

RL78/I1Eコード生成学習

2016/6/1 Rev. 1.00

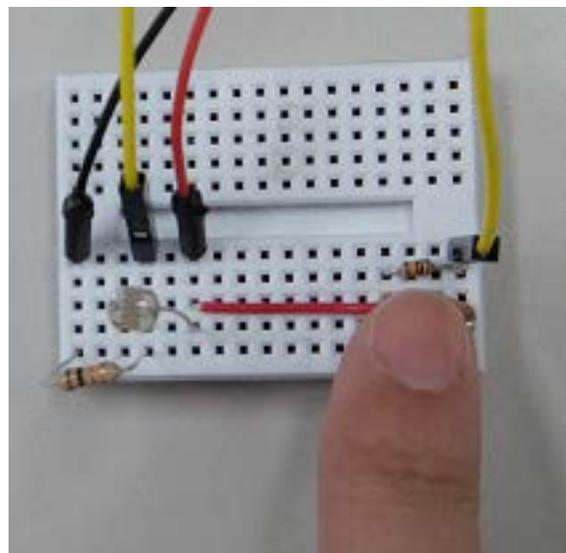
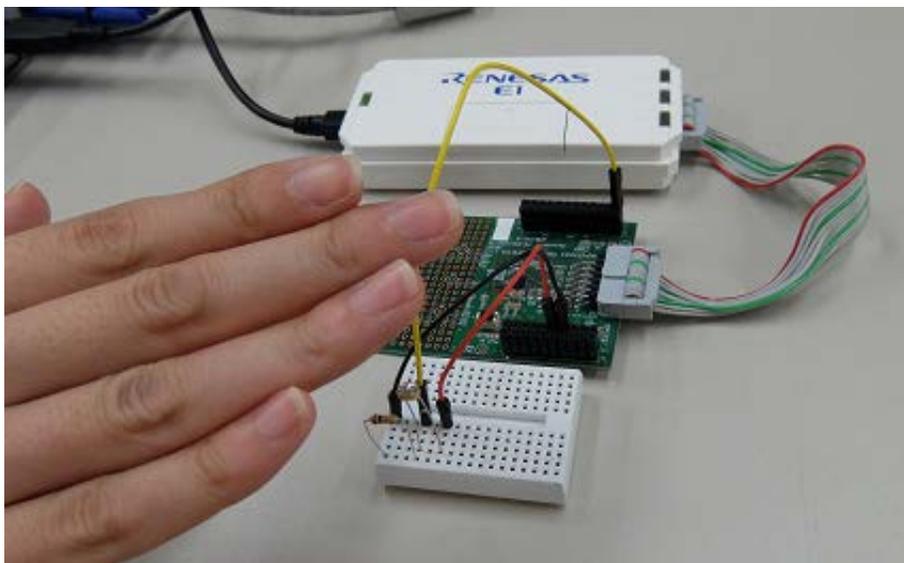
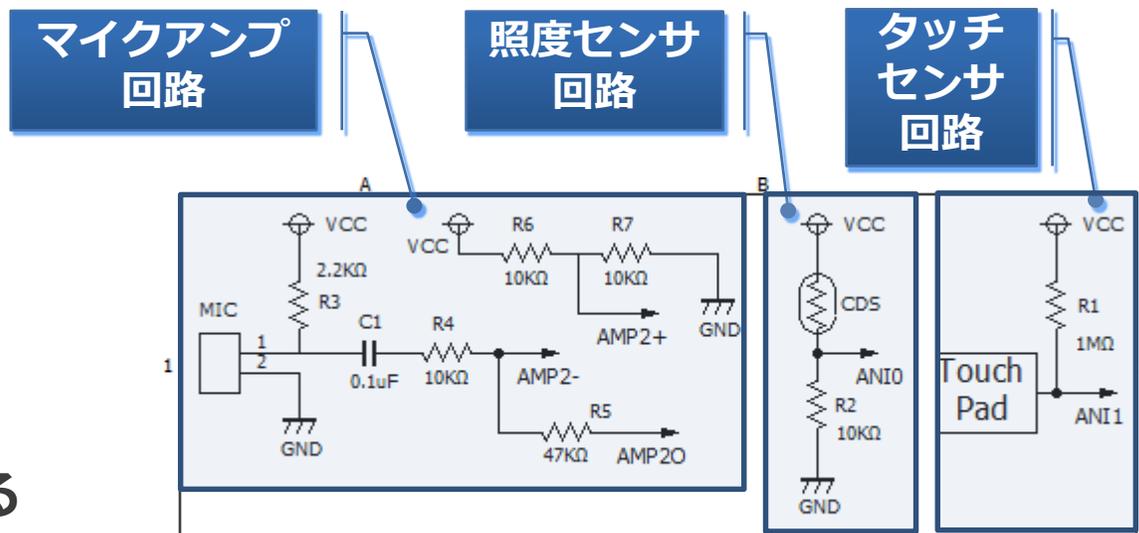
ソフトウェア事業部 ソフトウェア技術部

ルネサスシステムデザイン株式会社

R20UT3809JJ0100

演習内容

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



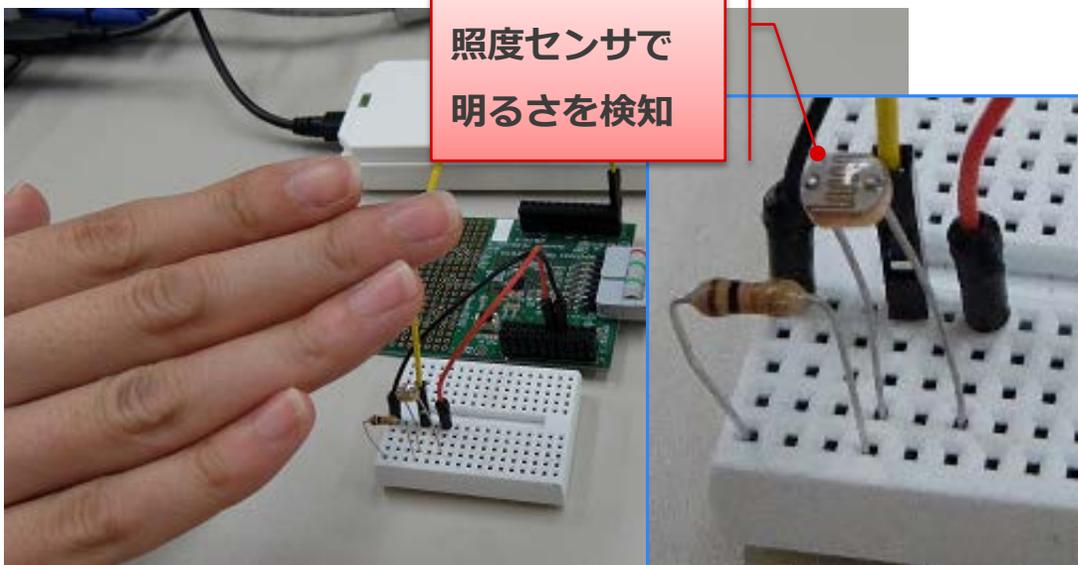
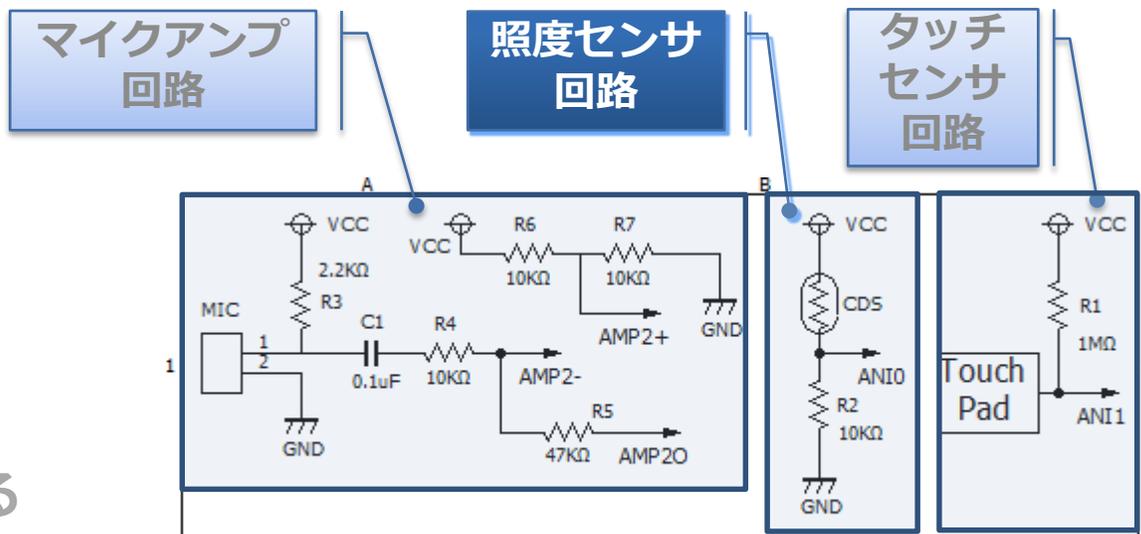
目次

- 演習A. 照度センサを使って明るさを検知
- 演習B. 簡易タッチセンサでLEDをON/OFF
- 演習C. マイクを使って音でLEDの輝度を変える

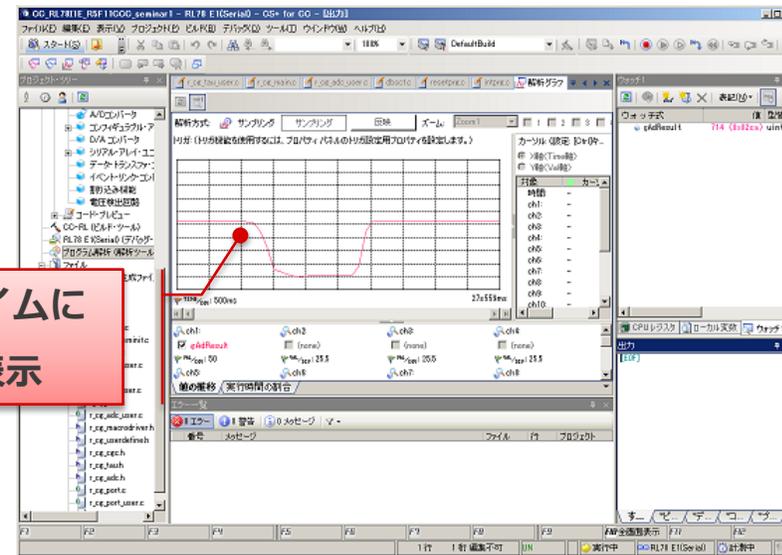
▪はじめに	ページ 02	
▪演習内容	ページ 03	
▪演習A	ページ 05	演習A. 照度センサを使って明るさを検知
▪プロジェクト作成	ページ 06	
▪周辺機能設定	ページ 10	
▪コード生成とプログラム編集	ページ 15	
▪プログラム実行とデバッグ	ページ 22	
▪プログラムの解説	ページ 29	
▪演習B	ページ 30	演習B. 簡易タッチセンサでLEDをON/OFF
▪周辺機能設定	ページ 32	
▪コード生成とプログラム編集	ページ 35	
▪プログラム実行とデバッグ	ページ 40	
▪プログラムの解説	ページ 43	
▪演習C	ページ 44	演習C. マイクを使って音でLEDの輝度を変える
▪周辺機能設定	ページ 46	
▪コード生成とプログラム編集	ページ 50	
▪プログラム実行とデバッグ	ページ 56	
▪最後に	ページ 57	

演習A

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



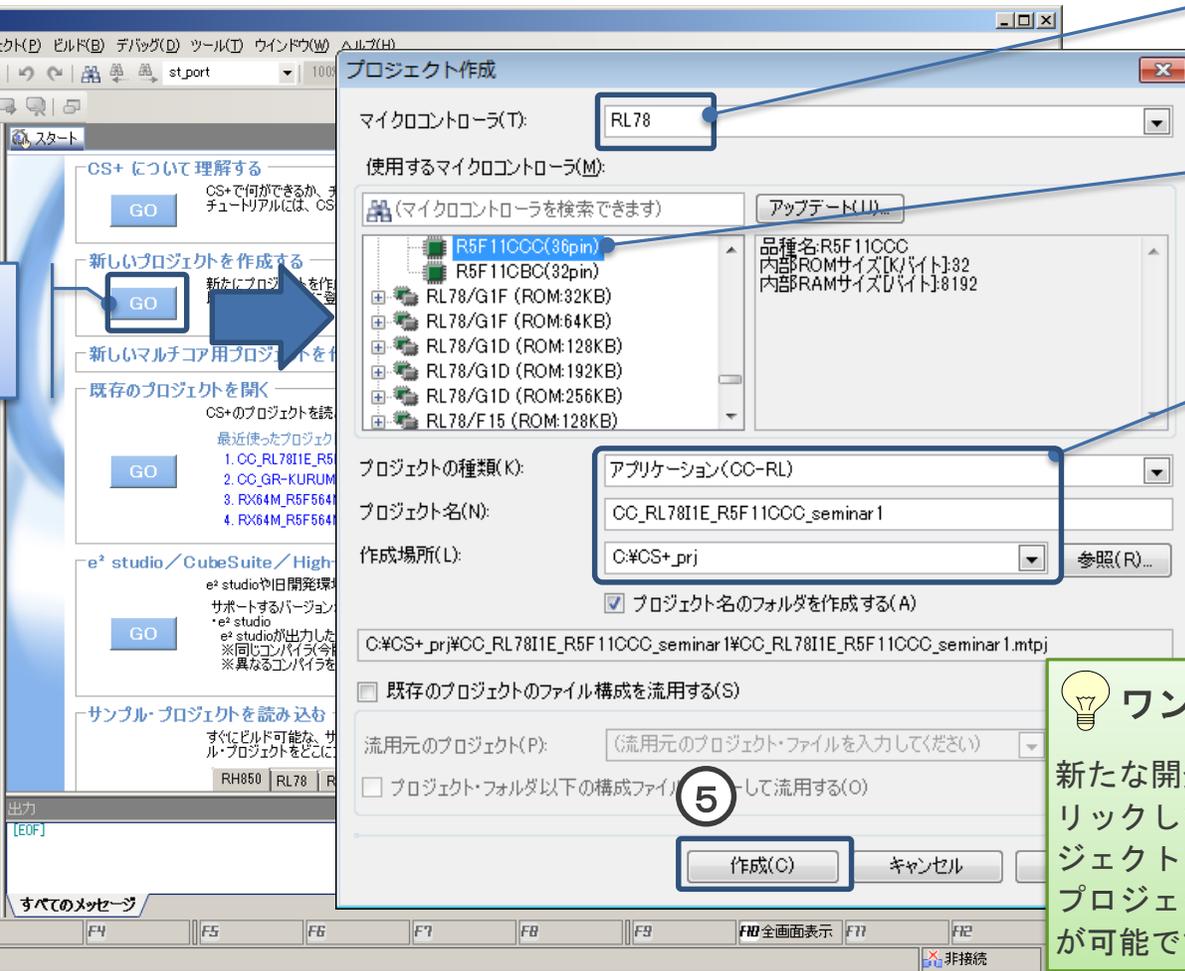
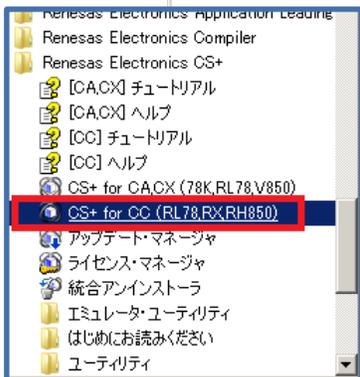
CS+上でリアルタイムに照度センサの値を表示



プロジェクト作成

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

① CS+起動後に新しいプロジェクトを作成



② 「RL78」を選択

③ 使用するマイクコントローラを選択
「RL78/I1E R5F11CCC(36pin)」

④ プロジェクトの種類を選択
「アプリケーション(CC-RL)」
プロジェクト名を入力
「CC_RL78I1E_R5F11CCC_seminar1」

💡 ワンポイント
新たな開発でCS+を使い始めるときは、**スタート(S)** ボタンをクリックしてください。スタート・パネルが表示され、新しいプロジェクトを作成したり、最近使ったプロジェクトや、お気に入りのプロジェクトを開いたりなど、簡単にプロジェクトを作成/開くことが可能です。

プロジェクト作成(プラグインの管理)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

7

使用するプラグインにチェックを入れる
「コード生成プラグイン」
「コード生成/端子図プラグイン」
「プログラム解析プラグイン」
「端子配置図プラグイン」

8

「OK」を選択、
表示に従い製品を再起動する

6
ツール → プラグインの管理
を選択

プロジェクトの情報を保存するファイルの名前です。

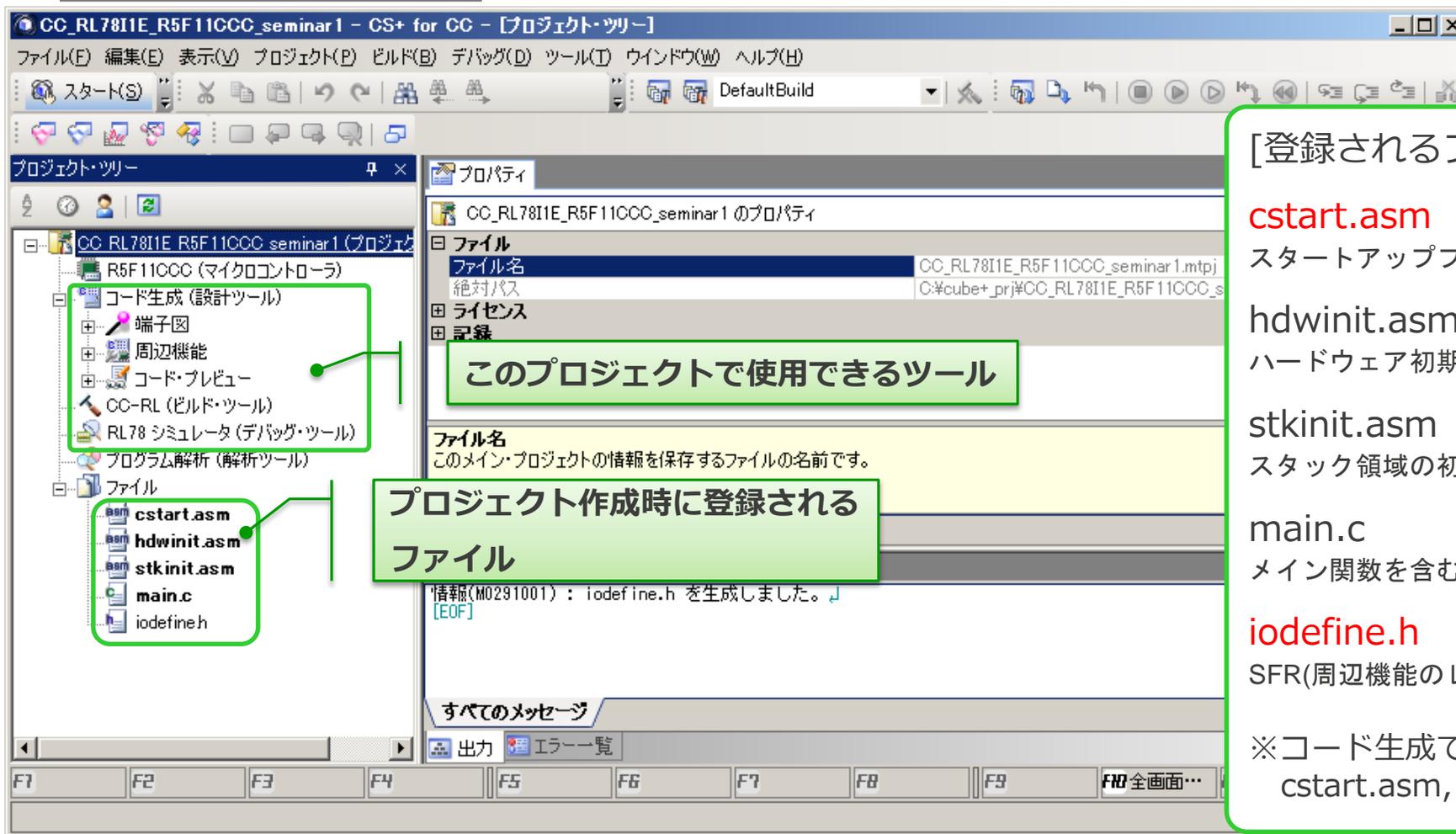
モジュール名	説明
<input checked="" type="checkbox"/> IronPythonコンソール・プラグイン	IronPythonのコマンドとCS+拡張機能が使
<input checked="" type="checkbox"/> Quick and Effective tool solution - QE	アプリケーション開発に便利なツールをセッ
<input checked="" type="checkbox"/> RH850用コード生成	デバイスドライバを自動生成および端子配
<input checked="" type="checkbox"/> アップデート・マネージャ・プラグイン	CubeSuite+ アップデート・マネージャと連携
<input checked="" type="checkbox"/> エディタ・パネル	エディタ・パネルのプラグインです。
<input checked="" type="checkbox"/> コード生成プラグイン	デバイスドライバを自動生成するプラグインです。(V850, 78K0, 78K0R, RL78/G12,
<input checked="" type="checkbox"/> コード生成/端子図プラグイン	デバイスドライバを自動生成および端子配置を表示するプラグインです。(V850, 78K0,
<input type="checkbox"/> スタック見直しツール	スタック使用量をツリー形式で表示するツールです。
<input type="checkbox"/> デバッグ・コンソール・プラグイン	標準I/Oをサポートするデバッグ・コンソール・プラグインです。
<input checked="" type="checkbox"/> プログラム解析プラグイン	プログラムの解析を行うプラグインです。
<input type="checkbox"/> リアルタイムOSタスク・アナライザ・プラグイン(共通部)	リアルタイムOSが組み込まれたプログラムの解析を行うプラグインです。
<input type="checkbox"/> リアルタイムOSビルド設定プラグイン(共通部)	リアルタイムOSのビルド情報を設定するプラグインです。
<input type="checkbox"/> リアルタイムOSリソース情報表示プラグイン(共通部)	リアルタイムOSの資源情報を表示するプラグインです。
<input type="checkbox"/> リアルタイムOS解析制御プラグイン(共通部)	リアルタイムOS情報の解析と管理をするプラグインです。
<input checked="" type="checkbox"/> 端子配置プラグイン	デバイスの端子配置を行うプラグインです。

ワンポイント

コード生成ツールに関連するプラグイン、プログラム解析プラグインは、初期状態では使用しない設定になっているため、コード生成と解析グラフは表示されません。

プロジェクト作成 (プロジェクト作成時に登録されるファイル)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



[登録されるファイル一覧]

cstart.asm

スタートアップファイル。リセット後に最初に実行される。

hdwinit.asm

ハードウェア初期化処理を行う。

stkinit.asm

スタック領域の初期化を行う。

main.c

メイン関数を含む。

iodefine.h

SFR(周辺機能のレジスタ)を定義したファイル

※コード生成で使うファイルは以下
cstart.asm, iodefine.h

プロジェクト作成 (コード生成の機能について)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

端子図配置表, 端子配置図:
マイコンの設定状態を示す

端子配置図

コード生成が設定可能な
周辺機能を表示

周辺機能設定

ワンポイント

コード生成は、周辺機能のドライバをCソース出力するツールです。USB、CANなどミドルウェアと組み合わせて使用する周辺機能には、対応していません。

周辺機能設定 (コード生成での設定について)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

2 選択された周辺機能の詳細設定を行う

1 周辺機能を選択して設定

周辺機能

- 共通/クロック発生回路
- ポート機能
- タイマ・アレイ・ユニット
- タイマRJ
- タイマRG
- リアルタイム・クロック
- インターバル・タイマ
- クロック出力/ブザー出力制御
- ウォッチドッグ・タイマ
- PGA + ΔΣ A/Dコンバータ
- A/Dコンバータ
- コンフィギュラブル・アンプ
- D/Aコンバータ
- シリアル・アレイ・ユニット
- データ・トランスファ・コントローラ
- イベント・リンク・コントローラ
- 割り込み機能
- 電圧検出回路

クロック設定

VDD設定

- 高速メイン・モード 4.0(V) ≤ VDD ≤ 5.5(V)
- 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(V)

メイン・システム・クロック (fMAIN) ソースの設定

- 高速オンチップ・オシレータ・クロック (fHOCO)
- PLL出力クロック (fPLL)
- 高速システム・クロック

24ビット ΔΣ A/Dコンバータの動作クロック (fDSAD) ソースの設定

- 高速オンチップ・オシレータ・クロック (fHOCO)
- PLL出力クロック (fPLL/2)
- 高速システム・クロック

RTCの動作クロック (fRTC) ソースの設定

- 停止
- 高速オンチップ・オシレータ・クロック (fHOCO)
- 高速システム・クロック

高速オンチップ・オシレータ・クロック (fHOCO) 設定

- 動作
- 周波数: 32 (MHz)

高速システム・クロック (fMX) 設定

- 動作
- X1発振 (fX)
- 外部クロック入力 (fEX)
- 周波数: 5 (MHz)
- 発振安定時間: 2¹⁸/fX (52428.8 μs)
- 48 (MHz)

このデモで設定する周辺機能

- ・ **共通/クロック発生回路** (デフォルト設定で使用)
→ メイン・システム・クロック、高速オンチップ・オシレータ・クロック (fHOCO)
高速オンチップ・オシレータ (fHOCO) 32MHz
CPUと周辺クロック fHOCOを使用
- ・ **タイマ・アレイ・ユニット**
→ 1ミリ秒のインターバルタイマを設定
- ・ **ウォッチドッグタイマ**
→ 未使用にする
- ・ **A/Dコンバータ**
→ A0端子のA/D変換を行う

周辺機能設定 (共通/クロック発生回路)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

③

**ダブルクリックで
周辺機能を選択**

④

**デフォルトで使用する
ので変更しない**

⑤

**オンチップ・デバッグ
設定する**

④

💡 ワンポイント

オンチップ・デバッグ設定は、CC-RL (ビルド・ツール) のリンク・オプション[デバイス]項目へ設定が反映されます。

周辺機能設定 (タイマ・アレイ・ユニット)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

⑦ チャンネル0、インターバル・タイマ選択

⑥ タイマ・アレイ・ユニットを選択

⑧ チャンネル0タブを選択

⑨ インターバル時間に1msを設定

ワンポイント

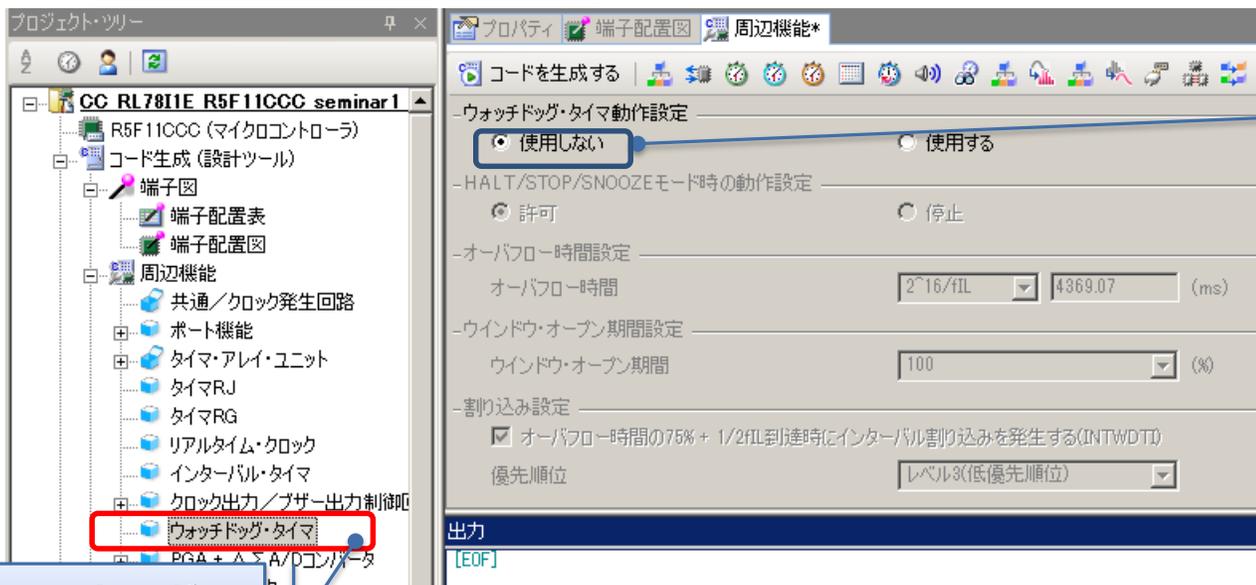
数値を入力する箇所では、マイコンの設定範囲を超えた場合、エラーを表示します。例えば、インターバル時間に20nsと入力すると設定範囲外となり、エラーを表示します。

エラーメッセージ: (入力値が不正です。)

設定範囲の情報: 63~67108864000

周辺機能設定 (ウォッチドッグ・タイマ)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



10
ウォッチドッグ・
タイマを選択

11
使用しない
にチェックする

 **ワンポイント**

プロジェクト・ツリー・パネルの各周辺機能のアイコンは、以下を意味しています。

 : 使用している機能、  : 未使用の機能、  : 設定内容に問題が発生、  : 設定内容の更新が必要

周辺機能設定 (A/Dコンバータ)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

The screenshot shows the '周辺機能' (Peripheral Function) settings for the A/D converter. The following steps are indicated by callouts:

- 12**: A/Dコンバータを選択 (Select A/D Converter)
- 13**: 使用するにチェックする (Check 'Use')
- 14**: 入力端子にANIOを設定 (Set ANIO as input terminal)
- 15**: 変換チャンネルにANIOを選択 (Select ANIO as conversion channel)
- 16**: $2.7 \leq AVDD$ を設定 (Set $2.7 \leq AVDD$)
- 17**: 他はデフォルトの設定 (Others are default settings)

💡 ワンポイント
 基準電圧を変えると、変換時間モードと変換時間の選択範囲が変更されます。電圧の入力が必要なのは、設定時間に影響するためです。

コード生成とプログラム編集

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

① コードを生成する を押下

② ファイルが生成されプロジェクトツリーに登録される

出力

```
M0409000:cg_src#r_cg_main.cを生成しました。↓
M0409000:cg_src#r_cg_systeminit.cを生成しました。↓
M0409000:cg_src#r_cg_macrodriver.hを生成しました。↓
M0409000:cg_src#r_cg_userdefine.hを生成しました。↓
M0409000:cg_src#r_cg_cg.cを生成しました。↓
M0409000:cg_src#r_cg_cg.c/hを生成しました。↓
M0409000:cg_src#r_cg_cg_user.cを生成しました。↓
M0409000:cg_src#r_cg_cg.hを生成しました。↓
M0409000:cg_src#r_cg_tau.cを生成しました。↓
M0409000:cg_src#r_cg_tau_user.cを生成しました。↓
M0409000:cg_src#r_cg_tau.hを生成しました。↓
M0409000:cg_src#r_cg_adc.cを生成しました。↓
M0409000:cg_src#r_cg_adc_user.cを生成しました。↓
M0409000:cg_src#r_cg_adc.hを生成しました。↓
M0409003:ファイルの生成を完了しました。↓
[EOF]
```

すべてのメッセージ *コード生成 *ラピッド・ビルド

生成されるファイル一覧

必ず生成されるファイル

- r_cg_main.c
- r_cg_systeminit.c
- r_cg_macrodriver.h
- r_cg_userdefine.h
- r_cg_cg.c /h
- r_cg_cg_user.c

使用する周辺機能に応じて生成されるファイル
以下は、タイマ・アレイ・ユニット、A/Dコンバータ
を設定すると出力されるファイル

- r_cg_tau.c /h
- r_cg_tau_user.c
- r_cg_adc.c/h
- r_cg_adc_user.c

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (生成されるファイルの説明)

生成ファイルのルール

r_cg_main.c	メイン関数
r_cg_systeminit.c	hdwinit()を含む周辺機能の初期化
r_cg_macrodriver.h	コード生成で使うマクロ/定数定義
r_cg_userdefine.h	ユーザが使うマクロ/定数定義
r_cg_cgc.c /h	
r_cg_cgc_user.c	周辺機能、クロックに関する設定
r_cg_周辺機能.c / .h	周辺機能の初期化と制御するAPIを含む
r_cg_周辺機能_user.c	周辺機能の割り込み関数を含む。 ユーザは割り込み関数内に処理を追加する。

ワンポイント

周辺機能を制御するAPIには、周辺機能の開始/停止や、結果の取得、データの送受信などがあります。

例えば r_cg_tau.c には、タイマを開始/停止するAPIが含まれます。

- ▶ タイマ開始 : R_TAU0_Channel0_Start()
- ▶ タイマ停止 : R_TAU0_Channel0_Stop()

ユーザが R_TAU0_Channel0_Start() を呼び出すとタイマが開始します。周期割り込みで呼び出される割り込みハンドラ内に処理を記述します。

- ▶ カウント完了 : r_tau0_channel0_interrupt()

このプロジェクトでは1ミリ秒ごとに上記関数が呼ばれます。

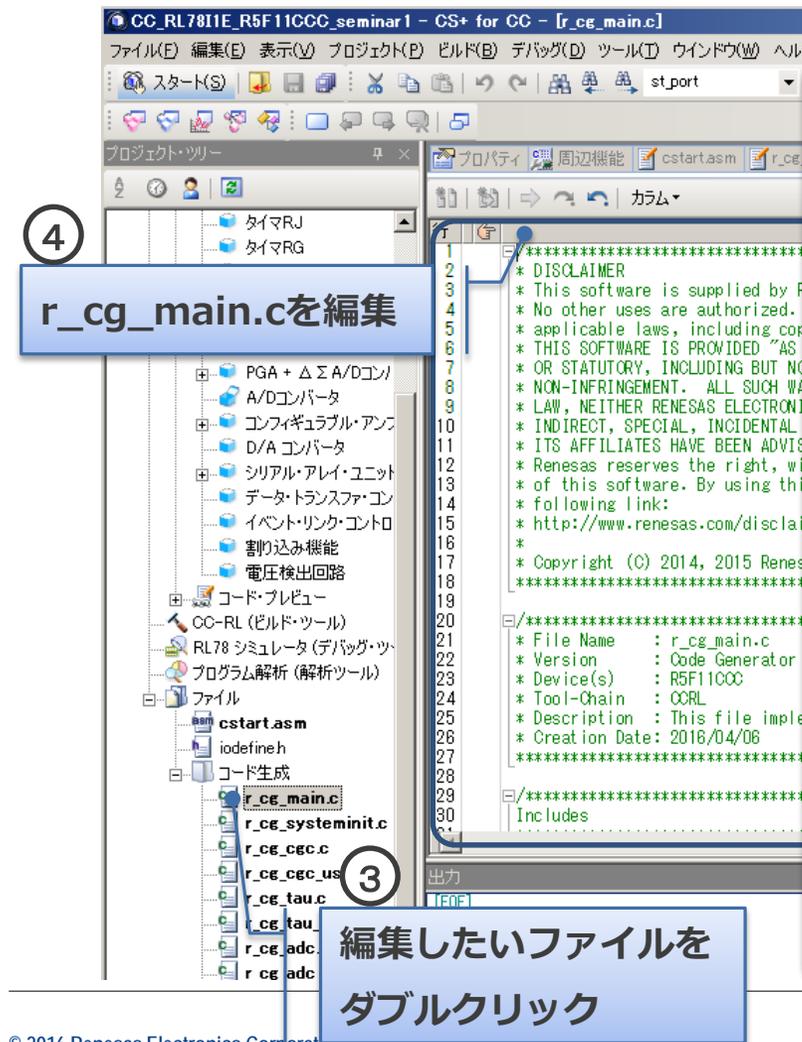
コード生成ツールが出力するAPIの詳細は、下記のユーザズマニュアルをご参照ください。

http://japan.renesas.com/products/tools/coding_tools/coding_assistance/cg_p/Documentation.jsp

CS+ コード生成ツール 統合開発環境 ユーザズマニュアル RL78 API リファレンス編 (R20UT3102JJ0102)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (ソースファイルの編集)



④
r_cg_main.cを編集

③
編集したいファイルを
ダブルクリック

💡 ワンポイント

ソースファイルを開くと/*Start ~*/、/*End~*/という記述があります。このコメント間にコードを書いてください。
コメントの外に書かれたコードは「コード生成」を実行すると記述が消去されてしまいます。

include の追加を書く場合

```
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
#include "r_cg_tau.h"
#include "r_cg_adc.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"
```

プリAGMA命令を書く場合

```
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

グローバル変数を書く場合

```
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
void R_MAIN_UserInit(void);
```

```
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/* Function Name: main
 * Description : This function implements main function.
 * Arguments : None
 * Return Value : None
 */
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */

/* Function Name: R_MAIN_UserInit
 * Description : This function adds user code before implementing
main function.
 * Arguments : None
 * Return Value : None
 */
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    EI();
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

mainの処理を記述

ユーザの初期化処理を記述

ユーザ関数を新規に追加するときに記述

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_cg_main.c)

```

46  /*****
47  Global variables and functions
48  *****/
49  /* Start user code for global. Do not edit comment generated here */
50  uint16_t gAdResult;           // AD変換結果
51  /* End user code. Do not edit comment generated here */
52
53  void R_MAIN_UserInit(void);
54  /*****
55  * Function Name: main
56  * Description : This function implements main function.
57  * Arguments   : None
58  * Return Value: None
59  *****/
60  void main(void)
61  {
62  R_MAIN_UserInit();
63  /* Start user code. Do not edit comment generated here */
64  while (1U)
65  {
66  ;
67  }
68  /* End user code. Do not edit comment generated here */
69  }

```

```

70  /*****
71  * Function Name: R_MAIN_UserInit
72  * Description : This function adds user code before implementing main function.
73  * Arguments   : None
74  * Return Value: None
75  *****/
76  void R_MAIN_UserInit(void)
77  {
78  /* Start user code. Do not edit comment generated here */
79  R_TAU0_Channel0_Start(); // タイマチャンネル0開始
80  EI();
81  /* End user code. Do not edit comment generated here */
82  }

```



ワンポイント

コード生成で使用する変数のデータ型は、r_cg_macrodriver.hに定義されています。

```

typedef signed char    int8_t;
typedef unsigned char  uint8_t;
typedef signed short   int16_t;
typedef unsigned short uint16_t;
typedef signed long    int32_t;
typedef unsigned long  uint32_t;
typedef unsigned short MD_STATUS;

```

コード生成とプログラム編集 (r_cg_tau_user.c)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

```

29  /******
30  Includes
31  *****/
32  #include "r_cg_macrodriver.h"
33  #include "r_cg_tau.h"
34  /* Start user code. Do not edit comment generated here */
35  #include "r_cg_adc.h"
36  /* End user code. Do not edit comment generated here */
37  #include "r_cg_userdefine.h"
38
39  /******
40  Pragma directive
41  *****/
42  #pragma interrupt r_tau0_channel0_interrupt(vect=INTTM00)
43  .
44  .
52  /******
53  * Function Name: r_tau0_channel0_interrupt
54  * Description  : This function INTTM00 interrupt service routine.
55  * Arguments   : None
56  * Return Value: None
57  *****/
58  __interrupt static void r_tau0_channel0_interrupt(void)
59  {
60  /* Start user code. Do not edit comment generated here */
61  R_ADC_Start();
62  /* End user code. Do not edit comment generated here */
63  }

```

A/DのAPIを使うための宣言

A/D変換の開始

コード生成とプログラム編集 (r_cg_ad_user.c)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

```

45  /******
46  Global variables and functions
47  *****/
48  /* Start user code for global. Do not edit comment generated here */
49  extern uint16_t gAdResult;
50  /* End user code. Do not edit comment generated here */
51
52  /******
53  * Function Name: r_adc_interrupt
54  * Description  : None
55  * Arguments   : None
56  * Return Value : None
57  *****/
58  __interrupt static void r_adc_interrupt(void)
59  {
60  /* Start user code. Do not edit comment generated here */
61  R_ADC_Stop();
62  R_ADC_Get_Result( &gAdResult );
63  /* End user code. Do not edit comment generated here */
64  }
65  /* Start user code for adding. Do not edit comment generated here */
66  /* End user code. Do not edit comment generated here */
67

```

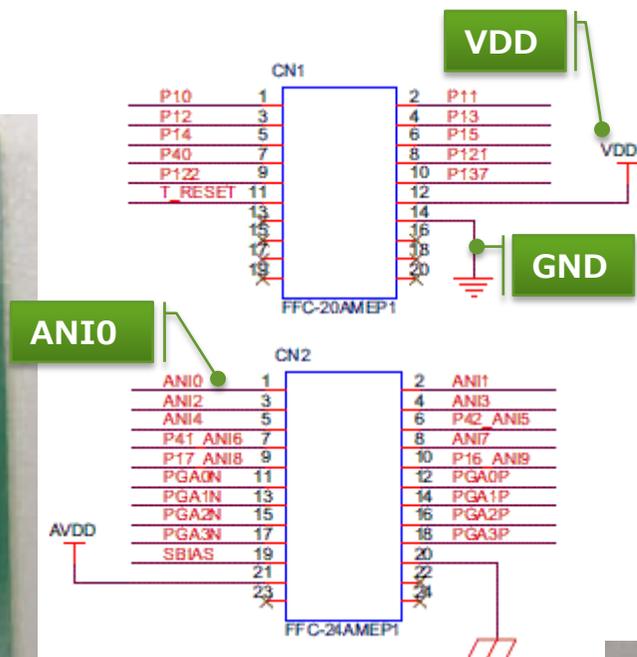
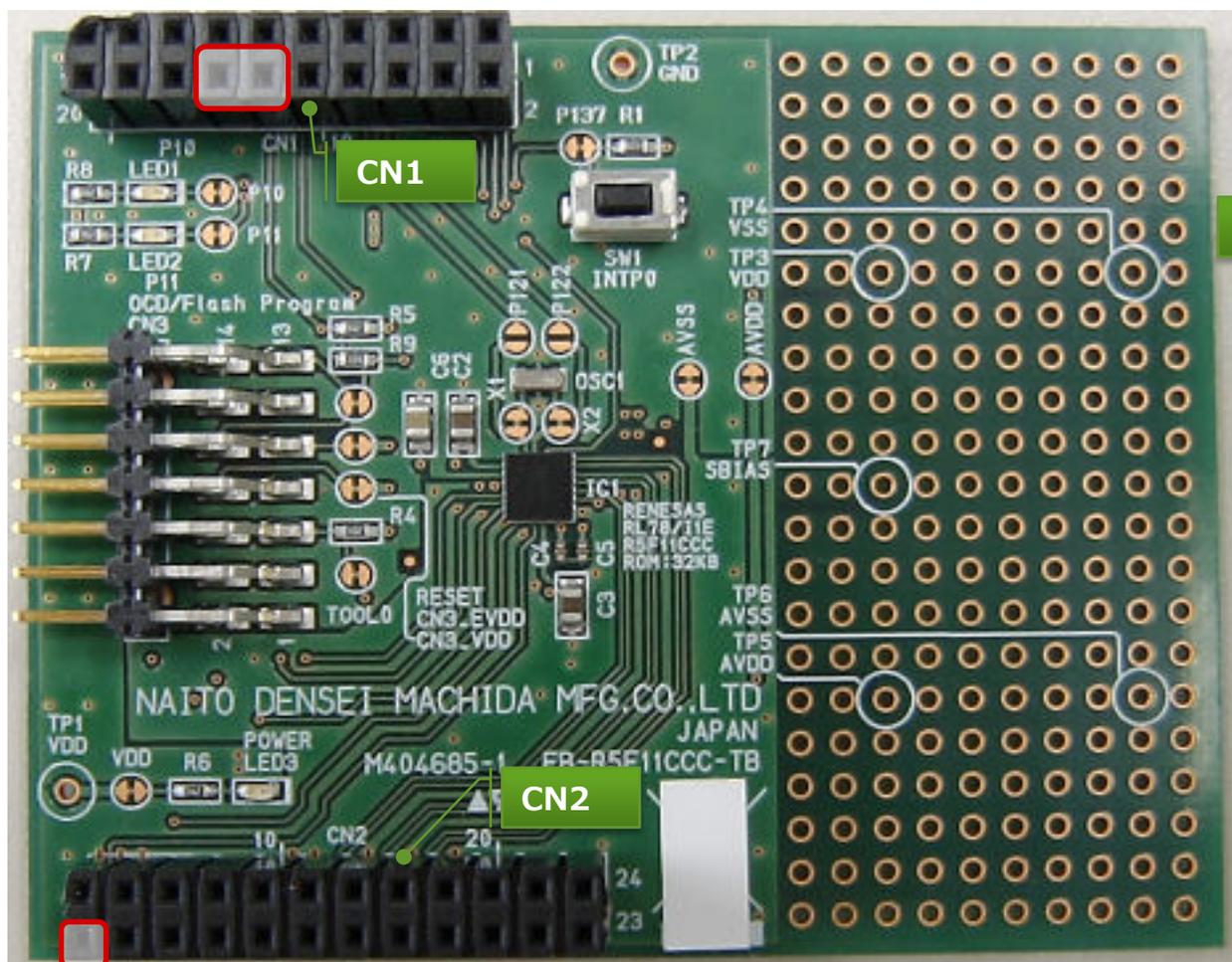
A/D変換が終了するごとに呼ばれる割り込み関数

A/D変換を停止

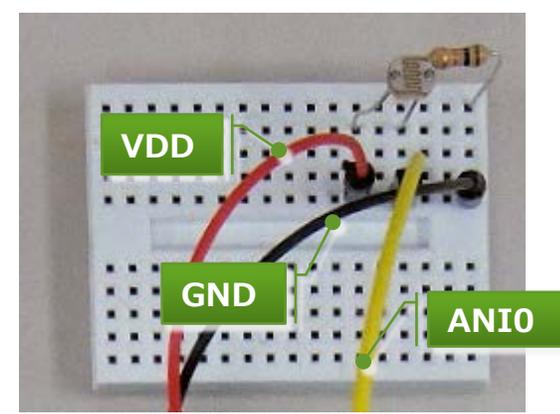
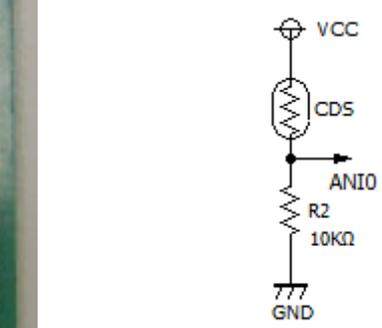
A/D変換結果を変数へ代入

ハードウェア設定

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



CN1 12pin VDD
 CN1 14pin GND
 CN2 1pin ANIO
 上記をボードに接続



- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

プログラムの実行とデバッグ (デバッグツールの設定)

The screenshot shows the IDE interface with the following elements:

- Project Tree (Left):** Shows the project structure for 'CC_RL78I1E_R5F11CCC_seminar1'. A right-click context menu is open over the 'RL78 シミュレータ(S)' folder, with 'RL78 E1(Serial)(L)' selected.
- Code Editor (Center):** Displays C code with pragmas for interrupt and global variables.
- Property Window (Top Right):** Shows the 'RL78 E1(Serial) のプロパティ' window. The 'ターゲットボードとの接続' section has 'エミュレータから電源供給をする(最大200 mA)' set to 'はい'.
- Property Window (Bottom Right):** Shows the 'RL78 E1(Serial) のプロパティ' window. The '実行中のメモリ・アクセス' section has '実行を一瞬停止してアクセスする' and '実行中に表示更新を行う' both set to 'はい', and '表示更新間隔[ms]' set to '100'.

① デバッグ・ツール上で右クリック、「使用するデバッグツール」→RL78 E1(Serial) を選択

② E1のプロパティを開き、「エミュレータから電源を供給する」→ “はい”

③ デバッグ・ツール設定タブを選択し、「実行を一瞬してアクセスする」→ “はい”
「表示更新間隔(ms)」→ “100”

プログラムの実行とデバッグ (プログラムのビルド&ダウンロード)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

④ [デバッグ]メニューから、「ビルド&デバッグ・ツールヘダウンロード」を選択

⑤ ダブルクリックで変数を選択し、右クリックでメニューを表示

⑥ メニューよりウォッチ1に登録を選択

The screenshot shows the IDE interface with the following elements:

- Menu:** The 'Build & Debug' menu is open, showing 'Build & Debug Tools Header Download (B) F6'.
- Watch Window:** A '進捗状況' (Progress) dialog box is displayed, indicating connection status for 'RL78 E1(Serial)'. It contains the text: 'RL78 E1(Serial) に、接続処理中です。' and 'エミュレータ・ファームウェアの更新が必要な場合、自動的に更新を行います。接続が完了するまではUSBおよび電源は切断しないでください。' with a 'キャンセル' (Cancel) button.
- Code Editor:** The 'r_cg_main.c' file is open, showing the 'main' function. A variable 'uint16_t AdResult;' is highlighted, and a context menu is open over it, with 'ウォッチ1に登録(R)' (Add to Watch 1) selected.
- Annotations:** Blue callouts with numbers 4, 5, and 6 point to the menu selection, the variable selection, and the watch registration respectively.

プログラムの実行とデバッグ (デバッグ画面の説明)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

```
49 /* Start user code for global. Do not edit comment generated here */
50
51 uint16_t gAdResult; // AD変換結果
52
53 /* End user code. Do not edit comment generated here */
54
55 void R_MAIN_UserInit(void);
56
57 /******
58 * Function Name: main
59 * Description : This function implements main function.
60 * Arguments : None
61 * Return Value : None
62 *****/
63 void main(void)
64 {
65     R_MAIN_UserInit();
66     /* Start user code. Do not edit comment generated here */
67     while (1U)
68     {
69     }
70 }
71 /* End user code. Do not edit comment generated here */
72
73 /******
74 * Function Name: R_MAIN_UserInit
75 * Description : This function adds user code before implement
76 * Arguments : None
77 *****/
```

カーソル位置が現在のPC値

プログラムがダウンロードされ
mainの先頭まで実行されている

プログラム実行中でも表示される

ワンポイント

E1のプロパティでダウンロード後にmain()シンボルまでプログラムが起動する設定になっています。リセット直後からの動作を確認した場合は「CPUリセット後に指定シンボルまで実行する」→「いいえ」にしてください。

RL78 E1(Serial)のプロパティ	
ダウンロード	
ダウンロードするファイル	[1]
ダウンロード後にCPUをリセットする	はい
ダウンロード・モードの選択	スピード優先
ダウンロード前にフラッシュROMを消去する	いいえ
イベント設定位置の自動変更方法	イベントを保留にする
予約領域の上書きをチェックする	はい
デバッグ情報	
CPUリセット後に指定シンボル位置まで実行する	はい
指定シンボル	_main
メモリ使用量の上限サイズ[Mバイト]	500

ダウンロードするファイル
ダウンロードするファイルを指定します。[]ボタンを押下するとダウンロード・ファイル ダイアログが開きます。ダイアログで、ダウンロードするファイルを指定してください。

接続用設定 / デバッグ・ツール設定 / **ダウンロード・ファイル設定** / フック処理設定

プログラムの実行とデバッグ (デバッグ画面の説明)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

逆アセンブルとソースを混合して表示します。

混合表示でアセンブラも表示

汎用レジスタも表示可能

ウォッチ1

ウォッチ式 値 型情報(N)

gAdResult 0 (0x0000) uint16_t

ローカル変数 ウォッチ1

CPUレジスタ

レジスタ名 値

汎用レジスタ

- AX(RP0) 0x0000
- BC(RP1) 0x0000
- DE(RP2) 0x0010
- HL(RP3) 0xfe20

制御レジスタ

- PC 0x00166
- PSW 0x46
- SP 0xfe1c
- ES 0x00
- CS 0x00

出力

[EOF]

プログラム実行中でも表示される

ワンポイント

プログラム実行中に変数を表示させるためには、E1のデバッグツール設定で「実行を一瞬停止してアクセスする」"はい"にします。この一瞬という時間はウォッチに登録する変数の数で変わります。変数が1つなら、停止する時間はおよそ3μ秒です。

RL78 E1 (Serial) のプロパティ

- メモリ
- メモリ・マッピング [10]
- メモリ書き込み時にベリファイを行う (はい)
- 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 (はい)
- 実行中に表示更新を行う (はい)
- 表示更新間隔[ms] 100
- ブレーク
- トレース
- 入力信号のマスク
- TARGET RESET信号をマスクする (いいえ)
- INTERNAL RESET信号をマスクする (いいえ)

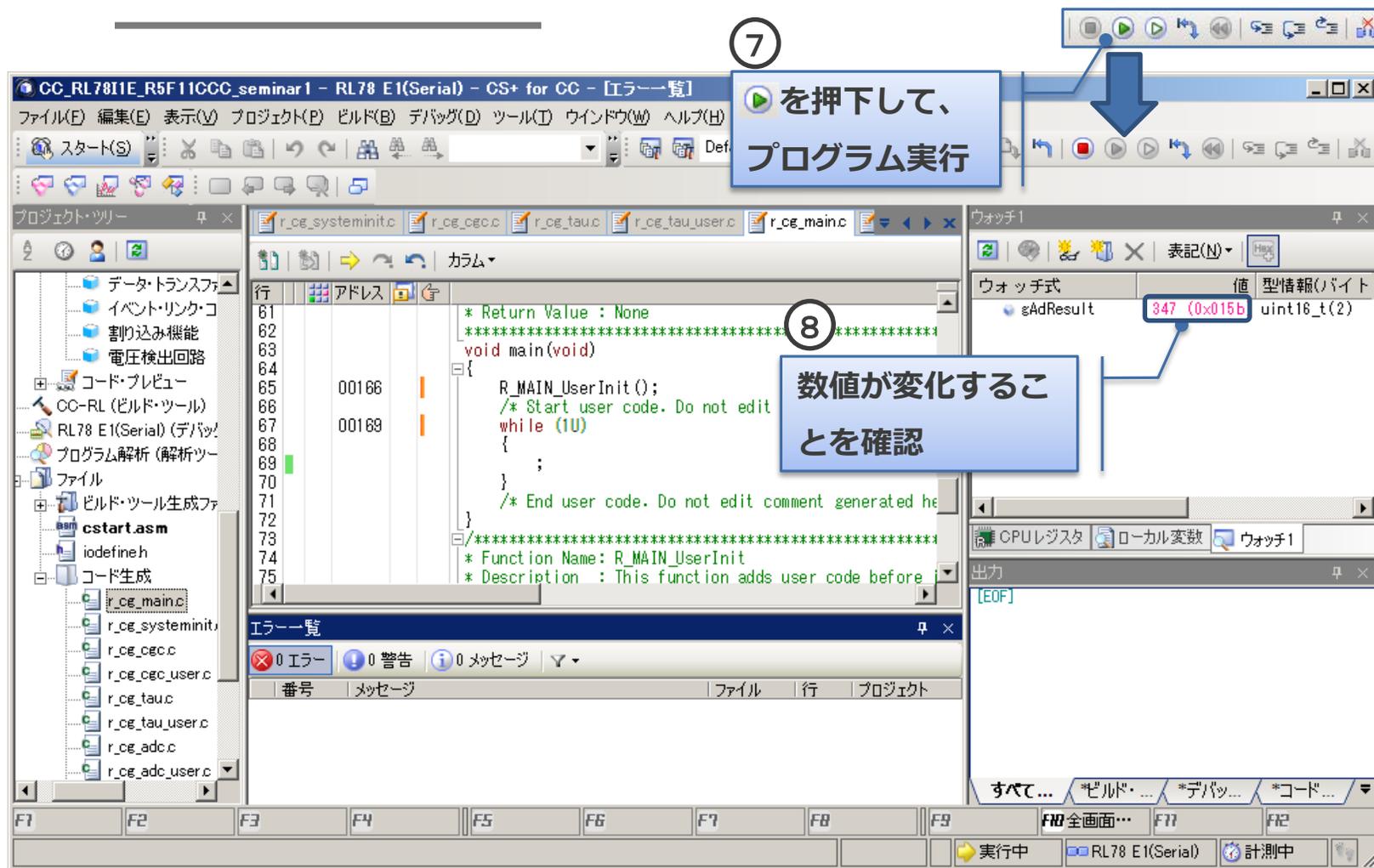
メモリ・マッピング

メモリ・マッピングの追加、削除、開始アドレスもしくは終了アドレスの変更は、[...]ボタンを選択し、メモリ・マッピングダイアログで行ってください。

プログラムの実行とデバッグ (ウォッチパネルで確認)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

⑦ を押下して、プログラム実行



```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit
    while (10)
    {
        ;
    }
    /* End user code. Do not edit comment generated here
}
/* Function Name: R_MAIN_UserInit
 * Description : This function adds user code before
```

⑧ 数値が変化することを確認

ウォッチ1

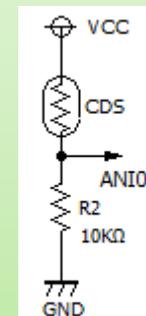
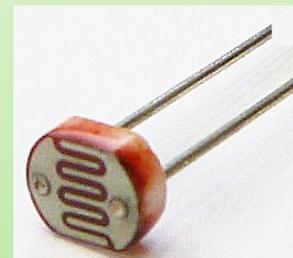
ウォッチ式	値	型情報(バイト)
gAdResult	347 (0x015b)	uint16_t(2)

エラー一覧

番号	メッセージ	ファイル	行	プロジェクト
0	エラー			
0	警告			
0	メッセージ			

💡 ワンポイント

CDSは暗いと抵抗値が高くなり、明るいとき抵抗値が低くなります。回路は抵抗分圧しています。そのため、CDSに光を当てると抵抗値が下がりANIOの電圧が上がります。逆に手をかざして暗くするとANIOの電圧が下がります。



プログラムの実行とデバッグ (解析グラフ表示)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

The screenshot shows the Renesas IDE interface with several callouts and annotations:

- 9** ウォッチ登録と同じ要領で解析グラフに登録 (Register the analysis graph in the same manner as the watch register).
- 10** プログラム解析を開く (Open program analysis).
- 11** 値の推移を開く (Open value change).
- 12** 自動調整を“行わない”、1グリッドあたりの時間を“500ms”にする (Set automatic adjustment to “Off”, and set the time per grid to “500ms”).
- 13** 1グリッドあたりの値を“50”、オフセット1を“-700”にする (Set the value per grid to “50”, and set offset 1 to “-700”).

The analysis graph settings window is visible on the right, showing:

- 解析方式: リアルタイム・サンプリング方式 (Real-time sampling method)
- 自動調整: 行わない (Off)
- 1グリッドあたりの時間[Time/Div]: 500ms
- グラフの種類: 折れ線グラフ (Line graph)
- チャンネル 1: 変数名/アドレス 1, 型/サイズ 1, 1グリッドあたりの値[Val/Div] 1, オフセット 1, 色 1
- チャンネル 2: 変数名/アドレス 2, 型/サイズ 2, オフセット 自動

The variable `gAdResult` is shown with a value of 50 and an offset of -700.

プログラムの実行とデバッグ (解析グラフ表示)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

14
を押下して、
プログラム実行

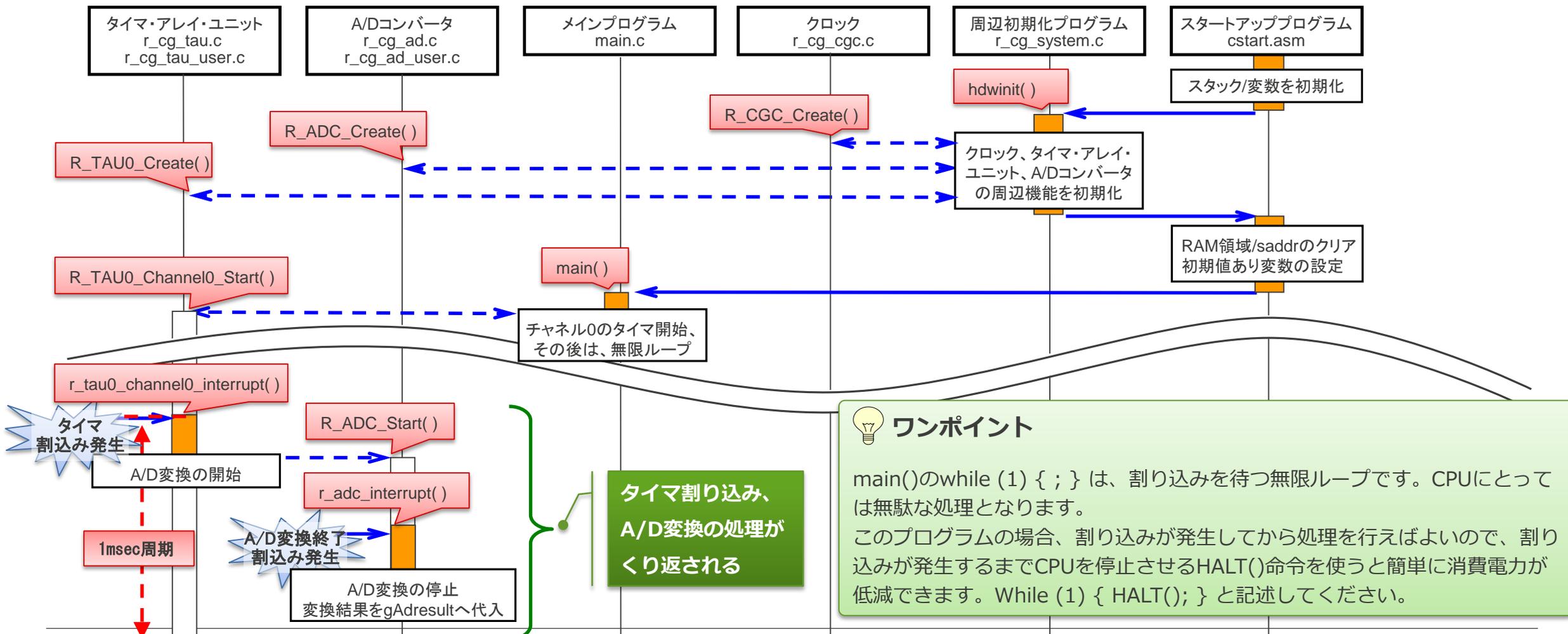
17
実行中も値が更新
される

💡 **ワンポイント**
表示の更新は100msが最小値です。そのため、オシロスコープのようにリアルタイムに波形を取得することはできません。

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

CPUを占有する処理

プログラムの解説 (リセットからmain()の処理について)



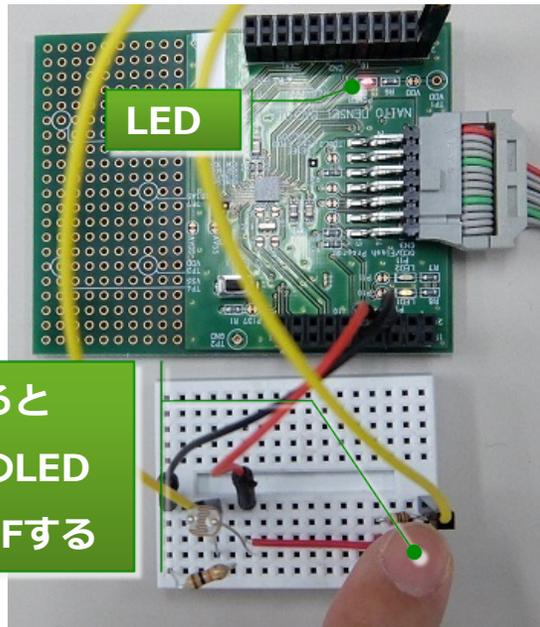
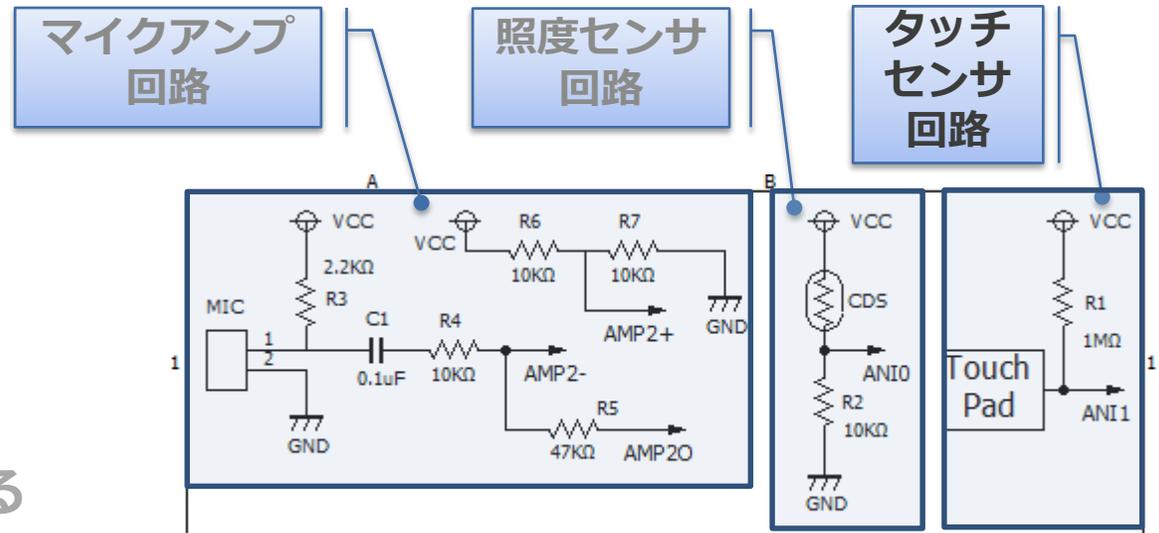
💡 ワンポイント

main()のwhile (1) { ; } は、割り込みを待つ無限ループです。CPUにとっては無駄な処理となります。このプログラムの場合、割り込みが発生してから処理を行えばよいので、割り込みが発生するまでCPUを停止させるHALT()命令を使うと簡単に消費電力が低減できます。While (1) { HALT(); } と記述してください。

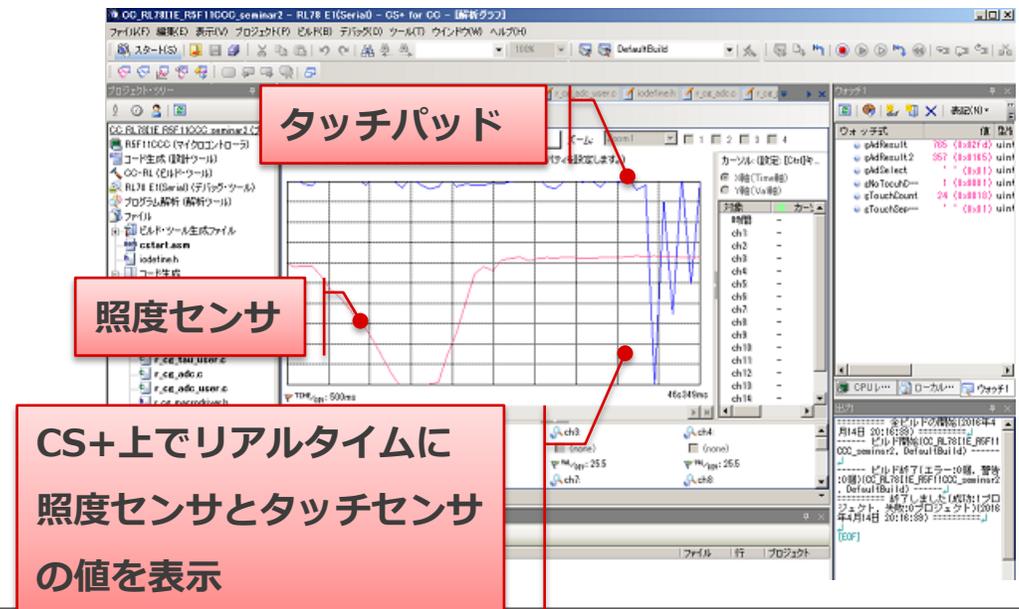
タイマ割り込み、
A/D変換の処理が
くり返される

演習B

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



タッチすると
ボード上のLED
がON/OFFする



処理のポイント

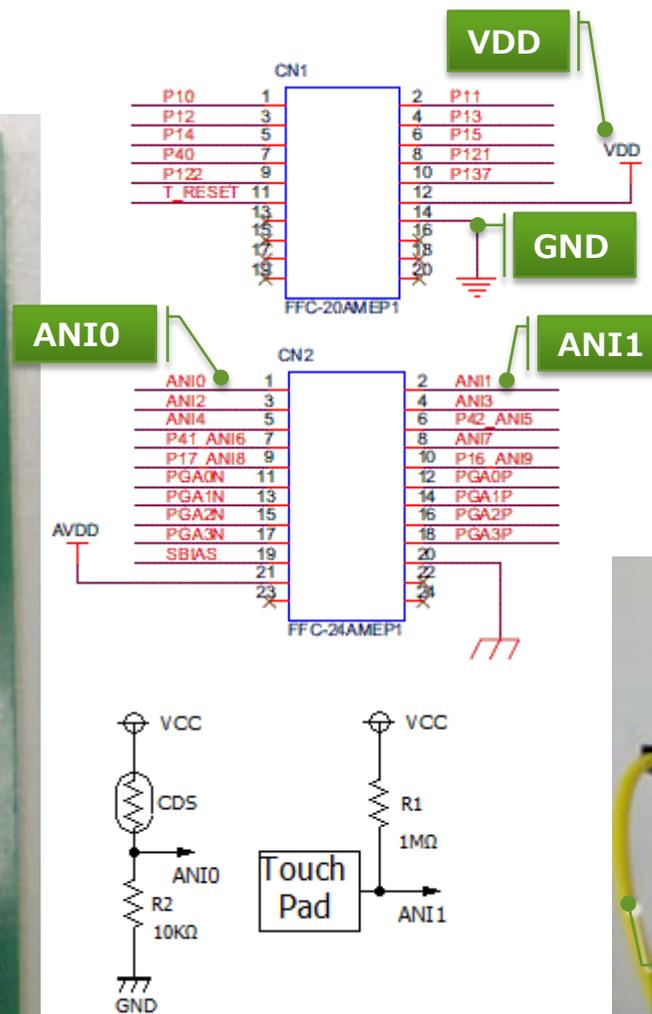
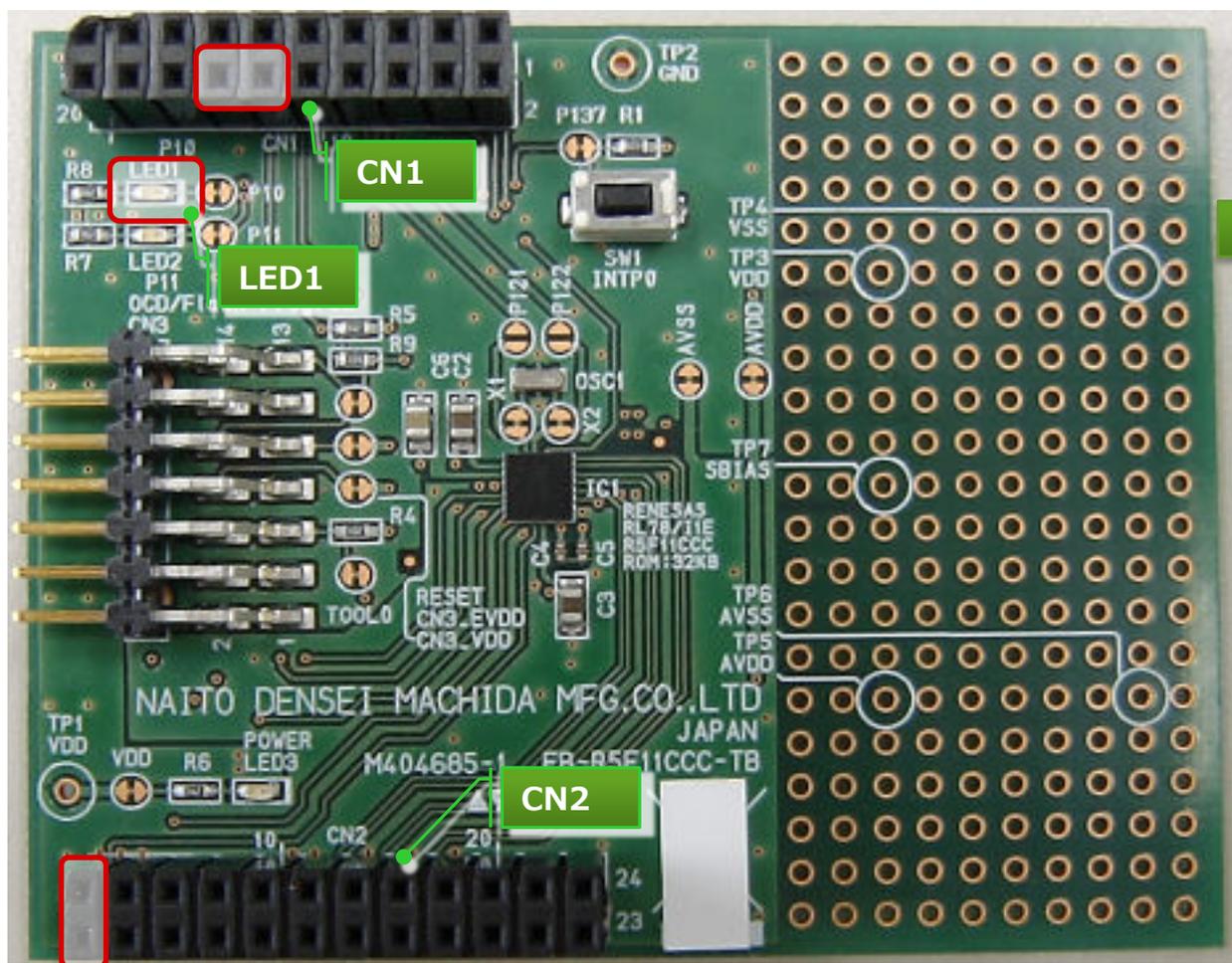
接続は下記の通り
 ANI0:照度センサ
 ANI1:タッチパッド
 Port10: LED

タッチパッドに触るとボード上のLED1(P10)がON/OFFする

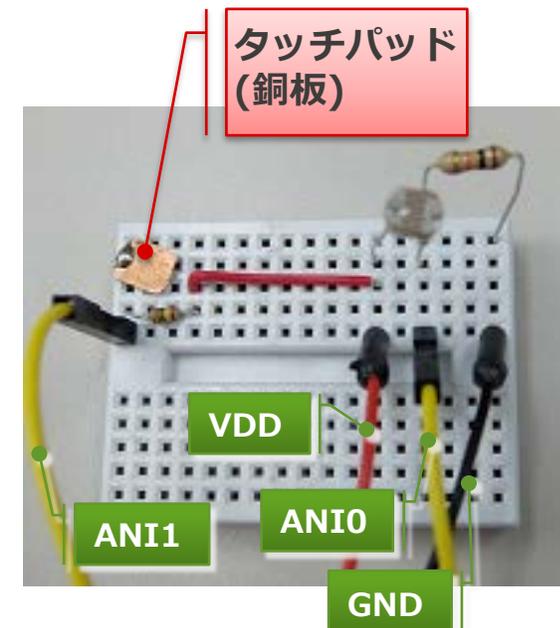
1msecごとにA/D変換を行い、ANI0とANI1を1msecごとに切り替える

ハードウェア設定

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



CN1 12pin VDD
 CN1 14pin GND
 CN2 1pin ANI0
 CN2 2pin ANI1
 ANI1へ接続する回路を追加



周辺機能設定 (A/Dコンバータ)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

① A/Dコンバータ
を選択

② 入力端子に
ANI1を追加

下へスクロール

他の設定は変更しない

周辺機能設定

- A/Dコンバータ動作設定
 - 使用しない
 - 使用する
- コンパレータ動作設定
 - 停止
 - 許可
- 分解能設定
 - 10 ビット
 - 8 ビット
- VREF(+)設定
 - AVDD
 - 内部基準電圧
- VREF(-)設定
 - AVSS
- トリガ・モード設定
 - ソフトウェア・トリガ・モード
 - ハードウェア・トリガ・ノーウェイト・モード
 - ハードウェア・トリガ・ウェイト・モード
- 動作モード設定
 - 連続セレクト・モード
 - 連続スキャン・モード
 - ワンショット・セレクト・モード
 - ワンショット・スキャン・モード

アナログ入力端子設定(高精度チャンネル)

<input checked="" type="checkbox"/> ANI0	<input checked="" type="checkbox"/> ANI1	<input type="checkbox"/> ANI2	<input type="checkbox"/> ANI3
<input type="checkbox"/> ANI4	<input type="checkbox"/> ANI5	<input type="checkbox"/> ANI6	<input type="checkbox"/> ANI7
<input type="checkbox"/> ANI8	<input type="checkbox"/> ANI9		

変換開始チャンネル設定: ANI0

変換時間設定

基準電圧	2.7 ≤ AVDD ≤ 5.5 (V)
変換時間モード	標準1
変換時間	1216/fCLK 38 (μs)

変換結果上限/下限値設定

- ADLL ≤ ADCRH ≤ ADULで割り込み要求信号(INTAD)を発生
- ADUL < ADCRHまたはADLL > ADCRHで割り込み要求信号(INTAD)を発生

上限値(ADUL)	255
下限値(ADLL)	0

割り込み設定

- A/Dの割り込み許可(INTAD)

優先順位: レベル3(低優先順位)

周辺機能設定 (ポート)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

8 Port1を選択

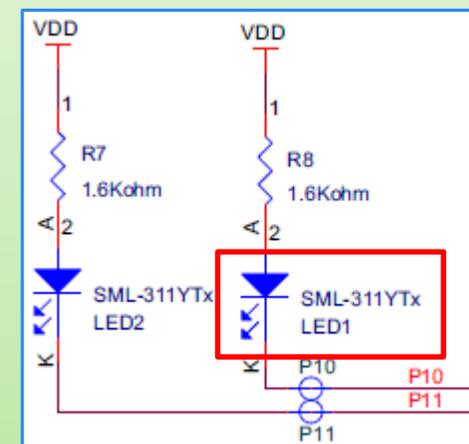
9 P10を出力に設定

10 1を出力にチェック

11 周辺機能の設定が完了後
“コードを生成する”押下
生成したソースを次ページ
より編集する

💡 ワンポイント

RL78/I1EボードはP10, P11にLEDが接続されています。LEDを点灯させるには、P10, P11をLowにします。Lowで動作することをアクティブ・ローと呼びます。1ポートにつき電流は、ローレベル出力で最大40mAです。抵抗を介してLEDが接続されているので、LEDが点灯しても5mA程度しか流れません。LEDを接続するとき、ドライバICを必ず使わなくてもよいのです。



- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_cg_main.c)

```
47 /*****
48 Global variables and functions
49 *****/
50 /* Start user code for global. Do not edit comment generated here */
51
52 uint16_t gAdResult; // AD変換結果
53 uint16_t gAdResult2; // AD変換結果2
54 uint8_t gAdSelect; // 変換するA/Dチャネル 0 = ANI0, 1= ANI1
55 uint16_t gTouchCount; // タッチパネルに触れていると判断した数
56 uint16_t gNoTocuhCount; // タッチパネルに触ってない回数
57 uint8_t gTouchSeparate; // タッチパネルに触っている状態 0=OFF, 1=ON
58
59
60 /* End user code. Do not edit comment generated here */
61
62 void R_MAIN_UserInit(void);
63 /*****
64 * Function Name: main
65 * Description : This function implements main function.
66 * Arguments : None
67 * Return Value : None
68 *****/
69 void main(void)
70 {
71 R_MAIN_UserInit();
72 /* Start user code. Do not edit comment generated here */
73 while (1U)
74 {
75 ;
76 }
77 /* End user code. Do not edit comment generated here */
78 }
```

```
79 /*****
80 * Function Name: R_MAIN_UserInit
81 * Description : This function adds user code before implementing main function.
82 * Arguments : None
83 * Return Value : None
84 *****/
85 void R_MAIN_UserInit(void)
86 {
87 /* Start user code. Do not edit comment generated here */
88
89 R_TAU0_Channel0_Start(); // タイマチャネル0 開始
90 gAdSelect = 0;
91 gTouchCount = 0;
92 gNoTocuhCount =0;
93 EI();
94
95 /* End user code. Do not edit comment generated here */
96 }
```

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_tau_user.c)

```
29 /*****  
30 Includes  
31 *****/  
32 #include "r_cg_macrodriver.h"  
33 #include "r_cg_tau.h"  
34 /* Start user code for include. Do not edit comment generated here */  
35 #include "r_cg_adc.h"  
36 /* End user code. Do not edit comment generated here */  
37 #include "r_cg_userdefine.h"  
38  
39 /*****  
40 Pragma directive  
41 *****/  
42 #pragma interrupt r_tau0_channel0_interrupt(vect=INTTM00)  
43 /* Start user code for pragma. Do not edit comment generated here */  
44 extern uint8_t gAdSelect;  
45 /* End user code. Do not edit comment generated here */  
46  
47 /*****  
48 Global variables and functions  
49 *****/  
50 /* Start user code for global. Do not edit comment generated here */  
51 /* End user code. Do not edit comment generated here */  
52
```

```
53 /*****  
54 * Function Name: r_tau0_channel0_interrupt  
55 * Description : This function INTTM00 interrupt service routine.  
56 * Arguments : None  
57 * Return Value : None  
58 *****/  
59 static void __near r_tau0_channel0_interrupt(void)  
60 {  
61 /* Start user code. Do not edit comment generated here */  
62 ad_channel_t ch;  
63  
64 switch ( gAdSelect )  
65 {  
66 case 0:  
67 ch = ADCHANNEL0;  
68 break;  
69  
70 case 1:  
71 ch = ADCHANNEL1;  
72 break;  
73  
74 default:  
75 ch = ADCHANNEL0;  
76 break;  
77 }  
78 R_ADC_Set_ADChannel( ch );  
79 R_ADC_Start();  
80  
81 /* End user code. Do not edit comment generated here */  
82 }  
83
```

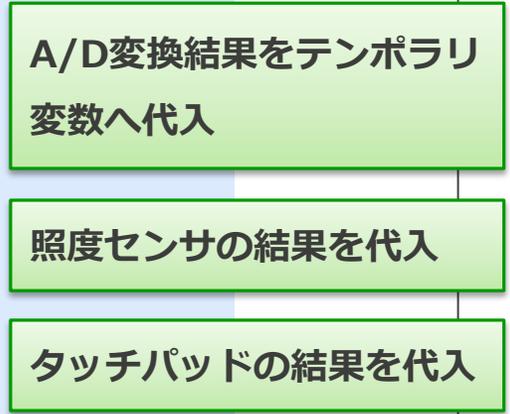
A/D変換を開始するときに
チャンネル番号を切り替える

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_adc_user.c)

```
20 /*****  
21 * File Name   : r_cg_adc_user.c  
22 * Version    : Code Generator for RL78/I1E V1.02.00.06 [12 Aug 2015]  
23 * Device(s)  : R5F11CCC  
24 * Tool-Chain : CCRL  
25 * Description : This file implements device driver for ADC module.  
26 * Creation Date: 2016/04/08  
27 *****/  
.  
.  
.  
45 /*****  
46 Global variables and functions  
47 *****/  
48 /* Start user code for global. Do not edit comment generated here */  
49 extern uint16_t gAdResult;  
50 extern uint16_t gAdResult2;  
51 extern uint8_t gAdSelect;  
52 extern uint16_t gTouchCount;  
53 extern uint16_t gNoTocuhCount;  
54 extern uint8_t gTouchSeparate;  
55 /* End user code. Do not edit comment generated here */  
56  
57 /*****  
58 * Function Name: r_adc_interrupt  
59 * Description  : None  
60 * Arguments   : None  
61 * Return Value: None  
62 *****/  
63 static void __near r_adc_interrupt(void)  
64 {
```

```
65 /* Start user code. Do not edit comment generated here */  
66 uint16_t result;  
67  
68 R_ADC_Stop();  
69 R_ADC_Get_Result( &result );  
70 switch ( gAdSelect )  
71 {  
72     case 0:  
73         gAdResult = result;  
74         break;  
75  
76     case 1:  
77         gAdResult2 = result;  
78  
79         if ( result > D_TOUCH_AD )  
80         {  
81             gTouchCount++;  
82             gNoTocuhCount = 0;  
83         }  
84         else if ( result > D_NOTOUCH_ADMIN )  
85         {  
86             gNoTocuhCount++;  
87         }  
88  
89         if ( gNoTocuhCount > D_NOTOUCH_COUNT )  
90         {  
91             gTouchCount = 0;  
92             gTouchSeparate = 0;  
93         }  
94     }
```



次ページへ

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_adc_user.c)

前ページから

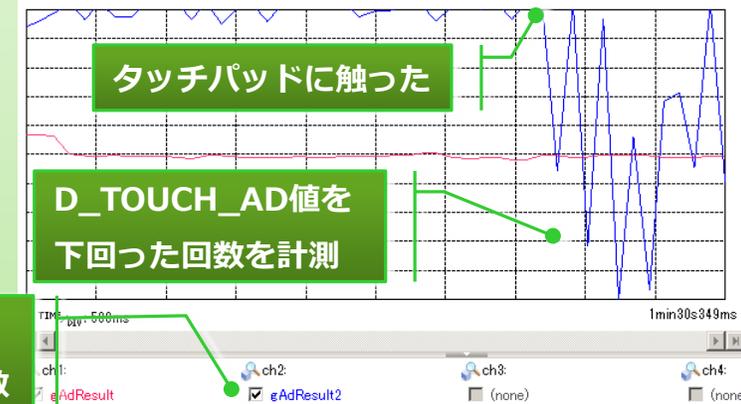
```
94
95     if ( (gTouchCount > D_TOUCH_COUNT) &&( gTouchSeparate == 0) )
96     {
97         // LED点灯と消灯を繰り返す
98         P1_bit.no0 ^= 1;
99         gTouchSeparate = 1;
100    }
101
102    break;
103
104    default:
105        gAdResult = result;
106        break;
107    }
108
109    gAdSelect++;
110    if ( gAdSelect >= D_MAXAD_CHANNEL )
111    {
112        gAdSelect = 0;
113    }
114
115    /* End user code. Do not edit comment
116 }
117
118
```

A/D変換の完了の最後の
処理で次に変換するに
チャンネル番号を指定する

ワンポイント

タッチパッドの処理が複雑になっていますが、理由があります。下図は、タッチパッドを触った時の解析グラフの波形です。触った瞬間に電圧降下し、充電と放電を繰り返します。人間はキャパシタ(コンデンサ)の役割と同等なので、このような波形になります。

触ったと判断するのは、A/D値がD_TOUCH_AD(100)を下回った回数をカウントする変数 **gTouchCount** が一定の大きさを超えた場合です。また、タッチパッドに触ったままの状態では、何もしないようにしています。



- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_cg_userdefine.h)

```
20  /*****
21  * File Name   : r_cg_userdefine.h
22  * Version    : Code Generator for RL78/I1E V1.02.00.06 [12 Aug 2015]
23  * Device(s)  : R5F11CCC
24  * Tool-Chain : CCRL
25  * Description : This file includes user definition.
26  * Creation Date: 2016/04/08
27  *****/
28  #ifndef _USER_DEF_H
29  #define _USER_DEF_H
30
31  /*****
32  User definitions
33  *****/
34
35  /* Start user code for function. Do not edit comment generated here */
36
37  #define D_MAX_AD_CHANNEL      2           // A/D変換するチャンネル数
38
39  #define D_TOUCH_AD           100         // タッチパッドに触ったと判断する値
40  #define D_NOTOUCH_ADMIN     900         // タッチパッド非接触時を判断するA/D最小値
41  #define D_TOUCH_COUNT       10         // タッチパッドを触ったと判断する回数
42  #define D_NOTOUCH_COUNT     100        // タッチパッドを触っていないと判断する回数
43
44  /* End user code. Do not edit comment generated here */
45  #endif
46
```

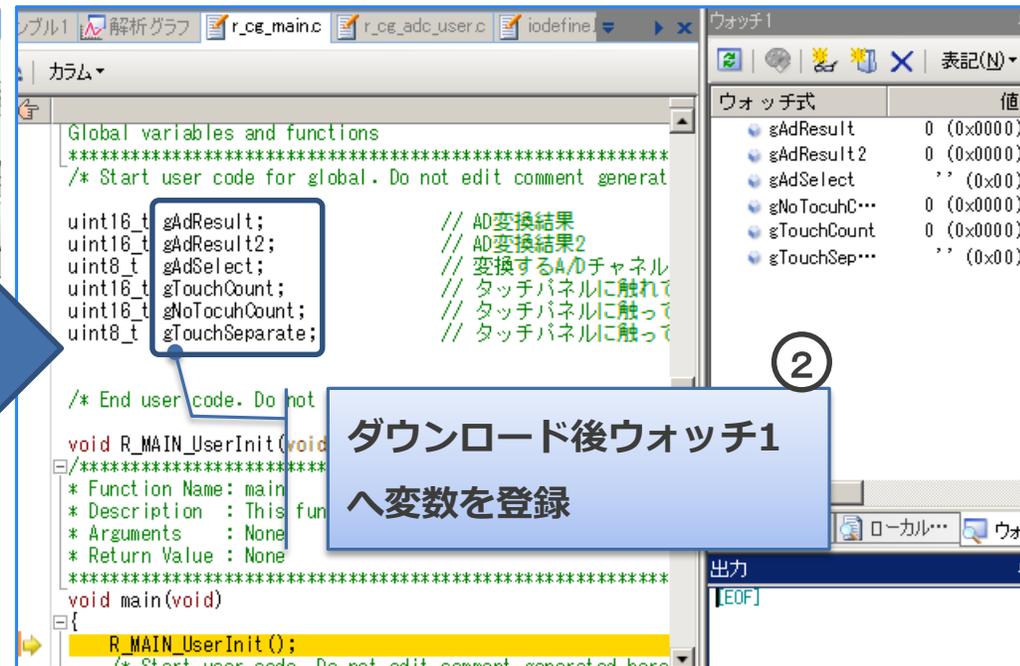
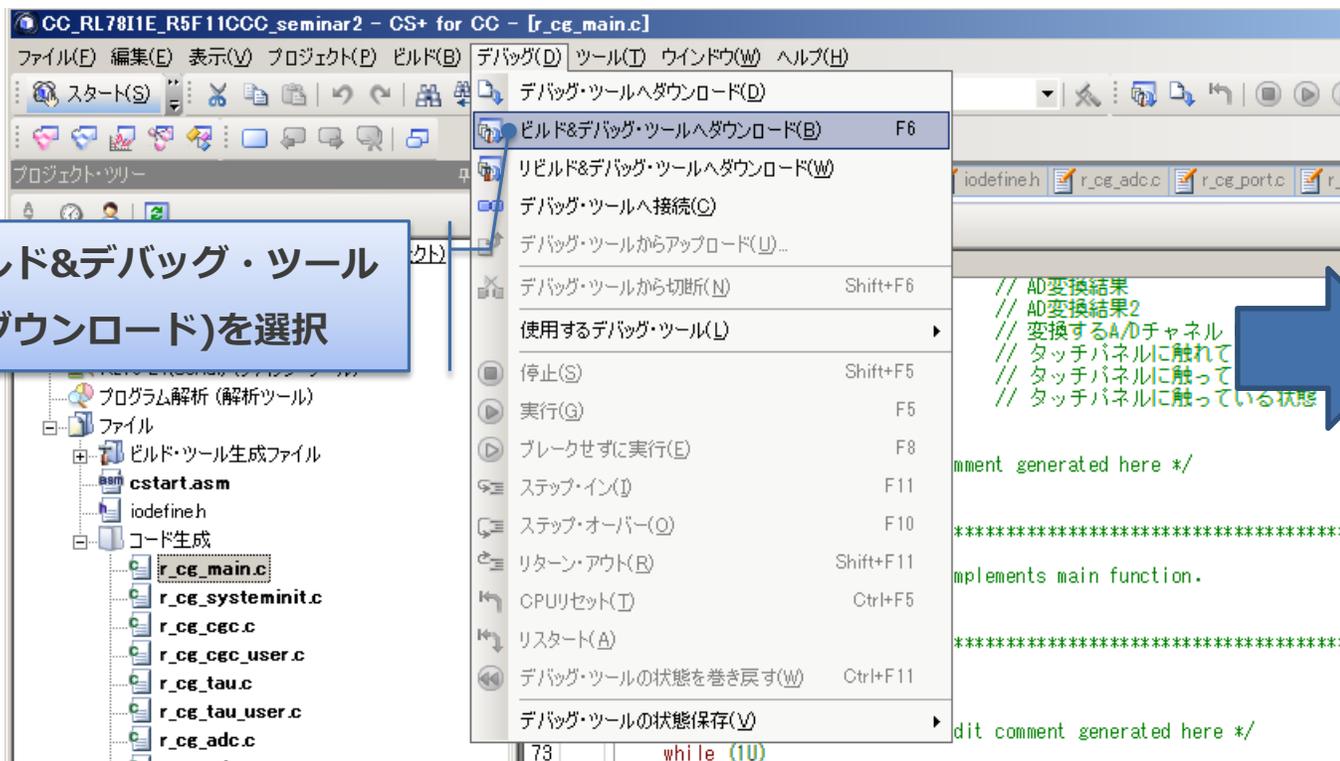
ワンポイント

r_cg_userdefine.hは、コード生成されるファイルには必ず includeされます。プログラムを記述するときにマジックナンバーを使わないためにユーザ専用で定義するファイルを用意しました。define だけでなく、マクロ命令、enum、typedefなどを、ここへ書いてください。

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

プログラムの実行とデバッグ (プログラムのビルド&ダウンロード)

①
ビルド&デバッグ・ツール
ヘダダウンロードを選択



プログラムの実行とデバッグ (解析グラフに登録)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

③ ウォッチ登録と同じ要領で解析グラフに登録

④ プログラム解析のプロパティ → "値の推移"タブを開いてチャンネル2の設定を行う

```

uint16_t gAdResult;
uint16_t gAdResult2;
uint8_t gAdSelect;
uint16_t gTouchCount;
uint16_t gNoTouchCount;
uint8_t gTouchSeparation;

void R_MAIN_UserInit(void)
{
    /* Start user code. */
    while (1U)
    {
        R_MAIN_UserInit();
    }
}
    
```

プログラム解析のプロパティ

カーソルBの色	PaleTurquoise
ズーム枠1の色	64, 255, 10, 79
ズーム枠2の色	64, 91, 228, 22
ズーム枠3の色	64, 5, 109, 239
ズーム枠4の色	64, 255, 84, 28

色 1

チャンネル 2

変数名/アドレス 2

型/サイズ 2

1グリッドあたりの値[Val/Div] 2

オフセット 2

色 2

チャンネル 3

解析方式

値の推移グラフのデータ取得方法を選択します。

リアルタイム・サンプリング方式ではプログラム停止

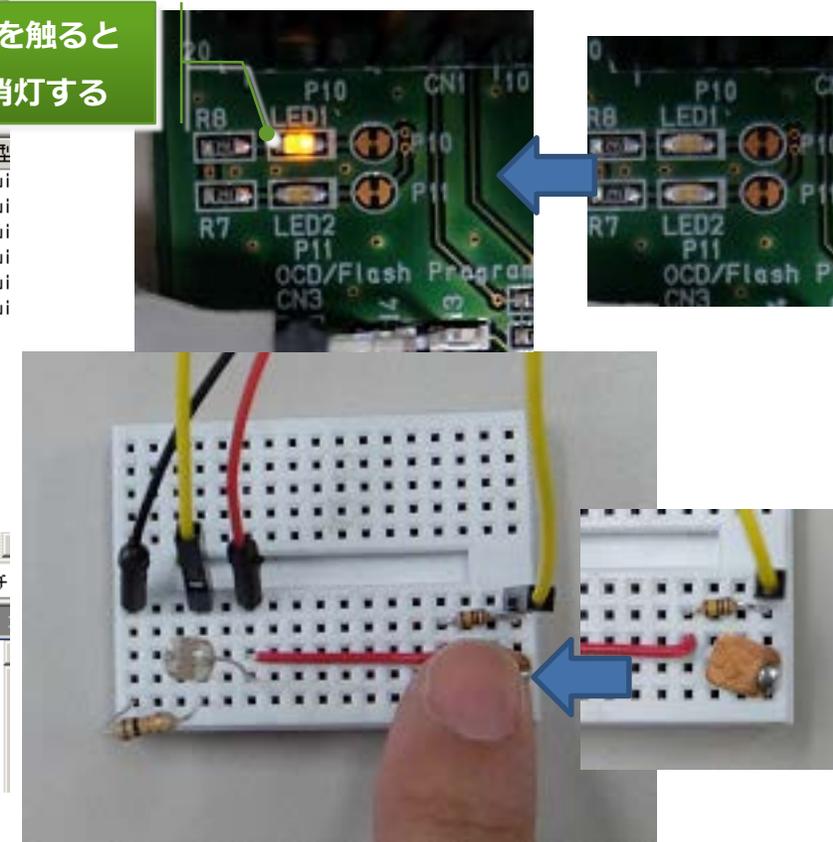
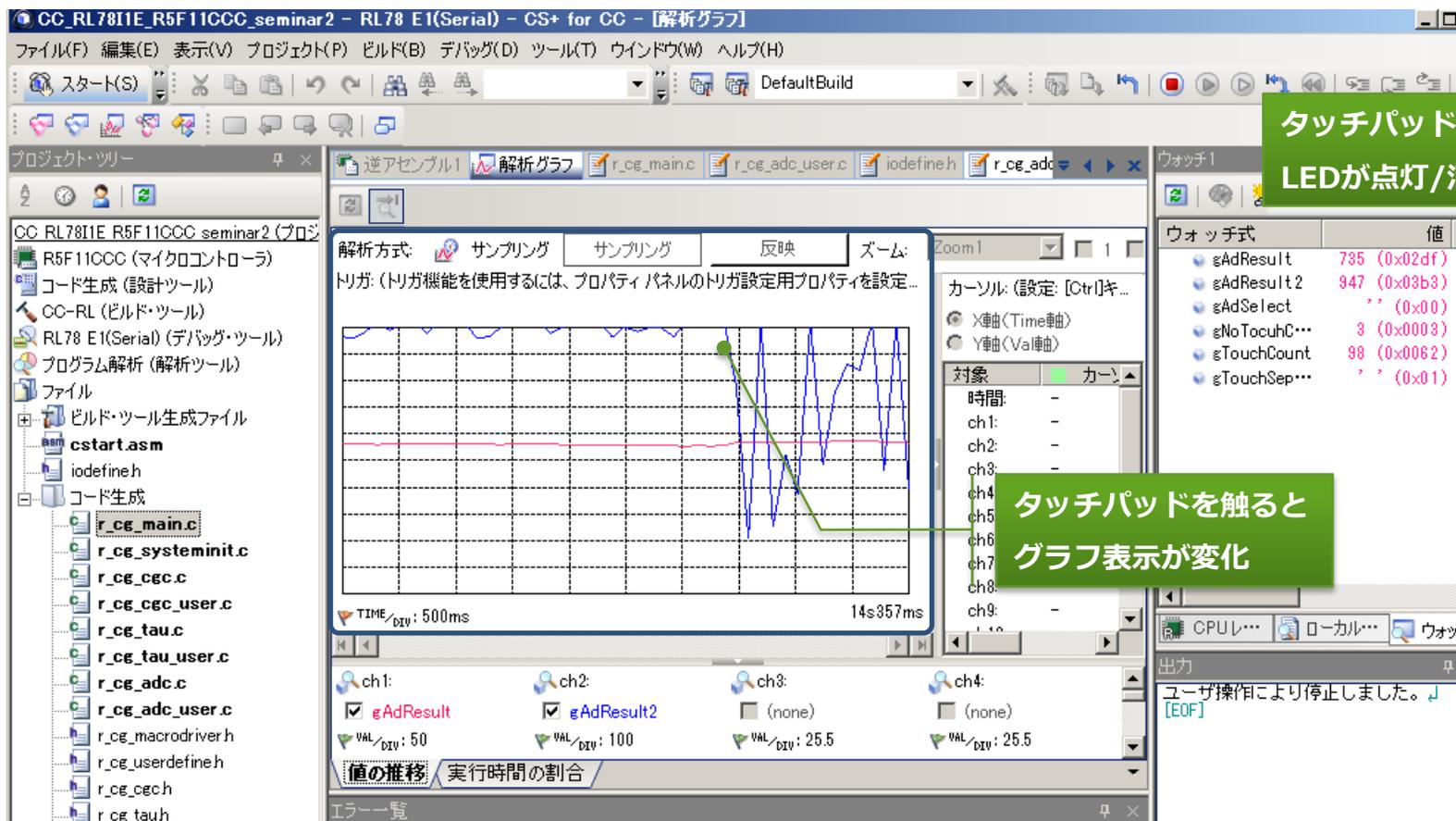
設定 値の推移

ワンポイント

グラフの表示が見つらいときは、オフセット値や1グリッドあたりの値を大きくすると表示を調整できます。

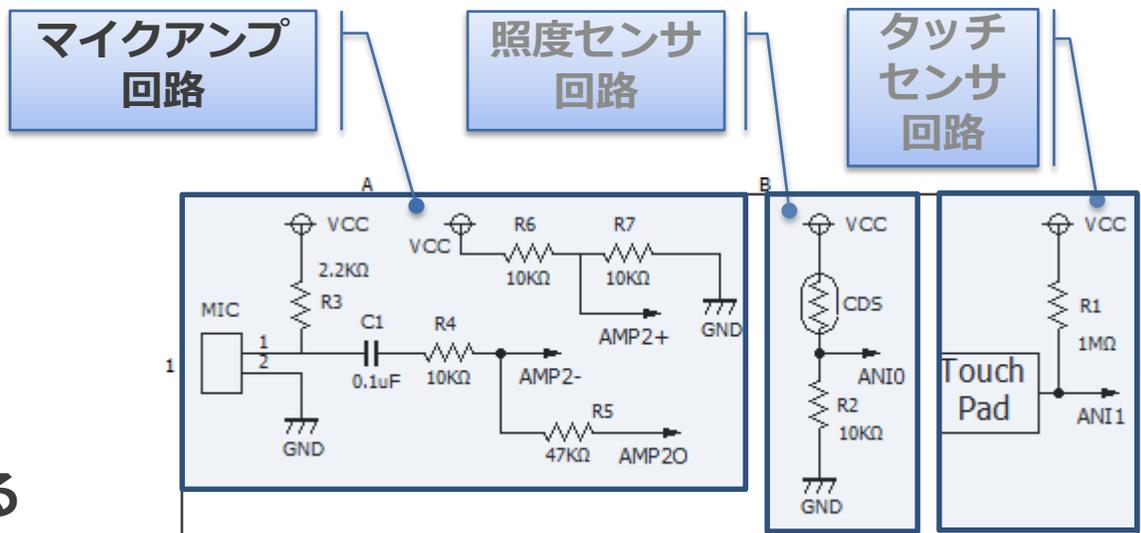
プログラムの実行とデバッグ (解析グラフ表示)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

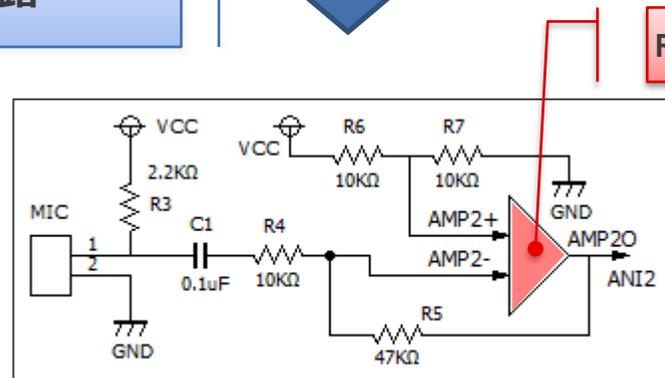


演習C

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



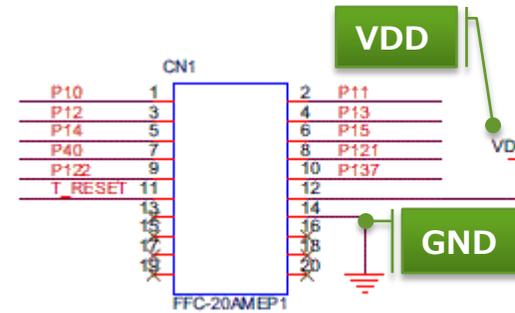
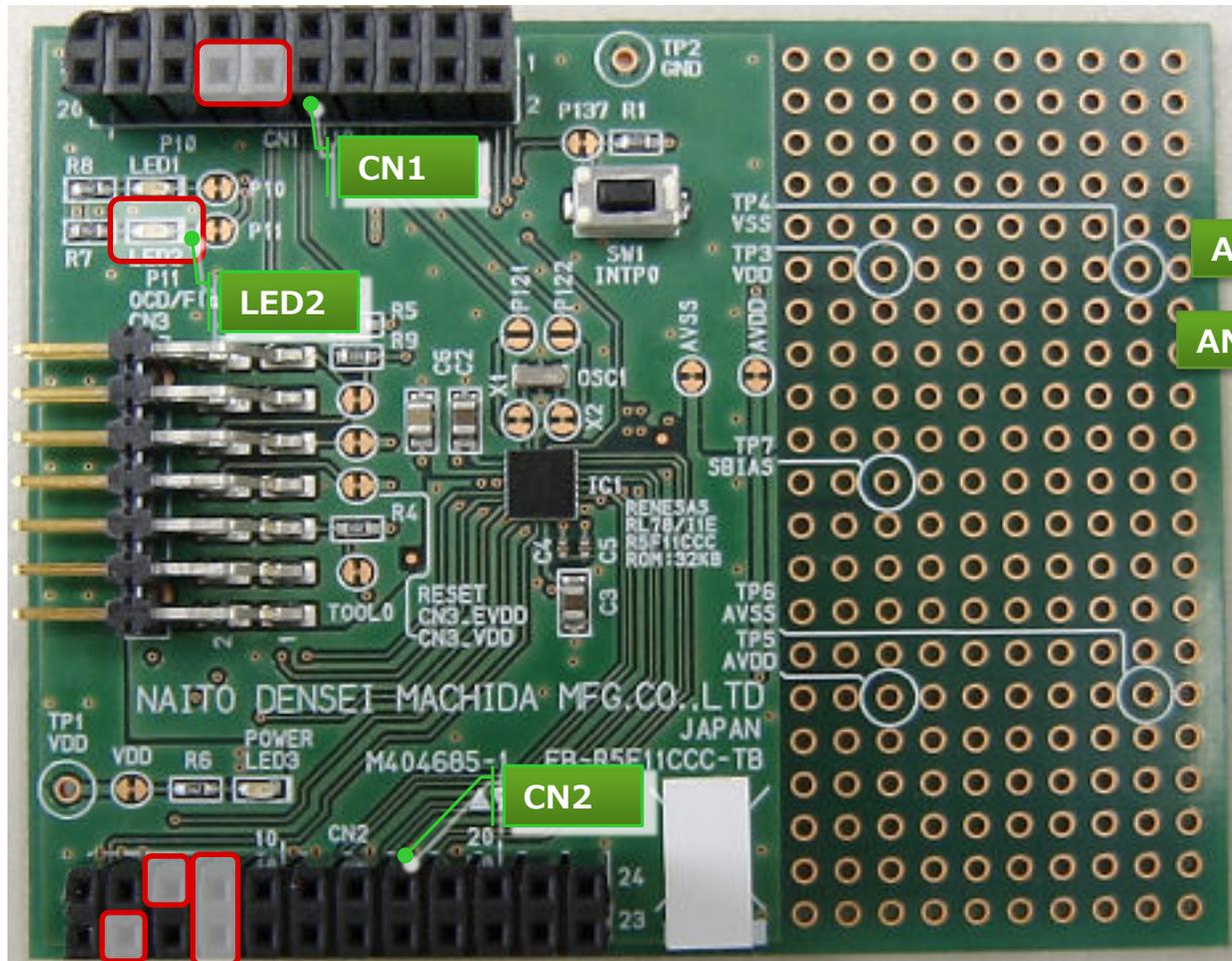
RL78/I1Eを含めた
等価回路



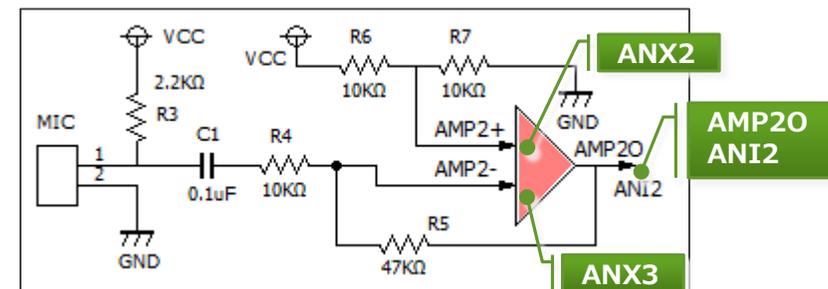
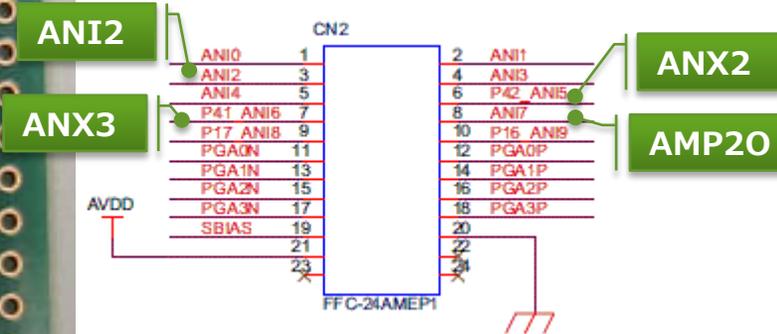
RL78/I1Eコンフィギュラブル・アンプ

ハードウェア設定

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



CN1 12pin	VDD
CN1 14pin	GND
CN2 3pin	ANI2
CN2 6pin	ANX2
CN2 7pin	ANX3
CN2 8pin	AMP20



周辺機能設定 (A/Dコンバータ)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

① A/Dコンバータを選択

② 入力端子に ANI2を追加

下へスクロール

他の設定は変更しない

プロジェクト・ツリー

CC RL78IIE R5F11CCC seminar1

R5F11CCC (マイクロコントローラ)

コード生成 (設計ツール)

端子図

端子配置表

端子配置図

周辺機能

共通/クロック発生回路

ポート機能

タイマ・アレイ・ユニット

タイマRJ

タイマRG

リアルタイム・クロック

インターバル・タイマ

クロック出力/ブザー出力制御

ウォッチドッグ・タイマ

PGA + ΔΣ A/Dコンバータ

A/Dコンバータ

マクロベータラブル・アップ

マクロベータラブル・ダウン

マクロベータラブル・ストップ

マクロベータラブル・リセット

マクロベータラブル・リセット・タイムアウト

マクロベータラブル・リセット・タイムアウト・コントロール

マクロベータラブル・リセット・タイムアウト・コントロール

割り込み機能

プロパティ 端子配置図 周辺機能*

コードを生成する

A/Dコンバータ動作設定

使用しない 使用する

コンパレータ動作設定

停止 許可

分解能設定

10 ビット 8 ビット

VREF(+)

AVDD 内部基準電圧

VREF(-)

AVSS

トリガ・モード設定

ソフトウェア・トリガ・モード

ハードウェア・トリガ・ノーウェイト・モード

ハードウェア・トリガ・ウェイト・モード

INTTM01

動作モード設定

連続セレクト・モード 連続スキャン・モード

ワンショット・セレクト・モード ワンショット・スキャン・モード

アナログ入力端子設定 (高精度チャネル)

変換開始チャネル設定

ANI0 ANI1 ANI2 ANI3

ANI4 ANI5 ANI6 ANI7

ANI8 ANI9

変換時間設定

変換開始チャネル設定 ANI0

基準電圧 2.7 ≤ AVDD ≤ 5.5 (V)

変換時間モード 標準1

変換時間 1216/CLK 38 (μs)

変換結果上限/下限値設定

ADLL ≤ ADCRH ≤ ADULで割り込み要求信号(INTAD)を発生

ADUL < ADCRHまたはADLL > ADCRHで割り込み要求信号(INTAD)を発生

上限値(ADUL) 255

下限値(ADLL) 0

割り込み設定

A/Dの割り込み許可(INTAD)

優先順位 レベル3(低優先順位)

周辺機能設定 (タイマ・アレイ・ユニット)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

③ タイマ・アレイ・ユニットを選択

④ タイマ・アレイ・ユニット1を選択

⑤ 一般設定タブでチャンネル0,1をPWM出力(マスタ/スレーブ)にする

⑥ PWMマスター周期を10msecに設定

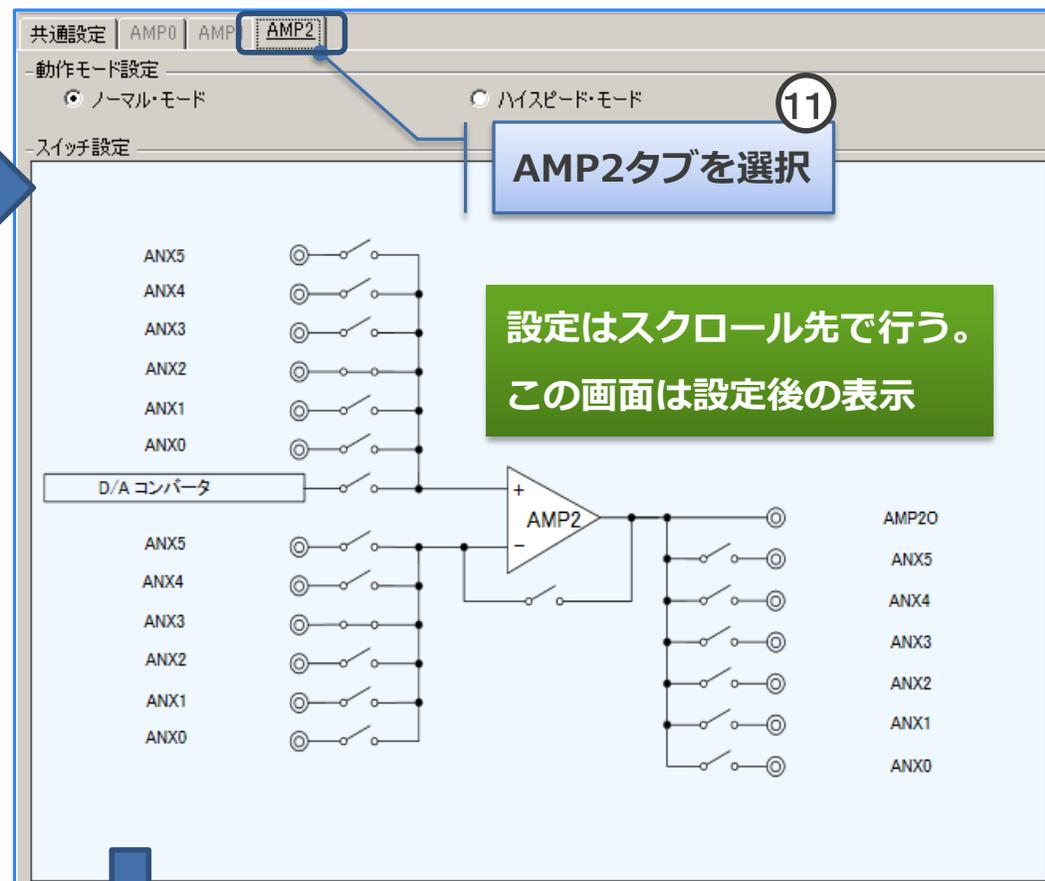
⑦ チャンネル1(スレーブ)タブを選択

⑧ 下記の設定にする
デューティー 0
初期値 1
アクティブ・ハイ

設定時の波形が表示される

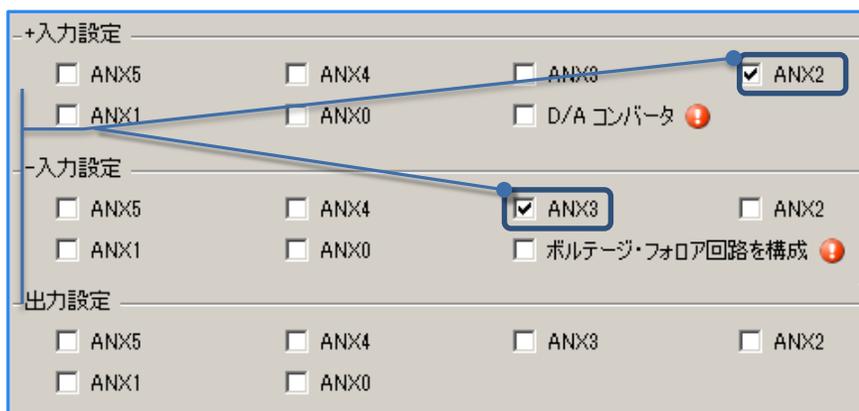
周辺機能設定 (コンフィギュラブル・アンプ)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



⑨ コンフィギュラブル・アンプを選択

⑫ +入力を"ANX2"
-入力を"ANX3"
それぞれチェックする



下へスクロール

周辺機能設定 (ポート)

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

Port1タブが前面に表示される

Port1を選択

P11を出力, 1を出力にチェック

ワンポイント

! にマウスカーソルを合わせると下記の文言が表示されます。

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。
P12はTO11で使われています。

P12は兼用端子として多くの機能が割り当てられています。今は、すでにTO11機能が使われているため "P12" としては使えないことを示しています。

端子番号	端子名	検索端子名	選択機能
D2	P41/ANI6/AMP1P/ANX3		ANX3
D3	ANI3/AMP0P/ANX1		設定されてい
D4	P12/SCK01/SCL01/TI11/TO11/INTP3/PCLBUZ0/TRGIOB/TRJ00		TO11

総括: 12 使用中の端子; 0 紛争; 0 警告

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_cg_main.c)

```
48 /*****
49 Global variables and functions
50 *****/
51 /* Start user code for global. Do not edit comment generated here */
52
53 uint16_t gAdResult; // AD変換結果
54 uint16_t gAdResult2; // AD変換結果2
55 uint16_t gAdResult3; // AD変換結果3
56 uint8_t gAdSelect; // 変換するA/Dチャンネル 0 = ANI0, 1= ANI1
57 uint16_t gTouchCount; // タッチパネルに触れていると判断した数
58 uint16_t gNoTocuhCount; // タッチパネルに触ってない回数
59 uint8_t gTouchSeparate; // タッチパネルに触っている状態 0=OFF, 1=ON
60 uint16_t gTmCounter // タイマカウンタ
61
62 /* End user code. Do not edit comment generated here */
63
64 void R_MAIN_UserInit(void);
65 /*****
66 * Function Name: main
67 * Description : This function implements main function.
68 * Arguments : None
69 * Return Value : None
70 *****/
71 void main(void)
72 {
73 R_MAIN_UserInit();
74 /* Start user code. Do not edit comment generated here */
75 while (1U)
76 {
77 ;
78 }
79 /* End user code. Do not edit comment generated here */
80 }
```

```
81 /*****
82 * Function Name: R_MAIN_UserInit
83 * Description : This function adds user code before implementing main function.
84 * Arguments : None
85 * Return Value : None
86 *****/
87 void R_MAIN_UserInit(void)
88 {
89 /* Start user code. Do not edit comment generated here */
90
91 R_TAU0_Channel0_Start(); // タイマチャンネル0 開始
92 gAdSelect = 0;
93 gTouchCount = 0;
94 gNoTocuhCount = 0;
95 gTmCounter = 0;
96 R_CAMP2_Start();
97 EI();
98
99 /* End user code. Do not edit comment generated here */
100 }
```

アンプは最初から
使えるようにする

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_tau_user.c)

```
29 /*****  
30 Includes  
31 *****/  
32 #include "r_cg_macrodriver.h"  
33 #include "r_cg_tau.h"  
34 /* Start user code for include. Do not edit comment generated here */  
35  
36 #include "r_cg_adc.h"  
37  
38 /* End user code. Do not edit comment generated here */  
39 #include "r_cg_userdefine.h"  
40  
41 /*****  
42 Pragma directive  
43 *****/  
44 #pragma interrupt r_tau0_channel0_interrupt(vect=INTTM00)  
45 #pragma interrupt r_tau1_channel0_interrupt(vect=INTTM10)  
46 #pragma interrupt r_tau1_channel1_interrupt(vect=INTTM11)  
47 /* Start user code for pragma. Do not edit comment generated here */  
48  
49 extern uint8_t gAdSelect;  
50 extern uint16_t gTmCounter;  
51  
52 /* End user code. Do not edit comment generated here */  
53  
54 /*****  
55 Global variables and functions  
56 *****/  
57 /* Start user code for global. Do not edit comment generated here */  
58 /* End user code. Do not edit comment generated here */
```

```
60 /*****  
61 * Function Name: r_tau0_channel0_interrupt  
62 * Description : This function INTTM00 interrupt service routine.  
63 * Arguments : None  
64 * Return Value : None  
65 *****/  
66 static void __near r_tau0_channel0_interrupt(void)  
67 {  
68 /* Start user code. Do not edit comment generated here */  
69 ad_channel_t ch;  
70 switch ( gAdSelect )  
71 {  
72 case 0:  
73 ch = ADCHANNEL0;  
74 break;  
75  
76 case 1:  
77 ch = ADCHANNEL1;  
78 break;  
79  
80 case 2:  
81 ch = ADCHANNEL2;  
82 break;  
83  
84 default:  
85 ch = ADCHANNEL0;  
86 break;  
87 }  
88 R_ADC_Set_ADChannel( ch );  
89 R_ADC_Start();  
90 gTmCounter++;  
91 /* End user code. Do not edit comment generated here */  
92 }
```

チャンネル番号2の変換結果
処理を追加

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_tau_user.c)

```
95 /*****  
96 * Function Name: r_tau1_channel0_interrupt  
97 * Description : This function INTTM10 interrupt service routine.  
98 * Arguments : None  
99 * Return Value : None  
100 *****/  
101 static void __near r_tau1_channel0_interrupt(void)  
102 {  
103     /* Start user code. Do not edit comment generated here */  
104  
105     // LED P11点灯  
106     P1_bit.no1 = 0;  
107  
108     /* End user code. Do not edit comment generated here */  
109 }  
110 /*****  
111 * Function Name: r_tau1_channel1_interrupt  
112 * Description : This function INTTM11 interrupt service routine.  
113 * Arguments : None  
114 * Return Value : None  
115 *****/  
116 static void __near r_tau1_channel1_interrupt(void)  
117 {  
118     /* Start user code. Do not edit comment generated here */  
119  
120     // LED P11消灯  
121     P1_bit.no1 = 1;  
122  
123     /* End user code. Do not edit comment generated here */  
124 }  
125  
126 /* Start user code for adding. Do not edit comment generated here */  
127 /* End user code. Do not edit comment generated here */
```

💡 ワンポイント

コード生成で10msecのPWMを作成しました。そのPWMに対する割り込みは2度発生します。下図はデューティ比75%の場合です。75%がLED点灯時間、25%がLED消灯時間となります。プログラム実行中にデューティ比を変更してLEDの輝度を変えています。

チャンネル0(マスタ) | チャンネル1(スレーブ)

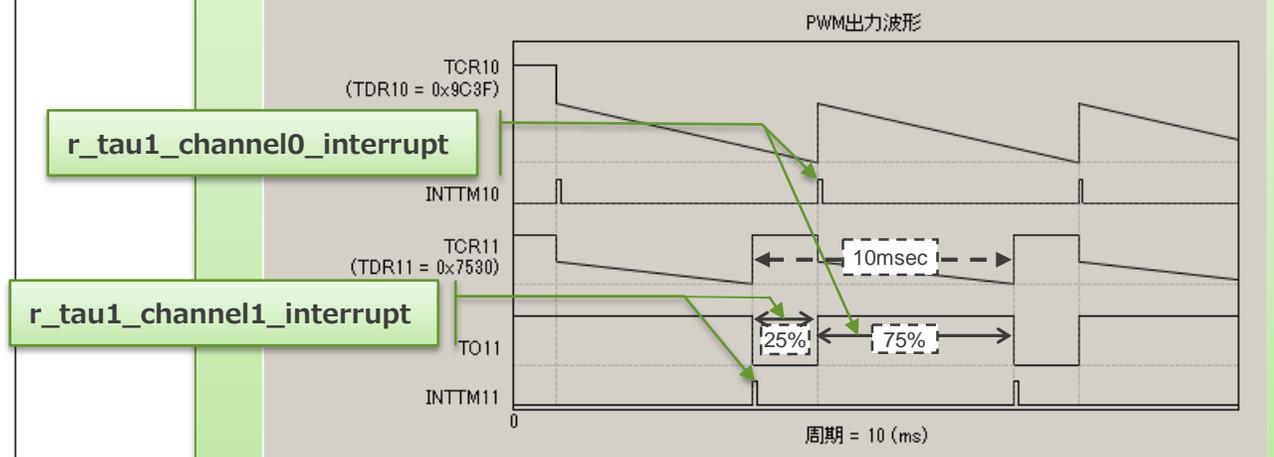
-PWMデューティ設定

デューティ: 75 (%) (実際の値: 75%)

-出力設定

初期出力値: 1

出力レベル: アクティブ・ハイ



- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_adc_user.c)

```
20 /*****
21 * File Name   : r_cg_adc_user.c
22 * Version    : Code Generator for RL78/I1E V1.02.00.06 [12 Aug 2015]
23 * Device(s)  : R5F11CCC
24 * Tool-Chain : CCRL
25 * Description: This file implements device driver for ADC module.
26 * Creation Date: 2016/04/08
27 *****/
28
29 /*****
30 Includes
31 *****/
32 #include "r_cg_macrodriver.h"
33 #include "r_cg_adc.h"
34 /* Start user code for include. Do not edit comment generated here */
35 #include "r_cg_tau.h"
36 /* End user code. Do not edit comment generated here */
37 #include "r_cg_userdefine.h"
38
39 :
40
41 /*****
42 Global variables and functions
43 *****/
44 /* Start user code for global. Do not edit comment generated here */
45
46 extern uint16_t gAdResult;
47 extern uint16_t gAdResult2;
48 extern uint16_t gAdResult3;
49 extern uint8_t gAdSelect;
50 extern uint16_t gTouchCount;
51 extern uint16_t gNoTocuhCount;
52 extern uint8_t gTouchSeparate;
53 extern uint16_t gTmCounter;
54 /* End user code. Do not edit comment generated here */
```

```
60 /*****
61 * Function Name: r_adc_interrupt
62 * Description  : None
63 * Arguments   : None
64 * Return Value : None
65 *****/
66 static void __near r_adc_interrupt(void)
67 {
68     /* Start user code. Do not edit comment generated here */
69     uint16_t result;
70
71     R_ADC_Stop();
72     R_ADC_Get_Result( &result );
73     switch ( gAdSelect )
74     {
75     case 0:
76         gAdResult = result;
77         break;
78
79     case 1:
80         gAdResult2 = result;
81
82         if ( result > D_TOUCH_AD )
83         {
84             gTouchCount++;
85             gNoTocuhCount = 0;
86         }
87         else if ( result > D_NOTOUCH_ADMIN )
88         {
89             gNoTocuhCount++;
90         }
91     }
```

次ページへ

- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_adc_user.c)

前ページから

```
92  if ( gNoTocuhCount > D_NOTOUCH_COUNT )
93  {
94      gTouchCount = 0;
95      gTouchSeparate = 0;
96  }
97
98  if ( (gTouchCount > D_TOUCH_COUNT) &&( gTouchSeparate == 0) )
99  {
100     // LED点灯と消灯を繰り返す
101     P1_bit.no0 ^= 1;
102     gTouchSeparate = 1;
103 }
104 break;
105
106 case 2:
107     gAdResult3 = result;
108     if ( gAdResult3 > D_MIC_ADTP )
109     {
110         comp = ( gAdResult3 - D_MIC_ADTP ) * 1000;
111         if ( comp > D_MIC_LEDMAXPWM )
112         {
113             comp = D_MIC_LEDMAXPWM;
114         }
115         if ( ( gTmCounter > D_MIC_LEDCHANGE ) | ( comp > TDR11 ) )
116         { // LEDの明るさを100msecごとに変更する
117             TDR11 = comp;
118             gTmCounter = 0;
119         }
120     }
```

```
121     else
122     {
123         if ( gTmCounter > D_MIC_LEDOFFDEALY )
124         {
125             TDR11 = 0;
126             gTmCounter = 0;
127         }
128     }
129     break;
130
131     default:
132         gAdResult = result;
133         break;
134 }
135
136 gAdSelect++;
137 if ( gAdSelect >= D_MAXAD_CHANNEL )
138 {
139     gAdSelect = 0;
140 }
141
142 /* End user code. Do not edit comment generated here */
143 }
144
145 /* Start user code for adding. Do not edit comment generated here */
146 /* End user code. Do not edit comment generated here */
```

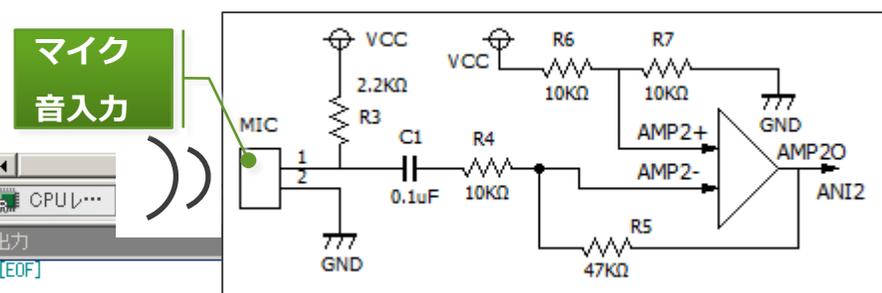
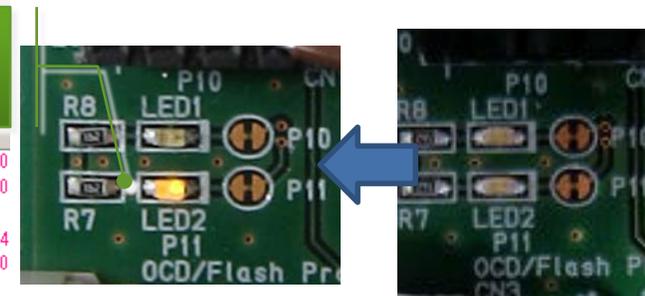
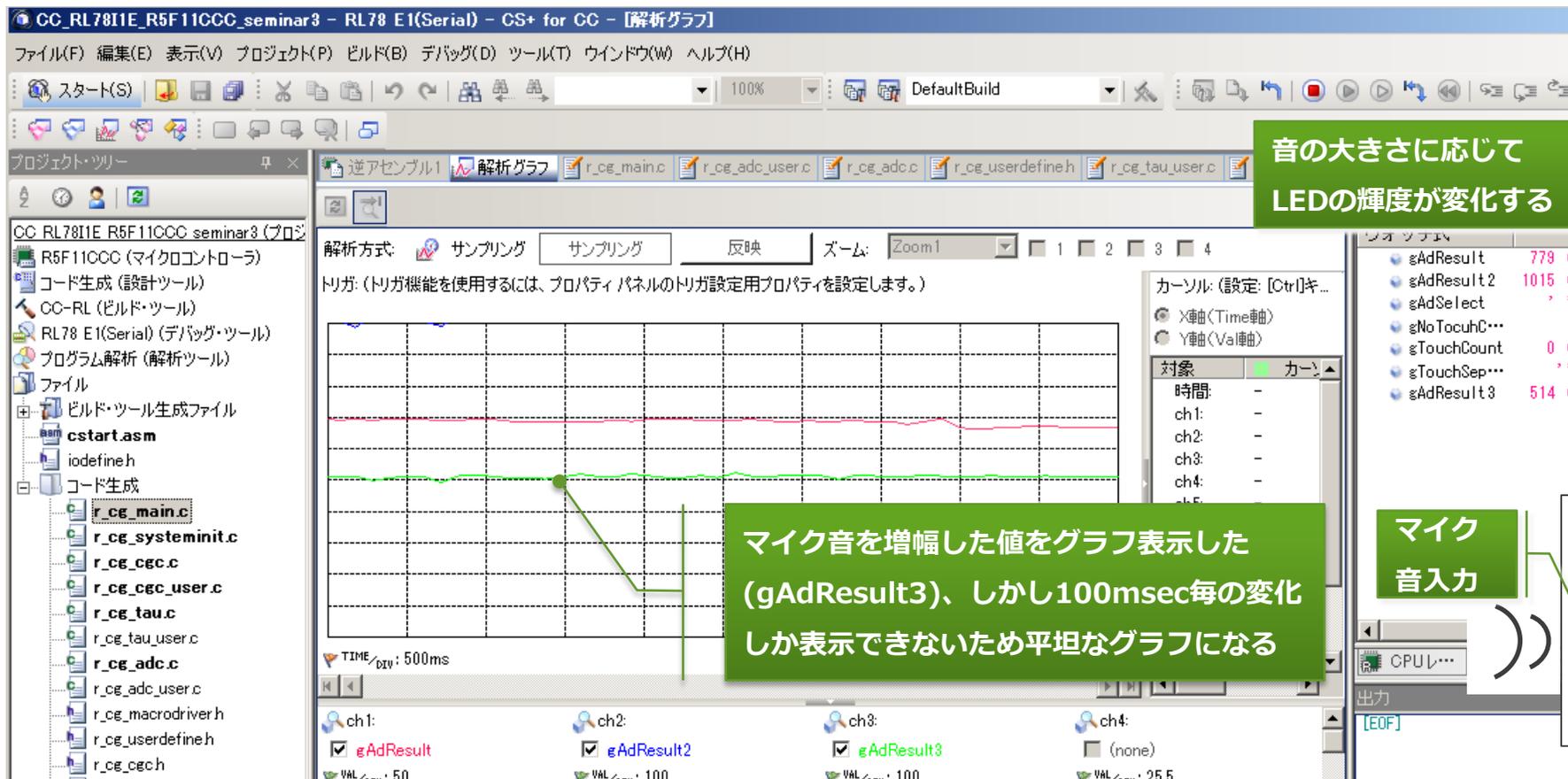
- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える

コード生成とプログラム編集 (r_cg_userdefine.h)

```
20  /*****
21  * File Name   : r_cg_userdefine.h
22  * Version    : Code Generator for RL78/I1E V1.02.00.06 [12 Aug 2015]
23  * Device(s)  : R5F11CCC
24  * Tool-Chain : CCRL
25  * Description : This file includes user definition.
26  * Creation Date: 2016/04/08
27  *****/
28  #ifndef _USER_DEF_H
29  #define _USER_DEF_H
30
31  /*****
32  User definitions
33  *****/
34
35  /* Start user code for function. Do not edit comment generated here */
36
37  #define D_MAX_AD_CHANNEL      3           // A/D変換するチャンネル数
38
39  #define D_TOUCH_AD           100        // タッチパッドに触ったと判断する値
40  #define D_NOTOUCH_ADMIN     900        // タッチパッド非接触時を判断するA/D最小値
41  #define D_TOUCH_COUNT       10         // タッチパッドを触ったと判断する回数
42  #define D_NOTOUCH_COUNT     100        // タッチパッドを触っていないと判断する回数
43
44  #define D_MIC_AD_TYP         550        // マイクA/D入力の無音状態値
45  #define D_MIC_LED_OFF_DEALY  500       // LEDを消去するまでの時間 msec
46  #define D_MIC_LED_CHANGE    100       // LEDの輝度を変えるタイミング msec
47  #define D_MIC_LED_MAX_PWM   40000     // LEDの輝度を変えるPWMの最大値
48
49  /* End user code. Do not edit comment generated here */
50  #endif
```

プログラムの実行とデバッグ (解析グラフ表示)

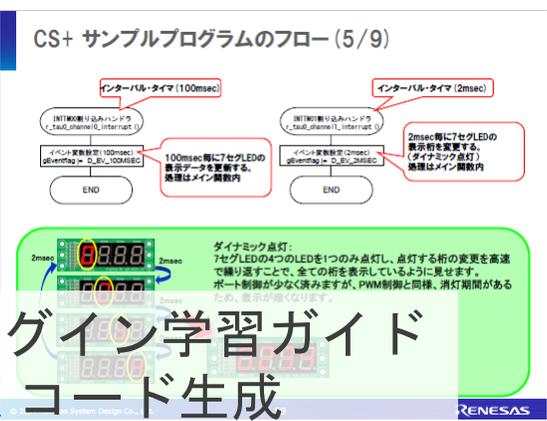
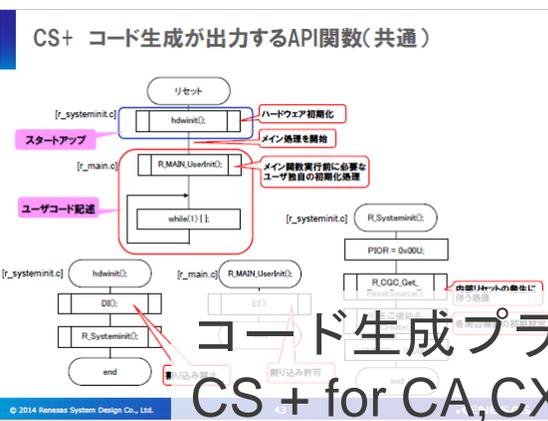
- A. 照度センサを使って明るさを検知
- B. 簡易タッチセンサでLEDをON/OFF
- C. マイクを使って音でLEDの輝度を変える



最後に

他にもコード生成のガイドがあるのでご活用ください。

<https://www.renesas.com/ja-jp/products/software-tools/tools/code-generator/code-generator-plug-in.html>



RL78/I1Eコード生成
RL78/I1E + 脈拍センサデモ
コンフィギュラブル・アンプ使用例

文書番号 R20U3745J0100
ツールビジネス課 ツール技術部
ルネサスシステムデザイン株式会社

汎用アンプを使ったデモ概要

センサで脈拍を検出する
アンプで増幅した脈拍をグラフ表示する
CS+上でリアルタイムに脈拍をグラフ表示

3.プログラム編集
4.ビルド

コード生成プラグイン学習ガイド CS + for CA,CX コード生成

RL78/I1Eコード生成 脈拍センサデモ コンフィギュラブル・アンプ使用例 CS + for CA,CX コード生成 e2 studio コード生成

CS+ デバッグ RL78/G13

使用するデバッグツール (既定) [e2studio] を変更
標準となるオプションを設定した場合、そのプロパティは太字表示されます。

CS+ プログラム実行 (サンプルプログラム動作確認)

SW1押下: 7セグLEDに0から9の数字と、(トント)が表示されます。
SW2押下: ボテンションメータで設定した値のA/D変換値(0~1023)が7セグLEDに表示されます。
SW4押下: PWMで設定した周波数でLED (P63) が点灯します。(LED (P62) は比較用に点灯します)
CPUボードのSW1押下: リセット状態に戻ります。
SW3押下: PCでキー入力したデータをターミナルに表示します。(コールバック)

プログラムの実行 (プログラム実行中の表示)

gAdResultの値がリアルタイムでグラフ表示される
8.グラフがスクロール表示

e2 studioでデバッグツールの設定 (プログラムの実行)

20.プログラムを実行
21.プログラム実行中でも変数表示が更新される
gAdResult変数がリアルタイムでグラフ表示される