User's Manual

32

RX210 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／RX200 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.00 2012.6

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、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。
2. 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。
3. 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に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
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。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。
9. 本資料に記載されている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。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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。

(2012. 4)

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。

RSKRX210 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	RSKRX210 ユーザーズマニュアル	R20UT0302JG
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	RSKRX210 ソフトウェアマニュアル	R20UT0305JG (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	RSKRX210 チュートリアル	R20UT0303JG
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	RSKRX210 クイックスタートガイド	R20UT0304JG
回路図	CPU ボードの回路図	RSKRX210 CPU ボード回路図	R20UT0301JG
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	RX210 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

1.2 関数名について

本マニュアル中の関数名と実際のサンプルコード中の関数名が一部異なる場合がありますが、本マニュアルで記載されている内容と機能的違いはございません。

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。

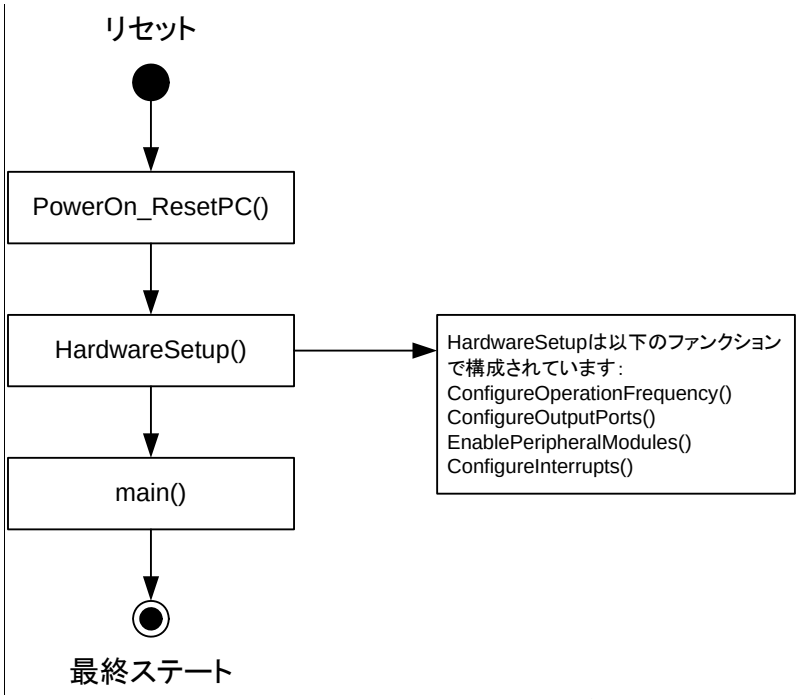

図 2-1: サンプルコードの基本構成

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 R_CMT_CreateOneShot R_CGC_Control
ConfigureOutputPorts	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	サンプルコードで使用する外部割込みを設定します。	R_INTC_CreateExtInterrupt R_INTC_CreateFastInterrupt

表 2-1: HardwareSetup 関数

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。

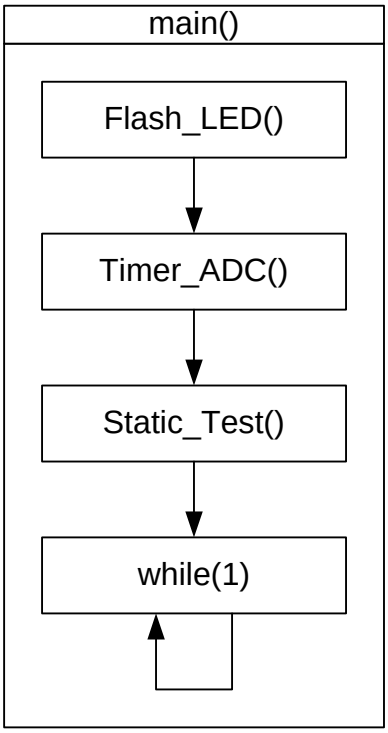

図 3-1: Tutorial フロー

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 Flash_LED 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タイマ割り込みを作り出します。スイッチが押されるか、LED が 200 回トグル出力されるまで LED のトグル出力が繰り返されます。

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 Timer_ADC 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_ADConversion 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_ADConversion 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 Timer_ADC をコールし、タイマおよび ADC 割り込みのセットアップ後、ステップ④のコールバック関数と並行して動作する Static_Test 関数をコールします。Static_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、表示内容は初期表示に戻ります。

⑥ その後、コードは無限ループ処理に入りステップ④の周期的割り込みを待ちます。

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。

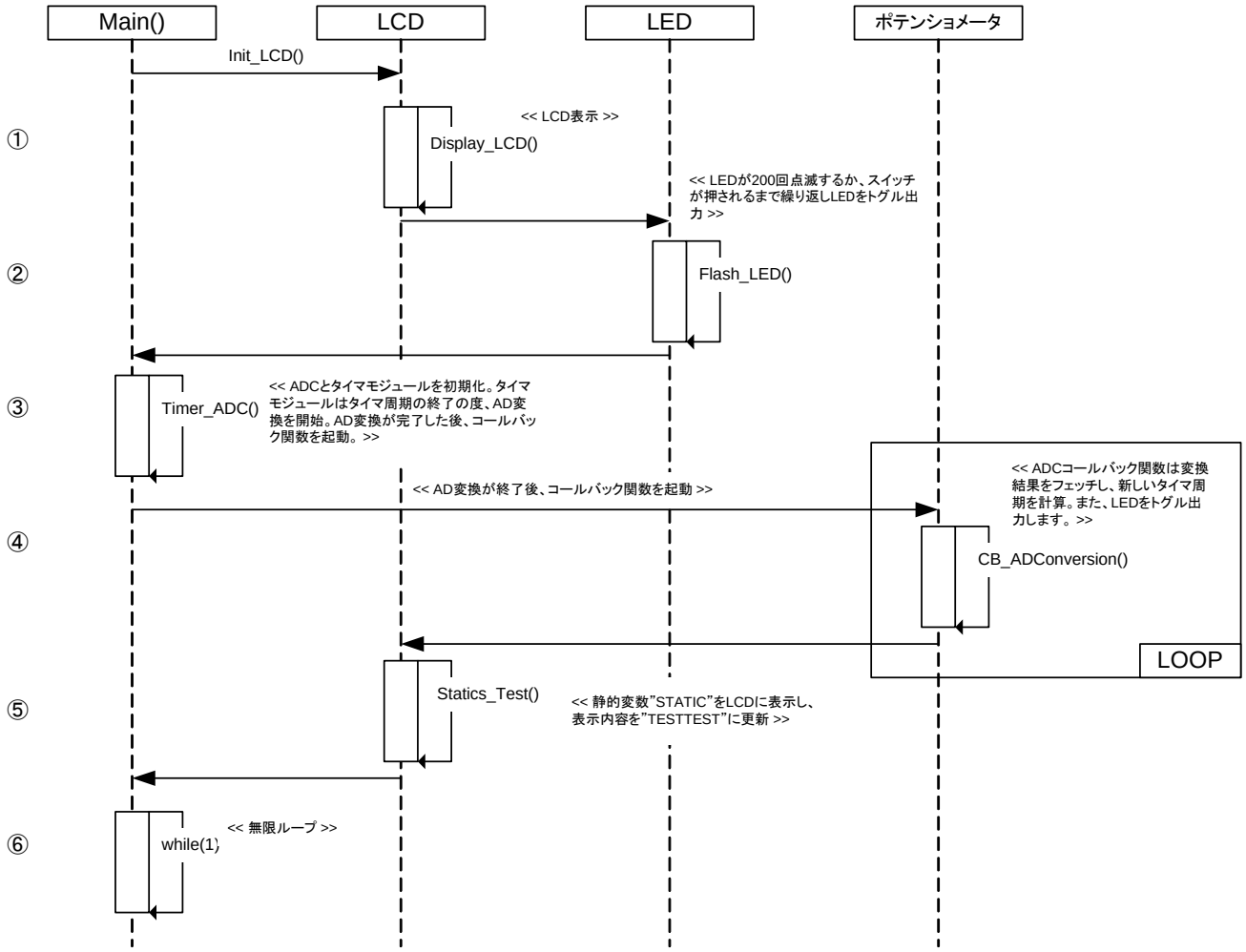

図 3-2: Tutorial フロー

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
Flash_LED	R_CMT_Create
	R_CMT_Destroy
Start_Timer	R_CMT_Create
Start_ADC	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_ADC_12_Control
CB_ADConversion	R_ADC_12_Read
	R_CMT_Control

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 ADC12_Repeat

本サンプルコードはリピートモード（連続スキャンモード）による 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。

4.1.1 オペレーション

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② リピートモード（スキャンモード）用の ADC モジュールを初期化するために Init_ADCXRepeat 関数をコールします。また、関数はインターバルタイマ割り込み（コールバック関数 CB_Timer_ADCXRepeat）を初期化します。
③ コードは無限ループ内でタイマ割り込み発生を待ちます。CB_Timer_ADCXRepeat 関数は現在の AD 変換結果をフェッチし、LCD に変換結果を表示します。

4.1.2 シーケンス

ADC12_Repeat サンプルのプログラム実行フローを**図 4-1** に示します。

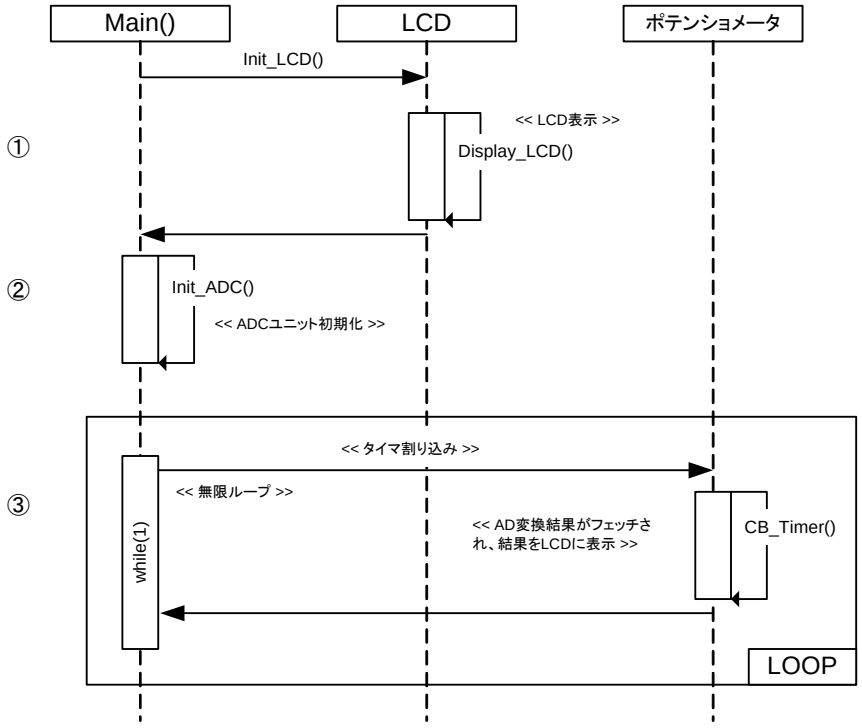

図 4-1: ADC12_Repeat フロー

4.1.3 RPDL

ADC12_Repeat 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_ADC12Repeat	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_CMT_Create
	R_ADC_12_Control
CB_Timer_ADC12Repeat	R_ADC_12_Read

表 4-1: ADC12_Repeat 用関数

4.2 Async_Serial

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。

4.2.1 オペレーション

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（サンプルコードのヘッダ欄でインストラクションを確認できます）。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_Async 関数をコールし、非同期モード用に SCI ユニットを初期化します。関数はさらに SCI ユニットがデータを受信するたびに割り込みを生成する SCI 受信割り込みを形成します。最後に、タイマユニットを形成します。これはコールバック関数 CB_Timer_Async をコールするために使用されて周期的な割り込みを生成します。
③ コードは無限ループ内でタイマ割り込み発生を待ちます。コールバック関数 CB_Timer_Async はタイマ割り込みによってコールされ、gSCI_Flag のステータスをチェックします。フラグが真の時、0 から 9 のインクリメント繰り返す数字を送信します。フラグが偽の時、送信を行わずに関数を抜けます。また、送信中は LED0 が点灯し続け、LED1 は消灯します。
④ SCI ユニットがデータを受信した場合、コールバック関数 CB_SCIReceive_Async がコールされます（ユーザがターミナルのプログラムに文字を入力する場合、コールバック関数がコールされます）。関数はキーボードの入力をフェッチします。文字が”z”の場合、gSCI_Flag 変数を偽にセットします。その他入力はフラグに真がセットされます。一旦”z”が押されたら LED1 が点灯し、送信を再開するために別のキーが押されるまで LED0 は消灯し続けます。

4.2.2 シーケンス

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

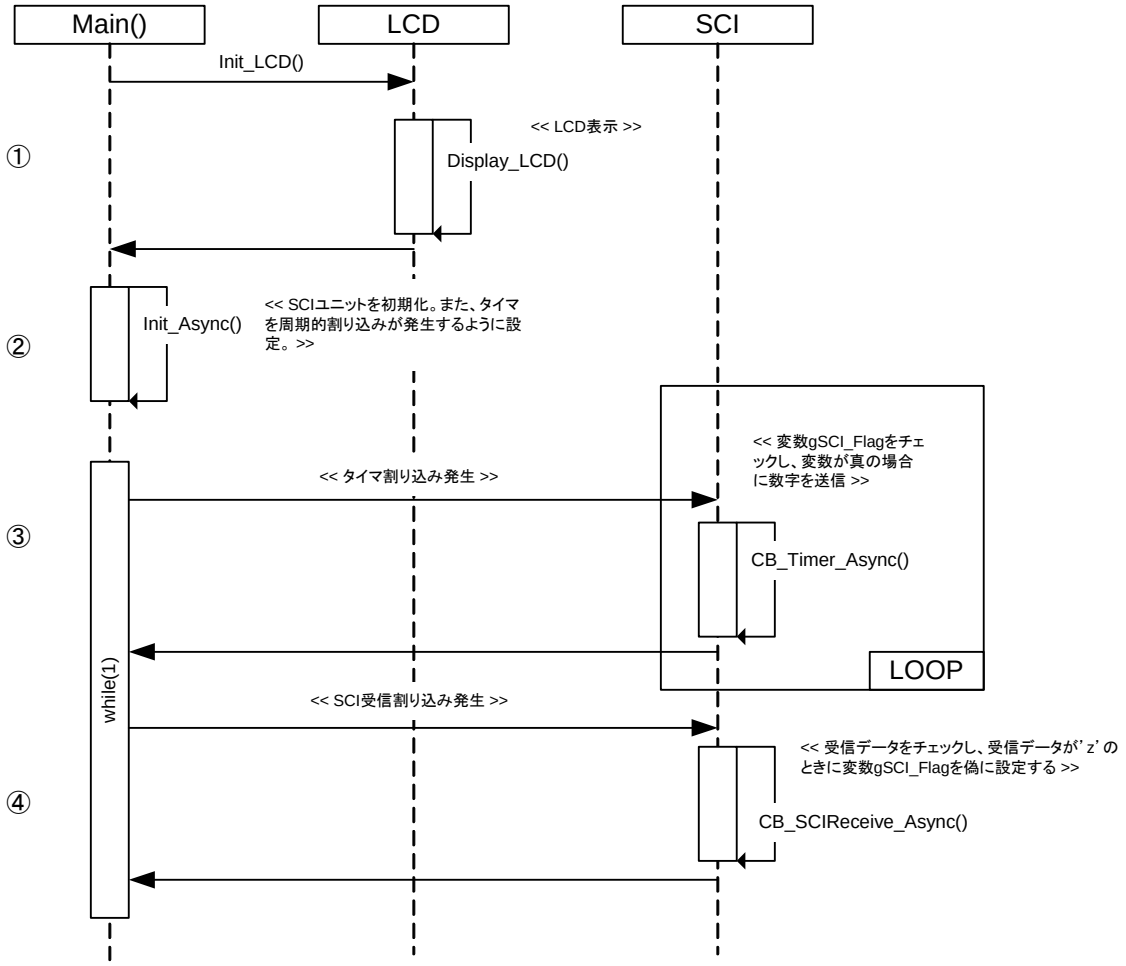

図 4-2: Async_Serial フロー

4.2.3 RPDL

Async_Serial 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_Async	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
	R_CMT_Create
Transmit_Async	R_SCI_Send
	R_IO_PORT_Write
CB_Timer_Async	R_SCI_GetStatus
CB_SCIReceive_Async	R_IO_PORT_Write
	R_SCI_Receive

表 4-2: Async_Serial 用関数

4.3 CRC

本サンプルコードは任意のキーボードからの入力のチェックサム計算を実行する CRC ユニットのデモコードです。

4.3.1 オペレーション

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（サンプルコードのヘッダ欄でインストラクションを確認できます）。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_CRC 関数をコールし、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。
③ コードは無限ループ内で SCI 受信割り込み発生を待ちます。キーボードからの入力ごとにコールバック関数 CB_SCIRecieve がコールされます。関数は文字データをフェッチして 16 進数の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。
④ Calculate_CRC 関数は受信文字を CRC レジスタに入れ、計算されたチェックサムをフェッチし、CB_SCIReceive_CRC 関数に値を返します。
⑤ コードは Calculate_CRC 関数からコールバック関数まで戻り、ターミナルへ受信文字、チェックサムを含んだストリングを送信します。その後、無限ループに戻り、再びキーボード入力を待ちます。

4.3.2 シーケンス

CRC サンプルのプログラム実行フローを**図 4-3** に示します。

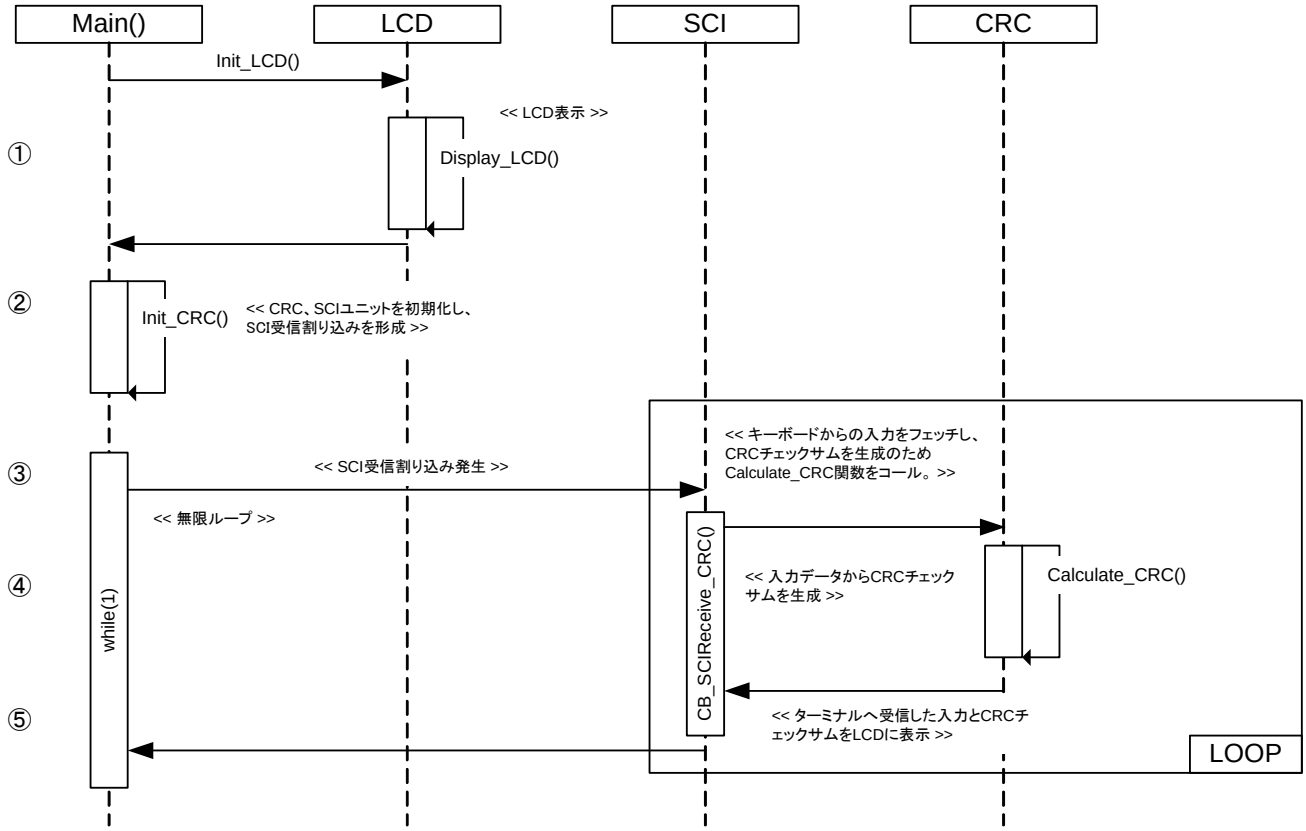

図 4-3: CRC_Calc フロー

4.3.3 RPDL

CRC 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_CRC	R_CRC_Create
	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
CB_SCIReceive_CRC	R_SCI_GetStatus
	R_SCI_Send
	R_SCI_Receive
Calculate_CRC	R_CRC_Write
	R_CRC_Read

表 4-3: CRC_Calc 用関数

4.4 DMAC

本サンプルコードは DMAC のデモコードです。DMAC ユニットを形成し、グローバル変数 gDMA_DataBuff へデータ転送を行います。

4.4.1 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_DMAC 関数をコールし、連続するデータ転送用の DMAC チャネルを形成します。転送モードは自動的に各転送後に終点アドレスをインクリメントします。コールバック関数 CB_DMACTransferEnd_DMAC はすべての転送が完了するとコールされます。コードが無限ループにエントリする前に DMAC は有効になり転送オペレーションが開始されます。
③ すべてのデータ転送が完了すると、コールバック関数 CB_DMACTransferEnd_DMAC がコールされ、LED1 を点灯させて転送オペレーションが終了したことを示します。

4.4.2 シーケンス

DMAC サンプルのプログラム実行フローを**図 4-4** に示します。

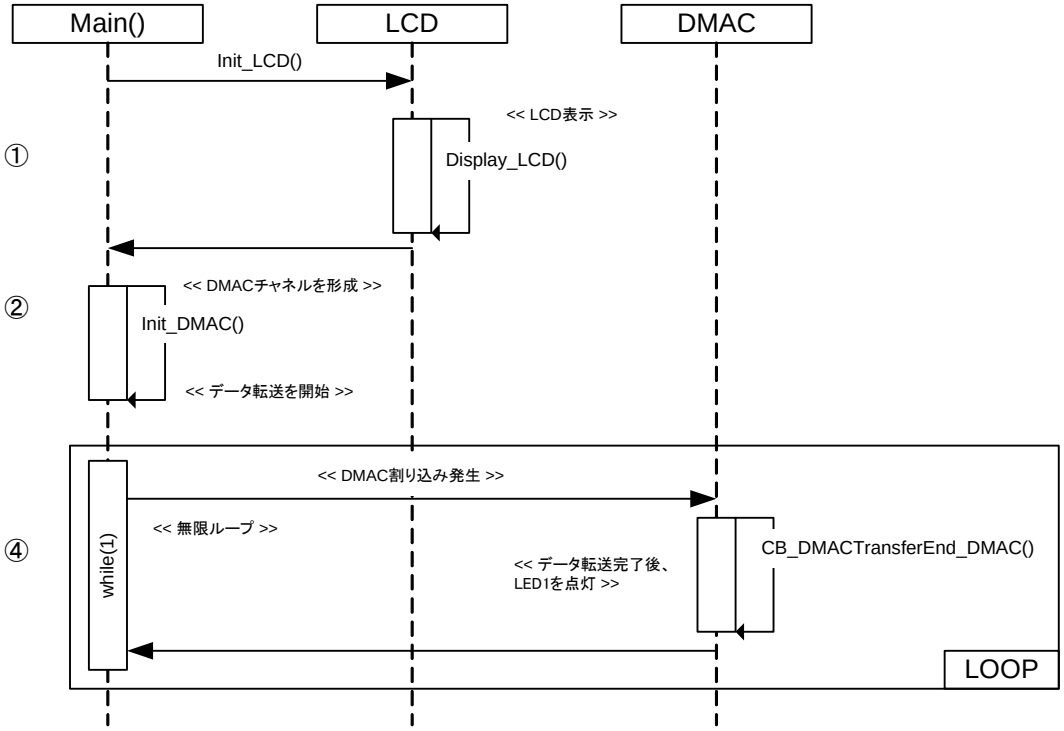

図 4-4: DMAC フロー

4.4.3 RPDL

DMAC 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_DMAC	R_DMAC_Create
	R_DMAC_Control
	R_INTC_Modify
CB_DMACTransferEnd_DMAC	R_DMAC_GetStatus
	R_IO_PORT_Write
	R_DMAC_Control

表 4-4: DMAC 用関数

4.5 DTC

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。

4.5.1 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 ユニットを形成します。DTC 転送は AD データレジスタの内容をグローバル変数配列 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。スイッチ SW3 が押されると、コールバック関数 CB_Switch_DTC がコールされます。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。転送回数の残りがない場合、関数は gDTC_Destination の内容をクリアし、転送先アドレスを先頭に戻すよう DTC 転送を再設定します。

4.5.2 シーケンス

DTC サンプルのプログラム実行フローを**図 4-5** に示します。

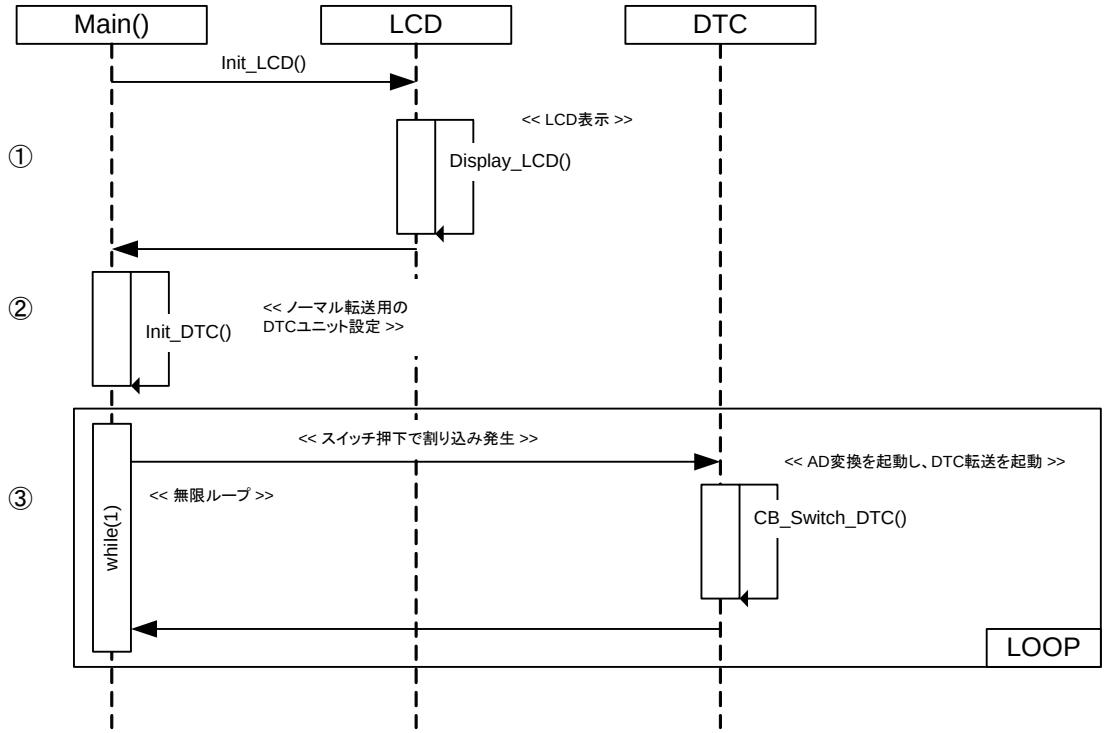

図 4-5: DTC フロー

4.5.3 RPDL

DTC 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_DTC	R_DTC_Set
	R_DTC_Create
	R_DTC_Control
	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_INTC_Write
	R_INTC_Modify
CB_Switch_DTC	R_DTC_GetStatus
	R_DTC_Control
	R_INTC_Write
	R_ADC_12_Control
	R_IO_PORT_Modify

表 4-5: DTC 用関数

4.6 LVD

本サンプルコードは LVD（低電圧検出回路）のデモコードです。電圧検出レベル以下に低下する場合、割り込みを発生させます。

4.6.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄のインストラクションに従ってボードを設定してください。ボードの PWR コネクタへ電圧を可変できる外部電源を接続し、初期電圧 5V に設定してください。

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_VDET 関数をコールし、VCC 電源が 4.0V および 1.9V まで低下する場合に割り込みを生成できるように LVD ユニットを初期化します。また、関数は周期的にボード上の LED をトグル出力するために CMT タイマを形成します。
③ サンプルコードは無限ループに入ります。周期的な割り込みによってコールバック関数 CB_Timer_VDET がコールされます。グローバル変数 gLEDsync_flg がセットされる場合、LED0～LED2 を同期してトグルし（LED3 は非同期でトグル）、gLEDsync_flg がセットされていない場合、LED0～LED3 すべてがトグルされます。
④ 入力電圧が 4.0V 未満に低下する場合、電圧検出割り込みによってコールバック関数 CB_LVD2_VDET がコールされます。関数は gLEDsync_flg をセットし、LED0～LED2 を消灯させ、電圧が 4.0V 以上に戻るまで待機します。
⑤ 入力電圧が 1.9V 未満に低下する場合、ノンマスカブルの電圧検出割り込みによってコールバック関数 CB_NMI_VDET がコールされます。これは他のコールバック関数が動作していても割り込まれます。関数は LED0～LED3 すべてを消灯させ、電圧が 1.9V 以上に戻るまで待機します。

4.6.2 シーケンス

LVD サンプルのプログラム実行フローを**図 4-6** に示します。

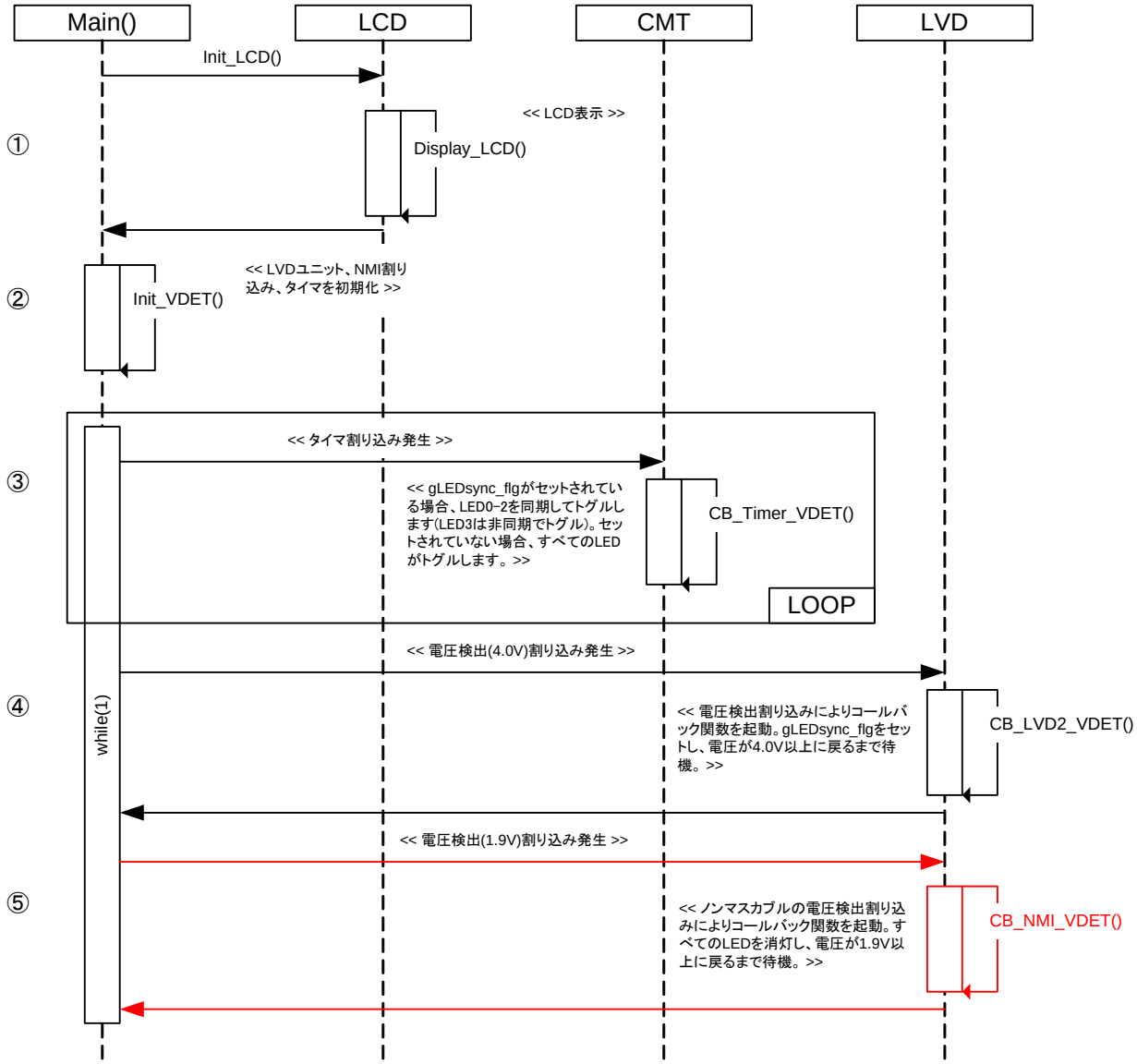

図 4-6: LVD フロー

4.6.3 RPDL

LVD 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_VDET	R_LVD_Create
	R_INTC_CreateExtInterrupt
	R_CMT_Create
CB_Timer_VDET	R_IO_PORT_Modify
CB_LVD2_VDET	R_CMT_Control
	R_IO_PORT_Write
	R_LVD_GetStatus
	R_INTC_Write
CB_NMI_VDET	R_IO_PORT_Write
	R_LVD_GetStatus
	R_INTC_ControlExtInterrupt

表 4-6: LVD 用関数

4.7 Power_Down

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、リアルタイムクロックによりモード遷移を行います。

4.7.1 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'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、低費消費電力とリアルタイムクロックに関連するレジスタを設定します。
③ サンプルコードは無限ループに入り、直ちに MCU はスタンバイモードに切り替わります。1 秒後、RTC ユニットはスタンバイモードから MCU を復帰させ、コールバック関数 CB_RTC_PowerDown をコールします。
④ CB_RTC_PowerDown 関数は RTC ユニットから時間を読み、現在の時間（プログラム開始からの経過時間）を表示している LCD の表示を更新します。その後、プログラムは無限ループに戻り、ステップ③と④を繰り返します。

4.7.2 シーケンス

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

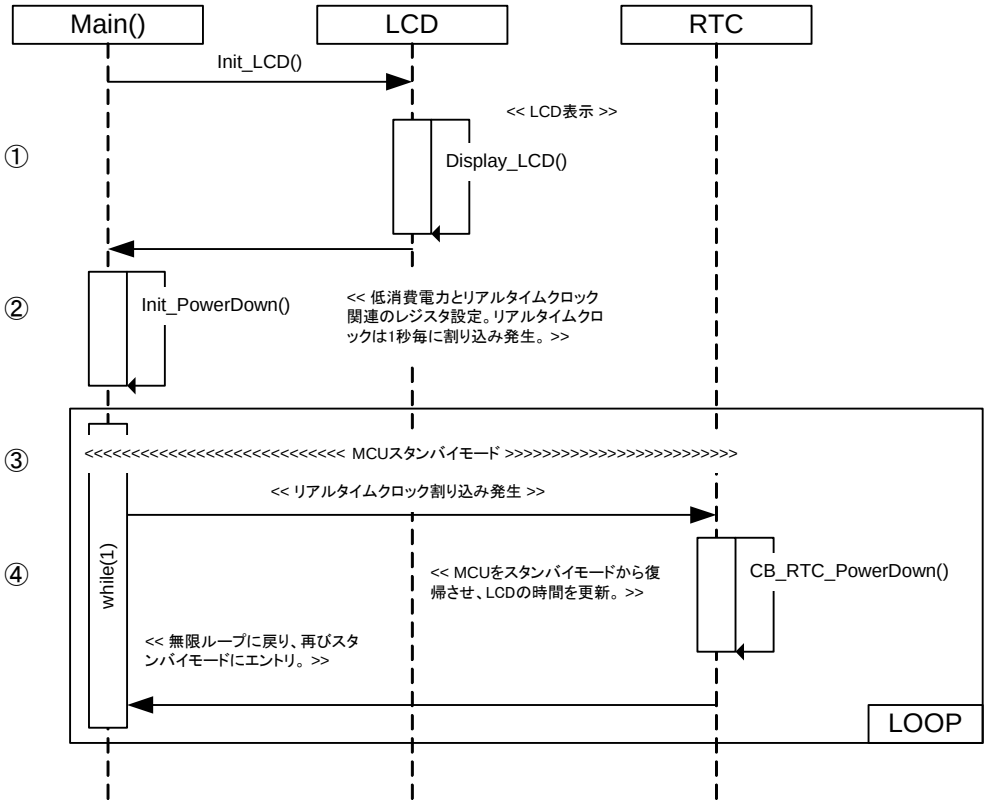

図 4-7: Power_Down フロー

4.7.3 RPDL

Power_Down 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_PowerDown	R_RTC_Create
	R_RTC_Control
	R_LPC_Create
CB_RTC_PowerDown	R_RTC_Read
	R_CMT_CreateOneShot

表 4-7: Power_Down 用関数

4.8 Timer_Mode

本サンプルコードはタイマによって 1KHz の波形を出力します。

4.8.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄のインストラクションに従ってタイマ出力ピンにオシロスコープを接続してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② 1KHz 周期、デューティ比 50%のタイマ出力をするためにタイマチャネルを形成します。
③ サンプルコードは無限ループに入り、タイマチャネルはタイマ出力し続けます。

4.8.2 シーケンス

Timer_Mode のプログラム実行フローを**図 4-8** 示します。

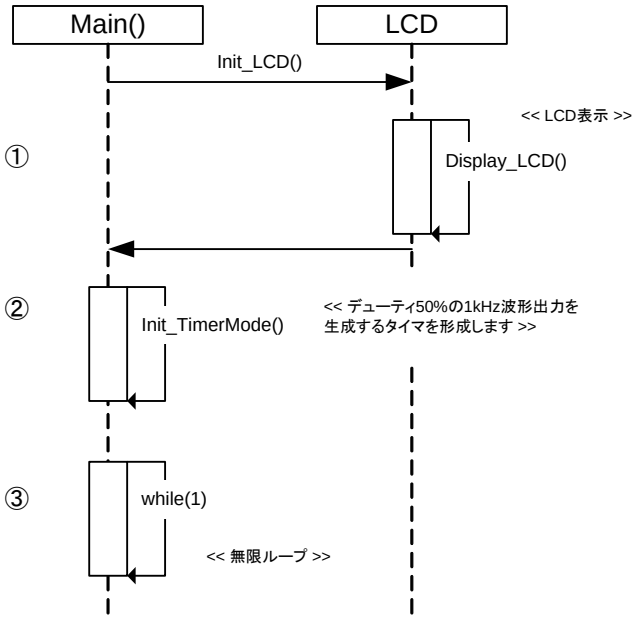

図 4-8: Timer_Mode フロー

4.8.3 RPDL

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

関数	RPDL 関数
Init_TimerMode	R_MTU2_Set
	R_MTU2_Create
	R_MTU2_ControlChannel
CB_Timer_TimerMode	R_IO_PORT_Modiy

表 4-8: Timer_Mode 用関数

4.9 PWM

本サンプルコードはタイマユニットを使ったデモコードでデューティーサイクルを変えながら波形出力します。スイッチが押されるとデューティーサイクルが固定されます。

4.9.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄のインストラクションに従ってタイマ出力ピンにオシロスコープを接続してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_PWM 関数をコールし、デューティーサイクル 10%に初期設定された 1KHz の波形を生成するタイマチャネルを形成します。また、関数はインターバル割り込みを発生させるために第 2 のタイマを形成し、割り込み毎にコールバック関数 CB_Timer_PWM をコールします。関数はさらにスイッチ割り込みとコールバック関数を形成します。
③ 第 2 のインターバルタイマは各期間でコールバック関数 CB_Timer_PWM をコールします。この関数が実行される毎に、メインタイマのデューティーサイクルはインクリメントされます。デューティーサイクルは 10%～90%の間で変化し、90%を超えると 10%に戻ります。
④ スイッチが押されると、コールバック関数 CB_Switch_PWM がコールされます。関数はデューティーサイクルを現在の値に固定し、LCD に現在の値を表示します。

4.9.2 シーケンス

PWM サンプルのプログラム実行フローを**図 4-9** に示します。

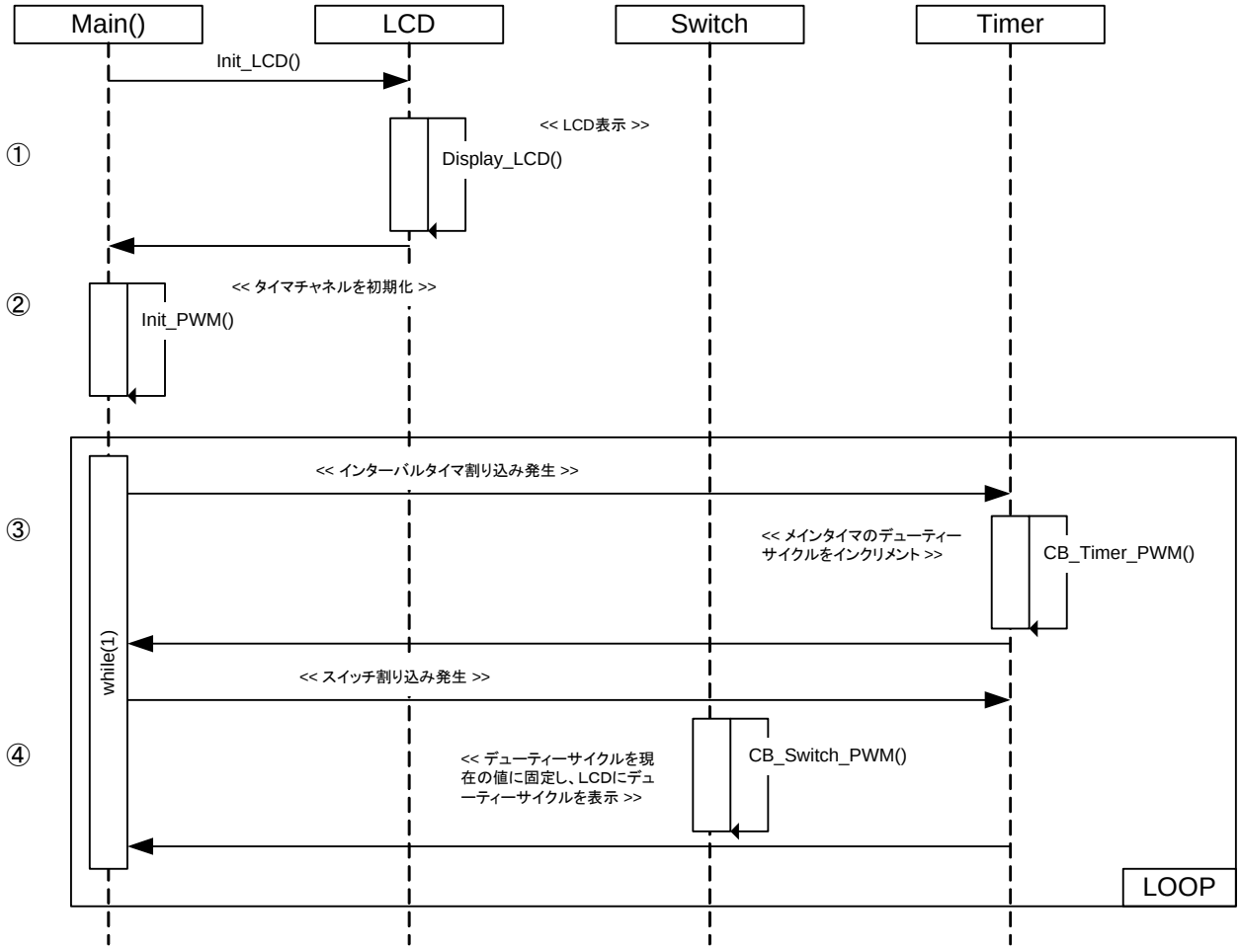

図 4-9: PWM フロー

4.9.3 RPDL

PWM 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_PWM	R_TMR_Set
	R_TMR_CreatePeriodic
CB_Timer_PWM	R_TMR_ControlPeriodic

表 4-9: PWM 用関数

4.10 WDT

本サンプルコードはウォッチドッグタイマのデモコードです。ポテンショメータによって調整された割合でウォッチドッグタイマのカウントを定期的にリセットします。リセット間隔がウォッチドッグタイマ周期よりも遅くなると、WDT オーバーフローが発生し、LED の点滅を停止します。

4.10.1 オペレーション

サンプルコードを実行する前に、ボード上のポテンショメータ RV1 を反時計回り一杯に回してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_WDT 関数はウォッチドッグタイマユニット、ADC ユニットおよびタイマユニットを形成します。ADC ユニットはリピートモード（連続スキャンモード）に設定し、タイマは周期的な割り込みによってコールバック関数 CB_Timer_WDT を実行できるように設定します。
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、割り込みが発生してコールバック関数 CB_Timer_WDT が実行されます。関数は WDT のカウントをリセットし、LED をトグル出力し、AD 変換結果をフェッチします。ポテンショメータの調整によって WDT リセット間隔を変更できます。
④ リセット間隔がウォッチドッグタイマの周期よりも長くなると、ウォッチドッグタイマはオーバーフロー割り込みが発生し、コールバック関数 CB_Overflow_WDT をコールします。この関数は LED の点滅を停止し、LCD に”Watchdog Overflow”を表示します。その後、プログラムは関数内の無限ループで待機します。

4.10.2 シーケンス

WDT サンプルのプログラム実行フローを**図 4-10** に示します。

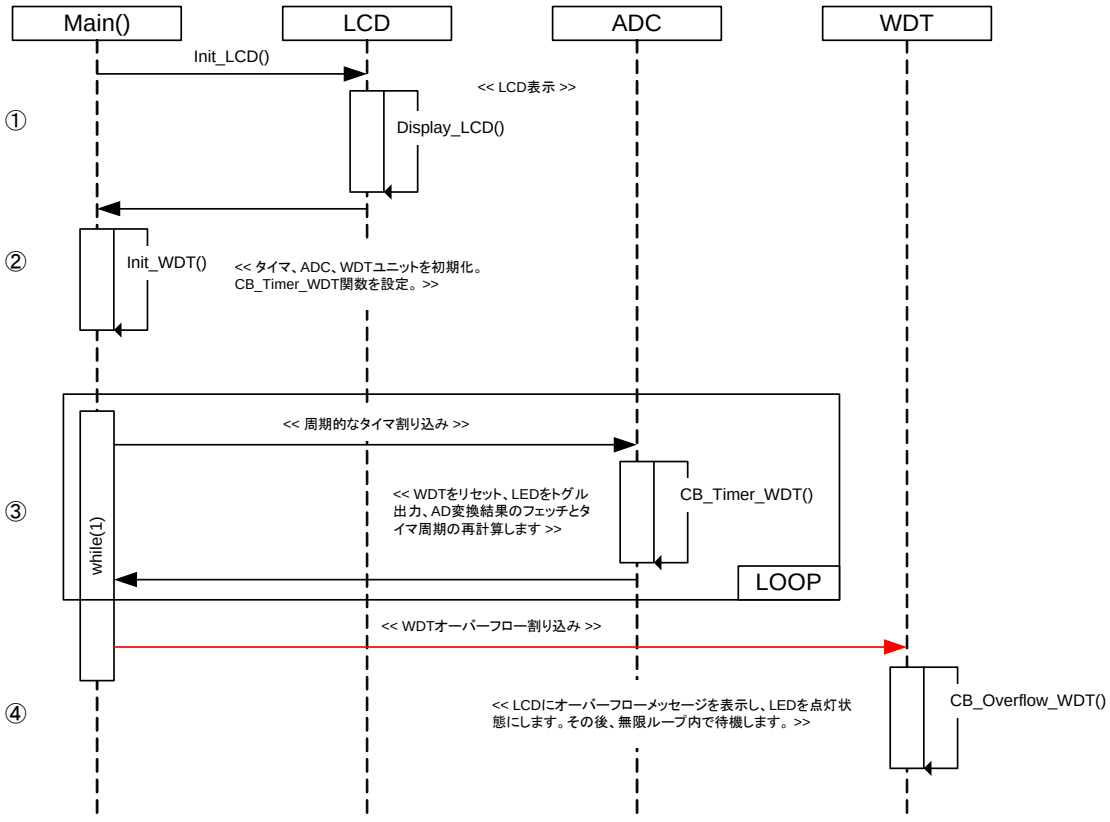

図 4-10: WDT フロー

4.10.3 RPDL

WDT 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_WDT	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_ADC_12_Control
	R_CMT_Create
	R_WDT_Set
	R_INTC_CreateExtInterrupt
	R_WDT_Control
CB_Timer_WDT	R_WDT_Control
	R_ADC_12_Read
	R_IO_PORT_Modify
	R_ADC_12_Control
	R_CMT_Control
CB_WDTOverflow	R_IO_Port_Write

表 4-10: WDT 用関数

4.11 RTC

本サンプルコードは 32.768KHz クロック源を使ったリアルタイムクロックのデモコードです。LCD に 24 時間「hh:mm:ss」表示します（開始は「00:00:00」）。

4.11.1 オペレーション

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_RTC 関数をコールし、RTC 設定を初期化して、2 つのコールバック関数 CB_Alarm_RTC、CB_1HZ_RTC を形成します。
③ サンプルコードは無限ループに入り、1 秒毎の割り込みによって RTC コールバック関数 CB_1HZ_RTC が実行されます。関数は RTC ユニットから時間（スタートからの経過時間）をフェッチし、LCD に表示します。
④ 時間がアラーム時間と一致する場合、再び割り込みが発生します。この割り込みは LED1 を点灯させるコールバック関数 CB_Alarm_RTC を実行します。

4.11.2 シーケンス

RTC サンプルのプログラム実行フローを**図 4-11** に示します。

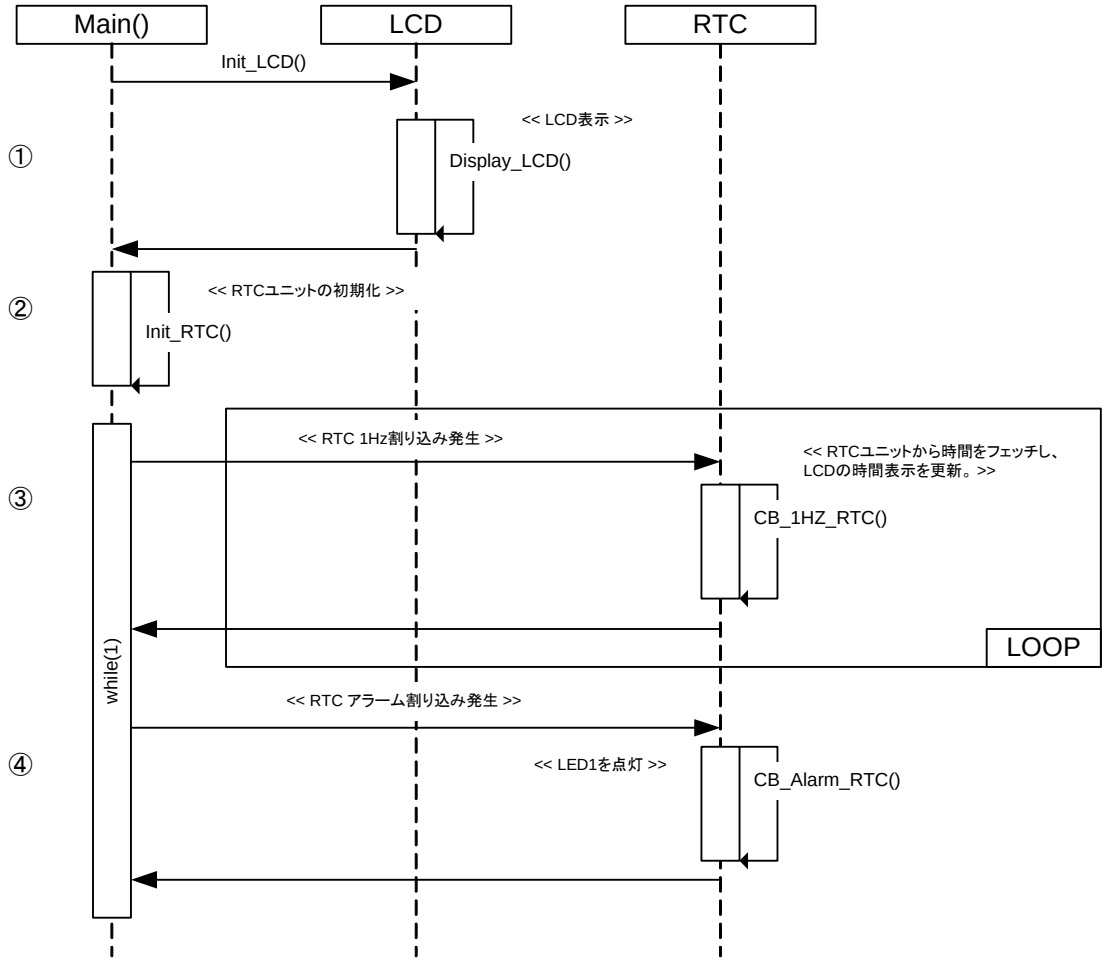

図 4-11: RTC フロー

4.11.3 RPDL

RTC 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_RTC	R_RTC_Create
	R_RTC_Control
CB_1HZ_RTC	R_RTC_Read
CB_Alarm_RTC	R_IO_PORT_Write

表 4-11: RTC 用関数

4.12 IIC_Master

本サンプルコードはマスターモードにおいて 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

4.12.1 オペレーション

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。

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。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。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。

④ 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。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。

4.12.2 シーケンス

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

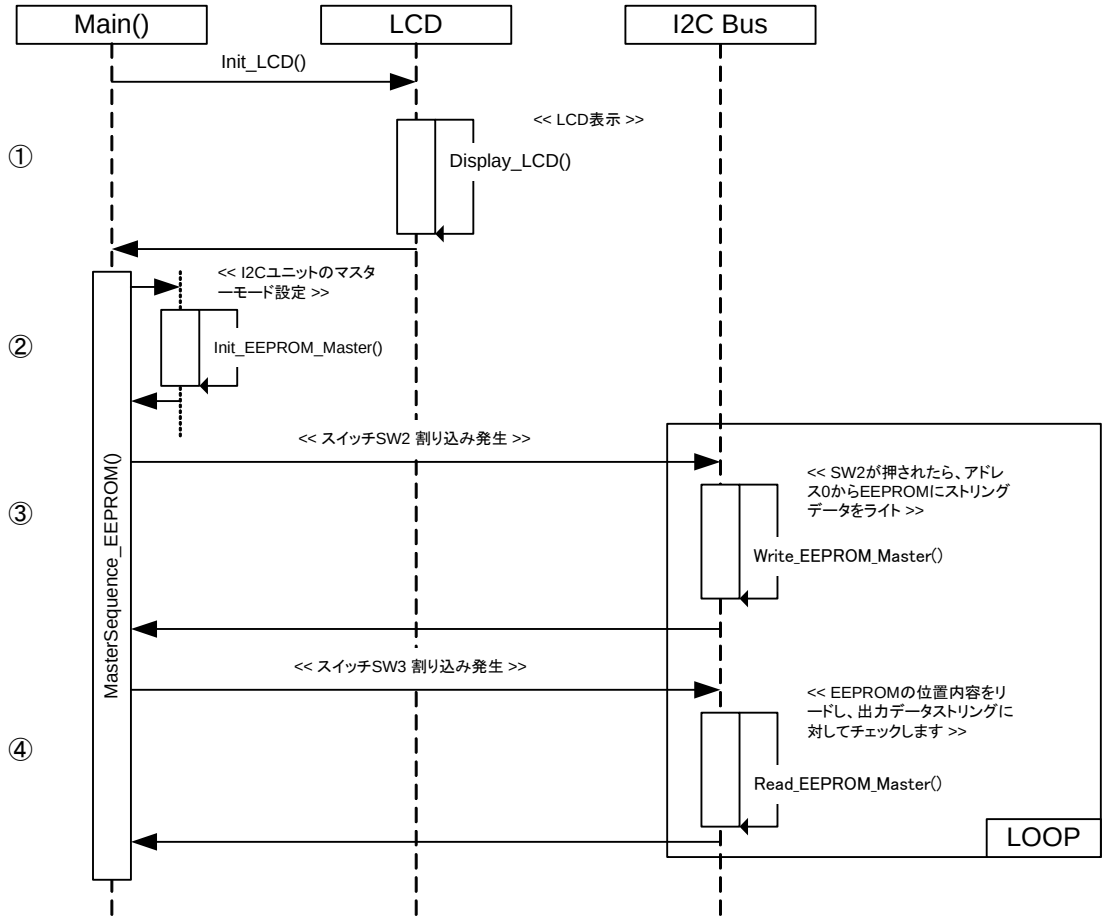

図 4-12: IIC_Master フロー

4.12.3 RPDL

IIC_Master 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_Master	R_IIC_Set
	R_IIC_Create
Write_EEPROM_Master	R_IIC_MasterSend
	R_CMT_CreateOneShot
Read_EEPROM_Master	R_IIC_MasterSend
	R_IIC_MasterReceive
	R_CMT_CreateOneShot
CheckStatus_EEPROM_Master	R_IIC_GetStatus
BusActivity_IIC	R_IO_PORT_Write

表 4-12: IIC_Master 用関数

4.13 SPI_Loopback

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。

4.13.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄のインストラクションに従ってボードの設定を変更してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_SPI 関数がコールされ、SPI ユニットを初期化し SPI 受信割り込みコールバックを形成します。またスイッチコールバック関数を形成します。
③ サンプルコードは無限ループに入ります。スイッチが押されるとコールバック関数 CB_Switch がコールされます。この関数は AD 変換結果を読み、SPI ループバックによって結果を送信します。
④ CB_Switch 関数中で送信されたデータは SPI ループバックによって受信し、割り込みを発生します。SPI 受信割り込みは CB_SPIReceive コールバック関数をコールし、フェッチした受信データとオリジナルの値を比較します。比較結果が一致するする場合、LCD に結果を表示します。送受信データ間で不一致がある場合、LCD にエラーが表示されます。

4.13.2 シーケンス

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

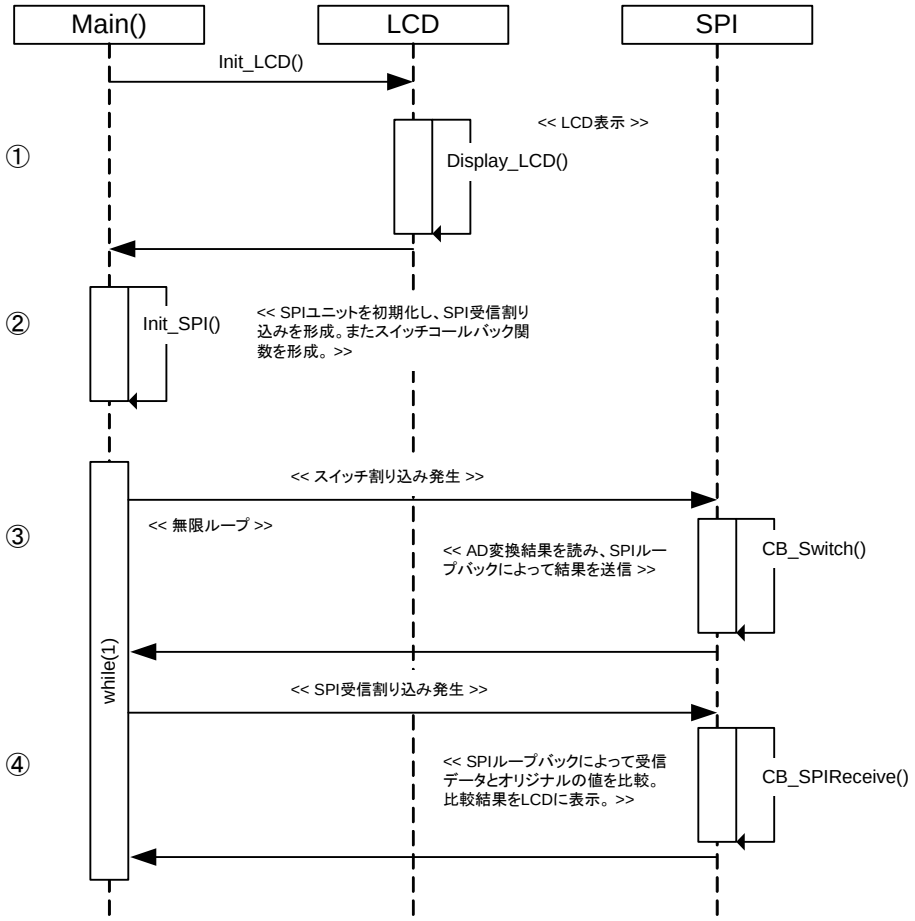

図 4-13: SPI_Loopback フロー

4.13.3 RPDL

SPI_Loopback 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_SPI	R_SPI_Create
	R_SPI_Control
	R_SPI_Command
	R_ADC_12_Set
	R_ADC12_Create_Unit
	R_ADC_12_CreateChannel
CB_Switch	R_ADC12_Control
	R_ADC12_Read
	R_SPI_Transfer

表 4-13: SPI_Loopback 用関数

4.14 Temperature Sensor

本サンプルコードは温度センサのデモコードです。ADC チャネルを使用して、内蔵温度センサの値を摂氏度に変換します。

4.14.1 オペレーション

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_TempSensor 関数をコールし、ADC モジュールをリピートモード（連続スキャン）に初期化します。さらに定期的にコールバック関数 CB_Timer_TempSensor をコールするインターバルタイマ割り込みを初期化します。
③ サンプルコードは無限ループに入ります。タイマ割り込み関数 CB_Timer_TempSensor が定期的にコールされ、現在の AD 変換結果をフェッチします。その後、摂氏度に変換され、LCD に変換結果を表示します。

4.14.2 シーケンス

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

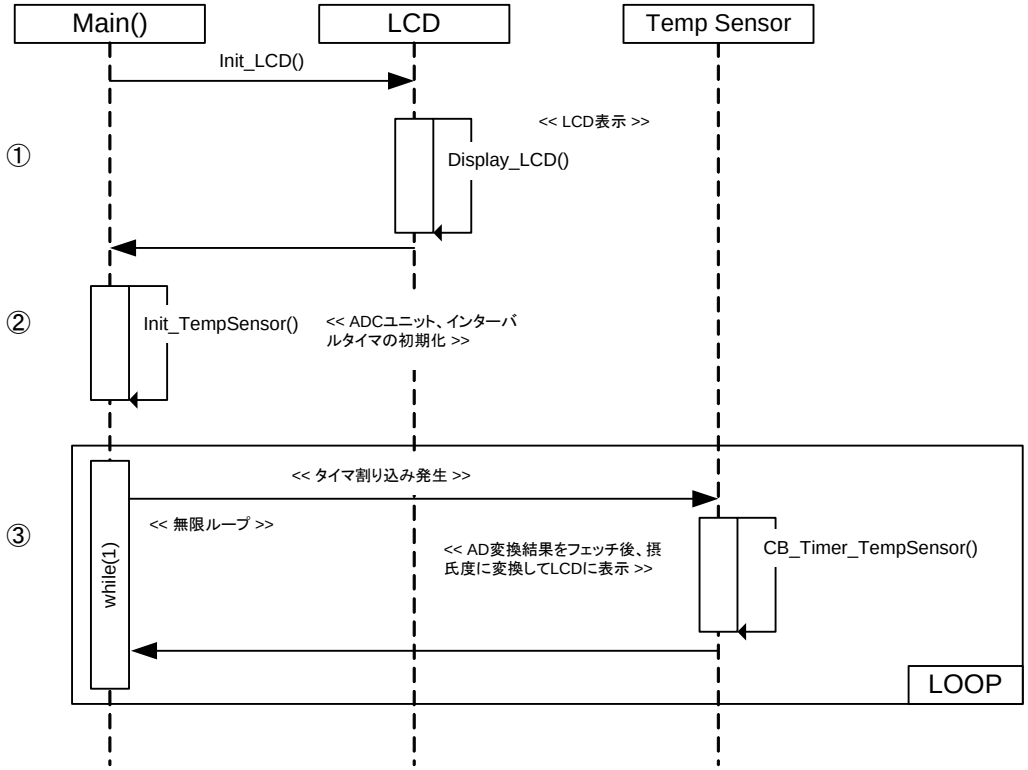

図 4-14: Temperature Sensor フロー

4.14.3 RPDL

Temperature Sensor 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_TempSensor	R_RWP_Control
	R_ADC_12_CreateUnit
	R_CMT_Create
	R_TS_Create
	R_TS_Control
CB_Timer_TempSensor	R_ADC_12_Read
	R_ADC_12_Control

表 4-14: Temperature Sensor 用関数

4.15 Analog_Compare

本サンプルコードはコンパレータモジュールのデモコードです。入力電圧とリファレンス電圧を比較します。

4.15.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄のインストラクションに従ってボードの設定を変更してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_AnalogCompare 関数をコールし、コンパレータユニットを形成します。ユニットはコンパレータへの入力電圧がリファレンス電圧よりも高い時にコールバック関数 CB_Comparator_AnalogCompare を実行します。
③ サンプルコードは無限ループに入ります。入力電圧がリファレンス電圧より高い時、コンパレータ割り込みが発生し、CB_Comparator_AnalogCompare 関数がコールされます。関数は入力電圧がリファレンス電圧より高いことを LCD と LED に示します。さらに入力電圧がリファレンス電圧以下になった場合に発生するコンパレータ割り込みを再構築します。
④ 入力電圧がリファレンス電圧以下になる場合、コンパレータは別の割り込みを発生し、CB_Comparator_AnalogCompare 関数をコールします。関数は入力電圧がリファレンス電圧より低いことを LCD と LED に示します。さらに入力電圧がリファレンス電圧より高くなった場合に発生するコンパレータ割り込みを再構築します。入力が再び高くなると、サンプルコードはステップ③と④を繰り返します。

4.15.2 シーケンス

Analog_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。

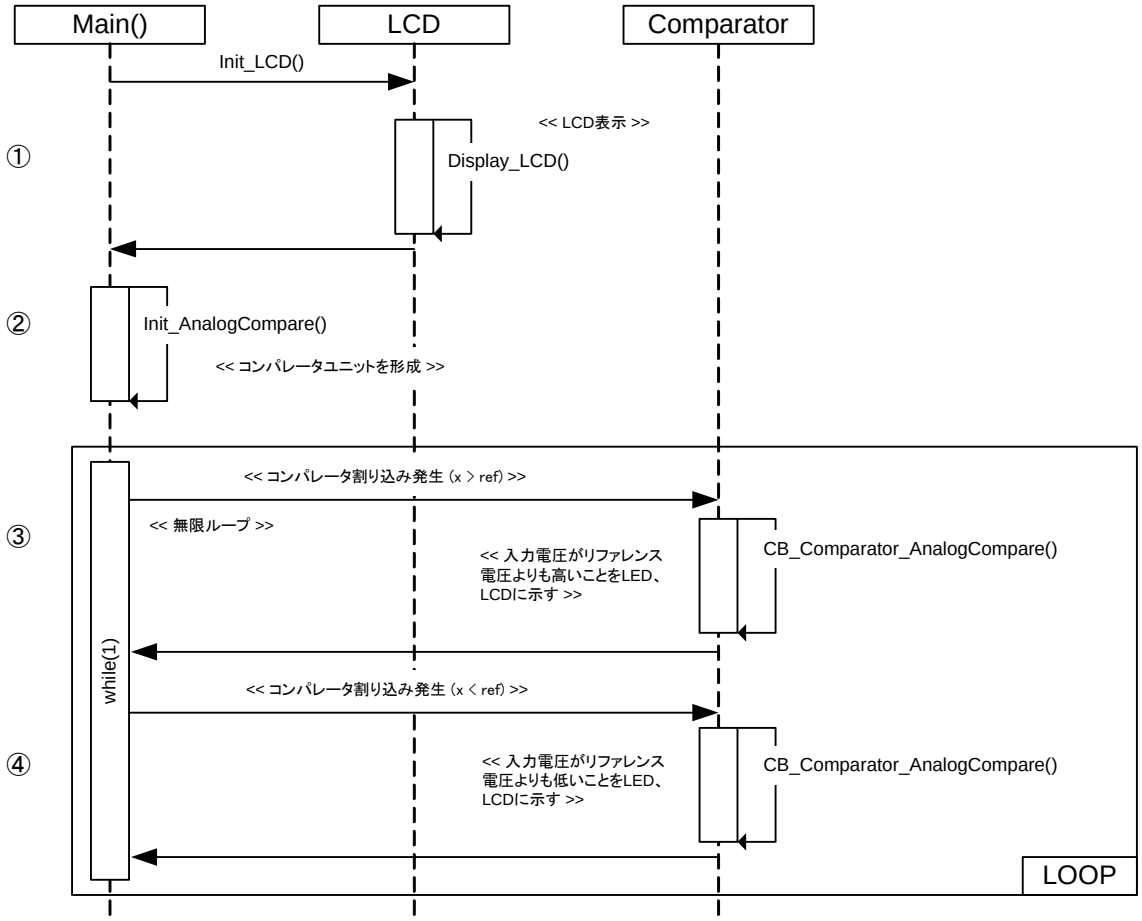

図 4-15: Analog_Compare フロー

4.15.3 RPDL

Analog_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_AnalogCompare	R_CPA_Create
CB_Comparator_AnalogCompare	R_CPA_GetStatus
	R_IO_PORT_Write
	R_CMT_CreateOneShot

表 4-15: Analog_Compare 用関数

4.16 Data Operation Circuit (DOC)

本サンプルコードはデータ演算回路のデモコードです。ターミナルソフトからのユーザ入力と基準となる 16 ビットのデータの比較結果をターミナルソフトに返します。

4.16.1 オペレーション

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（サンプルコードのヘッダ欄でインストラクションを確認できます）。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_DOC 関数をコールし、PC ターミナルソフトとの通信用、データ受信時のコールバック関数 CB_SCIRecieve_DOC 実行用に SCI ユニットを形成します。また、基準となる 16 ビットのデータ（グローバル変数 gCompareReference_DOC に設定します）への入力の比較を実行するために DOC ユニットを形成します。
③ サンプルコードは無限ループに入ります。キーを押すと、コールバック関数 CB_SCIReceive_DOC 関数がコールされます。関数はキー入力をフェッチし、ASCII 文字コードに変換します。その後、関数は値を DOC ユニットへ渡します。DOC ユニットはユーザ入力と基準データの比較を実行し、入力の一致/不一致をターミナルに返します。その後、サンプルコードは無限ループに戻り、次の入力を待ちます。

4.16.2 シーケンス

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

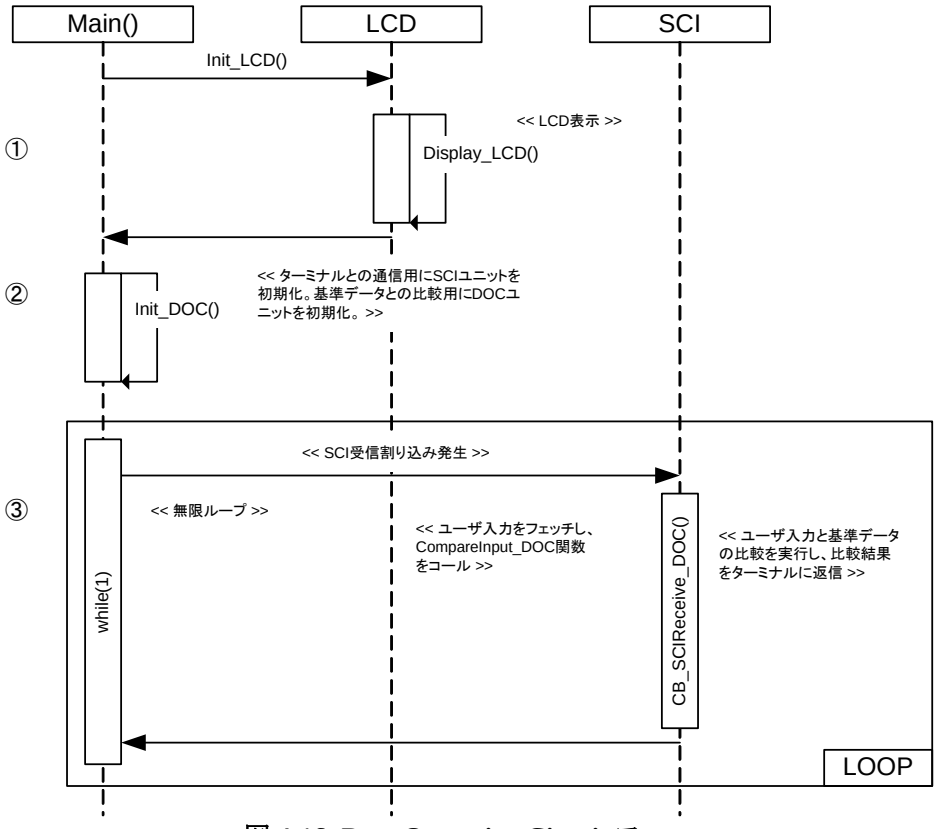

図 4-16: Data Operation Circuit フロー

4.16.3 RPDL

Data Operation Circuit 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_DOC	R_DOC_Create
	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
	R_SCI_Control
CB_SCIReceive_DOC	R_SCI_Send
	R_DOC_Write
	R_DOC_Read
	R_SCI_GetStatus
	R_SCI_Receive

表 4-16: Data Operation Circuit 用関数

4.17 Event Link Controller (ELC)

本サンプルコードはイベントリンクコントローラのデモコードです。スイッチが押されると、AD 変換を起動し、変換後に ELC ユニットは自動的に選択したポートピンをトグル出力します。

4.17.1 オペレーション

サンプルコードを実行する前に、サンプルコードのヘッダ欄に記載の指定されたピンを接続してください。

① LCD モジュールを初期化し、LCD にサンプル名とインストラクションを表示します。
② Init_ELC 関数をコールし、ADC ユニットおよび ELC ユニットを形成します。ELC ユニットは AD 変換完了後にポートピンをトグルするよう設定され、スイッチ割り込みはコールバック関数 CB_Switch_ELC をコールするよう設定されます。
③ サンプルコードは無限ループに入ります。スイッチが押されると、CB_Switch_ELC 関数が実行されます。押されたスイッチが SW1 の場合、関数は AD 変換を開始し、変換結果をストリングに変換します。その後、ストリングは LCD に表示され、関数は無限ループに戻ります。
④ AD 変換が完了すると、直ちに ELC は CPU を中断せずに選択されたポートピンをトグルします。

4.17.2 シーケンス

Event Link Controller サンプルのプログラム実行フローを**図 4-17** に示します。

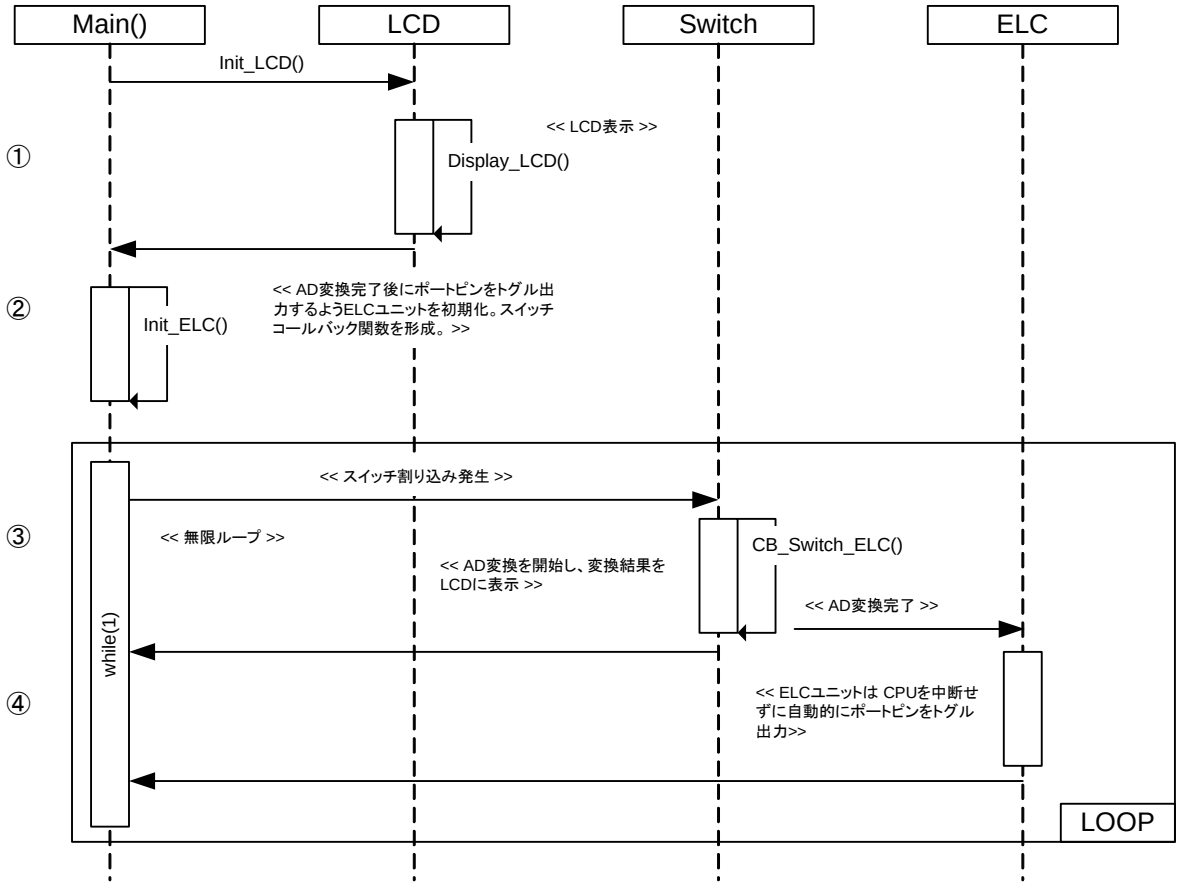

図 4-17: Event Link Controller フロー

4.17.3 RPDL

Event Link Controller で使用される関数、RPDL 関数を**表 4-17** に示します。

関数	RPDL 関数
Init_ELC	R_ADC_12_Set
	R_ELC_Create
	R_ELC_Control
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_IO_PORT_Set
	R_IO_PORT_Write
CB_Switch_ELC	R_ADC_12_Control
	R_ADC_12_Read

表 4-17: Event Link Controller 用関数

4.18 Low_Power

本サンプルコードは低消費電力モードのデモコードです。MCU はローパワー（CPU 動作）、スリープモード（CPU 停止）および通常動作間を遷移します。

4.18.1 オペレーション

本サンプルコードは一度 MCU へコードをダウンロード後、エミュレータと LCD モジュールを外した状態で実行してください。MCU の消費電流の測定についてはサンプルコードのヘッダ欄に記載されたインストラクションを参照してください。

① サンプルコードは低消費電力関数を形成するために Init_LowPower 関数をコールします。消費電力を抑えるために必要ない周辺を切り離し、コールバック関数 CB_Switch_LowPower を実行するためにスイッチ割り込みを形成します。

② コードは IdleFunction_LowPower 関数に入ります。この関数は CPU が動作していることを示すために LED0 をトグル出力します。スイッチが押されると、CB_Switch_LowPower 関数がコールされ、押されたスイッチによって MCU の電力モードを変更します。

SW1：	SW2：	SW3：
電力モード：通常	電力モード：ソフトウェアスタンバイ	電力モード：通常
動作電力制御：低速動作モード 2	動作電力制御：前モードが有効	動作電力制御：高速動作モード

4.18.2 シーケンス

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

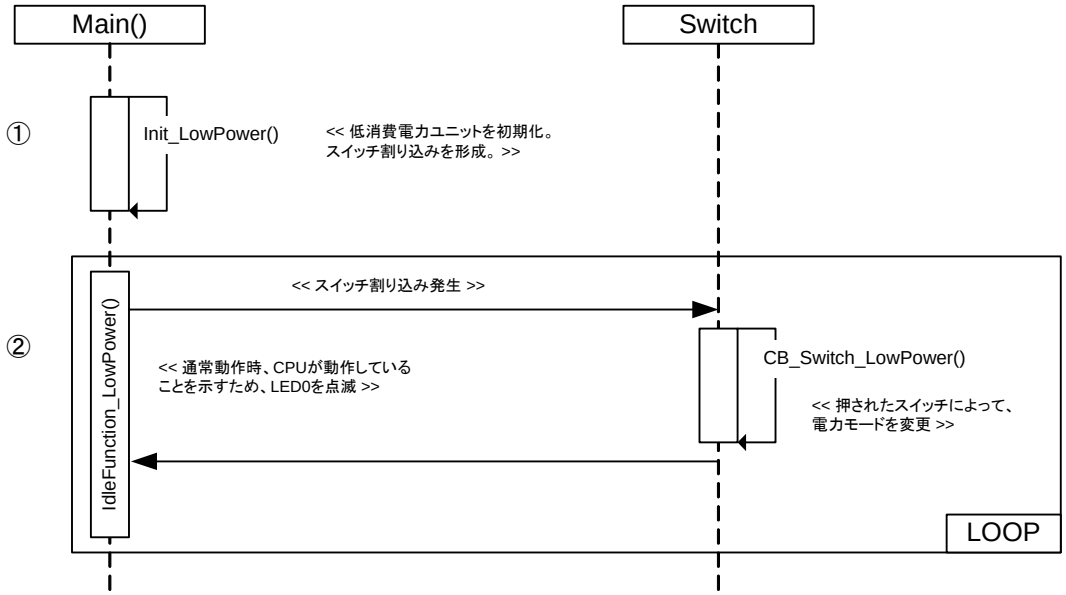

図 4-18: Low_Power フロー

4.18.3 RPDL

Low_Power 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
Init_LowPower	R_LPC_Create
	R_CGC_Control
	R_DMAC_Destroy
IdleFunction_LowPower	R_IO_PORT_Modify
	R_CMT_CreateOneShot
	R_CGC_Set
	R_CGC_Control
	R_LPC_Create
	R_IO_PORT_Write
	R_LPC_Control

表 4-18: Low_Power 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。

RX210 マイクロコントローラに関する詳細情報は、RX210 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/rskrx210 （日本サイト）
http://www.renesas.com/rskrx210 （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： software_support-eu@lm.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	RSKRX210 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	2011.08.01	ー	初版発行
2.00	2012.06.29	24	表 4-7 RPDL 関数 R_LPC_Control を削除。
		36	表 4-14 RPDL 関数 R_TS_Create、R_TS_Control を追加。
		39	表 4-16 RPDL 関数 R_SCI_Control を追加。

RSKRX210 ソフトウェアマニュアル

発行年月日　2012 年 6 月 29 日　Rev.2.00

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/contact/

Colophon 1.1

RX210 グループ

R20UT0305JG0200

ルネサス エレクトロニクス株式会社