RZ/T1 グループ
CAN インタフェースサンプルプログラム

R01AN3109JJ0120
Rev.1.20
2018.12.12

要旨

本アプリケーションノートでは、RZ/T1 Evaluation Board 上に実装される CAN コントローラ 2 チャネル (CAN0、CAN1) の内の CAN0 を使用しての通信を行うサンプルプログラムについて説明します。

CAN インターフェースサンプルプログラムの特長を以下に示します。

- 送信機能
  送信バッファを使用したメッセージ送信
  送受信 FIFO（送信モード）バッファを使用したメッセージ送信
- 受信機能
  受信バッファを使用したメッセージ受信
  受信 FIFO バッファを使用したメッセージ受信
  送受信 FIFO（受信モード）バッファを使用したメッセージ受信
- テスト機能
  セルフテストモード 0（外部ループバックモード）
    送信バッファ → 受信バッファ
    送信バッファ → 受信 FIFO バッファ
    送信バッファ → 送受信 FIFO（受信モード）バッファ
    送受信 FIFO（送信モード）バッファ → 送受信 FIFO（受信モード）バッファ
  セルフテストモード 1（内部ループバックモード）
    送信バッファ → 受信バッファ
    送信バッファ → 受信 FIFO バッファ
    送信バッファ → 送受信 FIFO（受信モード）バッファ
    送受信 FIFO（送信モード）バッファ → 送受信 FIFO（受信モード）バッファ
- 通信速度
  1Mbps、500Kbps、125Kbps に対応（メニューより選択可）

制限事項

本サンプルプログラムには以下の制限事項があります。

(1) 使用チャネル固定。(CAN0)

(2) メッセージフォーマット固定。(標準 ID(0x120)、データフレーム)

(3) 受信ルール固定。

受信ルールテーブルページ番号：0

受信ルール数：1（テーブル番号：0）

受信ルール ID：標準 ID(0x120)、データフレーム

(4) 使用バッファ固定。

送信バッファ番号：0

受信バッファ番号：1

受信 FIFO バッファ番号：0

送受信 FIFO（送信モード）バッファ番号：0

送受信 FIFO（送信モード）バッファにリンクさせる送信バッファ番号：2

送受信 FIFO（受信モード）バッファ番号：1

(5) その他

以下の機能については対応していません。

送信アボート、送信キューによる送信、送信履歴機能、ゲートウェイ機能、テスト機能（標準テストモード、リッスンオンリモード、RAM テスト、チャネル間通信テスト)、RSCAN RAM のエラー検出／訂正

対象デバイス

RZ/T1 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

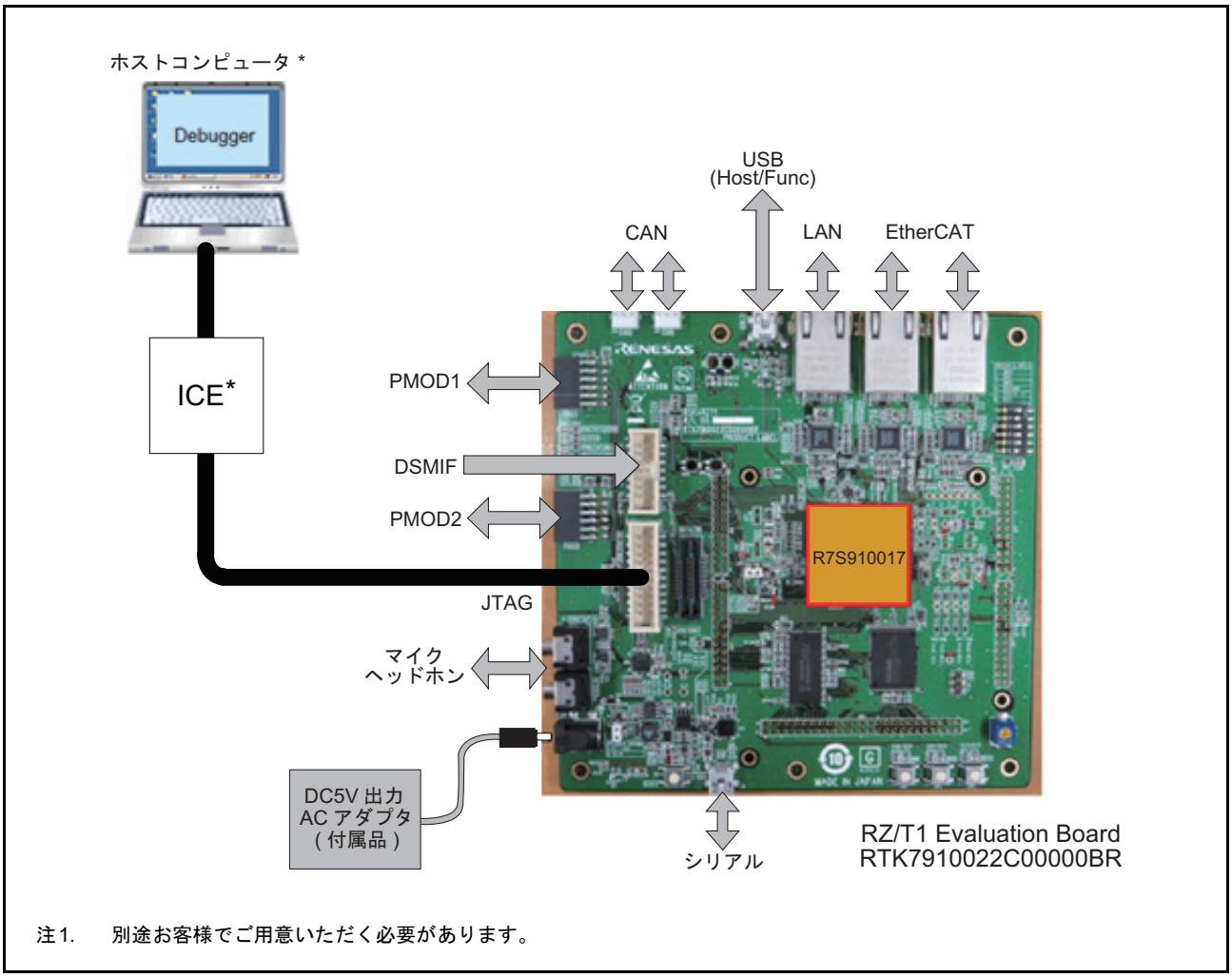
目　次

1.　仕様

表 1.1 に使用する周辺機能と用途を、図 1.1 にサンプルコード実行時の動作環境を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
CAN インターフェース（RSCAN） CAN0	本 LSI を使用して CAN バス上でデータを送信、受信するために使用します。
消費電力低減機能	RSCAN モジュールの起動/停止制御(MSTPCRB1)
割り込みコントローラ(ICUA)	RSCAN の割り込み制御 CAN グローバルエラー（ベクタ 262） CAN0 エラー（ベクタ 263） CAN1 エラー（ベクタ 264） CAN 受信 FIFO（ベクタ 104） CAN0 送受信 FIFO 受信完了（ベクタ 105） CAN0 送信（ベクタ 106） CAN1 送受信 FIFO 受信完了（ベクタ 107） CAN1 送信（ベクタ 108）
I/O ポート	CAN0：CRXD0（入力）PC6 CAN0：CTXD0（出力）P67 CAN1：CRXD1（入力）PC7 CAN1：CTXD1（出力）P66



注 1.　別途お客様でご用意いただく必要があります。

図 1.1　動作環境

2. 動作環境

本アプリケーションノートのサンプルコードは、下記の環境を想定しています。

表2.1　動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUクロック（CPUCLK）：450MHz
動作電圧	電源電圧（I/O）：3.3V
統合開発環境	IARシステムズ製 Embedded Workbench® for Arm Version 8.20.2 Arm製 DS-5™ 5.26.2 RENESAS製 e²studio 6.1.0
動作モード	SPIブートモード 16ビットバスブートモード
CAN動作モード	グローバル・ストップ・モード グローバル・リセット・モード グローバル・テスト・モード グローバル動作モード チャネル・ストップ・モード チャネル・リセット・モード チャネル待機モード チャネル通信モード
ターミナルソフトの通信設定	• 通信速度：115200bps • データ長：8ビット • パリティ：なし • ストップビット長：1ビット • フロー制御：なし • 改行コード（受信）：CR • 改行コード（送信）：CR
使用ボード	RZ/T1 Evaluation Board （以下、評価ボード） (RTK7910022C00000BR)
使用デバイス （ボード上で使用する機能）	シリアルインタフェース（USB-Mini Bコネクタ J8） CANコントローラ（RSCAN） 　ISO11898-1仕様準拠（標準フレーム／拡張フレーム）

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RZ/T1 グループ初期設定アプリケーションノート（R01AN2554JJ）

注 . 本アプリケーションノートで記載しないレジスタに関しては、RZ/T1 グループ 初期設定アプリケーションノートで設定した値のまま使用します。

4. 周辺機能説明

消費電力低減機能、I/O ポート、マルチファンクションピンコントローラ（MPC）等 CAN インターフェースに関連する機能については、『RZ/T1 グループ・ユーザーズマニュアルハードウェア編』を参照してください。

5. ハードウェア説明

5.1 使用端子一覧

表 5.1 に使用端子と機能を示します。

表5.1　使用端子と機能

チャネル	端子名	入出力	内容
CAN0	CRXD0	入力	CAN0受信データ入力端子
	CTXD0	出力	CAN0送信データ出力端子
CAN1	CRXD1	入力	CAN1受信データ入力端子
	CTXD1	出力	CAN1送信データ出力端子

5.2 参考回路

図 5.1 にブロック図を示します。

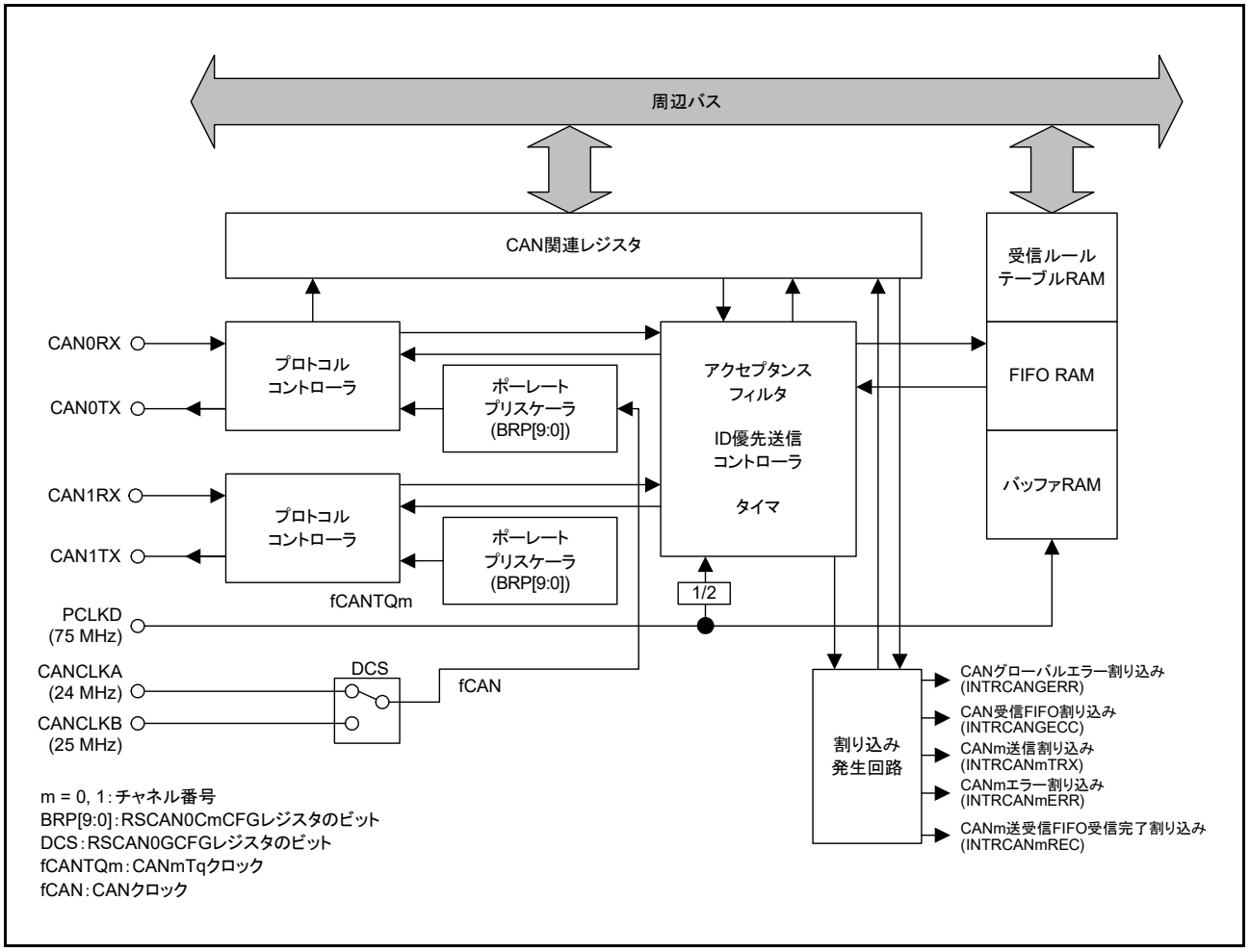


図 5.1　RSCAN のブロック図

6. CAN コンフィグレーション

6.1 CAN コンフィグレーション

CAN コンフィグレーションでは、CAN 通信を行う場合に必要な機能の設定を行います。CAN コンフィグレーションは、MCU リセットやバス異常検出、ウェイクアップなどの後に CAN 通信を開始、再開するときに実施してください。

CAN コンフィグレーションが実施可能なモードは以下のとおりです。CAN 状態（モード）については「6.2 CAN 状態 (モード) 遷移」を参照ください。

- グローバル・リセット・モード
- チャネル・リセット・モード
- チャネル待機モード

CAN コンフィグレーション時に設定が必要な機能を以下に示します。各処理の詳細については次章以降を参照ください。

- CAN 状態（モード）遷移
- 通信速度
- グローバル機能
- 受信ルール・テーブル
- バッファ
- グローバル・エラー割り込み
- チャネル機能

(1) MCU リセット後の CAN コンフィグレーション

MCU リセット後に CAN モジュール全体の初期化処理を実施します。

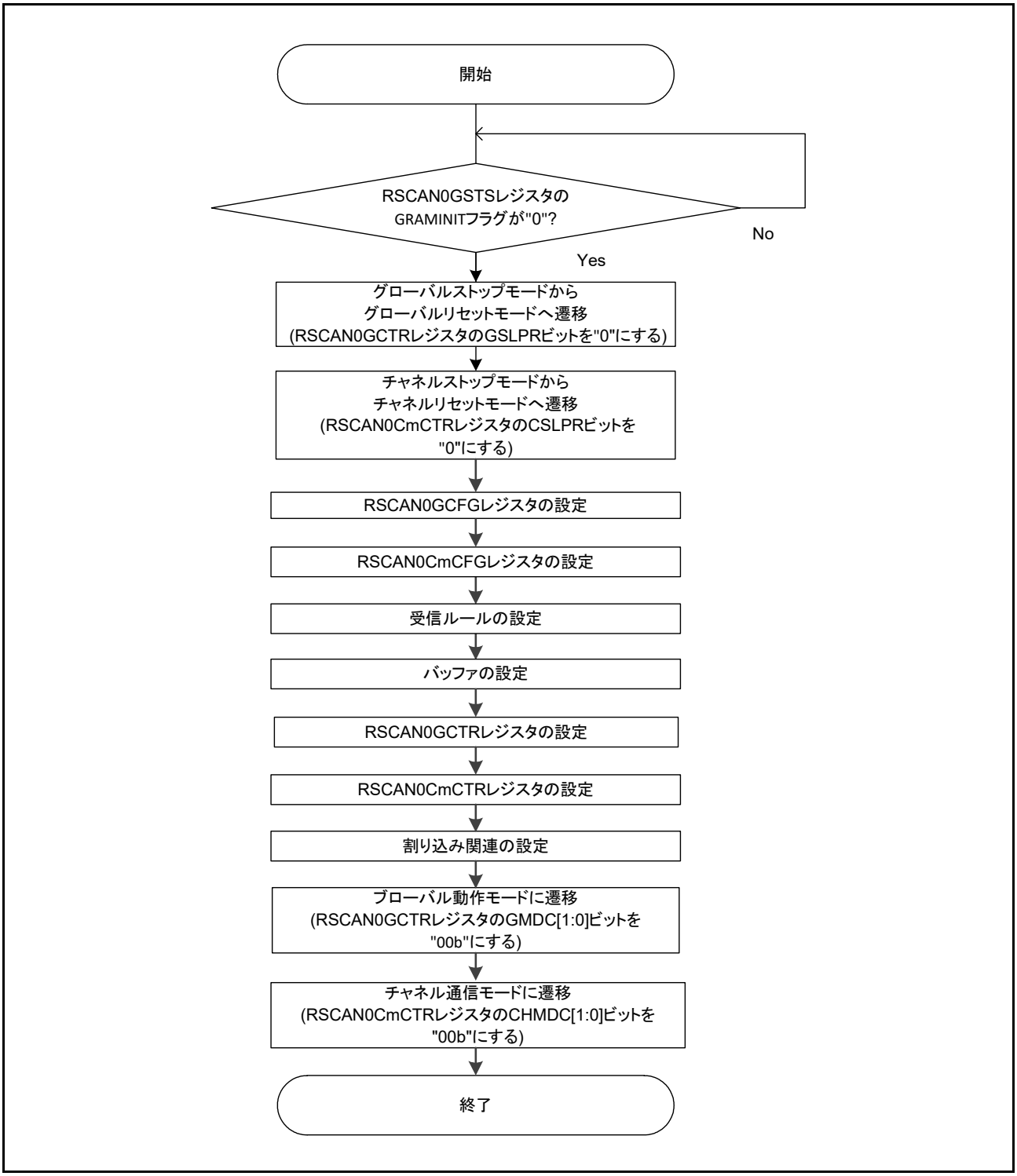


図 6.1　　MCU リセット後の CAN コンフィグレーション手順

6.2 CAN 状態（モード）遷移

CAN モジュールはチャネル全体（以下、グローバル）の状態とチャネルの状態（モード）を持ちます。
以下に CAN モジュールの持つ状態（モード）を示します。

- グローバル・モード
  - グローバル・ストップ・モード
  - グローバル・リセット・モード
  - グローバル・テスト・モード
  - グローバル動作モード

- チャネル・モード
  - チャネル・ストップ・モード
  - チャネル・リセット・モード
  - チャネル待機モード
  - チャネル通信モード

6.2.1 グローバル・モード

CAN モジュール全体のモードです。
図 6.2 にグローバル・モードの遷移図を示します。
なお、グローバル・モードの遷移により、チャネルのモードが変化する場合があります。

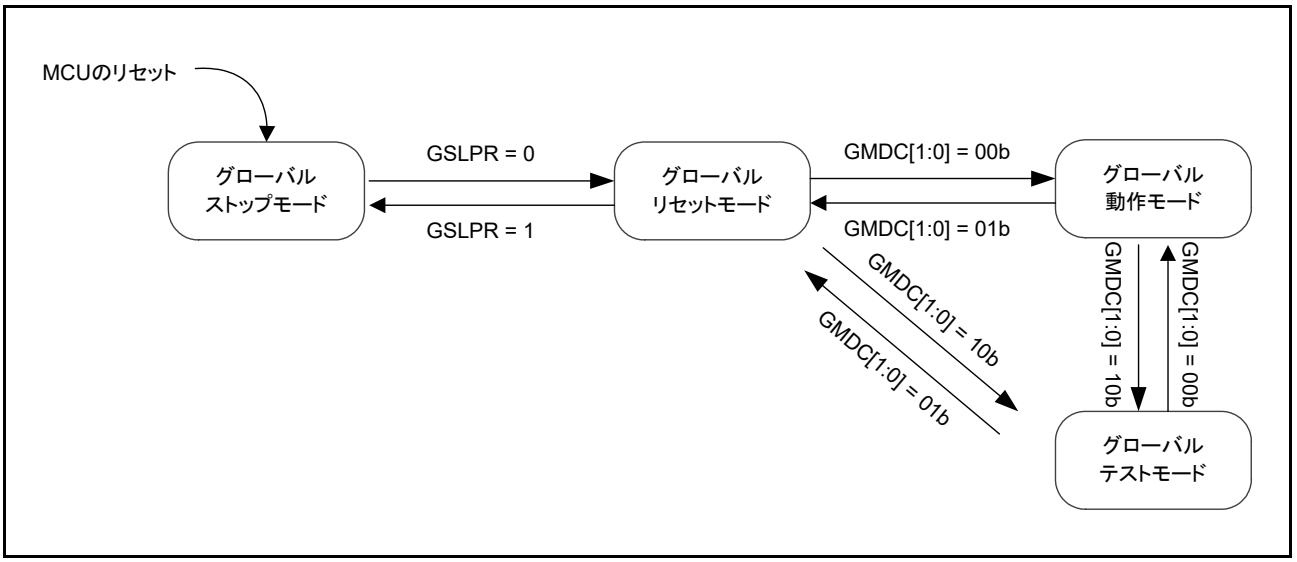


図 6.2 グローバル・モードの遷移図

(1) グローバル・ストップ・モード
CAN モジュールのクロックが停止するモードです。CAN のクロックが停止するため、消費電力が低減されます。CAN 関連レジスタの読み出しは可能ですが、書き込みはしないでください。レジスタ値は保持されます。

(2) グローバル・リセット・モード
CAN モジュール全体の設定を行うモードです。グローバル・リセット・モードに遷移すると、一部レジスタが初期化されます。

(3) グローバル・テスト・モード
テスト関連レジスタの設定を行うモードです。グローバル・テスト・モードに遷移すると、CAN 通信は停止します。

(4) グローバル動作モード
CAN モジュール全体を動作させるモードです。CAN 通信を行う場合は、グローバル動作モードに遷移する必要があります。

6.2.2 チャネル・モード

チャネルを制御するモードです。

図 6.3 にチャネル・モードの状態遷移図を示します。

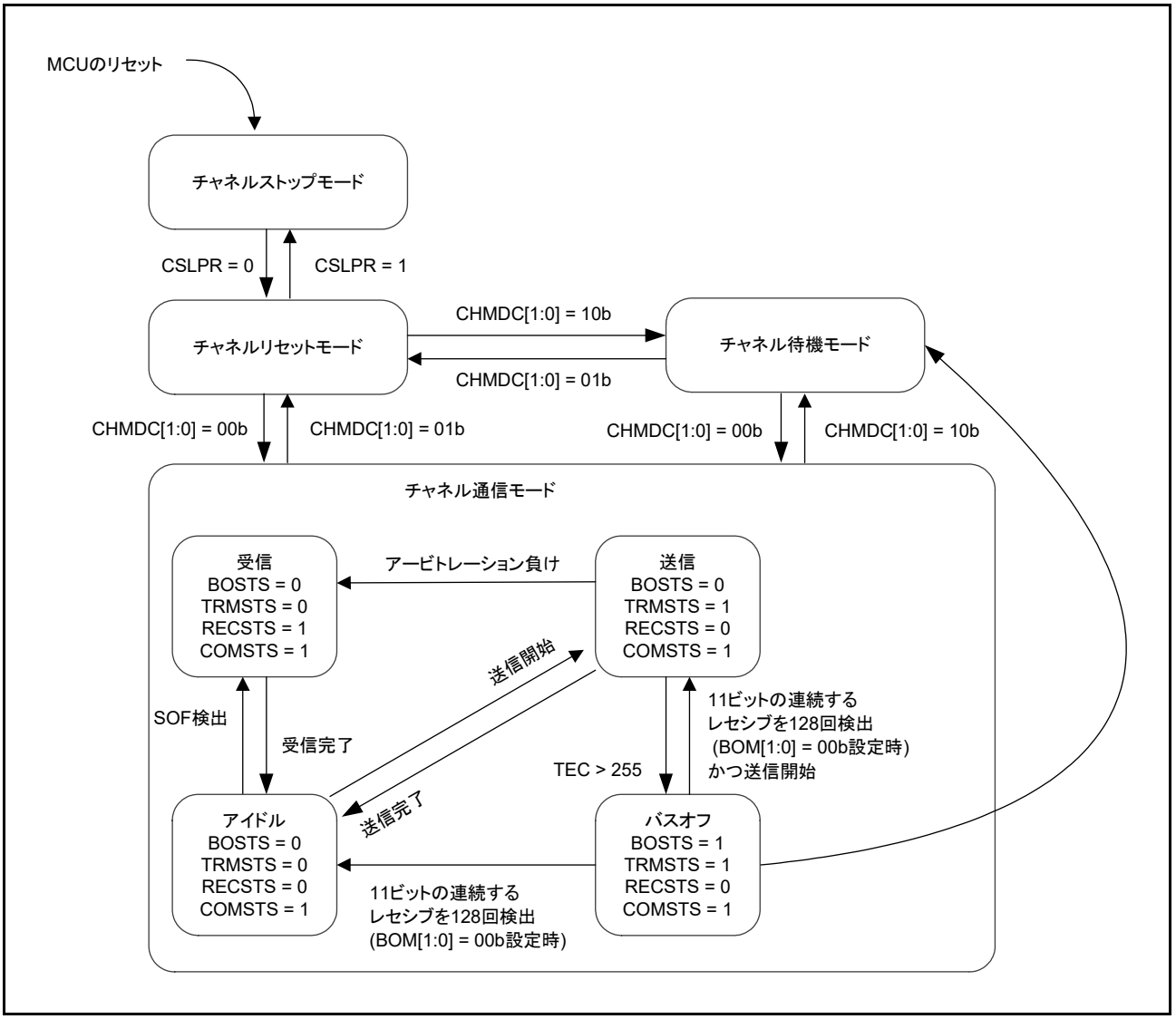


図 6.3 チャネル・モードの状態遷移図

(1) チャネル・ストップ・モード
チャネルに供給するクロックが停止するモードです。供給するクロックが停止するため消費電力が低減されます。該当チャンネルの CAN 関連レジスタの読み出しは可能ですが、書き込みはしないでください。レジスタ値は保持されます。

(2) チャネル・リセット・モード
チャネルの設定を行うモードです。チャネル・リセット・モードに遷移すると、一部チャネル関連レジス タが初期化されます。

(3) チャネル待機モード
チャネルのテスト関連レジスタの設定を行うモードです。チャネル待機モードに遷移すると、該当チャネルの CAN 通信は停止します。

(4) チャネル通信モード
CAN 通信を行うモードです。CAN 通信時、各チャネルは以下の通信状態を持ちます。

- アイドル
  受信も送信もしていない状態。
- 受信
  他のノードから送られてきたメッセージを受信している状態。
- 送信
  メッセージを送信している状態。
- バスオフ
  CAN 通信から遮断されている状態。

6.2.3 グローバル・モードの遷移によるチャネル・モードの変化

グローバル・モードの遷移により、チャネルのモードが変化する場合があります。
表 6.1 にグローバル・モード設定によるチャネル・モードの変化を示します。

表6.1 グローバル・モード設定によるチャネル・モードの変化

設定前の チャネル・モード	設定後のチャネル・モード			
	グローバル動作	グローバルテスト	グローバルリセット	グローバルストップ
チャネル通信	チャネル通信	チャネル待機	チャネルリセット	遷移禁止
チャネル待機	チャネル待機	チャネル待機	チャネルリセット	遷移禁止
チャネルリセット	チャネルリセット	チャネルリセット	チャネルリセット	チャネルストップ
チャネルストップ	チャネルストップ	チャネルストップ	チャネルストップ	チャネルストップ

6.3　通信速度

CAN 通信を行う通信速度を設定します。通信速度を決定するためには以下の処理を行う必要があります。

- ビット・タイミングの設定
- 通信速度の算出

6.3.1　CAN ビット・タイミングの設定

本 CAN モジュールの CAN ビット・タイミング設定では、通信フレームの 1 ビットを 3 つのセグメントで構成しています。

図 6.4 にビットのセグメントとサンプル・ポイントを示します。

これらのセグメントのうち、Time Segment 1(以下、TSEG1 という)、Time Segment 2(以下、TSEG2 という) は、サンプル・ポイントを指定するもので、これらの値を変えることでサンプリングするタイミングを変更できます。

このタイミング設定の最小単位を 1 Time Quantum(以下、Tq という) といい、CAN モジュールに入力された クロック周波数とボー・レート・プリスケーラ値で決められます。

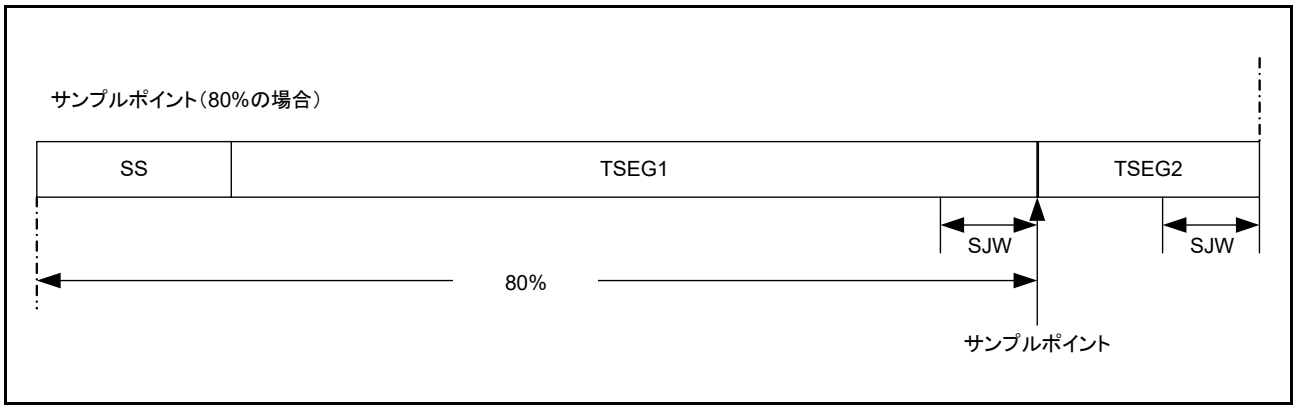


図 6.4　ビットのセグメントとサンプル・ポイント

上図の各セグメントについて、以下に説明します。

- SS：シンクロナイゼーション・セグメント
  インターフレーム・スペース中に、レセシブからドミナントへのエッジをモニタして同期をとるセグメントです。
  インターフレーム・スペース (Interframe Space) は、インターミッション (Intermission)、サスペンド・トランスミッション (Suspend Transmission)、バスアイドル (Bus Idle) で構成されます。バスアイドル中では、全ノードが送信を開始することができます。
- TSEG1：タイム・セグメント 1
  CAN バス上の物理的な遅延を吸収するセグメントです。バス上の物理的な遅延は、CAN バス上の遅延、入力コンパレータ遅延、および、出力ドライバ遅延の総和の 2 倍になります。
- TSEG2：タイム・セグメント 2
  周波数の誤差によるフェーズ・エラーを補償するセグメントです。
- SJW：リシンクロナイゼーション・ジャンプ幅
  フェーズ・エラーによっておこる位相誤差を補償するために、タイム・セグメントを延長または短縮する長さです。

(1) ビット・タイミングの条件
各セグメントの設定と制限事項は以下のとおりです。

- 各セグメントの設定
  - SS　　= 1Tq 固定
  - TSEG1　　= 4 ～ 16Tq の範囲で設定
  - TSEG2　　= 2 ～ 8Tq の範囲で設定
  - SJW　　= 1 ～ 4Tq の範囲で設定
  - SS+TSEG1+TSEG2 = 8 ～ 25Tq
- TSEG1、TSEG2 の制限
  - TSEG1 ＞ TSEG2 ≧ SJW（ただし、SJW=1 のとき TSEG2 ≧ 2）

6.3.2　通信速度の算出

通信速度は、CAN モジュールのクロック源である CAN クロック (f_{CAN})、ボー・レート・プリスケーラ分周値、および 1 ビットの Tq 数で決まります。f_{CAN} は CPU/ 周辺ハードウェア・クロックを 2 分周したクロックと X1 クロックのいずれかを使用可能です。

表 6.2 に主な通信速度の実現例と**表 6.3** にビット・タイミング設定例を示します。

表6.2　主な通信速度の実現例

fCAN / 通信速度	40 MHz	32 MHz	24 MHz	16 MHz	8 MHz
1 Mbps	8 Tq (5) 20 Tq (2)	8 Tq (4) 16 Tq (2)	8 Tq (3) 12 Tq (2) 24 Tq (1)	8 Tq (2) 16 Tq (1)	8 Tq (1)
500 Kbps	8 Tq (10) 20 Tq (4)	8 Tq (8) 16 Tq (4)	8 Tq (6) 12 Tq (4) 24 Tq (2)	8 Tq (4) 16 Tq (2)	8 Tq (2) 16 Tq (1)
250 Kbps	8 Tq (20) 20 Tq (8)	8 Tq (16) 16 Tq (8)	8 Tq (12) 12 Tq (8) 24 Tq (4)	8 Tq (8) 16 Tq (4)	8 Tq (4) 16 Tq (2)
125 Kbps	8 Tq (40) 20 Tq (16)	8 Tq (32) 16 Tq (16)	8 Tq (24) 12 Tq (16) 24 Tq (8)	8 Tq (16) 16 Tq (8)	8 Tq (8) 16 Tq (4)

注1.　（ ）内の数字はボーレートプリスケーラ分周値

表6.3　ビット・タイミング設定例

1ビット	設定値 (Tq)				サンプルポイント (%) * 図6.4を参照
	SS	TSEG1	TSEG2	SJW	
8 Tq	1	4	3	1	62.50
	1	5	2	1	75.00
10 Tq	1	6	3	1	70.00
	1	7	2	1	80.00
16 Tq	1	10	5	1	68.75
	1	11	4	1	75.00
20 Tq	1	12	7	1	65.00
	1	13	6	1	70.00

6.3.3 CAN ビット・タイミングと通信速度の設定手順

図 6.5 に CAN ビット・タイミングと通信速度の設定手順を示します。
これらの設定は CAN コンフィグレーション中に実施してください。

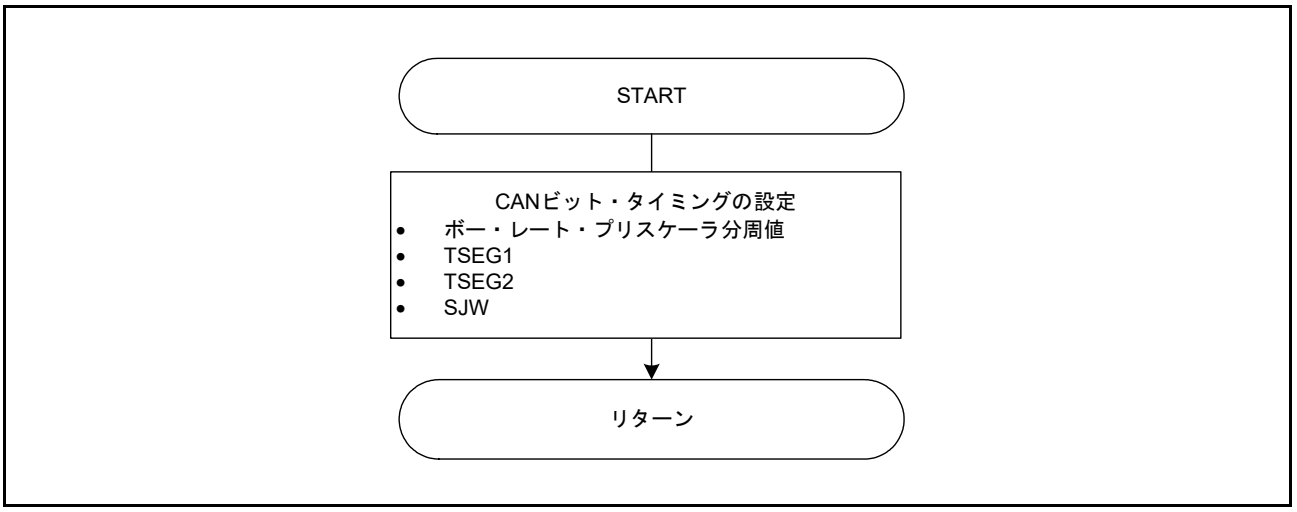


図 6.5 CAN ビットタイミングと通信速度の設定手順

6.4 グローバル機能

CAN モジュール全体（すべてのチャネル）で共通な以下の機能を設定します。

- 送信優先順位の設定
- DLC チェックの設定
- DLC 置換機能の設定
- ミラー機能の設定
- CAN クロック源の設定
- タイム・スタンプ・クロックの設定

6.4.1 送信優先順位の設定

同一チャネル内で複数の送信バッファから送信要求が出された場合の送信の優先順位を設定します。

送信優先順位はすべてのチャネルで共通で、チャネルごとに設定することはできません。判定方法は以下の 2 つから選択できます。

- ID 優先
  格納したメッセージ ID の優先順位に基づいてメッセージが送信されます。ID の優先順位は CAN 仕様に規定されている CAN バス・アービトレーション規定に準拠します。送信バッファ、送信モードに設定した送受信 FIFO バッファに格納したメッセージの ID が判定対象になります。送受信 FIFO バッファの場合は、FIFO 内の最も古いメッセージが優先順位判定の対象になります。メッセージが送受信 FIFO バッファから送信中の場合、同じ FIFO バッファにある次のメッセージが優先順位判定の対象になります。2 つ以上のバッファに同じメッセージ ID が設定されている場合は、より小さい番号のバッファが優先されます。
- 送信バッファ番号優先
  送信要求があるバッファの中で、最も小さい送信バッファ番号のメッセージが最初に送信されます。送受信 FIFO バッファが送信バッファにリンクしている場合は、リンク先の送信バッファ番号で判定されます。

どちらの送信優先順位を選択しても、アービトレーション・ロストまたはエラーが発生し、再送信される場合、送信の優先順位判定が再度実行されます

6.4.2 DLC チェックの設定

DLC チェック機能の許可、禁止を設定します。

DLC チェック機能を許可にすると、アクセプタンス・フィルタ処理を通過したメッセージに対して、DLC フィルタ処理を実施します。

DLC チェック機能を禁止にすると、アクセプタンス・フィルタ処理を通過したメッセージに対して、DLC フィルタ処理は実施されません。

DLC チェックでは、メッセージの DLC 値が受信ルールに設定した DLC 値以上の場合、DLC フィルタ処理を通過します。受信メッセージの DLC 値が受信ルールの DLC 値より小さい場合は、DLC フィルタ処理を通過しません。この場合、メッセージは受信バッファや FIFO バッファに格納されず、DLC エラーとなります。

6.4.3　DLC 置換機能の設定

DLC 置換機能の許可、禁止を設定します。

DLC 置換機能は DLC チェック機能が許可の場合のみ有効です。

DLC 置換機能を許可にしているときに、DLC フィルタ処理を通過した場合、受信メッセージの DLC 値の代わりに、受信ルールの DLC 値がバッファに格納されます。この場合、受信ルールの DLC 値を超えるデータ・バイトには H'00 が格納されます。

DLC 置換機能を禁止にしているときに、DLC フィルタ処理を通過した場合、受信メッセージの DLC 値がバッファに格納されます。この場合、受信メッセージのすべてのデータ・バイトがバッファに格納されます。

6.4.4　ミラー機能の設定

ミラー機能の許可、禁止を設定します。ミラー機能を許可にすると、自らが送信したメッセージを受信できます。

ミラー機能許可時、他の CAN ノードが送信したメッセージを受信するときは、ミラー機能未使用にした受信ルールがデータ処理に使用されます。自らが送信したメッセージを受信するときは、ミラー機能使用にした受信ルールがデータ処理に使用されます。

6.4.5　CAN クロック源の設定

CAN クロック源である CAN クロック (f_{CAN}) を設定します。CAN クロック源として使用可能なクロックを以下に示します。

- CPU/ 周辺ハードウェア・クロックを 2 分周したクロック
- X1 クロック

図 6.6 に CAN クロック発生回路を示します。

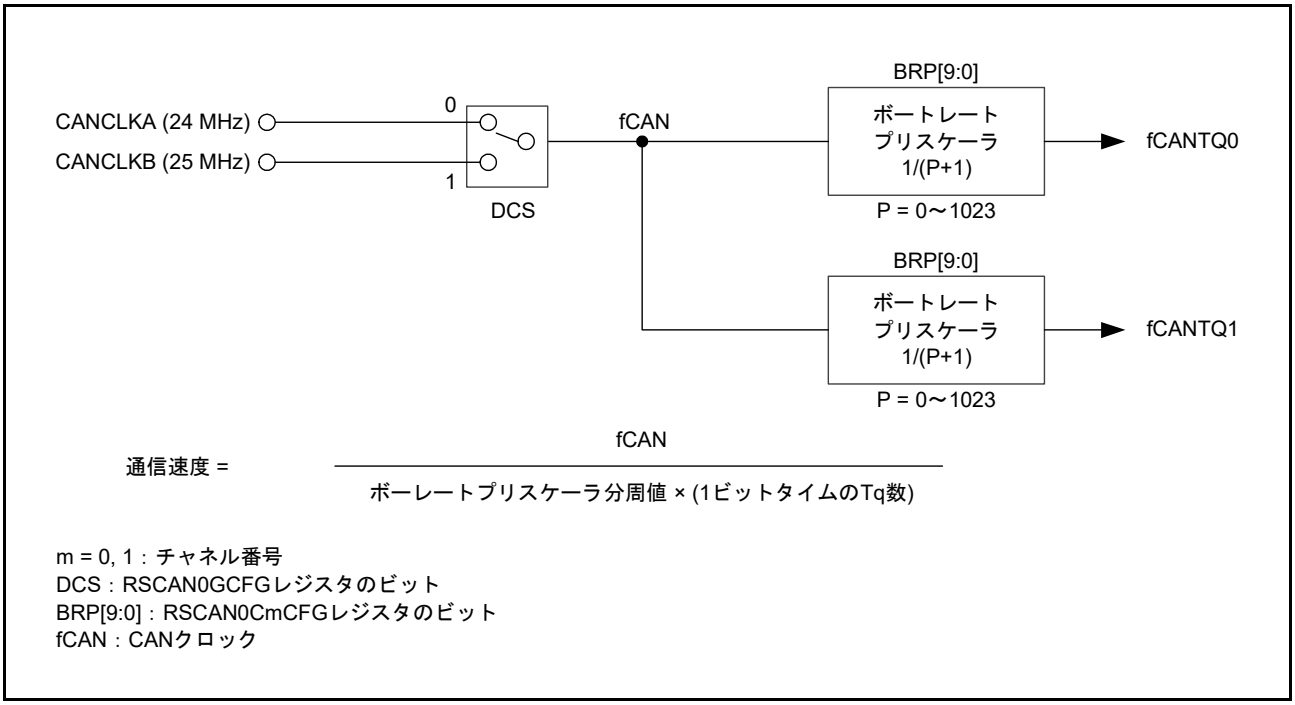


図 6.6　CAN クロック発生回路

6.4.6　タイム・スタンプ・クロックの設定

タイム・スタンプ・カウンタに使用するクロック源、および分周比を設定します。

タイム・スタンプは、メッセージの受信時間を記録するために使用する 16 ビットのフリーランカウンタです。タイム・スタンプ・カウンタ値は、メッセージの SOF(スタート・オブ・フレーム) 注 1 のタイミングで取り込まれ、メッセージ ID やデータと共に、受信バッファや FIFO バッファに格納されます。

タイム・スタンプに使用するクロックは以下から選択可能です。

- CPU/ 周辺ハードウェア・クロックを 2 分周したクロック
- CANi ビット・タイム・クロック

注 1.　スタート・オブ・フレーム (StartOfFrame) : フレームの開始を表すフィールド。

図 6.7 にタイム・スタンプ機能のブロック図を示します。

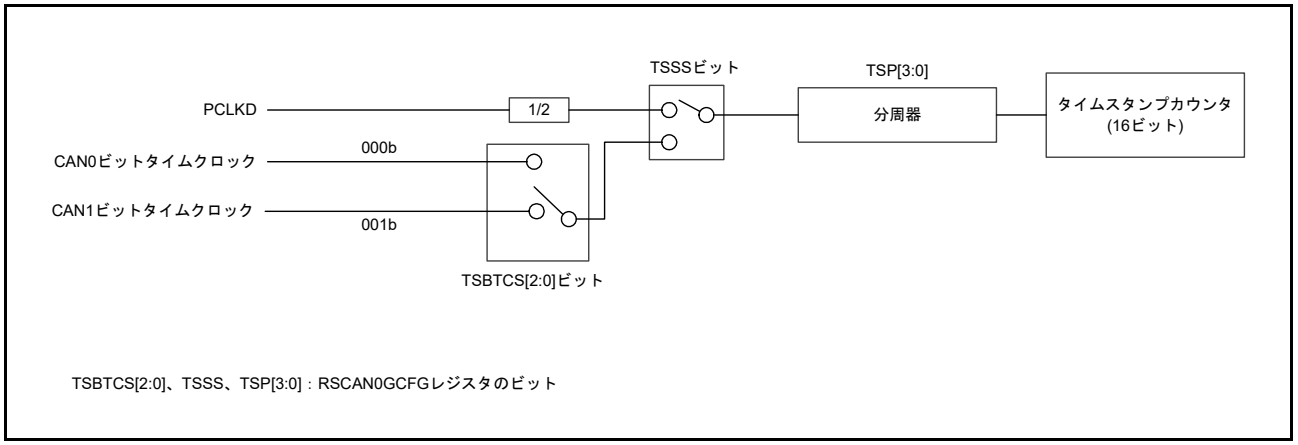


図 6.7　タイム・スタンプ機能のブロック図

6.4.7 グローバル機能の設定

図 6.8 にグローバル機能の設定手順を示します。
これらの設定は CAN コンフィグレーション中に実施してください。

図 6.8 グローバル機能の設定手順

6.5 受信ルール・テーブル

受信メッセージのフィルタリングを行うための受信ルール・テーブルを設定します。

受信ルール・テーブルを用いたデータ処理により、選別されたメッセージを指定したバッファへ格納します。データ処理には、アクセプタンス・フィルタ処理、DLC フィルタ処理、ルーティング処理、ラベル付加処理、ミラー機能があります。

受信ルールで以下の設定を行う必要があります。

- 受信ルール数の設定
- IDE/RTR/ID の設定
- 受信ルール対象メッセージの設定
- IDE マスク /RTR マスク /ID マスクの設定
- DLC チェックの設定
- 受信ルール・ラベルの設定
- 格納バッファの設定

6.5.1 受信ルール数の設定

チャネルに使用する受信ルール数を設定します。

1 ページに 16 の受信ルールを登録できます。

チェック処理は、一番小さい番号の受信ルールから昇順にチェックを開始します。受信メッセージの比較対象ビットが受信ルールとすべて一致したとき、または一致する受信ルールがないまますべてのチェックを終了したときにフィルタ処理は停止します。一致する受信ルールが無い場合、受信バッファや FIFO バッファに格納されません。

6.5.2 IDE/RTR/ID の設定

受信メッセージのID フォーマット（標準ID または拡張ID）、フレーム・フォーマット（データ・フレームまたはリモート・フレーム）、受信ID を設定します。

6.5.3 受信ルール対象メッセージの設定

他の CAN ノードが送信したメッセージ（RSCAN0GAFLIDj レジスタの GAFLLB ビットを "0"）にすると、他の CAN ノードが送信したメッセージを受信する場合に、受信ルールを用いたデータ処理を行います。

ミラー機能使用時に自らが送信したメッセージ（GAFLLB ビットを "1"）にすると、自らが送信したメッセージを受信する場合に、受信ルールを用いたデータ処理を行います。

6.5.4 IDE マスク /RTR マスク /ID マスクの設定

IDE/RTR/ID で設定した値のマスク値を設定します。

IDE マスク /RTR マスク /ID マスクでマスクされなかったビットがアクセプタンス・フィルタ処理で有効になります。

6.5.5 DLC チェックの設定

DLC チェック許可時に受信メッセージの DLC 値と比較する受信ルールの DLC 値を設定します。

6.5.6 受信ルール・ラベルの設定

フィルタ処理を通過したメッセージをバッファに格納するときに付加される 12 ビットのラベル情報を設定します。

ラベルは任意に設定できます。また、受信メッセージのラベルはプログラムで自由に使用することができます。例えば、ラベルに受信するチャネル番号を設定しておけば、受信 FIFO バッファ内の同一 ID のメッセージがどのチャネルで受信されたかを確認することが可能です。

6.5.7 格納バッファの設定

DLC フィルタ処理を通過したメッセージを格納するバッファを設定します。

格納先として選択可能なバッファを以下に示します。

- 受信バッファ n（1 つの受信ルールに対して、1 バッファだけ選択可能）
- 受信 FIFO バッファ m
- 送受信 FIFO バッファ k（受信モード）

1 つの受信ルールに対して、最大 2 バッファまで格納バッファを選択できます。ただし、格納先として受信バッファは 1 バッファしか選択できません。

6.5.8 受信ルールの使用例

受信ルールの使用例を以下に示します。

- 使用例 1

  以下のメッセージを受信する場合の各レジスタの例を示します。

  - ID フォーマット : 標準 ID
  - メッセージフォーマット : データ・フレーム
  - ミラー機能 : 他の CAN ノードのメッセージ受信
  - 受信 ID : 120h / 121h / 122h / 123h
  - DLC : 受信メッセージの DLC ≧ 6
  - ラベル : 010h
  - 格納先バッファ : 受信バッファ 3、受信 FIFO バッファ 0、1

受信ルール ID レジスタ (RSCAN0GAFLIDj)

受信可能メッセージ	GAFLIDE	GAFLRTR	GAFLLB	GAFLID[28:0]
120h	0	0	0	B'- --- ---- ---- ---- -001 0010 0000
121h				B'- --- ---- ---- ---- -001 0010 0001
122h				B'- --- ---- ---- ---- -001 0010 0010
123h				B'- --- ---- ---- ---- -001 0010 0011

受信ルールマスクレジスタ (RSCAN0GAFLMj)

GAFLIDEM	GAFLRTRM	GAFLIDM[28:0]
1	1	B'0 0000 0000 0000 0000 0111 1111 1100

受信ルールポインタ 0 レジスタ (RSCAN0GAFLP0j)

GAFLDLC[3:0]	GAFLPTR[11:0]	GAFLRMV	GAFLRMDP[6:0]
6	010h	1	3

受信ルールポインタ 1 レジスタ (RSCAN0GAFLP1j)

RSCAN0GAFLP1j[17:0]	GAFLFDPr[7:0]
B'00 0000 0000 0000 0000	B'0000 0011

- 使用例 2

  以下のメッセージを受信する場合の各レジスタの例を示します。

  - ID フォーマット　　：標準 ID
  - メッセージフォーマット　　：リモート・フレーム、データ・フレーム
  - ミラー機能　　：他の CAN ノードのメッセージ受信
  - 受信 ID　　：130h
  - DLC　　：DLC チェック未使用
  - ラベル　　：130h
  - 格納先バッファ　　：受信 FIFO バッファ 0、送受信 FIFO バッファ 0

受信ルール ID レジスタ (RSCAN0GAFLIDj)

受信可能メッセージ	GAFLIDE	GAFLRTR	GAFLLB	GAFLID[28:0]
130h （データ）	0	0	0	B'- --- ---- ---- ---- -001 0011 0000
130h （リモート）	0	1	0	B'- --- ---- ---- ---- -001 0011 0000

受信ルールマスクレジスタ (RSCAN0GAFLMj)

GAFLIDEM	GAFLRTRM	GAFLIDM[28:0]
1	0	B'0 0000 0000 0000 0000 0111 1111 1111

受信ルールポインタ 0 レジスタ (RSCAN0GAFLP0j)

GAFLDLC[3:0]	GAFLPTR[11:0]	GAFLRMV	GAFLRMDP[6:0]
0	130h	0	0

受信ルールポインタ 1 レジスタ (RSCAN0GAFLP1j)

RSCAN0GAFLP1j[17:0]	GAFLFDPr[7:0]
B'00 0000 0000 0000 0001	B'0000 0001

6.5.9　受信ルールテーブルの設定手順

図 6.9 に受信ルールテーブルの設定手順を示します。
これらの設定は CAN コンフィグレーション中に実施してください。

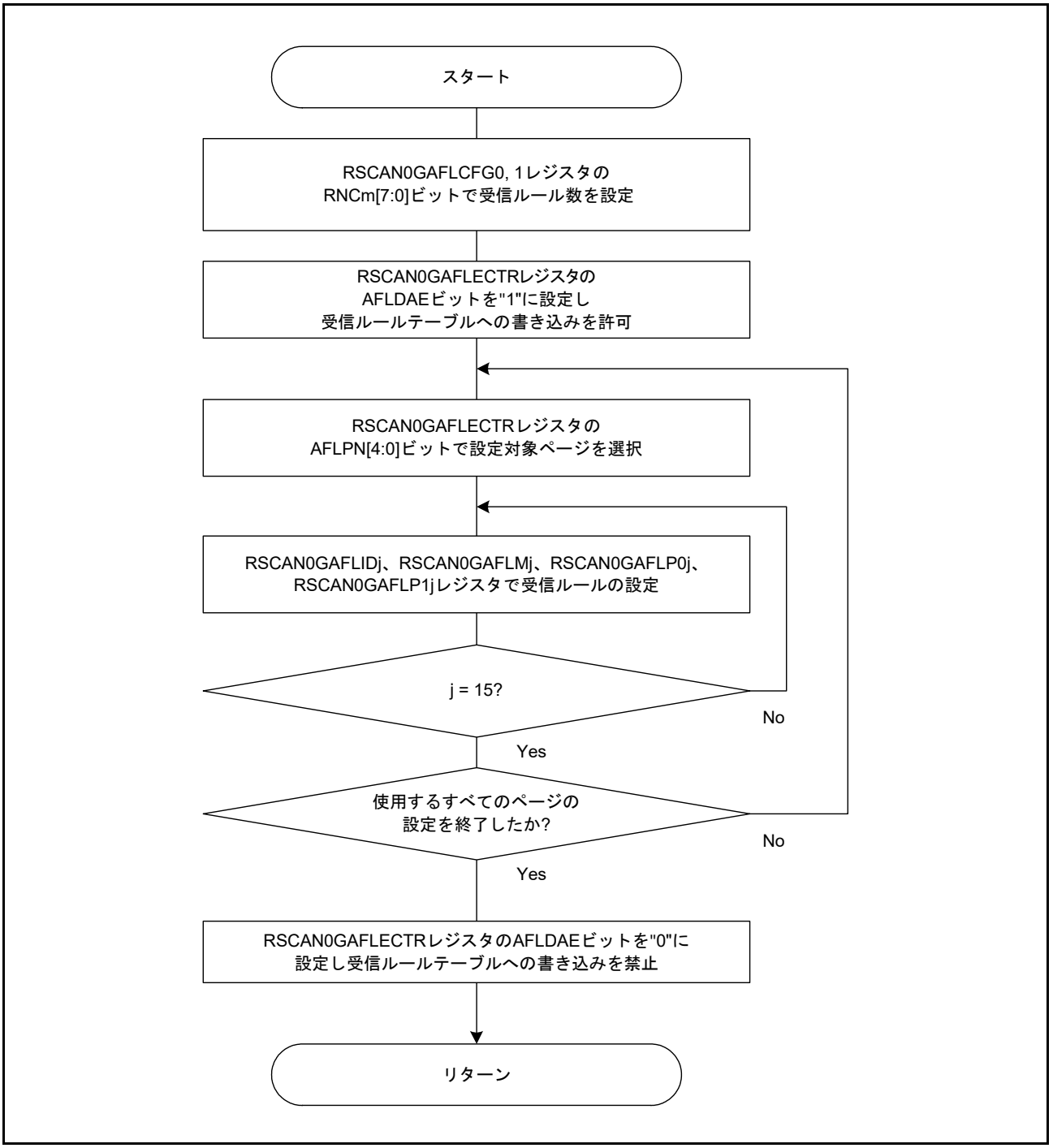


図 6.9　受信ルール・テーブルの設定手順

6.6　バッファ、FIFO バッファ

送受信で使用するバッファ、FIFO バッファの設定を行います。以下のバッファ、FIFO バッファを設定する必要があります。

- 受信バッファの設定
- 受信 FIFO バッファの設定
- 送受信 FIFO バッファの設定
- 送信バッファの設定
- 送信履歴バッファの設定

図 6.10 に各種バッファの構成を示します。

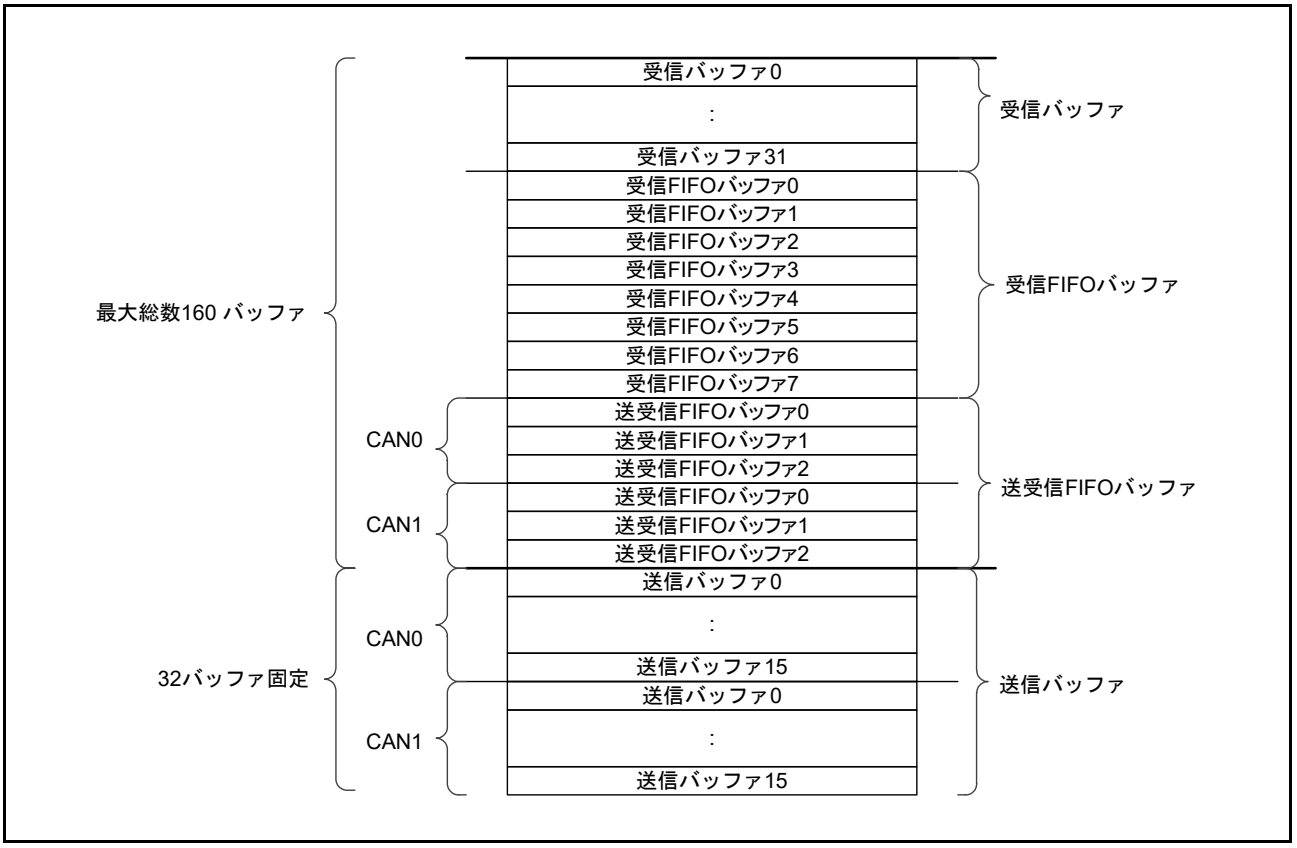


図 6.10　バッファの構成

6.6.1 受信バッファの設定

受信バッファへ割り当てるバッファ数を設定します。受信バッファへは 0 ～ 31 バッファを割り当てられます。受信バッファ数に "0" を設定すると、受信バッファは使用できません。

受信バッファ関連の割り込みはないため、割り込み関連の設定はありません。

6.6.2 受信 FIFO バッファの設定

受信 FIFO バッファを使用するために必要な設定を以下に示します。

- バッファ数の設定
- 割り込み許可 / 禁止、割り込み要因の設定

(1) バッファ数の設定

受信 FIFO バッファへ割り当てるバッファ数を設定します。

受信 FIFO バッファは 8 本あります。

受信 FIFO バッファを使用しない場合は、受信 FIFO バッファコンフィグレーション／制御レジスタ（RSCAN0RFCCx）の RFE（受信 FIFO バッファ許可ビット）ビットを "0"、および RFDC[2:0]（受信 FIFO バッファ段数設定ビット）を "000" に設定してください。

(2) 割り込み許可 / 禁止、割り込み要因の設定

受信 FIFO 割り込み許可 / 禁止の設定、割り込み要因を設定します。受信 FIFO 割り込みを使用する場合、割 り込み要因は以下から選択できます。

- 受信 FIFO バッファコンフィグレーション／制御レジスタ（RSCAN0RFCCx）の RFIGCV[2:0] ビットで設定した条件に達したときに受信 FIFO 割り込み発生（RSCAN0RFCCx レジスタの RFIM ビットが "0"）
- 1 メッセージ受信が完了するごとに受信 FIFO 割り込み発生（RSCAN0RFCCx レジスタの RFIM ビットが "1"）

6.6.3　送受信 FIFO バッファの設定

送受信 FIFO を使用するために必要な設定を以下に示します。

- バッファ数の設定
- 割り込み許可 / 禁止、割り込み要因の設定
- 送受信 FIFO モードの設定
- インターバル・タイマ・カウンタの設定（送信モード）
- 送信バッファ・リンクの設定（送信モード）

(1) バッファ数の設定

送受信 FIFO バッファのバッファ数を設定します。
各チャネルに 3 本あります。（CAN0：0 ～ 2、CAN1：3 ～ 5）
送受信 FIFO バッファを使用しない場合は、送受信 FIFO バッファコンフィグレーション／制御レジスタ（RSCAN0CFCCk）の CFE（送受信 FIFO バッファ許可ビット）ビットを "0"、および CFDC[2:0]（送受信 FIFO バッファ段数設定ビット）を "000" に設定してください。

(2) 割り込み許可 / 禁止、割り込み要因の設定

各送受信 FIFO バッファの割り込み許可 / 禁止の設定、割り込み要因を設定します。
送受信 FIFO モードごと に設定できる割り込み要因を以下に示します。

送受信FIFOモード	CFIMビット	
受信モード	0	受信メッセージ数がCFIGCV[2:0]ビットで設定した条件に達したとき、FIFO受信割り込み要求発生
	1	1メッセージ受信ごとにFIFO受信割り込み要求発生
送信モード	0	メッセージ送信完了によってバッファが空になったとき、FIFO送信割り込み要求発生
	1	1メッセージ送信が完了するごとにFIFO送信割り込み要求発生

なお、送受信 FIFO 送信割り込みは CANi 送信割り込みの発生要因となります。CANi 送信割り込み発生要因を以下に示します。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

(3) 送受信 FIFO モードの設定

送受信 FIFO バッファのモードを設定します。受信モード、送信モードのいずれかに設定できます。

- 受信モード
  受信 FIFO として動作します。
- 送信モード
  送信 FIFO として動作します。

(4) インターバル・タイマ・カウンタの設定（送信モード）

インターバル・タイマ・カウンタのカウント・ソース、送信間隔を設定します。インターバル・タイマ・カウンタは送信モードで有効です。

(5) 送信バッファ・リンクの設定（送信モード）
送受信 FIFO バッファを送信バッファにリンクさせます。送信バッファへのリンクは送信モードのみ有効です。

6.6.4 送信バッファの設定

各送信バッファの送信完了割り込み許可 / 禁止を設定します。
送信バッファは 1 チャネルにつき 16 バッファあり、送信バッファ、送受信 FIFO バッファ（送信モード）への リンク用のいずれかで使用することが可能です。

なお、送信完了割り込みは CANi 送信割り込みの要因となります。CANi 送信割り込み発生要因を以下に示します。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

6.6.5 送信履歴バッファの設定

送信履歴バッファを使用するために必要な設定を以下に示します。送信履歴バッファは 1 チャネルにつき 16 個の送信履歴データを格納できます。

- 格納対象バッファの設定
- 割り込み許可 / 禁止、割り込み要因の設定

(1) 格納対象バッファの設定
送信履歴バッファへ送信履歴データを格納する対象（送信元）バッファを設定します。格納する対象バッファは以下から選択できます。
また、メッセージを送信する際に、そのメッセージの送信履歴データを格納するかどうかを設定できます。

- 送受信 FIFO バッファ
- 送信バッファ、送受信 FIFO バッファ

(2) 割り込み許可 / 禁止、割り込み要因の設定
送信履歴割り込み許可 / 禁止の設定、割り込み要因を設定します。送信履歴バッファ割り込み要因を以下に示します。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

6.6.6　バッファの設定手順

図 6.11 に受信バッファ、受信 FIFO バッファの設定手順を、図 6.12 に送受信 FIFO バッファ、送信バッファ、送信履歴バッファの設定手順を示します。

これらの設定は CAN コンフィグレーション中に実施してください。

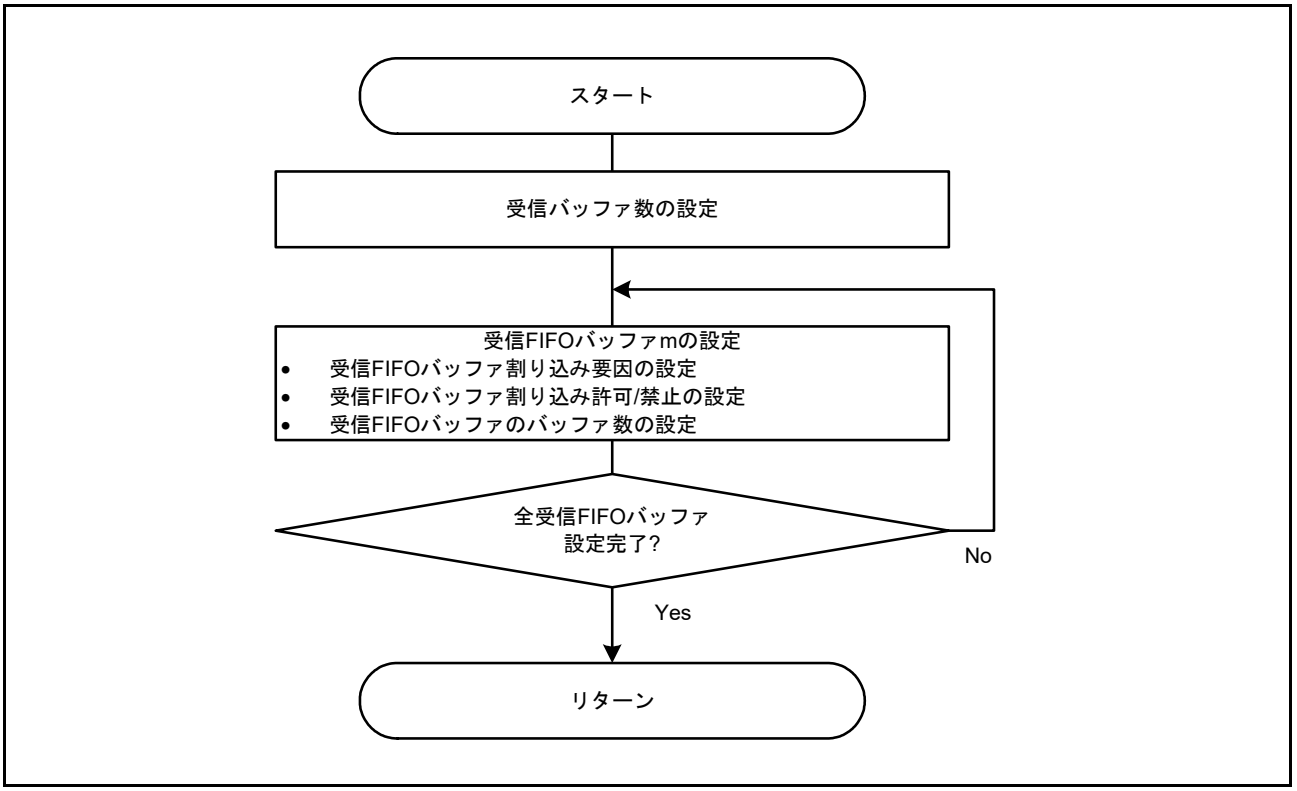


図 6.11　受信バッファ、受信 FIFO バッファの設定手順

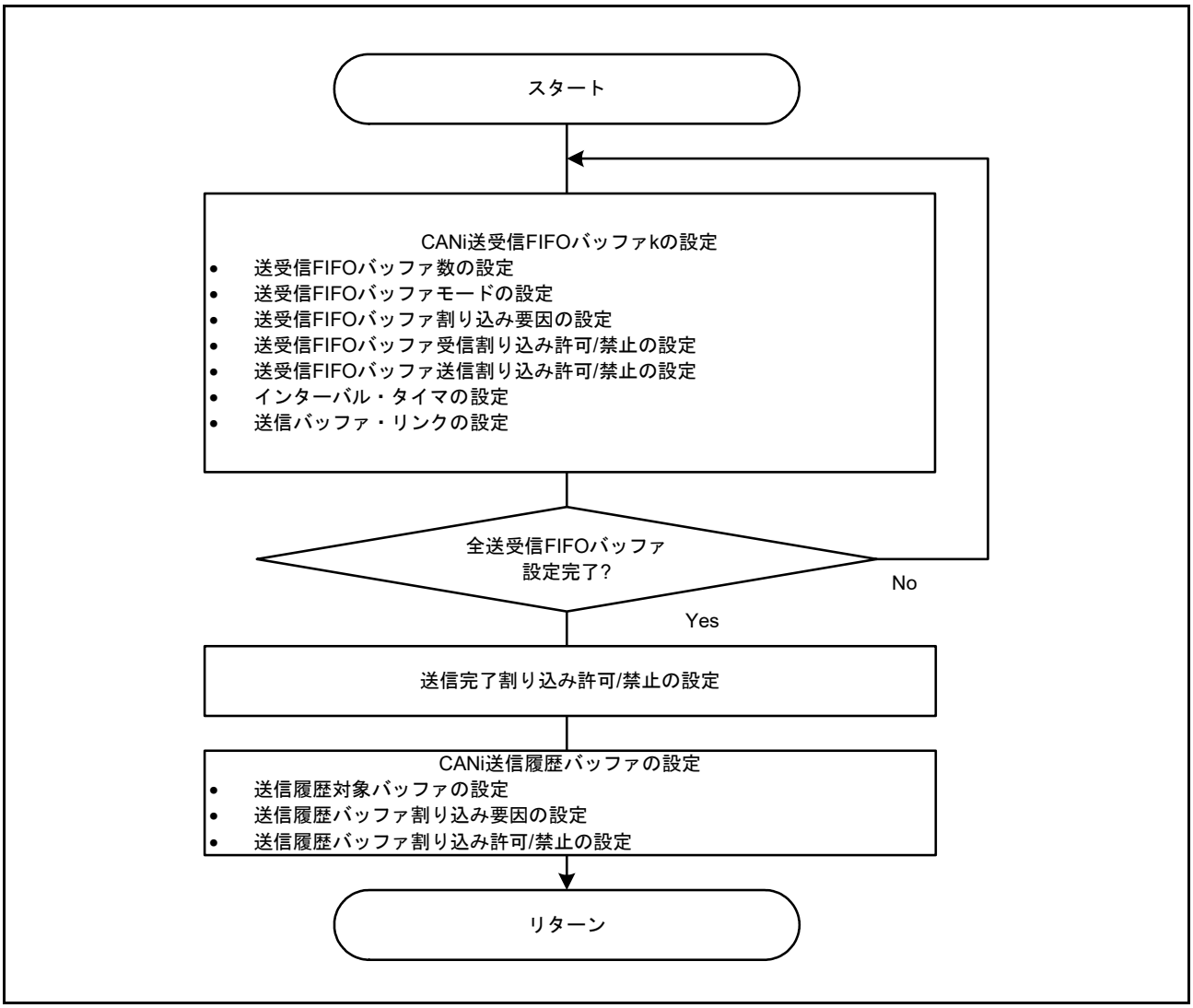


図 6.12　送受信 FIFO バッファ、送信バッファ、送信履歴バッファの設定手順

6.7 グローバル・エラー割り込み

グローバル・エラー割り込みの設定を行います。対応する割り込み許可ビットを許可にしているとき、CAN モジュールから割り込み要求が出力されます。割り込みの発生は、割り込みコントローラの割り込み制御レジスタの設定にも依存します。

6.7.1 グローバルエラー割り込みの設定

グローバル・エラー割り込みの発生要因を以下に示します。

- DLC チェック・エラー
- FIFO メッセージ・ロスト
- 送信履歴バッファ・オーバフロー

(1) DLC チェック・エラー

DLC チェック許可時に、アクセプタンス・フィルタ処理通過後の DLC チェックで、受信メッセージの DLC が受信ルールの DLC よりも小さい場合に検出されます。

(2) FIFO メッセージ・ロスト

受信 FIFO バッファ、送受信 FIFO バッファが FIFO フルの状態で、更に新しい受信メッセージを FIFO に格納しようとした場合に検出されます。

(3) 送信履歴バッファ・オーバフロー

送信履歴バッファがフルの状態で、更に新しい送信履歴データを送信履歴バッファに格納しようとした場合に検出されます。

6.7.2　グローバル・エラー割り込みの設定手順

図 6.13 にグローバル・エラー割り込みの設定手順を示します。
これらの設定は CAN コンフィグレーション中に実施してください。

図 6.13　グローバル・エラー割り込みの設定手順

6.8 チャネル機能

チャネルごとに持つ以下の機能の設定を行います。

- チャネル・エラー割り込みの設定
- 送信アボート割り込みの設定
- バスオフ復帰モードの設定
- エラー表示モードの設定
- 通信テストモードの設定

6.8.1 CANi エラー割り込み

CANi エラー割り込みの許可、禁止の設定を行います。チャネル・エラー割り込みの発生要因を以下に示します。

- バス・エラー
- エラー・ワーニング
- エラー・パッシブ
- バスオフ開始
- バスオフ復帰
- オーバロード・フレーム送信
- バス・ロック
- アービトレーション・ロスト

(1) バス・エラー

以下のいずれか 1 つでも検出した場合に割り込みが発生します。

- ACK デリミタでフォーム・エラーを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の ADERR ビットが "1"
- ドミナントを送信したにもかかわらず、レセシブを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の B0ERR が "1"
- レセシブを送信したにもかかわらず、ドミナントを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の B1ERR ビットが "1"
- CRC エラーを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の CERR ビットが "1"
- ACK エラーを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の AERR ビットが "1"
- フォーム・エラーを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の FERR ビットが "1"
- スタッフ・エラーを検出
  チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）の SERR ビットが "1"

(2) エラー・ワーニング

エラーワーニング状態（受信エラー・カウンタまたは送信エラー・カウンタ＞ 95）を検出した場合に割り込みが発生します。受信エラー・カウンタまたは送信エラー・カウンタが最初に 95 を超えたときのみ割り込みが発生します。

(3) エラー・パッシブ
エラー・パッシブ状態 (受信エラー・カウンタまたは送信エラー・カウンタ＞ 127) を検出した場合に割り込 みが発生します。受信エラー・カウンタまたは送信エラー・カウンタが最初に 127 を超えたときのみ割り込みが発生します。

(4) バスオフ開始
バスオフ状態 (送信エラー・カウンタ＞ 255) を検出した場合に割り込みが発生します。バスオフ復帰モードの設定が、バスオフ状態でチャネル待機モードへ遷移でバスオフ状態になった場合も割り込みが発生します。

(5) バスオフ復帰
11 ビットの連続するレセシブを 128 回検出してバスオフ状態からの復帰を検出した場合に割り込みが発生します。

(6) オーバロード・フレーム送信
受信または送信を行う場合に、オーバロード・フレームの送信条件を検出した場合に割り込みが発生しま す。

(7) バス・ロック
バス・ロックを検出した場合に割り込みが発生します。
チャネル通信モード時に CAN バス上に 32 ビットの連続するドミナントを検出するとバス・ロックと判断します。

(8) アービトレーション・ロスト
アービトレーション・ロストを検出した場合に割り込みが発生します。

6.8.2 CANi 送信アボート割り込み

送信アボート割り込みの許可、禁止の設定を行います。送信アボート割り込みを許可に設定している場合、送信アボート完了を検出したときに割り込みが発生します。

なお、送信アボート割り込みは CANi 送信割り込みの発生要因となります。CANi 送信割り込み発生要因を以下に示します。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

6.8.3 バスオフ復帰モードの設定

バスオフ復帰時の動作を設定します。

チャネル制御レジスタ（RSCAN0CmCTR）の BOM[1:0] ビット設定は以下のとおりです。

- 0 0 : ISO11898-1 仕様準拠
- 0 1 : バスオフ開始でチャネル待機モードへ遷移
- 1 0 : バスオフ終了でチャネル待機モードへ遷移
- 1 1 : バスオフ中にプログラムによる要求でチャネル待機モードへ遷移

6.8.4 エラー表示モードの設定

CAN バス・エラーが発生した場合に、チャネル・エラー・フラグレジスタ（RSCAN0CmERFL）のビット 14 ～ 8 にエラーが表示されます。その表示方は以下のとおり設定します。

- 最初に発生したエラー情報のみ表示（RSCAN0CmCTR レジスタの ERRD ビット "0"）
  最初に発生したエラー・フラグのみが "1" になります。同時に複数のエラーが発生した場合、検出された複数のエラーのフラグはすべて "1" になります。
- 発生したすべてのエラー情報を表示（RSCAN0CmCTR レジスタの ERRD ビット "1"）
  発生順に関係なく、発生したエラーのフラグはすべて "1" になります。

6.8.5 通信テストモードの設定

通信テストモードを設定します。テスト機能を使用することで、CAN トランシーバや MCU による CAN 通信の自己診断テスト、RAM の自己診断テストを行うことができます。

6.8.6 チャネル機能の設定手順

図 6.14 にチャネル機能の設定手順を示します。
これらの設定は CAN コンフィグレーション中に実施してください。

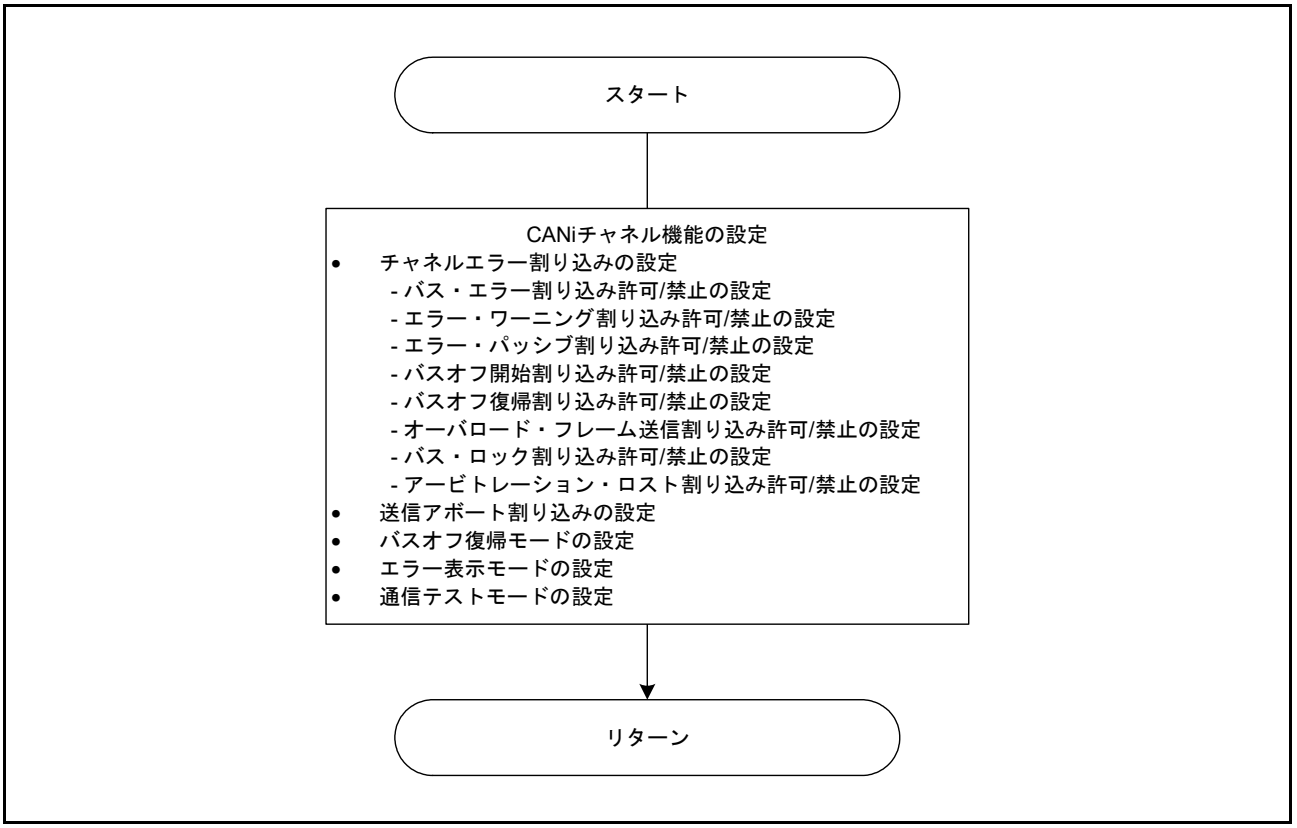


図 6.14 チャネル機能の設定手順

6.9　各状態で実施するCANコンフィグレーション処理

以下に各状態で実施するコンフィグレーション処理を示します。

注．　○：設定必要　×：設定不要　－：設定不可

6.9.1　CAN状態（モード）遷移

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
グローバル・モード遷移	○	○	—	—
チャネル・モード遷移	○	○	○	○

6.9.2　グローバル機能の設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
送信優先順序の設定	○	×	—	—
DLCチェックの設定	○	×	—	—
DLC置換え機能の設定	○	×	—	—
ミラー機能の設定	○	×	—	—
クロックの設定	○	×	—	—
タイム・スタンプ・クロックの設定	○	×	—	—
インターバルタイマプリスケーラの設定	○	×	—	—

6.9.3　通信速度の設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
ビット・タイミングの設定	○	×	×	×
通信速度の設定	○	×	×	×

6.9.4　受信ルール・テーブルの設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
受信ルール・テーブルの設定	○	×	—	—

6.9.5 バッファの設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
受信バッファの設定	○	×	—	—
受信FIFOバッファの設定	○	×	—	—
送受信FIFOバッファの設定	○	×	×	×
送信バッファの設定	○	×	×	×
送信履歴バッファの設定	○	×	×	×

6.9.6 グローバル・エラー割り込みの設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
グローバル・エラー割り込みの設定	○	×	—	—

6.9.7 チャネルの設定

処理	CANコンフィグレーション			
	MCU リセット後	グローバル リセット・モード後	チャネル リセット・モード後	チャネル 待機モード後
チャネル機能の設定	○	×	×	×

7. 受信

7.1 受信機能

CAN メッセージ受信を行う場合に使用可能な機能を以下に示します。各処理の詳細については次章以降を参照してください。

- 受信バッファを用いた受信
- 受信 FIFO バッファを用いた受信
- 送受信 FIFO バッファを用いた受信

7.2 受信バッファを用いた受信

全チャネルで共有する受信バッファは、0 ～ n+1 の範囲で使用できます。同じ番号の受信バッファに格納するメッセージは上書きされるため、最新の受信データが読み出せます。

受信バッファで受信した場合、割り込みは発生しません。

受信したメッセージを受信バッファに格納する処理が始まると、受信バッファ n が新しいメッセージありになります。(受信バッファ新データレジスタ 0（RSCAN0RMND0）の RMNS ビットが "1"）

受信バッファ ID レジスタ（RSCAN0RMIDq)、受信バッファポインタレジスタ（RSCAN0RMPTRq)、受信バッファデータフィールド 0 レジスタ（RSCAN0RMDF0q)、受信バッファデータフィールド 1 レジスタ（RSCAN0RMDF1q）から読み出せます。

7.2.1　受信バッファの読み出し手順

図 7.1 に受信バッファの読み出し手順を示します。

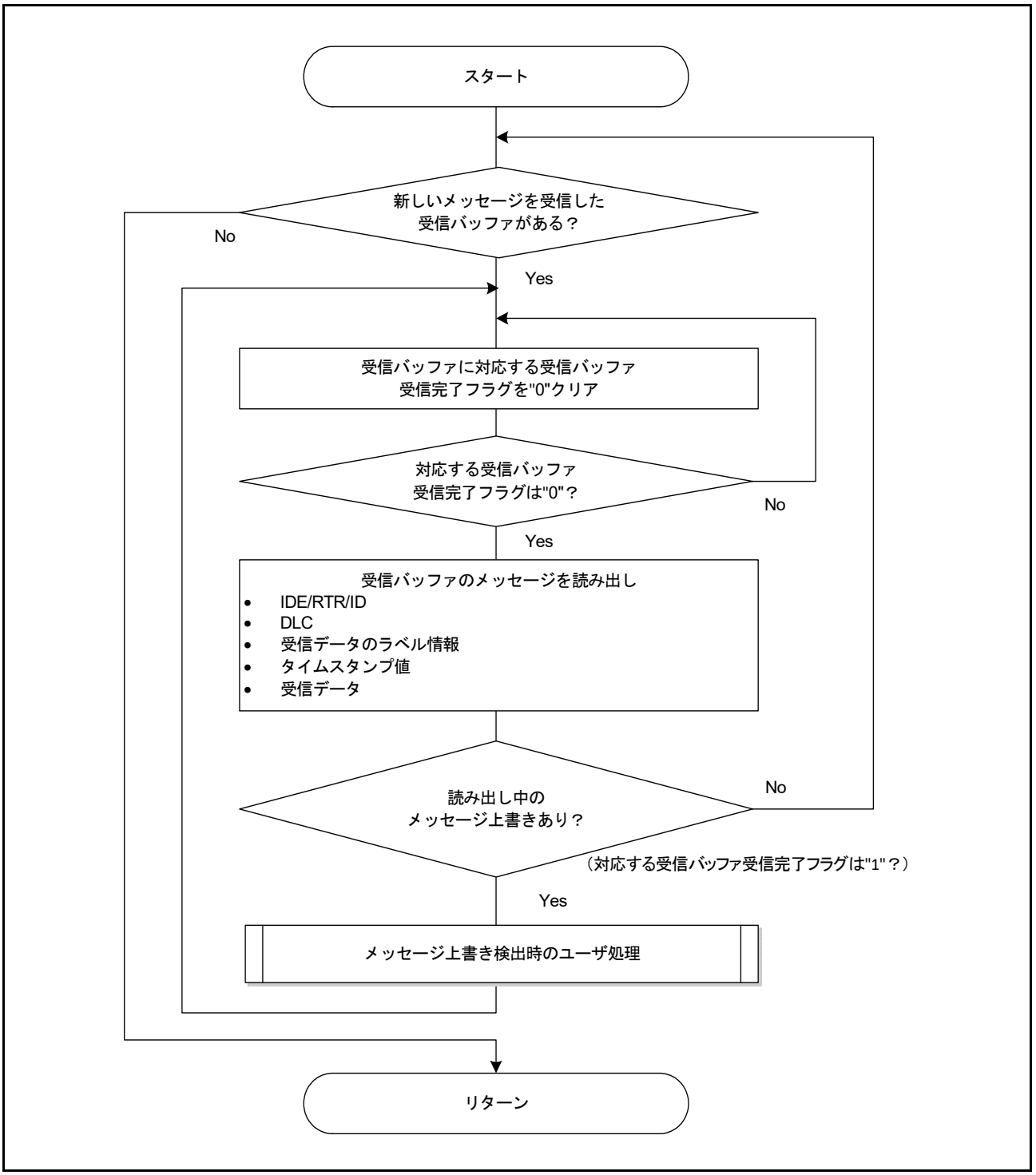


図 7.1　受信バッファの読み出し手順

7.3 受信 FIFO バッファを用いた受信

全チャネルで共有する受信 FIFO バッファが 8 本あります。各受信 FIFO バッファにバッファ数分メッセージを保存することができます。

受信メッセージが受信 FIFO バッファへ格納されると、対応するメッセージ数表示カウンタ（受信 FIFO バッファステータスレジスタ（RSCAN0RFSTSx）の RFMC[7:0]）の値がインクリメントされます。

受信メッセージは、受信 FIFO バッファアクセス ID レジスタ（RSCAN0RFIDx）、受信 FIFO バッファアクセスポインタレジスタ（RSCAN0RFPTRx）、受信 FIFO バッファアクセスデータフィールド 0 レジスタ（RSCAN0RFDF0x）、受信 FIFO バッファアクセスデータフィールド 1 レジスタ（RSCAN0RFDF1x）から読み出せます。

メッセージ数表示カウンタの値が FIFO バッファのバッファ数値（RSCAN0RFCCx レジスタの RFDC ビットで設定した値）に一致したとき、受信 FIFO バッファがフル（RSCAN0RFSTSx レジスタの RFFLL ビットが "1"）になります。

受信 FIFO バッファからすべてのメッセージを読み出したとき、受信 FIFO バッファが空（RSCAN0RFSTSx レジスタの RFEMP ビットが "1"）になります。

7.3.1　受信 FIFO バッファの読み出し手順

図 7.2 に受信 FIFO バッファの読み出し手順を示します。

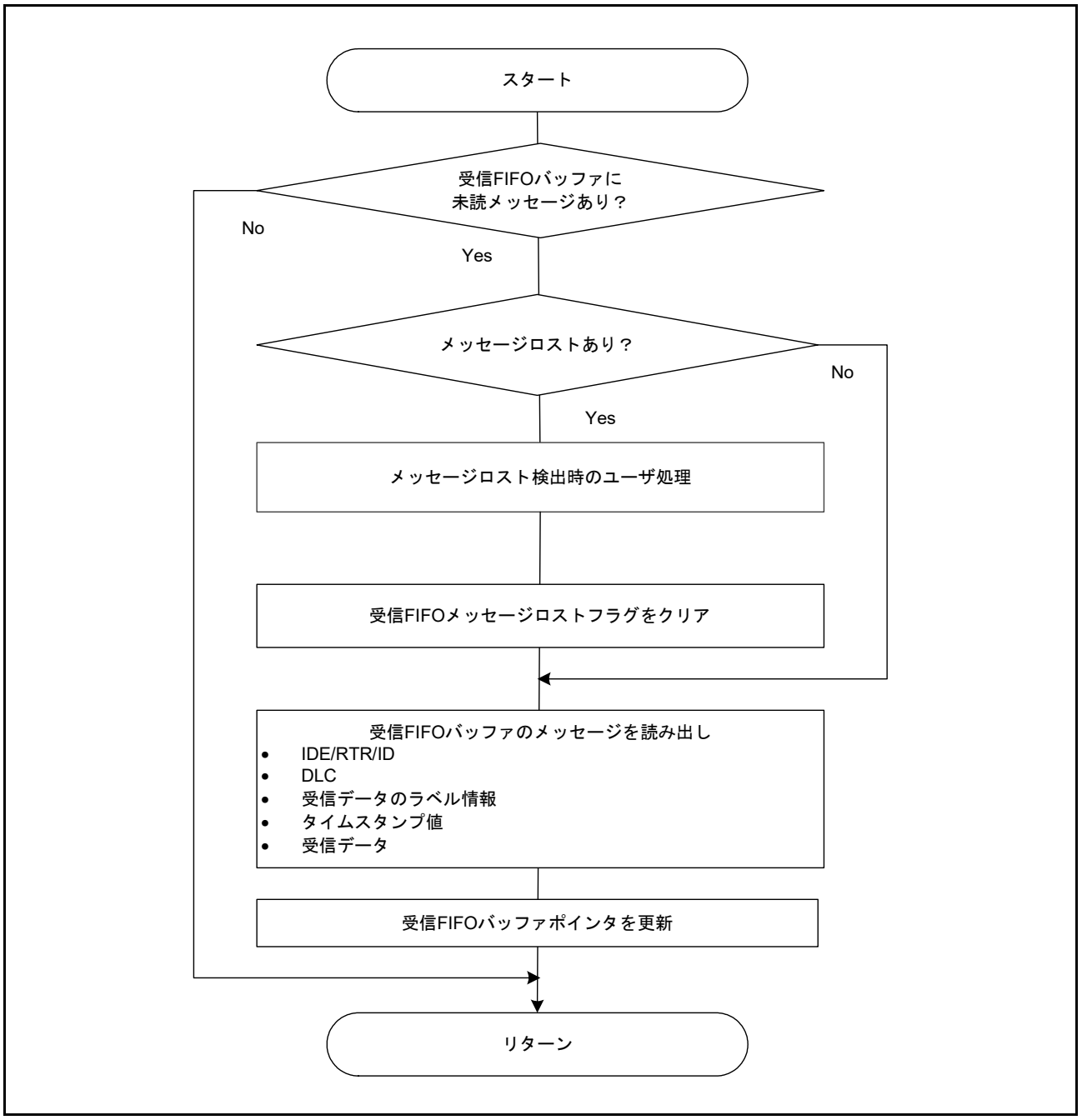


図 7.2　受信 FIFO バッファの読み出し手順

7.3.2 受信 FIFO 関連の割り込み処理

(1) 受信 FIFO 割り込み処理

受信 FIFO 割り込みを許可していれば、RSCAN0RFCCx レジスタの RFIM ビットの設定で選択した条件を満たした ときに受信 FIFO 割り込みが発生します。

割り込み要求が発生している状態（RSCAN0RFSTSx レジスタの RFIF ビットが "1"）受信 FIFO バッファの使用を禁止にしても割り込み要求フラグは自動的に "0" になりません。プログラムで "0" にしてください。

受信 FIFO 割り込みの許可、禁止は RSCAN0RFCCx レジスタの RFIE ビットで受信 FIFO バッファごとに設定できます。受信 FIFO 割り込みの要因を以下に示します。

- RSCAN0RFCCx レジスタのCFIGCV[2:0]ビットで設定した条件に達したときに受信 FIFO割り込み要求発生 (RSCAN0RFCCx レジスタの RFIM ビットが "0")
- 1メッセージ受信が完了するごとに受信 FIFO割り込み要求発生(RSCAN0RFCCx レジスタのRFIM ビットが "1")

受信 FIFO 割り込みを発生させるためには、対応する割り込み許可ビットが "1" である割り込み要求フラグをすべて "0" にする必要があります。

(2) グローバル・エラー割り込み処理

FIFO メッセージ・ロスト割り込みを許可していれば、受信 FIFO バッファのメッセージ・ロスト検出時にグローバル・エラー割り込みが発生します。FIFO メッセージ・ロスト割り込みの許可、禁止は RSCAN0GCTR レジスタの MEIE ビットでモジュール全体に共通で設定できます。

7.4 送受信 FIFO バッファを用いた受信

送受信 FIFO バッファは受信モード、送信モードのいずれかで使用することが可能です。

本章では、受信モードのみを記載します。

各チャネル専用の送受信 FIFO バッファが 1 チャネルにつき 3 本ずつあります。受信モードに設定した送受信 FIFO バッファは受信 FIFO バッファと同じく、バッファ数分のメッセージを保存することができます。

受信メッセージが受信モードに設定した送受信 FIFO バッファへ格納されると、対応するメッセージ数表示カウンタ (RSCAN0CFSTSk レジスタの CFMC[5:0] ビット) の値がインクリメントされます。

受信メッセージは、送受信 FIFO バッファアクセス ID レジスタ（RSCAN0CFIDk）、送受信 FIFO バッファアクセスポインタレジスタ（RSCAN0CFPTRk）、送受信 FIFO バッファアクセスデータフィールド 0 レジスタ（RSCAN0CFDF0k）、送受信 FIFO バッファアクセスデータフィールド 1 レジスタ（RSCAN0CFDF1k）から読み出すことができます。送受信 FIFO バッファは古いメッセージから読み出せます。

メッセージ数表示カウンタの値が送受信 FIFO バッファのバッファ数値 (RSCAN0CFCCk レジスタの CFDC[2:0] ビットで設定した値) に一致したとき、送受信 FIFO バッファがフル (RSCAN0CFSTSk レジスタの CFFLL フラグが "1") になります。

送受信 FIFO バッファからすべてのメッセージを読み出したとき、送受信 FIFO バッファが空 (RSCAN0CFSTSk レジスタの CFEMP フラグが "1") になります。

7.4.1　送受信 FIFO バッファ読み出し手順

図 7.3 に送受信 FIFO バッファの読み出し手順を示します。

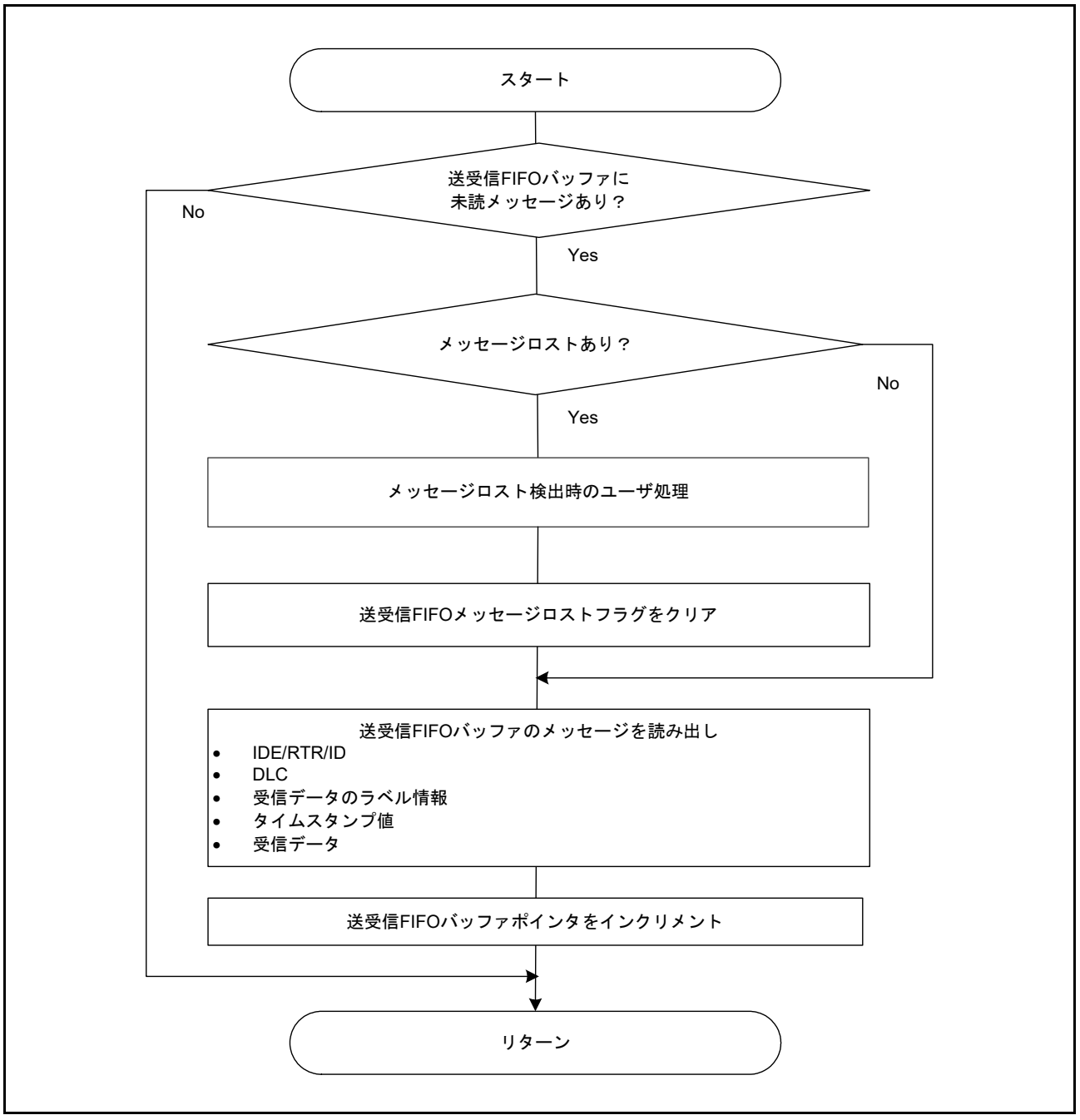


図 7.3　送受信 FIFO バッファの読み出し手順

7.4.2 送受信 FIFO バッファ（受信モード）の割り込み処理

(1) 送受信 FIFO 受信完了割り込み処理

送受信 FIFO 割り込みを許可していれば、RSCAN0CFCCk レジスタの CFIM ビットの設定で選択した条件を満たしたときに送受信 FIFO 割り込みが発生します。

割り込み要求が発生している状態（RSCAN0CFSTSk レジスタの CFRXIF ビットが "1"）で送受信 FIFO バッファの使用を禁止にしても割り込み要求フラグは自動的に "0" になりません。プログラムで "0" にしてください。

送受信 FIFO 割り込みの許可、禁止は RSCAN0CFCCk レジスタの CFRXIE ビットで送受信 FIFO バッファごとに設定できます。送受信 FIFO 割り込みの要因を以下に示します。

- RSCAN0CFCCkレジスタのCFIGCV[2:0]ビットで設定した条件に達したときに送受信 FIFO割り込み要求発生 (RSCAN0CFCCk レジスタの CFIM ビットが "0")
- 1メッセージ受信が完了するごとに受信 FIFO割り込み要求発生(RSCAN0CFCCkレジスタのCFIMビットが "1")

送受信 FIFO 割り込みを発生させるためには、対応する割り込み許可ビットが "1" である割り込み要求フラグをすべて "0" にする必要があります。

(2) グローバル・エラー割り込み処理

FIFO メッセージ・ロスト割り込みを許可していれば、送受信 FIFO バッファのメッセージ・ロスト検出時にグローバル・エラー割り込みが発生します。FIFO メッセージ・ロスト割り込みの許可、禁止は RSCAN0GCTR レジスタの MEIE ビットでモジュール全体に共通で設定できます。

8. 送信

8.1 送信機能

CAN メッセージ送信を行う場合に使用可能な機能を以下に示します。各処理の詳細については次章以降を参照ください。

- 送信バッファを用いた送信
- 送受信 FIFO バッファを用いた送信
- 送信履歴バッファ機能

8.2 送信バッファを用いた送信

送信バッファを用いて、データ・フレームまたはリモート・フレームを送信します。

送信バッファは 1 チャネルにつき 16 バッファあり、送信バッファ、送受信 FIFO バッファ (送信モード) へのリンク用のいずれかで使用することが可能です。

送信バッファの機能を以下に示します。

- メッセージ送信機能
- 送信アボート機能
- ワンショット送信機能 (再送信禁止機能)

8.2.1 メッセージ送信機能

データ・フレームまたはリモート・フレームを送信する機能です。

送信バッファに対して送信要求をセット（RSCAN0TMCp レジスタの TMTR ビットを "1"）すると、メッセージを送信できます。

送信結果は、対応する RSCAN0TMSTSp レジスタの TMTRF[1:0] フラグで確認できます。

送信が成功すると、

- 送信完了：送信アボート要求なし（TMTRF[1:0] フラグが B'10）
- 送信完了：送信アボート要求あり（TMTRF[1:0] フラグが B'11）

送信バッファごとに送信完了時の割り込み許可、禁止を RSCAN0TMIEC0 レジスタの TMIEp ビットで設定できます。

8.2.2　送信バッファからのメッセージ送信手順

図 8.1 に送信バッファからのメッセージ送信手順を示します。

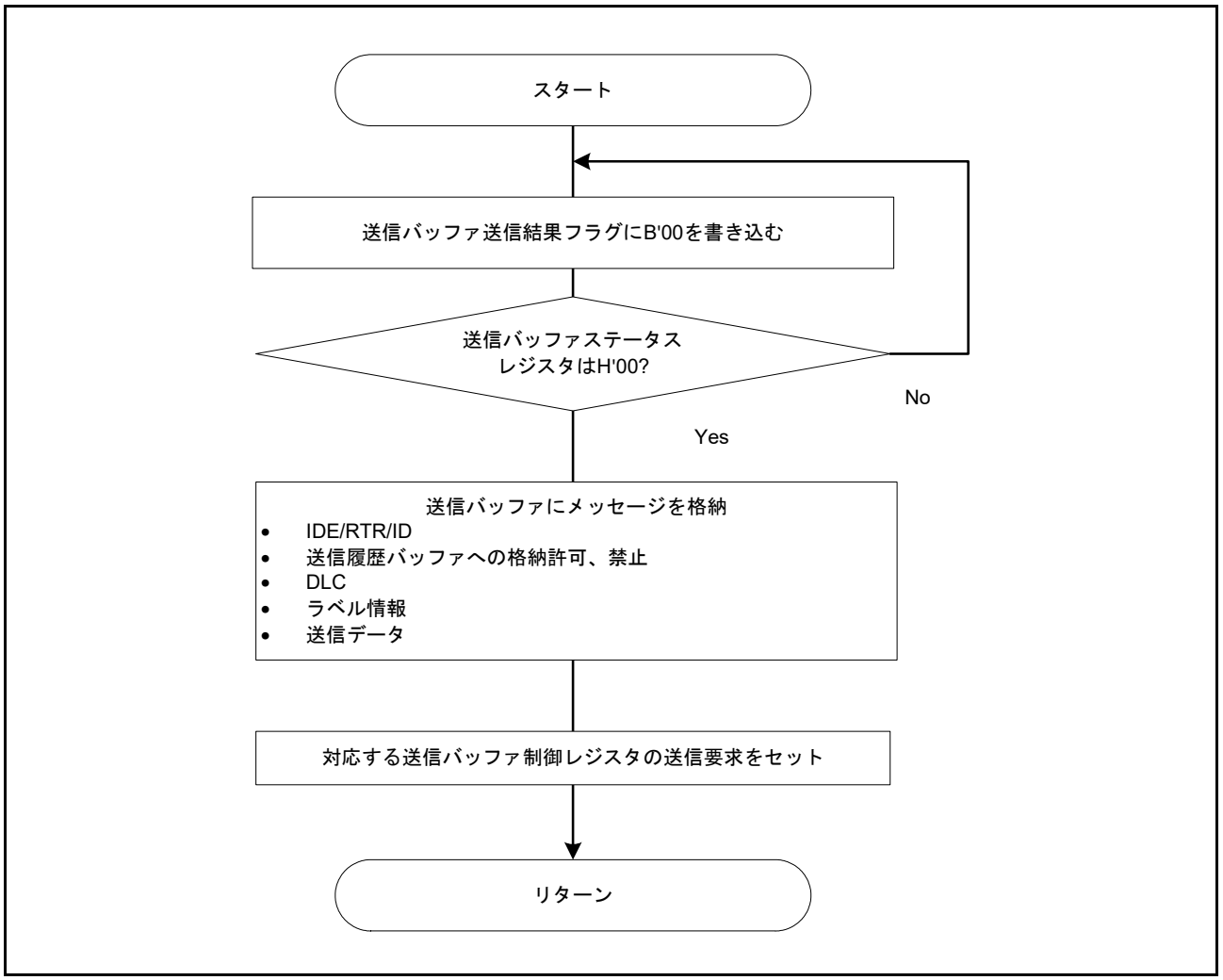


図 8.1　送信バッファからのメッセージ送信手順

8.2.3　送信アボート機能

2 つ以上のノードが同時に送信を始めた場合、CAN ID の優先度が低いメッセージのノードはアービトレーション負けとなります。アービトレーションに勝つか、CAN バスがアイドル状態のときに送信しない限り、メッセージの送信が正常に終了しません。

このようなときの再送信中のメッセージを破棄するための送信アボート機能があります。

送信アボート機能は、1 つのメッセージ送信に制限時間を設けたいときや緊急な優先順位の高いメッセージを送信するときなどに有効です。

送信要求がある（RSCAN0TMSTSp レジスタの TMTRM ビットが "1"）送信バッファに対して、送信アボート要求を発行（RSCAN0TMCp レジスタの TMTAR ビットを "1"）すると送信要求が取り消されます。

送信アボート要求を発行した後、実際にアボートされるタイミングを以下に示します。

- 送信中のメッセージまたは送信の優先順位判定で次の送信に決定しているメッセージ
  - アービトレーション・ロストが発生したとき
  - エラーが発生したとき
- 上記以外のメッセージ
  - 送信アボート要求を発行したとき

送信アボートが完了すると、RSCAN0TMSTSp レジスタの TMTRF[1:0] フラグが B'01 になり、送信要求が取り消されます。（TMTRM ビットが "0"）

送信中のメッセージまたは送信の優先順位判定で次の送信に決定しているメッセージに送信アボート要求を発行した後、アービトレーション・ロストやエラーが発生せず正常に送信完了した場合は、

- 送信完了：送信アボート要求あり（TMTRF[1:0] フラグが B'11）

8.2.4　送信アボート手順

図 8.2 に送信アボート手順を示します。

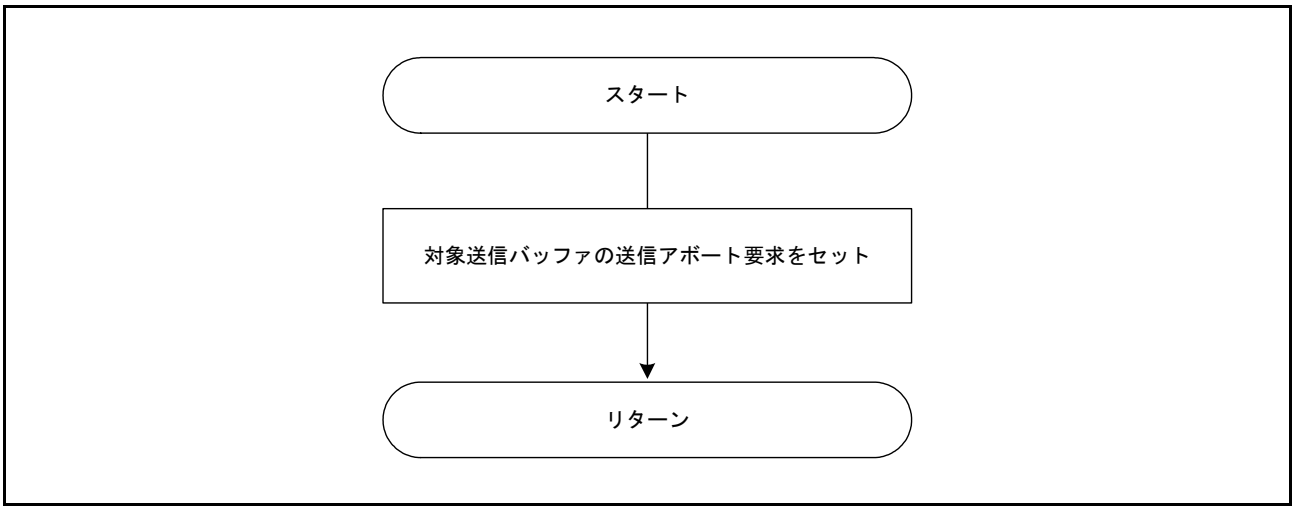


図 8.2　送信アボート手順

8.2.5　ワンショット送信機能

メッセージ送信要求発行時にワンショット許可（RSCAN0TMCp レジスタの TMCM ビットを "1"）にすると、1 回だけ送信を行います。アービトレーション・ロストまたはエラー発生時は再送信を行いません。

ワンショット送信の結果は、RSCAN0TMSTSp レジスタの TMTRF[1:0] フラグで確認できます。

ワンショット送信が成功すると、送信バッファ送信結果ステータスが下記になります。

- 送信完了：送信アボート要求なし（TMTRF[1:0] フラグが B'10）
- 送信完了：送信アボート要求あり（TMTRF[1:0] フラグが B'11）

アービトレーション・ロストまたはエラーが発生した場合は、送信バッファ送信結果ステータスが下記になります。

- アボート完了（TMTRF[1:0] フラグが B'01）

8.2.6　ワンショット送信手順

図 8.3 にワンショット送信手順を示します。

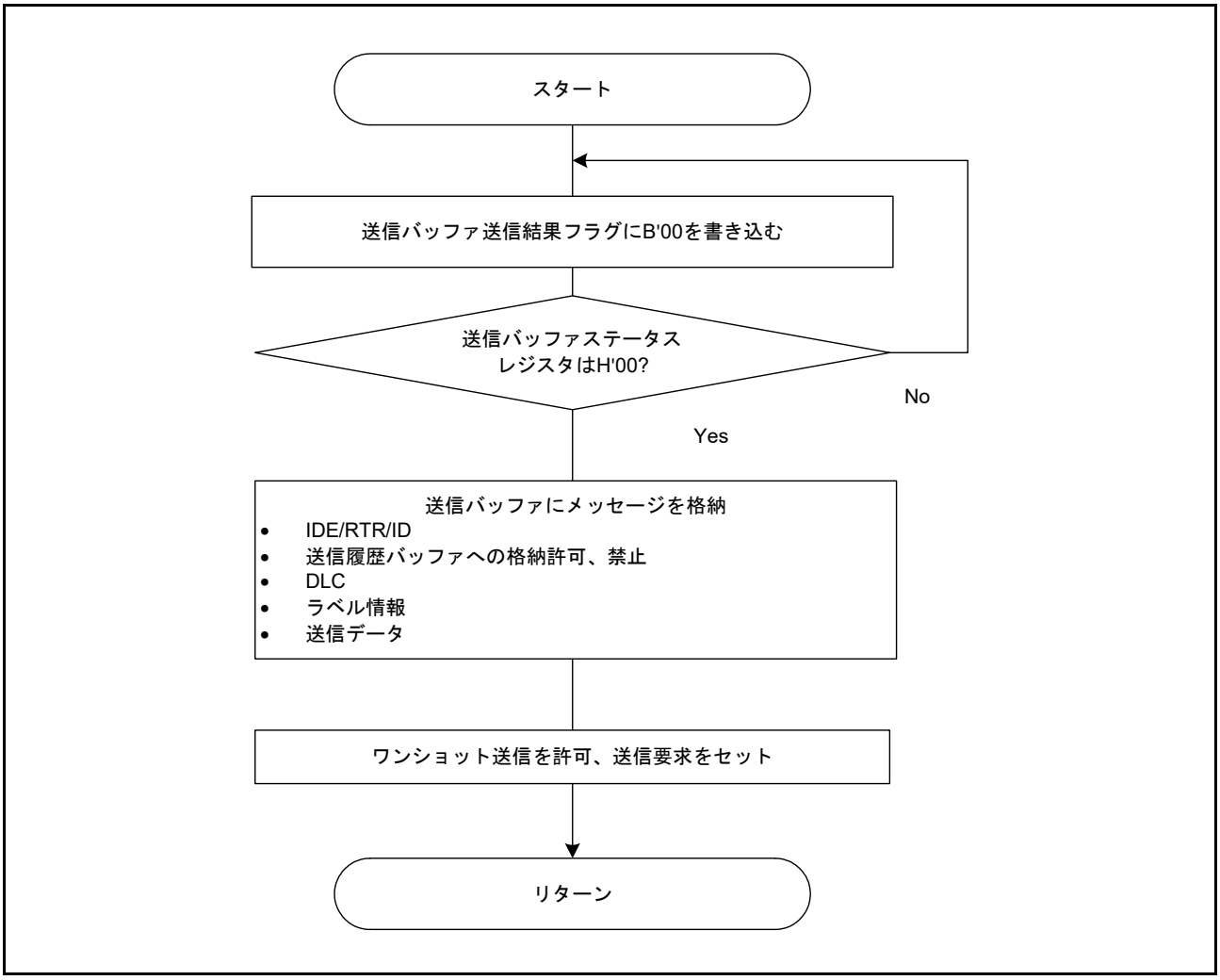


図 8.3　ワンショット送信手順

8.2.7 送信バッファの割り込み処理

(1) 送信完了割り込み処理

送信完了割り込みを許可していれば、送信完了したときに CANi 送信割り込みが発生します。
送信完了割り込みの許可、禁止は RSCAN0TMIEC0 レジスタの TMIEp ビットで送信バッファごとに設定できます。
CANi 送信割り込みは以下の要因を共用しています。複数の割り込み要因を使用する場合は、必要に応じて割り込み内で要因を判別してください。
なお、CANi 送信割り込みの発生要因は、RSCAN0GTINTSTS0 レジスタでも確認できます。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

(2) 送信アボート割り込み処理

送信アボート割り込みを許可していれば、送信アボート完了したときに CANi 送信割り込みが発生します。送信アボート割り込みの許可、禁止は RSCAN0CmCTR レジスタの TAIE ビットでチャネル毎に設定できます。ただし、送信完了：アボート要求あり (TMTRF[1:0] フラグが B'11) の場合には送信アボート割り込みは発生せず、送信完了割り込みが発生します。
CANi 送信割り込みは以下の要因を共用しています。複数の割り込み要因を使用する場合は、必要に応じて割り込み内で要因を判別してください。
なお、CANi 送信割り込みの発生要因は、RSCAN0GTINTSTS0 レジスタでも確認できます。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

8.2.8　送信完了または送信アボート完了後の処理手順

(1) 送信完了または送信アボート完了後の処理手順（割り込み未使用）

図 8.4 に送信完了または送信アボート完了後の処理手順（割り込み未使用）を示します。

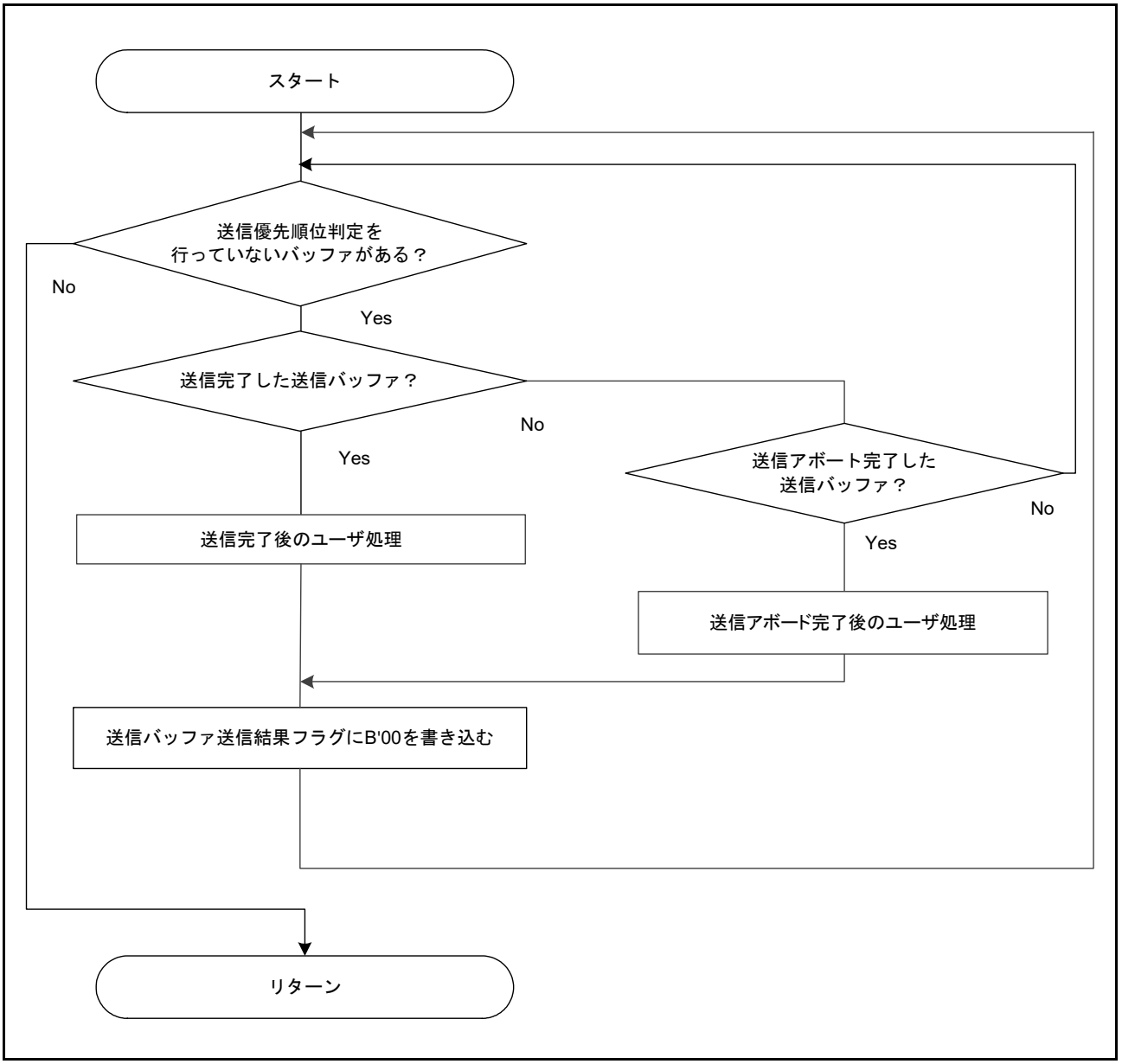


図 8.4　送信完了または送信アボート完了後の処理手順（割り込み未使用）

(2) 送信完了後の処理手順（割り込み使用時）

図 8.5 に送信完了後の処理手順（割り込み使用時）を示します。

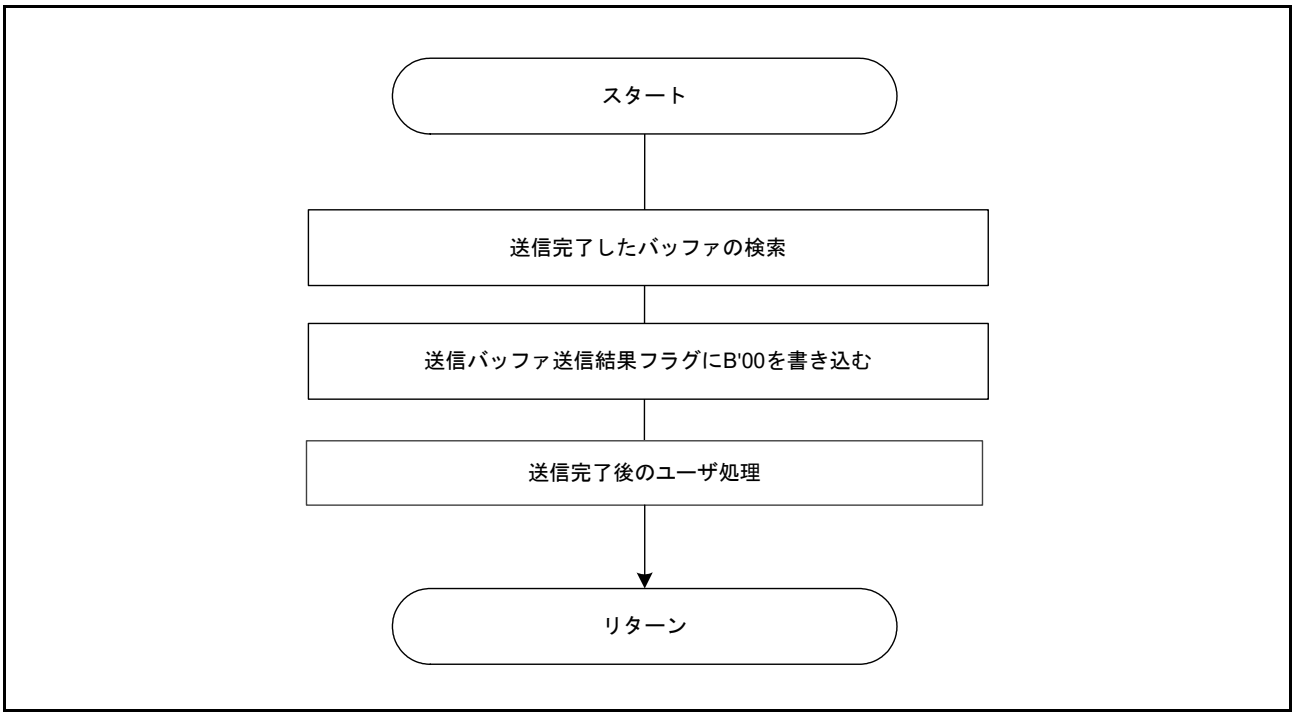


図 8.5　送信完了後の処理手順（割り込み使用時）

(3) 送信アボート完了後の処理手順（割り込み使用時）

図 8.6 に送信アボート完了後の処理手順（割り込み使用時）を示します。

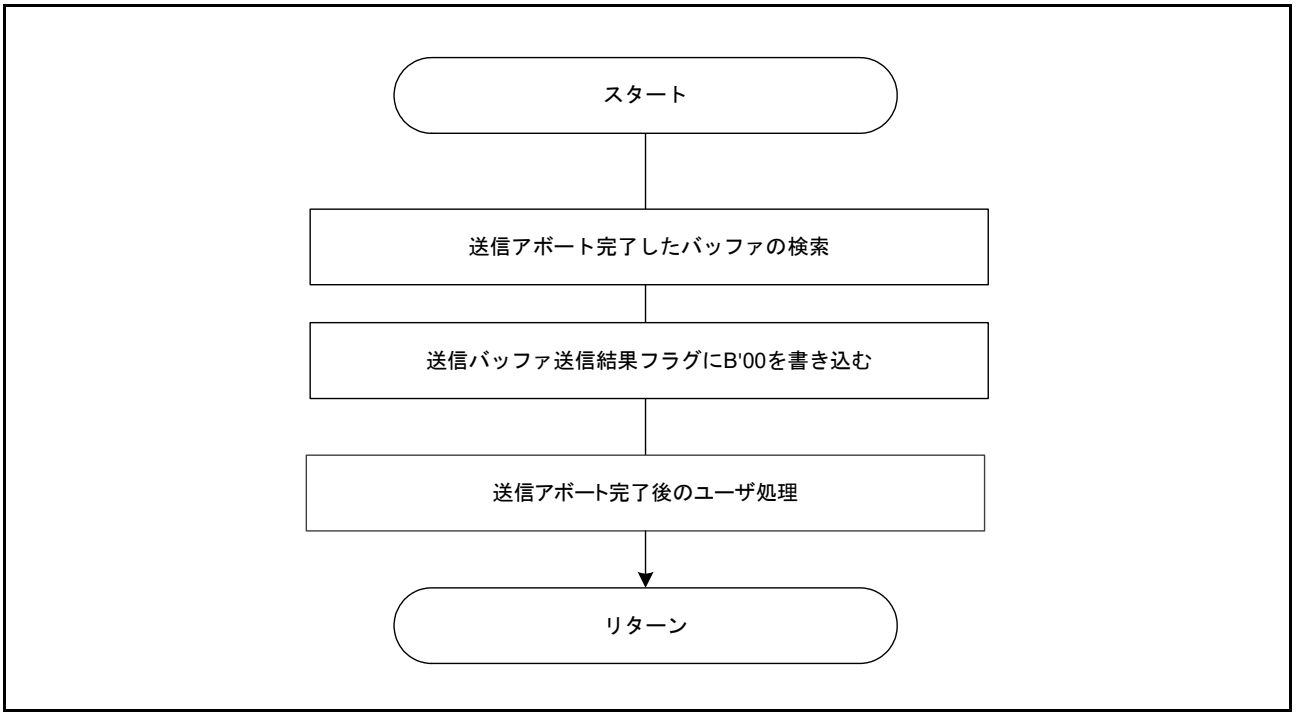


図 8.6　送信アボート完了後の処理手順（割り込み使用時）

8.3 送受信 FIFO バッファを用いた送信

送受信 FIFO バッファを用いて、データ・フレームまたはリモート・フレームを送信します。

送受信 FIFO バッファは 1 チャネルにつき 3 本あり、1 本の FIFO バッファに最大 128 メッセージ格納できます。最初に格納し たメッセージから順に送信されます。

送受信 FIFO バッファは受信モード、送信モードのいずれかで使用することが可能です（送信モードのみを説明します)。

送受信 FIFO バッファが持つ送信機能を以下に示します。

- メッセージ送信機能
- 送信アボート機能
- インターバル送信機能

8.3.1 メッセージ送信機能

データ・フレームまたはリモート・フレームを送信する機能です。

送受信 FIFO バッファに格納したメッセージは格納した順番で送信されます。

8.3.2　送受信 FIFO からのメッセージ送信手順

図 8.7 に送受信 FIFO からのメッセージ送信手順を示します。

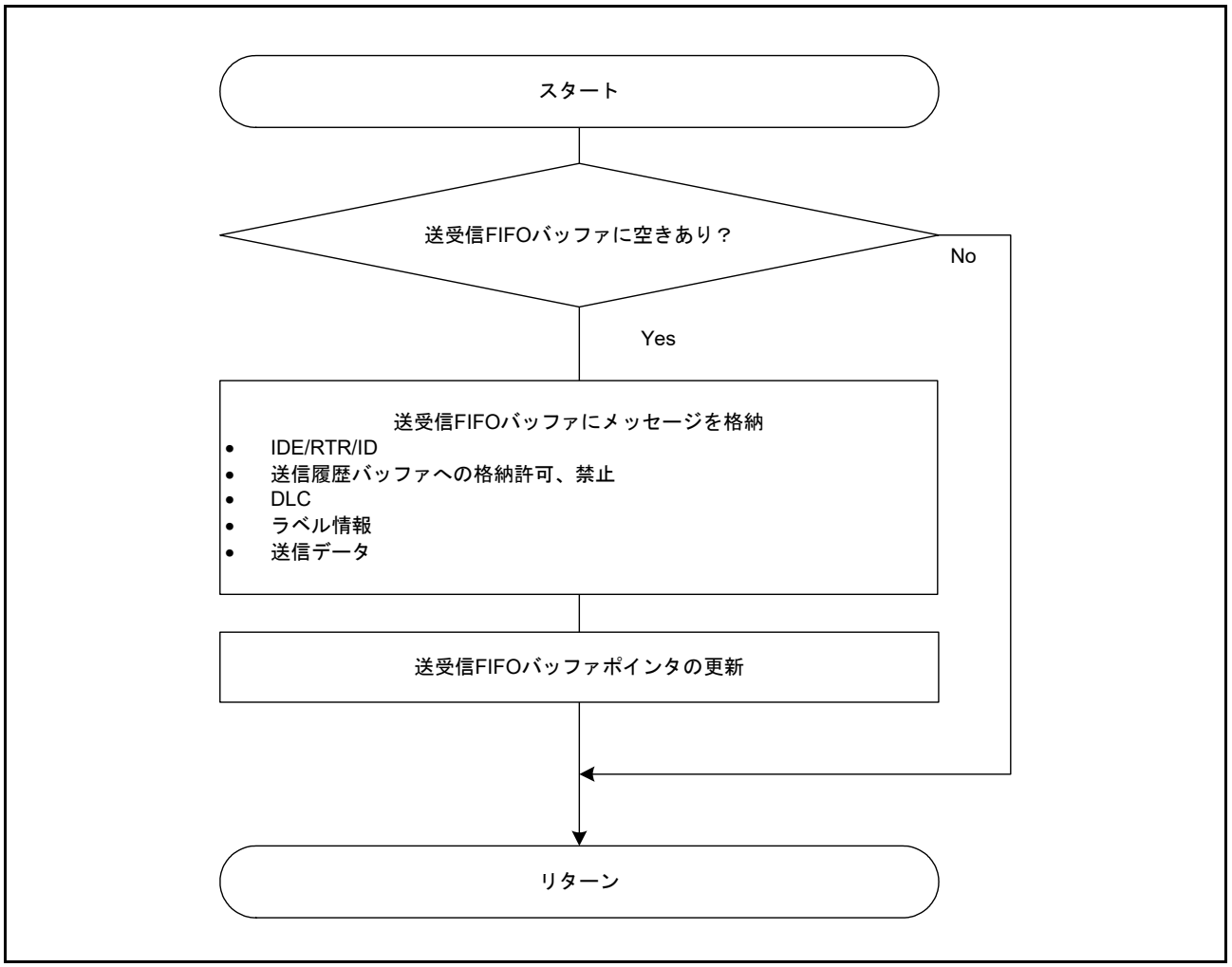


図 8.7　送受信 FIFO からのメッセージ送信手順

8.3.3 送信アボート機能

送受信 FIFO バッファの使用を禁止することで、送受信 FIFO バッファ内のメッセージをアボートすることができます。送受信 FIFO バッファのアボートでは送信中のメッセージだけでなく、送受信 FIFO バッファ内のすべてのメッセージがアボートされます。

送受信 FIFO バッファのアボート完了の確認は送受信 FIFO バッファの空を確認することで実施できます。

送受信 FIFO バッファの送信アボート完了による割り込みは発生しません。ただし、送信中にアボートすると送信完了による送受信 FIFO 送信完了割り込みが発生する場合があります。

8.3.4 インターバル送信機能

送信モードに設定した送受信 FIFO バッファ使用時に、同一送受信 FIFO バッファから連続してメッセージを送信する場合、メッセージ送信間のインターバル時間を設定できます。

8.3.5 送受信 FIFO バッファ（送信モード）の割り込み処理

(1) 送受信 FIFO 送信割り込み処理

送受信 FIFO 送信完了割り込みを許可していれば、RSCAN0CFCCk レジスタの CFIM ビットの設定で選択した条件を満たしたときに CANi 送信割り込みが発生します。

CANi 送信割り込みは以下の要因を共用しています。複数の割り込み要因を使用する場合は、必要に応じて割り込み内で要因を判別してください。

なお、CANi 送信割り込みの発生要因は、RSCAN0GTINTSTS0 レジスタでも確認できます。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

送信モード時の送受信 FIFO 送信完了割り込みの要因を以下に示します。

- メッセージ送信完了によってバッファが空になったときに送受信 FIFO 送信完了割り込み要求発生
- 1 メッセージ送信が完了するごとに送受信 FIFO 送信完了割り込み要求発生

8.4 送信履歴バッファ機能

送信完了したメッセージの情報 (送信履歴データ) を送信履歴バッファに格納できます。チャネルごとに１つの送信履歴バッファを持ち、送信履歴バッファには 16 個の送信履歴データを格納できます。

8.4.1 送信履歴データ格納機能

メッセージ送信元のバッファの種類とメッセージごとに送信履歴データを格納するかどうかを設定できます。メッセージ送信元のバッファの種類は、コンフィグレーション時に設定できます。

送信履歴データを格納するかどうかの設定、およびラベル・データの設定は各メッセージの送信時に設定できます。

送信が成功した後に以下の情報が送信履歴データとして送信履歴バッファへ格納されます。

- バッファ・タイプ
  格納されたメッセージが送信されたバッファ (送信バッファ、または送受信 FIFO バッファ) の種類
- バッファ番号
  送信元の送信バッファ、または送受信 FIFO バッファの番号
- ラベル・データ
  送信メッセージのラベル情報。ラベル情報は送信メッセージ格納時に設定できます。

8.4.2 送信履歴バッファの読み出し手順

図 8.8 に送信履歴バッファの読み出し手順を示します。

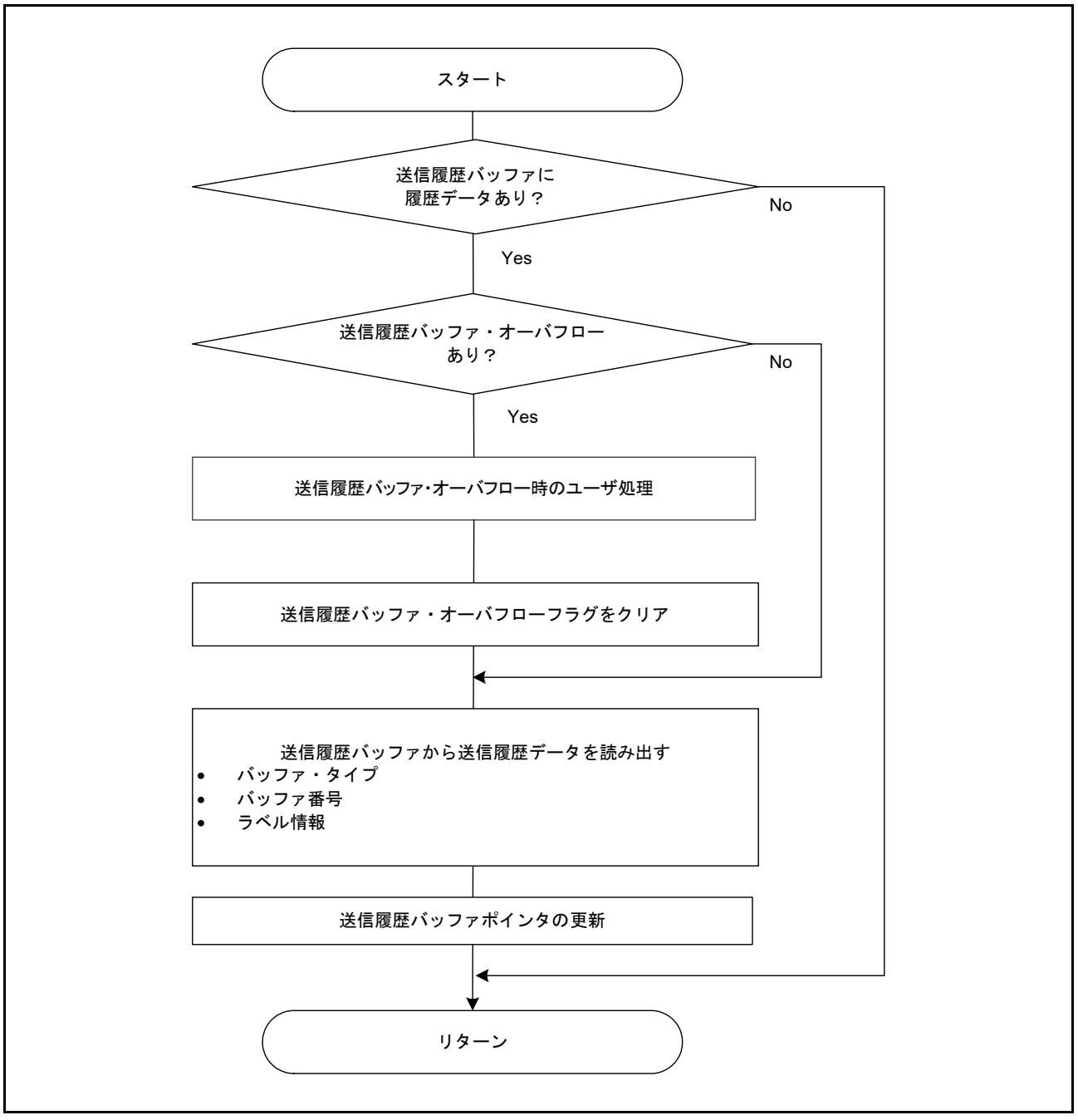


図 8.8 送信履歴バッファの読み出し手順

8.4.3 送信履歴バッファの割り込み処理

(1) 送信履歴割り込み処理

送信履歴割り込みを許可していれば、RSCAN0THLCCm レジスタの THLIM ビットの設定で選択した条件を満たしたときに CANi 送信割り込みが発生します。

CANi 送信割り込みは以下の要因を共用しています。複数の割り込み要因を使用する場合は、必要に応じて割り込み内で要因を判別してください。

なお、CANi 送信割り込みの発生要因は、RSCAN0GTINTSTS0 レジスタでも確認できます。

- CANi 送信完了割り込み
- CANi 送信アボート割り込み
- CANi 送受信 FIFO 送信完了割り込み
- CANi 送信履歴割り込み

(2) グローバル・エラー割り込み処理

送信履歴バッファ・オーバフロー割り込みを許可していれば送信履歴バッファのオーバフロー検出時にグローバル・エラー割り込みが発生します。送信履歴バッファ・オーバフロー割り込みの許可、禁止は RSCAN0GCTR レジスタの THLEIE ビットでモジュール全体に共通で設定できます。

9. CAN 関連の割り込み

9.1 CAN 関連の割り込み

CAN 関連の割り込みの許可／禁止のために対応する割り込み要求の設定を行います。
以下に使用可能な CAN 関連割り込みを示します。

- グローバル割り込み
  1. CAN 受信 FIFO 割り込み
  2. CAN グローバル・エラー割り込み
- チャネル割り込み
  1. CANi 送信割り込み
     - CANi 送信完了割り込み
     - CANi 送信アボート割り込み
     - CANi 送受信 FIFO 送信完了割り込み
     - CANi 送信履歴割り込み
     - CANi 送信キュー割り込み
  2. CANi 送受信 FIFO 受信完了割り込み
  3. CANi エラー割り込み

9.1.1 CAN 関連割り込みの設定手順

図 9.1 に CAN 関連割り込みの設定手順を示します。

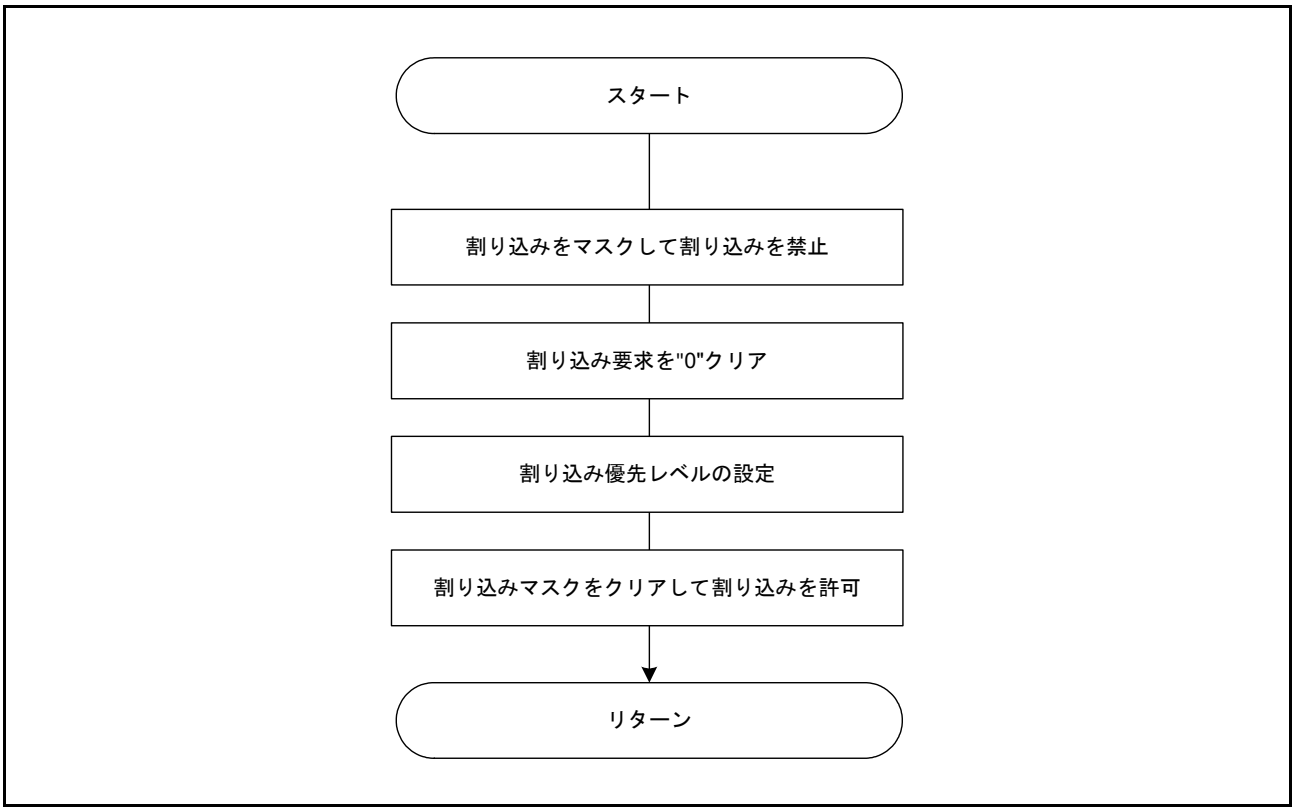


図 9.1 CAN 関連割り込みの設定手順

10.　ソフトウェア説明

10.1　動作概要

RSCAN サンプルプログラムの機能概要を「表 10.1　動作概要」に示します。また、図 10.1 にシステムブロック図を示します。

表 10.1　動作概要

機能	概要
兼用端子設定	• PC6: CAN0 CRXD0 • P67: CAN0 CTXD0 • PC7: CAN1 CRXD1 • P66: CAN1 CTXD1
CANの通信チャネル	• チャネル0（CAN0）を使用
割り込み要因（割り込み優先度）	• CANグローバル・エラー (3) • CAN0エラー (4) • CAN1エラー (4) • CAN受信FIFO (5) • CAN0送受信FIFO受信完了 (5) • CAN1送受信FIFO受信完了 (5) • CAN0送信 (5) • CAN1送信 (5)
転送速度設定	• 1 M[bps]
動作モード	• 送信モード（評価ボード2台接続時の送信側） 送信バッファを使ったメッセージ送信 送受信FIFO（送信モード）バッファを使ったメッセージ送信 • 受信モード（評価ボード2台接続時の受信側） 受信バッファを使ったメッセージ受信 受信FIFOバッファを使ったメッセージ受信 送受信FIFO（受信モード）バッファを使ったメッセージ受信 • 受信を受け付けながらの送信テスト • テストモード（評価ボード1台のみでのセルフテスト） セルフテストモード0（外部ループバックモード） セルフテストモード1（内部ループバックモード）
動作概要	メニューより動作モードを選択
動作結果表示	コンソールに結果出力

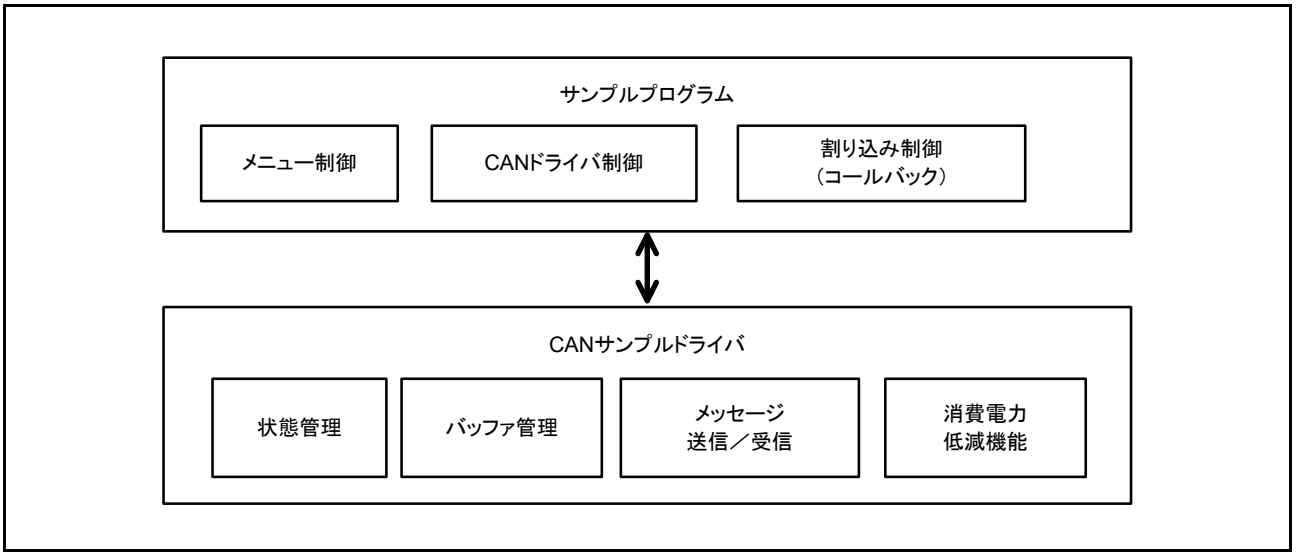


図 10.1　システムブロック図

10.1.1 プロジェクト設定

開発環境となる EWARM 上で使用されるプロジェクト設定については、RZ/T1 グループ 初期設定アプリケーションノートに記載しています。

10.1.2 使用準備（セルフテスト）

本サンプルプログラムを使用して、CAN 通信のセルフテストを行うことができます。
開発環境に接続している評価ボード単体で確認ができます。

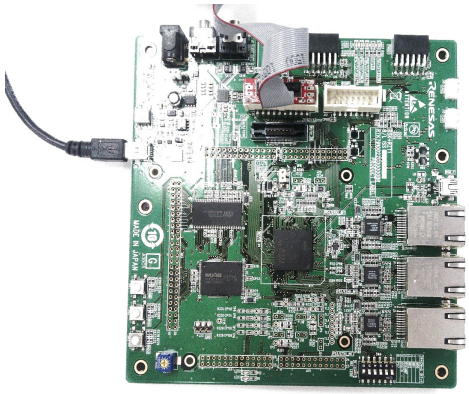


図 10.2 サンプルプログラムのセルフテスト実行時の構成

10.1.3　使用準備（送信テスト・受信テスト）

開発環境に接続している評価ボード（評価ボード (A)）と、別の PC にセットアップされた開発環境に接続している評価ボード（評価ボード (B)）の 2 台が必要です。

評価ボード (A) と評価ボード (B) を CAN ケーブル (注 1) で接続します。（お互いの CAN1 コネクタに接続）

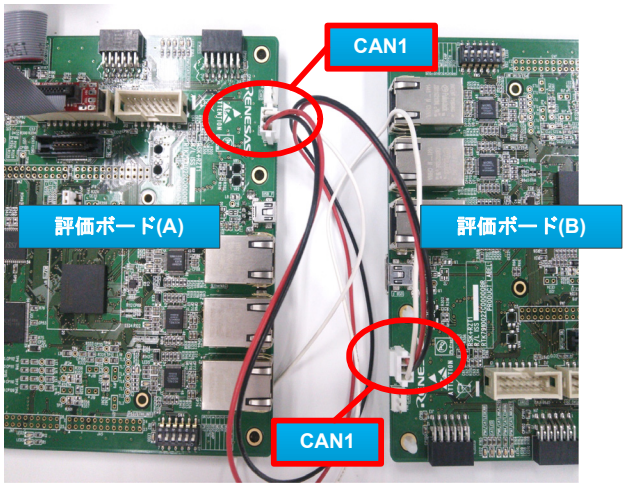


図 10.3　サンプルプログラムの送信テスト・受信テスト実行時の構成

注 1.　ボード (A)CAN コネクタ 1(J15) の CAN-H 端子とボード (B)CAN コネクタ 1(J15) の CAN-H 端子をケーブルで接続。
ボード (A)CAN コネクタ 1(J15) の CAN-L 端子とボード (B)CAN コネクタ 1(J15) の CAN-L 端子をケーブルで接続。

10.1.4　ターミナルソフト（Tera Term）

本サンプルプログラムは、FIFO 内蔵シリアルコミュニケーションインターフェース（SCIFA）の調歩同期式通信を用い、ホスト PC と RS-232C インターフェースの COM ポート通信を行います。

ホスト PC にてターミナルソフト（Tera Term）を起動してシリアルポートの設定（ボーレート：115200）を行ってください。また、送受信時の改行コードは "CR" に設定します。

- 転送速度：115200 bps
- キャラクタ長：8 ビット
- ストップビット長：1 ビット
- パリティ機能：なし
- ハードウェアフロー制御：なし

図 10.4　Tera Term：端末の設定

下記の例は、シリアルポートを "COM4" を使用した場合。

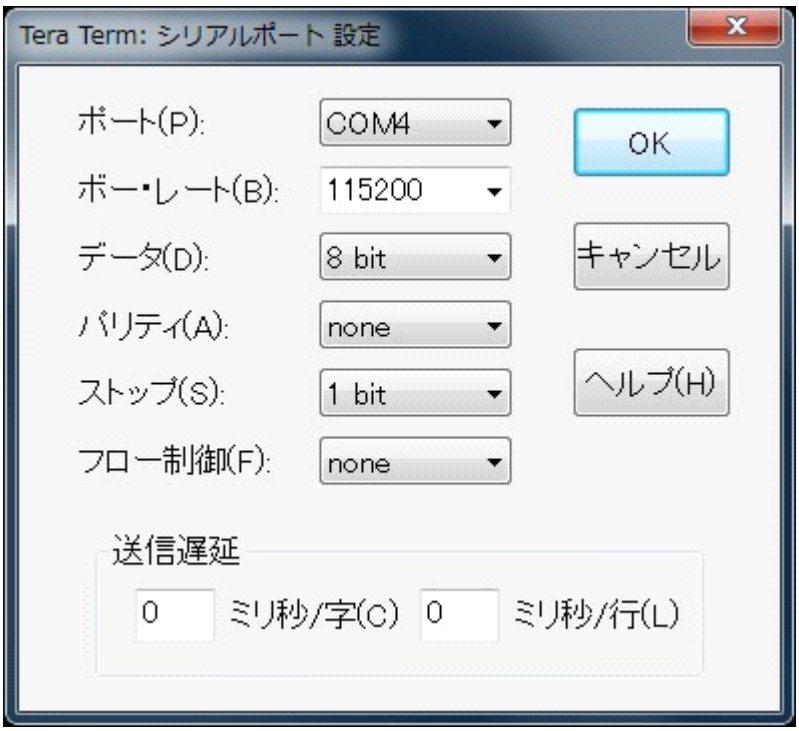


図 10.5　Tera Term：シリアルポートの設定

10.1.5 サンプルプログラムの機能

ターミナルソフト（Tera Term）を起動しておき、本サンプルプログラムを起動します。
コンソールのメニューよりサンプルプログラムの機能を選択してください。

- メインメニュー

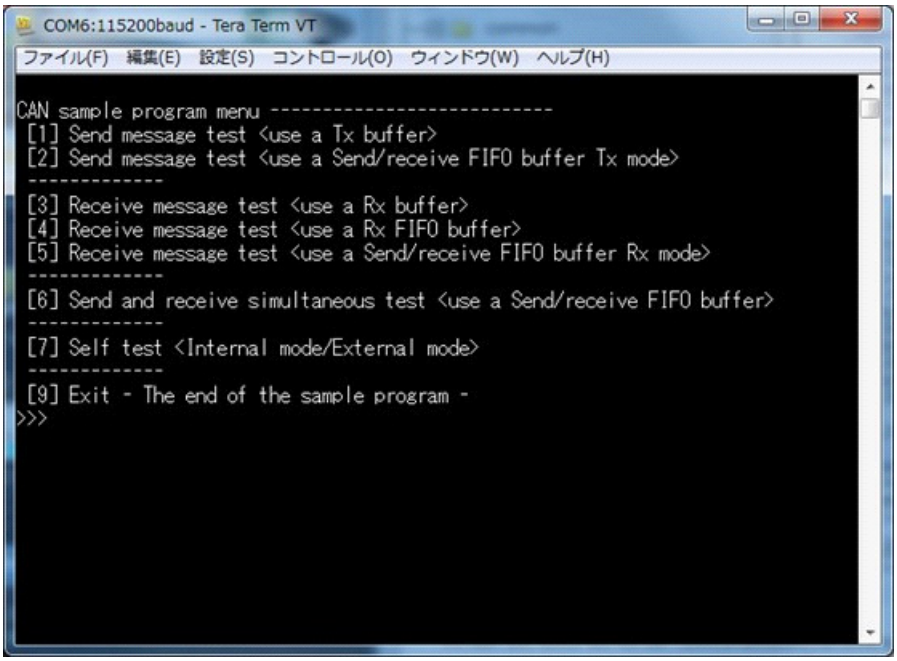


図 10.6 サンプルプログラムのメインメニュー

メニューの各項目の内容を以下に説明します。
[1] Send message test <uses a Tx buffer>
送信バッファからメッセージを送信するテスト。
[2] Send message test <uses a Send/receive FIFO buffer Tx mode>
送受信 FIFO バッファ（送信モード）からメッセージを送信するテスト。
[3] Receive message test <uses a Rx buffer>
受信バッファでメッセージを受信するテスト。
[4] Receive message test <uses a Rx FIFO buffer>
受信 FIFO バッファでメッセージを受信するテスト。
[5] Receive message test <uses a Send/receive FIFO buffer Rx mode>
送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト。
[6] Send and receive simultaneous test < uses a Send/receive FIFO buffer >
受信を受け付けながらの送信テスト。
[7] Self test <Internal mode/External mode>
セルフテスト選択メニュー。
[9] Exit - The end of the sample program -
サンプルプログラム終了。

- セルフテストメニュー

本メニューを使って、評価ボード (A) 単体でのセルフテスト（外部ループバック、内部ループバック）の動作確認を行うことができます。

外部ループバックモード、内部ループバックモードはメニューから切り替え可能です。

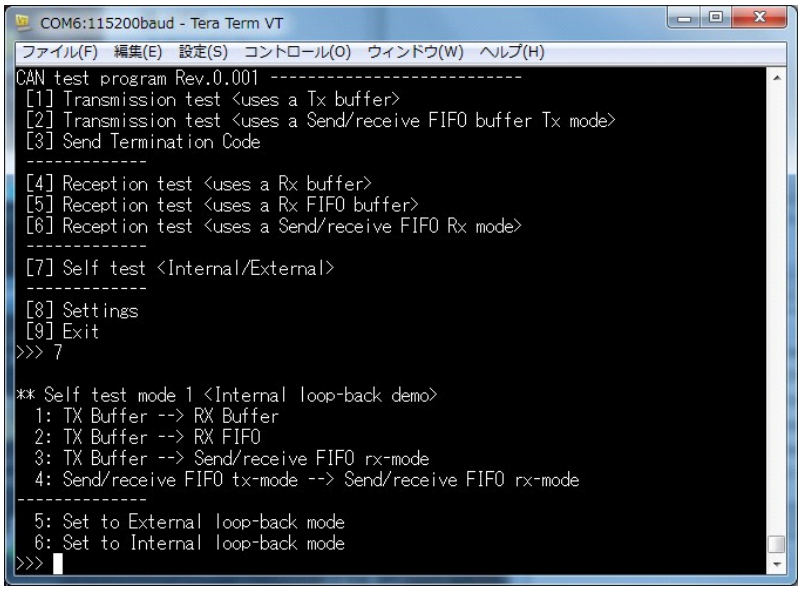


図 10.7　[7] Self test <Internal/Externa> 選択時のセルフテスト選択メニュー

各セルフテストの内容を以下に説明します。

1: Tx Buffer → Rx Buffer

送信バッファからメッセージを送信し、受信バッファでメッセージを受信するテスト。

2: Tx Buffer → Rx FIFO buffer

送信バッファからメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト。

3: Tx Buffer → Send/receive FIFO buffer Rx mode

送信バッファからメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト。

4: Send/receive FIFO buffer Tx mode → Send/receive FIFO buffer Rx mode

送受信 FIFO バッファ（送信モード）からメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト。

5: Set to External loop back mode

セルフテストモード 0（外部ループバックモード）の選択。

6: Set to Internal loop back mode

セルフテストモード 1（内部ループバックモード）の選択。

10.1.6 サンプルプログラム設定値

本サンプルプログラムは、以下の設定値にて動作します。

- チャネル：CAN0（固定）
- 通信速度： 1 Mbps（固定）
- 送信メッセージ：1 メッセージ（8 byte）の繰り返し
- 受信ルール：1 ルール（メッセージ ID：0x120 のみ受信可）（固定）
- 送信バッファ番号：0（固定）
- 受信バッファ番号：1（固定）
- 受信 FIFO バッファ番号：0（固定）
- 送受信 FIFO バッファ番号（送信モード）：0（固定）
- 送受信 FIFO バッファ番号（受信モード）：1（固定）
- 送受信 FIFO バッファのリンク先送信バッファ番号：2（固定）

サンプルデータ

- メッセージ ID：0x120（固定）
- メッセージタイプ：標準 ID（固定）
- データフォーマット：データ・フレーム（固定）
- メッセージデータ：00h ～ FFh（8byte を 1 個のメッセージとして送信）

受信バッファ

- バッファサイズ：1024byte（サイズを超えての受信時、先頭から上書き）

10.1.7　送信テスト

- 準備
  開発環境に接続している評価ボード (A) と、別の PC にセットアップされた開発環境に接続している評価ボード (B) をケーブルで接続してください。10.1.3　**使用準備（送信テスト・受信テスト）**を参照してください。

- 操作
  1. 受信側（別の PC にセットアップされた開発環境に接続している評価ボード (B)）を受信モードで待たせます。（評価ボード (B) で、メインメニューの [3]、[4]、[5] の何れかを選択）
  2. 送信側（評価ボード (A)）で、メインメニューの [1] または [2] を選択してください。
  3. 送信側（評価ボード (A)）からメッセージを送信し続けます。
     送信テストを終了する場合は、送信側で何かキーを押下してください。
  4. 送信テストが終了します。また、受信側も受信モードから抜けます。

- 送信データ
  メッセージ ID：0x120
  メッセージタイプ：標準 ID（値 0）
  データフォーマット：データ・フレーム（値 0）
  メッセージデータ：一回の送信メッセージは、8byte で構成する
  （サンプルでは、0x00 ～ 0xFF までの連続するデータを使用しています）
  デリミタコード：メッセージ終了コード
  （サンプルでは、0x00 を 8byte 続けて送信することでメッセージの終了を判断しています）

- 結果
  下記に送受信 FIFO バッファ（送信モード）を使ったメッセージ送信テストのテスト結果を示します。

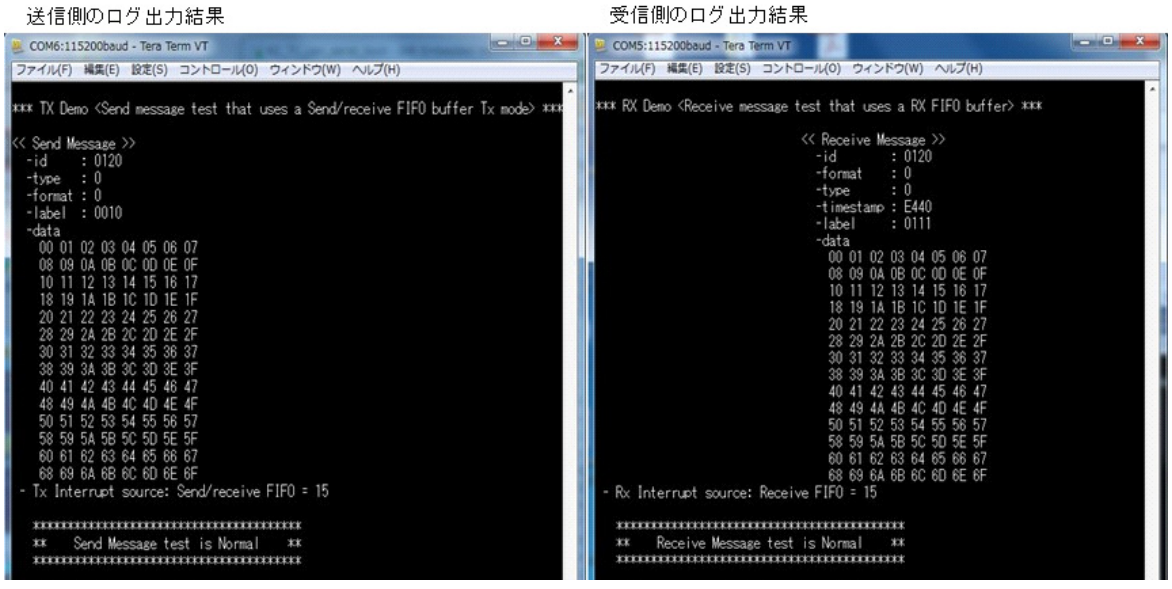


図 10.8　サンプルプログラムの送信テスト結果例

- 注意事項
  上記の操作項において、受信側（別の PC にセットアップされた開発環境に接続している評価ボード (B)）が受信モードになっていない状態で、送信側（評価ボード (A)）からメッセージを送信した場合、下記のエラーとなります。
  受信側（評価ボード (B)）の準備を終えてから、送信側（評価ボード (A)）よりメッセージを送信してください。

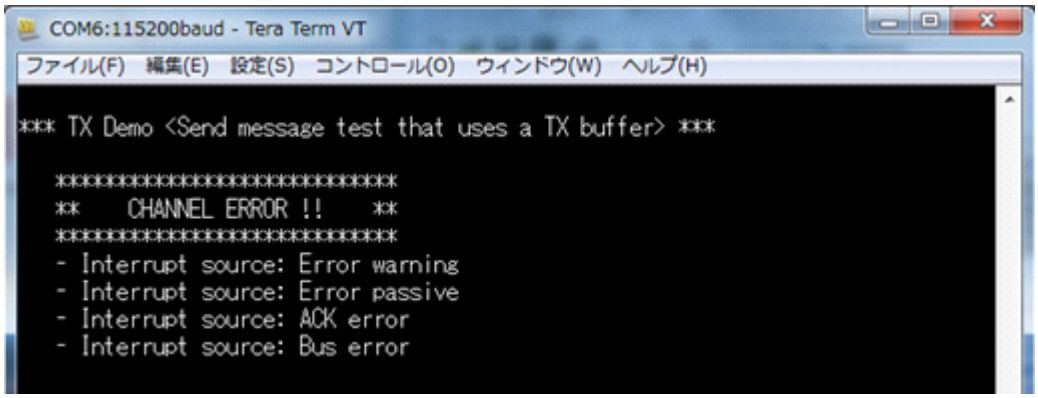


図 10.9　サンプルプログラムの送信テストエラー結果例

10.1.8　受信テスト

- 準備
  開発環境に接続している評価ボード (A) と、別の PC にセットアップされた開発環境に接続している評価ボード (B) をケーブルで接続してください。10.1.3　**使用準備（送信テスト・受信テスト）**を参照してください。

- 操作
  1. 評価ボード (A) のメインメニューで [3]、[4]、[5] の何れかを選択し、受信モードにしてください。
  2. 別のPCにセットアップされた開発環境に接続している評価ボード(B)のメインメニューで[1]または[2]を選択してください。
  3. 送信側（評価ボード (B)）からメッセージを送信し続けます。
     受信側（評価ボード (A)）の受信テストを終了する場合は、送信側（評価ボード (B)）で何かキーを押下してください。
  4. 送信側（評価ボード (B)）は、送信テストを終了し、受信側（評価ボード (A)）も受信モードから抜けます。

- 結果
  下記に送受信 FIFO バッファ（受信モード）でメッセージを受信するテストのテスト結果を示します。

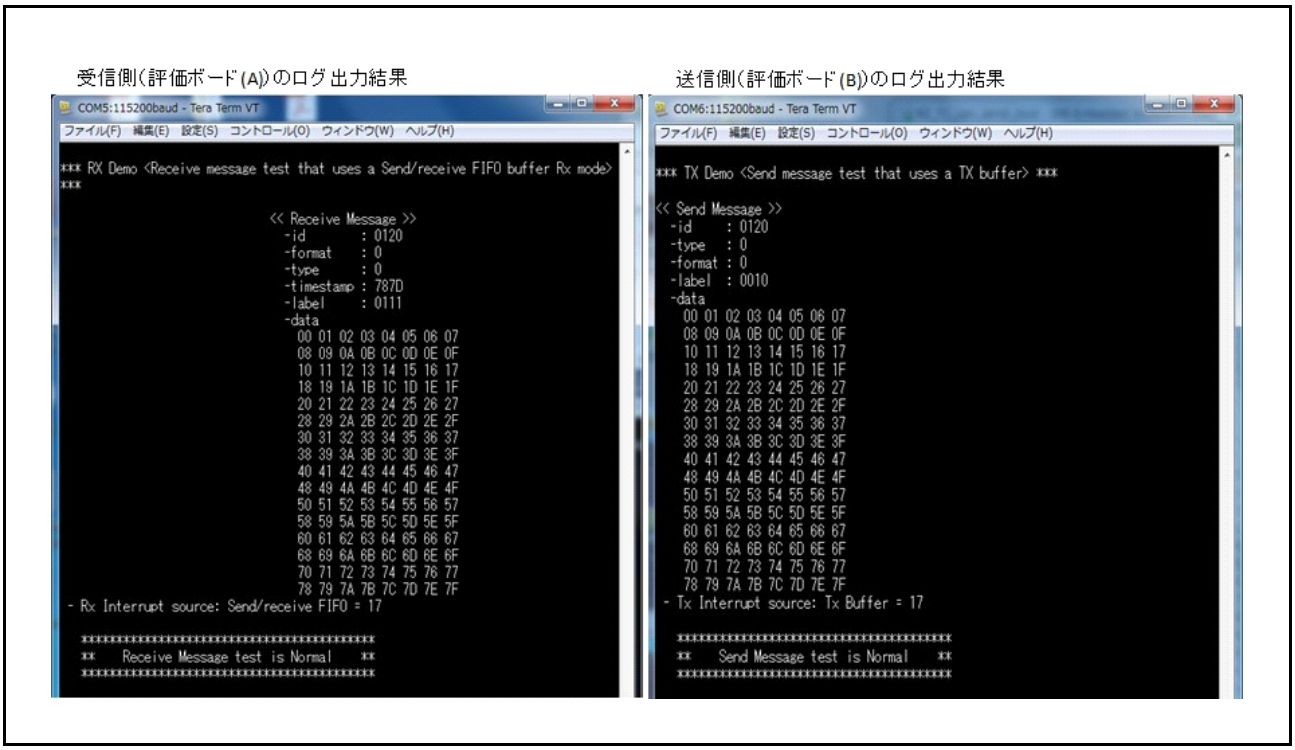


図 10.10　サンプルプログラムの受信テスト結果例

10.1.9　受信を受け付けながらメッセージ送信テスト

- 準備
  開発環境に接続している評価ボード (A) と、別の PC にセットアップされた開発環境に接続している評価ボード (B) をケーブルで接続してください。10.1.3　**使用準備（送信テスト・受信テスト）**を参照してください。

- 操作
  1. 評価ボード (A) 側のメインメニューで [6] を選択します。
  2. キー押下を促すガイダンスが出力されます。
  3. 別のPCにセットアップされた開発環境に接続している評価ボード(B)側のメインメニューで同じく [6] を選択してください。
  4. キー押下を促すガイダンスが出力されます。
  5. 両方にキー押下を促すガイダンスが出力されたことを確認後、評価ボード (B) 側で何かキーを押下してください。

  1 ～ 5 の操作を行うと、評価ボード (A) 側、評価ボード (B) 側から共にメッセージを送信し続けます。
  テストを終了する場合は、評価ボード (A) 側、評価ボード (B) 側、何れかで何かキーを押下してください。
  評価ボード (A) 側、評価ボード (B) 側、共に受信を受け付けながらメッセージ送信テストを終了します。

- 結果
  下記に受信を受け付けながらメッセージ送信するテストのテスト結果を示します。

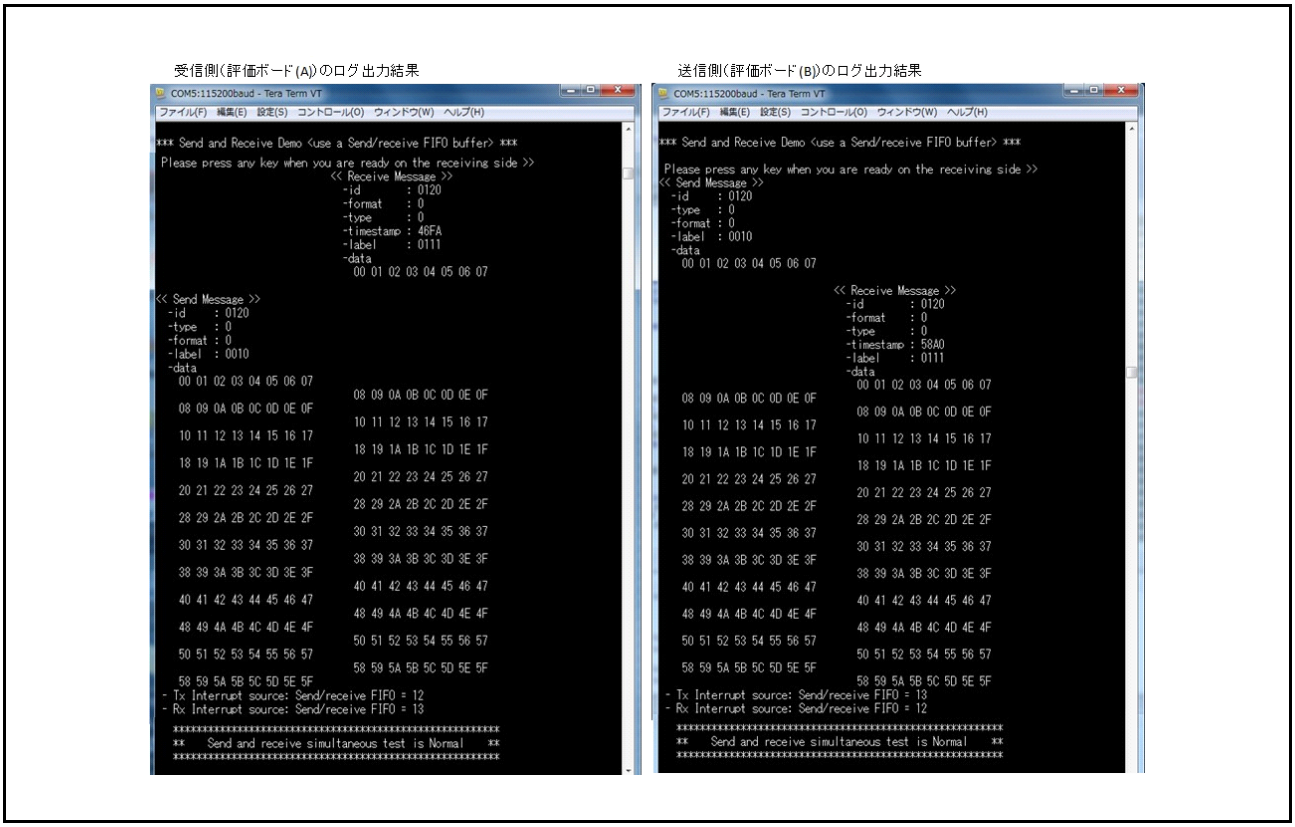


図 10.11　サンプルプログラムの受信を受け付けながらメッセージ送信テスト結果例

10.1.10　セルフテスト

- 準備
  開発環境に接続している評価ボード (A) のみを使用します。10.1.2　使用準備（セルフテスト）を参照してください。

- 操作
  1. 評価ボード (A) のメインメニューから [7] を選択し、セルフテストメニューを表示させます。
     テストを行いたい項目を選択してください。
  2. セルフテストメニューの [5] 外部ループバックモードまたは [6] 内部ループバックモードから選択してください。(デフォルトは、内部ループバックモードです)

- 結果
  下記に送信バッファからメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するセルフテストの結果を示します。

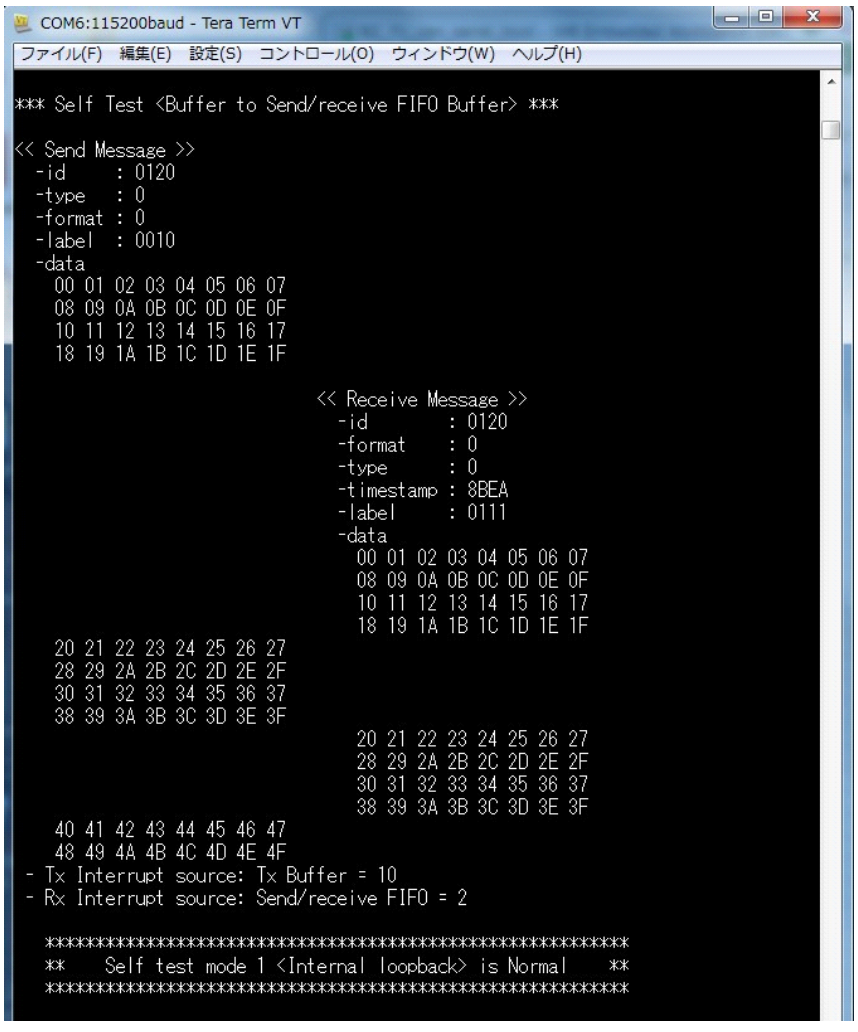


図 10.12　サンプルプログラムのセルフテスト結果例

10.2　使用割り込み一覧

表 10.2 にサンプルコードで使用する割り込みを示します。

表10.2　サンプルコードで使用する割り込み

割り込み（要因ID）	優先度	処理概要
CANグローバル・エラー (CANGE)	CAN_IR_PRIORITY_262_CANERR_GL	グローバル・エラー発生処理（ベクタ番号：262） • DLCエラー検出 • FIFOメッセージ・ロスト検出 • 送信履歴バッファ・オーバフロー検出
CAN0エラー (CANIE0)	CAN_IR_PRIORITY_263_CANERR_CH0	チャネル0エラー発生処理 (ベクタ番号：263) • チャネルバスエラー検出 • エラー・ワーニング検出 • エラー・パッシブ検出 • バスオフ開始検出 • バスオフ復帰検出 • オーバロード検出 • チャネルバスロック検出 • アービトレーション・ロスト検出 • スタッフ・エラー検出 • フォーム・エラー検出 • ACKエラー検出 • CRCエラー検出 • レセシブビットエラー検出 • ドミナントビットエラー検出 • ACKデリミタエラー検出
CAN1エラー (CANIE1)	CAN_IR_PRIORITY_264_CANERR_CH1	チャネル0エラー発生処理 （ベクタ番号：264）同上
CAN受信FIFO(CANRFI)	CAN_IR_PRIORITY_104_CANRFI	受信FIFOバッファメッセージ受信処理（ベクタ番号：104）
CAN0送受信FIFO受信完了 (CANFIR0)	CAN_IR_PRIORITY_105_CANFIR0	送受信FIFO（受信モード）バッファメッセージ受信処理（ベクタ番号：105）
CAN0送信(CANTI0)	CAN_IR_PRIORITY_106_CANTI0	送信完了処理 （ベクタ番号：106） • 送信完了検出 • 送信アボート完了検出 • 送受信FIFO（送信モード）送信割り込み要求検出
CAN1送受信FIFO受信完了 (CANFIR1)	CAN_IR_PRIORITY_107_CANFIR1	送受信FIFO（受信モード）バッファメッセージ受信処理（ベクタ番号：107）
CAN1送信(CANTI1)	CAN_IR_PRIORITY_108_CANTI1	送信完了処理（ベクタ番号：108） • 送信完了検出 • 送信アボート完了検出 • 送受信FIFO（送信モード）送信割り込み要求検出

10.3 固定幅整数一覧

表 10.3 にサンプルコードで使用する固定幅整数を示します。これらの型は標準ライブラリにて定義されています。

表10.3 サンプルコードで使用する固定幅整数

シンボル	内容
int8_t	8ビット整数、符号あり
int16_t	16ビット整数、符号あり
int32_t	32ビット整数、符号あり
int64_t	64ビット整数、符号あり
uint8_t	8ビット整数、符号なし
uint16_t	16ビット整数、符号なし
uint32_t	32ビット整数、符号なし
uint64_t	64ビット整数、符号なし

10.4　定数 / エラーコード一覧

表 10.4 にサンプルコードで使用する定数を示します。

表10.4　サンプルコードで使用する定数 (1 / 3)

定数名	設定値	内容
CAN_NUM	2	CANモジュールのチャネル数
CAN_CH_0	0	チャネル0（CAN0）
CAN_CH_1	1	チャネル1（CAN1）
CH_BUFFER_MAX	16	各チャネルで使用可能な送信バッファ数
CH_FIFO_BUFFER_MAX	3	各チャネルで使用可能な送受信FIFOバッファ数
DATA_MAX	8	1回の送信可能なメッセージデータ数
CAN_TX_BUFFER	0	状態フラグ（送信バッファ使用）
CAN_TX_FIFO	1	状態フラグ（送受信FIFOバッファ使用）
CAN_TX_HISTORY	2	状態フラグ（送信履歴）
CAN_TX_QUEUE	3	状態フラグ（送信キュー）
CAN_RX_BUFFER	0	状態フラグ（受信バッファ使用）
CAN_RX_RX_FIFO	1	状態フラグ（受信FIFOバッファ使用）
CAN_RX_FIFO	2	状態フラグ（送受信FIFO（受信モード）使用）
CAN_MODULE_ON	0	ストップ状態解除
CAN_MODULE_OFF	1	ストップ状態に遷移
CAN_STANDARD	0	標準ID
CAN_EXTENDED	1	拡張ID
CAN_DATA_FRAME	0	データ・フレーム
CAN_REMOTE_FRAME	1	リモート・フレーム
CAN_RULE_PAGE_MAX	8	受信ルール・ページ最大数
CAN_RULE_TABLE_MAX	16	受信ルール・テーブル最大数
CAN_RX_FIFO_BUFFER_MAX	8	受信FIFOバッファ最大数
CAN_RX_BUFFER_MAX	32	受信バッファ最大数
CAN_RULE_NUM_MAX	64	受信ルール最大数
CAN_RX_MODE	0	受信モード
CAN_TX_MODE	1	送信モード
CAN_GATEWAY_MODE	2	ゲートウェイモード
CAN_FIFO_MSG_0	0	送受信FIFOバッファ段数（0メッセージ）
CAN_FIFO_MSG_4	1	送受信FIFOバッファ段数（4メッセージ）
CAN_FIFO_MSG_8	2	送受信FIFOバッファ段数（8メッセージ）
CAN_FIFO_MSG_16	3	送受信FIFOバッファ段数（16メッセージ）
CAN_FIFO_MSG_32	4	送受信FIFOバッファ段数（32メッセージ）
CAN_FIFO_MSG_48	5	送受信FIFOバッファ段数（48メッセージ）
CAN_FIFO_MSG_64	6	送受信FIFOバッファ段数（64メッセージ）
CAN_FIFO_MSG_128	7	送受信FIFOバッファ段数（128メッセージ）
GL_MODE_STOP	0	グローバル・ストップ・モード
GL_MODE_RESET	1	グローバル・リセット・モード
GL_MODE_TEST	2	グローバル・テスト・モード
GL_MODE_OPE	3	グローバル動作モード
CAN_GL_OPE	0	グローバル動作モードへ遷移
CAN_GL_RESET	1	グローバル・リセット・モードへ遷移
CAN_GL_TEST	2	グローバル・テスト・モードへ遷移

表10.4　サンプルコードで使用する定数 (2 / 3)

定数名	設定値	内容
CH_MODE_STOP	0	チャネル・ストップ・モード
CH_MODE_RESET	1	チャネル・リセット・モード
CH_MODE_WAIT	2	チャネル待機モード
CH_MODE_COMM	3	チャネル通信モード
CAN_CH_COMM	0	チャネル通信モードへ遷移
CAN_CH_RESET	1	チャネル・リセット・モードへ遷移
CAN_CH_WAIT	2	チャネル待機モードへ遷移
GL_TEST_RAMTEST	0	RAMテスト
GL_TEST_COMMTEST	1	チャネル間通信テスト
CH_TEST_STANDARD	0	標準テストモード
CH_TEST_LISTENONLY	1	リッスンオンリモード
CH_TEST_SELF0	2	セルフテストモード0（外部ループバックモード）
CH_TEST_SELF1	3	セルフテストモード1（内部ループバックモード）
CANCLKA_CLK	24000000u	CANクロック（24MHz）
CANCLKB_CLK	25000000u	CANクロック（25MHz）
CAN_INTR_DISABLE	0	割り込み禁止
CAN_INTR_ENABLE	1	割り込み許可
CAN_IR_PRIORITY_262_CANERR_GL	3	優先順位（CANグローバルエラー）
CAN_IR_PRIORITY_263_CANERR_CH0	4	優先順位（CAN0エラー）
CAN_IR_PRIORITY_264_CANERR_CH1	4	優先順位（CAN1エラー）
CAN_IR_PRIORITY_104_CANRFI	5	優先順位（CAN受信FIFO）
CAN_IR_PRIORITY_105_CANFIR0	5	優先順位（CAN0送受信FIFO受信完了）
CAN_IR_PRIORITY_106_CANTI0	5	優先順位（CAN0送信）
CAN_IR_PRIORITY_107_CANFIR1	5	優先順位（CAN1送受信FIFO受信完了）
CAN_IR_PRIORITY_108_CANTI1	5	優先順位（CAN1送信）
CAN_HVA_WRITE_DATA	0u	HVA書き込みデータ
CAN_OK	0u	戻り値：正常
CAN_EMPTY	1u	戻り値：バッファエンプティ
CAN_NG	0xFFFFFFFFu	戻り値：エラー発生
CAN_INTR_TX_END	1	チャネル送信割り込み要因：送信完了
CAN_INTR_ABORT_END	2	チャネル送信割り込み要因：アボート送信完了
CAN_INTR_FIFO_REQ	3	チャネル送信割り込み要因：送受信FIFO（送信モード）送信完了
CAN_INTR_QUEUE_REQ	4	チャネル送信割り込み要因：送信キュー要求
CAN_INTR_HISTORY_REQ	5	チャネル送信割り込み要因：送信履歴要求
CAN_INTR_FIFO_EMPTY	1	受信FIFOバッファエンプティ
CAN_INTR_FIFO_FULL	2	受信FIFOバッファフル
CAN_INTR_FIFO_LOST	3	受信FIFOバッファメッセージ・ロスト
CAN_INTR_FIFO_TX_MESSAGE	4	送受信FIFO送信割り込み要求
CAN_INTR_FIFO_RX_MESSAGE	5	送受信FIFO受信割り込み要求
CAN_BUS_ERR	1	エラー・フラグ（バス・エラー）
CAN_ERR_WARNING	2	エラー・フラグ（エラー・ワーニング）
CAN_ERR_PASSIVE	3	エラー・フラグ（エラー・パッシブ）
CAN_BUS_OFF_START	4	エラー・フラグ（バスオフ開始）
CAN_BUS_OFF_RETURN	5	エラー・フラグ（バスオフ復帰）
CAN_OVER_LOAD	6	エラー・フラグ（オーバロード）

表10.4　サンプルコードで使用する定数 (3 / 3)

定数名	設定値	内容
CAN_BUS_LOCK	7	エラー・フラグ（チャネルバスロック）
CAN_ARBITRATION_LOST	8	エラー・フラグ（アービトレーション・ロスト）
CAN_STAFF_ERR	9	エラー・フラグ（スタッフ・エラー）
CAN_FORM_ERR	10	エラー・フラグ（フォームエラー）
CAN_ACK_ERR	11	エラー・フラグ（ACKエラー）
CAN_CRC_ERR	12	エラー・フラグ（CRCエラー）
CAN_RECESSIVE_BIT_ERR	13	エラー・フラグ（レセシブビットエラー）
CAN_DOMINANT_BIT_ERR	14	エラー・フラグ（ドミナントビットエラー）
CAN_ACK_DELIMITER_ERR	15	エラー・フラグ（ACKデリミタエラー）
CAN_DLC_ERR	1	エラー・フラグ（DLCエラー）
CAN_FIFO_MSG_LOST_ERR	2	エラー・フラグ（FIFOメッセージ・ロスト）
CAN_HISTORY_OVERFLOW_ERR	3	エラー・フラグ（送信履歴バッファ・オーバフロー）
CAN0_CRXD0_P30_VAL	0x10	MPC：CAN0 CRXD0 設定値（未使用）
CAN0_CRXD0_PC6_VAL	0x10	MPC：CAN0 CRXD0 設定値
CAN0_CTXD0_P60_VAL	0x10	MPC：CAN0 CTXD0 設定値（未使用）
CAN0_CTXD0_P67_VAL	0x10	MPC：CAN0 CTXD0 設定値
CAN1_CRXD1_PC3_VAL	0x10	MPC：CAN1 CRXD1 設定値（未使用）
CAN1_CRXD1_PC7_VAL	0x10	MPC：CAN1 CRXD1 設定値
CAN1_CTXD1_P61_VAL	0x10	MPC：CAN1 CTXD1 設定値（未使用）
CAN1_CTXD1_P66_VAL	0x10	MPC：CAN1 CTXD1 設定値
CAN1_CTXD1_PB3_VAL	0x10	MPC：CAN1 CTXD1 設定値（未使用）
CAN_GL_STATUS_BIT	0x00000007u	RSCAN0GSTS レジスタマスクビット
CAN_CH_STATUS_BIT	0x00000007u	RSCAN0CmSTS レジスタマスクビット
GCFG_REG_INIT	0x00000013u	RSCAN0GCFG レジスタ初期値
TMIEC0_REG_DISABLE_LOW	0x0000FFFFu	RSCAN0TMIEC0 レジスタ TMIEp(p = 15～0)マスクビット
TMIEC0_REG_DISABLE_HIGH	0xFFFF0000u	RSCAN0TMIEC0 レジスタ TMIEp(p = 31～16)マスクビット
CAN_CH_STOP_MODE	0x00000004u	RSCAN0CmCTR レジスタチャネルストップモード
CAN_REL_CH_STOP_MODE	0xFFFFFFFBu	RSCAN0CmCTR レジスタチャネルストップモード解除
TIME_QUANTUM_MIN	8	(*1) ビット・タイミング設定範囲 SS + TSEG1 + TSEG2 = 8 ~ 25Tq の範囲で設定
TIME_QUANTUM_MAX	25	(*1) ビット・タイミング設定範囲 SS + TSEG1 + TSEG2 = 8 ~ 25Tq の範囲で設定
SAMPLE_POINT	0.666666667	(*1) サンプル・ポイント(%) 本サンプルでは、2/3の値をサンプル・ポイントにしています。
FIFO_UPDATE	0x000000FFu	FIFOバッファポインタ制御設定値

注1.　6.3.1　CANビット・タイミングの設定の（1）　ビット・タイミングの条件、ならびにRZT1グループユーザーズマニュアル ハードウェア編の「35.9.1.2　ビット・タイミングの設定」を参照してください。

10.5　関数一覧

表 10.5 に関数一覧を示します。

表10.5　関数一覧

関数名	概要
R_CAN_Open	CANモジュールの起動
R_CAN_Close	CANモジュールの停止
R_CAN_GlobalControl	グローバル・モードの遷移
R_CAN_ChannelControl	チャネル・モードの遷移
R_CAN_SetBitrate	通信速度の設定
R_CAN_UseBufferEntry	送信・受信で使用するバッファの登録
R_CAN_SetRxFifoBuffer	受信FIFOバッファの設定
R_CAN_SetFifoBuffer	送受信FIFOバッファの設定
R_CAN_ReleaseFifoBuffer	送受信FIFOバッファの開放
R_CAN_ReleaseRxFifoBuffer	受信FIFOバッファの開放
R_CAN_ReleaseBuffer	送信バッファ、受信バッファの開放
R_CAN_GetTxBufferStatus	送信バッファステータスの読み出し
R_CAN_WriteBuffer	送信メッセージを送信バッファに書き込む
R_CAN_GetFifoStatus	送受信FIFOバッファステータスの読み出し
R_CAN_WriteFifo	送信メッセージを送受信FIFOバッファに書き込む
R_CAN_Tx	送信開始
R_CAN_RxSet	受信設定
R_CAN_ReadBuff	受信バッファから受信メッセージの読み出し
R_CAN_ReadRxFifo	受信FIFOバッファから受信メッセージの読み出し
R_CAN_ReadFifo	送受信FIFOバッファから受信メッセージの読み出し
R_CAN_GetFifoMessageNum	送受信FIFOバッファの未読メッセージ数の取得
R_CAN_GetRxFifoMessageNum	受信FIFOバッファの未読メッセージ数の取得
R_CAN_SetCommTestMode	テスト設定
R_CAN_ResetTestMode	テスト設定を解除しチャネル通信モードへ遷移
R_CAN_SetInterruptHandler	割り込みハンドラの登録
R_CAN_SetInterruptEnableDisable	CANモジュール割り込みベクタの割り込み許可・禁止
R_CAN_GetInterruptSource	割り込み要因の取得
R_CAN_ClearInterruptSource	割り込み要因のクリア
main	サンプルプログラムメイン処理

10.6 構造体 / 共用体 / 列挙型一覧

以下に、サンプルコードで使用する構造体 / 共用体 / 列挙型を示します。

- can_vector_t

  RSCAN 割り込みベクタ使用／未使用選択用構造体

```
typedef struct {
    union {
        uint8_t         BYTE;
        struct {
            uint8_t         CANGE:1;        /* CAN global error interrupt */
            uint8_t         CANIE0:1;       /* CAN0 error interrupt */
            uint8_t         CANIE1:1;       /* CAN1 error interrupt */
            uint8_t         CANRFI:1;       /* CAN reception FIFO interrupt */
            uint8_t         CANFIR0:1;      /* CAN0 transmission-and-reception FIFO interrupt */
            uint8_t         CANTI0:1;       /* CAN0 transmission interrupt */
            uint8_t         CANFIR1:1;      /* CAN1 transmission-and-reception FIFO interrupt */
            uint8_t         CANTI1:1;       /* CAN1 transmission interrupt */
        } BIT;
    } SELECT;
} can_vector_t;
```

- can_callback_t

  コールバック関数登録用構造体

```
typedef struct {
    void    (*pintr_ge)(void);          /* Pointer to user callback function */
    void    (*pintr_ie0)(void);         /* Pointer to user callback function */
    void    (*pintr_ie1)(void);         /* Pointer to user callback function */
    void    (*pintr_rfi)(void);         /* Pointer to user callback function */
    void    (*pintr_fir0)(void);        /* Pointer to user callback function */
    void    (*pintr_ti0)(void);         /* Pointer to user callback function */
    void    (*pintr_fir1)(void);        /* Pointer to user callback function */
    void    (*pintr_ti1)(void);         /* Pointer to user callback function */
} can_callback_t;
```

- can_handle_t

  コールバック関数登録用構造体

```
typedef struct {
    bool            ch_opened;
    can_callback_t  can_callback;
} can_handle_t;
```

- can_rx_rule_t
  受信ルールテーブル構造体

```
typedef struct {
    uint32_t        buf_type;       /* Type of the buffer */
                                    /* Transmission: CAN_TX_BUFFER, CAN_TX_FIFO */
                                    /* Reception: CAN_RX_BUFFER, CAN_RX_RX_FIFO, CAN_RX_FIFO */
    uint32_t        rule_page;      /* Reception rule page number */
    uint32_t        rule_table;     /* Reception rule table number */
    uint32_t        rule_id;        /* Message ID */
    uint32_t        rule_type;      /* Message type (data frame/remote frame) */
    uint32_t        rule_format;    /* Message format (standard ID/extended ID) */
    uint32_t        rule_label;     /* Message label */
    uint32_t        rule_dlc_check; /* DLC checking */
    uint32_t        rule_mask;      /* Mask */
} can_rx_rule_t;
```

- udata_t
  4 バイト長データ共用体

```
typedef union udata {
    uint32_t        LONG;
    uint8_t         BYTE[4];
} udata_t;
```

- can_tx_message_t
  送信メッセージデータ構造体

```
typedef struct {
    uint32_t        id;             /* Message ID */
    uint32_t        type;           /* 0: Data frame/1: Remote frame */
    uint32_t        format;         /* 0: Standard ID/1: Extended ID */
    uint32_t        length;         /* Message data length */
    udata_t         data_h;         /* Message data */
    udata_t         data_l;         /* Message data */
    uint32_t        history_label;  /* Label */
    uint32_t        buf_type;       /* Type of the buffer to be used (buffer or FIFO) */
} can_tx_message_t;
```

- can_rx_message_t

  受信メッセージデータ構造体

```
typedef struct {
    uint32_t        id;             /* Message ID */
    uint32_t        format;         /* 0: Standard ID/1: Extended ID */
    uint32_t        type;           /* 0: Data frame/1: Remote frame */
    uint16_t        timestamp;      /* Timestamp data */
    uint16_t        label;          /* Label information */
    uint32_t        length;         /* Message length */
    udata_t         data_h;         /* Message data */
    udata_t         data_l;         /* Message data */
} can_rx_message_t;
```

- gl_err_source_t

  グローバル・エラー発生要因カウント構造体

```
typedef struct {
    uint32_t        total_num;              /* Total count of the global errors */
    uint32_t        dlc_error_num;          /* DLC error count */
    uint32_t        fifo_message_lost_num;  /* FIFO message loss count */
    uint32_t        history_overflow_num;   /* Transmission history overflow count */
} gl_err_source_t;
```

- ch_err_source_t

  チャネル・エラー発生要因カウント構造体

```
typedef struct {
    uint32_t        total_num;              /* Total count of the channel errors */
    uint32_t        bus_error_num;          /* Bus error count */
    uint32_t        error_warning_num;      /* Error warning count */
    uint32_t        error_passive_num;      /* Error passive count */
    uint32_t        bus_off_start_num;      /* Bus-off start count */
    uint32_t        bus_off_return_num;     /* Bus-off return count */
    uint32_t        overload_num;           /* Overload count */
    uint32_t        bus_lock_num;           /* Bus lock-up count */
    uint32_t        arbitration_lost_num;   /* Arbitration loss count */
    uint32_t        staff_error_num;        /* Staff error count */
    uint32_t        form_error_num;         /* Form error count */
    uint32_t        ack_error_num;          /* ACK error count */
    uint32_t        crc_error_num;          /* CRC error count */
    uint32_t        recessive_bit_error_num;/* Recessive bit error count */
    uint32_t        dominant_bit_error_num; /* Dominant bit error count */
    uint32_t        ack_delimiter_error_num;/* ACK delimiter error count */
} ch_err_source_t;
```

- ch_tx_source_t
  送信割り込み発生要因カウント構造体

```
typedef struct {
    uint32_t        total_num;          /* Total count of the transmission interrupts */
    uint32_t        intr_sorce;         /* Interrupt source */
    uint32_t        tx_buf_end_num;     /* Count of successful transmissions from the transmission buffer */
    uint32_t        tx_fifo_end_num;    /* Count of successful transmissions from the transmission-and-reception
                                           FIFO buffer in transmission mode */
    uint32_t        tx_abort_num;       /* Transimission abortion count */
    uint32_t        tx_queue_num;       /* Transmission queue count */
    uint32_t        tx_history_num;     /* Transmission history data count */
} ch_tx_source_t;
```

- can_intr_source_t
  割り込み要因情報構造体

```
typedef struct {
    uint32_t        rx_fifo_num;        /* Reception FIFO interrupt count */
    uint32_t        ch_fifo_receive_num;/* Transmission-and-reception FIFO reception interrupt count */
    ch_tx_source_t  tx_source;          /* Transmission interrupt source */
    gl_err_source_t gl_err;             /* Global error interrupt source */
    ch_err_source_t ch_err;             /* Channel error interrupt source */
} can_intr_source_t;
```

- can_used_buffer_t
  使用バッファ情報構造体

```
typedef struct {
    uint32_t        use_tx_buf_no;      /* Transmission buffer number */
    uint32_t        use_rx_buf_no;      /* Reception buffer number */
    uint32_t        use_rx_fifo_no;     /* Reception FIFO buffer number */
    uint32_t        use_fifo_txmode_no; /* Number of the transmission-and-reception FIFO buffer in transmission
                                           mode */
    uint32_t        use_fifo_rxmode_no; /* Number of the transmission-and-reception FIFO buffer in reception mode
                                           */
    uint32_t        use_fifo_link_buf_no;/* Number of the buffer to which the FIFO buffer is linked to */
} can_used_buffer_t;
```

- can_tx_intr_sts_t
  送信割り込み要求のステータス情報構造体

```
typedef struct {
    union {
        uint8_t     BYTE;
        struct {
            uint8_t     TMTRF:2;    /* Result of transmission from the transmission buffer */
            uint8_t     CFTXIF:1;   /* Request for transmission-and-reception FIFO transmission interrupt */
            uint8_t     TXQIF:1;    /* Request for transmission queue interrupt */
            uint8_t     THLIF:1;    /* Request for transmission history interrupt */
            uint8_t     :3;
        } BIT;
    } STS;
} can_tx_intr_sts_t;
```

- can_tx_history_t
  送信履歴情報構造体

```
typedef struct {
    uint32_t    buf_type;   /* Buffer type */
    uint32_t    buf_no;     /* Buffer number */
    uint32_t    label;      /* Label data */
} can_tx_history_t;
```

10.7　関数仕様

サンプルコードの関数仕様を示します。

10.7.1　R_CAN_Open

R_CAN_Open	
概　要	本モジュールを使用する際に最初に使用する関数です。
ヘッダ	r_can_api.h
宣　言	void R_CAN_Open(uint32_t ch, uint32_t frequency);
説　明	CAN 通信を開始するための初期設定をします。引数で、使用するチャネル、通信速度を設定します。チャネルの状態が “ 未初期化状態の場合、次の処理を行います。 ー API 使用する各種変数の初期化 ー CAN のモジュールストップ状態の解除 ー ポートの入出力設定 ー CAN モジュールをグローバル・リセット・モードへ遷移 ー 選択チャネルをチャネル・リセット・モードへ遷移 ー CAN 通信で使用する CAN レジスタの初期化 ー CAN 通信の通信速度の設定
引　数	uint32_t ch　：チャネル番号 uint32_t frequency　：通信速度
リターン値	なし
補足	なし

10.7.2　R_CAN_Close

R_CAN_Close	
概　要	CAN の通信を終了し、CAN モジュールを開放する際に使用する関数です。
ヘッダ	r_can_api.h
宣　言	void R_CAN_Close(uint32_t ch);
説　明	CAN 通信を終了するための設定をします。引数で指定したチャネルを無効にします。本関数では次の処理を行います。 ー CAN のモジュールストップ状態への遷移 ー CAN 割り込みの禁止 再度通信を開始するには、R_CAN_Open ( ) (初期化関数) をコールする必要があります。通信中に強制的に停止した場合、その通信は保証しません。
引　数	uint32_t ch　：チャネル番号
リターン値	なし
補足	なし

10.7.3　R_CAN_GlobalControl

R_CAN_GlobalControl		
概　要	RSCAN モジュール全体の制御を行います。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_GlobalControl(uint32_t mode);	
説　明	RSCAN モジュール全体の状態を制御するグローバル・モードを引数で設定したモードに遷移させます。 － グローバル・ストップ・モード モジュール全体のクロックを停止させ、低消費電力を実現。 － グローバル・リセット・モード モジュール全体の初期設定を行うためのモード。 － グローバル・テスト・モード テスト設定（RAM テスト、チャネル間通信テスト）のためのモード － グローバル動作モード モジュール全体を動作可能にします。※ 通常は、このモードとなります。	
引　数	uint32_t mode	：グローバル・モード GL_MODE_OPE：グローバル動作モードに遷移する GL_MODE_RESET：グローバル・リセット・モードに遷移する GL_MODE_STOP：グローバル・ストップ・モードに遷移する GL_MODE_TEST：グローバル・テスト・モードに遷移する
リターン値	なし	
補足	なし	

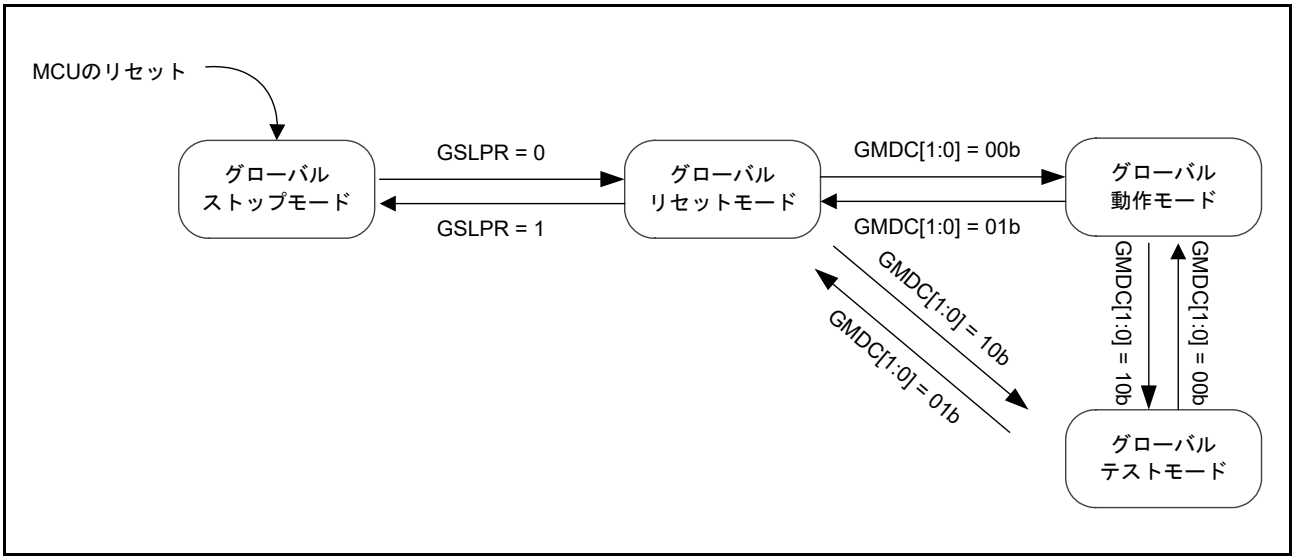


図 10.13　グローバル・モードの遷移図

10.7.4　R_CAN_ChannelControl

R_CAN_ChannelControl		
概　要	チャネルを制御します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_ChannelControl(uint32_t ch, uint32_t mode);	
説　明	引数で指定したチャネルの状態を設定します。 － チャネル・ストップ・モード チャネルのクロックを停止させるモードです。 － チャネル・リセット・モード チャネルの初期設定を行うためのモードです。 － チャネル待機モード CAN 通信を停止させ、チャネルのテストを許可するモードです。 － チャネル通信モード CAN 通信を行うモードです。※ 通常は、このモードです。	
引　数	uint32_t ch	：チャネル番号
	uint32_t mode	：チャネル・モード CH_MODE_STOP：チャネル・ストップ・モードへ遷移する CH_MODE_RESET：チャネル・リセット・モードへ遷移する CH_MODE_WAIT：チャネル待機モードへ遷移する CH_MODE_COMM：チャネル通信モードへ遷移する
リターン値	なし	
補足	なし	

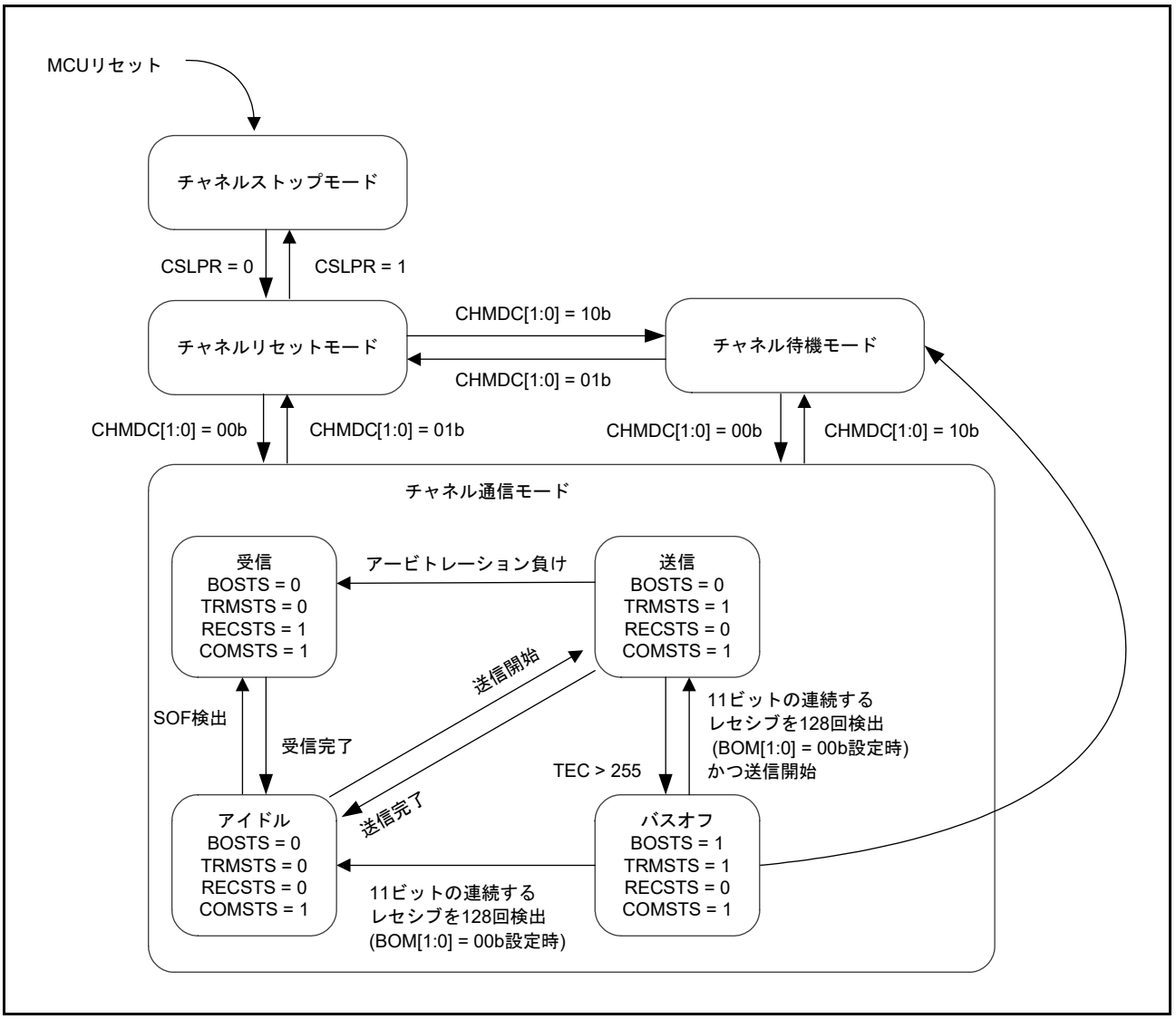


図 10.14　チャネル・モードの状態遷移図

10.7.5　R_CAN_SetBitrate

R_CAN_SetBitrate	
概　要	CAN 通信の通信速度を設定します。
ヘッダ	r_can_api.h
宣　言	void R_CAN_SetBitrate(uint32_t ch, uint32_t frequency);
説　明	引数で指定した値を用いて CAN 通信の通信速度を決定します。 6.3　通信速度を参照してください。
引　数	uint32_t ch　：チャネル番号 uint32_t frequency　：通信速度 BAUD_RATE_1MBPS：1Mbps BAUD_RATE_500KBPS：500Kbps BAUD_RATE_125KBPS：125Kbps
リターン値	なし
補足	なし

10.7.6　R_CAN_UseBufferEntry

R_CAN_UseBufferEntry	
概　要	CAN 通信で使用する各種バッファの情報を登録します。
ヘッダ	r_can_api.h
宣　言	void R_CAN_UseBufferEntry(can_used_buffer_t * obj);
説　明	CAN 通信で使用する各種バッファの情報を登録します。 ー 送信バッファの番号 ー 受信バッファの番号 ー 受信 FIFO バッファの番号 ー 送受信 FIFO（送信モード）バッファの番号 ー 送受信 FIFO（受信モード）バッファの番号 ー送受信 FIFO（送信モード）バッファにリンクさせる送信バッファの番号
引　数	can_used_buffer_t * obj　：バッファ関連情報構造体のポインタ
リターン値	なし
補足	なし

10.7.7　R_CAN_SetRxFifoBuffer

R_CAN_SetRxFifoBuffer	
概　要	受信 FIFO バッファの使用を許可します。
ヘッダ	r_can_api.h
宣　言	void R_CAN_SetRxFifoBuffer(uint32_t ch, can_rfcc_t * obj);
説　明	CAN 通信で、受信 FIFO バッファを使ったメッセージ受信を許可します。 メッセージを受信すると、CAN 受信 FIFO 割り込みが発生します。 この関数を呼び出す前に、R_CAN_UseBufferEntry() 関数にて、事前に使用する受信 FIFO バッファの番号を登録しておいてください。 受信したメッセージは、R_CAN_ReadRxFifo() 関数を使って読み出すことができます。 7.3　受信 FIFO バッファを用いた受信を参照してください。
引　数	uint32_t ch　：チャネル番号 can_rfcc_t * obj　：受信 FIFO バッファ設定情報
リターン値	なし
補足	なし

10.7.8　R_CAN_SetFifoBuffer

R_CAN_SetFifoBuffer		
概　要	送受信 FIFO バッファの使用を許可します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_SetFifoBuffer(uint32_t ch, uint32_t mode, can_cfcc_t * obj);	
説　明	CAN 通信で、送受信 FIFO バッファを使ったメッセージ送信、メッセージ受信を許可します。 － 送信モード 送受信 FIFO（送信モード）バッファに書き込んだメッセージの送信が完了すると、送信完了割り込み（割り込み要因：送受信 FIFO 送信完了）が発生します。 8.3　送受信 FIFO バッファを用いた送信を参照してください。 － 受信モード 送受信 FIFO（受信モード）バッファにメッセージを受信すると、送受信 FIFO 受信完了の割り込みが発生します。 この関数を呼び出す前に、R_CAN_UseBufferEntry() 関数にて、事前に使用する送受信 FIFO バッファの番号を登録しておいてください。 受信したメッセージは、R_CAN_ReadFifo () 関数を使って読み出すことができます。 7.4　送受信 FIFO バッファを用いた受信を参照してください。	
引　数	uint32_t ch	：チャネル番号
	uint32_t mode	：モード CAN_TX_MODE：送信モード CAN_RX_MODE：受信モード
	can_cfcc_t * obj	：送受信 FIFO バッファ設定情報
リターン値	なし	
補足	なし	

10.7.9　R_CAN_ReleaseFifoBuffer

R_CAN_ReleaseFifoBuffer		
概　要	CAN 通信で使用した送受信 FIFO バッファを開放します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_ReleaseFifoBuffer(uint32_t ch, uint32_t mode);	
説　明	CAN 通信で使用した送受信 FIFO バッファを開放します。	
引　数	uint32_t ch	：チャネル番号
	uint32_t mode	：モード CAN_TX_MODE：送信モード CAN_RX_MODE：受信モード
リターン値	なし	
補足	なし	

10.7.10　R_CAN_ReleaseRxFifoBuffer

R_CAN_ReleaseRxFifoBuffer		
概　要	CAN 通信で使用した受信 FIFO バッファを開放します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_ReleaseRxFifoBuffer(uint32_t ch);	
説　明	CAN 通信で使用した受信 FIFO バッファを開放します。	
引　数	uint32_t ch	：チャネル番号
リターン値	なし	
補足	なし	

10.7.11　R_CAN_ReleaseBuffer

R_CAN_ReleaseBuffer		
概　要	CAN 通信で使用したバッファを開放します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_ReleaseBuffer(uint32_t ch, uint32_t mode);	
説　明	CAN 通信で使用したバッファを開放します。	
引　数	uint32_t ch	：チャネル番号
	uint32_t mode	：モード CAN_TX_MODE：送信モード CAN_RX_MODE：受信モード
リターン値	なし	
補足	なし	

10.7.12　R_CAN_GetTxBufferStatus

R_CAN_GetTxBufferStatus		
概　要	送信バッファの状態を読み出します。	
ヘッダ	r_can_api.h	
宣　言	uint32_t R_CAN_GetTxBufferStatus(uint32_t ch);	
説　明	送信バッファの状態を読み出します。	
引　数	uint32_t ch	：チャネル番号
リターン値	0：送信中で無い 1：送信中	
補足	なし	

10.7.13　R_CAN_WriteBuffer

R_CAN_WriteBuffer	
概　要	送信バッファにメッセージを書き込みます。
ヘッダ	r_can_api.h
宣　言	void R_CAN_WriteBuffer(uint32_t ch, can_tx_message_t * msg);
説　明	送信メッセージを送信バッファに書き込む処理です。 送信するメッセージの ID、データフォーマット、送信されるメッセージのデータ長、ラベル情報、送信データを can_tx_message_t 構造体に格納し、本関数の引数に渡します。 8.2　送信バッファを用いた送信を参照してください。
引　数	uint32_t ch　：チャネル番号 can_tx_message_t * msg　：送信メッセージ情報
リターン値	なし
補足	なし

10.7.14　R_CAN_GetFifoStatus

R_CAN_GetFifoStatus	
概　要	送受信 FIFO バッファの状態を読み出します。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_GetFifoStatus(uint32_t ch, uint32_t mode);
説　明	送受信 FIFO バッファの状態を読み出します。
引　数	uint32_t ch　：チャネル番号 uint32_t mode　：モード CAN_TX_MODE：送信モード CAN_RX_MODE：受信モード
リターン値	0：送受信 FIFO バッファフルではない 1：送受信 FIFO バッファフル
補足	なし

10.7.15　R_CAN_WriteFifo

R_CAN_WriteFifo	
概　要	送受信 FIFO（送信モード）バッファにメッセージを書き込みます。
ヘッダ	r_can_api.h
宣　言	void R_CAN_WriteFifo(uint32_t ch, can_tx_message_t * msg);
説　明	送信メッセージを送受信 FIFO（送信モード）バッファに書き込む処理です。 送信するメッセージの ID、データフォーマット、送信されるメッセージのデータ長、ラベル情報、送信データを can_tx_message_t 構造体に格納し、本関数の引数に渡します。 8.3　送受信 FIFO バッファを用いた送信を参照してください。
引　数	uint32_t ch　：チャネル番号 can_tx_message_t * msg　：送信メッセージ情報
リターン値	なし
補足	なし

10.7.16　R_CAN_Tx

R_CAN_Tx	
概　要	CAN 通信の送信を開始します。
ヘッダ	r_can_api.h
宣　言	void R_CAN_Tx(uint32_t ch, can_tx_message_t * msg);
説　明	CAN 通信の送信を開始する処理です。 － 送信バッファを用いた送信 送信バッファの送信要求ビットを "1"（送信を要求する）にします。 － 送受信 FIFO（送信モード）バッファを用いた送信 送受信 FIFO バッファ許可ビットを "1"（送受信 FIFO バッファを使用する）にします。
引　数	uint32_t ch　：チャネル番号 can_tx_message_t * msg　：送信メッセージ情報
リターン値	なし
補足	なし

10.7.17　R_CAN_RxSet

R_CAN_RxSet	
概　要	受信を許可します。
ヘッダ	r_can_api.h
宣　言	void R_CAN_RxSet(uint32_t ch, can_rx_rule_t * rule);
説　明	メッセージを受信するための受信ルールを設定する処理です。 受信ルール情報を can_rx_rule_t 構造体に格納し、本関数の引数に渡します。 6.5　受信ルール・テーブルを参照してください。
引　数	uint32_t ch　：チャネル番号 can_rx_rule_t * rule　：受信ルール情報
リターン値	なし
補足	なし

10.7.18　R_CAN_ReadBuff

R_CAN_ReadBuff	
概　要	受信したメッセージを受信バッファから読み出します。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_ReadBuff(uint32_t ch, can_rx_message_t * obj);
説　明	受信メッセージを受信バッファから読み出す処理です。 7.2　受信バッファを用いた受信を参照してください。
引　数	uint32_t ch　：チャネル番号 can_rx_message_t * obj　：受信メッセージ格納領域
リターン値	CAN_OK：受信バッファからの読み出し正常 CAN_EMPTY：受信バッファに新しいメッセージがない
補足	なし

10.7.19　R_CAN_GetRxFifoMessageNum

R_CAN_GetRxFifoMessageNum	
概　要	受信 FIFO バッファの未読メッセージ数取得。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_GetRxFifoMessageNum(void);
説　明	受信 FIFO バッファの未読メッセージの数を返します。 7.3　受信 FIFO バッファを用いた受信を参照してください。
引　数	なし
リターン値	未読メッセージ数
補足	なし

10.7.20　R_CAN_ReadRxFifo

R_CAN_ReadRxFifo	
概　要	受信したメッセージを受信 FIFO バッファから読み出します。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_ReadRxFifo(can_rx_message_t * obj);
説　明	受信メッセージを受信バッファから読み出す処理です。 7.3　受信 FIFO バッファを用いた受信を参照してください。
引　数	can_rx_message_t * obj　：受信メッセージ格納領域
リターン値	CAN_OK：受信 FIFO バッファからの読み出し正常 CAN_EMPTY：受信 FIFO バッファに未読メッセージなし（バッファ空） CAM_LOST：FIFO メッセージ・ロスト
補足	なし

10.7.21　R_CAN_GetFifoMessageNum

R_CAN_GetFifoMessageNum	
概　要	送受信 FIFO バッファの未読メッセージ数の取得。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_GetFifoMessageNum(uint32_t ch);
説　明	送受信 FIFO バッファの未読メッセージ数を返します。 7.4　送受信 FIFO バッファを用いた受信を参照してください。
引　数	uint32_t ch　：チャネル番号
リターン値	未読メッセージ数
補足	なし

10.7.22　R_CAN_ReadFifo

R_CAN_ReadFifo	
概　要	受信したメッセージを送受信 FIFO バッファから読み出します。
ヘッダ	r_can_api.h
宣　言	uint32_t R_CAN_ReadFifo(uint32_t ch, can_rx_message_t * obj);
説　明	受信メッセージを受信バッファから読み出す処理です。 7.4　送受信 FIFO バッファを用いた受信を参照してください。
引　数	uint32_t ch　：チャネル番号 can_rx_message_t * obj　：受信メッセージ格納領域
リターン値	CAN_OK：送受信 FIFO バッファからの読み出し正常 CAN_EMPTY：送受信 FIFO バッファに未読メッセージなし（バッファ空） CAM_LOST：FIFO メッセージ・ロスト
補足	なし

10.7.23　R_CAN_SetCommTestMode

R_CAN_SetCommTestMode		
概　要	通信テストモードを選択します。	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_SetCommTestMode(uint32_t ch, uint32_t mode);	
説　明	通信テストモードを選択します。 通信テストモードで、以下の選択が可能です。 － 標準テストモード － リッスンオンリモード － セルフテストモード 0（外部ループバックモード） － セルフテストモード 1（内部ループバックモード）	
引　数	uint32_t ch	：チャネル番号
	uint32_t mode	：テストモード CH_TEST_STANDARD：標準テストモード CH_TEST_LISTENONLY：リッスンオンリモード CH_TEST_SELF0：セルフテストモード 0（外部ループバックモード） CH_TEST_SELF1：セルフテストモード 1（内部ループバックモード）
リターン値	なし	
補足	なし	

10.7.24　R_CAN_ResetTestMode

R_CAN_ResetTestMode		
概　要	通信テストのクリア	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_ResetTestMode(uint32_t ch);	
説　明	R_CAN_SetCommTestMode() 関数で設定した通信テストモードをクリアします。 通信テスト終了後、必ずこの API 関数をコールしてください。	
引　数	uint32_t ch	：チャネル番号
リターン値	なし	
補足	なし	

10.7.25　R_CAN_SetInterruptHandler

R_CAN_SetInterruptHandler	
概　要	CAN 通信で使用する割り込みハンドラから呼び出されるコールバック関数の登録
ヘッダ	r_can_api.h
宣　言	void R_CAN_SetInterruptHandler(uint32_t ch, can_callback_t * pcallback);
説　明	CAN 通信で使用する割り込みハンドラから呼び出されるコールバック関数を登録します。 CAN 通信の割り込みハンドラは以下のとおりです。 － CAN グローバルエラー － CAN0 エラー － CAN1 エラー － CAN 受信 FIFO － CAN0 送受信 FIFO 受信完了 － CAN0 送信 － CAN1 送受信 FIFO 受信完了 － CAN1 送信 * pcallback コールバック関数情報構造体のポインタ。 この構造体のメンバにコールバック関数名を記述します。使用しない場合は、NULL としてください。
引　数	uint32_t ch　：チャネル番号 can_callback_t * pcallback　：コールバック関数情報
リターン値	なし
補足	なし

10.7.26　R_CAN_SetInterruptEnableDisable

R_CAN_SetInterruptEnableDisable		
概　要	CAN 通信で使用する割り込みハンドラを許可／禁止	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_SetInterruptEnableDisable(uint32_t enable_disable);	
説　明	CAN 通信で使用する割り込みハンドラの許可／禁止を行います。 CAN 通信の割り込みハンドラは以下のとおりです。 － CAN グローバル・エラー － CAN0 エラー － CAN1 エラー － CAN 受信 FIFO － CAN0 送受信 FIFO 受信完了 － CAN0 送信 － CAN1 送受信 FIFO 受信完了 － CAN1 送信	
引　数	uint32_t enable_disable	：許可／禁止 CAN_INTR_DISABLE：禁止 CAN_INTR_ENABLE：許可
リターン値	なし	
補足	なし	

10.7.27　R_CAN_GetInterruptSource

R_CAN_GetInterruptSource		
概　要	割り込み要因の取得	
ヘッダ	r_can_api.h	
宣　言	void R_CAN_GetInterruptSource(can_intr_source_t * obj);	
説　明	各割り込み発生時の割り込み要因情報を返します。	
引　数	can_intr_source_t * obj	：割り込み要因情報格納領域
リターン値	なし	
補足	なし	

10.7.28　R_CAN_ClearInterruptSource

R_CAN_ClearInterruptSource	
概　要	割り込み要因情報のクリア
ヘッダ	r_can_api.h
宣　言	void R_CAN_ClearInterruptSource(void);
説　明	各割り込みの割り込み要因をクリアします。
引　数	なし
リターン値	なし
補足	なし

10.7.29　main

Main	
概　要	サンプルプログラムのメイン関数です。
ヘッダ	―
宣　言	void main(void)
説　明	サンプルプログラムのメイン処理です。 処理内容は 10.8　フローチャートを参照してください。
引　数	なし
リターン値	なし
補足	なし

10.8　フローチャート

10.8.1　　メイン処理

本サンプルプログラムは、メニューから確認項目を選択する方式をとっています。
メニューの内容は、10.1　**動作概要**を参照してください。

図 10.15 にサンプルコードのメイン処理のフローチャートを示します。

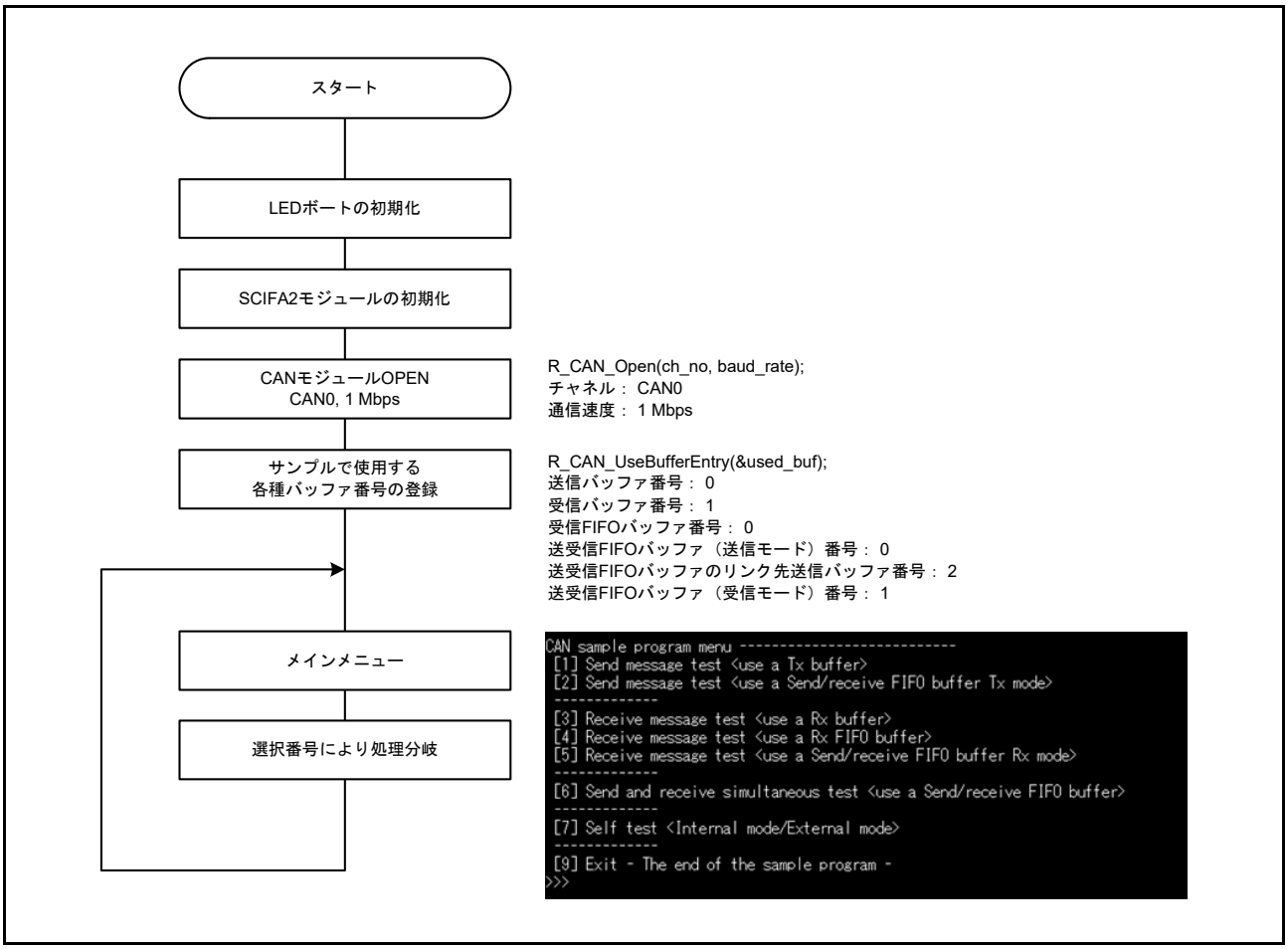


図 10.15　サンプルコードのメイン処理

10.8.2　送信テスト

送信テストは、「送信バッファを使ったメッセージ送信」と「送受信 FIFO バッファ (送信モード) を使ったメッセージ送信」の 2 通りがあり、メニューから選択することができます。

メニューの内容は、10.1　**動作概要**を参照してください。

（1）void tx_demo_buffer(void)
　　送信バッファを使ったメッセージ送信処理
（2）void tx_demo_fifo(void)
　　送受信 FIFO バッファ（送信モード）を使ったメッセージ送信処理
（3）uint32_t write_buffer(uint32_t msg_type, tx_data_t * obj)
　　送信バッファ関連レジスタに送信メッセージを書き込む処理
（4）uint32_t write_fifo(uint32_t msg_type, tx_data_t * obj)
　　送受信 FIFO バッファ関連レジスタに送信メッセージを書き込む処理

図 10.16 ～図 10.19 にそれぞれの機能のフローチャートを示します。

(1) 送信バッファを使ったメッセージ送信処理

関数名：void tx_demo_buffer(void)

以下に送信バッファを使ったメッセージ送信処理のフローチャートを示します。

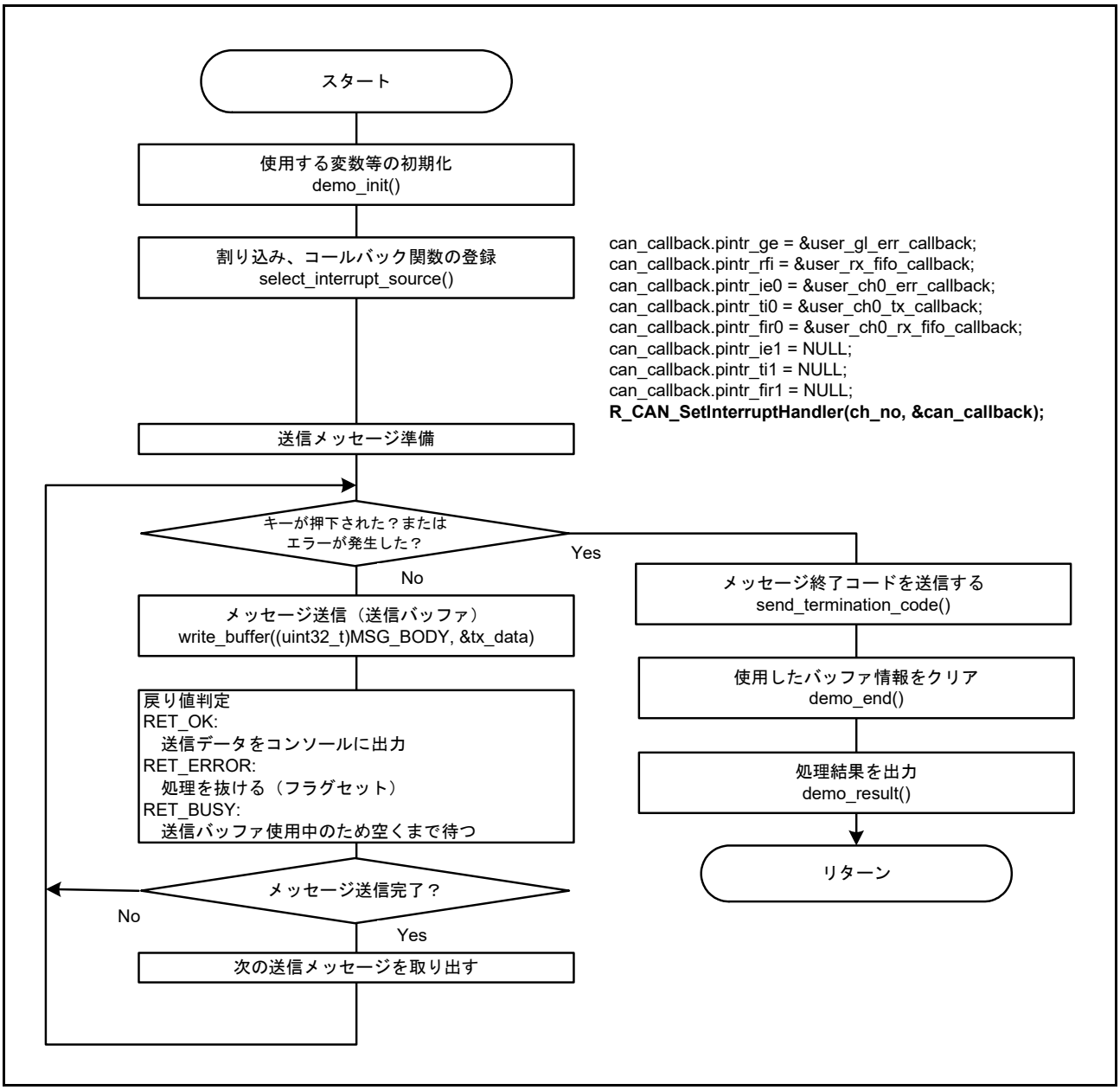


図 10.16　サンプルコードの送信バッファを使ったメッセージ送信処理

（2）送受信 FIFO バッファ（送信モード）を使ったメッセージ送信処理

関数名：void tx_demo_fifo(void)

以下に、送受信 FIFO バッファ（送信モード）を使ったメッセージ送信処理のフローチャートを示します。

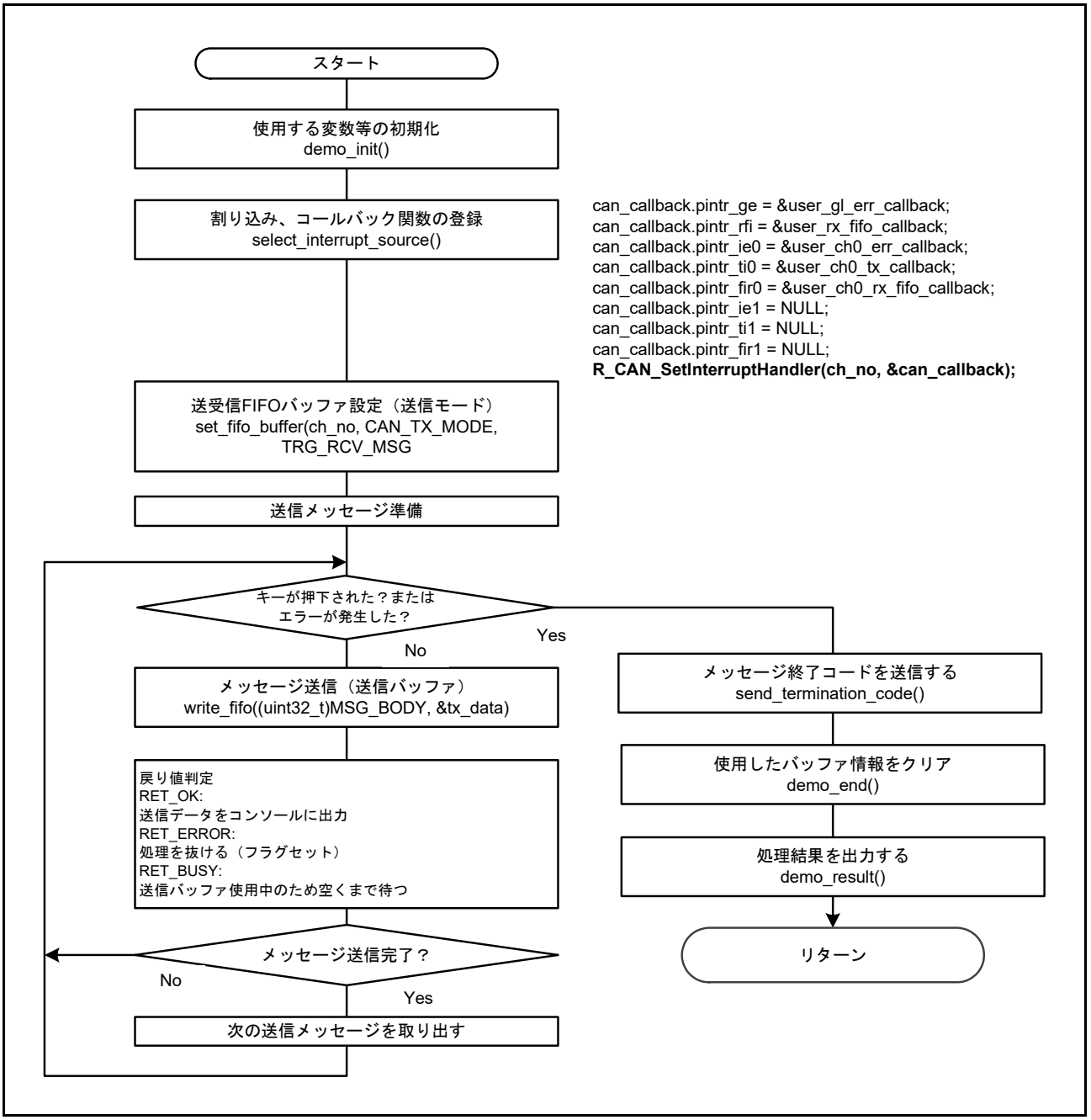


図 10.17　サンプルコードの送受信 FIFO バッファ (送信モード) を使ったメッセージ送信処理

(3) 送信バッファ関連レジスタに送信メッセージを書き込む処理
関数名：uint32_t write_buffer(uint32_t msg_type, tx_data_t * obj)
以下に、送信バッファ関連レジスタに送信メッセージを書き込む処理のフローチャートを示します。

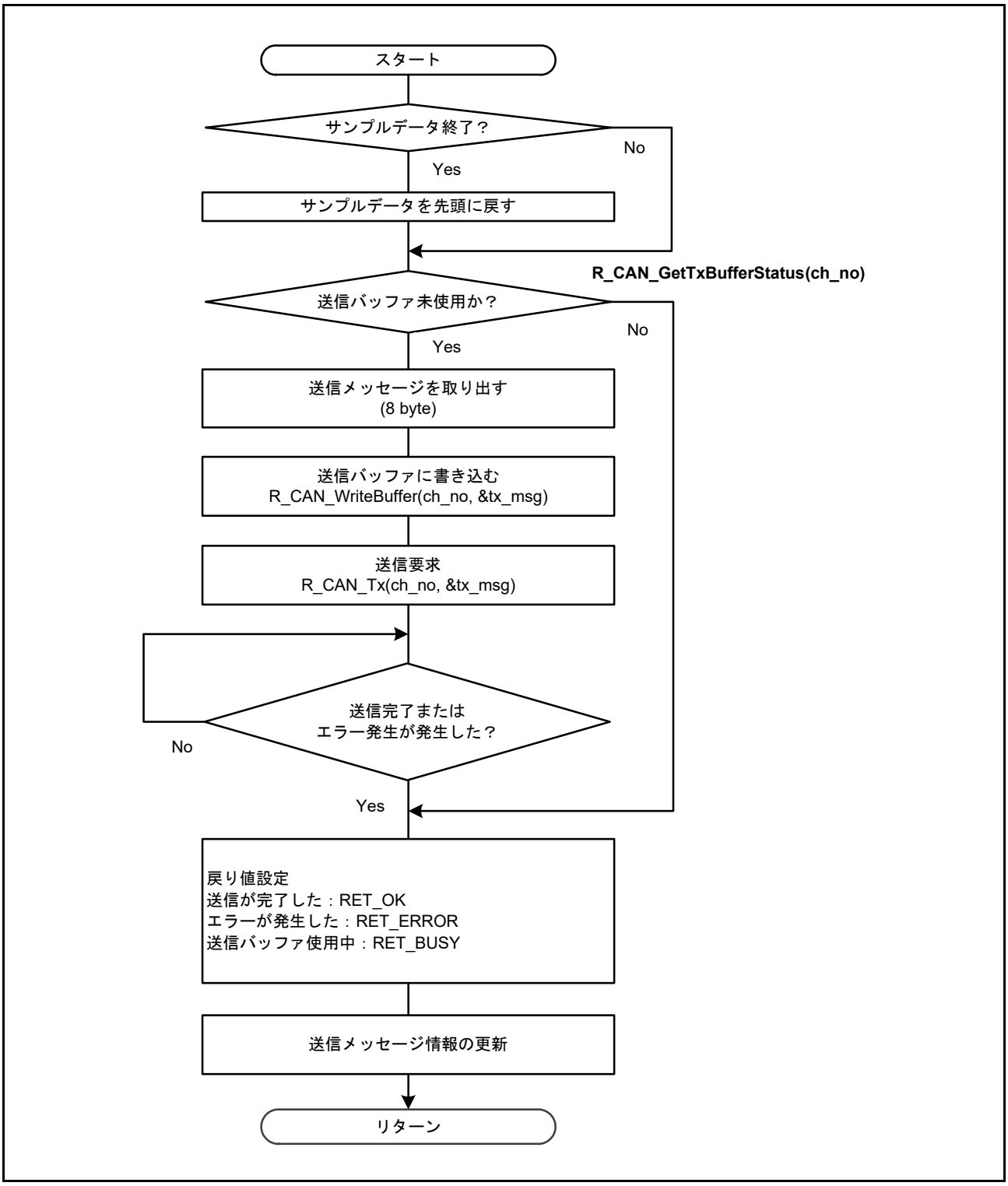


図 10.18　サンプルコードの送信バッファ関連レジスタに送信メッセージを書き込む処理

(4) 送受信 FIFO バッファ関連レジスタに送信メッセージを書き込む処理

関数名：uint32_t write_fifo(uint32_t msg_type, tx_data_t * obj)

以下に、送受信 FIFO バッファ関連レジスタに送信メッセージを書き込む処理のフローチャートを示します。

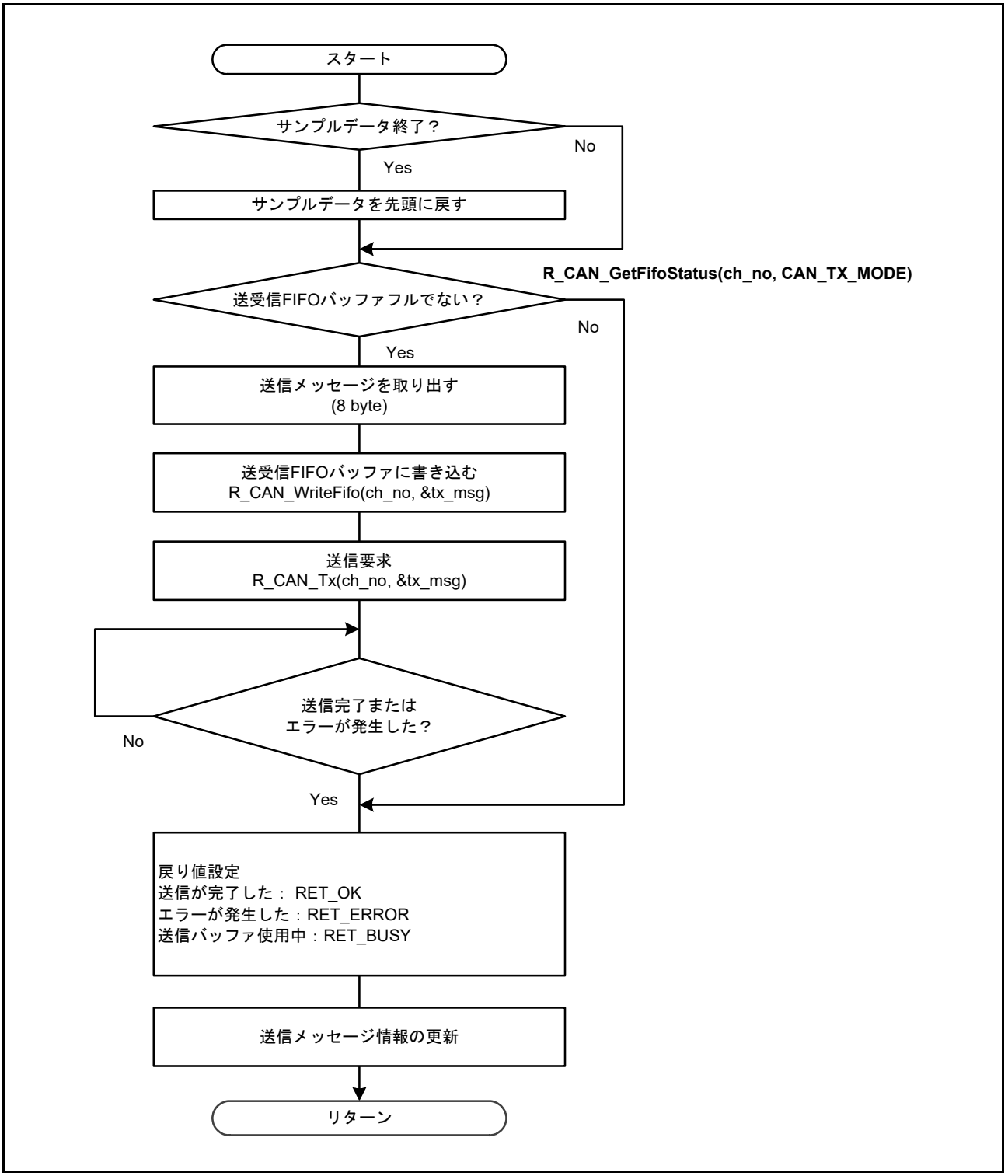


図 10.19　サンプルコードの送受信 FIFO バッファ関連レジスタに送信メッセージを書き込む処理

10.8.3　受信テスト

受信テストは、「受信バッファを使ったメッセージ受信」、「受信 FIFO バッファを使ったメッセージ受信」および「送受信 FIFO バッファ（受信モード）を使ったメッセージ受信」をメニューから選択することができます。

メニューの内容は、10.1　**動作概要**を参照してください。

（1）void rx_demo_buffer(void)
　受信バッファを使ったメッセージ受信処理
（2）void rx_demo_rx_fifo(void)
　受信 FIFO バッファを使ったメッセージ受信処理
（3）void rx_demo_fifo(void)
　送受信 FIFO バッファ（受信モード）を使ったメッセージ受信処理
（4）uint32_t read_buffer(rx_data_t * obj)
　受信バッファから受信メッセージを読み出す処理
（5）void read_rx_fifo(rx_data_t * obj)
　受信 FIFO バッファから受信メッセージを読み出す処理
（6）void read_fifo(uint32_t ch, rx_data_t * obj)
　送受信 FIFO バッファから受信メッセージを読み出す処理

図 10.20 ～**図 10.25** にそれぞれの機能のフローチャートを示します。

(1) 受信バッファを使ったメッセージ受信処理

関数名：void rx_demo_buffer(void)

以下に、受信バッファを使ったメッセージ受信処理のフローチャートを示します。

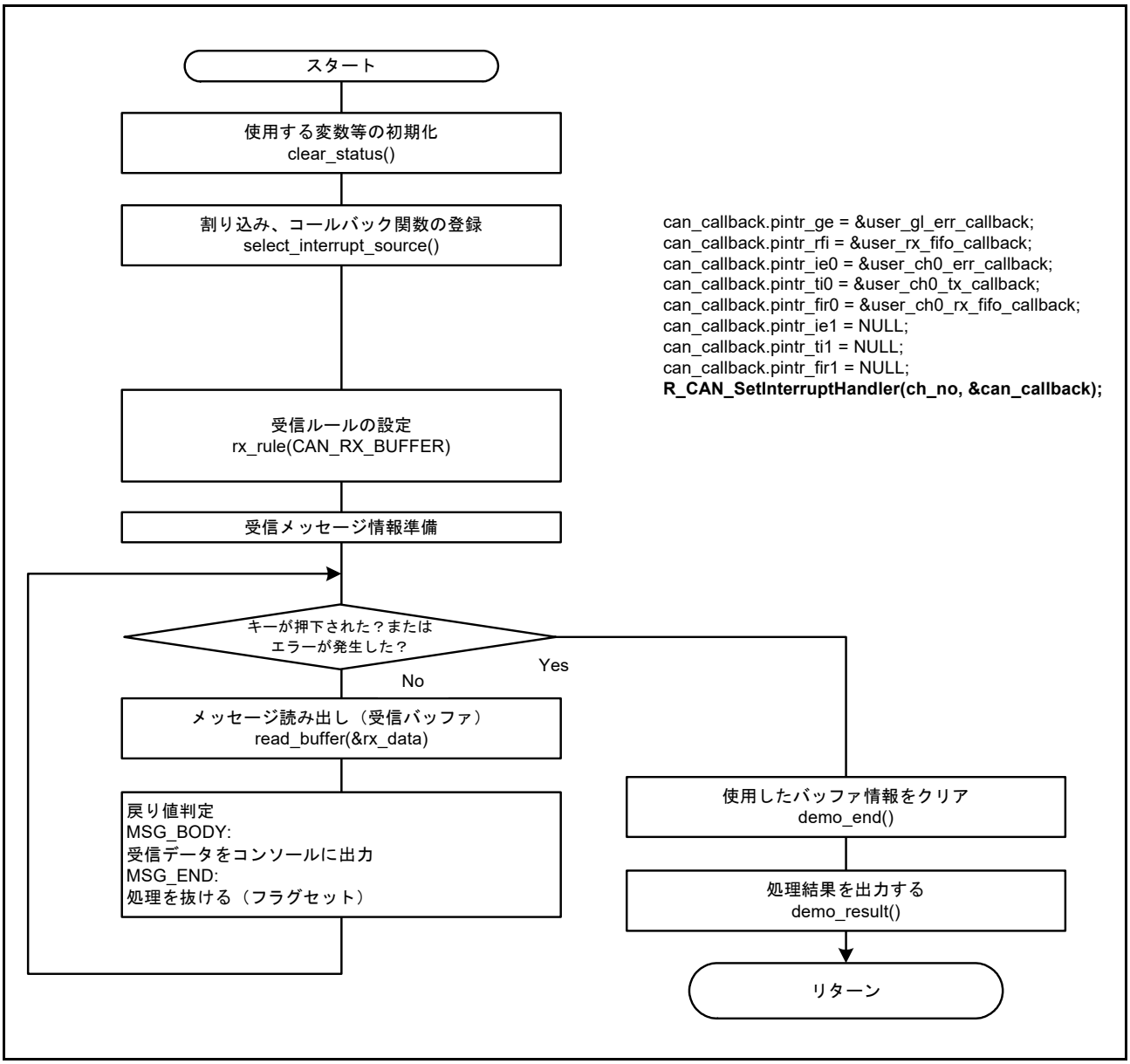


図 10.20　サンプルコードの受信バッファを使ったメッセージ受信処理

(2) 受信 FIFO バッファを使ったメッセージ受信処理

関数名：void rx_demo_rx_fifo(void)

以下に、受信 FIFO バッファを使ったメッセージ受信処理のフローチャートを示します。

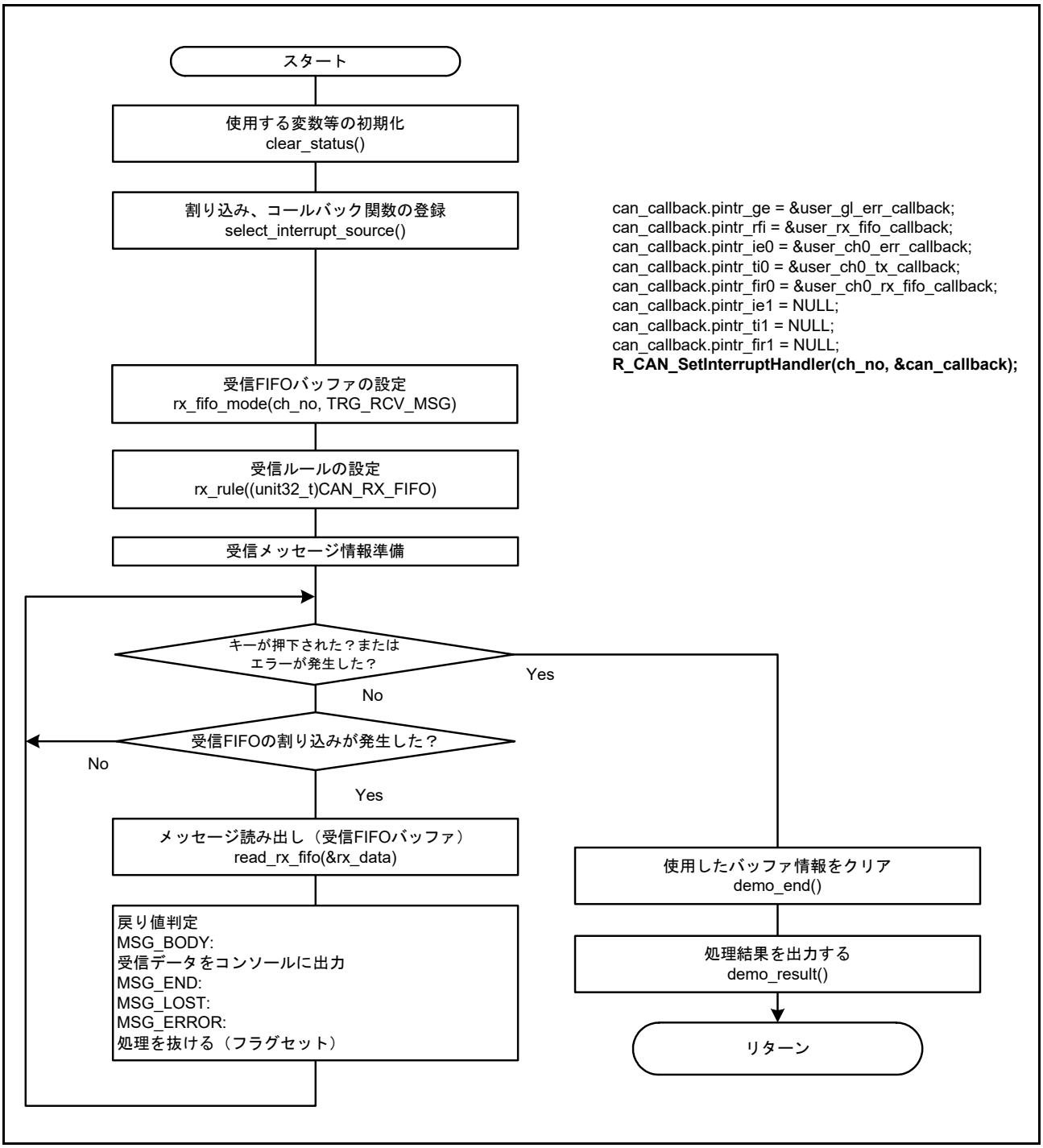


図 10.21　サンプルコードの受信 FIFO バッファを使ったメッセージ受信処理

（3）送受信 FIFO バッファ（受信モード）を使ったメッセージ受信処理

関数名：void rx_demo_fifo(void)

以下に、送受信 FIFO バッファ（受信モード）を使ったメッセージ受信処理のフローチャートを示します。

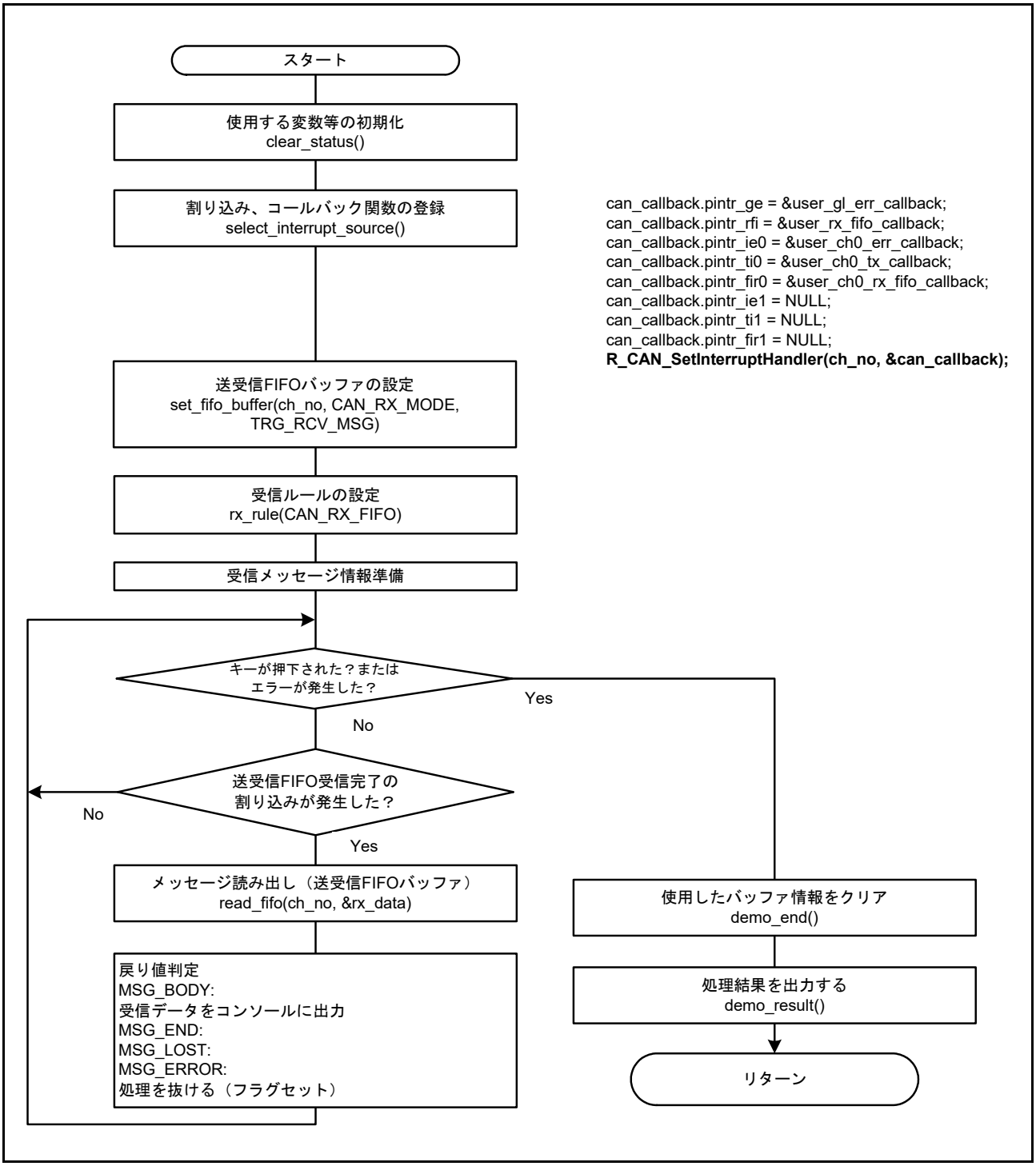


図 10.22　サンプルコードの送受信 FIFO バッファ（受信モード）を使ったメッセージ受信処理

(4) 受信バッファから受信メッセージを読み出す処理
関数名：uint32_t read_buffer(rx_data_t * obj)
以下に、受信バッファから受信メッセージを読み出す処理のフローチャートを示します。

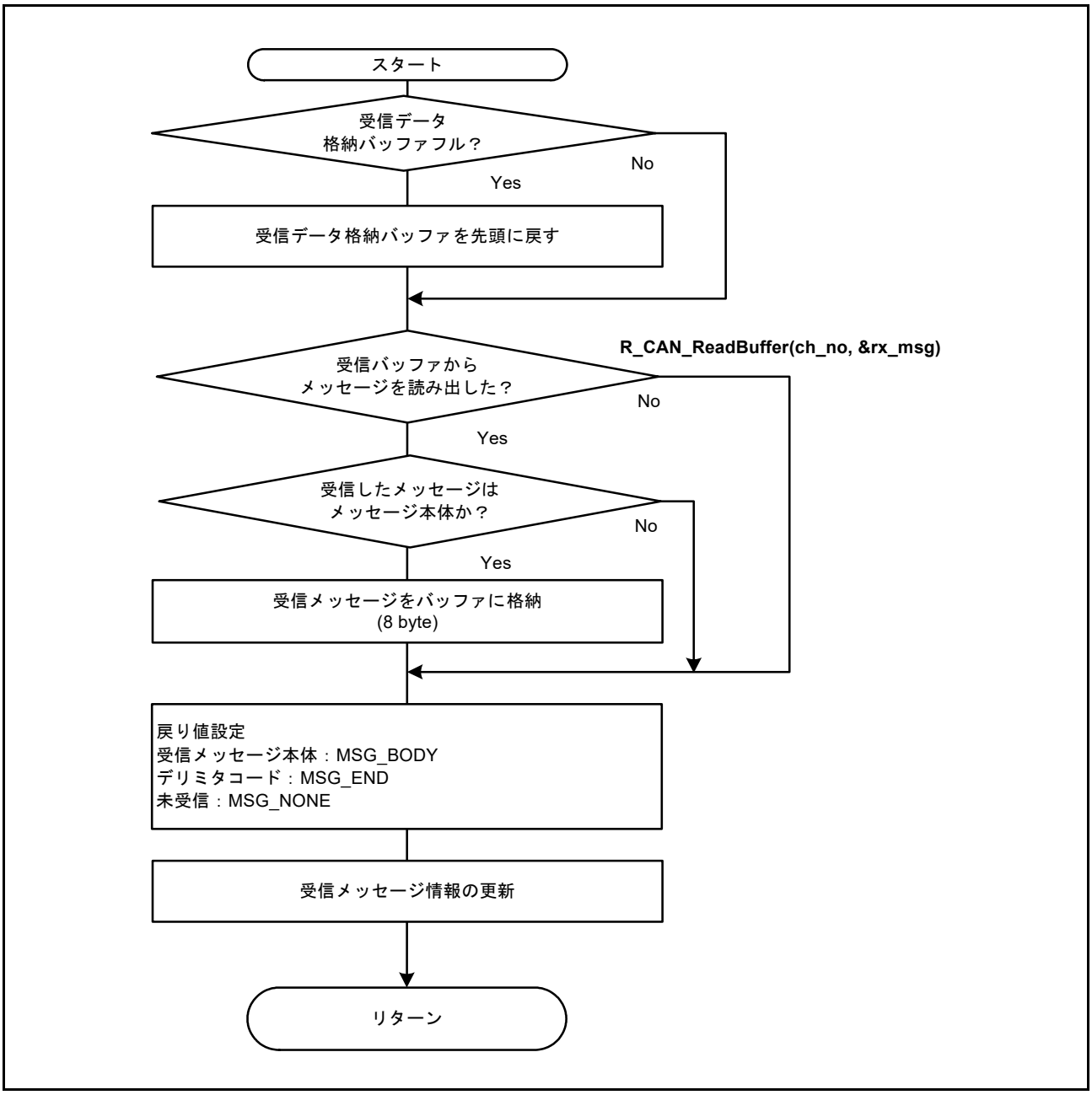


図 10.23　サンプルコードの受信バッファから受信メッセージを読み出す処理

(5) 受信 FIFO バッファから受信メッセージを読み出す処理

関数名：void read_rx_fifo(rx_data_t * obj)

以下に、受信 FIFO バッファから受信メッセージを読み出す処理のフローチャートを示します。

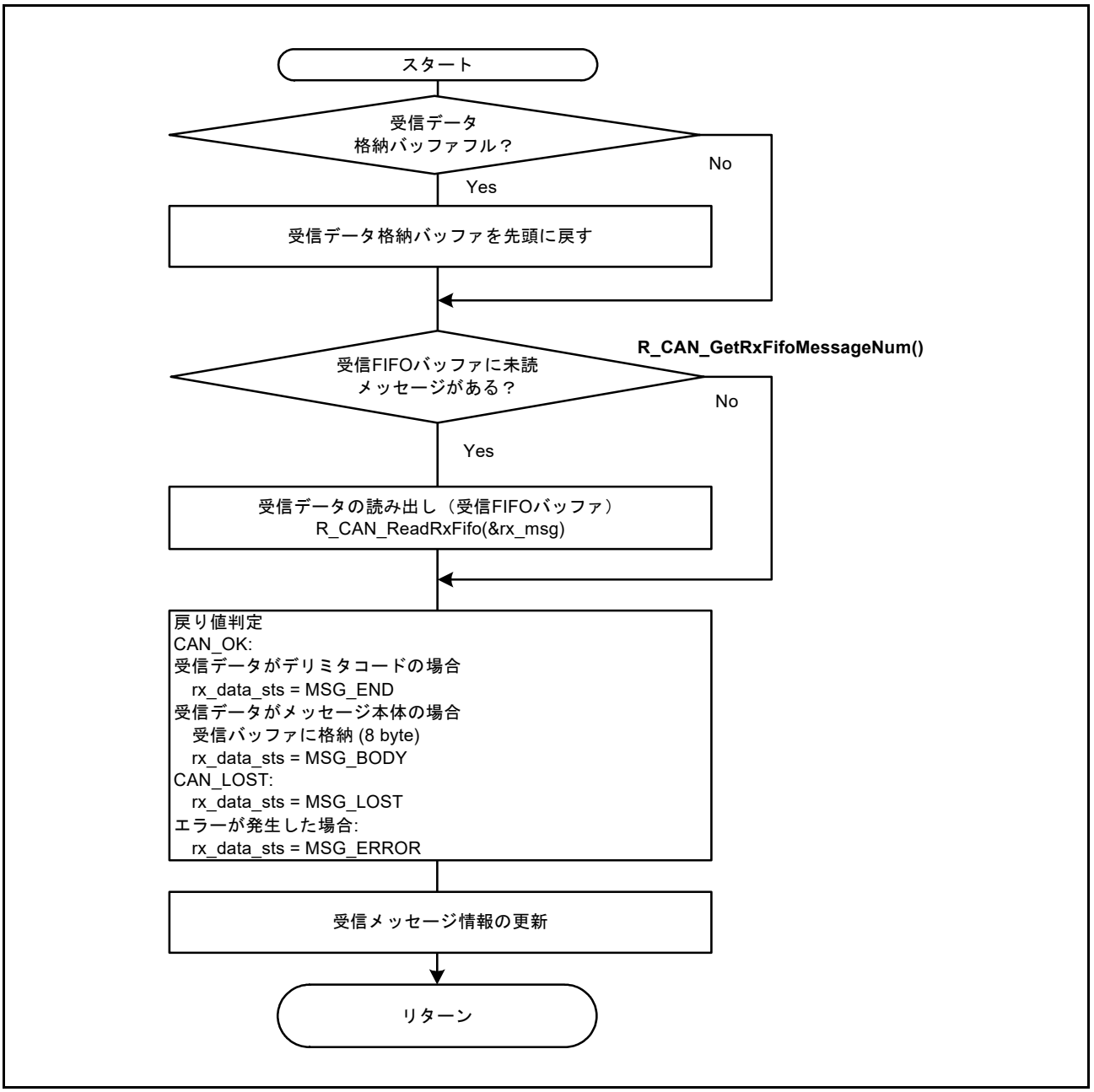


図 10.24　サンプルコードの受信 FIFO バッファから受信メッセージを読み出す処理

(6) 送受信 FIFO バッファから受信メッセージを読み出す処理

関数名：void read_fifo(uint32_t ch, rx_data_t * obj)

以下に、送受信 FIFO バッファから受信メッセージを読み出す処理のフローチャートを示します。

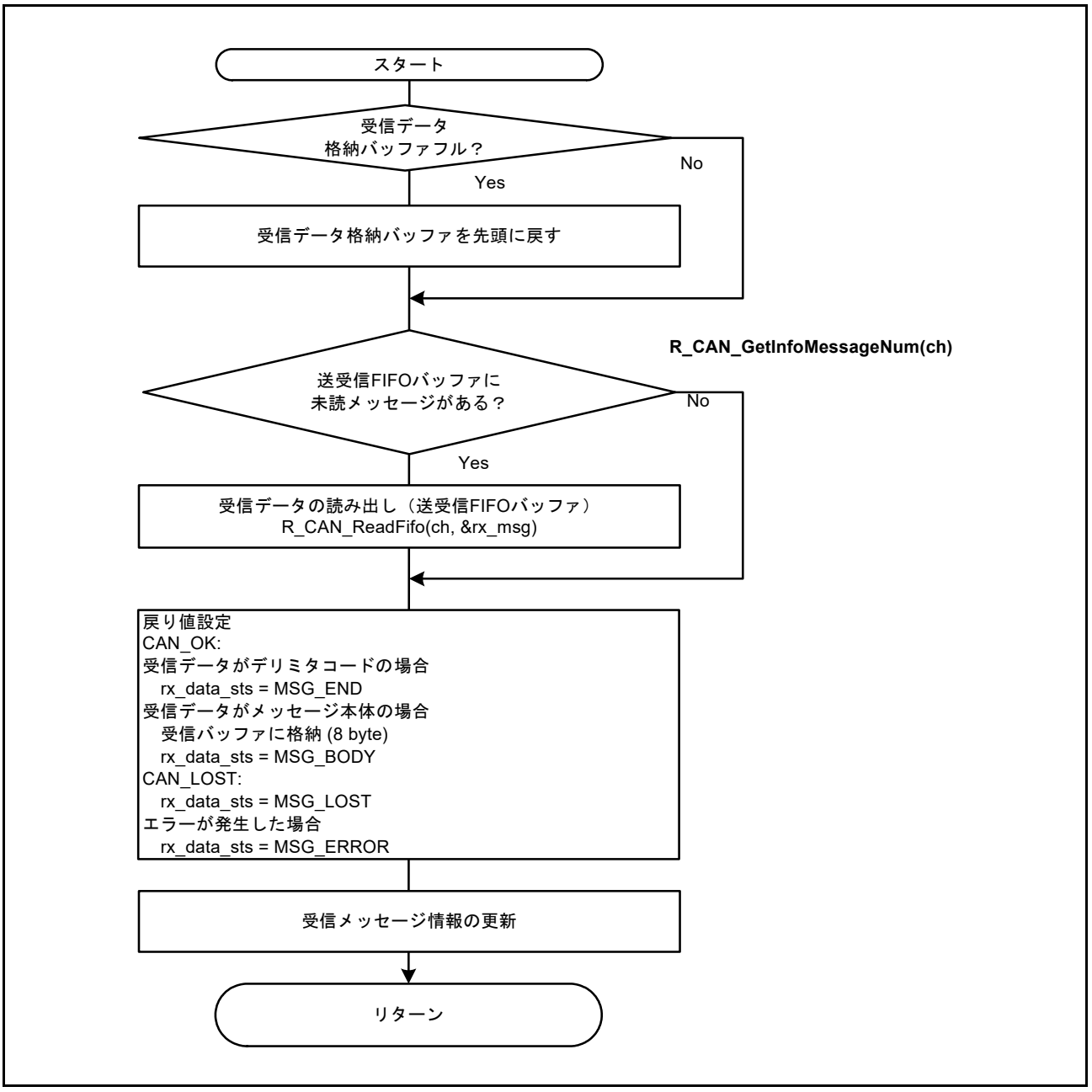


図 10.25　サンプルコードの送受信 FIFO バッファから受信メッセージを読み出す処理

10.8.4　受信を受け付けながらメッセージ送信を行うテスト

メッセージを受信しながらメッセージを送信するテストです。

本サンプルでは、送受信 FIFO バッファ（受信モード）を使ってメッセージを受信し、送受信 FIFO バッファ（送信モード）を使ってメッセージを送信します。

メニューの内容は、10.1　**動作概要**を参照してください。

本テストを行うには以下の関数を使用します。

（1）void trx_demo_fifo(void)

メッセージを受信しながらメッセージを送信するテスト

図 10.26 に本機能のフローチャートを示します。

(1) メッセージを受信しながらメッセージを送信するテスト

関数名：void trx_demo_fifo(void)

以下に、メッセージを受信しながらメッセージを送信するテストのフローチャートを示します。

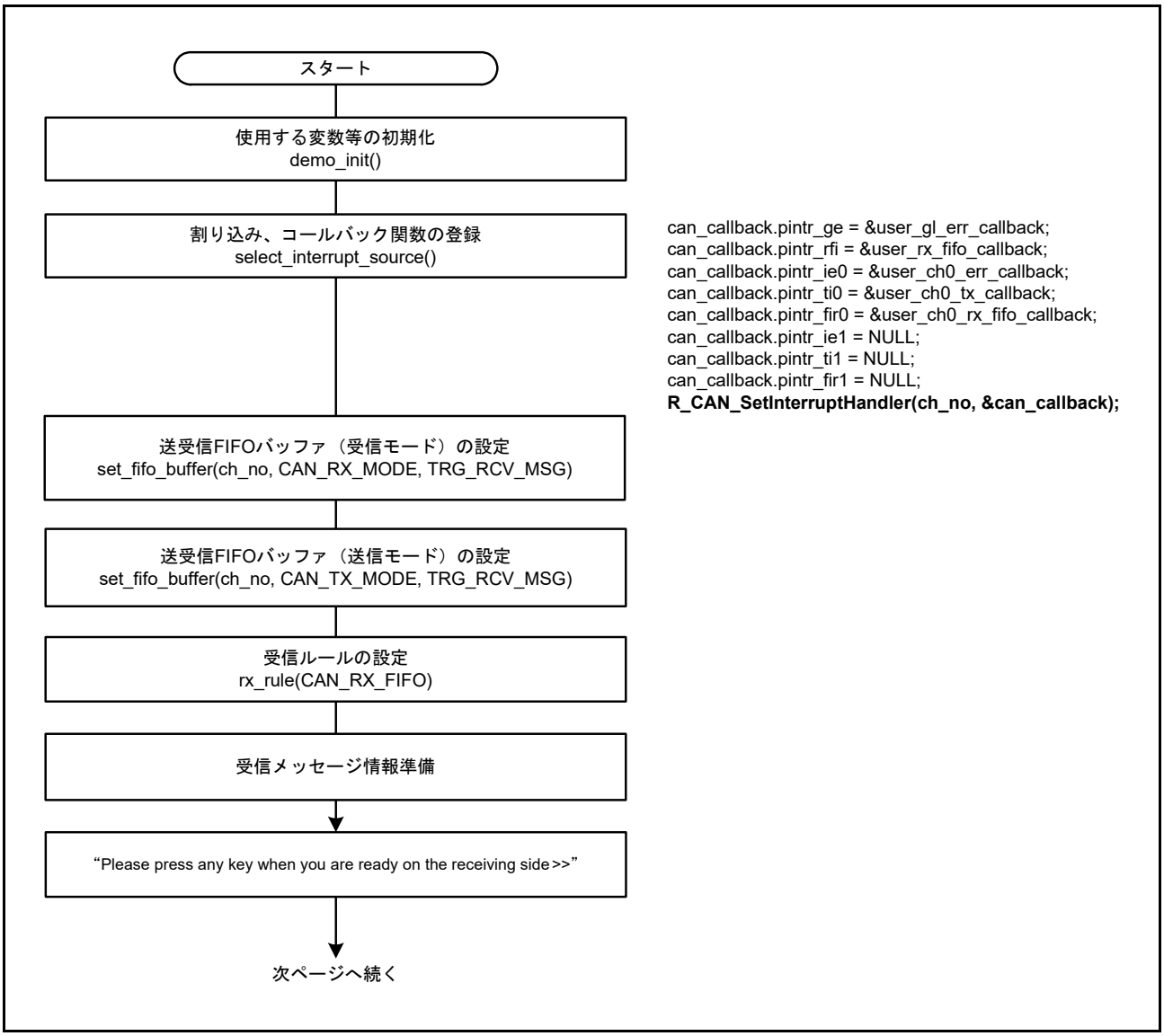


図 10.26　サンプルコードのメッセージを受信しながらメッセージを送信するテスト (1/2)

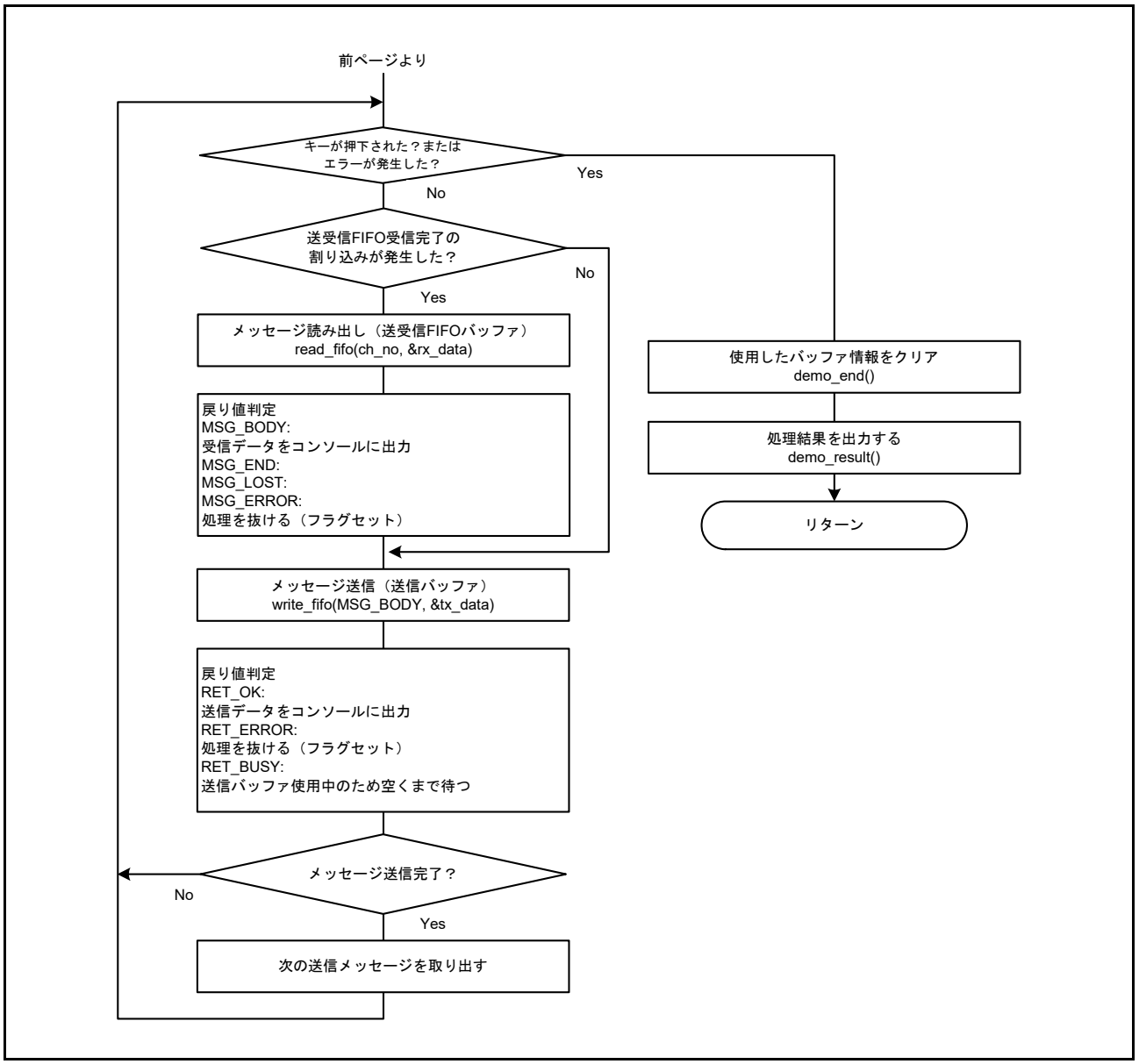


図 10.26　サンプルコードのメッセージを受信しながらメッセージを送信するテスト (2/2)

10.8.5　セルフテスト

開発環境に接続している評価ボードのみを使用して CAN 通信テストを行うことができます。
本サンプルプログラムでは、メッセージの送信・受信を行うセルフテストモードを用意しています。
また、セルフテストモード 0（外部ループバックモード）とセルフテストモード 1（内部ループバックモード）の切り替えが可能です。メニューから選択してください。
メニューの内容は、10.1　**動作概要**を参照してください。

(1) セルフテストモード 0（外部ループバックモード）
CAN トランシーバを含めたチャネルのループバックテストを行うモードです。
図 10.27 にセルフテストモード 0 選択時の接続を示します。

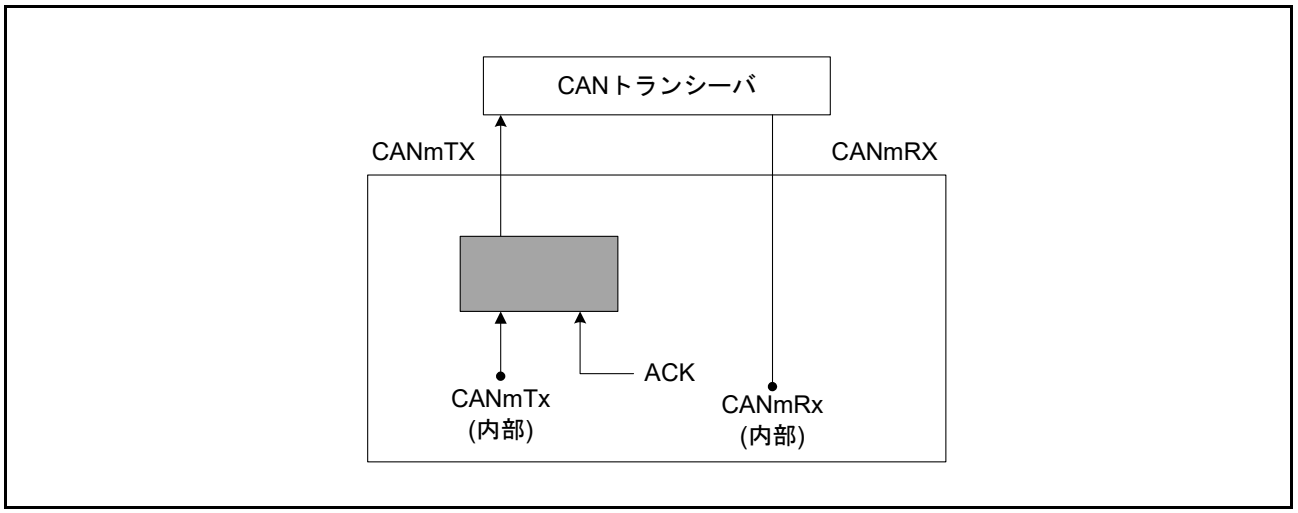


図 10.27　セルフテストモード 0 選択時の接続

(2) セルフテストモード 1（内部ループバックモード）
送信したメッセージを受信したメッセージとして取り扱い、送信したメッセージをバッファに格納するモードです。
図 10.28 にセルフテストモード 1 選択時の接続を示します。

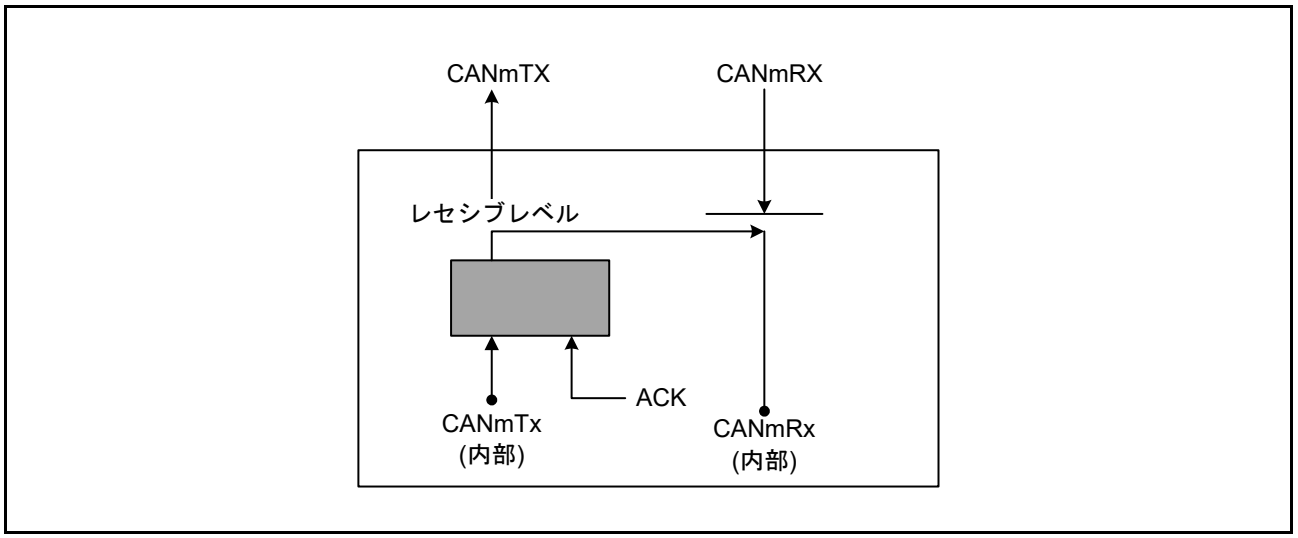


図 10.28　セルフテストモード 1 選択時の接続

下記の 4 種類のセルフテストを用意しています。

- 送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテスト
- 送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト
- 送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト
- 送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト

送信側の送信方法は

- 送信バッファを使用時
  1 メッセージ送信を繰り返します。
- 送受信 FIFO バッファ（送信モード）を使用時
  1 メッセージ送信を繰り返します。

受信側の受信方法は

- 受信 FIFO バッファで受け取る場合
  受信 FIFO バッファフルで割り込み発生（受信 FIFO バッファ段数：4 メッセージ）
- 送受信 FIFO バッファ（受信モード）で受け取る場合
  送受信 FIFO バッファフルで割り込み発生（送受信 FIFO バッファ段数：4 メッセージ）

各テストを行うには以下の関数を使用します。

(1) void selftest_buf_to_buf(void)
送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテスト

(2) void selftest_buf_to_rx_fifo(void)
送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト

(3) void selftest_buf_to_fifo(void)
送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト

(4) void selftest_fifo_to_fifo(void)
送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト

図 10.29 ～図 10.32 にそれぞれの機能のフローチャートを示します。

(1) 送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテスト
関数名：void selftest_buf_to_buf(void)
以下に、送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテストのフローチャートを示します。

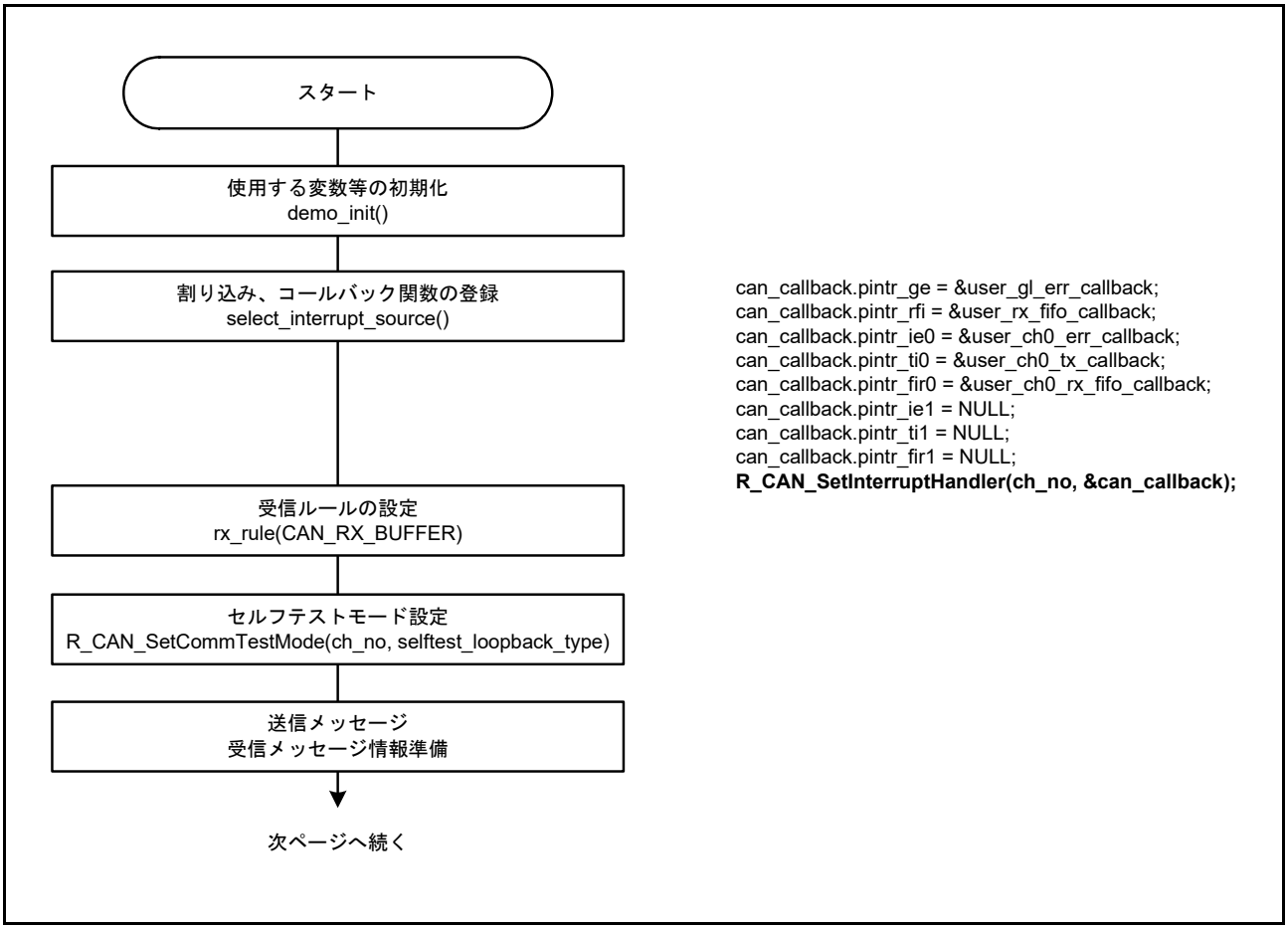


図 10.29　サンプルコードの送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテスト（続く）

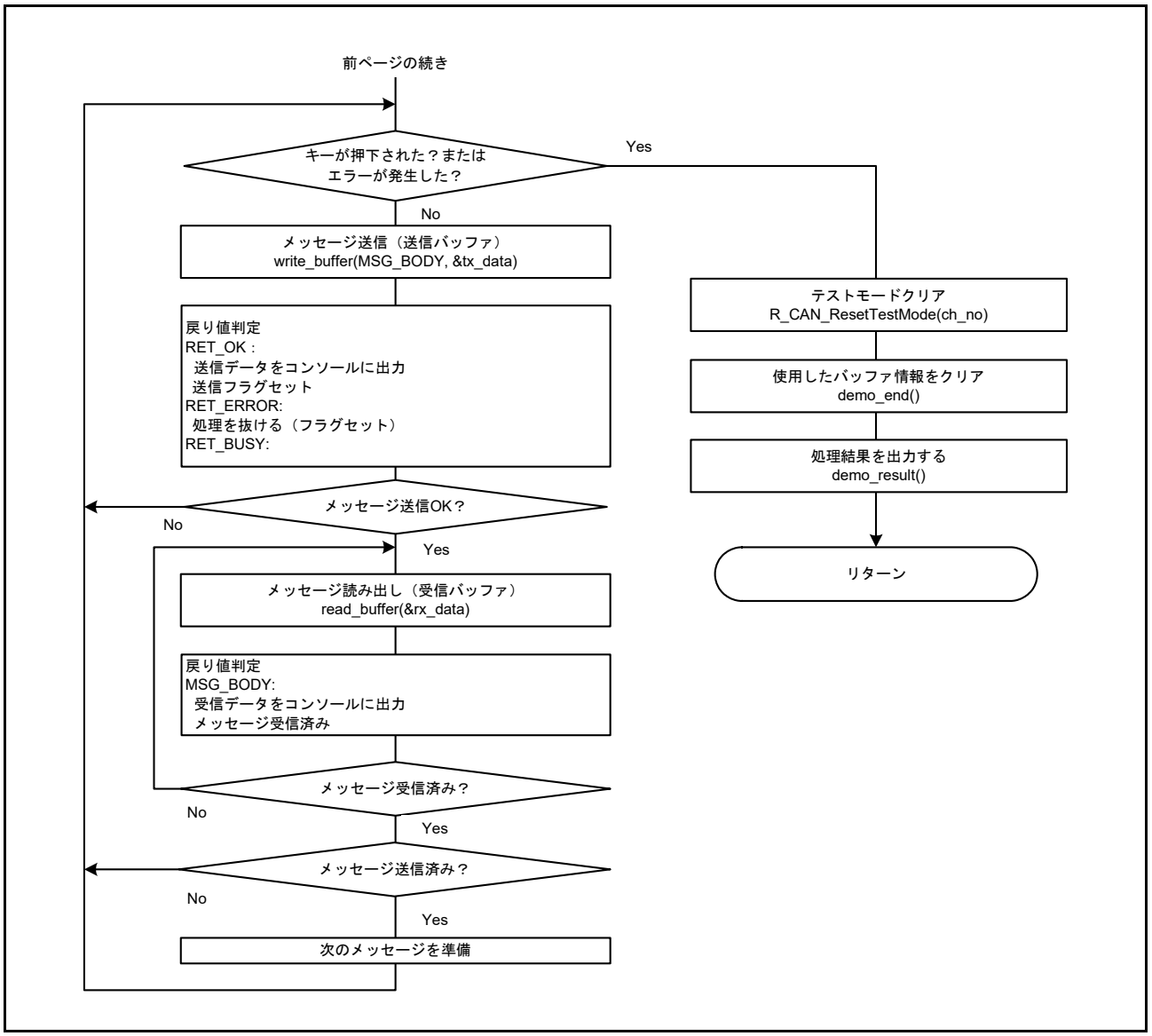


図 10.29　サンプルコードの送信バッファを使ってメッセージを送信し、受信バッファでメッセージを受信するテスト

(2) 送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト
関数名：void selftest_buf_to_rx_fifo(void)
以下に、送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテストのフローチャートを示します。

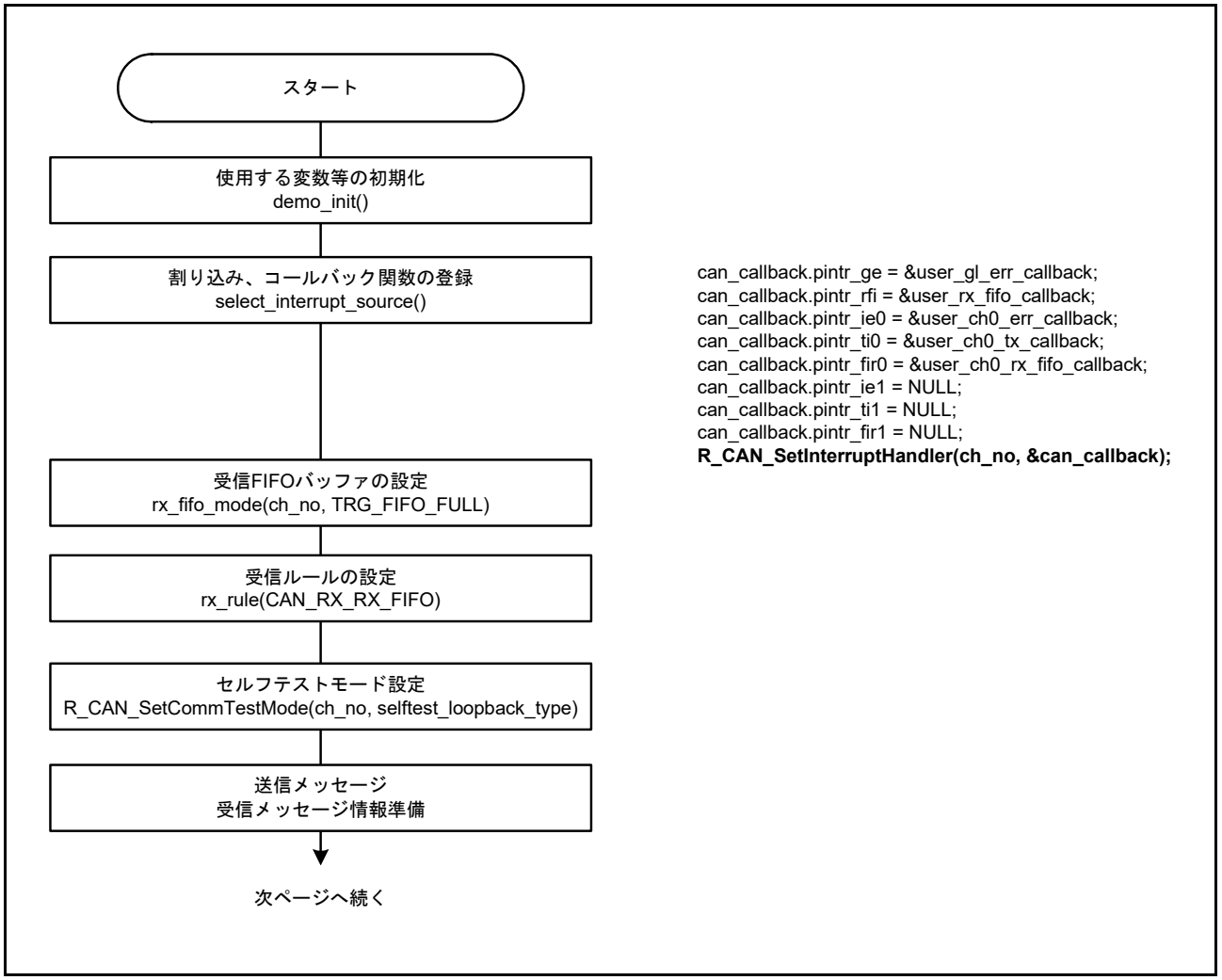


図 10.30　サンプルコードの送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト（続く）

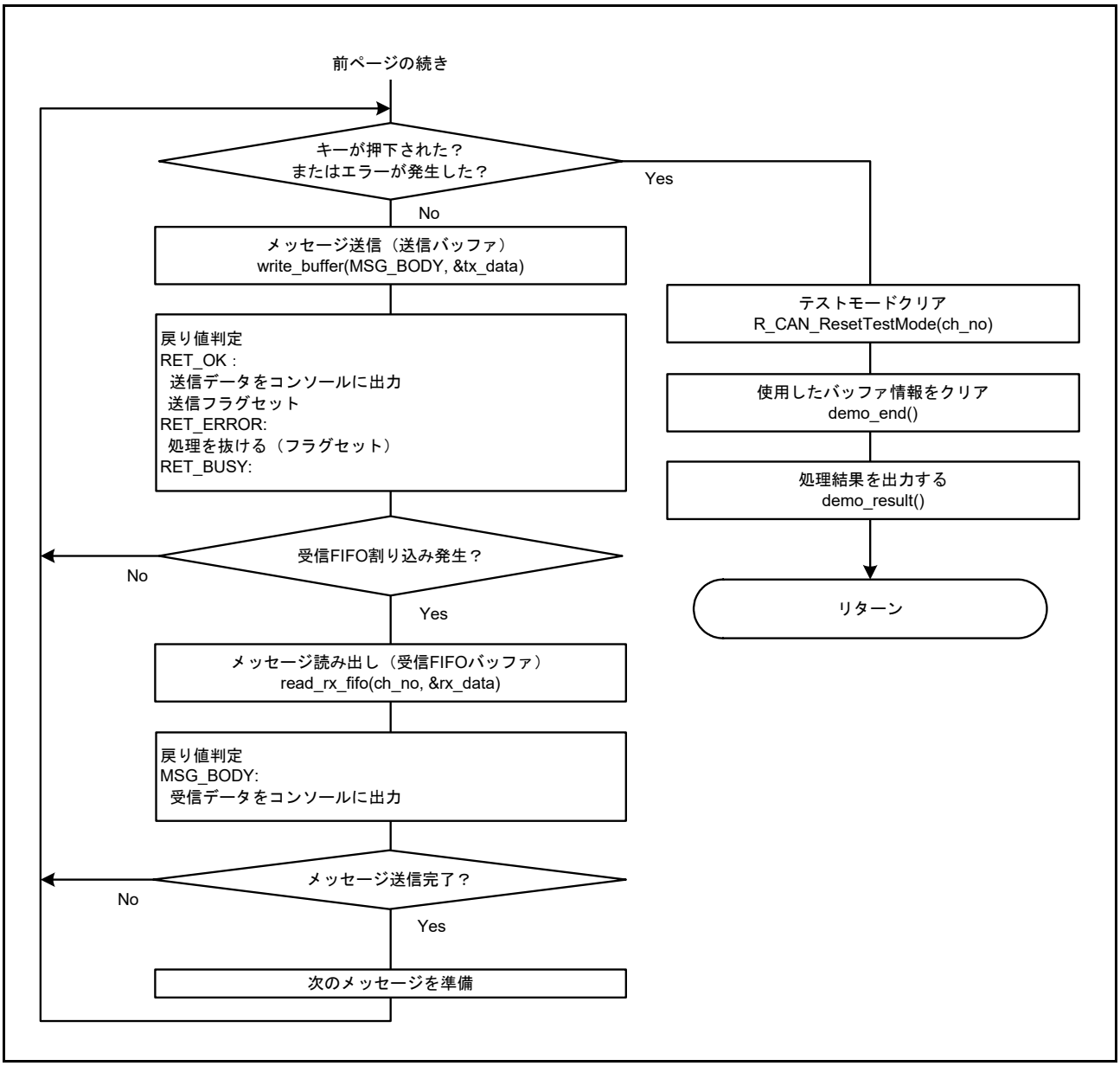


図 10.30　サンプルコードの送信バッファを使ってメッセージを送信し、受信 FIFO バッファでメッセージを受信するテスト

（3）送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト
関数名：void selftest_buf_to_fifo(void)
以下に、送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテストのフローチャートを示します。

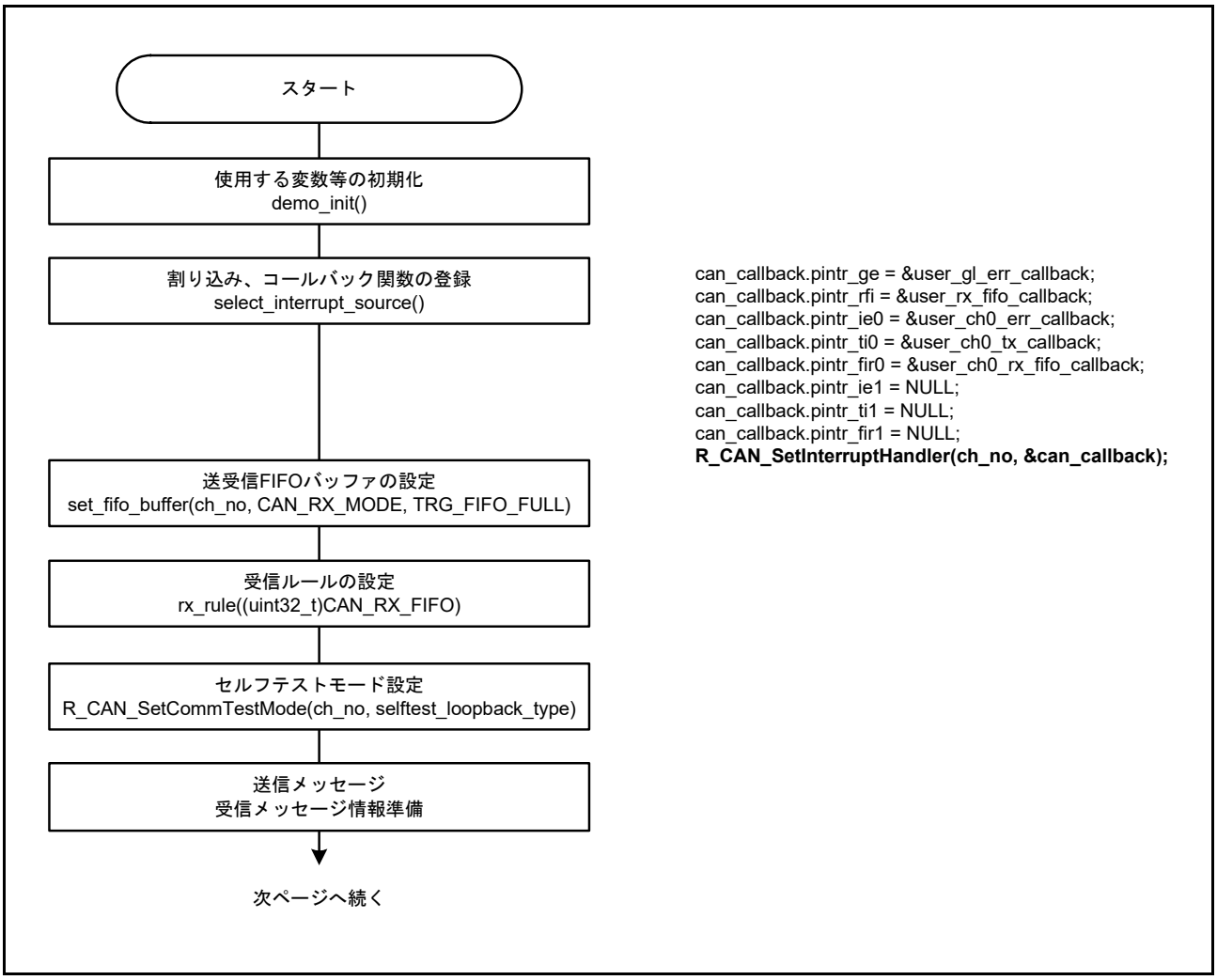


図 10.31　サンプルコードの送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト（続く）

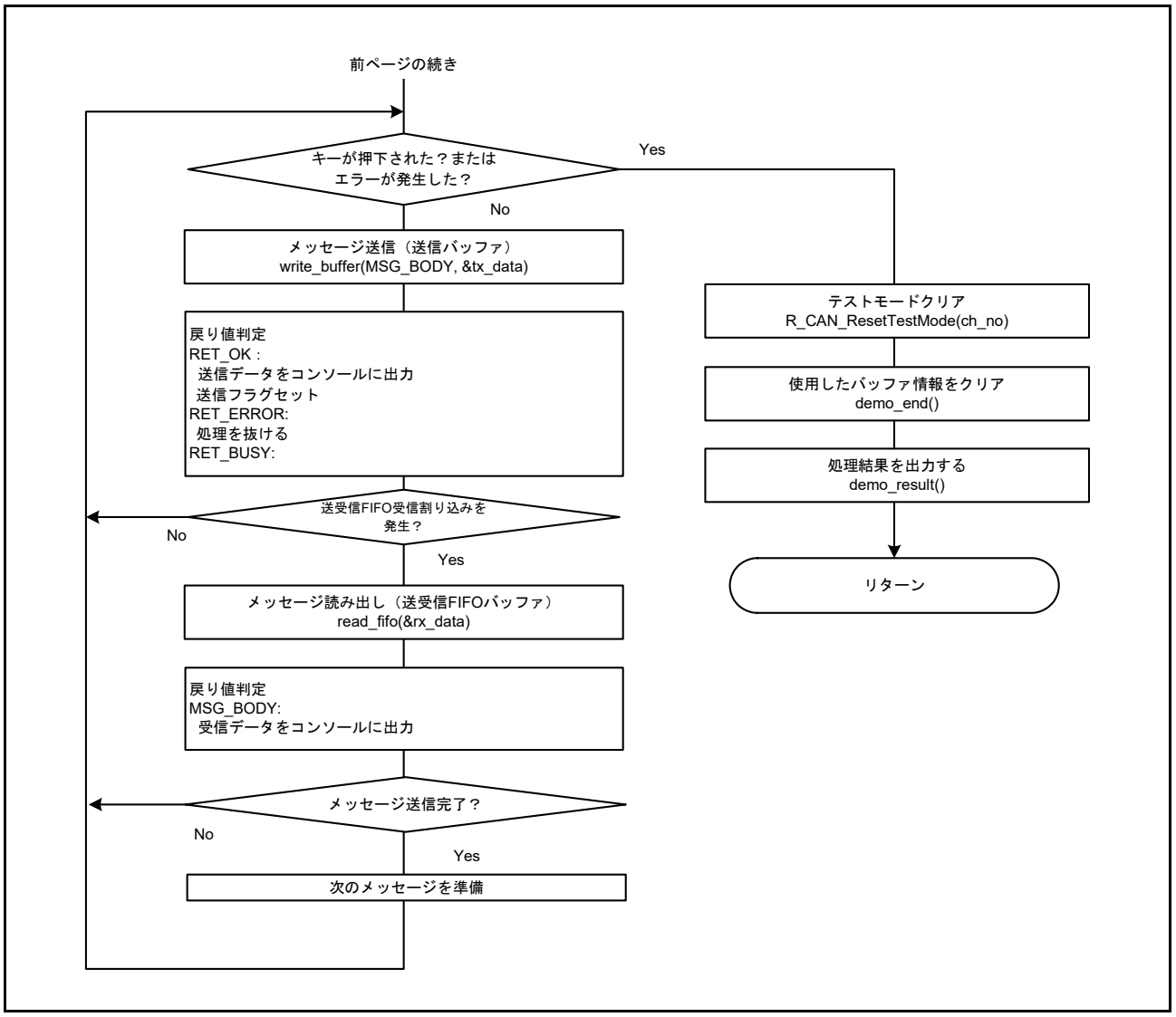


図 10.31　サンプルコードの送信バッファを使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト

（4）送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト
関数名：void selftest_fifo_to_fifo(void)
以下に、送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテストのフローチャートを示します。

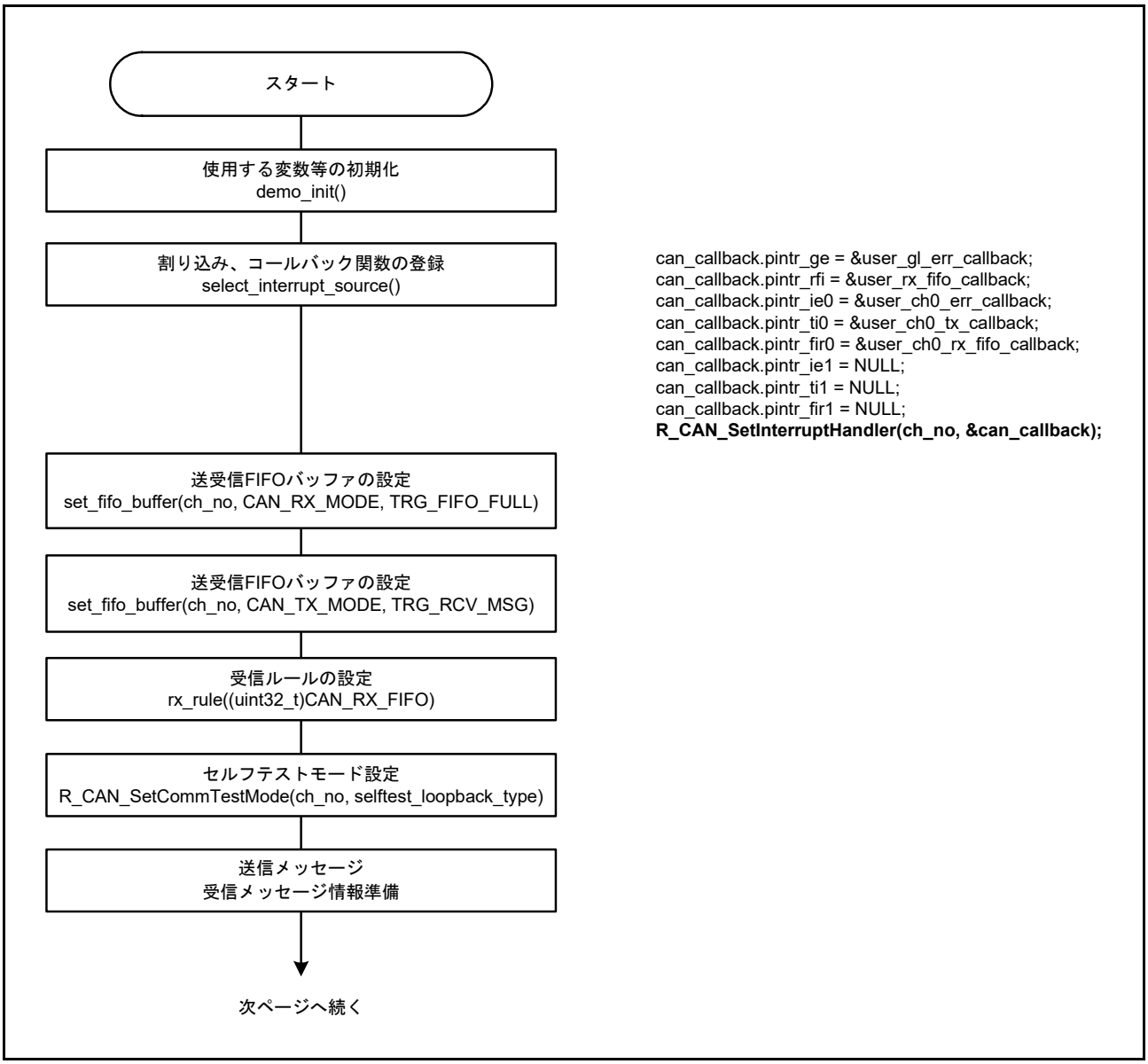


図 10.32　サンプルコードの送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト（続く）

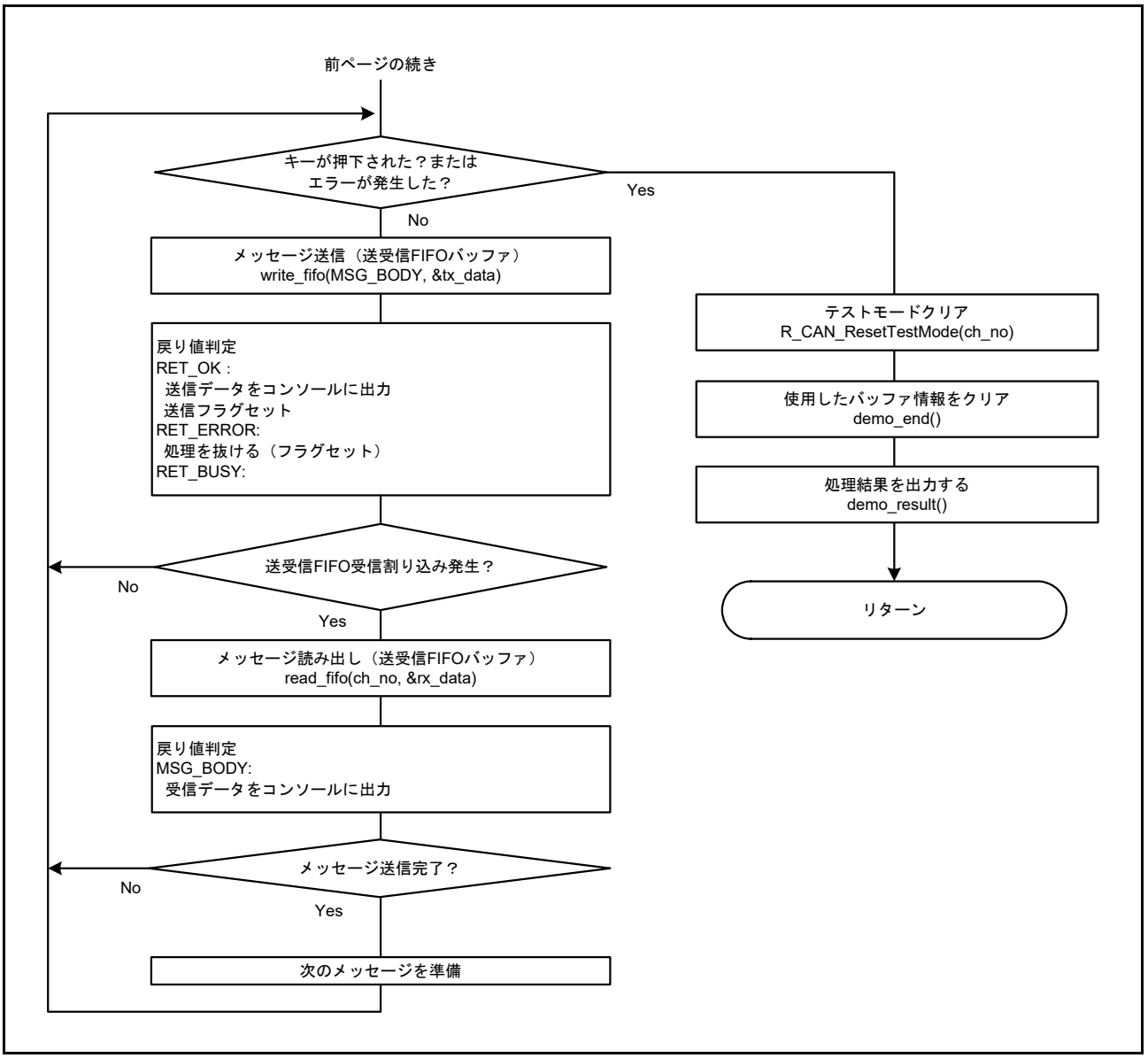


図 10.32　サンプルコードの送受信 FIFO バッファ（送信モード）を使ってメッセージを送信し、送受信 FIFO バッファ（受信モード）でメッセージを受信するテスト

10.8.6　コールバック処理

本サンプルプログラムは、割り込み発生時に呼ばれる割り込みハンドラと、割り込みハンドラから呼ばれるコールバック関数を実装しています。

（1）RSCAN:CANGE　CAN グローバル・エラー
　割り込みハンドラ：void user_gl_err_isr(void)
　コールバック関数：void user_gl_err_callback(void)
（2）RSCAN:CANIE0　CAN0 エラー
　割り込みハンドラ：void user_ch0_err_isr(void)
　コールバック関数：void user_ch0_err_callback(void)
（3）RSCAN:CANIE1　CAN1 エラー
　割り込みハンドラ：void user_ch1_err_isr(void)
　コールバック関数：void user_ch1_err_callback(void)
（4）RSCAN:CANRFI　CAN 受信 FIFO 割り込み
　割り込みハンドラ：void user_rx_fifo_isr(void)
　コールバック関数：void user_rx_fifo_callback(void)
（5）RSCAN:CANTI0　CAN0 送信割り込み
　割り込みハンドラ：void user_ch0_tx_isr(void)
　コールバック関数：void user_ch0_tx_callback(void)
（6）RSCAN:CANTI1　CAN1 送信割り込み
　割り込みハンドラ：void user_ch1_tx_isr(void)
　コールバック関数：void user_ch1_tx_callback(void)
（7）RSCAN:CANFIR0　CAN0 送受信 FIFO 受信完了割り込み
　割り込みハンドラ：void user_ch0_rx_fifo_isr(void)
　コールバック関数：void user_ch0_rx_fifo_callback(void)
（8）RSCAN:CANFIR1　CAN1 送受信 FIFO 受信完了割り込み
　割り込みハンドラ：void user_ch1_rx_fifo_isr(void)
　コールバック関数：void user_ch1_rx_fifo_callback(void)
（9）SCIFA:RXIF2　SCIFA 受信 FIFO データフル割り込み
　割り込みハンドラ：void scifa_key_input_isr(void)
　コールバック関数：void key_handler_callback(void)

- 割り込みハンドラの登録方法
  割り込みハンドラは、以下の API 関数を用いて行ってください。
  void R_ICU_Regist(uint32_t vec_num, uint32_t type, uint32_t priority, uint32_t isr_addr);
  uint32_t vec_num：ベクタ番号
  uint32_t type：割り込み検出タイプ
  uint32_t priority：割り込み優先レベル
  uint32_t isr_addr：割り込みハンドラ関数のアドレス

  また、割り込みの許可／禁止は、以下の API 関数を用いてください。
  void R_ICU_Disable(uint32_t vec_num);
  void R_ICU_Enable(uint32_t vec_num);
  uint32_t vec_num：ベクタ番号

- コールバック関数の登録方法
  can_handle_t gb_can_handles[CAN_NUM];

```
typedef struct {
    void (*pintr_ge)(void);          /* Pointer to user callback function. */
    void (*pintr_ie0)(void);         /* Pointer to user callback function. */
    void (*pintr_ie1)(void);         /* Pointer to user callback function. */
    void (*pintr_rfi)(void);         /* Pointer to user callback function. */
    void (*pintr_fir0)(void);        /* Pointer to user callback function. */
    void (*pintr_ti0)(void);         /* Pointer to user callback function. */
    void (*pintr_fir1)(void);        /* Pointer to user callback function. */
    void (*pintr_ti1)(void);         /* Pointer to user callback function. */
} can_callback_t;

typedef struct {
    bool             ch_opened;
    can_callback_t   can_callback;
} can_handle_t;
```

図 10.33 ～図 10.38 にそれぞれの機能のフローチャートを示します。

(1)（RSCAN:CANGE）CAN グローバル・エラー発生時に呼ばれるコールバック関数

割り込みハンドラ：void user_gl_err_isr(void)

コールバック関数：void user_gl_err_callback(void)

CAN モジュール全体で、エラーが発生した場合に割り込みハンドラ処理から呼び出されます。

以下に、グローバル・エラー発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

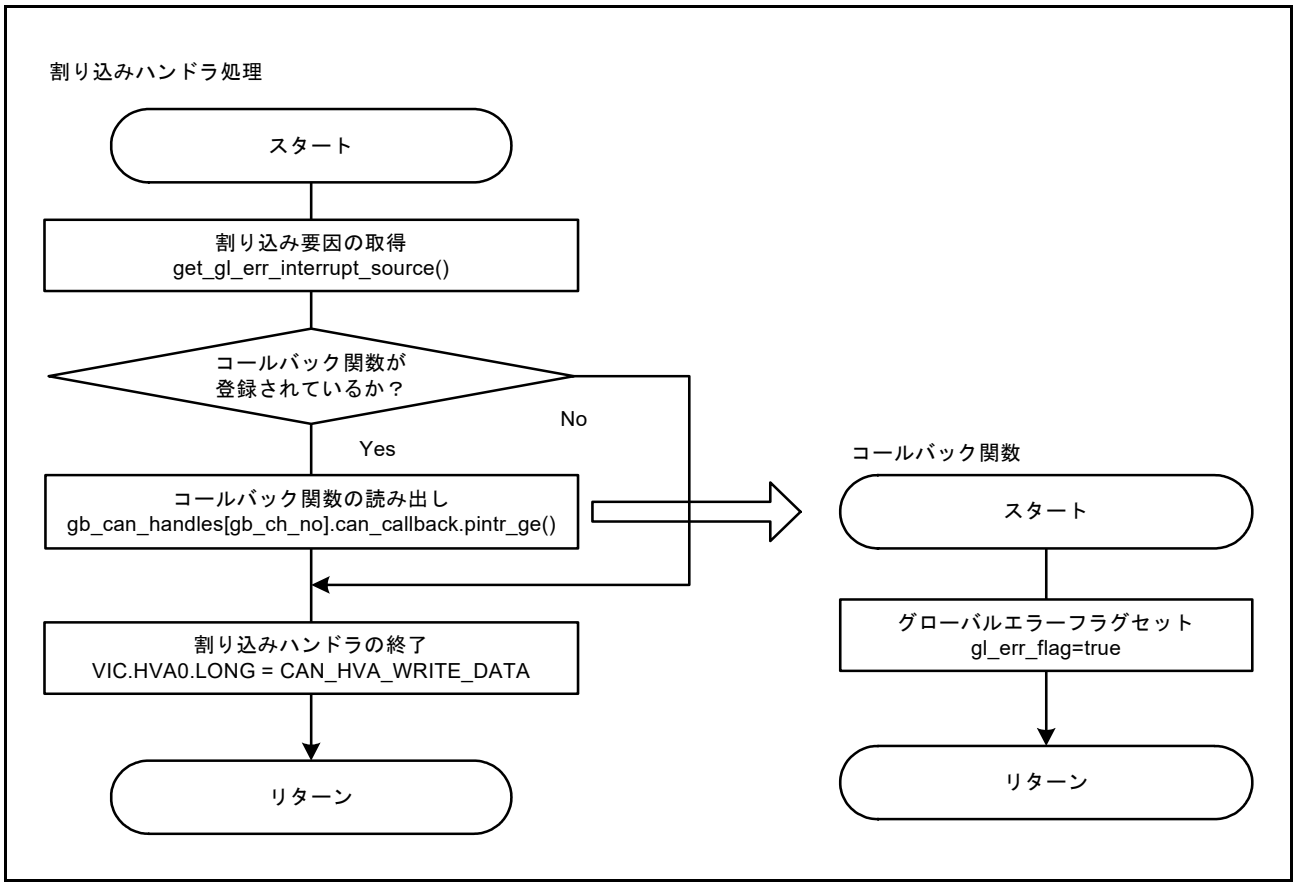


図 10.33　サンプルコードの CAN グローバル・エラー発生時に呼ばれるコールバック関数処理

（2）（RSCAN:CANIEm）CANm エラー発生時に呼ばれるコールバック関数

割り込みハンドラ：void user_ch0_err_isr(void)、void user_ch1_err_isr(void)

コールバック関数：void user_ch0_err_callback(void)、void user_ch1_err_callback(void)

CAN モジュールのチャネル（CAN0、CAN1）でエラーが発生した場合に割り込みハンドラ処理から呼び出されます。

以下に、チャネル・エラー発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

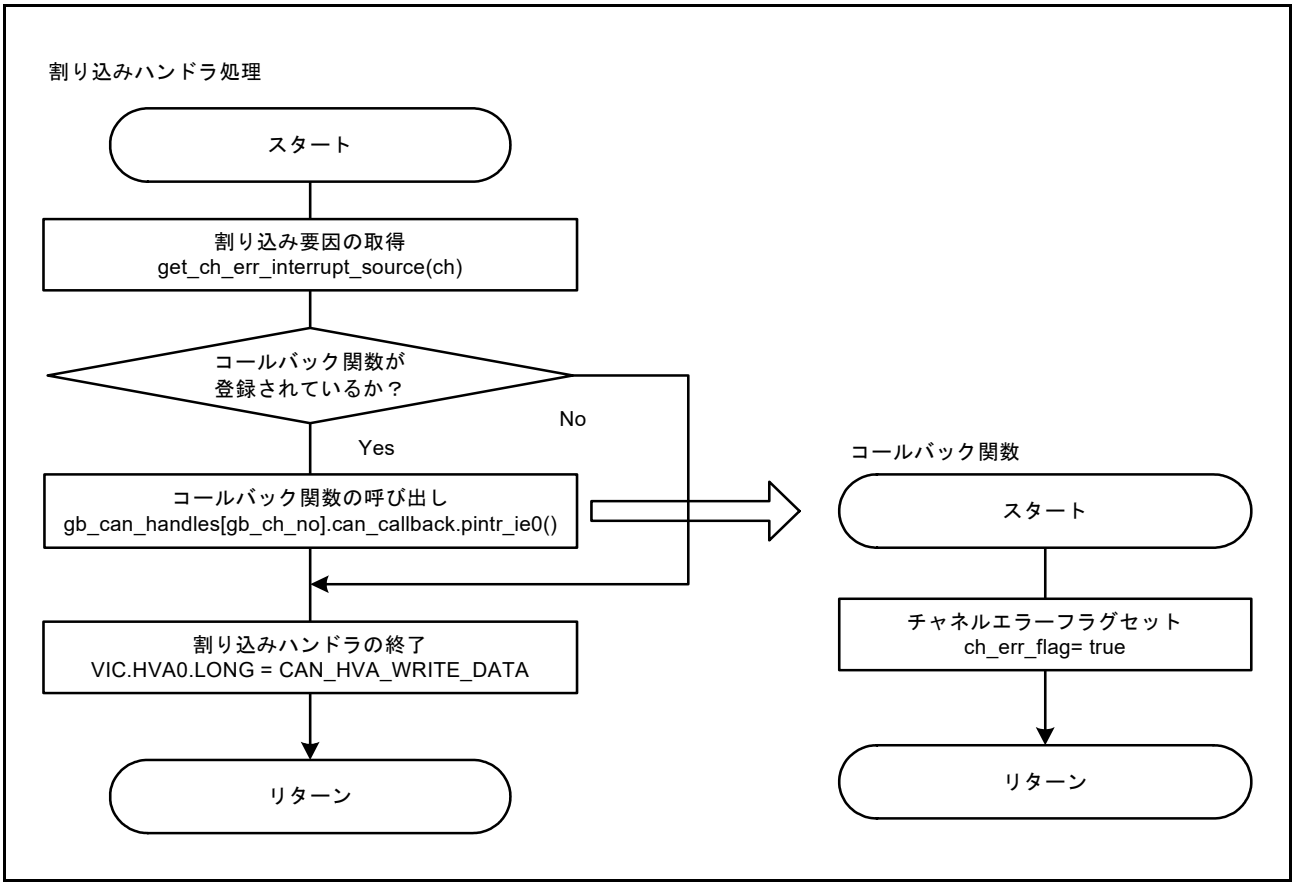


図 10.34　サンプルコードの CANm エラー発生時に呼ばれるコールバック関数処理

（3）（RSCAN:CANRFI）CAN 受信 FIFO 割り込み発生時に呼ばれるコールバック関数

割り込みハンドラ：void user_rx_fifo_isr(void)

コールバック関数：void user_rx_fifo_callback(void)

受信 FIFO バッファを使って受信する設定にしておくと、受信 FIFO バッファにメッセージが溜まると割り込みハンドラ処理から呼び出されます。

以下に、受信 FIFO 割り込み発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

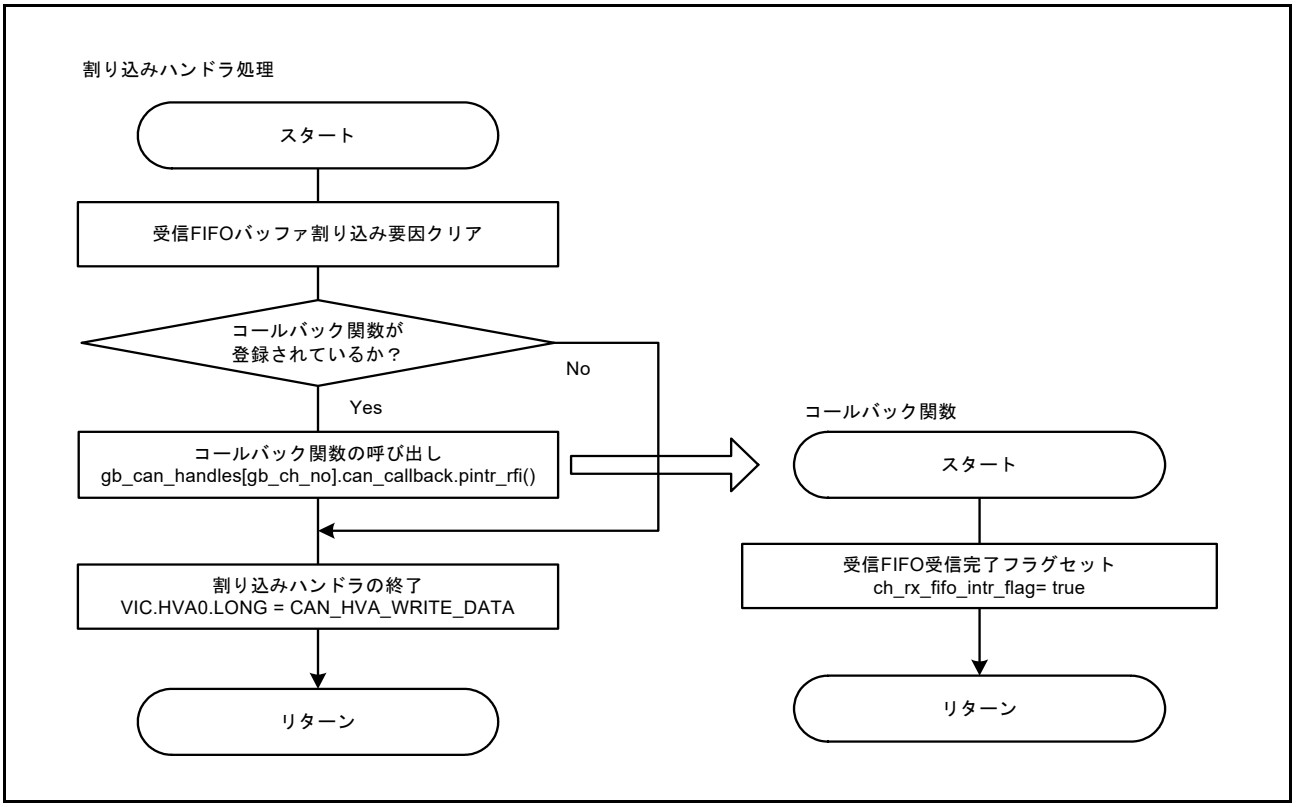


図 10.35　サンプルコードの CAN 受信 FIFO 割り込み発生時に呼ばれるコールバック関数処理

（4）（RSCAN:CANTIm）CANm 送信割り込み発生時に呼ばれるコールバック関数

割り込みハンドラ：void user_ch0_tx_isr(void)、void user_ch1_tx_isr(void)

コールバック関数：void user_ch0_tx_callback(void)、void user_ch1_tx_callback(void)

メッセージの送信が完了すると割り込みハンドラ処理から呼び出されます。

以下に、送信完了割り込み発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

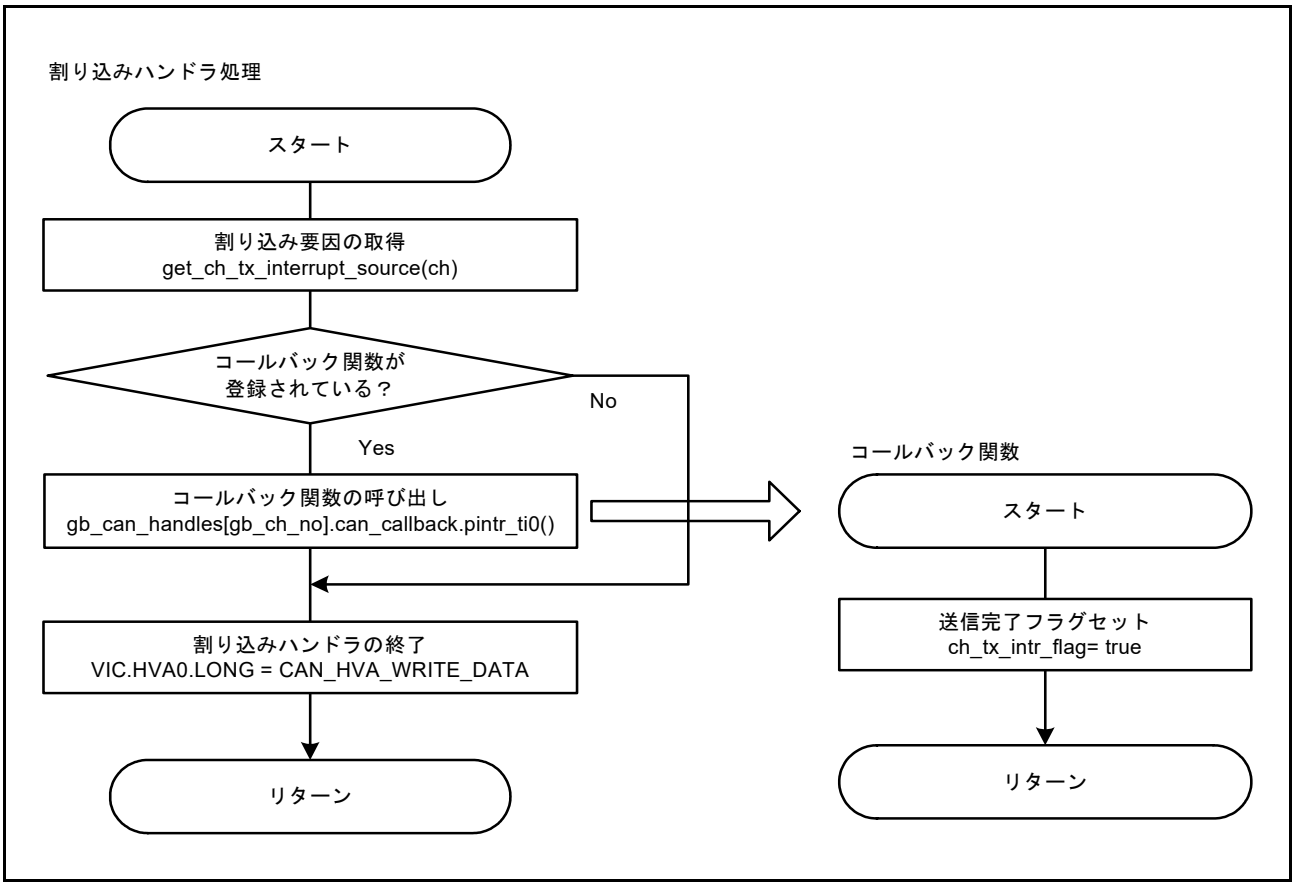


図 10.36　サンプルコードの CANm 送信割り込み発生時に呼ばれるコールバック関数処理

（5）（RSCAN:CANFIRm）CANm 送受信 FIFO 受信完了割り込み発生時に呼ばれるコールバック関数

割り込みハンドラ：void user_ch0_rx_fifo_isr(void)、void user_ch1_rx_fifo_isr(void)

コールバック関数：void user_ch0_rx_fifo_callback(void)、void user_ch1_rx_fifo_callback(void)

送受信 FIFO バッファを使って受信する設定にしておくと、送受信 FIFO バッファにメッセージが溜まると割り込みハンドラ処理から呼び出されます。

以下に、送受信 FIFO 受信完了割り込み発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

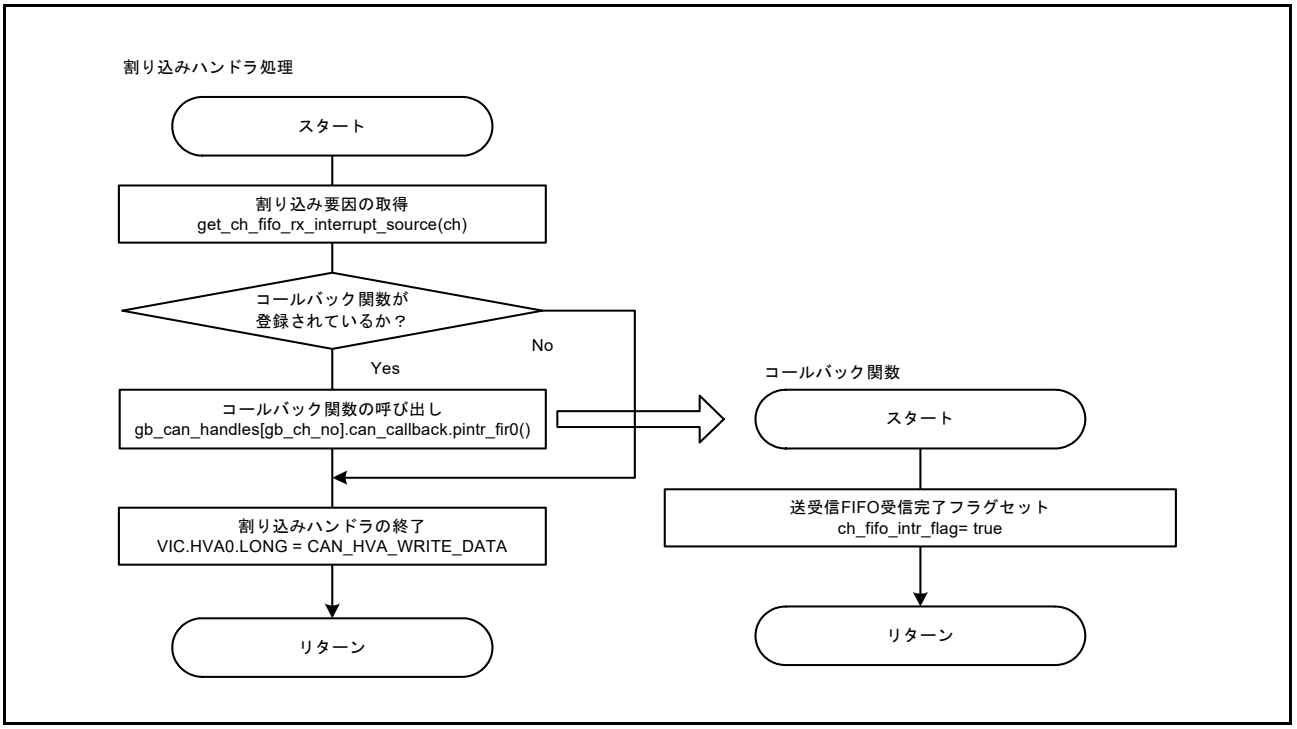


図 10.37　サンプルコードの CANm 送受信 FIFO 受信完了割り込み発生時に呼ばれるコールバック関数処理

（6）（SCIFA:RXIF2）受信 FIFO データフル割り込み発生時に呼ばれるコールバック関数

割り込みハンドラ：void scifa_key_input_isr(void)

コールバック関数：void key_handler_callback(void)

SCIFA のキー入力の割り込み発生時にハンドラから呼ばれます。

本サンプルプログラムは、送信時、メッセージを繰り返し受信側に送信つづけます。途中でメッセージの送信を中止させるために以下のコールバック関数を用意しています。送信中に何かキーが押下されるとキー押下フラグがセットされます。

以下に、キー入力の割り込み発生時に呼ばれるハンドラ処理とその中から呼ばれるコールバック処理のフローチャートを示します。

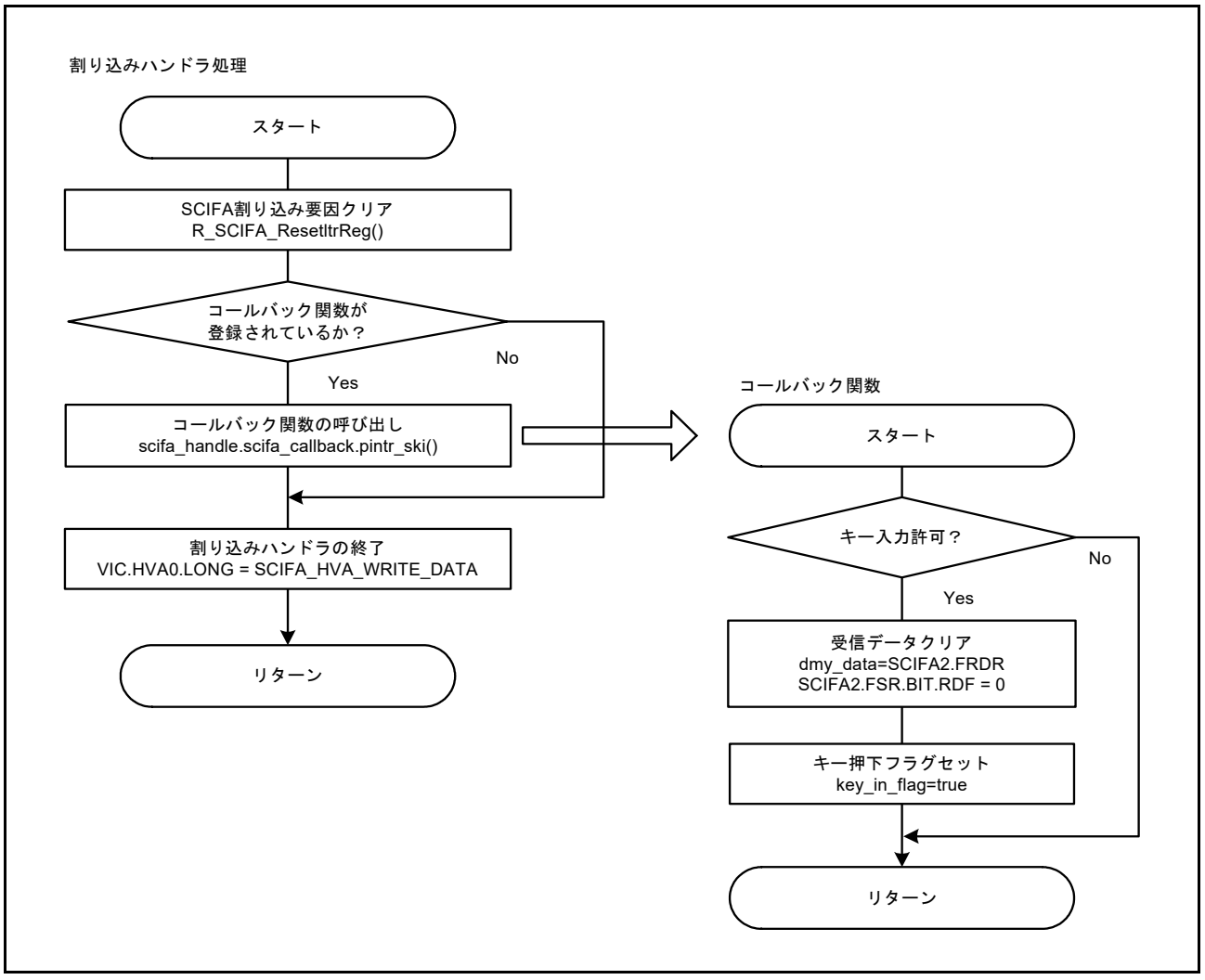


図 10.38　サンプルコードの SCIFA の受信 FIFO データフル割り込み発生時に呼ばれるコールバック関数処理

11. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

12.　参考ドキュメント

- ユーザーズマニュアル：ハードウェア
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- RZ/T1 CPU ボード RTK7910022C00000BR ユーザーズマニュアル
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- テクニカルアップデート／テクニカルニュース
  （最新の情報をルネサス エレクトロニクスホームページから入手してください。）

- ユーザーズマニュアル：開発環境
  IAR 統合開発環境（IAR Embedded Workbench for Arm）に関しては、IAR ホームページから入手してください。
  （最新版を IAR ホームページから入手してください。）

  Arm ソフトウェア開発ツール（Arm Compiler toolchain、Arm DS-5 等）に関しては、Arm ホームページから入手してください。
  （最新版を Arm ホームページから入手してください。）

  ルネサス エレクトロニクスソフトウェア開発ツール（e2studio 等）に関しては、ルネサス エレクトロニクスホームページから入手してください。
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

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/contact/

改訂記録	CANインタフェースサンプルプログラム アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.10	2017.09.14	—	初版発行
1.20	2018.12.12	2. 動作環境	
		8	表2.1 動作環境　統合開発環境の内容変更
		10. ソフトウェア説明	
		104	10.7.27 R_CAN_GetInterruptSource　説明、因数を修正
		12. 参考ドキュメント	
		141	ARM→Armに変更

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 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・システムの設計において、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て生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等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。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   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、産業用ロボット等

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の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。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 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・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配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。

(Rev.4.0-1 2017.11)

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 エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：https://www.renesas.com/contact/