Tutorial 1A

創造設計第二 TA：田原 康平

平成 21 年度 10 月 7 日

1. はじめに

Tutorial（試作検討） 1A では創造設計第 2 で使用するマイコンの開発環境になれることを主な目的とする．また，全 Tutorial を通じて，マイコンボードは，VStone 社製マイコンボード (VS-WRC003) を使用することを前提としている．

今回の創造設計では使用するマイコンの指定はされていないが，上記以外のマイコン及びマイコンボードを使用する場合は，TA 側でサポートすることは困難であるため，他のマイコンを使用する際は完全に自己責任で行うこと．

使用するボードセットはメカトロニクスラボで使用したものと同一のものなので，今回の Tutorial ではメカトロニクスラボの復習が主な内容となっている．実際に取り組む内容としては以下のようになっている．

- 開発環境の理解
- ポート入出力
- モニタプログラムを用いたデバッグ
- RAM 化
- 割り込み（タイマ割り込みと外部割り込み）
- ROM 化

まず，開発環境に慣れるためにサンプルプログラムを実行し，動作を確認する．次に割り込みの方法（タイマ割り込みと外部割り込み）について確認する．最後に Tutorial 1B で作成したメカニカルスイッチのチャタリング防止回路を動作確認するために，スイッチの信号を外部入力とする割り込みプログラムを作成し，マイクロコントローラのフラッシュメモリに書き込んで ROM 化を行う．

警告	各グループのメンバー全員が内容を理解できるように，確認し合って進めること．

警告	Tutorial のプログラムや資料は必要に応じて創造設計第二のホームページ (`http://www.cm.ctrl.titech.ac.jp/ss2-2010/technical.html`) からダウンロードすること．今回は ss2shisaku01.zip をダウンロードし，適当な場所に解凍しておくこと．本 zip ファイルには,MONITOR.MOT,sample フォルダ, 参考資料フォルダが格納されている．sample フォルダに本 Tutorial で用いるサンプルプログラムが入っているので確認すること．参考資料フォルダには，公開されている技術資料などの pdf を格納している．

2. 開発環境

本年度の創造設計から，H8 のマイコン使用するため開発環境として，ルネサステクノロジ社製の統合開発環境である High-performance Embedded Workshop (HEW) と，デバッガである HTerm を使用します．また，ROM への書き込みには Flash Development Toolkit(FDT) を使用する．

HEW,HTerm,FDT に関する詳細なマニュアルはルネサステクノロジのホームページならびに，メカトロニクスラボ用の加嶋先生のページ (`http://www.cyb.mei.titech.ac.jp/~kashima/ml/ml.html`，学内専用) を参考にすること．

注意	プロジェクトのワーキングディレクトリとして, 全角文字や空白を入れてはいけない．

警告	T ドライブは学生に書き込み権限はありません．昨年の創造設計のデータが残っていので，注意すること（昨年と今年では使用するマイコンが異なる）．

Table 1: 入出力ポート数

DC モータ	2
RC サーボ	0 (VS-WRC004 という拡張ボードを使用することで使用可能)
アナログ入力	2 (コネクタの増設で最大 4)
ブザー	1
LED	1 (1 つは電源用 LED のため，基本的には使用しない)

警告 HEW のデフォルトのワーキングディレクトリは C ドライブになっている．しかし，H3-310 のシステムの構成上 C ドライブ上に新しく生成されたファイルはシステムをシャットダウンすると消去されてしまう．そのため，作業は Z ドライブ上か，配布された USB メモリ上で行うようにする

注意 新規プロジェクトの作成方法や，サンプルプログラムのコンパイル，デバッガ実行のための ROM をモニタプログラムに戻す方法，デバッガの起動方法はメカトロニクスラボ資料を参照すること．ただし，HEW,HTerm,FDT でマイコンと接続する際は COM3 を選択すること．

2.1 入出力ポート

VS-WRC003 には多くの入出力ポートが用意されている．デフォルトで用意されているポートの数は以下のとおりである．

しかし， VS-WRC003 は配線図が公開されているため，入出力ポートを増やすことができる．

今回は後にメカニカルスイッチを用いるため，ボード上の集合 IO(CN10/EXT) を利用できるようにする．

具体的にはピンをハンダ付けするだけである．(この作業はメカトロニクスラボで行っているはずである．もし，ピンがたっていないのであれば，TA に申し出ること．Tutorial 1 中のみに限って，ピンを提供するので，創造工房で半田付けを行う.)

入出力ポートの詳細（ボート番号やレジストリ名）はメカトロニクスラボの資料やルネサスのサポートページ `http://www.vstone.co.jp/top/products/robot/beauto/cdownload.html` を参照すること．

本 Tutorial で用いるのは，LED 用の P64/FTOPA1 と ブザー用の P76/TMOV のみである．

3. LED1 を点灯させる

本節では LED1 を点灯させることを通じて，入出力ポートの設定方法を復習すると共に，開発環境に慣れることを目的とする．

3.1 モニタプログラムの書き込み

デフォルトの状態で配布されている VS-WRC003 には LCD(液晶パネル) が取り付けられていないため，本 Tutorial ではプログラムのデバッグをモニタプログラムを使用して行う．

サンプルプログラムの入った zip ファイルを解凍すると MONITOR.MOT が入っているので，FDT を用いてマイコンの ROM 領域にモニタプログラムを書き込む．

書き込みの手順は `http://www.cyb.mei.titech.ac.jp/~kashima/ml/doc/HEW_samplemanual_v0.2.pdf` を参考にすること．

まず，PC とマイコンをシリアルケーブルで接続し，FDT を起動する．その際，デバイスマネージャでマイコンが COM3 で接続されていることを確認する．

以下の情報と上記 pdf を参考にしながら，モニタプログラムを書き込む．

- デバイスとカーネルの選択 Full Name ： H8/36064F
- デバイス設定 入力クロック： 14.7456[MHz]

3.2 モニタプログラムの動作確認

次に HTerm を使用して，モニタプログラムが正常に書き込まれているか確認する．

PC とマイコンが電源が入っている状態で接続されている時に，HTerm を起動すると，自動的にマイコンと HTerm が接続され，白いウィンドウが表示される．表示されなかったり，エラーが出た際には， COM の設定や HTerm の設定を再度確認する．

白いウィンドウが表示されている状態で，ボード上の電源の隣のボタンを押すと，以下のメッセージが表示される．

```
H8/36064 Series Normal Mode Monitor Ver. 2.0B
Copyright (C) 2003 Renesas Technology Corp.
```

このメッセージが表示されれば，モニタプログラムは正常に作動している．また，同時に HTerm とマイコンが正常に接続されていることもこのメッセージにより確認できる．

3.3 サンプルプログラム

本節ではサンプルプログラムを用いて，HEW，HTerm の使い方になれる．

本節の目的はサンプルプログラムを HEW でコンパイルし，HTerm で RAM 領域に書き込み，マイコンの LED1 を点灯させることを目的とする．

3.3.1 新規プロジェクトの作成

HEW を起動すると，**Fig.** 1 に示すダイアログが表示される．[新規プロジェクトワークスペースの作成] を選択し [OK] をクリックし．

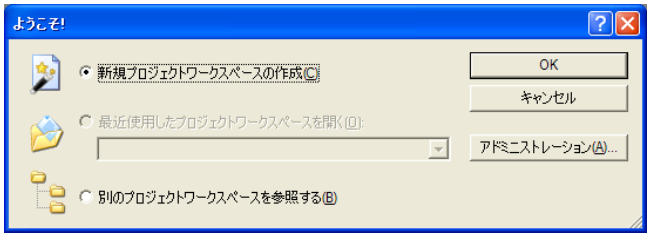

Fig. 1: 起動時のダイアログ

次に，ワークスペース名などを設定するダイアログが表示される (**Fig.** 2)．ワークスペース名とディレクトリを設定し [OK] をクリック．プロジェクト名はワークスペース名で自動的に補完されます．また，他の設定はそのままで構わない．

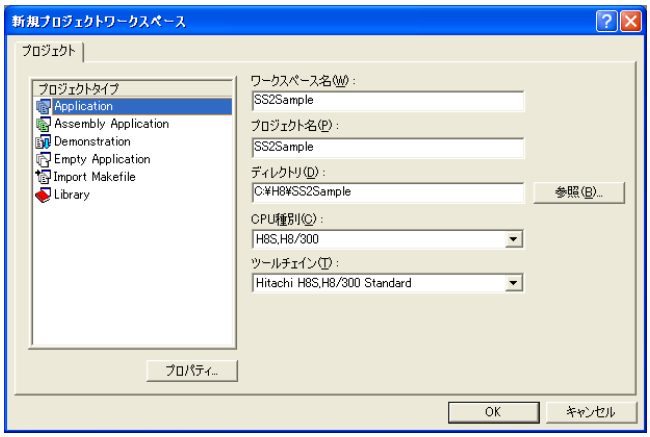

Fig. 2: ワークスペース名の設定

次のステップでは CPU の種類を設定する (**Fig.** 3)．CPU シリーズは [300H]，CPU タイプは [36064] に設定する．本来はこれ以降も多くの設定項目があるが，今回はそれらについてはすべてデフォルトのままにしておく．．[完了] をクリックすると設定完了．

最後にプロジェクトの概要が表示されますので，CPU シリーズと CPU タイプについて確認し [OK] をクリック (**Fig.** 4)．

これで，プロジェクトが作成されている．同時に HEW によっていくつかのファイルが自動生成される．

しかし，デフォルトで生成されるプロジェクトのままでは，モニタプログラムを用いたデバッグを行うことができない[1]ので，設定を変更する．

右上のビルドオプションが [Dubug] になっていることを確認した上で，メニューバーの [ビルド] → [H8S,H8/300 Standard Toolchain ...] を選択．

[最適化リンカ] タブでカテゴリを [セクション] に選択し，Table 2 のように設定する．

[1] モニタプログラムを ROM 領域に書き込んでいるため，コンパイルして生成される実行ファイルは RAM 領域に書き込まれる．しかし，モニタプログラムによって RAM 領域で動作させることができるメモリアドレスが規定されているため，アドレスを再設定する必要がある．

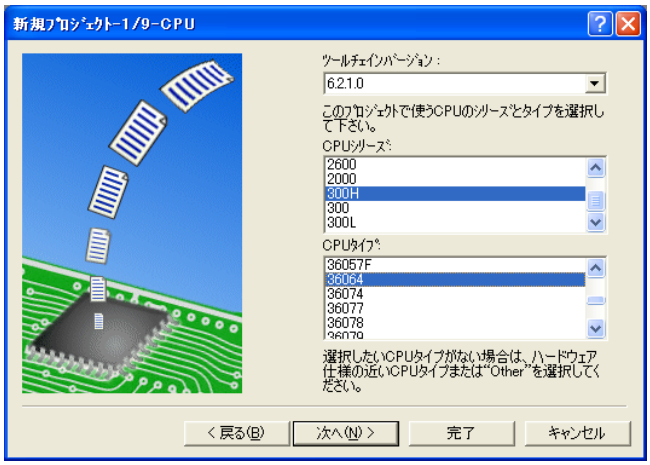

Fig. 3: CPU の設定

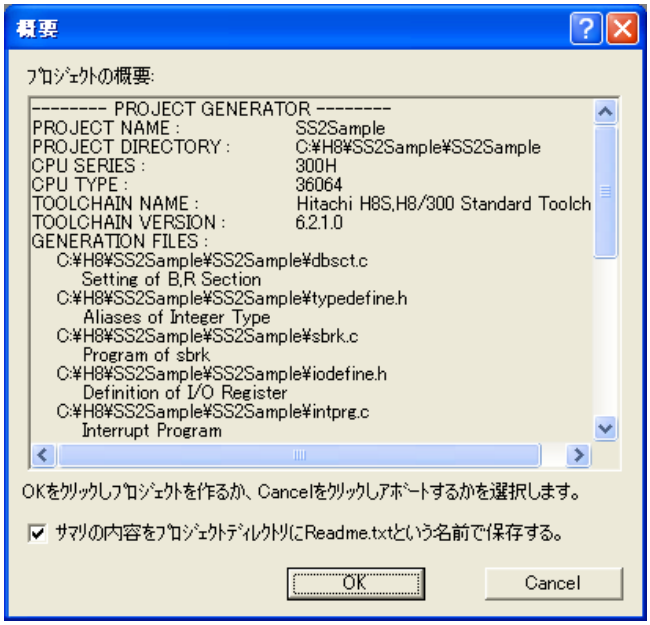

Fig. 4: 概要の確認

Table 2: Address - Section の設定

Address	Section
0x0000F840	CV0
0x0000F890	PResetPRG,PIntPRG,P,C$DSEC,C$BSEC,D
0x0000FB80	B,R
0x0000FE80	S

これで，モニタプログラムでデバッグを行うことができるプロジェクトを作成することができた．

3.3.2 サンプルプログラム

前節で作成されたプロジェクトで，HEW が自動生成したファイルを Tutorial 1A を行うために，サンプルプログラムに書き換える．

ダウンロードした shisaku01.zip の sample フォルダの中の sample.c,resetprg.c,intprg.c,iodefine.h をコピーして，作成したプロジェクトのファイルと置き換える．

HEW を起動したまま，置き換えを行うと，自動的に書き換えたファイルが読み込まれているはずである．

これで，本 Tutorial を行う準備が整ったことになる．

3.3.3 LED を点灯させるプログラムの作成

サンプルプロジェクトをそのままコンパイルしても，何も実行されない．そこで，本節では sample.c 内の main 関数を書き換えて，LED1 を点灯させる．

課題 1a-1：LED1 の点灯

1. サンプルコード内の課題 1a-1 の部分を変更し，P64 を出力ポートに設定する．(BIT0 から数えて 5 番目の BIT4 を 1 にする．16 進数表記を用いる．)

sample.c の変更ができたら，コンパイルを行う．ビルドには Debug ビルドと Release ビルドがあるが，Debug ビルドに設定されいることを確認した後に,「すべてをビルド」を行う．

0 Errors となれば，コンパイルが完了である．同時に，実行ファイルである，sample.abs が Debug フォルダに生成されている．

3.3.4 RAM 領域への書き込みと実行

HTerm を用いて作成された sample.abs をマイコンの RAM 領域に書き込む．

3.2 節が正常に完了している状態で，[コマンド] → [Load] もしくは F9 を押して sample.abs を選択する．

ロードが完了すると「ソースプログラムを表示しますか？」と聞かれるので「はい」をクリック．

これで実行ファイルを RAM 領域に書き込むことができた．

この状態で F5 を押すことで，プログラムが実行される．

LED1 が点灯すれば，完了である．

注意 この状態ではプログラムに終了コードが設定されていないため，ボードの電源を切ることで強制終了する．その後に，電源を入れて再接続を行う．

4. ポーリング

一定の間隔で繰り返しポートの値を読み込むことをポーリングという．入力ポートの電圧 (H/L) をプログラム中で変数の値 (0/1) として扱う[2]．前回の値を保持しておき今回の値と比較すれば，入力の変化 (エッジ) を検出することができる．次節でタイマ割り込みを用いた．ポーリングによるリアルタイム制御プログラムについて説明する．

5. 割り込み

割り込みとは，実行中の処理を一時的に中断して，他の処理（割り込み処理）を行うことをいう．また，割り込み処理が終了した後には元の実行していた処理を再開させることができる．割り込み処理を行う主な利点としては次のようなものがある．

- スイッチやセンサ入力などの外部入力によって行う処理を確実に実行できる．(外部割り込み)
- 定期的な処理を確実に実行できる．(タイマ割り込み)
- プログラム全体の処理効率が上がる．

[2]電圧とレジスタ値の関係は正論理である (つまりレジスタを 1 にすると電圧が出る)．ポート入力は負論理となっている

5.1 タイマ割り込み

タイマによるカウント値を利用して，一定周期で割り込みを行うことをタイマ割り込みという．

本ボードには割り込み用のタイマとして，タイマ B1, タイマ V, タイマ Z0, タイマ Z1 が用意されている．各タイマの詳細はメカトロニクスラボの資料を参照すること．本 Tutorial ではタイマ B1 を用いて 5[ms] 毎のタイマ割り込みを行う．

本節の目標はタイマ割り込みを用いて LED1 を 0.5 秒間隔で点滅させることである．

5.1.1 タイマ割り込みの設定

本節では，タイマ B1 を用いたタイマ割り込みの具体的な設定を述べる．

```
void main(void){
  //すべての割り込みを禁止
  set_imask_ccr(1);
  // TB1 : 5 [ms] ごとに割り込みを入れる
  TB1.TMB1.BIT.RLD = 1;
  TB1.TMB1.BIT.CKS = 2;
  TB1.TLB1 = 112;
  IENR2.BIT.IENTB1 = 1;
  IRR2.BIT.IRRTB1 = 0;
  // すべての割り込みを許可
  set_imask_ccr(0);
}

// 割り込み関数の記述
__interrupt(vect=29) void INT_TimerB1(void)
{
  IRR2.BIT.IRRTB1 = 0;

  割り込み処理の内容
}
```

上記のように割り込みを使用する場合は，**一旦全ての割り込みを禁止した後に，割り込みの設定をした後に割り込みの許可を与える必要がある．**また，割り込みに使用する関数の宣言は別途行う必要がある．サンプルプログラムでは，resetprg.c で INT_TimerB1(void) の宣言を行っている．ただし，宣言を行った関数は全ての関数に対して処理を記述する必要があるので，注意する．何もしない場合は

```
__interrupt(vect=29)voidINT_TimerB1(void){}
```

のように関数の内容を空にすれば良い．タイマ割り込み以外にもイベント割り込みなどに使用する割り込み関数も存在するので注意する．

課題 1a-2：LED1 の点滅

1. サンプルコード内の課題 1a-2 の部分のコメントアウトを外す．全部で 4 箇所です．
2. サンプル中の ????? を変更して，0.5 秒間隔で LED1 を点滅させる．
   LED1flag という 0,1 の値をとるフラグが用意されています．main 関数内に LED1flag の値によって，LED1 を点灯，消灯させる条件分岐が記述されています．

この課題によって，電源スイッチの横のボタンでプログラムを終了させることができるようになっている．原理は，ボタンのレジスタの値 IO.PDR5.BIT.B5 の値によって，reset というフラグを切り替え，reset = 1 のときに，無限ループを break によって抜け出るようにしている．

5.2 イベント割り込み

イベント割り込みとは入力ポートの電圧が変化したときに，実行中の処理を一旦中断し，特定の関数 (割り込みハンドラ) を実行する．割り込みハンドラの終了後，元の処理が再開される．例えばメカニカルスイッチなどの外部パルスの立ち上がり（または立ち下がり）を検出し，割り込み処理を実行することができる．

本節では以下の課題を通して，メカニカルスイッチによって，ブザーの ON/OFF をおこなうプログラムを作成することを目標とする．

課題 1a-3：ブザーの ON/OFF を切り替える

1. サンプルプログラムから課題 1a-3 に関連する部分のコメントアウトを外す（3 箇所）
2. sensor という変数の値により，ブザーの ON/OFF を行うコードを main 関数のループの中に記述する．
3. イベントの発生により，sensor の値が反転するようにイベント割り込みの処理を記述する．

以上のコードが書けたらとりあえずコンパイルをして，実行させる．
その際に，イベントとしてメカニカルスイッチを使用する．しかし，メカニカルスイッチは，チャタリング防止回路作成のために使用していはずなので，22[kΩ] の抵抗を使用して，スイッチ回路を作成する．(抵抗をクリップではさめば良い)
その際に Tutorial 1B の回路図を参考にすること．抵抗の取り付け方により，スイッチ ON で電圧が上がるのか下がるのかが変わるので，設定したい方を選択する．
デフォルトは立ち下がりをスイッチ ON として検出するようにしている．
スイッチ回路が作成できたら，回路に Vcc,GND,Signal を接続する．どの端子に接続するかは `http://www.vstone.co.jp/top/products/robot/beauto/cdownload.html` を参考にする．ただし，Signal は P50 ポートに接続すること．

Vcc と GND は絶対にショートさせないこと．最悪の場合，過電流によりマイコンが焼けて壊れます．

スイッチの ON/OFF を切り替えて，ブザーが ON/OFF されることを確認する．

5.2.1 チャタリング防止

前節の状態で，プログラムを実行し，スイッチを ON/OFF すると，チャタリングという現象が起きたはずである．
チャタリングとは ON/OFF が切り替わる瞬間に，高周波で ON/OFF が切り替わってしまう現象である．
本節ではチャタリングの防止を行い，前節の目標を達成することを目標とする．
チャタリング防止を行う方法として，ソフトウェア上で実現する方法と，ハードウェア（回路）上で実現する方法がある．回路上でチャタリング防止を行う方法は Tutorial 1B で解説するため，ここではソフトウェア上で実現する方法を紹介する．
チャタリングは ON/OFF の切り替わりの瞬間のみで生じるので，最初の立ち上がり（もしくは立ち下がり）を検出したら，ある一定時間（チャタリングが落ち着くまでの時間），ブザーを ON/OFF するフラグを変更しないようにすれば良い．

課題 1a-4：チャタリング防止

1. 最初の立ち上がり（もしくは立ち下がり）が発生した時刻を記録し，一定時間 sensor フラグの値が変更されないように課題 1a-3 で作成したプログラムを変更する．ここで，時間を記録する変数として，sensorONtime という変数が用意してある．

コンパイルして実行し，プログラムが正常に動作すれば，Tutorial 1A は終了です．お疲れ様でした．
ここまでの作業が完了すると，マイコンは，以下のような動作を行うことができるようになっているはずです．

- 0.5 秒おきに LED1 が点滅する．
- メカニカルスイッチにより，ブザーを ON/OFF できる．
- 電源スイッチ横のボタンにより，プログラムを終了できる．

6. プログラムの ROM 化

本 Tutorial ではデバッグにモニタプログラムを用いて，RAM 領域に実行ファイルを書き込むことによってデバッグを行った．
しかし，H8 マイコンの RAM 領域は 2k バイトと少ないので，創造設計を通じてプログラムを作成していくと，おそらく RAM 不足に陥る．
そこで，本節では，モニタプログラムを用いずに，プログラムの実行ファイルを直接 ROM 領域に書き込んで実行する方法を紹介する．

HEW でプロジェクトを開くと左上に [Debug] と表示されていると思う．これが，ビルドオプションを指定するもので，別に [Release] というオプションが用意されている．このオプションを指定することで，プログラムを ROM 領域に書き込むための実行ファイルがビルドをすることで生成される．実行ファイルの拡張子は [.mot] である．

このとき，ビルドを行うといくつかのエラーメッセージが表示される場合がある．これは，ビルドオプションの変更により，[Release] では 3.3.1 節で設定したオプションがまだ設定されていないためである．

そこで，Table 2 を参考にして設定を行う．ただし，今回は ROM 領域に書き込むため，Address は変更する必要はない．

この.mot ファイルを モニタプログラムを書き込む際に用いた FDT を用いて，ROM 領域に書き込む．手順はモニタプログラムを書き込む手順と同じなので，省略する．

書き込みに成功したら，ケーブを外し，電源を再投入すると，プログラムが実行される．

7. 個人 PC での開発環境の構築

H8 マイコンの開発環境は I3-310 では整っています．しかし，個人所有の PC で開発を行ないたいという人が多いと思います．

開発に必要な HEW,HTerm,FDT のインストール方法は前述の加嶋先生のホームページからたどることができるのでそこからインストールを行います．

I3-310 以外での環境の動作確認としては，TA の田原によって，Windows 7 Ultimate (64bit) で開発が行えることは確認しています．（ただし，個人的な感想は COM の設定が不安定でした．）

個人の PC で開発を行うことは完全に自己責任で行ってください．

また，HEW はバージョンごとの互換性がほぼありません．I3-310 にインストーされているバージョンは古いので，同じバージョンのソフトウェアをインストールすることを推奨する．