SmartRA 学習キット
チュートリアル 4

ルネサス エレクトロニクス社　RA マイコン搭載
HSB シリーズマイコンボード　評価キット

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子

REV.1.0.0.0

ー目　次ー

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

1. 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複写・複製・転載はできません。
4. 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的･用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。
ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。
保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。
本製品を使った二次製品の保証は致し兼ねます。

安全上のご注意

製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上でお読み下さい。

表記の意味

警告 取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性がある事が想定される

注意 取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが可能性がある事が想定される

絵記号の意味

	**一般指示** 使用者に対して指示に基づく行為を強制するものを示します		**一般禁止** 一般的な禁止事項を示します
	**電源プラグを抜く** 使用者に対して電源プラグをコンセントから抜くように指示します		**一般注意** 一般的な注意を示しています

警告

以下の警告に反する操作をされた場合、本製品及びユーザシステムの破壊・発煙・発火の危険があります。マイコン内蔵プログラムを破壊する場合もあります。

1. 本製品及びユーザシステムに電源が入ったままケーブルの抜き差しを行わないでください。
2. 本製品及びユーザシステムに電源が入ったままで、ユーザシステム上に実装されたマイコンまたはIC等の抜き差しを行わないでください。
3. 本製品及びユーザシステムは規定の電圧範囲でご利用ください。
4. 本製品及びユーザシステムは、コネクタのピン番号及びユーザシステム上のマイコンとの接続を確認の上正しく扱ってください。

発煙・異音・異臭にお気付きの際はすぐに使用を中止してください。

電源がある場合は電源を切って、コンセントから電源プラグを抜いてください。そのままご使用すると火災や感電の原因になります。

⚠注意

以下のことをされると故障の原因となる場合があります。

1. 静電気が流れ、部品が破壊される恐れがありますので、ボード製品のコネクタ部分や部品面には直接手を触れないでください。
2. 次の様な場所での使用、保管をしないでください。

   ホコリが多い場所、長時間直射日光があたる場所、不安定な場所、衝撃や振動が加わる場所、落下の可能性がある場所、水分や湿気の多い場所、磁気を発するものの近く
3. 落としたり、衝撃を与えたり、重いものを乗せないでください。
4. 製品の上に水などの液体や、クリップなどの金属を置かないでください。
5. 製品の傍で飲食や喫煙をしないでください。

ボード製品では、裏面にハンダ付けの跡があり、尖っている場合があります。

取り付け、取り外しの際は製品の両端を持ってください。裏面のハンダ付け跡で、誤って手など怪我をする場合があります。

CD メディア、フロッピーディスク付属の製品では、故障に備えてバックアップ(複製)をお取りください。

製品をご使用中にデータなどが消失した場合、データなどの保証は一切致しかねます。

アクセスランプがある製品では、アクセスランプの点灯中に電源を切ったり、パソコンをリセットをしないでください。

製品の故障や、データ消失の原因となります。

本製品は、医療、航空宇宙、原子力、輸送などの人命に関わる機器やシステム及び高度な信頼性を必要とする設備や機器などに用いられる事を目的として、設計及び製造されておりません。

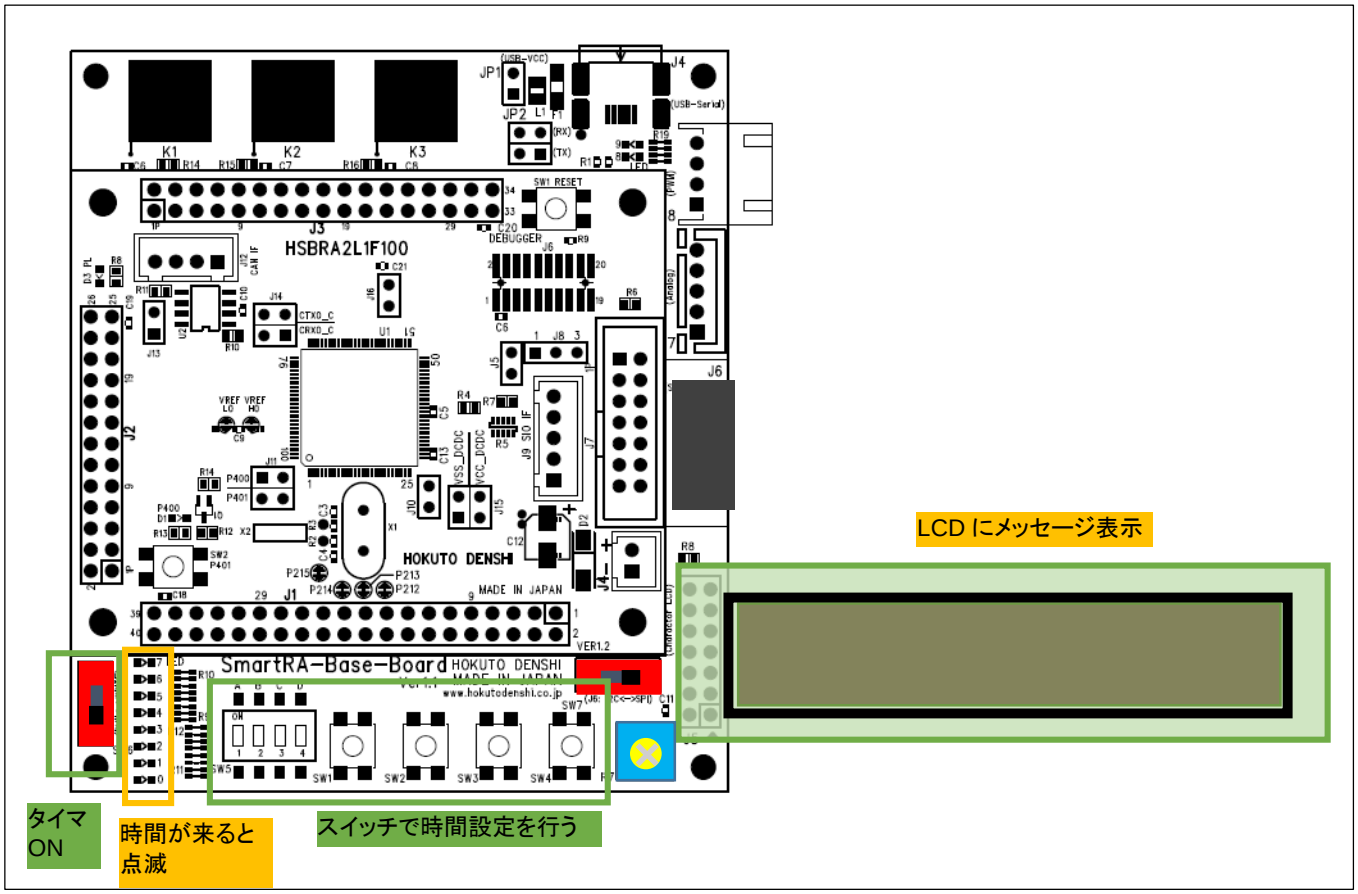
医療、航空宇宙、原子力、輸送などの設備や機器、システムなどに本製品を使用され、本製品の故障により、人身や火災事故、社会的な損害などが生じても、弊社では責任を負いかねます。お客様ご自身にて対策を期されるようご注意ください。

1. RA2L1_RTC_TIMER

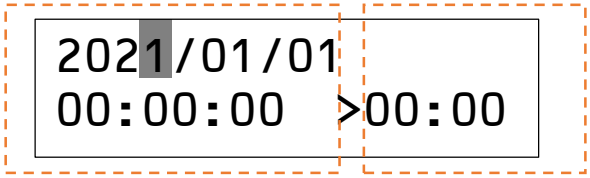
マイコン内蔵のリアルタイムクロックタイマ機能を使用したチュートリアルです。

リアルタイムクロックは、時計と関連するクロックで、1 秒（や 1 分や 1 時間）をカウントするのに適しています。また、カレンダ機能と連動しており、実時間（2021/5/10 15:09:30 など）をカウントします。

1.1. プログラムの動作



プログラムを実行すると、まず現在時刻の時間合わせモードとなります。



現在時刻　　タイマ

SW1	SW2	SW3	SW4
設定項目：戻る	設定項目：進む	値：減らす	値：増やす

現在時間の設定には、SW1~SW4 を使います。

・まずは、年合わせになっているので、合っていれば、SW2 で次の項目（月）に移動してください
・SW4 で 01→02→03 と値を増やす事ができます。SW3 は値を減らす場合に押します
・年、月、日、時、分を順次合わせてください
・分まで合わせて、SW2 を押すと、秒のところにカーソルが移動します。秒は、SW3, SW4 で設定はできません
・秒のところにカーソルを合わせて、SW4 を押すと設定完了です

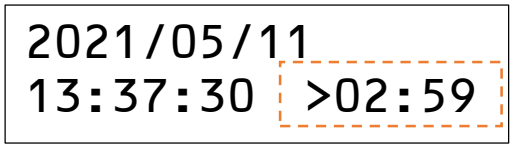
現在時間がカウントを開始します。これは、マイコンの RTC リアルタイムクロックという機能の、カレンダカウントモードです。

このサンプルプログラムは、現在時刻の表示以外に、タイマ（ラーメンタイマ、キッチンタイマの様な機能）を持たせており、DIP スイッチでタイマ時間をセットします。ON 側に倒すと、

SW5-A	SW5-B	SW5-C	SW5-D
+8 分	+4 分	+2 分	+1 分

の時間設定となります。3 分の時間設定を行いたいときは、SW5-C, SW5-D を ON 側に倒してください。（~15 分まで 1 分単位での設定が可能です）

タイマスタートは、SW6 です。SW6 を上側に倒すと、タイマがスタートします。



枠内がタイマで、この時間が 0 になると、8 個の LED が点滅します。（LED の点滅を止めるのは、SW1~SW4 のいずれかを押してください。）

SW6 を下側に倒すと、タイマは停止します。

※SW6 を上側（WRITE 側）にして、電源投入やマイコンリセットするとプログラム書き込みモードとなり、サンプルプログラムは実行されませんので、注意してください

RTC は、最近の家電には必ず搭載されている機能かと思います。（炊飯器等でも、現在時間が表示されているものが多いかと思います。）

組み込みの世界では、マイコンと接続したセンサーの情報を記録する際、実時間のデータとセットになっていると役に立つケースがあります。

1.2. RA2L1 のクロック系

ここで、RA2L1 のクロック系に関して、触れておきたいと思います。

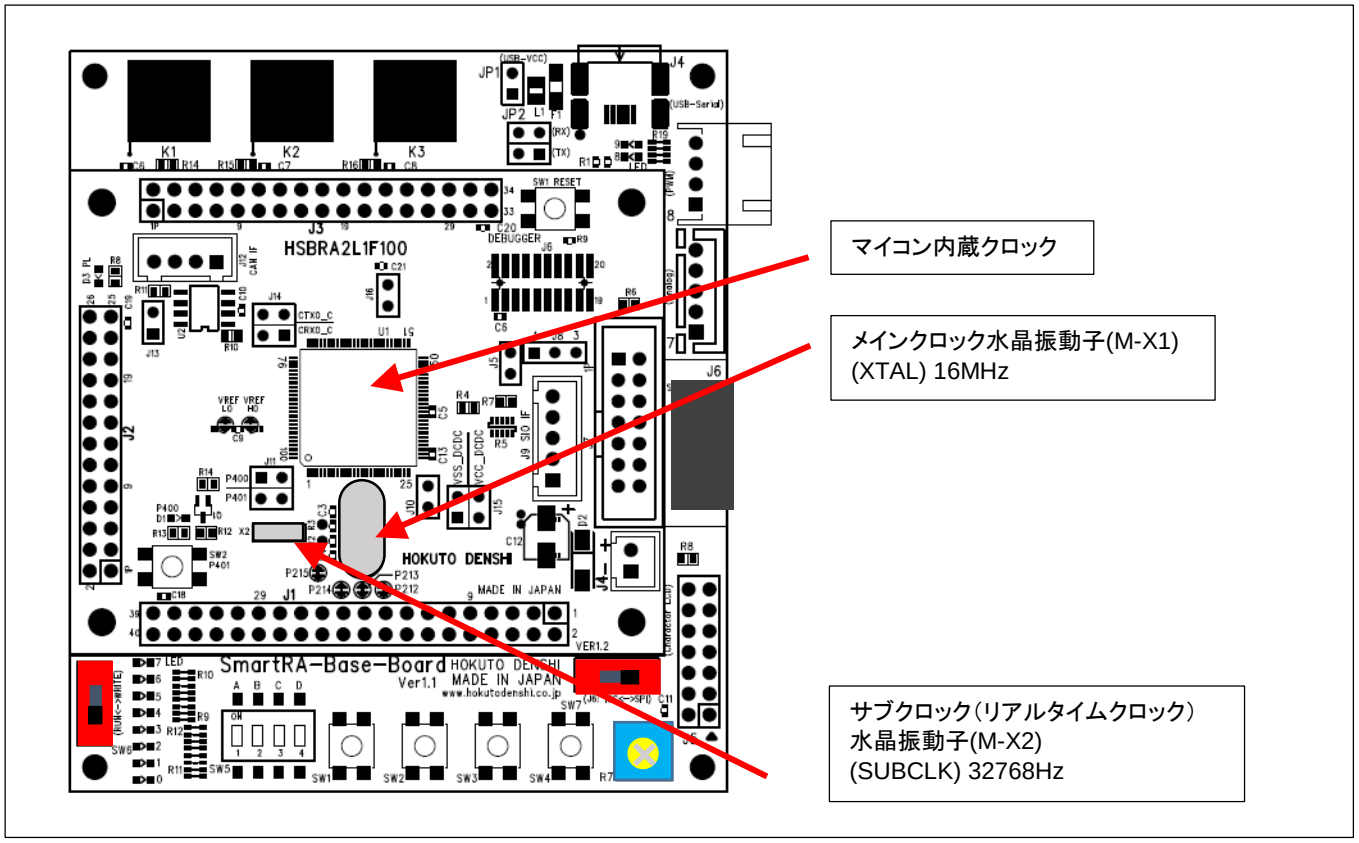


図 1-1 クロック

マイコンのクロックは、マイコン内部で生成しているクロックと、外付けの水晶振動子をベースにするものがあります。

表 1-1 クロックソース

クロック名称	周波数	発振源	備考
XTAL	16MHz	水晶振動子(M-X1)	CAN のクロックソースに使用される
LOCO	32768Hz	マイコン内蔵	SUBCLK の代替で使用される
MOCO	8MHz	マイコン内蔵	起動時デフォルトクロック
SUBCLK	32768Hz	水晶振動子(M-X2)	
HOCO	24/32/48/64MHz	マイコン内蔵	一般的には CPU クロックとして使用、周波数を選択可能
IWDT	15kHz	マイコン内蔵	ウォッチドッグタイマ向け

※クロック名称は FSP やハードウェアマニュアル記載の名称

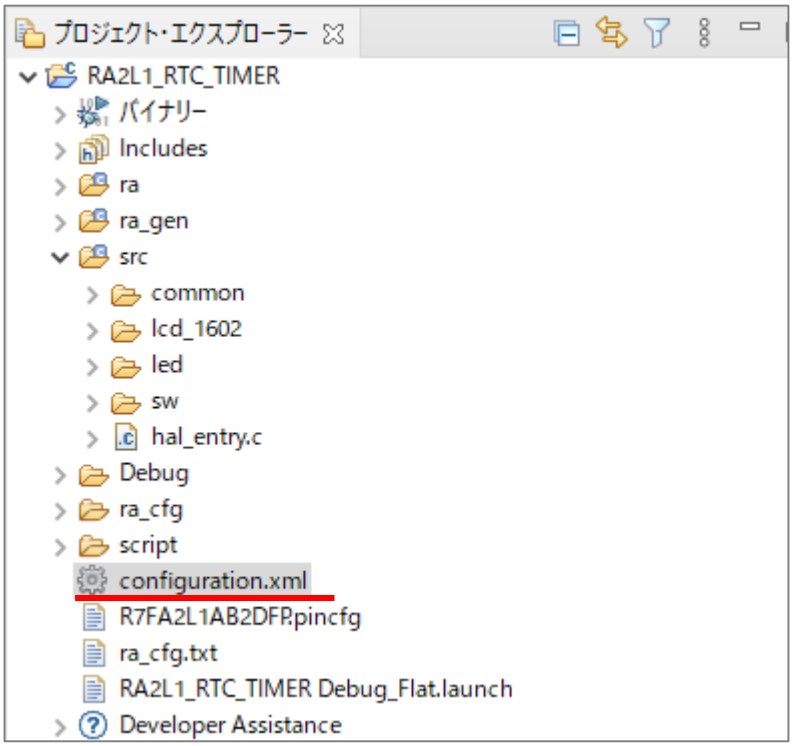
クロック源としては、表 1-1 があり、マイコンのシステムクロックは、IWDT を除く 5 種類から選択可能ですが、一般的には HOCO をシステムクロックに使います。RA2L1 は、動作周波数 48MHz(max)なので、HOCO=48MHz に設定して使用するのが、一番高速に動作させることができます。
（マイコンの消費電力を抑えたいときは、MOCO や LOCO をシステムクロックに設定することもできます。）

一般的に、水晶振動子は非常に正確な周波数で発振しますので、周波数の精度が必要な場合は、XTAL や SUBCLK を使うのが有利です。

なお、RA2L1 のマイコン内蔵クロックの精度は以下の通りです。

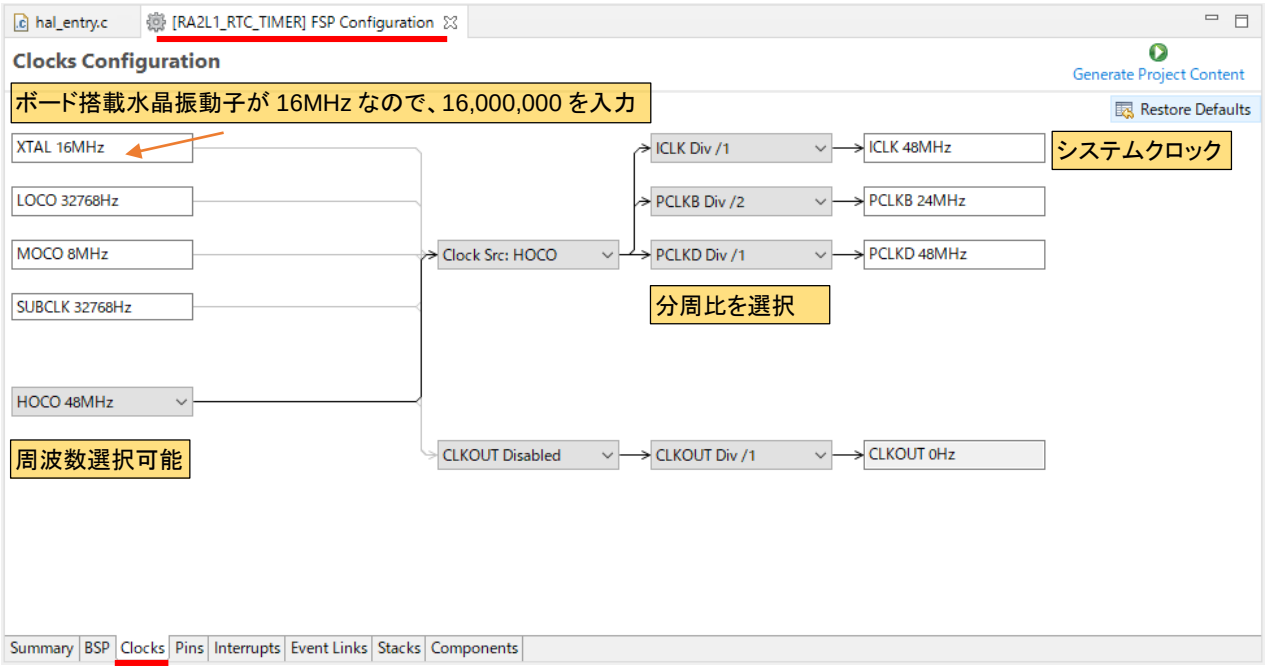
クロック名称	周波数	クロック精度
LOCO	32768Hz	±15%
MOCO	8MHz	±15%
HOCO	24/32/48/64MHz	±1%
IWDT	15kHz	±15%

HOCO は、マイコン内部で発振させているクロックとしては非常に精度が良く、±1%が保証されています。



クロック設定は、FSP の Configuration.xml 内の

1.3. FSP の設定

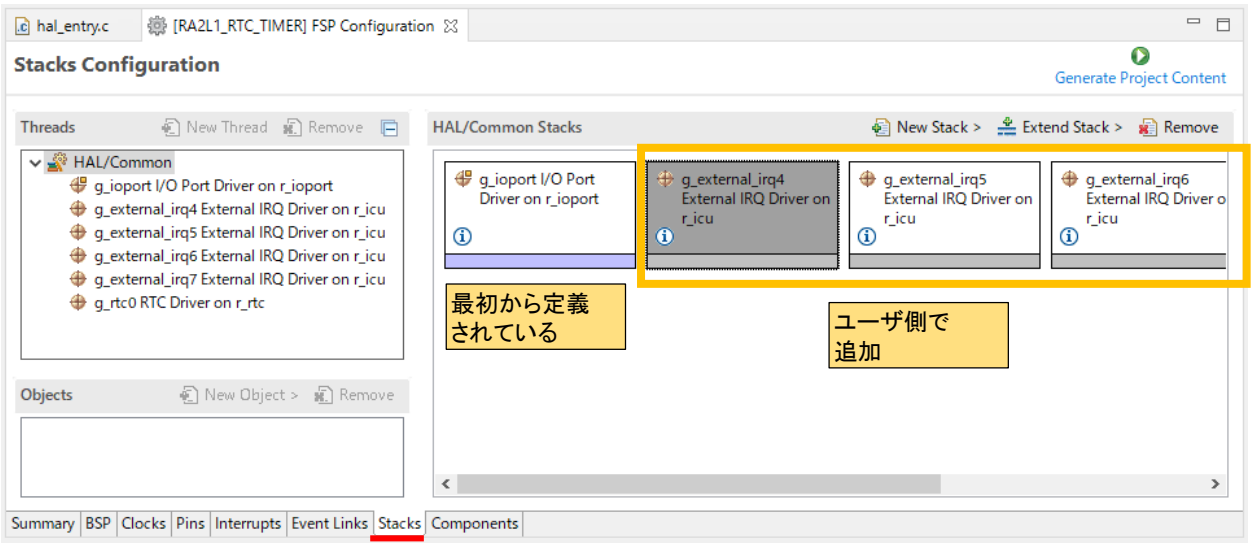


Clocks タブで設定します。本チュートリアルに限らず、基本的には上記の設定で問題ありません。

次に、FSP の設定で、RTC（リアルタイムクロック）とプッシュスイッチの割り込み設定を行っています。本チュートリアルでは、プッシュスイッチの読み取りに、外部割込み(IRQ)を使っています。

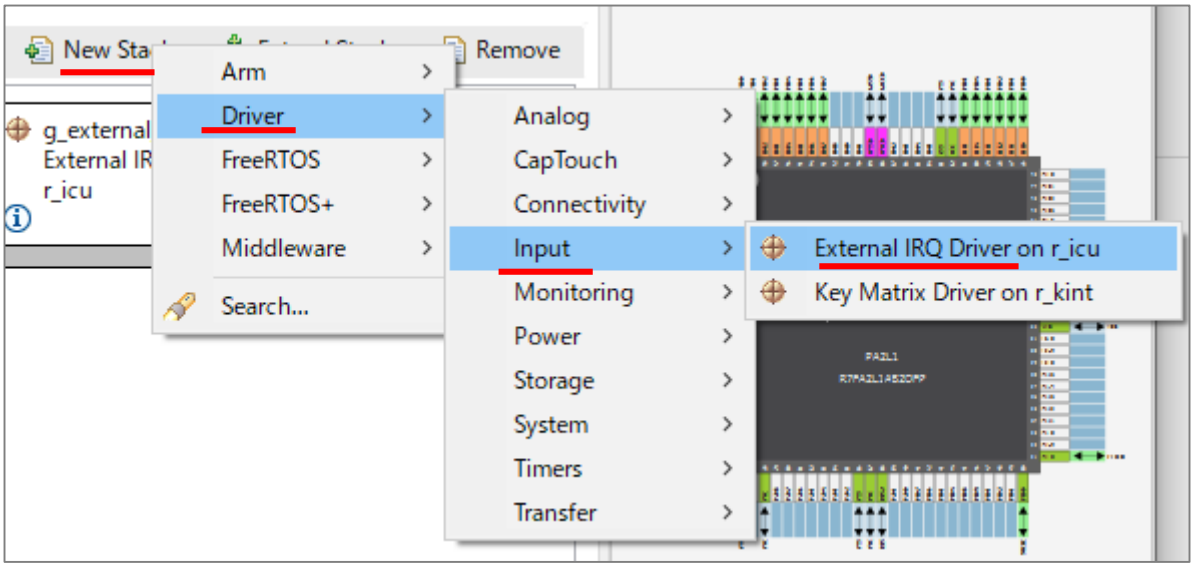
※実は、RA2L1_LCD のチュートリアルでも使用していましたが、RA2L1_LCD のチュートリアルでは説明していませんでしたので、ここで説明致します

－IRQ 設定－



FSP の Stacks タブで色々な機能を追加。

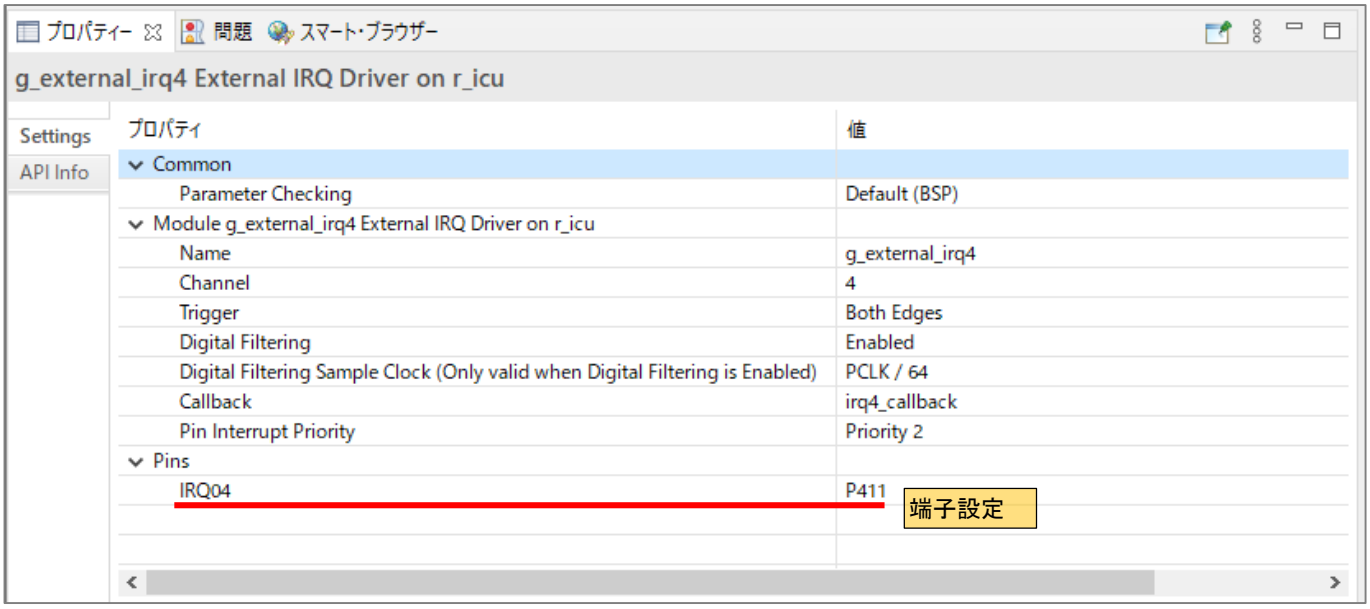
FSP 設定(FSPConfiguraion.xml)の、Stacks というタブがあり、この部分にマイコンで使う種々の機能を追加していく、というのが FSP の基本的な使い方となります。(New Stack で追加を行っていく)



例えば、New Stack – Driver – Input – External IRQ Driver

で、追加すると、IRQ(外部端子割り込み)の機能を追加できます。(本チュートリアルでは追加済みです)

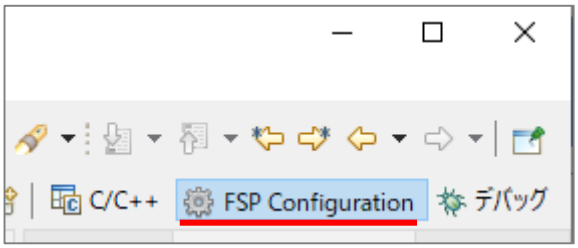
IRQ を追加すると



上記の様なプロパティ選択画面となります。

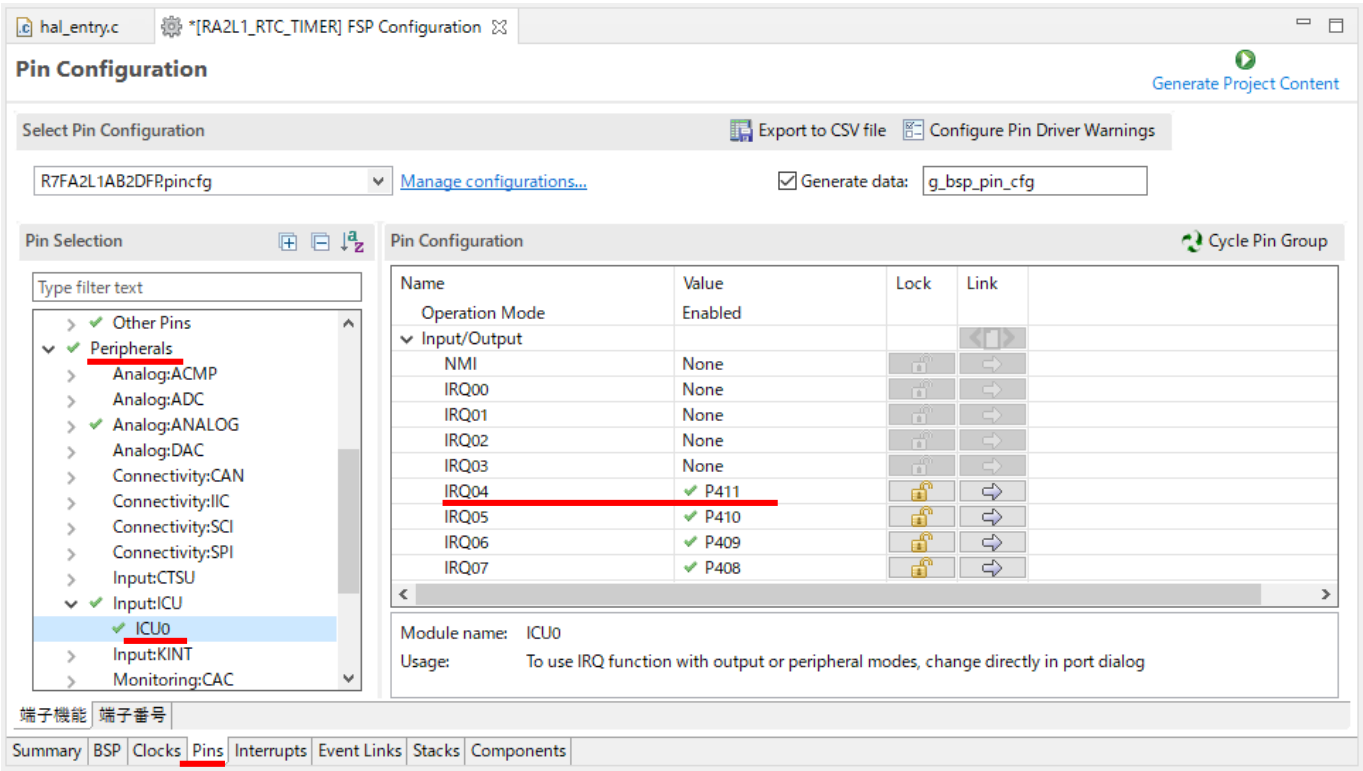
設定項目	設定値	備考
Name	g_external_irq4	名称は任意ですが、初期値 g_external_irq0 を 4 に変更しています
Channel	4	P411(B-SW4)が IRQ4 に割り当てているので、4 に変更
Trigger	Both Edges	立ち上がりと立下り両方を見る様に設定
Digital Filtering	Enabled	デジタルフィルタを使うように設定
Digital Filtering SampleClock	PCLK/64	PCLKB(=24MHz)の 1/64, 375kHz を設定
Callback	irq4_callback	設定値は任意、「機能名_callback」が推奨
Pin Interrupt Priority	Priority 2	割り込み優先度, 2 に設定

プロパティ設定画面が出てこない場合は、



表示ペインの設定で、FSP Configuration を選択してください。

－端子設定－



Pins タブの Peripherals – Input:ICU – ICU0 の IRQ04 に P411 を設定します。B-SW1~3 の P408~P410 も同様に IRQ7~IRQ5 に設定します。

プッシュスイッチと端子、IRQ の関係をまとめる以下の様になります。

スイッチ	端子	IRQ	コールバック関数
B-SW1	P408	IRQ7	irq7_callback
B-SW2	P409	IRQ6	irq6_callback
B-SW3	P410	IRQ5	irq5_callback
B-SW4	P411	IRQ4	irq4_callback

ここで、コールバック関数ですが、今 IRQ のエッジが Both Edges に設定しているので、スイッチを「押したとき」（P408 が L に変化した際）と「離したとき」（P408 が H に変化した際）に、このコールバック関数が呼ばれる事となります。コールバック関数内に、処理内容を記載しておけば、スイッチが押されたとき（もしくは離されたとき）に特定の処理を実行することができます。

コラム　ポーリングと割り込み処理

スイッチの読み取りや、マイコンの外部から発信されたデータの受信等、マイコン側からすると、いつ発生するか判らないイベントを処理する方法として、ポーリングと割り込みがあります。

・ポーリング

プログラム内で常に（定期的に）イベントが発生しているかを見に行く手法です。単純ですが、確認のために、CPU リソースが使われます。

```
while(1)    メインループ
{
    sw = sw_read();
    if((sw & 0xf) != 0xf)
    {
        [スイッチの処理]
    }

    [メインのプログラム処理]
}
```

メインループ内に
スイッチが押された
事を確認する処理が入る

・割り込み

イベントが発生した場合、CPU に通知されますので、CPU はイベントを確認する必要がありません。

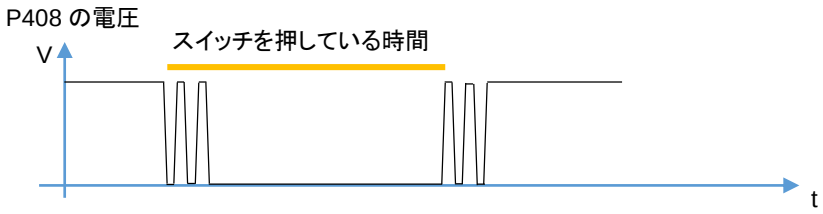
```
while(1)    メインループ
{
    [メインのプログラム処理]
}

irq4_callback()
{
    [スイッチの処理]
}
```

スイッチが押された時のみ
メインループとは別にスイッチの処理を行う

一般的には、いつ発生するか判らないイベントを処理する手法としては、割り込みを使う方がスマートです。

コラム　チャタリング

機械的なスイッチ（プッシュスイッチや DIP スイッチ等）を操作した際には、通常チャタリングが発生します。

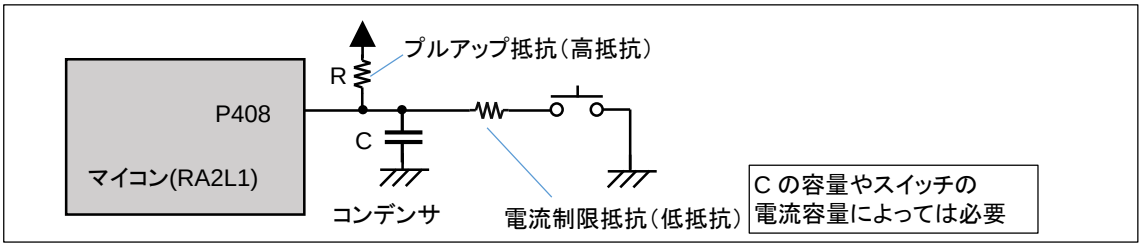


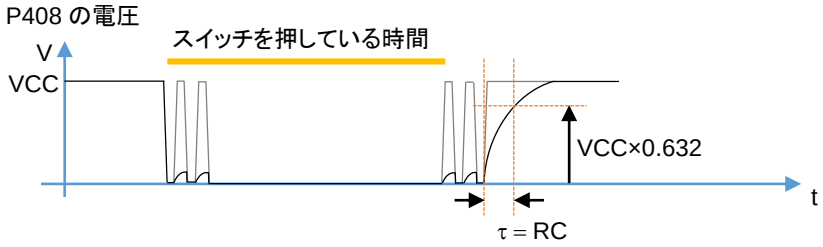
B-SW1 を押した場合、P408 の電圧は 0V（付近）。スイッチを押さない場合は、5V（=VCC 電位）となります。この時の電圧は、スイッチを押した瞬間と離した瞬間に、0-5V 間を何度か行ったり来たりする（スイッチ内部の金属の電極が跳ね返りを起こしている）事があります。

P408 の L/H をプログラムで処理すると、人は 1 回しかボタンを押していないつもりでも、プログラム上は、何回か押して離してを繰り返していると判断されるという事が起こります。（食券を 1 枚しか買っていないつもりでも、何枚も購入した事になってしまうなど。）

チャタリングを処理する方法はいくつか考えられます。

(1)ハードウェアで対策する



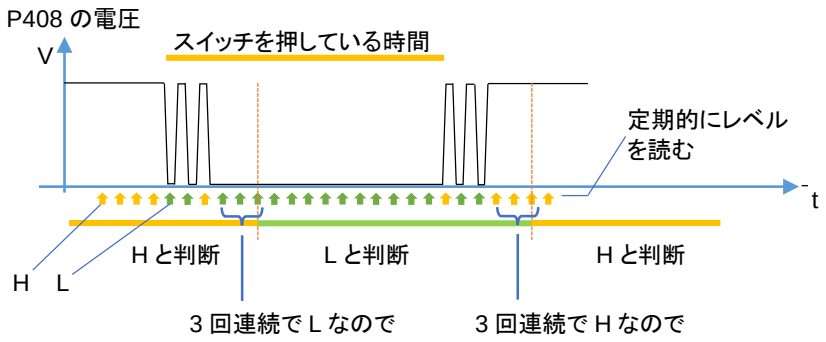


スイッチで駆動される信号線にコンデンサを入れると、コンデンサがない場合の波形（灰色）に対し、立下りはほぼ変わらず、立ち上がりが遅くなります（立ち上がり波形が鈍ります）。どのぐらい遅くなるかは、プルアップ抵抗(R)とコンデンサの容量(C)によって決まります。波形立ち上がり時、全体(VCC=5V)の 63.2%に達する時間を「時定数」といい、R と C の積となります。R=10kΩ, C=0.1μF の時、時定数(τ)は 1ms となります。

時定数が短すぎると、チャタリングをフィルタリングする事が出来ない（立ち上がりが速く、灰色の波形に近づく）ですし、極端に長いとスイッチの応答が悪くなります（スイッチを離したとき、直ぐに離したと認識されない）。

コラム　チャタリング

(2)ソフトウェアで処理する



プログラムで定期的に端子のレベル(L か H か)を取得して、何回か連続して同じレベルであれば、、スイッチ押している、離していると判断する手法です。過去の L/H の情報を保存したり、何回か連続という条件判断を入れなければならないので、多少面倒です。

(3)マイコンのデジタルフィルタを使う

これは、(2)でやっている事をマイコン内蔵の機能で行うという手法になります。L/H が切り替わったと判断する基準は「3 回連続」です。回数は選ぶことができません。定期的にレベルを読む周期は、ある程度設定が可能で、PCLKB(=24MHz)~PCLKB/64(=375kHz)です。PCLKB/64 に設定した場合、3 回連続は 8μs になりますので、8μs より狭いパルスのチャタリングを除去できるこ事となります。

本チュートリアルでは、(3)の手法を使っています。試行を行ってみると、たまにチャタリングを除去する事ができないケースもありました。そのため、サンプルプログラムでは、ソフトウェア的な不感帯(一度押したら一定時間(150ms)読み取りをキャンセルする)を設けています。

(1)は、部品点数、価格面で不利です。(2)は、マイコンのリソース(CPU 時間、多少のメモリ)を食うので、演算速度の遅い安価なマイコンではためらわれる事も考えられます。(3)は、スイッチに合わせて細かな調整(判定回数や、自由な判定周期設定)ができず、変更できるパラメータ(判定周期のみ)も限定されます。

チャタリングは、(ボタンが思った様に反応しないのは非常にストレスですので)機器の使い勝手に関わる問題ですが、意外に対応が難しかったりします。

本チュートリアルでは、プッシュスイッチ(B-SW1~B-SW4)を、ディジタルフィルタを有効化した外部割り込みで処理しており、B-SW1(P408/IRQ7)は以下の様に処理しています。

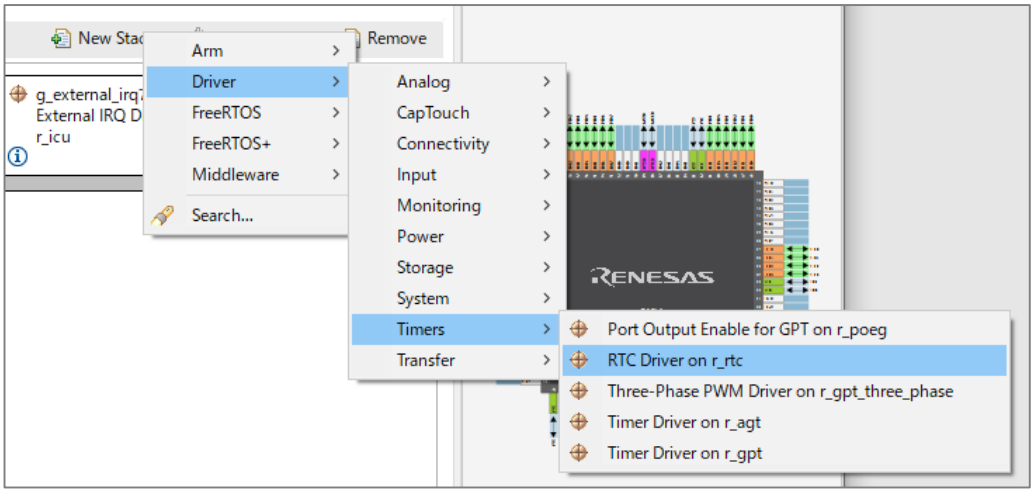
sw¥sw.c

```
void irq7_callback(external_irq_callback_args_t * p_args)
{
    (void) p_args;

    //本割り込み関数はSW1が変化した際に呼び出される

    if(R_PORT4->PIDR_b.PIDR8 == SW_OFF)
    {
        g_sw_state[SW1] = SW_OFF;          //g_sw_state[SW1] に現在のSW1の状態を保存
        g_sw_off_flag[SW1] = FLAG_SET;      //SW1 がOFFに変化したフラグを立てる
    }
    else
    {
        g_sw_state[SW1] = SW_ON;
        g_sw_on_flag[SW1] = FLAG_SET;       //SW1 がONに変化したフラグを立てる
    }
}
```

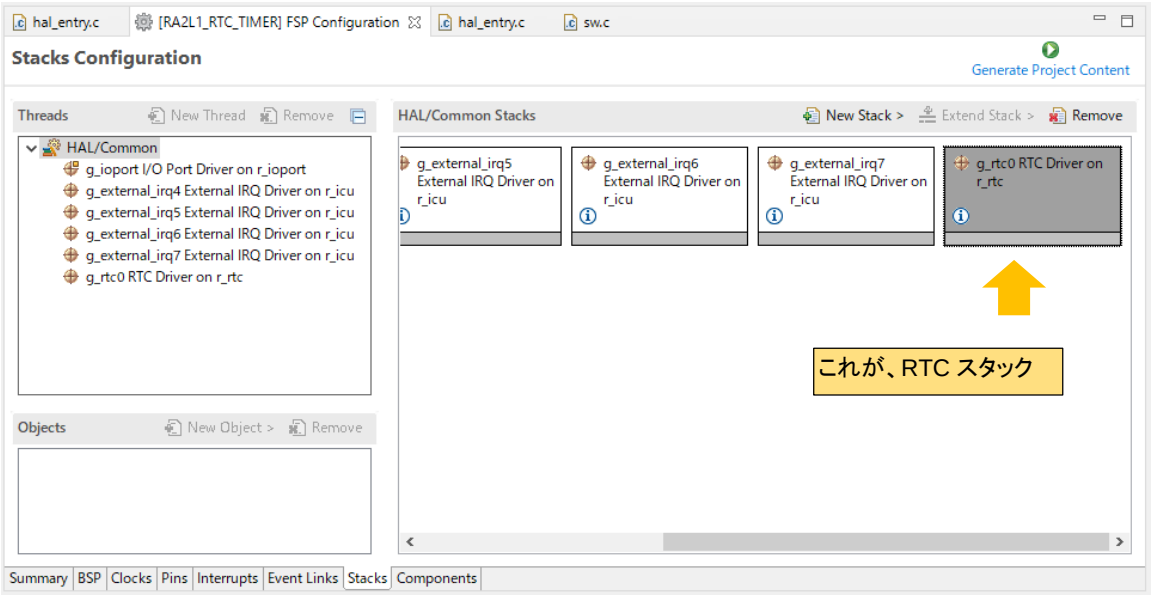
g_sw_state はグローバル変数で、割り込みが掛った場合（スイッチが押されたとき、もしくは離されたとき）にその時のスイッチの状態を保存します（=現在のスイッチの状態を保存する変数）。

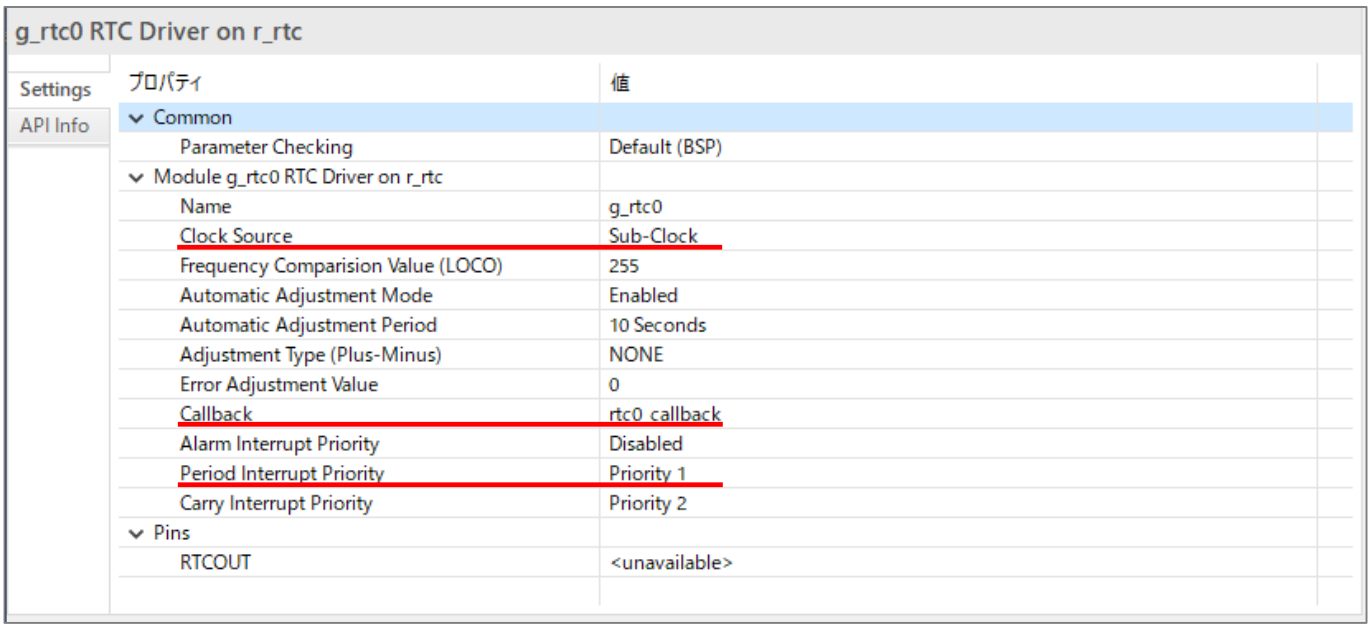
g_sw_off_flag[]は、スイッチが離されたときにフラグが立つ変数です。同様に、g_sw_on_flag は、スイッチが押されたときにフラグが立つ変数です。フラグをクリアした時から、フラグをチェックした時までに、スイッチが押されたかを判断する用途で使用します。

ここまでは、プッシュスイッチの処理に関しての説明でしたが、本チュートリアルのメインは、RTC（リアルタイムクロック）を使用したアプリケーションです。



FSP の Stacks のタブで、New Stack – Driver – Timers – RTC Driver on r_rtc を追加しています。

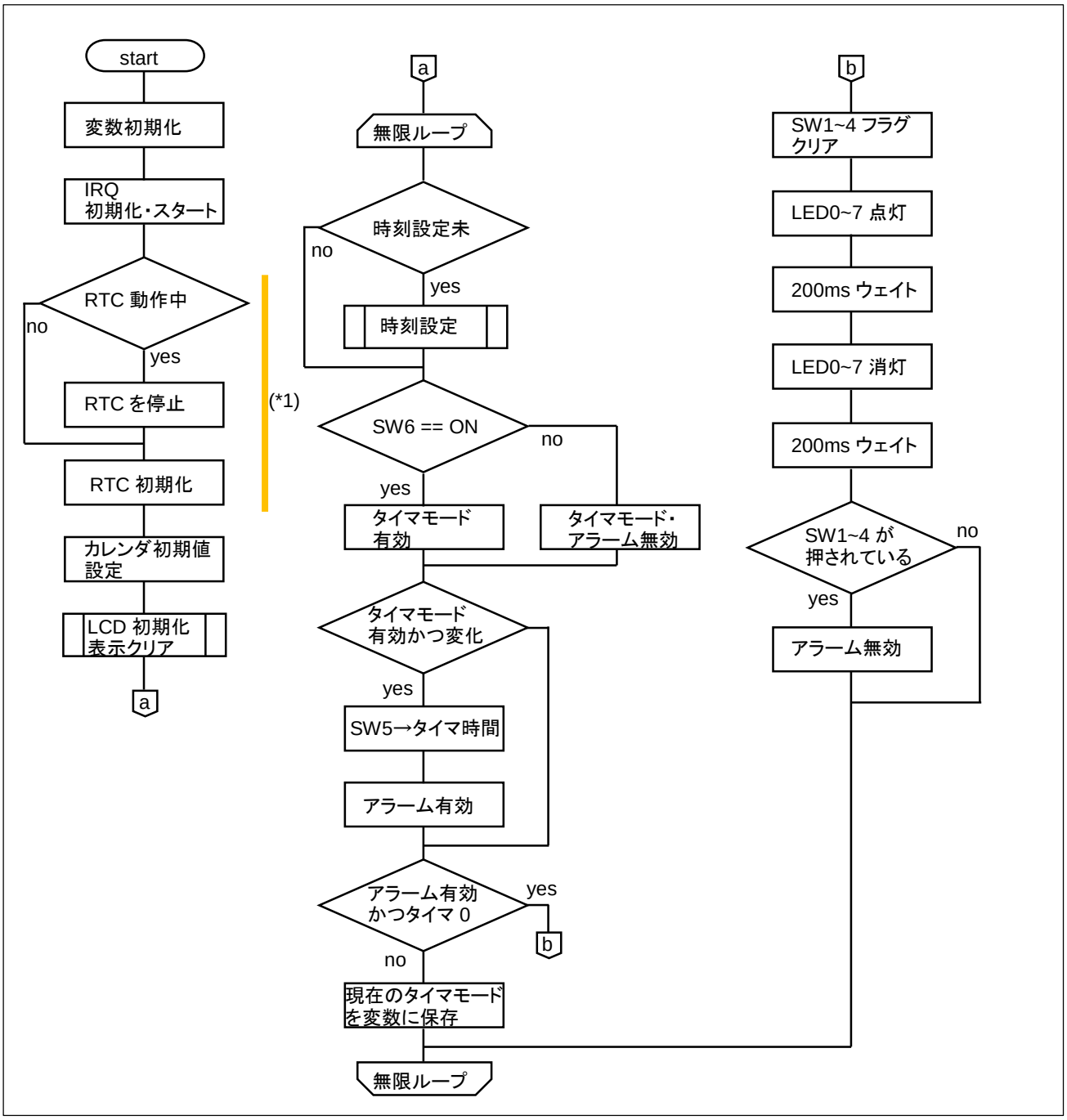




設定項目	設定値	備考
Clock Source	Sub-Clock	LOCO も選択可能であるが、精度が良い水晶振動子ベースのサブクロックを選択
Callback	rtc0_callback	RTC 割り込みを使用しているので、コールバック関数を定義
Period Interrupt Priorify	Priority 1	周期割り込みを使っているので、優先度を定義

上記設定を行っています。

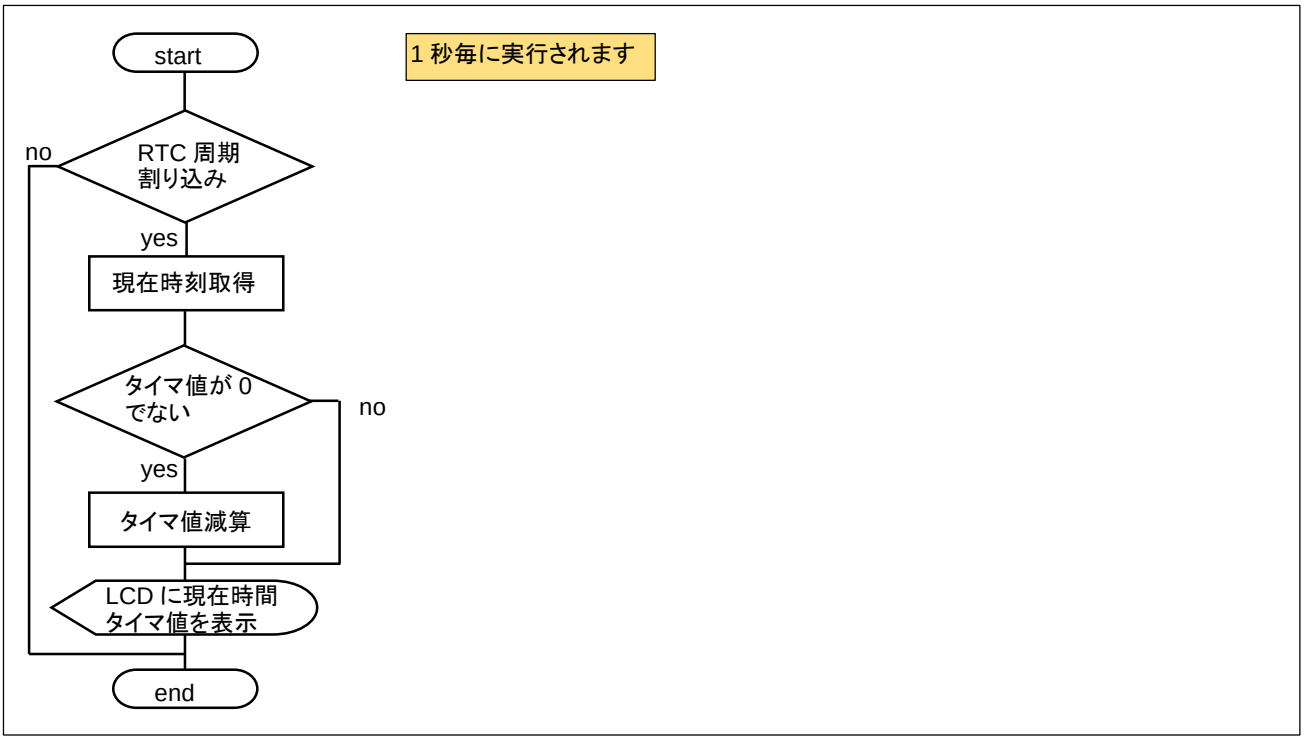
1.4. フローチャート



(*1)RTC は、リセットで初期化（停止）されないため、RTC が動作している場合は一度止めた後で初期化処理を行う

全体のフローチャートは上記の様になります。

ーRTC 割り込みフローチャートー



RTC 周期割り込み（1 秒毎）に、割り込みが入る様に設定しており、この割り込みルーチン内で LCD の表示を更新しています。（LCD の表示更新は、1 秒に 1 回行われます）

hal_entry.c（抜粋、IRQ の設定、有効化）

```
    //IRQ4
    R_ICU_ExternalIrqOpen(&g_external_irq4_ctrl, &g_external_irq4_cfg);
    R_ICU_ExternalIrqEnable(&g_external_irq4_ctrl);

    //IRQ5
    R_ICU_ExternalIrqOpen(&g_external_irq5_ctrl, &g_external_irq5_cfg);
    R_ICU_ExternalIrqEnable(&g_external_irq5_ctrl);

    //IRQ6
    R_ICU_ExternalIrqOpen(&g_external_irq6_ctrl, &g_external_irq6_cfg);
    R_ICU_ExternalIrqEnable(&g_external_irq6_ctrl);

    //IRQ7
    R_ICU_ExternalIrqOpen(&g_external_irq7_ctrl, &g_external_irq7_cfg);
    R_ICU_ExternalIrqEnable(&g_external_irq7_ctrl);
```

B-SW1~SW4 の処理で、IRQ4~7 を使っています。IRQ の設定は、FSP のスタック追加、設定で行っていますが、

・IRQ 初期化(R_ICU_ExternalIrqOpen)
・IRQ 有効化(R_ICU_ExternalIrqEnable)

の処理は、hal_entry.c 内で行っています。

hal_entry.c（抜粋、RTC の設定）

```
    rtc_time_t set_time =
    {
        //2021/1/1 00:00:00  リセット後の設定前初期値
        .tm_sec  = 0,          //秒
        .tm_min  = 0,          //分
        .tm_hour = 0,          //時
        .tm_mday = 1,          //日
        .tm_wday = 1,          //曜日は設定しない
        .tm_mon  = 1,          //月
        .tm_year = 121,        //1990年基準（2021年の場合は121）
    };

    R_RTC_Open(&g_rtc0_ctrl, &g_rtc0_cfg);

    R_RTC_CalendarTimeSet(&g_rtc0_ctrl, &set_time);

    R_RTC_PeriodicIrqRateSet(&g_rtc0_ctrl, RTC_PERIODIC_IRQ_SELECT_1_SECOND);
```

RTC でカレンダを設定する rtc_time_t 構造体を定義します。年は、1990 年を 0 とした値で取り扱います（ライブラリの仕様）。

・RTC の初期化(R_RTC_Open)
・RTC の時刻設定(R_RTC_CalendarTimeSet)
・RTC の周期割り込みのタイミングの定義(R_RTC_PeriodicIrqRateSet)

を行っています。

hal_entry.c（抜粋、RTC の時間取得）

```
    rtc_time_t get_time;

    R_RTC_CalendarTimeGet(&g_rtc0_ctrl, &get_time);
```

・RTC の現在時刻の取得(R_RTC_CalendarTimeGet)

rtc_time_t 構造体に、現在時刻の結果が格納されます。

例、get_time.tm_mon に月データが入ります。

コラム　BCD とは

RTC では、時間等のデータがレジスタに格納されていますが、格納値は BCD です。

BCD とは、BinaryCodedDecimal の略で、2 進化 10 進符号の事です。

日本語にしても意味不明ですが、分データを保持しているレジスタは、

R_RTC->RMINCNT

です。現在の時間が 35 分の時レジスタ値は

0x35 （16 進数で 35）

となります。

普通は、10 進数で 35 の場合、16 進数だと、0x23 となりますので、マイコン内部では 0x23 となっていても良い気がするのですが、マイコン内部での取り扱いは、0x35 です。

16 進数をそのまま直読できるので、LCD に分を表示させる際 hex 表示で、

lcd_write_hex(R_RTC->RMINCNT);

とやれば良いです。時刻を表示する際に、16 進→10 進の変換等考えなくても良いという事となります。

イメージ的には、表示処理が楽という事でしょうか。

その反面、現在の時間を 35 分に設定したいという場合、

R_RTC->RMINCNT = 35;

とやると、現在時刻は 23 分になってしまいます。値をセットする場合は、多少面倒です。

35 分に設定する場合、

R_RTC->RMINCNT = 0x35;
R_RTC->RMINCNT = 53;

のどちらかとなります。

コラム　BCD とは

FSP の API 関数を使う場合は、

```
rtc_time_t set_time;

R_RTC_CalendarTimeGet(&g_rtc0_ctrl, &set_time);   //現在時刻を set_time に格納
set_time.tm_min = 35;   //分の情報を書き換え（35 分を設定）
R_RTC_CalendarTimeSet(&g_rtc0_ctrl, &set_time);   //新しい時間を設定
```

の様に、分のデータを 10 進数で扱います。

RTC が内部で BCD を使っているという事を意識する必要はありません。

マイコン内蔵の RTC 以外でも、独立した市販の RTC モジュールでも時間のデータを BCD で取り扱うケースは多いですので、RTC のプログラムを作成する場合は、データが通常の 16 進数か、BCD かを意識するようにしてください。

取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2021.5.11	—	初版発行

お問合せ窓口

最新情報については弊社ホームページをご活用ください。

ご不明点は弊社サポート窓口までお問合せください。

株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7

TEL 011-640-8800　FAX 011-640-8801

e-mail：support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用)

URL:http://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RA マイコン搭載
HSB シリーズマイコンボード 評価キット

SmartRA 学習キット
チュートリアル 4

株式会社 北斗電子

 2021 年 5 月 11 日改訂 REV.1.0.0.0 (210511)