AVR（1chipマイクロコンピュータ）をBASICで使ってみる

2014.01.20　JA1VCW

A.　はじめに

1chip マイクロコンピュータ（以下マイコンと略します）を自分でプログラムできると、最近良く使われるシリアル設定のICが使いやすくなります。 また、計算ができますので値の変換等も可能です。
マイコンのプログラムと言うと難しそうですが、やって見るとそんなに難しいものではありません。
今回は入門の、門の前くらいです。（このくらいの知識しか無いのが本当の所です）
まあ、あまりお金もかからないし時間もつぶせますので、もしよろしければ遊んで見ませんか。
ただし、何事もそうですが少しの興味と気力は必要ですし、深みにはまるとそれはそれで少し大変です。

インターネットで検索しますと、多くの方が本稿と同様な文章を書かれています。
私個人が経験したり思ったりした事を、私なりに書いてみました。

B.　この文章の目的

・マイコン使うと何ができる？
・どうやったら働かせることができるの？

という方々に

・ふーんそんなもんかい
・ひょっとしたら出来るかも
・チャレンジしてみるか

と思ってもらうことを目的とします。

従って必要なことをなるべく簡単に述べたいので、良くご存知の方が

・こんな非能率なプログラムをかいて・・・
・こういう命令を使ったら簡単になるのに・・・
・こんなこと常識じゃん・・・

などと思われる場合が多々出て来ると思いますが、目をつむってください。

C.　方針

次のような方針で述べてゆきたいと思います。
私の知っている範囲の内容です。 知らない事もたくさんありますので、その場合はご容赦ください。

目次

1.　マイコンを使うと何ができるか

1）DDS、PLL、AD変換器、DA変換器などの条件設定で、シリアルデータで設定する場合が増えています。 この場合マイコンを使用すると、簡単に且つ自由に設定ができます。

2）マイコンに接続されている外部回路の状態をマイコン内に取り込んで、その状態に応じてあらかじめ決めてある範囲で判断し、a）の設定が変えられます。

3）複雑な計算を実行させて表示させる事が出来ます。　例えばmWをdBmに変換して表示する、カウンタの数値を読み取りオフセットを加算して表示するなど。

4）カウンタやAD,DA変換器を内蔵するようなマイコンもありますので、それを使うと周波数カウンタや電圧の測定、発生等が比較的簡単に可能となります。

2.　単純化したマイコン

マイコンを単純化すると図のようになります。

①プログラム

人がこのマイコンに実施させたい事を規定に従って(命令と言います）書きます。 プログラムはひとつひとつの命令の羅列によって成り立っています。 命令はStatementともいいます。

②処理部

①の命令の内容を分析理解して実行します。

③とのデータのやり取りを行います。

④を通して外部機器とのデータのやり取りを行います。

③記憶部

メモリです。 いろいろな数値などのデータを保持します。 一般にマイコンの電源を切ると内容は消えます。 量は少ないですが電源を切っても消えないメモリも有します。

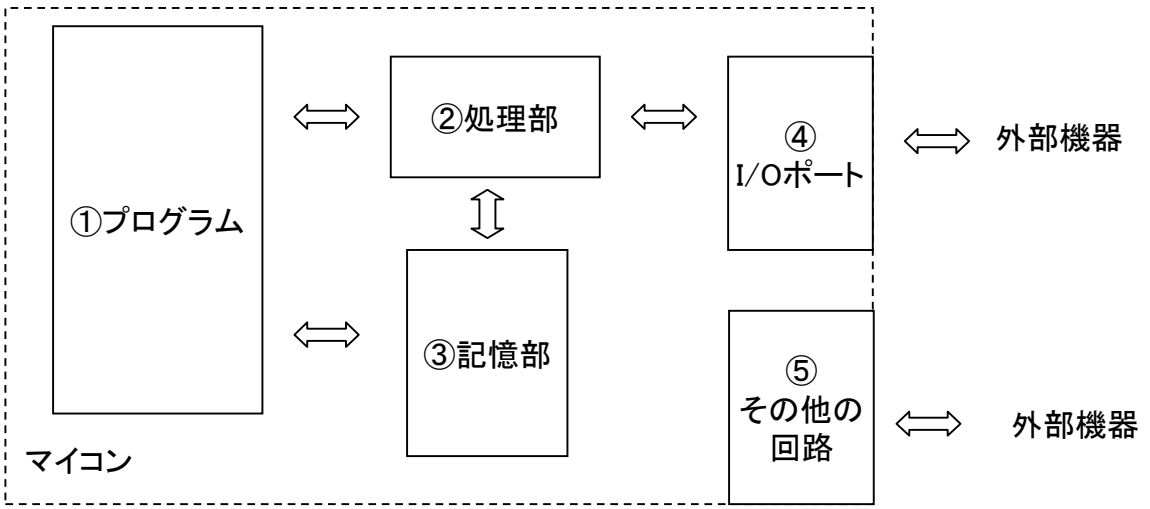
④IOポート

外部機器とのインターフェイスで、マイコンの種類によって使用できる数が違います。

通常ロジックレベルで入出力されます。 この状態を②で監視、コントロールできます。

⑤その他の回路、機能

A/D,D/A変換器およびそのマルチプレクサ、タイマ／カウンタ、UART、ウオッチドッグタイマ、インタラプト等　便利な機能がいっぱい。　今回は手始めなので使いません。



3.　プログラム言語

3．1　プログラムとは

マイコンはプログラムに書かれている命令を順番に処理してゆきます。 プログラムについては
1)プログラムはその製作者が明確な意図をもって作成する、マイコンに対する指示書です。
2)マイコンはプログラムに書かれているように処理しますので、書かれていないことや間違って書いた場合は製作者の意図通りには処理してくれません。
3)プログラムは必要な動作を漏れなく記述する必要があります。
4)プログラムの記述には規則や文法があって、それに則って記述しなければいけません。
これはマイコンにとって理解しやすく、また製作者に作成を容易にし、且つ他人が見た場合にわかりやすくするための工夫です。
プログラムに対する責任は99%以上作成者にあります。 正しく動かないのはほとんど作成者のせいです。
プログラムは製作者の意図そのものですので、他の人の書いたプログラムを解析したり理解したりする事は生まれも育ちも違うひとの意図を理解することになり、なかなか大変です。

3．2　言語

前述のとおりプログラムはマイコンに対する指示書です。 そして指示の方法はいろいろあります。
それを言語と言ったりします。
1)アセンブリ言語
マイコンのハードウェアに対して、直接指示を行う言語です。 機械語と言うこともあります。
a)すべての言語のなかで一番細部に渡って指示が可能な言語。 ハードウェアと1対1に対応します。
b)細かく指示できる代わりに、細かく指示しないと意図したように働かない事が多いです。
c)マイコンによって書き方が違うので、A社とB社では書式自体が同じではない。 つまりマイコンのメーカーを変えると、そのメーカーの製品に対応するアセンブリ言語を使用しなければなりません。
（まあ、でも似たような命令がありますので全く1から覚えなおさなければならないようなことは無い様です。 しかし、書式は変わります）。
2)コンパイラを使用する言語
1）のc)項の不便を解消するために1）の言語の前段階で処理を行う事を前提に書き方を統一し、マイコンの仕様の違いを無くそうとした言語。 C、BASIC、FORTH等があります。
a)あらかじめ使いやすいようにプログラムの体系がきめてあって、その体系内の規定に従って命令を書きます。
b)そのようにして命令を書き、コンパイラというソフトウェアで処理するとどのようなマイコンでも同じ動作になります。 ただしマイコンによってコンパイラが違います。
すなわちA社のxxマイコン用のコンパイラ、B社のyyマイコン用のコンパイラという様に、マイコンに適合したコンパイラ使用しなければなりません。

今回はマイコンにATMEL社のAVRを使用します。 そしてコンパイラとしてBASCOM　AVR というソフトウェアを使用することにします。
BASCOM はBASIC という言語の書き方でプログラムを書くと　AVR というマイコンで動作するような機械語に変換します。　 それをAVR 本体に書き込むと動作します。
C という言語もあってむしろこちらのほうが世の中に普及していますが、入門用としてBASICにしました。

他にもMICROCHIP社のPIC というマイコンもあります。

３．３ BASCOMの使い方

BASCOMにはたくさんの機能があるのですべてを網羅するのはここでは不可能です。
単にハードウェアをコントロールするだけでしたら、そんなに難しいことではありません。

３．３．１ BASCOM AVR の入手方法

こちらのホームページ（HP）に詳しく載っています。
http://www.ne.jp/asahi/shared/o-family/ElecRoom/AVRMCOM/BASCManu/BASCMupw.html

書き込み器を指定しますが、私の場合は自作のものですので ［External Programmer］ でした。
このHP はいろいろな重要情報を与えてくれます。 ぜひ一通り読むことをお勧めします。
また命令などの詳しい説明もあり、プログラム開発には非常に役立ちます。（むしろ必須です）
私の場合は命令の動作の説明はほとんどこのHPを利用させてもらっています。（何しろ日本語）

３．３．２ スタートする

BASCOMがインストールできたら、 BASCOM のアイコンをクリックしてスタートします。

３．３．３ Editする

最初は File => New とすると noname1 という名前の画面が開きます。
この中に直接プログラムを書いてゆきます。
命令語は太字で表示されます。 またプルダウンメニューが表れて選択するような場合もあります。
メモ帳などでプログラムを書いて、Copy&Paste してもOKです。
ただし正しく入力されていることを確認の事。

書き終わったら File => save as.. でファイル名を付けてsaveしておきます。 Save するとその名前が付いてそのファイル名で開けます。

３．３．４ コンパイルする。

Edit の状態ではそのまま、従来のファイルを使用する場合は File => Open でファイルを開きます。
その状態で Program => Compile でコンパイルができます。
エラーがあれば窓の下の方に表示されるので修正します。
コンパイルが終了するといくつものファイルが生成されますが、重要なのは xxx.hex というファイルで
この内容を書き込み器に転送することによってAVR がプログラムされます。

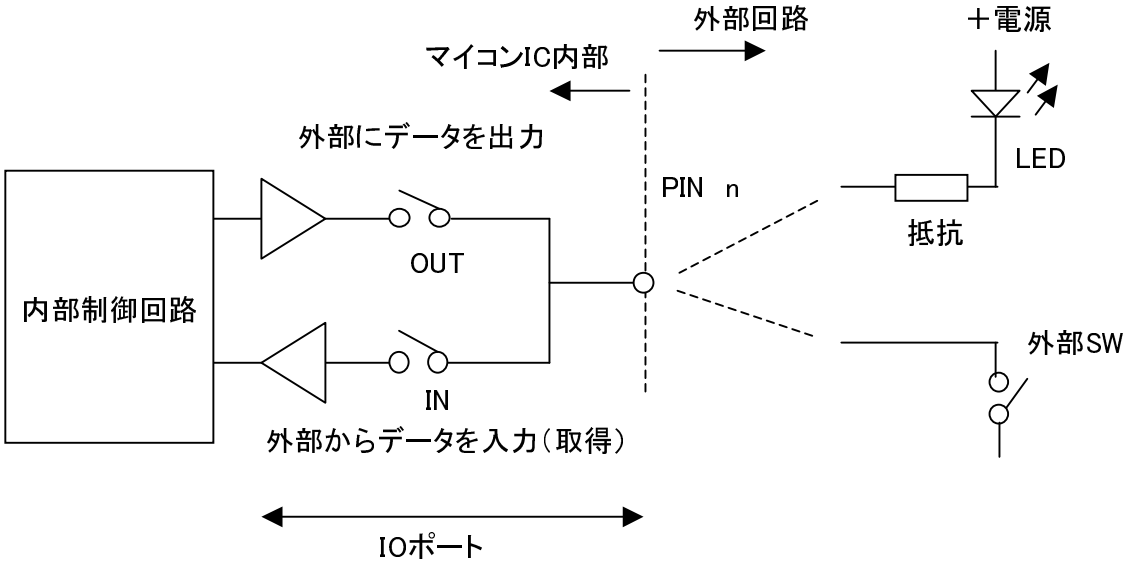
また xxx.rpt というファイルが生成されます。 通常ではファイルを開けませんが、これはASCIIファイル
なのでメモ帳で開けます。 いろいろと記述がありますが、12行目にROMIMAGEという項目 があります。
これはプログラムの大きさを表している様です。

.hexファイルをAVRに書き込む方法は後述します。

4.　実際のハードウェア

ATMEL社のAVR を使いますが、基本的にマイコンとマイコンによって制御される外部回路は、マイコン内のIOポートと呼ばれる部分を使用して制御されます。

4．1　単純化したIOポートについて



図は単純化したIOポートです。

1）PINn に接続する外部回路が（マイコンから見て）出力になるか、（マイコンから見て）入力になるかを決めます。 それは回路を設計するときに設計者が決め、そのような回路の接続を行います。
図ではLED を制御したければ出力に、外部SWの状態を監視したい場合は入力になります。

2）出力に設定したときにはOUTのSWをon に、INのSWをoff にし、PINn にLogic”0”を設定すれば点灯。Logic”1”を設定すれば消灯します。

3）入力に設定したときにはOUTのSW を off に、IN のSW を on にし、PINnの状態を見れば外部SW が on かoff かを知ることができます。

4）基本的にIOポートはデジタルなので、出力は“0Vまたは電源電圧”、入力は”0Vまたは電源電圧”の電圧値が入出力されます。（アナログを扱える機能もありますが、本稿では省略）

5）OUT、IN のSW をon/offしたり、PINnに値を設定したり状態を見たりすることは、そのための命令が用意されていて、その命令をプログラムに書けば実行するときにそのように動作します。

6）実際のIOポートはもっとずっと複雑で、入力に設定したときにプルアップを接続したり、出力のときに3ステートのバッファにしたり、出力した状態を読み込んだり いろいろな機能があります。
それらの機能はすべて専用の命令があってそれによってコントロールができます。
しかし、今回はこの単純化したIOポートが理解できれば十分です。

4．2　どのようなAVR を使用するか

ATmega328Pを使用します。

これを選定した理由は

・プログラムメモリが32kbyte まで使用できる。(BASCOMでは無償版では4kbyte の制限あり)

・IOポートが23本。 小さなコントローラとして適当な数です。

・28ピンDIP なので扱いが楽。 この上位は40ピンDIP かまたはQFP になるので扱いにくい。

・安価 (これを書いている時点で¥250-(秋月電子通商))

・何回もプログラムの書き込みができます。 10,000回を保証しているので、プログラムを変更=>書き換えという方法でデバッグできます。 以前100回というマイコンもありましたが、書き換えるたびに正の字を書いていました。 10,000回だったら書き込回数を気にせずにデバッグできます。

実際の使用方法としては次のようです。

1)PBn, PCn, PBn はすべてIOポートに設定することができます。

2) ()内は別の機能です。 IOポートとプログラムで切り替えて使えるものもあります。

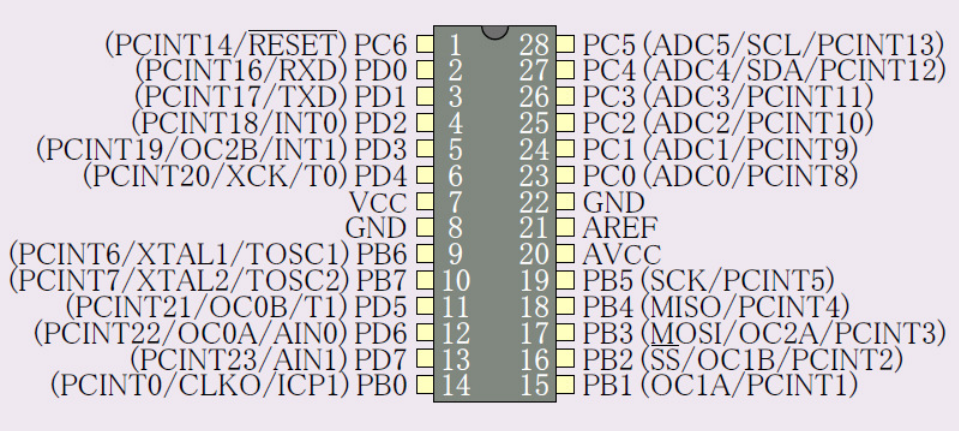
3)一部のIOポートはクロック発振用の水晶発振子が接続されたり、プログラム書き込み用のコントロール端子に割り当てられたりしていますので、それらは特別にリザーブされます。

4)とても多機能ですので、ここですべてを説明することはできません。 取説を参照ください。
幸い、インターネット内に日本語の説明もあります。(メーカーで翻訳したものではないようです。)
http://www.avr.jp/
等があります。

5)書き込み用として *RESET, SCK, MISO, MOSHI およびクロック用としてXTAL1, XTAL2
これらは最初はIOポートには使わないようにしています。 (IOポートが不足した時には使う場合もあります)

6)最初はこの回路(後ページ)から始めます。

図はATmega328Pのピン配置です。



4．3　開発システムについて

　これだけあれば開発が可能であるという最小のシステムです。

費用は概算（2014.01.10時点）で、販売店によって変わります。

1）ハードウェア

・ターゲットボード　　AVR を含みます。 いろいろな部品をふくめても本稿のレベルでしたら
　　¥1k- ～ ¥2k- でしょう。 LCD が一番高価ですが ＜ ¥1k- 程度。

・プログラムライタ　　純正品で ¥3k- 強。
　　自作で ＜ ¥1k- 。

・電源　ケーブル等　　必要に応じて。

2）ソフトウェア

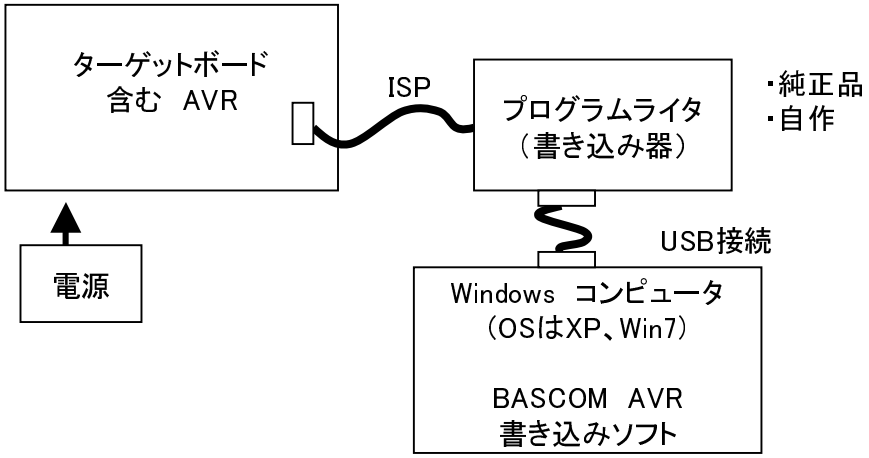
・BASCOM　AVR　　AVR用ベーシックコンパイラ。
　　無償版　　無償。 ただし生成プログラムが約4kbyte以下である制限あり。
　　若干のバグもある様です。（バグも内容がわかっていれば仕様！）
　　製品版　　制限なし。 ¥10k- ～ ¥15k- ??

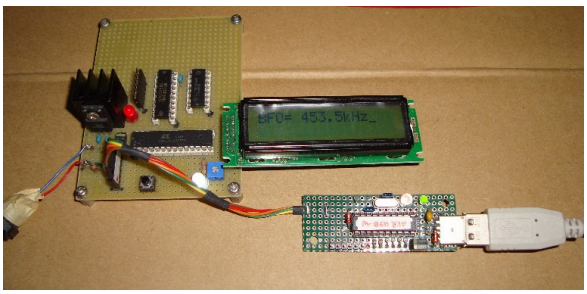
・書き込みソフト　　純正品のプログラムライタ使用の場合はその中に含まれているはず。
　　自作　　hidspx.exe フリーソフト。 別の拙稿参照のこと。（AVR書き込み器）

3）コンピュータ

　現在　Core2 CPU 6600 2.4GHz、 Memory 4G、 Win7

　以前　Celeron 2.4GHz、 Memory 1G、 WinXP でも十分使用できました。





実際にこれだけです。

・ターゲットボード　　横長のICがAVR
　　12V => 5V のレギュレータ付き
　　余計な回路も少し付いています
　　LCD も付けました

・右下　自作書き込み器　（USB接続）
・電源（写っていません）
・コンピュータ（写っていません）

5.　回路を考えます

1)目的･･･AVR をBASIC で動かすための練習用回路を作ります。

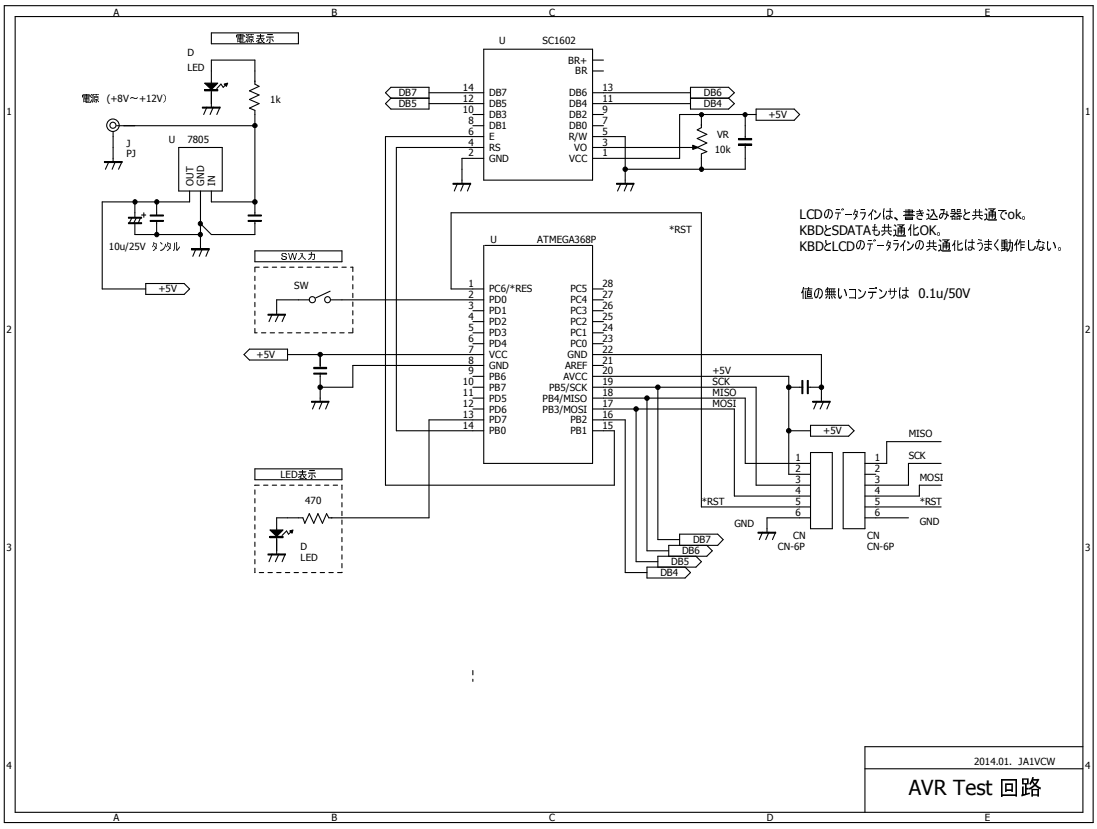
2)出力のためにLED1個と、入力用にSW1を1個配置する。

3)LED用にIOポートPD7 を使用。 PD7･･IOポート“D”のbit7の意味。

4)SW入力としてPD0 を使用。　PD0･･ IOポート“D”のbit0の意味。
IOポートは“D”である必要は無く、“C”でも“B”でも可。 Bit もどこでも可。 ただし書き込み回路とは重複しないほうが良い。 IOポートをたくさん使ってもう無いよということになったら書き込み用のポートを、動作を良く理解して使用可。

5)BASCOMの特徴の一つとして16文字＊2行のLCD ディスプレイを接続すると、LCD 命令で文字列を表示することができます。 これは変数も表示できますのでデバッグ等に大変便利です。
（本当のことを言うと不可欠です。） LCD接続回路を付けておきます。 コネクタです。
4)で書き込み回路と重ならないほうが良いと書いてありますが、今回は大丈夫です。 ピンの節約から意図的に重ねてあります。

6)IOポートのリストを作ります。 作らないとデバッグの時にとても効率が悪い。 必須です。



IOポートのリスト

Port	Bit	I/O	CPUpin	初期値	信号名	内容
D	7	O	13	L	LEDout	LEDを制御。 ″1″で点灯
D	0	I	2	―	SWin	SW入力。 スイッチonで″0″
B	5	O	19	―	DB7	LCD用データ BASCOMの命令で使用
B	4	O	18	―	DB6	
B	3	O	17	―	DB5	
B	2	O	16	―	DB4	
B	1	O	15	―	Enbl	LCDイネーブル BASCOMの命令で使用
B	0	O	14	―	RS	LCDイネーブル BASCOMの命令で使用

6.　プログラムを書いてみる

いくつかのプログラムを作ってみました。
この例では少し注意があります。 これらは本稿に記してあるすべてのプログラムで発生しています。

1)プログラムに使用される文字は大文字と小文字ですがBASCOMのEditorでは区別が無く自動的に修正されます。 本稿ではPowerpointの性質で区別できません。 実際と少し違う場合があります。
2)コメントの開始はシングルクオートですが、'に変換されてしまって直しかたがわかりません。
3)細かい命令の動作はとても説明しきれないので、**BASCOM AVRの入手方法に載せたHP**を参照願います。 本稿では必要最小限で、内容が理解できる程度の説明とします。
4)本稿のプログラムの中のコメントや説明に日本語が書いてありますが、実際のBASCOMの無償版プログラムではバグがあって、日本語は避けた方が良いようです。
5)実際にAVRに書き込んで動作させていますが、何かの間違いで動かなかった場合はご勘弁ください。簡単なプログラムなのですぐに直せるでしょう。
6)LEDの点滅から始めます。 どこにでもあるプログラムですが、物には順序ともいいますので・・・・

6.1　LEDの点滅　(LED.bas)

LEDを100msごとに点滅させるプログラムです。

```
Rem  LEDtest                    'Remと書くとその後改行までコメントとなります

$regfile = "m328pdef.dat"       'ATmega328Pを使用するときに書きます。 他のシリーズでは当然
                                '使用するマイコンに合わせて書き換えます。
$crystal = 1000000              'クロックを1MHzに設定します。①
Config   Portd.7 = Output       'LEDをIOポートPD7に接続してありますのでPD7を出力に設定します。

Do                              'プログラム制御命令で このDoから下のLoopの間を繰り返します。
   Set   Portd.7                'PD7をLogic"1"にセット。 Logic1 はその端子に約5Vを出力。点灯。
   Waitms 100                   'ここで100ms待ちます。 単に待つだけなので点灯したままです。③
   Reset   Portd.7              'PD7をLogic"0"にセット。 Logic0 はその端子に約0Vを出力。消灯。
   Waitms 100                   'ここで100ms待ちます。 単に待つだけなので消灯したままです。
Loop
End                             'プログラムの記述の終了を宣言します。
```

①この周波数設定はヒューズビットと関係します。 デフォルトでは内部クロック8MHz、クロック分周1/8という設定なので1MHzとなります。 ヒューズビットについては後出します。
②Do～Loop の間に書いた命令を無限に繰り返します。 またこの間の命令の書き出しにスペースが入れてあるのは単に見やすくするためです。 この部分がブロックですということを示しています。
③Waitms なので ms 単位になります。 ①の設定と関係します。 時間をコンパイラが計算してくれるようで、ハードウェアは1MHzなのに①の設定を2MHzにすると実際には200ms待つ事になります。

これをBASCOMでコンパイルして、その出力ファイルを書き込み器を使ってマイコンに書き込めば
書き込終わった直後からLEDが約100ms間隔で点滅します。

前記命令の動作はコンパイラによって厳密に定義されいて、その書式に従って書かなければいけません。
なお‘以降改行までの間は、コメントとして何を書いても良い事になっていて、プログラムのメモとして
使用します。 コメントはできる限り書くこと。

6．2 LEDの点滅 その2 （LED2.bas）

LEDを乱数の時間で点滅させるプログラムです。

```
Rem  LEDtest                    ‘Remと書くとその後改行までコメントとなります

$regfile = ″m328pdef.dat“
$crystal = 1000000
Config Portd.7 = Output

Dim Tm As byte                  ‘Tmという名前の変数を、byte型で定義します ①

Do
    Tm = rnd(255)               ‘0～254までの乱数を生成して、変数Tmに代入 ②
    gosub Light                 ‘サブルーチンに移行します
Loop

Light:                          ‘サブルーチンの始まり
    Set   Portd.7
    Waitms Tm                   ‘Tmに代入されている値の時間だけ待ちます。
    Reset   Portd.7
    Waitms Tm
Return
End
```

①変数を定義します。 プログラム内で使用する変数はすべて使う前に名前と型の宣言が必要です。
型とは変数の内容によって決まるもので、例えばDouble型は大きい数値をしまえますがメモリを
8byte使います。 そこにByte型のように1byteの値をしまうとメモリが無駄になります。
また、ちがった型の変数を代入しようとすると、エラーやワ―ニングをだして、何かおかしくないか?
という注意を促したりします。（代入できるような例外はあります。）
②乱数を発生します。 0～（）内の数値-1 までの乱数です。

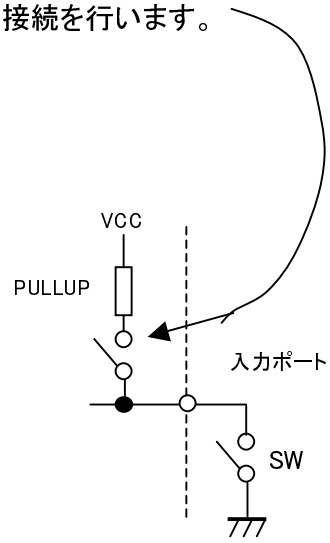
このようにするとサブルーチンの動作でLEDがTm の時間発光して、Tm の時間消灯します。
そしてまた違う乱数の値がTm にセットされて再びサブルーチンに移行します。
従って光る時間がランダムになるようなLED発光が実現できます。

６．３　SWの読み込み（SW1.bas）

IOポートに接続されているSWの状態を読んで、それに応じてLEDを点滅します。

```
Rem  SW  read

$regfile = "m328pdef.dat"
$crystal = 1000000
Config Portd.7 = Output        'PD7を出力に設定します
Config Portd.0 = input         'PD0を入力に設定します  ①

Dim Tm As Byte

Reset   Portd.7                '初期設定  LEDをoff  ②
Portd = &H01                   '入力ピンをプルアップします  ③

Start:
   Tm = Pind.0                 'PD0の内容を変数Tm に代入します  ④
   If   Tm = 1 Then            'Tm が1  であったら  PD7 を"H"にします  ⑤
      Set   Portd.7
   Else                        'そうでない場合は  PD7 を"L"にします  ⑥
      Reset   Portd.7
   End If
Goto Start                     'ラベルStart に制御を移します
End
```

①SW を入力に接続します。
　SW はピンとGNDの間に接続しているので、SW を押すとピンがGNDに接続されます。
②パワーon のスタート時にはLED を消灯します。
③入力ピンにはプルアップ機能がありますので、この命令でプルアップの接続を行います。
④SWの状態を変数Tm に取り込みます。
⑤Tm が１すなわちSWがoff の時はPD7をセット（点灯）
⑥Tm が1以外の時すなわちSWがon の時はPD7をリセット（消灯）
Start から Goto Start の間を永遠に実行します。
これでSWを押したときにのみLEDが消えるような動作になりました。



6．4 LCDdisplayのテスト(LCDdisp.bas)

LCDdisplayに文字を表示します。

前項のSW1.bas をLCDdispleyに文字や変数を表示するように変更します。 SW on の時は上の行に LED on 1 と、SW offの時は LED off 0 と表示します。

```
Rem  LCDdisp

$regfile = "m328pdef.dat"
$crystal = 1000000
Config Portd.7 = Output
Config Portd.0 = Input

Config Lcdmode = Port                    'この6行は、16文字2行のLCDを使うための設定です。 ①
Config Lcdbus = 4                        'データ転送モードを4bit モードに
Config Lcdpin = Pin , Db4 = Portb.2 , Db5 = Portb.3    'データの4bitをIOポートに割付け
Config Lcdpin = Pin , Db6 = Portb.4 , Db7 = Portb.5    'データの4bitをIOポートに割付け
Config Lcdpin = Pin , E = Portb.1 , Rs = Portb.0       'コントロールの信号をIOポートに割付け
Config Lcd = 16 * 2                                    'LCDは16文字2行のタイプを使う

Dim Tm As Byte

Reset  Portd.7
Portd = &H01
Cls                              'LCDの文字を消去

Start:
  Tm = Pind.0
  If Tm = 1 Then                 'IF～then～else 構文
     Set  Portd.7
     Upperline                   '上の行を指定して
     Lcd "LED on " ; Tm          ' LED on の文字とTmの内容 (=1)を表示 ②
     Lowerline                   '下の行を指定して
     Lcd "              "        '下の行の文字を消すために空白を16文字表示
  Else
     Reset  Portd.7
     Upperline                   '上の行を指定して
     Lcd "              "        '上の行の文字を消すために空白を16文字表示
     Lowerline                   '下の行を指定して
     Lcd "LED off " ; Tm         ' LED off の文字とTmの内容 (=0)を表示
  End If
Goto Start

End
```

①特定のLCDディスプレイを使用する場合です。 ほぼ決まり文句。 HP参照のこと。

②ダブルクオート" "内の文字はそのまま表示されます。 変数はその値が表示されます。

これもかなりバリエーションがありますので、HPを参照してください。

LCDに表示するプログラムがコンパイル時に追加されるために、プログラムメモリを消費します。

このプログラムで 892byte 使用していました。

６．５　DDSにデータをセットする（setDDS.bas）

DDSの周波数設定を行います。 ここでは例としてBFO用で453.5kHz を発生します。
値は固定になっていますが、変数にしてそれをロータリーエンコーダのパルスやテンキーのデータで変えてやれば任意の周波数を発生できます。 DDS位ですと少し複雑になりますが難しくはありません。

６．５．１　予備調査、設計など

1）周波数と周波数データ

DDSはAD9834で、28bitのアキュムレータを持ちます。 クロックとはDDSに供給するクロックです。
nを周波数データとして　　出力周波数＝（クロック周波数／2^28）＊n
出力周波数：453.5kHz、クロック周波数：12.8MHz　なので、周波数データnは上式から　&H　911EB8

2）AD9834の設定の方法は次のようです。（他にも設定モードはありますが、簡単な方法です）

下表のように28bitを14bit*2の分割したデータを含む16bitのデータを、上から順に3つ送ることによって目的の周波数を発生できます。

AD9834　設定レジスタ

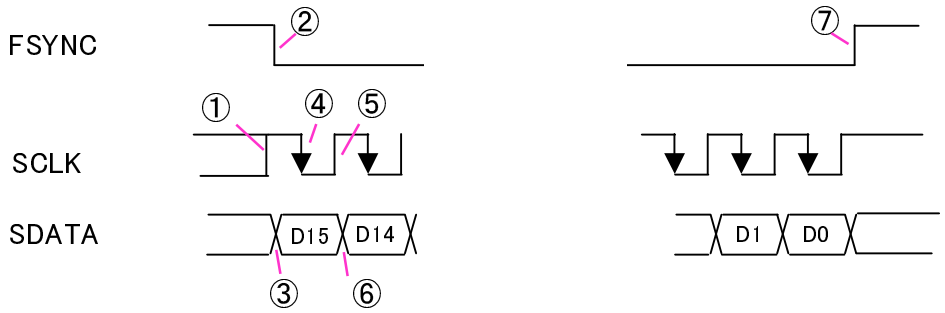
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	Hex表示
ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	&H 2000
ﾃﾞｰﾀﾚｼﾞｽﾀ L	0	1	MSB				28bitの周波数データのうち　下14bit									LSB	&H 5EB8
ﾃﾞｰﾀﾚｼﾞｽﾀ H	0	1	MSB				28bitの周波数データのうち　上14bit									LSB	&H 4244

周波数データは表示の右端の値を上から順に16bitを1ワードとして送ればよい。

3）AD9834のデータ設定タイミング

ソフトでタイミングを取って設定しますが、送出時間が長いのでクリチカルなタイミングはこの場合は必要がなく、信号の順序が正しければよいので次のようになります。

1ワード分のデータ送出



これをことばで書いてみると　イニシャル状態を　FSYNC：　“H”　SCLK,SDATA：　“X”（不定）として

①SCLKを　“H”
②FSYNCを　“L”
③SDATAに　“D15”　のデータを出力。
④SCLKを　“L”
⑤SCLKを　“H”
⑥SDATAに　“D14”　のデータを出力。
　　以下　“D0”まで④～⑥を繰り返す
⑦FSYNCを　“H”

これで16bit　1ワード分の設定ができますので、内容を前記のデータとして3ワードを送出すれば設定完了です。

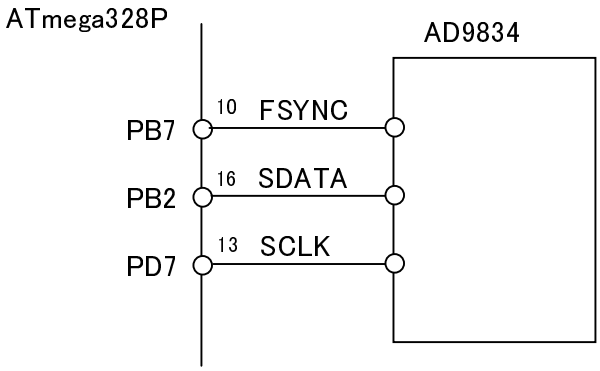
4)IOポートのリストを作る

IOポートのリスト （DDS追加）

	Bit	I/O	CPUpin	初期値	信号名	内容
D	7	O	13	H	SCLK/LED	DDS SCLK/LEDを制御 共用。 ″1″で点灯
D	0	I	2	ー	SWin	SW入力。 スイッチonで″0″
B	7	O	10	H	FSYNC	DDS FSYNC
B	5	O	19	ー	DB7	LCD用データ BASCOMの命令で使用
B	4	O	18	ー	DB6	
B	3	O	17	ー	DB5	
B	2	O	16	ー	SDATA/DB4	DDS DATA LCDDB4 共用
B	1	O	15	ー	Enbl	LCDイネーブル BASCOMの命令で使用
B	0	O	14	ー	RS	LCDイネーブル BASCOMの命令で使用

データを共用しているところがあります。 ５．の5)項です。 試してあります。
LCDのデータ信号はLCD命令でコントロールされていて、他のデータと供用してもOKの場合があります。
（かならず試してから使用の事。 ダメな場合もあるかも? 今回はOKでした）

5)DDS追加回路



6．5．2 実際のプログラムです。
このページはいろいろな定義と、電源投入時のイニシャル設定部分になります。

```
Rem  setDDS  DDS: AD9834

$regfile = "m328pdef.dat"
$crystal = 1000000

Const Bfofql = &H5EB8                    'DDS data (L)  453.5kHzのDDSデータ。 下14bit
Const Bfofqh = &H4244                    'DDS data (H)  453.5kHzのDDSデータ。 上14bit
Const Bfof = 453.5                       '表示用周波数

Config Lcdmode = Port                    'LCDのポートの定義
Config Lcdbus = 4
Config Lcdpin = Pin , Db4 = Portb.2 , Db5 = Portb.3
Config Lcdpin = Pin , Db6 = Portb.4 , Db7 = Portb.5
Config Lcdpin = Pin , E = Portb.1 , Rs = Portb.0
Config Lcd = 16 * 2
Config Portd.0 = Input

Config Portb.7 = Output                  'DDS FSYNC            DDSの同期信号
Config Portd.7 = Output                  'DDS SCLK/test LED    DDSデータ書き込みクロック
Config Portb.2 = Output                  'DDS SDATA            DDSデータ

Dim Tm As Byte                           '変数の定義
Dim Dadt As Word , Dadt1 As Word
Dim Bfofreq As Single                    'BFO freq  浮動小数点型を使ってみました
Dim Wk1 As Byte , Wk2 As Byte            '汎用 レジスタ
Dim Wchr As String * 6

Rem    ------------              電源投入時のイニシャライズ
Reset Portd.7                            'DDSの書き込みクロックを “L”
Set Portb.7                              'Fsyncを “H”
Portd = &H01                             'SW入力のプルアップ
Cls                                      'LCDの表示クリア
Rem    ------------
```

```
Rem    ------------              メインプログラム
Start:
   Tm = Pind.0                          'SWの状態を読みます
   Bfofreq = Bfof                       '周波数を表示するために数値を変数に代入
   If   Tm = 0 Then                     'もしSWが押されたら
      Gosub Wddsdata                    'DDSに設定データを送出
      Wchr = Str(bfofreq)               '周波数の数値を文字に変換して
      Wchr = Format(wchr , "00000" )    '文字変数をフォーマットして
      Cls                               'LCDをクリアして
       Lcd "BFO= " ; Wchr ; "kHz"       'LCDに表示
   End If
Goto Start
Rem    ------------              メインプログラム

Rem   ------ set DDS data  -----サブルーチン
Wddsdata:
     Dadt = &H2000                      'control word  を用意して
     Gosub Ddsdata                      '16bitを送出
     Dadt = Bfofql                       'lower data  を用意して
     Gosub Ddsdata                       '16bitを送出
     Dadt = Bfofqh                      'upper data  を用意して
     Gosub Ddsdata                      '16bitを送出
Return

Rem   ------ write DDS data  -----サブルーチン  前記の6．5．1  3)の事項を忠実に実行する。
Ddsdata:
    Set Portd.7                         'SCLKを  "H"
   Reset Portb.7                        'FSYNCを  "L"
   For Wk1 = 1 To 16                    'データbit16個をMSBから順にSDATAに送出
       Rotate Dadt , Left , 1           '①
       Wk2 = Dadt And &H0001
       If   Wk2 = 1 Then                'もしデータが  "1"ならば
           Set Portb.2                  'SDATAを  "H"  にする
       Else                             'そうでなければ(データ=0)
           Reset Portb.2                'SDATAを  "L"  にする
       End If
       Reset Portd.7                    'SCLKを  "L"  データがDDSに取り込まれる
       Set Portd.7                      'SCLKを  "H"  SCLKを元にもどす
   Next Wk1
   Set Portb.7                          'FSYNCを  "H"
Return

End
```

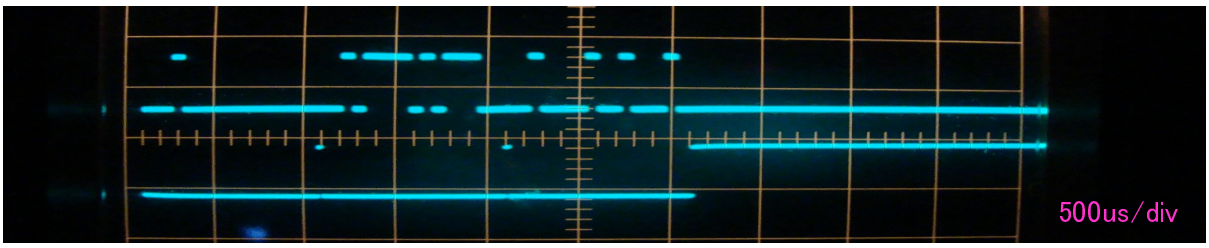
①16bitのMSBがLSBに移動、15～0bitは左にシフト。

この程度のプログラムで1482byteメモリを消費しています。 BASCOM AVR の無償版の4kbyteのおよそ1／3強です。

6）波形観測

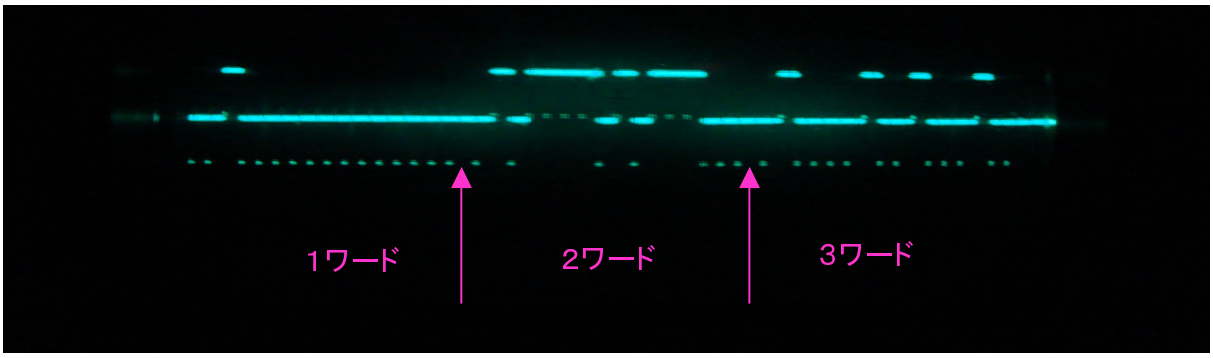
本来ならば実際にICを接続して確認するのですが、今回ICがなかったので波形を観測します。

① 全体の波形



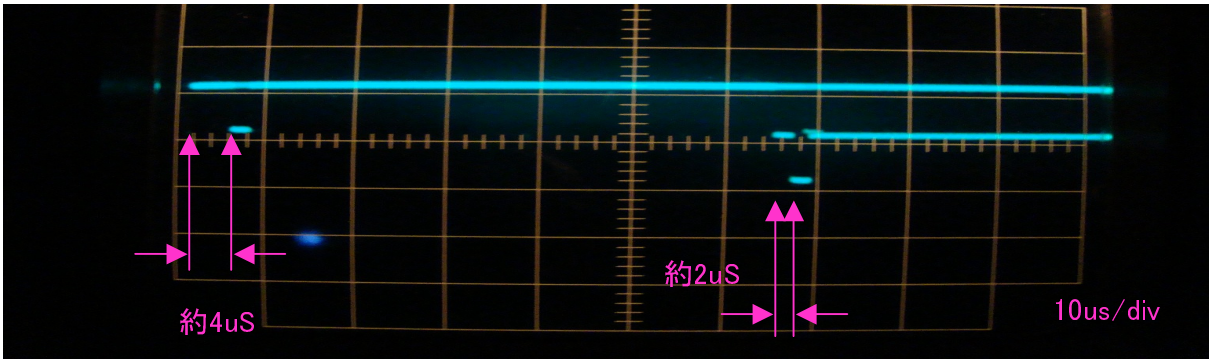
3ワードの送出に約3ms程度かかっています。 人が関わる場合は十分です。（ダイアルを回すなど）
最近使用されるような 1000パルス／回転等というロータリーエンコーダなどでは、速く回転するとミスカウントする場合があります。 この時のマイコンのクロックは内部発振器で 1 MHzでした。

② 送出データの確認



SDATAとSCLKをオシロスコープのADDmode（ch1とch 2の加算モード）を使用して観測。 データとクロックの様子が分かります。 すなわち SCLKの抜けているところ（良く見ると実際はレベルがシフトしている）がSDATAの“H”なので、左からSCLKを数えてみると、1ワード目から &H 2000 、 &H 5EB8 、 &H 4244 となっている事が確認できます。 このような安物のディレイスイープのないオシロでも十分役立ちます。

③ データセットアップ・ホールド



同様にSDATAとSCLKをオシロスコープのADDmodeを使用して観測。
データの立ち上がりでトリガをかけるとこのようになります。 立ち上がりと下がりで データークロック間のセットアップの時間が変わっています。 これはソフトの動きが違っているためです。
立ち上がりで 約4us 下がりで 約2us 程度です。 これはICのセットアップを十分満足しています。

６．６　プログラムの基本構造

プログラムは基本的に次のような形になります。

定義　：
　プログラムの内部や外部で使う条件をコンパイラに知らせるための記述
　・マイコンの定義ファイル
　・使用クロックの周波数
　・IOポートの定義

変数、定数の宣言　：
　プログラムの内で使用される変数、定数はすべて使用前にその名前、型、(定数等では)値が宣言されていなければなりません

ハードウェアの初期設定　：
　電源投入時に最初に実行しなければならないこと
　・出力ポートの初期の状態の設定　（必須）
　・表示部のクリア

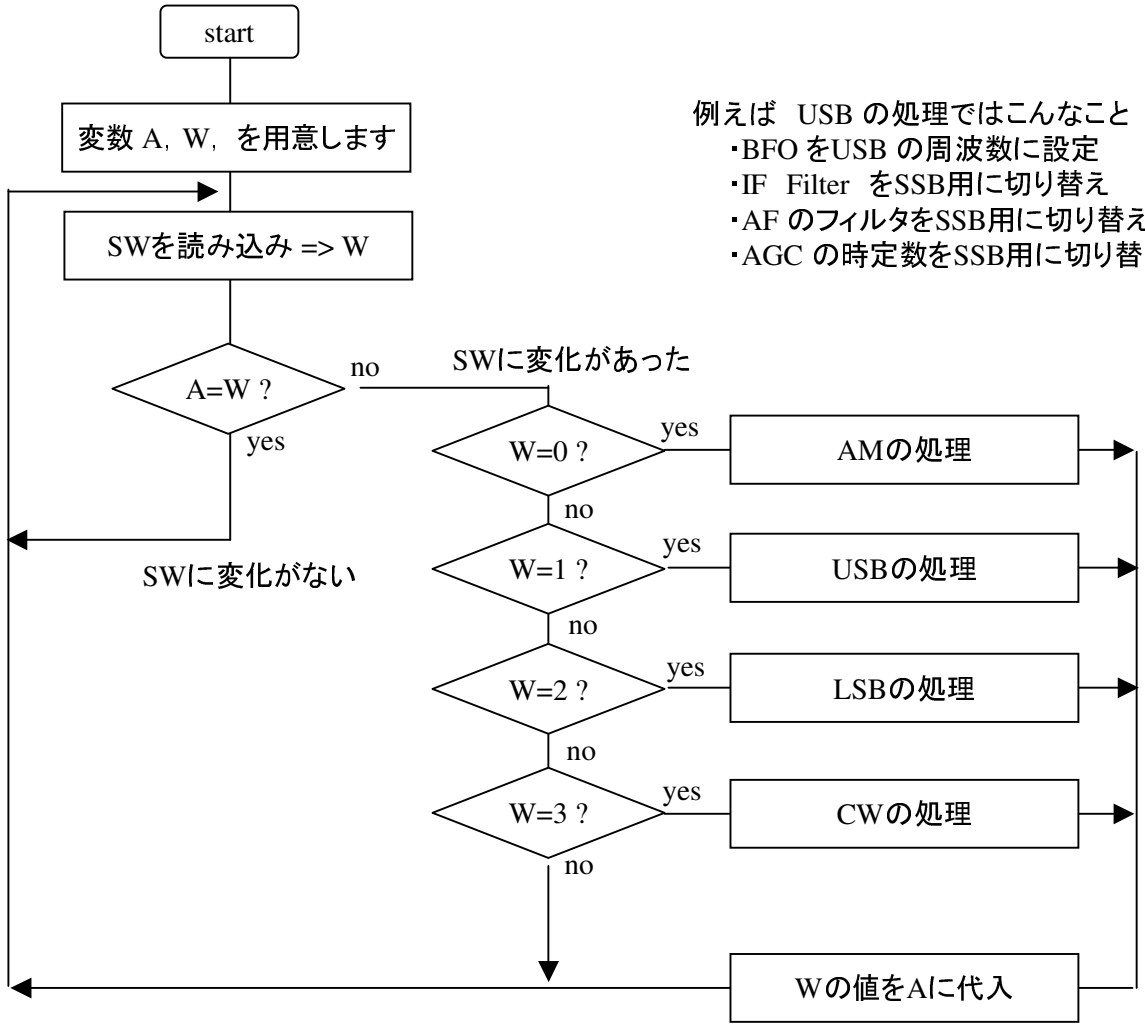
動作させたいプログラム　：
　・プログラムの本体
　・サブルーチン群

６．７　制御プログラム

今回のマイコンの仕事は機器を制御することです。 この場合はどの様にするかというと御用聞きです。仕事がないかと1軒1軒聞いて回ります。 仕事があった場合はその仕事を行い仕事を終了すれば再び聞きに回ります。 これを永遠に続けます。

例えば受信機でEmission　SWで、AM、USB、LSB、CW を切り替える事を考えます。
SW入力は4ポジションなのでIOポートは2bitあればよいでしょう。 AM : 0　USB : 1　LSB : 2　CW : 3　の信号がマイコンに入力されるようにハードウェアを設計します。
そして、プログラムは次のように動作させます。（次ページ）
・通常は　A=W　をチェックするループを実行しています。
・SWが切り替えられた時(A<>W)にそれを検出して、1回だけSWのポジションに応じた設定を行います。
それ以上は特に説明は不要でしょう。

Emission　SWで、AM、USB、LSB、CW を切り替えるプログラム



例えば　USB の処理ではこんなこと
- BFO をUSB の周波数に設定
- IF　Filter　をSSB用に切り替え
- AF のフィルタをSSB用に切り替え
- AGC の時定数をSSB用に切り替え

６．８　無償版の若干の注意

タダということで若干の注意があります。

1)コンパイラにバグがあります。

http://www.ne.jp/asahi/shared/o-family/ElecRoom/AVRMCOM/BASCManu/BASCManu.html

http://www.a-phys.eng.osaka-cu.ac.jp/hosoda-g/site1/BASCOM_Kuse.html

わかっているバグ全部ではないかもしれませんが、多くがこれらのHPに掲載されています。

2)変数に浮動小数点型を使用したり、命令に超越関数(三角関数やlogなど)を使用すると
一気にプログラム領域(メモリ)を消費します。 それ自身が複雑なので処理が多いからなのです。
次の例(次ページ)は浮動小数点の掛け算とLOG(自然対数)と表示だけです。
1872Byte消費します。 もう、無償版の制限4kbyteの半分近いです。
しかし、私はこれをアセンブリ言語でプログラミングする気にはなりません。（やって見てください）
超越関数などは一度でも“使う”ということでそのプログラムが組み込まれるので、その後式が
何回も出てきても極端にメモリを消費することは無いようです。 これは作ってみないとわかりません。
従って多く数値計算をさせる場合は製品版を使うか、“C”等に乗り換えたほうが良いと考えます。
“C”は無償でメモリFULLまで使えるコンパイラがあるはずです。(確認はしていません)
コントローラなどとして使用する場合は、実例で示す通り、結構な仕事をしてくれます。

浮動小数点や超越関数を使ったプログラム例

```
Rem  kansuu4

$regfile = "m328pdef.dat"
$crystal = 1000000

Config Lcdmode = Port
Config Lcdbus = 4
Config Lcdpin = Pin , Db4 = Portb.2 , Db5 = Portb.3
Config Lcdpin = Pin , Db6 = Portb.4 , Db7 = Portb.5
Config Lcdpin = Pin , E = Portb.1 , Rs = Portb.0
Config Lcd = 16 * 2

Dim A As Single
Dim B As Single
Dim Ans As Single

Cls

Start:
      A = 2.5
      B = 3.1
      Ans = A * B
      Ans = Log(ans)
      Lcd "ANS= " ; Ans
Stop
End

Rem   メモリ消費  1874 byte
```

ちなみにこの答えは　小数点以下6桁程度までは電卓と同じ表示になりました。

7.　プログラムをマイコンに書き込む

7．1　プログラムをマイコンに書き込む方法

自作の書き込み器を使用します。 書き込み器とそのプログラムの扱いは拙稿 “AVR書き込み器”に述べてありますのでご参照ください。
またインターネットでHIDaspxで参照するといろいろな情報が得られます。 いちどは目を通してください。
純正の書き込み器を使用する場合はインターネットで調べてください。（すみません持っていませんので）

1）DOS窓、BASCOM　Edit窓、BASICのプログラムのある窓を開きます。（次のページ参照）
2）書き込みプログラムは　hidspx.exe　を使用します。　そのプログラムの存在するファイルを開けて
　hidspx.exe をDOS窓に　D&D　します。
3）DOS窓をアクティブにして、キーボードからsp（スペース）をキーインします。
4）BASICのプログラムのある窓にはコンパイル時に生成された　xxx.hex　のファイルがありますので
　それをDOS窓に　D&Dします。
5）DOS窓をアクティブにして、キーボードからEnterをキーインします。
　すると、AVRに転送が始まり、数秒間で終了します。 終了直後に転送したプログラムが動作します。
これでコンパイル後初めてのプログラムの実行ができました。

デバッグする場合は、
1）BASCOM　Edit窓でプログラムを修正、コンパイルします。
2）DOS窓をアクティブにして、F5を押すと直前のコマンドが再度セットされて前項5）の状態になります
　ので、そのままEnterをキーインします。
　するとAVRに転送が始まり、数秒間で終了します。 終了直後に転送したプログラムが動作します。
3）まちがった操作をしても前のいくつかのコマンドが残っている場合がありますのでF5を再度押して
　みてください。

この1）、2）を繰り返す事によって短時間でソースプログラムの変更=>実機での動作 が可能になってデバッグできます。

プログラムのシミュレータもありますが（BASCOM窓でシミュレーション可能）、ソースプログラムを変更し、実機で確認できる方がやりやすいように私は感じました。

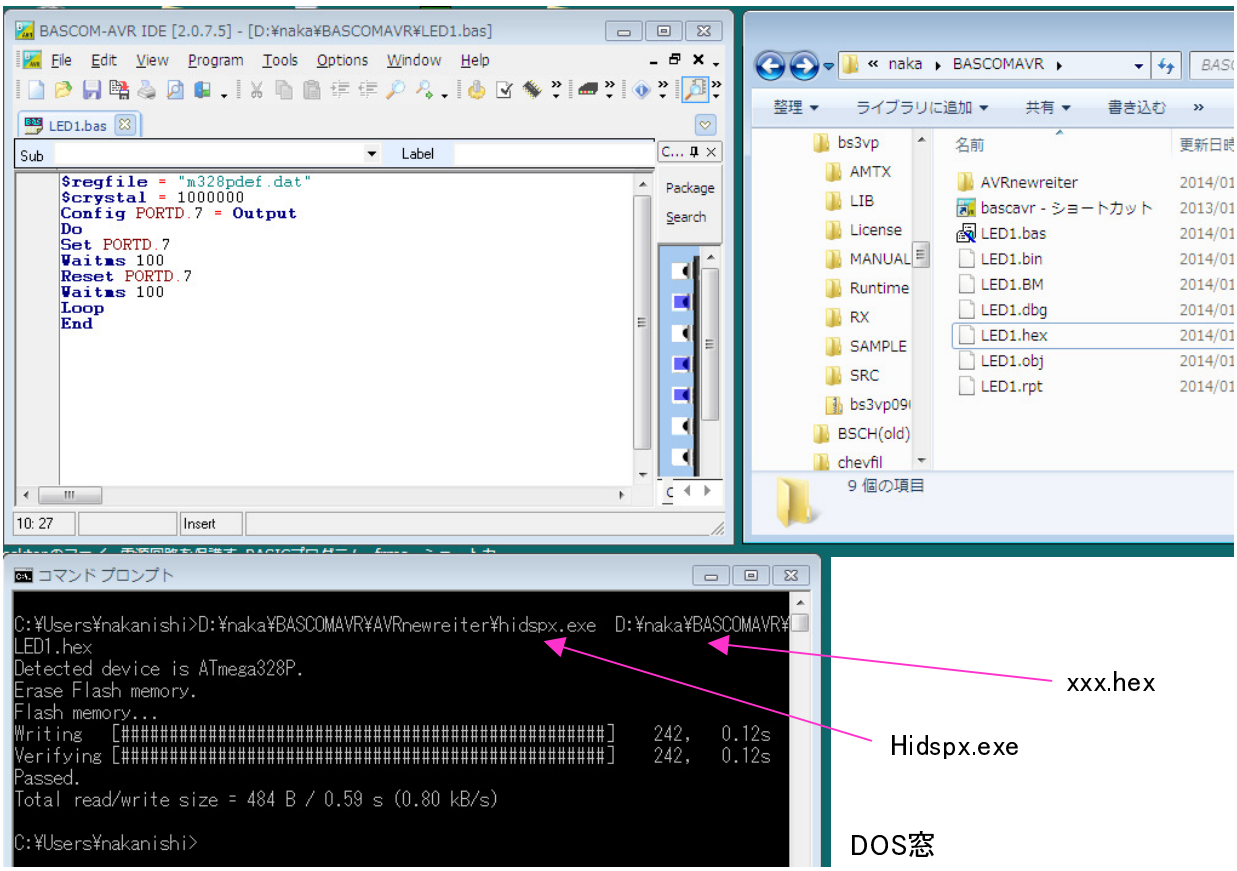
7．2　ヒューズビットについて

　AVRにはプログラムを書き込む以外にヒューズビットをプログラムしないといけません。
ヒューズはAVR のいくつかの動作モードを決めます。　例えばクロックに外部発振器か内部発振器を使うかとか、クロックを分周するとか、その他があります。
インターネットで “AVR　ヒューズ” で検索しますと、関係情報がたくさん出てきます。
そちらを参照ください。
hidspx.exe　ですと　***¥hidspx.exe　–rf　で現在設定されている状況が読み出せます。
後出のようなリストが出ますが、詳しい説明を読まないと何だかわからない場合があります。

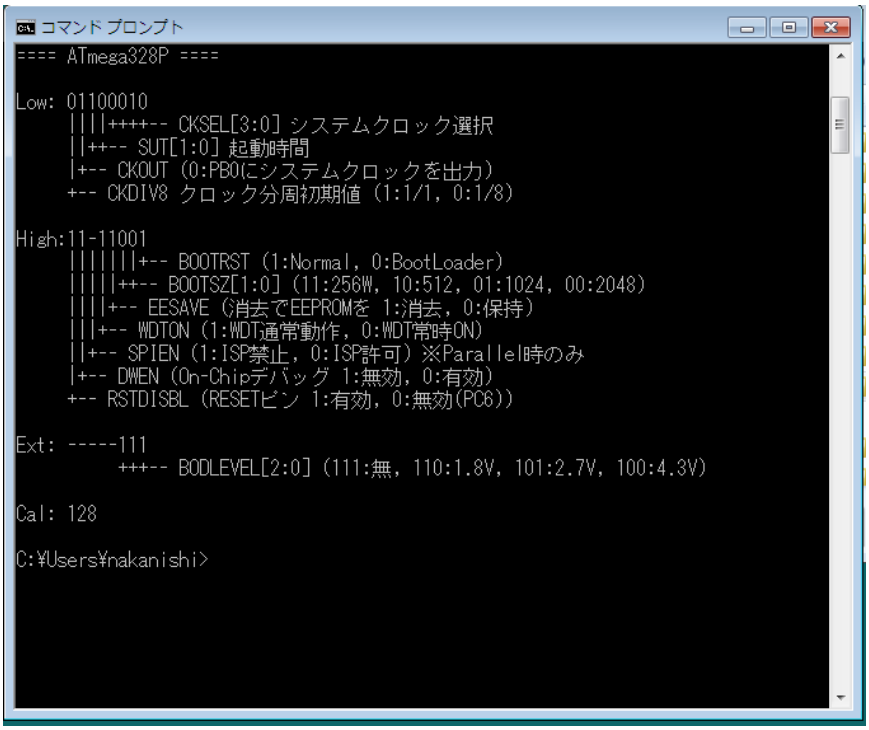
AVRの書き込みの様子

BASCOM　窓

BASICのプログラムのある窓



ヒューズの内容の読み出し結果



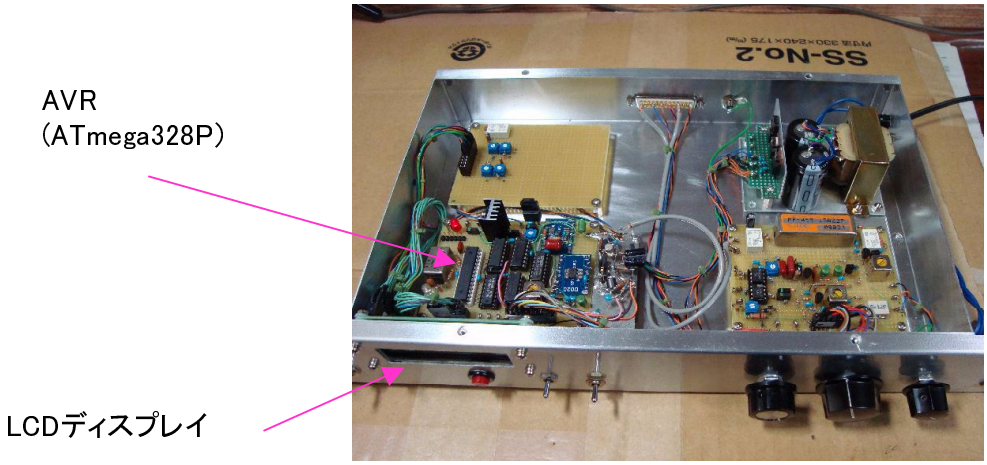
8.　実例

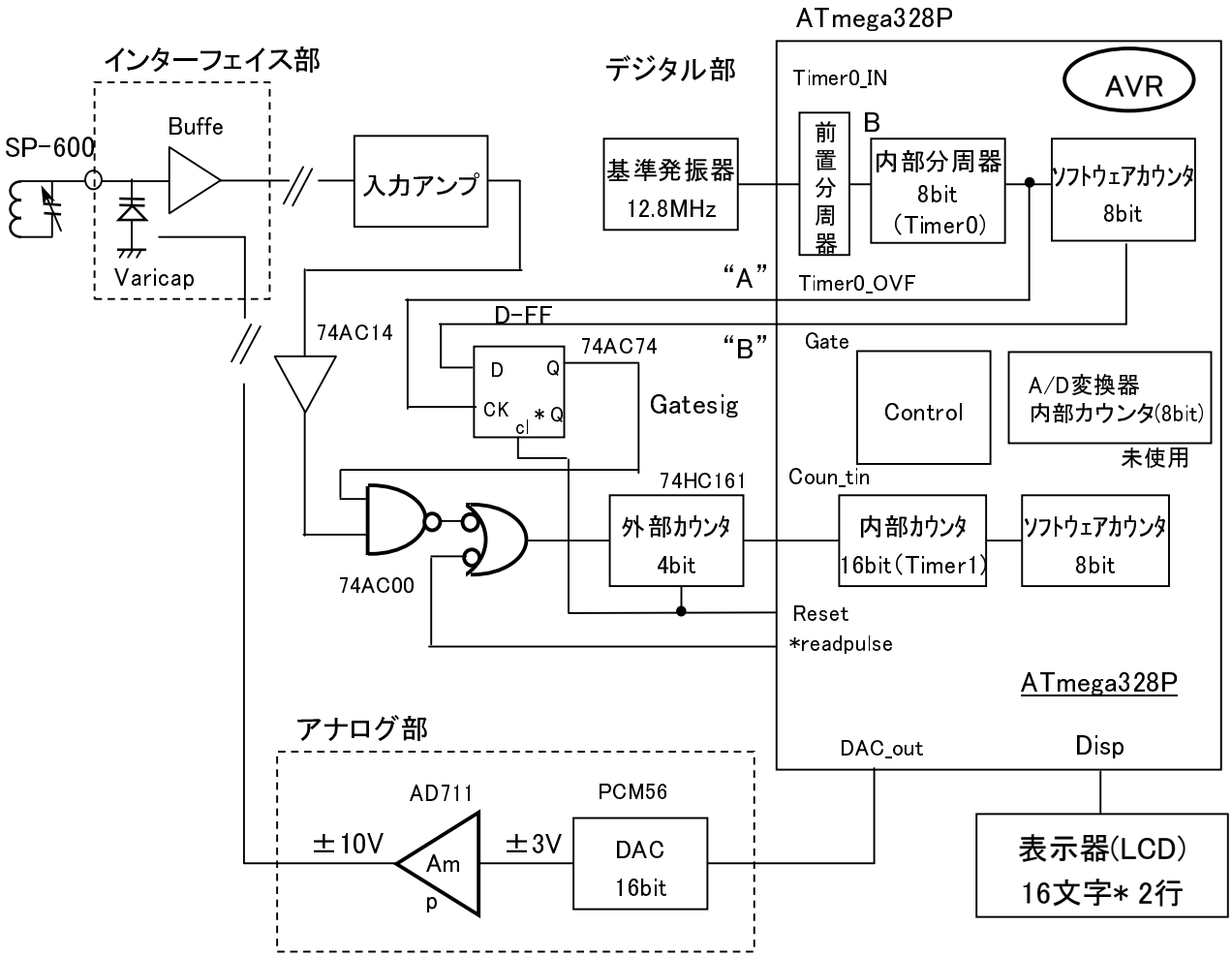
ハマーランド社製の受信機(1953年製)に付けたSSBアダプタです。

以前書きましたので、詳しくは拙稿を参照のこと。

- ・カウンタ機能
- ・BFOに使っているDDSのコントロール
- ・AFC系のコントロール
- ・AFC用のDA変換器のコントロール
- ・LCD表示

主な機能はこれらで、4kbyteギリギリでした。





D.　感想

1）そうはいってもやはり使えるようなプログラムにするためには、少し勉強が必要だと思います。
インターネットでいろいろな方がプログラムを作っていますので、見ましょう。
結構わかりずらかったりしますが、やさしいと思われるものを選んで見ましょう。

2）作ったプログラムが思ったように動かないのは、ほとんどが（全部とは言わないが）本人のミスです。
・命令をまちがって理解している。
・変数の型をまちがえている。 代入などで小数点以下が切り捨てられているなど。
・処理自体の考え方がまちがっている。
こんなところが多いようです。
コンパイラでは、
・取説などにおける説明不足。
・実際にバグがある。　はっきりバグとして発表されているものは、それに従うこと。

3）命令の動作がわからない時には、実際に使ってテストしてみること。

4）もし初心者でしたら、プログラムの経験者と友達になりましょう。 知っている人がみれば自明でも分からないで調べまくる事は大変エネルギーと時間を使うことになります。
どんどん聞いてうるさがられましょう。 知っている人はちゃんと教えてくれるものです。
BASICなんか使ったこと無いよといわれたら、命令の説明を提示して読んでもらいましょう。
まがりなりにもプログラムの経験者ならば、BASICを使ったことが無くても命令の説明を読めば分かるように解説することはできるでしょう。

5）ちょっと凝ったなプログラムを書くと、やはり4kbyteの制限に引っ掛かります。
表示にバリエーションを持たせようとか、超越関数、便利な命令（キーボードSWとか）を多用するとあっという間です。 その意味では“簡単に動くよ!、大きなプログラムは無理だよ！” と言うのは2階へあげて梯子をはずすような感じが若干ありますが、プログラムやそれを使ってハードウェアを動かすイメージができればここでは大成功です。
他の言語なども似たような感じですので無駄にはならないと思います。 少なくともハードウェアの変更なしに他の言語に移行はできます。
“C”は無償で全プログラムメモリを使用できるようなものがあるはずです。
インターネットで検索してください。 例えば　AVR　C言語　等です。 いっぱい出てきます。

6）“C”がわかっている人はBASICなど使う必要はないのですが、慣れていない方にはBASICが良いのかなと、私が思って勝手に思ってこんな文章を書いてみました。

7）プログラムは一生懸命考えて、作って、バグを退治して、動いたときが楽しいのです。

以上

Homepageの表示をクリックしてもそのページにに移行しない場合があります。 私の未熟で理由がわかりません。
その場合はお手数ですが手入力でお願いいたします。