[第1章]

開発ソフト，プログラマは低価格だがAVRは高機能マイコン

AVRを知るにはまず動かしてみよう

1-1　AVRマイコンとは

AVRはアトメル社の開発した8ビットRISCタイプのMCU（Micro Controller Unit）です．

アトメル社は1984年にアメリカ合衆国のサンノゼに誕生しました．社長はギリシャ系アメリカ人のジョージ・パレゴス氏で，当初世界最速のEPROMでデビューし，その後も不揮発性メモリ分野で特色ある製品をリリースしてきました．パラレルEEPROMでは，世界で最初にエラー・ディテクション・コレクション機能を搭載し10万回の書き換えを可能にした製品を商品化，EPROMでやはり世界で最初に低電圧3V対応の製品を出したのもアトメル社です．また，ベンチャ企業には珍しく，自社ファブ（製造工場）を早い時期からもったことでも有名です．現在アメリカ，ドイツ，フランス，イギリスなどにIC製造の前工程の生産拠点をおき，デザイン・センタも世界各地にもつようになりました．アセンブリに関しては東南アジアの協力会社に依託しています．

AVRもこの展開の一つで，現在アトメル社のノルウェー工場で設計開発が行われています．ここ2～3年，MCU業界では世界でもっとも出荷数の伸び率が高いと評価され，2004年度は年率40％を超える勢いで世界中の組み込み製品に採用されてきています．アトメル社の製品ラインナップの中でも3分の1を占めるほどになってきました．現在アトメル社には，32ビットMCUとしてARMコアをもつファミリ，8ビットはAVRと8051コアをもつ製品，4ビットにMARC4などの製品がラインナップされていますが，自社製のASIC，ASSP製品のほとんどはAVRとARMコアを内蔵して作られています．

AVRはノルウェーの大学生二人が卒業研究で開発し，その後この二人ごとアトメル社へ移籍し，アトメル・ノルウェーとなり，ノルウェー工場で開発されていることはよく知られていることです．AVRの名前の由来もその二人のイニシャルからとったとする説が有力です．

1-2　AVRを動かしてみよう

AVRのことを知ってもらうには，いろいろ解説しても始まらないと思います．とにかく少しでも動かしてみる，動かしてみればいかに使い勝手がよいかがわかってもらえると信じています．そこで，いきなりですが，AVRを動かすところからはじめます．動かすといってもハードウェアを持ってい

ない人もいるわけですから，ここでは，誰でも無償で使うことのできるフリーウェアでAVR開発のためのWindowsパソコン上で動作するAVR Studioを使って，クロス統合化開発環境上でAVRの動きを追ってみることからはじめます．AVR Studioは，アトメル社のホームページからダウンロードすることができます．

続いて，STK500というアトメル社の供給しているスタータ・キットを使用して，AVRにプログラムを書き込んで走らせる手順を説明します．STK500を持っている人は実際にデバイスを動かしてみることができます．AVRを実際に動かすためには最低限書き込み器が必要です．もしSTK500を持っていない場合は，AVRISPと呼ばれる簡易書き込み器を入手されることをお勧めします．書き込み器をわざわざ自作する(書き込み器の製作から始める解説書を多くのMCUで見受けるが，これはAVRに関する限りむだであると思う)よりも手間がかかりませんし，とても低価格で入手できます．これとチップを入手して，ユニバーサル基板に組み立てれば，ここでの話はすぐに実行できます．

その昔パソコン上でアプリケーション・プログラムを書いて走らせようとして暴走し，システムを壊してインストールをしなおすなどということを経験された方もいるかもしれません．AVRを開発，デバッグする際にそのようなことはほとんどありません．ほとんどというのは微妙なところですが，いくつかの注意点を守れば，デバイスを壊したり，ツールを壊したりするケースはまずありえないので，どんどん自分の思いついたアイデアを実機でチェックしてもらいたいと思います．もし何らかの操作ミスで動かなくなっても，ほとんどのケースでは修復できます．

1-3　AVR Studioのインストール

まず，AVR Studioをインストールします．アトメル社のサイト(`http://www.atmel.com/products/avr/`)へアクセスし，左端のメニューからTools & Softwareをクリックし，Design Softwareの欄からAVR Studio4をクリックするとSoftwareの欄にAVR Studio4.xxの項目が現れますから，ディスク型のアイコンをクリックしてプログラムをダウンロードします．.xxのところは番号が大きいほうが新しいバージョンです．この際，保存をクリックして適当なフォルダを作りダウンロードしておきます．

AVR Studio4をダウンロードする際，その近くにサービス・パック(SPx)の表示がある場合，一緒にダウンロードしておきましょう(**図1-1**参照)．

ダウンロードしたファイルをダブル・クリックして自動解凍します．パソコン型のセットアップ・アイコンをダブル・クリックすると，セットアップ・ウィザードが開始されます．これからは画面の指示に従ってインストールを行ってください．Windows 2000やXPなどでAdministratorとユーザを分けて利用されている方は，必ずAdministrator特権でインストールしてください(インストール後はユーザ・モードで動く)．

途中ライセンスの許諾の可否についての問い合わせ画面が出てきますが，ひととおり読んで問題がなければ，I　accept…をクリックしてさらにNextをクリックします．フリーウェアなので，使用する分には問題はないですし，配布もOKのようですが，ソフトの改造や中身の解読は許されていないようです．

また，USBのドライバをインストールするか否かの問い合わせも途中で現れますが，JTAGICE-

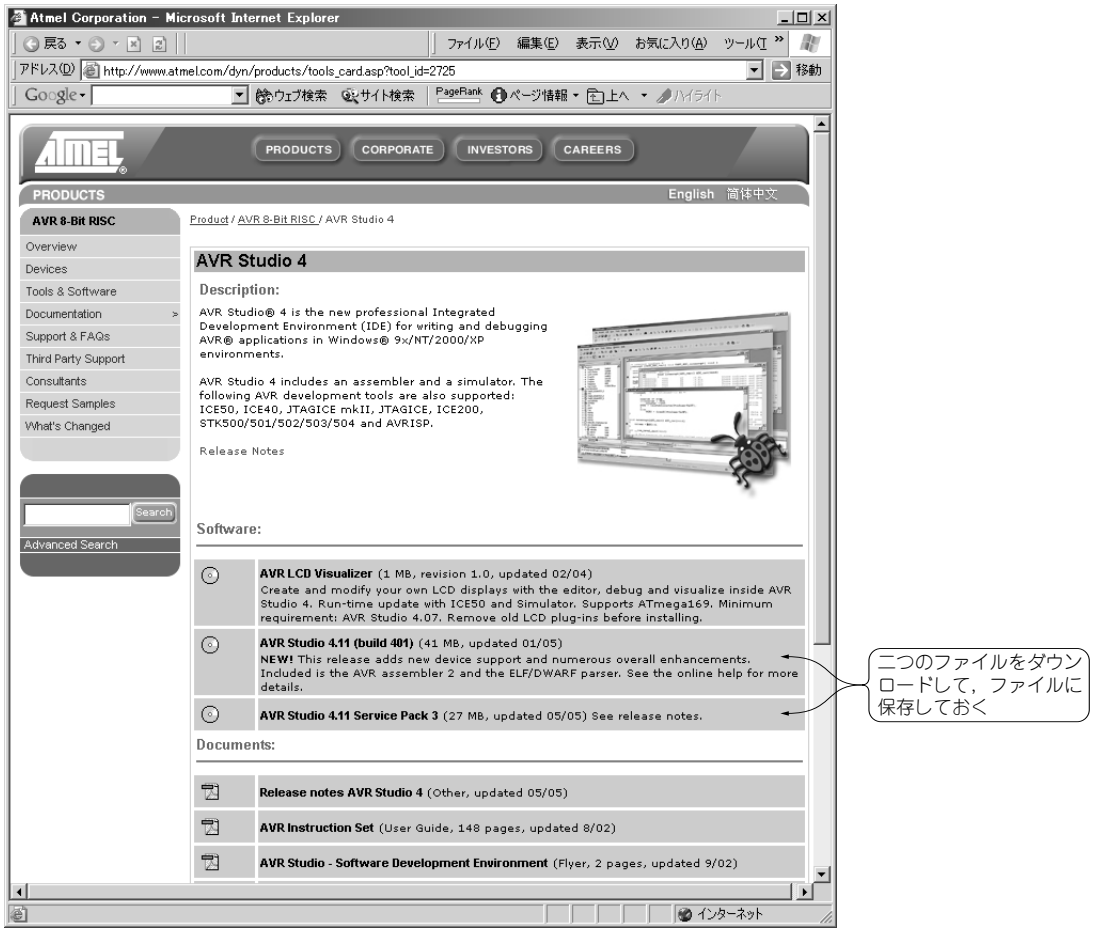

図1-1　アトメル社のAVRStudio ダウンロード・サイト

mkIIやICE40/50を使うことがあるようであれば，同時にインストールしておくとよいでしょう．使えないOSもあるようなので，メッセージに注意してください．必要に応じてシステムを一度再起動してから，サービス・パックのインストールを行います．アイコンをダブル・クリックして画面の指示に従ってください．

1-4　アセンブラでのプログラミング

アセンブラでプログラムを作り，オブジェクト・コード（実際にAVRに書き込むためのコード）を生成するためにはAVR Studioを使用して行います．

プロジェクトを作る

パソコン上で，スタート→（すべての）プログラム→ATMEL AVR Tools→AVR Studio4.xx をク

図1-2　AVR スタートアップ画面

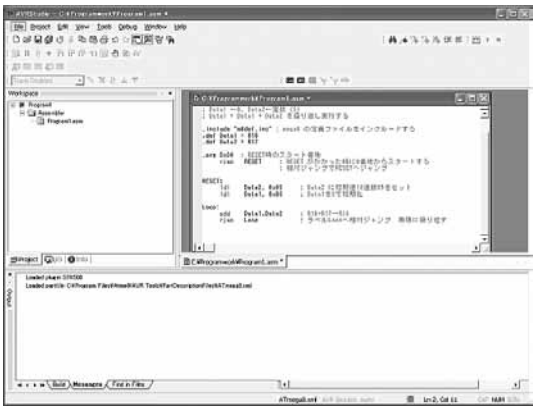

図1-3　アセンブラ・ソースの入力画面

リックしてAVR Studioを立ち上げます（デスクトップにショートカットを作っておくと便利）．**図1-2**のような画面が表示されます．

このWelcome to AVR Studio ウィンドウ（正しくはProject Wizardという）から新しくプロジェクトを起こす場合は，Create New Project（既存のプロジェクトを開く場合はRecent Projectsの中からファイルを選択しOpen）をクリックすると，次のウィンドウが現れます．

Project Nameにプロジェクト名を入力し，Locationのディレクトリを指定します．新しいフォルダを作ることもできるので，トレーニング用のフォルダを作っておきましょう．入力がすんだら，Next>> をクリックします．ここではシミュレータでのシミュレーションを行うので，Debug PlatformのリストからAVR Simulatorを選択して網掛け状態にしておきます．

Device欄からデバイスを選択します．これからプログラムを書くためのデバイスです．この章ではATmega8を用いるのでATmega8をクリックし，Finishをクリックします．次の画面が表示されます（**図1-3**）．中央の …program1.asmの編集ウィンドウにアセンブラのソース・コードを入力して，ソース・ファイルを作成することができます．

● プログラムの入力

図1-3の中のC:¥Programwork¥Program1.asmに示されるように，**リスト1-1**の内容を入力してみます．“；”はコメント行で，半角英字で入力しておけば，これ以降の行には，全角漢字も入力できます．改行した後は半角英数字に戻さなければなりません．

それではとりあえず，入力してみてください．.で始まるステートメントは擬似命令と呼ばれるもので，実際にコンピュータは実行しませんが，プログラムをコントロールするための命令です．AVRのアセンブラでは，16進数を表現するときに先頭に0xとつけてその後に10C5などの16進数を続けて記述します．すなわち0x10C5と書くことで16進数がアセンブラに伝えられます．$10C5と書いても同じ意味になります．インテル系で使われる表現，010C5Hなどは受け付けられません．

入力が終わったらファイルをセーブします．Fileメニューから Saveをクリックします．

リスト1-1　加算機能を確認するサンプル・プログラム

```
; Program1
; AVRStudioを用いてもっとも単純なプログラムを作る
; Data1 ←0，Data2←定数(5)
; Data1 = Data1 + Data2を繰り返し実行する

.include "m8def.inc"                    ; mega8の定義ファイルをインクルードする
.def Data1 = R16
.def Data2 = R17

.org 0x00 ; リセット時のスタート番地
     rjmp    RESET                      ; リセットがかかったときに0番地からスタートする
                                        ; 相対ジャンプでRESETへジャンプ

RESET:
          ldi     Data2, 0x05           ; Data2に初期値16進数05をセット
          ldi     Data1, 0x00           ; Data1を0で初期化

Loop:
          add     Data1,Data2           ; R16+R17→R16
          rjmp    Loop                  ; ラベルLoopへ相対ジャンプ　永久に繰り返すプログラム
```

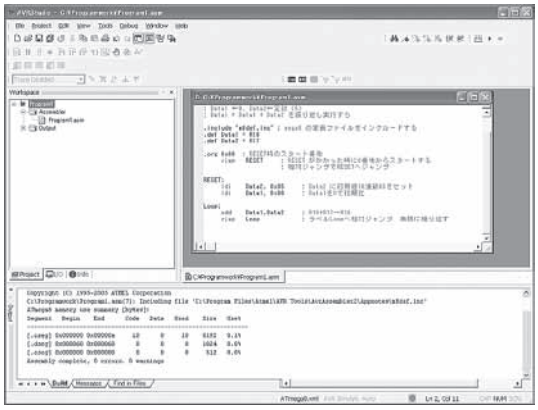

図1-4　アセンブル後の表示

● アセンブル

次に，アセンブルを行います．アセンブルはBuild→Buildをクリックします．何も誤りがなければ，画面の下のほうにあるOutputウィンドウに`Assembly complete, 0 errors 0 warnings`と表示されます(**図1-4**)．

もしスペル・ミスなどがあれば，赤色の表示とともにError Messageが表示されます．どこがエラーかを知るには，メニュー・バーのProjectをクリックし，プルダウン・メニューからNext Error (F4)をクリックすることで，エラーのある行にカーソルを移動させることができます．スペルなどをチェックして修正を加えます．修正後再度Buildでアセンブルし，エラーがなくなるまで修正とアセンブルを繰り返します．

1-5　AVRをシミュレータで動かしてみよう

ここからは，シミュレーションを行ってプログラムの動きを見ていきます．Debug→Start Debuggingをクリックすると，Work Spaceの表示がレジスタおよびI/Oレジスタ表示に変わります．Register16-31の前にある“＋”をクリックしレジスタを展開しておきます(**図1-5**)．さらにProcessorも同様に展開しておきます．

これでシミュレーションの下準備の完了です．

● ステップ実行

Debug→Step Intoをクリックしてみます．ソース・ファイルの黄色い矢印が移動したはずです．プログラムが1ステップ実行され，次に実行される命令を矢印は示しています．Work SpaceのProcessorの下にあるProgram Counterの値も0x0000から0x0001へ変わります．rjmp命令を実行したことで2サイクルかかったことがCycle Counterに示されています．さらにFrequencyが4 MHz，Stop Watch　0.5 μsであることが示されます．4 MHzでの1サイクルが0.25 μsなので，2クロック・サイクルで0.5 μsでrjmp命令が実行されたことがわかります．

さらにStep intoをクリックしていくと，1ステップずつ命令を進めることができます．Step Intoの代わりにF11を押すことでもステップを進めることができます(4 MHzの設定になっていないこともある．そのときは，Debugメニューから AVR Simulator Optionsを選択して周波数の変更を行うことができる)．

何回かステップを進めていくときにWorkspaceのR16，R17も注目します．ldi命令でデータを

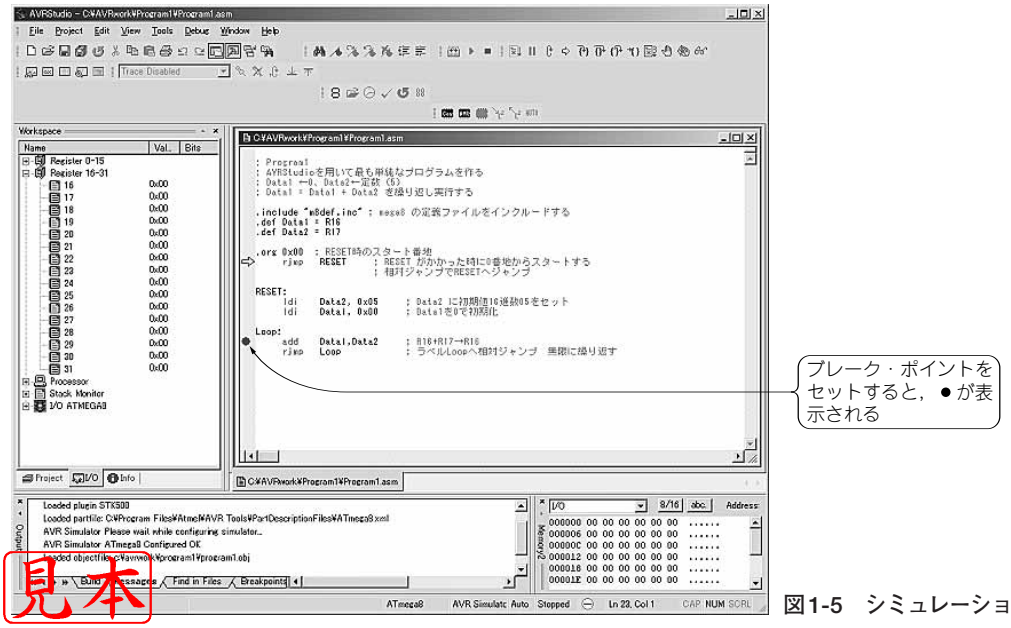

図1-5　シミュレーション画面

ロードし，add命令で加算された結果が表示されていきます．16進数で表示されているのでわかりにくいですが，立派に演算を実行していることがわかります．

Debug→Auto Stepを次にクリックすると，繰り返しステップが実行され，そのつどWorkspaceにアップデートされた結果が表示されます．停止するには，Debug→Breakをクリックします．

● ブレーク・ポイント

Debug→Runでプログラムを走らせると，ストップした時点で結果を表示します．ブレーク・ポイントを検知した時点かDebug→Breakなどで，強制的にストップさせることができます(**図1-6**)．

ブレーク・ポイントは，ソース・ウィンドウのステートメントにカーソルを置いて，左マウス・ボタンをクリックし，ツール・バーからDebug→Toggle Breakpointをクリックするか，F9を押す，あるいはマウスの右ボタンを押してプルダウン・メニューからToggle Breakpointを選択してセットできます．トグルになっているので，押すたびにセット→リセットを繰り返すことができます．

設定されたブレーク・ポイントの上にカーソルを当て，右マウス・ボタンをクリックすると，プルダウン・メニューが開きます．ここでBreakPoint Propertiesにカーソルをあてクリックし，ブレークの条件を設定してみます．continue execution after the view have been updated.にチェックをつけてOkをクリックしてブレーク・ポイントの一つにさしかかったとき，Viewを更新し，継続してRunするモードを設定しておき，F5を押してRunさせるとレジスタの内容がブレークにさしかかるたびに更新されます．この場合，ブレーク・ポイントではブレークしません．

このように，AVRの動きを非常に簡単に見ることができます．ビット数の大きなコンピュータではアプリケーションを動かすまでの初期化という儀式だけでも大きなプログラム・ステップを取ってしまいますが，AVRはそのようなことにプログラム・スペースを費やす必要はほとんどありません．最低限必要な初期化だけでシステムを作ることができます．

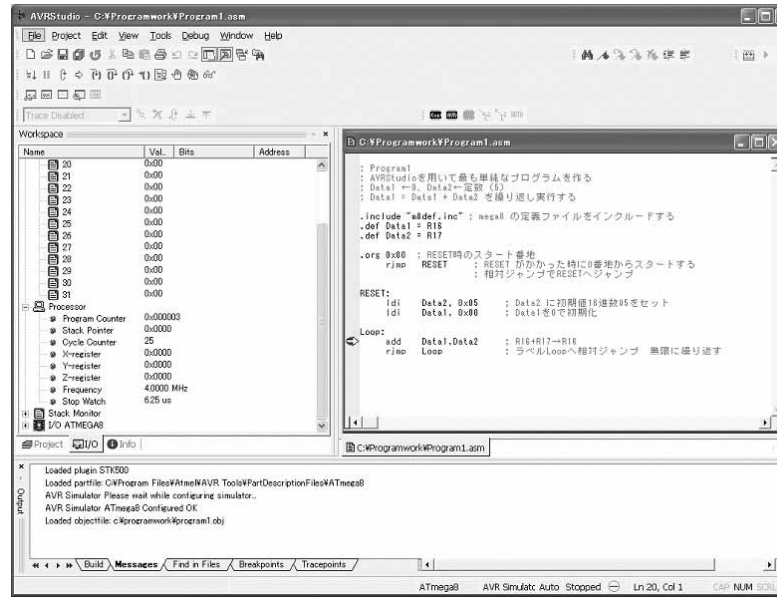

図1-6 シミュレーション中のAVRStudio

1-6　STK500スタータ・キット

前節ではシミュレーションでAVRを動かしましたが，ここでは実際のAVRチップで動作をさせてみましょう．

ただし，ハードウェアがなければ実機確認はできません．一番手っとり早い方法は，アトメル社から出ているスタータ・キット“STK500”を入手してそれで実験することです．しかし，STK500はコンピュータが普及する前の時代から考えるととても安くなったとはいえ，個人で購入される方にはまだ高価と思われるかもしれません．ここでは，STK500以外にユニバーサル基板に部品を配線して作った場合でも同様にAVRを動かして見られるように解説します．

その場合でも最低限の書き込みツールAVRISPは用意してください．

STK500は，**図1-7**のようにボード上に40ピンまでの数種類のDIPソケットが並べられているボードです．この上には，さらにISP〔イン・システム・プログラム：AVRが実装されたシステム(ボード)上でプログラムをする〕用のプログラマ回路，ボードで使用するための電源電圧を発生する回路，クロック周波数を発生するクロック発生回路，デバッグをするときに利用することができる8ビットのLED表示機，やはりデバッグ時に利用できる入力信号を与えるための8ビットの入力スイッチ，外部とのコミュニケーションを行うためのシリアル・ポートなどが組み込まれています．

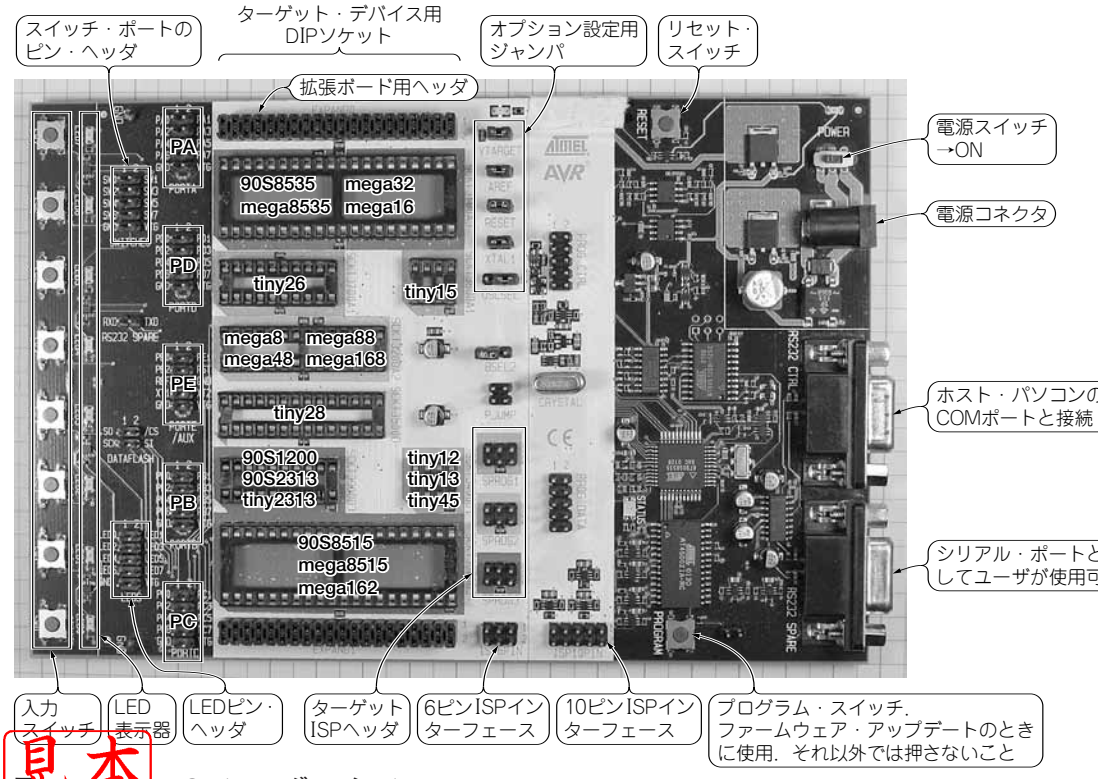

図1-7　STK500のコンフィグレーション

AVRのもつI/Oポートは，10ピンのヘッダ・コネクタへ出されているので，LED，スイッチ以外のユーザがユニバーサル基板に作ったインターフェースともケーブルを介して接続することができるように工夫されています．

1-7　STK500を動かす

それではSTK500を使う準備をします．

AVR Studioを立ち上げてNew Projectをクリックし，Program2と名前をつけて，**リスト1-2**のプログラムを入力していきます．デバイスは，ATmega8で設定していますが，STK500に付属のサンプルを使う場合，付属しているサンプルIC名でもかまいません．この場合，プログラムの中の.includeの“m8def.inc”を変更します．

リスト1-2　STK500上で動かすサンプル・プログラム

```
; Program2
; STK500のLEDを光らせる簡単なプログラムを書く
; 出力ポートの初期化
; 時間を遅らせるサブルーチンを使用するためにスタック・ポインタを
; 初期化しサブルーチン・コールを行う

.include "m8def.inc"
.def Temp = R16
.def CNT1 = R17
.def CNT2 = R18

.org 0x0000       ;
          rjmp    RESET
Delay1:   ldi     CNT1,    0xFF     ; ディレイ・サブルーチン・エントリ
Loop1:    ldi     CNT2,    0xFF     ; カウンタ値の初期化;
Loop2:    dec     CNT2              ; カウンタ値をマイナス1，結果が0になるとSREG:Z=1
          brbc    1, Loop2          ; SREGのビット1(Z)がクリアされていればLoop2へブランチ
          dec     CNT1
          brbc    1, Loop1          ;
          ret                       ; サブルーチン・リターン

RESET:
          ldi     Temp, low(RAMEND) ; RAMの最終番地をバイト単位で
          out     SPL, Temp               ; スタック・ポインタへセット
          ldi     Temp, high(RAMEND)      ; メモリ領域の終わりであるRAMENDは
                                          ; 各.defファイルの中で定義されている
          out     SPH, Temp
          ldi     Temp, 0xFF
          out     DDRB, Temp        ; PORTBのデータ・ディレクション・レジスタを出力にセット

Loop:
          out     PORTB, Temp       ; Tempレジスタのデータを PORTBにセット
          rcall   Delay1            ; ディレイ・サブルーチンを相対コール
          com     Temp              ; Tempレジスタのデータをビットごとに反転
          rjmp    Loop              ;
```

たとえばAT90S8515を使うなら，“8515def.inc”に置き換えます．AVR Studioを標準インストールした場合，ディレクトリ¥Program Files¥Atmel¥AVR Tools¥AvrAssmbler¥Appnotesフォルダの下にdefファイルが置かれており，この中から該当するデバイスのファイル名を見つけて記述します．

入力がすんだらBuildでオブジェクト・コードを作成し，シミュレーションにかけることができるので動かしてみてください．ソフトウェア・タイマに時間がかかるので，ブレーク・ポイントを「out PORTB, Temp」の行に設定し，ブレーク・ポイントのプロパティでブレーク・ポイントでビューを更新後，継続モードを設定しておくと，PORTBの変化のようすが見られるでしょう．

次は，いよいよSTK500と接続して，実デバイスでの動作の確認です．

● 電源を用意する

まず電源を接続します．電源はキットには入っていませんが，電源ケーブルは付属しています．実験用の電源を使うか，手持ちのACアダプタでも電圧とピンの径が合えば使うことができます．電圧は10～15 VのDCで，極性は問いません．電源コネクタに接続します．接続時には，電源スイッチは切っておきましょう．

● STK500とパソコンをシリアルで接続

ホスト・パソコンのシリアル・ポートとSTK500の通称RS-232C CTRLポートを，付属のグレイ・ケーブルで接続します．RS-232C CTRLコネクタはDsub 9ピン・コネクタで，ボード・エッジの中央付近にあります．

最近のノート・パソコンでシリアル・ポートがサポートされていない機種もありますが，USB-RS-232C変換アダプタなどを使うこともできます．しかし，動作が安定しないケースもあるので，できればシリアル・ポートをもったパソコンを用意することをお勧めします．

● AVRチップをソケットに差し込む

ICチップをICソケットに装着します．ATmega8を持っている人はSCKT3200A2の緑色に色分けされたソケットに装着します．ICの向きに気をつけてピンを曲げないように装着してください．ICの向きはRS-232Cコネクタの反対側に1ピン・マークを向けます．AT90S8515を使う場合はSCKT3000D3(赤色)に装着します．AVRチップとソケットの対応は第5章で説明します．

● 書き込み用ISPケーブルを接続

次に，ISP用のケーブルを接続します．6ピンのヘッダ・コネクタISP 6PINとそれぞれの装着したデバイスに相当する色に対応するSPROGヘッダ・コネクタに付属のフラット・ケーブル6芯を接続します．フラット・ケーブルはひねったりせずに，1ピン・マークをフラット・ケーブルの1ピン・マーク(ケーブルに色が塗られている)にそれぞれあわせて接続してください．ちなみにATmega8の場合，SPROG2の緑色ヘッダ・コネクタに接続します．

LEDのヘッダ・コネクタLEDSとPORTBヘッダを10芯のフラット・ケーブルで接続してください．方向などをまちがえないようにするのは同様です．

● ジャンパの設定

次に，ジャンパの設定が必要です．基本的には出荷時（デフォルト）のジャンパのままでかまいません．VT_{ARGET}，A_{REF}，RESET，$XTAL_1$のジャンパ・ポストにジャンパがささった状態にします．OSCSELは文字を正面に見て右側にジャンパされていればOKです．$BSEL_2$，PJUMPはジャンパをはずしておきます．

これで準備が完了です．

● 動作確認

AVR Studioが先ほどのRun状態であれば，Breakします．デバッグ・モードであれば，Stop Debuggingでこのモードから抜けます．

STK500の電源を投入します．このときVT_{ARGET}ジャンパの上にあるLEDが点灯します．もし点灯しなければ電源の接続やICの向き，ケーブルの接続などに問題があるので，チェックしてください．

Tools→Program AVR→Auto Connectをクリックします．STK500のウィンドウが表示されます．このときProgramボタンがアクティブでない場合は，Programタブをクリックして，Program画面を表示させます．STK500を立ち上げた直後にメッセージ・ウィンドウにエラーが出る場合，クローズド・チェック・ボックスをクリックして一度ウィンドウを閉じてから再度同じ手続きを繰り返します．このとき電源を立ち上げなおす必要があるかもしれません．

さらにエラーが出るようであれば，それぞれの接続を確認します．電源が確実に入っていなかったり，Disconnected Modeを誤ってクリックしてしまった場合，もう一度Toolsからはじめますが，Program AVRの次にAuto Connectを選択しないでConnectでポートを指定して，パソコンが割り当てているCOMポートの番号を選択してConnectします（通常はCOM1）．

● プログラムの書き込みと動作確認

STK500のウィンドウが表示されたら，デバイス・ボックスの値をATmega8（あるいは装着したデバイス名）にします（**図1-8**）．Programming modeはISPに●（チェック）が入っているようにします．マウスでクリックすれば指定できます．Filesボックスにこれから書き込むためのプログラム名を表示させます．右端の...ボタンを押すと，ファイルのリスト・ウィンドウが開かれます．ファイルの場所を指定し，表示されるプログラム・ネームをクリックします．ここでは`Program2.hex`です．

続いてProgramボタンをクリックします．どうですか？ LEDが点滅を始めたと思います．ATmega8を使っている場合8ビットの出力をしたにもかかわらず6ビット分しか点滅しないと思います．これは，点滅しない2ビット分がSTK500上では水晶発振器（オシレータ）の回路に接続されているためで，もし内部RCオシレータを使用するように回路接続をしてあるボードを使用すれば，すべてを点滅させることができます．

ここで注意することは，STK500上の「PROGRAMスイッチ」は絶対に押さないことです．通常STK500を使うとき，このスイッチを押す必要はありません．もし誤って押すと，AVRStudioと通信ができなくなることがあります．このスイッチは，AVRStudioを新しいバージョンに更新したときにSTK500のファームウェアが改訂されることがあり，このときファームウェアの内容を書き換える

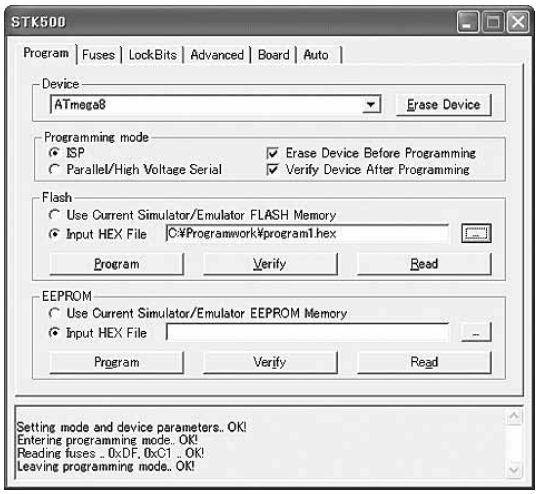

図1-8　STK500の設定ウィンドウ

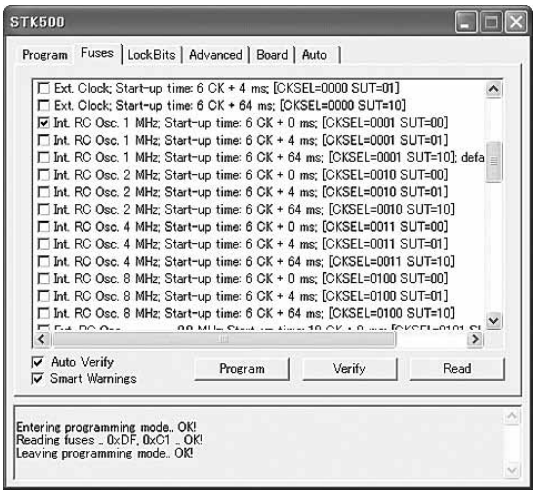

図1-9　ヒューズ設定ウィンドウ

ために押すスイッチです．

もし誤ってファームウェアを壊してしまったときには，マニュアル・ファームウェア・アップグレードという方法で修復を行うことができます．これについては，開発ツールの第5章で説明します．

引き続きFusesタブをクリックし，ヒューズ設定画面にします(**図1-9**)．

● ヒューズ・ビットの意味と変更例

ヒューズ・ビットとはAVRに特徴的な機能で，AVRのシステム機能の設定に使われます．システム・クロックに何を使うか？ デバッグ・モードは使えるようにしておくか？ ブラウン・アウト・ディテクタ(低電圧検出回路の閾値電圧)は何Vに設定するか？ など，リセット直後にAVRがどのようなセットアップでスタートするかを決める項目が含まれています．これはAVRのプログラムの中では変更することができません．項目が多いですから，右側のスライダを操作して全体を見渡しておいてください．

ここでは，internal RC Oscillatorの周波数を変えてみます．デフォルトで1 MHzになっているはずです．4 MHzと書かれたうちの一つのボックスをクリックしチェック・マークを入れて，Programボタンを押してください．どうですか？ LEDの点滅速度が非常に速くなったはずです．

これは何をしているかというと，ATmega8などmegaシリーズに搭載されている，内部RCオシレータの発振周波数をヒューズ・ビットのプログラムで変更してみたわけです．ATmega8の場合は1，2，4，8 MHzを選択することができるようになっています．残念ながら古い設計のAT90Sシリーズは内部RCオシレータをもっていません．したがって，AT90S8515で実験した場合はこの周波数を変える実験はできないことになります．

● プログラムの内容

ここでリスト**1-2**の内容を見ておきましょう．このプログラムでは，サブルーチンDelay1を用

いてソフトウェア・タイマを使っています．サブルーチンを用いるためにスタック・ポインタを設定し，サブルーチン・コールしたあとで戻り先の番地を確保するためのメモリ領域を設定しています．RAMENDはincludeファイルに定義されていて，それぞれのデバイスに固有のアドレスが取り込まれます．これについては第3章で説明します．

PORTBに0xFFを出力し，ソフトウェア・タイマで時間を遅らせた後，PORTBに反転データである0x00を出力します．続いてDelay1で時間を遅らせ，反転データを出力，これを永遠に繰り返します．

STK500は，STK500上のAVRだけでなくユーザの作成したボード上のAVRにプログラムを書き込む機能ももっています．これはSTK500のATmega8にプログラムするときにも使用した，ISPの信号を用いて行います．ユーザのターゲット・ボードにこの信号をフラット・ケーブルで接続して行います．次の節で説明するAVRISPと同じことができるわけです．

1-8　AVRISPを用いたボード上のAVRへのプログラミング

図1-10にATmega8の評価用ボードの回路例を，外観を**写真1-1**に示します．

この回路例では6ピンのヘッダがISPコネクタとなっています(**写真1-2**)．AVRISPを購入すると本体から出ているケーブルが10芯のフラットとなっているようですが，6芯を用いたほうが省スペースになりますし，10ピンだからとくに性能が良いということもないので，6ピンに交換しておきます．

AVRISP本体裏面の4隅に楕円の穴があり中に爪が見えるので，マイナス・ドライバを使ってこの爪を外側へ押せば蓋を外すことができます．付属の6芯フラット・ケーブル(10芯と6芯の両方を出しておいても問題ないようだ)と交換します．AVRISPの電源はこのケーブルを介してターゲット・ボードから供給されます．評価用ボードには8ビットのLED表示器，3ビットのプッシュ・スイッチ，RS-232シリアル・インターフェースを設けてありますが，今回の実験ではLED表示器だけ使っています(**写真1-3**)．

ボードを設計する際には，V_{CC}とGNDピンの近くにバイパス・コンデンサ0.1μF程度を必ず入れてください．またAV_{CC}は，必ずV_{CC}と等しいかそれ以下の電源を加えます．オープンで使用することがないようにしてください．

内蔵のA-Dコンバータをより高精度で使うには，この電源(AV_{CC})に安定度の高いリプルの少ない電源を供給する必要があります．V_{CC}から作る場合，V_{CC}に約10μHのインダクタを接続してもう反対側をAV_{CC}に接続し，AV_{CC}とGND間に0.1μFのセラミック・コンデンサを入れます．外部リファレンス電圧を使用しない場合，0.1μFのセラミック・コンデンサをA_{REF}とGND間に入れると安定度が増すとされています．

その他，ISPのインターフェースですが，ISPでのプログラムが終わったら，通常のI/O機能として使用することができます．ただし，ISPのインターフェースがAVRの出力ピンとして使用する場合はまず問題がないのですが，入力として設計され外部信号とAVRのピンの間にバッファICなどをつけていると，このICにより信号がスタックしてしまいプログラムを行えない場合があります(**図1-11**)．アトメル社は，このICとAVRピンの間に6.8 kΩ以上のできるだけ高い抵抗を直列に入れることを推奨していますが，むしろ抵抗の代わりにジャンパを入れることをお勧めします．

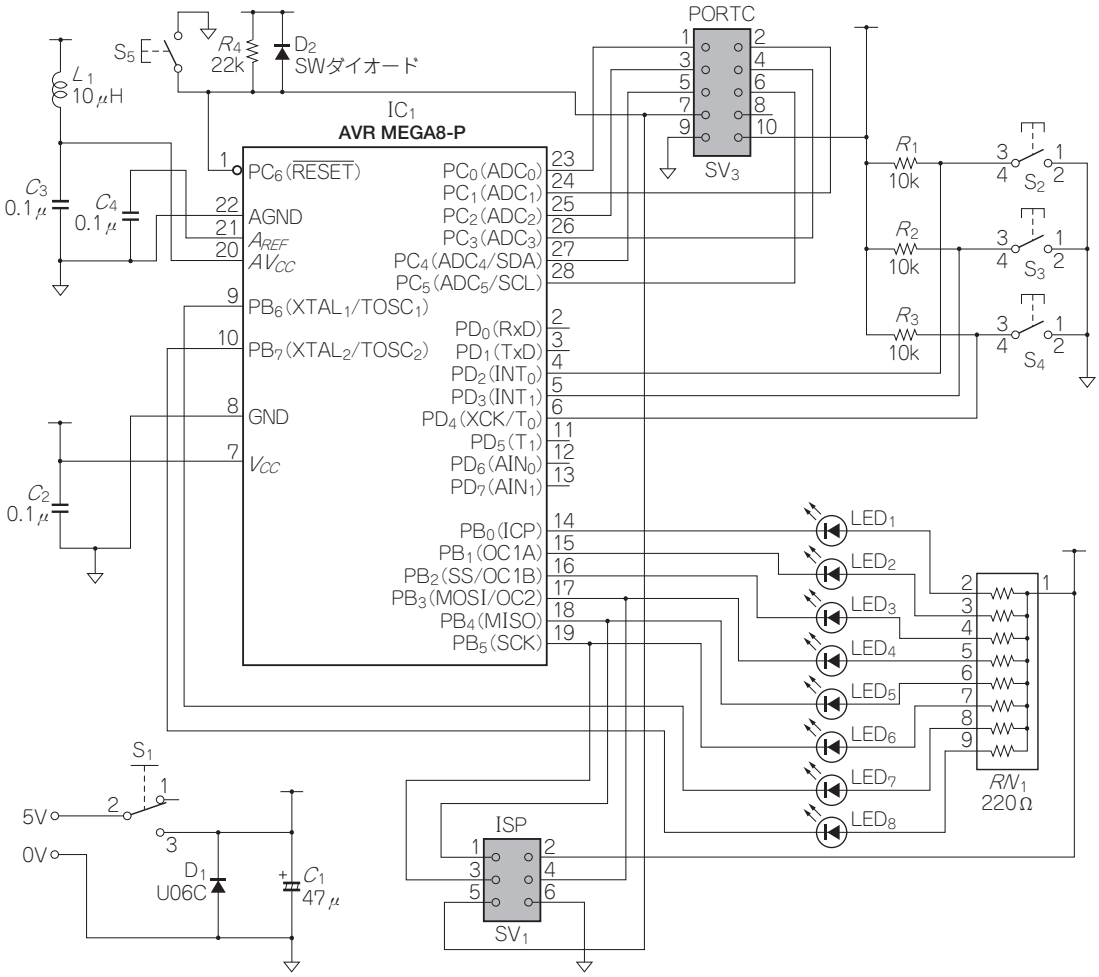

図1-10　ATmega8評価用ボード回路例

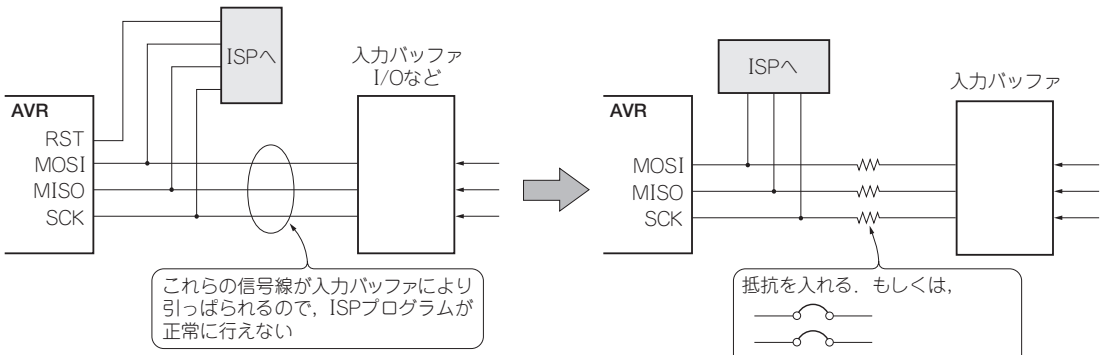

図1-11　ISPプログラムの注意点

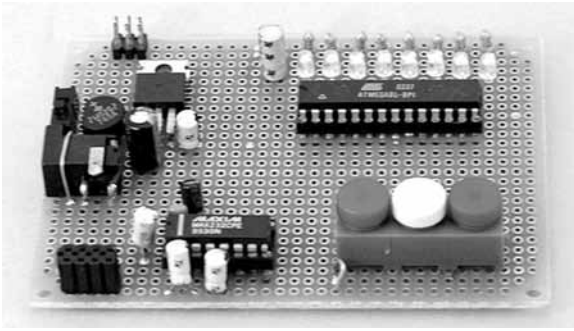

写真1-1　手作り評価ボードの例

写真1-2　AVRISPの外観

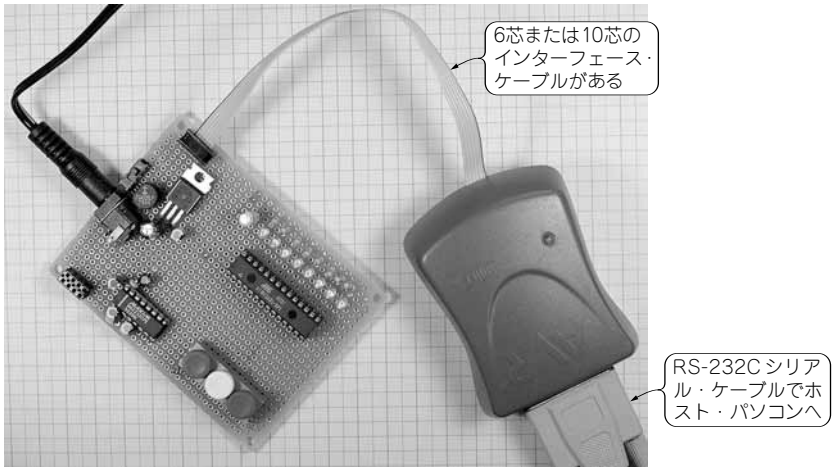

写真1-3　AVRISPと手作り評価ボードの接続

AVRISPとパソコンはシリアル・ケーブルで接続します．電源はターゲット・ボードから供給します．後は，STK500でプログラムしたときと同じ手続きで，ターゲット・ボード上のAVRにプログラムすることができます．

Tools→Program AVR→Auto Connectを選択するとウィンドウが表示されるので，STK500で行った手続きで，Programおよびヒューズ・ビットのプログラムができます．インターフェースが接続されていれば即，表示器の点滅を見ることができます．ヒューズ・ビットのプログラムでオシレータの周波数を変えてプログラムすれば，点滅の速さを変えることもできます．

1-9　1クロック1インストラクション

このMCUの特徴である1クロック1インストラクション(命令)を説明する前に，古くから使われてきた構造のコンピュータの動作を復習しておきます．伝統的なコンピュータと動作を比較することで，AVRのパワーを明確にできると思われます．