

RX610 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

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが無いことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）
特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

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っていると、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。

RSKRX610 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	RSKRX610 ユーザーズマニュアル	R20UT0423JG
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	RSKRX610 ソフトウェアマニュアル	R20UT0428JG (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グ方法の説明	RSKRX610 チュートリアル	R20UT0424JG
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	RSKRX610 クイックスタートガイド	R20UT0425JG
回路図	CPU ボードの回路図	RSKRX610 CPU ボード回路図	R20UT0427EG
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	RX610 グループ ハードウェアマニュアル	RJJ09B0488

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 Transfer Controller	データ転送コントローラ
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.....	7
1.1 目的.....	7
2. サンプルコードコンセプト.....	8
2.1 サンプルコードの構成.....	8
2.2 サンプルコードリスト.....	9
3. Tutorialサンプル.....	10
3.1 Tutorial.....	10
3.1.1 説明.....	10
3.1.2 オペレーション.....	11
3.1.3 シーケンス.....	12
3.1.4 RPDL.....	12
3.2 Application.....	13
3.2.1 説明.....	13
4. 周辺機能サンプル.....	14
4.1 ADC_OneShot.....	14
4.1.1 オペレーション.....	14
4.1.2 シーケンス.....	15
4.1.3 RPDL.....	15
4.2 ADC_Repeat.....	16
4.2.1 オペレーション.....	16
4.2.2 シーケンス.....	16
4.2.3 RPDL.....	17
4.3 Async_Serial.....	17
4.3.1 オペレーション.....	17
4.3.2 シーケンス.....	18
4.3.3 RPDL.....	19
4.4 CRC.....	19
4.4.1 オペレーション.....	19
4.4.2 シーケンス.....	20
4.4.3 RPDL.....	20
4.5 Data_Flash.....	21
4.5.1 オペレーション.....	21
4.5.2 シーケンス.....	22
4.5.3 RPDL.....	22
4.6 DMAC.....	23
4.6.1 オペレーション.....	23
4.6.2 シーケンス.....	23
4.6.3 RPDL.....	24
4.7 DTC.....	24
4.7.1 オペレーション.....	24
4.7.2 シーケンス.....	25
4.7.3 RPDL.....	25
4.8 Power_Down.....	26
4.8.1 オペレーション.....	26
4.8.2 シーケンス.....	27
4.8.3 RPDL.....	27
4.9 PWM_Mode.....	28
4.9.1 オペレーション.....	28
4.9.2 シーケンス.....	29

4.9.3	RPDL	29
4.10	Sync_Serial	30
4.10.1	オペレーション	30
4.10.2	シーケンス	31
4.10.3	RPDL	31
4.11	Timer_Capture	32
4.11.1	オペレーション	32
4.11.2	シーケンス	33
4.11.3	RPDL	33
4.12	Timer_Compare	34
4.12.1	オペレーション	34
4.12.2	シーケンス	34
4.12.3	RPDL	34
4.13	Timer_Event	35
4.13.1	オペレーション	35
4.13.2	シーケンス	35
4.13.3	RPDL	35
4.14	Timer_Mode	36
4.14.1	オペレーション	36
4.14.2	シーケンス	36
4.14.3	RPDL	36
4.15	WDT	37
4.15.1	オペレーション	37
4.15.2	シーケンス	38
4.15.3	RPDL	39
5	追加情報	40

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>

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。

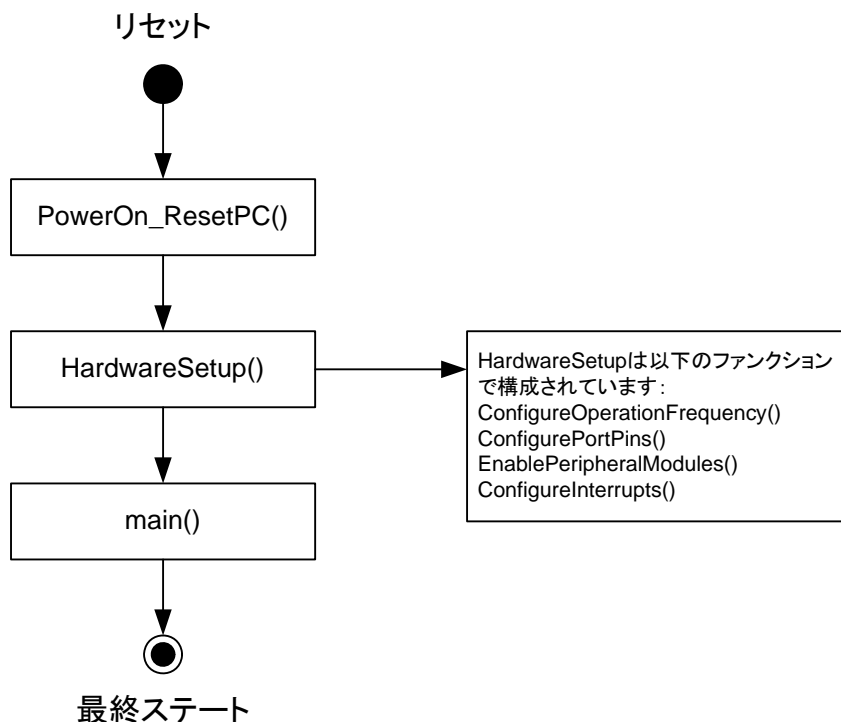


図 2-1: サンプルコードの基本構成

HardwareSetup 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 R_INTC_CreateFastInterrupt

表 2-1: HardwareSetup 関数

2.2 サンプルコードリスト

RSKRX610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
ADC_OneShot	10bit ADC モジュール（ワンショットモード）のデモコード
ADC_Repeat	10bit ADC モジュール（リピートモード）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
PWM_Mode	TPU モジュール（PWM モード）のデモコード
Power_Down	消費電力低減機能のデモコード
CRC	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	TMR モジュール（イベント）のデモコード
Timer_Mode	TMR モジュール（タイマモード）のデモコード
DMAC	DMAC モジュールのデモコード
Data_Flash	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

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“Tutorial”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。

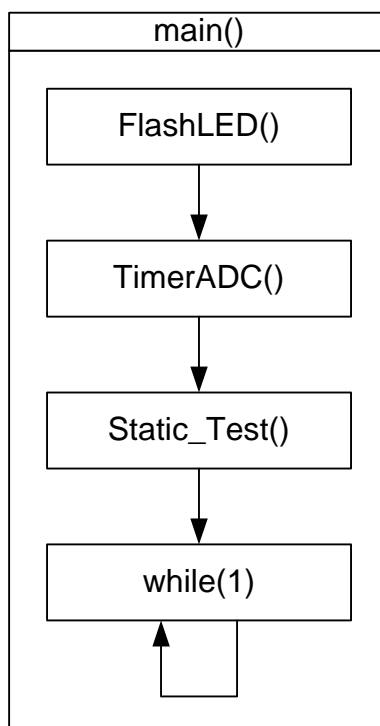


図 3-1: Tutorial フロー

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_TMR 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_TMR 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。

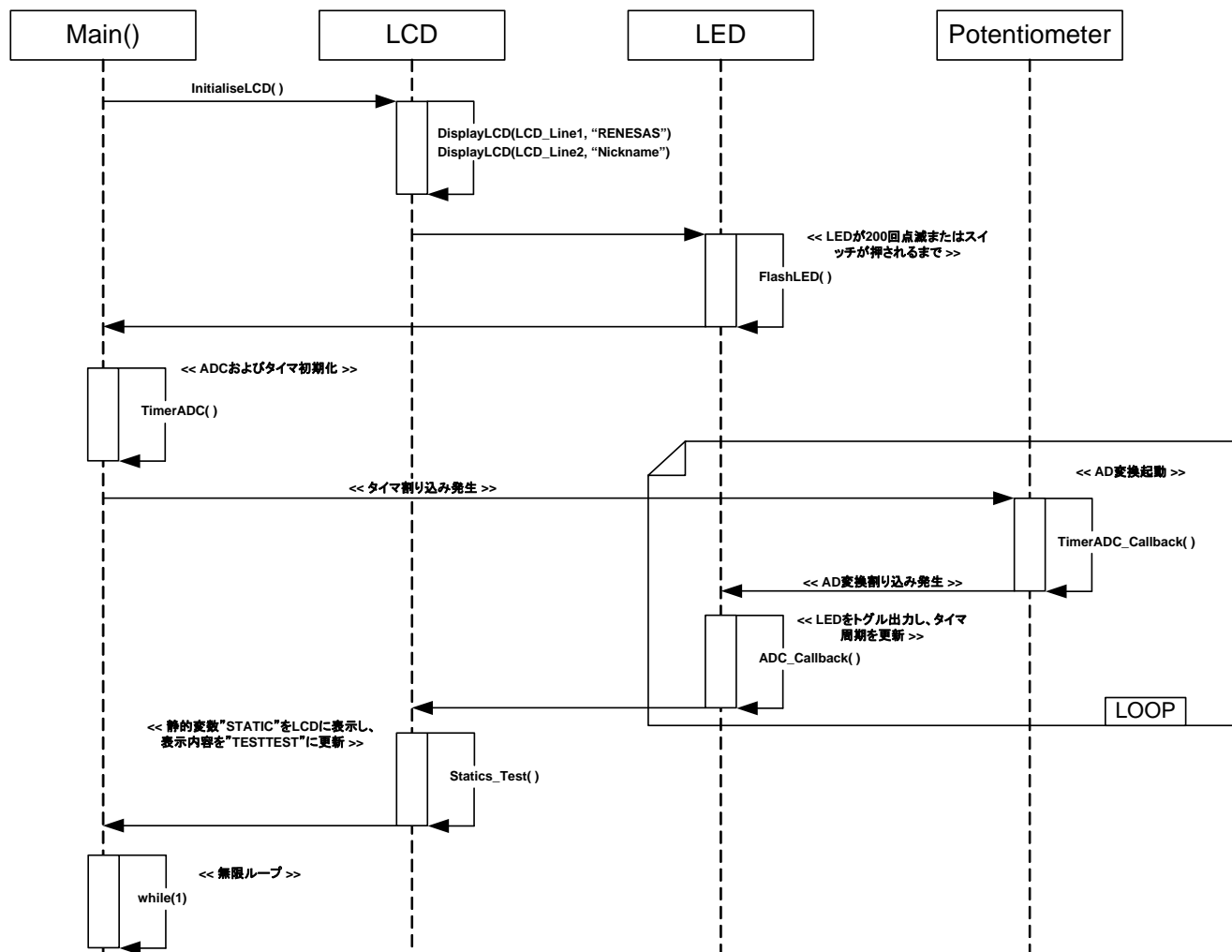


図 3-2: Tutorial フロー

3.1.4 RPD L

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
FlashLED	R_TMR_CreatePeriodic
	R_TMR_Destroy
ToggleLEDs	R_IO_PORT_Modify
StartTimer	R_TMR_CreateUnit
StartADC	R_ADC_10_Create
CB_TimerADC	R_ADC_10_Control
CB_ADConversion	R_ADC_10_Read
	R_TMR_ControlUnit

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 ADC_OneShot

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。

4.1.1 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. ADC ユニットを初期化する `Init_ADC` 関数をコールします。サンプルコードは割り込みを使用せず、変換が完了したかどうかをポーリングします。
3. 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。

4.1.2 シーケンス

ADC_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。

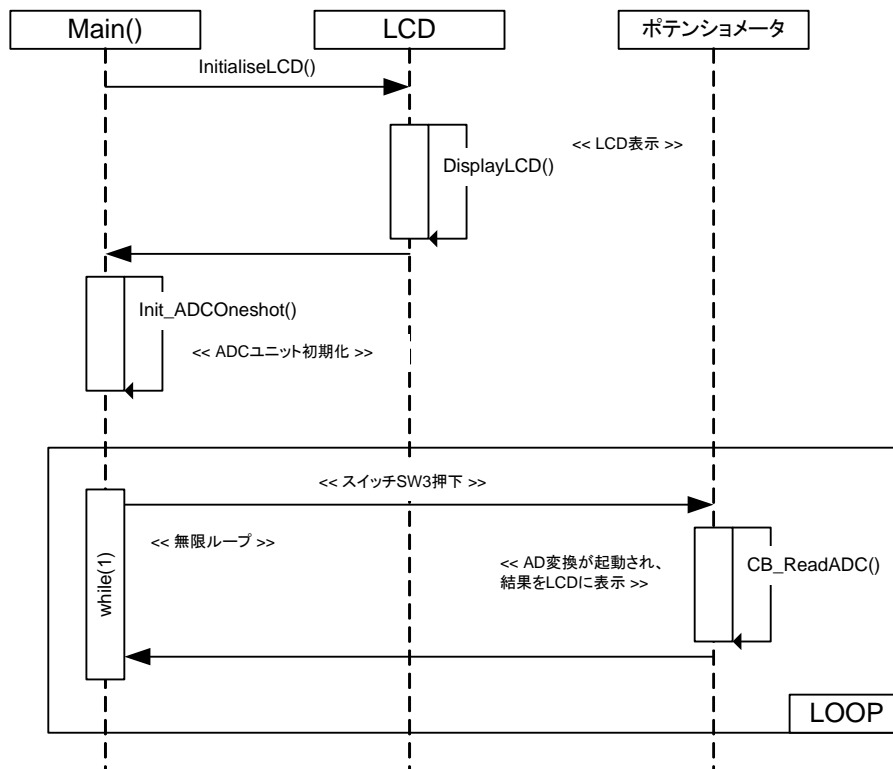


図 4-1: ADC_OneShot フロー

4.1.3 RPDL

ADC_OneShot 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
Init_ADC	R_ADC_10_Create
CB_ReadADC	R_ADC_10_Control
	R_ADC_10_Read

表 4-1: ADC_OneShot 用関数

4.2 ADC_Repeat

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。

4.2.1 オペレーション

1. LCD モジュールを初期化し、LCD にサンプルコード名を表示します。
2. ADC_Init 関数をコールし、ADC ユニットおよび TMR モジュールを初期化します。
3. ADC チャンネル 0 は変換完了ごとに AD 変換結果を読むために使用するコールバック関数 CB_ADC で形成されます。TMR チャンネル 2 はタイマオーバーフロー上のコールバック関数 CB_TMR によってサービスされる割り込みを生成するために形成されます。そして、CB_TMR は AD 変換を開始します。

4.2.2 シーケンス

ADC_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。

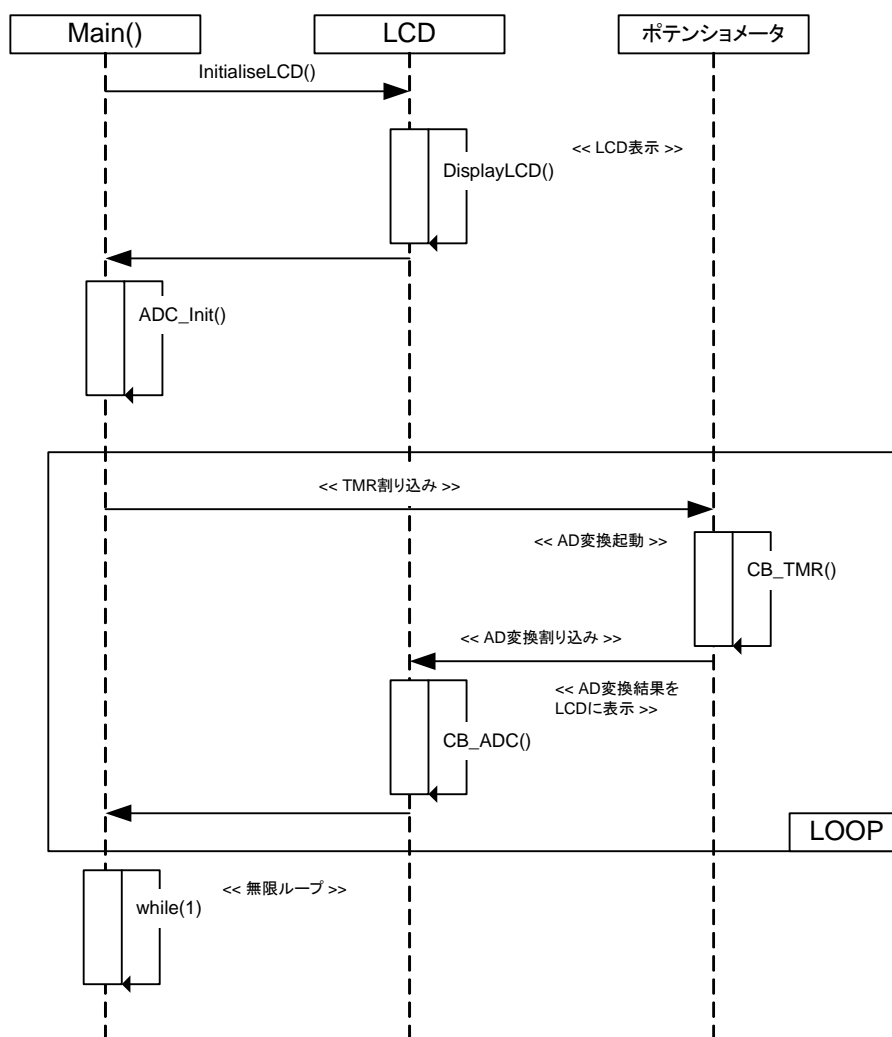


図 4-2: ADC_Repeat フロー

4.2.3 RPD_L

ADC_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
ADC_Init	R_ADC_10_Create
	R_TMR_CreatePeriodic
CB_TMR	R_ADC_10_Control
CB_ADC	R_ADC_10_Read

表 4-2: ADC_Repeat 用関数

4.3 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。

4.3.1 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、インストラクションを表示します。
3. UART モジュールの非同期通信のためのチャンネル 1 を初期化するために Async_Init 関数がコールされます。また、200ms ごとに周期的な割り込みを生成するためにコールバック関数 CB_TMR を備えた TMR ユニット 1 を形成します。
4. その後、コードは無限ループで待機します。
5. CB_TMR はターミナルへ文字出力を制御するために使用されます。UART チャンネルのステータスを取得し、エラーが検出されない場合、文字送信するために Transmit_Async 関数をコールします。送信前に、Transmit_Async 関数は SCI ステータスフラグを立てるかどうかに最初にチェックします。ステータスが 1 にセットされている場合、送信される文字が '9' を超えているかどうかチェックします。越えている場合、'0' にリセットして送信します。
6. ターミナルへの文字受信は、コールバック関数 CB_SCIReceive が実行されます。関数は文字が 'z' かチェックします。'z' の場合はターミナル上への文字出力を停止させるためにステータスフラグを偽にセットします。LED0、LED1 は送信ステータスを示すために点灯と消灯を繰り返します。
7. 受信文字が 'z' でない場合は、ステータスフラグは真にセットされ、文字出力は継続します。SCI チャンネルは別の受信割り込みの準備ができてリセットされます。

4.3.2 シーケンス

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

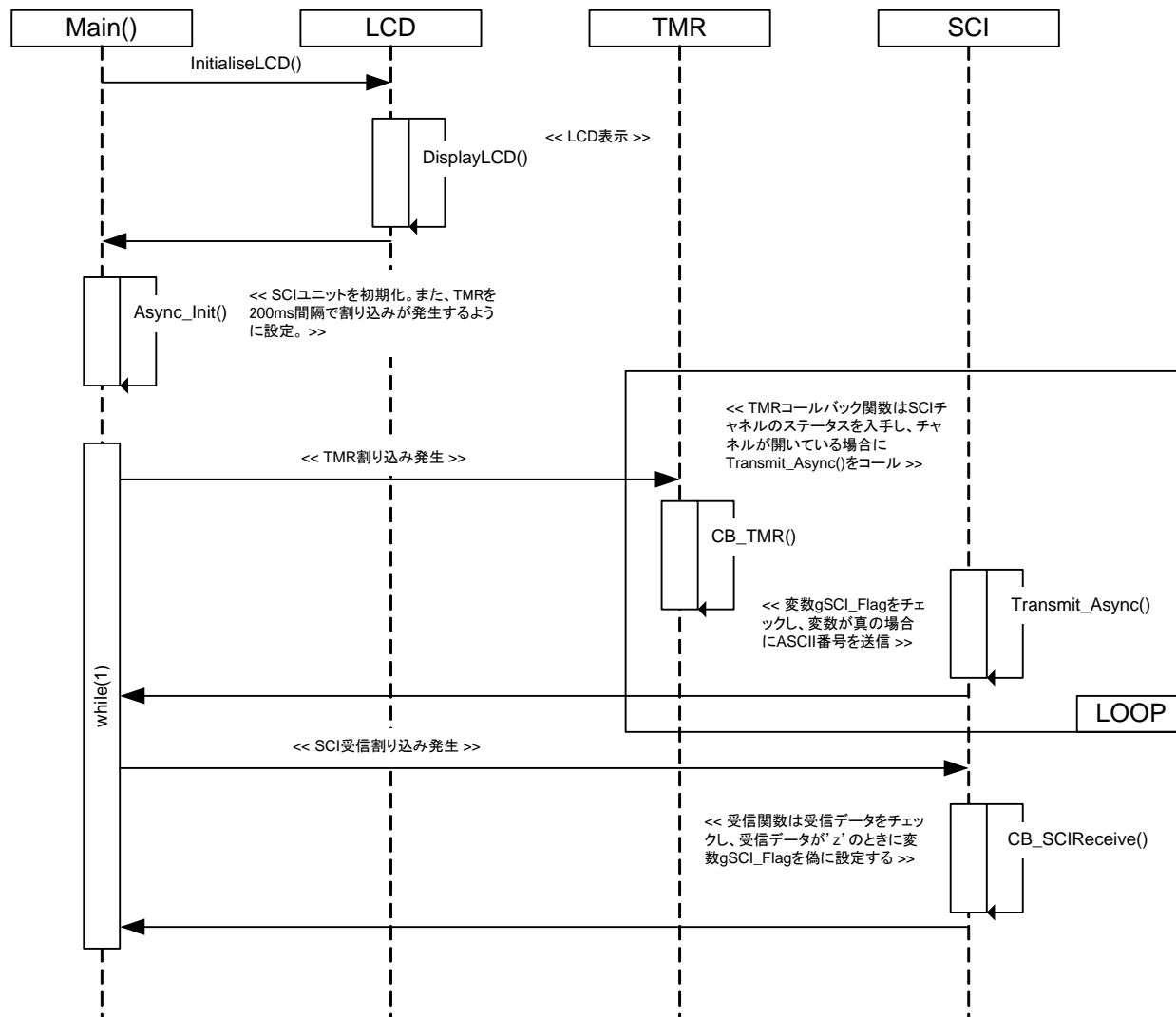


図 4-3: Async_Serial フロー

4.3.3 RPD_L

Async_Serial 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
Async_Init	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
	R_TMR_CreatePeriodic
CB_TMR	R_SCI_GetStatus
Transmit_Async	R_SCI_Send
	R_IO_PORT_Write
CB_SCIReceive	R_IO_PORT_Write
	R_SCI_Receive

表 4-3: Async_Serial 用関数

4.4 CRC

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。

4.4.1 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. CRC_Init 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。

4.4.2 シーケンス

CRC サンプルのプログラム実行フローを図 4-4 に示します。

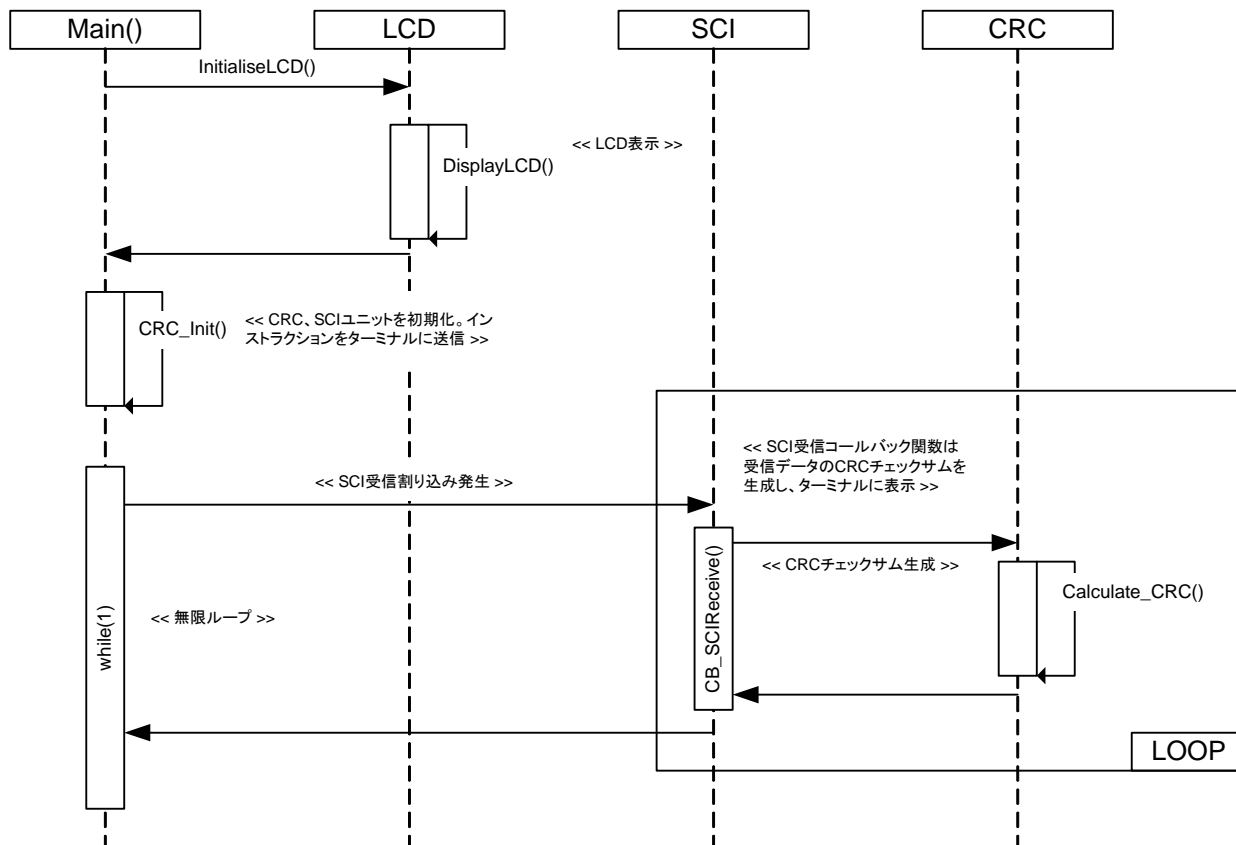


図 4-4: CRC フロー

4.4.3 RPD L

CRC 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
CRC_Init	R_CRC_Create
	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
Calculate_CRC	R_CRC_Write
	R_CRC_Read
CB_SCIReceive	R_SCI_GetStatus
	R_SCI_Send
	R_SCI_Receive

表 4-4: CRC 用関数

4.5 Data_Flash

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。

4.5.1 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。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。
7. 新しい書き込みを行うには、スイッチ SW1 を押してください。

4.5.2 シーケンス

Data_Flash サンプルのプログラム実行フローを図 4-5 に示します。

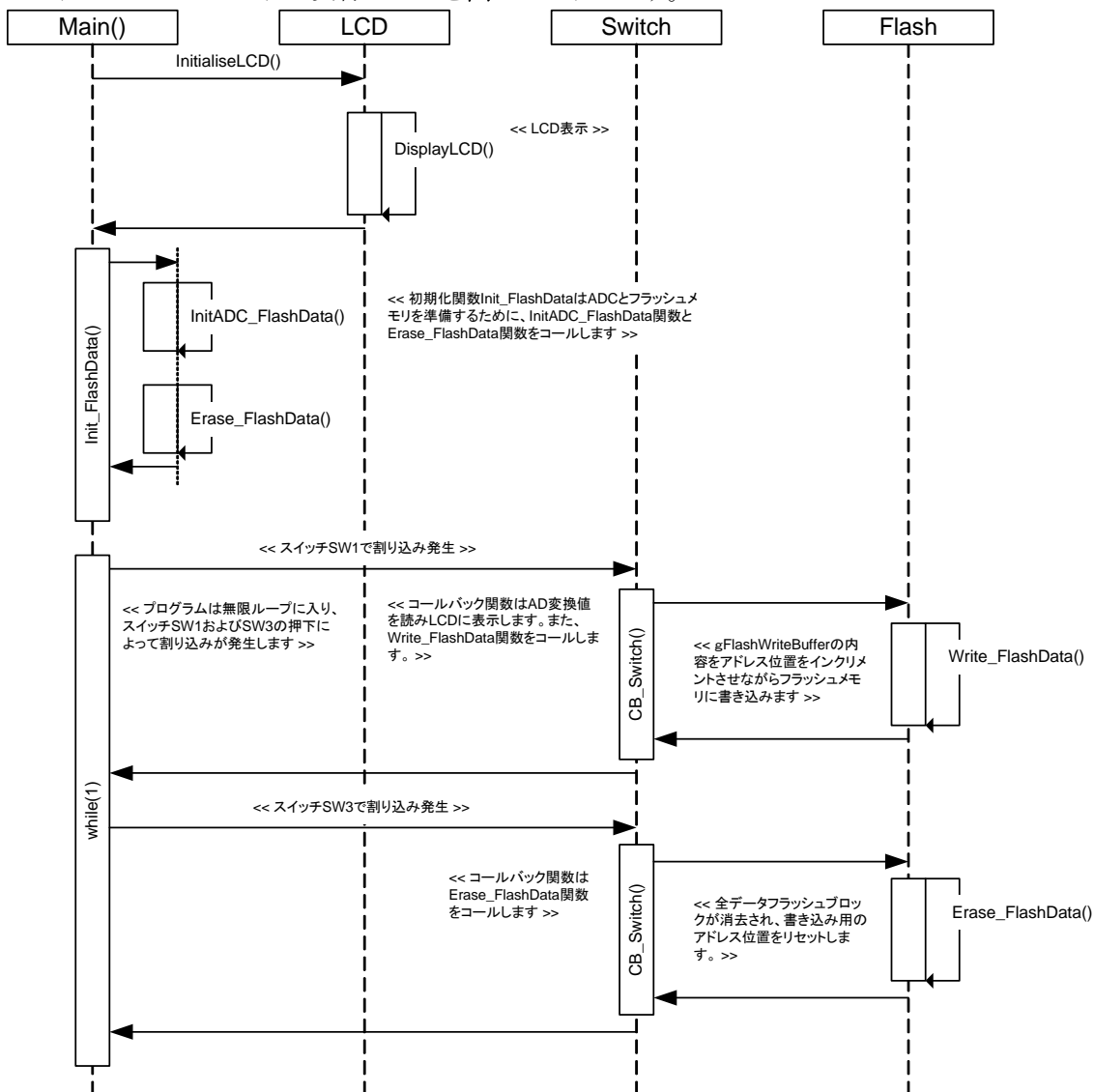


図 4-5: Data_Flash フロー

4.5.3 RPDL

Data_Flash 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_FlashData	R_ADC_10_Create
CB_Switch	R_ADC_10_Control
	R_ADC_10_Read

表 4-5: Data_Flash 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 RX600 (REU05B0131-0141) を参照ください。

4.6 DMAC

本サンプルコードは DMAC ユニットを使用してグローバル変数 `gDMA_DataBuff` へのデータ転送を行います。

4.6.1 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. 16 ビットの DMAC チャンネル 0 を形成するために `Init_DMAC` 関数がコールされます。転送モードは自動的に転送の後にデータ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_DMAC` はすべての転送が完了するとコールされます。
3. コードが無限ループに入る前に DMAC は利用可能なり転送オペレーションが開始されます。
4. すべてのデータ転送が完了すると、コールバック関数 `CB_DMAC` が実行された、転送オペレーションが完了したことを示すために LED1 を点灯させます。

4.6.2 シーケンス

DMAC サンプルのプログラム実行フローを図 4-6 に示します。

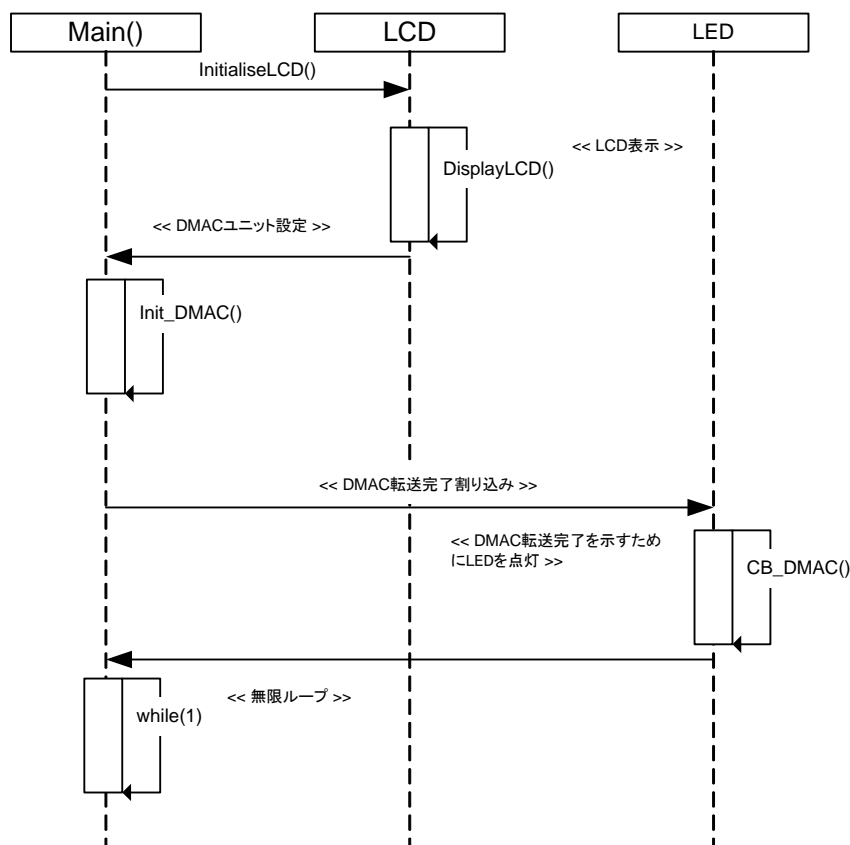


図 4-6: DMAC フロー

4.6.3 RPD_L

DMAC 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_DMAL	R_DMAL_Create
	R_DMAL_Control
DMAL_callback	R_IO_PORT_Modify

表 4-6: DMAL 用関数

4.7 DTC

本サンプルコードは DTC ユニッモ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 によつて転送します。

4.7.1 オペレーション

1. LCD モジュールを初期化し、LCD にインストラクシヨンを表示します。
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 ユニッモおよび ADC ユニッモを設定します。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モ 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。

4.7.2 シーケンス

DTC サンプルのプログラム実行フローを図 4-7 に示します。

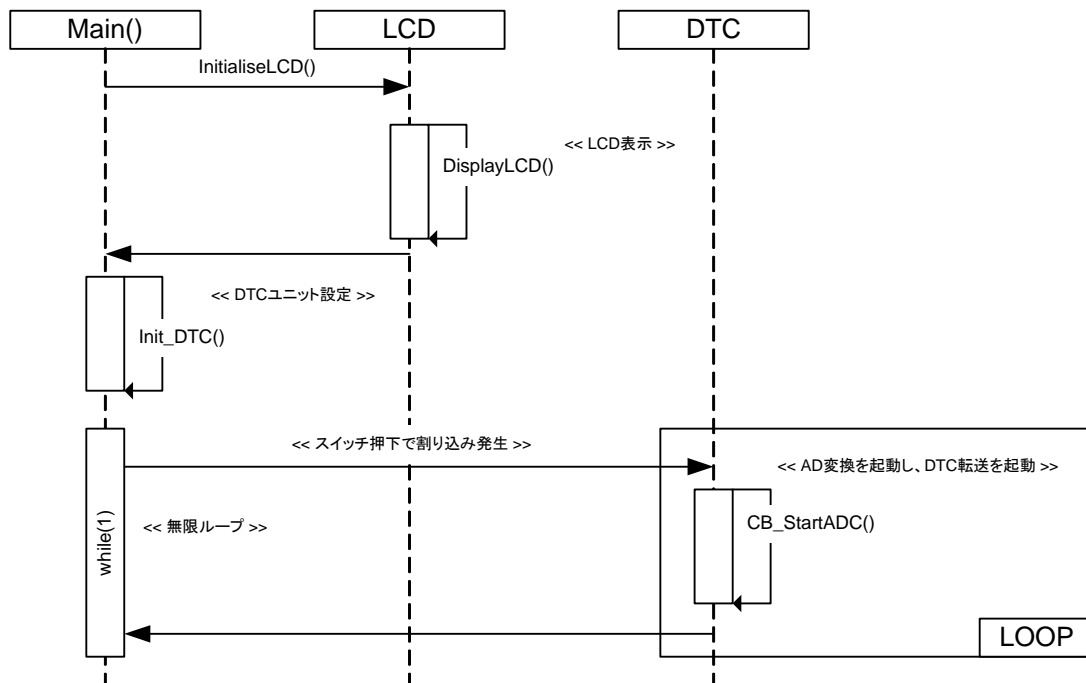


図 4-7: DTC フロー

4.7.3 RPD L

DTC 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_DTC	R_DTC_Set
	R_DTC_Create
	R_ADC_10_Create
	R_DTC_Control
CB_StartADC	R_DTC_GetStatus
	R_DTC_Control
	R_INTC_Write
	R_ADC_10_Control

表 4-7: DTC 用関数

4.8 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、スイッチによりスタンバイモードにエントリします。

4.8.1 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、低消費電力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。

4.8.2 シーケンス

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

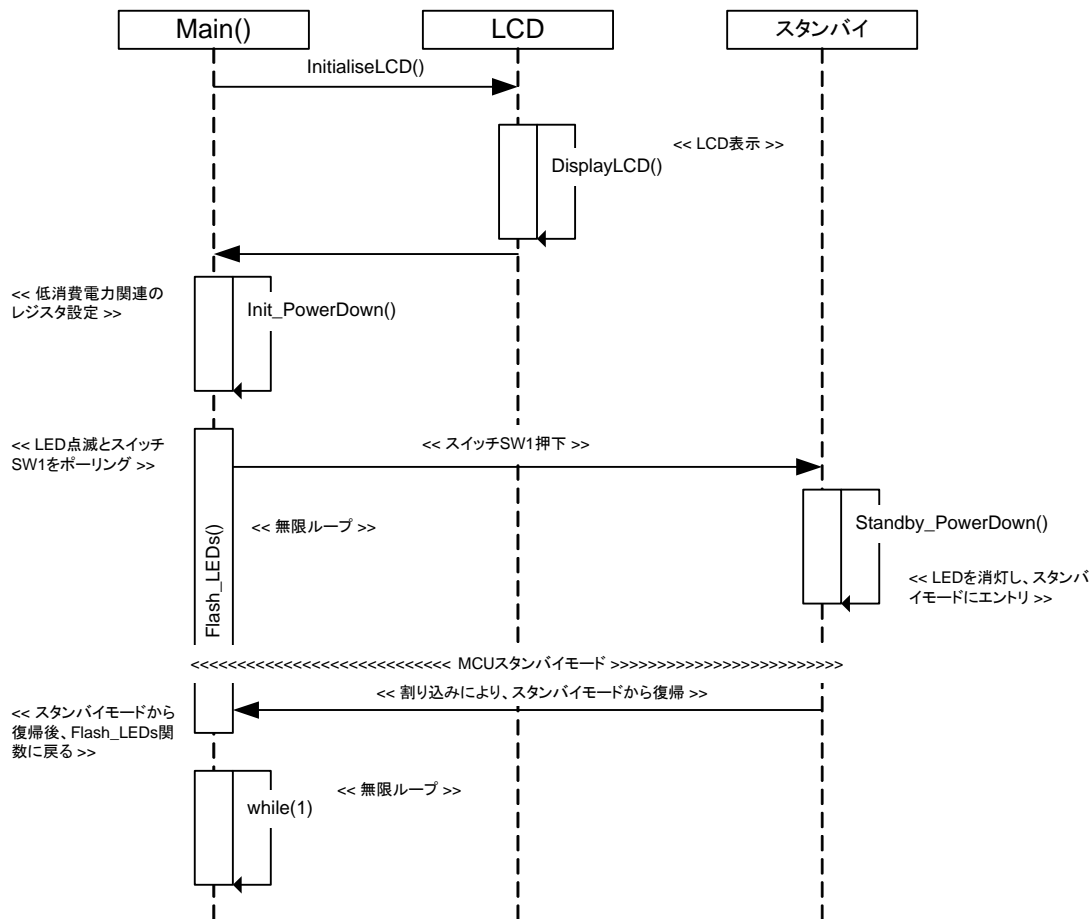


図 4-8: Power_Down フロー

4.8.3 RPD L

Power_Down 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
Init_PowerDown	R_LPC_Create
Flash_LEDs	R_CMT_CreateOneShot
	R_IO_PORT_Modify
	R_IO_PORT_Write
Standby_PowerDown	R_IO_PORT_Write
	R_LPC_Control

表 4-8: Power_Down 用関数

4.9 PWM_Mode

本サンプルコードはタイマの PWM 機能を使って 1kHz の波形を出力します。PWM 波形のデューティサイクルは 10% から 90% の範囲で繰り返し変化します。スイッチを押すとデューティサイクルは固定されます。

4.9.1 オペレーション

1. サンプルコードを実行する前に、ボード上の JA2.Pin20 にオシロスコープ等の測定装置を接続してください。
2. サンプルコードは 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。
3. `Init_PWM` 関数をコールし、10% のデューティサイクルを備えた周期的な 1kHz の出力信号を生成するために TMR ユニットのチャンネル 0 を形成します。TMR ユニットのチャンネル 1 は周期的な割り込み生成のために形成されます。これは 10% から 90% までデューティサイクルの変化を制御するために使用されます。
4. 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。
5. `CB_TMR` はチャンネル 1 の割り込みによってデューティサイクルを更新するためにコールされます。

4.9.2 シーケンス

PWM_Mode サンプルのプログラム実行フローを図 4-9 に示します。

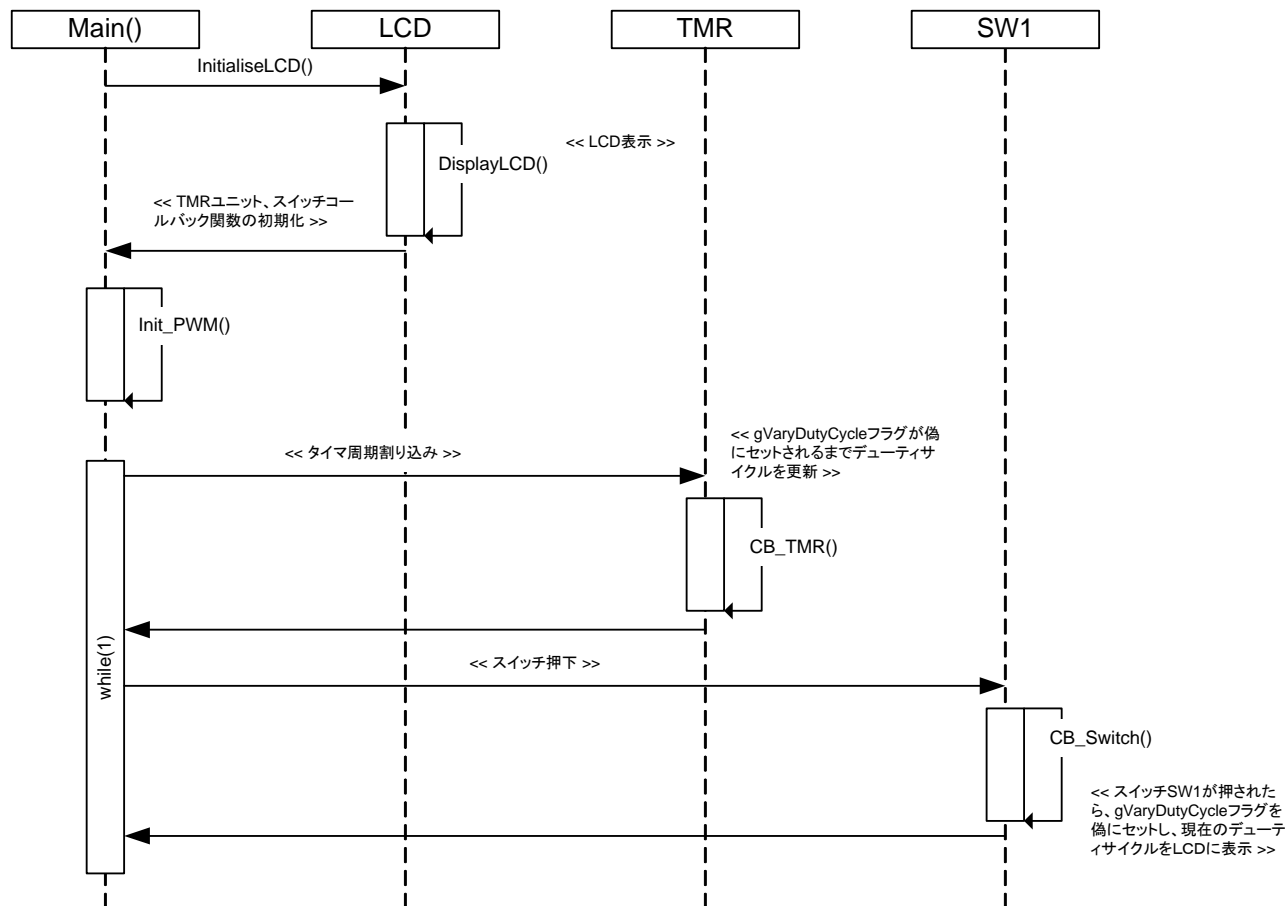


図 4-9: PWM_Mode フロー

4.9.3 RPD L

PWM_Mode 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
PWM_Init	R_TMR_CreatePeriodic
CB_TMR	R_TMR_ControlPeriodic

表 4-9: PWM_Mode 用関数

4.10 Sync_Serial

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 チャンネル間をループバックさせ通信します。

4.10.1 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. `SCI3toSCI1Transfer_Sync` 関数をコールし、SCI チャンネルを設定します。その後、チャンネル 3 からチャンネル 1 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_SCI1Receive` 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. `SCI1toSCI3Transfer_Sync` 関数をコールし、チャンネル 1 からチャンネル 2 にデータを送信します。データを受信すると `CB_SCI3Receive` 関数がコールされます。
5. `CB_SCI3Receive` 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。

4.10.2 シーケンス

Sync_Serial サンプルのプログラム実行フローを図 4-10 に示します。

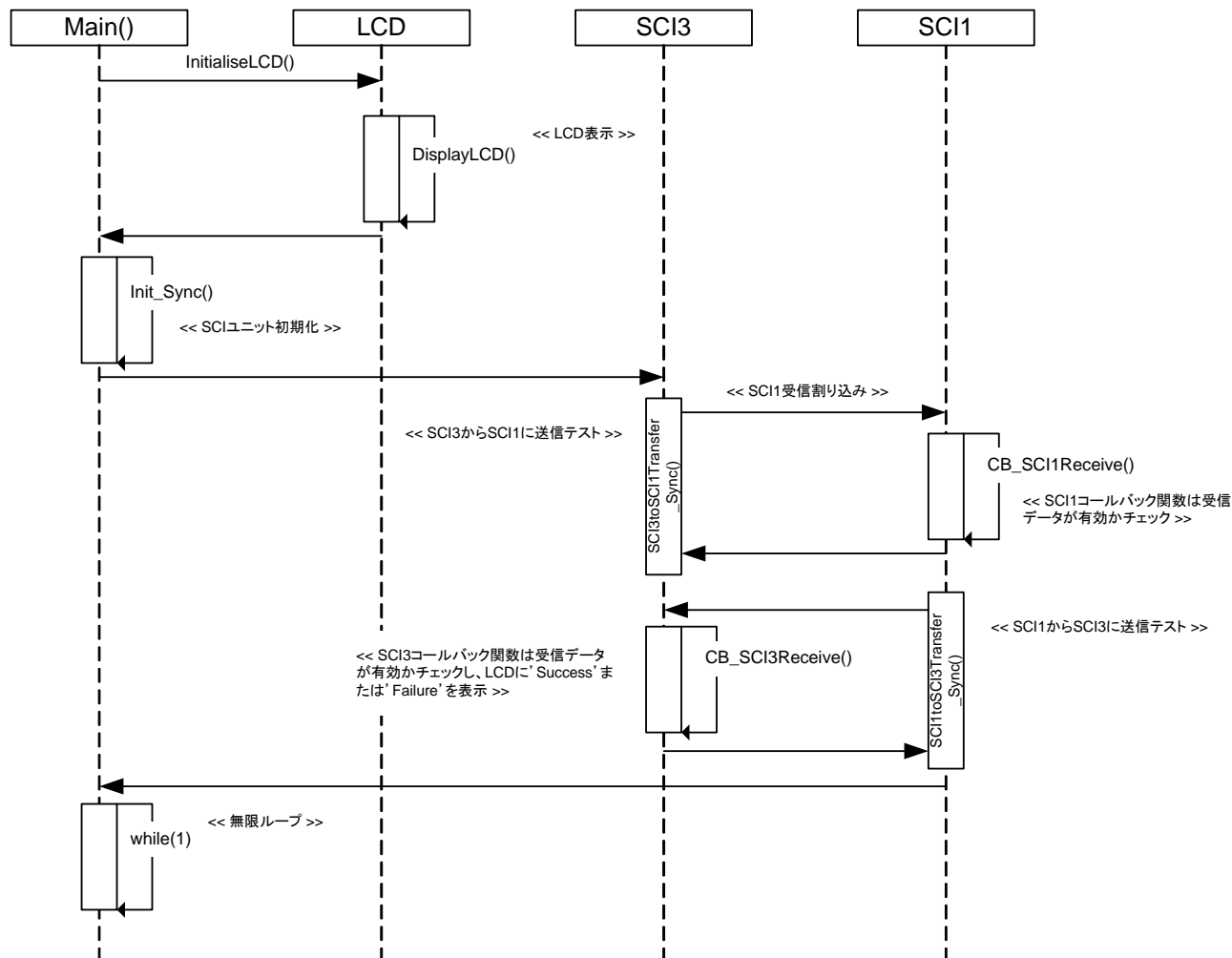


図 4-10: Sync_Serial フロー

4.10.3 RPDL

Sync_Serial 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
SCI_Init	R_SCI_Create
	R_SCI_Receive
SCI_Send	R_IO_PORT_Modify
	R_SCI_Send
SCI_RX_callback	R_IO_PORT_Modify

表 4-10: Sync_Serial 用関数

4.11 Timer_Capture

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 に期間表示されます。

4.11.1 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 TMR ユニット 1 を設定します。TMR ユニット 1 は停止されており、カウント値は 0 にリセットされます。またスイッチコールバック関数 CB_SwitchPress と CB_SwitchRelease がセットされます。
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 usCount をリセットし、TMR 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 TMR タイマ割り込みによって変数 usCount 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 TMR タイマを停止させ、変数 usCount 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。

4.11.2 シーケンス

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

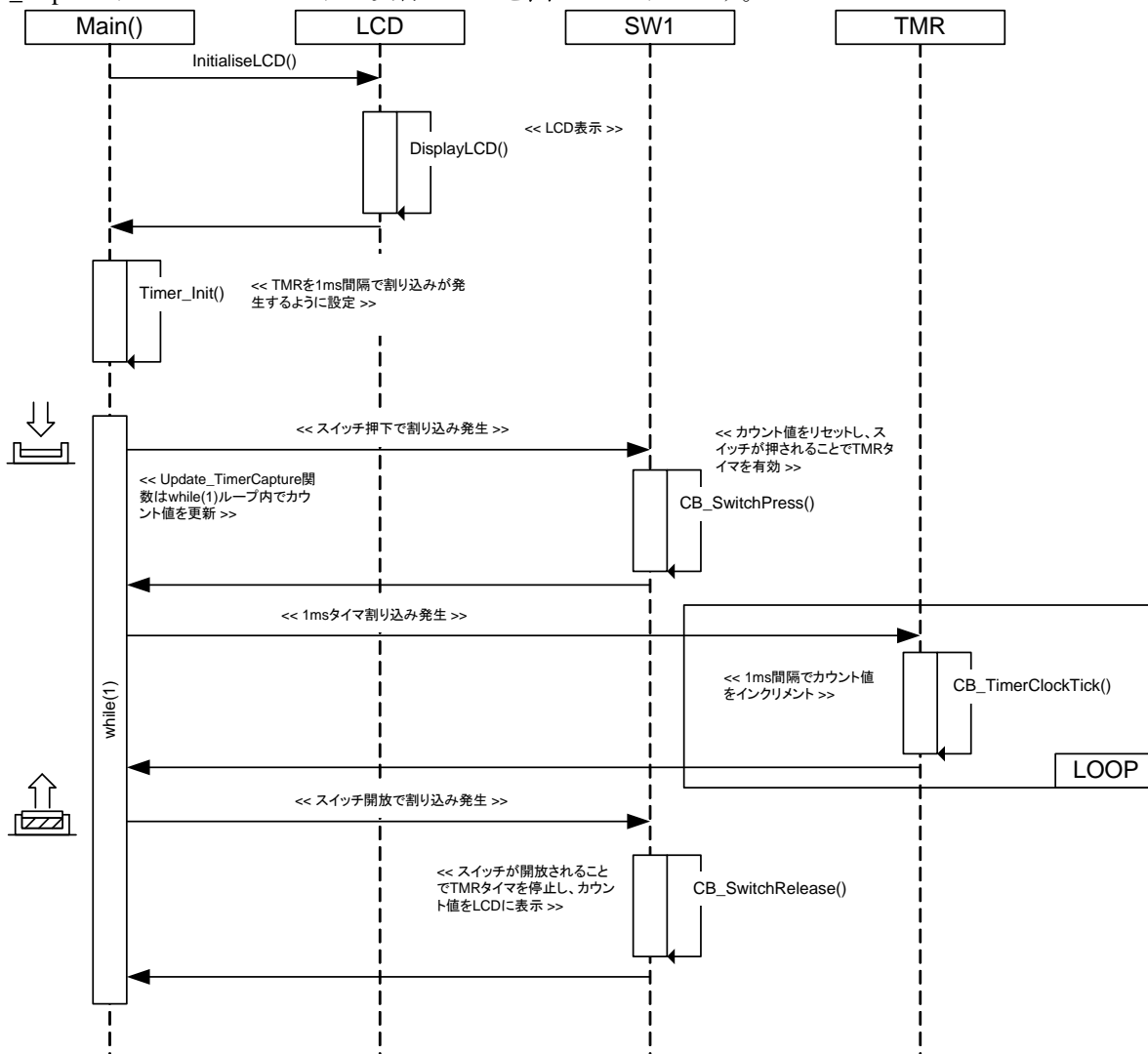


図 4-11: Timer_Capture フロー

4.11.3 RPDL

Timer_Capture 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
Timer_Init	R_TMR_CreatePeriodic
	R_TMR_ControlUnit
CB_SwitchPress	R_TMR_ControlUnit
	R_IO_PORT_Modify
CB_SwitchRelease	R_TMR_ControlUnit
	R_IO_PORT_Modify

表 4-11: Timer_Capture 用関数

4.12 Timer_Compare

本サンプルコードは TMR 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。

4.12.1 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 '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 TMR 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 TMR タイマ割り込みによってコールバック関数 CB_TMR 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。

4.12.2 シーケンス

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

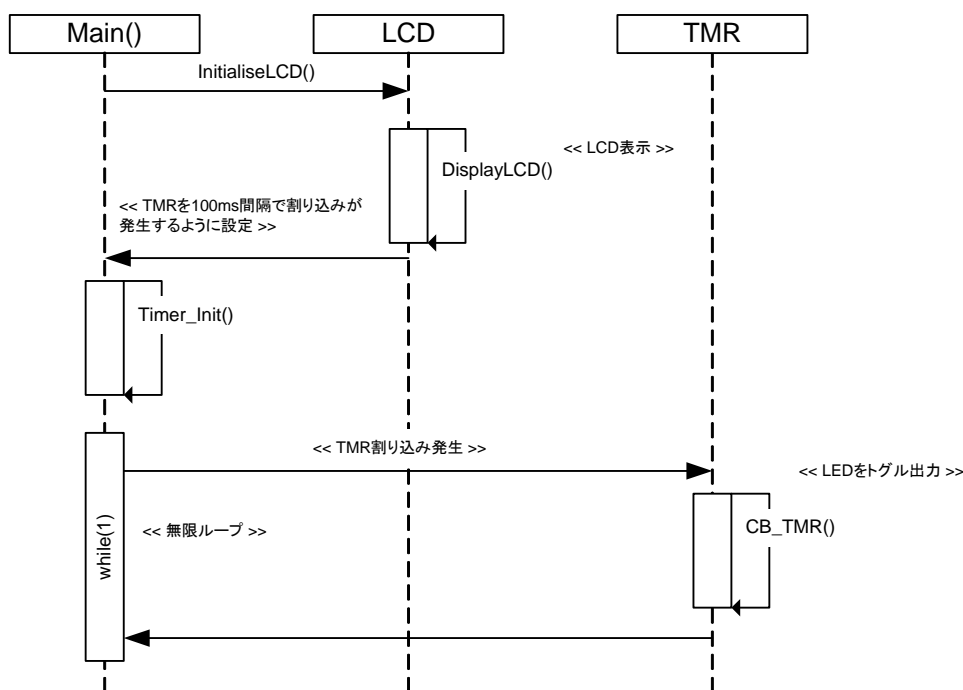


図 4-12: Timer_Compare フロー

4.12.3 RPDL

Timer_Compare 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
Timer_Init	R_TMR_CreateChannel
CB_TMR	R_IO_PORT_Modify

表 4-12: Timer_Compare 用関数

4.13 Timer_Event

本サンプルコードは TMR ユニットのチャンネル 2 を使ってスイッチ入力を計測します。

4.13.1 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 'Push SW2'、2 行目に 'Count: 0' 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 TMR ユニットのチャンネル 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 TMR 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 周期で割り込みが発生し、CB_TMR_UpdateLCD コールバック関数がコールされます。

4.13.2 シーケンス

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

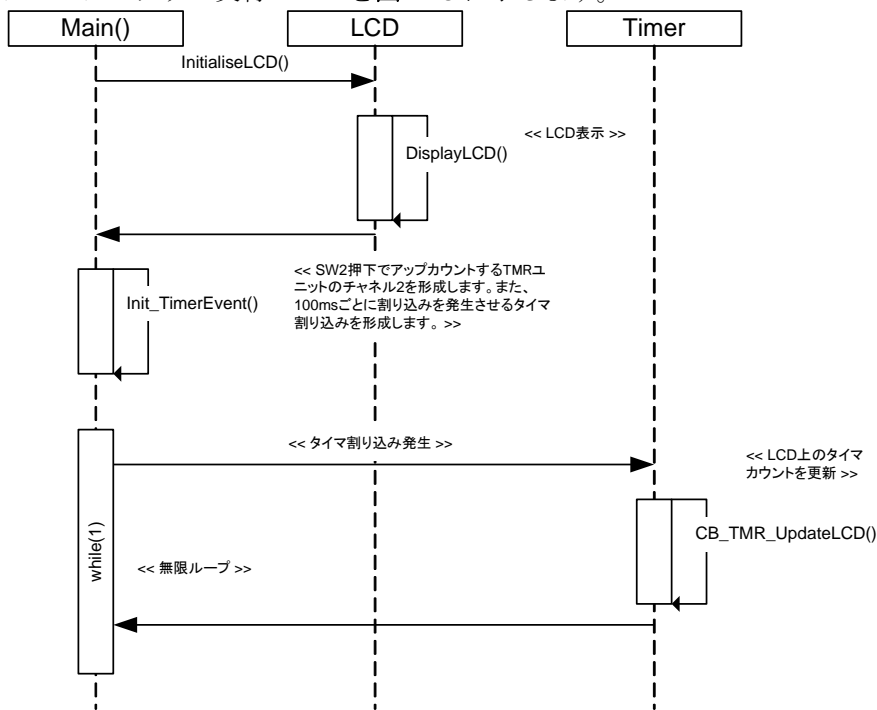


図 4-13: Timer_Event フロー

4.13.3 RPDL

Timer_Event 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_Timer	R_TMR_CreateChannel
	R_TMR_CreatePeriodic
TMR_callback	R_TMR_ReadChannel

表 4-13: Timer_Event 用関数

4.14 Timer_Mode

本サンプルコードはタイマ機能を使って 1kHz の周期的な波形を出力します。

4.14.1 オペレーション

1. LCD モジュールを初期化し、LCD の 1 行目に '1KHz'、2 行目に 'J2-pin20' を表示します。
2. その後、1kHz でデューティサイクル 50% の周期的な方形波を出力するために TMR ユニットのチャンネル 3 を形成します。タイマ出力は LED0 上でも観測することができます。
3. サンプルコードは無限ループに入り、タイマによって周期的な波形を出力し続けます。

4.14.2 シーケンス

Timer_Mode のプログラム実行フローを図 4-14 示します。

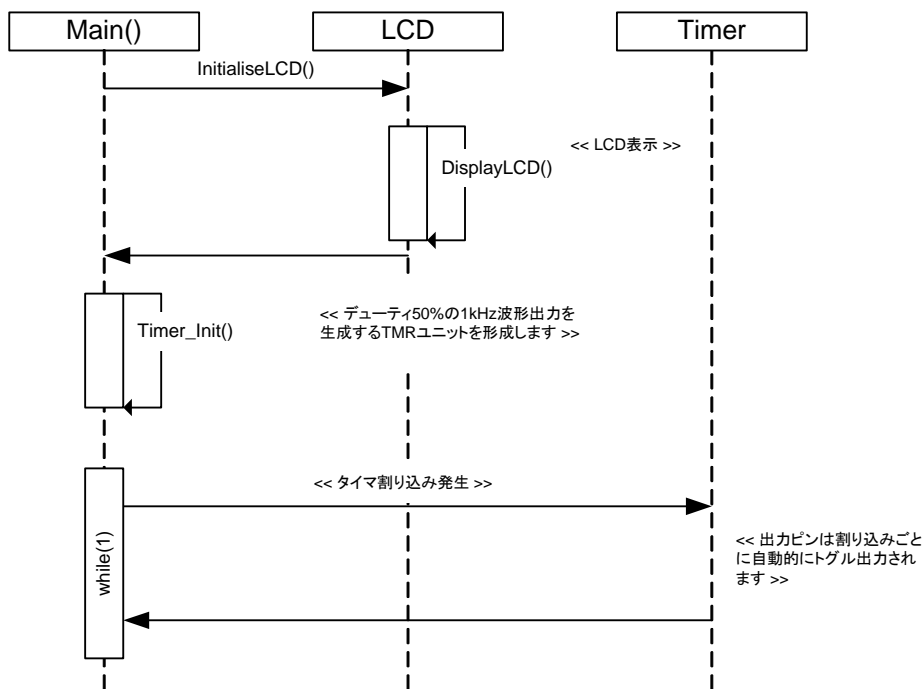


図 4-14: Timer_Mode フロー

4.14.3 RPD L

Timer_Mode で使用される関数、RPDL 関数を表 4-14 示します。

関数	RPDL 関数
Timer_Init	R_TMR_CreatePeriodic
Timer_callback	R_IO_PORT_Modify

表 4-14: Timer_Mode 用関数

4.15 WDT

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タイマ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。タイマ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。

4.15.1 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_WDT` 関数をコールし、周期的な割り込みを発生させる TMR、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_StartADC` が実行されます。
4. コールバック関数 `CB_WDT_Reset` 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、変数 `TMR_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_StartADC` 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”WDT 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。

4.15.2 シーケンス

WDT サンプルのプログラム実行フローを図 4-15 に示します。

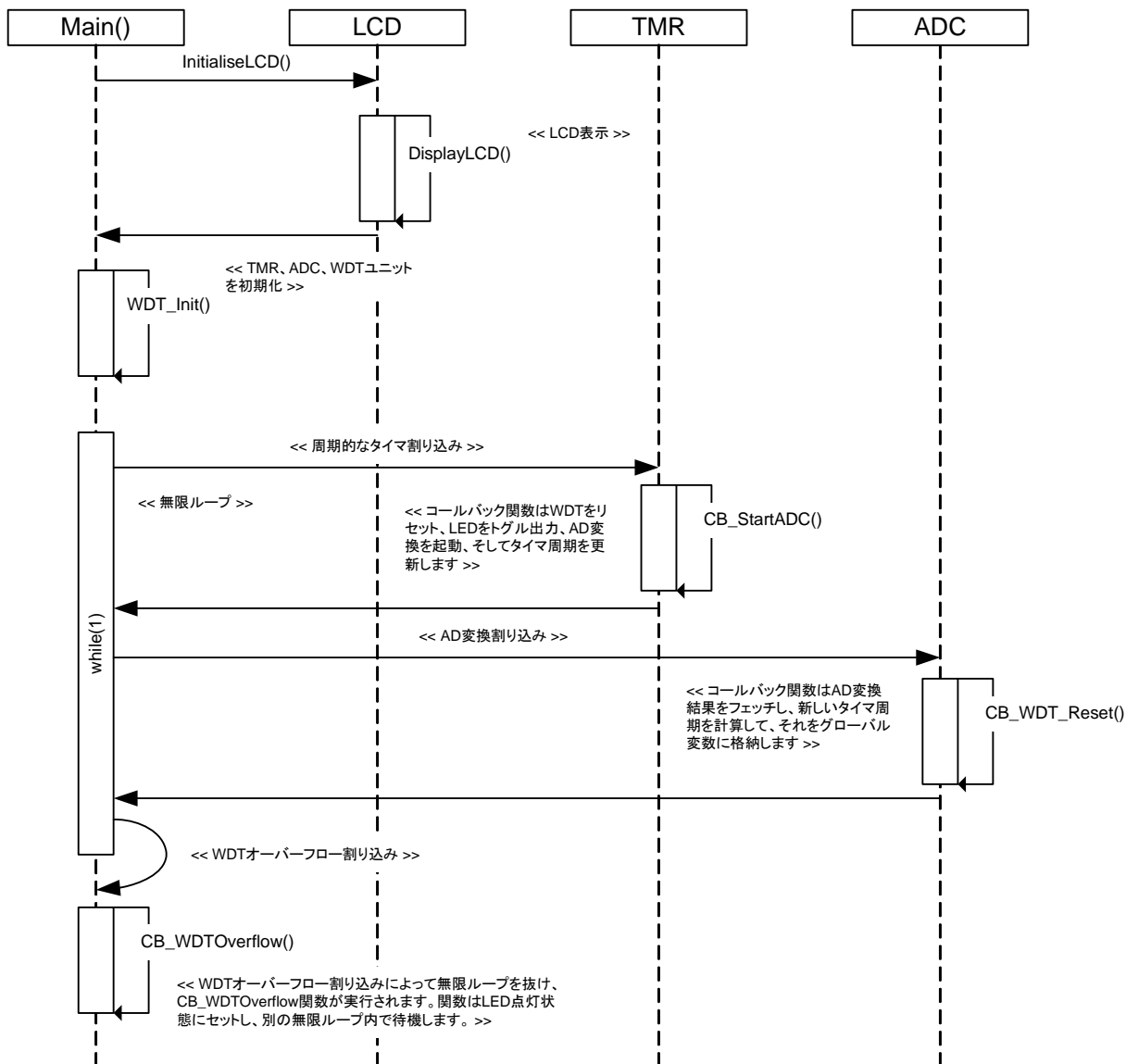


図 4-15: WDT フロー

4.15.3 RPDL

WDT 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
WDT_Init	R_TMR_CreateUnit
	R_WDT_Create
	R_ADC_10_Create
CB_StartADC	R_ADC_10_Control
CB_WDT_Reset	R_WDT_Control
	R_ADC_10_Read
	R_IO_PORT_Modify
	R_TMR_ControlUnit
CB_WDTOverflow	R_IO_PORT_Modify

表 4-15: WDT 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。

RX610 マイクロコントローラに関する詳細情報は、RX610 グループハードウェアマニュアル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 ファミリコンパイラパッケージユーザズマニュアル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/renesas_starter_kits (日本サイト)
http://www.renesas.com/renesas_starter_kits (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

本書の内容の一部または全てを予告無しに変更することがあります。
本書の著作権はルネサス エレクトロニクス株式会社にあり、ルネサス エレクトロニクス株式会社の書面での承諾無しに、本書の一部または全てを複製することを禁じます。

© 2011 Renesas Electronics Corporation. All rights reserved.
© 2011 Renesas Electronics Europe Ltd. All rights reserved.
© 2011 Renesas Solutions Corporation. All rights reserved.

改訂記録	RSKRX610 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	2010.03.19	－	初版発行
2.00	2011.02.07	－	ドキュメント番号、ドキュメントスタイルを更新

RSKRX610 ソフトウェアマニュアル

発行年月日 2011年2月7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/inquiry>

RX610 グループ