RL78 マイコン学習セット マニュアル 入門編

第4版 2021.10.11 詳細下記 第1版 2015.4.16

【 製品概要 】

本マニュアルはRL78/I1A R5F107DE(38ピン)マイコンを使ったマイコン学習セットの開発環境構築、ソフトウエアインストゥール手順、添付CDのサンプルプログラムの動作について解説されています。

入門編ではマイコンの基本的なハードウエアのアクセス方法、プログラムの書き方をサンプルプログラ ムを参考に学び、習熟度をチェックするために、演習プログラムの課題を自分で考えます。

ルネサスエレクトロニクス社の統合開発環境CS+ for CC における開発方法について多く記述 してあります。

※本学習セット開発にはルネサスエレクトロニクス社製E2 Liteが必要です。

【 第4版での変更点 】

1. デバックツールE1の製造中止に伴いましてE2 Lite(以降 E2L)でのデバックに変更しました。



1. 学習環境、事前準備

- 1-1. 学習環境
 - a:学習セット 同梱物
 - b: BCRL78107 CPU部の特徴
 - c: E2 Liteエミュレータ (デバッカ)
 - d:無償のCS+、RL78用Cコンパイラのダウンロード
 - e: CDコピー、デバイスドライバD2XXのインストゥール
 - f:RL78とH8/300H、R8Cの速度比較
 - f-1:ポートアクセス速度の比較
 - f-2:乗除演算速度の比較
- 1-2 動作、デバック
 - a:CS+起動、コンパイル、書き込み、動作

b:新しいプログラムを作る CS+ 操作

b-1:A/D、リセット、ウオッチドッグ設計上の注意点

- b 2 :自動生成されたプログラム
- b-3:E2Lから電源供給
- b-4:コード生成後の初期値の変更
- b-5:変数を見る
- b-6:変数変化を実行中に確認する

2. サンプルプログラム

2 — 1.	キー入力 samp	le1
	プログラム	:点滅する LED をキー入力で消灯
	演習プログラム	: 押されたキーの LED を点滅
2 – 2.	USB通信 samp	l e 2
	プログラム	:ABCDをパソコン側に送信
	演習プログラム	:キー入力でABCDと送信
2 — 3 .	A/D変換 samp	l e 3
	プログラム	:A/D変換データをパソコン側に送信
	演習プログラム	:A/D値を0-5Vに換算しパソコン側に送信。
2 — 4.	PWM samp	l e 4
	プログラム	:LED輝度連続可変
	演習プログラム	:LED輝度階段状可変
2 — 5.	割り込み samp	l e 5
	プログラム	: 割り込みでLD1点灯
	演習プログラム	:メインでLD1点滅、割り込みでLD2点灯、消灯

1-1. 学習環境

a:学習セット同梱物	
RL78学習ボード	1
CD(サンプルプログラム、デバイスドライバ、ドキュメント)	1
マニュアル(本誌) 入門、実用	各
電源ケーブル、USBケーブル	各
モーター	1
サーミスタ	1



※開発に必要なルネサスエレクトロニクス社製デバッカE2Lは同封されておりません。別途必要です。 但し、プログラムの検討、コンパイルは、無料のCS+(後述)で行うことが出来ます。

1 1

複数の人間の学習において

A. E2L+本ボード+CS+インストゥール済みパソコンを用意

B. プログラムの検討、コンパイルは他のパソコンで行い、実行だけAのパソコンに席を移る といった使い方で、人数分用意しなくても効率よく学習することは可能だと思います。もちろん、各人に 各台数あるのが、時間的な効率は一番良いです。

b: BCRL78107 CPUボード部の特徴

学習ボードのマイコン部分は弊社BCRL78107CPUボードと同じです。

●高性能、低消費電力、低コストな新設計RL78コアを使用。1.39DMIPS/MHz、46µA/
 MHz。32MHz±1%の高精度内蔵オシレータ ※1

●RL78/I1A(R5F107DE)は産業、インフラ、情報アプリケーションに特化した強力な周辺 機能(高性能PWMタイマ、LIN-bus、DALI通信機能)を搭載。38ピン。

●内蔵高速オシレータ 32MHz(2.7~5.5V)。最小命令実行時間31.25nsec。

●内蔵低速オシレーター 15KHz(TYP) CPUクロックとしては使用不可。

●メモリ容量 フラッシュROM64Kバイト、RAM4Kバイト、データフラッシュ4Kバイト。 電源を切ってもデータが保持されるEEPROM 25LC256(容量32、768BYTE)搭載 ラ イブラリ添付※2

●動作電圧電流 3.3V~5.5V、16mA TYPE(5V、USB使用、32MHz動作時)
 最低2.7Vから動作可能(BCRL78107Sタイプ ※2)

●豊富な周辺機能

I/Oポート 合計34、A/D変換器:10ビット分解能 11ch、プログラマブルゲインアンプ 6

ch、UART 3ch(1chはLIN-bus、DMX512、DALI通信対応)

タイマ8ch(PWM出力3ch、1nsec分解能可能、64MHzPLL+ディザリング)、乗除算・ 積和演算器内蔵、オンチップデバック機能内蔵

●USB搭載 ドライバIC FTDI社 FT232RL搭載。※2

●デバッカE2Lによるデバック用コネクタ搭載。C言語による1行実行、ブレークポイント、変数参照 等可能です。

※1 速度比較は本マニュアル 1-1 f:RL78とH8/300H、R8Cの速度比較をご参照下 さい。

※2 学習ボードはCPU+デバック用コネクタ、USBインターフェイス+EEPROM搭載のBCR
 L78107Mが使用されています。

CPU部大きさ(部品面)



USBミニBコネクタ、FT232RL、25LC256は裏面搭載。



概要

E2Lエミュレータは、ルネサス主要マイコンに対応したオンチップデバッギングエミュレータです。基 本的なデバッグ機能を有した低価格の購入しやすい開発ツールで、フラッシュプログラマとしても使用可 能です。

C言語ソースデバックが可能で、1 行実行、ブレークポイント設定、変数、レジスタ、メモリ参照等々、

従来であれば高価なICEしか出来なかった機能が、安価に実現されています。変数をウオッチ窓に登録 し、実行中を含めて数値を見ながらデバック出来ます。

また、使い方もHEW(統合開発環境)のE8aと同じで、経験があれば半日で、無くても1日で必要な 操作を会得することが出来ると思います。

マイコンとの通信として、シリアル接続方式とJTAG接続方式の2種類に対応しています。使用可能な デバッグインタフェースは、ご使用になるマイコンにより異なります。

対応MPU RL78,RX,RA,RE



E2LをPCのUSBに接続するとwindowsが自動的にデバイスドライバをインスツールします。 続いて、ネットから開発環境CubeSuite+とCコンパイラの最新版ダウンロードを行います。

d:無償のCS+、RL78用Cコンパイラのダウンロード

プログラムの開発はルネサスエレクトロニクス社の統合開発環境CS+ for CC でC言語を用い動作させることができます。CD添付のサンプルプログラムはこの環境下で作成されています。無償版 をダウンロードして使用します。

ネット検索で→「CS+ 無償ダウンロード」の検索で表示されます。

統合開発環境 CS+ (旧CubeSuite+)

製品名	仕様・性能	試用期限
統合開発環境 CS+ for CC (RL78, RX, RH850用) 製品ページ 評価版ダウンロード	 ・試用期限内は製品板(professional版)と同じ。試用期限を過ぎる と各MCUにより以下の制限があります。 (RH850ファミリ) リンクサイズを256Kバイト以内に制限しています。professional 版の機能は使用できません。 (RXファミリ) リンクサイズを128Kバイト以内に制限しています。 professional版の機能は使用できません。 (RL78ファミリ) リンクサイズを64Kバイト以内に制限しています。professional 版の機能は使用できません。 ・コンパイラ※、デバッカを同梱。 ・※ CC-RL CC-RK CC-RH 	60日 初めて評価板ソフトウェアツールをインストールした後、最初にビルドを行った日から60日間の営用期間があります。 試用期間内は、機能に制限はありません。 61日目以降は、リンクサイズ、professional版の機能が制限されます。

Cコンパイラ等も同梱されていています。ルネサスエレクトロニクス株式会社に登録が必要ですが、質問のときにも必要なのでしておいて、損はないと思います。

ダウンロード出来ましたら、指示に従い展開して下さい。

e:開発セット添付CDコピー、デバイスドライバD2XXのインストゥール

事前にCDの中のホルダを例えばC:¥WrokSpace¥にコピーしてください。WorkSpaceはCS+をインストゥールすると自動形成されます。

共	有▼ 書き込む	新しいフォルダー			
*	名前	^	更新日時	種類	サイズ
No.	📙 RL78STUDY		2015/04/30 13:34	ファイル フォル	
	📙 USBDRV		2015/04/30 13:34	ファイル フォル	
III	👢 ドキュメント		2015/04/30 13:34	ファイル フォル	

初めて、RL78学習ボードをパソコンにUSBケーブルで接続するとOSがFT232RLのデバイ スドライバを要求してきますが、Windows Updateに登録されているため、ユーザーは何も しなくても最新のデバイスドライバが自動的にインスツールされます。

正常にインストゥールされると、以下のように2つのデバイスが確認出来ます。



f:RL78とH8/300H、R8Cの速度比較

RL78は、製造中止がアナウンスされているH8/3048の替わりに検討される方も多いと思われま すが、実行速度はどうなのでしょうか? 開発環境を含めて以前より進化していなければ使う意味がない とお考えの方も多いかと思われます。

f-1 ポートアクセス速度比較

単純なポートアクセスプログラムで比較してみます。 RL78のポートを1,0繰り返すプログラムです。



オシロスコープでP20、P21波形を観測すると6.38732MHzという周波数でポートの1,0 を繰り返すことが分かります。(クロック32MHz)



TDS 2012 - 15:00:07 2013/08/13

この命令の詳細は

w h	ile	ə (1U)	
{			
P 2	=	0 x 0 0 ;	//ポートを0にする
P 2	=	Oxff;	//ポートを1にする
}			//上行にジャンプする

という3つの動作を行っています。波形が1から0に落ちて、上がる手前の時間が1命令の実行時間です。 波形上約30nsec程度なので、カタログ値 31.25nsecと大きく相違は無いように思います。 1クロックで1命令実行はRISC並みですね。1の時間が0に比べて長いのはポートを1にする、上行 にジャンプするの2命令実行しているからです。

H8/300Hコアを代表してH8/36109を使用しました。基板名BCH8361409。HEW で同じ意味のコードを書き込みテストします。H8/300日コアはH8/3048やH8/3052と 同じです。



ポートEを繰り返し、0、1しています。波形を観測すると828.067КHzとなりました。

6.38732MHz÷828.067KHz≒7.7倍高速という驚きの結果になりました。(クロッ ク20MHz)クロックを同じにしても、4.8倍違います。



663.601KHzとなりました。

f-2 乗除演算速度の比較

演算速度はどの程度違うでしょうか? 32bitの乗算、除算を行ってみました。 演算前にポートを立てて、演算後にポートを下ろすことにより、演算実行時間をオシロで観測しています。

H8-36109 約30µsecでした。



R8C/M12Aの場合 約15.5µsecでした。



RL78の場合 約3.8µsecでした。

ソースファイル



ソース+逆アセンブラ





以上の結果をまとめると

CPU⊐ア	クロック	ポートアクセス	乗除演算
R L 7 8	3 2 M H z	6. 38MHz	3. 8µsec
H 8 – 3 0 0 H	2 0 M H z	0.82MHz	30µѕес
R 8 C	2 0 M H z	0.66MHz	15.5µsec
結論		R L 7 8がH 8 - 3 0	R L 7 8 が H 8 - 3 0
		0Hの7.7倍、R8C	0Hの7.8倍、R8C
		の9.6倍高速。	の4倍高速。

※測定結果はいずれも弊社製品比較です。

一般に設計が新しいCPUの方が、製造プロセスが微細化されている分、同じ機能であれば安価に製造できます。 RL78は従来より優れたアーキテクチャのコアに、乗除・積和演算器、10進補正回路等、高度な機能も内蔵し、 かつ、今までより低消費電力、安価を目指して開発されたようです。

結論として、従来、H8/3048等をご使用の方々にも安心して使っていただける性能をもったCPU だと思います。

1-2 動作、デバック

a:CS+起動、コンパイル、書き込み、動作



CDに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

CS+を起動します。ここでは例としてRL78STUDY¥sample1を動作させます。キーを押 すと上のLEDが点滅するプログラムです。

初めてのときは ファイル → ファイルを開く → sample1.mtpjをダブルクリックしま す。

🚳 sample1.mtpj	2015/04/17 14:59	MTPJ ファイル	284 KB
sample1.ビーリバーエレクトロニクス	2015/04/30 13:09	MTUD ファイル	233 KB

🔕 sa	mple1 - CS+ for CC - [プロパティ]
7711	ル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B)
R	79-H(S) 🔳 🖪 🎒 i 🐰 🛅 🖄 🤊
9	
ار 🔛	םטֿיבלאישטר 🕂 🕂
V E	0 2 3
שי ק-איקבבקא-ק	a samplet (プロジェクト) ■ RSF107DE (マイクロコントローラ) ■ 第子配置 (詩計ツール) ■ コード生成 (詩計ツール) ■ CC-RL (ビルド・ツール) ■ プログラム解析 (解析ツール) ■ プログラム解析 (アーマール) ■ フード生成 ■ 「r_cg_cgc.c ■ r_cg_port.c ■ r_cg_ort.c ■ r_cg_userdefine.h ■ r_cg_cgc.h ■ r_cg_cgc.h
	and a start and a start start and

r_main. cが中央に表示されます。とりあえず、実行してみます。E2Lのケーブルを基板のCN

1に挿入します。電源はE2Lから供給しますので、USB接続は不要です。(写真ご参考)



「デバック・ツールヘプログラムを転送」をクリック。



正常に転送できると、今まで表示されていなかったプログラムの絶対アドレスが表示されます。E2L から電源3.3VがCPU基板に供給されます。



正常にダウンロード出来ると命令のアドレス等が表示されます。ここまでいかなかった場合、E2Lのインストゥールをご検証願います。

次に、プログラムを動作させます。「CPUリセット後、プログラムを実行」をクリック。

·ドウ(W) ヘルプ(H)			
ि ि ि × ि ि × ि • • • • • • • • • • • •	ित्र हेत्र दूत्र 👯		
	CPUリセット後、	プログラムを実行します。	L

4つのLED D1, 2, 3, 4が点滅したら正常に動作しています。CS+の右下部「RUN 実行中」 が表示されます。

)RUN 읒 実行中 🔍 RL78 E2 Lite

ここで キー SW1が押されている間、PORT2には0が出力され、LEDは消灯するはずです。そうなりますでしょうか?

ここまで確認できましたら、一度止めます。	
へルプ(H)	
1 = = = = (* ()	
実行中のプログラムを停止します。	(Shift+F5)

main 関数のwaitの数値に2か所、0を減らして



E)	表示	(V)	7	^r D:	ジェ	クト	-(P)	ピ	ルト	∹(B)	デバ	ッグ(D)	ツ-
J		0	*	Þ	ß	5	0	8	4	A			•	10
	2	יע	1/l	/C :	指定		1た]	頁目	を仍 刀匠	存し	します。 r main	(Ci	trl+	s)

プログラムを変更しましたから「ビルド後、デバック・ツールヘプログラムを転送」をクリック。

≪ 🔯 🗘 ⊨ 🖲 💿 🔊 🖓 🛞 🗟 ٿ 🕌
ビルド後デバッグ・ツールヘプログラムをダウンロードします。(F6)

「CPUリセット後、プログラムを実行」をクリック。 LEDの点灯が先ほどより、早く点滅するのが目視できましたでしょうか?

P12のデータを読み込んだり、その値を3ビット左にシフトしたり、反転させたりしているので、値が 変動します。

次に、ブレークポイントの設定を行ってみます。一度、	プログラムを停止させます。
へルプ(H)	
 Image: State of the state of th	
実行中のプログラムを停止します。	(Shift+F5)

マウスを 00195 番地のここにもって行き、左クリックで設定です。手の形が出ます。解除は設定後、再ク リック。



「CPUリセット後、プログラムを実行」をクリック。先ほど設定した行でプログラムカウンタが停止し ます。黄色は現在のプログラムカウンタの位置です。まだコードは実行されていません。



ステップオーバーで1行実行。

🖉 🕫 🚽 🕘 🕑 🗭 🖒 ĉE 🚰 🖓	
ステップ・オーバー実行を行います。	(F10)



プログラムカウンターは 00198 で、1 行前の P 1 2 = 0 x f 0 が実行されました。L D 1, 2, 3, 4 の点灯が確認できると思います。



P 1 2 = 0 x f 0;

のP12の部分をマウスの左ボタンを押したままでドラッグ、右クリックで「ウオッチ1に登録」で現在 のP12の内容が表示されます。

ウォッチ1		ą 🗙
2 🛞 🧏 🖏	★ 表記(N) - 100	
ウォッチ式 育1 P2	値 型情報(バイト数) 0×f0 SFR[R/W 1.8](1)	
ブレークポイントを創	ຊ除し「プログラを現在の位置から実行」すると │	:
	ļ	
🔨 🗟 🗗 🐂	🖲 💽 🗠 🛞 🖓 🕢	
	プログラムを現在の位置から実行します。(F	-5)

ウオッチ1窓にあるP2の値がプログラムの進行に伴い、OxOO、OxfOと繰り返すのが確認できると思います。

以上、大急ぎでしたが、プログラムの書き換え、コンパイル、E2Lへのダウンロード、実行、ブレーク ポイント設定、ウオッチ窓設定、動作の概要です。

b:新しいプログラムを作る

CS+でのプログラム開発は、例えばHEWと比べると大きく異なる部分があります。その一つはプロ グラムを書く前に、端子機能を入力すること(端子機能を入力しないと、プログラムが書けません)です。 これはハードウエア的に端子の割り振りが終了していないといけないことになります。

二つ目は 割り振りにより決まる、端子を使用するための関数がコード生成機能で自動的に生成される ことです。例えばSIOを使用するように端子を割振り、コード生成でSIOを使用すると設定し、「コ ード生成」ボタンをクリックすることにより、SIOを使用するためのイニシャル、送信、受信関数が作 成されます。これによりプログラマはそれらを書く必要がありません。アプリケーションのみに集中でき るように考えられています。

経験のある方ほど他と大きく異なる開発方法に戸惑いがあるかもしれません。すこし操作してみれば、C S+の機能が、より簡単に、より短時間に、より正確に開発が行えるよう考慮されているのが理解できる と思います。例えばハード的に入力しか使えない端子を出力で使おうとしても設定出来ませんので、ミス が発生しません。ソフトウエアの部分でも、提供される関数を使用することにより、バグの発生を抑え、 品質が底上げされます。

開発の詳細は順を追って説明します。

CS+を開くと、以下のような画面が表示されます。新しいプロジェクト作成はファイル→新規作成→新しいプロジェクトを作成 を実行します。

新しいプロジェク	ハを作成する 新たにプロジェクトを作成します。 既存のプロジェクトに登録されているファイル構成を流用して、作成することも可能です。
GU 新しい	クロジェクトを作成するための、 プロジェクト 作成 ダイアログを開きます。

新しいプロジェクトを作成します。上記または下記の方法



使用するマイクロコントローラはR5F107DE(38pin)を設定します。プロジェクト名を設定 して、「作成」をクリックします。

プロジェクト作成	×
マイクロコントローラ(<u>I</u>):	RL78
使用するマイクロコントローラ(<u>M</u>):	
▲(マイクロコントローラを検索でき	きます) アップデート(<u>U</u>)
 RL78/G14 (ROM192KB RL78/G14 (ROM256KB RL78/11 A (ROM256KB RL78/11 A (ROM34KB) R5F1 07AE(30pin) RL78/F1 2 (ROM54KB) RL78/F1 2 (ROM16KB) RL78/F1 2 (ROM16KB) RL78/F1 2 (ROM16KB) 	日種名:R5F107DE 内部ROMサイズ[V)パイト]64 内部RAMサイズ[V]パト]4096
プロジェクトの 種類(<u>K</u>):	アプリケーション(CA78KOR)
プロジェクト 名(<u>N</u>):	test_RL78107
作成場所(<u>L</u>):	C:¥WorkSpace
	☑ プロジェクト 名のフォルダを作成する(▲)
C:¥WorkSpace¥test_RL78107¥	test_RL78107.mtpj
📃 既存のプロジェクト のファイル構	成を流用する(<u>S</u>)
流用元のプロジェクト(<u>P</u>):	(流用元のプロジェクト・ファイルを入力してください) <
🗆 プロジェクト・フォルダ以下の構	成ファイルをコピーして 流用する(<u>O</u>)
	作成(<u>©</u>) キャンセル ヘルプ(<u>H</u>)

「ツール」→「プラグインの管理」→「追加機能」で「コード生成プラグイン」「端子配置プラグイン」 にチェックが入っていることを確認、無い場合入れます。

次にコード生成(設計ツール)→クロック発生回路をクリックします。 クロック発生回路より順次設定していきます。

プロジェクト・ツリー	ф.	×
2 🕜 🙎 🔳		
ba2_3chPWM_RL78 (プロ:	ジェク	1
R5F107DE (マイクロコン)	-0-	ラ)
🎤 端子配置 (設計ツール)		
一國 端子配置表		
端子配置図		
コード生成 (設計ツール)		
- 6 クロック発生回路		
─ ┛ポート		
ー・シリアル		
ー [®] A/Dコンバータ		
- ⁻ 917		
リアルタイム・クロック		
インターバル・タイマ		
ー [©] DMAコントローラ		
└─● プログラマブル・ゲイン	 アン 	ップ
▲CA78KOR (ビルド・ツール)	
➡ RL78 シミュレータ (デバッ	グ・	ツ-
プログラム解析 (解析ツー)	L)	
[▶] ファイル		
📑 スタートアップ		

ここでは高速メインモード2.7V~5.5Vを選択し、内蔵32MHzクロックを選択します。

	コードさ	E成*		
📓 端子配置へ反映 当 コード生成(G) 🍶 🎲	a 3	A. Ø & □ Ø ♣ □ ₱ ₱		
端子割り当て設定 クロック設定 オンチップ・デバッグ設定	定リセ	ット要因確認 安全機能		
- 動作モード 設定 -				
◎ 高速メイン・モード 2.7(V) ≤ VDD ≤ 5.5(V)		◎ 高速メイン・モード 2.4(V) ≤ VDD	≦ 5.5(∨)	◎ 低速メイン・モード 1.8(V) ≦ VDD ≦ 5.5(V)
-ソース・クロック選択設定				
 高速オンチップオシレータクロック(fIH) 		高速システム・クロック(fMX)		
- 高速オンチップオシレータクロック設定				
☑ 動作	周波数	32	▼ (MHz)	
- 高速システム・クロック設定				
🥅 動作				
④ ×1 発振(f×)		◎外部クロック入力(fEX)		
周波数		5	(MHz)	
発振安定時間		52428.8 (2°18/fMX)	(su)	
-PLL 出力クロック(fPLL)設定				
動作 月	司波数	64	(MHz)	
-メイン・システム・クロック(fMAIN)設定				
メイン・システム・クロック (fMAIN)		32000 (f1H)	▼ (kHz)	
-サブシステム・クロック(fSUB)設定				
動作				
④ XT1 発振(fXT)		◎ 外部クロック入力(fEXS)		
周波数		32.768	(kHz)	
XT1 登振回路の登振モード違択		低酒書發振	*	

オンチップ・デバック設定は使用するを選択

端子割り当て設定	クロック設定	ブロック図	オンチップ・	・デバッグ設定	リセット要因確認	安全機能	データ・フラッシュ
-オンチップ・デバッグ 重	协作設定 ——	5	ά.				
◎ 使用しない				◎ 使用	する		

他はRL78学習ボードでは特に変更する必要はありません。

次にポートの設定を行います。端子を入力で使うのか、出力で使うのか等を設定します。ポート以外で使 用する場合、使用しないを選択します。例PO3→RXD1で使用するので、ポートとしては使用しない。 後に説明するシリアルでアクティブにしているので、警告が表示されます。

2017ティー 端子配置表 増コード	"生成★】 📽 端子配置図 🖉 r_main.c
3 端子配置へ反映 3 コード生成(() ポート0 + ト1 + ト0 + ト2 + ト	G) 👗 🗊 🖋 🖉 💪 🔞 ֎ 🗐 🧐 💑 🔓 🦚 🥠
-P02	*4 M=1*7 M=1*12 M=1*13 M=1*14 M=1*20
 ● 使用しない ● 入力 ● 出力 	内蔵プルアップ N-ch 1 N-ch 1
 ● 使用しない ○ 入力 ● (000000000000000000000000000000000000	01:以下の端子と競合しています。この機能を使用する場合は競合する機能の設定を無効にしてください。 12はTxD1で使われています。
● 使用しない ○ 入力 ○ 出力 [-P06	 内蔵プルアップ 1
◎ 使用しない ◎ 入力 ◎ 出力 [□ 内蔵ブルアップ □ 1

次にシリアルを設定します。PO3, PO2のRXD1、TXD1を使用しますから、UART1、送信 /受信機能を選択します。データビット長、ボーレート等それぞれ送受信で設定がありますので、注意願 います。

/ 🚰 プロパティ 📑	端子配置表* 💕	端子配	置図* *		-15	生成	*		
🔣 端子配置へ反明	👷 📲 コード生	成(G)	📥 📬	<u>ي</u>	5	<u>6</u>	٢	8	
SAUO SAU4 IIC.	AO								
チャネル UARTO	UART1 CSI00								
- 機能									
チャネルロ	使用しない	•							
チャネル1	使用しない	•							
チャネル2	UART1	•	送信/	受信	機育	Ĕ		•	
チャネル3	使用しない	-							

📓 端子配置へ反映 当 コード生成(G	s) 🔬 💷 📽 🍠 💊 🖉 🖉 🗸
SAU0 SAU4 IICA0	
チャネル UARTO UART1 CSI00	
受信 送信	
-データ・ビット 長設定	
⑦ 7ビット ③ 8ビット	
-データ転送方向設定	20
LSB	MSB (
-パリティ設定	
💿 パリティなし 🛛 🔘 ୦パリティ	◎ 奇数パリティ ◎ 偶数パリティ
-ストップ・ビット 長設定	
1ビット固定です	
-受信データ・レベル設定	
④ 標準	🔘 反転
-転送レート設定	
ボー・レート	38400 ▼ (bps) (誤差:+0.16%
-割り込み設定	
受信完了割り込み設定(INTSR1)	低 🔻
🔲 Iラー割り込み設定(INTSREI)	低 🔻
-コールバック機能設定	
☑ 受信完了	✓ 15-
A	

以下は上記設定し「コード生成」後、PO2、PO3を入力とか出力に設定しようとした場合に出される 警告。設定できません。

出力 INTPOIdP137で使われています。↓ ₩0403001:以下の端子と競合しています。この機能を使用する場合は競合する機能の設定を無効にしてください。↓ RxD1はP03で使われています。↓ TxD1はP02で使われています。↓ 次にA/Dコンバータを設定します。今回は基準電圧(+)=電源電圧、基準電圧(-)=GND電位、ソ フトウエアトリガモードです。コンパレーター動作「許可する」。

	and have been been been been been been been be	and the second	1000
端子配置へ反映 9	🏐 コード生成(G) 🍰 🗊 💓 🦨	🕰 🔞 🖉 🔲 🚳 🚠 🗅 独	131
-A/Dコンパータ動作設定			
🔘 使用しない		◎ 使用する	
コンパレータ動作設定			
🔘 停止		④許可	
分解能設定			
10ビット		8ビット	
VREF(+)設定			
VDD	O AVREFP	◎ 内部基準電圧	
VREF(-)設定			
VSS		O AVREFM	
トリガ・モード 設定 ―――			
◎ ソフトウエア・トリカ	ブ・モ ード		
🔘 ハードウェア・トリナ	ゴ・ノーウエイト・モード		
◎ ハードウェア・トリナ	ゴ・ウエイト・モード		
INTERNO1	•		
動作モード設定			
◎ 連続セレクト・モー	-**	◎ 連続スキャン・モード	
◎ ワンショット・セレク	ル・モード	◎ ワンショット・スキャン・モー	-1*
ANIO - ANI2, (PGA	AOUT), ANI4 - ANI7アナログ入力端子記	定 ANIO - ANI2, (PGAOU	T), ANI4 - ANI7
ANI16 - ANI19アナ	ログ入力端子設定		
🔽 ANI1 6	📝 ANI1 7	📝 ANI18	📝 ANI1 9
変換開始チャネル設	定	ANIO	

b-1:A/D設計上の注意点

1点注意しなくてはならないのが、例えばANI5を使いたい場合、ANIO~ANI2、ANI4もA / Dコンバータ入力になってしまうので、ハード設計時に注意が必要です。 (選択できるA/D入力)

94(1)2200	
ANIO – ANI2, (PGAOUT), ANI4 – ANI	17
ANIO – ANI2, (PGAOUT), ANI4 – ANI	6
ANIO – ANI2, (PGAOUT), ANI4 – ANI	5
ANIO – ANI2, (PGAOUT), ANI4	
ANIO – ANI2, (PGAOUT)	
ANIO - ANI2	
ANIO - ANI1	
ANIO	
すべて デジタル	

特に割り込みは使用しません。

ANI16 - ANI19アナログ入力端子設定		
🔲 ANI1 6 🧕 👘 🔲 ANI1 7 🥥	🕅 ANI1 8 0	📃 ANI1 9 0
変換開始チャネル設定	ANIO	•
変換時間設定		
基準電圧	3.6 ≦ VDD ≦ 5.5	• ()
変換時間モード	標準1	•
変換時間	38 (1216/fCLK)	▼ (µ
	あげ みつ)た 祭井	
変換結果上限/下限値設定 ◎ ADLL≦ADCRH≦ADULで割り込み要求信号(I	NTADを発生	
変換結果上限/下限値設定 ● ADLL≦ADCRH≦ADULで割り込み要求信号(I ● ADUL <adcrhまたはadll>ADCRHで割り込</adcrhまたはadll>	INTAD)を発生 み要求信号(INTAD)を発生	
変換結果上限/下限値設定 ● ADLL≦ ADCRH≦ ADULで割り込み要求信号(I ● ADUL< ADCRHまたはADLL> ADCRHで割り込 上限値(ADUL)	INTAD)を発生 み要求信号(INTAD)を発生 255	
変換結果上限/下限値設定 ● ADLL≦ ADCRH≦ ADULで割り込み要求信号(I ● ADUL< ADCRHまたはADLL> ADCRHで割り込 上限値(ADUL) 下限値(ADUL)	NT AD)を発生 み要求信号(INT AD)を発生 255 0	
変換結果上限/下限値設定 ● ADLL≦ADCRH≦ADULで割り込み要求信号(I ● ADUL <adcrhまたはadll>ADCRHで割り込 上限値(ADUL) 下限値(ADLL) 割り込み設定</adcrhまたはadll>	INTAD)を発生 み要求信号(INTAD)を発生 255 0	
変換結果上限/下限値設定 ● ADLL≦ ADCRH≦ ADULで割り込み要求信号(I ● ADUL< ADCRHまたはADLL> ADCRHで割り込 上限値(ADUL) 下限値(ADLL) 割り込み設定 ■ A/Dの割り込み許可(INTAD)	NTAD	

タイマ、ウオッチドグタイマ、リアルタイムクロックは特に使用しません。ウオッチドグタイマはデホル トで「使用する」がチェックされていますので、「使用しない」をチェックします。これを忘れると定期 的にリセットが発生しますので、注意して下さい。

- リオッナトック・タイマ 動17F設定 - ② 使用しない	◎ 使用する
-HALT/STOP/SNOOZE モード	時の動作設定
④ 許可	◎ 停止
-オーバフロー時間設定	
オーバフロー時間	4369.07 (2°16/fIL) 🔹 (ms)
-ウインドウ・オープン 期間設定 —	
ウインドウ・オープン期間	100 💌 (%)
- 割り込み設定	4
☑ オーバフロー時間の75%到	達時にインターバル割り込みを発生する(INTWDTI)
優失順位	(IF.

インターバルタイマを使用する場合、以下のように設定します。

例として1msecに1回、割り込みが入るインターバルタイマを設定します。

/ 🚰 プロパティ 📑 端子	子配置表* 💕 端子酮	记置図* 📲 🗉	コードさ	主成*							
📓 端子配置へ反映	当 コード生成(G)	🦾 💷 💕	5	<u>a</u> Ø	2		٩	aio		12	M
-インターバル・タイマ動作 ◎ 使用しない -インターバル時間設定	F設定	◉ 使用する									
インターバル時間		1		ms	•) (実	際の	値:	1)		
-割り込み設定 ―――											
🛛 📝 インターバル信号	検出(INTIT)										
優先順位		低			•]					

電圧検出回路の電圧を設定します。本基版は電源ON時のリセットを内蔵のパワーオンリセット回路(POR)と電圧検出回路(LVD)の併用で行っています。PORだけですと、1.51Vでリセット解除 されますので、LVDで電圧を設定します。一般に、電池動作(3.0Vあたり)で動作させる場合はP ORのみ、3.3Vや5Vで動作させる場合はLVDを設定してリセット電圧を2.81V~で動作させ ます。デフォルトでは「使用しない」になっていて、「使用する」に変更しないと、デバック中は問題な くても、単独で動作させたときに、電源投入時、不安定な動作になります。



上記設定を終えたら「コード生成」をクリックします。



r_main.cを初め、設定内容を反映したソースファイルが生成されます。

プロジェクトをリビルドしてみます。

・ル(T) ウインドウ(W) ヘルプ(H)	
00% - 🗟 🚱 🔨 🐂 🔍 🕲 🖉 🖓	3 K
<mark>プロジェクトをリビルドします。 (Shift</mark> : 置図/雪コード生成/雪 r_main.c	+F7)
ビルド リビルド デバッカにダウンロード	

--ビルド後、デバッカにダウンロード

ビルド終了(エラー:0個、警告:0個)と出れば正常です。

出力
>r_cg_it.cվ >r_cg_it_user.c」 >DefaultBuild¥test_RL78107.lmfJ >DefaultBuild¥test_RL78107.hexJ ビルド終了(エラー:0個,警告:0個)J ======= 終了しました(成功:1ブロジェクト,失敗:0ブロジェクト)(2013年8月12日 15:57:50) ========J ຢ
[EUF] ↓すべてのメッセージ

b-2:自動生成されたプログラム

以下省略

Systeminit(void)関数の中身

R_PORT_Create(); //ポート初期化 R_CGC_Create(); //クロック初期化 ; R_ADC_Create(); //A/Dコンバータ初期化 ; などが自動生成され、電源ON時に自動的に実行されます。

b-3:E2Lから電源供給

デバッカにはE2Lを使用します。E2Lから電源を供給する設定は

プロジェクト・ツリー 🛛 🕂 🗙	TO/Fr Mine	
2 @ 2 2	SI 18 E2 Lite 012018 T 4	
■ ■		64 4095 4 内蔵クロックを使用する 内蔵クロックを使用する シスラム
D アイル Stantasm Stantasm Stantasm Stantasm D フィル Stantasm D フィル restantasm D フィル restantasm C 「restemintc C 」	 ▼ 121 レージの押除る: T31 レージの押からの T31 レージの押からの T31 レージの構築 T31 レージのを通知になるる(最大200mA) 供給電圧(い) マラックション セネコジテルロ マラックション セネコジテルロ Tフラッション 建参換える許可する フィン・20・モードを使用する 起動時に、フラッシュ ROMを消去する 	はい 33V 回回 000000000000000000 はい はい といえ

RL78E2Lite(Serial)(デバック・ツール)を右クリック→プロパティで上記画面にな りますので、エミュレータから電源を供給するを「はい」、電圧は3.3V固定です(E1は3.3Vと 5Vが選べました)。外部電源を使用する場合、ダウンロード前に電源をONさせる必要があります。

ビルド後、ダウンロードを行いE2Lとうまく通信が出来るとデバックのためのボタンがアクティブになります。

CPUリセット 現在の位置からプログラム動作

CPUリセット後、動作をクリックするとプログラムが初めから動作します。

b-4:コード生成後の初期値の変更

「コード生成」後、プログラムをある程度書いた後の仕様の変更に、再び「コード生成」を行うと、既に プログラムを書いた部分が初期化されて消えてしまいます!

そうならないためにStart user code、、、とEnd user code、、、の間に ユーザープログラム、コメント、定数、変数宣言等を書いてください。

/* Start user code for global. Do not edit comment generated here */

ここにユーザーコードを書けば、「コード生成」を繰り返しても、消されることはありません!!

/* End user code. Do not edit comment generated here */

b-5:変数を見る



R_ADC_Start();		//А	D変換開始
R_AUC_Get_Result(ad_dat	1	ウォッチ1 に登録(R)	
	<u>14</u>	解析グラフに登録(E)	
P3 0 = 1:	4	アクション・イベントの登録(A)	
P3.0 = 0;	×	切り取り(T) Ctrl+X	

見たい変数をコピーして右クリック→ウオッチ1に登録

ウオッチ1			4 ×
2 🔍 🐉 🗓 🗙	表記(N)- 😹		
ウォッチ式	値	型情報(バイト数)	アドレス メモ
🛯 ad_data	0 (0x0000)	unsigned short(2)	OxfefdO

b-6:変数変化を実行中に確認する

ウォッチ式		値	型情報(バイト数)	アドレス メ
👻 ad_data	0	(0x0000)	unsigned short(2)	OxfefdO
adcs 📷		0×0	SFR[R/W 1](1ビッ…	Oxfff30.7
🗊 ADMO		0x00	SFR[R/W 1.8](1)	0xfff30
🗊 ADM1		0x20	SFR[R/W 1.8](1)	0xfff32
📷 ADM2		0x00	SFR[R/W 1.8](1)	0xf0010
😜 loop		?	?	?
🖽 😜 ad_buffer		""	unsigned char…	Oxfefd4
👽 enc_p	0	(0x0000)	unsigned short(2)	0xfefe8
😔 enc_m	0	(0x0000)	unsigned short(2)	Oxfefea
🗊 PM20		0x80	SFR[R/W 1.8](1)	0xf0510

ウオッチウインドウはデバックに非常に便利な窓ですが、初期設定では動作中は更新されません。そこで、

\[\begin{aligned} & \vee \equiv \equi		
プロジェクト・ツリー 🛛 🕈 🗙	/ 🔊 逆アセンブル/ 🥤 r_main.c/ 🥤 r_cg_intc_user.c/ 📓 r_cg_i	intc.ç/ 🗹 r_cg_it.ç/ 🗹 r_systeminit.ç/ 🗹 r_cg
2 3 2 2	🔐 RL78 E1(Serial) のプロパティ	
ba2_3chPWM_RL78 (プロジェク ▲ ■ R5F107DE (マイクロコントロー タ 端子配置 (設計ツール) ■ コード生成 (設計ツール) ▲ CA78KOR (ビルド・ツール) ■ RL78 E1(Serial) (デパッグ・ツー ④ プログラム解析 (解析ツール) ■ ファイル ■ ジルド・ツール生成ファイル ■ スタートアップ ■ コード生成 - ● r_main.c	 ★ メモリ メモリ・マッピング メモリ書き込み時にパリファイを行う ★ 実行中のメモリ・アクセス 実行を一瞬停止してアクセスする 実行中に表示更新を行う 表示更新間隔[ms] ブレーク ▲ 入力信号のマスク TARGET RESET 信号をマスクする INTERNAL RESET 信号をマスクする 	[9] はい はい はい いいえ いいえ

RL78 E2L(Serial)(デバック・ツール)→プロパティ→デバックツール設定で実行中の メモリアクセス、表示更新を「はい」にすると、実行中でも変数の変化が確認できます。

下記例はsprintfでad_buffにeep_dataの値が格納されるのをリアルタイムで表示 しています。

into_user.c	r og intc.c 🛛 🗹 r og it.c 🖓 r systeminit.c 🖓 r og i	t_user.c / 🗹 r_cg_serial.c / 👔	2 JUNET	- x	ウォッチ1		4
					🔳 🧶 🤮 🗙 🛪	表記(N)- 阙	
					ウォッチ式	値 型情報(バイト数)	アドレス >
					🔍 ad_data	0 (0x0000) unsigned short(2)	0xfefd0
prom test					ad CS	0x0 SFR[R/W 1](1ピッ…	0xfff30.7
					📆 ADMO	0x01 SFR[R/W 1.8](1)	0xfff30
	eep init():				📷 ADM 1	0x20 SFR[R/W 1.8](1)	Oxfff32
					📷 ADM2	0x00 SFR[R/W 1.8](1)	0 x f 00 1 0
	eep_wr16(0_250);	//TEMP	25 0°C		S loop	??	?
	een_wr16(2,500);	//THICK	500pm		🖃 👻 ad_buffer	"" unsigned char…	Oxfefd4
	$eep_wr16(4, 100);$	//CAL	1 00		0]	'e' (0x65) unsigned char(1)	Oxfefd4
	ccp_wiio(4,100),	TTOAL	1.00		€ [1]	'e' (0x65) unsigned char(1)	0xfefd5
	$\cos data = \cos cd16(0)$				€ [2]	'p' (0x70) unsigned char(1)	Oxfefd6
	eep_uata - eep_lulu(0),	sintf(ad buffas "app = V(dVoVr" app data); (/ad buff(=10)#/		T:	😜 [3]	''(0x20) unsigned char(1)	Oxfefd7
	sprinti(ad_burrer, eep - %4d#n#r,eep_da	if ,eep_data);		ASUTT	🥥 [4]	'=' (0x3d) unsigned char(1)	0xfefd8
	K_UARTI_Send(ad_butter,sizeot(a	ad_butter));	//uart=//		€ [5]	''(0x20) unsigned char(1)	0xfefd9
	tx_end_wait();			=	0] 😡	''(0x20) unsigned char(1)	Oxfefda
					🥥 [7]	'2' (0x32) unsigned char(1)	Oxfefdb
					(8]	'5' (0x35) unsigned char(1)	Oxfefdc
while(1)				🥥 [9]	'0' (0x30) unsigned char(1)	0 x f e f d d
					😜 [10]	'' (OxOa) unsigned char(1)	Oxfefde
					🥥 [11]	'' (OxOd) unsigned char(1)	Oxfefdf
					€ [12]	'' (0x00) unsigned char(1)	OxfefeO
					🥥 [13]	'' (0x00) unsigned char(1)	Oxfefe1
/* Start us	ser code. Do not edit comment ger	herated here */			€ [14]	'' (0x00) unsigned char(1)	0xfefe2
while (111)	ber bedet be not bart commert ger	ioracou noto "y			😜 [15]	'' (0x00) unsigned char(1)	Oxfefe3
1					💊 [16]	'' (0x00) unsigned char(1)	Oxfefe4
1					😜 [17]	'' (OxOO) unsigned char(1)	Oxfefe5

なお、使用端子や動作プログラムが同じような構成のものの場合、ホルダをコピーし、ホルダ名、m t p j ファイルの名前を変更すれば、それまでの設定はそのまま使えます。変更もその上から行うことが出来ます。

2. サンプルプログラム

2-1 sample1 キー入力でLEDを点灯

【 概要 】

LED1, 2, 3, 4が点滅し、SW1キーを押すと消灯します。マイコンの基礎であるポートの入出力 を学習します。電源はE2Lから供給しますので、USBケーブルは接続しないでください。以下の写真 ご参照。





【 ハードウエア 】

キーはRA1でプルアップされており、入力ポートP121, 122, 123, 124は押されないとき

1 (+3.3V)、押されたとき0 (0V GNDレベル)になります。出力ポートP23,25,26, 27はトランジスタバッファTD62003の入力に接続されていて、入力I × が1 (+3.3V)で出 力O × がON(出力が0V近傍まで落ち、LEDが点灯します)、0(0V GNDレベル)でOFF(出 力が+3.3V近傍まで上がり、LEDが消灯します)

x = 1 ~ 7

回路図に5Vとあるのは、USBケーブルをさして使用するsample2以降はUSBからの電圧を電 源にするので5Vになります。このsample1だけはE2Lから電源を供給するので3.3Vになり ます。

【 コード生成 ポートの設定 】

あらかじめ、ポートP121、122、123、124は入力に設定されています。

プロジェクト・ツリー 🛛 🕈	🔄 プロパティ 📲 コード 生成 📝 r main.c 💕 端子	子配置図 デ 端
2 🕜 🙎 🔳	■ 端子配置へ反映 ● コード生成(G)	511 ml A (
■ Sample1 (プロジェクト)	ポート0 ポート1 ポート2 ポート3 ポート4 ポ	-ト7 <u>ポート12</u>
■ A 端子配置 (設計ツール)	-P120 ④ 使用しない 〇 入力 〇 出力 〇 内慮	気プルアップ
□ ¹ コード生成 (設計ツール)	-P121	
€ 	-P122	
- シリアル	-P123	
● タイマ	-P124	

ポート24,25,26,27は出力に設定されています。



【 プログラム 】

```
①void main(void)
{
(2)
      R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
3
      while (1U)
    {
                    P2 = 0x00;
(4)
                                                   //LD1,2,3,4 OFF
5
                    wait(1000000);
                                                   //時間待ち
6
                    P2 = 0xf0;
                                                   //LD1,2,3,4 ON
                    wait(100000);
                                                   //時間待ち
```

```
while(P12.1 == 0) //SW1 が押されているとループ
{
P2 = 0x00; //LD1,2,3,4 OFF
}
```

```
}
```

(7)

/* End user code. Do not edit comment generated here */

}

【 解説 】

①void main(void)

メイン関数です。電源ONで自動的に実行されます。

② R_MAIN_UserInit();

コード生成によって自動的に作られた初期設定関数をコールしています。この初期設定はメインルーチン の下(同じファイル)のところにあります。

③ while (1U)

以下の { }の中を無限ループします。10は数字の1で、unsigned型であると明示しています。

(4) P2 = 0x00; //LD1,2,3,4 OFF

wait(1000000);

P2 = 0xf0;

ポート2の出力データを0にしています。P2.4、P2.5、P2.6、P2.7が0になるので、L EDが4つ共消灯します。

バイト	P 2							
ビット	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
2進数	0	0	0	0	0	0	0	0
表記								
16進	0				0			
数表記								

5

//時間待ち

人間の目でみて点滅が分かるように時間待ちをしています。100000という数を-1して、0になったら抜けます。

```
void wait(uint32 t wtime)
```

```
while(wtime != 0)
{
    wtime--;
}
```

}

{

6

//LD1,2,3,4 ON

ポート2の上位4ビット出力データを1にしています。P2.4、P2.5、P2.6、P2.7が1に なるので、LEDが4つ共点灯します。

バイト	P 2							
ビット	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

重み	8	4	2	1	8,	٨	4	2	1
2進数	1	1	1	1	0		0	0	0
表記									
16進	f				0				
数表記									

2進数1111が16進数ではfになる理由ですが、4ビット毎にデータが1の重みを加算します。 P2.7は重み8、P2.6は4、P2.5は2、P2.4は1 ですので、8+4+2+1=15となります。P2.3~P2.0は0です。

1	0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1
0											0	1	2	3	4	5
進																
数																
1	0	1	2	3	4	5	6	7	8	9	а	b	с	d	е	f
6																
進																
数																

10進数表記で15は16進数表記ではf(F)になります。なので、0xf0と書きます。頭に付ける 0xは以降の数が16進表記であることを表しています。

 $\overline{\mathcal{O}}$

while(P12.1 == 0) //SW1 が押されているとループ { P2 = 0x00; //LD1,2,3,4 OFF }

SW1は押さないときは何もしません。P12.1の配線がRA1の10KΩでプルアップ(電源に吊ら れること)されていますので、電圧は+3.3Vとなり、入力したデータは1となります。

SW1は押されるとGNDと接続されますので、OV、入力データはOとなります。 プログラムでは押されたとき、データOのときにwhileで無限ループとなり、P2にOを出していま すので、LEDが全て消灯します。離すと点滅が再開します。

【 演習 】

sample1で動作を理解できた方は、演習問題に進んで下さい。課題は キー入力で押したところのLEDを点滅させて下さい。

回答例 sample1_a

回答例は例であり、課題に合致していれば、書き方がこの通りである必要はありません。

以下省略

それぞれはそれぞれの会社の登録商標です。

フォース及びFORCE®は弊社の登録商標です。

- 1. 本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。
- 2. 本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。
- 3. 本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。
- 4. 本文章は許可なく転載、複製することを堅くお断りいたします。

お問い合わせ先:

〒350-1213 埼玉県日高市高萩1141-1
TEL 042(985)6982
FAX 042(985)6720
Homepage: http://beriver.co.jp
e-mail: info@beriver.co.jp
有限会社ビーリバーエレクトロニクス ©Beyond the river Inc. 20190204