EV-200-USB-3
（ RL78/I1B 評価キット ）
ユーザーズ・マニュアル
ソフトウェア編
第１版

※ 本ユーザーズ・マニュアル（以下本書）に記載の全ての情報は発行時点のものであり、
予告なしに仕様を変更することがあります。

株式会社 内藤電誠町田製作所

目次

1.　はじめに

　この度は、EV-200-USB-3（RL78/I1B評価キット）をご購入いただき誠にありがとうございました。

本書は、EV-200-USB-3（RL78/I1B評価キット）のソフトウェアについて説明します。「ユーザーズ・マニュアル　ハードウェア編」を十分に理解したうえでお読み下さい。
以下、EV-200-USB-3（RL78/I1B評価キット）は「本製品」と記述します。
ご使用前に本書をよくお読みのうえ、正しく使用して下さい。

本製品は、ルネサスエレクトロニクス株式会社製 電力メーター/計測機器向けマイコンRL78/I1Bを搭載した評価キットです。

また、関連文書として以下に示した資料を使用します。

関連文書	文書番号
EV-200-USB-3(RL78/I1B評価キット) ユーザーズ・マニュアル ハードウェア編	SJAA-150008
EV-200-USB-3(RL78/I1B評価キット) ユーザーズ・マニュアル 拡張シリアルI/F　コマンド仕様編	SJAA-150011
EV-200-USB-3(RL78/I1B評価キット) ユーザーズ・マニュアル 電力測定用アプリケーション編	SJAA-150012
EV-200-USB-3(RL78/I1B評価キット) アプリケーションノート	SJAA-150010

下記の注意事項をご確認のうえ、ご使用下さい。

⚠ 注意

- **本ソフトウェアに関する財産権、所有権、知的財産権、その他一切の権限は弊社に帰属します。**

- **本ソフトウェアに関して弊社に断わりなく複製、他のソフトウェアへの流用、第三者に開示等しないようお願い致します。**

- **本ソフトウェアは無償で提供されますが、なんの欠陥もないという無制限の保証を行うものではありません。**

- **本書の内容を参照される場合は、お客様の責任において行って下さい。**
  ご使用によって生じたお客様または第三者の損害に関し、弊社は一切その責任を負いません。

- **第三者の特許件、著作権その他の知的財産の侵害などに関し、弊社はその責任を一切負いません。**

- **本書に記載された内容は万全を期して慎重に作成していますが、誤りがないことを保証するものではありません。**
  本書内容の誤りによってお客様に生じた損害においても、弊社は一切その責任を負いません。

- **お客様がWEBサイトからコンテンツをダウンロードすることにより、お客様のコンピュータ、またはネットワーク環境等に支障･障害が生じた場合、弊社はいかなる理由によるものでも一切責任を負いません。また、これらの事象によって生じた損害等についても、弊社は一切責任を負いません。**

- **弊社WEBサイトの利用はお客さまの責任において行われるものとします。**

- **弊社WEBサイトのコンテンツの作成および配信に関与する者は、当サイトの利用によって生じたあらゆる損害に対して、一切の責任を負いません。**

2.　ソフトウェア構成

2.1. 機能概要

本ソフトウェアの機能概要を以下に示します。

１）　電力測定機能

有効電力、無効電力、皮相電力、電圧、電流を測定します。また、過電圧、過電流の検出機能も有しています。

２）　ユーザプログラムI/F機能

ユーザプログラムを実装するためのI/F関数が用意されています。

３）　LCD表示機能

測定した電力値の表示等に使用できるLCD表示機能を備えています。

４）　スイッチ入力機能

LCDの表示切替等に使用できるスイッチ入力機能を備えています。

５）　磁気センサ機能

磁気センサにより磁気異常を検出できます。

６）　ケース開放検出機能

ケース開閉検出スイッチによりケース開放を検出できます。

７）　スリープモード移行機能

AC電源の遮断を検出し、スリープモードへ移行することができます。

2.2. 構成図

図 2-1 に本ソフトウェアの構成図を示します。

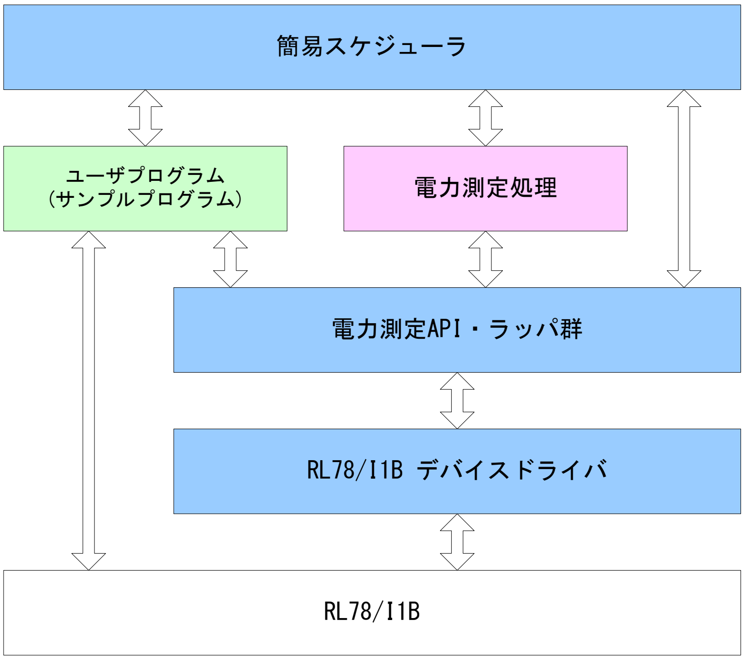


図 2-1 ソフトウェア構成図

１）　簡易スケジューラ

電力測定処理とユーザプログラムの I/F 関数を順番に呼び出します。
詳細は 2.2. 簡易スケジューラを参照して下さい。

２）　電力測定処理

RL78/I1B の 24 ビット⊿ΣADC の測定結果を基に電力の計算を行います。

３）　ユーザプログラム(サンプルプログラム)

ユーザ向けのプログラムを実装する I/F 関数です。サンプルプログラムとして
コンソールプログラムとスイッチ/LCD 操作が実装されています。

４）　電力測定 API・ラッパ群

ユーザプログラムからの電力測定開始操作やパラメータ設定、電力測定処理が
行った電力計算の結果取得等の機能を提供する API(以下、電力操作 API と記述
します)と RL78/I1B デバイスドライバのラッパ関数群です。

電力操作 API は、グローバル変数とグローバル変数を操作するためのマクロという形式で提供しています。

５）　RL78/I1B デバイスドライバ

RL78/I1B の周辺機能の初期化および操作を行うプログラムです。
基本的に CS+のコード生成機能を用いて自動生成された関数を使用しています。

2.3. RAM 空間のマッピング

図 2-2 に本ソフトウェアの RAM 空間のメモリマッピングを示します。

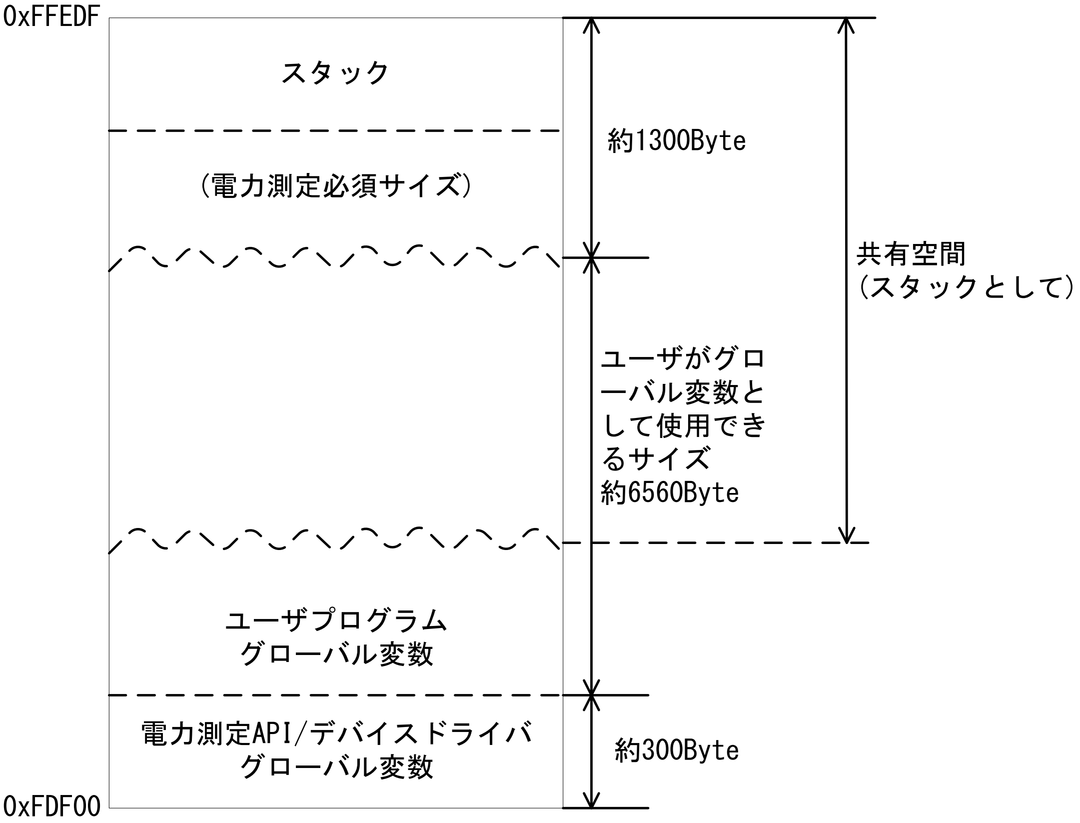


図 2-2 RAM 空間メモリマッピング

電力測定処理で使用する電力測定 API のグローバル変数の領域と電力測定に必要な最低限の大きさのスタック領域以外に関しては、ユーザが自由に使用できます。
ただし、ユーザプログラムで使用するグローバル変数領域をスタック領域が上書きしないように注意して下さい。
また、ユーザが使用できる RAM サイズの数値は目安です。コンパイル環境によって変化する可能性があります。

2.4. ユーザ用 I/F 関数

本ソフトウェアにユーザが任意の処理を追加する場合は、ユーザ用の I/F 関数に処理を追加します。
表 2-1 にユーザ用 I/F 関数の一覧を示します。

表 2-1 ユーザ用 I/F 関数一覧

処理	関数仕様	用途
ユーザ初期化処理 I/F	void em_user_init(void)	ユーザプログラム用の初期化処理を記述
ユーザプログラム I/F1	void em_user_program1(unsigned char gs)	電力測定処理前に実行するユーザプログラムを記述
ユーザプログラム I/F2	void em_user_program2(unsigned char gs)	
ユーザプログラム I/F3	void em_user_program3(unsigned char gs)	
ユーザプログラム I/F4	void em_user_program4(unsigned char gs)	電力測定処理後に実行するユーザプログラムを記述
ユーザプログラム I/F5	void em_user_program5(unsigned char gs)	
ユーザプログラム I/F6	void em_user_program6(unsigned char gs)	
電力測定開始時コールバック I/F	void em_user_start_measure(void)	電力測定処理開始直前に実行する処理を記述
WDT 処理 I/F	void em_user_wdt(void)	WDT タイムアウト発生時の処理を記述

また、他にウォッチドッグ・タイマのカウンタをクリアする関数も提供します。

表 2-2 ユーザ用関数

処理	関数仕様	用途
WDT クリア	void em_clear_wdt(void)	WDT のカウンタをクリアする

2.5. 状態遷移図

図 2-3 に本ソフトウェアの状態遷移図を示します。

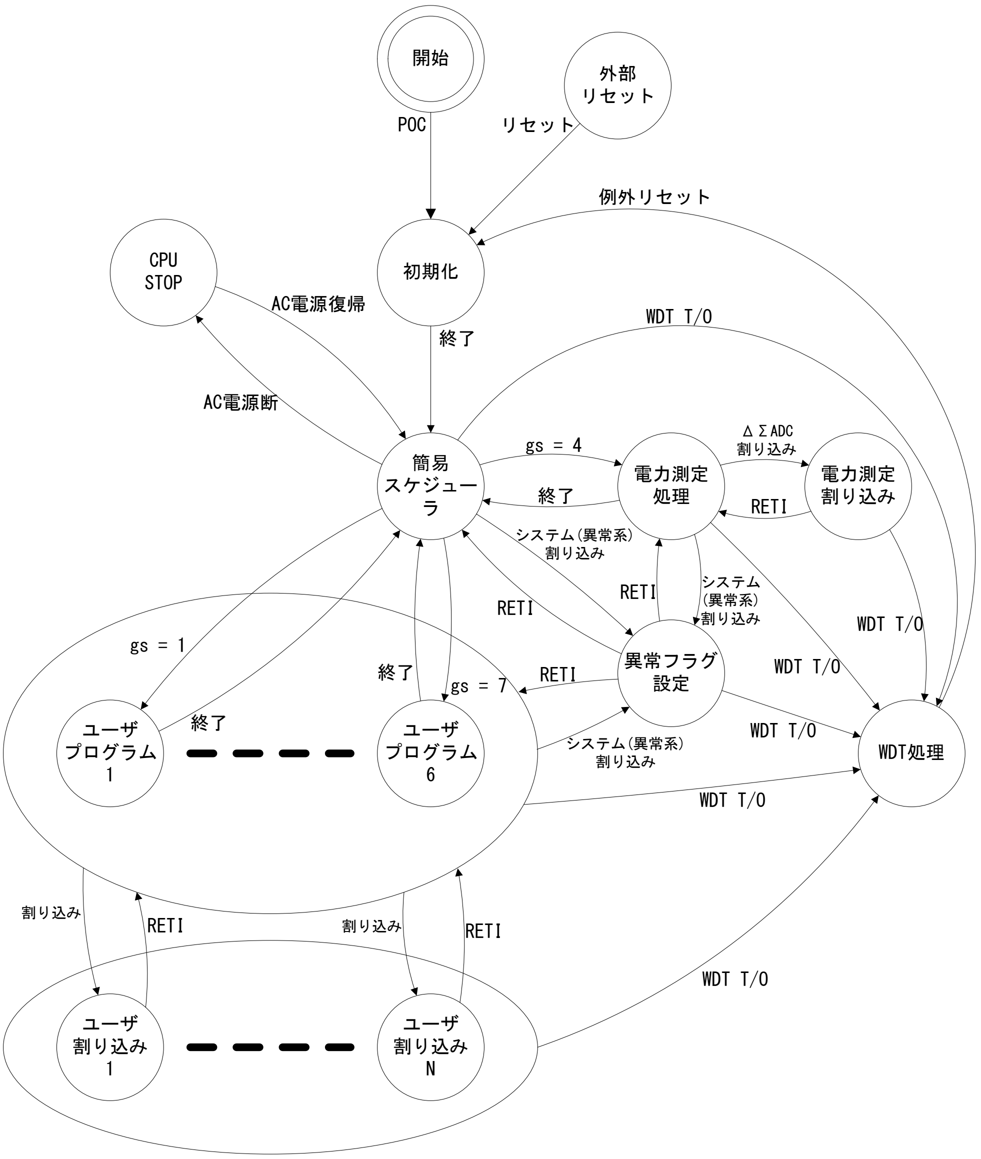


図 2-3 状態遷移図(全体)

パワーオンリセットの後、本ソフトウェアは初期化処理へと遷移します。
初期化処理ではユーザ初期化処理 I/F 関数が呼ばれます。
その後、簡易スケジューラへと遷移し、ユーザプログラム I/F 関数 1～6 が排他的に呼び出されます。
ウォッチドッグ・タイマのタイムアウトが発生すると、WDT 処理へ遷移します。
WDT 処理では WDT 処理 I/F 関数が呼び出された後、初期化処理へと遷移します。
また、AC 電源が遮断された場合は CPU STOP へ遷移しスリープモードに移行します。
この状態で再び AC 電源が供給されると簡易スケジューラへ遷移し、ユーザプログラム I/F 関数の呼び出しが再開されます。

2.6. 初期化

図 2-4 に初期化処理のフローチャートを示します。

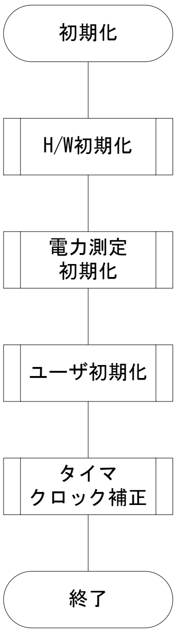


図 2-4 初期化処理フローチャート

１） H/W 初期化

RAM のクリアや RL78/I1B デバイスドライバによる RL78/I1B 周辺機能の初期化を行います。

２） 電力測定初期化

電力測定 API の初期化を行います。

３） ユーザ初期化

このユーザ初期化でユーザ初期化処理 I/F 関数が呼び出されますので、ユーザ初期化処理 I/F 関数にユーザプログラムに必要な初期化処理を記述して下さい。
オプションバイトの設定で高速オンチップ・オシレータ・クロックは 24MHz を初期値としていますが、ユーザ初期化処理で設定の変更をすることができます。ユーザ初期化処理 I/F 関数でクロック設定を変更した場合、電力測定処理で使用するタイマの設定値は次のタイマクロック補正で適切な値に自動調整されます。

４）　タイマクロック補正

電力測定 API で使用するタイマの設定値を、現在設定されているクロック周波数で適切に動作するように調整します。
ユーザ初期化でクロック設定を変更した場合でも、このタイマクロック補正によってタイマの設定値が適切な値に調整されますので、特にユーザ側で修正をする必要はありません。
また、ユーザ初期化処理 I/F 関数以外(ユーザプログラム 1～6 等)で高速オンチップ・オシレータ・クロックを変更してもタイマクロックの補正は実行されませんので、注意して下さい。

2.7. 簡易スケジューラ

図 2-5 に簡易スケジューラのフローチャートを示します。

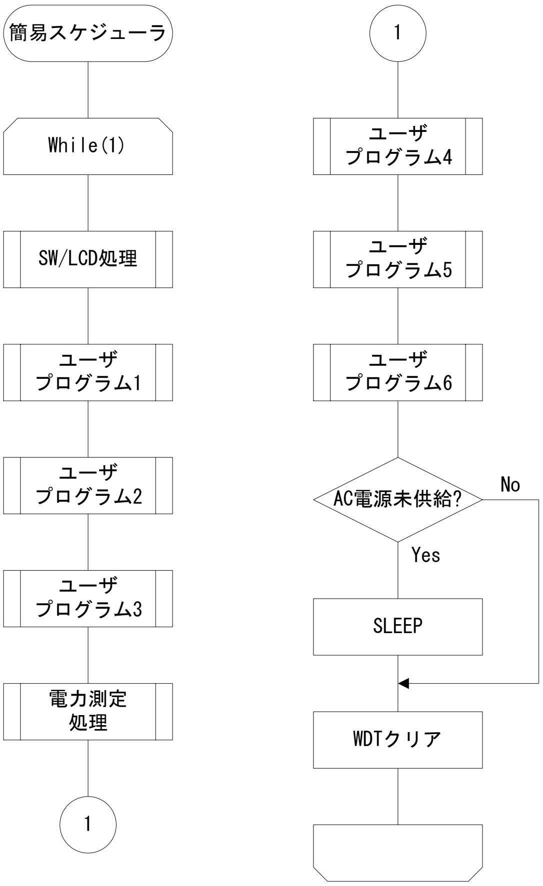


図 2-5 簡易スケジューラ フローチャート

１） SW/LCD 処理

サンプルプログラムのスイッチ/LCD 操作を呼び出ます。

２） ユーザプログラム 1～3

ユーザ用 I/F 関数 1～3 を呼び出します。ユーザ用 I/F 関数 3 にはサンプルプログラムのコンソールプログラム(コマンド受付部分)が実装されています。

３） 電力測定処理

電力操作 API を使用した電力測定処理を実行します。電力測定処理は電力測定開始フラグ[*1] をオンにすることで処理が開始されます。

*1 詳細は 4.ユーザ I/F API を参照。

４）　ユーザプログラム 4～6

ユーザ用 I/F 関数 4～6 を呼び出します。ユーザ用 I/F 関数 4 にはサンプルプログラムのコンソールプログラム(電力測定結果の出力部分)が実装されています。

５）　SLEEP 処理

AC 電源の供給状態を判定してスリープモードに入ります。スリープモードを使用する場合は、電池の接続が必要になります。詳細に関しては、EV-200-USB-3(RL78/I1B 評価キット) ユーザーズ・マニュアル ハードウェア編を参照して下さい。

６）　WDT クリア

ウォッチドッグ・タイマのクリアを行います。ウォッチドッグ・タイマのクリアは簡易スケジューラでも行っていますが、ユーザプログラム 1～6 で長時間の処理が必要なプログラムを実装する場合は、各々の処理内でウォッチドッグ・タイマのクリアを行って下さい。

2.8. 電力測定処理

図2-6に電力測定処理の状態遷移図を示します。

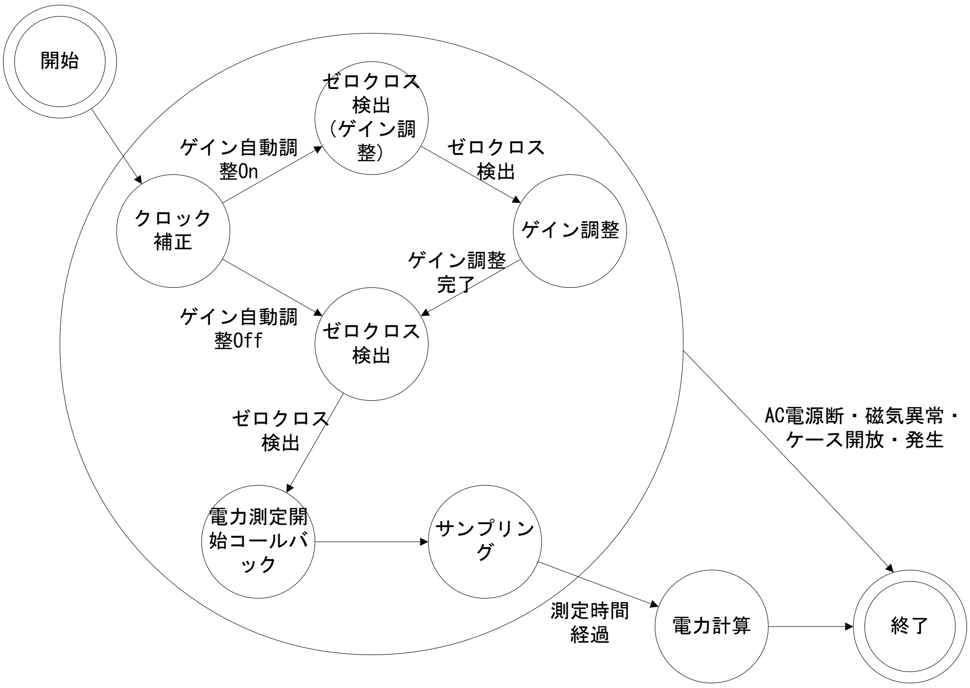


図 2-6 電力測定処理 状態遷移

1）　クロック補正

RL78/I1Bの高速オンチップ・オシレータ・クロック周波数補正機能を使用してクロック周波数の補正を行います

2）　ゼロクロス検出(ゲイン調整)

入力電圧、入力電流のゲイン設定がオートゲインモードに設定されている場合は、ゲイン調整処理(プリサンプリング処理)に遷移します。ゼロクロス検出(ゲイン調整)では、ゲイン調整処理を行う前にゼロクロスを検出します。

3）　ゲイン調整

ゲイン調整処理(プリサンプリング処理)を実行、適切なゲイン値を設定します。

4）　ゼロクロス検出

電力測定処理を行う前にゼロクロスを検出します。

５）　電力測定開始コールバック

ゼロクロス検出後、サンプリング開始前に電力測定開始コールバック I/F 関数が呼ばれます。この電力測定開始コールバック I/F 関数にユーザが処理を記述することで、正確なサンプリング開始時間の取得を行うことができます。

６）　サンプリング

RL78/I1B の⊿ΣADC を用いて入力電圧と入力電流のサンプリングを行います。

７）　電力計算

入力電圧と入力電流のサンプリング結果を基に電力の計算を行います。

８）　終了

電力測定処理中の割り込みについては、電力測定に使用する割り込みを除き、全てマスクされます。電力測定中にマスクされた割り込みが発生した場合は、電力測定処理終了時に割り込み優先度に従って順次割り込みが発生します。

2.9. 異常系処理

電力測定処理中に異常が発生した場合は、発生した異常に対応したエラーフラグ[*1]が設定され、電力測定処理が中断されます。
表 2-3 にエラーフラグの一覧を示します。

表 2-3 エラーフラグ

フラグ値	意味
0x0000 0000	正常
0x0000 0001	ゼロクロス未検出
0x0000 0002	過電圧検出
0x0000 0004	過電流検出
0x0000 0008	AC 電源断検出
0x0000 0010	電力測定異常検出
0x0000 0020	磁気センサ異常検出
0x0000 0040	ケース開放検出

*1　　詳細は 4. ユーザ I/F API 参照

1） ゼロクロス未検出

図 2-7 にゼロクロス検出のイメージ図を示します。

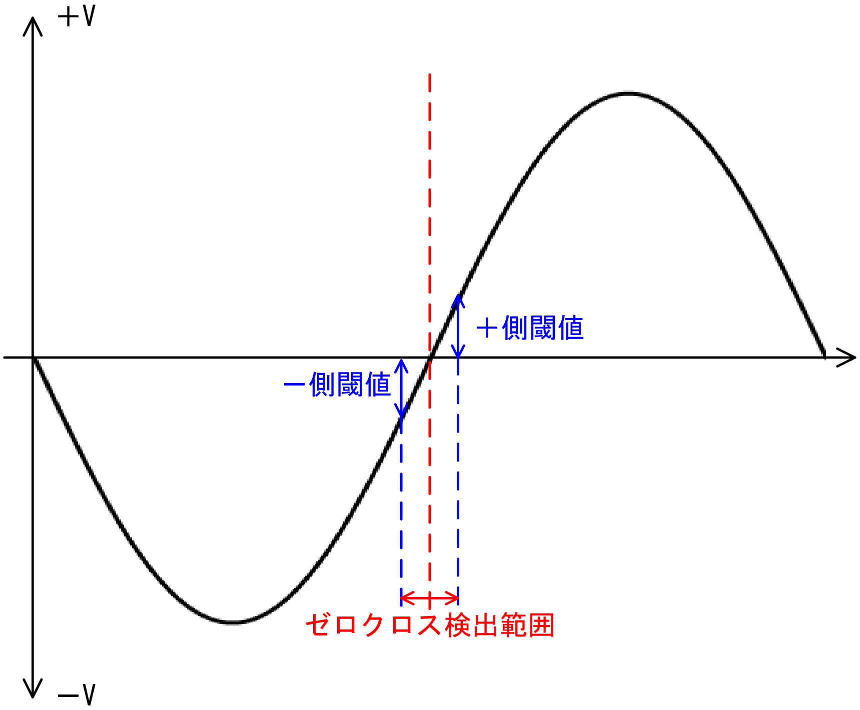


図 2-7 ゼロクロス検出

電力測定のサンプリングを実行する前に、電圧のゼロクロスを検出します。
本ソフトウェアでは、以下の二つの条件を充たしたときにゼロクロス検出としています。

・⊿ΣADC の電圧のサンプリング値が－から＋に変化
・⊿ΣADC の電圧のサンプリング値がゼロクロストリガ閾値*1 以下
(閾値は＋側、－側の双方に摘要されます。)

以下にゼロクロス検出の例を示します。

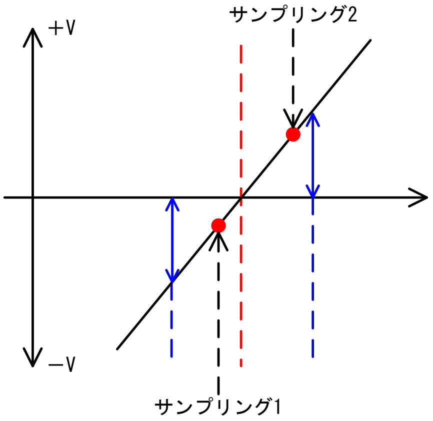


図 2-8 では⊿ΣADC の電圧のサンプリング値であるサンプリング 1、サンプリング 2 が両方とも閾値以下となりゼロクロス検出範囲に入っています。この場合は、より 0V に近いサンプリング 1 をゼロクロスとして検出します。

図 2-8 ゼロクロス検出例 1

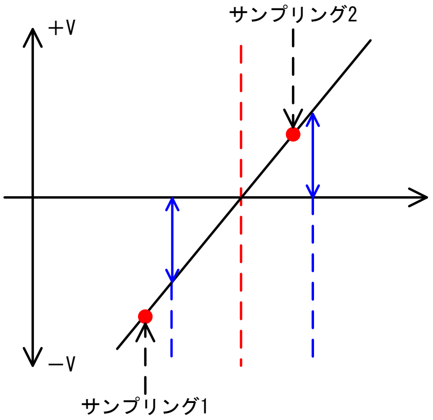


図 2-9 ではサンプリング 1 がー側閾値外、サンプリング 2 が＋側閾値以下なので、サンプリング 2 をゼロクロスとして検出します。

図 2-9 ゼロクロス検出例 2

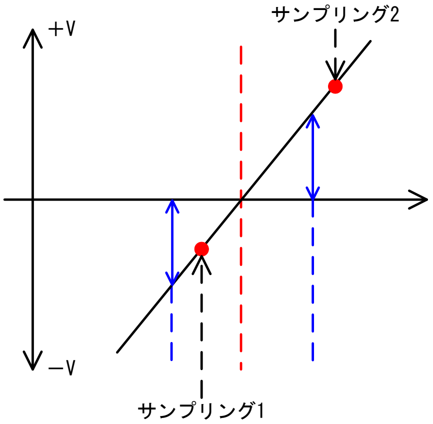


図 2-10 ではサンプリング 1 がー側閾値以下、サンプリング 2 が＋側閾値外なので、サンプリング 1 をゼロクロスとして検出します。

図 2-10 ゼロクロス検出例 3

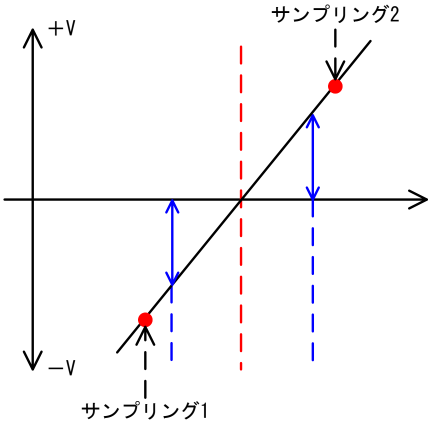


図 2-11 ではサンプリング 1、サンプリング 2 が両方とも閾値外なので、ゼロクロスは検出されません。

図 2-11 ゼロクロス検出例 4

図 2-11 のような状態が続き、ゼロクロスが 200ms 間検出できなかった場合は、ゼロクロス未検出となり、電力測定処理が中断されます。

２）　過電圧検出

電力測定中に過電圧閾値[*1] 以上の電圧を検出した場合、過電圧検出となり、電力測定が中止されます。閾値の値は実効値ではなく瞬時値です。

３）　過電流検出

電力測定中に過電流閾値[*1] 以上の電流を検出した場合、過電流検出となり、電力測定が中止されます。閾値の値は実効値ではなく瞬時値です。

４）　AC 電源断検出

電力測定中に AC 電源が遮断された場合、AC 電源遮断検出となり、電力測定が中止されます。また、電池を接続していない場合はこの機能は使用できません。

５）　電力測定異常検出

何らかの原因で⊿Σ ADC の動作に異常が生じた場合、電力測定異常となり、電力測定が中止されます。具体的には、⊿Σ ADC の測定結果が入力電圧の許容値を超えた場合や、割り込みが一度も発生せずに測定時間が経過してしまった場合を電力測定異常のエラーと定義しています。

６）　磁気センサ異常検出

電力測定中に磁気センサの異常を検出した場合、磁気センサ異常検出となり、電力測定が中止されます。また、電力測定を実行していない状態でも磁気センサ異常は検出されます。この場合は、磁気センサが正常な状態に戻るまで電力測定処理は実行できません。

７）　ケース開放検出

電力測定中にケース開放を検出した場合、ケース開放検出となり、電力測定が中止されます。また、電力測定を実行していない状態でもケース開放は検出されます。この場合は、ケースが正常な状態に戻るまで電力測定処理は実行できません。

８）　ウォッチドッグ・タイマのタイムアウト

この異常状態は電力測定処理中のみに発生するものではありません。
ソフトウェアが暴走し、ウォッチドッグ・タイマのタイムアウトが発生した場合、WDT 処理 I/F 関数が呼ばれ、その後に不正命令実行の割り込みを利用して H/W 初期化処理に制御を移します。また、電力測定操作 API で取得できるリセット要因[*1]に『ウォッチドッグ・タイマによるリセット』が設定されます。ユーザ初期化処理 I/F 関数でリセット要因を取得し、適切な初期化を行って下さい。

*1　詳細は 4. ユーザ I/F API 参照

3. セットアップ

3.1. 開発環境

表 3-1 に、本ソフトウェアの開発環境について記述します。

表 3-1 開発環境

開発環境	
統合開発環境	CS+ for CC (V5.01.00.06 以上)
C コンパイラ	CC-RL (V1.02.00.07 以上)
コードライブラリ	RL78/I1B コードライブラリ (V1.02.02.01 以上)
デバッグツール	E1 エミュレータ (V3.01.00.06 以上)

開発言語	
使用言語	C 言語 (一部にアセンブラを使用)
その他	CS+ for CC のコード生成機能を使用

本ソフトウェアを使用するためには CS+と E1 エミュレータが必要になります。CS+は表 3-1 に示したバージョン以上のものを使用して下さい。また、本ソフトウェアは CS+の無償版でもコンパイルできる事を確認しています。

E1 エミュレータ：ルネサスエレクトロニクス株式会社（http://japan.renesas.com/）製オンチップデバッギングエミュレータです。尚、購入する際は購入元にお問い合わせ下さい。
CS+：ルネサスエレクトロニクス株式会社（http://japan.renesas.com/）製統合開発環境です。尚、購入する際は購入元にお問い合わせ下さい。

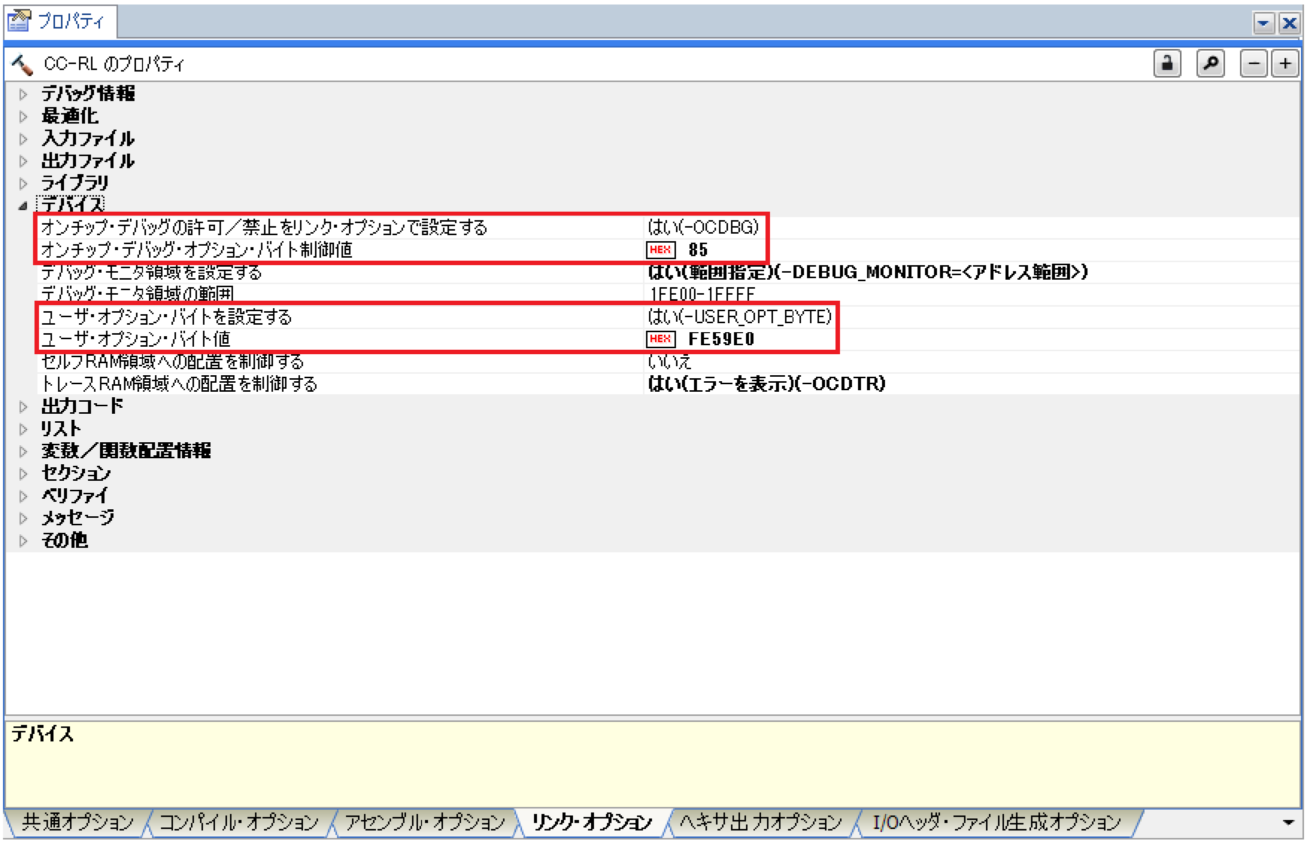
CS+の PC へのインストールおよび E1 エミュレータのセットアップ、基本的な使用方法に関しては、それぞれのマニュアルを参照して下さい。

3.2. 書き込み・デバッグのための設定

本ソフトウェアは弊社HP(http://sys.ndk-m.com)からダウンロードが可能です。
CS+の『既存のプロジェクトを開く』からダウンロードした本ソフトウェアのプロジェクトファイルを選択し、プロジェクトを開いて下さい。
本ソフトウェアをビルドし、本製品へ書き込み・デバッグするためには、オプション・バイトの設定とE1エミュレータの電源供給の設定が必要になります。

1)　オプションバイトの設定

CS+の『プロジェクトツリー』から『CC-RL(ビルド・ツール)』をダブルクリックし、CC-RLのプロパティを表示させて下さい。表示されたCC-RLのプロパティから『リンク・オプション』のタブを選択します。『リンク・オプション』タブ内の『デバイス』を開きます。



オンチップ・デバッグの許可／禁止をリンク・オプションで設定する	はい(-OCDBG)
オンチップ・デバッグ・オプション・バイト制御値	HEX 85
デバッグ・モニタ領域を設定する	はい(範囲指定)(-DEBUG_MONITOR=<アドレス範囲>)
デバッグ・モニタ領域の範囲	1FE00-1FFFF
ユーザ・オプション・バイトを設定する	はい(-USER_OPT_BYTE)
ユーザ・オプション・バイト値	HEX FE59E0
セルフRAM領域への配置を制御する	いいえ
トレースRAM領域への配置を制御する	はい(エラーを表示)(-OCDTR)

図 3-1 CS+ CC-RLのプロパティ

表 3-2 に示すように設定を行って下さい。

表 3-2 リンク・オプション デバイスの設定

設定項目	設定値	備考
オンチップ・デバッグの許可／禁止をリンク・オプションで設定する	はい(-OCDBG)	
オンチップ・デバッグ・オプション・バイト制御値	85	オプションバイトの 0x000C3 に該当
ユーザ・オプションバイトを設定する	はい(-USER_OPT_BYTE)	
オプション・バイト値	FE59E0[*1]	オプションバイトの 0x000C0～000C2 に該当

*1　オプションバイトのアドレス 0x000C1 に該当する 0x59 はデフォルトでは LVD オフの設定(0xFF)になっていますので、設定値は FEFFE0 となっています。

表 3-3 に本ソフトウェアのオプションバイトの設定を示します。

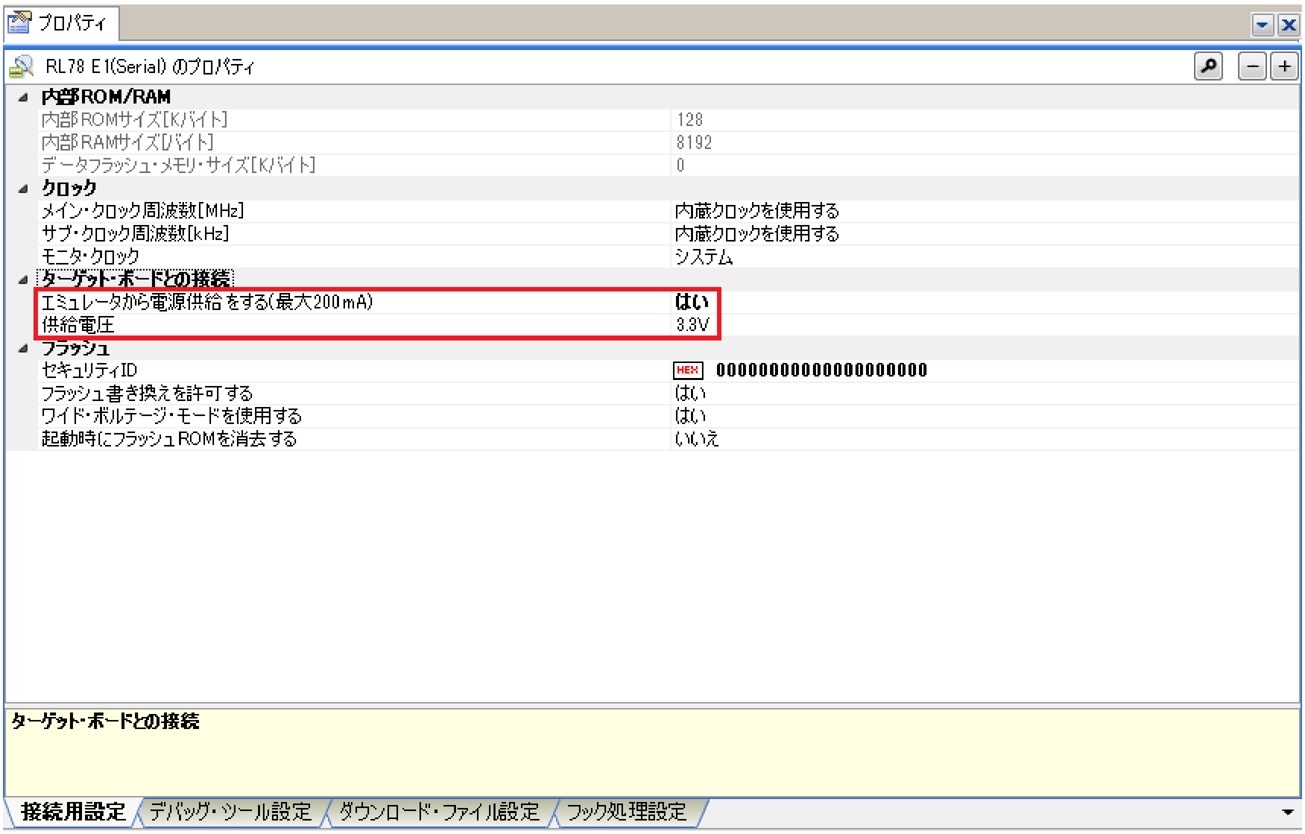
表 3-3 オプションバイト設定

アドレス	設定値	内容
0x000C0/0x010C0	0xFE	ウォッチドッグ・タイマ インターバル割り込み使用 ウインド・オープン期間 100% カウンタ動作許可 オーバーフロー時間 3799.18ms HALT/STOP モード時カウンタ動作停止
0x000C1/0x010C1	0x59	V_{LVDH} 立ち上がり電圧 2.61V V_{LVDH} 立ち下がり電圧 2.55V 割り込みモード
0x000C2/0x010C2	0xE0	フラッシュ動作 HS(高速メイン)モード 高速オンチップ・オシレータ・クロック 24MHz
0x000C3/0x010C3	0x85	オンチップ・デバッグ動作許可 セキュリティ ID 認証失敗時、フラッシュメモリのデータ消去はしない

デバッグ時にウォッチドッグ・タイマを無効にしたい場合は、アドレス 0x000C0 の設定値を 0xFE から 0xEE に変更して使用して下さい。
LVD の検出電圧 VLVDH および VLVDH は A/D コンバータを内部基準電圧で動作させることが出来る電圧値(2.4V 以上)を考慮して選択しています。また、選択している LVD の検出電圧値はバッテリ・バックアップ・モードへの切り替え検出電圧よりも高い電圧値であるため、LVD とバッテリ・バックアップ・モードの両方を使用する場合は注意して下さい。
オプションバイト設定に関する詳細は RL78/I1B のユーザーズマニュアルを参照して下さい。

２）　E1 エミュレータ電源供給設定

CS+の『プロジェクトツリー』から『RL78 E1(Serial)(デバッグ・ツール)』をダブルクリックし、RL78 E1(Serial)のプロパティを表示させて下さい。もし、デバッグ・ツールに RL78 E1(Serial)が選択されていない場合は、『プロジェクトツリー』のデバッグ・ツールを右クリックしてメニューを表示させ、『使用するデバッグ・ツール』から『RL78 E1(Serial)』を選択した後にプロパティを表示させて下さい。表示された RL78 E1(Serial)のプロパティから『接続用設定』のタブを選択します。『接続用設定』タブ内の『ターゲット・ボードとの接続』を開きます。



内部ROM/RAM	
内部ROMサイズ[Kバイト]	128
内部RAMサイズ[バイト]	8192
データフラッシュ・メモリ・サイズ[Kバイト]	0
クロック	
メイン・クロック周波数[MHz]	内蔵クロックを使用する
サブ・クロック周波数[kHz]	内蔵クロックを使用する
モニタ・クロック	システム
ターゲット・ボードとの接続	
エミュレータから電源供給をする(最大200mA)	**はい**
供給電圧	3.3V
フラッシュ	
セキュリティID	HEX 00000000000000000000
フラッシュ書き換えを許可する	はい
ワイド・ボルテージ・モードを使用する	はい
起動時にフラッシュROMを消去する	いいえ

ターゲット・ボードとの接続

図 3-2 CS+ RL78 E1(Serial)のプロパティ

表 3-4 に示すように設定を行って下さい。

表 3-4 接続用設定 ターゲット・ボードとの接続設定

設定項目	設定値	備考
エミュレータから電源供給をする(最大 200mA)	はい	本製品と E1 エミュレータを接続するには、E1 エミュレータからの電源供給が必要です。
供給電圧	3.3V	

3.3. 書き込み・デバッグ

オプションバイトとE1エミュレータの電源供給設定が終了したら、本製品のE1 I/FにE1エミュレータを接続します。E1エミュレータの接続は本製品に電源を供給していない状態で行って下さい。E1エミュレータを接続後に本製品へ電源を供給します。
この状態で、本ソフトウェアをビルドし、CS+の『デバッグ』メニューから『デバッグツールへのダウンロード』を選択する事で、本製品へソフトウェアが書き込まれ、デバッグが可能になります。

3.4. サンプルプログラム

１） コンソールプログラム

拡張シリアルI/Fを介してパソコン等と接続し､コマンドを送信することで電力測定を実行するプログラムです。コマンド受信部分がユーザプログラムI/F3に、電力測定結果の出力部分がユーザプログラムI/F4にそれぞれサンプルとして実装されています。
詳細に関しては、EV-200-USB-3(RL78/I1B評価キット)ユーザーズ・マニュアル コマンド仕様編を参照して下さい。

２） スイッチ/LCD操作

LCDおよびLCD操作スイッチを使用したメニュー操作により電力測定を実行するプログラムです。em_lcd_swという関数に実装されています。
詳細に関しては、EV-200-USB-3(RL78/I1B評価キット)ユーザーズ・マニュアル ハードウェア編を参照して下さい。

4.　ユーザI/F API

本ソフトウェアで提供する電力測定APIは電力測定処理を制御するためのグローバル変数とグローバル変数への設定を補助するマクロで構成されています。

4.1. グローバル変数一覧

表4-1にソフトウェアで使用するグローバル変数の一覧を示します。

表 4-1 変数一覧

変数名	型	概要
em_g_active_power	float	電力測定の結果(有効電力)
em_g_reactive_power	float	電力測定の結果(無効電力)
em_g_apparent_power	float	電力測定の結果(皮相電力)
em_g_volt	float	電力測定の結果(電圧)
em_g_current	float	電力測定の結果(電流)
em_g_error	unsigned int	エラーフラグ
em_g_sampling_freq	unsigned int	サンプリング周波数選択
em_g_measure_time	unsigned int	測定時間設定
em_g_current_device	unsigned char	電流測定デバイス選択
em_g_volt_gain	unsigned char	電圧ゲイン
em_g_current_gain_ct	unsigned char	電流(CT)ゲイン
em_g_current_gain_s	unsigned char	電流(シャント)ゲイン
em_g_dsadc_init_wait	unsigned int	⊿ΣADCセットアップ時間
em_g_zerocross_trigger	unsigned char	ゼロクロストリガ選択
em_g_zerocross_th	float	ゼロクロストリガ閾値
em_g_over_volt_th	float	過電圧閾値
em_g_over_current_th	float	過電流閾値
em_g_measure	unsigned char	電力測定開始フラグ
em_g_reset_state	unsigned char	リセット要因
em_g_sleep	unsigned char	スリープモードフラグ
em_g_magnetic	unsigned char	磁気センサフラグ
em_g_case_open	unsigned char	ケース開放フラグ
em_g_ac_return[8]	unsigned char	AC電源復帰フラグ
em_g_sw	unsigned char	LCD切替スイッチ押下フラグ

4.2. 設定用グローバル変数

電力測定処理の設定に用いるグローバル変数とその設定補助マクロを以下に示します。

[変数] unsigned int em_g_sampling_freq

概要	サンプリング周波数選択
説明	RL78/I1Bの⊿ΣADCのサンプリング周波数を設定します。 2[kHz]と4[kHz]から選択可能です。
数値	2000 ： 2[kHz] 4000 ： 4[kHz](初期値)
関連マクロ	EM_SAMPLING_FREQ_2K EM_SAMPLING_FREQ_4K
備考	なし

[マクロ名] EM_SAMPLING_FREQ_2K

概要	⊿ΣADCサンプリング周波数設定(2k)
説明	⊿ΣADCのサンプリング周波数を2kに設定します。
引数	なし
リターン値	なし
備考	em_g_sampling_freqに2000を設定

[マクロ名] EM_SAMPLING_FREQ_4K

概要	⊿ΣADCサンプリング周波数設定(4k)
説明	⊿ΣADCのサンプリング周波数を4kに設定します。
引数	なし
リターン値	なし
備考	em_g_sampling_freqに4000を設定

[変数] unsigned int em_g_measure_time

概要	測定時間設定
説明	電力測定処理の測定時間を設定します。単位は0.1[s]で、最大30.0[s]まで設定可能です。0を設定すると、1周期分のみ電力測定が行われます。
数値	0～300 (初期値 ： 10)
関連マクロ	なし
備考	なし

[変数] unsigned char em_g_current_device

概要	電流測定デバイス選択
説明	電力測定処理で電流を測定するデバイスを設定します。CT とシャント抵抗のどちらかを選択可能です。
数値	0x01 : CT(初期値)
	0x02 : シャント抵抗
関連マクロ	EM_DEVICE_CT
	EM_DEVICE_SHUNT
備考	なし

[マクロ名] EM_DEVICE_CT

概要	電流測定デバイス設定(CT)
説明	電流を測定するデバイスを CT に設定します。
引数	なし
リターン値	なし
備考	em_g_current_device に 0x01 を設定

[マクロ名] EM_DEVICE_SHUNT

概要	電流測定デバイス設定(シャント)
説明	電流を測定するデバイスをシャント抵抗に設定します。
引数	なし
リターン値	なし
備考	em_g_current_device に 0x02 を設定

[変数] unsigned char em_g_volt_gain

概要	電圧ゲイン
説明	RL78/I1B の⊿Σ ADC の電圧入力チャネルゲインを設定します。オートゲインモードに設定すると電力測定処理時に適切なゲインがその設定されます。
数値	0x00 : オートゲインモード(初期値)
	0x01 : 固定ゲインモード(1 倍)
	0x02 : 固定ゲインモード(2 倍)
	0x04 : 固定ゲインモード(4 倍)
	0x08 : 固定ゲインモード(8 倍)
	0x10 : 固定ゲインモード(16 倍)
関連マクロ	EM_VGAIN_AUTO
	EM_VGAIN_1
	EM_VGAIN_2
	EM_VGAIN_4
	EM_VGAIN_8
	EM_VGAIN_16
備考	なし

［マクロ名］ EM_VGAIN_AUTO

概要	電圧ゲイン設定(オート)
説明	電圧ゲインをオートゲインモードに設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x00 を設定

［マクロ名］ EM_VGAIN_1

概要	電圧ゲイン設定(1 倍)
説明	電圧ゲインを固定ゲインモード(1 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x01 を設定

［マクロ名］ EM_VGAIN_2

概要	電圧ゲイン設定(2 倍)
説明	電圧ゲインを固定ゲインモード(2 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x02 を設定

［マクロ名］ EM_VGAIN_4

概要	電圧ゲイン設定(4 倍)
説明	電圧ゲインを固定ゲインモード(4 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x04 を設定

［マクロ名］ EM_VGAIN_8

概要	電圧ゲイン設定(8 倍)
説明	電圧ゲインを固定ゲインモード(8 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x08 を設定

［マクロ名］ EM_VGAIN_16

概要	電圧ゲイン設定(16 倍)
説明	電圧ゲインを固定ゲインモード(16 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_volt_gain に 0x10 を設定

[変数] unsigned char em_g_current_gain_ct

概要	電流(CT)ゲイン
説明	RL78/I1Bの⊿ΣADCの電流入力(CT)チャネルゲインを設定します。オートゲインモードに設定すると電力測定処理時に適切なゲインがその設定されます。
数値	0x00 ： オートゲインモード(初期値) 0x01 ： 固定ゲインモード(1倍) 0x02 ： 固定ゲインモード(2倍) 0x04 ： 固定ゲインモード(4倍) 0x08 ： 固定ゲインモード(8倍) 0x10 ： 固定ゲインモード(16倍) 0x20 ： 固定ゲインモード(32倍)
関連マクロ	EM_CGAIN_CT_AUTO EM_CGAIN_CT_1 EM_CGAIN_CT_2 EM_CGAIN_CT_4 EM_CGAIN_CT_8 EM_CGAIN_CT_16 EM_CGAIN_CT_32
備考	なし

[マクロ名] EM_CGAIN_CT_AUTO

概要	電流(CT)ゲイン設定(オート)
説明	電流(CT)ゲインを自動に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ctに0x00を設定

[マクロ名] EM_CGAIN_CT_1

概要	電流(CT)ゲイン設定(1倍)
説明	電流(CT)ゲインを固定ゲインモード(1倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ctに0x01を設定

[マクロ名] EM_CGAIN_CT_2

概要	電流(CT)ゲイン設定(2倍)
説明	電流(CT)ゲインを固定ゲインモード(2倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ctに0x02を設定

[マクロ名] EM_CGAIN_CT_4

概要	電流(CT)ゲイン設定(4倍)
説明	電流(CT)ゲインを固定ゲインモード(4倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ctに0x04を設定

[マクロ名] EM_CGAIN_CT_8

概要	電流(CT)ゲイン設定(8 倍)
説明	電流(CT)ゲインを固定ゲインモード(8 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ct に 0x08 を設定

[マクロ名] EM_CGAIN_CT_16

概要	電流(CT)ゲイン設定(16 倍)
説明	電流(CT)ゲインを固定ゲインモード(16 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ct に 0x10 を設定

[マクロ名] EM_CGAIN_CT_32

概要	電流(CT)ゲイン設定(32 倍)
説明	電流(CT)ゲインを固定ゲインモード(32 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_ct に 0x20 を設定

[変数] unsigned char em_g_current_gain_s

概要	電流(シャント)ゲイン
説明	RL78/I1B の⊿ΣADC の電流入力(シャント抵抗)チャネルゲインを設定します。オートゲインモードに設定すると電力測定処理時に適切なゲインがその設定されます。
数値	0x00 : オートゲインモード(初期値) 0x01 : 固定ゲインモード(1 倍) 0x02 : 固定ゲインモード(2 倍) 0x04 : 固定ゲインモード(4 倍) 0x08 : 固定ゲインモード(8 倍) 0x10 : 固定ゲインモード(16 倍) 0x20 : 固定ゲインモード(32 倍)
関連マクロ	EM_CGAIN_S_AUTO EM_CGAIN_S_1 EM_CGAIN_S_2 EM_CGAIN_S_4 EM_CGAIN_S_8 EM_CGAIN_S_16 EM_CGAIN_S_32
備考	なし

[マクロ名] EM_CGAIN_S_AUTO

概要	電流(シャント)ゲイン設定(オート)
説明	電流(シャント)ゲインをオートゲインモードに設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x00 を設定

[マクロ名] EM_CGAIN_S_1

概要	電流(シャント)ゲイン設定(1 倍)
説明	電流(シャント)ゲインを固定ゲインモード(1 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x01 を設定

[マクロ名] EM_CGAIN_S_2

概要	電流(シャント)ゲイン設定(2 倍)
説明	電流(シャント)ゲインを固定ゲインモード(2 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x02 を設定

[マクロ名] EM_CGAIN_S_4

概要	電流(シャント)ゲイン設定(4 倍)
説明	電流(シャント)ゲインを固定ゲインモード(4 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x04 を設定

[マクロ名] EM_CGAIN_S_8

概要	電流(シャント)ゲイン設定(8 倍)
説明	電流(シャント)ゲインを固定ゲインモード(8 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x08 を設定

[マクロ名] EM_CGAIN_S_16

概要	電流(シャント)ゲイン設定(16 倍)
説明	電流(シャント)ゲインを固定ゲインモード(16 倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_s に 0x10 を設定

[マクロ名] EM_CGAIN_S_32

概要	電流(シャント)ゲイン設定(32倍)
説明	電流(シャント)ゲインを固定ゲインモード(32倍)に設定します。
引数	なし
リターン値	なし
備考	em_g_current_gain_sに0x20を設定

[変数] unsigned int em_g_dsadc_init_wait

概要	⊿ΣADCセットアップ時間
説明	RL78/I1Bの⊿ΣADCのセットアップ時間を設定します。単位は10[ms]です。
数値	初期値 : 15(150[ms])
関連マクロ	なし
備考	unsigned intの最大値まで設定可能です。

[変数] unsigned char em_g_zerocross_trigger

概要	ゼロクロストリガ選択
説明	ゼロクロス検出の方法をソフトウェアとハードウェアで選択できます。ハードウェア検出を選択した場合は、RL78/I1Bのコンパレータを用いたゼロクロス検出処理が実行されます。
数値	0x01 : ソフトウェア検出(初期値) 0x02 : ハードウェア検出
関連マクロ	EM_ZEROCROSS_SOFT EM_ZEROCROSS_HARD
備考	なし

[マクロ名] EM_ZEROCROSS_SOFT

概要	ゼロクロストリガ設定(ソフト)
説明	ゼロクロストリガをソフトウェアに設定に設定します。
引数	なし
リターン値	なし
備考	em_g_zerocross_triggerに0x01を設定

[マクロ名] EM_ZEROCROSS_HARD

概要	ゼロクロストリガ設定(ハード)
説明	ゼロクロストリガをハードウェアに設定に設定します。
引数	なし
リターン値	なし
備考	em_g_zerocross_triggerに0x02を設定

[変数] float em_g_zerocross_th

概要	ゼロクロストリガ閾値
説明	ゼロクロス検出の閾値を設定します。ゼロクロストリガ選択がソフトウェア検出の場合に設定値が有効になります。単位はVです。
数値	初期値 : 11.3
関連マクロ	なし
備考	なし

[変数] float em_g_over_volt_th

概要	過電圧閾値
説明	過電圧検出の閾値を設定します。単位はVです。実効値ではなく瞬時値を設定して下さい。
数値	初期値 : 382.0
関連マクロ	なし
備考	なし

[変数] float em_g_over_current_th

概要	過電流閾値
説明	過電流検出の閾値を設定します。単位はAです。実効値ではなく瞬時値を設定して下さい。
数値	初期値 : 22.0
関連マクロ	なし
備考	なし

[変数] unsigned char em_g_measure

概要	電力測定開始フラグ
説明	電力測定処理の開始/停止を設定します。『開始』に設定すると電力測定処理が実行されます。『停止』に設定すると電力測定処理が実行されません。また、電力測定処理が実行されると値が『停止』に設定されます。
数値	0x00 : 停止 (初期値) 0x01 : 開始
関連マクロ	EM_MEASURE_STOP EM_MEASURE_START
備考	なし

[マクロ名] EM_MEASURE_STOP

概要	電力測定処理停止
説明	電力測定処理の実行フラグを停止に設定します。
引数	なし
リターン値	なし
備考	em_g_measure に 0x00 を設定

[マクロ名] EM_MEASURE_START

概要	電力測定処理開始
説明	電力測定処理の実行フラグを開始に設定します。
引数	なし
リターン値	なし
備考	em_g_measure に 0x01 を設定

4.3. 電力測定結果確認用グローバル変数

電力測定処理の結果が格納されるグローバル変数を以下に示します。電力測定処理の結果を参照したい場合は、これらのグローバル変数を読み出して下さい。

[変数] float em_g_active_power

概要	電力測定の結果(有効電力)
説明	電力測定処理で測定された有効電力の値が設定されます。単位は W です。 有効電力の値が必要な場合にこの変数を直接読み出して下さい。
数値	初期値 : 0.0
関連マクロ	なし
備考	なし

[変数] float em_g_reactive_power

概要	電力測定の結果(無効電力)
説明	電力測定処理で測定された無効電力の値が設定されます。単位は var です。 無効電力の値が必要な場合にこの変数を直接読み出して下さい。
数値	初期値 : 0.0
関連マクロ	なし
備考	なし

[変数] float em_g_apparent_power

概要	電力測定の結果(皮相電力)
説明	電力測定処理で測定された皮相電力の値が設定されます。単位は VA です。 皮相電力の値が必要な場合にこの変数を直接読み出して下さい。
数値	初期値 : 0.0
関連マクロ	なし
備考	なし

[変数] float em_g_volt

概要	電力測定の結果(電圧)
説明	電力測定処理で測定された電圧の値が設定されます。単位は V です。 電圧の値が必要な場合にこの変数を直接読み出して下さい。
数値	初期値 : 0.0
関連マクロ	なし
備考	なし

[変数] float em_g_current

概要	電力測定の結果(電流)
説明	電力測定処理で測定された電流の値が設定されます。単位は A です。 電流の値が必要な場合にこの変数を直接読み出して下さい。
数値	初期値 : 0.0
関連マクロ	なし
備考	なし

4.4. 各種フラグ用グローバル変数

電力測定処理等で設定される各種フラグ用のグローバル変数とその確認用マクロを以下に示します。

[変数] unsigned int em_g_error

概要	エラーフラグ
説明	電力測定処理中に発生したエラーの要因がフラグとして設定されます。
数値	0x0000 0000 : 正常(初期値)
	0x0000 0001 : ゼロクロス未検出
	0x0000 0002 : 過電圧検出
	0x0000 0004 : 過電流検出
	0x0000 0008 : AC 電源断検出
	0x0000 0010 : 電力測定異常
	0x0000 0020 : 磁気センサ異常検出
	0x0000 0040 : ケース開放検出
関連マクロ	EM_IS_SUCCESS
	EM_IS_ZEROCROSS_ERR
	EM_IS_OVER_VOLT_ERR
	EM_IS_OVER_CURRENT_ERR
	EM_IS_AC_OFF_ERR
	EM_IS_DEVICE_ERR
	EM_IS_MAGNETIC_ERR
	EM_IS_CASE_OPEN_ERR
備考	エラーに関する詳細は 3.1.異常系処理を参照

[マクロ名] EM_IS_SUCCESS

概要	正常状態確認
説明	電力測定処理が正常終了したことを確認します。
引数	なし
リターン値	0 : エラー発生
	0 以外 : 正常終了
備考	em_g_error の値を確認

[マクロ名] EM_IS_ZEROCROSS_ERR

概要	ゼロクロス未検出確認
説明	ゼロクロス未検出が発生したことを確認します。
引数	なし
リターン値	0 : ゼロクロス未検出は発生していない
	0 以外 : ゼロクロス未検出発生
備考	em_g_error の値を確認

［マクロ名］EM_IS_OVER_VOLT_ERR

概要	過電圧検出確認
説明	過電圧検出が発生したことを確認します。
引数	なし
リターン値	0：過電圧検出は発生していない 0以外：過電圧検出発生
備考	em_g_errorの値を確認

［マクロ名］EM_IS_OVER_CURRENT_ERR

概要	過電流検出確認
説明	過電流検出が発生したことを確認します。
引数	なし
リターン値	0：過電流検出は発生していない 0以外：過電流検出発生
備考	em_g_errorの値を確認

［マクロ名］EM_IS_AC_OFF_ERR

概要	AC電源断確認
説明	AC電源断が発生したことを確認します。
引数	なし
リターン値	0：AC電源断は発生していない 0以外：AC電源断発生
備考	em_g_errorの値を確認

［マクロ名］EM_IS_DEVICE_ERR

概要	電力測定異常確認
説明	電力測定異常が発生したことを確認します。
引数	なし
リターン値	0：電力測定異常は発生していない 0以外：電力測定異常発生
備考	em_g_errorの値を確認

［マクロ名］EM_IS_MAGNETIC_ERR

概要	磁気センサ異常検出確認
説明	磁気センサ異常が発生したことを確認します。
引数	なし
リターン値	0：磁気センサ異常は発生していない 0以外：磁気センサ異常発生
備考	em_g_errorの値を確認

［マクロ名］EM_IS_CASE_OPEN_ERR

概要	ケース開放検出確認
説明	ケース開放が発生したことを確認します。
引数	なし
リターン値	0：ケース開放は発生していない 0以外：ケース開放発生
備考	em_g_errorの値を確認

[変数] unsigned char em_g_reset_state

概要	リセット要因
説明	RL78/I1B のリセット要因が設定されます。
数値	0x00 : パワーオンリセット 0x01 : ウォッチドッグ・タイマのタイムアウトによるリセット
関連マクロ	EM_IS_POWERON_RESET EM_IS_WOTCHDOG
備考	なし

[マクロ名] EM_IS_POWERON_RESET

概要	リセット要因確認(パワーオンリセット)
説明	リセット要因がパワーオンリセットかどうか確認します。
引数	なし
リターン値	0 : パワーオンリセットではない 0 以外 : パワーオンリセットがリセット要因
備考	em_g_reset_state の値を確認

[マクロ名] EM_IS_WOTCHDOG

概要	リセット要因確認(ウォッチドッグ・タイマのタイムアウト)
説明	リセット要因がウォッチドッグ・タイマのタイムアウトかどうか確認します。
引数	なし
リターン値	0 : ウォッチドッグ・タイマのタイムアウトではない 0 以外 : ウォッチドッグ・タイマのタイムアウトがリセット要因
備考	em_g_reset_state の値を確認

[変数] unsigned char em_g_sleep

概要	スリープモードフラグ
説明	AC 電源の供給が遮断された場合にスリープモードフラグが設定されます。このフラグが設定されていると、簡易スケジューラは RL78/I1B を STOP モードに設定し、本製品はスリープモードに移行します。スリープモード中は AC 電源の供給状態を監視し、AC 電源の供給を検出すると通常動作へ戻ります。
数値	0x00 : 通常動作中 0x01 : スリープモード
関連マクロ	EM_IS_NOT_SLEEP EM_IS_SLEEP
備考	なし

[マクロ名] EM_IS_NOT_SLEEP

概要	非スリープモード確認
説明	スリープモードではないことを確認します。
引数	なし
リターン値	0 : スリープモード 0 以外 : スリープモードではない
備考	em_g_sleep の値を確認

[マクロ名] EM_IS_SLEEP

概要	スリープモード確認
説明	スリープモードであることを確認します。
引数	なし
リターン値	0 : スリープモードではない 0以外 : スリープモード
備考	em_g_sleepの値を確認

[変数] unsigned char em_g_magnetic

概要	磁気センサフラグ
説明	磁気センサの現在の状態が設定されます。磁気センサ異常検出状態の場合、電力測定開始フラグを『開始』に設定しても電力測定処理は実行されません。
数値	0x00 : 異常なし 0x01 : 磁気センサ異常検出
関連マクロ	EM_IS_NOT_MAGNETIC EM_IS_MAGNETIC
備考	なし

[マクロ名] EM_IS_NOT_MAGNETIC

概要	磁気センサ状態確認(異常)
説明	磁気センサの状態が異常であるかを確認します。
引数	なし
リターン値	0 : 正常 0以外 : 異常
備考	em_g_magneticの値を確認

[マクロ名] EM_IS_MAGNETIC

概要	磁気センサ状態確認(正常)
説明	磁気センサの状態が正常であるかを確認します。
引数	なし
リターン値	0 : 異常 0以外 : 正常
備考	em_g_magneticの値を確認

[変数] unsigned char em_g_case_open

概要	ケース開放フラグ
説明	ケース開閉検出スイッチの現在の状態が設定されます。ケース開放状態の場合、電力測定開始フラグを『開始』に設定しても電力測定処理は実行されません。
数値	0x00 ： ケース閉鎖 0x01 ： ケース開放
関連マクロ	EM_IS_NOT_CASE_OPEN EM_IS_CASE_OPEN
備考	なし

[マクロ名] EM_IS_NOT_CASE_OPEN

概要	ケース開放状態確認(非開放)
説明	ケースが開放されていないことを確認します。
引数	なし
リターン値	0 : 開放されている 0 以外 : 開放されていない
備考	em_g_case_open の値を確認

[マクロ名] EM_IS_CASE_OPEN

概要	ケース開放状態確認(開放)
説明	ケースが開放された状態であるかを確認します。
引数	なし
リターン値	0 : 開放されていない 0 以外 : 開放されている
備考	em_g_case_open の値を確認

[変数] unsigned char em_g_ac_return[8]

概要	AC 電源復帰フラグ
説明	AC 電源が遮断した後、AC 電源が復帰した場合にフラグが設定されます。AC 電源復帰フラグを参照する場合は、必ず簡易スケジューラから与えられたアプリケーション番号を添字またはマクロの引数に使用して下さい。
数値	0x00 ： 異常なし 0x01 ： AC 電源復帰
関連マクロ	EM_IS_NOT_AC_RETURN(n) EM_IS_AC_RETURN(n) EM_CLEAR_AC_RETURN(n)
備考	なし

[マクロ名] EM_IS_NOT_AC_RETURN

概要	AC 電源復帰フラグ確認(復帰無し)
説明	AC 電源の復帰が発生していないことを確認します。
引数	n : アプリケーション番号 (簡易スケジューラからユーザプログラム 1～6 に渡される引数 gs を設定して使用して下さい)
リターン値	0 : AC 電源復帰発生 0 以外 : AC 電源復帰発生していない
備考	em_g_ac_return[n]の値を確認

［マクロ名］ EM_IS_AC_RETURN

概要	AC電源復帰フラグ確認(復帰有り)
説明	AC電源の復帰が発生していることを確認します。
引数	n：アプリケーション番号 (簡易スケジューラからユーザプログラム1～6に渡される引数gsを設定して使用して下さい)
リターン値	0：AC電源復帰発生していない 0以外：AC電源復帰発生
備考	em_g_ac_return[n]の値を確認

［マクロ名］ EM_CLEAR_AC_RETURN

概要	AC電源復帰フラグクリア
説明	AC電源復帰フラグをクリアします。
引数	n：アプリケーション番号 (簡易スケジューラからユーザプログラム1～6に渡される引数gsを設定して使用して下さい)
リターン値	なし
備考	em_g_ac_return[n]の値を変更

［変数］ unsigned char em_g_sw

概要	LCD切替スイッチ押下フラグ
説明	LCD切替スイッチの押下状態が設定されます。LCD切替スイッチは短押し(2秒以内)と長押し(2秒以上)に対応しています。また、電力測定処理の実行中はLCD切替スイッチの入力は受け付けていません。
数値	0x00 ： 押下無し 0x01 ： 短押し 0x02 ： 長押し
関連マクロ	EM_IS_SW_NOT_PUSH EM_IS_SW_SHORT_PUSH EM_IS_SW_LONG_PUSH EM_SW_CLEAR_PUSH
備考	なし

［マクロ名］ EM_IS_SW_NOT_PUSH

概要	LCD切替スイッチ押下フラグ確認(押し下げ無し)
説明	LCD切替スイッチの押下がないことを確認します。
引数	なし
リターン値	0：押下あり 0以外：押下なし
備考	em_g_swの値を確認

［マクロ名］ EM_IS_SW_SHORT_PUSH

概要	LCD 切替スイッチ押下フラグ確認(短押し)
説明	LCD 切替スイッチが短押しされたことを確認します。
引数	なし
リターン値	0：短押しされていない 0 以外：短押しされた
備考	em_g_sw の値を確認

［マクロ名］ EM_IS_SW_LONG_PUSH

概要	LCD 切替スイッチ押下フラグ確認(長押し)
説明	LCD 切替スイッチが長押しされたことを確認します。
引数	なし
リターン値	0：長押しされていない 0 以外：長押しされた
備考	em_g_sw の値を確認

［マクロ名］ EM_SW_CLEAR_PUSH

概要	LCD 切替スイッチ押下フラグクリア
説明	LCD 切替スイッチ押下フラグをクリア(押下なし状態に変更)します。
引数	なし
リターン値	なし
備考	em_g_sw の値を変更

4.5. ユーザ I/F API 使用サンプル

電力測定操作 API を使用したサンプルソースを以下に示します。

1） 電力測定開始処理

```
/* ユーザプログラム 1～3 に実装することを想定しています。 */
void  em_user_program1(unsigned char gs ) {
    /* スリープモードの場合は処理をしない */
    if (EM_IS_SLEEP) {
        return;
    }

    /* 電源電圧復帰判定 */
    if (EM_IS_AC_RETURN(gs)) {
        /* 電源電圧復帰に関する処理がある場合はここに記述 */
        EM_CLEAR_AC_RETURN(gs);
    }

    /* ⊿ΣADC のサンプリング周波数設定 */
    EM_SAMPLING_FREQ_4K;         /* 4kHz に設定 */

    /* ⊿ΣADC のセットアップ時間設定 */
    em_g_dsadc_init_wait = 15;   /* 150ms に設定 */

    /* 電力測定時間を設定 */
    em_g_measure_time = 10;      /* 1.0 秒に設定 */

    /* 電流測定デバイス選択 */
    EM_DEVICE_CT;                /* CT を選択 */

    /* 各入力ゲインを設定 */
    EM_VGAIN_AUTO;               /* 電圧入力ゲインを自動に設定 */
    EM_CGAIN_CT_AUTO;            /* 電流(CT)入力ゲインを自動に設定 */
    EM_CGAIN_S_AUTO;             /* 電流(シャント抵抗)入力ゲインを自動に設定 */

    /* ゼロクロス設定 */
    EM_ZEROCROSS_SOFT;             /* ゼロクロストリガをソフトウェアに設定 */
    em_g_zerocross_th = 11.3;    /* ゼロクロストリガ閾値を 11.3V に設定 */

    /* 過電圧・過電流設定 */
    em_g_over_volt_th = 382.0;   /* 過電圧閾値を 382.0V に設定 */
    em_g_over_current_th = 22.0; /* 過電圧閾値を 22.0A に設定 */

    /* 電力測定処理開始 */
    EM_MEASURE_START;            /* 電力測定開始 */
}
```

2） 電力測定結果処理

```
/* ユーザプログラム4〜6に実装することを想定しています。 */
void  em_user_program4(unsigned char gs ) {
    char str[64];

    /* スリープモードの場合は処理をしない */
    if (EM_IS_SLEEP) {
        return;
    }

    /* エラー判定 */
    if (EM_IS_SUCCESS){
        /* 正常時の処理 */
        sprintf( str, “%5f,%5f,%5f,%5f,%5f¥n”, em_g_active_power,   /* 有効電力 */
                                                 em_g_reactive_power, /* 無効電力 */
                                                 em_g_apparent_power, /* 皮相電力 */
                                                 em_g_volt,           /* 電圧 */
                                                 em_g_current );      /* 電流 */
    }
    else if (EM_IS_ZEROCROSS_ERR) {
        /* ゼロクロスエラー時の処理 */
    }
    else if (EM_IS_OVER_VOLT_ERR) {
        /* 過電圧エラー時の処理 */
    }
    else if (EM_IS_OVER_CURRENT_ERR) {
        /* 過電流エラー時の処理 */
    }
    else if (EM_IS_AC_OFF_ERR) {
        /*  AC電源断エラー時の処理 */
    }
    else if (EM_IS_DEVICE_ERR) {
        /* 電力測定エラー時の処理 */
    }
    else if (EM_IS_MAGNETIC_ERR) {
        /* 磁気センサエラー時の処理 */
    }
    else if (EM_IS_CASE_OPEN_ERR) {
        /* ケース開放時の処理 */
    }
}
```

3）　各種フラグ参照処理

```
/* ユーザプログラムで各種フラグを参照する場合のサンプルです。 */
void  em_user_program5(unsigned char gs ) {
    /* スリープモードの場合は処理をしない */
    if (EM_IS_SLEEP) {
        return;
    }

    /* リセット後の処理 */
    if (EM_IS_POWERON_RESET){
        /* パワーオンリセット時の処理 */
    }
    else if (EM_IS_WOTCHDOG) {
        /* ウォッチドッグ・タイマのタイムアウト時の処理 */
    }

    /* 磁気センサ異常判定 */
    if (EM_IS_MAGNETIC) {
        /* 磁気センサ異常時の処理 */
    }

    /* ケース開放判定
    if (EM_IS_CASE_OPEN) {
        /* ケース開放時の処理 */
    }

    /* スイッチ処理 */
    if (!EM_IS_SW_NOT_PUSH) {
        /*  押下有り */
        if (EM_IS_SW_SHORT_PUSH) {
            /* 短押し時の処理 */
        } else if (EM_IS_SW_LONG_PUSH) {
            /* 長押し時の処理 */
        }
        EM_SW_CLEAR_PUSH; /* スイッチ押下フラグクリア */
    }
}
```

5. 付録

5.1. ファイル一覧

表 5-1 に本ソフトウェアを構成するファイル一覧を示します。本ソフトウェアは CS+によるコード生成機能を利用して作成されています。備考欄に自動生成と記述されているファイルはコード生成機能で生成されたファイルです。

表 5-1 ファイル構成(1/2)

ディレクトリ	ファイル名	概要	備考
.¥	cstart.asm	スタートアップ処理	
	hdwinit.asm	ハードウェア初期化	
	iodefine.h	I/O 定義	
	stkinit.asm	スタック初期化	
.¥cg_src¥	r_cg_adc.c	A/D コンバータドライバ	自動生成
	r_cg_adc.h	A/D コンバータドライバヘッダ	自動生成
	r_cg_adc_user.c	A/D コンバータ割り込み	自動生成
	r_cg_bup.c	バッテリバックアップドライバ	自動生成
	r_cg_bup.h	バッテリバックアップドライバヘッダ	自動生成
	r_cg_bup_user.c	バッテリバックアップ割り込み	自動生成
	r_cg_cgc.c	クロック発生回路ドライバ	自動生成
	r_cg_cgc.h	クロック発生回路ドライバヘッダ	自動生成
	r_cg_cgc_user.c	クロック発生回路割り込み	自動生成
	r_cg_comp.c	コンパレータドライバ	自動生成
	r_cg_comp.h	コンパレータドライバヘッダ	自動生成
	r_cg_comp_user.c	コンパレータ割り込み	自動生成
	r_cg_dsadc.c	⊿ΣADC ドライバ	自動生成
	r_cg_dsadc.h	⊿ΣADC ドライバヘッダ	自動生成
	r_cg_dsadc_user.c	⊿ΣADC 割り込み	自動生成
	r_cg_hofc.c	周波数補正ドライバ	自動生成
	r_cg_hofc.h	周波数補正ドライバヘッダ	自動生成
	r_cg_hofc_user.c	周波数補正割り込み	自動生成
	r_cg_iica.c	IIC ドライバ	自動生成
	r_cg_iica.h	IIC ドライバヘッダ	自動生成
	r_cg_iica_user.c	IIC 割り込み	自動生成
	r_cg_intp.c	外部割込みドライバ	自動生成
	r_cg_intp.h	外部割込みドライバヘッダ	自動生成
	r_cg_intp_user.c	外部割込み	自動生成
	r_cg_lcd.c	LCD ドライバ	自動生成
	r_cg_lcd.h	LCD ドライバヘッダ	自動生成
	r_cg_lcd_user.c	LCD 割り込み	自動生成
	r_cg_main.c	メイン処理	自動生成
	r_cg_port.c	ポートドライバ	自動生成

表 5-2 ファイル構成(2/2)

ディレクトリ	ファイル名	概要	備考
.¥cg_src¥	r_cg_port.h	ポートドライバヘッダ	自動生成
	r_cg_port_user.c	ポート割り込み	自動生成
	r_cg_rtc.c	リアルタイムクロックドライバ	自動生成
	r_cg_rtc.h	リアルタイムクロックドライバヘッダ	自動生成
	r_cg_rtc_user.c	リアルタイムクロック割り込み	自動生成
	r_cg_sau.c	シリアルドライバ	自動生成
	r_cg_sau.h	シリアルドライバヘッダ	自動生成
	r_cg_sau_user.c	シリアル割り込み	自動生成
	r_cg_systeminit.c	システム初期化	自動生成
	r_cg_tau.c	タイマドライバ	自動生成
	r_cg_tau.h	タイマドライバヘッダ	自動生成
	r_cg_tau_user.c	タイマ割り込み	自動生成
	r_cg_wdt.c	ウォッチドッグ・タイマドライバ	自動生成
	r_cg_wdt.h	ウォッチドッグ・タイマドライバヘッダ	自動生成
	r_cg_wdt_user.c	ウォッチドッグ・タイマ割り込み	自動生成
.¥em¥	em_api.h	電力測定 API ヘッダ	
	em_api_clock.c	電力測定 API クロック補正	
	em_api_init.c	電力測定 API 初期化	
	em_api_lcd.c	電力測定 API LCD 制御	
	em_api_led.c	電力測定 API LED 制御	
	em_api_wdt.c	電力測定 API WDT 制御	
	em_cycle.c	サイクル検出	
	em_eeprom.c	EEPROM 操作	
	em_ep_measure.c	電力測定	
	em_ep_measure_if.c	電力測定 I/F	
	em_g_variable.c	電力測定 API グローバル変数	
	em_intr.c	電力測定 外部割込み制御	
	em_lcd_sw.c	スイッチ/LCD 制御 サンプル	
	em_simple_scheduler.c	簡易スケジューラ	
	em_sleep.c	スリープモード制御	
	em_sw.c	LCD 切替スイッチ制御	
	em_timer.c	電力測定 API タイマ制御	
.¥include¥	em_common.h	電力測定 API 共通ヘッダ	
	em_config.h	電力測定 API コンフィグレーションヘッダ	
.¥user_if¥	user_if.c	ユーザプログラム I/F	
	user_if.h	ユーザプログラム I/F ヘッダ	
.¥wrapper¥	em_wrapper.c	ラッパ関数	
	em_wrapper.h	ラッパ関数ヘッダ	

5.2. 関数一覧

表 5-3 にソフトウェアで使用する関数の一覧を示します。

表 5-3 関数一覧(1/5)

関数名	ファイル	概要
void R_ADC_Set_AC_Off_Observ(void)	r_cg_adc.c	AC 電源 OFF 監視処理
void R_ADC_Set_AC_On_Observ(void)	r_cg_adc.c	AC 電源 ON 監視処理
void R_ADC_Set_AC_Value_Get(void)	r_cg_adc.c	A/D 変換値取得
int R_BUP_Is_Vbat(void);	r_cg_bup.c	バッテリバックアップ状態判定
void R_BUP_Enable_Intr(void);	r_cg_bup.c	電源切替割り込み許可
void R_BUP_Disable_Intr(void);	r_cg_bup.c	電源切替割り込み禁止
void R_COMP1_Start_Ref1(void)	r_cg_comp.c	コンパレータ動作開始
void R_DSADC_Set_Sampling_Freq(uint8_t freq)	r_cg_dsadc.c	サンプリング周波数設定
void R_DSADC_Channel0_Set_PGAGain(uint8_t gain)	r_cg_dsadc.c	チャネル 0 の ゲイン設定
void R_DSADC_Channel1_Set_PGAGain(uint8_t gain)	r_cg_dsadc.c	チャネル 1 の ゲイン設定
void R_DSADC_Channel2_Set_PGAGain(uint8_t gain)	r_cg_dsadc.c	チャネル 2 の ゲイン設定
MD_STATUS R_IICA0_Master_Continue_Send(uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num, uint8_t wait)	r_cg_iica.c	IIC 送信処理
MD_STATUS R_IICA0_Master_Continue_Receive(uint8_t adr, uint8_t * const rx_buf, uint16_t rx_num, uint8_t wait)	r_cg_iica.c	IIC 受信処理
MD_STATUS R_IICA0_StopCondition_Check(uint8_t wait)	r_cg_iica.c	IIC ストップコンディションチェック
void R_IICA0_Adjust_Clock(void)	r_cg_iica.c	IIC クロック調整
void R_RTC_Start_Intr_Masked(void)	r_cg_rtc.c	RTC 動作開始
void R_UART2_Adjust_Clock(void)	r_cg_sau.c	UART2 クロック調整
void R_TAU0_Adjust_Clock(void)	r_cg_tau.c	TAU0 クロック調整
void R_TAU0_Channel1_Effective(void)	r_cg_tau.c	TAU0 チャネル 1 動作有効化
void R_TAU0_Channel1_Ineffective(void)	r_cg_tau.c	TAU0 チャネル 1 動作無効化
void R_TAU0_Channel2_Set_And_Start(uint16_t tdr)	r_cg_tau.c	TAU0 チャネル 2 動作開始
int em_select_clock_ext(void)	em_api_clock.c	外部クロック選択

表 5-4 関数名(2/5)

関数名	ファイル	概要
int em_select_clock_hoco(unsigned char div)	em_api_clock.c	高速オンチップ・オシレータ・クロック選択
void em_handler_hofc(void)	em_api_clock.c	周波数補正 割り込み処理
void em_clock_corection(void)	em_api_clock.c	周波数補正
void em_api_init(void)	em_api_init.c	電力測定 API 初期化
void em_adjust_clock(void)	em_api_init.c	タイマクロック 補正処理
void em_api_init_reset_state(uint8_t reset_flag, uint8_t porsr_flag)	em_api_init.c	リセット要因設定処理
void em_putc_lcd(int column, char c)	em_api_lcd.c	LCD 表示(1 桁)
void em_putd_lcd(int column, char on)	em_api_lcd.c	LCD ドット表示
void em_puts_lcd(char *s)	em_api_lcd.c	LCD 表示(全桁一括)
void em_handler_led_blink(void)	em_api_led.c	LED 点滅割り込み処理
void em_led_on(unsigned char n)	em_api_led.c	LED 点灯
void em_led_off(unsigned char n)	em_api_led.c	LED 消灯
void em_led_blink(unsigned char n)	em_api_led.c	LED 点滅
int em_is_led_blink(void)	em_api_led.c	LED 点滅判定
void em_led_enable(void)	em_api_led.c	LED 点灯/点滅有効化
void em_led_disable(void)	em_api_led.c	LED 点灯/点滅無効化
void em_clear_wdt(void)	em_api_wdt.c	WDT カウンタクリア
void em_wdt(void)	em_api_wdt.c	WDT タイムアウト処理
void em_handler_cycle(void)	em_cycle.c	サイクル検出 割り込み処理
void em_start_detect_cycle(void)	em_cycle.c	サイクル検出開始
void em_stop_detect_cycle(void)	em_cycle.c	サイクル検出停止
void em_get_cycle_state(void)	em_cycle.c	サイクル検出状態取得
int em_send_iic_eeprom(unsigned char iic_addr, unsigned char *data, unsigned short num)	em_eeprom.c	EEPROM への IIC 送信処理
int em_ack_polling_iic_eeprom(unsigned char iic_addr)	em_eeprom.c	IIC ACK ポーリング処理
int em_receive_iic_eeprom(unsigned char iic_addr, unsigned char *data, unsigned short num)	em_eeprom.c	EEPROM からの IIC 受信処理
void em_handler_iic_error(MD_STATUS flag)	em_eeprom.c	IIC 転送エラー処理
void em_handler_iic_receive(void)	em_eeprom.c	IIC 受信完了処理
void em_handler_iic_send(void)	em_eeprom.c	IIC 送信完了処理
int em_read_byte_eeprom(unsigned short addr, unsigned char *c)	em_eeprom.c	EEPROM 読み出し(1Byte)
int em_read_eeprom(unsigned short addr, unsigned char *s, unsigned int num)	em_eeprom.c	EEPROM 読み出し(連続)
int em_write_byte_eeprom(unsigned short addr, unsigned char c)	em_eeprom.c	EEPROM 書き込み(1Byte)

表 5-5 関数名(3/5)

関数名	ファイル	概要
int em_write_eeprom(unsigned short addr, unsigned char *s, unsigned int num)	em_eeprom.c	EEPROM 書き込み(連続)
void em_eeprom_adjust_clock(void)	em_eeprom.c	EEPROM クロック調整
int em_over_judge(int32_t measure_value, int32_t threshold)	em_ep_measure.c	過電圧/過電流判定
int32_t em_normalize_dsadc_result(uint32_t dsadc_result, int32_t offset)	em_ep_measure.c	⊿ΣADC 測定 データ校正
void em_normalize_gain(EM_PROFILE *p_prof, unsigned char auto_perm)	em_ep_measure.c	⊿ΣADC ゲイン校正
void em_set_measure_value(EM_PROFILE *p_prof, int32_t measure_volt, int32_t measure_current)	em_ep_measure.c	電圧・電流測定値積算
int em_zero_cross_no_th(EM_PROFILE *p_prof, int32_t measure_volt)	em_ep_measure.c	ゼロクロス処理 (閾値なし)
int em_zero_cross(EM_PROFILE *p_prof, int32_t measure_volt)	em_ep_measure.c	ゼロクロス処理
void em_handler_dsadc(void)	em_ep_measure.c	⊿ΣADC 変換完了処理
void em_active_intr(void)	em_ep_measure.c	電力測定割り込み 有効化
int em_ep_init(EM_PROFILE* p_prof)	em_ep_measure.c	電力測定処理初期化
int em_ep_pre_sampling(EM_PROFILE *p_prof)	em_ep_measure.c	電力測定 プリサンプリング処理
int em_ep_measure(EM_PROFILE *p_prof)	em_ep_measure.c	電力測定処理
int em_ep_calc(EM_PROFILE *p_prof)	em_ep_measure.c	電力計算処理
float em_get_active_power(EM_PROFILE* p_prof)	em_ep_measure.c	有効電力取得
float em_get_reactive_power(EM_PROFILE* p_prof)	em_ep_measure.c	無効電力取得
float em_get_apparent_power(EM_PROFILE* p_prof)	em_ep_measure.c	皮相電力取得
float em_get_volt(EM_PROFILE* p_prof)	em_ep_measure.c	電圧取得
float em_get_current(EM_PROFILE* p_prof)	em_ep_measure.c	電流取得
void em_ep_measure_if(unsigned char gs)	em_ep_measure_if.c	電力測定処理実行
void em_handler_magnetic(void)	em_intr.c	磁気センサ 割り込み処理
void em_handler_case(void)	em_intr.c	ケース開閉検出 スイッチ割り込み 処理
void em_mask_all_intr(unsigned char* intr_mask)	em_intr.c	全割り込みマスク
void em_restore_mask_intr(unsigned char* intr_mask)	em_intr.c	割り込みマスク復旧
short em_gain_to_multi(unsigned char gain)	em_lcd_sw.c	ゲイン値→2 の乗数変換
void em_edit_int_str(char*s, short num, unsigned int digit, unsigned int min_digit)	em_lcd_sw.c	整数→文字列変換

表 5-6 関数一覧(4/5)

関数名	ファイル	概要
void em_edit_num_str(char *s, float num, int *dot)	em_lcd_sw.c	浮動小数→文字列変換
void em_lcd_sw(unsigned char gs)	em_lcd_sw.c	スイッチ/LCD 操作
void em_simple_scheduler(void)	em_simple_scheduler.c	簡易スケジューラ
void em_handler_adc_off_observe(void)	em_sleep.c	AC 電源 Off 監視 割り込み処理
void em_handler_adc_on_observe(void)	em_sleep.c	AC 電源 On 監視 割り込み処理
void em_handler_vbat(void)	em_sleep.c	バッテリ・バックアップ・モード遷移 割り込み処理
void em_sleep(void)	em_sleep.c	スリープモード遷移
void em_handler_tau01(uint8_t sw_level)	em_sw.c	LCD 切替スイッチ押下割り込み処理
void em_handler_tau02_sw(void)	em_sw.c	LCD 切替スイッチ長押し判定 割り込み処理
void em_handler_tau02(void)	em_timer.c	タイマ・アレイ・ユニット チャネル 2 割り込み処理
void em_timer_start_1ms(void)	em_timer.c	1ms インターバル タイマ開始
void em_timer_start_10ms(void)	em_timer.c	10ms インターバル タイマ開始
void em_timer_start_100ms(void)	em_timer.c	100ms インターバル タイマ開始
void em_timer_stop(void)	em_timer.c	インターバル タイマ停止
int em_is_timeout(unsigned long time)	em_timer.c	インターバルタイマ タイムアウト判定
void em_user_init(void)	user_if.c	ユーザ初期化処理 I/F
void em_user_wdt(void)	user_if.c	WDT 処理 I/F
void em_user_start_measure(void)	user_if.c	電力測定開始時 コールバック I/F
void em_user_program1(unsigned char gs)	user_if.c	ユーザプログラム I/F1
void em_user_program2(unsigned char gs)	user_if.c	ユーザプログラム I/F2
void em_user_program3(unsigned char gs)	user_if.c	ユーザプログラム I/F3

表 5-7 関数一覧(5/5)

関数名	ファイル	概要
void em_user_program4(unsigned char gs)	user_if.c	ユーザプログラムI/F4
void em_user_program5(unsigned char gs)	user_if.c	ユーザプログラムI/F5
void em_user_program6(unsigned char gs)	user_if.c	ユーザプログラムI/F6
void em_wrapper_intp6_handler(void)	em_wrapper.c	INTP6 割り込みハンドララッパ
void em_wrapper_intp7_handler(void)	em_wrapper.c	INTP7 割り込みハンドララッパ
void em_wrapper_sw_handler(void)	em_wrapper.c	LCD 切替スイッチ割り込みハンドララッパ
void em_wrapper_eeprom_wp(uint8_t wp)	em_wrapper.c	EEPROM ライトプロテクト制御ラッパ
void em_wrapper_set_dsadc_gain_volt(uint8_t gain)	em_wrapper.c	ΣADC 電圧ゲイン設定ラッパ
void em_wrapper_set_dsadc_gain_ct(uint8_t gain)	em_wrapper.c	ΣADC 電流(CT)ゲイン設定ラッパ
void em_wrapper_set_dsadc_gain_shunt(uint8_t gain)	em_wrapper.c	ΣADC 電流(シャント)ゲイン設定ラッパ
void em_wrapper_get_dsadc_volt(uint32_t * const buffer)	em_wrapper.c	⊿ΣADC 電圧変換値取得ラッパ
void em_wrapper_get_dsadc_ct(uint32_t * const buffer)	em_wrapper.c	⊿ΣADC 電流(CT)変換値取得ラッパ
void em_wrapper_get_dsadc_shunt(uint32_t * const buffer)	em_wrapper.c	⊿ΣADC 電流(シャント)変換値取得ラッパ
void em_wrapper_adc_ac_monitor_ch(void)	em_wrapper.c	AC 電源監視用 ADC チャネル設定ラッパ
void em_wrapper_led_on(unsigned char n)	em_wrapper.c	LED 点灯ラッパ
void em_wrapper_led_off(unsigned char n)	em_wrapper.c	LED 消灯ラッパ

5.3. 利用可能なマイコン資源

1）　内部/外部リソース

表 5-8 に RL78/I1B の内部リソース（周辺機能）一覧を示します。ユーザの列で○になっている機能はユーザプログラムで使用が可能です。

表 5-8 内部リソース（周辺機能）

機能		システム			システム（異常系）			電力測定			ﾘﾌｧﾚﾝｽ			ユーザ			備考
		初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	
ポート	P00, P01	○	×	×	×	×	×	×	×	×	×	○	○	×	○	○	拡張シリアル I/F
	P02－P07	○	×	×	×	×	×	×	×	×	×	×	×	×	○	○	PMOD I/F
	P1	○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	LCD
	P2	○	○	×	×	×	×	×	×	×	×	×	×	×	×	×	ANI, IVCMP
	P30, P31	○	×	○	×	×	×	×	×	○	×	×	×	×	×	○	LED
	P32, P33	○	×	×	×	×	×	×	×	×	×	×	×	×	○	○	拡張シリアル I/F
	P40	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	E1 I/F
	P41, P42, P44	○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	LCD切り替えスイッチ、ケース開閉スイッチ、磁気センサ
	P43	○	×	×	×	×	×	×	×	×	×	×	×	×	○	○	拡張シリアル I/F
	P6	○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	EEPROM
	P7	○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	LCD
	P8	○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	LCD
	P121, P122	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	高速オンチップ・オシレータ
	P123, P124	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	サブシステム・クロック
	P125	○	×	×	×	×	×	×	×	×	×	×	×	×	○	○	PMOD I/F
	P126, P127	○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	LCD
クロック発生回路		○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	
高速オンチップオシレータクロック周波数補正機能		×	×	×	×	×	×	×	○	○	×	×	×	×	×	×	
タイマ・アレイ・ユニット	チャネル0	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	拡張シリアル I/F
	チャネル1	○	×	×	×	×	×	×	×	×	×	○	×	×	○	×	LCD切り替えスイッチ
	チャネル2	×	×	×	×	×	×	○	○	○	×	×	×	×	×	×	電力測定用の時間管理 100msインターバルタイマ
	チャネル3	×	×	×	×	×	×	×	×	×	○	○	○	○	○	○	インターバルタイマとしてのみ使用可能
	チャネル4	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
	チャネル5	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
	チャネル6	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
	チャネル7	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
リアルタイムクロック		○	○	○	×	×	×	×	×	×	×	×	×	×	○	○	
サブシステム・クロック周波数測定回路		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
12ビットインターバルタイマ		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
8ビットインターバルタイマ		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
クロック出力/ブザー出力制御回路		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	兼用端子の関係で使用不可
ウォッチドッグタイマ		○	×	○	○	×	○	×	×	○	×	×	○	×	×	○	
A/Dコンバータ		○	○	○	○	○	○	○	○	○	×	×	×	×	×	×	AC断、温度センサ
温度センサ		×	×	×	×	×	×	○	○	○	×	×	×	×	×	×	
24ビットΔΣA/Dコンバータ		×	×	×	×	×	×	○	○	○	×	×	×	×	×	×	
コンパレータ		×	×	×	×	×	×	○	○	○	×	×	×	×	×	×	
シリアル・アレイ・ユニット		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	PMOD I/F, 拡張シリアル I/F
シリアル・インタフェースIICA		○	×	×	×	×	×	×	×	×	×	○	○	×	○	○	EEPROM
IrDA		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	兼用端子の関係で使用不可
LCDコントローラ/ドライバ		○	×	×	×	×	×	×	×	×	×	×	○	×	×	○	
データ・トランスファ・コントローラ		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	
割り込み機能		○	○	○	×	×	×	○	○	○	×	○	○	×	○	○	ユーザ側で使用する以外の割り込み設定の変更は不可
スタンバイ機能		○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	
リセット機能		○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	
パワーオンリセット回路		○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	
電圧検出回路		○	×	×	×	×	×	×	×	×	×	×	×	×	×	×	
バッテリバックアップ機能		○	○	○	×	×	×	×	×	×	×	×	×	×	×	×	
発振停止検出回路		○	×	×	○	×	×	×	×	×	×	×	×	×	×	×	
安全機能		×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	

表 5-9 に本製品の外部リソース(ハードウェア機能)一覧を示します。ユーザの列で○になっている機能はユーザプログラムで使用が可能です。

表 5-9 外部リソース(ハードウェア機能)

機能	システム			システム(異常系)			電力測定			ﾘﾌｧﾚﾝｽ			ユーザ			備考
	初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	初期化	読み込み	書き込み	
LCD	○	×	×	×	×	×	×	×	×	×	×	○	×	×	○	
LCD切り替えスイッチ	○	×	×	×	×	×	×	×	×	×	○	×	×	○	×	
ケース開閉スイッチ	○	○	×	○	○	×	×	×	×	×	×	×	×	×	×	
磁気センサ	○	○	×	○	○	×	×	×	×	×	×	×	×	×	×	
EEPROM	○	×	×	×	×	×	×	×	×	×	○	○	×	○	○	
LED	○	×	×	×	×	×	×	×	×	×	×	○	×	×	○	
PMOD I/F	×	×	×	×	×	×	×	×	×	×	×	×	○	○	○	ポートの初期化はシステム側で実施
拡張シリアルI/F	×	×	×	×	×	×	×	×	×	○	○	○	○	○	○	

2） 割り込みリソース

表 5-10 に RL78/I1B の割り込みリソースの一覧を示します。ユーザの列で〇になっている割り込みはユーザプログラムで使用が可能です。

表 5-10 割り込みリソース

デフォルト・プライオリティ	優先順位レベル	名称	トリガ	システム	システム（異常系）	電力測定	リファレンス	ユーザ	備考
0	0	INTWDTI	WDTインターバル	○	○	×	×	×	
1	0	INTLVI	電圧検出	○	×	×	×	×	(予約)電池残量確認
19	0	INTRTIT	RTC補正タイミング	○	×	×	×	×	(予約)
20	0	INTFM	周波数測定完了	○	×	×	×	×	(予約)
24	0	INTAD	AD変換終了	○	○	×	×	×	AD断割り込み
25	0	INTRTC	RTC定周期/アラーム	○	×	×	×	×	SLEEP時のSnooze制御
30	0	INTP6	端子入力エッジ検出	○	○	×	×	×	異常割り込み(磁気検出)
31	0	INTP7	端子入力エッジ検出	○	○	×	×	×	異常割り込み(ケース)
39	0	INTOSDC	発振停止検出	○	○	×	×	×	(予約)異常検出
42	0	INTVBAT	電源切り替え検出	○	×	×	×	×	
22	1	INTTM02	タイマ・チャネル02	×	×	○	×	×	電力測定用の時間管理 100msインターバルタイマ
27	1	INTDSAD	ΔΣADC変換完了	×	×	○	×	×	
32	1	INTCMP0	コンパレータ検出0	×	×	○	×	×	(予約)
33	1	INTCMP1	コンパレータ検出1	×	×	○	×	×	ゼロクロス検出
38	1	INTCR	高速オンチップ・オシレータ・クロック周波数補正完了	×	×	○	×	×	
2	2～	INTP0	端子入力エッジ検出	×	×	×	×	○	拡張シリアル
3	2～	INTP1	端子入力エッジ検出	×	×	×	×	○	PMOD
4	-	INTP2	端子入力エッジ検出	×	×	×	×	×	
5	-	INTP3	端子入力エッジ検出	×	×	×	×	×	
6	-	INTP4	端子入力エッジ検出	×	×	×	×	×	
7	2～	INTP5	端子入力エッジ検出	×	×	×	×	○	PMOD
8	2～	INTST2	UART2送信完了、バッファ空	×	×	×	○	○	拡張シリアル
9	2～	INTSR2	UART2受信完了	×	×	×	○	○	拡張シリアル
10	2～	INTSRE2	UART2受信エラー	×	×	×	○	○	拡張シリアル
11	2～	INTST0	UART0送信完了、バッファ空	×	×	×	×	○	PMOD
		INTSCI00	CSI00転送完了、バッファ空	×	×	×	×	○	PMOD
	-	INTIIC00	IIC00転送完了	×	×	×	×	×	
12	2～	INTTM00	タイマ・チャネル00	×	×	×	○	○	拡張シリアル
13	2～	INTSR0	UART0受信完了	×	×	×	×	○	PMOD
14	2～	INTSRE0	UART0受信エラー	×	×	×	×	○	PMOD
	-	INTTM01H	タイマ・チャネル01(上位8ビット)	×	×	×	×	×	
15	2～	INTST1	UART1送信完了、バッファ空	×	×	×	×	○	PMOD
	-	INTIIC10	IIC10転送完了	×	×	×	×	×	
16	2～	INTSR1	UART1受信完了	×	×	×	×	○	PMOD
17	2～	INTSRE1	UART1受信エラー	×	×	×	×	○	PMOD
	-	INTTM03H	タイマチャネル03(上位8ビット)	×	×	×	×	×	
18	2～	INTIICA0	IICA0通信完了	×	×	×	○	○	EEPROM
21	2～	INTTM01	タイマ・チャネル01(16ビット/下位8ビット)	×	×	×	○	○	LCD切替スイッチ
23	2～	INTTM03	タイマ・チャネル03(16ビット/下位8ビット)	×	×	×	○	○	
26	2～	INTIT	12ビットインターバルタイマ	×	×	×	×	○	
28	2～	INTTM04	タイマ・チャネル04	×	×	×	×	○	
29	2～	INTTM05	タイマ・チャネル05	×	×	×	×	○	
34	2～	INTTM06	タイマ・チャネル06	×	×	×	×	○	
35	2～	INTTM07	タイマ・チャネル07	×	×	×	×	○	
36	2～	INTIT00	8ビットインターバルタイマ・チャネル00	×	×	×	×	○	
37	2～	INTIT01	8ビットインターバルタイマ・チャネル01	×	×	×	×	○	
40	2～	INTIT10	8ビットインターバルタイマ・チャネル10	×	×	×	×	○	
41	2～	INTIT11	8ビットインターバルタイマ・チャネル11	×	×	×	×	○	

6.　改版履歴

版数	日付	内容
1 版	2016/08/01	新規作成