電子おもちゃの作り方

Embedded system design using AT90S2313

慶應義塾大学・環境情報学部教授・武藤佳恭
takefuji@sfc.keio.ac.jp
慶應義塾大学 SFC 研究所 研究所員・江藤　潔
esoc@sfc.keio.ac.jp

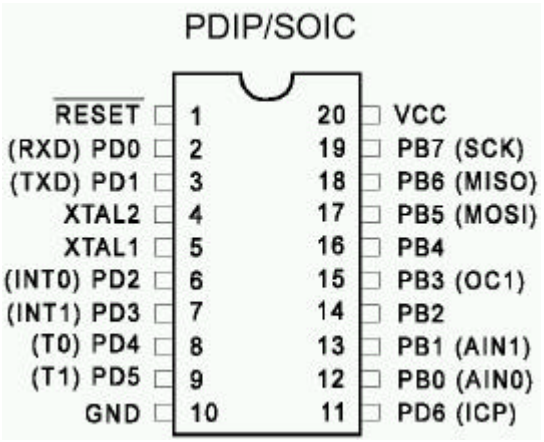
はじめに

この本では、実際に秋葉原に売っている部品を使って自分で自由自在に電子おもちゃを設計・製作するための入門書である。義務教育（中学生）の知識で理解できることを前提としている。電子部品（抵抗、キャパシタ（コンデンサと呼んでは世界に通用しない)、ダイオード、トランジスタ、LED、LCD、フラッシュメモリ付きマイクロコントローラチップなど）の使い方を基礎から応用まで解説する。理論よりも、実用性に重点を置いている。

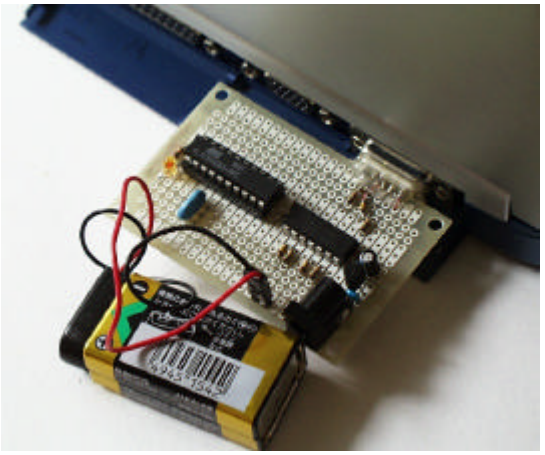
この本では、電子おもちゃ構築のための最新のハードウエアから製作のために必要なソフトウエア開発ツール環境（AVR-GCC: ATMEL 社のマイクロコントローラのための GNU　GCC コンパイラ）にも具体的に触れながら、できるだけ快適な環境（安価・どこまでも探求できる）でプログラム製作できるように心がけている。

この本では、ネットワーク・インターネットに接続可能なおもちゃを構築するためのインターフェース技術・通信技術（シリアル通信：RS232C 通信）に触れながら、必要最低限の知識で自分の望む電子おもちゃを構築できるようにしている。

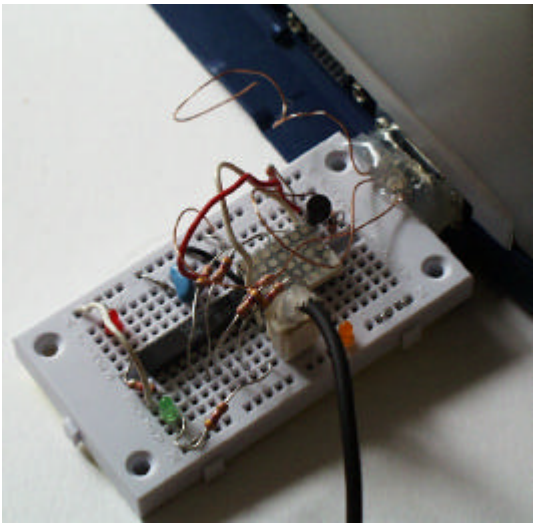
キーワード：抵抗、キャパシタ、ダイオード、LED、トランジスタ、フラッシュメモリ付きマイクロコントローラ、AVR-GCC, ATMEL 社、GNU　GCC C コンパイラ、シリアル通信、RS232C 通信、インターネット。



２０ピン AT90S2313 **チップ**



AT90S2313 **チップへのプログラム書き込み回路**



テスト回路の動作風景

もくじ

１．電子おもちゃ構築のための環境

１．１　Windows95, 98, NT,2000 上で必要なソフトウエア

最新のフラッシュメモリ付きマイクロコントローラはおもちゃの頭脳として働らいてくれる。フラッシュメモリとは不揮発性メモリで、電池を抜いてもメモリ内の中身は消えない。マイクロコントローラ内部へ自分のプログラムを格納し、動作させるためには、自分のプログラムをパソコン開発環境で構築し、そのプログラムを転送する必要がある。開発環境はシリアル通信可能な Windows コンピュータならば、OS は Windows95/98/NT あるいは Windows2000、なんでもかまわない。

マイクロコントローラの中でも、安価（１個４００円ぐらい）で UART 機能（シリアル通信の RS２３２C）を備え、自由に使える 15 ピンの I/O に加え、複数のタイマー、いろいろな割り込み機能を備えている、ATMEL 社 20 ピンの AT90S2313 を中心に解説する。

Windows コンピュータ(intel x86 系列)上で必要な開発ツールは、クロスコンパイラ（intel x86 マシン上で AT90S2313 マシンのコードを生成する）が必要である。この本では、AVR-GCC（GNU　GCC　C コンパイラ）を用いる。なぜならば、無料でダウンロードできるばかりでなく、コンパイラのソースコードなども手に入り、とことん勉強したければ、いくらでも探求できる環境なのである。

AT90S2313 用のプログラムコード（実行・転送可能なコード）は AVR-GCC コンパイラにより、インテル hex フォーマットに変換され、AVRSS 転送プログラムを用いて、Windows コンピュータから AT90S2313 のマイクロコントローラ内部へ転送できる。

必要なソフトウエア：
AVR-GCC：GNU　GCC　C コンパイラ
AVRSS：　シリアル転送プログラム PC→AT90S2313

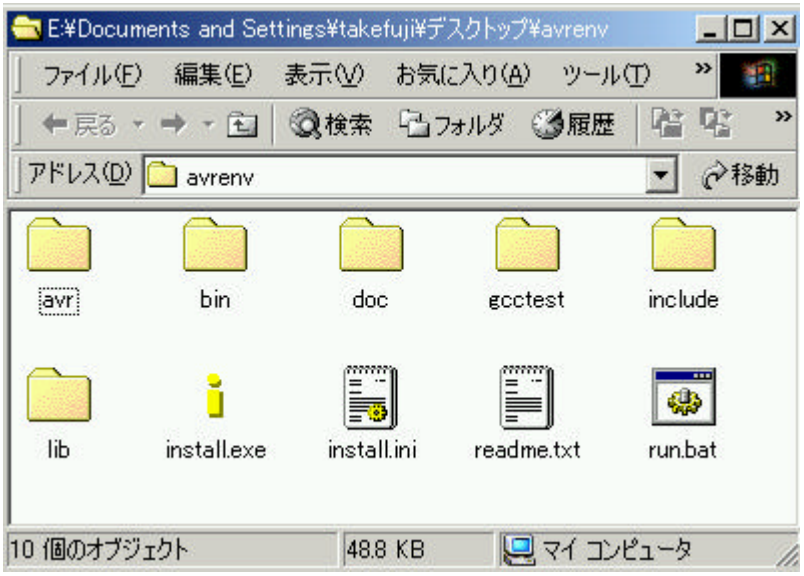
1. 電子おもちゃ構築のための環境

１．２　プログラミング環境の設定

AVR-ENV 環境を構築するには、avrenv.zip ファイルをダウンロードする。ダウンロードするには、右ボタンクリックを用いる。avrenv.zip ファイルには、AVR-GCC　GNU C コンパイラ環境と AVRSS 環境の両方を含む。また、ここで紹介している例題プログラムもすべて含んでいる。AVR-GCC は GNU　C クロスコンパイラの環境であり、Windows マシン上で AT90S2313 マイクロコントローラなどの最終実行ファイル（インテル　HEX ファイルと呼ばれる：xxx.hex）を構築できる。また、最終実行ファイルを PC からマイクロコントローラに書き込むソフトウエアが必要であるが、そのためのプログラムが AVRSS である。
http://www.sfc.keio.ac.jp/~takefuji/kenkyukai/toy.html から avrenv.zip ファイルをダウンロードし、その zip ファイルを解凍する。Windows 上で zip ファイルを解凍するには Lhasa 解凍プログラムなどを次のサイトからあらかじめダウンロードする。

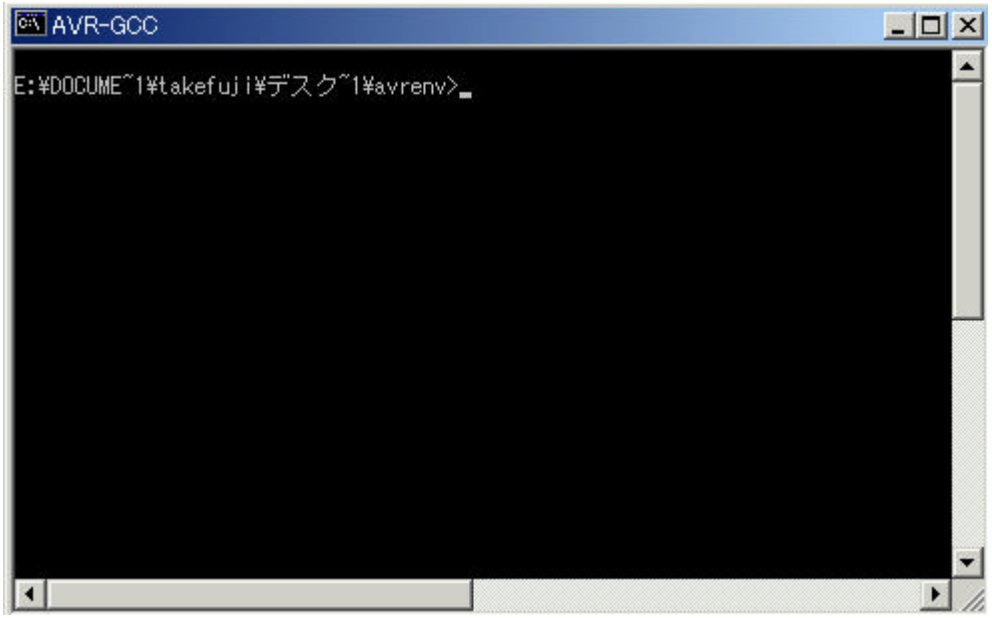
http://www.forest.impress.co.jp/library/lhasa.html

avrenv.zip を解凍すると、次のようなファイル構成となる。

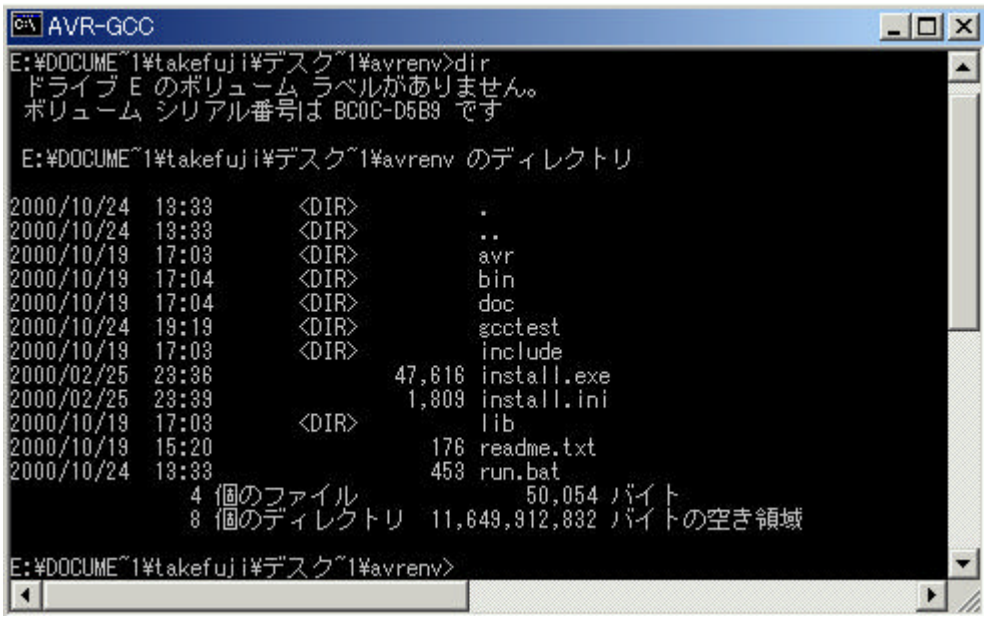


ここで、install.exe をダブルクリックし、AVR-GCC クロスコンパイラと AVRSS

書き込みプログラムをインストールする。インストールがうまく完了すると、デスクトップ上に AVR-GCC のアイコンができるはずである。
AVR－GCC の詳しい説明は
http://members.nbci.com/_XMCM/volkeroth/index_e.htm
また、AVRSS の詳しい説明は、
http://elm-chan.org/reports/avrx/report.html を参照せよ。
AVR-GCC をダブルクリックして、AVR-GCC や AVRSS をうまく起動できるか確かめてみる。ダブルクリックすると、次のようなウィンドウが出てくる。



ここで、dir コマンドを実行すると avrenv のファイル構成がわかる。



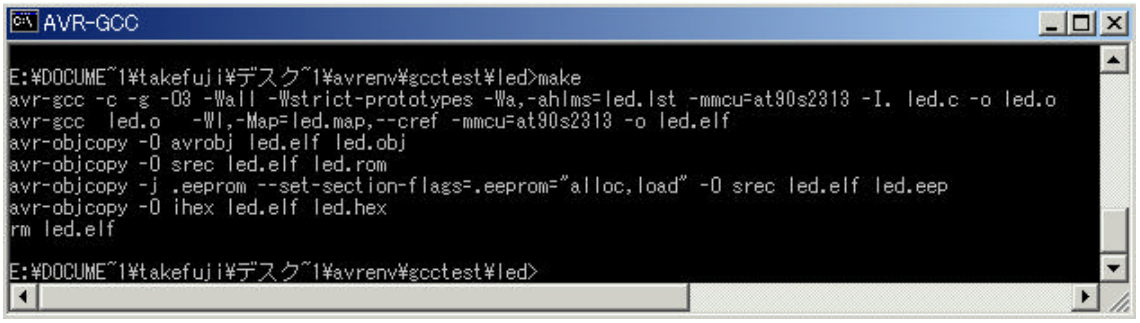
ここで、試しに gcctest のディレクトリに移動し、更に、led のディレクトリに移動し、make して実行ファイルを構築してみる。ディレクトリへの移動は cd コマンドを用いる。例えば、

```
cd  gcctest¥led
```

実行ファイルを構築するには、make コマンドを入力しリターンキーを押す。

```
make
```

make コマンドの役目は最終実行ファイル(ここでは、lex.hex)を自動的に生成することである。



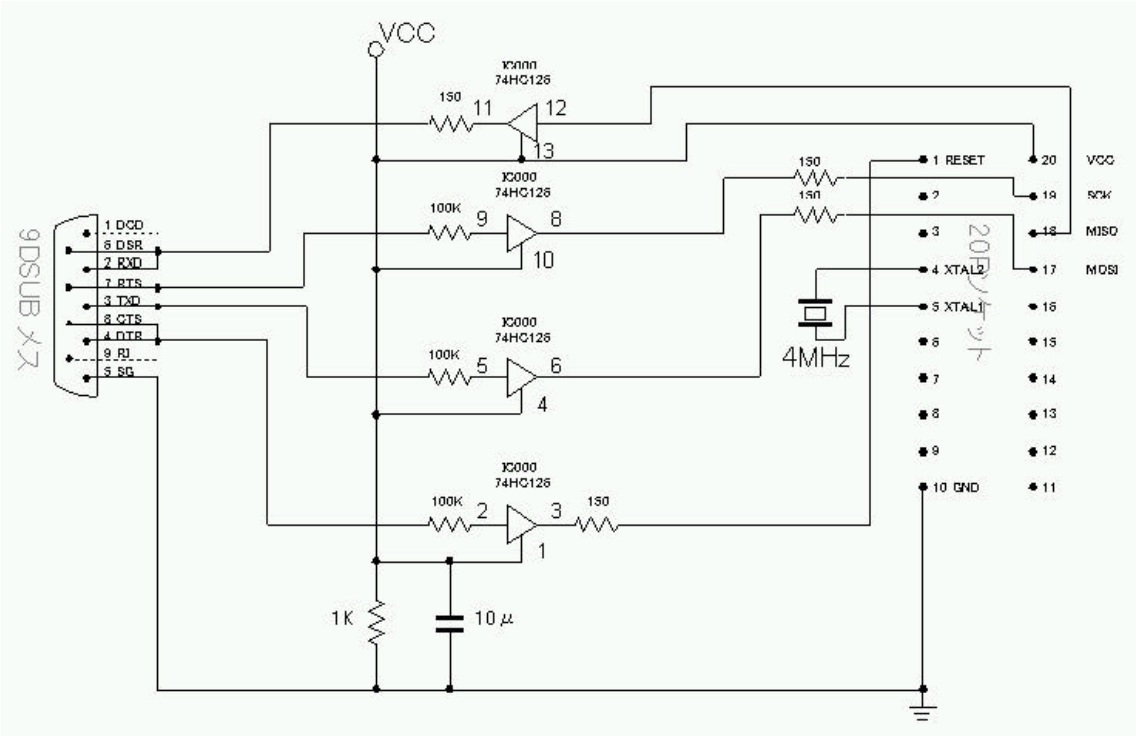
ここで出来上がった led.hex ファイルを avrss コマンドを用いて AT90S2313 にプログラム転送してみる。

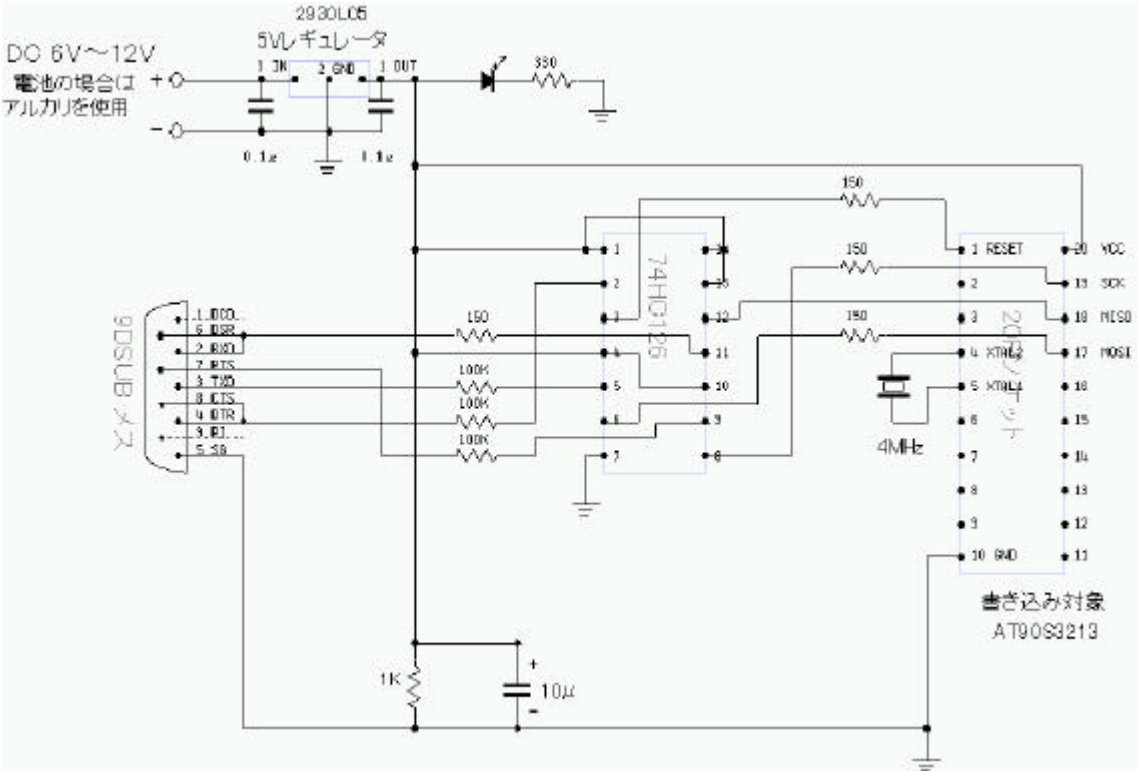
```
avrss  led.hex
```

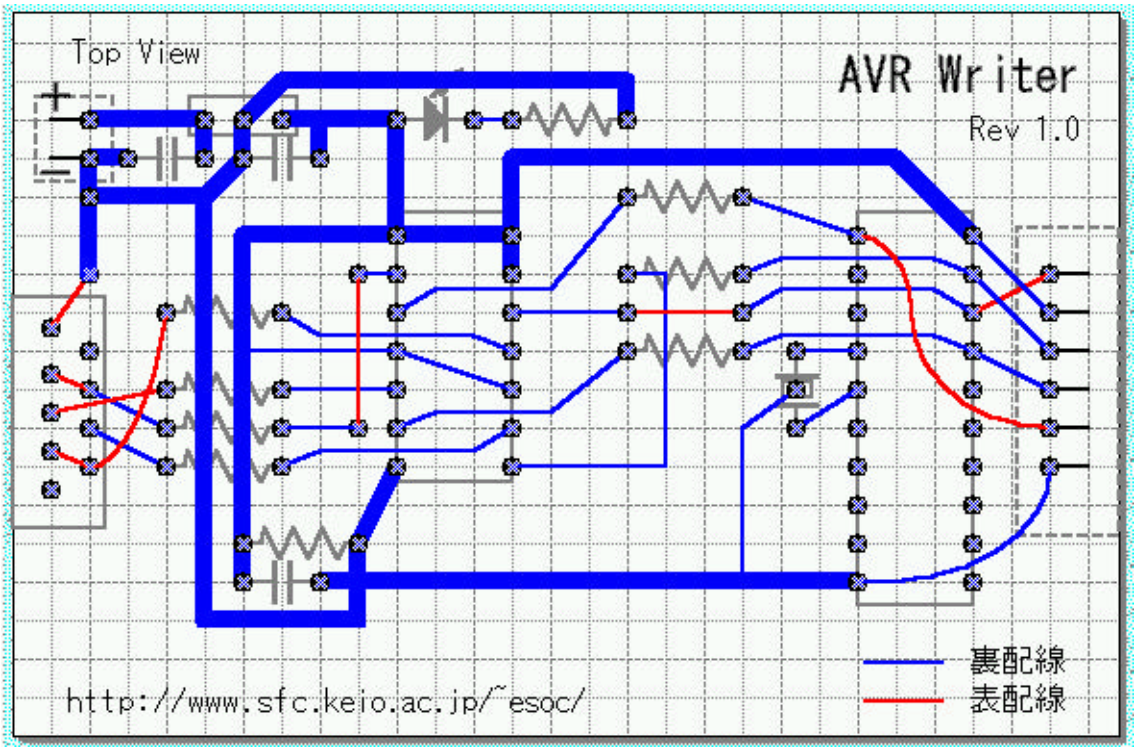
と入力し、リターンキーを押すと、簡単に実行可能な hex ファイルをマイクロコントローラチップへ転送できる。実際にはプログラムライタやプログラムローダ回路が必要である。

１．３　PC→AT90S2313　プログラムライタ回路

秋葉原へ行って、５００円以内で作れる、簡単なライタ回路を次に示す。必要な主な部品はスリーステートバッファーIC、３本足水晶発振子、抵抗、キャパシタ、９DSUB メスコネクタと２０ピン IC ソケットだけである。回路図と部品表などを次に示す。電源は DC５V を用いるか、３端子レギュレータ（例えば、９V 電池を使って DC５V 電源を作る。）を用いる。AT90S2313 の内臓発振回路と３本足水晶発振子を使うと、所望の発振周波数回路を構築できる。AT90S2313 では、１つの命令は 1/2、１、２、３、４クロック数のいづれかで実行できる。なんと、ほとんどの分岐命令は 1/2 クッロクで実行する。すなわち、10Mhz クロックを用いた場合、１つの分岐命令実行に 50ns である。


VCC
IC000
74HC126
150
100K
1K
10μ
4MHz
9DSUB メス
20Pソケット
1 DCD
6 DSR
2 RXD
7 RTS
3 TXD
8 CTS
4 DTR
9 RI
5 SG
1 RESET
2
3
4 XTAL2
5 XTAL1
6
7
8
9
10 GND
20 VCC
19 SCK
18 MISO
17 MOSI
16
15
14
13
12
11

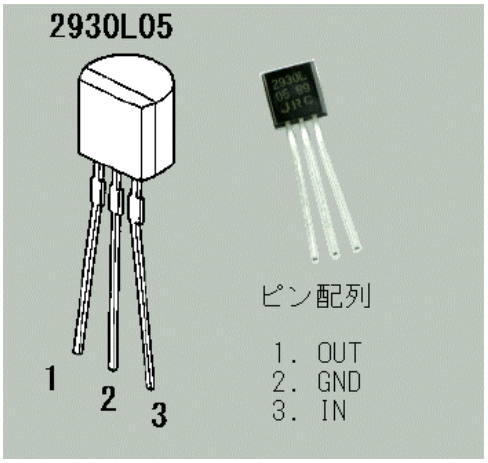

2930L05
5Vレギュレータ
DC 6V～12V
電池の場合は
アルカリを使用
+
−
1 IN
2 GND
3 OUT
330
0.1μ
150
100K
1K
10μ
74HC126
9DSUB メス
20Pソケット
4MHz
1 RESET
2
3
4 XTAL2
5 XTAL1
6
7
8
9
10 GND
20 VCC
19 SCK
18 MISO
17 MOSI
16
15
14
13
12
11
書き込み対象
AT90S3213



部品リスト(千:千石、秋:秋月)

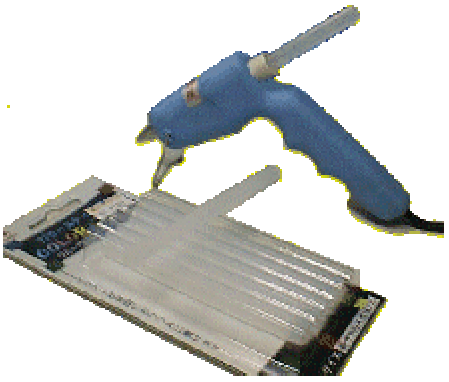
- **基盤**(25Pin X 15Pin:**秋** or **千**)
- RS232c **コネクタ、メス**(**秋**)
- 74HC126 (**千**)
- AT90S2313(**秋**)
- 2930L05(**秋**)
- 4MHz(**セラロック**:**秋**)
- 0.1µF X 2(**秋** or **千**)
- 10µF(**秋** or **千**)
- 150O X 4(**秋**)
- 330O(**秋**)
- 1KO(**秋**)
- 100KO X 3(**秋**)
- 20Pin IC **ソケット**(**千**)

注意！:2930L05 の取り付け方向について



2930L05 は取り付ける際に方向を区別する必要があります。右図のように、regulate する前の電圧を 3 ピンに入力し、regulate された電圧出力を 1 ピンから得るという仕様になっています。2930L05 を上から見た場合、輪郭が平らな面と丸い面がありますが、前ページの AVR Writer と書いてある図でいえば、その平らな面が図中で下方向に位置するように設置することになります。

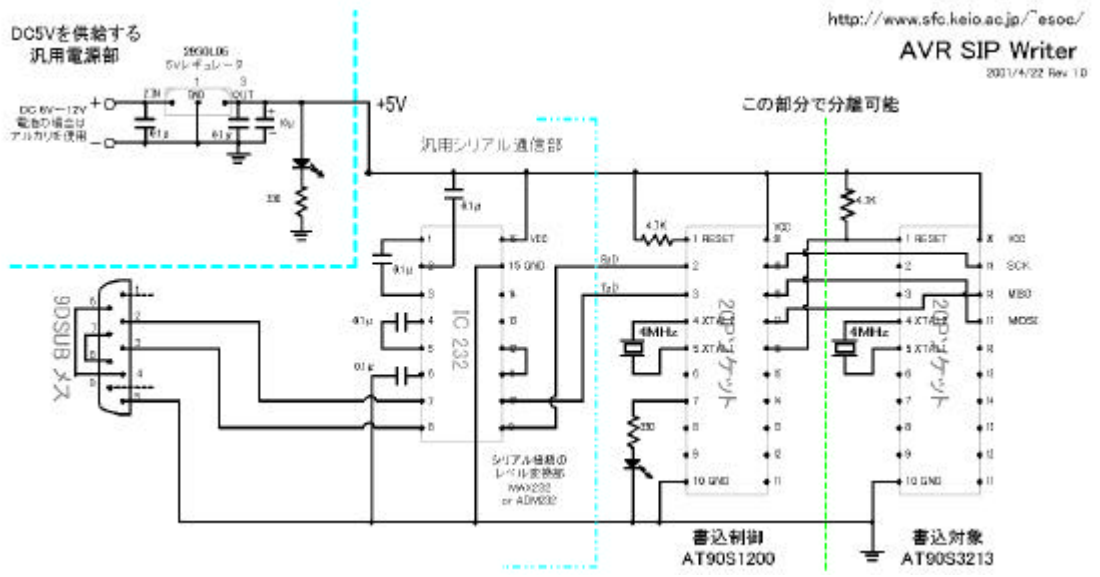
また、主な道具としては、つぎのようなものが必要である。



シリアルインターフェースを利用することによってこのような単純な回路で AT90S2313 へのプログラムライター回路が実現できる。

１．４ PC→AT90S2313　USB 経由による プログラムライタ回路

シリアルインターフェースの無いラップトップコンピュータから書き込む場合、USB ポートをシリアルインターフェースに変換して書き込むという選択が考えられる。書き込む際の同期の問題から、すでに紹介した回路そのままでは動作せず、プログラム書き込み用のプログラムをあらかじめ書き込んでおいたマイクロコントローラーを使用することで書き込みを可能としている。そのためには AT902313 の仲間の AT901200 というマイクロコントローラーを使用する。あらかじめ AT901200 に転送しておくプログラムとそのソースコードは以下の URL からダウンロードすることができる。
プログラム：http://www.sfc.keio.ac.jp/~esoc/avr/writer/1200/AVRPGM01.HEX
ソース：http://www.sfc.keio.ac.jp/~esoc/avr/writer/1200/AVRPGM01.ASM

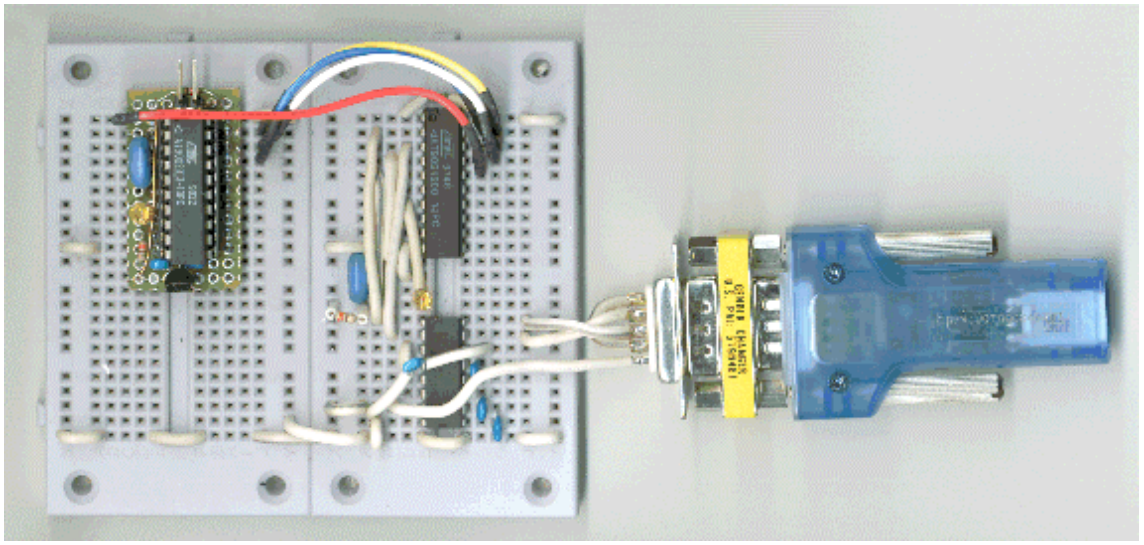


部品リスト(千:千石、秋:秋月)

- RS232c コネクタ、メス(秋)
- ADM232AAN (秋)
- AT90S1200(秋) 注意：AT90S1200A は不可
- AT90S2313(秋)

・2930L05(秋)
・4MHz X 2(セラロック:秋)
・LED X2(好きな色の小さなもの:秋)
・0.1μF X 6(青色:秋 or 千)
・10μF(電解、円柱:秋 or 千)
・330O X 2(秋)
・4.7KO X 2(秋)
・USB-RS232C 変換機(IO DATA 製　USB-RSAQ2(COM5)で動作確認済み)

この書き込み回路を利用する場合には、前述の avrss コマンドではなく、http://village.infoweb.ne.jp/~update/taka/avr_pgm/avrpgm.htm からダウンロードすることのできる「AKI-AVR プログラマキット」を使用する。



図：USB 接続の書き込み回路実装例

２．　電子部品の機能と簡単な使い方

この章では抵抗、キャパシタ、ダイオード、LED、トランジスタなどの基本部品の使い方を中心に説明する。

２．１ 抵抗とキャパシタ

抵抗はΩ（オーム）で表す。例えば１KΩの抵抗は３つのカラーコードで表される。４つめの色は誤差を表す。

（茶・黒・赤）＝１０　ｘ　１０^{2}＝１０００Ω

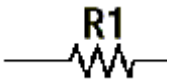
黒　茶　赤　橙　黄　緑　青　紫　灰　白
０　１　２　３　４　５　６　７　８　９

単位

p(ピコ)　n(ナノ)　μ(マイクロ)　m(ミリ)　K(キロ)　M(メガ)　G（ギガ）T(テラ)

１０$^{-12}$　１０$^{-9}$　１０$^{-6}$　１０$^{-3}$　１０$^{+3}$　１０$^{+6}$　１０$^{+9}$　１０$^{+12}$

（黄・紫・赤）＝４７　ｘ　１０^{2}＝４．７KΩ

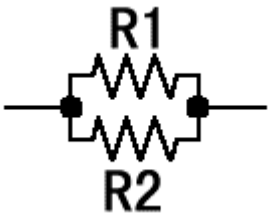
記号は 

で表現できる。

直列つなぎの抵抗値ＲはＲ＝Ｒ１＋Ｒ２

R1=1KΩ、R2=4.7KΩではＲ＝5.7KΩ

並列つなぎの抵抗値Ｒは 1／R＝1／R1+1／R2
R1=R2=1KΩではＲは５００Ωとなる。



上の抵抗値Ｒを導いてみる。

オームの法則　V（電圧）＝I（電流）　x　R（抵抗）

１．　R１とR２両端の電圧はVに等しい。

V＝i1　**x**　R1　　i1 は R1 に流れる電流

V＝I2　**x**　R2　　i2 は R2 に流れる電流

２．　電流 i の総和は i1+i2 **である。**

i=i1　+　i2

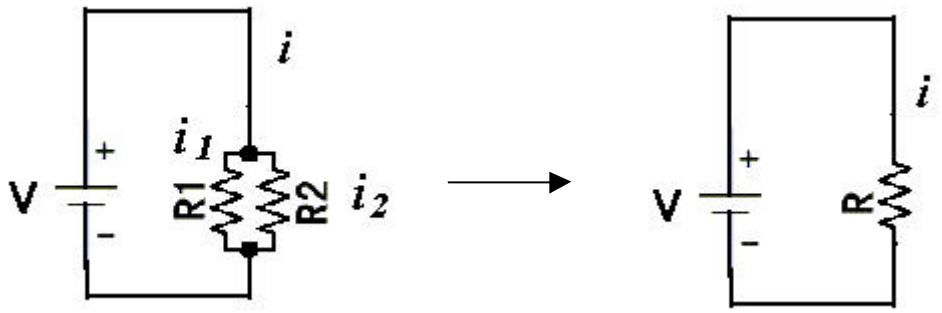
３．　電圧V＝i（電流）　x　R(抵抗)である。

V＝i　x　R

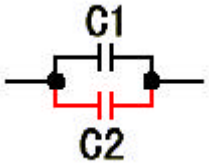
１．　より　V＝i　x　R --→　i=V／R、

２．に１．を代入し３．より、i=i1　+　i2=V／R1+V／R2＝**V／Rとなる。つまり、**V／R1+V／R2＝**V／Rとなり、両辺をVで割ると、**

1／R＝1／R1+1／R2 **となる。**



並列つなぎのキャパシタ値CはC＝C1+C2



直列つなぎのキャパシタの値Cは 1／C=1／C1+1／C2

オームの法則を用いて、1／C=1／C1+1／C2 を導いてみる。
電荷量ＱはＱ＝Ｖ（電圧）　ｘ　Ｃ（キャパシタンス）
1. Q1=C1 x V1
2. Q2=C2 x V2
3. V=V1+V2
4. Q=C x V
5. Q=Q1=Q2

1／C=Ｖ／Ｑ＝（V1+V2）／Ｑ=(Q1／C1+Q2／C2)／Q より、
1／C=1／C1+1／C2

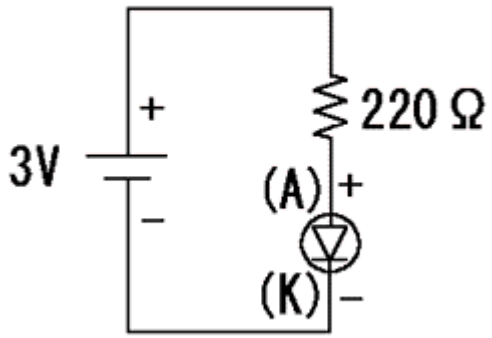
キャパシタは色の変わりに３つの数字で表現される場合がある。
例えば、１０３の場合は１０　ｘ　１０^3＝10000ｐF＝0.01uF(マイクロファラッド)

２．２　抵抗、ダイオード、LED

ダイオードもＬＥＤも一方向に電流がながれやすい素子である。
また、障壁電圧（ＬＥＤでは１．８Ｖぐらい、シリコンダイオードは０．８Ｖぐらい）よりも高い電圧を加えなければ電流は流れない。

例えば、下の図のように３Ｖの電池をつないだ場合、障壁電位を超えているので、自動的にダイオードならば０．８Ｖ、ＬＥＤならば１．８Ｖの電位が両端で保たれる。すなはち、抵抗の両端はＬＥＤならば１．２Ｖ（＝３－１．８）、ダイオードならば２．２Ｖ（＝３－０．８）になる。
電圧Ｖ＝電流Ｉ　ｘ　抵抗Ｒ　なので、
ダイオードの場合、電流ＩはＩ＝２．２／２２０＝０．０１Ａ＝１０mA とな

る。

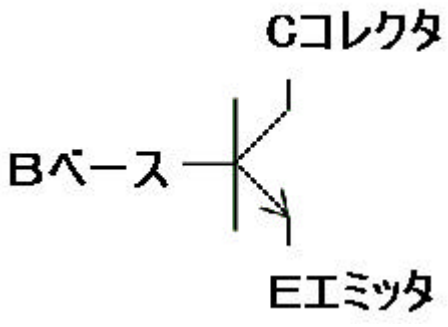


ダイオードの記号は  である。

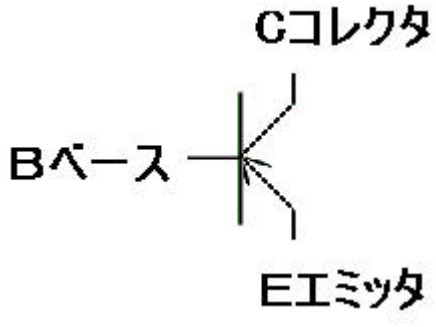
２．３　抵抗とバイポーラトランジスタ

バイポーラトランジスタにはＰＮＰ型とＮＰＮ型の２種類があり、実際には４種類の記号がある。（２ＳＡ，２ＳＢ，２ＳＣ，２ＳＤ）

２ＳＡｘｘｘ　　ＰＮＰ型
２ＳＢｘｘｘ　　ＰＮＰ型
２ＳＣｘｘｘ　　ＮＰＮ型
２ＳＤｘｘｘ　　ＮＰＮ型

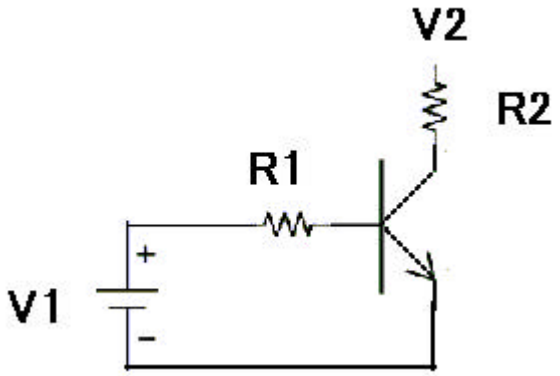


ＮＰＮ型トランジスタ

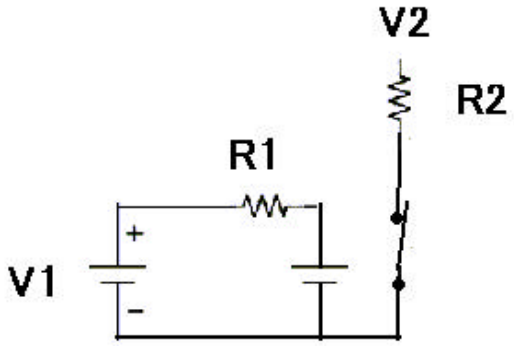


ＰＮＰ型トランジスタ

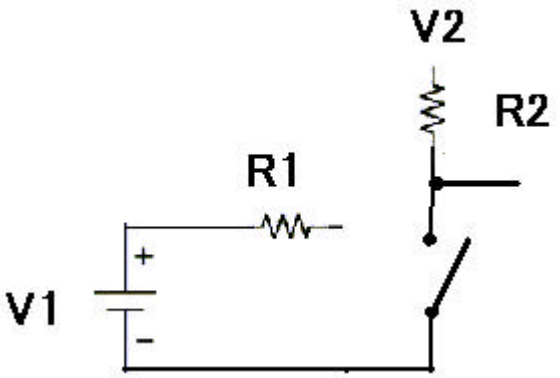
バイポーラトランジスタをスイッチとして使う場合、ベータとエミッタ間の障壁電位（０．７５Ｖ）を超える電位を順方向に加えるとベース・エミッタ間は 0.75V になり、コレクタ・エミッタ間は同電位になる。



Ｖ１＞０．７５Ｖの場合コレクタ・エミッタ間はスイッチＯＮになり、コレクタ・エミッタは同電位つまり、０Ｖとある



Ｖ１＜０．７５Ｖの場合、コレクタ・エミッタ間はスイッチＯＦＦになる。



例えば、Ｖ１＝３Ｖ、Ｖ２＝５Ｖ、Ｒ１＝Ｒ２＝４．７ＫΩでは、
Ｒ１の抵抗には（３Ｖ－０．７５Ｖ）／R1 の電流が流れる。また、コレクタ・エミッタ間は同電位になり、コレクタの出力は０Ｖとなる。

例えば、Ｖ１＝－１Ｖ，Ｖ２＝５Ｖ，Ｒ１＝Ｒ２＝４．７ＫΩでは、
コレクタ・エミッタ間はスイッチＯＦＦになり、コレクタの電位はＶ２となる。

２．４　　その他の役立つ電子部品

ここでは、水晶発振素子、ＣＭＯＳ論理ゲート、３端子レギュレータ、モータ駆動ＩＣ，タイマーＩＣ、の機能と使い方を例を交えながら説明する。

２．５　　その他の役立つ部品

・ブレッドボード
ブレッドボードとは表面に無数の穴が空いた板状の物体で、基盤にはんだ付けすることなしに回路を作る手助けとなるものです。あいている穴に部品や配線を突っ込むと中のクリップで挟まれるようになっているので、ブロック感覚で回路を組み立てることが可能です。下図の赤い部分で示されたように、複数の穴が内部で繋がっています。特にボードの上下に分かれて平行に走っているライン 2 本は縦方向につながっているほかのラインに接続しやすいので、電源の+と GND で利用します。配線を自作する場合は 0.5mm から 0.8mm の単芯の銅線を利用します。身近な物ですとインターフォンの配線が 0.8mm でホームセンターで入手できます。線を剥く長さは 5mm から 7mm が適当です。

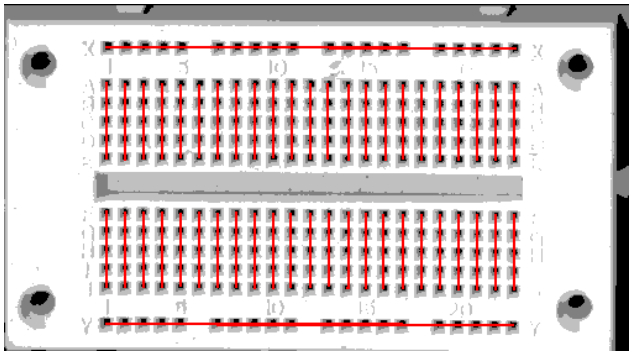


図 ブレッドボード内部の接続の様子

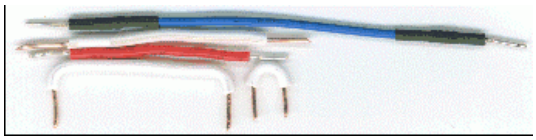
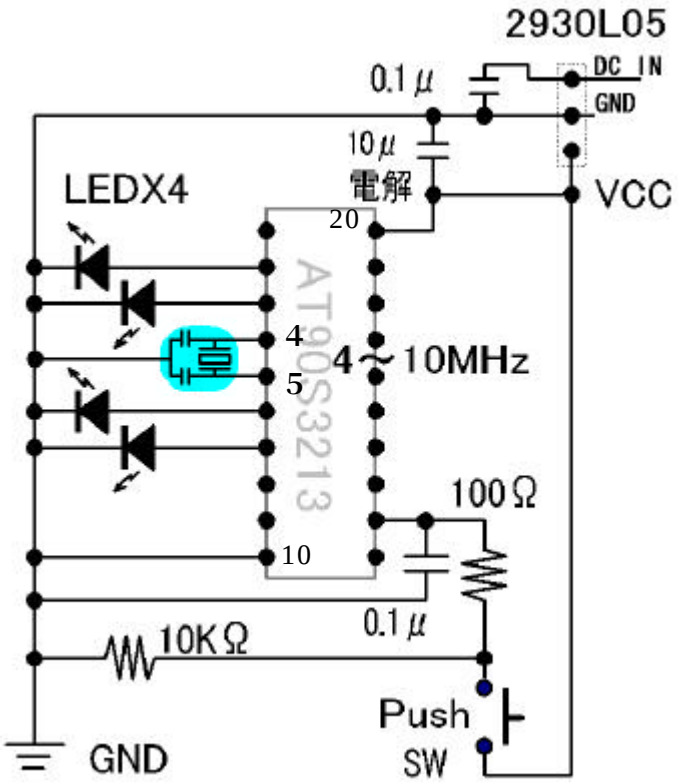


図 接続用配線

２．６ 最も単純な回路構成



右図の例は AT902313 にプログラムを書き込んだあと、実際に AT902313 単体で動かす際の最も単純な回路構成の一例である。基本的には４番 pin と 5 番 pin にセラロックを接続し、20 番 pin に電圧供給、10 番 pin とセラロックの真中の端子をグラウンド、すなわち０V に接続すれば回路は成り立ち、AT902313 は動き出します。
右図の例では安定した電圧を供給するためにライタと同じく 2930L05 を使用している他、動作を確認するための LED とプッシュスイッチ(単純なボタン)を接続してあります。

ここではボタンを押すと LED が光るというプログラムの例を紹介します。

```
/*  Title:    AVR-GCC test program for at90s2313
    Author:   eto
    Mail-to:  esoc@sfc.keio.ac.jp
    Date:     4/2001
    Purpose:  SW on LED
    Software: AVR-GCC  needed
    Hardware: ATS90S2313

    PortD LED and PortB SW
    PB0=SW0
    PD3=LED0
*/
```

```
#include
#include
typedef unsigned char  io8bit;

int main( void )
      {
      io8bit nowled ;
      outp(0xff,DDRD);       /* PD1-PD6 PortD for output */
      outp(0xfe,DDRB);       /* PB0-PB1 PortB for input  */
      nowled = 0x00;         /* LED init all Low        */

      for (;;)
            {
            if((inp(PINB) & 0x01) != 0x00)  /* check switch */
                  nowled = 0x01;   /* LED0 on set */
            else
                  nowled = 0x00;   /* LED off set */

            outp(nowled, PORTD);   /* LED out */
            }
      }
```

３．　ソフトウエアを C 言語で組んでみよう。

ＡＶＲ－ＧＣＣのＣコンパイラを使って AT90S2313 のプログラムを組んでみる。よく使うＣプログラムの重要コマンドをあげてみる。

#include <io2313.h> **(データ構造やパラメータを定義しているファイル参照)**

#define nop() asm volatile ("nop" ::)　**（C プログラム内でアセンブラ命令を定義）**

ここでは、シークエンシャルなプログラム、割り込みプログラム（タイマー、外部割込み)、シリアル通信プログラム、新しいデバイスとのインターフェース

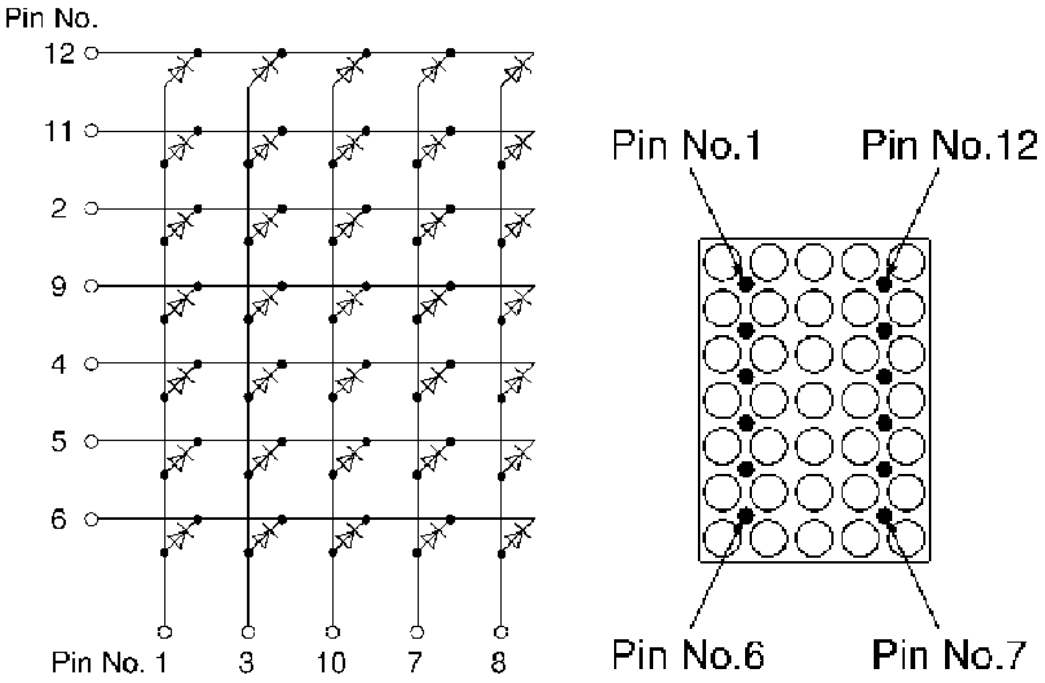
など、役に立つプログラム例を示す。

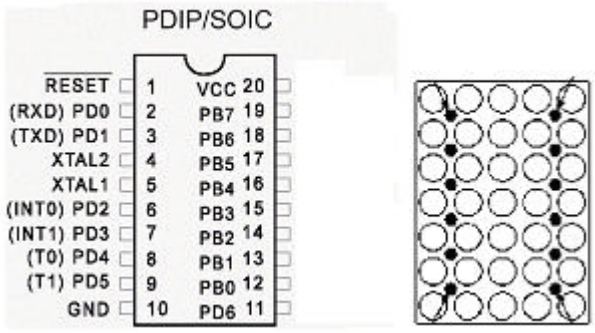
３．１　おもちゃのＣプログラム例１：LEDを点滅させる

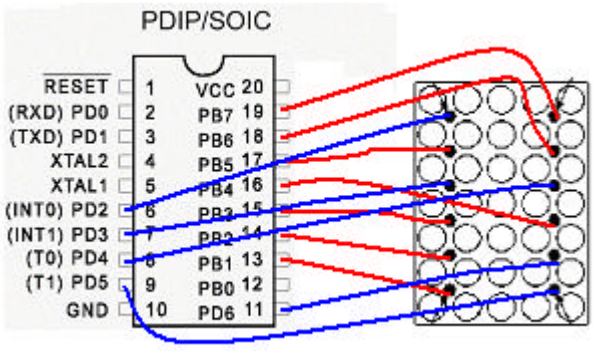
５ｘ７のLEDマトリックスを使って表示装置を作ってみる。
５ｘ７のLEDマトリックスには３５個のLEDがあり、１２ピンのパッケージになっている。例えば、一番左上のLEDを光らせるためには、P1（ピン番号１）にHigh信号とP12にLow信号を与える必要がある。
(p12,p11,p2,p9,p4,p5,p6)=(L,H,H,H,H,H,H)
(p1,p3,p10,p7,p8)=(H,L,L,L,L)

一番右下のLEDを光らすためには、
(p12,p11,p2,p9,p4,p5,p6)=(H,H,H,H,H,H,L)
(p1,p3,p10,p7,p8)=(L,L,L,L,H)

(p12,p11,p2,p9,p4,p5,p6)=(L,H,H,H,H,H,H)
(p1,p3,p10,p7,p8)=(H,L,H,L,H)とすると、上の行のLEDが１０１０１と光る。







青い線には４．７ＫΩの抵抗を直列して接続します。

次のプログラムは、一番上の５つの LED の内１つを光らせながら、つぎつぎにシフトするものです。ここでは、C プログラムの中で nop()という、アセンブラプログラムをどう呼びだしている。
Port B7 から PortB １の７bit と Port D６から Port D2 の５bit をそれぞれ出力に定義するためには、

```
outp(0xff,DDRB);            /* PB7-PB1 of PortB for output */
outp(0x7e,DDRD);            /* PD6-PD1 of PortD for output */
```

のようにする。
０ｘｆｆは１６進数を表現し、8-bit すべてが１である。

このプログラムでは、8-bit の tate の変数は tate=(p12,p11,p2,p9,p4,p5,p6,PB0)。
ここでは、PB0 は未使用。また、5-bit の yoko 変数は、
yoko=(p8,p7,p10,p3,p1,PD1,PD0)。　　PD1 と PD0 は未使用。

```
tate = 0x7f;      は p12 だけを L にしている、また
yoko = 0x40;      は p8 だけを H にしている。

for (i=0; i<255; i++)   /* outer delay loop */
```

```
 for(j=0; j<255;j++) /* inner delay loop */
    for(k=0;k<10;k++)
      nop();            /* delay */
```

のプログラムは単純に遅延時間を作っている。

```
/*   Title:    AVR-GCC Port I/O program for AT90S2313
    Author:   takefuji
    Date:     10/2000
    Purpose:   Flash the top row of  five LEDs by manipulating Port
                B7-B1(7-bit) and Port D6-D2(5-bit)
    Software: AVR-GCC   needed
     Hardware: ATS90S2313 and 5x7 LED matrix

  5x7 LED matrix
p12 +  +  +  +  +
p11 +  +  +  +  +
p2  +  +  +  +  +
p9  +  +  +  +  +          +  indicates an LED.
p4  +  +  +  +  +
p5  +  +  +  +  +
p6  +  +  +  +  +
  p1 p3 p10 p7 p8

     Mail-to:      takefuji@sfc.keio.ac.jp

PB7=p12,PB6=p11,PB5=p2,PB4=p9,PB3=p9,PB2=p4,PB1=p5,PB1=p6
PD6=p8,PD5=p7,PD4=p10,PD3=p3,PD2=p1
*/

#include <io2313.h>
#include <iomacros.h>
#define  nop() asm volatile ("nop" ::)
typedef unsigned char  io8bit;

int main( void )
{
```

```
    io8bit tate, yoko,i, j, k;

    outp(0xff,DDRB);            /* set PB7-PB1 of PortB for output */
    outp(0x7e,DDRD);            /* set PD6-PD1 of PortD for output */

    tate = 0x7f;                /* tate init value p12=L */
    yoko = 0x40;                    /* yoko init value p8=H */

   for (;;) {
       outp(tate, PORTB);      /* p12-p8 LED on */
       outp(yoko, PORTD);      /* p12-p8 LED on */
        for (i=0; i<255; i++)   /* outer delay loop */
          for(j=0; j<255;j++) /* inner delay loop */
               for(k=0;k<10;k++)
                   nop();           /* delay */
        yoko <<=1;        /* shift 1bit */
   if(!yoko) yoko = 0x04;
        outp(yoko, PORTD);
     }
}
```

課題：　３５個すべてのLEDのひとつづつを点灯させるプログラムを構築せよ。
課題：　文字を表示できるようなプログラムを構築せよ。
課題：　好きな文字列を表示するバナー表示器を構築せよ。

３．２　おもちゃのＣプログラム例２：タイマーと割り込み

ここでは、割り込みとマルチタスクのメカニズムを説明しながら、タイマーの割り込みプログラムを説明する。３．１では遅延時間を作るのに、for ループを使って実現していたが、タイマーを使って同様なことができる。ここでは、割り込み処理のメカニズムとマルチタスクができる原理をまず説明する。

割り込みとは、日常茶飯事おきている。例えば宿題をしている最中に携帯電話がなったとすると宿題をやる行為を一時中断して携帯電話のビジネスを済ませ、再び宿題モードに戻る行為を割り込み処理という。

マルチタスク処理とは、あたかも複数のジョブを同時に処理しているように見せかけている処理のことをいう。単純ラウンドロビン方式とはタイマーを使って定期的に割り込みを CPU にかけ均等に複数のジョブを片付けていく方式である。
割り込み処理がないとすると、CPU は１つのジョブしか処理できない。

タイマーがオーバーフローを起こすと割り込みを発生し、タイマー割り込み処理するプログラムである。　割り込みが発生すると、CPU は割り込み処理プログラムを開始する。実際には、割り込みの種類によって開始する番地（飛んでいく番地）が違うだけである。割り込み処理プログラムへ飛んでいくと、割り込み処理を抜け出さないと、永久に割り込み前の処理へ戻れない。通常、割り込み処理をしている最中に別の割り込みを許す場合を多重割り込みという。
ここでは、8-bit タイマーを用いて、タイマーオーバーフローで割り込みを発生させる。
タイマーオーバーフロー割り込みの設定は、

１．TIMSK レジスタの TCNT0 を１にセットする。
２．タイマーのクロックは内部クロックを用い、１０２４分周する。
＊ 外部信号でも、内部クロックでも選択できる。
＊ 分周数も選択できる。
３．タイマーカウンターの中身をセットする。
４．グローバル割り込みをイネーブルにする。

```
outp(0x02, TIMSK);          TCNT0 overflow をイネーブルにする。
outp(0,   TCNT0);           TCNT0 の中身を０にする。
outp(5,   TCCR0);           CPU クロックの分周数を１０２４にする。
sei();                      グローバル割り込みのイネーブル。
```

タイマーオーバーフロー割り込みが発生すると、SIGNAL(SIG_OVERFLOW0) のプログラムにジャンプする。
タイマーカウンターの中身を割り込みプログラムでもう一度初期化している。

```
outp(0, TCNT0);
```

```
/*
    Title:     AVR-GCC timer interrupt program for AT90S2313
    Author:    takefuji
    Date:      10/2000
```

```
    Purpose:  Flash the LEDs on Port B by using a 8-bit timer interrupt
    needed
    Software: AVR-GCC to compile needed
    Hardware: ATS90S2313
    Note:     Mail to takefuji@sfc.keio.ac.jp
*/

#include <io2313.h>
#include <interrupt.h>
#include <signal.h>

unsigned char led;


SIGNAL(SIG_OVERFLOW0)  /* signal handler for tcnt0 overflow interrupt
                                        */
{
   outp(~led, PORTB);   /* invert the output since a zero means: LED on */
    led=~led;
    outp(0, TCNT0);            /* reset counter to get this interrupt again */
}

int main(void)
{
    outp(0xff,DDRB);           /* use all pins on PortB for output */
    outp(0xff, PORTB);
    outp(0x02, TIMSK);                  /* enable TCNT0 overflow */
    outp(0,   TCNT0);                   /* reset TCNT0 */
    outp(5,   TCCR0);                   /* count with cpu clock/1024 */
    led= 0xff;
    sei();                      /* enable global interrupts */

    for (;;) {
    }             /* loop forever */
}
```

課題：16-bit **タイマーオーバーフローの割り込みプログラムを構築せよ。**
ヒント：SIGNAL(SIG_OVERFLOW1)、TIMSK、TCNT1H、TCNT1L、TCCR1B

３．３　おもちゃの C プログラム例３：UART シリアル通信

シリアル通信のレガシーである UART **を用いた、**RS232C **通信のプログラムを構築してみる。親機の** PC **から** RS232C **インターフェースを介してデータ受信プログラムを構築してみる。この実験では、シリアルデータは非同期に転送されてくるものとする。**RS232C **では、スタートビットに続き、７ビットあるいは８ビットデータ、更にパリティビット、更にストップビットが続く。**
ここでは、８ビットデータ、パリティなし、１ストップビットとする。

UART **の通信では、最低、**PD0**（受信）と** PD1**（送信）の２ビットと** GND**（グラウンド）の３本で通信できる。データを受信すると、割り込みを発生させ、受信したデータを** PB Port **に表示する。**
割り込みが発生すると、SIGNAL(SIG_UART_RECV) **のプログラムへ飛ぶ。**
設定には、UCR **レジスタの** RXCIE**、**TXCIE**、**RXEN**、**TXEN **のビットをセットする。ボーレートの設定は** UBRR **を２５にする。（４**Mhz **の** CPU **クロックを用いた場合、**UBRR=25 **は９６００**baud **設定に等しい。）**

```
/*
    Title:     AVR-GCC UART program for AT90S2313
    Author:    takefuji
    Date:      10/2000
   Purpose:  PC sends a byte data to UART of AT90S2313.
              The received data is displayed at PORTB.
             UART format: 9600 baud, 8bit, 1 stop-bit, no parity
    needed
   Software: AVR-GCC to compile
    needed
   Hardware: ATS90S2313 on 2313 board
   Note:     To contact me, mail to
                 takefuji@sfc.keio.ac.jp
*/

#include <io2313.h>
```

```
#include <interrupt.h>
#include <signal.h>
#include <iso646.h>

/*  4Mhz CPU, 9600 baud, 8-bit, 1-stop, 0 parity  UBRR=25 */

SIGNAL(SIG_UART_RECV)
/* signal handler for receive complete interrupt */
{
    unsigned char led;

   led = inp(UDR);        /* read byte for UART data buffer */

   outp(led, PORTB);     /* output received byte to PortB (LEDs) */
}

void uart_init(void)
/* initialize uart */
{
   /* enable RxD/TxD and ints */
   outp((1<<RXCIE)|(1<<TXCIE)|(1<<RXEN)|(1<<TXEN),UCR);
   /* set baud rate */
    outp(25, UBRR);
}


int main(void)
{
   outp(0xff ,DDRB);      /* PortB output */
   outp(0x00, PORTB);     /* switch LEDs on */

   uart_init();
    sei();                    /* enable interrupts */

   for (;;) {            /* loop forever */
     }
```

```
}
```

課題：PC から AT90S2313 に送られてきた 8-bit データを PC に送り返すプログラムを構築せよ。
課題：さらに、送られてきた文字を LED で表示せよ。

３．４　おもちゃの C プログラム例４：LCD に表示する

PC から RS232C を介して転送されてきたデータを LCD 表示器に表示するプログラムを構築するための、LCD 表示用のドライバプログラムを示す。

```
/************************************************************
  *   Program:    LCD driver
  *   Created:    Oct. 20, 2000
  *   Author:     takefuji
  *   Comments:   Akizuki LCD driver
  *   PortB0-3=DB0-DB3, PortB4=RS, PortB5=E
  *  contrast adj pin should be grounded.
************************************************************/

#include "lcd.h"
#include <io.h>

void lcd_delay(unsigned int p)
{
  unsigned int i;
  byte j;
  for(i=0;i<p;i++)
    for (j=0;j<10;j++); //10 to 20
}

void toggle_E()
{
  outp(inp(PORTB) | 0x20,PORTB);        // E=1 PortB5=E
  outp(inp(PORTB) & 0xDF,PORTB);        // E=0
}
```

```
/*-----------------------------------------------
 *  Clear display
 *  PortB0=DB4, PortB1=DB5, PortB2=DB6, PortB3=DB7
 *  PortB4=RS (RS=0 instruction;RS=1 data)
 *-----------------------------------------------*/
void lcd_cls()
{
  outp(0x00,PORTB); toggle_E();
  outp(0x01,PORTB); toggle_E();
  lcd_delay(1000);
}

/*-----------------------------------------------
 *  Home position
 *-----------------------------------------------*/
void lcd_home()
{
  outp(0x00,PORTB); toggle_E();
  outp(0x02,PORTB); toggle_E();
  lcd_delay(1000);
}

/*-----------------------------------------------
 *  LCD & Cursor control (Blinking, ON/OFF,...)
 *-----------------------------------------------*/
void lcd_control(byte disonoff, byte curonoff, byte curblink)
{ byte temp;
  outp(0x00,PORTB);
  toggle_E();
  temp=0x08;
  if (disonoff==1) temp|=0x04;
  if (curonoff==1) temp|=0x02;
  if (curblink==1) temp|=0x01;
  outp(temp,PORTB);
  toggle_E();
```

```
  lcd_delay(100);
}

/*---------------------------------------------
 *  DD address:
 * row0: 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
 * row1: C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
 *   row 0, row 1; column 0-F
 *---------------------------------------------*/
void lcd_goto(byte row, byte column)
{
  if(row==0x00)
   {
  outp(0x08,PORTB);
  toggle_E();
  outp(column,PORTB);
  toggle_E();}
  if(row==0x01)
  {outp(0x0C,PORTB);
   toggle_E();
   outp(column,PORTB);
   toggle_E();}
   lcd_delay(100);
}

/*---------------------------------------------
 *  Put character on LCD
 *---------------------------------------------*/
void lcd_putch(byte data)
{ byte temp;
  temp=data>>4;           //upper 4-bit data -> PortB
  temp=temp | 0x10;       //RS=1 data transfer
  outp(temp,PORTB);
  toggle_E();
  temp=data & 0x0F;       //lower 4-bit data -> PortB
  temp=temp | 0x10;       //RS=1 data transfer
```

```
  outp(temp,PORTB);
  toggle_E();
  lcd_delay(100);
}

/*---------------------------------------------
 *  Display null terminated string
  *--------------------------------------------*/
void lcd_putstr(byte *data)
{
      while(*data) {
         lcd_putch(*data);
          data++;
      }
}

/*---------------------------------------------
 *  Display 8bit bin value
  *--------------------------------------------*/
void printbin(byte x)
{
  byte i;
  for (i=128;i>0;i>>=1)
    {
     if ((x&i)==0) lcd_putch('0');
     else lcd_putch('1');
    }
}

/*---------------------------------------------
 *  Display byte in hex
  *--------------------------------------------*/
void printhex(byte i)
{
    byte hi,lo;
```

```
    hi=i&0xF0;              // High nibble
    hi=hi>>4;
    hi=hi+'0';
    if (hi>'9')
        hi=hi+7;

    lo=(i&0x0F)+'0';        // Low nibble
    if (lo>'9')
        lo=lo+7;

    lcd_putch(hi);
    lcd_putch(lo);
}


/*-------------------------------------------------
 *  Display initialization
 *-------------------------------------------------*/
void lcd_init(void)
{
 outp(0xFF,DDRB);
 lcd_delay(10000);
 outp(0x03,PORTB); toggle_E(); lcd_delay(500);
 outp(0x03,PORTB); toggle_E(); lcd_delay(100);
 outp(0x03,PORTB); toggle_E(); lcd_delay(500);
 outp(0x02,PORTB); toggle_E(); lcd_delay(100);
 outp(0x02,PORTB); toggle_E();
 outp(0x08,PORTB); toggle_E(); lcd_delay(100);
 outp(0x00,PORTB); toggle_E();
 outp(0x08,PORTB); toggle_E(); lcd_delay(100);
 outp(0x00,PORTB); toggle_E();
 outp(0x01,PORTB); toggle_E(); lcd_delay(100);
 outp(0x00,PORTB); toggle_E();
 outp(0x07,PORTB); toggle_E(); lcd_delay(100);
 lcd_cls();
}
```

```
int main(void)
{
    lcd_init();
    lcd_control(1,1,1);
    lcd_delay(1000);
    lcd_cls();
    lcd_putstr("Hello from ﾀｹﾌｼﾞ");
    lcd_goto(0x01,0x01);
    lcd_putstr("hello from keio");
      while(1);
}
```

lcd.h のヘッダーファイルは次のようになる。

```
/*****************************************************************
  *   Program:    LCD driver header
  *   Created:    Oct. 20, 2000
   *    Author:     takefuji
  *   Comments:   functions & definitions
  *****************************************************************/
// LCD is connected on PORTB in 4bit mode
#include <io.h>
#include <stdio.h>
#include <stdarg.h>

typedef unsigned char byte;

void lcd_delay(unsigned int p);
void toggle_E();
void lcd_cls();
void lcd_home();
void lcd_control(byte disonoff, byte curonoff, byte curblink);
void lcd_goto(byte row, byte column);
void lcd_putch(byte data);
void lcd_putstr(byte *data);
void printbin(byte x);
void printhex(byte i);
```

```
void lcd_init(void);
```

３．５　おもちゃの C プログラム例５：電力線搬送の X10 ドライバプログラム

家電装置をコントロールできる、一番簡単な電力線搬送プログラムを構築する。ここで使う技術は、X10 と呼ばれる電力線搬送方式である。

```
/*****************************************************************************
  *   Program:    X10 driver
  *   Created:    Oct. 26, 2000
  *   Author:     takefuji
  *   Comments:   X10 PL513/TW523 driver
  *                    PortD6=zero-detect(input),          PortD5=send(output),
PortD4=switch_on_off(input)
  * PL513(1:Black zero-detect; 2:Red GND; 3:Green GND; 4:Yellow Tx)
   * TW523(1:Black zero-detect; 2:Red GND; 3:Green Rx; 4:Yellow Tx)
  * three 1ms width pulses from rise of 0-detect (1.778ms blank separation
between them)
  * technical note of X-10 (powerhouse) has the details for pulse coding on
page 7
  * and commands on page 5.
  * start (1110), house codes A (0 1 1 0)=>(01 10 10 01)
  * key codes All units off (0 0 0 0 1)=>(01 01 01 01 10)
  * key codes All lights on (0 0 0 1 1)=>(01 01 01 10 10)
  * start+house+key+start+house+key for a command using 22 cycles
   *   2      4      5     2      4     5   = 22 cycles
*****************************************************************************
************/

#include <stdio.h>
#include <stdarg.h>
#include <io.h>
typedef unsigned char byte;

void x10_delay(unsigned int p)
```

```
{
  unsigned int i;       //one loop is 0.004587ms with 10Mhz
  byte j;
  for(i=0;i<p;i++)
    for (j=0;j<10;j++);
}

void x10_ldelay(unsigned int p)
{
  unsigned int i;       //one loop is 0.0459ms with 10Mhz
  byte j,k;
  for(i=0;i<p;i++)
    for (j=0;j<10;j++)
        for (k=0;k<10;k++);
}

/*-----------------------------------------
 * three pulses with zero-detect=High
  *-----------------------------------------*/
void three_pulse(void)
{
 sbi(PORTD,5);
 x10_delay(218);                 //1ms
 cbi(PORTD,5);
 x10_delay(388);                //1.778ms
 sbi(PORTD,5);
 x10_delay(218);                 //1ms
 cbi(PORTD,5);
 x10_delay(388);                //1.778ms
 sbi(PORTD,5);
 x10_delay(218);                 //1ms
 cbi(PORTD,5);
 x10_delay(22);                  //0.1ms

}
```

```
/*-----------------------------------------
 * No pulse with zero-detect=High
  *-----------------------------------------*/
void nopulse(void)
{
 x10_delay(218);                    //1ms
}

/*-------------generate pulse stream----------------*/
void pulse_strm(byte *data)
{
  while(*data) {
   if((*data)=='1'){
                    while((inp(PIND) & 0x40)==0x00){};   //if zero-detect 0=>1
then next
               three_pulse();
                    while((inp(PIND) & 0x40)==0x40){};   //if zero-detect 1=>0
then next
                nopulse();
                data++;}
   if((*data)=='0'){
                    while((inp(PIND) & 0x40)==0x00){};   //if zero-detect 0=>1
then next
                nopulse();
                    while((inp(PIND) & 0x40)==0x40){};   //if zero-detect 1=>0
then next
               three_pulse();
                data++;}
               }
}

/*-----------------------------------------------
 *  X10 initialization
  *-----------------------------------------------*/
void x10_init(void)
```

```
{
  outp(0x2F,DDRD);                  //PortD4=input,      PortD5=send(output),
PortD6=zero-detect(input)
  sbi(PORTD,4);      //pullup input for PortD4
  sbi(PORTD,6);      //pullup input for PortD6

}

void x10_start(void)
{
/*----------------start 1110--------------------------*/
    while((inp(PIND) & 0x40)==0x00){}; //if zero-detect 0=>1 then next
  three_pulse();
    while((inp(PIND) & 0x40)==0x40){}; //if zero-detect 1=>0 then next
  three_pulse();
    while((inp(PIND) & 0x40)==0x00){}; //if zero-detect 0=>1 then next
  three_pulse();
    while((inp(PIND) & 0x40)==0x40){}; //if zero-detect 1=>0 then next
  nopulse();
}

void turn_on(void)
{
x10_start();
/*----------0110 00011 (ON)-------------------------*/
pulse_strm("011000011");
 x10_start();
/*----------0110 00011 (ON)-------------------------*/
 pulse_strm("011000011");

 x10_ldelay(20000);
 x10_ldelay(20000);
 x10_ldelay(20000);

}
```

```
void turn_off(void)
{
x10_start();
/*----------0110 00001 (OFF)-------------------------*/
 pulse_strm("011000001");
x10_start();
/*----------0110 00001 (OFF)-------------------------*/
 pulse_strm("011000001");

 x10_ldelay(20000);
 x10_ldelay(20000);
 x10_ldelay(20000);
}

int main(void)
{
 x10_init();

 for(;;)
  {
  if((inp(PIND) & 0x10)==0x10) turn_off();      //switch open
  if((inp(PIND) & 0x10)==0x00) turn_on();       //switch pressed
 };
}
```

３．６　サーボコントロールの C プログラム例６

サーボモータをコントロールして、実際に動くものを作ってみる。サーボモータは、パルス幅で制御されており、サーボの方向・角度は２０ミリ秒ごとに送られてくるパルス幅により決定される。パルス幅は例えば、１ｍｓで時計の長針の５３分、２ｍｓで７分方向にサーボを動かすことができる。６０度方向を変えるには、０．６秒ぐらいは時間がかかる。ここでは、２０ｍｓおきにパルスを発生させ９０度をゆっくり往復するプログラム例を示す。

```
/*****************************************************************
```

```
 *   Program:    servo driver
 *   Created:    March 17.2001
 *   Author:     takefuji
 *   Comments:   servo driver
 *  Servo receives the control signal every 20 ms.
 *  1ms pulse width= 0 degree =218,1.5ms=45 degree=327
  *    2ms pulse width=90 degree=436
**********************************************************************/

#include <stdio.h>
#include <stdarg.h>
#include <io.h>
typedef unsigned char byte;

void delay(unsigned int p)
{
 unsigned int i;       //one loop is 0.004587ms with 10Mhz
 byte j;
 for(i=0;i<p;i++)
   for (j=0;j<10;j++);
}

void ldelay(unsigned int p)
{
 unsigned int i;       //one loop is 0.0459ms with 10Mhz
 byte j,k;
 for(i=0;i<p;i++)
   for (j=0;j<10;j++)
     for (k=0;k<10;k++);
}

/*-----------------------------------------------
 *  servo initialization
 *-----------------------------------------------*/
void init(void)
{
```

```
  outp(0x7e,DDRD);   //PortD1-6=output, PortD0=input
  outp(0xff,DDRB);
  sbi(PORTD,0);      //pullup input for PortD0

}

void zero(void)
{
  unsigned int i;
  for (i=0;i<109;i++)
    {
    outp(0x7e,PORTD);  //PortD will be on.
     delay(218);        //1ms=218
    outp(0x00,PORTD);  //PortD will be off.
    ldelay(414);       //19ms
    }
}

void zto45(void)
{
  unsigned int i;
  for (i=0;i<109;i++)
    {
    outp(0x7e,PORTD);  //PortD will be on.
     delay(219+i);       //1.5ms=327
    outp(0x00,PORTD);  //PortD will be off.
    ldelay(403);       //18.5ms
    }
}

void forty5to90(void)
{
  unsigned int i;
  for (i=0;i<109;i++)
    {
    outp(0x7e,PORTD);
```

```
   delay(328+i);        //2ms=436
   outp(0x00,PORTD);
   ldelay(392);         //18ms
   }
}

void nineto45(void)
{
  unsigned int i;
  for (i=0;i<109;i++)
   {
   outp(0x7e,PORTD);  //PortD will be on.
   delay(435-i);       //1.5ms=327
   outp(0x00,PORTD);  //PortD will be off.
   ldelay(403);       //18.5ms
   }
}

void forty5to0(void)
{
  unsigned int i;
  for (i=0;i<109;i++)
   {
   outp(0x7e,PORTD);  //PortD will be on.
   delay(326-i);       //1ms=218
   outp(0x00,PORTD);  //PortD will be off.
   ldelay(414);       //19ms
   }
}

int main(void)
{
 init();
  zero();
 for(;;)
  {
```

```
  zto45();
  forty5to90();
  nineto45();
   forty5to0();
 };
}
```

３．７　おもちゃの C プログラム例７

ワンダースワンに接続してみる。

３．８　おもちゃの C プログラム例８

AD 社の２軸 gyro センサと LCD を AT90S2313 に接続して簡単なゲームを構築してみる。

３．９　おもちゃの C プログラム例９

インターネットに接続してみる。

２．　１０　おもちゃの C プログラム例１０