

Applilet EZ PL で電源電圧監視を行う

RL78/G11 は 1.6V~5.5V の広い電圧範囲に対応し、少ない消費電力での動作が可能です。このため、バッテリーでの動作に適しています。1.5V の乾電池は放電終了電圧が 0.9V 程度なので、単 4 型の電池を 2 個使うことで、3.3V~1.8V で動作させることを考えてみます。

乾電池動作の場合には、電池が消耗したことを通知して、交換を促すことは意味がありません。

そこで、Applilet EZ PL for RL78 を使って、電源電圧の監視を行います。RL78/G11 には、1.45V の内部基準電圧が内蔵されているので、これを A/D コンバータで測定すると、VDD が 1.8V なら 0x0338 となります。これより大きい変換結果になったら、電源電圧が 1.8V を切ったとして外部に通知し、STOP モードに入ることにします。

1. Applilet EZ PL for RL78 でのマイクロコントローラ設定

「設定(S)」メニューの「マイクロコントローラの設定」を右の示すように設定します。

ウォッチドッグ・タイマは使用しません。

P125/RESET 端子はオンチップ・デバッグで使用するだけなので、リセット動作設定の「外部リセット端子の選択」は「使用する (RESET)」にしておきます。

今回は、1.8V を検出するので、「POR/SPOR 検出電圧」(電圧固定の POR が選択できるように見えるのは気になります) を 1.8V より低く設定します

(YRPBRL78G11 を使用する場合には、3V 以上には設定しないでください。そうしないと、以降はオンチップ・デバッグできなくなります)。

動作モードは、LS (低速メイン) モードに設定します。

高速オンチップ・オシレータの周波数は 8MHz に設定しておきます。

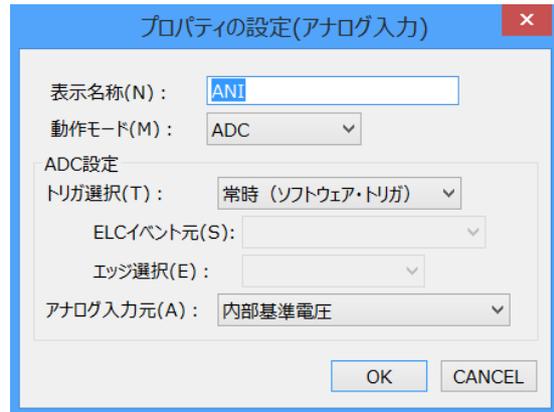


2. 使用するパネルの設定

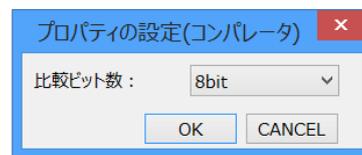
今回は、アナログ入力、コンパレータ、スタンバイ、デジタル出力及び論理和ゲートを使用します。

アナログ入力のパネル・プロパティは右のように設定します。

電源電圧は常時確認する必要があるので、「トリガ選択」は「常時（ソフトウェア・トリガ）」を選択し、「アナログ入力元」は「内部基準電圧」に設定しておきます。



コンパレータのパネル・プロパティは「比較ビット数」を「8bit」に設定します。



スタンバイは単に、STOP モードに入れるだけです。解除することはないのですが、設定が必要なので、INTP0 で解除するように設定しておきます。他の機能で INTP0 を使用している場合は、他の空いている要因に変更して下さい。



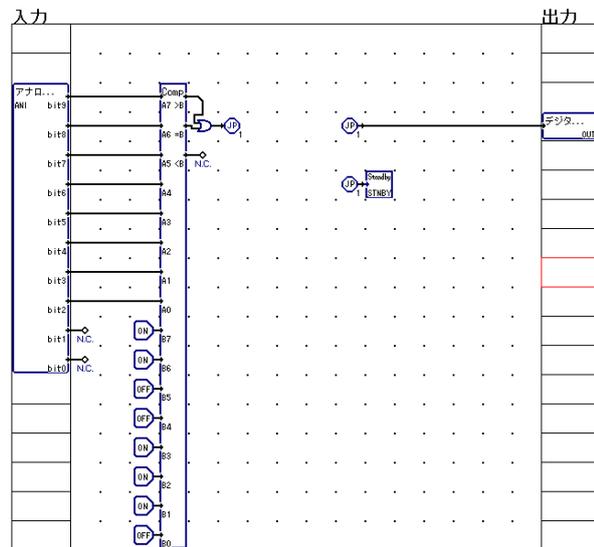
デジタル出力は P56 をアクティブ H に設定します。

最終的に作成したパネルは右のような構成になります。

アナログ入力の bit9～bit2 をコンパレータの A7～A0 に接続します。アナログ入力の bit1 と bit0 は未接続（NC）とします。

コンパレータの B 側の入力は B7, B6, B3～B1 は「Every ON」を接続し、残りの B 入力は「Every OFF」を接続します。

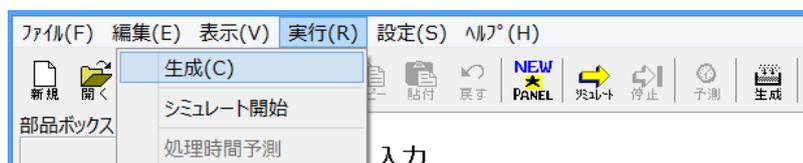
コンパレータ出力は「A>B」と「A=B」の論理和をとり、スタンバイ・パネルの入力とデジタル出力に接続します。



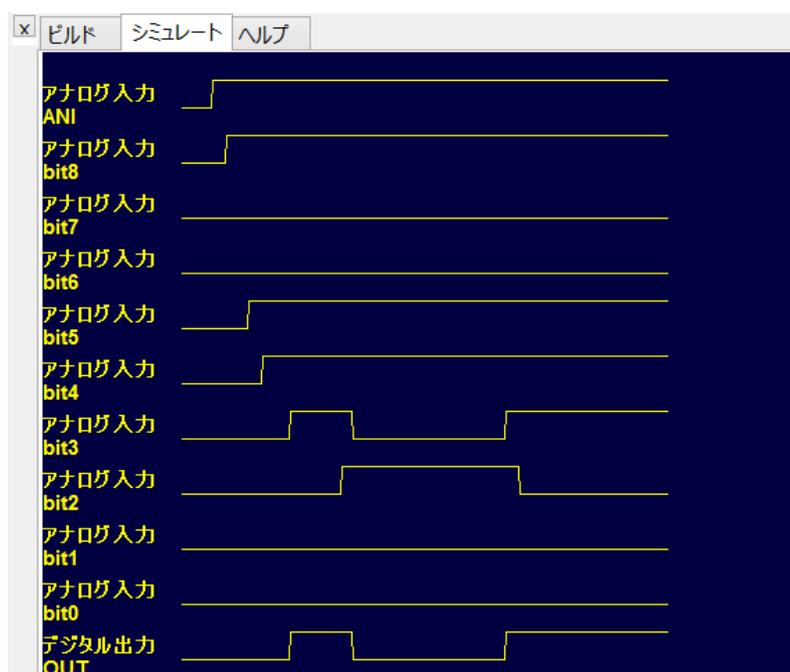
これ以外の機能が必要な場合は、必要なパネルを追加設定してください。ノイズ対応で、未使用の端子はデジタル出力に設定（設定することを推奨します）する場合には上の図の続き（下）に追加してください。

3. 結果の確認

パネルの定義が完了したら、結果を保存して、生成します。



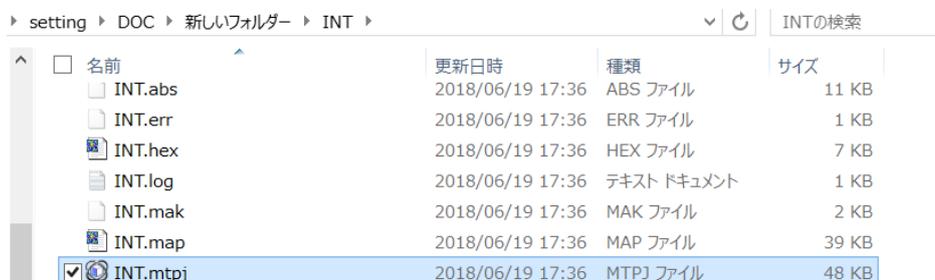
次にシミュレータで動作を確認します。なお、スタンバイはシミュレートできないので、デジタル出力で確認します。アナログ入力の上位からセットしていき、bit3をHにセットして、0x338以上になると、一番下にあるデジタル出力がHとなります。



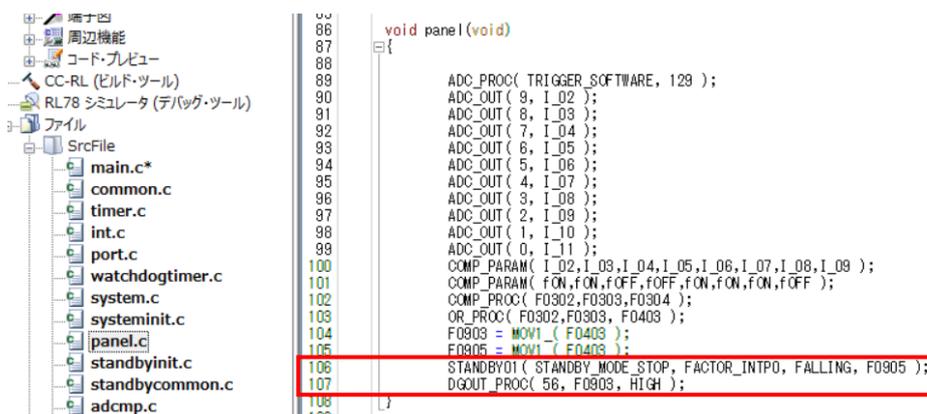
ここまで確認したら、以降は生成されたプロジェクトをCS+で確認します。

4. CS+での確認

プロジェクトファイルの保存先に指定したフォルダにあるCS+のプロジェクト・ファイル（今回は、INT.mtpj）をダブル・クリックしてCS+を起動します。



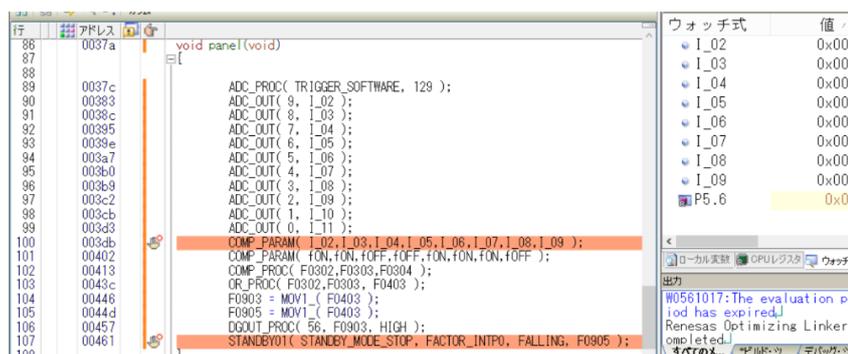
CS+が起動したら、左側のプロジェクト・ツリーの「ファイル」の「SrcFile」にある「panel.c」がApplileet EZ PL for RL78 で定義したパネル関係のcソース・ファイルが記述されたものです。この内容を、下に示します。



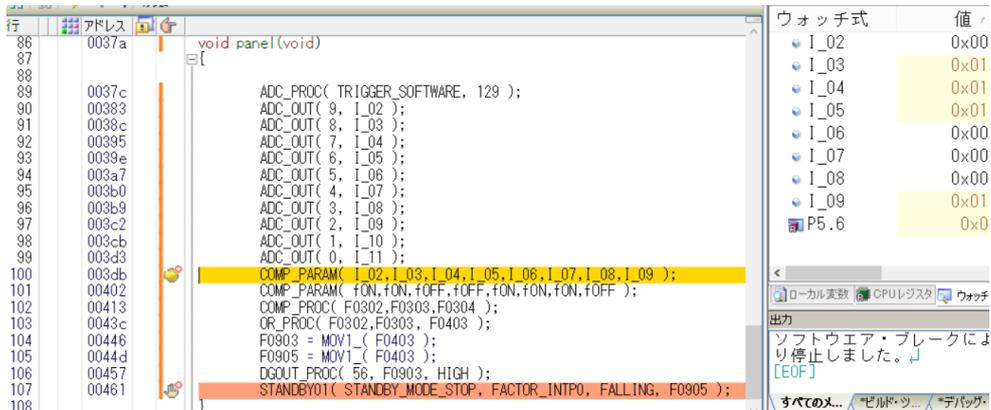
ここで、問題が見つかりました。106行目が、スタンバイ処理で、107行目がデジタル出力です。つまり、スタンバイにはいると、デジタル出力ができなくなることが分かります。

そこで、この2行の順序を入れ替えます。

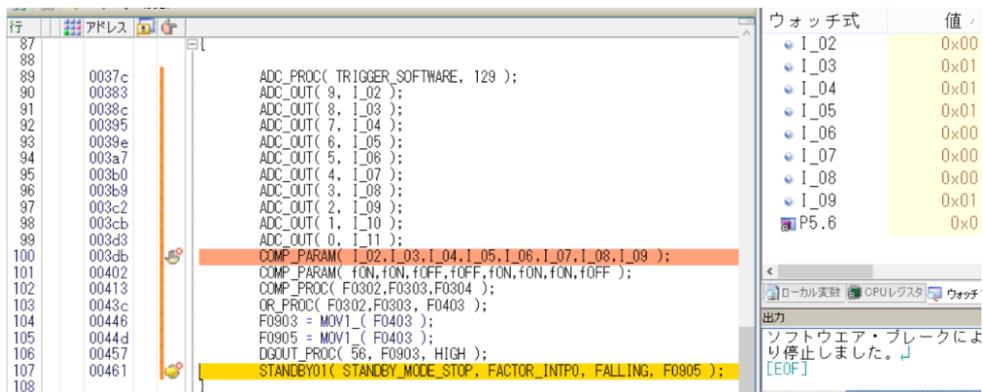
ビルドして、E2OB 経由でダウンロードした状態を下に示します。ここでは、コンパレータへの入力（100行目）とスタンバイの呼び出し（107行目）にブレークポイントを設定し、コンパレータへの入力をウォッチ式に登録しています。



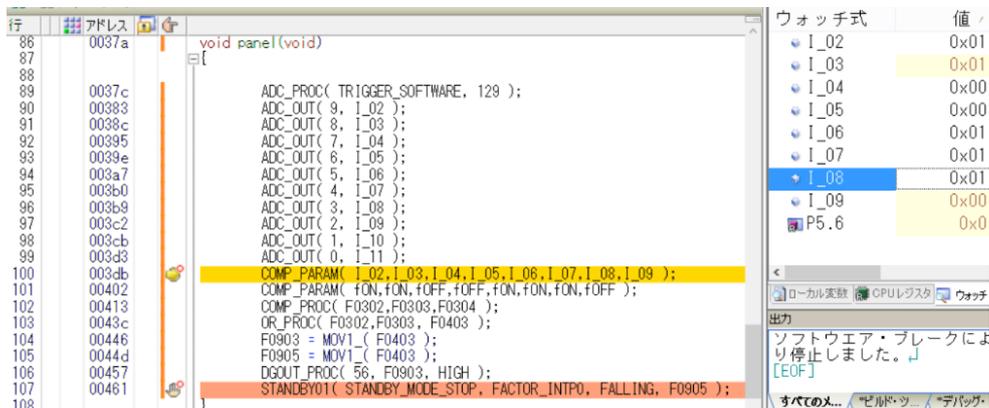
この状態で、ブレークポイントを有効にして実行してみます。



これで、I_02~I_09には A/D 変換結果の上位 8bit が格納されています。このまま、107 行目まで実行させます。このとき、P5.6 は 0 のままです。



引き続き、100 行目まで実行させます。この場合には、スタンバイから抜けて、100 行目の実行前でブレークします。E2OB では電源電圧を変更できないので、A/D 変換結果の I_02~I_09 の値をウォッチ式で、I_02、I_03、I_06~I_08 を 0x01、その他を 0x00 に設定します。この状態を下に示します。



この状態で、107 行目まで実行させます。

すると、下に示すように P5.6 がセットされているのが分かります。

The screenshot shows a debugger window with the following assembly code:

```

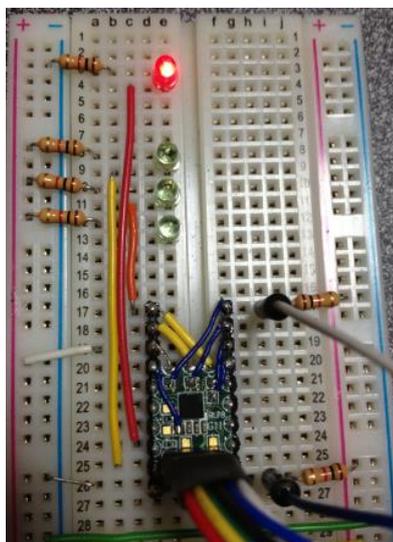
86 0037a void panel(void)
87 {
88
89 0037c     ADC_PROC( TRIGGER_SOFTWARE, 129 );
90 00383     ADC_OUT( 9, I_02 );
91 0038c     ADC_OUT( 8, I_03 );
92 00395     ADC_OUT( 7, I_04 );
93 0039e     ADC_OUT( 6, I_05 );
94 003a7     ADC_OUT( 5, I_06 );
95 003b0     ADC_OUT( 4, I_07 );
96 003b9     ADC_OUT( 3, I_08 );
97 003c2     ADC_OUT( 2, I_09 );
98 003cb     ADC_OUT( 1, I_10 );
99 003d3     ADC_OUT( 0, I_11 );
100 003db     COMP_PARAM( I_02, I_03, I_04, I_05, I_06, I_07, I_08, I_09 );
101 00402     COMP_PARAM( FON, FON, FOFF, FOFF, FON, FON, FON, FOFF );
102 00413     COMP_PROC( F0302, F0303, F0304 );
103 0043c     OR_PROC( F0302, F0303, F0403 );
104 00446     F0903 = MOV1( F0403 );
105 0044d     F0905 = MOV1( F0403 );
106 00457     DOUT_PROC( 56, F0903, HIGH );
107 00461     STANDBY01( STANDBY_MODE_STOP, FACTOR_INTPO, FALLING, F0905 );

```

The watch window on the right shows the following values:

ウォッチ式	値
I_02	0x01
I_03	0x01
I_04	0x00
I_05	0x00
I_06	0x01
I_07	0x01
I_08	0x01
I_09	0x00
P5.6	0x1

このとき、P5.6 で一番上の LED を点灯させることで確認しています。



さらに、実行させると CS+は下のよう実行中（STOP モード）で止まってしまい、ブレーク・ポイントまで来ません。

The screenshot shows a debugger window with the following assembly code:

```

86 0037a void panel(void)
87 {
88
89 0037c     ADC_PROC( TRIGGER_SOFTWARE, 129 );
90 00383     ADC_OUT( 9, I_02 );
91 0038c     ADC_OUT( 8, I_03 );
92 00395     ADC_OUT( 7, I_04 );
93 0039e     ADC_OUT( 6, I_05 );
94 003a7     ADC_OUT( 5, I_06 );
95 003b0     ADC_OUT( 4, I_07 );
96 003b9     ADC_OUT( 3, I_08 );
97 003c2     ADC_OUT( 2, I_09 );
98 003cb     ADC_OUT( 1, I_10 );
99 003d3     ADC_OUT( 0, I_11 );
100 003db     COMP_PARAM( I_02, I_03, I_04, I_05, I_06, I_07, I_08, I_09 );
101 00402     COMP_PARAM( FON, FON, FOFF, FOFF, FON, FON, FON, FOFF );
102 00413     COMP_PROC( F0302, F0303, F0304 );
103 0043c     OR_PROC( F0302, F0303, F0403 );
104 00446     F0903 = MOV1( F0403 );
105 0044d     F0905 = MOV1( F0403 );
106 00457     DOUT_PROC( 56, F0903, HIGH );
107 00461     STANDBY01( STANDBY_MODE_STOP, FACTOR_INTPO, FALLING, F0905 );

```

The watch window on the right shows the following values:

ウォッチ式	値
I_02	?
I_03	?
I_04	?
I_05	?
I_06	?
I_07	?
I_08	?
I_09	?
P5.6	?

これで、確認は完了です。簡単ですが、Appilet EZ PL for RL78 は結構使えそうです。

4. おまけ

未使用端子処理を含めた場合に生成される panel 関数を下に示します。赤く囲んだ部分が、未使用端子処理部分です。これらは、単純に出力ポートに設定すれば十分なので、削除しても構いません。

```
101 void panel(void)
102 {
103
104     ADC_PROC( TRIGGER_SOFTWARE, 129 );
105     ADC_OUT( 9, I_02 );
106     ADC_OUT( 8, I_03 );
107     ADC_OUT( 7, I_04 );
108     ADC_OUT( 6, I_05 );
109     ADC_OUT( 5, I_06 );
110     ADC_OUT( 4, I_07 );
111     ADC_OUT( 3, I_08 );
112     ADC_OUT( 2, I_09 );
113     ADC_OUT( 1, I_10 );
114     ADC_OUT( 0, I_11 );
115     COMP_PARAM( I_02, I_03, I_04, I_05, I_06, I_07, I_08, I_09 );
116     COMP_PARAM( FON, FON, FOFF, FOFF, FON, FON, FON, FOFF );
117     COMP_PROC( F0302, F0303, F0304 );
118     OR_PROC( F0302, F0303, F0403 );
119     F0903 = MOV1_( F0403 );
120     F0905 = MOV1_( F0403 );
121     DGOUT_PROC( 56, F0903, HIGH );
122     STANDBY01( STANDBY_MODE_STOP, FACTOR_INTPO, FALLING, F0905 );
123     DGOUT_PROC( 0, FOFF, LOW );
124     DGOUT_PROC( 1, FOFF, LOW );
125     DGOUT_PROC( 33, FOFF, LOW );
126     DGOUT_PROC( 20, FOFF, HIGH );
127     DGOUT_PROC( 21, FOFF, HIGH );
128     DGOUT_PROC( 22, FOFF, HIGH );
129     DGOUT_PROC( 23, FON, HIGH );
130     DGOUT_PROC( 30, FOFF, HIGH );
131     DGOUT_PROC( 31, FOFF, HIGH );
132     DGOUT_PROC( 32, FOFF, HIGH );
133     DGOUT_PROC( 51, FOFF, HIGH );
134     DGOUT_PROC( 52, FOFF, HIGH );
135     DGOUT_PROC( 53, FOFF, HIGH );
136     DGOUT_PROC( 54, FOFF, HIGH );
137     DGOUT_PROC( 55, FOFF, HIGH );
138
139 }
```

今回は、A/Dの機能を使った電源電圧検出で、これだけで使用するプログラムではありません。これに必要な機能のパネルを追加することで、やりたいこと実現してください。

以上