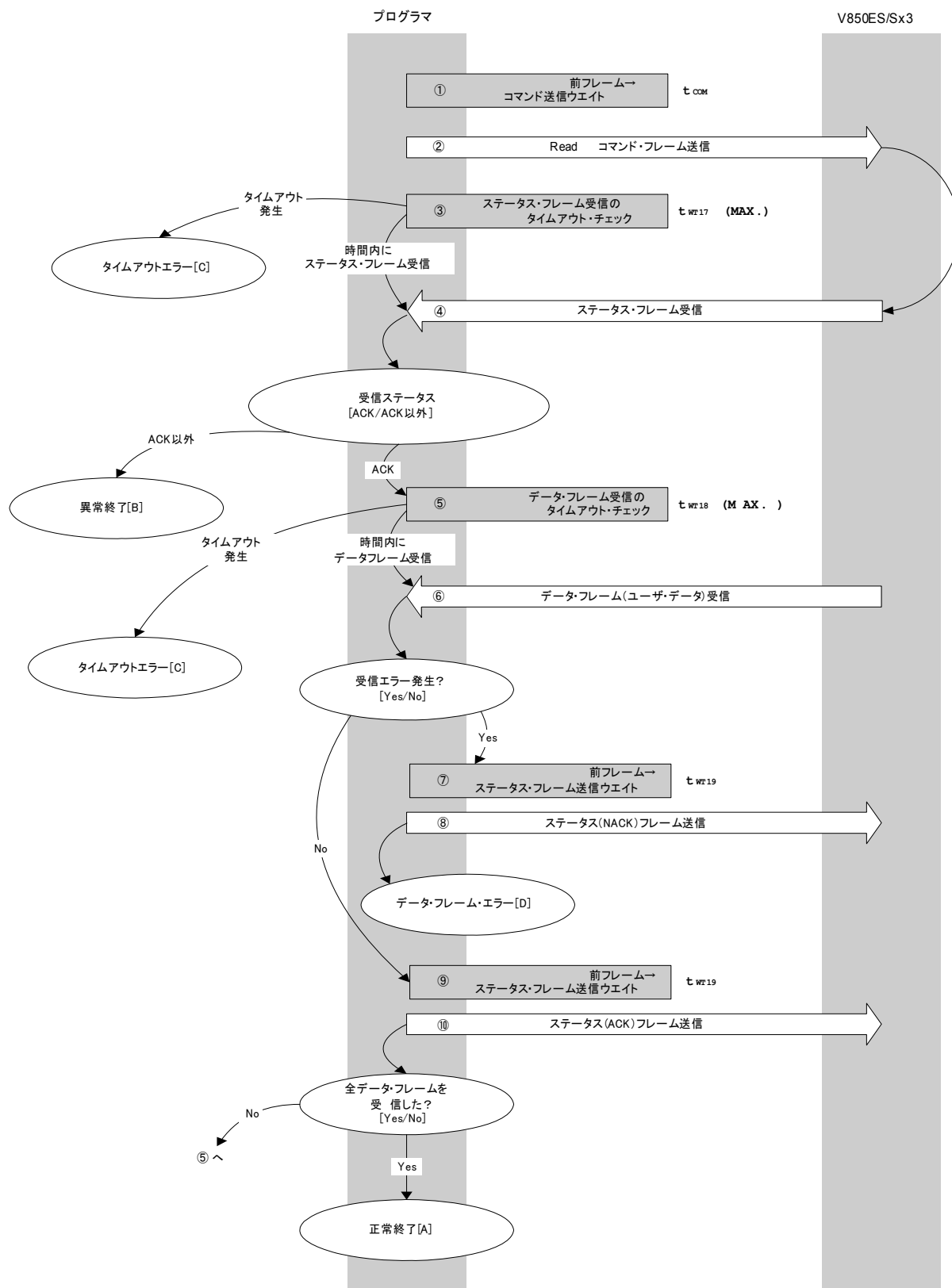
```
/*      Check internally verify      */
/*****
rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);      // get status frame
// switch(rc) {
//
//      case   FLC_NO_ERR:      return rc;      break; // case [A]
//      case   FLC_DFTO_ERR:    return rc;      break; // case [C]
//      default:                return rc;      break; // case [B]
// }

return rc;
}
```

6. 16 Readコマンド

6. 16. 1 処理手順チャート

Readコマンド処理手順



6. 16. 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、**Read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_{WT17} (MAX.))。
 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・フレーム受信結果 (ユーザ・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_{WT18} (MAX.))。
 受信したデータ・フレーム (ユーザ・データ) をチェックします。

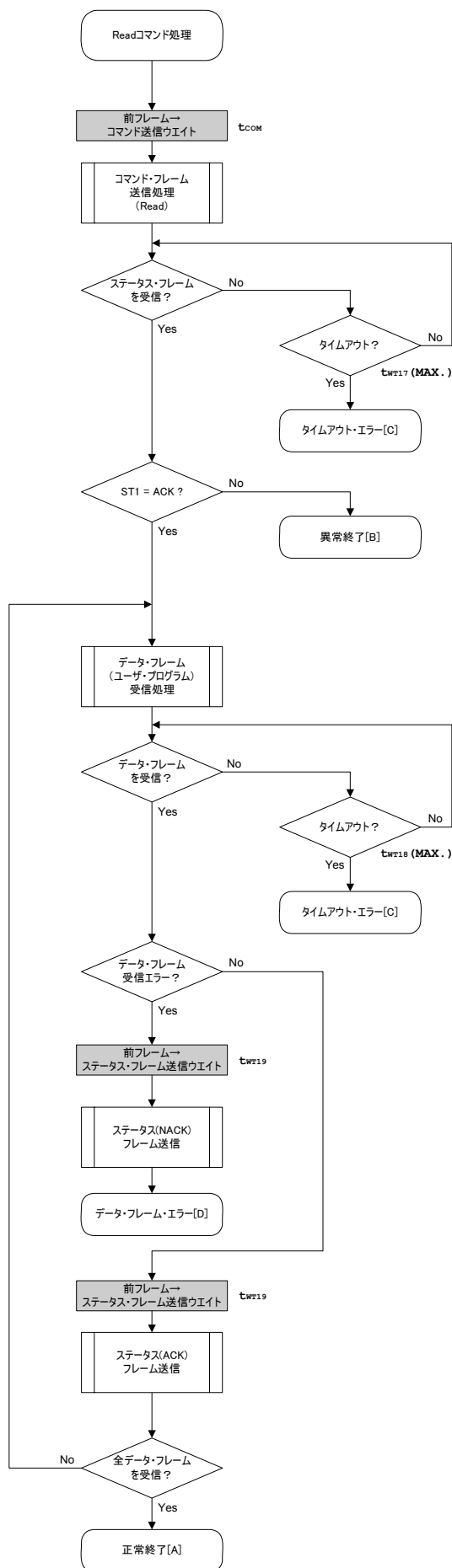
データ・フレームが正常の場合 : に進みます。
 データ・フレームが異常の場合 : に進みます。

直前のフレームからステータス (NACK) フレーム送信までのウェイトをします
 (ウェイト時間 t_{WT19})。
 データ・フレーム送信処理により、NACKフレームを送信します。
データ・フレーム・エラー[D] となります。
 直前のフレームからステータス (ACK) フレーム送信までのウェイトをします
 (ウェイト時間 t_{WT19})。
 データ・フレーム送信処理により、ACKフレームを送信します。
 全データ・フレームの受信が完了した場合は、**正常終了[A]** です。
 まだ受信すべきデータ・フレームが残っている場合は、より再実行します。

6. 16. 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、読み出し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	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「読み出し禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	ステータス・フレーム、またはデータ・フレームの受信でタイムアウトが発生しました。
データ・フレーム・エラー [D]		-	読み出し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. 16. 4 フロー・チャート



6. 16. 5 サンプル・プログラム

Readコマンド処理のサンプル・プログラムです。

```

/*****/
/*                                          */
/* Read command                          */
/*                                          */
/*****/
u16      fl_ua_read(u32 top, u32 bottom)
{
    u16    rc;
    u32    read_head;
    u16    len;
    u8     hooter;

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

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

    put_cmd_ua(FL_COM_READ, 7, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT17_MAX); // get status frame
    switch(rc) {
        case    FLC_NO_ERR:                break;
        // case    FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

    /*****/
    /*      receive user data              */
    /*****/
    read_head = top;

    while(1){

        rc = get_dfrm_ua(fl_rxdata_frm, tWT18_MAX); // get ROMdata from FLASH

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

                fl_wait(tWT19);
                put_sfrm_ua(FLST_NACK); // send status(NACK)
        }

        return rc;
        break;
    }

    fl_wait(tWT19);

```

```
put_sfrm_ua(FLST_ACK);

/*****
/*      save ROM data      */
*****/
if ((len = fl_rxddata_frm[OFS_LEN]) == 0)      // get length
    len = 256;

memcpy(read_buf+read_head, fl_rxddata_frm+2, len); // save to external RAM

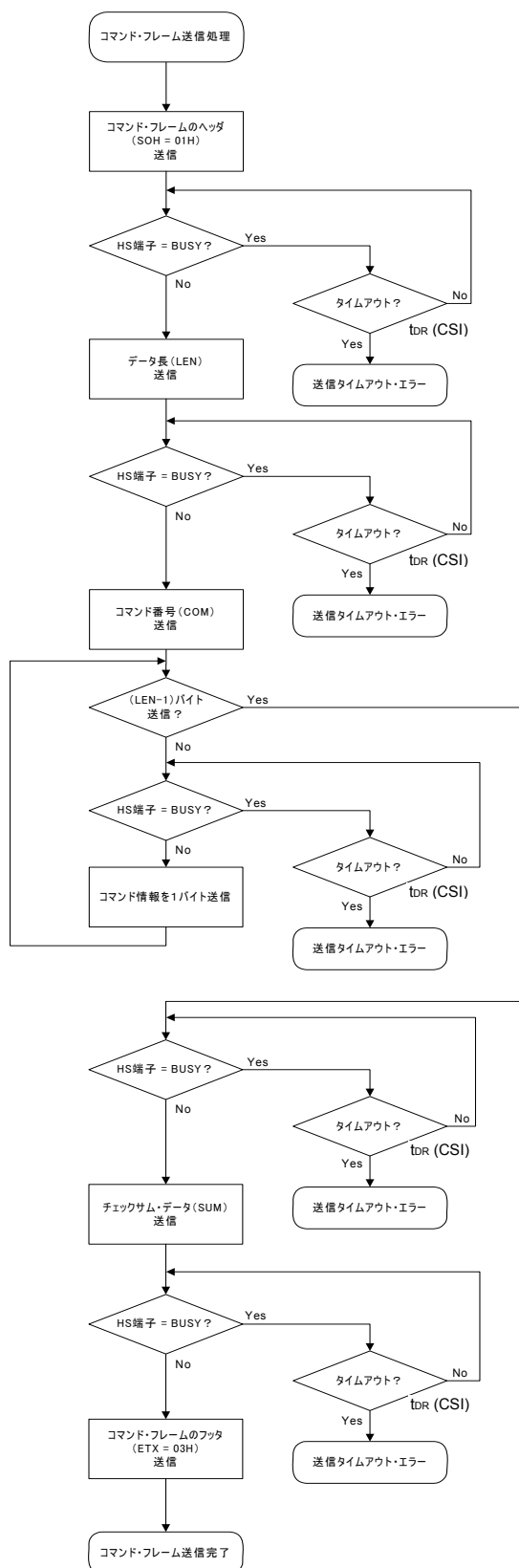
read_head += len;

/*****
/*      end check      */
*****/
hooter = fl_rxddata_frm[len + 3];
if (hooter == FL_ETB)      // end frame ?
    continue;              // no
break;                      // yes
}

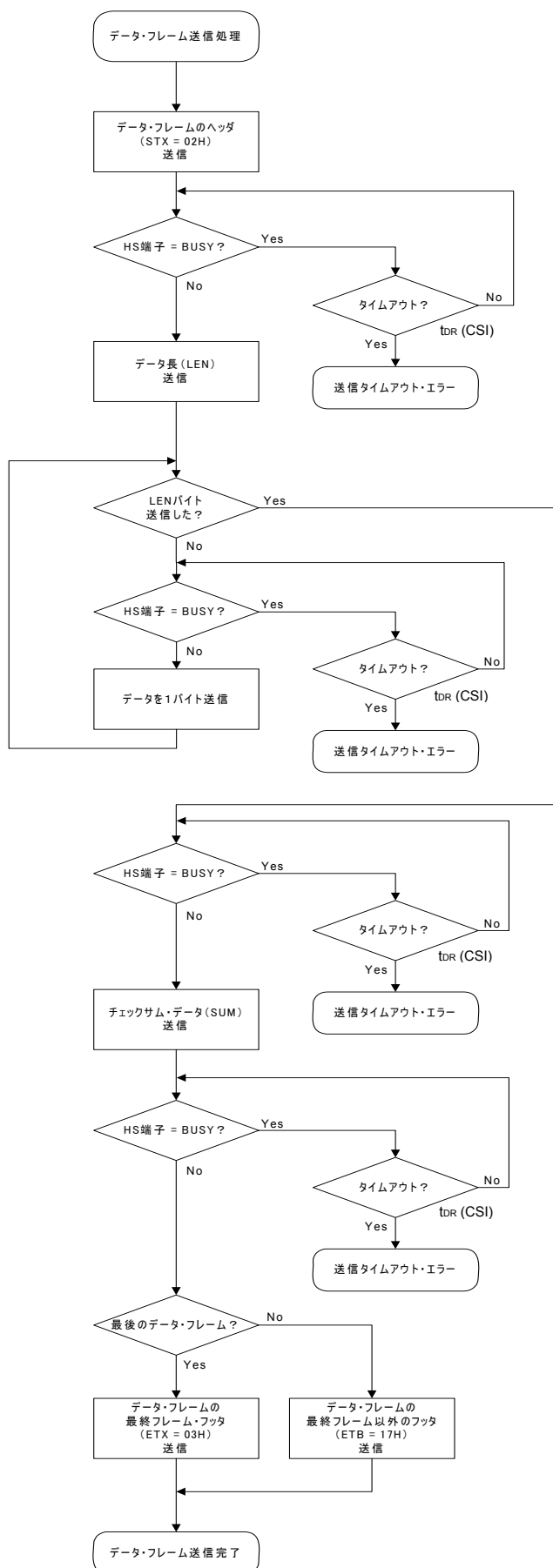
return FLC_NO_ERR;
}
```

第7章 3線式シリアルI/O ハンドシェイク対応(CSI+HS)通信方式

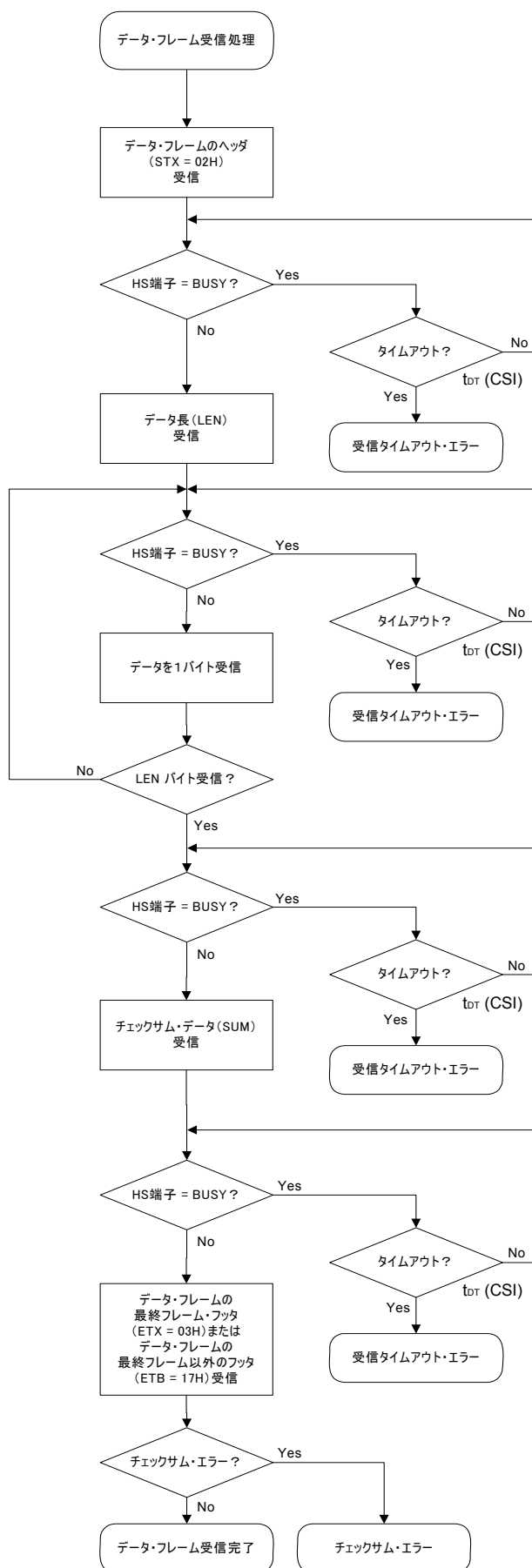
7.1 コマンド・フレーム送信処理のフロー・チャート



7.2 データ・フレーム送信処理のフロー・チャート



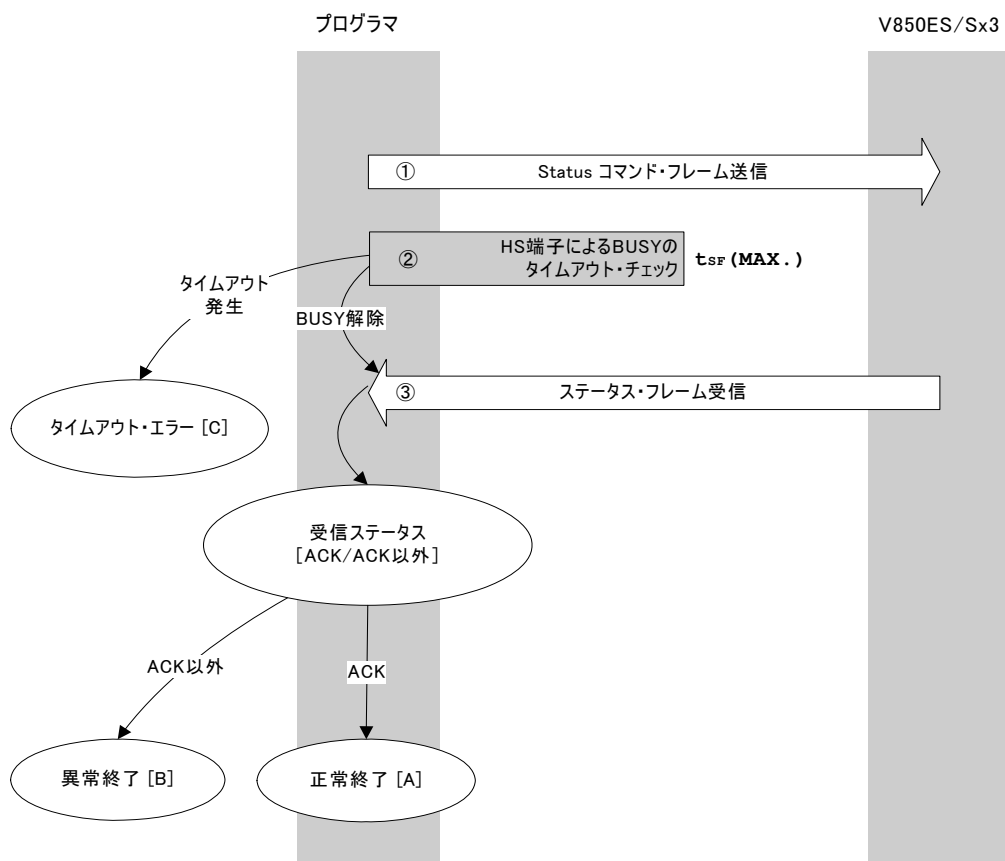
7.3 データ・フレーム受信処理のフロー・チャート



7.4 Statusコマンド

7.4.1 処理手順チャート

Statusコマンド処理手順



7.4.2 処理手順説明

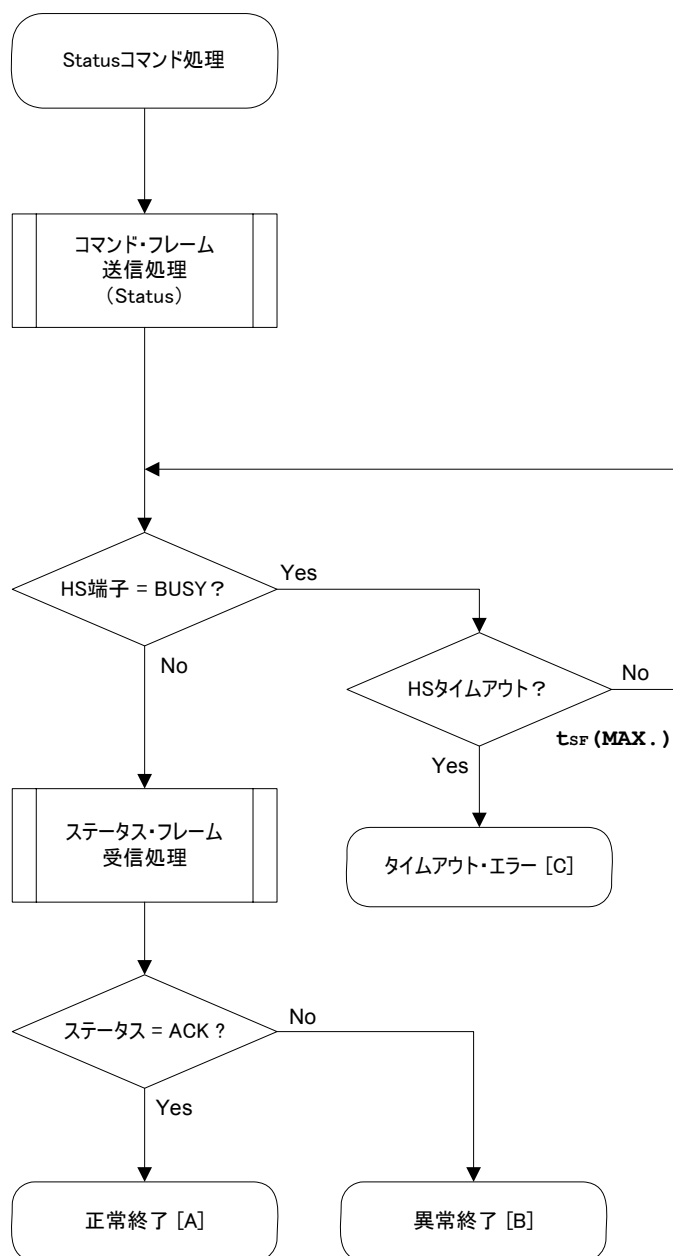
コマンド・フレーム送信処理により、**Statusコマンド**を送信します。
 HS端子によるV850ES/Sx3のBUSYチェックを行います。
 BUSYのタイムアウトが発生した場合は、**タイムアウト・エラー[C]**となります
 (タイムアウト時間 t_{SF} (MAX.))。
 ステータス・コードをチェックします。

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

7.4.3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	V850ES/Sx3 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	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	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	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。
	Write エラー	書き込みエラー
	MRG10 エラー	消去エラー
	MRG11 エラー	データ書き込み時の内部ベリファイ・エラー ,または ブランク・チェック・エラー
	ベリファイ・エラー	プログラマから送信されたデータとのベリファイで エラーが発生しました。
	プロテクト・エラー	Security Set コマンドで禁止した処理を実行しよう としました。
	否定応答 (NACK)	否定応答
タイムアウト・エラー [C]	-	HS のビジーにタイムアウトが発生しました。

7.4.4 フロー・チャート



7.4.5 サンプル・プログラム

Statusコマンド処理のサンプル・プログラムです。

```

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

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

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

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

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

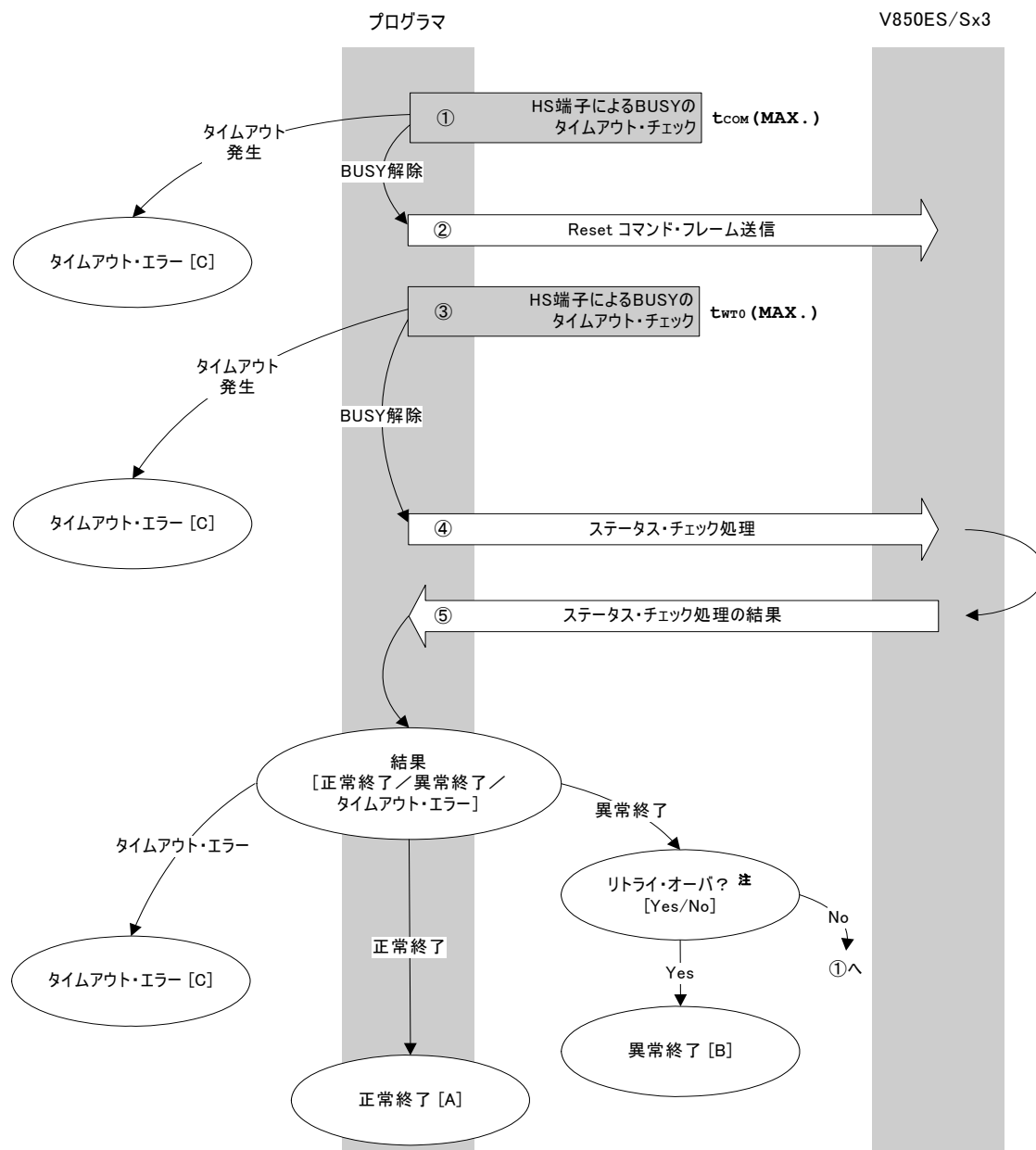
    return rc;          // case [A] or [B]
}

```

7.5 Resetコマンド

7.5.1 処理手順チャート

Resetコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

7.5.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYのタイムアウトが発生した場合は、**タイムアウト・エラー[C]**となります (タイムアウト時間 t_{COM} (MAX.))。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYのタイムアウトが発生した場合は、**タイムアウト・エラー[C]**となります (タイムアウト時間 t_{WTO} (MAX.))。

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。

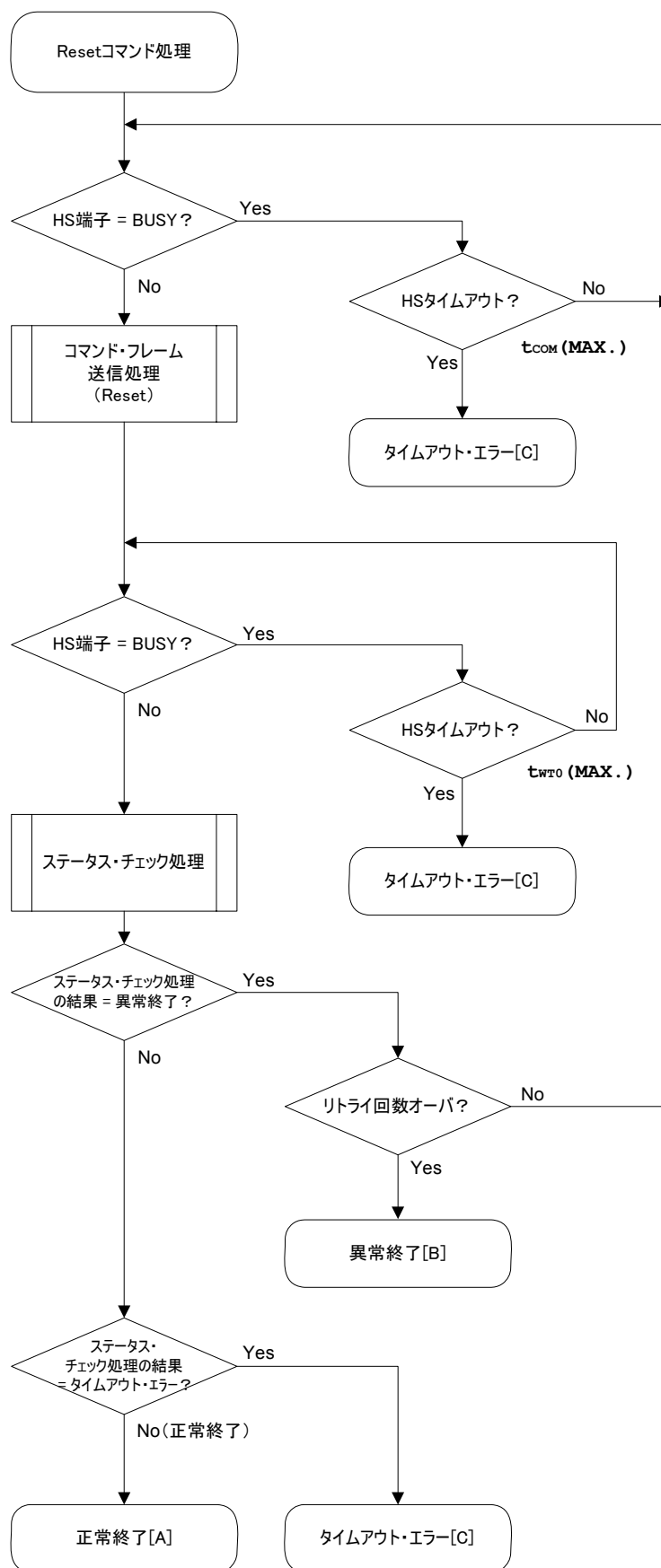
異常終了の場合 : リトライ・オーバでなければ より再実行します。
リトライ・オーバであれば、**異常終了[B]**です。

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V850ES/Sx3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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	ステータス・チェック処理がタイムアウト・エラーで終了した、またはHS端子のピジーでタイムアウトが発生しました。

7.5.4 フロー・チャート



7.5.5 サンプル・プログラム

Resetコマンド処理のサンプル・プログラムです。

```

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

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

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

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

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

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

    }

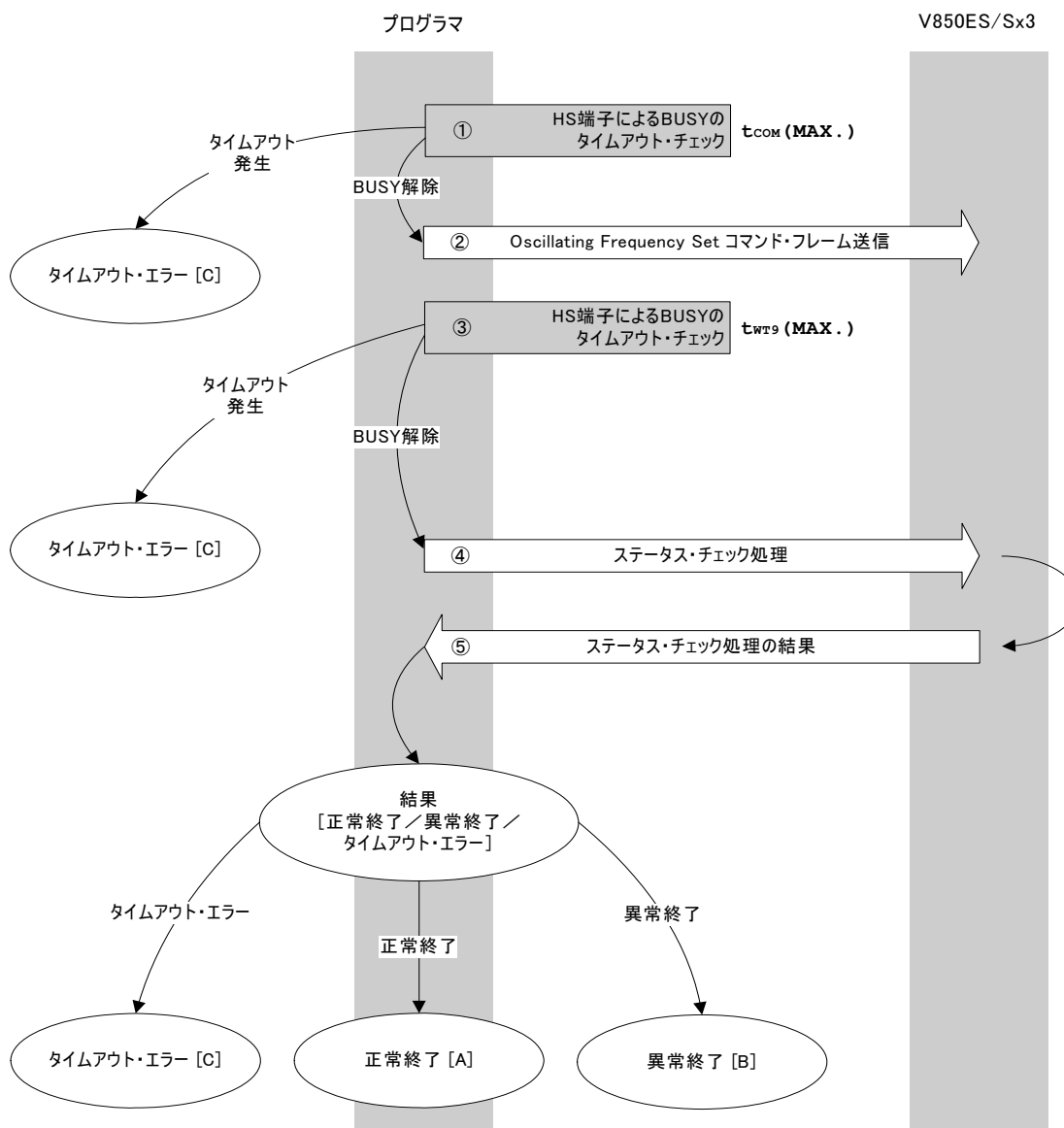
    // switch(rc) {
    //     case FLC_NO_ERR:    return rc;    break; // case [A]
    //     case FLC_HSTO_ERR: return rc;    break; // case [C]
    //     default:          return rc;    break; // case [B]
    // }
    return rc;
}

```

7.6 Oscillating Frequency Setコマンド

7.6.1 処理手順チャート

Oscillating Frequency Setコマンド処理手順



7.6.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー [C]**となります
(タイムアウト時間 $t_{COM}(MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー[C]**となります
(タイムアウト時間 $t_{WT9}(MAX.)$)。

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。

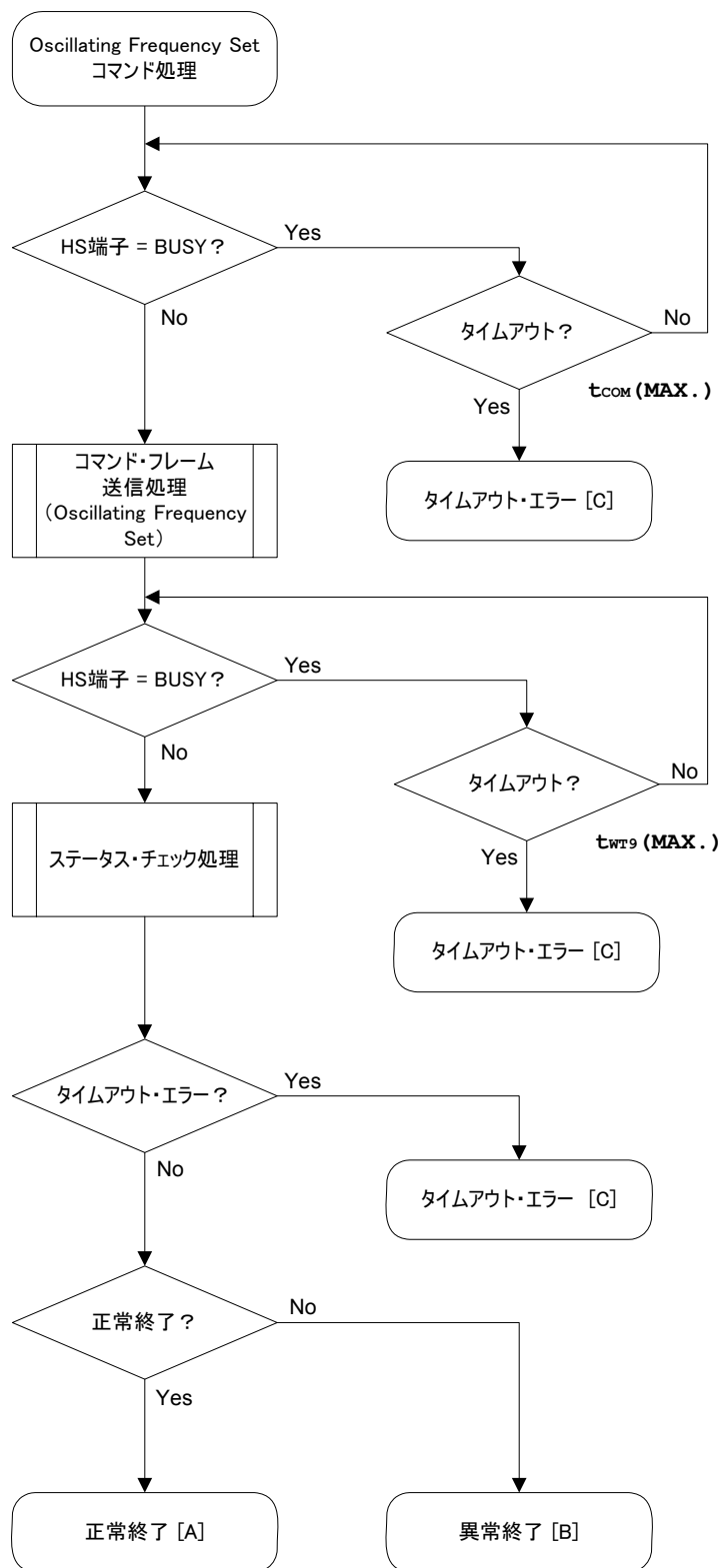
異常終了の場合 : **異常終了[B]** です。

タイムアウト・エラーの場合 : **タイムアウト・エラー[C]** です。

7.6.3 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、V850ES/Sx3 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	HS 端子のピジーでタイムアウトしました。

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-HS)
/*
/*
/*****
/* [i] u8 clk[4]    ... frequency data (D1-D4)
/* [r] u16          ... error code
/*****
/*****
u16      fl_hs_setclk(u8 clk[])
{
    u16    rc;

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

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

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

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

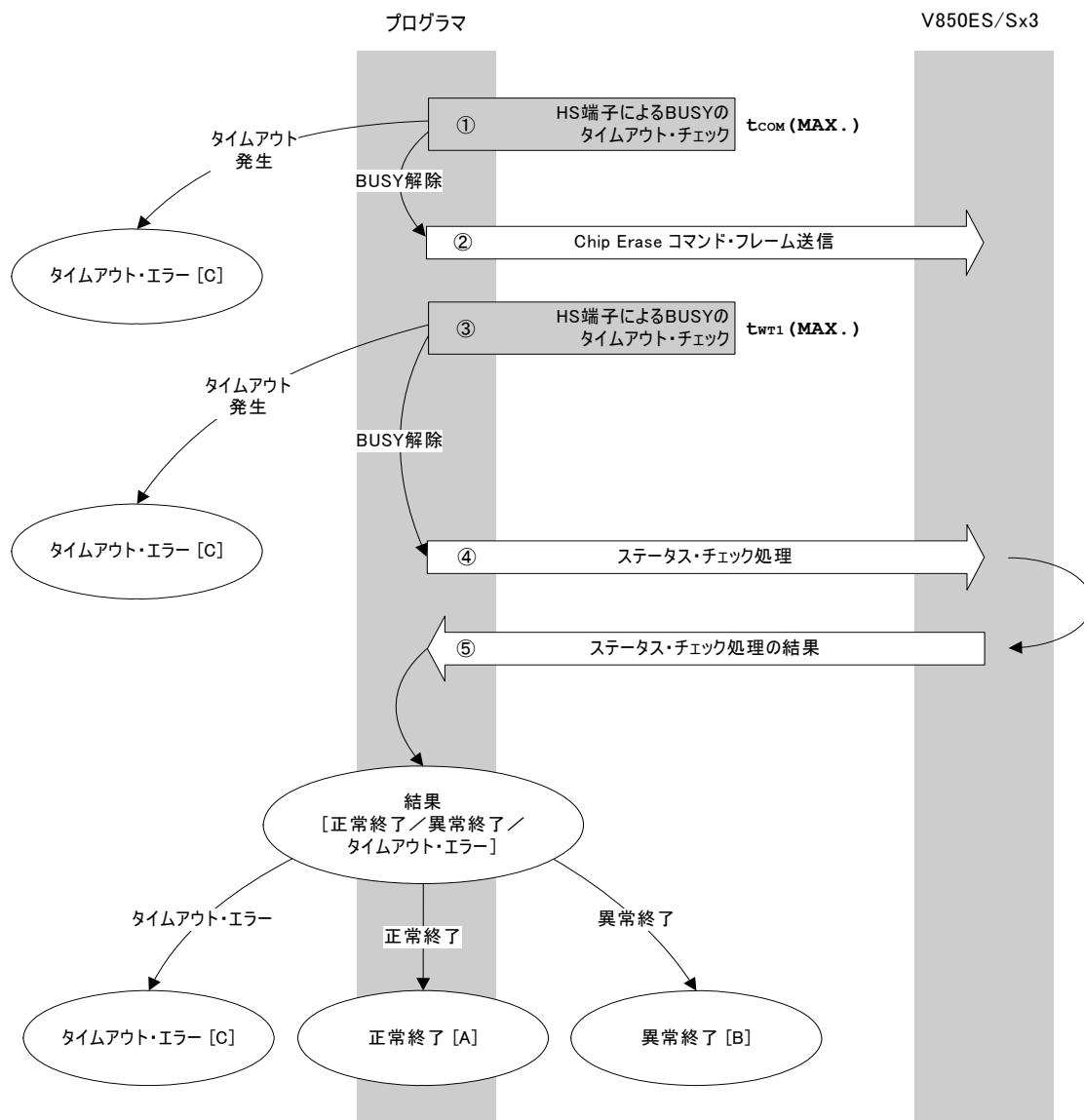
    rc = fl_hs_getstatus();           // get status frame
    // switch(rc) {
    //     case FLC_NO_ERR:    return rc;    break; // case [A]
    //     case FLC_HSTO_ERR: return rc;    break; // case [C]
    //     default:          return rc;    break; // case [B]
    // }
    return rc;
}

```

7.7 Chip Eraseコマンド

7.7.1 処理手順チャート

Chip Eraseコマンド処理手順



7.7.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー[C]**となります
(タイムアウト時間 t_{com} (MAX.))。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー[C]**となります
(タイムアウト時間 t_{WT1} (MAX.))。

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。

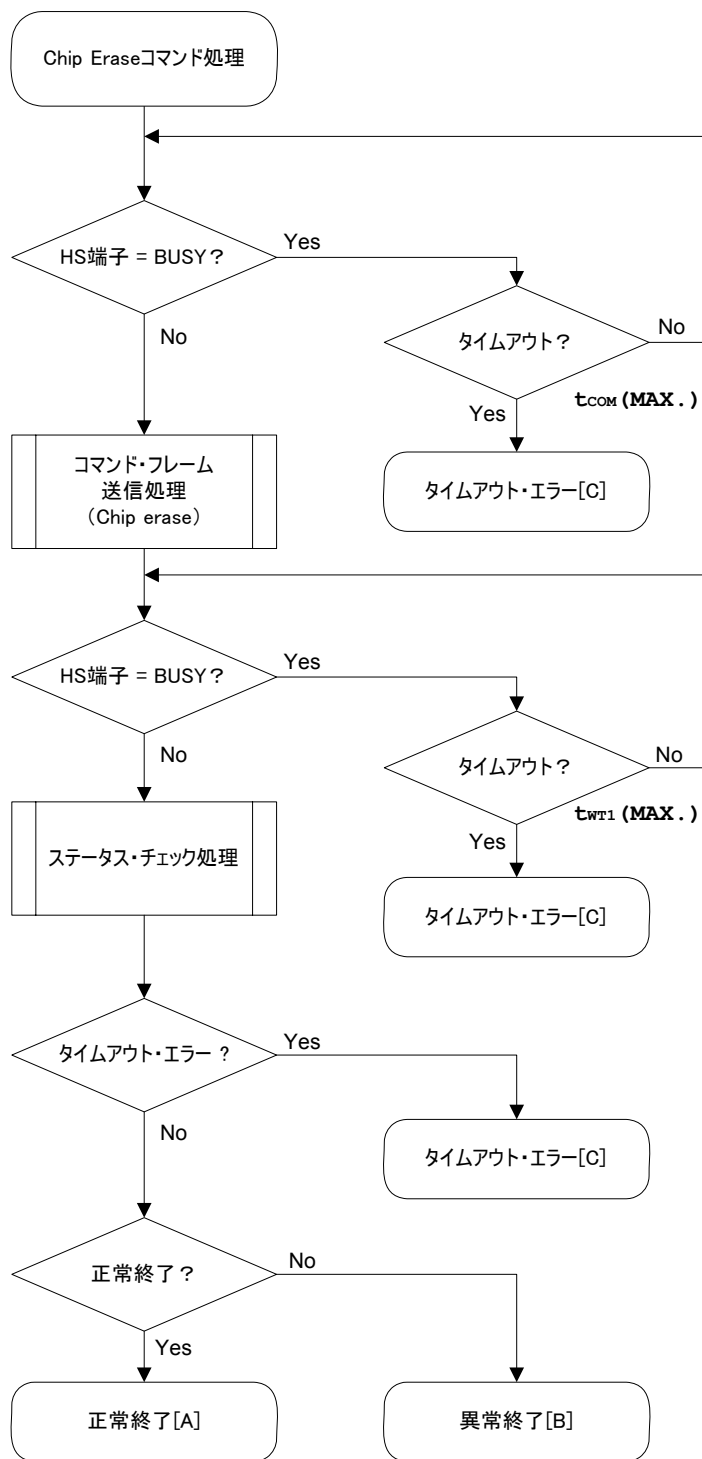
ステータス・チェック処理の結果に応じて次の処理を行います。

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。

7.7.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、チップ消去が正常に実行されたことを示します
異常終了 [B]	チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、Chip Erase コマンドが禁止となる設定になっています。
	否定応答 (NACK)	15H	・処理中にステータス・コマンド以外を受信しました。 ・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)。
	WRITE エラー	1CH	消去エラーが発生しました。
	MRG10 エラー	1AH	
MRG11 エラー	1BH		
タイムアウト・エラー [C]		-	HS 端子のピジーでタイムアウトしました。

7.7.4 フロー・チャート



7.7.5 サンプル・プログラム

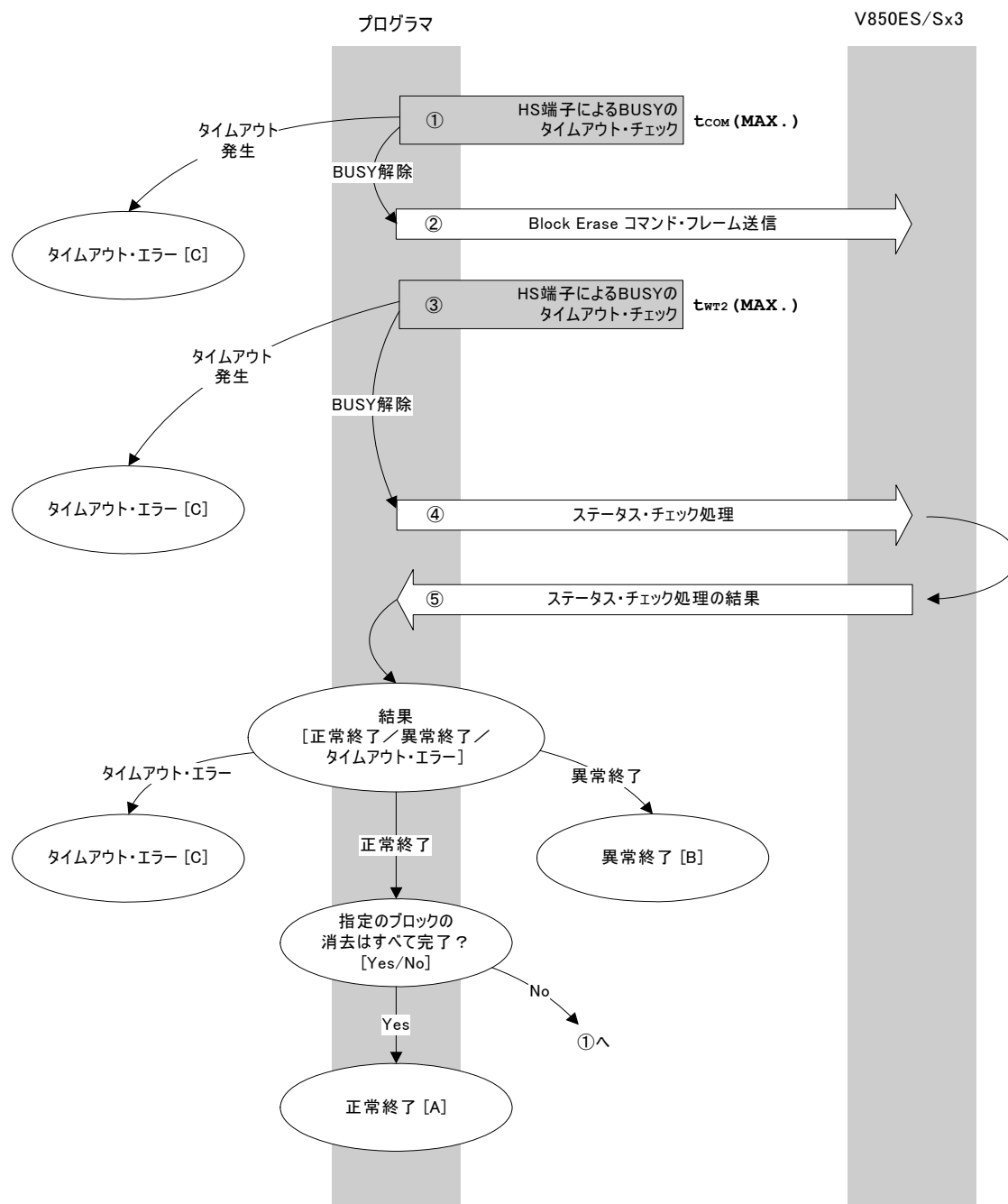
Chip Eraseコマンド処理のサンプル・プログラムです。

```
/******  
/*  
/* Erase all(chip) command (CSI-HS)  
/*  
/*  
/******  
/* [r] u16      ... error code  
/******  
u16      fl_hs_erase_all(void)  
{  
    u16      rc;  
  
    if (hs_busy_to(tCOM_MAX))  
        return  FLC_HSTO_ERR;                // t.o. detected  
  
    if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))  
                                                // send "ChipErase" command  
        return  rc;                          // case [C]  
  
    if (hs_busy_to(tWT1_MAX))  
        return  FLC_HSTO_ERR;                // case [C]  
  
    rc = fl_hs_getstatus();                    // get status frame  
// switch(rc) {  
//     case  FLC_NO_ERR:      return rc;      break; // case [A]  
//     case  FLC_HSTO_ERR:   return rc;      break; // case [C]  
//     default:              return rc;      break; // case [B]  
// }  
    return rc;  
}
```

7.8 Block Eraseコマンド

7.8.1 処理手順チャート

Block Eraseコマンド処理手順



7.8.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー[C]**となります
(タイムアウト時間 $t_{COM}(MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

BUSYがタイムアウトした場合は、**タイムアウト・エラー[C]**となります
(タイムアウト時間 $t_{WT2}(MAX.)$)。

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。

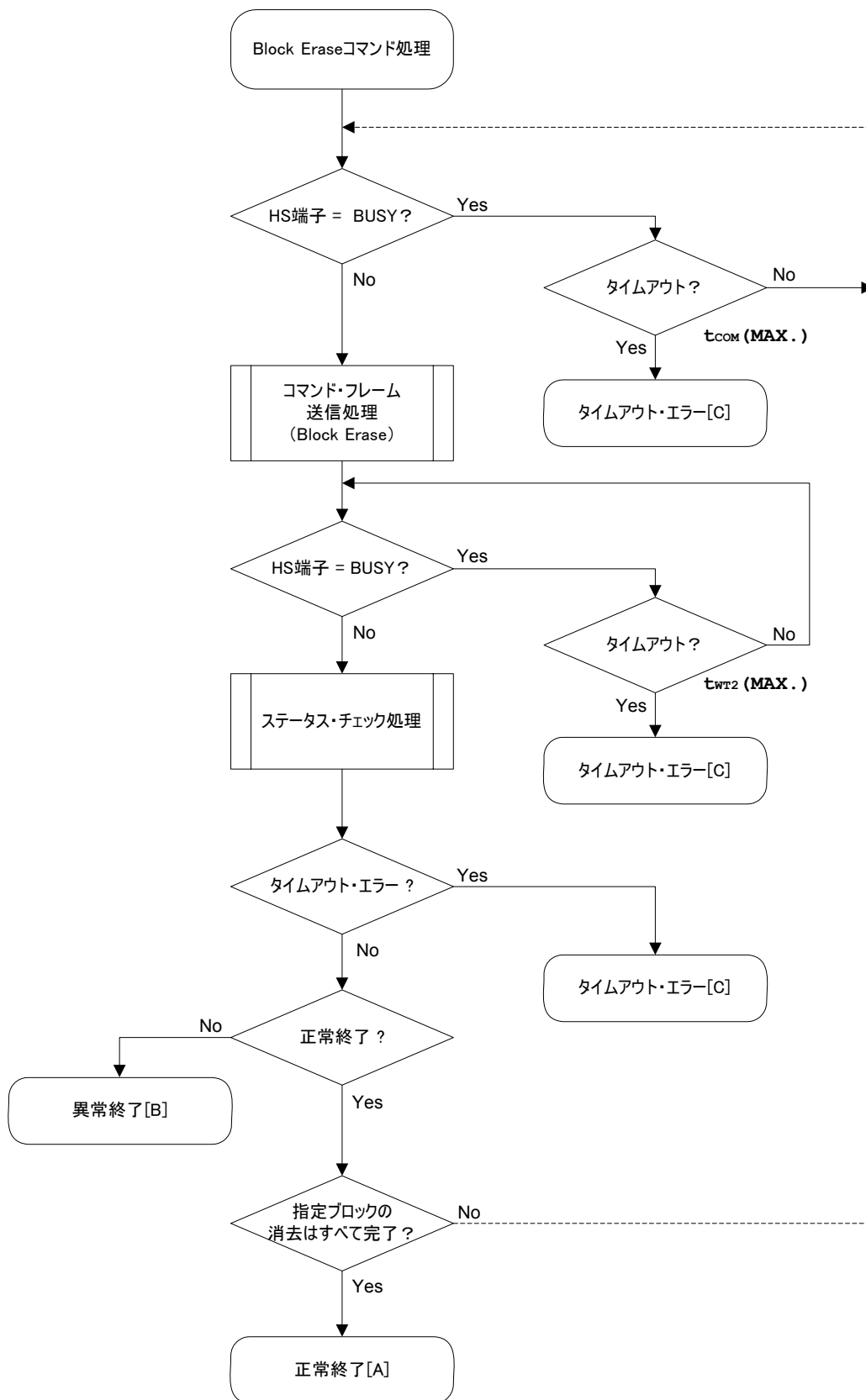
ステータス・チェック処理の結果に応じて次の処理を行います。

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

7.8.3 終了時の内容

終了内容	ステータス・コード	内容	
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、ブロック消去が正常に実行されたことを示します。	
異常終了 [B]	パラメータ・エラー	05H	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、Block Erase コマンドが禁止となる設定になっています。
	否定応答 (NACK)	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
MRG10 エラー	1AH	消去エラーが発生しました。	
タイムアウト・エラー [C]	-	HS 端子のビジーでタイムアウトしました。	

7.8.4 フロー・チャート



7.8.5 サンプル・プログラム

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

```

/*****
/*
/* Erase block command (CSI-HS)
/*
/*****
/* [i] u16 sblk    ... start block number
/* [i] u16 eblk    ... end block number
/* [r] u16        ... error code
/*****
u16      fl_hs_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);    // get tWT2(Max)

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

    if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 7, fl_cmd_prm))
        // send "Block Erase" command
        return rc;                      // case [C]

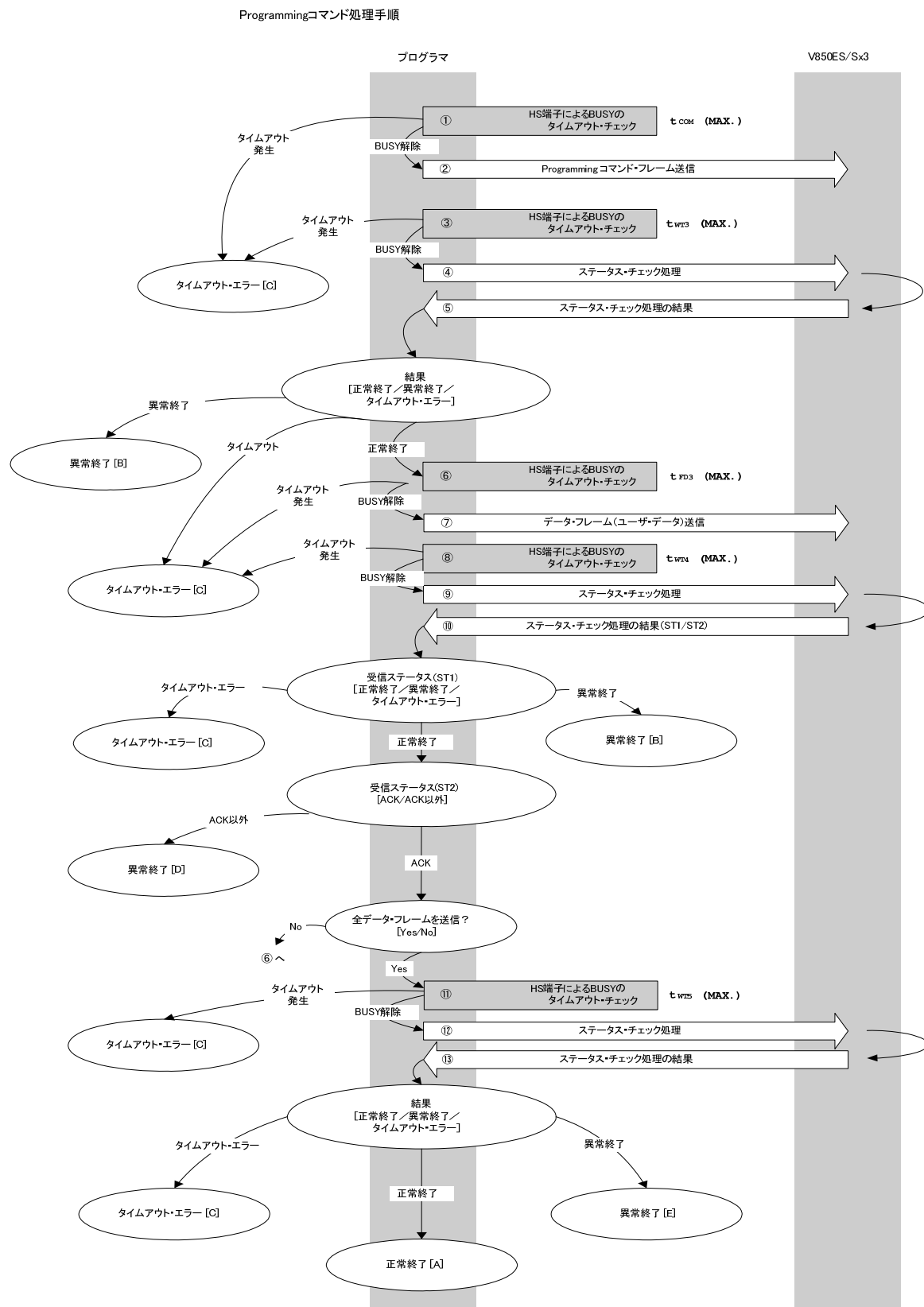
    if (hs_busy_to(wt2_max))
        return FLC_HSTO_ERR;            // t.o. detected :case [C]

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

```

7.9 Programmingコマンド

7.9.1 処理手順チャート



7.9.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。
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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT3} (MAX.)$)。
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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{FD3} (MAX.)$)。
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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT4} (MAX.)$)。
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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

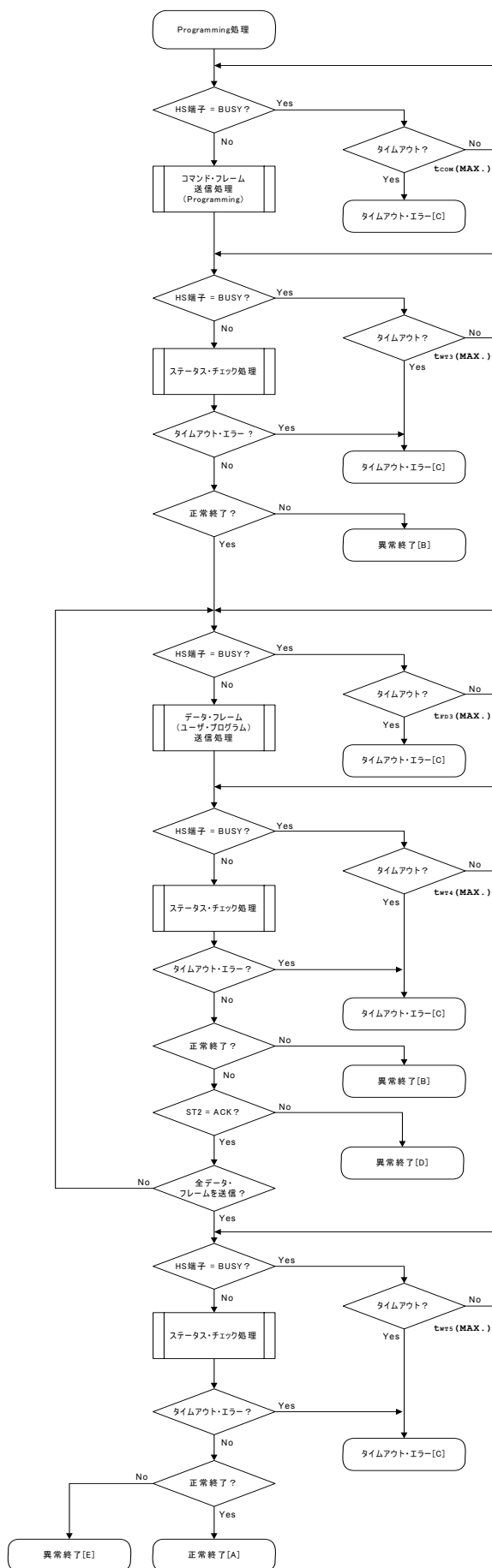
タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT5} (MAX.)$)。
ステータス・チェック処理により, ステータス・フレームを取得します。
ステータス・チェック処理の結果に応じて次の処理を行います。

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	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	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	セキュリティ設定で Programming コマンドが禁止となる設定になっています。
	否定応答 (NACK)	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	HS 端子のピジーでタイムアウトしました。
異常終了 [D] 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。

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-HS)
/*
/*****/
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****/
u16      fl_hs_write(u32 top, u32 bottom)
{
    u16      rc;
    u32      send_head, send_size;
    bool     is_end;
    u32      wt5_max;

    /*****/
    /*      set params
    /*****/
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    wt5_max = make_wt5_max(get_block_num(top, bottom));

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

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

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

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

    /*****/
    /*      send user data
    /*****/
    send_head = top;

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

```

```

        send_size = bottom - send_head + 1;
                // transmit size = (bottom - send_head)+1 byte
    }
    memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                // set data frame payload
    send_head += send_size;

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

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

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

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

}
/*****
/* Check internally verify */
*****/
if (hs_busy_to(wt5_max))
    return FLC_HSTO_ERR; // t.o. detected

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

```


7. 10. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。
 コマンド・フレーム送信処理にて **Verifyコマンド** を送信します。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT6} (MAX.)$)。
 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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{FD3} (MAX.)$)。
 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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

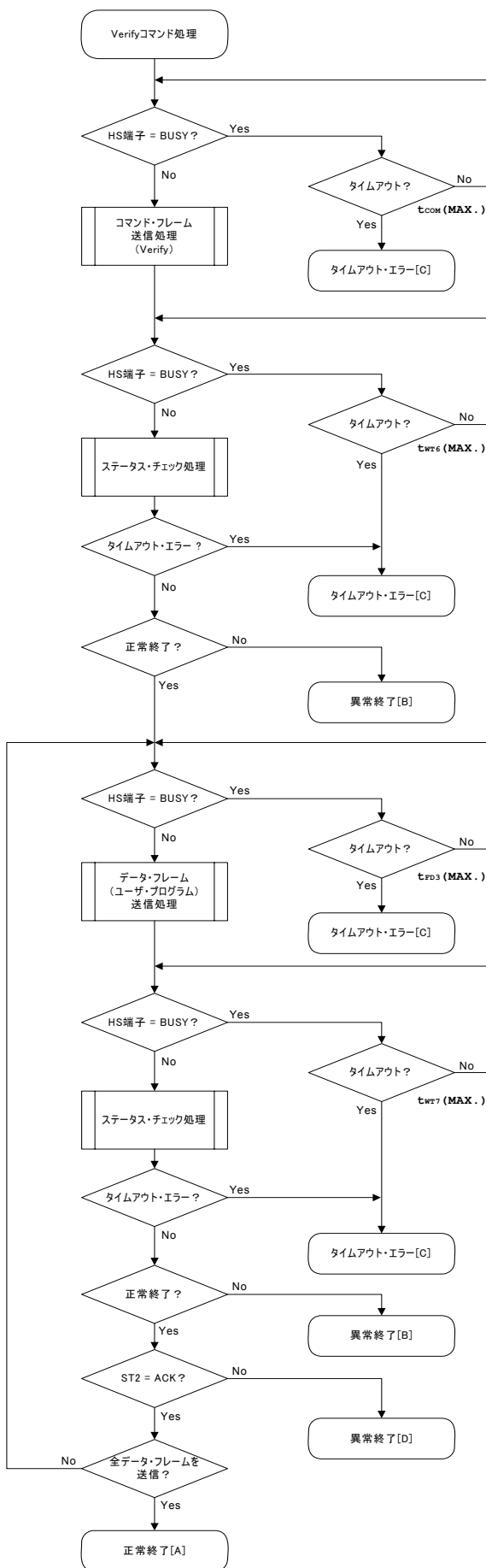
タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT7} (MAX.)$)。
 ステータス・チェック処理によりステータス・データを取得します。
 ステータス処理の結果 (ステータス・コード (ST1/ST2)) に応じて次の処理を行います (処理手順チャートやフロー・チャートも参照してください)。

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

7. 10. 3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ベリファイが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 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、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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	HS 端子のピジューでタイムアウトしました。
異常終了 [D]	ベリファイ・エラー	0FH	ベリファイに失敗しました。または、その他のエラーが発生しました。

7. 10. 4 フロー・チャート



7. 10. 5 サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

```

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

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

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

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

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

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

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

    /*****
    /*      send user data
    /*****
    send_head = top;

    while(1){

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

```

```
else{
    is_end = true;
    send_size = bottom - send_head + 1;
        // transmit size = (bottom - send_head)+1 byte
}
memcpy(fl_txdata_frm, buf+send_head, send_size); // set data frame payload
send_head += send_size;

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

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

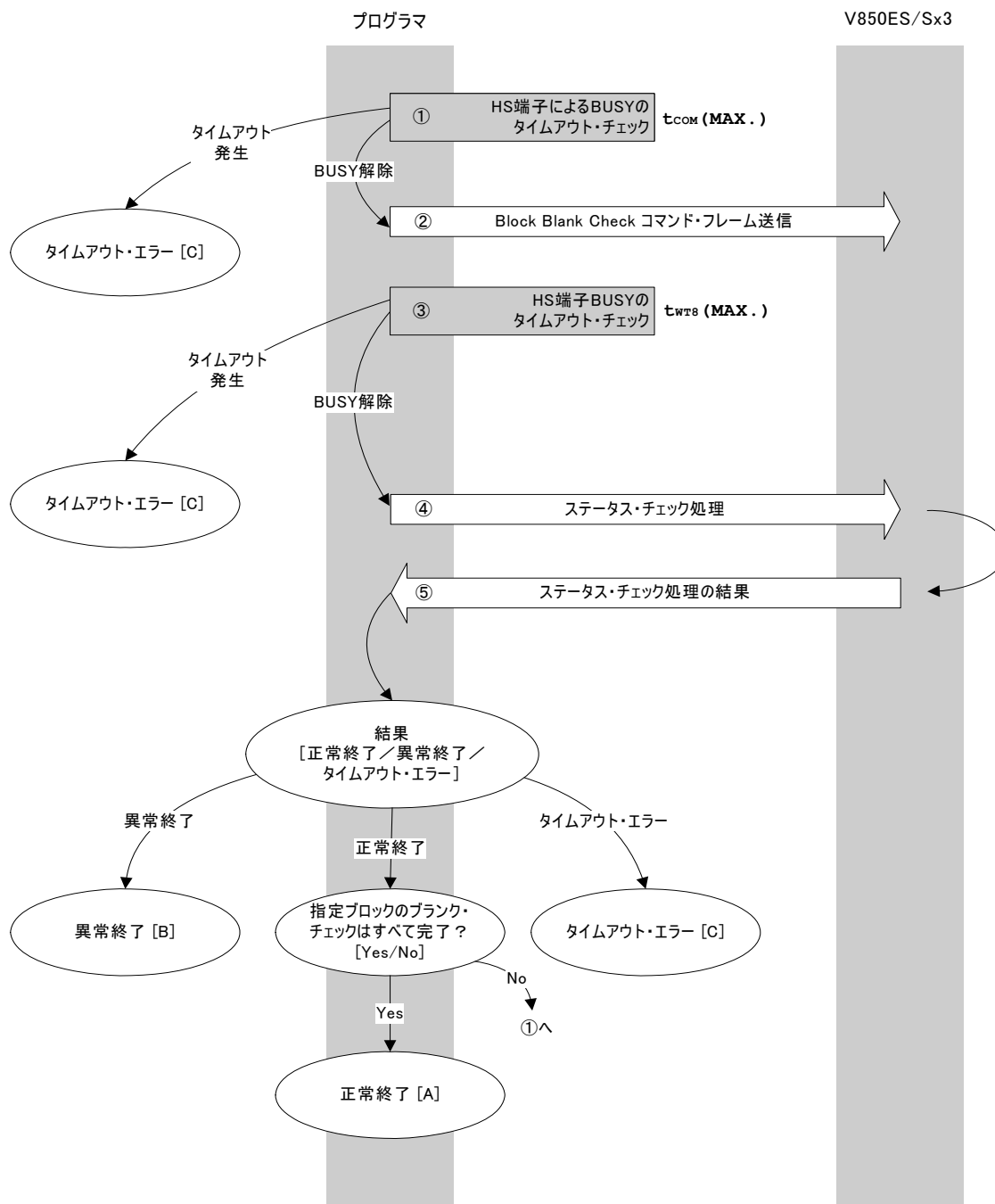
if (hs_busy_to(tWT7_MAX))
    return FLC_HSTO_ERR;           // t.o. detected

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

7. 11 Block Blank Checkコマンド

7. 11. 1 処理手順チャート

Block Blank Checkコマンド処理手順



7.11.2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。
 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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT8} (MAX.)$)。
 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。

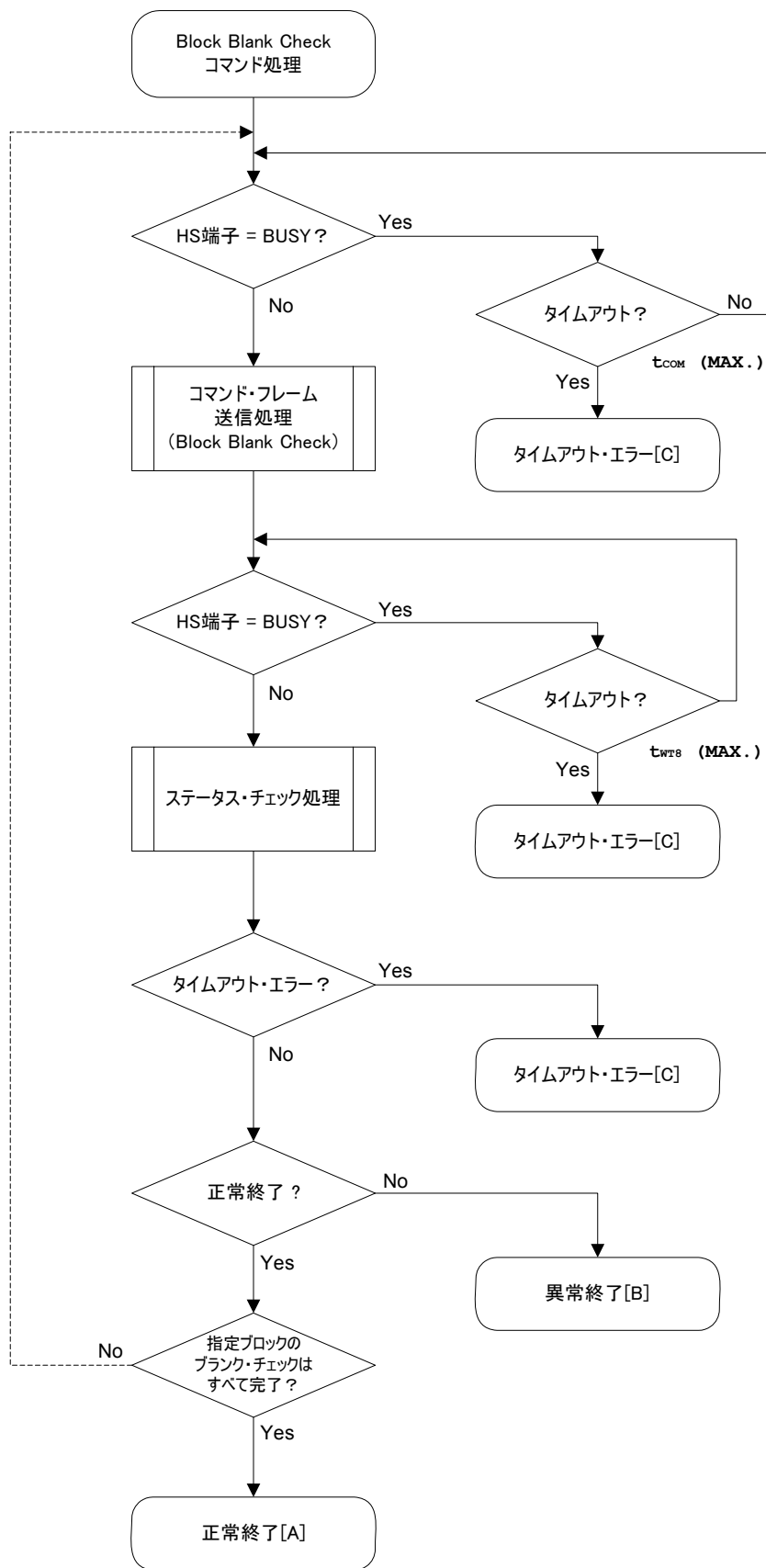
ステータス・チェック処理の結果に応じて次の処理を行います。

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

7.11.3 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。
	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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常の場合 (データ長 (LEN) 不正, ETX なしなど)
	MRG11 エラー	1BH	指定したブロックのフラッシュ・メモリがブランクではありません。
タイムアウト・エラー [C]	-	HS 端子のビジーでタイムアウトしました。	

7.11.4 フロー・チャート



7.11.5 サンプル・プログラム

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

```

/*****
/*
/* Block blank check command (CSI-HS)
/*
/*
/*****
/* [i] u16 sblk    ... start block number
/* [i] u16 eblk    ... end block number
/* [r] u16         ... error code
/*****
u16      fl_hs_blk_blank_chk(u16 sblk, u16 eblk)
{
    u16    rc;
    u32    wt8_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

    wt8_max = make_wt8_max(sblk, eblk); // get tWT8(Max)

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

    if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 7, fl_cmd_prm))
        return rc;                    // send "Block Blank Check" command
                                        // case [C]

    if (hs_busy_to(wt8_max))
        return FLC_HSTO_ERR;          // t.o. detected :case [C]

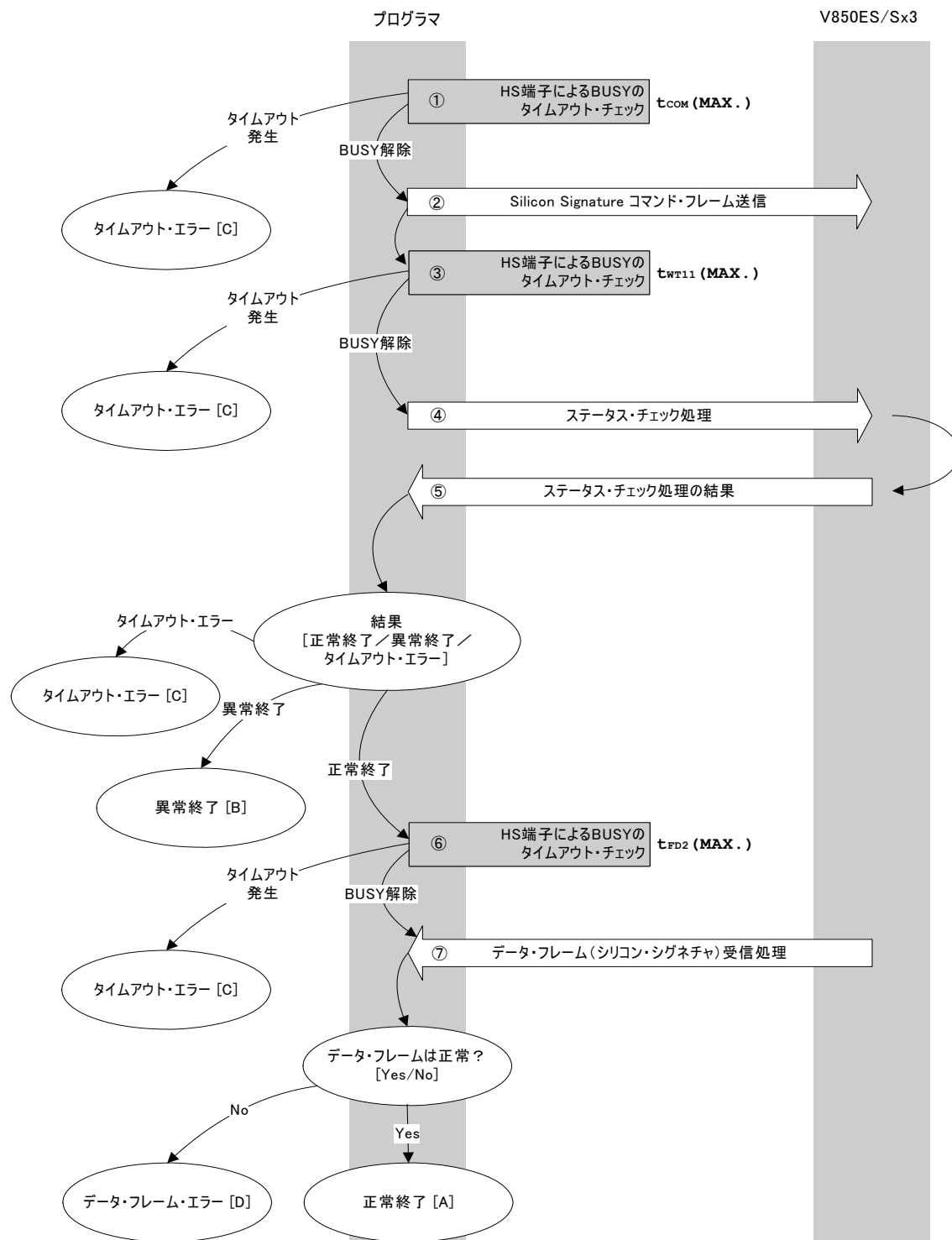
    rc = fl_hs_getstatus();            // get status frame
    // switch(rc) {
    //     case    FLC_NO_ERR:    return rc;    break; // case [A]
    //     case    FLC_HSTO_ERR: return rc;    break; // case [C]
    //     default:              return rc;    break; // case [B]
    // }
    return rc;
}

```

7. 12 Silicon Signatureコマンド

7. 12. 1 処理手順チャート

Silicon Signatureコマンド処理手順



7. 12. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT11} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{FD2} (MAX.)$)。

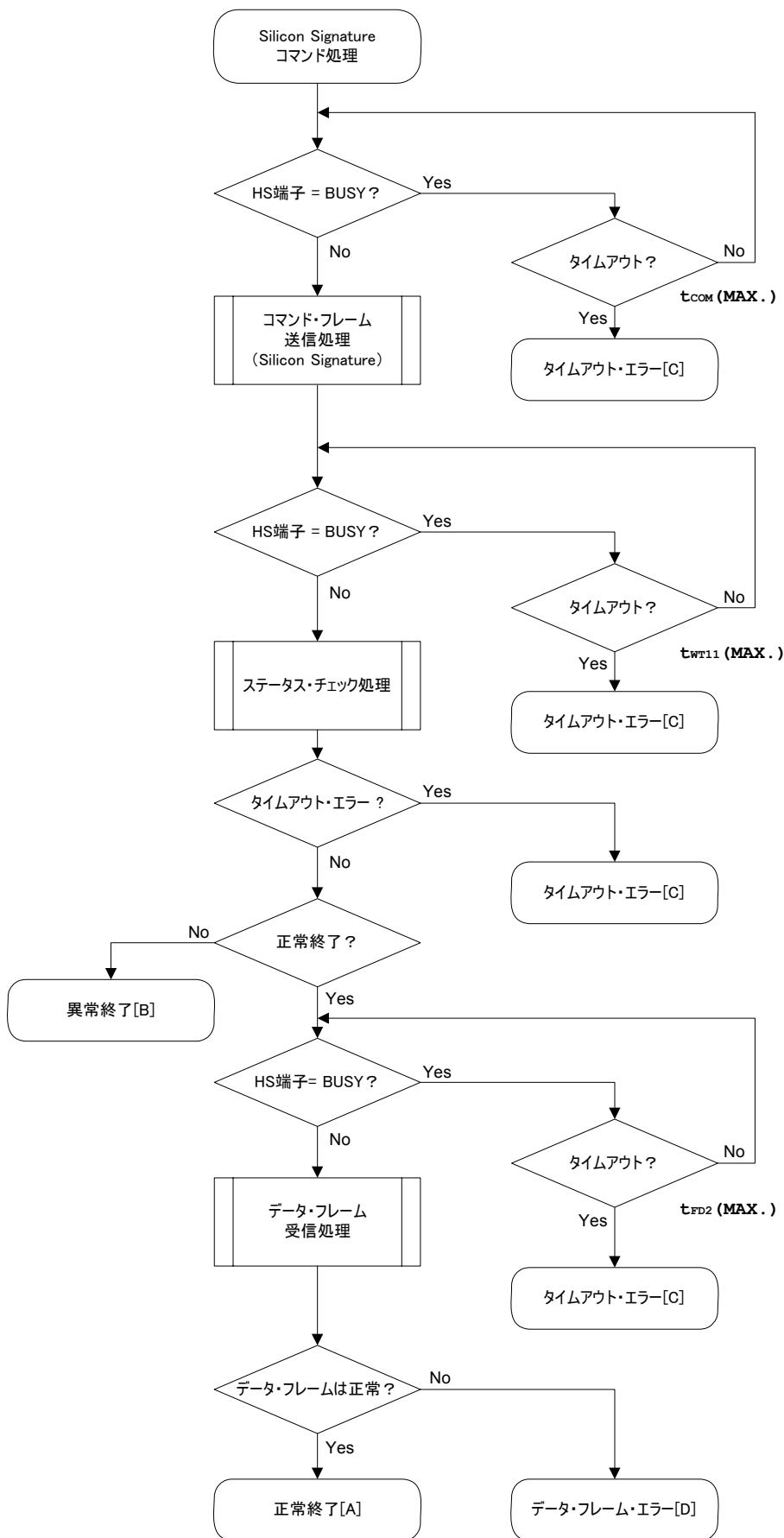
受信したデータ・フレーム (シリコン・シグネチャ・データ) をチェックします。

- 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	HS 端子のピジーでタイムアウトしました。
データ・フレーム・エラー [D]		-	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

7.12.4 フロー・チャート



7. 12. 5 サンプル・プログラム

Silicon Signatureコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Get silicon signature command (CSI-HS)
/*
/*
/*****
/* [i] u8 *sig... pointer to signature save area
/* [r] ul6 ... error code
/*****
ul6      fl_hs_getsig(u8 *sig)
{
    ul6      rc;

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

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

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

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

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

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

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

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

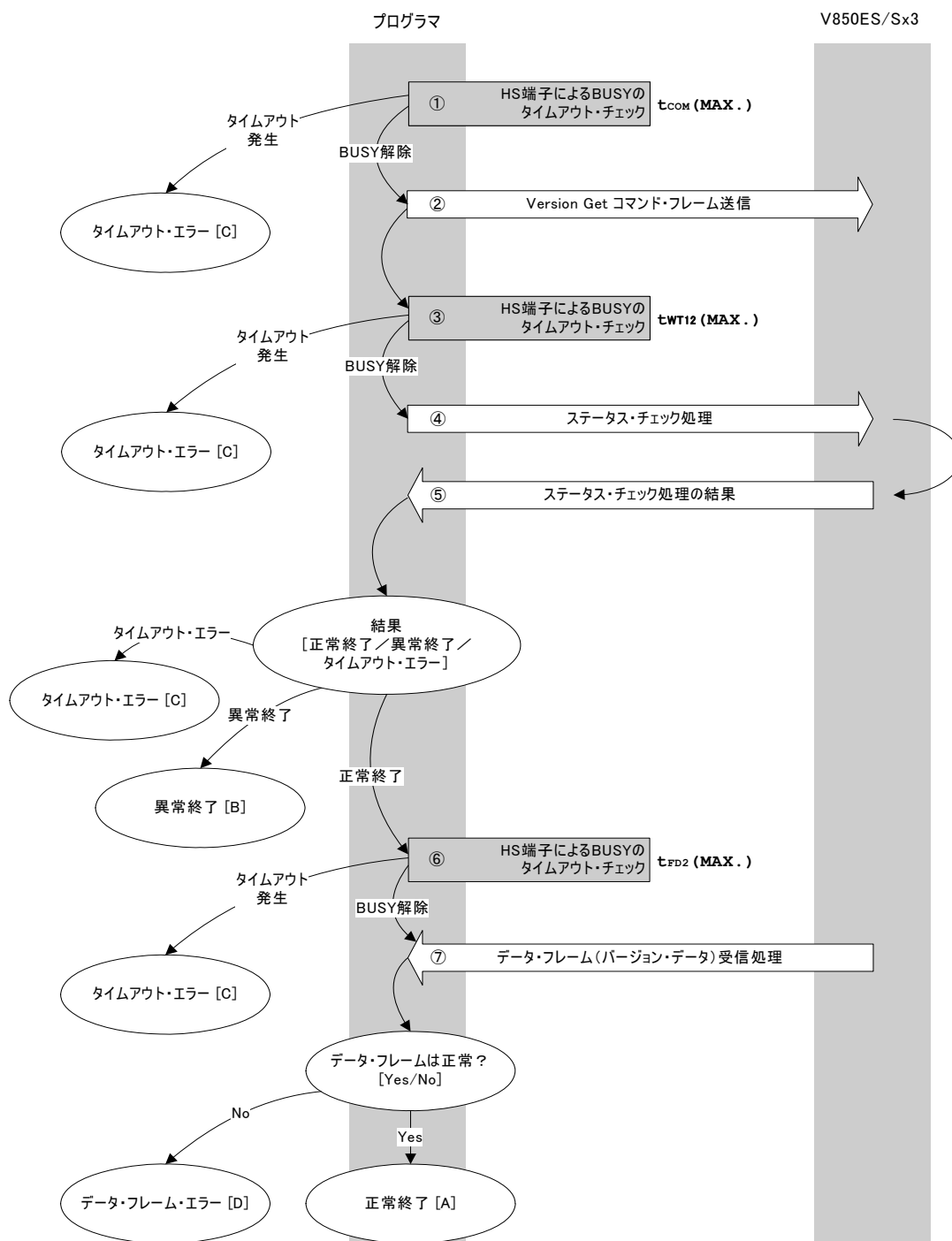
    return rc;                        // case [A]
}

```

7. 13 Version Getコマンド

7. 13. 1 処理手順チャート

Version Getコマンド処理手順



7. 13. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT12} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{FD2} (MAX.)$)。

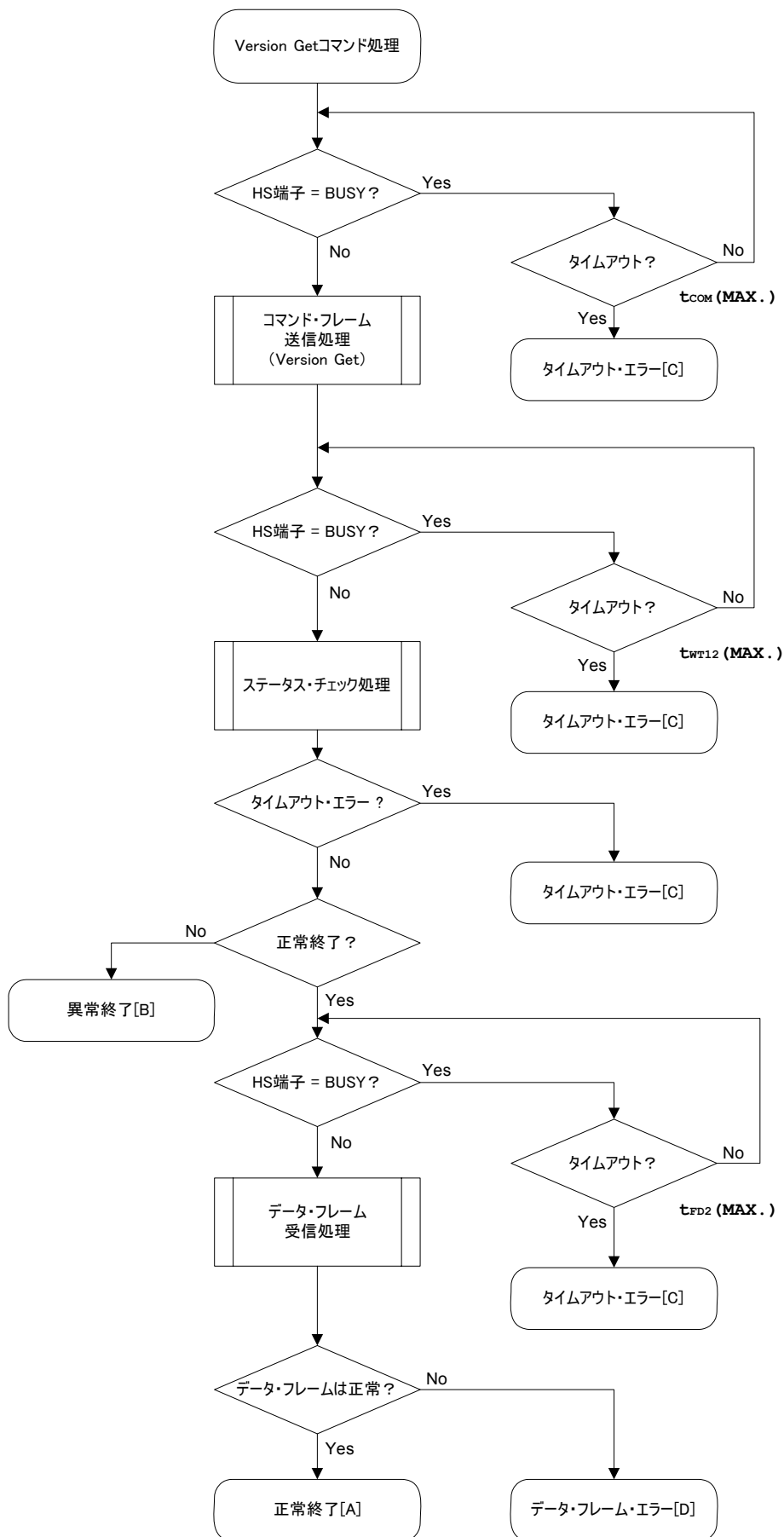
受信したデータ・フレーム (バージョン・データ) をチェックします。

- 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	HS端子のビジーでタイムアウトしました。
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-HS)
/*
/*
/*****
/* [i] u8 *buf      ... pointer to version data save area
/* [r] u16          ... error code
/*****
u16      fl_hs_getver(u8 *buf)
{
    u16      rc;

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

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

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

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

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

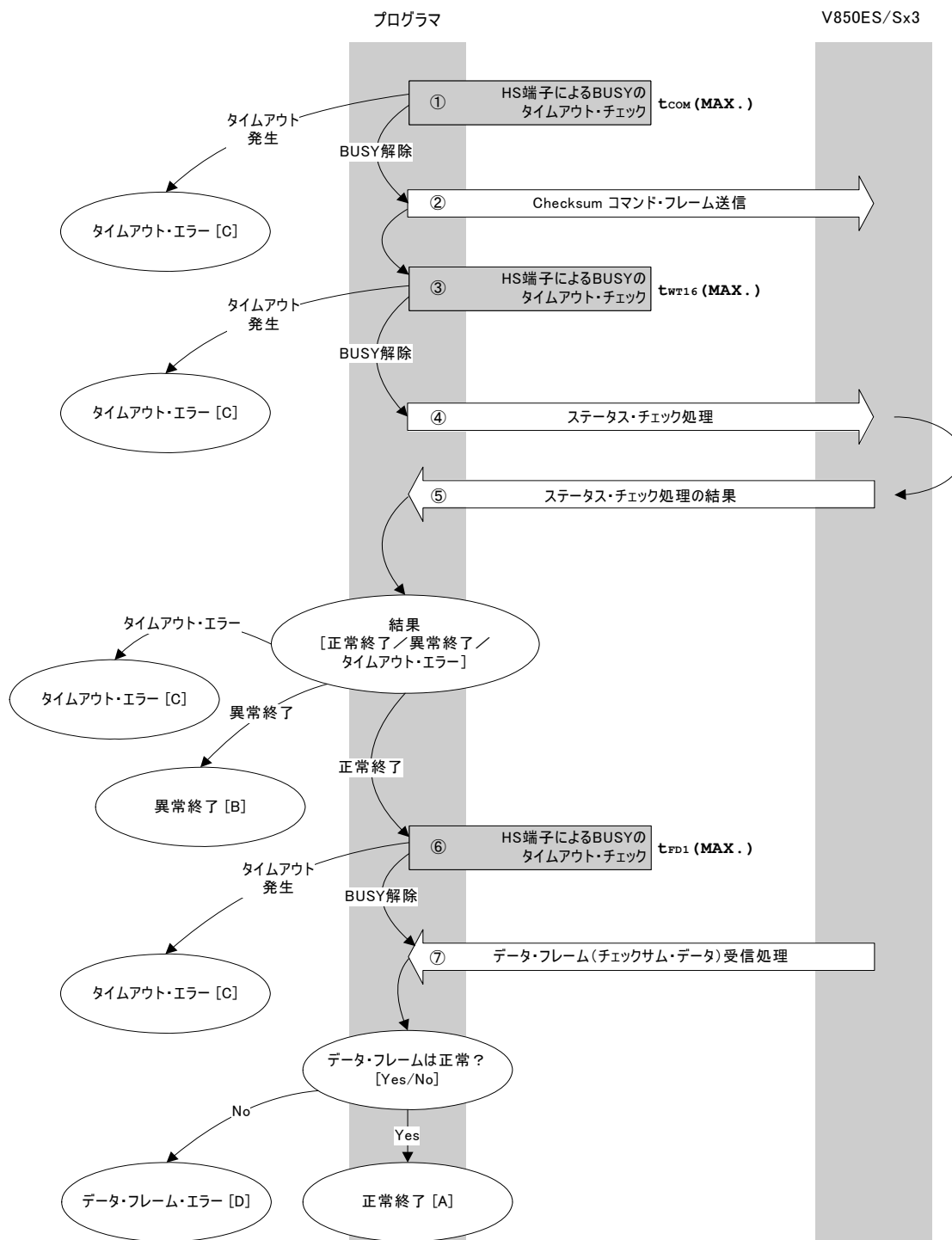
    rc = get_dfrm_hs(fl_rxdata_frm);   // get signature data
    switch(rc) {
        case FLC_NO_ERR:               break; // continue
        // case FLC_HSTO_ERR:          return rc; break; // case [C]
        default:                       return rc; break; // case [D]
    }
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD,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

```

7.14 Checksumコマンド

7.14.1 処理手順チャート

Checksumコマンド処理手順



7. 14. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{WT16} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{FD1} (MAX.)$)。

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。

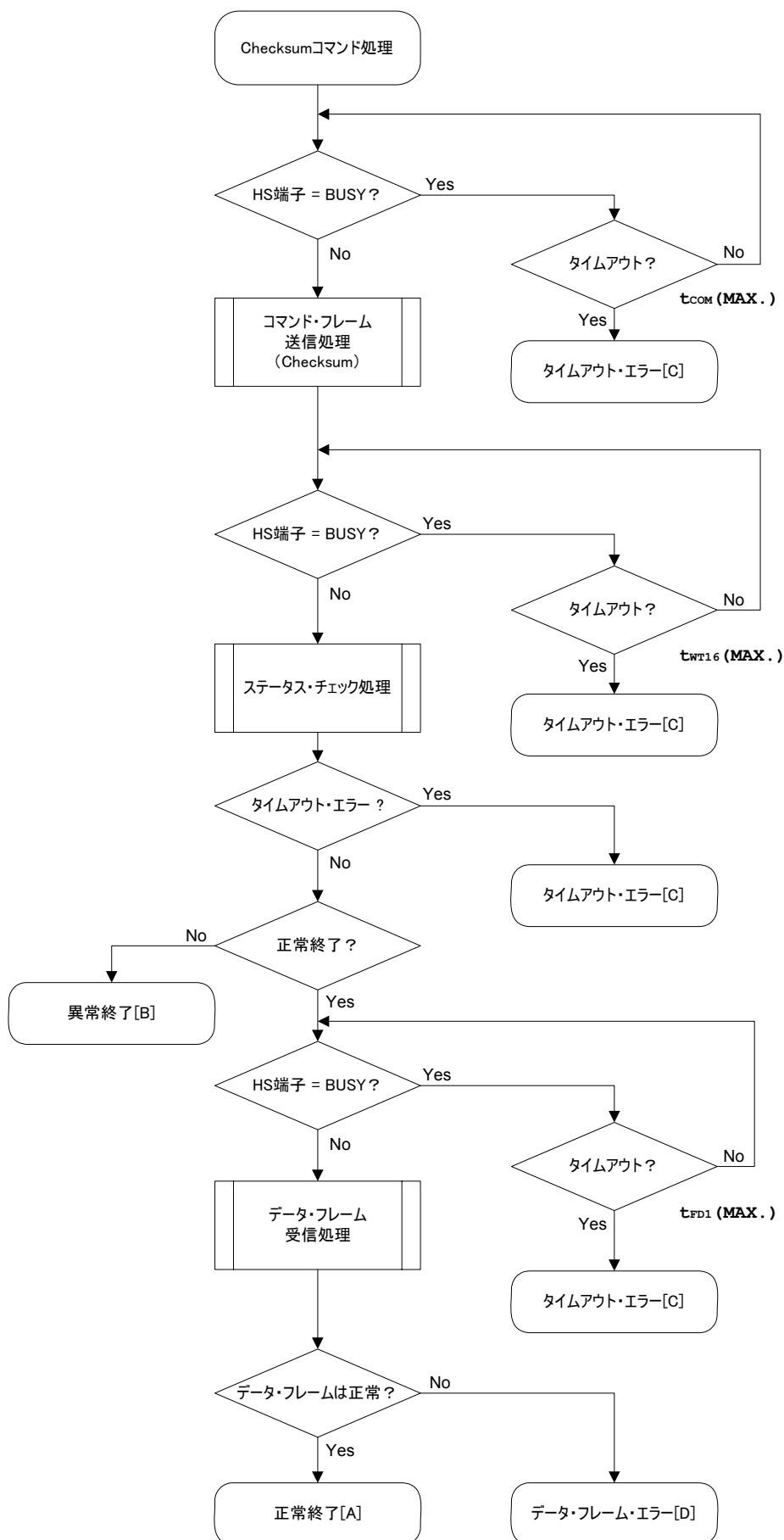
データ・フレームが正常の場合 : **正常終了[A]** です。

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, 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。
	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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	HS 端子のビジーでタイムアウトしました。	
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-HS)
/*
/*****
/* [i] u16 *sum    ... pointer to checksum save area
/* [i] u32 top     ... start address
/* [i] u32 bottom  ... end address
/* [r] u16        ... error code
/*****
u16      fl_hs_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;
    u32    fdl_max;

    /*****
    /*      set params
    /*****
    set_range_prm(fl_cmd_prm, top, bottom);          // set SAH/SAM/SAL, EAH/EAM/EAL
    fdl_max = get_fdl_max(get_block_num(top, bottom)); // get tFD1(Max)

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

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

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

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

    /*****
    /*      get data frame (Checksum data)
    /*****
    if (hs_busy_to(fdl_max))
        return FLC_HSTO_ERR;          // t.o. detected :case [C]

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

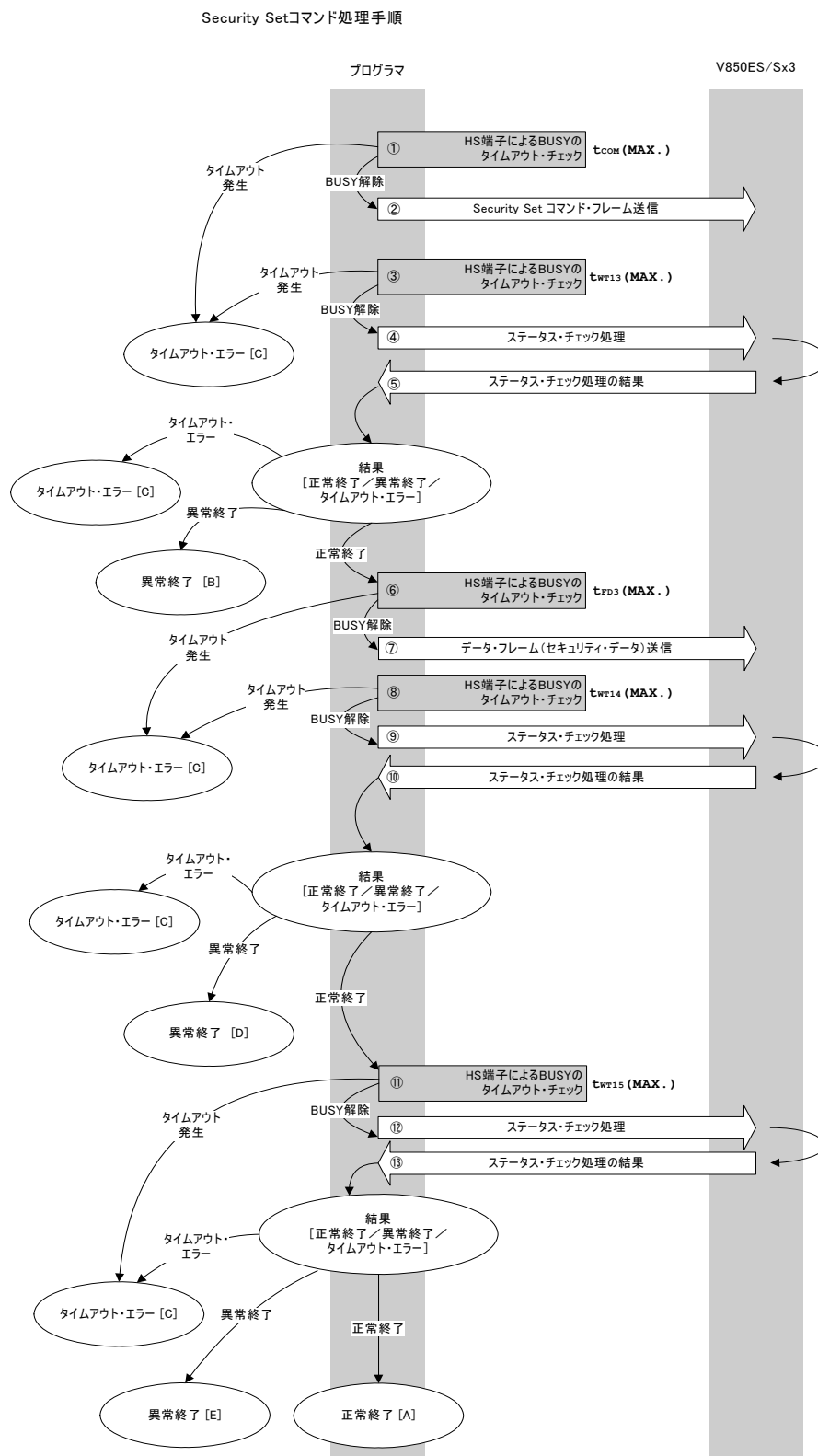
    switch(rc) {
        case    FLC_NO_ERR:            break; // continue
    //    case    FLC_HSTO_ERR:    return rc;    break; // case [C]
        default:            return rc;    break; // case [D]
    }
}

```

```
*sum = (fl_rxddata_frm[OFS_STA_PLD] << 8) + fl_rxddata_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



7. 15. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT13} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{FD3} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT14} (MAX.)$)。

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[D]** 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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT15} (MAX.)$)。

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。

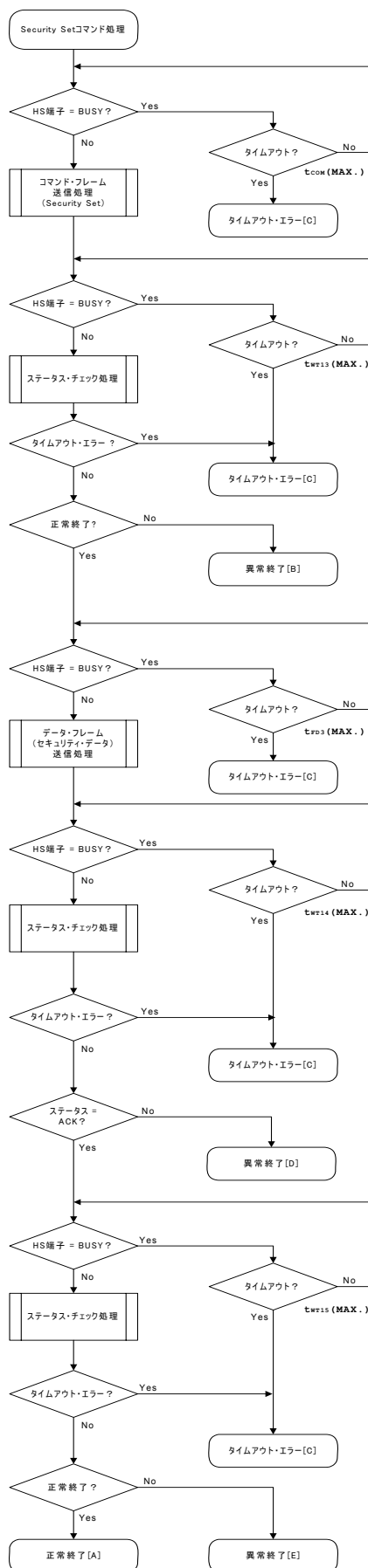
ステータス・チェック処理の結果に応じて次の処理を行います。

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[E]** 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]	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	<ul style="list-style-type: none"> ・ 処理中にステータス・コマンド以外を受信しました。 ・ コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D]	否定応答 (NACK)	15H	セキュリティ・データ・フレームが異常です。
	チェックサム・エラー	07H	送信したセキュリティ・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・データが次の場合のエラーです。 <ul style="list-style-type: none"> ・ セキュリティを禁止から許可に変更 ・ ブート・ブロック・クラスタ書き換え禁止設定状態でブート・ブロック・クラスタの最終ブロック番号の値を変更
	パラメータ・エラー	05H	セキュリティ・データが次の場合のエラーです。 <ul style="list-style-type: none"> ・ ブート・ブロック・クラスタの最終ブロック番号がデバイスの最終ブロック番号より大きい ・ リセット・ベクタ・ハンドラ・アドレスの値が "00000000H" 以外
異常終了 [E]	MRG10 エラー	1AH	書き込みエラーが発生しました。
	MRG11 エラー	1BH	
	WRITE エラー	1CH	

7.15.4 フロー・チャート



7. 15. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Set security flag command (CSI-HS)
/*
/*****
/* [i] u8 scf      ... Security flag data
/* [r] u16        ... error code
/*****
u16      fl_hs_setscf(u8 scf, u8 bot, u32 vect)
{
    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 | 0b11100000; // "FLG" (bit 7,6,5 must be '1')
    fl_txdata_frm[1] = bot;          // "BOT"

    fl_txdata_frm[2] = (u8)(vect >> 16); // "ADH"
    fl_txdata_frm[3] = (u8)(vect >> 8); // "ADM"
    fl_txdata_frm[4] = (u8) vect; // "ADL"

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

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

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

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

    /*****
    /*      send data frame (security setting data)
    /*****
    if (hs_busy_to(tFD3_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    if (rc = put_dfrm_hs(5, fl_txdata_frm, true)) // send securithi setting data
        return rc; // error detected :case [C]

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

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

```

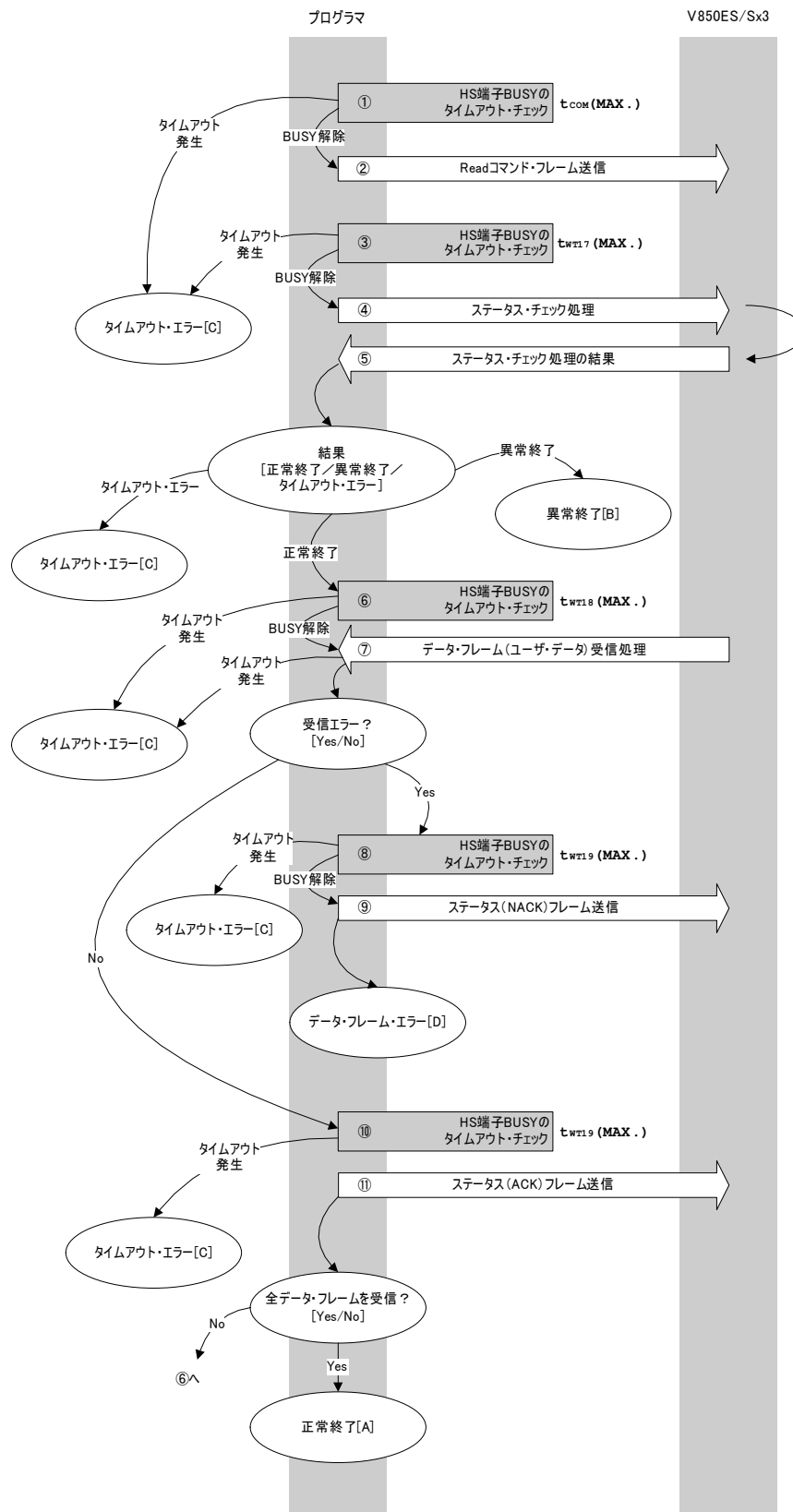
```
/*
*****
/*      Check internally verify      */
*****
if (hs_busy_to(tWT15_MAX))
    return FLC_HSTO_ERR;           // t.o. detected

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

7. 16 Readコマンド

7. 16. 1 処理手順チャート

Read コマンド処理手順



7. 16. 2 処理手順説明

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です (タイムアウト時間 $t_{COM} (MAX.)$)。

コマンド・フレーム送信処理により **Readコマンド** を送信します。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT17} (MAX.)$)。

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。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT18} (MAX.)$)。

データ・フレーム受信処理により、フラッシュROM内のデータ・フレーム (ユーザ・データ) を受信します。

正常終了の場合 : に進みます。
チェックサム・エラーなどの場合 : に進みます。
タイムアウト・エラーの場合 : **タイムアウト・エラー[C]** です。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT19} (MAX.)$)。

データ・フレーム送信処理により、NACKフレームを送信します。

データ・フレーム・エラー[D] です。

HS端子によるV850ES/Sx3のBUSYチェックを行います。

タイムアウトであれば **タイムアウト・エラー[C]** です

(タイムアウト時間 $t_{WT19} (MAX.)$)。

データ・フレーム送信処理により、ACKフレームを送信します。

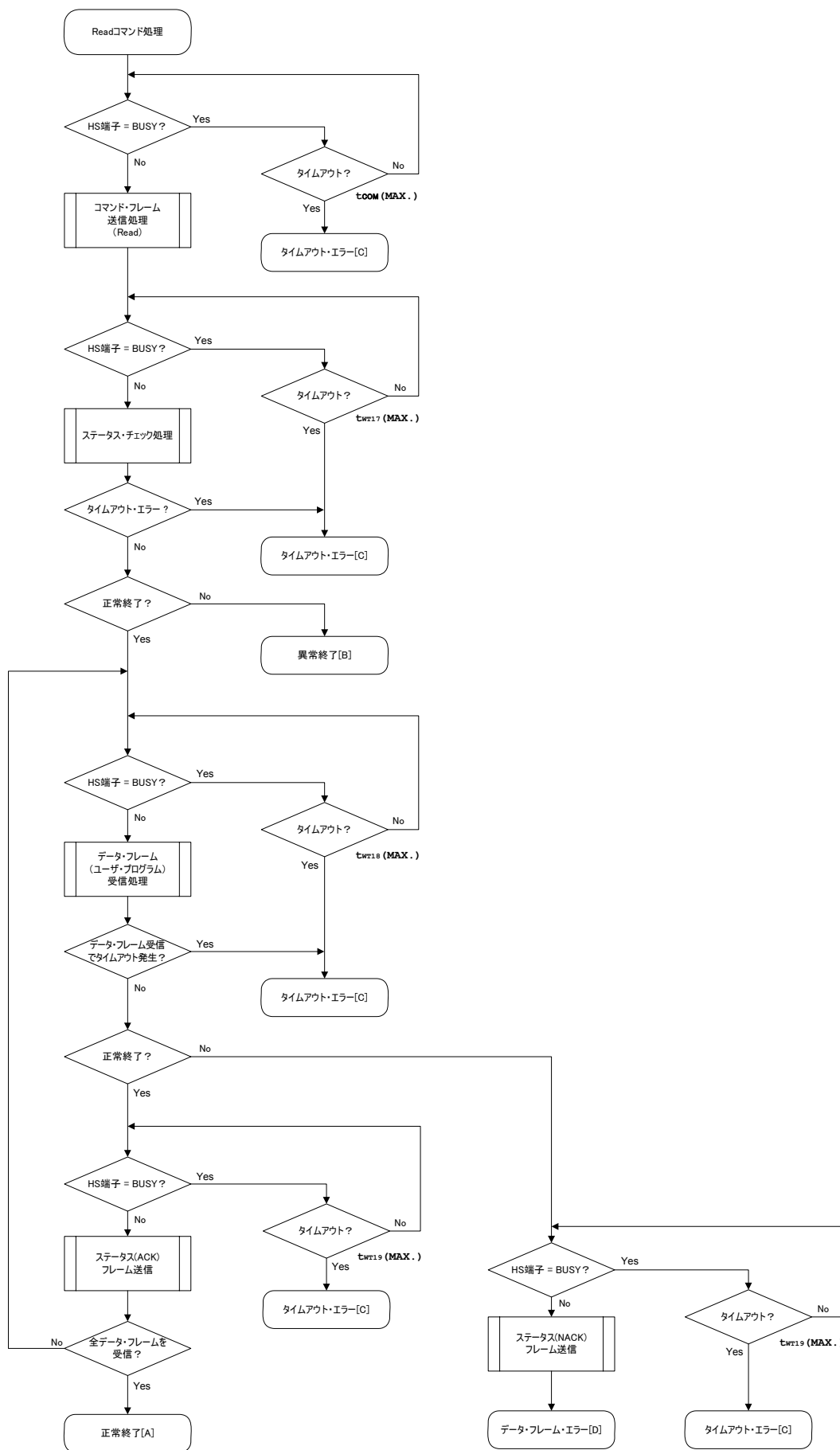
全データ・フレームを受信済みの場合は、**正常終了[A]** です。

まだ受信すべきデータ・フレームがある場合は、 から再実行します。

7. 16. 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、読み出し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	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「読み出し禁止」になっています。
	否定応答 (NACK)	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	HS 端子のビジーでタイムアウトしました。
データ・フレーム・エラー [D]		-	読み出し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。

7.16.4 フロー・チャート



7.16.5 サンプル・プログラム

Readコマンド処理のサンプル・プログラムです。

```

/*****
/*
/* Read command
/*
/*****
u16      fl_hs_read(u32 top, u32 bottom)
{
    u16    rc;
    u32    read_head;
    u16    len;
    u8     hooter;

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

/*****
/*      send command & check status
/*****
if (hs_busy_to(tCOM_MAX))
    return FLC_HSTO_ERR; // t.o. detected :case [C]

if (rc = put_cmd_hs(FL_COM_READ, 7, fl_cmd_prm))
    return rc;

if (hs_busy_to(tWT17_MAX))
    return FLC_HSTO_ERR; // t.o. detected :case [C]

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

/*****
/*      receive user data
/*****
read_head = top;

while(1){

    if (hs_busy_to(tWT18_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxddata_frm); // get ROM data from FLASH
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        case FLC_HSTO_ERR:      return rc; break; // case [C]

// case FLC_RX_DFSUM_ERR:
        default:                 // case [D]

        if (hs_busy_to(tWT19_MAX))
            return FLC_HSTO_ERR; // t.o. detected

        put_sfrm_hs(FLST_NACK);
// send status(NACK) frame

        return rc;
        break;

```

```
    }
    if (hs_busy_to(tWt19_MAX))
        return FLC_HSTO_ERR;    // t.o. detected

    put_sfrm_hs(FLST_ACK);        // send status(ACK) frame

    /* save ROM data */
    /* ***** */
    if ((len = fl_rxd_data_frm[OFS_LEN]) == 0)    // get length
        len = 256;

    memcpy(read_buf+read_head, fl_rxd_data_frm+2, len);
    // save to external RAM

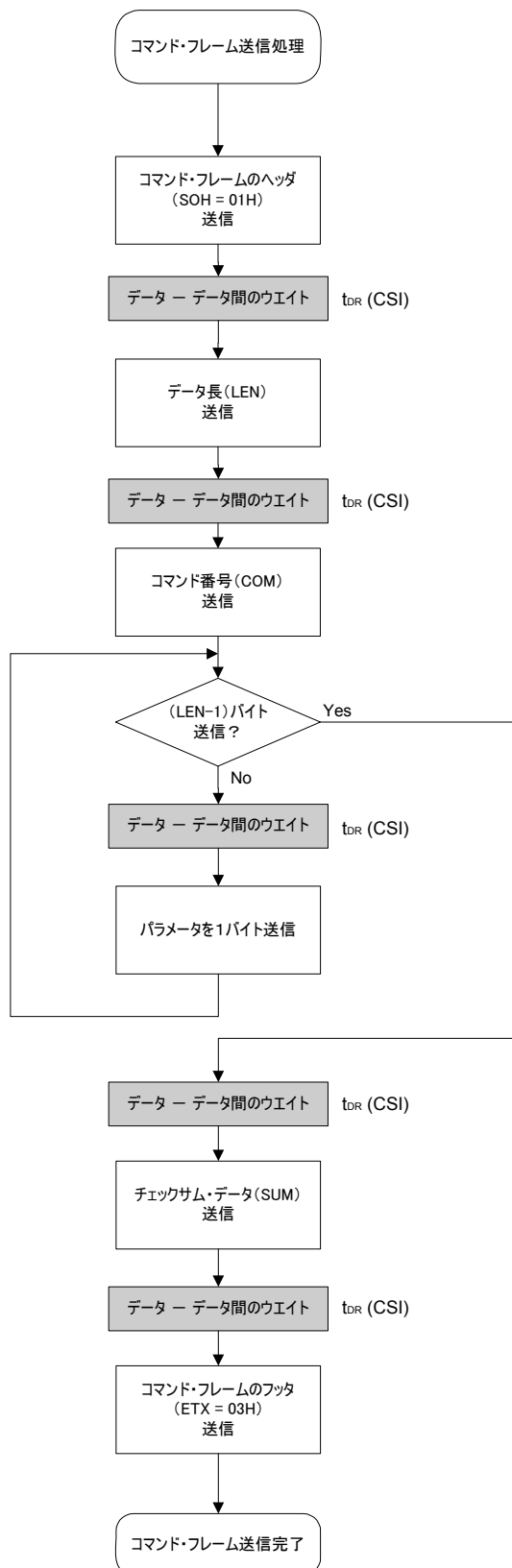
    read_head += len;

    /* end check */
    /* ***** */
    hooter = fl_rxd_data_frm[len + 3];
    if (hooter == FL_ETB)    // end frame ?
        continue;    // no
    break;    // yes
}

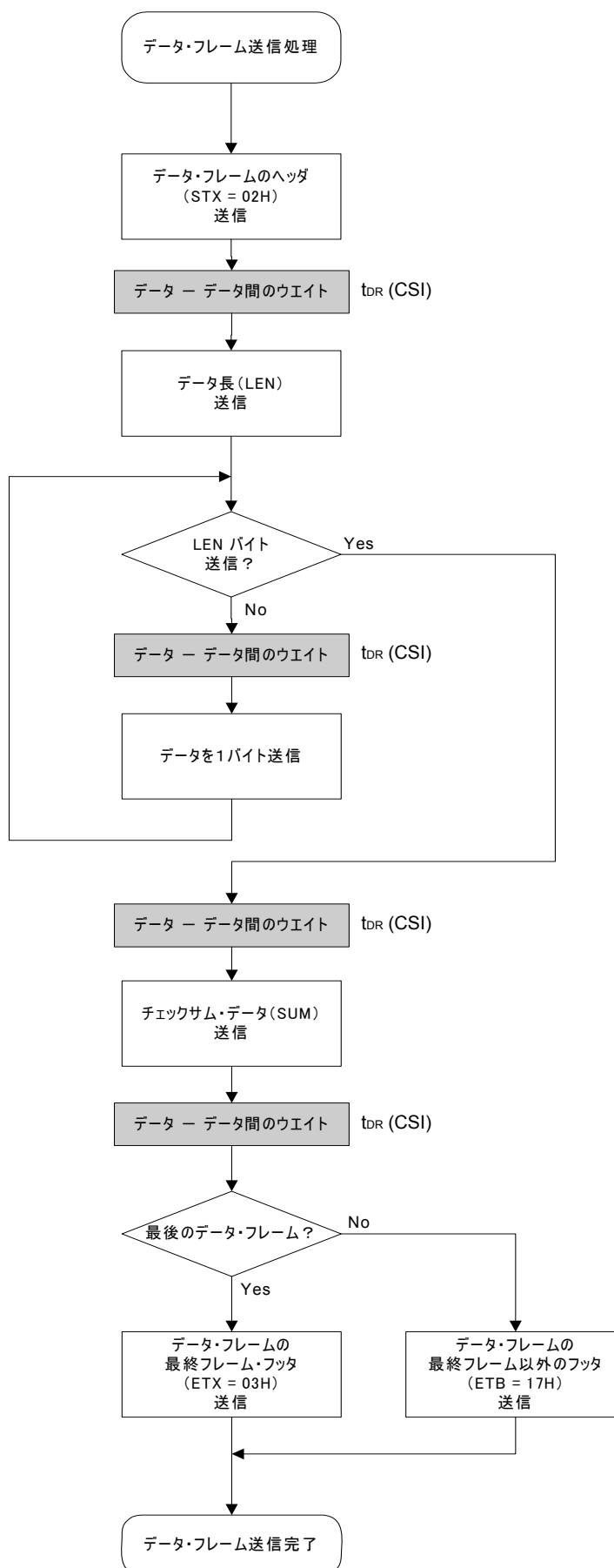
return FLC_NO_ERR;
}
```

第8章 3線式シリアルI/O (CSI) 通信方式

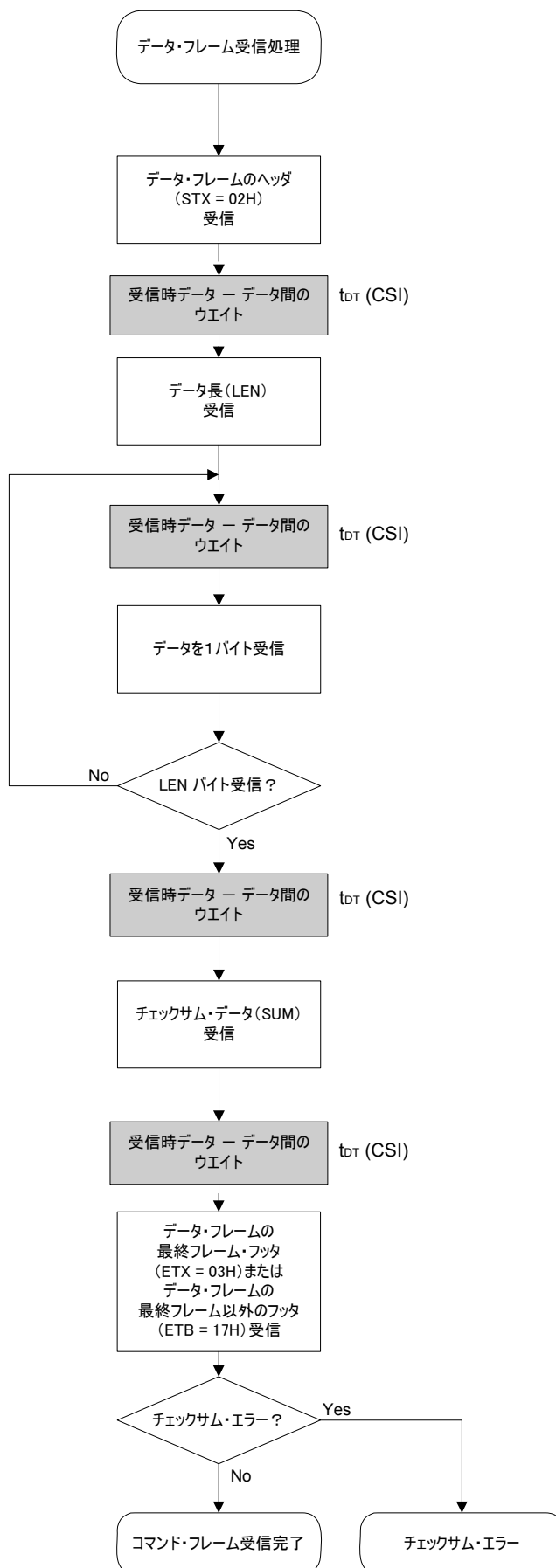
8.1 コマンド・フレーム送信処理のフロー・チャート



8.2 データ・フレーム送信処理のフロー・チャート



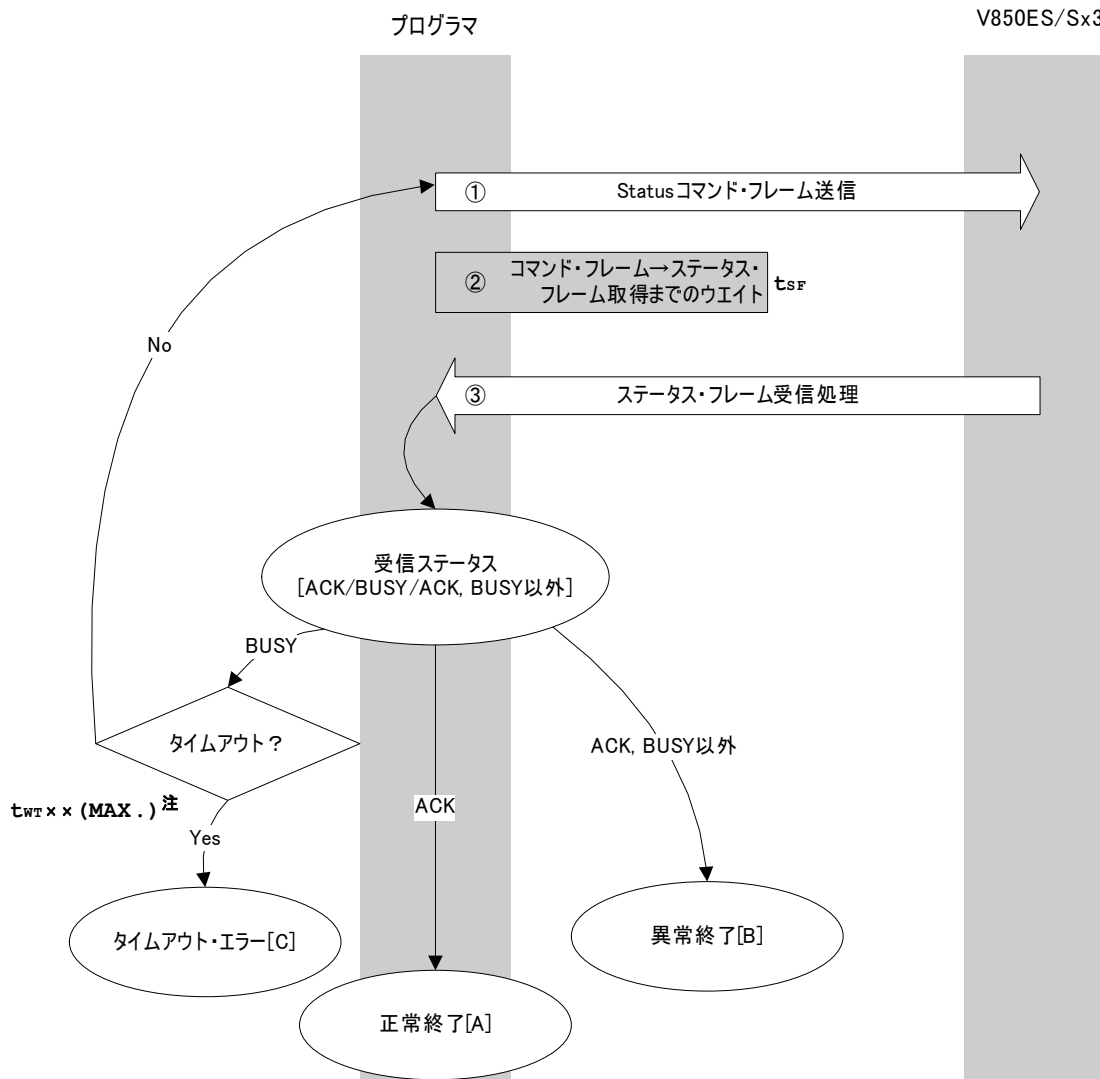
8.3 データ・フレーム受信処理のフロー・チャート



8.4 Statusコマンド

8.4.1 処理手順チャート

Statusコマンド処理手順



注 実行コマンドにより、適用スペックが異なります。

8.4.2 処理手順説明

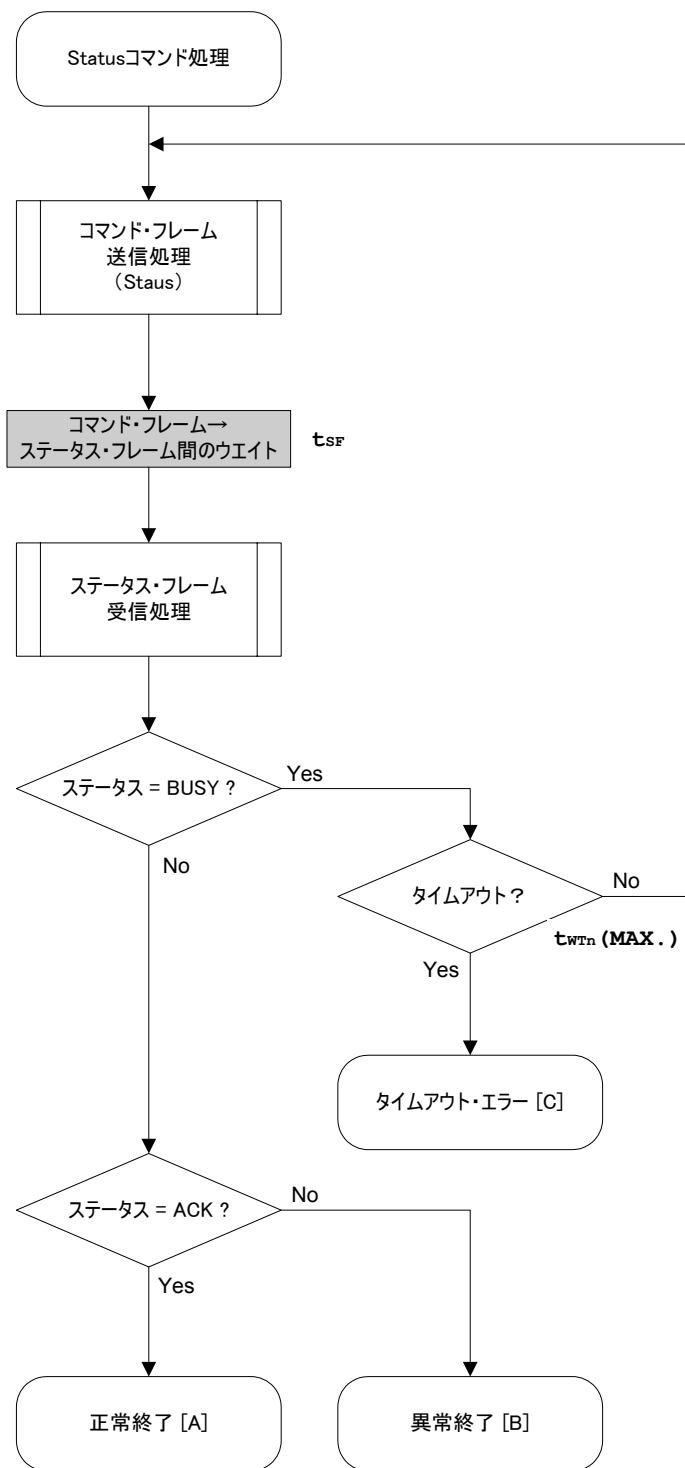
コマンド・フレーム送信処理により、**Statusコマンド**を送信します。
 コマンド送信からステータス・フレーム受信までのウェイトをします(ウェイト時間 t_{SF})。
 ステータス・コードをチェックします。

ST1 = ACKの場合 : **正常終了[A]**です。
ST1 = BUSYの場合 : タイムアウトをチェックします。タイム・アウト時間
 ($t_{WTO} (MAX.)$) は、この処理のパラメータとして与えられ
 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。

8.4.3 終了時の内容

終了内容	ステータス・コード	内 容	
正常終了 [A]	正常応答 (ACK)	06H	V850ES/Sx3 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。
	Write エラー	1CH	書き込みエラー
	MRG10 エラー	1AH	消去エラー
	MRG11 エラー	1BH	データ書き込み時の内部ペリファイ・エラー,または ブランク・チェック・エラー
	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と しました。
否定応答 (NACK)	15H	否定応答	
タイムアウト・エラー [C]	-	HS のビジーにタイムアウトが発生しました。	

8.4.4 フロー・チャート



8.4.5 サンプル・プログラム

Statusコマンド処理のサンプル・プログラムです。

```

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

        if (fl_rxddata_frm[OFS_LEN] == 2 && fl_st1 == FLST_ACK && fl_st2 == FLST_BUSY){
            if (check_flto())                      // time out ?
                return FLC_DFTO_ERR; // Yes, time-out // case [C]
            continue;
        }

        break;                                     // ACK or other error (but BUSY)
    }

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

```

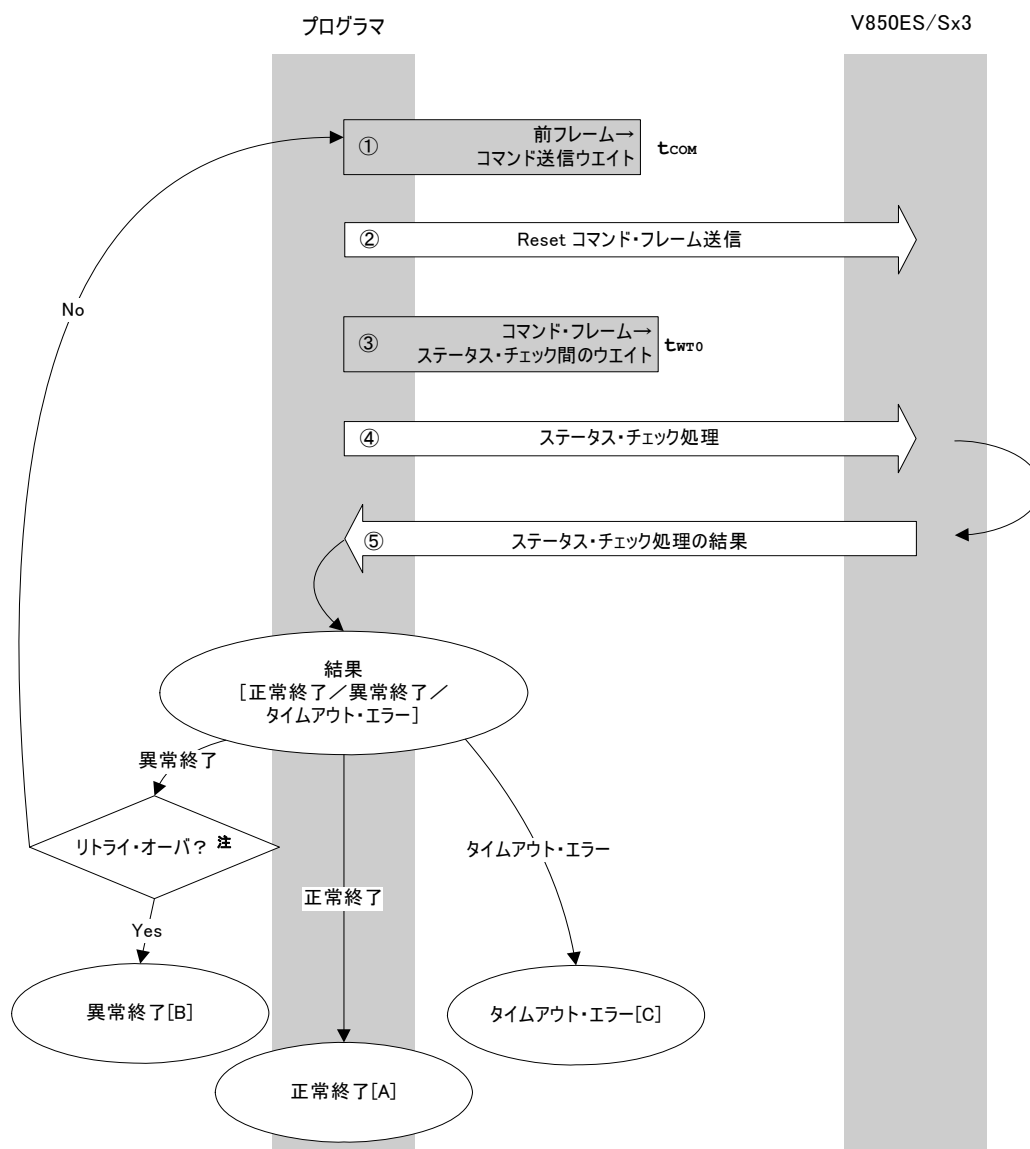
```
//      default:          return rc;    break; // case [B]
// }
    return rc;

}
```

8.5 Resetコマンド

8.5.1 処理手順チャート

Resetコマンド処理手順



注 リセット・コマンドの送信は16回 (MAX.) としてください。

8.5.2 処理手順説明

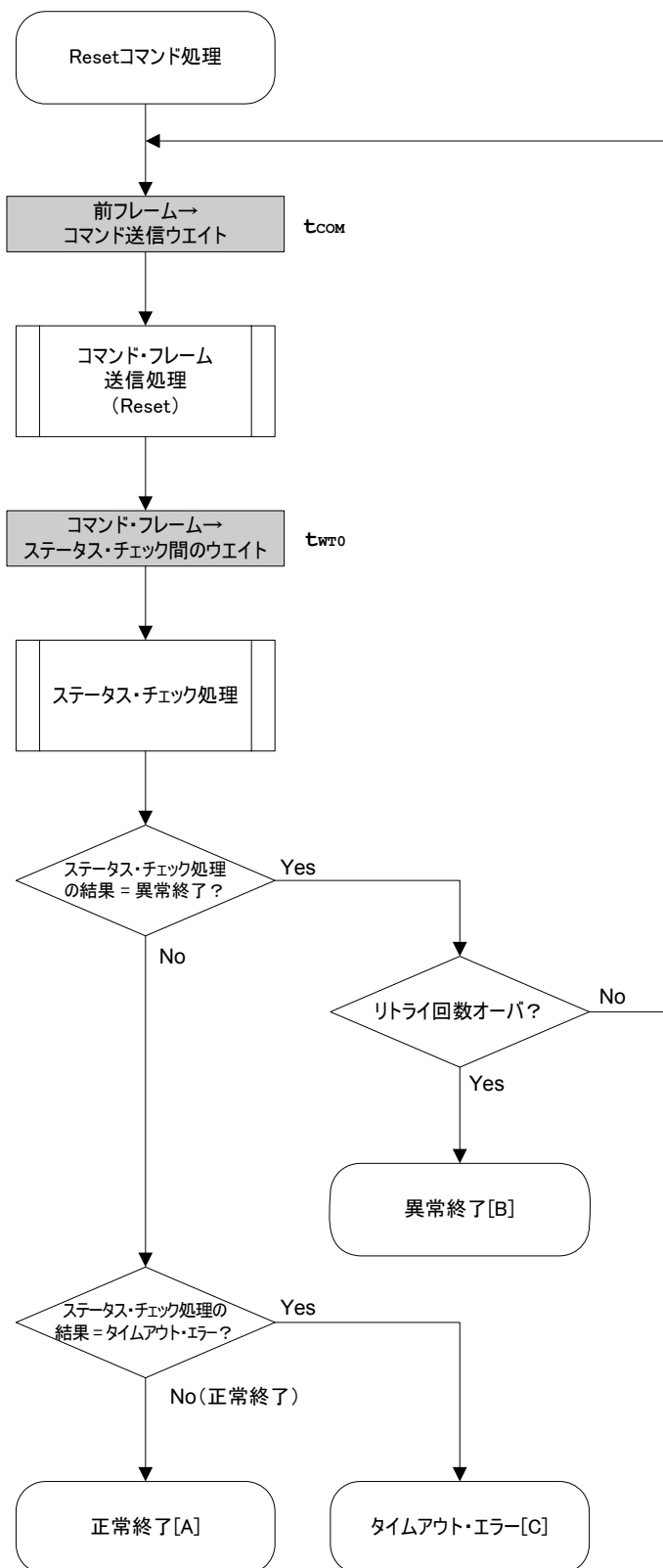
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 コマンド・フレーム送信処理により, **Resetコマンド** を送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WTR})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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。

8.5.3 終了時の内容

終了内容		ステータス・コード	内 容
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され, プログラマと V850ES/Sx3 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	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]		-	ステータス・チェック処理がタイムアウト・エラーで終了しました。

8.5.4 フロー・チャート



8.5.5 サンプル・プログラム

Resetコマンド処理のサンプル・プログラムです。

```

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

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

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

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

        fl_wait(tWT0);

        rc = fl_csi_getstatus(tWT0_MAX);            // get status

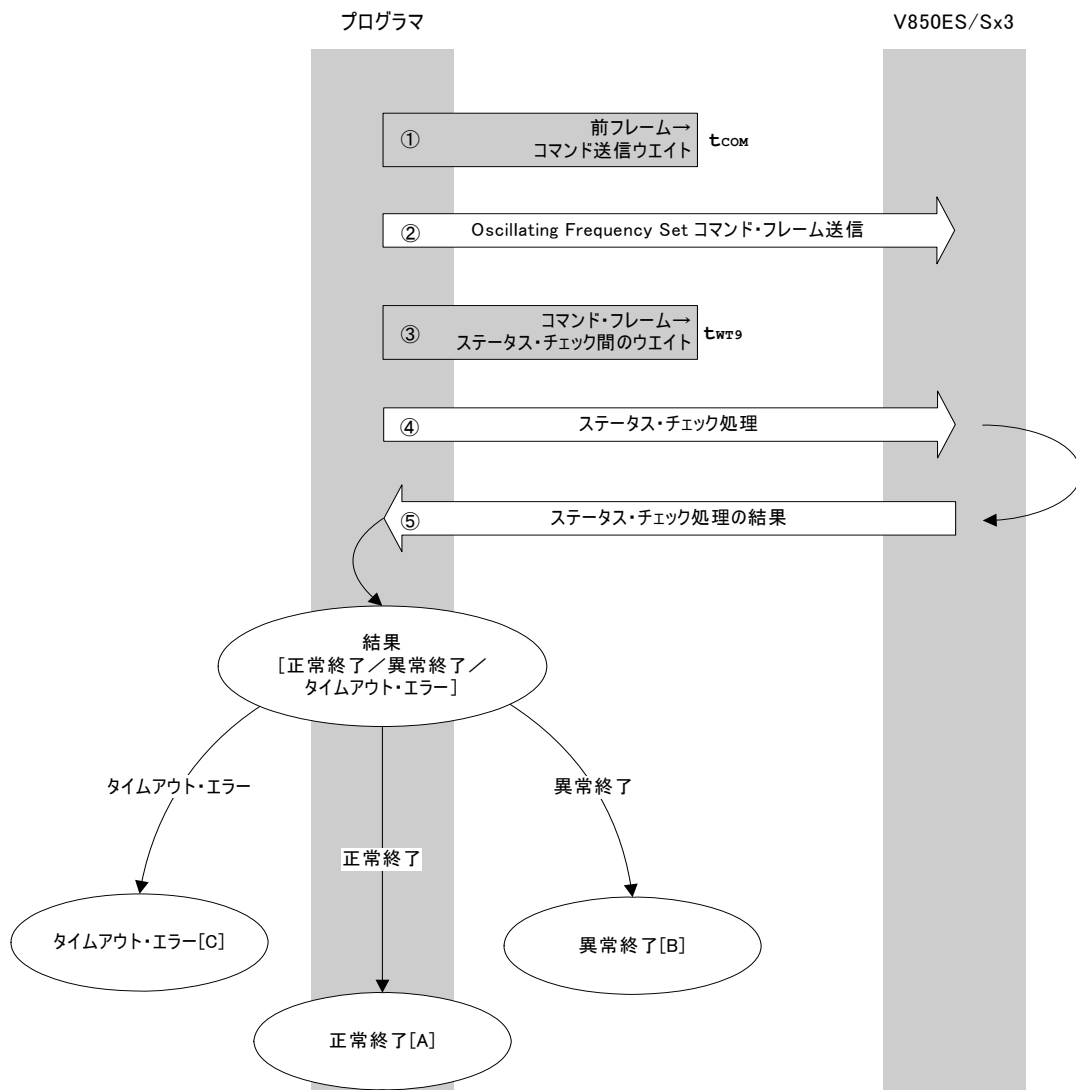
        if (rc == FLC_DFTO_ERR)                    // timeout error ?
            break;                                // yes // case [C]
        if (rc == FLC_ACK)                         // Ack ?
            break;                                // yes // case [A]
        //continue;                                // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //     case   FLC_NO_ERR:      return rc;      break; // case [A]
    //     case   FLC_DFTO_ERR:   return rc;      break; // case [C]
    //     default:               return rc;      break; // case [B]
    // }
    return rc;
}

```

8.6 Oscillating Frequency Setコマンド

8.6.1 処理手順チャート

Oscillating Frequency Setコマンド処理手順



8.6.2 処理手順説明

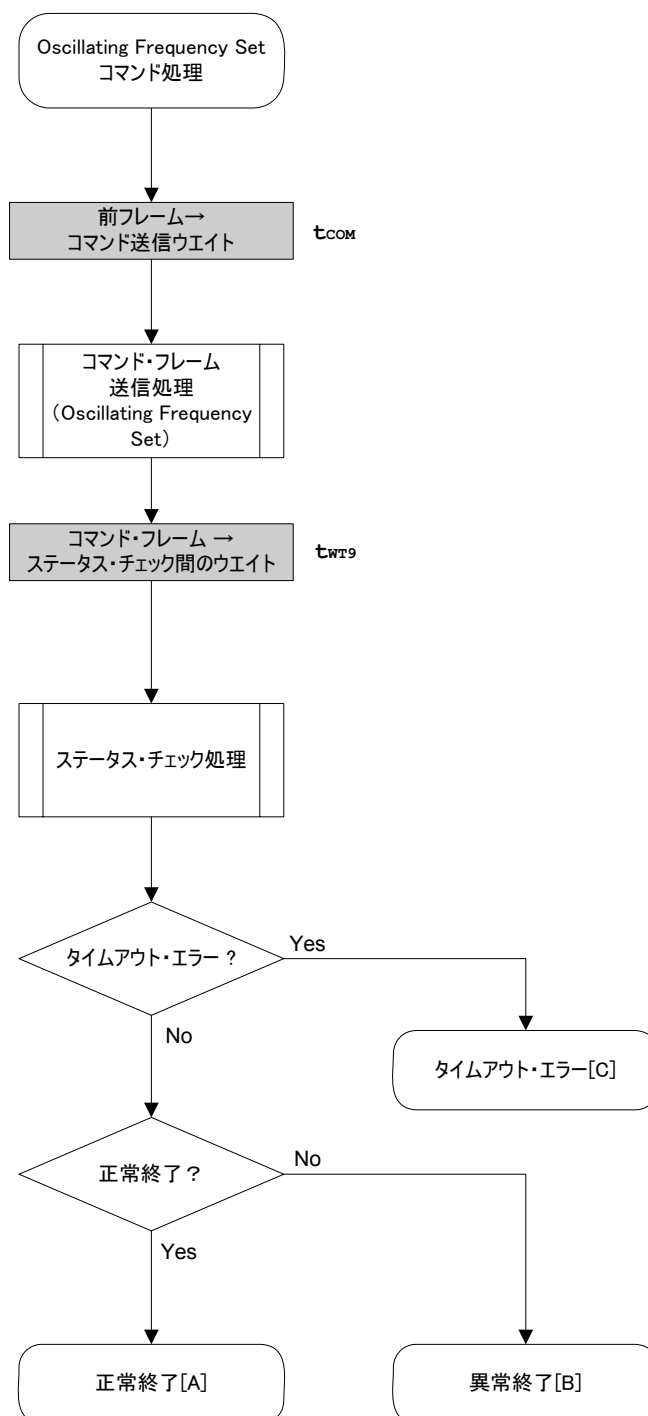
直前のフレームからコマンド送信までのウエイトをします (ウエイト時間 t_{COM})。
 コマンド・フレーム送信処理により、**Oscillating Frequency Setコマンド**を送信します。
 コマンド送信からステータス・チェック処理までのウエイトをします (ウエイト時間 t_{WTR9})。
 ステータス・チェック処理により、ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

8.6.3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、V850ES/Sx3 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 チェックサム・エラー 否定応答 (NACK)	05H	発振周波数値が範囲外です。
	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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームが受信できませんでした。

8.6.4 フロー・チャート



8.6.5 サンプル・プログラム

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

```

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

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

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

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

    fl_wait(tWT9);

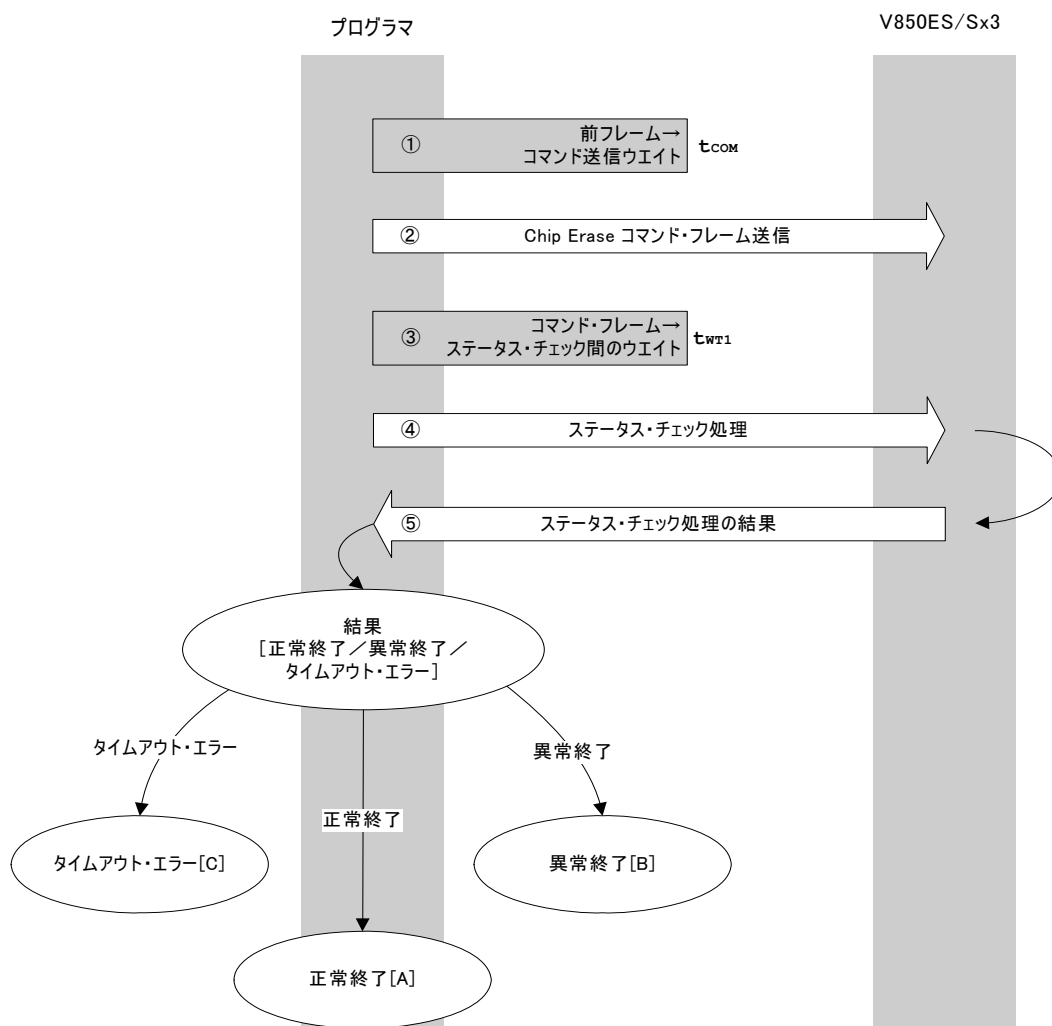
    rc = fl_csi_getstatus(tWT9_MAX); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

8.7 Chip Eraseコマンド

8.7.1 処理手順チャート

Chip Eraseコマンド処理手順



8.7.2 処理手順説明

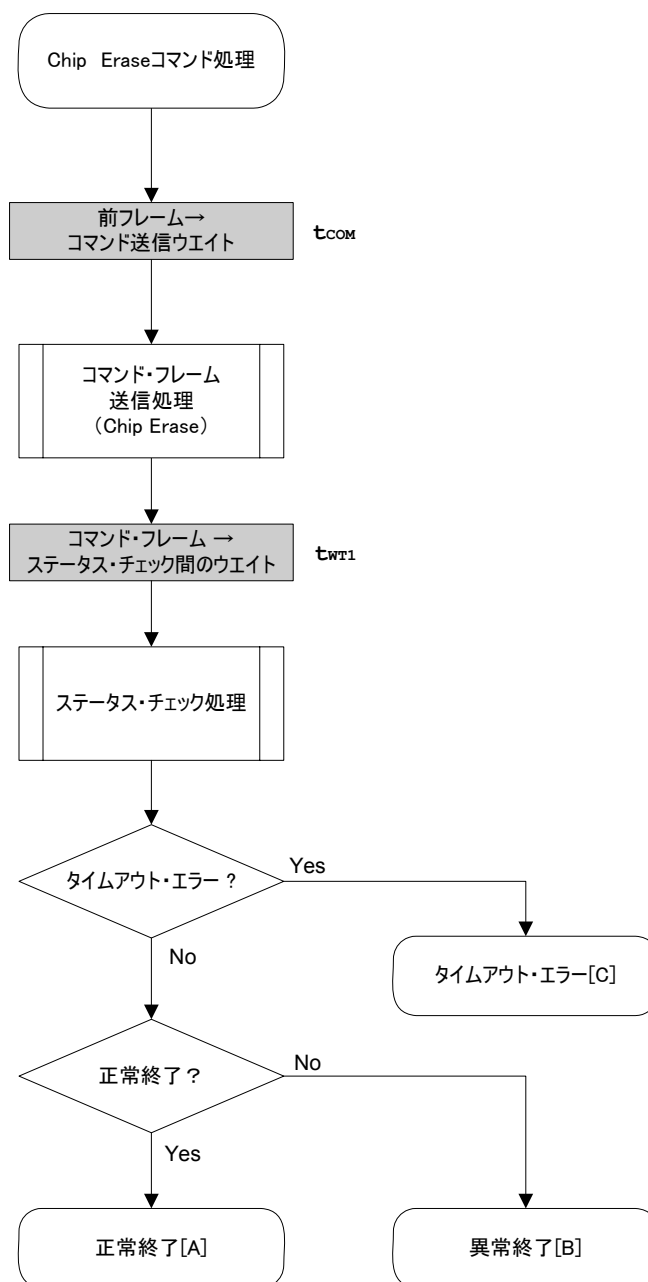
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 コマンド・フレーム送信処理により, **Chip Eraseコマンド** を送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WRT1})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて以下の処理を行います。

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。

8.7.3 終了時の内容

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

8.7.4 フロー・チャート



8.7.5 サンプル・プログラム

Chip Eraseコマンド処理のサンプル・プログラムです。

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

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

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

    fl_wait(tWT1);

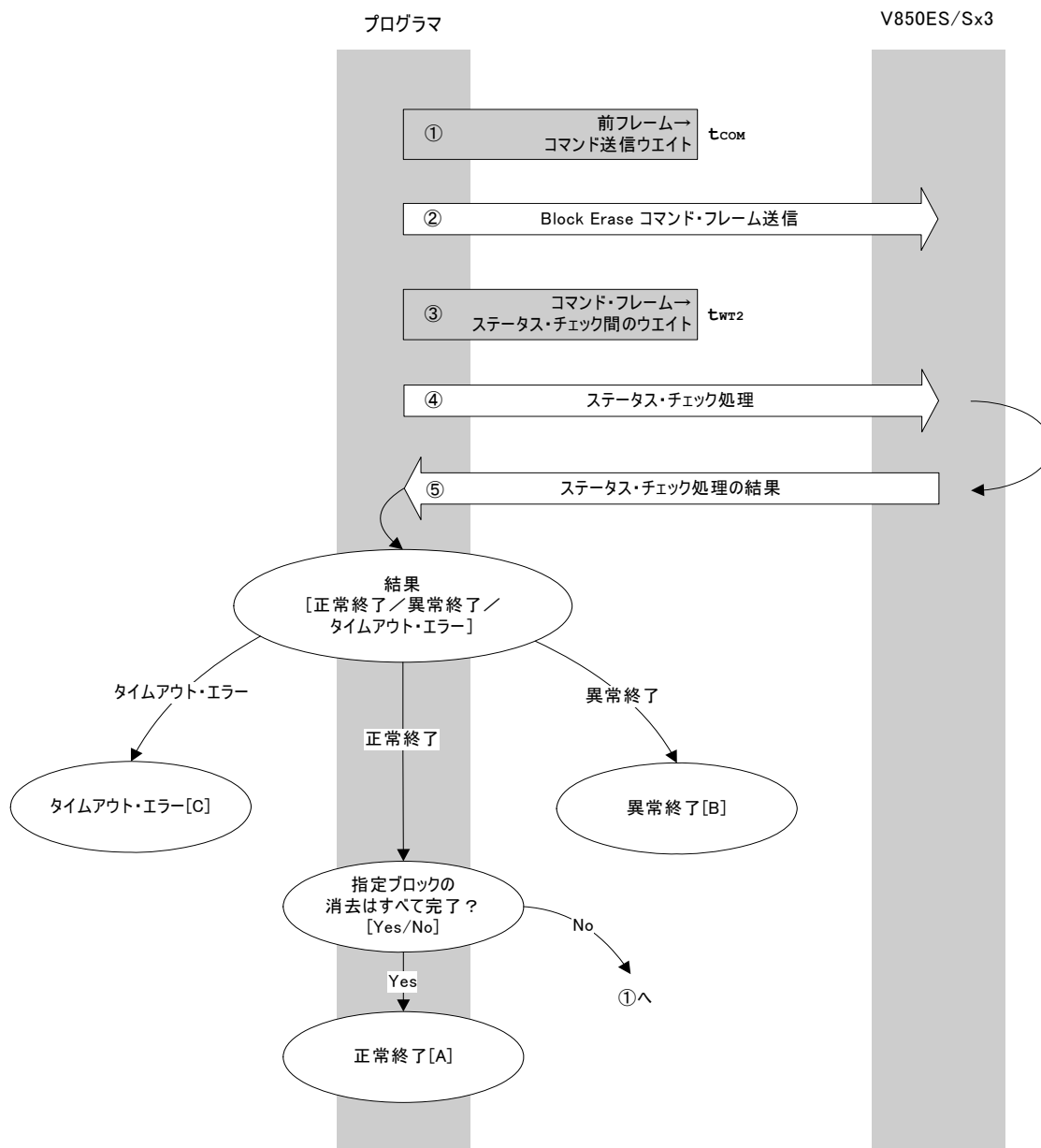
    rc = fl_csi_getstatus(tWT1_MAX);          // get status frame

    // 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;
}
```

8.8 Block Eraseコマンド

8.8.1 処理手順チャート

Block Eraseコマンド処理手順



8.8.2 処理手順説明

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_{WR2})。
 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 : 指定したブロックの消去がすべて完了していない場合は, ブロック番号を変えて より再実行します。
 指定したすべてのブロックの消去が完了した場合は, **正常終了[A]**です。

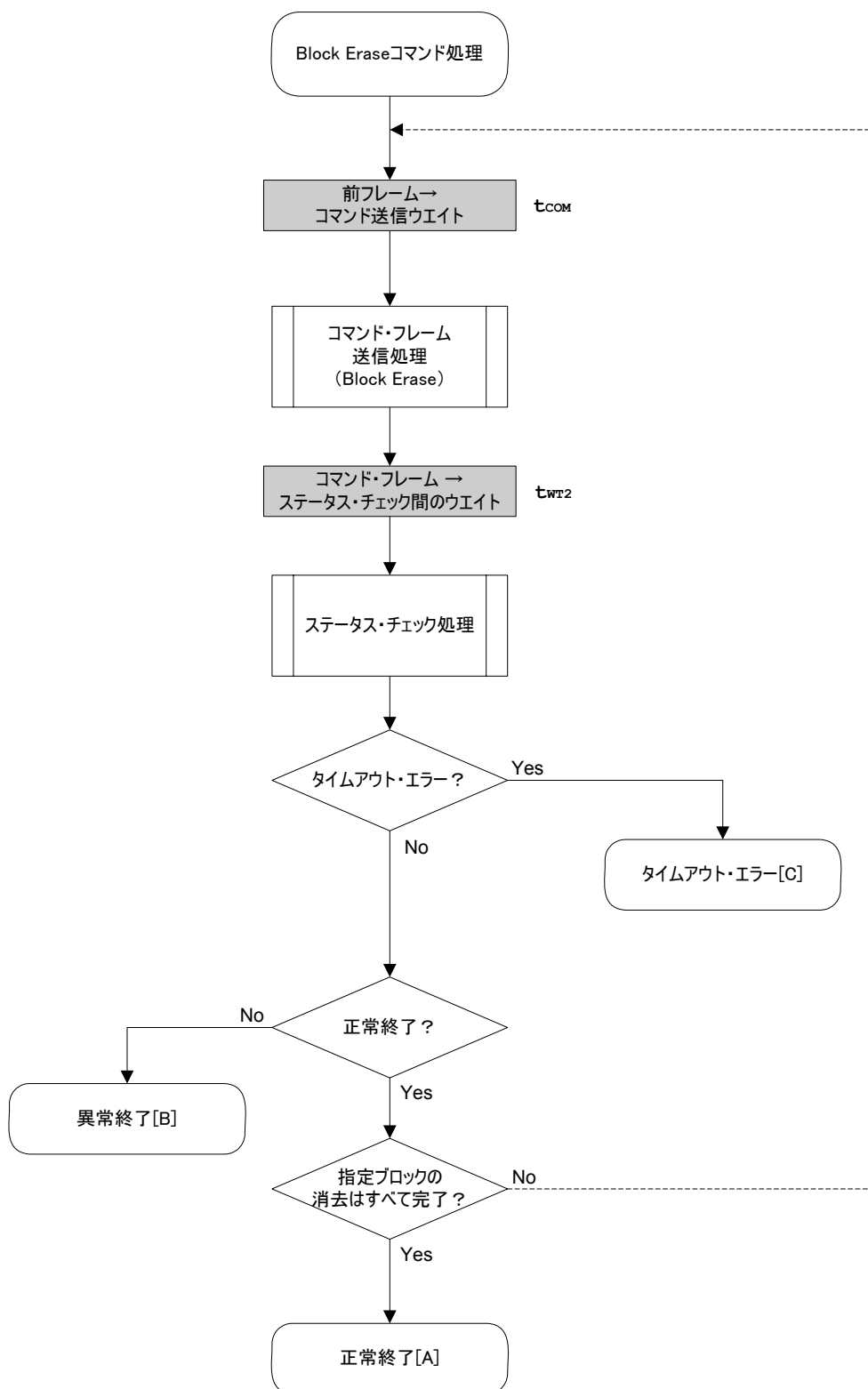
異常終了の場合 : **異常終了[B]**です。

タイムアウト・エラーの場合 : **タイムアウト・エラー[C]**です。

8.8.3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され, ブロック消去が正常に実行されたことを示します。
異常終了 [B] パラメータ・エラー	05H	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。
	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。
	10H	セキュリティ設定で ,Block Erase コマンドが禁止となる設定になっています。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレームデータが異常です (データ長 (LEN) 不正, ETX なしなど)。
MRG10 エラー	1AH	消去エラーが発生しました。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。

8.8.4 フロー・チャート



8.8.5 サンプル・プログラム

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

```

/*****
/*
/* Erase block command (CSI)
/*
/*****
/* [i] u16 sblk    ... start block number
/* [i] u16 eblk    ... end block number
/* [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);    // get tWT2 (Min)
    wt2_max = make_wt2_max(sblk, eblk); // get tWT2 (Max)

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

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

    fl_wait(wt2);

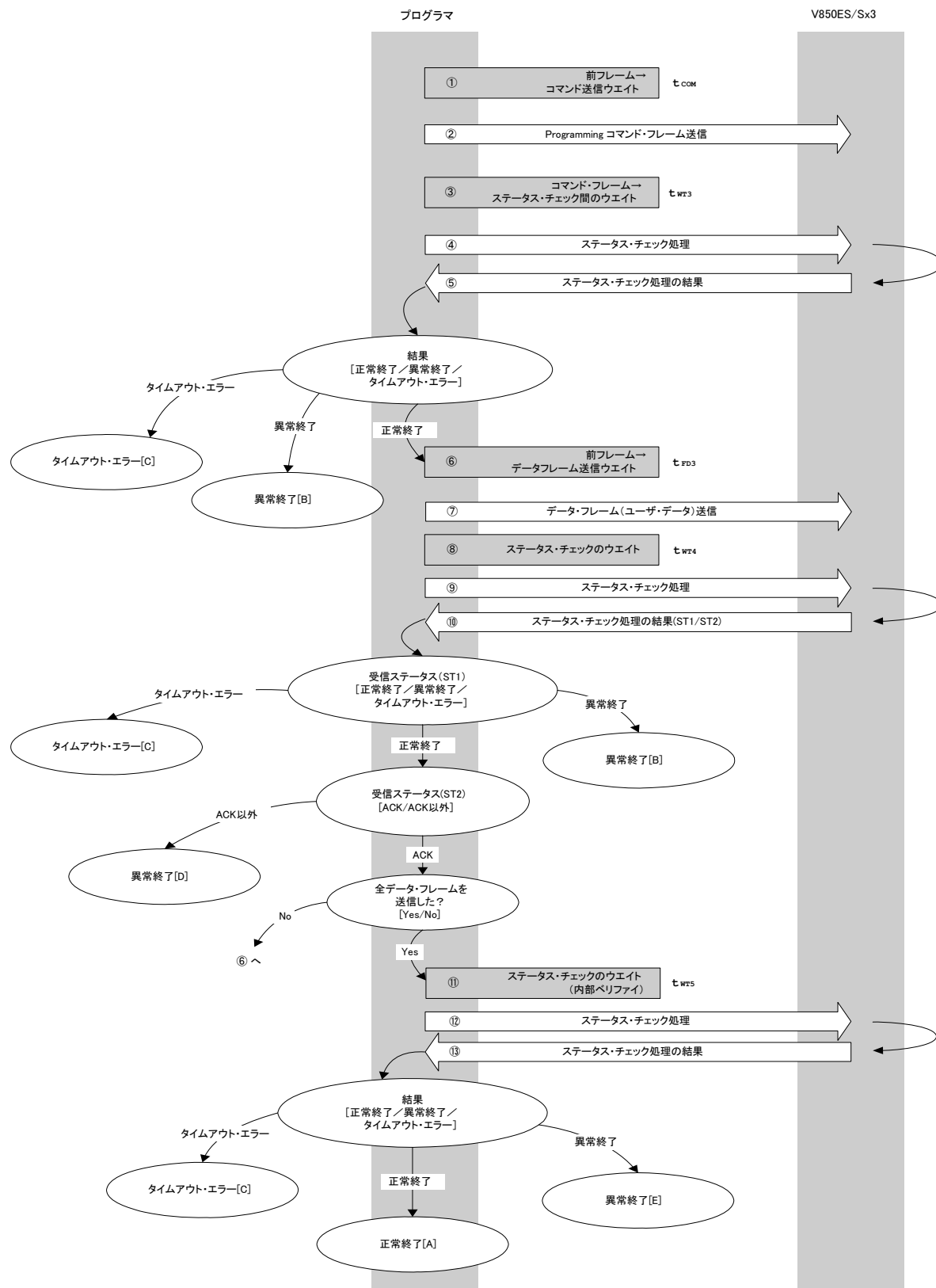
    rc = fl_csi_getstatus(wt2_max); // get status frame
    // switch(rc) {
    //
    //     case   FLC_NO_ERR:      return rc;      break; // case [A]
    //     case   FLC_DFTO_ERR:   return rc;      break; // case [C]
    //     default:                return rc;      break; // case [B]
    // }
   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

```

8.9 Programmingコマンド

8.9.1 処理手順チャート

Programmingコマンド処理手順



8.9.2 処理手順説明

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, V850ES/Sx3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。

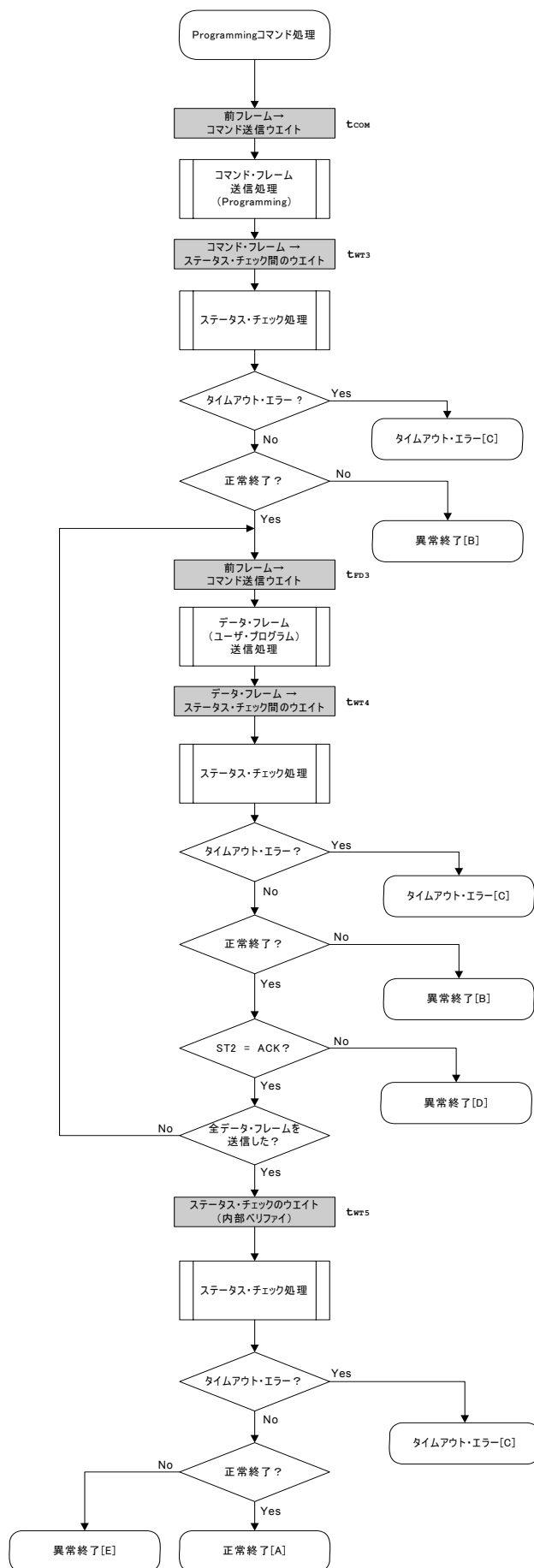
ステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WT5})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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。

8.9.3 終了時の内容

終了内容	ステータス・コード	内 容	
正常終了 [A]	正常応答 (ACK)	06H	コマンドが正常に実行され、ユーザ・データの書き込みが正常に終了したことを示します。
異常終了 [B]	パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 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、またはデータ・フレームのチェックサムが異常です。
	プロテクト・エラー	10H	セキュリティ設定で、Programming コマンドが禁止となる設定になっています。
	否定応答 (NACK)	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D]	WRITE エラー	1CH	書き込みエラーが発生しました。
異常終了 [E]	MRG11 エラー	1BH	内部ペリファイ・エラーが発生しました。

8.9.4 フロー・チャート



8.9.5 サンプル・プログラム

Programmingコマンド処理のサンプル・プログラムです。

```

/*****/
/*
/* Write command (CSI)
/*
/*****/
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****/
u16      fl_csi_write(u32 top, u32 bottom)
{
    u16      rc;
    u32      send_head, send_size;
    bool     is_end;
    u32      wt5, wt5_max;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL
    wt5      = make_wt5(get_block_num(top, bottom));
    wt5_max = make_wt5_max(get_block_num(top, bottom));

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

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

    rc = fl_csi_getstatus(tWT3_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_DFTO_ERR:    return rc; break; // case [C]
        default:                  return rc; break; // case [B]
    }

    /*****/
    /*      send user data
    /*****/
    send_head = top;

    while(1){

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

        memcpy(fl_txdata_frm, rom_buf+send_head, send_size);

```



```

// set data frame payload
send_head += send_size;

fl_wait(tFD3); // wait before sending data frame
put_dfrm_csi(send_size, fl_txdata_frm, is_end);
// send data frame (user data)
fl_wait(tWT4); // wait

rc = fl_csi_getstatus(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(wt5); // wait

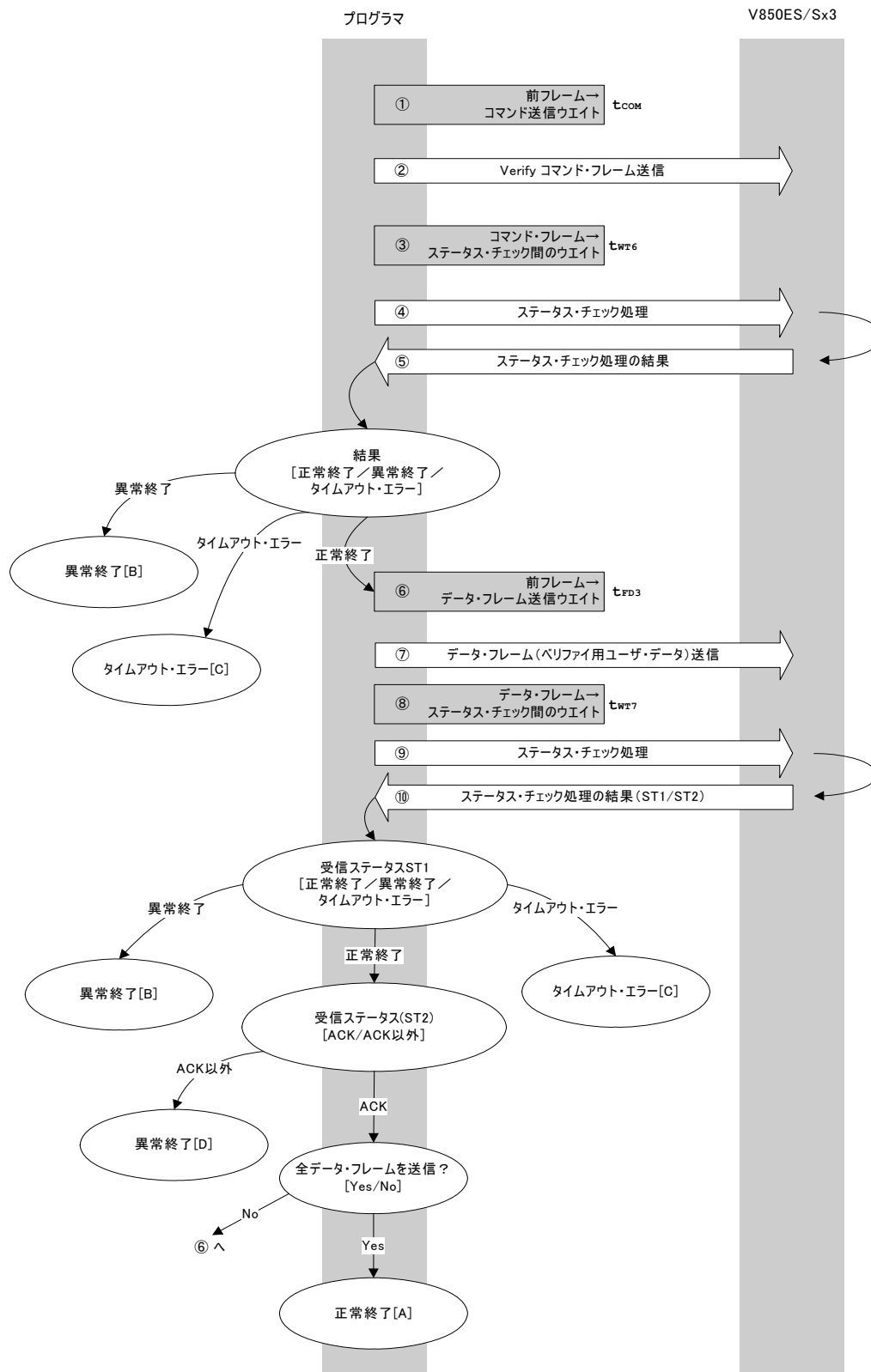
rc = fl_csi_getstatus(wt5_max); // get status frame
// switch(rc) {
// case FLC_NO_ERR: return rc; break; // case [A]
// case FLC_DFTO_ERR: return rc; break; // case [C]
// default: return rc; break; // case [E]
// }
return rc;
}

```

8. 10 Verifyコマンド

8. 10. 1 処理手順チャート

Verifyコマンド処理手順



8. 10. 2 処理手順説明

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。

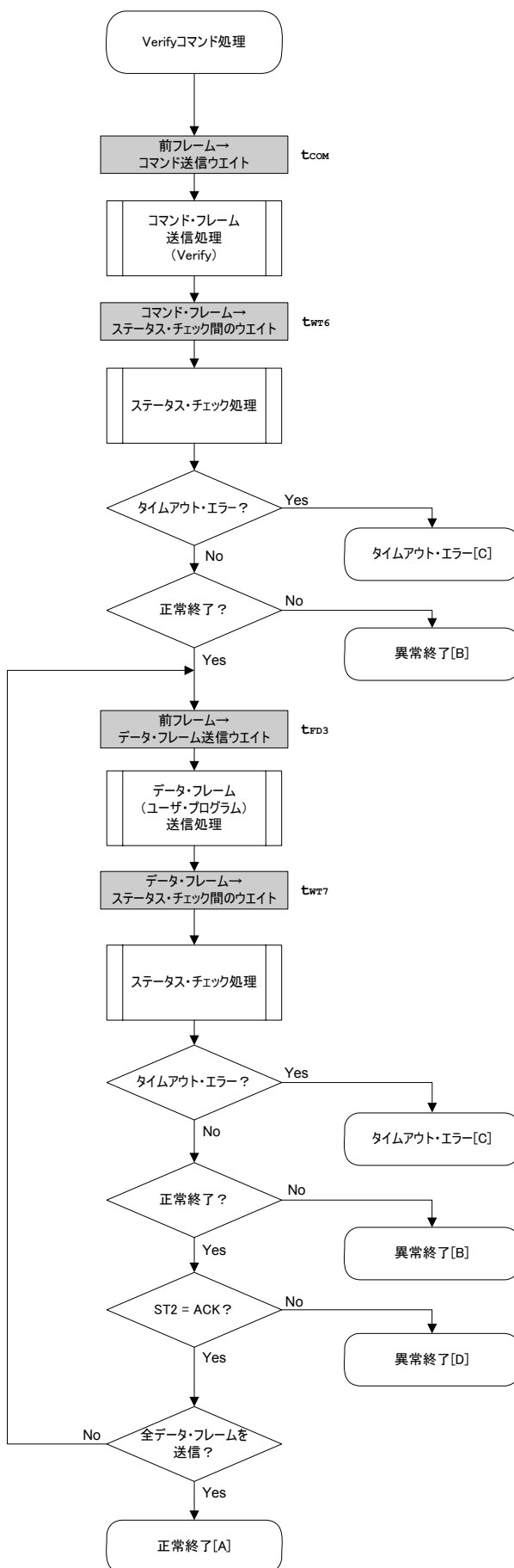
直前のフレームからデータ・フレーム送信までのウェイトをします (ウェイト時間 t_{FD3})。
 データ・フレーム送信処理により, ペリファイ用のユーザ・データを送信します。
 データ・フレーム送信からステータス・チェック処理までのウェイトをします
 (ウェイト時間 t_{WT7})。
 ステータス・チェック処理にてステータス・フレームを取得します。
 ステータス・チェック処理の結果 (受信ステータス (ST1/ST2)) に応じて次の処理を行います
 (処理手順チャートやフロー・チャートも参照してください)。

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。

8. 10. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され,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。
	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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D] 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。

8. 10. 4 フロー・チャート



8. 10. 5 サンプル・プログラム

Verifyコマンド処理のサンプル・プログラムです。

```

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

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

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

    rc = fl_csi_getstatus(tWT6_MAX); // get status frame
    switch(rc) {
        case    FLC_NO_ERR:                break; // continue
        // case    FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

    /*****/
    /*      send user data
    /*****/
    send_head = top;

    while(1){

        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, buf+send_head, send_size);
            // set data frame payload
        send_head += send_size;

```

```
fl_wait(tFD3); // wait before sending data frame
put_dfrm_csi(send_size, fl_txdata_frm, is_end); // send data frame
fl_wait(tWT7); // wait

rc = fl_csi_getstatus(tWT7_MAX); // get status frame
switch(rc) {
    case FLC_NO_ERR: break; // continue
//    case FLC_DFTO_ERR: return rc; break; // case [C]
    default: return rc; break; // case [B]
}
if (fl_st2 != FLST_ACK) { // ST2 = ACK ?
    rc = decode_status(fl_st2); // No
    return rc; // case [D]
}

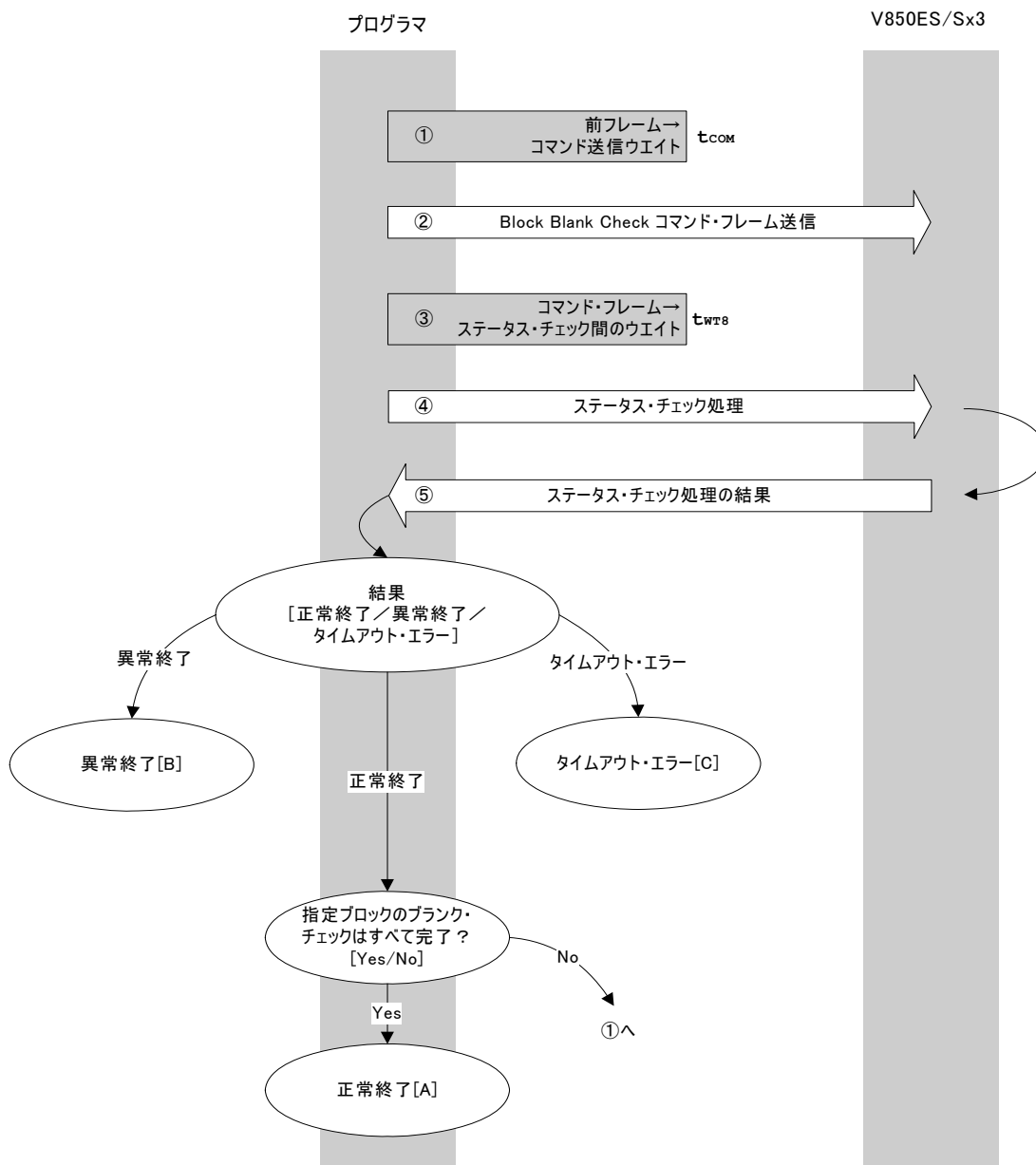
if (is_end) // send all user data ?
    break; // yes
//continue;

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

8. 11 Block Blank Checkコマンド

8. 11. 1 処理手順チャート

Block Blank Checkコマンド処理手順



8. 11. 2 処理手順説明

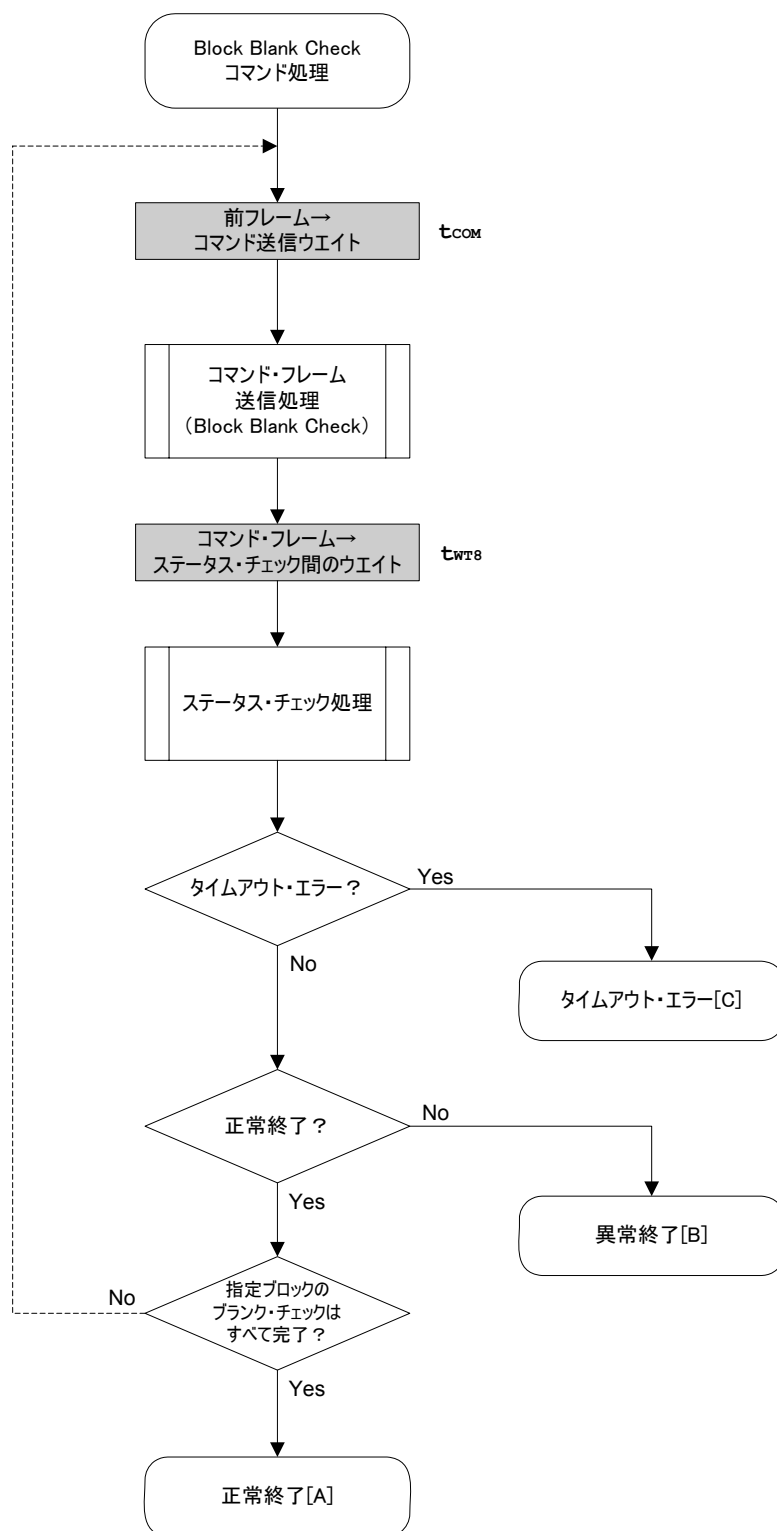
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{COM})。
 コマンド・フレーム送信処理により, **Block Blank Checkコマンド** を送信します。
 コマンド送信からステータス・チェック処理までのウェイトをします (ウェイト時間 t_{WTS})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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

8. 11. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され, 指定したブロックすべてがブランクであることを示します。
異常終了 [B] パラメータ・エラー	05H	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。
	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。
	15H	・ 処理中にステータス・コマンド以外を受信しました。 ・ コマンド・フレーム・データが異常の場合 (データ長 (LEN) 不正, ETX なしなど)
	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。

8. 11. 4 フロー・チャート



8. 11. 5 サンプル・プログラム

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

```

/*****/
/*
/* Block blank check command (CSI)
/*
/*****/
/* [i] u16 sblk    ... start block number
/* [i] u16 eblk    ... end block number
/* [r] u16         ... error code
/*****/
u16    fl_csi_blk_blank_chk(u16 sblk, u16 eblk)
{
    u16    rc;
    u32    wt8, wt8_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

    wt8    = make_wt8(sblk, eblk);    // get tWT8 (Min)
    wt8_max = make_wt8_max(sblk, eblk); // get tWT8 (Max)

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

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

    fl_wait(wt8);

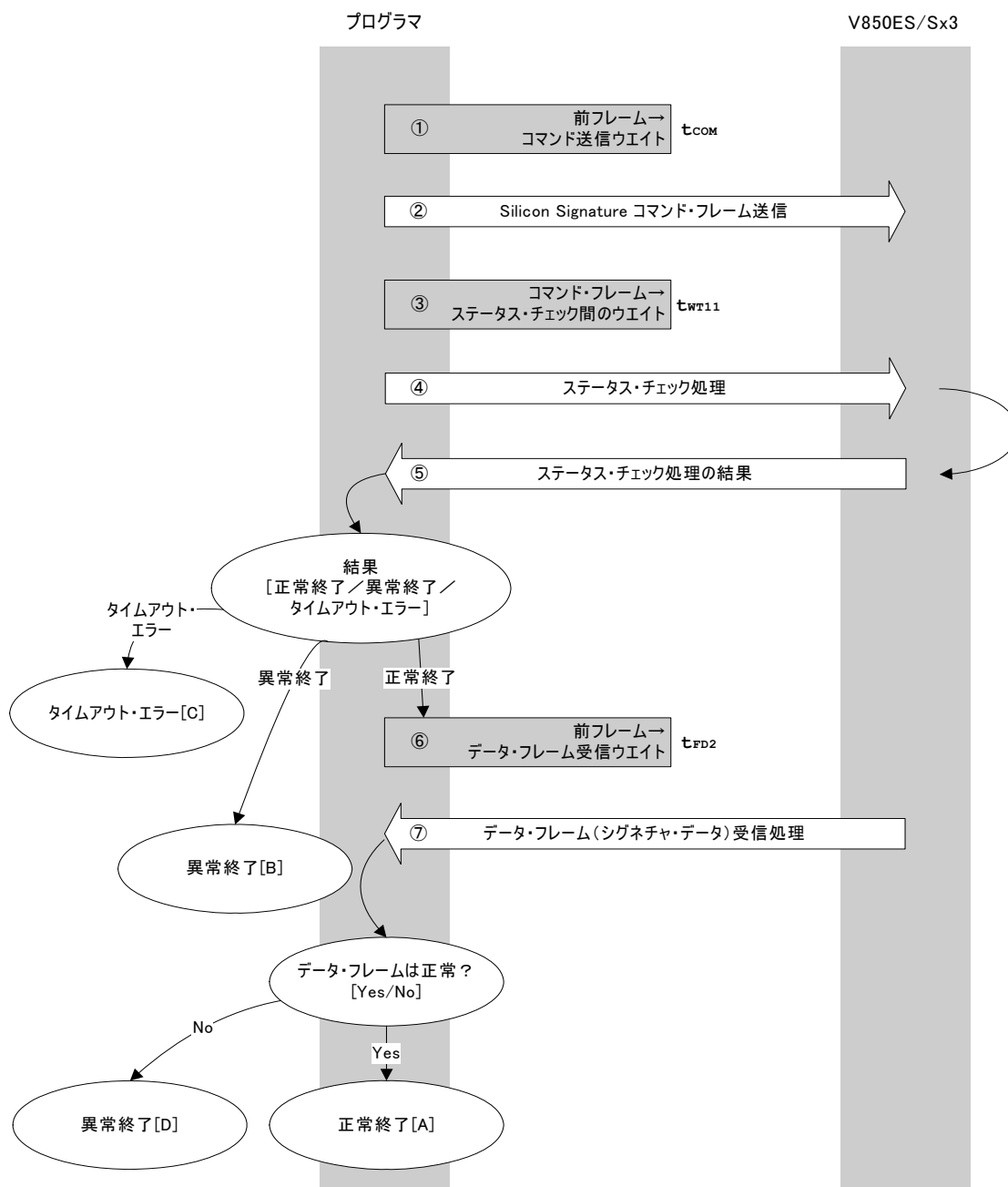
    rc = fl_csi_getstatus(wt8_max); // get status frame
    // switch(rc) {
    //
    //     case    FLC_NO_ERR:    return rc;    break; // case [A]
    //     case    FLC_DFTO_ERR: return rc;    break; // case [C]
    //     default:              return rc;    break; // case [B]
    // }
    return rc;
}

```

8. 12 Silicon Signatureコマンド

8. 12. 1 処理手順チャート

Silicon Signatureコマンド処理手順



8. 12. 2 処理手順説明

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_{WR11})。
 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。

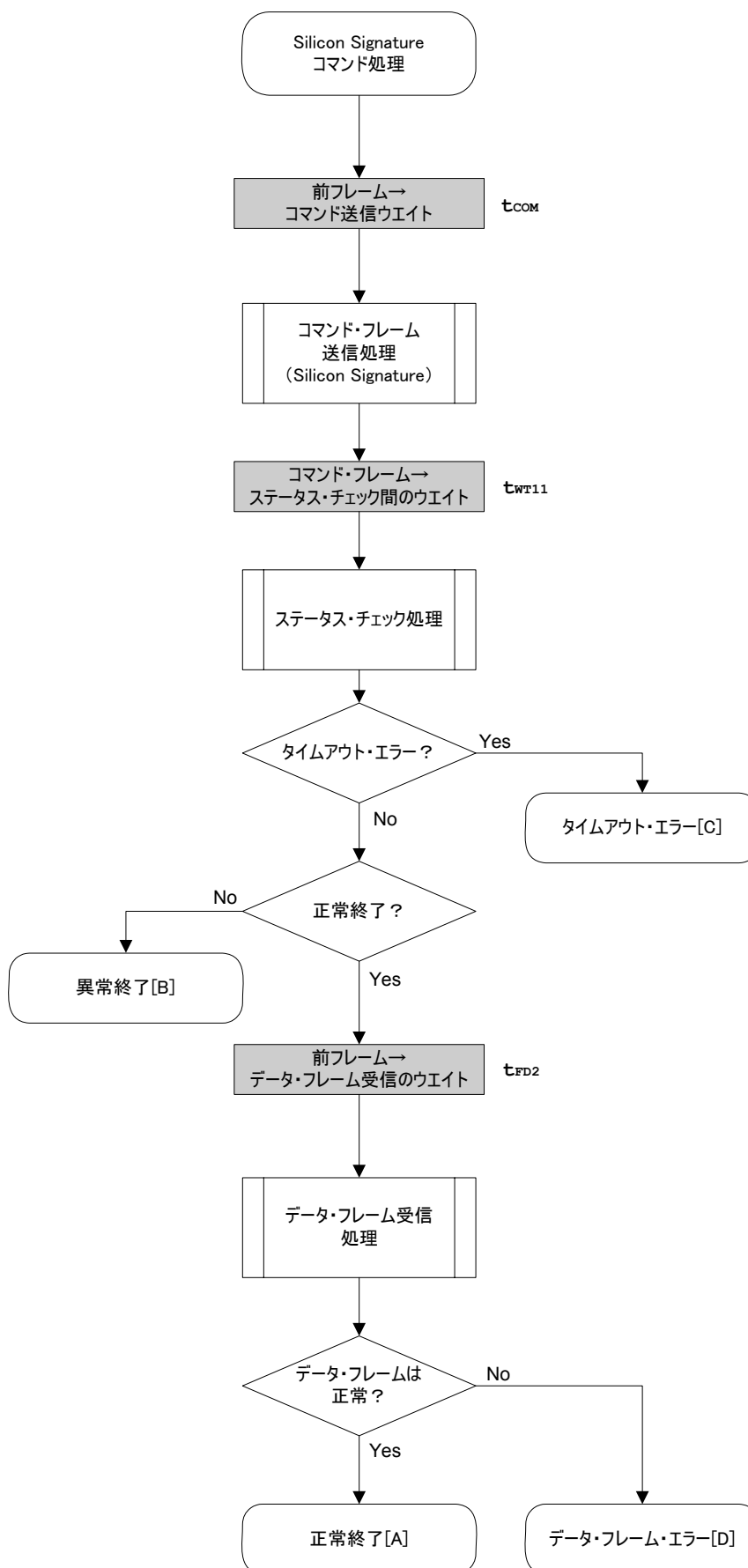
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{RD2})。
 受信したデータ・フレーム (シリコン・シグネチャ・データ) をチェックします。

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

8. 12. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され, シリコン・シグネチャを取得できたことを示します。
異常終了 [B] チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
データ・フレーム・エラー [D]	-	シリコン・シグネチャ・データとして受信したデータ・フレームのチェックサムが異常です。

8. 12. 4 フロー・チャート



8. 12. 5 サンプル・プログラム

Silicon Signatureコマンド処理のサンプル・プログラムです。

```

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

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

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

    fl_wait(tWT11);

    rc = fl_csi_getstatus(tWT11_MAX);    // get status frame
    switch(rc) {
        case    FLC_NO_ERR:              break; // continue
        // case    FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:              return rc;    break; // case [B]
    }

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

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

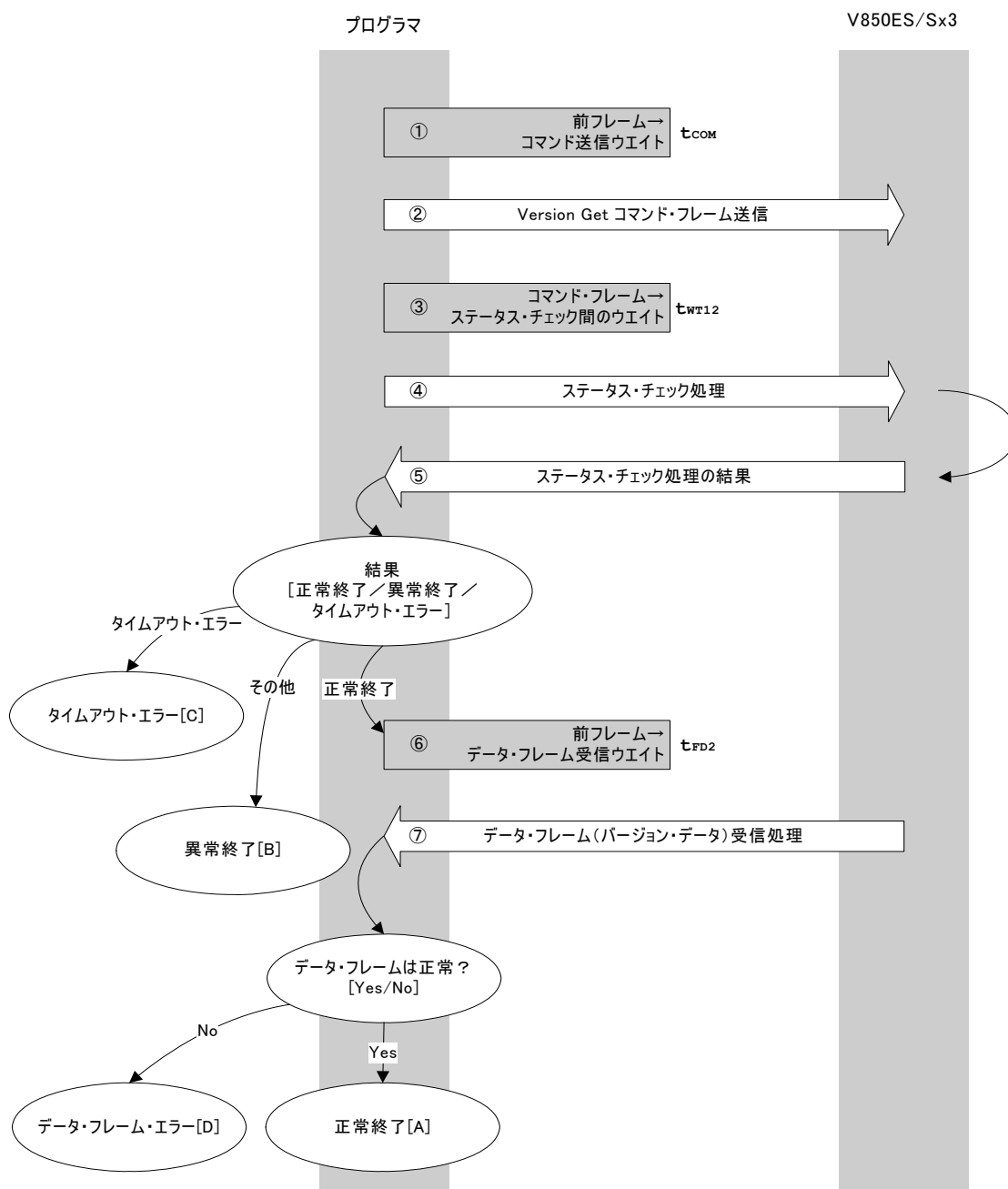
    if (rc){                      // if no error,
        return rc;                // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                // copy Signature data
    return rc;                    // case [A]
}

```

8. 13 Version Getコマンド

8. 13. 1 処理手順チャート

Version Getコマンド処理手順



8. 13. 2 処理手順説明

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_{wr12})。
 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。

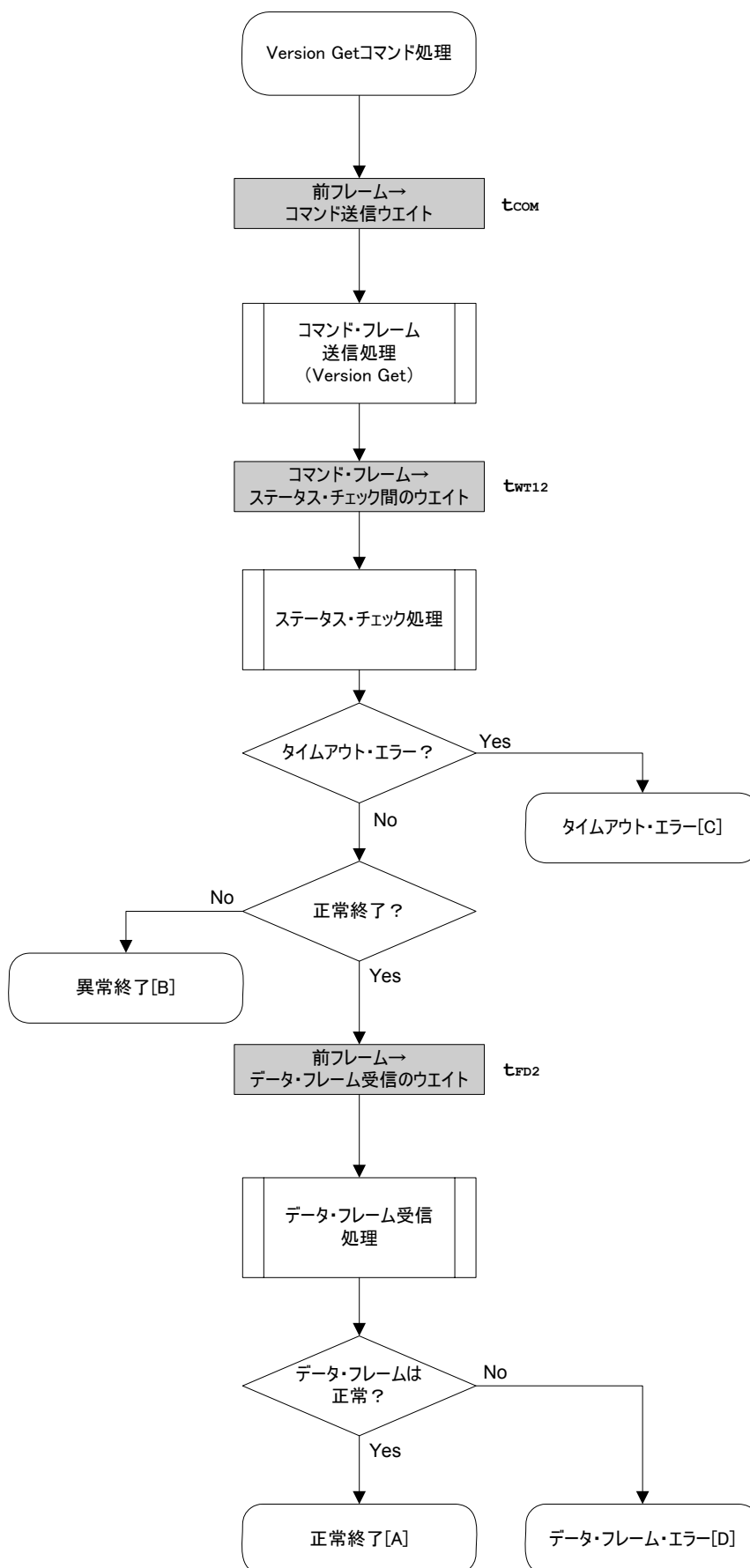
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{rd2})。
 受信したデータ・フレーム (バージョン・データ) をチェックします。

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。

8. 13. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され, バージョン・データを取得できたことを示します。
異常終了 [B] チェックサム・エラー	07H	送信したコマンド・フレームのチェックサムが異常です。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
データ・フレーム・エラー [D]	-	バージョン・データとして受信したデータ・フレームのチェックサムが異常です。

8. 13. 4 フロー・チャート



8. 13. 5 サンプル・プログラム

Version Getコマンド処理のサンプル・プログラムです。

```

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

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

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

    fl_wait(tWT12);

    rc = fl_csi_getstatus(tWT12_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]
    }

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

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

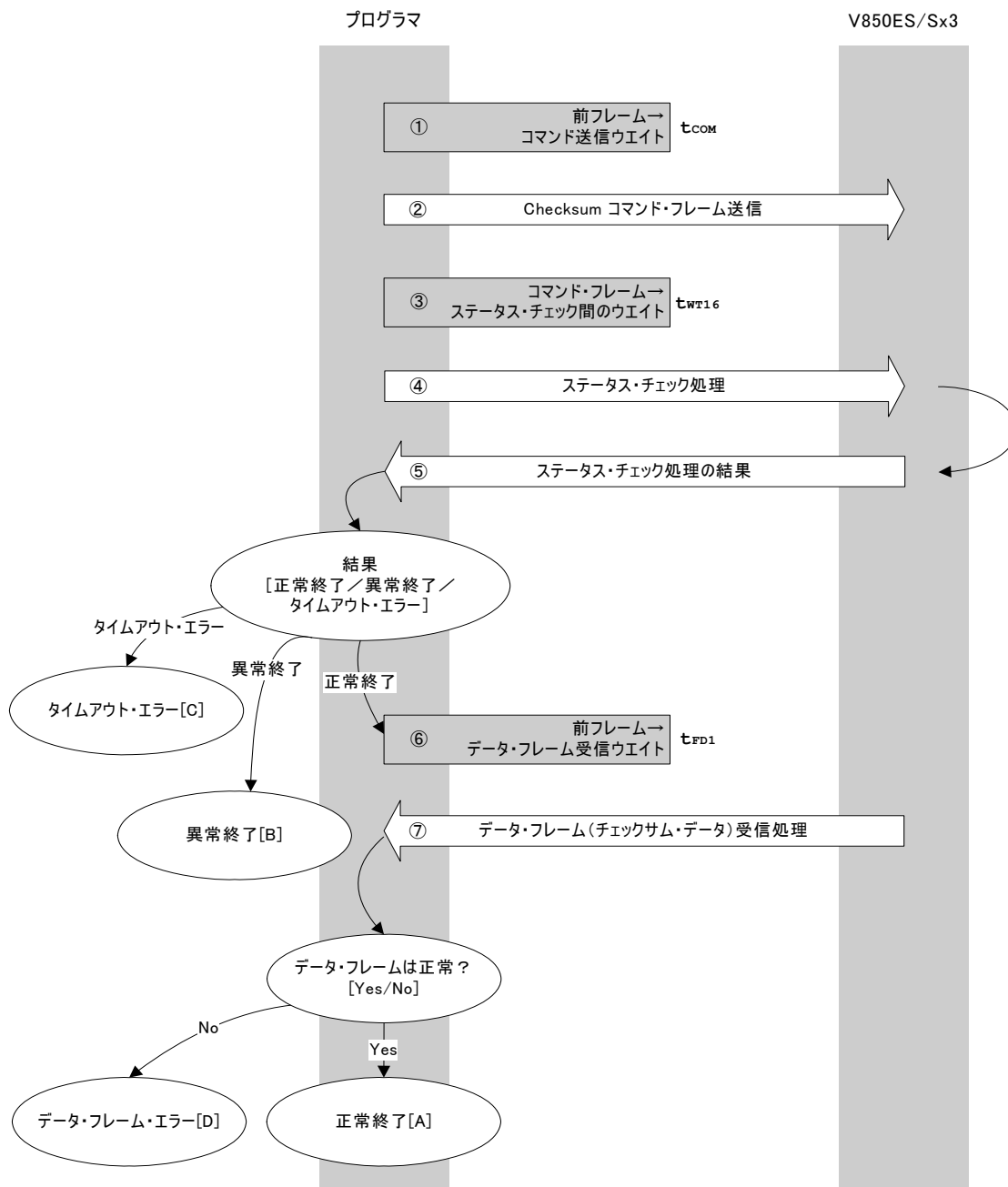
    if (rc){                                // if no error,
        return rc;                            // case [D]
    }
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                                // case [A]
}

```

8. 14 Checksumコマンド

8. 14. 1 処理手順チャート

Checksumコマンド処理手順



8. 14. 2 処理手順説明

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。

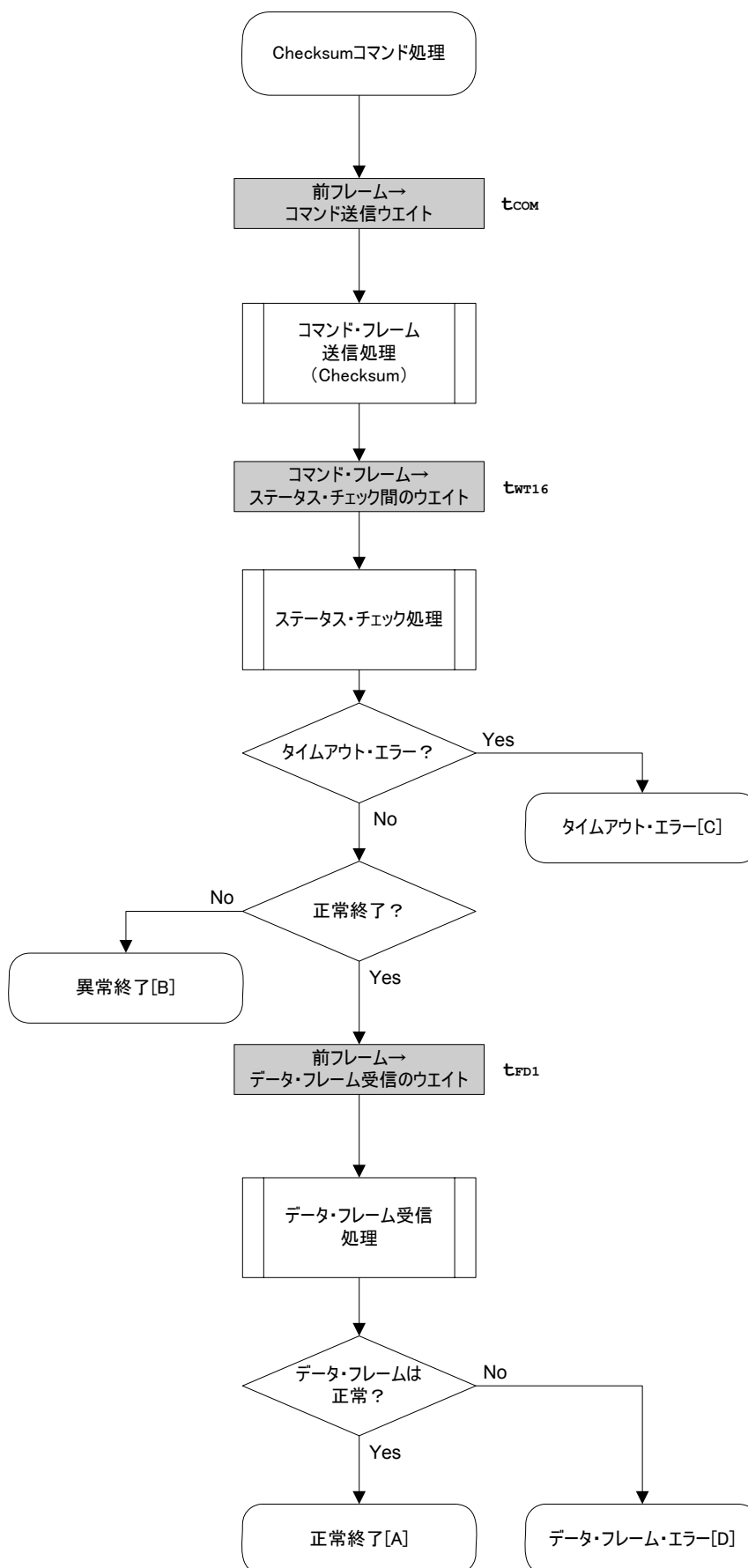
直前のフレームからコマンド送信までのウェイトをします (ウェイト時間 t_{RD1})。
 受信したデータ・フレーム (チェックサム・データ) をチェックします。

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。

8. 14. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、バージョン・データを取得できたことを示します。
異常終了 [B] パラメータ・エラー	05H	開始 / 終了アドレスがブロックの開始 / 終了アドレス以外で指定されています。
	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。
	15H	・処理中にステータス・コマンド以外を受信しました。 ・コマンド・フレーム・データが異常です (データ長 (LEN) 不正 , ETX なしなど) 。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
データ・フレーム・エラー [D]	-	チェックサム・データとして受信したデータ・フレームのチェックサムが異常です。

8. 14. 4 フロー・チャート



8. 14. 5 サンプル・プログラム

Checksumコマンド処理のサンプル・プログラムです。

```

/*****/
/*
/* Get checksum command (CSI)
/*
/*****/
/* [i] u16 *sum    ... pointer to checksum save area
/* [i] u32 top     ... start address
/* [i] u32 bottom  ... end address
/* [r] u16        ... error code
/*****/
u16      fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;
    u32    fd1;

    /*****/
    /*      set params
    /*****/
    // set params
    set_range_prm(fl_cmd_prm, top, bottom);          // set SAH/SAM/SAL, EAH/EAM/EAL
    fd1 = get_fd1(get_block_num(top, bottom));      // get tFD1 (Min)

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

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

    fl_wait(tWT16);

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

    /*****/
    /*      get data frame (Checksum data)
    /*****/
    fl_wait(fd1);

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

    if (rc){
        return rc;                                  // if error,
        // case [D]
    }

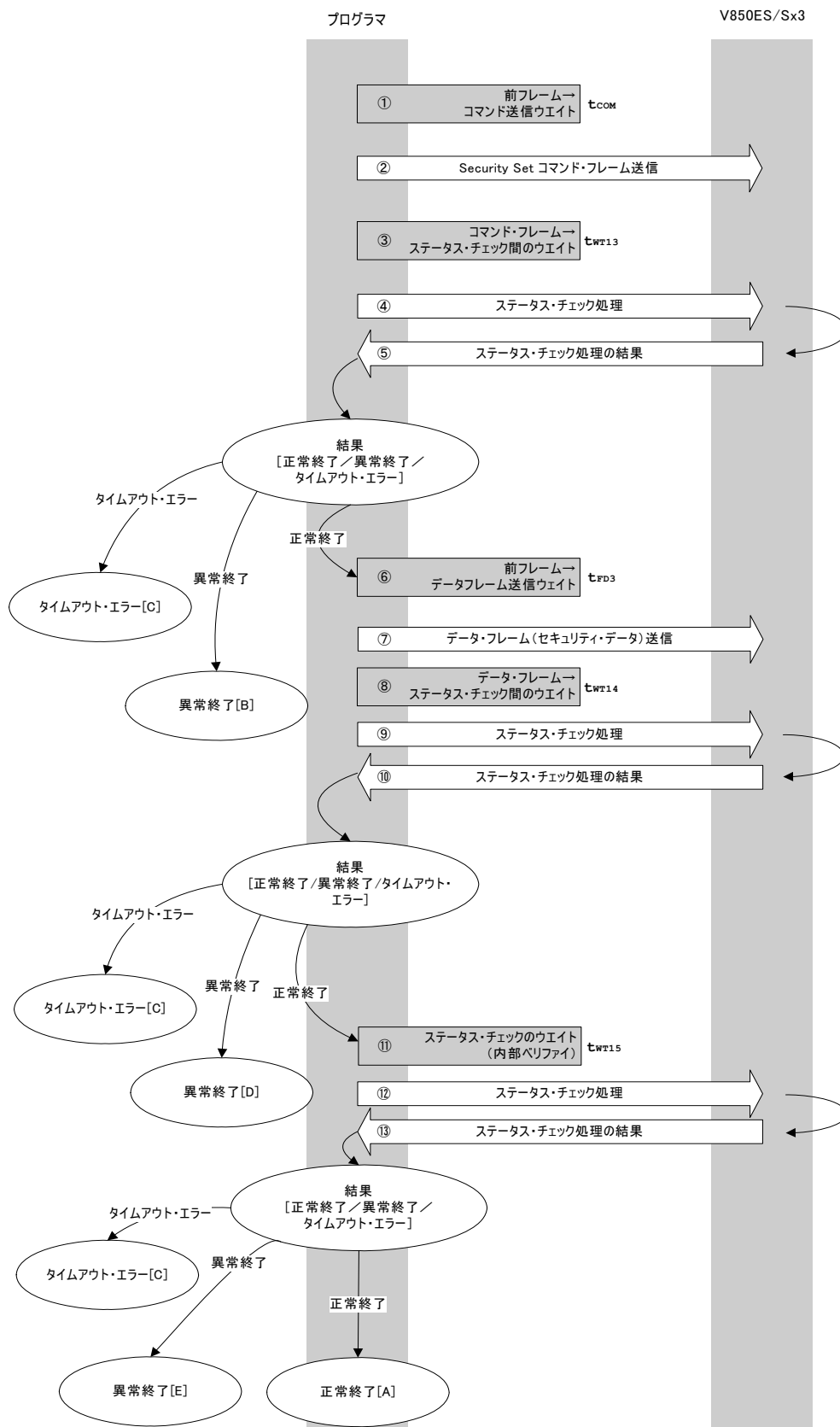
    *sum = (fl_rxddata_frm[OFS_STA_PLD] << 8) + fl_rxddata_frm[OFS_STA_PLD+1];
        // set SUM data
    return rc;                                       // case [A]
}

```

8. 15 Security Setコマンド

8. 15. 1 処理手順チャート

Security Setコマンド処理手順



8. 15. 2 処理手順説明

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, 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エイトをします (ウエイト時間 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エイトをします (ウエイト時間 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エイトをします (ウエイト時間 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。

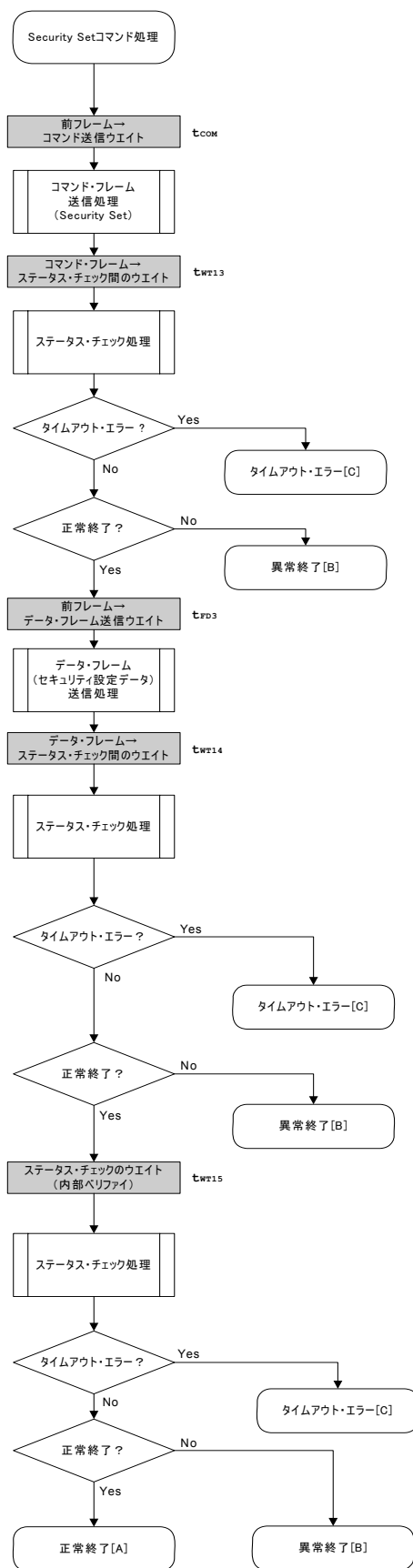
ステータス取得 (内部ベリファイ完了) までのウエイトをします (ウエイト時間 t_{WT15})。
 ステータス・チェック処理により, ステータス・フレームを取得します。
 ステータス・チェック処理の結果に応じて次の処理を行います。

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。

8. 15. 3 終了時の内容

終了内容	ステータス・コード	内 容
正常終了 [A] 正常応答 (ACK)	06H	コマンドが正常に実行され、セキュリティ設定データが正しく設定されたことを示します。
異常終了 [B] 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。
	15H	・ 処理中にステータス・コマンド以外を受信しました。 ・ コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
異常終了 [D] 否定応答 (NACK)	15H	セキュリティ・データ・フレームが異常です。
	07H	送信したセキュリティ・データ・フレームのチェックサムが異常です。
	10H	セキュリティ・データが次の場合のエラーです。 ・ セキュリティを禁止から許可に変更 ・ ブート・ブロック・クラスタ書き換え禁止設定状態でブート・ブロック・クラスタの最終ブロック番号の値を変更
	05H	セキュリティ・データが次の場合のエラーです。 ・ ブート・ブロック・クラスタの最終ブロック番号がデバイスの最終ブロック番号より大きい ・ リセット・ベクタ・ハンドラ・アドレスの値が "00000000H" 以外
異常終了 [E] MRG10 エラー	1AH	書き込みエラーが発生しました。
	1BH	
	1CH	

8. 15. 4 フロー・チャート



8. 15. 5 サンプル・プログラム

Security Setコマンド処理のサンプル・プログラムです。

```

/*****/
/*
/* Set security flag command (CSI)
/*
/*****/
/* [i] u8 scf      ... Security flag data
/* [r] u16         ... error code
/*****/
u16      fl_csi_setscf(u8 scf, u8 bot, u32 vect)
{
    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|= 0b11100000; // "FLG" (bit 7,6,5 must be '1')
fl_txdata_frm[1] = bot;         // "BOT"

fl_txdata_frm[2] = (u8)(vect >> 16); // "ADH"
fl_txdata_frm[3] = (u8)(vect >> 8);  // "ADM"
fl_txdata_frm[4] = (u8) vect;        // "ADL"

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

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

fl_wait(tWT13);                // wait

rc = fl_csi_getstatus(tWT13_MAX); // get status frame
switch(rc) {
    case FLC_NO_ERR:             break; // continue
// case FLC_DFTO_ERR: return rc; break; // case [C]
    default:                    return rc; break; // case [B]
}

/*****/
/*      send data frame (security setting data)
/*****/
fl_wait(tFD3);                 // wait before getting data frame

put_dfrm_csi(5, 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]
    }

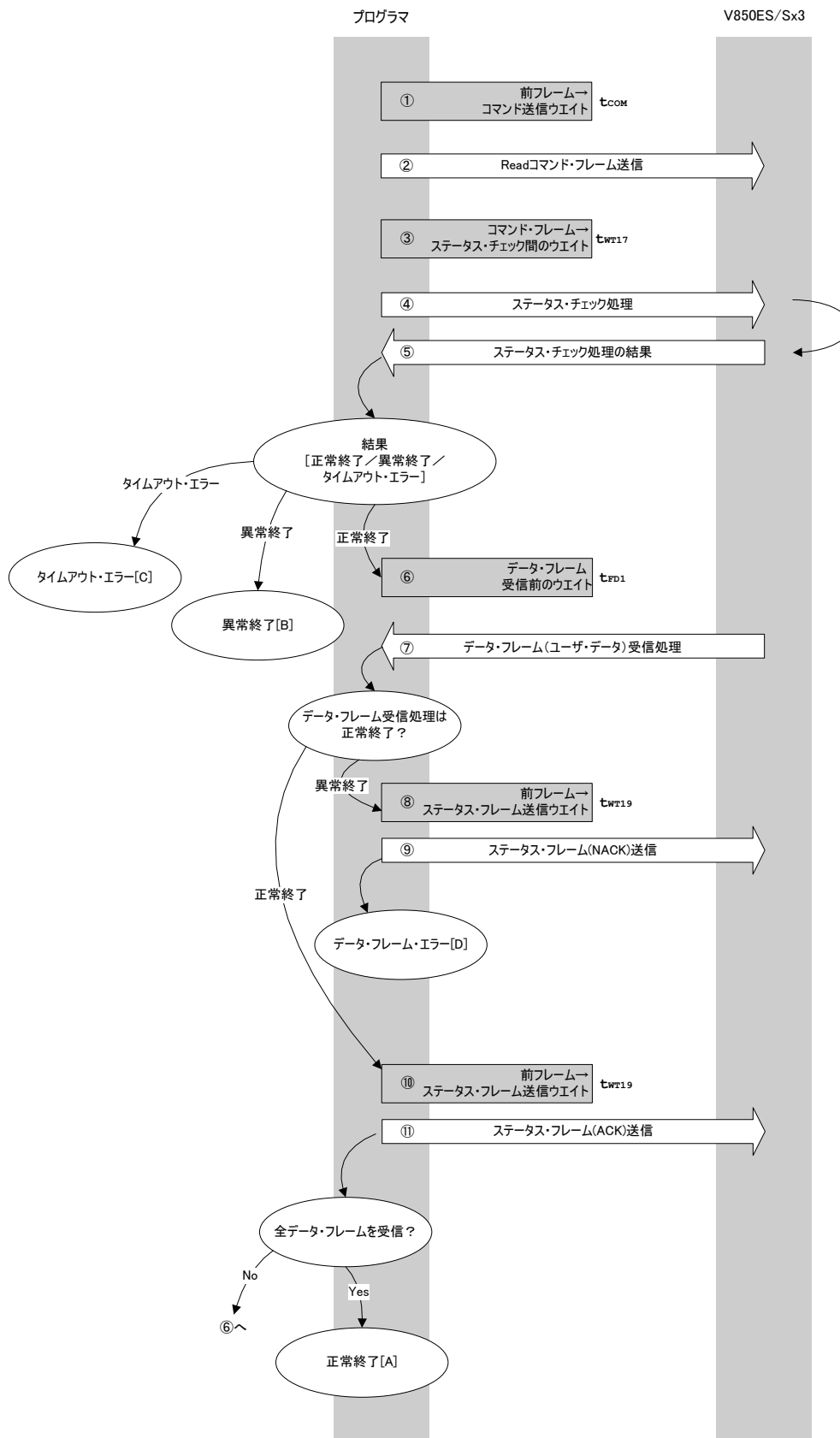
    /*****
    /*      Check internally verify      */
    /*****/
    fl_wait(tWT15);

    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;
}
```

8. 16 Readコマンド

8. 16. 1 処理手順チャート

Readコマンド処理手順



8. 16. 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、**Read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_{WT17})。
 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・フレーム受信までのウェイトをします (ウェイト時間 t_{WT18})。
 データ・フレーム受信処理により、データ・フレーム (ユーザ・データ) を受信します。
 受信処理の結果に応じて、次の処理を行います。

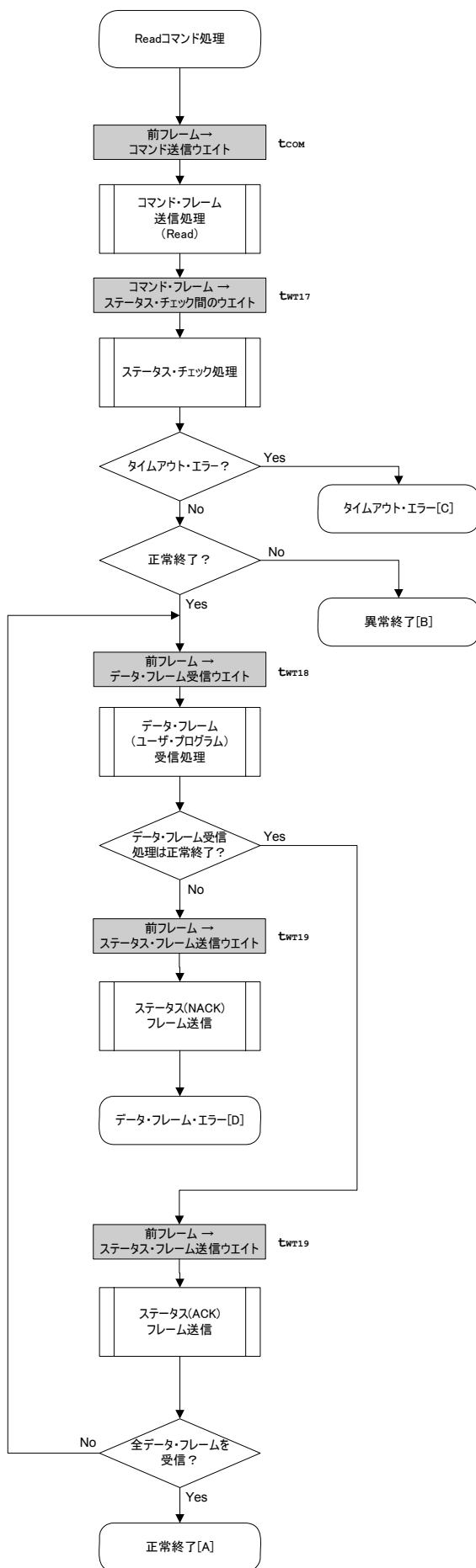
正常終了の場合 : に進みます。
 異常終了の場合 : に進みます。

直前のフレームからステータス (NACK) フレーム送信までのウェイトをします (ウェイト時間 t_{WT19})。
 データ・フレーム送信処理により、NACKフレームを送信します。
データ・フレーム・エラー[D]となります。
 直前のフレームからステータス (ACK) フレーム送信までのウェイトをします (ウェイト時間 t_{WT19})。
 データ・フレーム送信処理により、ACKフレームを送信します。
 全データ・フレームの受信が完了した場合は、**正常終了[A]**です。
 まだ受信すべきデータ・フレームが残っている場合は、より再実行します。

8. 16. 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、読み出し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	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。
	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。
	10H	セキュリティ設定で「読み出し禁止」になっています。
	15H	コマンド・フレーム・データが異常です (データ長 (LEN) 不正, ETX なしなど)。
タイムアウト・エラー [C]	-	規定の時間内にステータス・フレームの受信ができませんでした。
データ・フレーム・エラー [D]	-	読み出しデータとして受信したデータ・フレームのチェックサムが異常です。

8. 16. 4 フロー・チャート



8. 16. 5 サンプル・プログラム

Readコマンド処理のサンプル・プログラムです。

```

/*****/
/*
/* Read command (CSI)
/*
/*****/
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****/
u16      fl_csi_read(u32 top, u32 bottom)
{
    u16      rc;
    u32      read_head;
    u16      len;
    u8       hooter;

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

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

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

    put_cmd_csi(FL_COM_READ, 7, fl_cmd_prm); // send "Read" command

    fl_wait(tWT17);         // wait

    rc = fl_csi_getstatus(tWT17_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]
    }

    /*****/
    /*      receive user data
    /*****/
    read_head = top;

    while(1){
        fl_wait(tWT18);

        rc = get_dfrm_csi(fl_rxd_data_frm); // get ROM data from FLASH
        switch(rc) {
            case FLC_NO_ERR:   break; // continue
//          case FLC_RX_DFSUM_ERR:
            default:           // case [D]
                fl_wait(tWT19);
                put_sfrm_csi(FLST_NACK); // send status(NACK) frame
                return rc;
                break;
        }
    }
}

```



```
    }

    fl_wait(tWT19);
    put_sfrm_csi(FLST_ACK);          // send status(ACK) frame

    /* save ROM data */
    /* save ROM data */
    if ((len = fl_rxddata_frm[OFS_LEN]) == 0) // get length
        len = 256;

    memcpy(read_buf+read_head, fl_rxddata_frm+2, len);
                                                // save to external RAM

    read_head += len;

    /* end check */
    /* end check */
    hooter = fl_rxddata_frm[len + 3];
    if (hooter == FL_ETB) // end frame ?
        continue; // no
    break; // yes
}

return FLC_NO_ERR;
}
```

第9章 フラッシュ・メモリ・プログラミング・パラメータ特性

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V850ES/Sx3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、V850ES/Sx3各製品のユーザズ・マニュアルを参照のうえ、設計してください。

(1) フラッシュ・メモリ・パラメータ特性

(a) 動作クロックについて

V850ES/Sx3は、Oscillation Frequency Setコマンドでプログラマから指定されるメイン・クロック発振周波数 (f_x) の値によりメイン・クロック周波数 (f_{xx}) を変更します。

- ・ 2.5 MHz f_x 4.0 MHz : $f_{xx} = f_x \times 8$ (PLLモード)
- ・ 4.0 MHz < f_x 5.0 MHz : $f_{xx} = f_x \times 4$ (PLLモード)
- ・ 5.0 MHz < f_x 10.0 MHz : $f_{xx} = f_x$ (クロック・スルー・モード)

したがってOscillation Frequency Setコマンド以前(プログラマからのOscillation Frequency Setコマンド発行後のウエイト (t_{WT9})) までは $f_{xx} = f_x$ とし、それ以降は上記のように f_x によって異なった周波数を f_{xx} に代入してください。

備考 フラッシュ・メモリ・プログラミング・モード中に設定されるメイン・クロック周波数 (f_{xx}) は f_x に応じてV850ES/Sx3内部で自動的に設定されます。

(b) フラッシュ・メモリ・プログラミング・モード・セット時間

($T_A = -40 \sim +85 \text{ }^\circ\text{C}$, $V_{DD} = EV_{DD} = AV_{REF0} = AV_{REF1}$, $V_{SS} = EV_{SS} = AV_{SS} = 0 \text{ V}$, $C_L = 50 \text{ pF}$)

項 目	略号	条 件	MIN.	TYP.	MAX.
V_{DD} FLMD0	t_{DP}		1 ms		
FLMD0 RESET	t_{PR}		2 ms		
Count start time from RESET FLMD0 ^{注1}	t_{RP}		800 μs		
Count finish time from RESET FLMD0 ^{注1}	t_{RPE}				10 ms
FLMD0 counter high level width/low level width	t_{PW}		10 μs		100 μs
Wait for Reset command	t_{RC}	CSI, CSI + HS	0.3 s		
Wait for low level data1	t_{R1}	UART	0.3 s		
Wait for low level data2	t_{12}	UART	30000/ f_{XX}		
Wait for Reset command	t_{2C}	UART	30000/ f_{XX}		
Low level data1 width	t_{L1}	UART		注2	
Low level data2 width	t_{L2}	UART		注2	
FLMD0 counter rise time	t_R				1 μs
FLMD0 counter fall time	t_F				1 μs

注1. FLMD0端子信号の入力タイミングは ($t_{RP} + t_{RPE}$) $\div 2$ を標準値として推奨しています。

2. 9600 bps時の00Hデータ幅と同じ

(c) プログラミング特性

(TA = -40 ~ +85 °C, VDD = EVDD = AVREF0 = AVREF1, VSS = EVSS = AVSS = 0 V, CL = 50 pF)

項目	略号	条件	MIN.	MAX.	
Data ~ Data	tDR	Receive data frame	CSI, CSI + HS	192/fxx	
			UART	192/fxx	
	tDT	Send data frame	CSI, CSI + HS	177/fxx	
			UART	注	
Receive Status command frame ~ Send status frame	tsf	CSI, CSI + HS	3046/fxx		
Send status frame ~ Send data frame(1)	tFD1	CSI, CSI + HS	1425/fxx + 202676/fxx × M + 24 μs	1710/fxx + 243212/fxx × M + 29 μs	
		UART	注	1710/fxx + 243212/fxx × M + 29 μs	
Send status frame ~ Send data frame(2)	tFD2	CSI, CSI + HS	5685/fxx + 72 μs		
		UART	注		
Send status frame ~ Receive data frame(3)	tFD3	CSI, CSI + HS	3487/fxx + 36 μs		
		UART	3487/fxx + 36 μs		
Send status frame ~ Receive command frame	tCOM	-	730/fxx + 12 μs		

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

備考 M : ブロック数

fxx : メイン・クロック周波数

< tDR, tFD3, tCOM >

V850ES/Sx3は、直前の通信完了後、MIN.後から次の通信が可能となります。

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

< tDT, tsf, tFD2 >

V850ES/Sx3は、直前の通信完了後、MIN.後から次の通信が可能となります。

プログラマは、直前の通信完了後、MIN.時間経過後に次のデータの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。

< tFD1 >

V850ES/Sx3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V850ES/Sx3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。

CSI通信の場合、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。

UART通信の場合、V850ES/Sx3が、MIN. ~ MAX.時間の間にステータス・フレームを送信します。

(d) コマンド特性

(TA = -40 ~ +85 °C, VDD = EVDD = AVREF0 = AVREF1, VSS = EVSS = AVSS = 0 V, CL = 50 pF) (1/2)

コマンド	略号	条件	MIN.	MAX.
Reset	t _{WT0}	CSI, CSI + HS	255/f _{xx}	
		UART	注1	
Chip Erase	t _{WT1}	-	48024/f _{xx} + 98512 μs	52051/f _{xx} + 1943467 μs
Block Erase	t _{WT2}	-	7327/f _{xx} + (28413 μs + 308 μs × BM + 600/f _{xx}) + (... ^{注2}) + 72 μs	7327/f _{xx} + (284125 μs + 3072 μs × BM + 600/f _{xx}) + (... ^{注2}) + 72 μs
Program	t _{WT3}	CSI, CSI + HS	3472/f _{xx} + 48 μs	
		UART	注1	
	t _{WT4} ^{注3}	-	18765/f _{xx} + 603 μs	1035327/f _{xx} + 33090 μs
	t _{WT5}	CSI, CSI + HS	4249/f _{xx} + 38 μs + (259154/f _{xx} + 1191 μs) × M	5099/f _{xx} + 46 μs + (310985/f _{xx} + 1429 μs) × M
UART		注4	5099/f _{xx} + 46 μs + (310985/f _{xx} + 1429 μs) × M	
Verify	t _{WT6}	CSI, CSI + HS	517/f _{xx}	
		UART	注1	
	t _{WT7} ^{注3}	CSI, CSI + HS	6847/f _{xx} + 63 μs	
		UART	注5	
Block Blank Check	t _{WT8}	-	4416/f _{xx} + (20 μs + 308 μs × BM + 600/f _{xx}) + (... ^{注2}) + 24 μs	5300/f _{xx} + (24 μs + 369 μs × BM + 720/f _{xx}) + (... ^{注2}) + 29 μs
Oscillating Frequency Set	t _{WT9}	CSI, CSI + HS	10645/f _{xx}	
		UART	注1	
Baud rate set	t _{WT10}	UART	2984/f _{xx}	
Silicon Signature	t _{WT11}	CSI, CSI + HS	515/f _{xx}	
		UART	注1	
Version Get	t _{WT12}	CSI, CSI + HS	519/f _{xx}	
		UART	注1	

注1. プログラムはコマンド・フレーム送信前に受信許可にしておいてください。また、プログラムのタイムアウト時間は、3 s以上にしてください。

2. 同時選択処理を何回繰り返すかをBNとした場合に、表中の()内の計算をBMで指定される同時処理を行うサイズを変更しながら、加算してください。

例 同時処理を2→4→8というブロックに対して行うBlock EraseコマンドのMin.値の場合 (BN = 3)

$$7327/f_{xx} + (28413 \mu s + 308 \mu s \times 2 + 600/f_{xx}) + (28413 \mu s + 308 \mu s \times 4 + 600/f_{xx}) + (28413 \mu s + 308 \mu s \times 8 + 600/f_{xx}) + 72 \mu s$$

3. 64ワード・ユニット
4. プログラムは連続受信許可にしておいてください。また、プログラムのタイムアウト時間は、3 s以上にしてください。
5. プログラムはデータ・フレーム送信前に受信許可にしておいてください。また、プログラムのタイムアウト時間は3 s以上にしてください。

備考 M : ブロック数

BM : 同時選択処理ブロック数 (ブロック)

BN : 同時選択処理の実行回数 (表中の()の加算の繰り返し数)

f_{xx} : メイン・クロック周波数

(d) コマンド特性

(TA = -40 ~ +85 °C, VDD = EVDD = AVREF0 = AVREF1, VSS = EVSS = AVSS = 0 V, CL = 50 pF) (2/2)

コマンド	略号	条件	MIN.	MAX.
Security Setting	t _{WT13}	CSI, CSI + HS	469/f _{xx}	
		UART	注1	
	t _{WT14}	-	28893/f _{xx} + 1896 μs	29687/f _{xx} + 301429 μs
	t _{WT15}	CSI, CSI + HS	14274/f _{xx} + 28858 μs	14274/f _{xx} + 287334 μs
		UART	注2	14274/f _{xx} + 287334 μs
Checksum	t _{WT16}	CSI, CSI + HS	715/f _{xx}	
		UART	注1	
Read	t _{WT17}	CSI, CSI + HS	2074/f _{xx} + 24 μs	
		UART	注1	
	t _{WT18} ^{注3}	CSI, CSI + HS	13058/f _{xx} + 12 μs	
		UART	注4	
	t _{WT19}	-	148/f _{xx}	注5

- 注1. プログラマはコマンド・フレーム送信前に受信許可にしておいてください。また、プログラマのタイムアウト時間は、3 s以上にしてください。
2. プログラマは連続受信許可にしておいてください。また、プログラマのタイムアウト時間は、3 s以上にしてください。
3. 64ワード・ユニット
4. プログラマはステータス・フレーム送信前に受信許可にしておいてください。また、プログラマのタイムアウト時間は3 s以上にしてください。
5. プログラマからのステータス・フレーム受信待ち

備考 f_{xx} : メイン・クロック周波数

< t_{WT0} – t_{WT9}, t_{WT11} – t_{WT19} >

- ・ MIN. と MAX. の規定がある項目の場合

V850ES/Sx3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V850ES/Sx3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。

CSI通信の場合、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。

UART通信の場合、V850ES/Sx3が、MIN. ~ MAX. 時間の間にステータス・フレームを送信します。

- ・ 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。

< t_{WT10} >

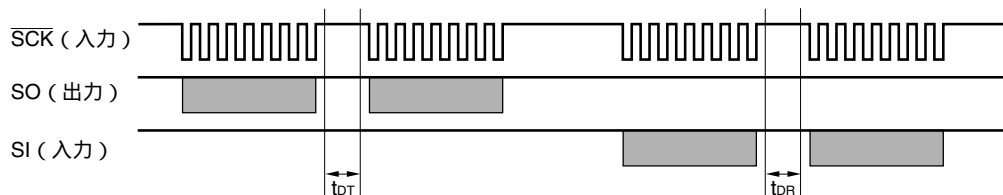
V850ES/Sx3は、直前の通信完了後、MIN. 後から次の通信が可能となります。

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

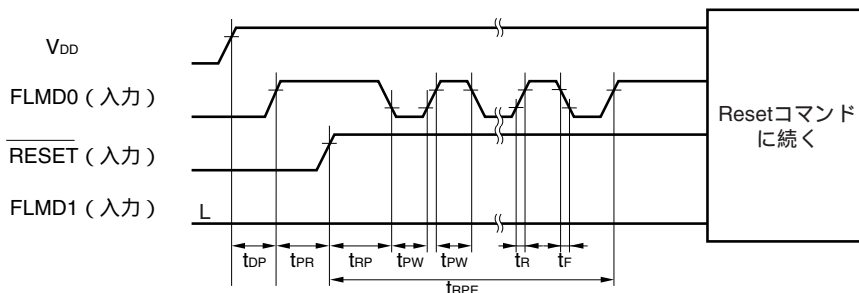
CSI通信タイミング

(1/3)

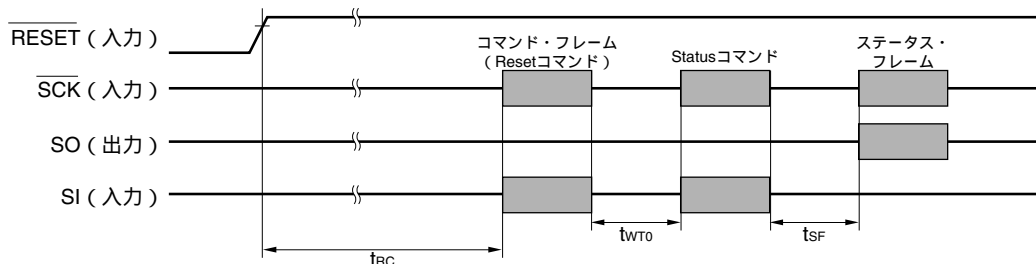
(a) データ・フレーム



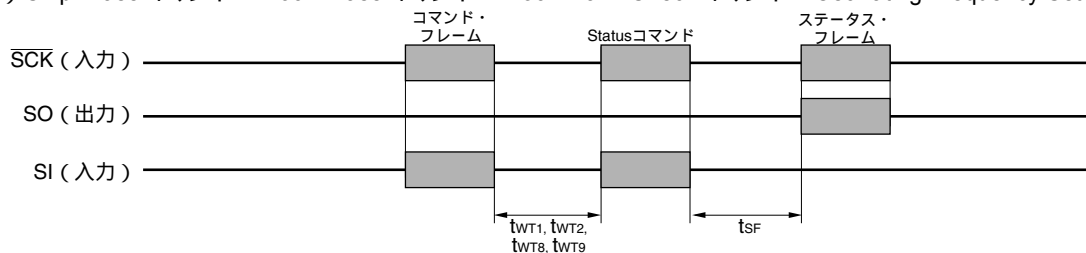
(b) プログラミング・モード設定



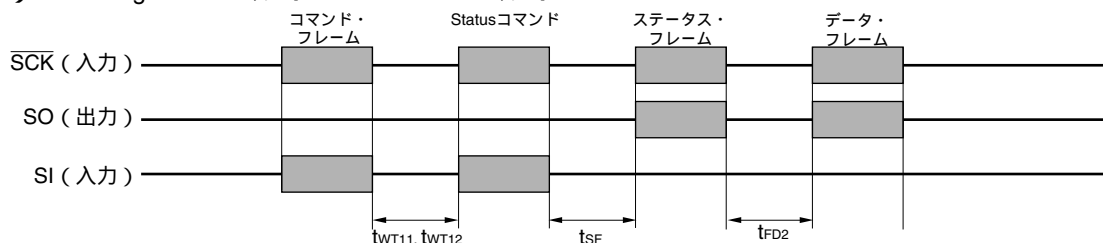
(c) Resetコマンド



(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



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

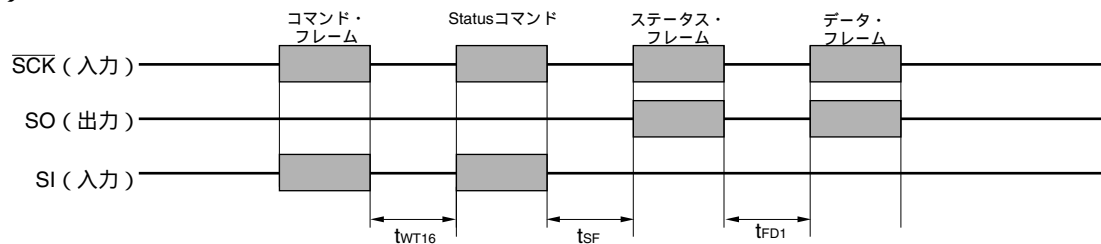


備考 SCK : SCKB0, SCKB3

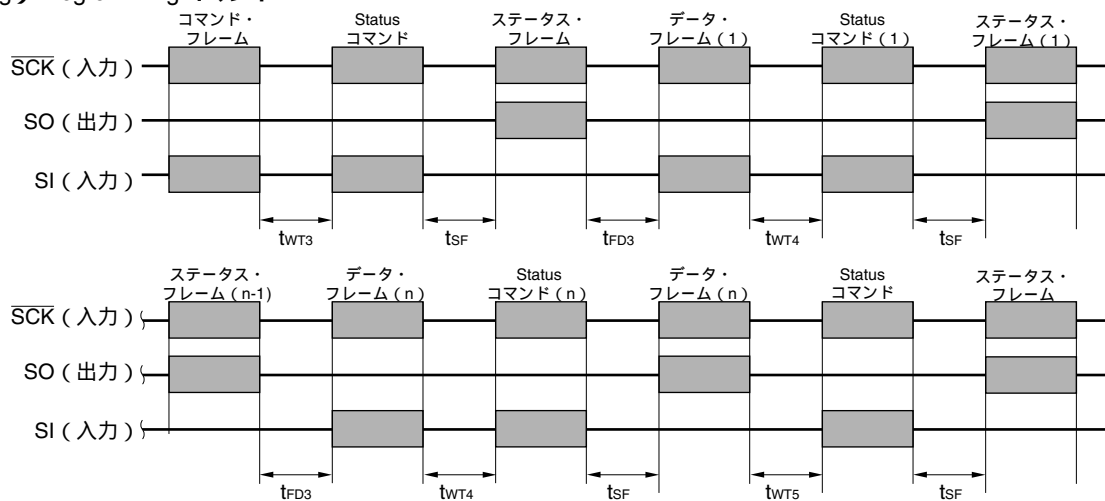
SO : SOB0, SOB3

SI : SIB0, SIB3

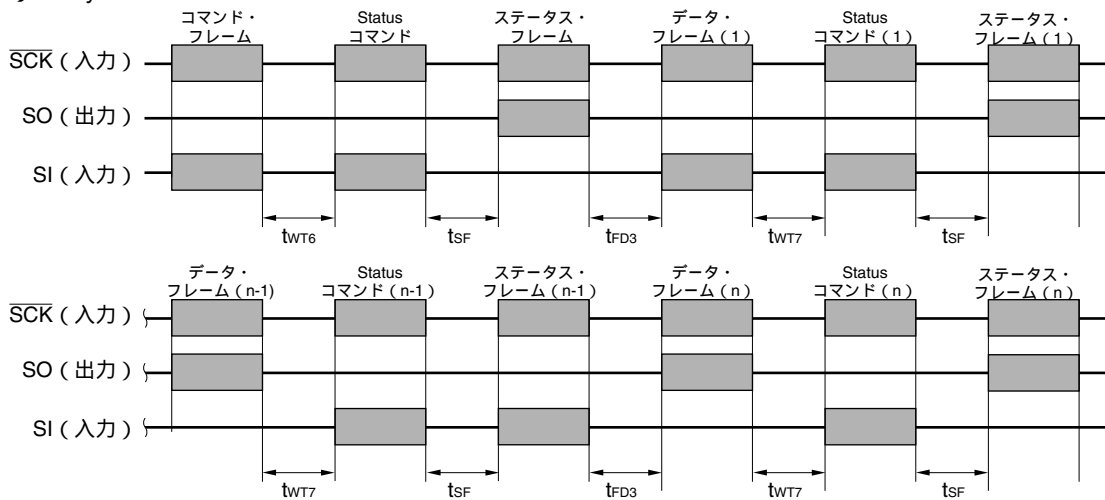
(f) Checksumコマンド



(g) Programmingコマンド



(h) Verifyコマンド

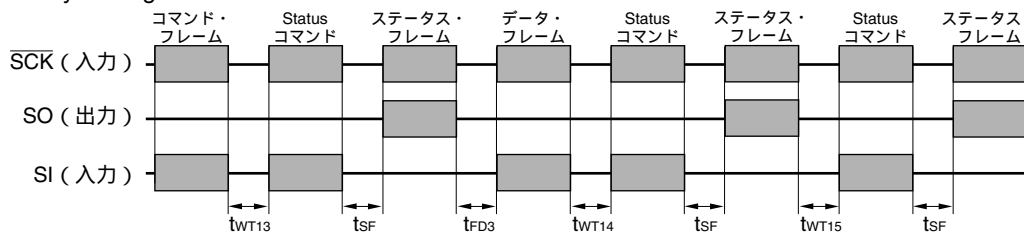


備考 SCK : SCKB0, SCKB3

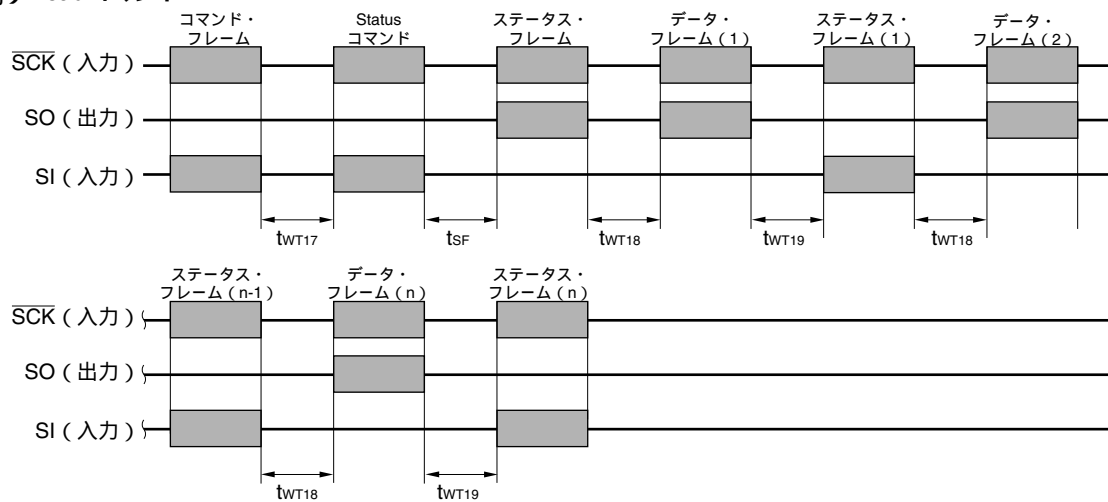
SO : SOB0, SOB3

SI : SIB0, SIB3

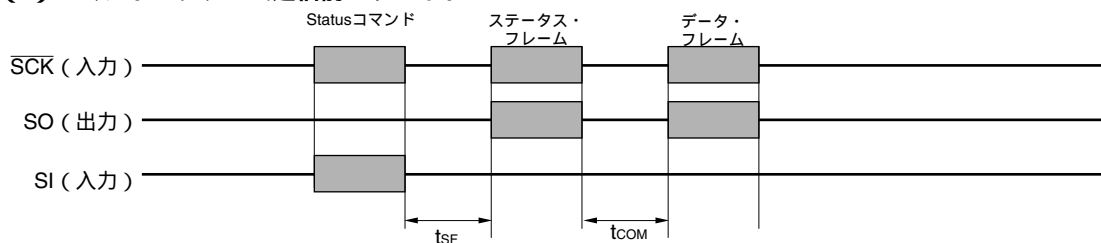
(i) Security Settingコマンド



(j) Readコマンド



(k) コマンド・フレーム送信前のウェイト



備考 $\overline{\text{SCK}}$: SCKB0, SCKB3

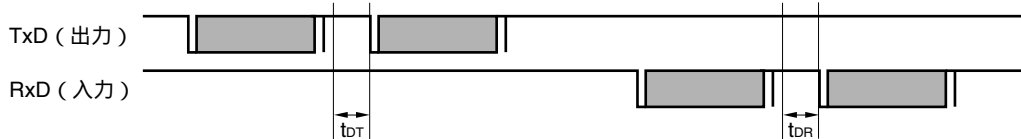
SO : SOB0, SOB3

SI : SIB0, SIB3

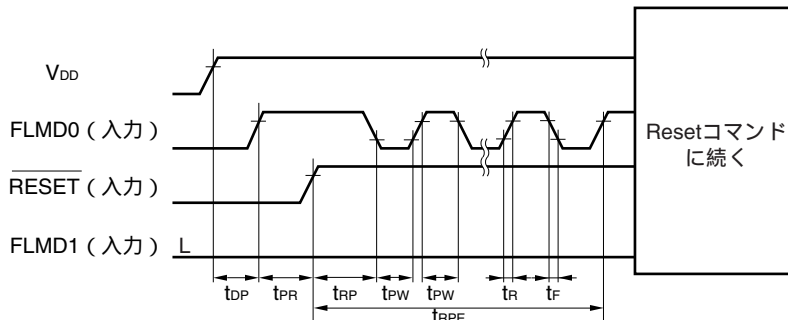
UART通信タイミング

(1/3)

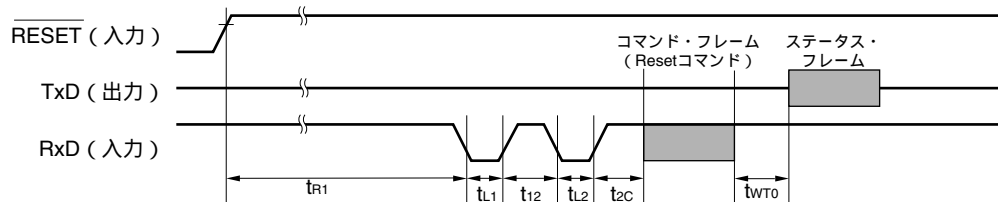
(a) データ・フレーム



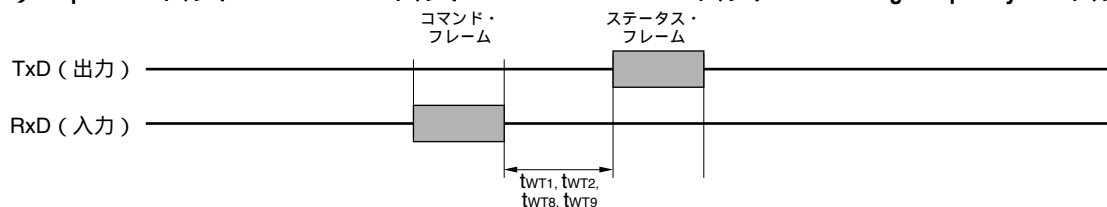
(b) プログラミング・モード設定



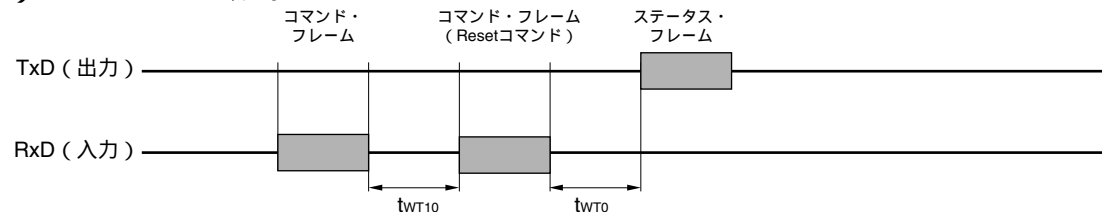
(c) Resetコマンド



(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



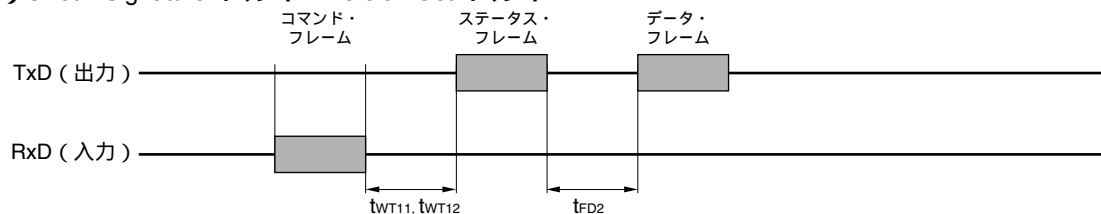
(e) Baud Rate Setコマンド



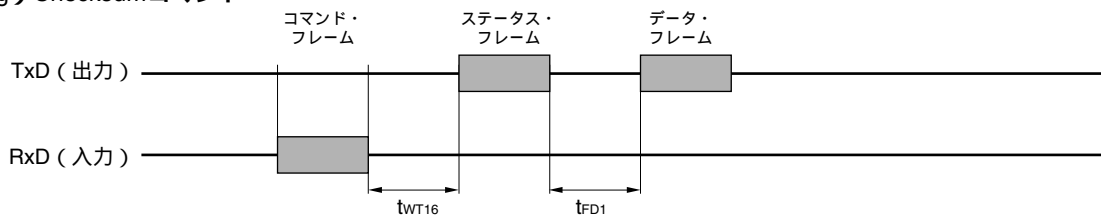
備考 TxD : TXDA0

RxD : RXDA0

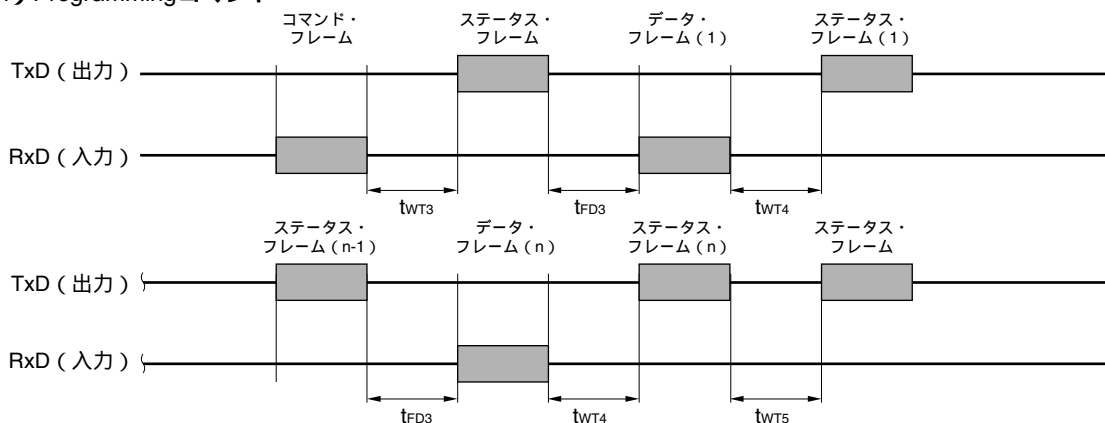
(f) Silicon Signatureコマンド / Version Getコマンド



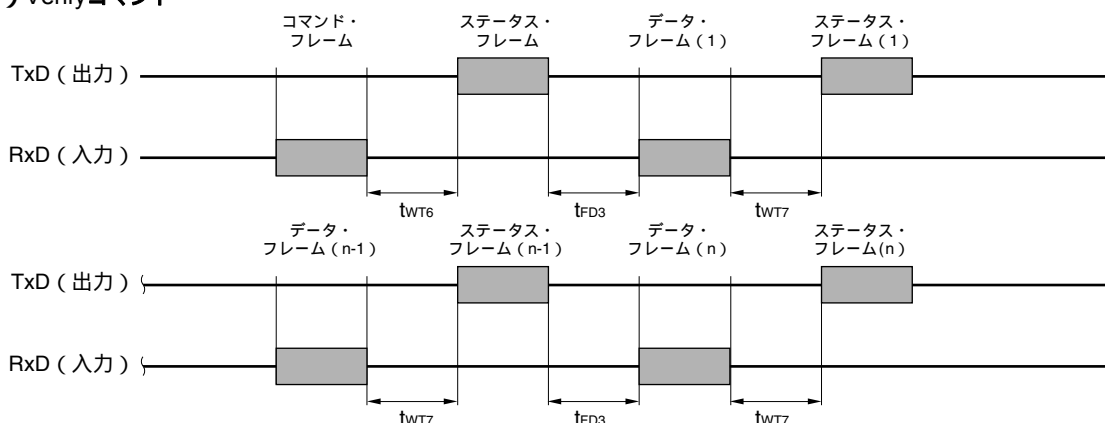
(g) Checksumコマンド



(h) Programmingコマンド



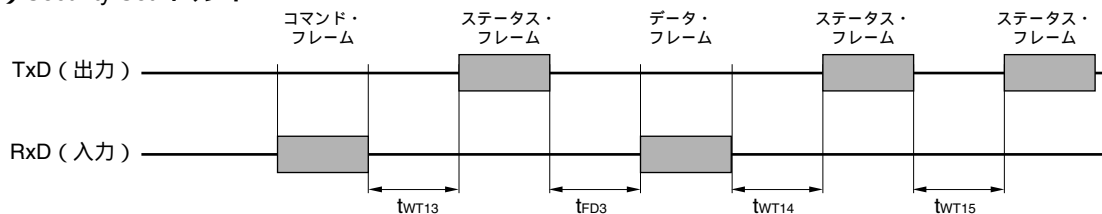
(i) Verifyコマンド



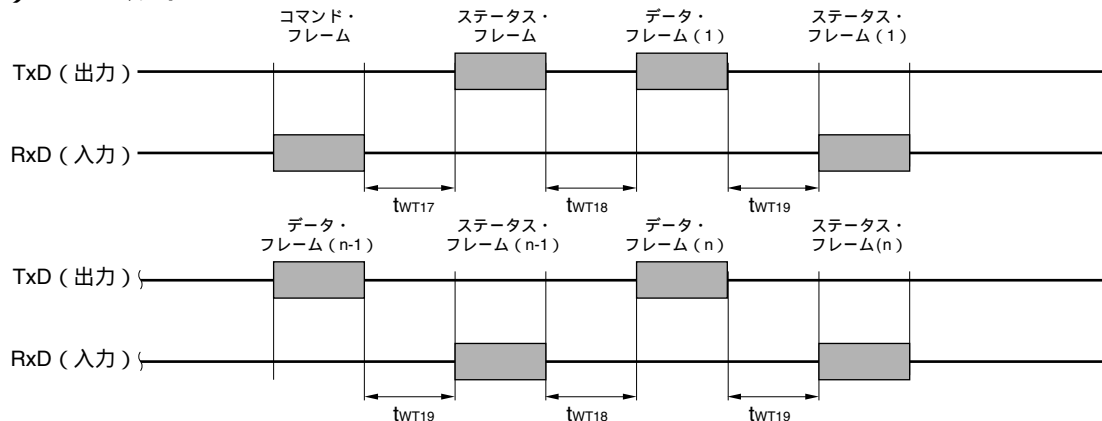
備考 TxD : TXDA0

RxD : RXDA0

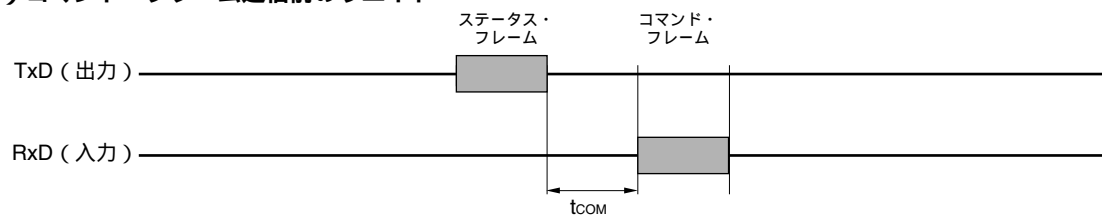
(j) Security Setコマンド



(k) Readコマンド



(l) コマンド・フレーム送信前のウェイト



備考 TxD : TXDA0
RxD : RXDA0

(2) 同時選択ブロック処理について

ブロック消去, ブランク・チェック, 内部ペリファイなどの処理では, 複数ブロックを同時に処理する“同時選択処理”を繰り返し実行することにより実現しています。

従って, ウェイト時間は, “同時選択処理”の実行時間の総和となります。

“同時選択処理の実行時間の総和”を算出するためには, 同時選択処理の実行回数 (BN) と同時選択処理ブロック数 (BM) を算出する必要があります。

(a) 同時選択処理ブロック数 (BM) と同時選択処理の実行回数 (BN)

BNは, 同時に処理するブロック数 (BM: 同時選択処理ブロック数) を求めながら, 算出します。

同時選択処理ブロック数 (BM) 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】

処理ブロック数 (ER_BKNUM) 同時選択処理ブロック数候補 (SSER_BKNUM)

【条件2】

開始ブロック番号 (ST_BKNO) ÷ 同時選択処理ブロック数候補 (SSER_BKNUM) = 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

【条件1】【条件2】【条件3】を満たした同時選択ブロック処理例を次に示します。

(例1) ブロック1-127の場合

<1> 最初の開始ブロック番号は1で、処理ブロック数が127であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1

よって、【条件3】を満たす数値は“1”となり、同時選択処理ブロック数 (BM) は“1”となることから、ブロック1のみ処理します。

<2> ブロック1を処理すると、次の開始ブロック番号は2で、処理ブロック数が126であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1, 2

よって、【条件3】を満たす数値は“2”となり、同時選択処理ブロック数 (BM) は“2”となることから、ブロック2, 3を処理します。

<3> ブロック2, 3を処理すると、次の開始ブロック番号は4で、処理ブロック数が124であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1, 2, 4

よって、【条件3】を満たす数値は“4”となり、同時選択処理ブロック数 (BM) は“4”となることから、ブロック4-7を処理します。

<4> ブロック4-7を処理すると、次の開始ブロック番号は8で、処理ブロック数が120であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1, 2, 4, 8

よって、【条件3】を満たす数値は“8”となり、同時選択処理ブロック数 (BM) は“8”となることから、ブロック8-15を処理します。

<5> ブロック8-15を処理すると、次の開始ブロック番号は16で、処理ブロック数が112であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1, 2, 4, 8, 16

よって、【条件3】を満たす数値は“16”となり、同時選択処理ブロック数 (BM) は“16”となることから、ブロック16-31を処理します。

<6> ブロック16-31を処理すると、次の開始ブロック番号は32で、処理ブロック数が96であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

さらに【条件2】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32

よって、【条件3】を満たす数値は“32”となり、同時選択処理ブロック数 (BM) は“32”となることから、ブロック32-63を処理します。

<7> ブロック 32-63 を処理すると、次の開始ブロック番号は 64 で、処理ブロック数が 64 であることから、【条件 1】を満たす数値は、次のようになります。

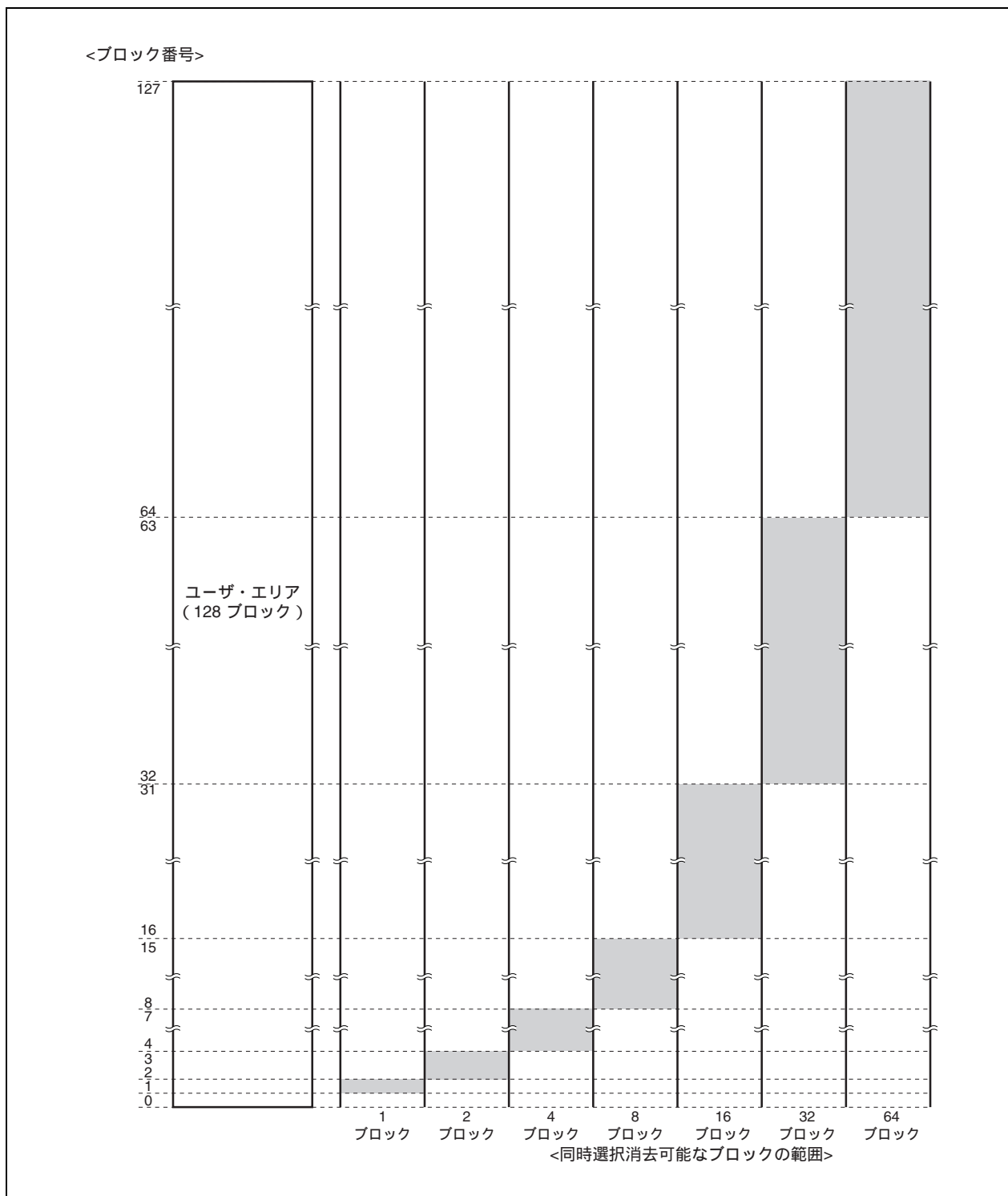
1, 2, 4, 8, 16, 32, 64

さらに【条件 2】を満たす数値は、次のようになります。

1, 2, 4, 8, 16, 32, 64

よって、【条件 3】を満たす数値は“ 64 ”となり、同時選択処理ブロック数 (BM) は“ 64 ”となることから、ブロック 64-127 を処理します。

以上より、ブロック 1-127 の場合、1, 2-3, 4-7, 8-15, 16-31, 32-63, 64-127 の 7 回、同時選択処理を実行するため、BN = 7 となります。



(例2) ブロック5-10の場合

<1> 最初の開始ブロック番号は5で、処理ブロック数が6であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4

さらに【条件2】を満たす数値は、次のようになります。

1

よって、【条件3】を満たす数値は“1”となり、同時選択処理ブロック数 (BM) は“1”となることから、ブロック5のみ処理します。

<2> ブロック5を処理すると、次の開始ブロック番号は6で、処理ブロック数が5であることから、【条件1】を満たす数値は、次のようになります。

1, 2, 4

さらに【条件2】を満たす数値は、次のようになります。

1, 2

よって、【条件3】を満たす数値は“2”となり、同時選択処理ブロック数 (BM) は“2”となることから、ブロック6, 7を処理します。

<3> ブロック6, 7を処理すると、次の開始ブロック番号は8で、処理ブロック数が3であることから、【条件1】を満たす数値は、次のようになります。

1, 2

さらに【条件2】を満たす数値は、次のようになります。

1, 2

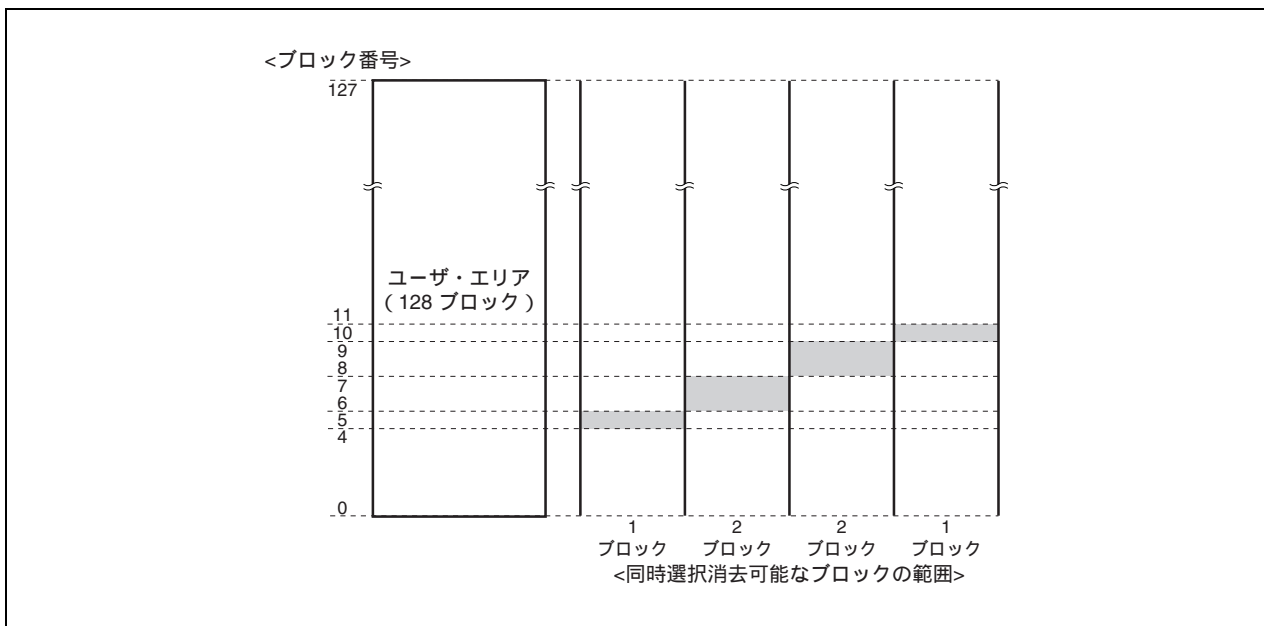
よって、【条件3】を満たす数値は“2”となり、同時選択処理ブロック数 (BM) は“2”となることから、ブロック8, 9を処理します。

<4> ブロック8, 9を処理すると、次の開始ブロック番号は10で、処理ブロック数が1であることから、【条件1】を満たす数値は、次のようになります。

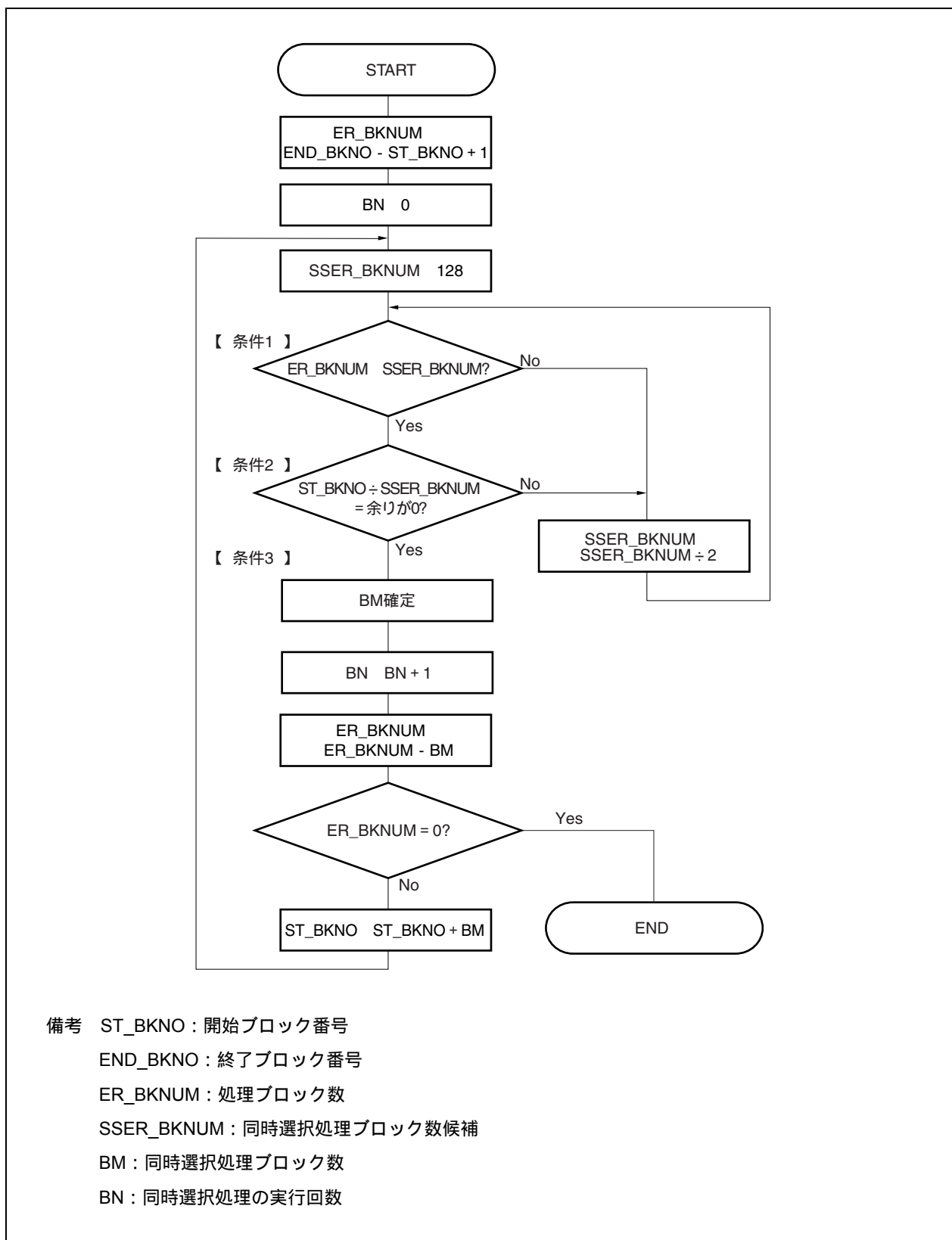
1

これは、【条件2】と【条件3】も満たしているため、同時選択処理ブロック数 (BM) は“1”となることから、ブロック10を処理します。

以上より、ブロック5-10の場合、5, 6-7, 8-9, 10の4回、同時選択処理を実行するため、BN = 4 となります。



【条件1】 【条件2】 【条件3】を満たしたBM, BNの求め方のフロー例を次に示します。

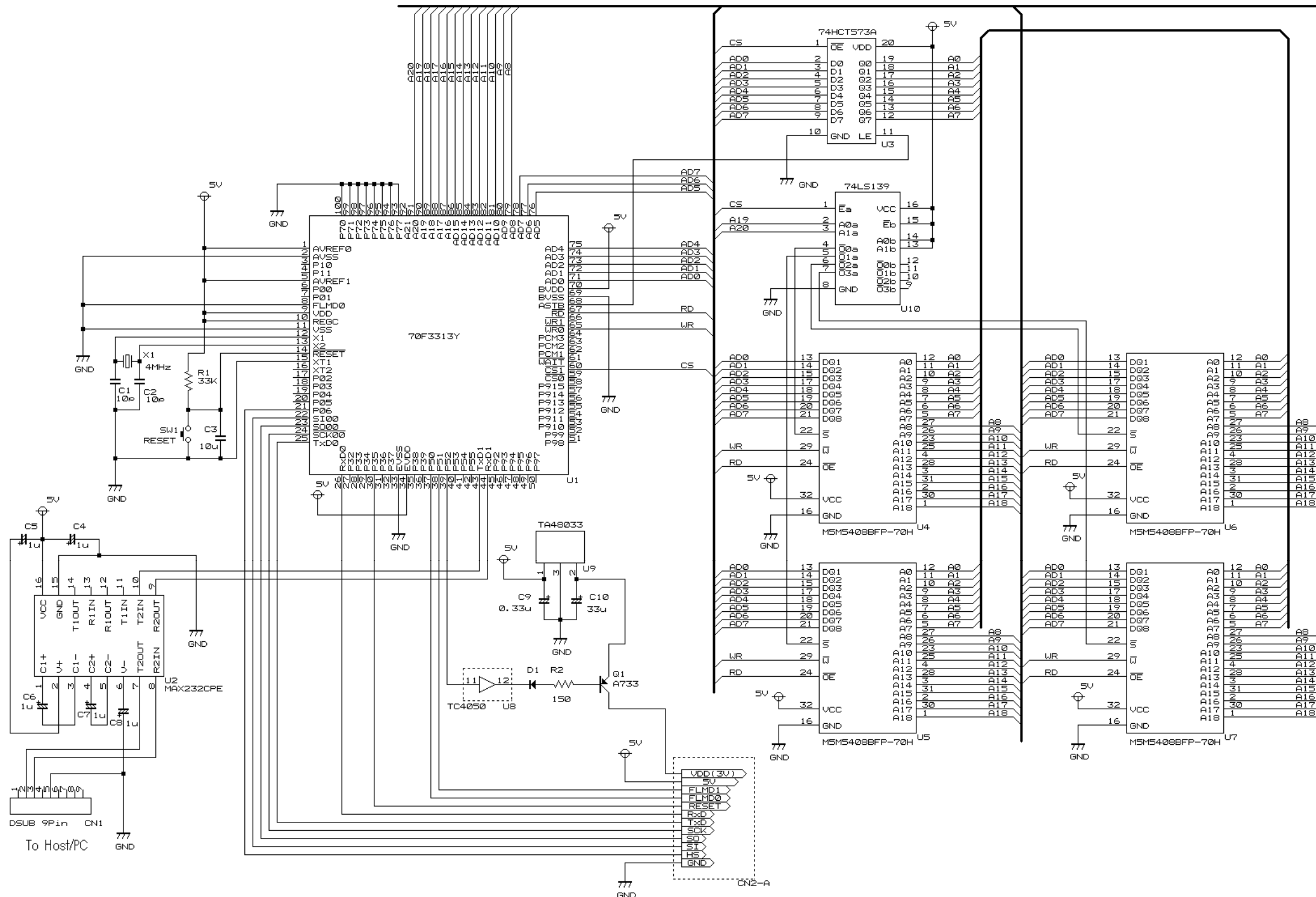


付録A 参考回路図

プログラマとV850ES/Sx3の参考回路図を図A - 1, A - 2に示します。

図A - 1 プログラマとV850ES/Sx3の参考回路図 (メイン・ボード)

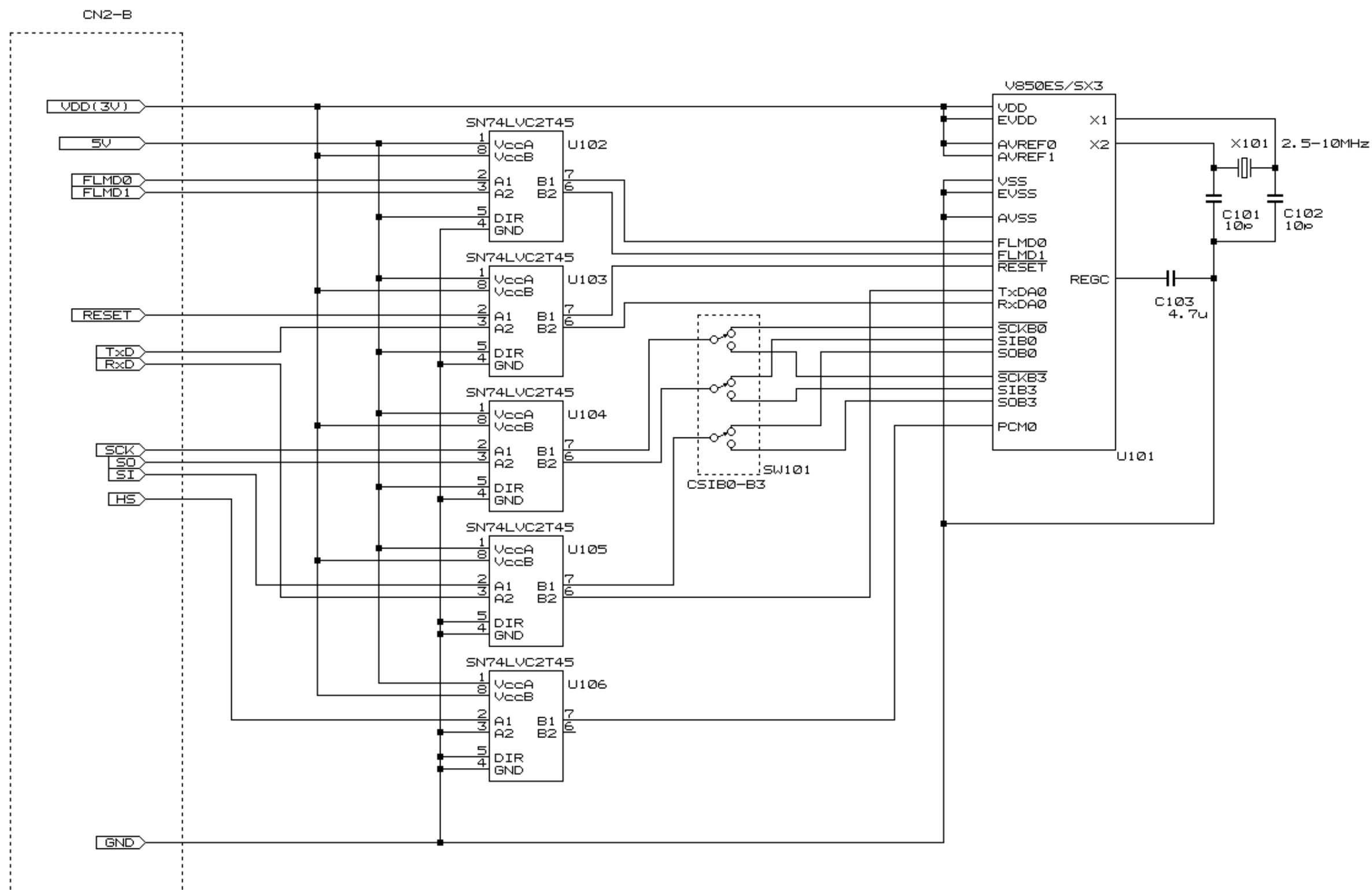
V850ES/Sx3 Flash Programmer sample application MAIN BOARD



備考 回路図中の未使用端子に関しては、各製品のユーザーズ・マニュアルにしたがい、端子処理を行ってください。

図A-2 プログラマとV850ES/Sx3の参考回路図 (ターゲット・ボード)

V850ES/Sx3 Flash Programmer sample application TARGET BOARD



備考 回路図中の未使用端子に関しては、各製品のユーザズ・マニュアルにしたがい、端子処理を行ってください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<http://japan.renesas.com/>

お問合せ先

<http://japan.renesas.com/inquiry>

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2007.7.13	-	初版発行
2.00	2011.12.28	p. 23	図2 - 5 フラッシュ・メモリ書き換え時の一般的なコマンド・フロー 修正
		p. 48	5. 10. 1 説 明 修正
		p. 57	5. 12. 1 説 明 修正
		p. 60	表5 - 4 セキュリティ・フラグ・フィールドと各コマンドの禁止 / 許可 修正
		p. 79	6. 7. 3 終了時の内容 修正
		p. 83	6. 8. 3 終了時の内容 修正
		p. 88	6. 9. 3 終了時の内容 修正
		p. 117	6. 15. 5 サンプル・プログラム 修正
		p. 140	7. 7. 3 終了時の内容 修正
		p. 144	7. 8. 3 終了時の内容 修正
		p. 149	7. 9. 3 終了時の内容 修正
		p. 180	7. 15. 5 サンプル・プログラム 修正
		p. 205	8. 7. 3 終了時の内容 修正
		p. 209	8. 8. 3 終了時の内容 修正
		p. 214	8. 9. 3 終了時の内容 修正
		p. 243	8. 15. 5 サンプル・プログラム 修正
		p. 250	第9章 フラッシュ・メモリ・プログラミング・パラメータ特性 ・ (1) フラッシュ・メモリ・パラメータ特性 (a) 動作クロックについて 修正
		p. 252	・ (1) フラッシュ・メモリ・パラメータ特性 (c) プログラミング特性 修正
p. 253	・ (1) フラッシュ・メモリ・パラメータ特性 (d) コマンド特性 修正		
p. 255	・ (1) フラッシュ・メモリ・パラメータ特性 CSI通信タイミング (e) Silicon Signatureコマンド / Version Getコマンド 修正		
p.259	・ (1) フラッシュ・メモリ・パラメータ特性 UART通信タイミング (h) Programmingコマンド 修正		

CMOSデバイスの一般的注意事項

- (1) 入力端子の印加波形：入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOSデバイスの入力がノイズなどに起因して、 V_{IL} (MAX.) から V_{IH} (MIN.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん、 V_{IL} (MAX.) から V_{IH} (MIN.) までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。
- (2) 未使用入力の処理：CMOSデバイスの未使用端子の入力レベルは固定してください。未使用端子入力については、CMOSデバイスの入力に何も接続しない状態で動作させるのではなく、プルアップかプルダウンによって入力レベルを固定してください。また、未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると、個別に抵抗を介して V_{DD} または GND に接続することが有効です。資料中に「未使用端子の処理」について記載のある製品については、その内容を守ってください。
- (3) 静電気対策：MOSデバイス取り扱いの際は静電気防止を心がけてください。MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレイやマガジン・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。
- (4) 初期化以前の状態 電源投入時、MOSデバイスの初期状態は不定です。電源投入時の端子の出力状態や出力設定、レジスタ内容などは保証しておりません。ただし、リセット動作やモード設定で定義している項目については、これらの動作ののちに保証の対象となります。リセット機能を持つデバイスの電源投入後は、まずリセット動作を実行してください。
- (5) 電源投入切断順序 内部動作および外部インタフェースで異なる電源を使用するデバイスの場合、原則として内部電源を投入した後に外部電源を投入してください。切断の際には、原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により、内部素子に過電圧が印加され、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源投入切断シーケンス」についての記載のある製品については、その内容を守ってください。
- (6) 電源OFF時における入力信号 当該デバイスの電源がOFF状態の時に、入力信号や入出力プルアップ電源を入れしないでください。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。

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情報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、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。
標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）
特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、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と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。



ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。

総合お問合せ窓口：<http://japan.renesas.com/inquiry>