H8マイコン開発環境構築の手引き

筆者　12S 奥野

1　前書き

我らのサークルにおいて、ソフトウェア関連の技術は優れていれどもハードウェア方面に関してはわりと近い存在である M ○ A や工○研究部に大きく遅れを取っている。M 科の友人から浴びせられる『あのサークルって何やってるの？』『江ノ島まで来て部屋にこもる生活って……』といった厳しい文言を受け、X68 も「俺らソフトだけじゃねえよ、見ろよこの肉体（BODY）？ワイルドだろぉ？」なオーラを出して他のサークルを威圧する力が必要だと感じた。

そこで、今回はハードウェアの制御についてこの記事を書くに至った。この記事では主に H8 マイコンを用いた組み込みソフト開発環境の導入について記す。組み込みソフトというと敷居が高そうと思われそうだが、別にそうでもないことを示すことができたらと思う。

2　概要

それでは、今回用いるマイコンボードと開発環境について紹介する。

2.1　マイコンボード

今回この記事では、H8 シリーズのマイコンボードを用いて説明を行う。H8 マイコンは平成 8 年に日立製作所の子会社のルネサスから発売されているボードであり、安価でそこそこの性能を持つちょうどいいマイコンである。秋月電子なら 3000 円程度あれば，H8 ボードを購入することができる。

マイコンを取り扱う場合に気をつけなければならないのは、『マイコンのどのピンがどこにつながっているか』ということ。

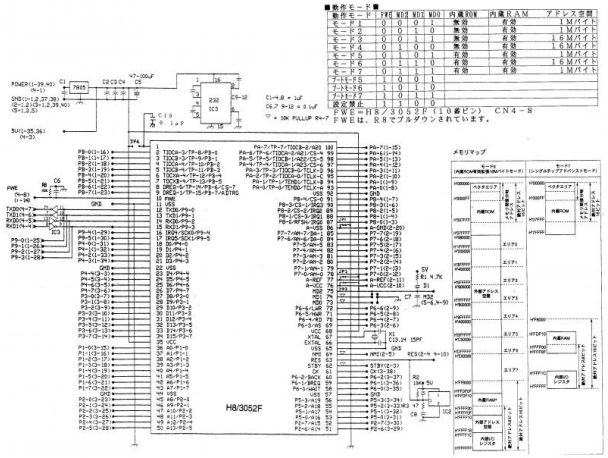

図 1．マイコンのピン配置図の一例

図 1 のような表で確認できるが，一見すると非常にややこしい。おそらく紙面上では潰れてしまって何がなんだかわからないとは思う。マイコンを購入すると、こういったピン配置図が必ず添付されているはずである。また、それが無くともマイコン自体のピン配置図は各種データシートにて大きい画像を確認できる（下に一例の URL を記述した。）。これから出ているピンを目で追っていく事でも確認できる。おおよその場合、添付されている説明書の中に，表 1 のような単に文字だけの表がある。

ピン番号	ポート	ピン番号	ポート
1	+5V	5	P22
2	P11	6	Rx00
3	P12	7	Tx00
4	P22	8	GND

表 1　ピン配置図表の例

この表の見方は、ピン番号 1 番から＋ 5V の電源が供給されており、2 番ピンと 3 番ピンがポート 1 の 1 番と 2 番、4 ピンと 5 ピンがポート 2、そして 6 番ピンが入力端子、7 番が出力端子で 8 番が GND になる。1 番の +5V でセンサに電源が行き、その末端が GND に帰る、というのが一般的なパターンである。

日立 16 ビットマイクロコンピュータ H8/3048 シリーズ　データシート
http://tokyo-ct.net/usr/matsu/learning/h8training/j602093_h83048.pdf

2.2　開発環境

開発環境はルネサス社製の統合開発環境 High-performance Embedded Workshop を用いる (通称 HEW) 。無償評価版であれば、評価機関が過ぎてもある一定サイズ以上のプログラムのコンパイルができないだけで、それ以外は通常の HEW として使うことができる。H8 版はルネサスの公式ホームページからダウンロードして、インストールすることができる。(http://japan.renesas.com/support/downloads/download results/C2000801-C2000900/ evaluation software h8c.jsp)

ちなみに、これをダウンロードするために，MyRenesas というページに登録する必要が有るため、注意が必要である。一応、日立製作所謹製のため心配は不要だと思うが、気にする人は回避したほうがいいのかもしれない。

3　準備

紹介が終わったところで、実際にマイコンボード使ってみようと言いたいところだが、まずは準備を整えなければならない。

3.1 電源の供給

マイコンボードは当たり前だが、電力の供給がなければ動かず、電池などで電力を供給してやる必要がある。そこで、電源供給の手段を用意しなければならない。秋月電子などで売っている H8 マイコンボードはだいたいが 5V 駆動なので、電源そのものは単三乾電池 4 本を直列に繋いでやるか、電池ボックスを購入すればよい。しかし、マイコンボードの電源供給ピンがボードによって異なるため、それはマイコンボードの説明書を見ながら実装していただくほかない。初めてであれば部品実装済みの組立品ボードを購入していただくのをおすすめする。

3.2 通信ケーブル

この HEW からマイコンボードにプログラムを転送するためのものを用意しなくてはならない。マイコンボードでは主にパソコンと通信するときに RS232C というシリアルポートの通信規格を用いており、USB ではなく主に D-Sub9 ピンアダプタにて行う。パソコンから送信するため、USB-RS232C の変換ケーブルを用意しなくてはならない。

図 2. USB − RS232C 変換ケーブル

さらに、RS232C からマイコンボードに伸ばすケーブルも作らなければならない。既成品を購入してもよいのだが、自作したほうがはるかに安価なため自作方法を紹介しよう。

3.3 ケーブルの自作

H8/3048 には書き込みボードが内蔵されているため、ケーブルをハンダ付けするだけで書き込みケーブルが完成する。この自作の際には、マイコンのピン配置図から Rx と Tx が伸びているピンを見つける必要がある。

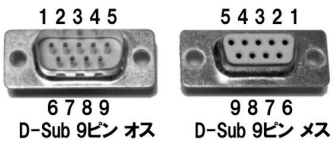

図 3　Dsub9 ピンコネクタのピン配置図

Dsub9 ピンのピン配置図は図 3 のようになっており、今回使用するのは 2 番の Rx（INPUT)、3 番の Tx(OUTPUT)、5 番の GND である。それらについて、2 番をマイコンの Tx(OUTPUT)、3 番をマイコンの Rx(INPUT)、5 番をマイコンの GND(VSS) に接続する。OUT から出た信号を IN がキャッチする格好になる。Rx と Rx を繋いで「つながらないぞ？」というミスもあるので、間違えないよう注意してほしい。

なお直接ハンダ付けしてしまうと PC にマイコンボードがくっつく形になってしまうため、コネクタを挟むなどしたほうが良い。完成図は図 4 のようになる。

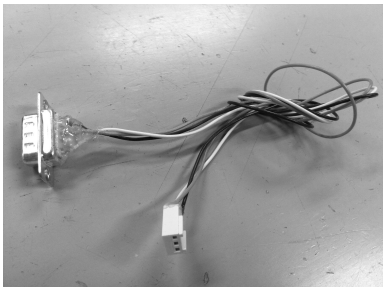

図 4　完成図

3.4　CpuWrite の入手

あとは、HEW から直接 H8 へと書き込みできるプラグインをインストールすれば、準備は完了。マイコンカーラリー ダウンロードページ

http://www.mcr.gr.jp/tech/download/main02.html

より、『●ルネサス統合開発環境用その他ソフト Ver1.40 2012.01.30』 をダウンロードして解凍する。そして、フォルダの中に『CpuWrite』の実行ファイルと「CpuWrite の登録方法」というテキストファイルがあるはずなので、それに従って登録してほしい。これで、準備は完了だ。

4　電子工作

準備は完了といえど、実はマイコンだけでは何もできない。マイコンにつなぐ何かを作らなければならないのである。そこで、簡単な電子工作の技術も必要となる。ここでは、簡単に LED を光らせることを目的とした回路を作ってみよう。

4.1　LED 発光回路

回路に必要なのは 3 つ。「VCC」「抵抗」「GND」である。VCC と GND とは、電池の＋と－を表す記号だと思って差し支えない、電池の＋側が VCC で、－側が GND である。VCC はマイコンの信号で十分なので、

[マイコンの信号線]-[抵抗]-[LED]-[GND]

のような回路を作ればよい。マイコンの信号線のことは置いておいて、まずは LED と抵抗を結んでそれらの GND の末端を一つの線にまとめる作業をしよう。LED と抵抗は秋月電子などで売っており、こちらは 300 円もあれば 8 個ずつ揃えることができる。それらをユニバーサル基板にハンダと配線材を使って適当に配線しよう。

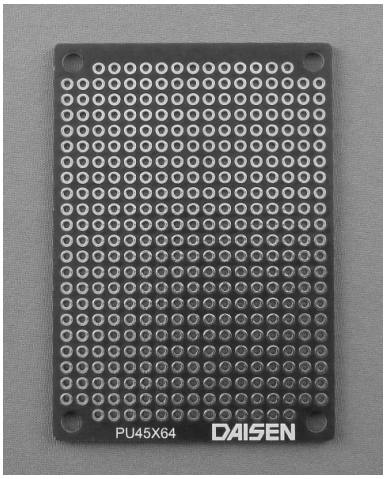

図 5　ユニバーサル基板の一例

4.2　マイコンとの接続

では次にマイコンとの接続だが、どこのポートを使うか決めて置かなければならない。ピンが別々の場所に散らばっているポートを使うと面倒なので、ここれは大抵の H8 マイコンでは固まって配置されているであろうポート 1 を例に進める。ポート 1 の 8 本に直接にハンダ付けするか、もしくはピンヘッダと 10 ピンコネクタのようなものを使うとよい。

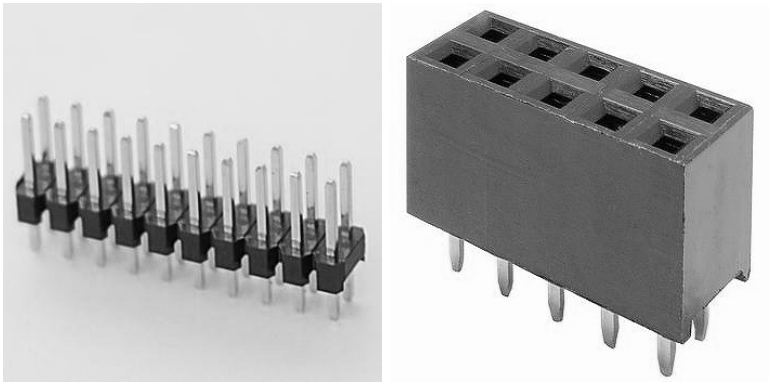

図 6　ピンヘッダと 10 ピンコネクタ

これを発光回路基板とマイコン基板に取り付けてやれば、これの抜き差しだけで接続できる。筆者としてもおすすめの方法である。

5　ワークスペースの設定

では、いよいよパソコンと接続してプログラムを入れてみよう。その手順を追って説明する。

5.1　ワークスペースの設定

まずは、HEW でワークスペースを作ってみよう。HEW を起動し、新規ワークスペースの作成を選択する。出てきたウィンドウに好きな、ワークスペース名とプロジェクト名を入れよう。ディレクトリや CPU 選別が『H8S,H8/300』になっていることを確認して OK を押す。
そして次のウィンドウでは、ツールチェイン番号はそのままで、CPU シリーズを選択する。H8/3048 であれば 300H シリーズである。そして、自分の使用している CPU を選択しよう。

すると、画面が開いて左にワークスペースができたのが確認できるはずだ。

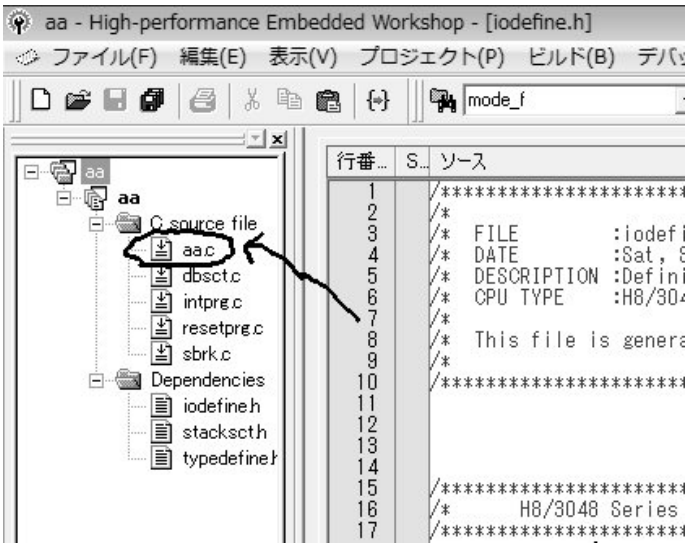

図 7　ワークスペース

では、このワークスペースの中の（プロジェクト名）.c を開いてみよう。すると、プログラムの原型のようなものが書いてあるはずである。ここにプログラムを書いて開発していくことになる。

5.2　プログラム

では、華のプログラムである。今回は「ポート 1 につないだ 8 個の LED を光らせる」というプログラムを書いてみよう。といっても勝手がわからないだろうから、サンプルを例にして説明する。

```
1  #include ”iodefine.h”
2   void main(void){
3        P1.DDR = 0xff;
4             while(1){
5                  P1.DR.BYTE = 0xff;
6             }
7  }
```

コード 1　入出力設定

これがサンプルである。あれ、意外と短いな。と思った方が多いかと思われるが、LED を光らせるだけならば特に難しいことはないのだ。
では、それぞれの部分を解説しよう。……とはいっても、C 言語とは何か、から解説していては埒が開かないので、この記事を見ていただいている方には、C 言語については多少の知識があるという前提で話させていただく。といってもそれほど高度なものは必要なく、if-else 文、for 文、while 文程度の知識があれば問題なくマイコン開発はできる。

5.2.1　include

今回は HEW に付属しているユーザ定義の関数系を呼び出している。C 言語においては、関数系を呼び出すときは#include のあとに< >で囲むが、HEW でユーザー定義の関数系を呼び出すときには< >ではなく” ”で囲うというルールがある。そのため、今回のコードにおいては #include”iodefine” という記述をしている。

5.2.2　**入出力設定**

関数の直前、コード 1 の 3 行目に記述してある『P1.DDR = 0xff』とは、ポートの入出力をどうするかという設定である。DR と DDR という設定には「このポートを入力として用いる」「このポートを出力として用いる」という意味が含まれている。

次の行の 0xff というのには、「では、このポートの何番目までを入出力に用いるか？」という設定である。ポートのピンは 8 本あるのだが、そのポートピンの 1 本目〜8 本目は 2 の（ピン番号－1）乗で表し、それを 16 進数に治してプログラムに書く。

例えば、ポート 1 の 1 番目のピンのみを入出力に用いたい場合は 0x01、2 番目のピンのみを使いたい場合は 0x02、1 番目と 2 番目を使いたい時は 0x03、5 番目と 3 番目を使いたい時は 0x18、のように記述する。
今回は LED を光らせるためにポート 1 のすべての端子を出力として利用するため、P1.DDR=0xff と記述する。

5.2.3　**関数内部**

関数内部、コード 1 の 4 行と 5 行目に記述してある『while(1)』と『P1.DR.BYTE = 0xff』だが，これは『while(1)』つまり永遠に『ポート 1 のバイトに 0xff を出力する』という意味である。先ほど設定したポート 1 の出力に 0xff という信号を送る動作を継続的に行なっている。

5.2.4 書き込み

では、このプログラムを実際にマイコンに書き込んでみよう。まずはこのプログラムを HEW 上部のビルドタブからビルドを選んでビルドする。その後マイコンの書き込みスイッチを書き込み側に倒し、パソコンと接続し、ツールタブの CPUWrite を実行して開始ボタンを押す。正しく進行すれば成功である。

マイコンに LED 発光回路を接続してマイコンの電源をつければ、LED が発光するはずである。

6 応用

入出力設定の簡単な拡張例を考えてみよう。たとえば「時間おきに LED が順に点滅する」というプログラムである。C 言語で書いているため、処理は C 言語とまったく同じである。

```
#include ”iodefine.h”
void main(void){
unsigned long i;
P1.DDR = 0xFF;


      while(1){
           P1.DR.BYTE = 0x01;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x02;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x04;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x08;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x10;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x20;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x40;
      for (i=0;i¡1000000;i++)
           P1.DR.BYTE = 0x80;
      for (i=0;i¡1000000;i++)
      }
}
```

コード 2　複数の入出力設定

これで、一定時間おきに順番に LED が光るプログラムが完成である。
このプログラムのように時間稼ぎに i++ のような手法を用いる場合、CPU のクロック数というものを意識する必要がある。CPU というのは御存知の通り我々の想像もつかないような速さで計算しているが、その計算にかかる時間は 0 ではない。その時間は CPU のクロック数から求まり、H8 マイコンの場合おおよそ 16MHz であることから、一秒間に 1600000 回の計算をしているということになる。そのため、一回 i++ を行うにはだいたい 625 ナノ秒必要ということになる。であれば、i++ の上限を調整することによって時間を調節できるということになる。上のプログラムの i の上限をいろいろ弄ってみて試していただきたい。

6.1 応用の一例

これができるようになるということは、複数の出力を同時に制御できるようになったということである。複数の出力を同時に制御することで使えるようになるものとしては、たとえば以下のようなものがある。

- 電光掲示板
- クリスマスツリーの電飾
- 7 セグメント LED

これからクリスマスの時期、クリスマスツリーを自分の思うように光らせたい！と思ったならばすぐにでも使えるだろう。

今回ここにて説明したことは、プログラム内部にやりたいことを記述してそれを出力させる方法、C 言語でいうところの printf 関数である。コントローラーなどからの信号を受け取り、それによって出力を変える scanf 関数のようなものももちろん実装可能で、ページ上の都合からこの記事では割愛したが、そちらもさほど難しいことではないため、意欲溢れる諸兄にはぜひ挑戦していただきたい。

なお、それらができれば自作のコントローラーというのも十分に可能である。スティックや R2 ボタンの押し込みなどを再現したいのならば、AD（アナログ－デジタル）変換をマスターする必要があるが、DDR コンのように『押したか，押されていないか』だけを見るようなコントローラーなら問題なく製作できるはずである。

7 最後に

さて、マイコンを手に入れて開発環境を手に入れたあなたは、一端の組み込みソフト開発者となった。マイコンに関する資料はネット上にいくらでも出まわっており、それらを見れば本当になんでもできる（参考の項に記述してあるので、興味があれば見ていただきたい）。

マイコンを用いたハード寄りのゲーム（赤外線ガンシューティングゲームだとか、デジタル野球盤など）とかを考えても良いし、モーターやセンサで駆動するものを作ってニコニコ技◯部などに投稿してみるのも良いと思う。無論簡単な道のりではないが、可能なのか不可能なのかというと

全て可能である。

これにより、ソフトだけではなくハード面でもゲーム開発に活用していただければ、よりゲーム開発の幅が広がるのではないかと思う。

8 参考

今回この記事を書くにあたり、以下のページを参考とした。

takurobo 工作室 http://homepage1.nifty.com/rikiya/
マイコンカーラリー .net ダウンロードページ http://www.mcr.gr.jp/tech/download/main01.html
ET ロボコンではじめるシステム制御 http://monoist.atmarkit.co.jp/mn/articles/1003/18/news094.html
H8 マイコンボードで動作する組み込み OS を自作してみよう！ http://monoist.atmarkit.co.jp/mn/articles/1103/31/news004.html
日立 16 ビットマイクロコンピュータ H8/3048 シリーズ　データシート http://tokyo-ct.net/usr/matsu/learning/h8training/j602093_h83048.pdf