User's Manual

32

RX62T グループ

Renesas Starter Kit ソフトウェアマニュアル

ルネサス 32 ビットマイクロコンピュータ
RX ファミリ
RX600 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.01 2011.7

ご注意書き

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

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

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

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

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

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

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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

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. リザーブアドレスのアクセス禁止
   【注意】リザーブアドレス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。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、RSK ハードウェア概要と電気的特性をユーザに理解していただくためのマニュアルです。様々な周辺装置を使用して、RSK プラットフォーム上のサンプルコードを設計するユーザを対象にしています。

このマニュアルは、RSK 製品の機能概観を含みますが、組み込みプログラミングまたはハードウェア設計ガイドのためのマニュアルではありません。また、RSK および開発環境のセットアップに関するその他の詳細は、チュートリアルに記載しています。

このマニュアルを使用する場合、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

RSKRX62T では次のドキュメントを用意しています。ドキュメントは最新版を使用してください。最新版はルネサスエレクトロニクスのホームページに掲載されています。

ドキュメントの種類	記載内容	資料名	資料番号
ユーザーズマニュアル	RSK ハードウェア仕様の説明	RSKRX62T ユーザーズマニュアル	RJJ10J2788
ソフトウェアマニュアル	Renesas Peripheral Driver Library（RPDL）を備えたサンプルコードの機能とその相互作用の説明	RSKRX62T ソフトウェアマニュアル	RJJ10J2791 (本マニュアル)
チュートリアル	RSK および開発環境のセットアップ方法とデバッギング方法の説明	RSKRX62T チュートリアル	RJJ10J2789
クイックスタートガイド	A4 紙一枚の簡単なセットアップガイド	RSKRX62T クイックスタートガイド	RJJ10J2790
回路図	CPU ボードの回路図	RSKRX62T CPU ボード回路図	RJJ99J0072
ユーザーズマニュアル ハードウェア編	ハードウェアの仕様（ピン配置、メモリマップ、周辺機能の仕様、電気的特性、タイミング）と動作説明	RX62T グループ ユーザーズマニュアル ハードウェア編	R01UH0034JJ

2. 略語および略称の説明

略語／略称	英語名	備考
ADC	Analogue-to-Digital Converter	A/D コンバータ
CPU	Central Processing Unit	中央処理装置
CRC	Cyclic Redundancy Check	巡回冗長検査
DTC	Data Trasfer Controller	データトランスファコントローラ
HEW	High-performance Embedded Workshop	ルネサス統合開発環境
IRQ	Interrupt Request	割り込み要求
LCD	Liquid Crystal Display	液晶ディスプレイ
LED	Light Emitting Diode	発光ダイオード
PC	Personal Computer	パーソナルコンピュータ
PLL	Phase Locked Loop	位相同期回路
RSK	Renesas Starter Kit+	ルネサススタータキット
SFR	Special Function Registers	周辺機能を制御するためのレジスタ
SCI	Serial Communication Interface	シリアルコミュニケーションインタフェース
WDT	Watch Dog Timer	ウォッチドッグタイマ

目次

1. 概要

1.1 目的

本 RSK はルネサスマイクロコントローラ用の評価ツールです。本マニュアルは、Renesas Peripheral Driver Library（RPDL）を備えたサンプルコードの機能とその相互関係について説明します。Renesas Peripheral Driver Library（以下 RPDL またはライブラリと称す）は、ルネサスエレクトロニクスによって作られたマイクロコントローラ用に統一された Application Programming Interface（API）がベースになっています。

本マニュアルは RPDL そのもののマニュアルではなく、サンプルコードで RPDL がどのように使用されているかを説明するものです。RPDL に関する詳細情報はルネサスウェブサイトの Peripheral Driver Generator（PDG）サイトを参照してください。

http://japan.renesas.com/pdg

注：
本 RSK の RPDL は暫定版で、本サンプルコードのみで動作することを確認したものです。

2. サンプルコードコンセプト

2.1 サンプルコードの構成

図 2-1 は全ての RSK サンプルコードの基本的な構成を示しています。最初の関数'Power_On_Reset_PC'と'HardwareSetup'はメインプログラムコードが実行される前にマイクロコントローラの設定を行う関数です。

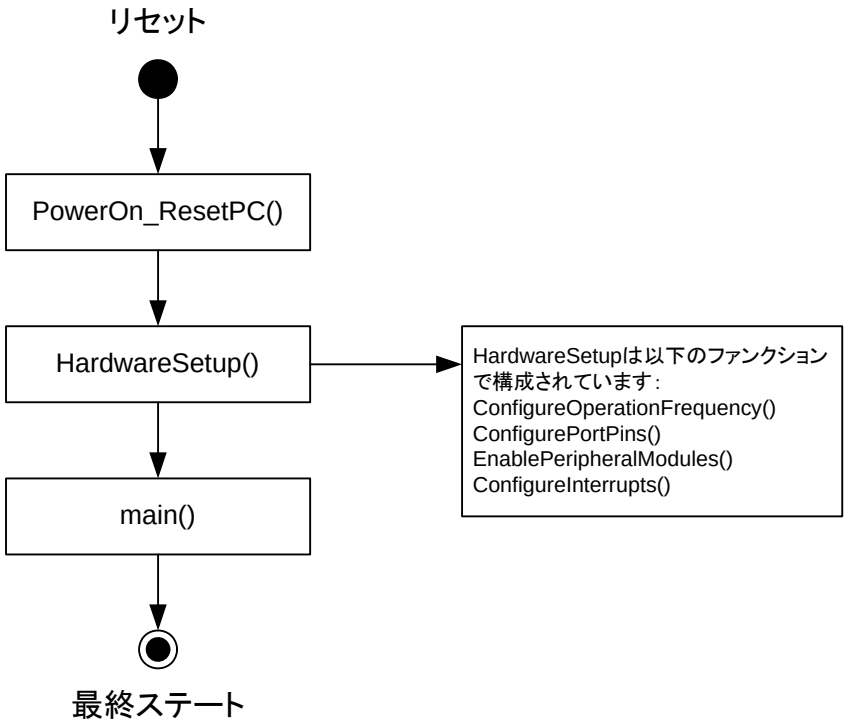


図 2-1: サンプルコードの基本構成

HarwareSetup 関数に含まれる関数、用途を**表 2-1** に示します。

関数	用途/機能	RPDL 関数
ConfigureOperatingFrequency	CPU メインクロック、バスクロック、周辺クロック、リアルタイムクロックおよび PLL 等の初期設定を行います。	R_CGC_Set
ConfigurePortPins	CPU ボード上の装置およびサンプルコードに合ったポートの入出力を設定します。また、ポートの初期レベルを設定します。	R_IO_PORT_Set R_IO_PORT_Write
EnablePeripheralModules	マイクロコントローラの周辺機能の許可/禁止を設定します。RPDL によって制御されません。	-
ConfigureInterrupts	ユーザスイッチに関連した外部割り込みピンを設定・許可し、対応するピンを入力として設定します。	R_INTC_CreateExtInterrupt*

表 2-1: HardwareSetup 関数

* RPDL 関数 R_INTC_CreateExtInterrupt は ConfigureInterrupts 関数によって間接的にコールされます。

2.2 サンプルコードリスト

RSKRX62T のサンプルコードリストを**表 2-2** に示します。

サンプルコード	内容
Tutorial	デバッガおよび RSK ハードウェアの基礎的な使用法を学ぶためのデモコード
Application	マイクロコントローラの初期化コードのみを含んだブランクプロジェクト
ADC10_OneShot	10bit ADC モジュール（ワンショットモード）のデモコード
ADC10_Repeat	10bit ADC モジュール（リピートモード）のデモコード
ADC12_OneShot	12bit ADC モジュール（ワンショットモード）のデモコード
ADC12_Repeat	12bit ADC モジュール（リピートモード）のデモコード
SPI	SPI モジュール（ループバック）のデモコード
LIN_Master	LIN モジュール（マスターモードまたはループバック）のデモコード
Async_Serial	SCI モジュール（非同期型）のデモコード
Sync_Serial	SCI モジュール（同期型）のデモコード
Power_Down	消費電力低減機能のデモコード
LVD	電圧検出回路 1 のデモコード
IIC_Master	I^2C ユニット（マスターモード）のデモコード
IIC_Slave	I^2C ユニット（スレーブモード）のデモコード
CRC_Calc	CRC モジュールのデモコード
Timer_Capture	TMR モジュール（キャプチャ機能）のデモコード
Timer_Compare	TMR モジュール（コンペアマッチ）のデモコード
Timer_Event	MTU3 ユニット（イベント）のデモコード
Timer_Mode	MTU3 ユニット（タイマモード）のデモコード
Flash_Data	FCU モジュールのデモコード
DTC	DTC モジュールのデモコード
WDT	ウォッチドッグタイマのデモコード
IWDT	独立ウォッチドッグタイマのデモコード

表 2-2: サンプルコードリスト

3. Tutorialサンプル

3.1 Tutorial

サンプルコード'Tutorail'はデバッガおよび RSK ハードウェア基礎的な使用方を学ぶためのサンプルコードです。

3.1.1 説明

Tutorial はポートピン制御、割り込み設定、C 変数初期化を行うために 3 つの関数をコールします。これらの関数を**図 3-1** に示します。

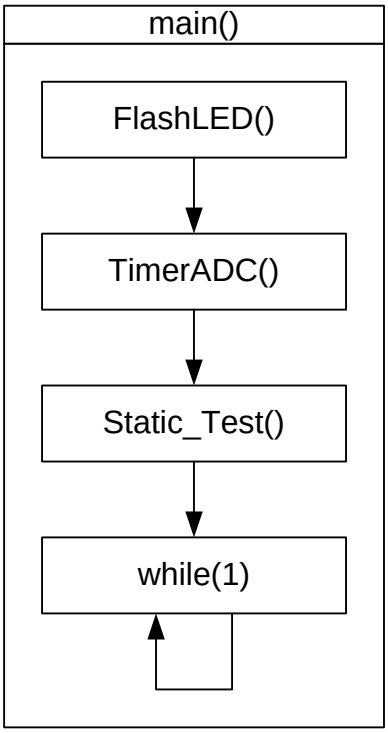


図 3-1: Tutorial フロー

3.1.2 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に’Renesas’、2 行目にマイクロコントローラのグループ名を表示します。
2. FlashLED 関数をコールします。この関数は繰り返し LED をトグル出力するために CMT 割り込みを作り出し、スイッチが押されるか、LED が 200 回トグル出力されるまでループ内で待機します。
3. その後、周期的に AD 変換を起動するために ADC ユニットおよびタイマユニットを形成する TimerADC 関数をコールします。ADC ユニットは AD 変換が完了するたびに、CB_ADCConversion 関数をコールするために形成されます。
4. タイマユニットの周期が経過すると、AD 変換を起動します。一旦、AD 変換が完了すればコールバック関数 CB_ADCConversion が実行されます。コールバック関数は AD 変換結果をフェッチし、新しいタイマ周期を計算するために AD 変換結果を使用します。さらに、コールバック関数は LED をトグルします。
5. TimerADC をコールし、タイマおよび ADC 割り込みのセットアップ後、Static_Test 関数をコールします。
6. Statics_Test 関数は LCD の 2 行目に’STATIC’を表示し、ストリング定数’TESTTEST’に表示内容を置き換えます。置き換えが完了すると、表示内容は(1)の初期表示に戻ります。その後、コードは無限ループ処理に入ります。

3.1.3 シーケンス

Tutorial のプログラム実行フローを**図 3-2** に示します。

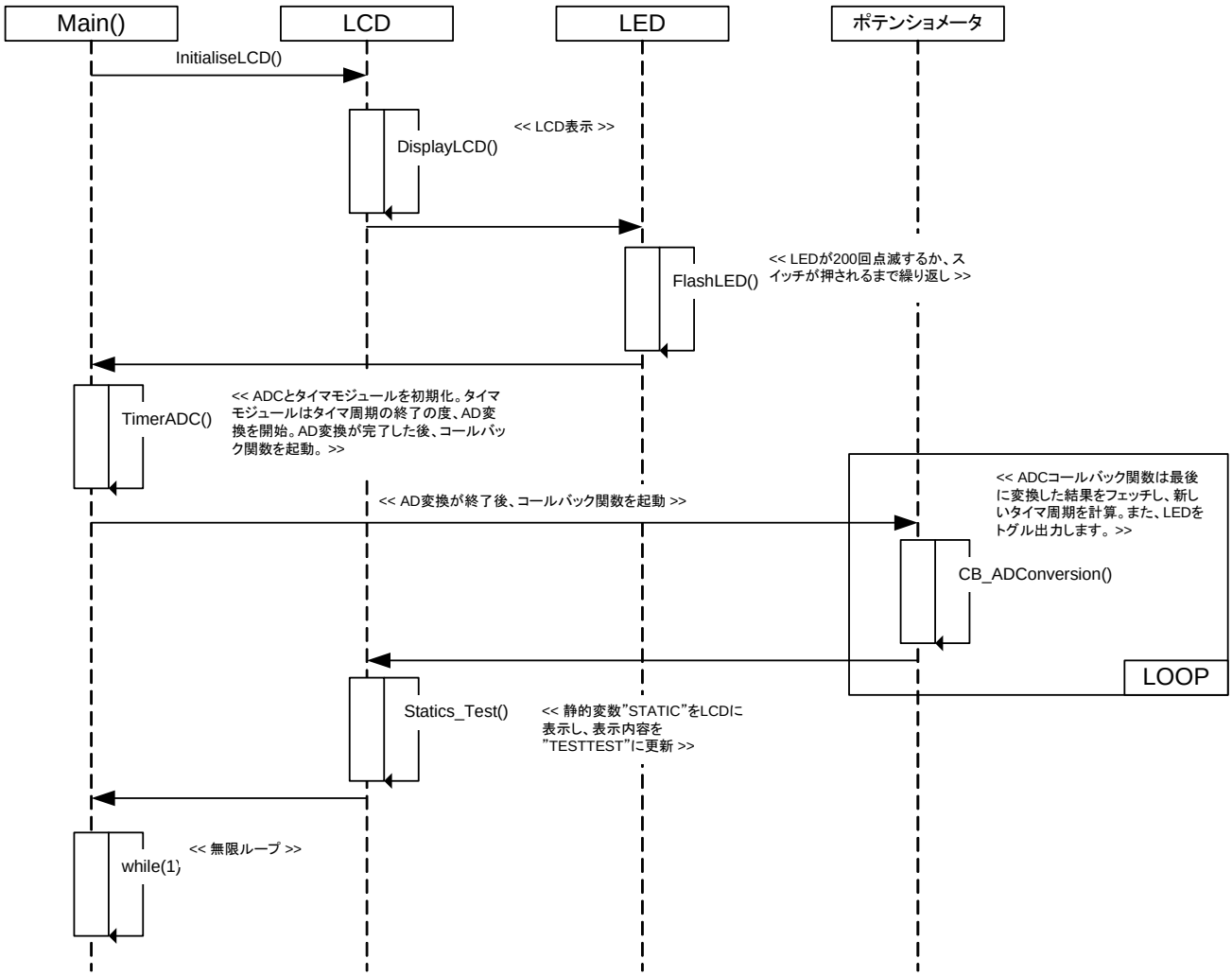


図 3-2: Tutorial フロー

3.1.4 RPDL

Tutorial で使用される関数、RPDL 関数を**表 3-1** に示します。

関数	RPDL 関数
FlashLEDs	R_TMR_Create
	R_CMT_Destroy
TimerADC	R_TMR_CreatePeriodic
	R_ADC_10_Create
CB_ADConversion	R_ADC_10_Read
	R_TMR_ControlPeriodic
CB_TimerADC	R_ADC_10_Control
Statics_Test	R_CMT_CreateOneShot

表 3-1: Tutorial 用関数

3.2 Application

3.2.1 説明

Application はユーザ自身でコード作成するために用意されたサンプルです。メイン関数が実行される前に実行されるハードウェア初期化および設定コードを含みますがメイン関数にはコードがありません。

ハードウェアの初期化および設定に関する詳細は、セクション 2 を参照してください。

4. 周辺機能サンプル

本セクション中のサンプルコードでは、初期化の例およびいくつかの周辺モジュールの使用法について説明します。また、周辺機能をデバッグする方法についても説明します。

4.1 ADC10_OneShot

4.1.1 説明

本サンプルコードはワンショットモード（シングルモード）による 10bit AD 変換のデモコードです。ボード上のスイッチ SW3 を押すと、ポテンショメータ RV1 の入力を AD 変換します。

ポテンショメータは簡易的にマイクロコントローラに可変アナログ入力供給をするために備え付けられています。AD 変換の精度は保証できませんので、予めご了承ください。

4.1.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。
2. ADC ユニットおよびスイッチのコールバック関数を設定する ADC10Oneshot 関数をコールします。
3. コードは無限ループ内で割り込みを待ちます。
4. スイッチ SW3 が押されると割り込みが発生し、コールバック関数 CB_ReadADC をコールして実行します。この関数は AD 変換を起動し、AD 変換結果をキャラクタストリングに変換して LCD にストリングを表示させます。
5. 再度スイッチ SW3 が押されると、再び(4)と同じ動作をします。

4.1.3 シーケンス

ADC10_OneShot サンプルのプログラム実行フローを**図 4-1** に示します。

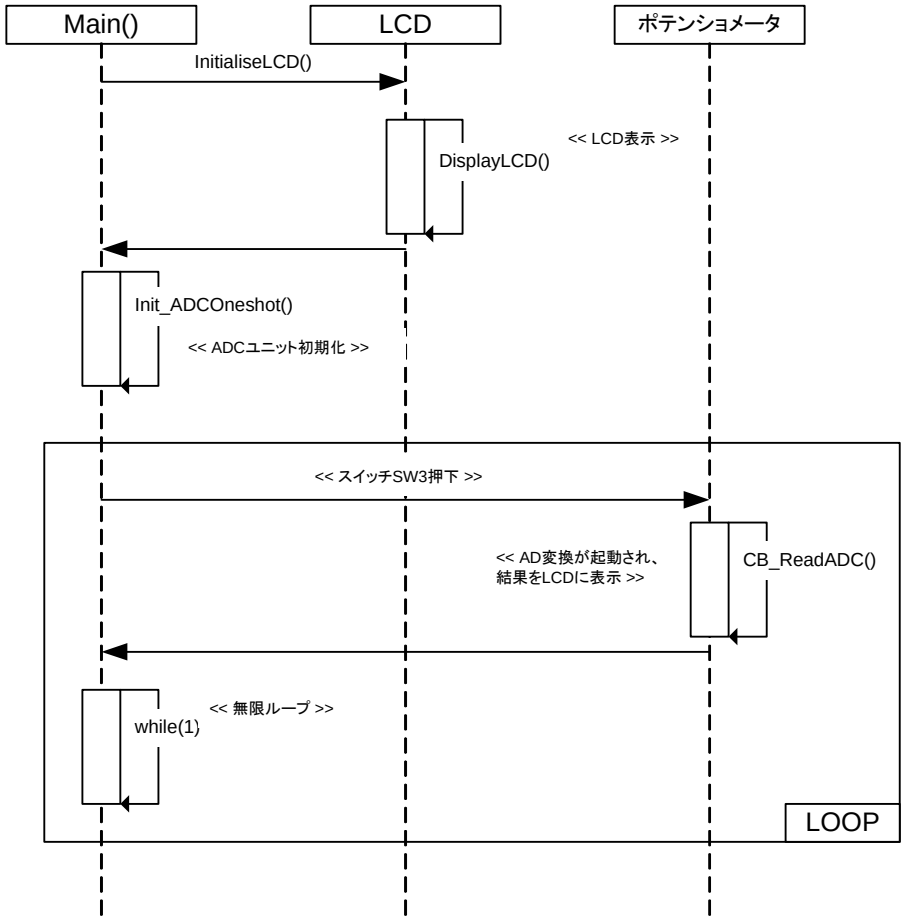


図 4-1: ADC10_OneShot フロー

4.1.4 RPDL

ADC10_Oneshot で使用される関数、RPDL 関数を**表 4-1** に示します。

関数	RPDL 関数
Init_ADCOneshot	R_ADC_10_Create
CB_ReadADC	R_ADC_10_Control
	R_ADC_10_Read

表 4-1: ADC10_Oneshot 用関数

4.2 ADC10_Repeat

4.2.1 説明

本サンプルコードはリピートモード（スキャンモード）による 10bit AD 変換のデモコードです。コードはボード上のポテンショメータ RV1 の入力を繰り返し AD 変換します。また、周期的なタイマ割り込みによって AD 変換値を更新して LCD モジュールに表示します。

ポテンショメータは簡易的にマイクロコントローラに可変アナログ入力供給をするために備え付けられています。AD 変換の精度は保証できませんので、予めご了承ください。

4.2.2 オペレーション

1. LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. ADC10Repeat 関数をコールします。この関数は ADC ユニットをリピートモード用に設定し、周期的な割り込みを発生させるための CMT（コンペアマッチタイマ）を設定します。そして、コールバック関数 CB_CMTADC をコールします。
3. コードは無限ループ内で CMT 割り込み発生を待ちます。
4. CB_CMTADC 関数は AD 変換結果をフェッチし、結果をキャラクタストリングに変換して LCD にストリングを表示させます。割り込みは 250ms ごとに発生します。

4.2.3 シーケンス

ADC10_Repeat サンプルのプログラム実行フローを**図 4-2** に示します。

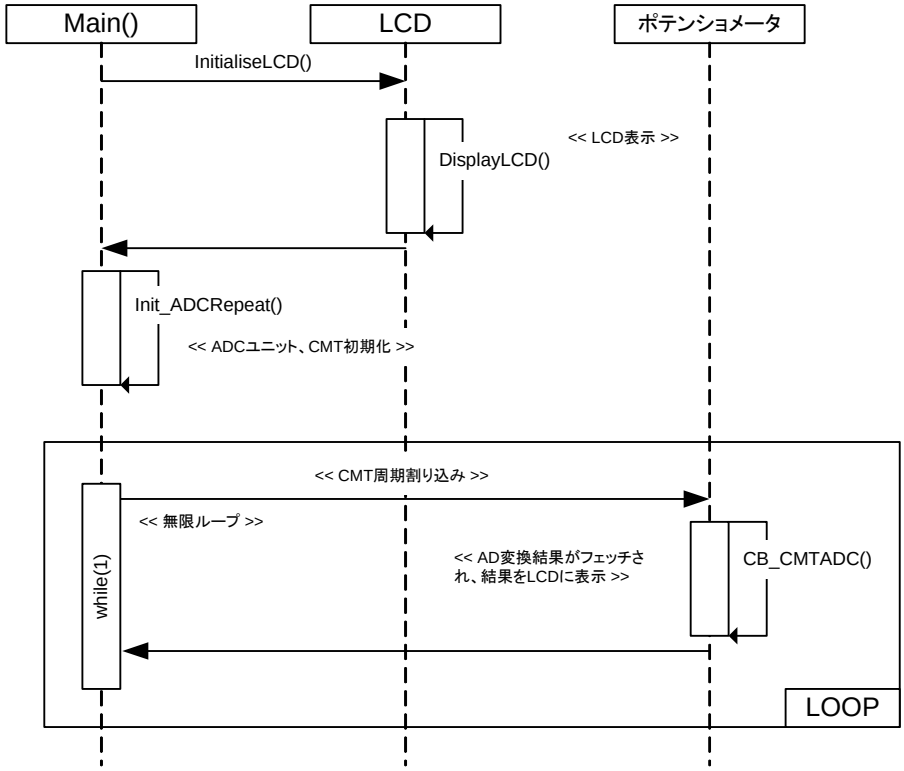


図 4-2: ADC10_Repeat フロー

4.2.4 RPDL

ADC10_Repeat で使用される関数、RPDL 関数を**表 4-2** に示します。

関数	RPDL 関数
Init_ADCRepeat	R_ADC_10_Create
	R_ADC_10_Control
	R_CMT_Create
CB_CMTADC	R_ADC_10_Read

表 4-2: ADC10_Repeat 用関数

4.3 ADC12_OneShot

4.3.1 説明

本サンプルコードはワンショットモード（シングルモード）による 12bit AD 変換のデモコードです。ボード上のスイッチ SW3 を押すと、ポテンショメータ RV1 の入力を AD 変換します。

ポテンショメータは簡易的にマイクロコントローラに可変アナログ入力供給をするために備え付けられています。AD 変換の精度は保証できませんので、予めご了承ください。

4.3.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。
2. ADC ユニットおよびスイッチのコールバック関数を設定する ADC12Oneshot 関数をコールします。
3. コードは無限ループ内で割り込みを待ちます。
4. スイッチ SW3 が押されると割り込みが発生し、コールバック関数 CB_ReadADC をコールして実行します。この関数は AD 変換を起動し、AD 変換結果をキャラクタストリングに変換して LCD にストリングを表示させます。
5. 再度スイッチ SW3 が押されると、再び(4)と同じ動作をします。

4.3.3 シーケンス

ADC12_Oneshot サンプルのプログラム実行フローを**図 4-3** に示します。

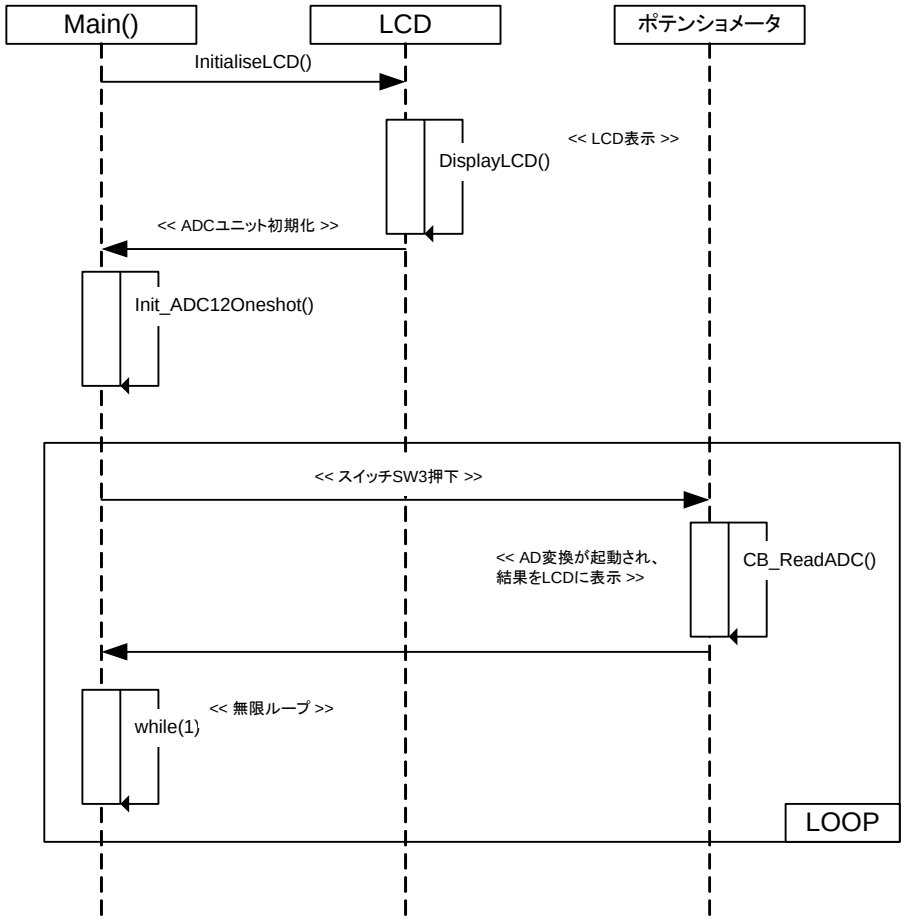


図 4-3: ADC12_Oneshot フロー

4.3.4 RPDL

ADC12_Oneshot で使用される関数、RPDL 関数を**表 4-3** に示します。

関数	RPDL 関数
Init_ADC12Oneshot	R_ADC_12_CreateUnit
CB_ReadADC	R_ADC_Control
	R_ADC_12_Read

表 4-3: ADC12_Oneshot 用関数

4.4 ADC12_Repeat

4.4.1 説明

本サンプルコードはリピートモード（スキャンモード）による 12bit AD 変換のデモコードです。コードはボード上のポテンショメータ RV1 の入力を繰り返し AD 変換します。また、周期的なタイマ割り込みによって AD 変換値を更新して LCD モジュールに表示します。

ポテンショメータは簡易的にマイクロコントローラに可変アナログ入力供給をするために備え付けられています。AD 変換の精度は保証できませんので、予めご了承ください。

4.4.2 オペレーション

1. LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. ADC12Repeat 関数をコールします。この関数は ADC ユニットをリピートモード用に設定し、周期的な割り込みを発生させるための CMT（コンペアマッチタイマ）を設定します。そして、コールバック関数 CB_CMTADC をコールします。
3. コードは無限ループ内で CMT 割り込み発生を待ちます。
4. CB_CMTADC 関数は AD 変換結果をフェッチし、結果をキャラクタストリングに変換して LCD にストリングを表示させます。割り込みは 250ms ごとに発生します。

4.4.3 シーケンス

ADC12_Repeat サンプルのプログラム実行フローを**図 4-4** に示します。

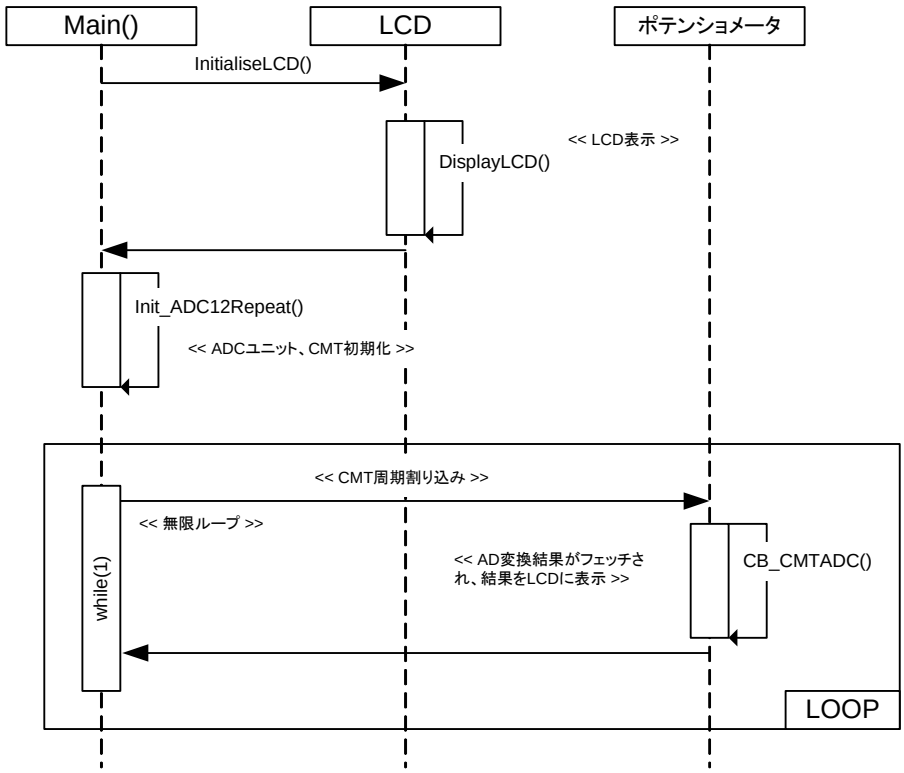


図 4-4: ADC12_Repeat フロー

4.4.4 RPDL

ADC12_Repeat で使用される関数、RPDL 関数を**表 4-4** に示します。

関数	RPDL 関数
Init_ADC12Repeat	R_ADC_12_CreateUnit
	R_ADC_12_Control
	R_CMT_Create
CB_CMTADC	R_ADC_12_Read

表 4-4: ADC12_Repeat 用関数

4.5 SPI

4.5.1 説明

本サンプルコードは SPI のデモコードです。SPI モジュールは内部の送受信ができるようにループバック通信用に設定されます。

4.5.2 オペレーション

1. LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. Init_SPI 関数をコールします。この関数はフレーム送信の数を 1 として SPI チャネルをマスターモードに設定します。ダイレクトデータ転送を備えたループバックモードでビット操作は使用されません。転送ビット長は 16 に設定されます。
3. ADC チャネルは送信データ用にシングルモードに設定されます。CB_Switch 関数はスイッチ開放のコールバック関数として指定されます。
4. コードは無限ループ内で待機します。
5. スイッチが押されると割り込みが発生します。CB_Switch 関数は押されたスイッチが SW3 だったかをチェックします。SW3 の場合には、転送が開始されます。
6. 送信されたデータは受信データと比較されます。転送が成功した場合は受信データで LCD 表示が更新されます。転送が失敗した場合は LCD に転送失敗が通知されます。

4.5.3 シーケンス

SPI サンプルのプログラム実行フローを**図 4-5** に示します。

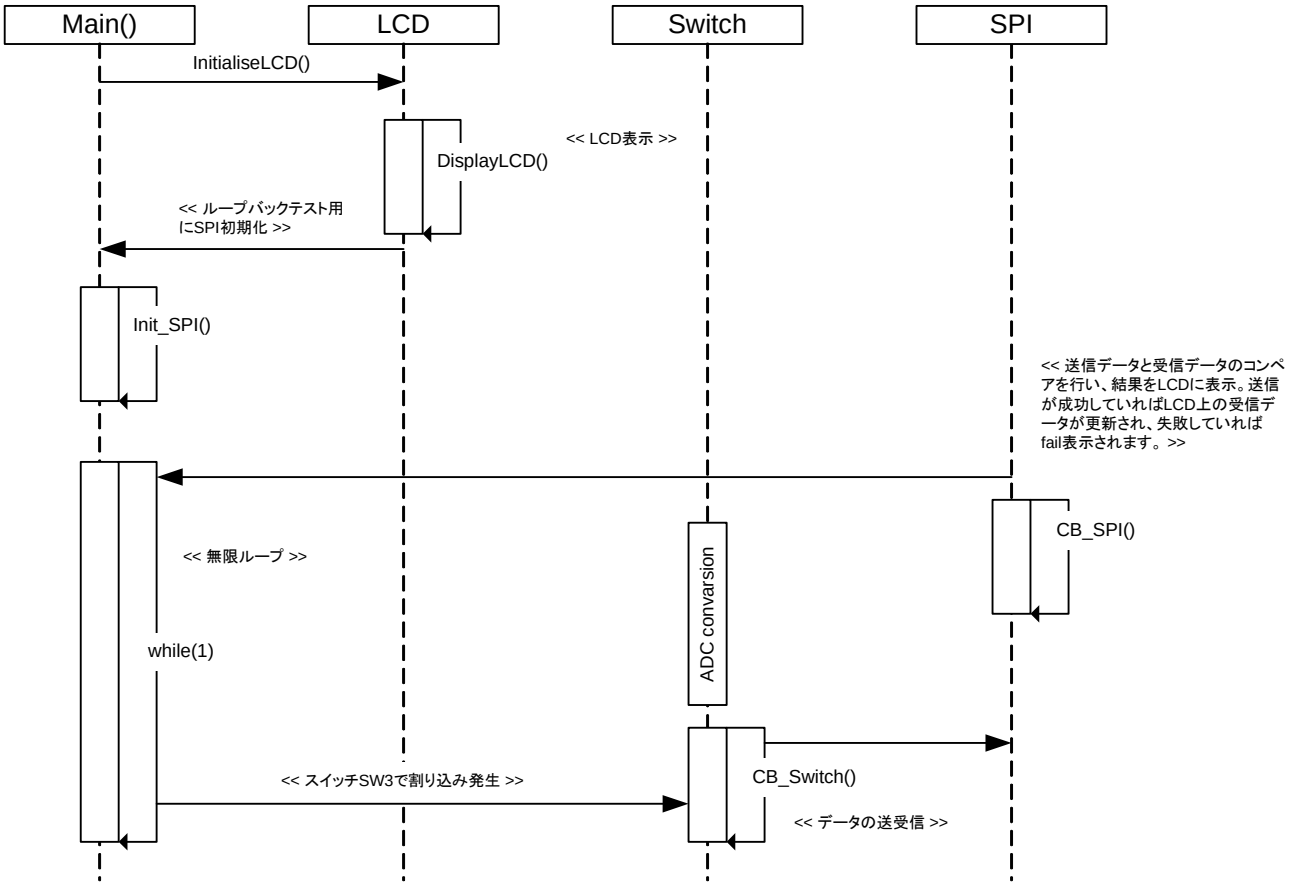


図 4-5: SPI フロー

4.5.4 RPDL

SPI で使用される関数、RPDL 関数を**表 4-5** に示します。

関数	RPDL 関数
Init_SPI	R_SPI_Create
	R_SPI_Control
	R_SPI_Command
	R_ADC_Create
CB_Switch	R_ADC_Control
	R_ADC_Read
	R_SPI_Transfer

表 4-5: SPI 用関数

4.6 LIN_Master

4.6.1 説明

本サンプルコードは LIN のデモコードです。LIN モジュールはユーザ設定によりマスターモードまたはループバックモードに設定できます。

4.6.2 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に選択された動作モード'LIN_Mstr'または'LIN_LOOP'、2 行目に'Push SW3'を表示します。
2. Init_LIN 関数をコールします。この関数はコールバック関数 CB_ReadStatus を備えておりマスターモードにおける LIN チャネルを形成します。マスターモードにおいては 1 つのスイッチコールバック関数 CB_FrameTransmit だけが形成され、ループバックモードにおいては 2 つのスイッチコールバック関数が形成されます。自己テストモードにおける LIN モジュールを形成し、シングルモードのための ADC チャネルを形成するために、スイッチ押下検出によってスイッチコールバック関数 CB_Self_test がコールされます。AD 変換を開始し、変換結果を読んで LCD に値を表示するために、スイッチ開放検出によってスイッチコールバック関数 LIN_Self_Test がコールされます。
3. その後、サンプルコードは無限ループ内で待機します。
4. マスターモードにおいて、スイッチ押下により割り込みが発生し、CB_FrameTransmit 関数とコールします。この関数は押されたスイッチが SW3 だったかどうかをチェックします。SW3 が押された場合に転送が開始されます。転送ステータスは転送が成功したかどうかを判断するのに使用されます。成功した場合、LCD に受信データが表示されます。失敗した場合、エラーメッセージが表示されます。ループバックモードにおいて、スイッチ SW3、自己テストモードにおける LIN モジュール、シングルモードのための ADC チャネルを設定します。スイッチの開放によって AD 変換を開始し、変換結果を 8 バイトの ASCII ストリングデータに変換します。バイトデータはビット反転されて LIN データバッファにロードされます。データは TX ピンを経由して TX ピンに接続された RX ピンに送られます。
5. 送信データは受信データと比較されます。送信データと受信データが一致（転送が成功）している場合、LCD の表示は受信データで更新されます。転送が失敗した場合、LCD に転送の失敗を通知します。

4.6.3 シーケンス

LIN_Master サンプルのプログラム実行フローを**図 4-6** に示します。

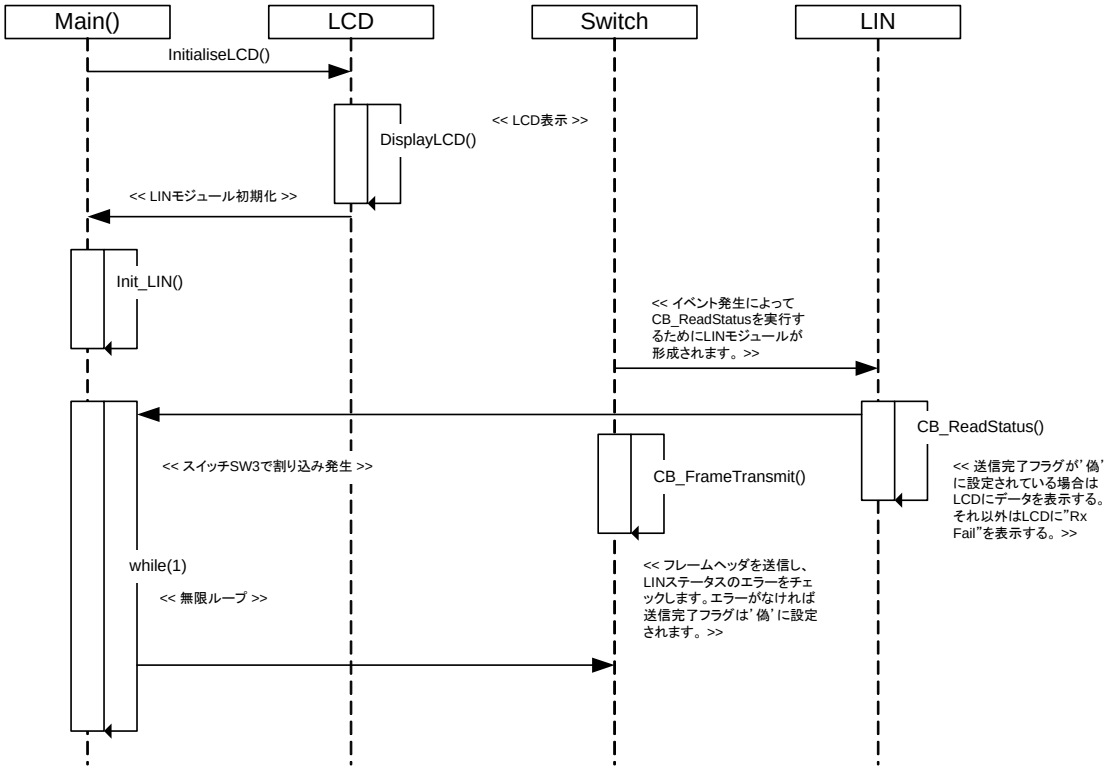


図 4-6: LIN_Master フロー

4.6.4 RPDL

LIN_Master で使用される関数、RPDL 関数を**表 4-6** に示します。

関数	RPDL 関数
Init_LIN	R_IO_PORT_Write
	R_IO_PORT_Set
	R_LIN_Create
	R_CMT_CreateOneShot
CB_FrameTransmit	R_CMT_CreateOneShot
	R_LIN_Transfer
	R_LIN_Read
	R_LIN_GetStatus
LIN_Self_Test	R_ADC_10_Control
	R_ADC_10_Read
	R_LIN_Transfer
	R_LIN_Read
CB_Self_Test	R_LIN_Control
	R_ADC_10_Create

表 4-6: LIN_Master 用関数

4.7 Async_Serial

4.7.1 説明

本サンプルコードは非同期モードによる SCI のデモコードです。RS232 ケーブルを経由して PC 上のターミナルソフトと通信します。

4.7.2 オペレーション

1. サンプルコードを実行する前に、RS232 ケーブルを経由して PC とボードを接続し、ターミナルソフトを起動します（サンプルコードのコメント欄でインストラクションを確認できます）。
2. サンプルコードは LCD モジュールを初期化し、LCD の 1 行目に’Async’、2 行目に’Serial’を表示します。
3. Init_Async 関数で SCI と CMT を設定し、LCD にインストラクションを送ります。その後、無限ループに入り割り込み発生を待ちます。
4. CMT は 100ms ごとに周期的な割り込みを発生させ、コールバック関数 CB_CMTTimer をコールします。
5. CB_CMTTimer 関数は SCI チャネルのステータスをフェッチし、チャネルが開いていたら Transmit_Async 関数をコールします。チャネルがビジーの場合、無限ループに戻ります。
6. Transmit_Async 関数は変数 gSCI_Flag をチェックし、変数が真のときに ASCII 番号（0 から 9 のインクリメントを繰り返す）をターミナルに表示します。変数が偽のときはターミナルには書かれずに関数から戻ります。

4.7.3 シーケンス

Async_Serial サンプルのプログラム実行フローを図 **4-7** に示します。

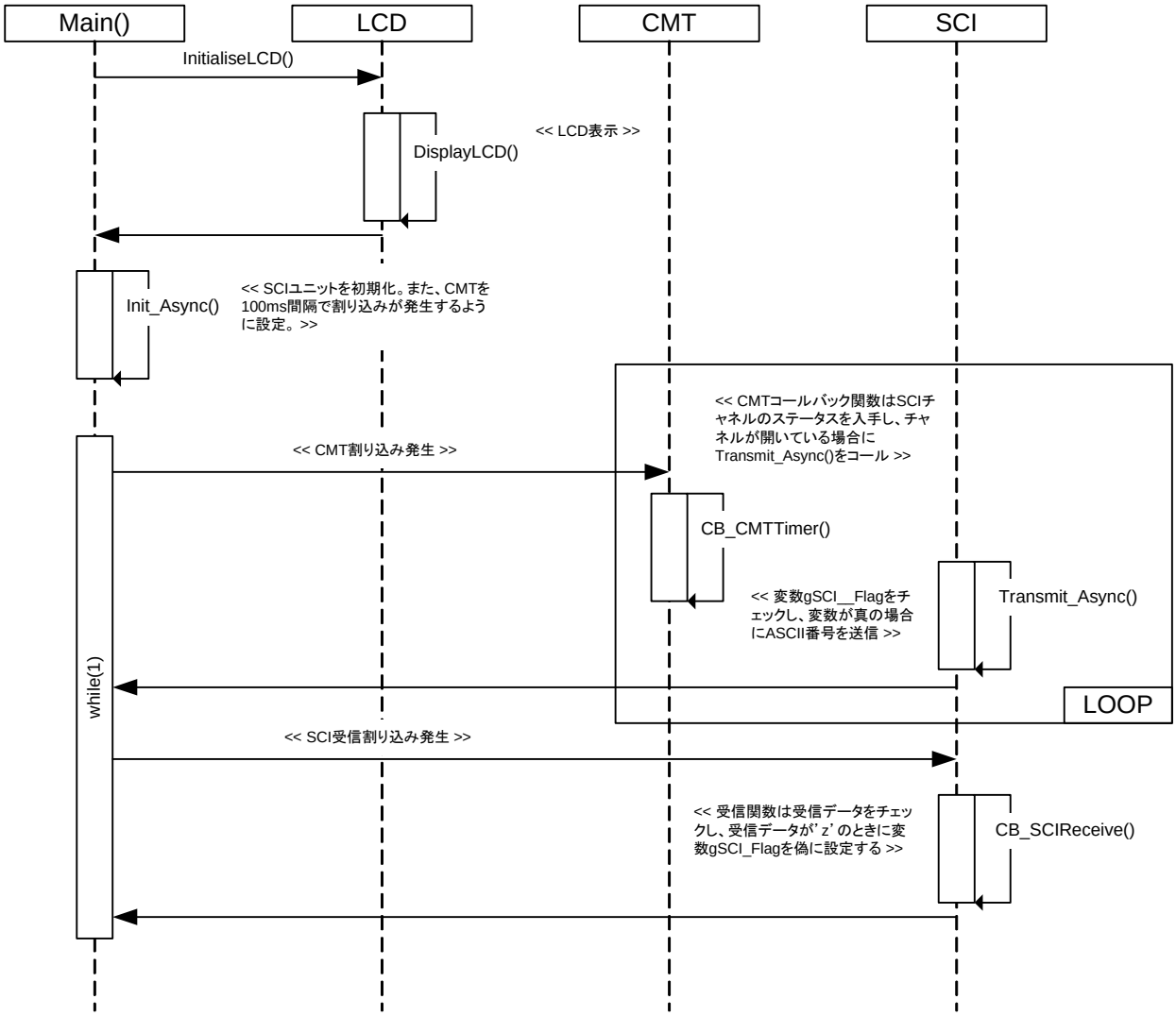


図 4-7: Async_Serial フロー

4.7.4 RPDL

Async_Serial で使用される関数、RPDL 関数を**表 4-7** に示します。

関数	RPDL 関数
Init_Async	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
	R_CMT_Create
CB_CMTTimer	R_SCI_GetStatus
Transmit_Async	R_SCI_Send
CB_SCIReceive	R_SCI_Receive

表 4-7: Async_Serial 用関数

4.8 Sync_Serial

4.8.1 説明

本サンプルコードは同期モードによる SCI のデモコードです。2 つの SCI チャネル間をループバックさせ通信します。

4.8.2 オペレーション

1. LCD モジュールを初期化し、サンプルコード名を表示します。
2. SCI データバッファをクリアする Init_Sync 関数をコールします。
3. SCI2toSCI0Transfer_Sync 関数をコールし、SCI チャネルを設定します。その後、チャネル 2 からチャネル 0 にストリングデータを送信します。データを受信すると CB_SCI0Receive 関数がコールされ、受信データが正しいかチェックします。
4. SCI0toSCI2Transfer_Sync 関数をコールし、チャネル 0 からチャネル 2 にデータを送信します。データを受信すると CB_SCI2Receive 関数がコールされます。
5. CB_SCI2Receive 関数は両方の送信が成功しているかチェックし、成功していれば LCD に”Success”を表示します。送信が失敗している場合、LCD に”Failure”を表示します。
6. その後、サンプルコードは無限ループに入ります。

4.8.3 シーケンス

Sync_Serial サンプルのプログラム実行フローを**図 4-8** に示します。

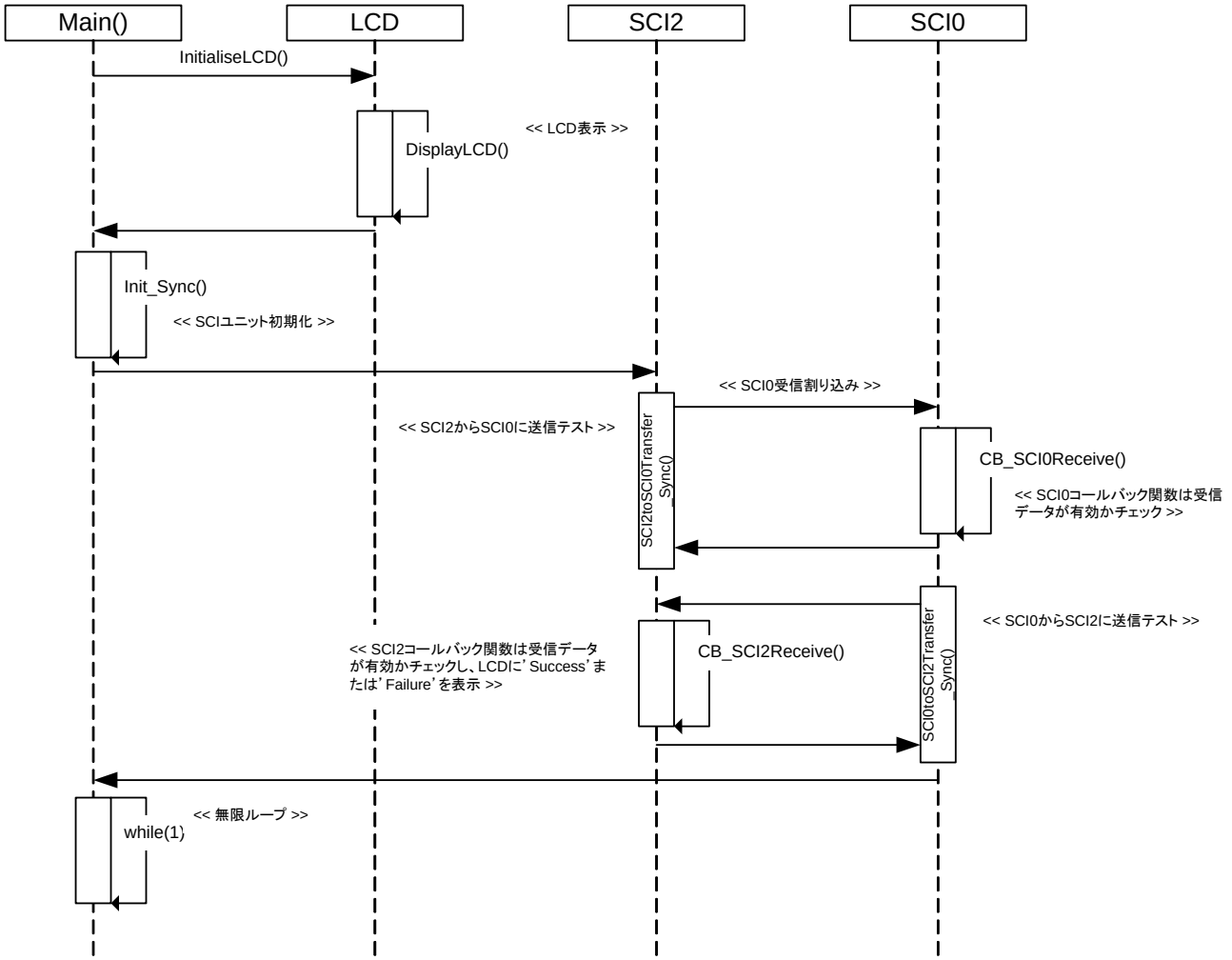


図 4-8: Sync_Serial フロー

4.8.4 RPDL

Sync_Serial で使用される関数、RPDL 関数を**表 4-8** に示します。

関数	RPDL 関数
SCI0toSCI2Transfer_Sync	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
SCI2toSCI0Transfer_Sync	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send

表 4-8: Sync_Serial 用関数

4.9 Power_Down

4.9.1 説明

本サンプルコードは低消費電力関連のレジスタを設定し、スイッチによりスタンバイモードにエントリします。

4.9.2 オペレーション

1. サンプルコードは LCD モジュールを初期化し、LCD の 1 行目に'Pwr Mode'、2 行目に現在のモード状態'Active'を表示します。
2. Init_PowerDown 関数をコールし、低費消費電量に関連するレジスタを設定します。
3. その後、CMT のワンショットタイマ機能を使って LED 点滅を制御するための Flash_LEDs 関数をコールします。関数は無限ループ内でスイッチ（変数 gSwitchFlag）をポーリングします。
4. スイッチ SW1 が押されると、Standby_PowerDown 関数がコールされます。
5. Standby_PowerDown 関数は LCD の 2 行目を'Standby'に変え、LED を消灯させます。そして変数 gSwitchStandbyReady のポーリングによりスタンバイに入るためにマイクロコントローラを準備し、変数が真であるまで待ちます。もしもスイッチが押され続けていると、関数はスタンバイに入るのを待っていることを示すために LED3 を点灯させます。スイッチが開放されると、LED は全て消灯させてスタンバイモードに入ります。
6. いずれかのスイッチを押すと、マイクロコントローラはスタンバイから復帰します。LCD の 2 行目は'Active'を表示し、LED が点灯します。

4.9.3 シーケンス

Power_Down サンプルのプログラム実行フローを**図 4-9** に示します。

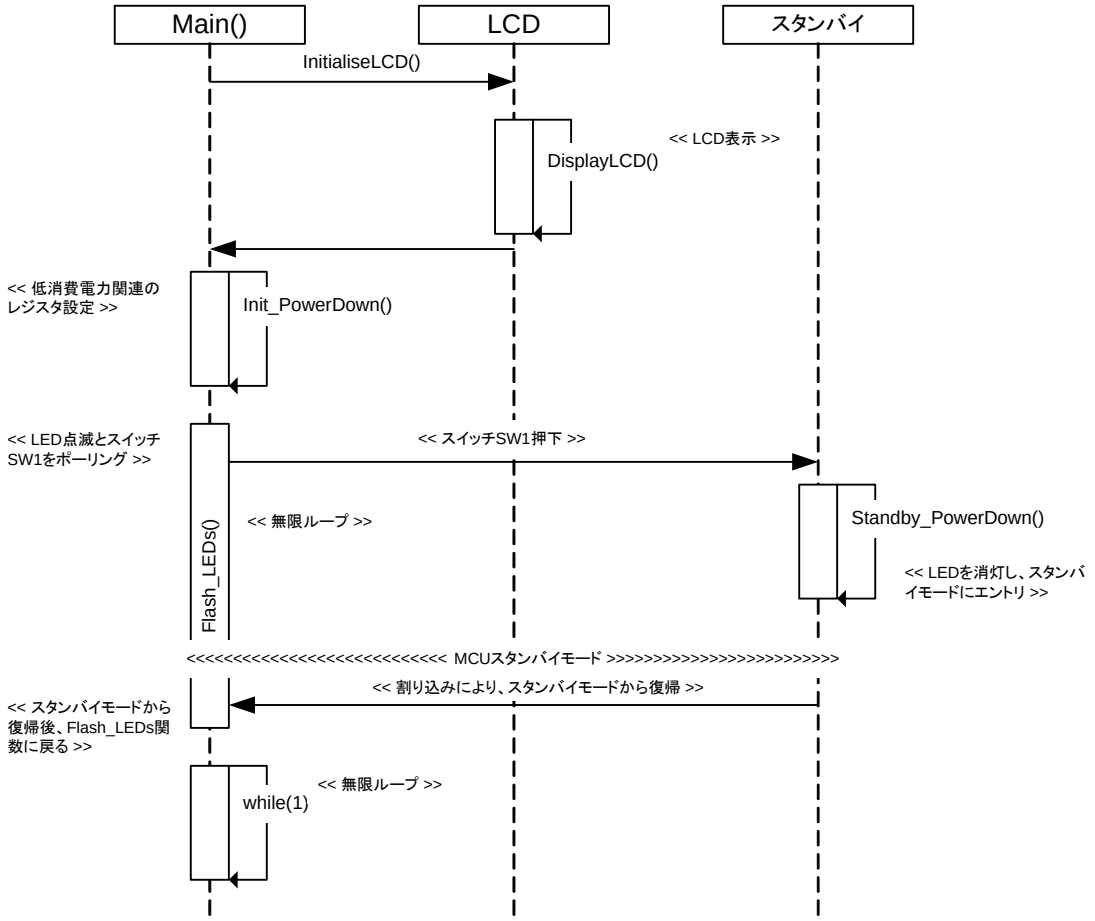


図 4-9: Power_Down フロー

4.9.4 RPDL

Power_Down で使用される関数、RPDL 関数を**表 4-9** に示します。

関数	RPDL 関数
Init_PowerDown	R_LPC_Create
Flash_LEDs	R_CMT_CreateOneShot
	R_IO_PORT_Modify
	R_IO_PORT_Write
Standby_PowerMode	R_IO_PORT_Write
	R_LPC_Control

表 4-9: Power_Down 用関数

4.10 LVD

4.10.1 説明

本サンプルコードは電圧検出回路 1 を設定し、電源電圧が検出レベルに達すると割り込みを発生させます。

4.10.2 オペレーション

1. サンプルコードは Init_LVD 関数をコールします。電源電圧が検出レベルに達すると割り込みを発生させるよう電圧検出回路 1（LVD1）を形成します。
2. LVD 割り込みはノンマスカブル割り込みでマスカブル割り込みとは異なって扱われます。LVD 割り込みはコールバック関数 CB_LVD1 と共に立ち下がりエッジ検出を備えた NMI として形成されます。CB_LVD1 関数はコールされるたびに LED0～LED2 を消灯し、LED3 を点灯させます。また、LED トグル出力に同期させるために 500ms 間隔ごとの周期的な割り込みを発生させるコールバック関数 CB_CMTimer を形成します。
3. その後、電圧検出および CMT タイムアウトの割り込みのたびにコードは無限ループに入ります。

4.10.3 シーケンス

LVD サンプルのプログラム実行フローを**図 4-10** に示します。

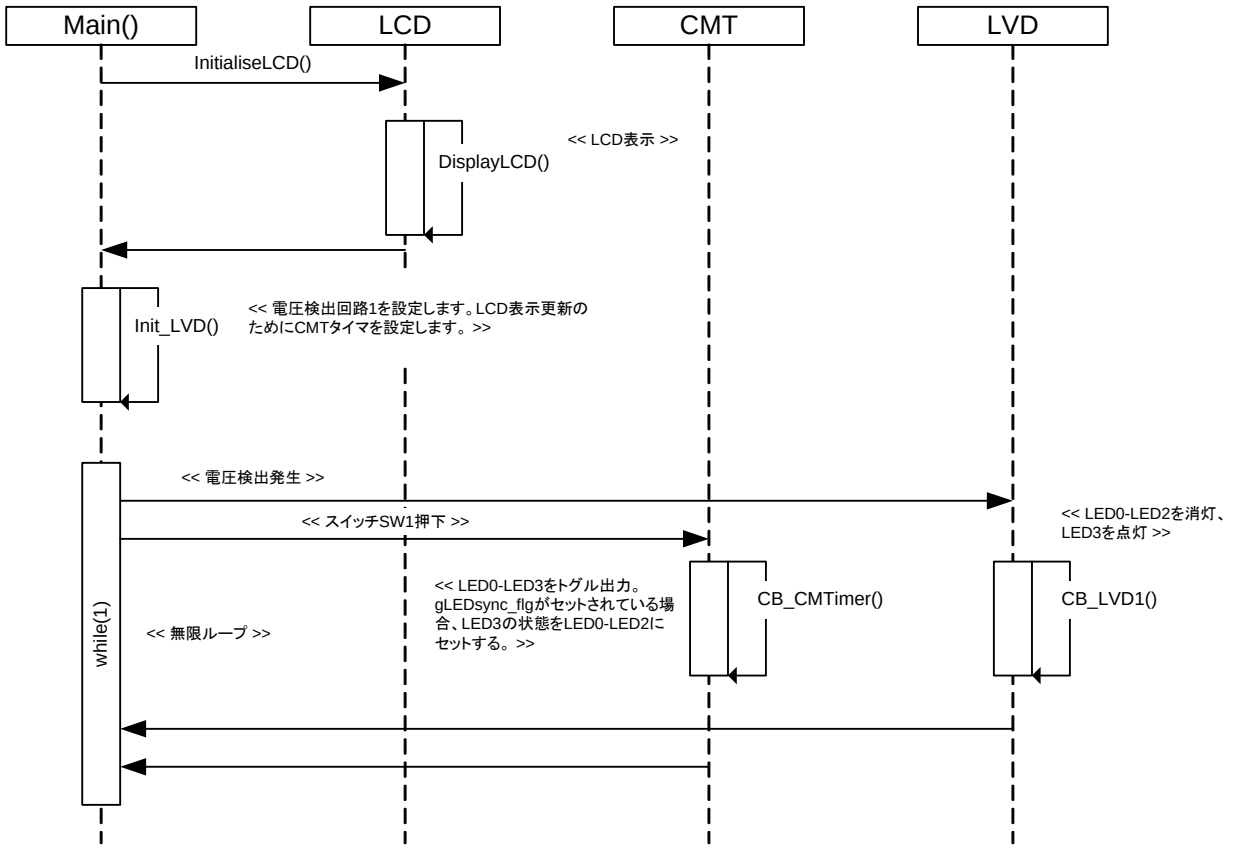


図 4-10: LVD フロー

4.10.4 RPDL

LVD で使用される関数、RPDL 関数を**表 4-10** に示します。

関数	RPDL 関数
Init_LVD	R_LVD_Control
	R_INTC_CreateExtInterrupt
	R_CMT_Create
CB_CMTimer	R_IO_PORT_Modify
CB_LVD1	R_CMT_Control
	R_IO_PORT_Write

表 4-10: LVD 用関数

4.11 IIC_Master

4.11.1 説明

本サンプルコードはマスターモードにおける IIC ユニットより EEPROM メモリへのリード/ライトオペレーションをデモします。サンプルコードはルネサスデバイスで動作するように設定されています。

・HN58X24512I, 1Mbit EEPROM, 1MHz

4.11.2 オペレーション

1. サンプルコードは LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. その後、サンプルコードは IIC マスターシーケンスループに入ります。そこでは Init_EEPROM_Master 関数が最初にコールされ、IIC ユニットはマスターモードに形成されます。
3. その後、ユーザスイッチをポーリングしながらマスターシーケンスは無限ループで待機します。
4. スイッチが押されるとスイッチコールバック関数を実行し、どのスイッチが押されたかチェックします。
5. スイッチ SW2 が押された場合、EEPROM ライトオペレーションは Write_EEPROM_Master 関数を使って実行されます。ライトオペレーションはストリング”XXRenesas IIC”をライトします。XX はライトオペレーションのたびにインクリメントされます。
6. ライトオペレーションは常に EEPROM の最初のメモリアドレスから開始し、ライトが完了した後に LCD に書かれたストリングデータ識別子を表示します。ライトオペレーションが失敗した場合、LCD は”Error W.”を表示します。
7. スイッチ SW3 が押された場合、EEPROM リードオペレーションは Read_EEPROM_Master 関数を使って実行されます。リードオペレーションはリード完了後に 0 から開始し、次の 16 バイト位置までインクリメントします。
8. リードデータが期待値”XXRenesas IIC”と一致する場合、データ識別子（XX）はリードオペレーションの成功を知らせるために LCD に表示されます。リードオペレーションが失敗した場合、LCD は”Error R.”を表示します。
9. IIC 動作が進行中の間、LED1 は点灯し続けます（転送が成功している間点滅します）。点灯が続く場合は IIC バスがロックアップしています。バスを再び開放するためにマスター側とスレーブ側の両方のデバイスをリスタートしてください。

4.11.3 シーケンス

IIC_Master サンプルのプログラム実行フローを**図 4-11** に示します。

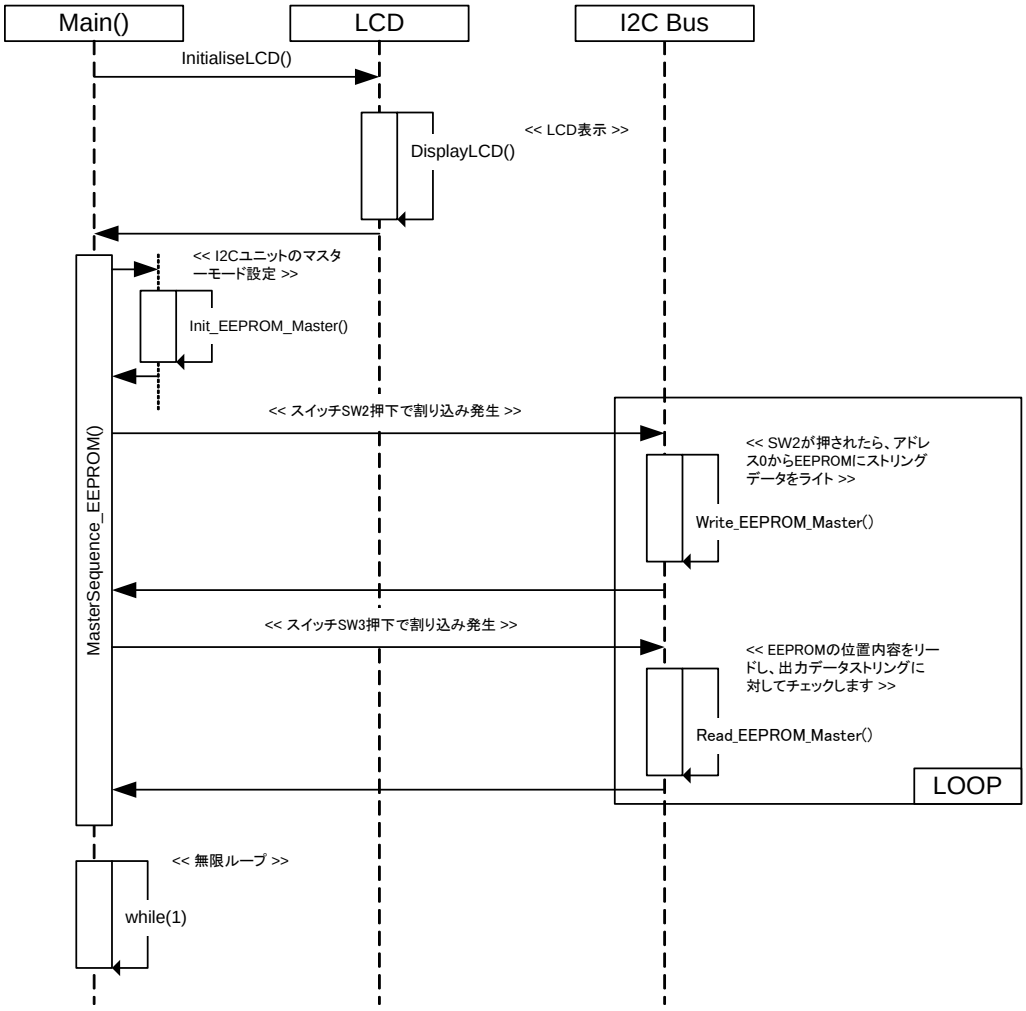


図 4-11: IIC_Master フロー

4.11.4 RPDL

IIC_Master で使用される関数、RPDL 関数を**表 4-11** に示します。

関数	RPDL 関数
Init_EEPROM_Master	R_IIC_Create
Write_EEPROM_Master	R_IIC_MasterSend
	R_CMT_CreateOneShot
Read_EEPROM_Master	R_IIC_MasterSend
	R_IIC_MasterReceive
	R_CMT_CreateOneShot

表 4-11: IIC_Master 用関数

4.12 IIC_Slave

4.12.1 説明

本サンプルコードはスレーブモードにおける IIC ユニットより 2k バイトの疑似 EEPROM メモリへのオペレーションをデモします。

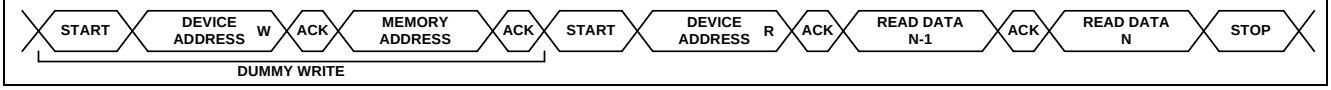
4.12.2 IICスレーブコマンド

(1) ライトオペレーション

- 疑似 EEPROM にライトするために、マスターはスタートコンディションの後に EEPROM デバイスアドレス（デフォルトアドレス：0xA0）を送り、スレーブからの ACK 信号を待ちます。
- マスターは 8 ビットの EEPROM メモリアドレス（0x0000 から 0x0800 まで有効）を送ることにより進行し、次にスレーブからの ACK 応答を待ちます。
- マスターは各データバイトのあとの ACK 応答と共に、バイトでライトータを送信し始めます。スレーブへのライトデータの最大バイト数は単一ライトオペレーション中で 16 です。
- 最終バイトが送られたら、マスターは処理を終了するストップ信号を送ります。
- 疑似 EEPROM の内部アドレスポインタはバイトライトごとに自動的にインクリメントされます。

(2) リードオペレーション

- アドレスが最大値に達するか、ストップコンディションが検出されまで、リードオペレーションは常に現在の内部疑似 EEPROM メモリポインタから開始し、次のバイトへ自動的にインクリメントされます。
- 任意のメモリ位置からリードするために、マスターは最初にダミーライトを要求された EEPROM メモリアドレスへ送ります。
- 現在の内部メモリ位置からリードするために、マスターは EEPROM デバイスアドレスの後にスタートコンディションを送ります。
- その後、EEPROM スレーブは ACK 信号で返答し、現在のメモリ位置にあるデータを送り、次のバイト位置へ内部ポインタを自動的にインクリメントします。
- 別のバイトを読むために、マスターは ACK 信号を送ります。必要なバイト数がリードされるまで、マスターはこの操作が繰り返され、ストップコンディションを含むオペレーションに終了します。
- リードオペレーションはアドレス範囲全体（0x0000 から 0x0800）からデータをリードするために使用することができ、ストップコンディションが検出されるか、最終メモリアドレスに到達するときに停止します。



4.12.3 オペレーション

1. サンプルコードは LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. その後、サンプルコードは IIC ユニットをスレーブモードに初期化し、疑似 EEPROM メモリ領域を準備します。
3. その後、BusMonitor_EEPROM_Slave 関数をコールします。この関数は IIC ユニットをポーリングし、データが IIC バス中のマスターデバイスから受信されるまで待ちます。
4. データが IIC バスによってマスターデバイスから受信されると BusReply_EEPROM 関数がコールされます。この関数は、マスターが有効なリード/ライト要求を送ったかどうかをチェックします。

5. 有効なライト要求が検出されると、BusReply_EEPROM 関数は Write_EEPROM_Slave 関数（疑似 EEPROM に受信データをライトする）をコールします。内部ポインタはマスターから受け取られたアドレスにセットされ、各バイトライトで自動的にインクリメントされます。
6. 有効なリード要求が検出されると、BusReply_EEPROM 関数は Read_EEPROM_Slave 関数をコールします。Read_EEPROM_Slave 関数は疑似メモリの最終アドレスに達するか、ストップコンディションが検出されるまでマスターデバイスへ疑似 EEPROM メモリの内容を送ります。
7. その後、サンプルコードは BusReply_EEPROMkannsuu に戻り、LCD にリード/ライトオペレーションの成功または失敗を表示させます。無効な要求がマスターから送られる場合、関数は LCD にエラーを表示させます。
8. その後、サンプルコードは BusMonitor_EEPROM_Slave 関数に戻り、IIC バス上の別のマスターコマンドを待ちます。

4.12.4 シーケンス

IIC_Slave サンプルのプログラム実行フローを**図 4-12** に示します。

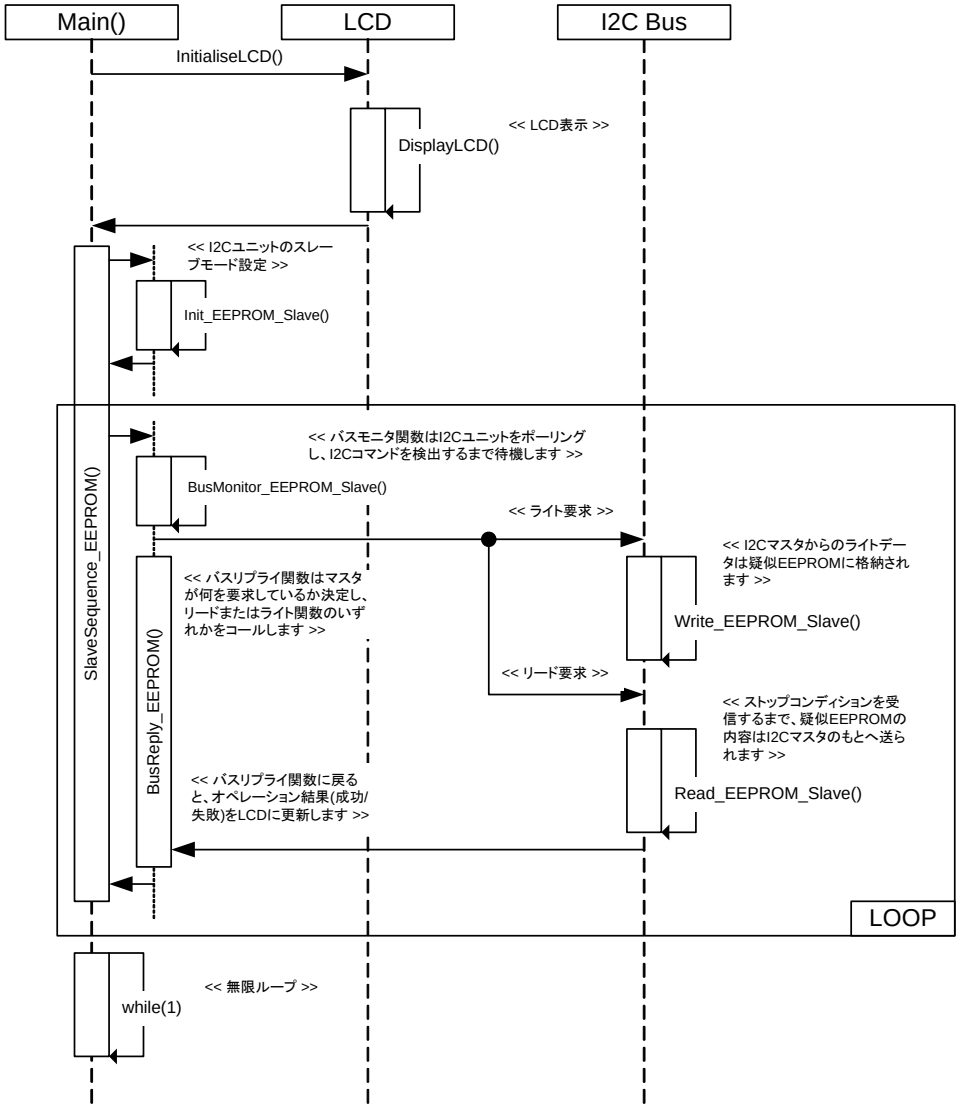


図 4-12: IIC_Slave フロー

4.12.5 RPDL

IIC_Slave で使用される関数、RPDL 関数を**表 4-12** に示します。

関数	RPDL 関数
Init_EEPROM_Slave	R_IIC_Create
Read_EEPROM_Slave	R_IIC_SlaveSend
BusMonitor_EEPROM_Slave	R_IIC_SlaveMonitor

表 4-12: IIC_Slave 用関数

4.13 CRC_Calc

4.13.1 説明

本サンプルコードはチェックサムを備えた SCI ターミナルから入力された文字の反復によって CRC ユニットをデモします。

4.13.2 オペレーション

1. サンプルコードを実行する前に、RS232 ケーブルを経由して PC とボードを接続し、ターミナルソフトを起動します（サンプルコードのコメント欄でインストラクションを確認できます）。
2. サンプルコードは LCD モジュールを初期化し、LCD の 1 行目に’CRC’、2 行目に’Calc’を表示します。
3. Init_CRC 関数をコールし、16bit の ANSI チェックサムを生成するための CRC ユニットと PC ターミナルと非同期通信するための SCI ユニットを設定します。
4. 関数は、さらにターミナルからデータを受信できるように割り込みを設定し、ターミナル画面にインストラクションを表示します。
5. その後、無限ループに入り割り込みを待ちます。
6. ターミナルにキー入力すると、SCI 割り込みはコールバック関数 CB_SCIReceive を実行します。この関数は受信した文字を取得し、チェックサムを生成するために Calculate_CRC 関数をコールします。
7. サンプルコードは Calculate_CRC 関数関数からコールバック関数まで戻り、ターミナルへの受信文字とチェックサムを含むストリングを書き込みます。
8. 無限ループに戻ると、再びキー入力があるまで待機します。

4.13.3 シーケンス

CRC_Calc サンプルのプログラム実行フローを**図 4-13** に示します。

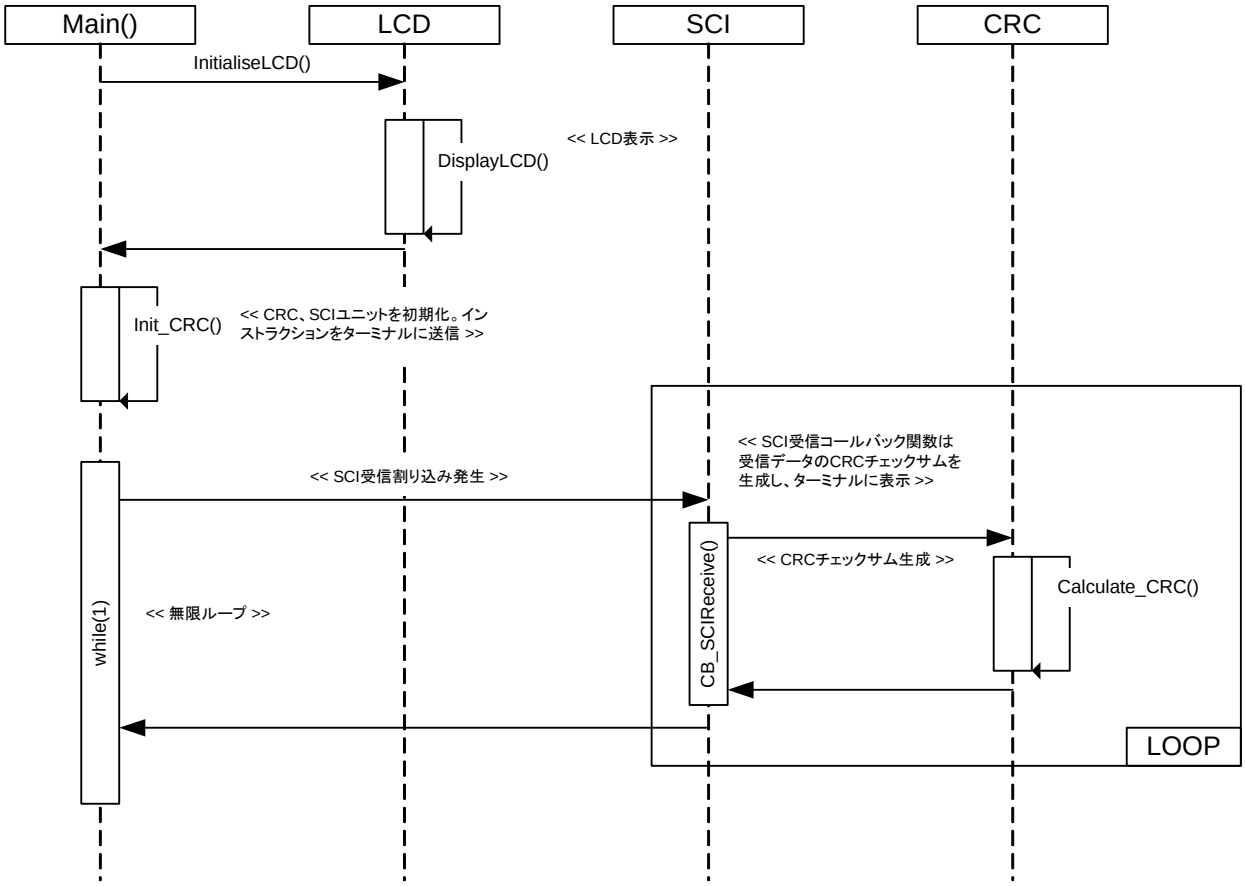


図 4-13: CRC_Calc フロー

4.13.4 RPDL

CRC_Calc で使用される関数、RPDL 関数を**表 4-13** に示します。

関数	RPDL 関数
Init_CRC	R_CRC_Create
	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
CB_SCIReceive	R_GetStatus
	R_SCI_Send
	R_SCI_Receive
Calculate_CRC	R_CRC_Write
	R_CRC_Read

表 4-13: CRC_Calc 用関数

4.14 Timer_Capture

4.14.1 説明

本サンプルコードはタイマによりスイッチ SW1 が押されている期間を LCD モジュールに表示します。スイッチが開放されると、LCD に millseconds 表示されます。

正確なスイッチの ON/OFF 推移を行うために、エッジ検出された後に 10ms の無反応期間を設けています。本サンプルコードはデモを目的としており、マイクロコントローラのタイマ精度を示すものではありません。

4.14.1 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に'Capture'、2 行目に'Push SW1'を表示します。
2. 1ms ごとの周期的な割り込みを発生されるために CMT タイマを設定します。CMT タイマは停止されており、カウント値は 0 にリセットされます。
3. その後、無限ループに入り割り込みを待ちます。
4. スイッチ SW1 が押されると CB_SwitchPress 関数をコールする割り込みが発生します。関数はカウント変数 usCapture_value をリセットし、CMT タイマを起動します。
5. スイッチ SW1 が押されている間、コールバック関数 CB_TimerClockTick は 1ms 間隔の CMT タイマ割り込みよって変数 usCapture_value をインクリメントします。
6. スイッチ SW1 が開放されると CB_SwitchRelease 関数をコールする割り込みが発生します。関数は CMT タイマを停止させ、変数 usCapture_value が 10 秒未満かどうかチェックする Update_TimerCapture 関数をコールします。10 秒未満の場合は LCD に時間を表示し、10 秒を越えていた場合は 10 秒を越えた時間であったことを示すメッセージを表示します。

4.14.3 シーケンス

Timer_Capture サンプルのプログラム実行フローを**図 4-14** に示します。

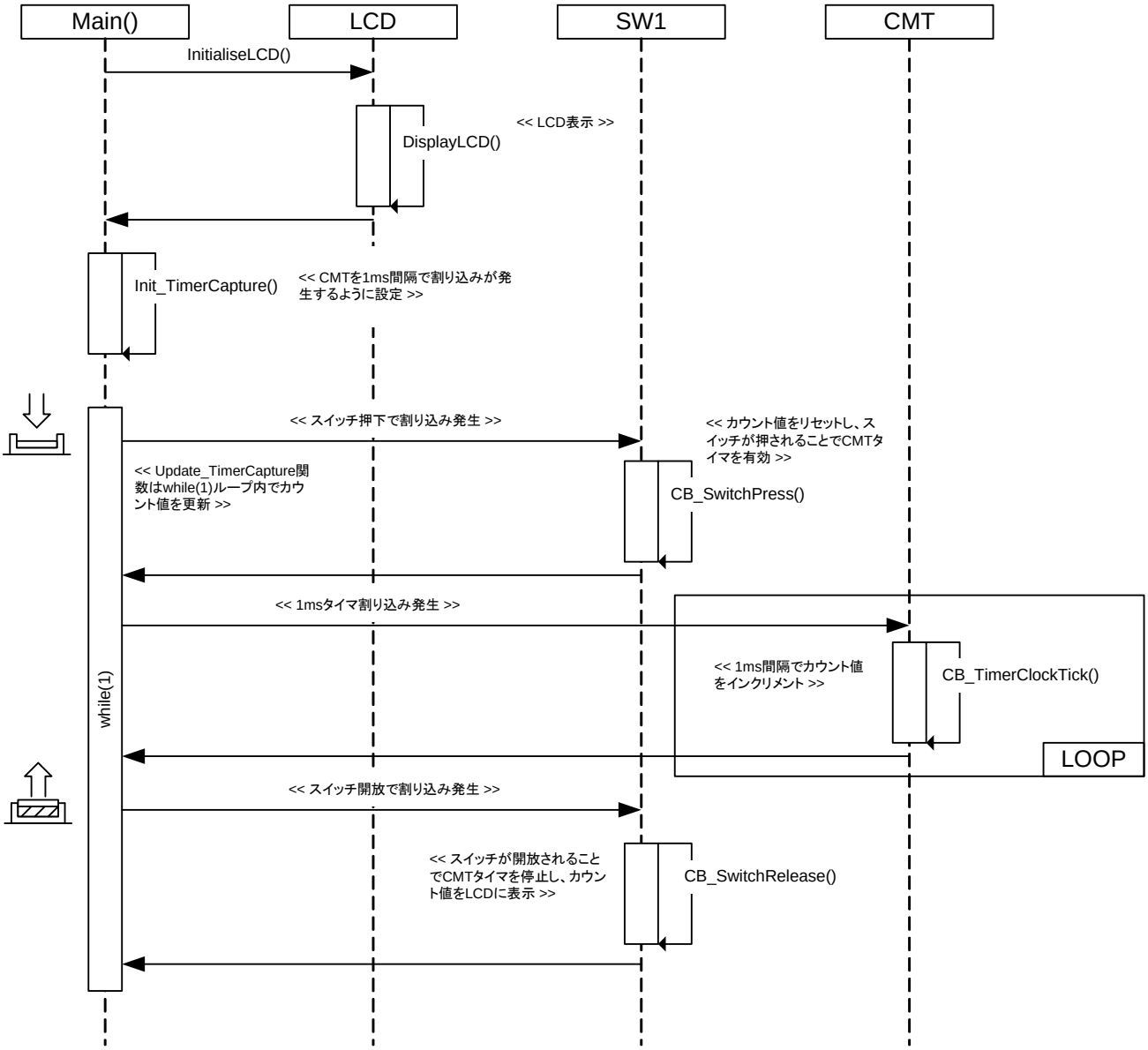


図 4-14: Timer_Capture フロー

4.14.4 RPDL

Timer_Capture で使用される関数、RPDL 関数を**表 4-14** に示します。

関数	RPDL 関数
Init_TimerCapture	R_CMT_Create
	R_CMT_Control
CB_SwitchPress	R_CMT_Control
CB_SwitchRelease	R_CMT_Control

表 4-14: Timer_Capture 用関数

4.15 Timer_Compare

4.15.1 説明

本サンプルコードは CMT タイマを設定し、コンペアマッチ割り込みが発生するたびにコールバック関数が実行されます。コールバック関数は LED をトグル出力します。

4.15.2 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に’Timer’、2 行目に’Compare’を表示します。
2. 100ms ごとの周期的な割り込みを発生させるために CMT タイマを設定します。
3. サンプルコードは無限ループに入り、100ms ごとの CMT タイマ割り込みによってコールバック関数 CB_CompareMatch がコールされます。
4. コールバック関数は LED をトグル出力します。

4.15.3 シーケンス

Timer_Compare サンプルのプログラム実行フローを**図 4-15** に示します。

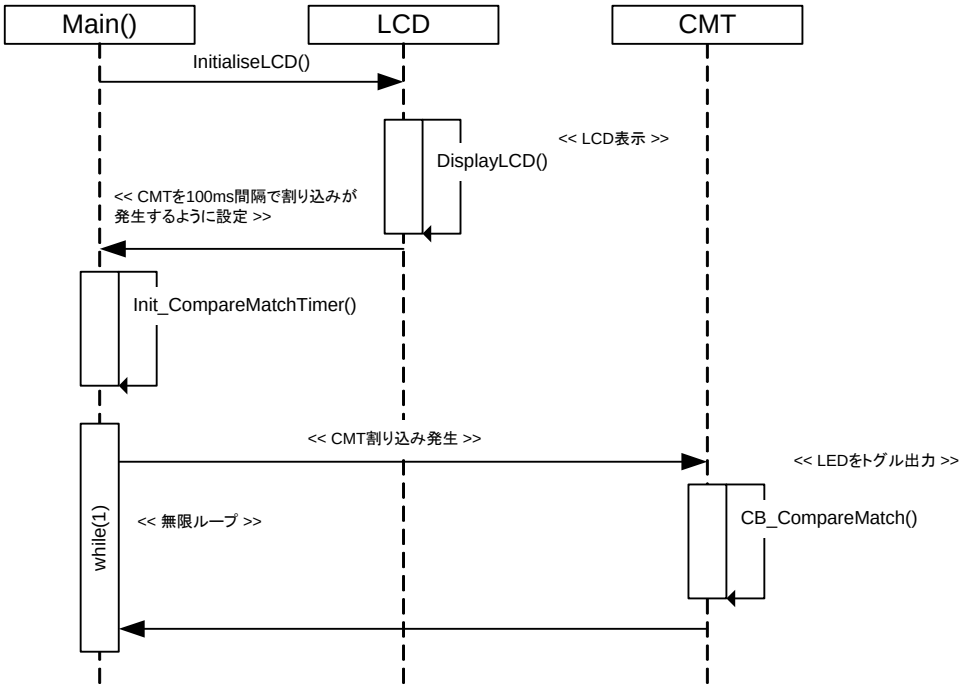


図 4-15: Timer_Compare フロー

4.15.4 RPDL

Timer_Compare で使用される関数、RPDL 関数を**表 4-15** に示します。

関数	RPDL 関数
Init_CompareMatchTimer	R_CMT_Create
CB_CompareMatch	R_IO_PORT_Modify

表 4-15: Timer_Compare 用関数

4.16 Timer_Event

4.16.1 説明

本サンプルコードはスイッチ SW2 を外部クロック入力源としたイベント機能のデモコードです。

4.16.2 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に'TR Event'、2 行目に'Push SW2'を表示します。
2. その後、サンプルコードはスイッチ SW2 立ち下りエッジを計測する MTU3 ユニットのチャネル 2 を形成し、LCD 上のカウント値を更新するために 100ms 周期で割り込みを生成する CMT タイマを形成します。
3. メインのループ処理に入ります。100ms 周期で CMT タイマ割り込みが発生し、CB_CompareMatch コールバック関数がコールされます。

4.16.3 シーケンス

Timer_Event サンプルのプログラム実行フローを**図 4-16** に示します。

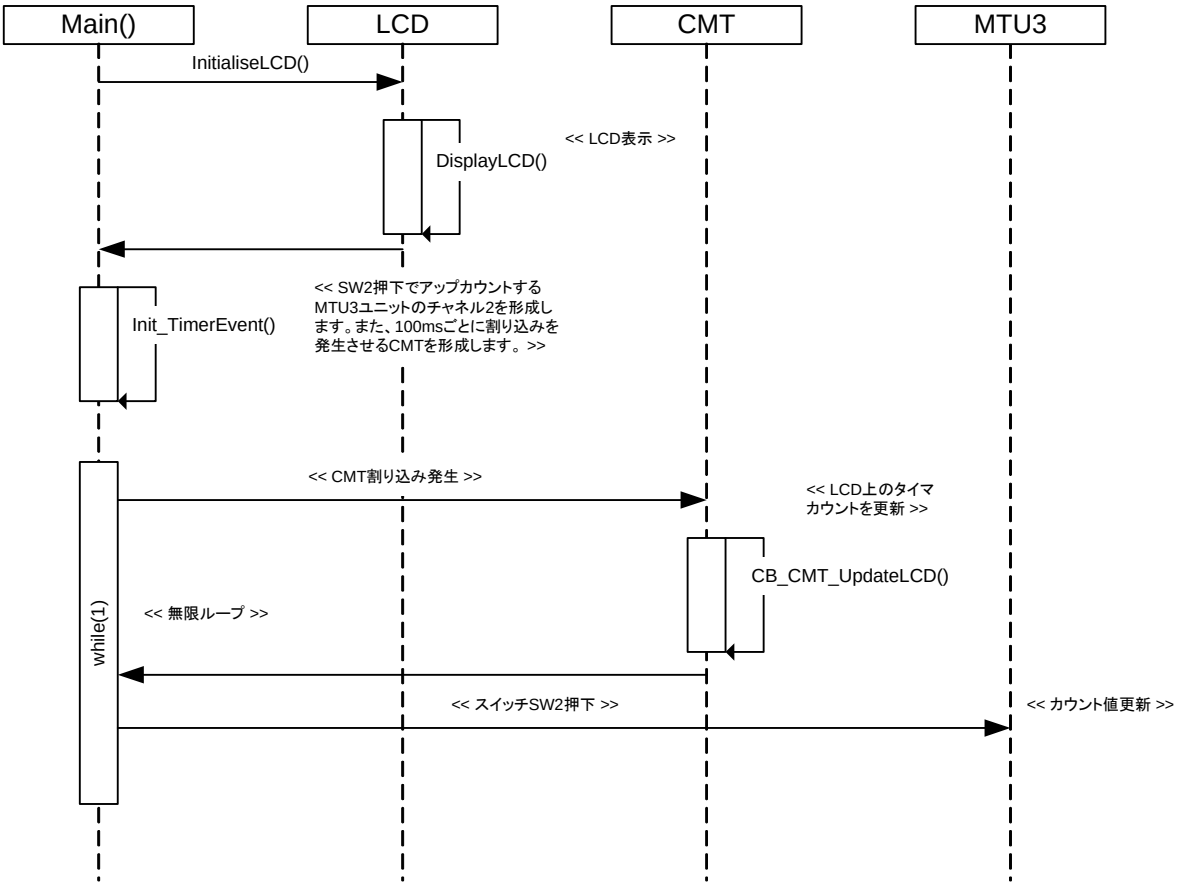


図 4-16: Timer_Event フロー

4.16.4 RPDL

Timer_Event で使用される関数、RPDL 関数を**表 4-16** に示します。

関数	RPDL 関数
Init_CompareMatchTimer	R_MTU3_Set
	R_MTU3_Create
	R_CMT_Create

表 4-16: Timer_Event 用関数

4.17 Timer_Mode

4.17.1 説明

本サンプルコードはコンペアマッチが発生するたびにタイマ出力ピンをトグル出力するよう MTU3 ユニットのチャネル 2 を設定します。

4.17.2 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に'1KHz'、2 行目に'J2-pin13'を表示します。
2. その後、1kHz の周期的な方形波を出力するために MTU3 ユニットのチャネル 2 を形成します。タイマはコンペアマッチのたびに出力ピン MTIOC2A をトグル出力します。
3. その後、無限ループに入ります。

4.17.3 シーケンス

Timer_Mode のプログラム実行フローを**図 4-17** 示します。

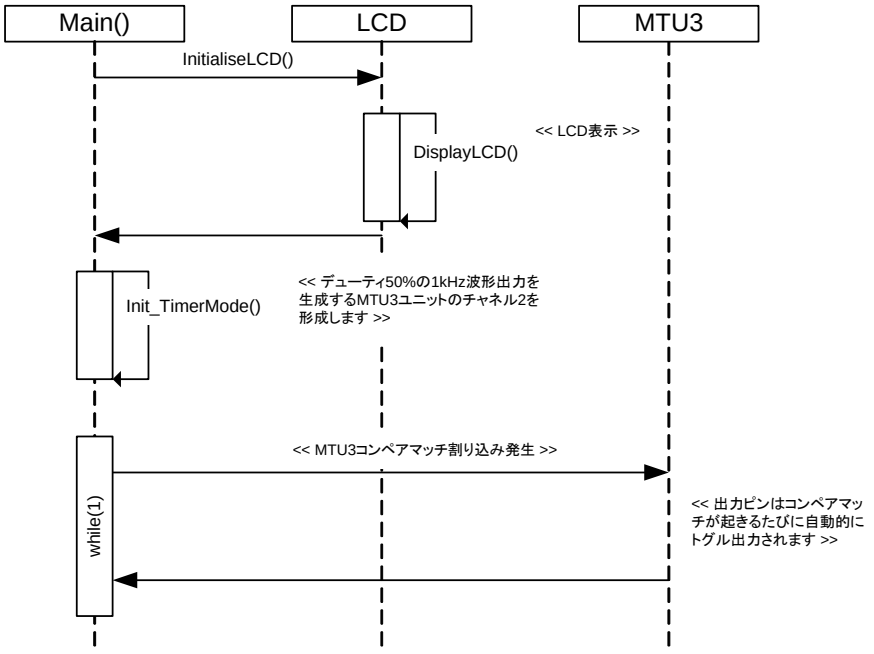


図 4-17Timer_Mode フロー

4.17.4 RPDL

Timer_Event で使用される関数、RPDL 関数を**表 4-17** 示します。

関数	RPDL 関数
Init_TimerMode	R_MTU3_Set
	R_MTU3_ControlChannel

表 4-17 Timer_Event 用関数

4.18 Flash_Data

4.18.1 説明

本サンプルコードはデータフラッシュを使ったデモコードです。スイッチが押されると AD 変換を行い、変換結果をデータフラッシュに書き込みます。

4.18.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。そして、Init_FlashData 関数をコールします。
2. 関数は FCU ユニットおよびデータフラッシュを設定します。データフラッシュにデータが書き込まれる前にフラッシュメモリを消去する関数がコールされます。また、サンプルコードは ADC ユニットおよびスイッチ割り込みも設定します。
3. サンプルコードは無限ループに入り、スイッチが押されるのを待ちます。
4. スイッチ SW1 が押されると、AD 変換を起動し変換結果をフラッシュに書き込みます。また、書き込まれた結果とアドレスを LCD に表示します。
5. 再びスイッチ SW1 が押されると、新しい AD 変換結果を次のフラッシュメモリアドレスに書き込みます。
6. スイッチ SW3 が押されると、データフラッシュブロックは消去され、書き込み先のアドレスはリセットされます。

4.18.3 シーケンス

Flash_Data サンプルのプログラム実行フローを**図 4-18** に示します。

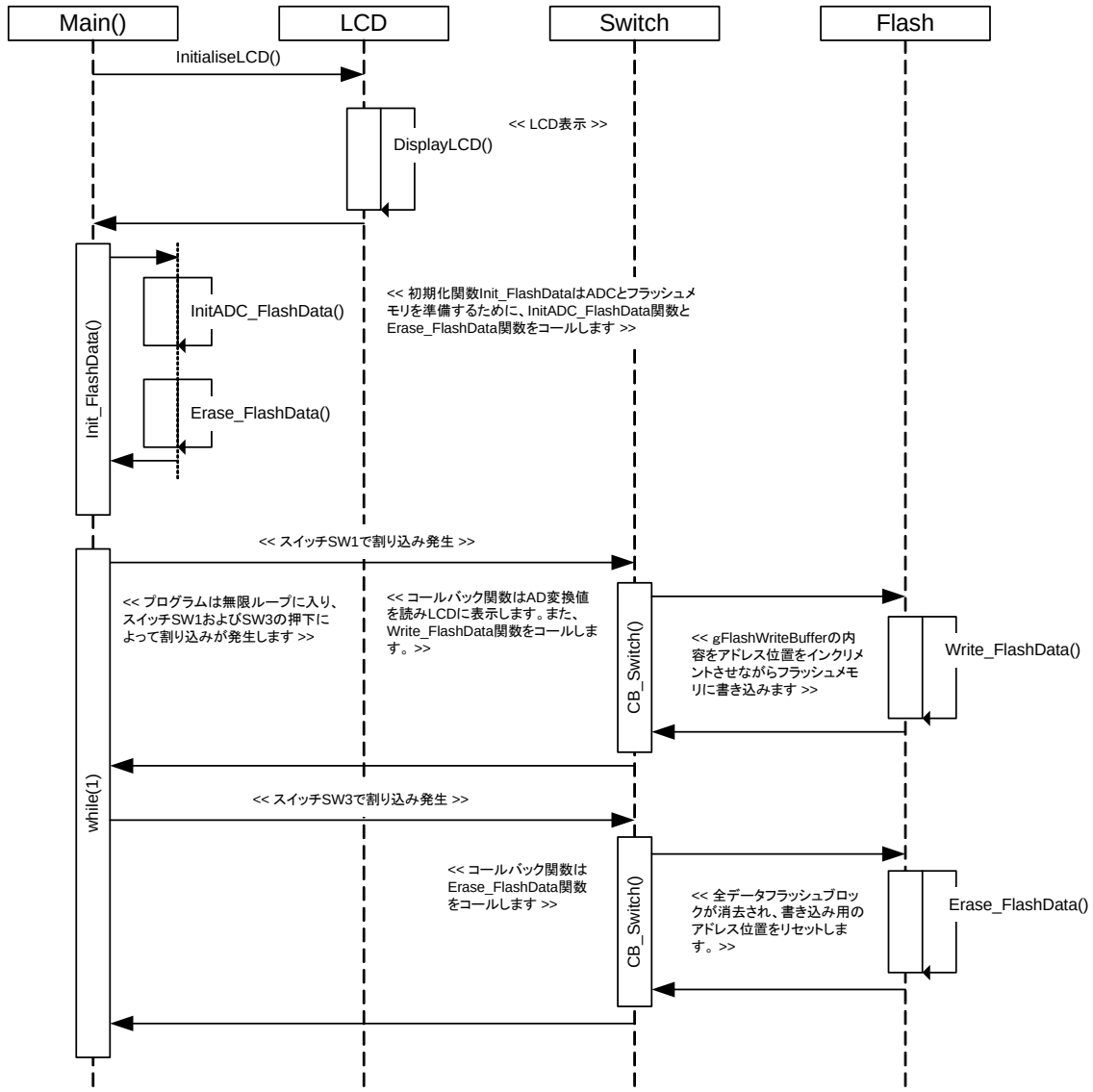


図 4-18: Flash_Data フロー

4.18.4 RPDL

Flash_Data で使用される関数、RPDL 関数を**表 4-18** に示します。

関数	RPDL 関数
InitADC_FlashData	R_ADC_10_Create
CB_Switch	R_ADC_10_Control
	R_ADC_10_Read

表 4-18: Flash_Data 用関数

本サンプルコードの詳細はアプリケーションノート Simple Flash API for RX（R01AN0544EU）を参照ください。

4.19 DTC

4.19.1 説明

本サンプルコードは DTC ユニットを使ったデモコードです。スイッチが押されると AD 変換を行い、変換結果を DTC によって転送します。

4.19.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。
2. Init_DTC 関数をコールし、AD 変換後に DTC 転送が起動するよう DTC ユニットおよび ADC ユニットを設定します。DTC 転送は AD データレジスタの内容をグローバル変数アレイ gDTC_Destination にインクリメントしながら転送する設定します。
3. サンプルコードは無限ループに入り、割り込みを待ちます。
4. スイッチ SW3 が押されると、コールバック関数 CB_Switch が実行されます。コールバック関数は残りの転送回数をチェックし、AD 変換を起動します。転送回数に残りがない場合、関数は gDTC_Destination の内容をクリアし、先頭から始められるよう DTC 転送を再設定します。

4.19.3 シーケンス

DTC サンプルのプログラム実行フローを**図 4-19** に示します。

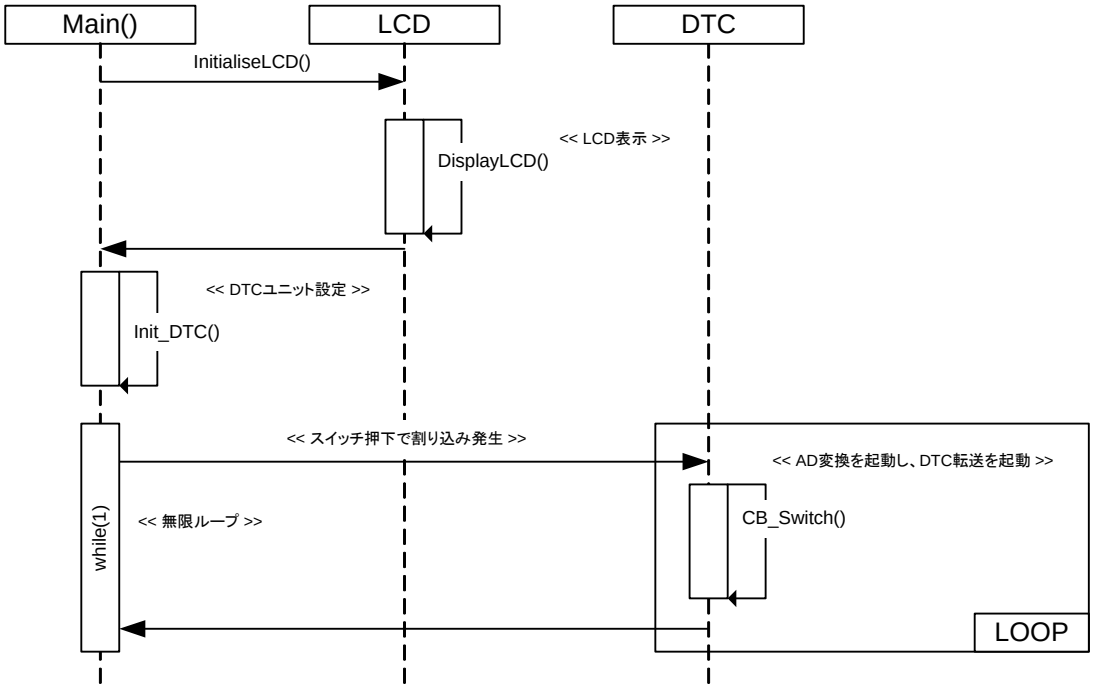


図 4-19: DTC フロー

4.19.4 RPDL

DTC で使用される関数、RPDL 関数を**表 4-19** に示します。

関数	RPDL 関数
Init_DTC	R_DTC_Set
	R_DTC_Create
	R_ADC_10_Create
	R_DTC_Control
CB_Switch	R_DTC_GetStatus
	R_DTC_Control
	R_INTC_Write
	R_ADC_10_Control

表 4-19: DTC 用関数

4.20 WDT

4.20.1 説明

本サンプルコードはタイマオーバーフロー割り込みおよびタイマをリセットするための周期的な CMT 割り込みによってウォッチドッグタイマをデモするコードです。CMT 割り込みの期間はポテンショメータ RV1 の調整により WDT オーバーフロー割り込みが発生するまで期間を短くすることができます。

4.20.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。
2. Init_WDT 関数をコールし、周期的な割り込みを発生させる CMT、WDT および ADC ユニットを設定します。ウォッチドッグタイマのオーバーフロー期間は～700ms に設定され、WDT がオーバーフローする場合に、コールバック関数 CB_WDTOverflow を実行する準備ができます。
3. サンプルコードは無限ループに入ります。タイマ期間が経過すると、コールバック関数 CB_CMTPeriod が実行されます。
4. コールバック関数は WDT カウントのリセット、LED のトグル出力および AD 変換の起動を行います。さらに、グローバル変数 gCMT_Period の値をタイマ期間として更新します。
5. AD 変換が完了すると、コールバック関数 CB_ADConversion が実行されます。関数は AD 変換結果をフェッチし、それを新しいタイマ期間を計算するために使用します。
6. タイマ期間が 700ms より大きな場合、ウォッチドッグタイマはオーバーフロー割り込みを発生させてオーバーフローします。
7. WDT オーバーフロー割り込みはコールバック関数 CB_WDTOverflow を実行します。この関数は LED を点灯状態にし、LCD に”Watchdog Overflow”を表示します。その後、関数は無限ループ内で待機します。

4.20.3 シーケンス

WDT サンプルのプログラム実行フローを**図 4-20** に示します。

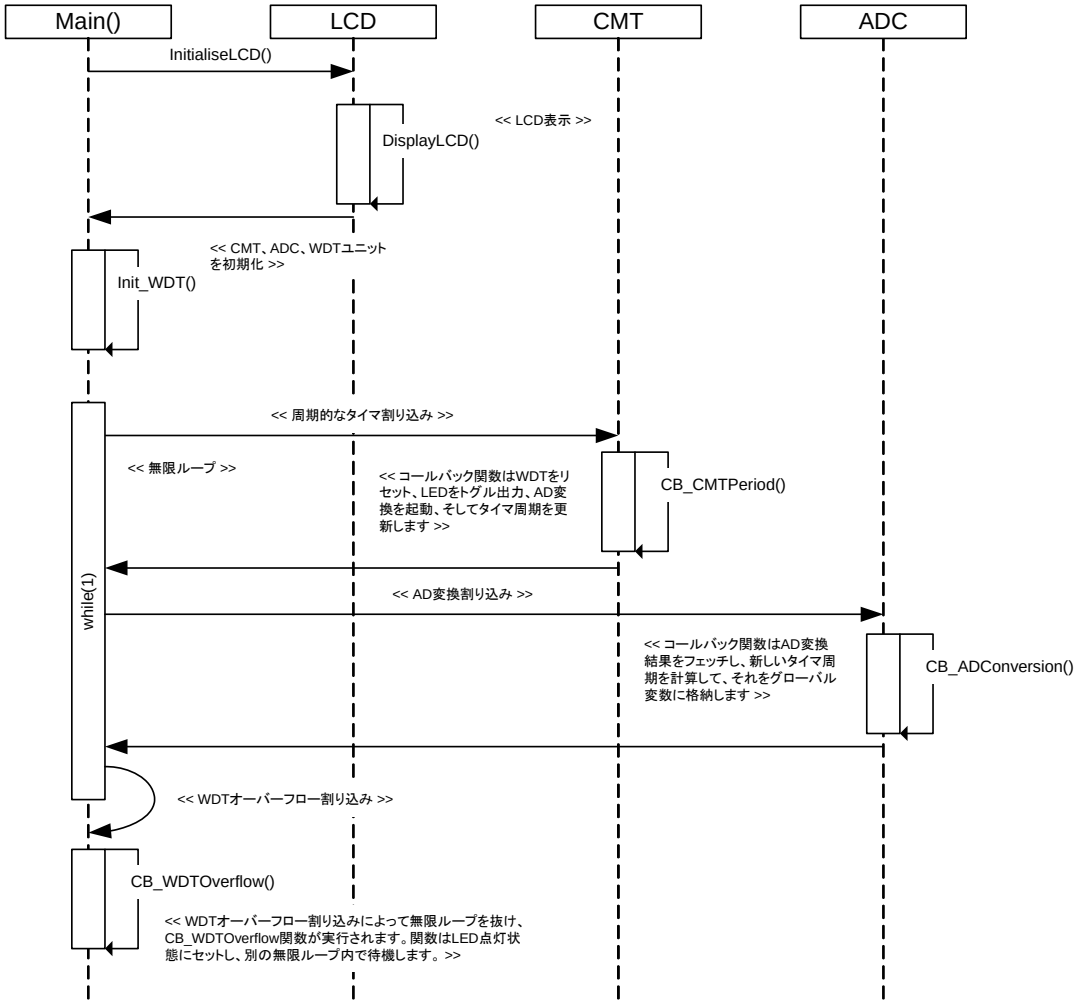


図 4-20: WDT フロー

4.20.4 RPDL

WDT で使用される関数、RPDL 関数を**表 4-20** に示します。

関数	RPDL 関数
Init_WDT	R_ADC_10_Create
	R_WDT_Create
	R_CMT_Create
CB_CMTPeriod	R_WDT_Control
	R_IO_PORT_Modify
	R_ADC_10_Control
	R_CMT_Control
CB_ADConversion	R_ADC_10_Read
CB_WDTOverflow	R_IO_PORT_Write

表 4-20: WDT 用関数

4.21 IWDT

4.21.1 説明

本サンプルコードはタイマアンダーフロー割り込みおよびタイマをリセットするための周期的な CMT 割り込みによって独立ウォッチドッグタイマをデモするコードです。CMT 割り込みの期間はポテンショメータ RV1 の調整により IWDT アンダーフロー割り込みが発生するまで期間を短くすることができます。

4.21.2 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクションを表示します。
2. その後、最後のリセットが IWDT アンダーフローによりプログラムを停止させるかどうかをチェックする Init_IWDT 関数をコールします。停止しなければ、周期的な割り込みを発生させる CMT、IWDT および ADC ユニットを設定します。ウォッチドッグタイマのアンダーフロー期間は～130ms に設定され、IWDT がアンダーフローする場合に、マイクロコントローラをリセットします。
3. サンプルコードは無限ループに入ります。タイマ期間が経過すると、コールバック関数 CB_CMTPeriod が実行されます。
4. コールバック関数は IWDT カウントのリセット、LED のトグル出力および AD 変換の起動を行います。さらに、グローバル変数 gCMT_Period の値をタイマ期間として更新します。
5. AD 変換が完了すると、コールバック関数 CB_ADConversion が実行されます。関数は AD 変換結果をフェッチし、それを新しいタイマ期間を計算するために使用します。
6. タイマ期間が 130ms より大きな場合、ウォッチドッグタイマはアンダーフロー割り込みを発生させます。
7. IWDT はデバイスをリセットし、すべての LED を点灯状態にし、LCD に”Watchdog Underflow”を表示します。その後、関数は無限ループ内で待機します。

4.21.3 シーケンス

IWDT サンプルのプログラム実行フローを図 **4-21** に示します。

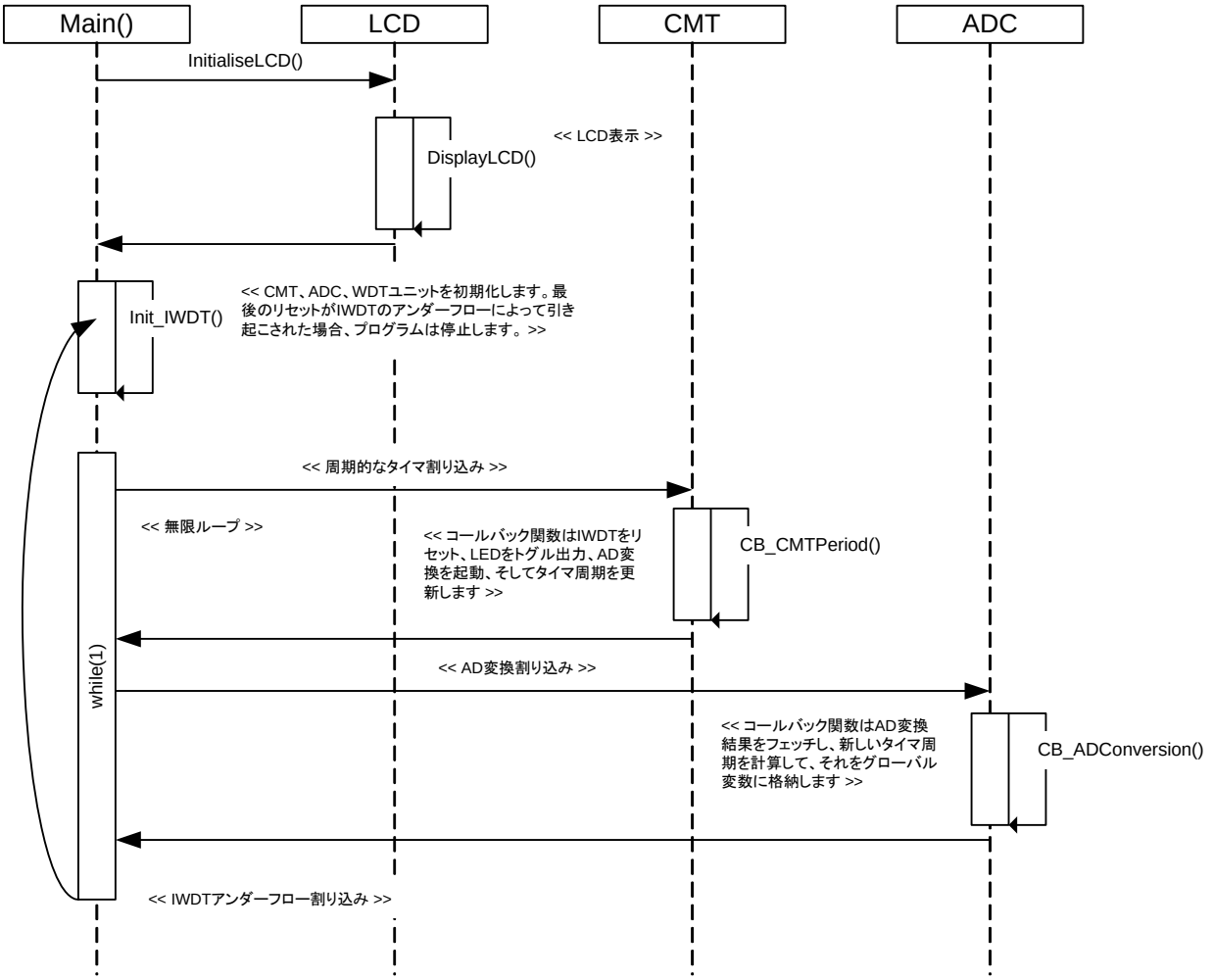


図 4-21: IWDT フロー

4.21.4 RPDL

IWDT で使用される関数、RPDL 関数を**表 4-21** に示します。

関数	RPDL 関数
Init_IWDT	R_ADC_10_Create
	R_IWDT_Set
	R_INTC_CreateExtInterrupt
	R_CMT_Create
CB_CMTPeriod	R_WDT_Control
	R_IO_PORT_Modify
	R_ADC_10_Control
	R_CMT_Control
CB_ADConversion	R_ADC_10_Read

表 4-21: IWDT 用関数

5 追加情報

サポート

High-performance Embedded Workshop の詳細情報は、CD またはウェブサイトに掲載のマニュアルを参照してください。

RX62T マイクロコントローラに関する詳細情報は、RX62T 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。

アセンブリ言語に関する詳細情報は、RX ファミリユーザーズマニュアルソフトウェア編を参照してください。

オンラインの技術サポート、情報等は以下のウェブサイトより入手可能です：

http://japan.renesas.com/rskrx62t （日本サイト）
http://www.renesas.com/rskrx62t （グローバルサイト）

オンライン技術サポート

技術関連の問合せは、以下を通じてお願いいたします。

アメリカ： techsupport.america@renesas.com
ヨーロッパ： tools.support.eu@renesas.com
日本： csc@renesas.com

ルネサスのマイクロコントローラに関する総合情報は、以下のウェブサイトより入手可能です：

http://japan.renesas.com/ （日本サイト）
http://www.renesas.com/ （グローバルサイト）

商標

本書で使用する商標名または製品名は、各々の企業、組織の商標または登録商標です。

著作権

改訂記録	RSKRX62T ソフトウェアマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.12.03	ー	初版発行
2.00	2011.05.26	19, 21	表 4-3、4-4 の RPDL 関数名を修正。 RPDL 関数 R_ADC_12_Create を R_ADC_12_CreateUnit に修正
		35	図 4-12 のシーケンス説明を修正。 誤：リード要求→Write_EEPROM_Slave 関数 ：ライト要求→Read_EEPROM_Slave 関数 ↓ 正：ライト要求→Write_EEPROM_Slave 関数 ：リード要求→Read_EEPROM_Slave 関数
		45	表 4-18 下の Flash API アプリケーションノートのドキュメント番号を変更。 REU05B0131 から R01AN0544EU に変更
2.01	2011.07.01	ー	社名修正

RSKRX62T ソフトウェアマニュアル

発行年月日　2011 年 7 月 1 日　Rev.2.01

発行　　株式会社ルネサスソリューションズ
　　〒532-0003　大阪府大阪市淀川区宮原 4-1-6

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

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

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）　(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

RX62T グループ

RJJ10J2791-0201

ルネサス エレクトロニクス株式会社