名称	入手先	正式名称
Atmega8	ストロベリー・リナックス/マルツパーツ館	ATMEL　ATMEGA8（DIP） [ATMEGA8-16PI]
ブレッドボード	秋月電子	ブレッドボード　EIC－301
USBコネクタ（Bタイプメス）	秋月電子	基板取付用USBコネクタ（Bタイプ，メス）
ユニバーサル基板	秋月電子	片面ユニバーサル基板　Cタイプ（72x47mm）
１2MHzクリスタルx 1	秋月電子	クリスタル　１2MHz（10個入）
１2MHzセラロックx 1	秋月電子	セラミック発振子（セラロック）コンデンサ内蔵タイプ　１2MHz
1.5KΩ x 1	秋月電子	カーボン抵抗（炭素皮膜抵抗）1／6W 1.5KΩ（100本入） [RD16S 1K5]
68Ω x 2	千石電商	タクマン電子　カーボン抵抗 1／4W 68Ω±5%　RD25 68Ω(100本入)
4.7KΩ x 1	秋月電子	カーボン抵抗（炭素皮膜抵抗）1／6W 4.7KΩ（100本入） [RD16S 4K7]
3.3Vレギュレータx 1	秋月電子	3端子レギュレータ　3．3V　500mA
セラミックキャパシタ1.5uF x 4	秋月電子	積層セラミックコンデンサー 1.5μF 25V（10個入）
セラミックキャパシタ22pF x 2	千石電商	三菱マテリアル　セラミックコンデンサ 50V22pF　HE40SJSL220J
単線(0.65mm)	タイガー無線	－
Atmega168	ストロベリー・リナックス/マルツパーツ館	ATMEL　ATMEGA168（DIP） [ATMEGA168-20PI]
Attiny2313	秋月電子	AVRマイコン　ATTINY2313－20PU
その他材料(LED・センサなど)	秋月電子	－
テスター	秋月電子	ポケット・デジタルマルチメータ　P－16


☆マルツパーツ館
・AVRマイコン(ATMega8)
・AVRマイコン(ATMega168)
・AVRマイコン(ATTiny85)
☆秋月電子
・ブレッドボード
・抵抗
・キャパシタ(コンデンサ)
・3.3Vレギュレータ
・USBコネクタ（Bタイプメス）
・ユニバーサル基板
・12MHzクリスタル
・12MHzセラロック
・AVRマイコン(ATTiny2313)
・テスター
☆マルツパーツ館(２号店)
・AVRマイコン(ATMega8)
・AVRマイコン(ATMega168)
・AVRマイコン(ATTiny85)
☆千石電商
・抵抗
・キャパシタ(コンデンサ)
・12MHzクリスタル
・12MHzセラロック
☆タイガー無線
・0.65mm単線
☆ITプラザ
・AVRマイコン(ATMega168)
☆若松通商(3F)
・キャパシタ(コンデンサ)
秋葉原
中央通り
総武本線
秋葉原UDX
秋葉原クロスフィールド
秋葉原ラジオ会館

電子おもちゃ

武藤佳恭

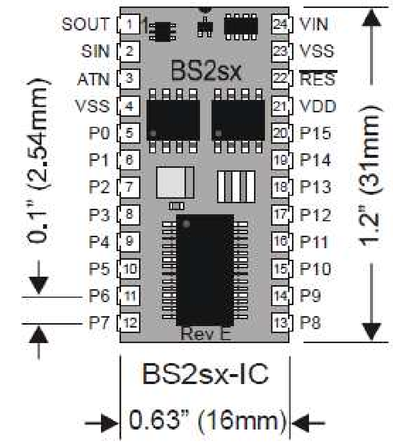
電子おもちゃ支援情報

- http://neuro.sfc.keio.ac.jp/kenkyukai/toy.htmlをクリックします。
- “はじめに”をクリックします。
- cygwinとWinAVRをインストールします。
- デスクトップ上のprogrammersNotepadをダブルクリックして、led0ディレクトリのled.cとmakefileをコンパイルしてみましょう。

一番簡単な開発環境

- Basic Stamp for Windows, Mac, Linux
- USB-RS232c ケーブル　1200円（秋月）

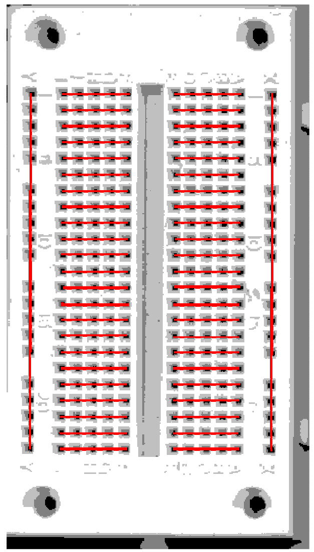
- BS2 3900円（秋月）
- BS2SX　4700円（秋月）



USB

USB-RS232c

ブレッドボード



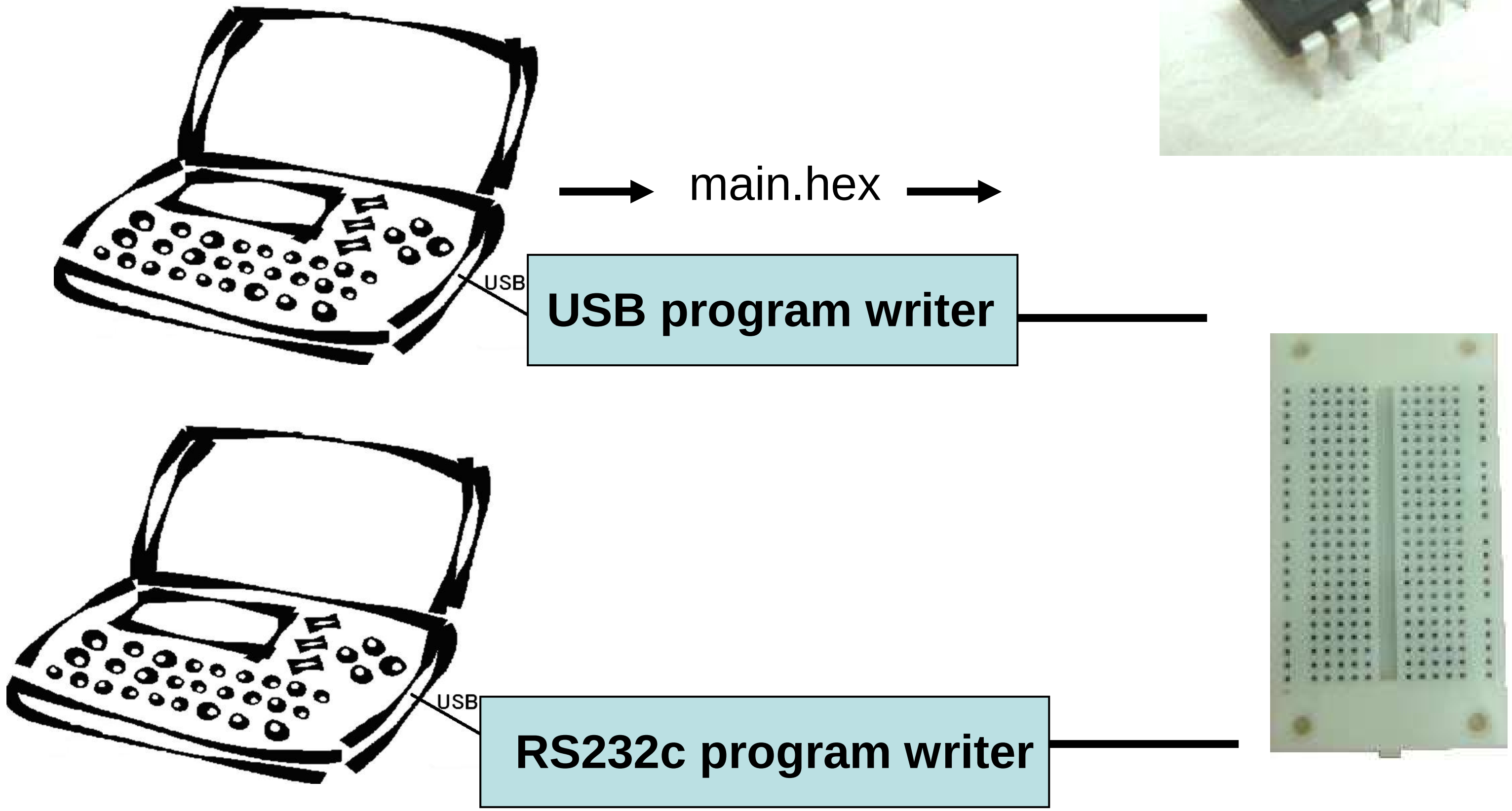
マイクロコントローラ(AVR)開発環境installation

- cygwin(libusb)
- winAVR
- プログラムライター(PCからAVRへプログラム転送:main.hex)

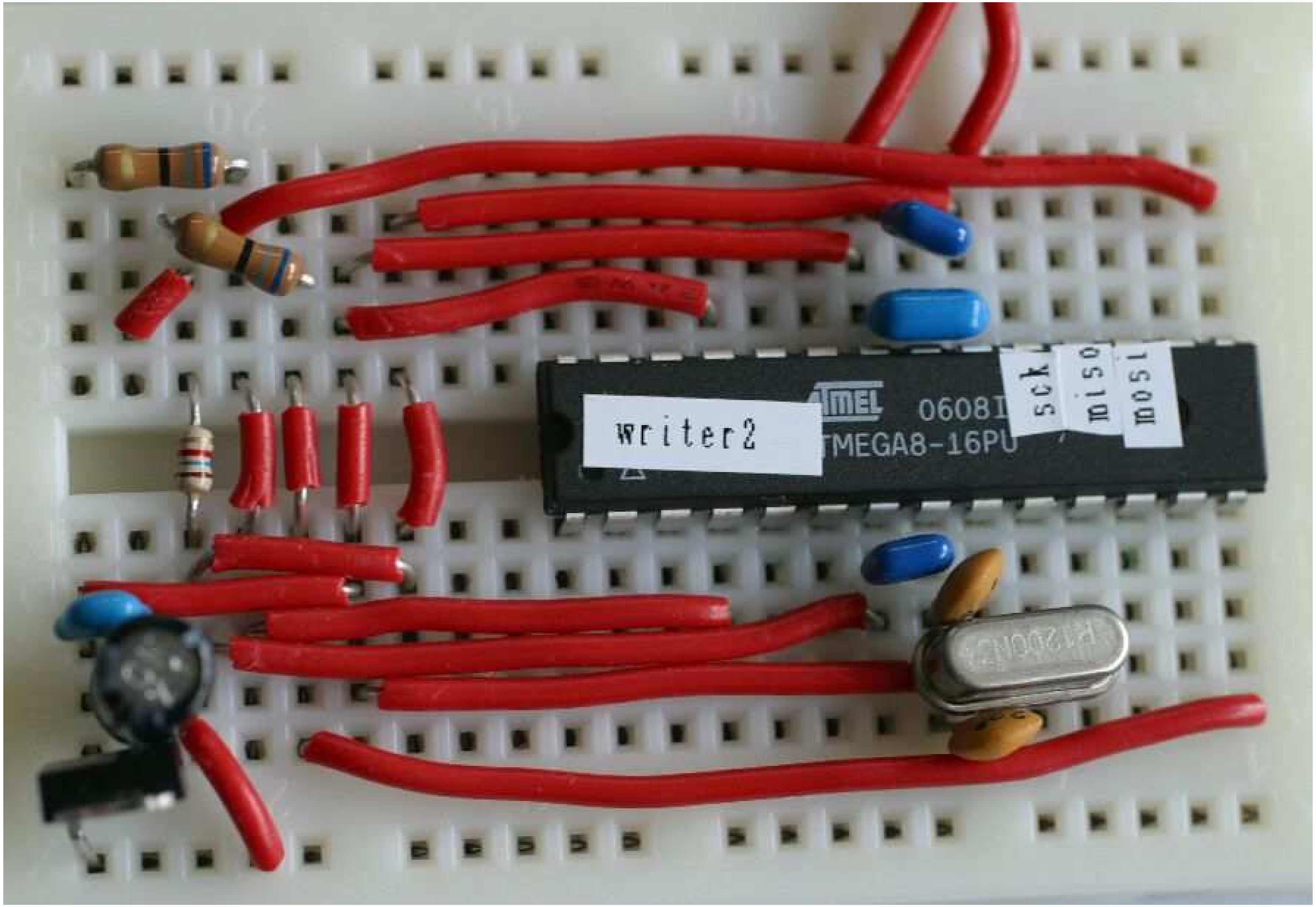
- ATtiny2313 120円(秋月) 20pin 2k 15port
- ATtiny26L　260円(秋月) 20pin 4k 16port AD
- ATmega168 500円(ストロベリー) 28pin 16k 23p AD
- ATmega88　400円(ストロベリー) 28pin 8k 23p AD
- ATmega8　400円(ストロベリー) 28pin 8k 23p AD
- ATmega48　300円(ストロベリー) 28pin 4k 23p AD
- ATtiny45 1100円/4 (ITプラザ) 8pin 4k 6port AD
- ATtiny13　200円(ITプラザ) 8pin 1k 6port AD

開発環境

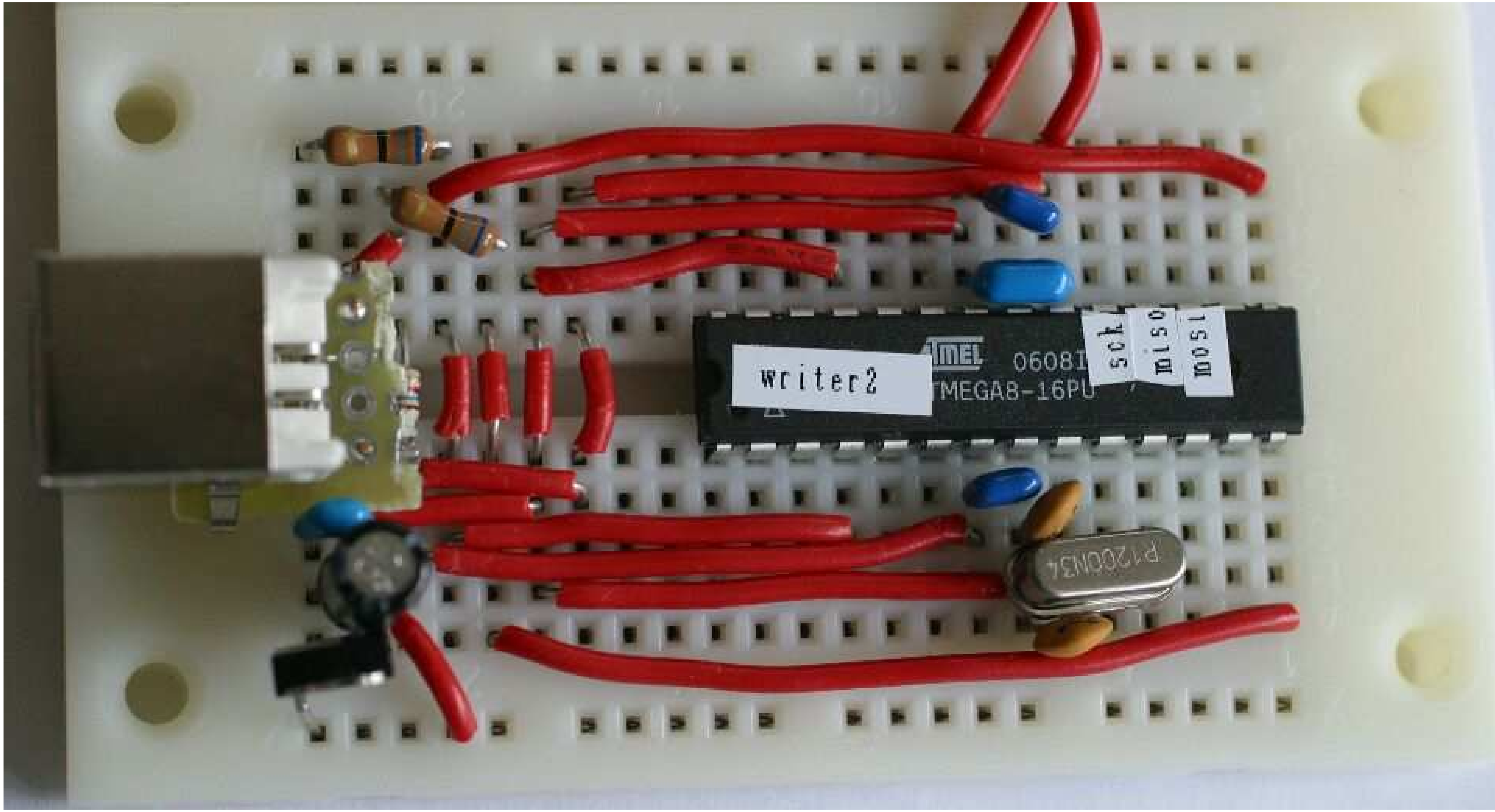
- USB program writer
- RS232c program writer

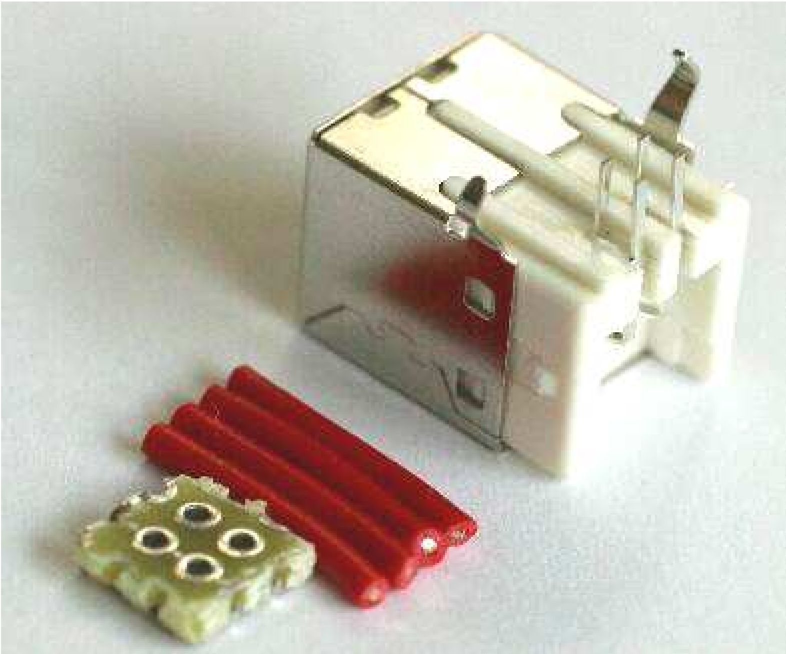


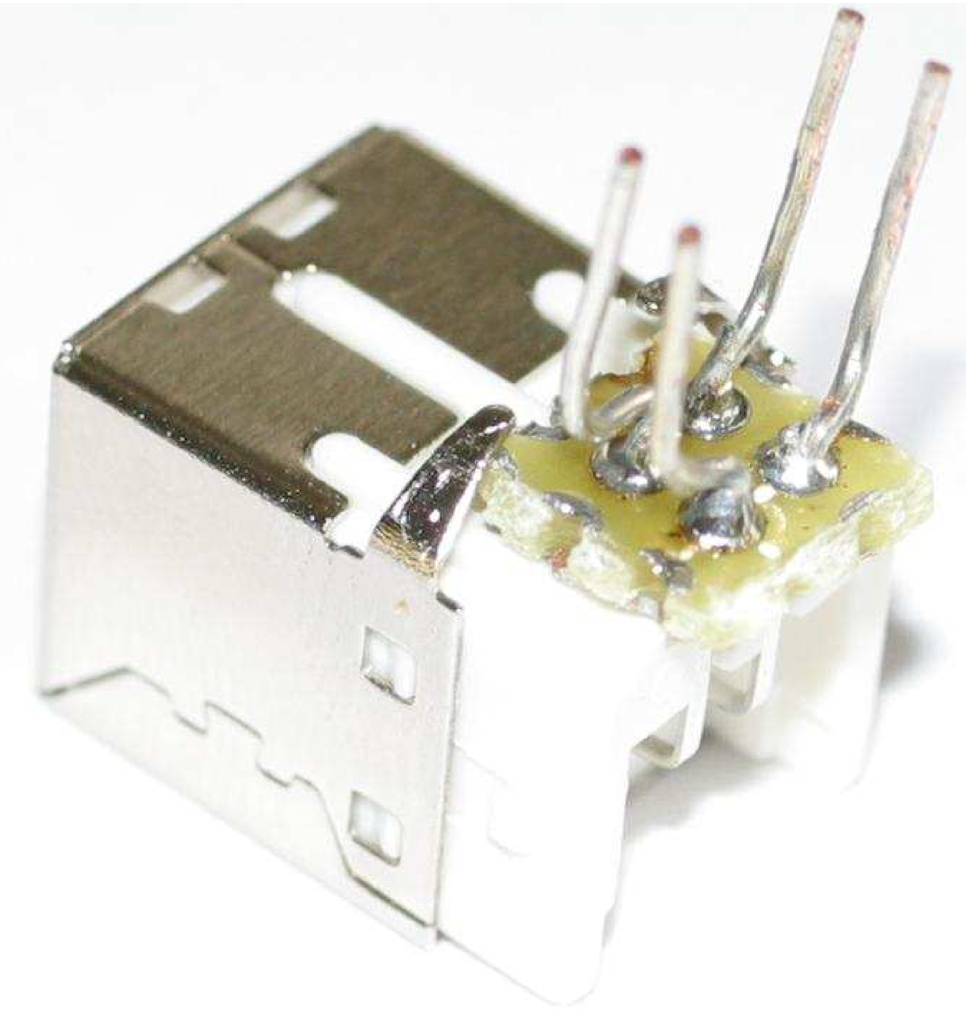
USB program writer

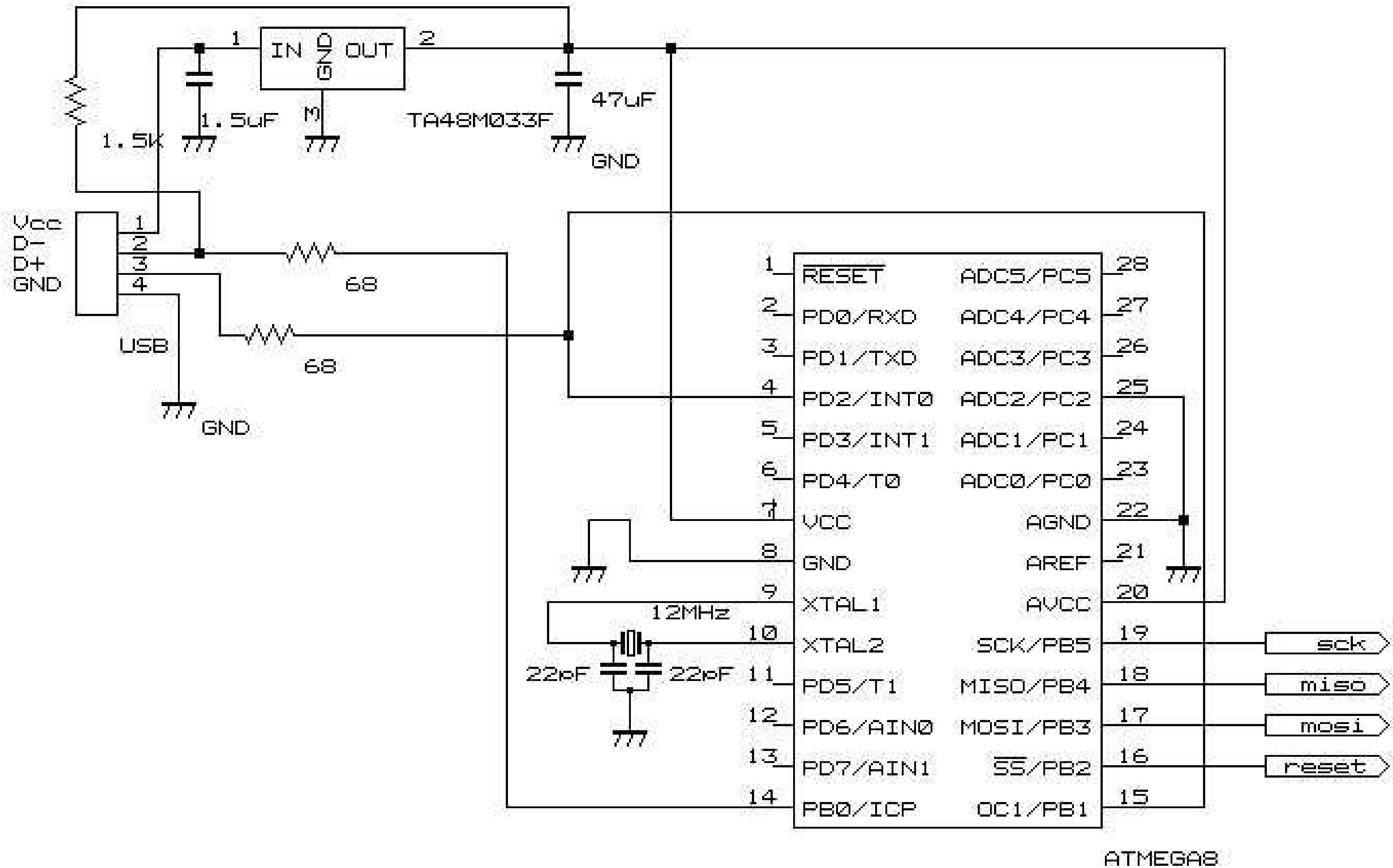


USB program writer





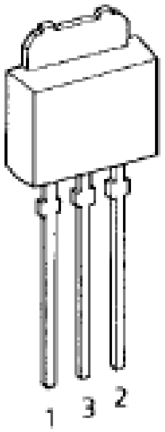


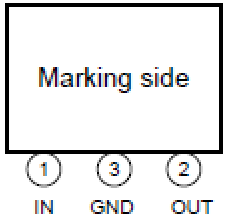


ATmega8 USB program writer by Takefuji

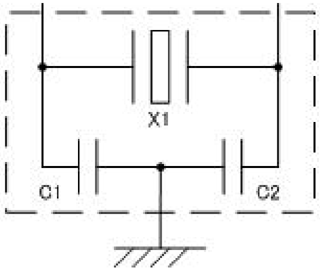
Program writerに必要な部品

- Atmega8（DIP２８ピン）
- ブレッドボード（45mm x 85mm 270穴)
- USBコネクタ（Bタイプ　メス）
- ユニバーサル基板
- １２MHzクリスタル（2ピン）x １
- １２MHzセラロック x １
- 1.5KΩ x 1, 68Ω x 2, 4.7KΩ x 1
- 3.3Vレギュレータ x １
- セラミックキャパシタ1.5uF x 4, 22pF x 2
- 単線

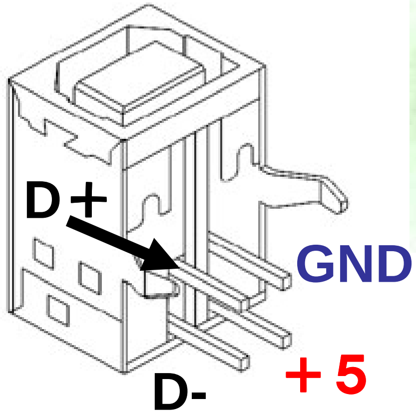




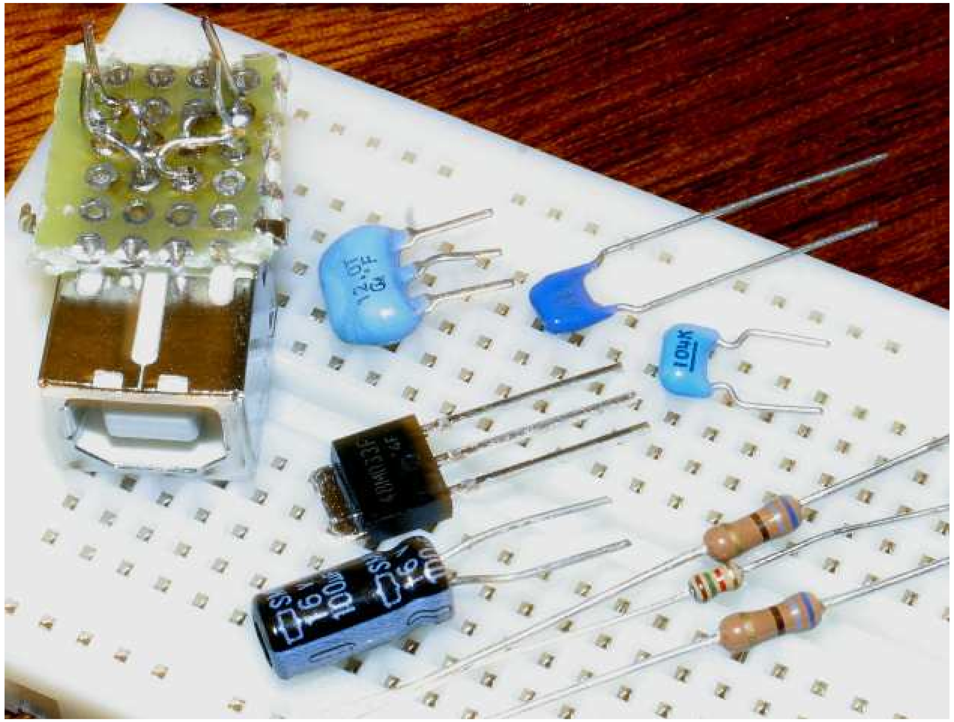
TA48M033F



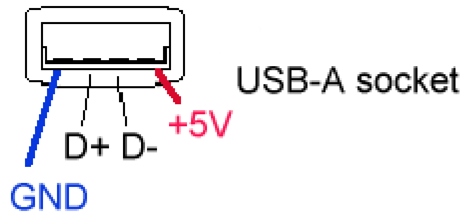
Ceralock



USB B Type PCB Female



USB socket on the computer



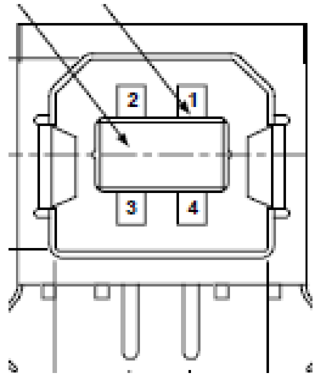


Table 6-1. USB Connector Termination Assignment

Contact Number	Signal Name	Typical Wiring Assignment
1	VBUS	Red
2	D-	White
3	D+	Green
4	GND	Black
Shell	Shield	Drain Wire

マイクロコントローラ（AVR) USB開発環境installation

- Cygwin:libusb
- winAVR
- プログラムライター（PCからAVRへプログラム転送：main.hex）
- /usr/lib/libusb/inf-wizard.exe for device driver generation
- c:¥cygwin¥lib¥libusb¥inf-wizard.exe

注意点：usbasp最新版のwin-driverを読み込む（libusb0.dll, libusb0.sys, usbasp.inf）

gccの使い方for firmware

Firmware main.hexの作り方

- WinAVRをインストールしたら、デスクトップ上のProgramersNotepadをダブルクリックする。
- 解凍したgcctest.zipのled0ディレクトリのled.cとmakefileファイルを開く。(Ctrl +O)
- Toolsバーの[WinAVR] Make Allを実行する。

led.hexファイルが出来ているはずです。

gccとavrdudeの使い方for firmware

Firmware main.hexの作り方

- avr-gccを立ち上げます。

make

avr-gcc….

- 書き込み(main.hexをマイクロコントローラに書き込み）

avrdude –c usbasp –p 2313 –U flash:w:main.hex:a

- 書き込み（ATmega8のfusesに書き込む）

make fusesをコマンドで実行

avrdude -c usbasp -p m8 -P /dev/usb/ttyUSB0 -u -U hfuse:w:0xc9:m -U lfuse:w:0xef:m

Applicationソフトウエアの作り方

- cygwinを立ち上げます。

gcc -o xxx xxx.c –lusb

device driverは、inf-wizard.exeを使って作ります。

- usbconfig.hの注意点

USB_CFG_DMINUS_BITは0 である必要がある。

portBであればPB0、portDであればPD0

USB_CFG_DPLUS_BITはint0に接続する必要がある。

データ転送（パソコン→AVR)

パソコン側ソフトウエア

- usb_control_msg(d,USB_TYPE_VENDOR|USB_RECIP_DEVICE|USB_ENDPOINT_IN,i, (0xff & j)|(0xff00 & (256*k)), (0x0ff & l)|(0xff00 & (256*m)),(char *)buffer, (0x0ff & n)|(0xff00 & (256*o)),5000);

↓

AVRファームウエア

- usbFunctionSetup(uchar data[8])

data[1]=i,data[2]=j,data[3]=k,data[4]=l,data[5]=m, data[6]=n,data[7]=o

データ転送(パソコン→AVR)

パソコン側ソフトウエア

```
usb_control_msg(d,USB_TYPE_VENDOR|USB_RECIP
  _DEVICE|USB_ENDPOINT_IN,i, (0xff & j)|(0xff00 &
  (256*k)), (0x0ff & l)|(0xff00 & (256*m)),(char *)buffer,
  (0x0ff & n)|(0xff00 & (256*o)),5000);
buffer[0]=replybuf[0],…,buffer[7]=replybuf[7]
```

AVRファームウエア

```
replybuf[0],…………..,replybuf[7]
usbFunctionSetup(uchar data[8])   {
static uchar replybuf[8];
usbMsgPtr=replybuf;               }
```

Device driver（PC側に常駐する）

- usbasp(プログラムライタ）のdevice driverは、

http://www.fischl.de/usbasp/から最新版のusbasp.tar.gzを解凍し、/bin/win-driverのドライバをインストールします。

- 他のdevice driverは、 inf-wizard.exeを使って、device driverを生成します。その生成したドライバをインストールします。

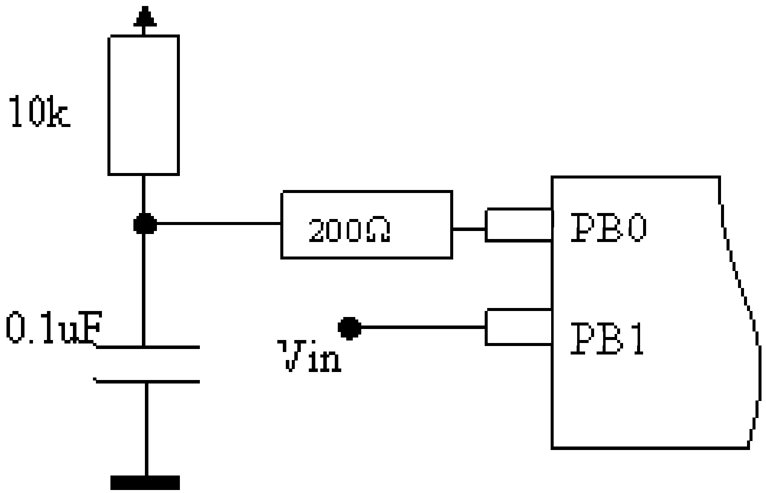
Device driverの問題点

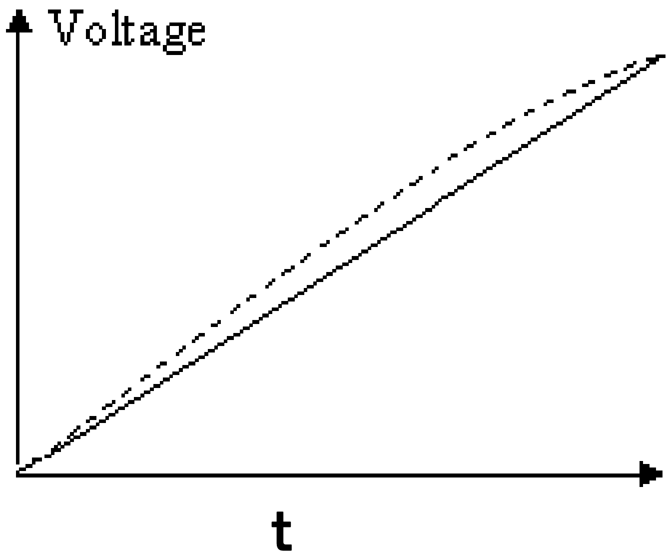
- usbasp(プログラムライタ）のdevice driverは、WinAVRのドライバに比べ古いために問題が起こる場合がある。

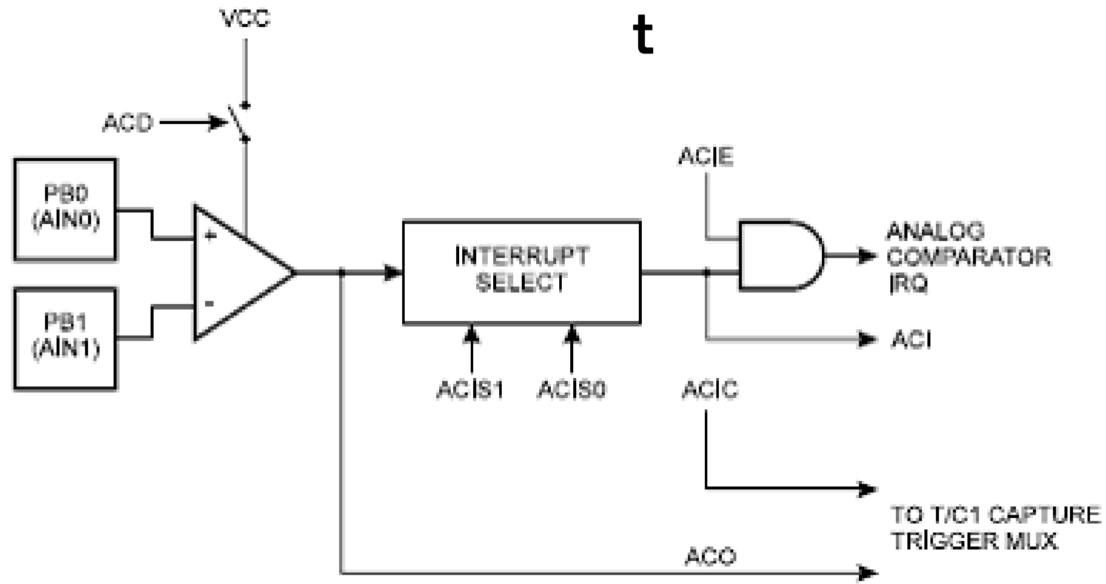
対処法：win-driverのusbasp.infファイルを変更します。（libusb00.sysにすべて変更）

同様に、libusb0.sysをlibusb00.sysに変更。

Analogデータのディジタルへの変換







積分回路

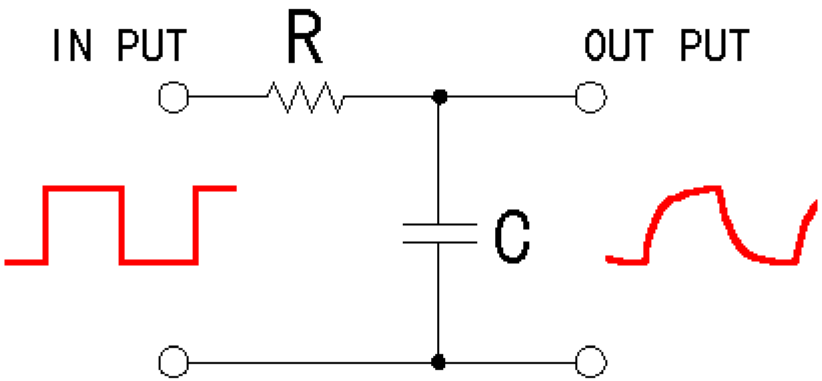
$$V = iR + Vc = RdQ/dt + Vc = Rd(CVc)/dt + Vc$$

$$Q = CVc$$

$$Ic = dQ/dt = i$$

$$V = RCdVc/dt + Vc$$

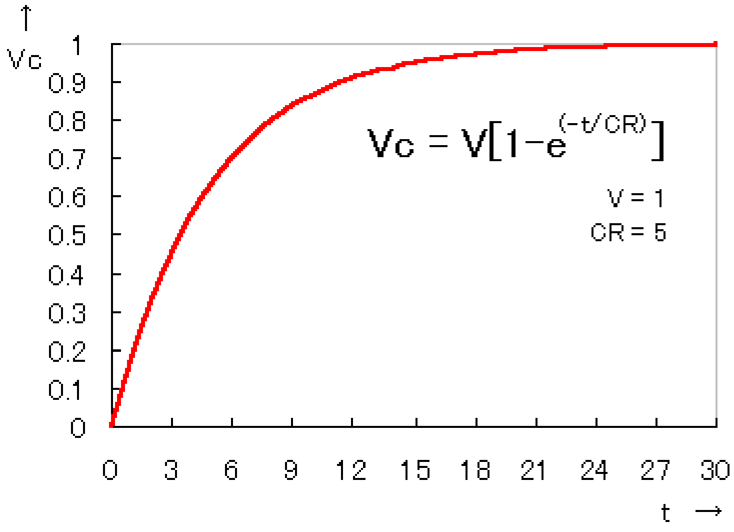
```
ode2('diff(v,t)+v=5, v, t);
```



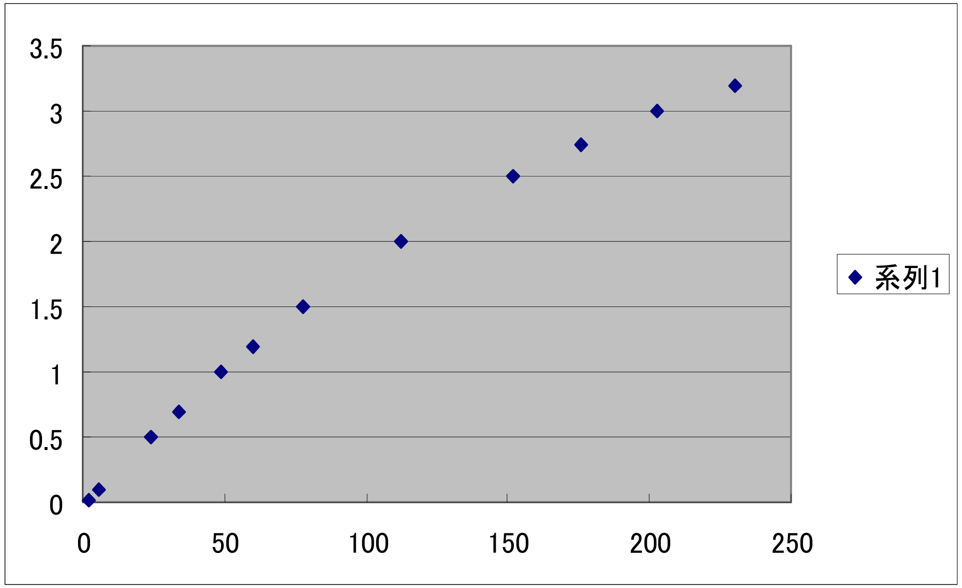
コンデンサ(C)の両端に加わる電圧(Vc)の変化は以下の式になります。

$$Vc = V[1-e^{-(t/CR)}]$$

上の式をグラフで表すと以下のようになります。

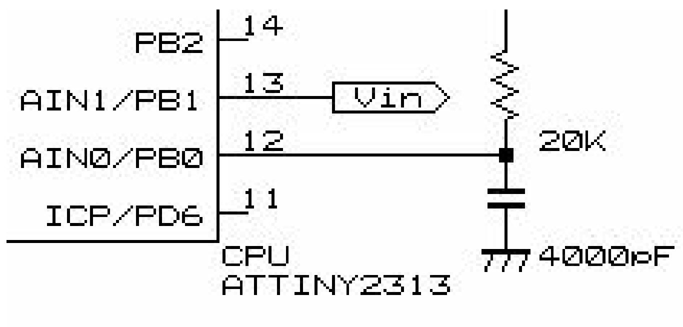


電圧

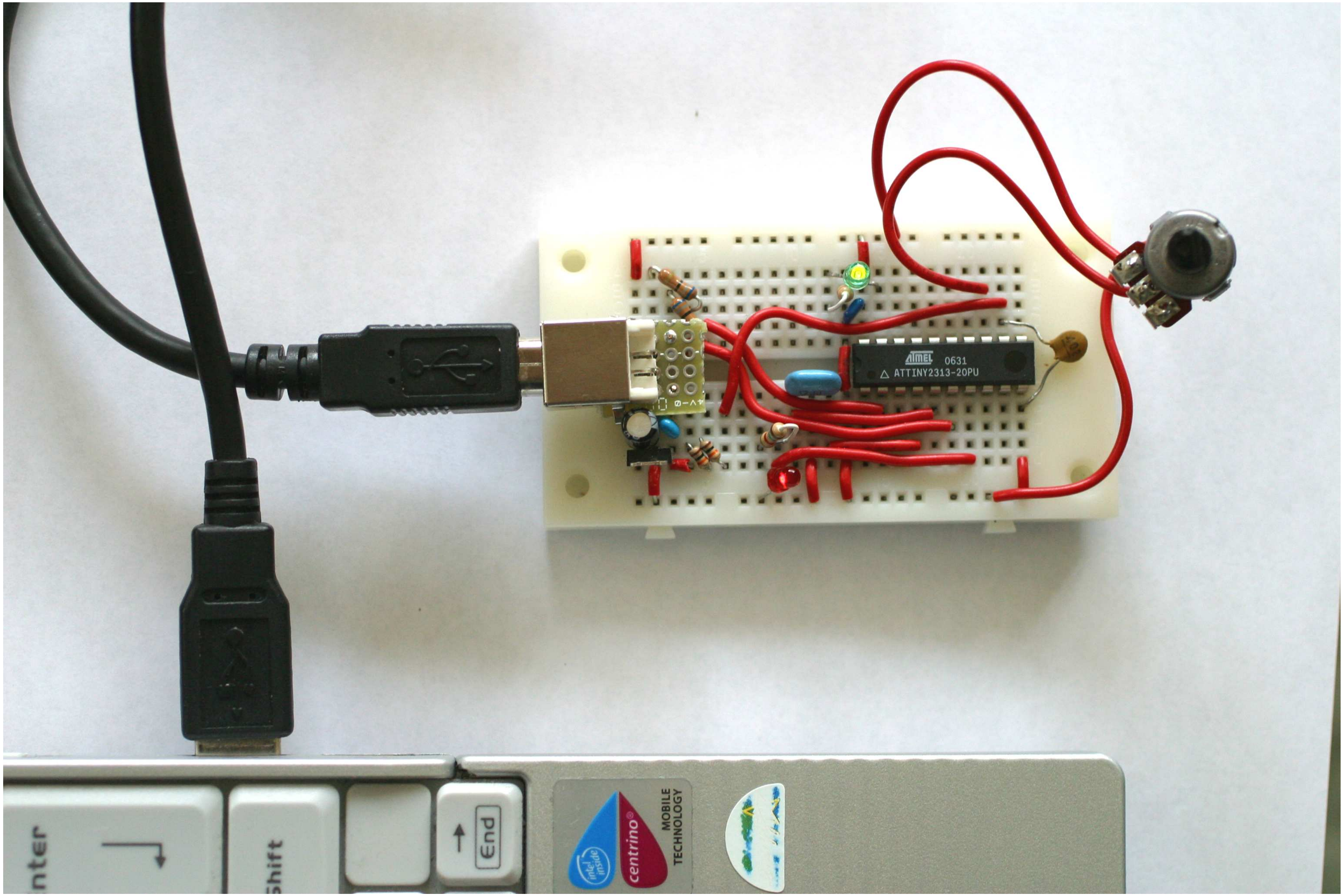


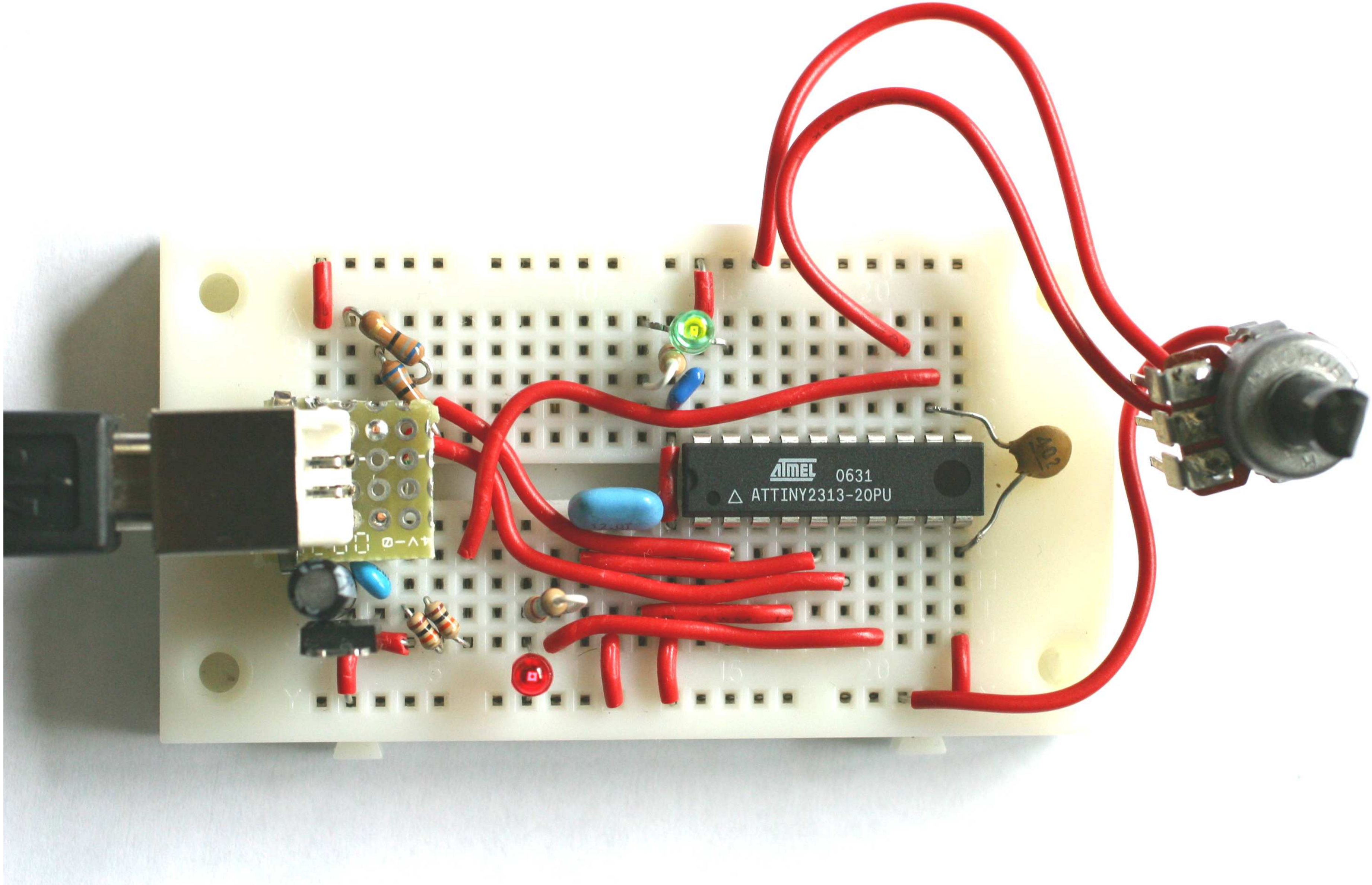
count

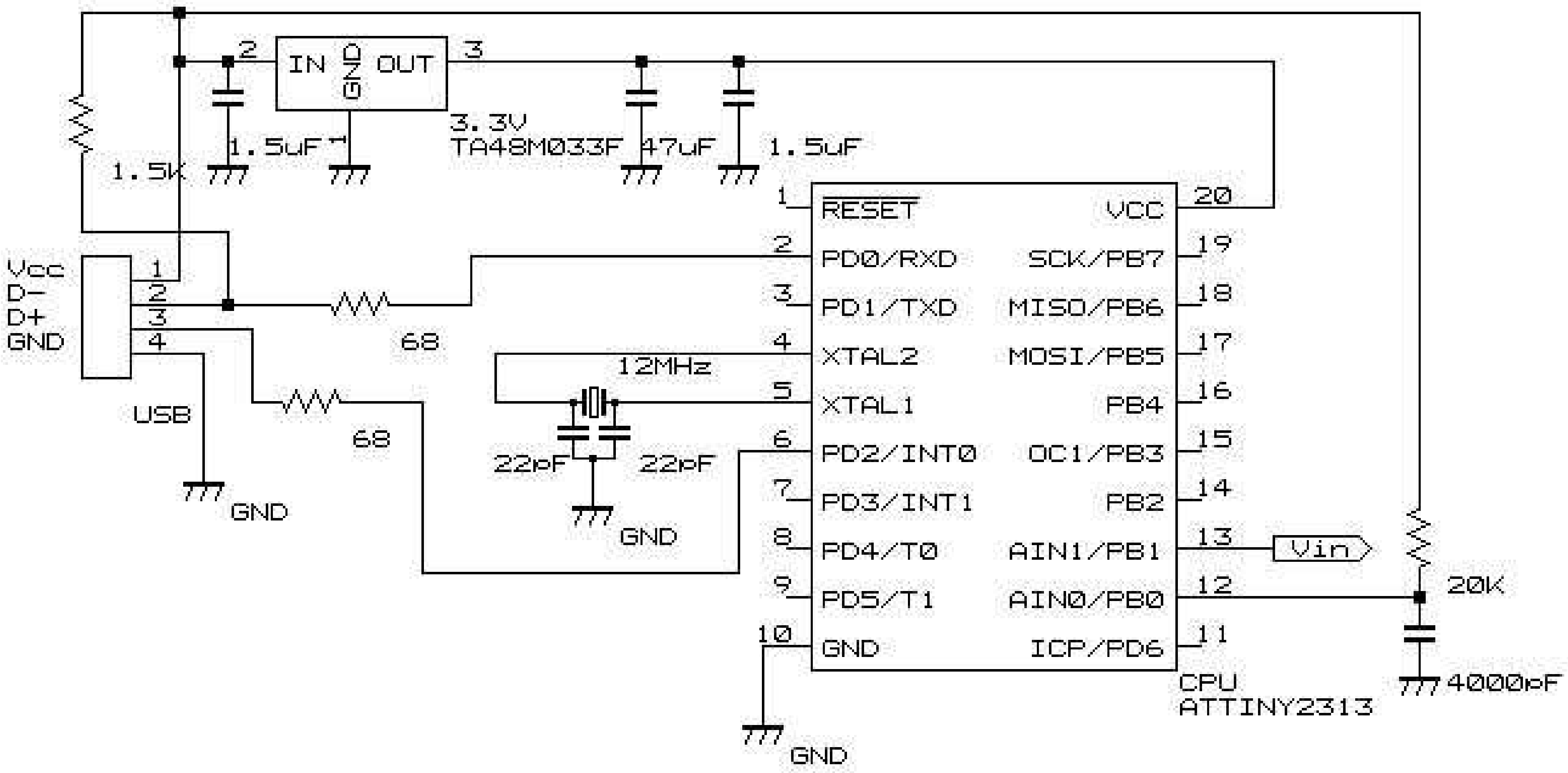
5V



AT90s2313 アナログ電圧の精度

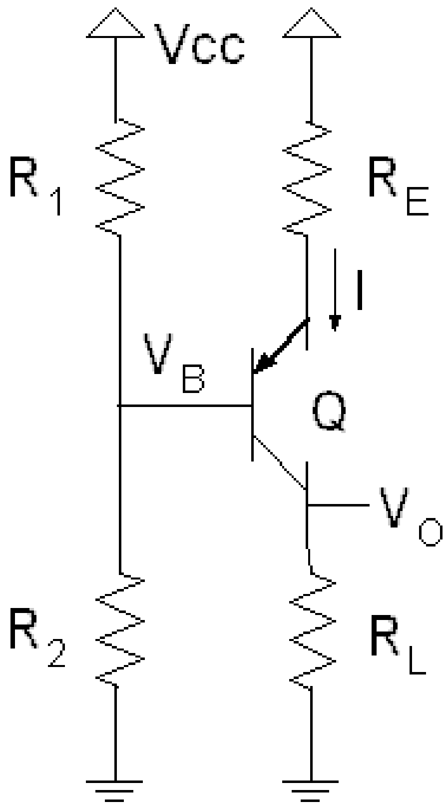

ATMEL
0631
ATTINY2313-20PU
centrino
MOBILE
TECHNOLOGY
End
Shift

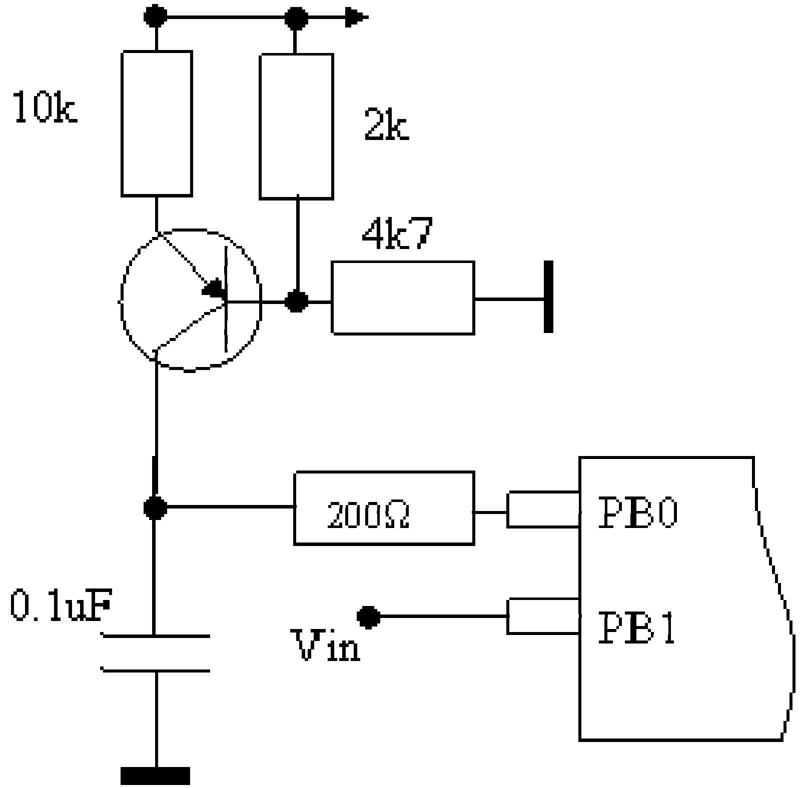

ATMEL 0631
ATTINY2313-20PU



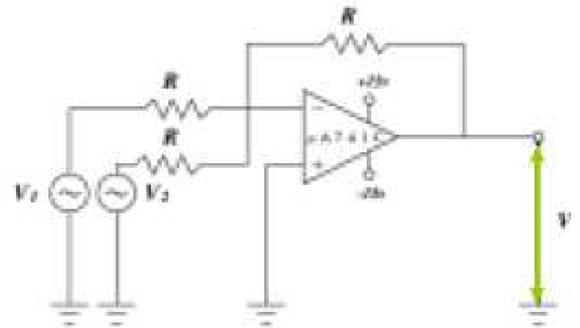
2313 analog入力回路

定電流回路

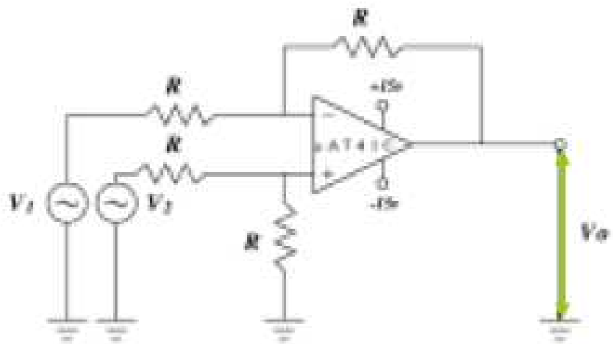




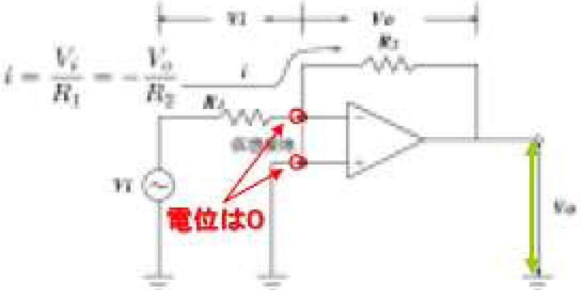
加算回路



減算回路



反転増幅回路



$A = \frac{V_o}{V_i} = -\frac{R_2}{R_1}$:位相が反転

加算回路の解析

一入力端子における電流の関係:

$$\frac{V_1}{R} + \frac{V_2}{R} = -\frac{V_o}{R}$$

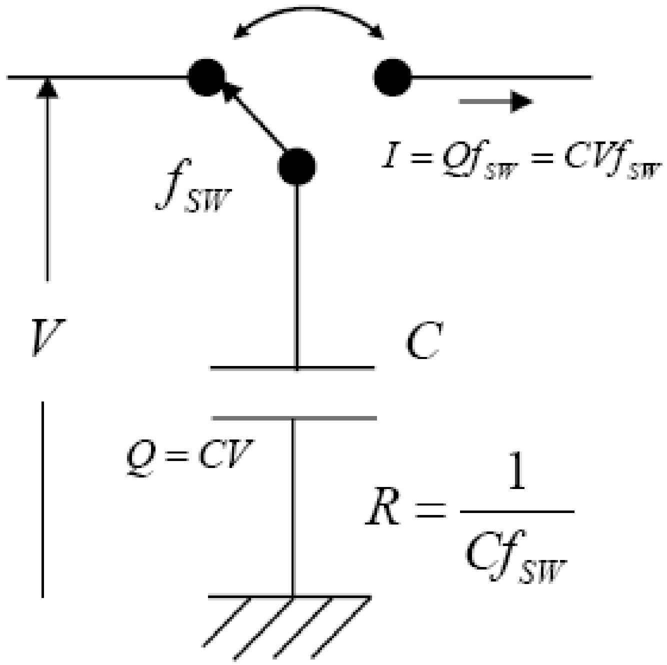
$$V_o = -(V_1 + V_2)$$

減算回路の解析

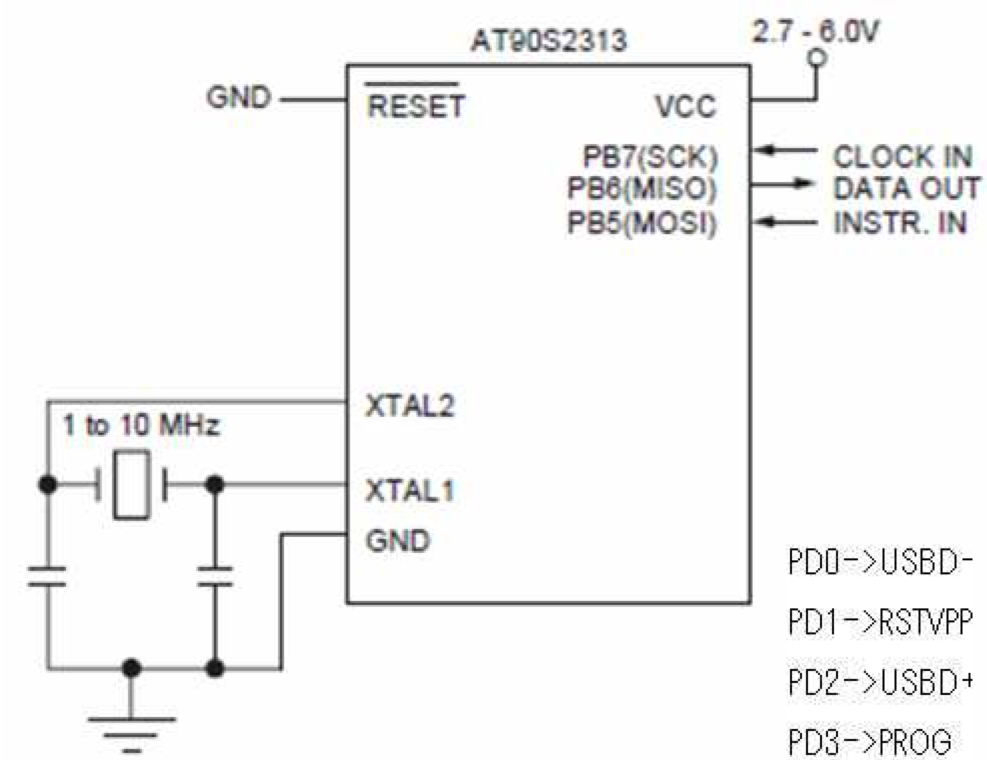
一入力端子における電流の関係:

$$\frac{1}{R}\left(V_1 - \frac{V_2}{2}\right) = \frac{1}{R}\left(\frac{V_2}{2} - V_o\right)$$

$$V_o = -(V_1 - V_2)$$



スイッチドキャパシタ回路の原理



PD0->USBD-

PD1->RSTVPP

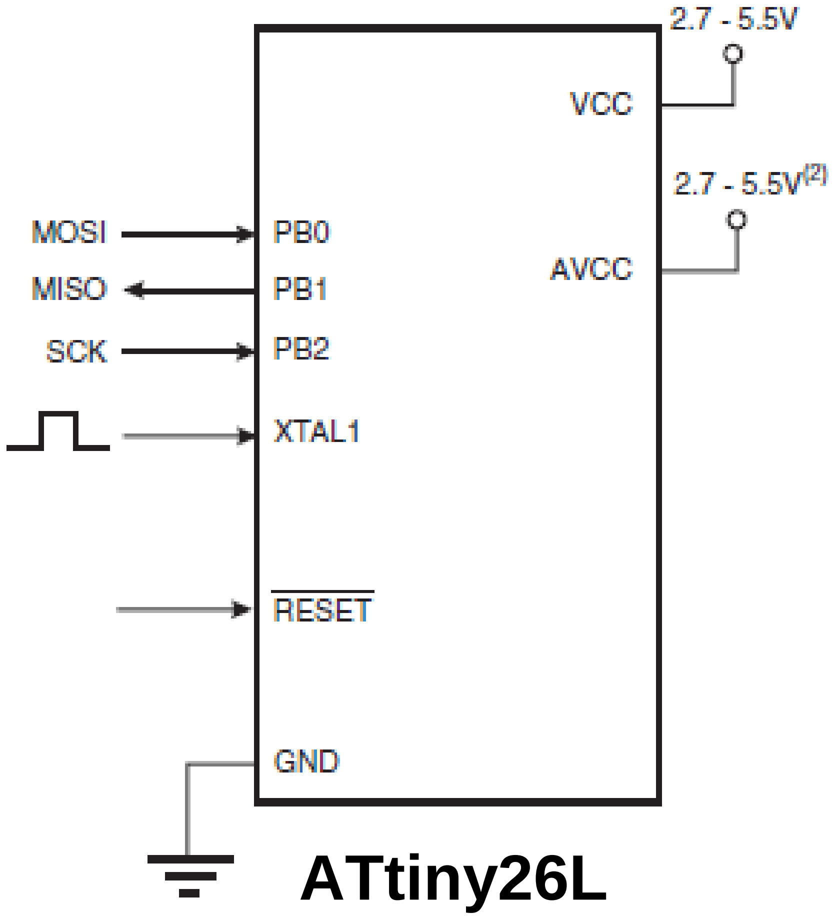
PD2->USBD+

PD3->PROG

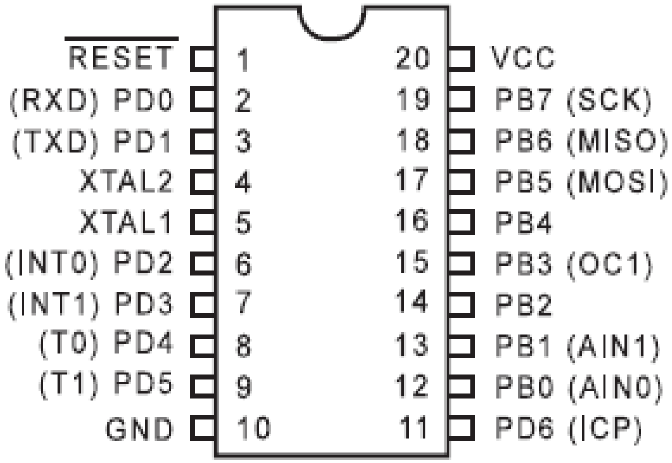
PD4->P33

PD5->P34

PD6->XTAL1

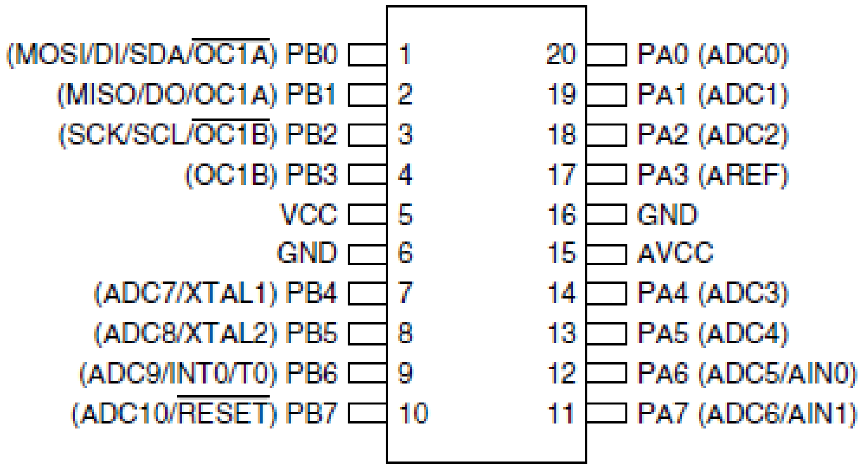


AT90S2313

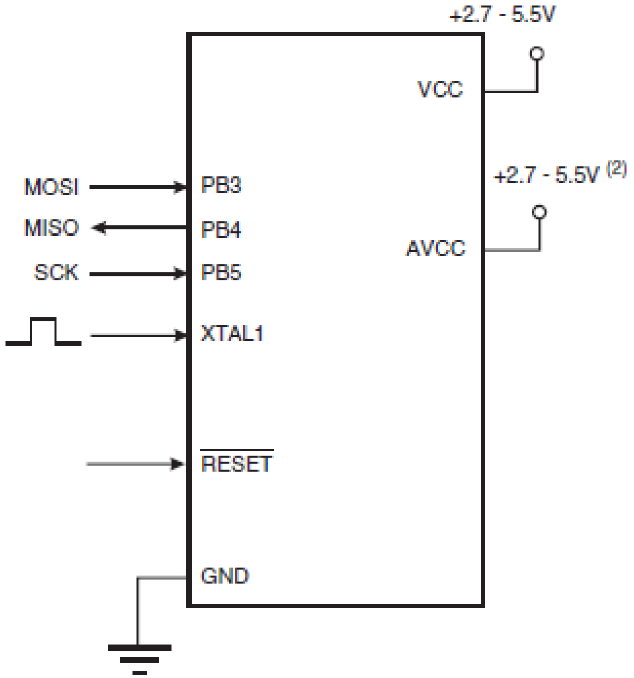


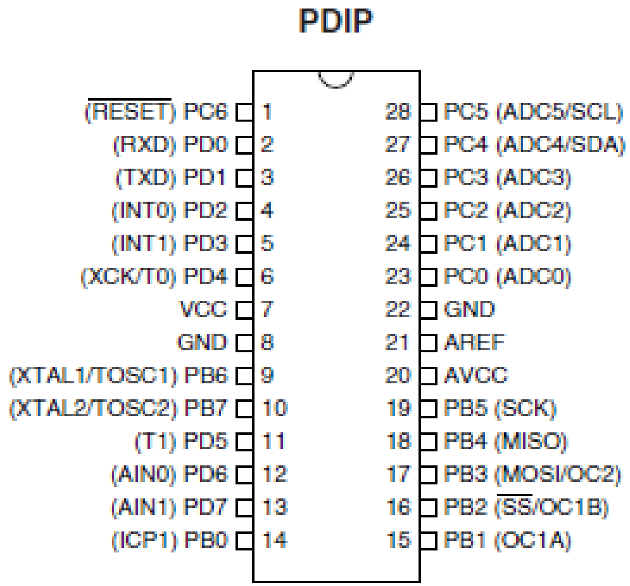
ATtiny26L

PDIP/SOIC

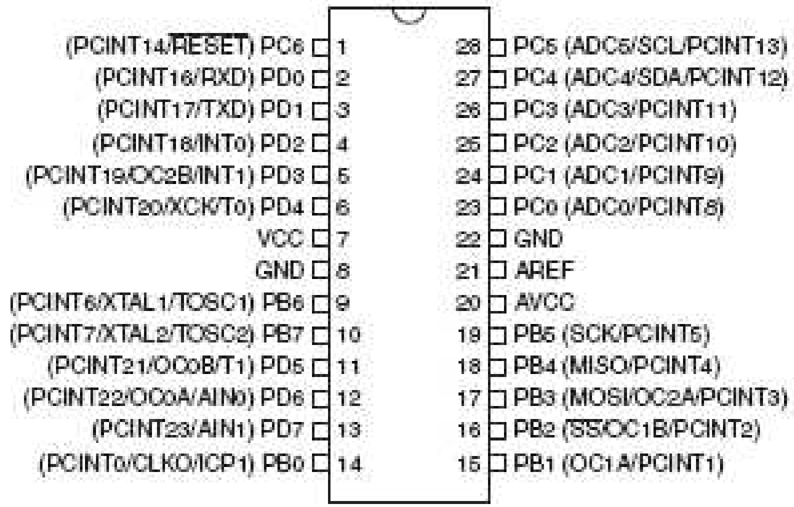


ATMEGA8

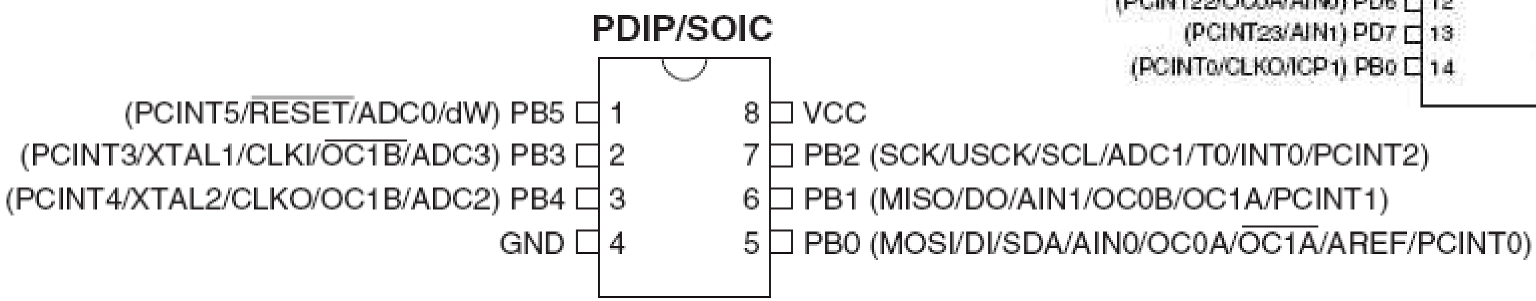




ATMEGA168



TINY45



HEX FORMAT

```
:10202E00216E016A0162215D217F218C017F01C336
:10203E000126011602294002 2AAD0172022A9A02D5
:10204E0029D00229D6022A1402287D02284302270B
 ^  ^    ^  ^                            ^
 |  |    |  |                            |
 |  |    |  |       checksum-------------+
 |  |    |  +-------data bytes
 |  |    +----------record type (00=data, 01=end of file)
 |  +---------------address for this line of data
 +------------------number of bytes of data in this line

Line #1: 16 bytes  @  0x202E to 0x203D  (8238 to 8253)
Line #2: 16 bytes  @  0x203E to 0x204D  (8254 to 8269)
Line #2: 16 bytes  @  0x204E to 0x205D  (8270 to 8285)
```

```
:10246200464C5549442050524F46494C4500464C33
|||||||||||                              CC->Checksum
|||||||||DD->Data
|||||||TT->Record Type
|||AAAA->Address
|LL->Record Length
:->Colon
```

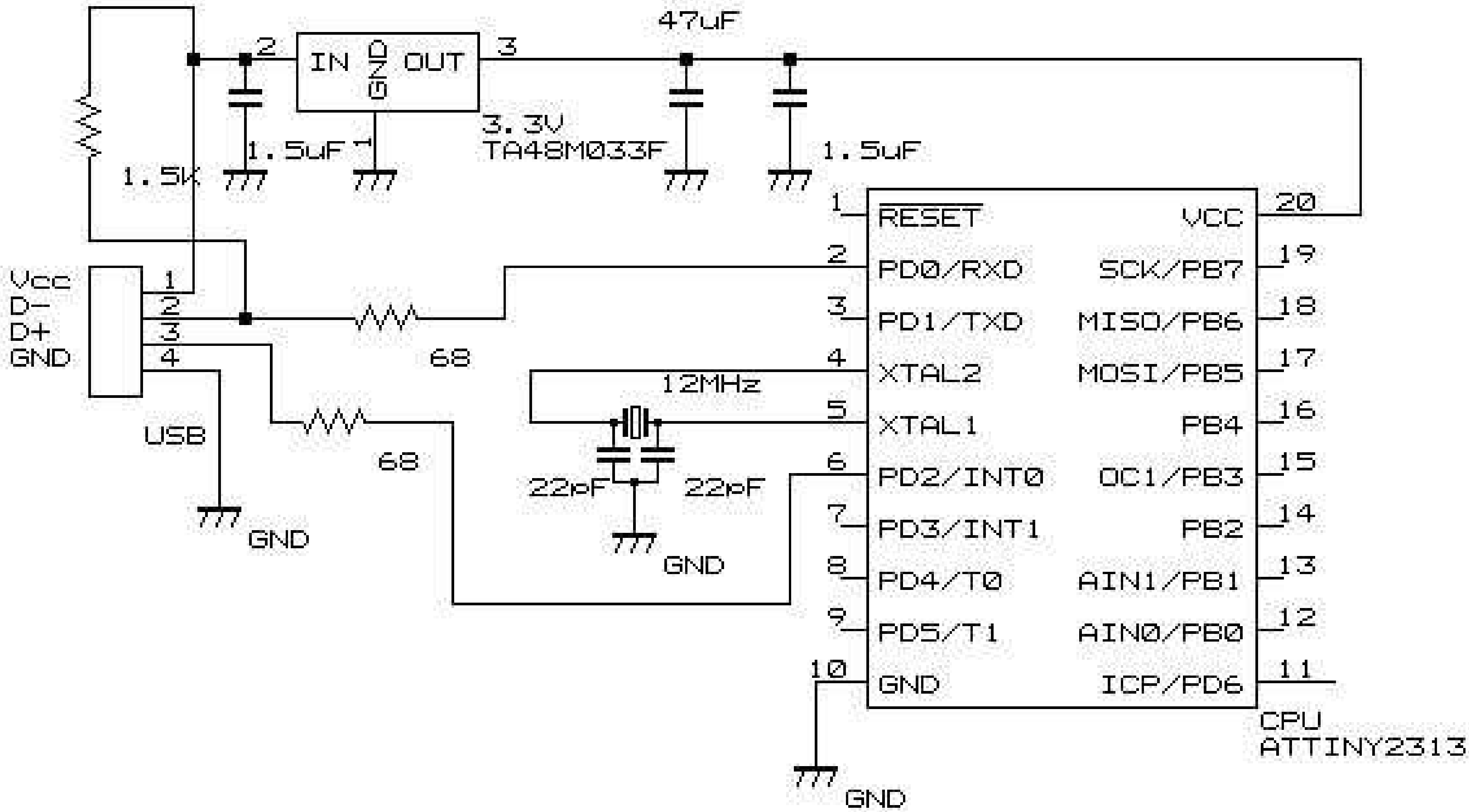
where:

- **10** is the number of data bytes in the record.
- **2462** is the address where the data are to be located in memory
- **00** is the record type 00 (a data record).
- **464C...464C** is the data.
- **33** is the checksum of the record.

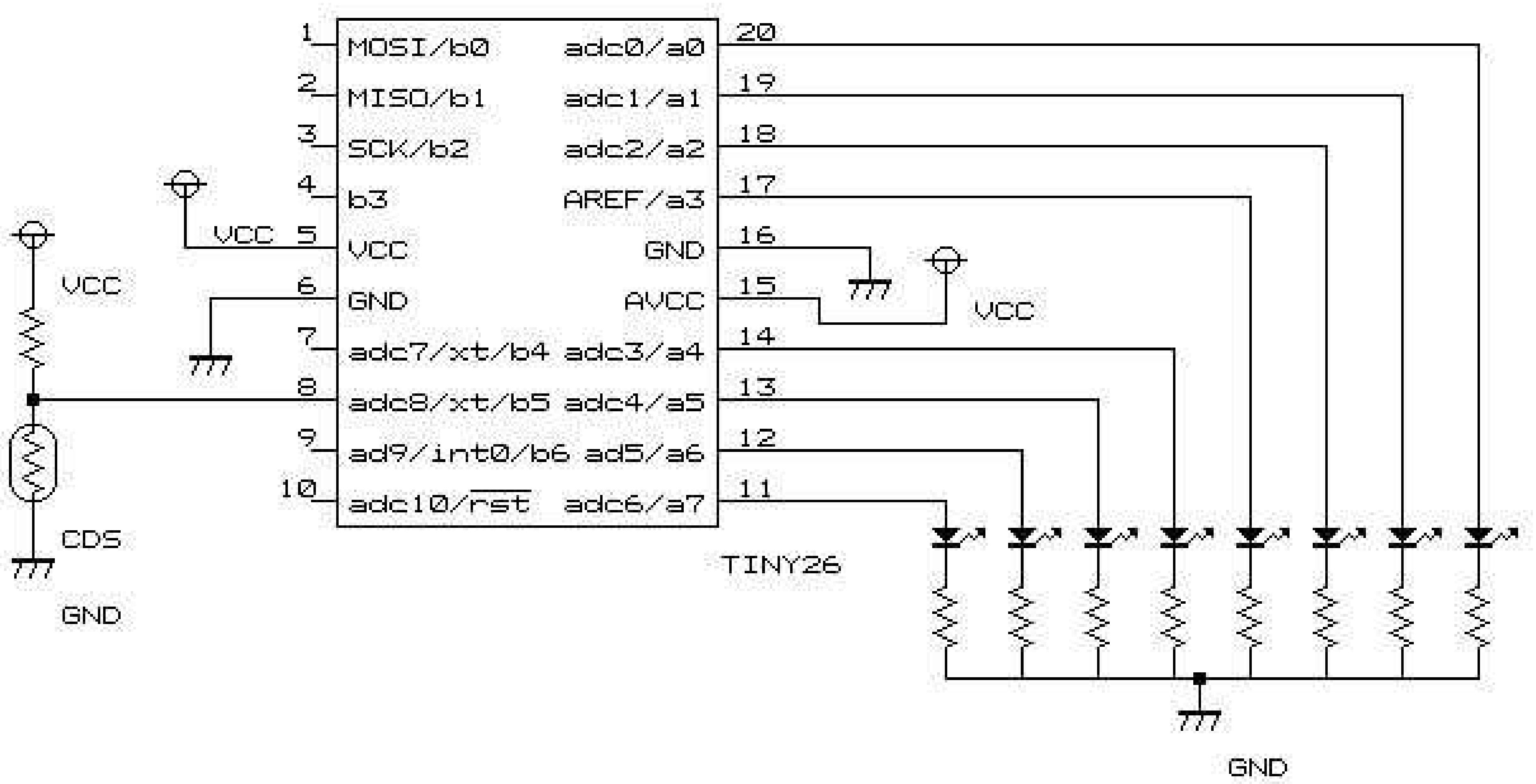
```
:00000001FF
```

vhere:

- **00** is the number of data bytes in the record.
- **0000** is the address where the data are to be located in memory. Th address in end-of-file records is meaningless and is ignored. An addre of 0000h is typical.
- **01** is the record type 01 (an end-of-file record).
- **FF** is the checksum of the record and is calculated as 01h + NOT(00h + 00h + 00h + 01h).



47uF
IN
GND
OUT
2
3
1
3.3V
TA48M033F
1.5uF
1.5uF
1.5K
Vcc
D-
D+
GND
USB
68
68
12MHz
22pF
22pF
GND
GND
GND
RESET
VCC
PD0/RXD
SCK/PB7
PD1/TXD
MISO/PB6
XTAL2
MOSI/PB5
XTAL1
PB4
PD2/INT0
OC1/PB3
PD3/INT1
PB2
PD4/T0
AIN1/PB1
PD5/T1
AIN0/PB0
GND
ICP/PD6
CPU
ATTINY2313


1 MOSI/b0
2 MISO/b1
3 SCK/b2
4 b3
VCC 5 VCC
6 GND
7 adc7/xt/b4
8 adc8/xt/b5
9 ad9/int0/b6
10 adc10/rst
20 adc0/a0
19 adc1/a1
18 adc2/a2
17 AREF/a3
16 GND
15 AVCC
14 adc3/a4
13 adc4/a5
12 ad5/a6
11 adc6/a7
TINY26
VCC
VCC
CDS
GND
GND

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x06	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

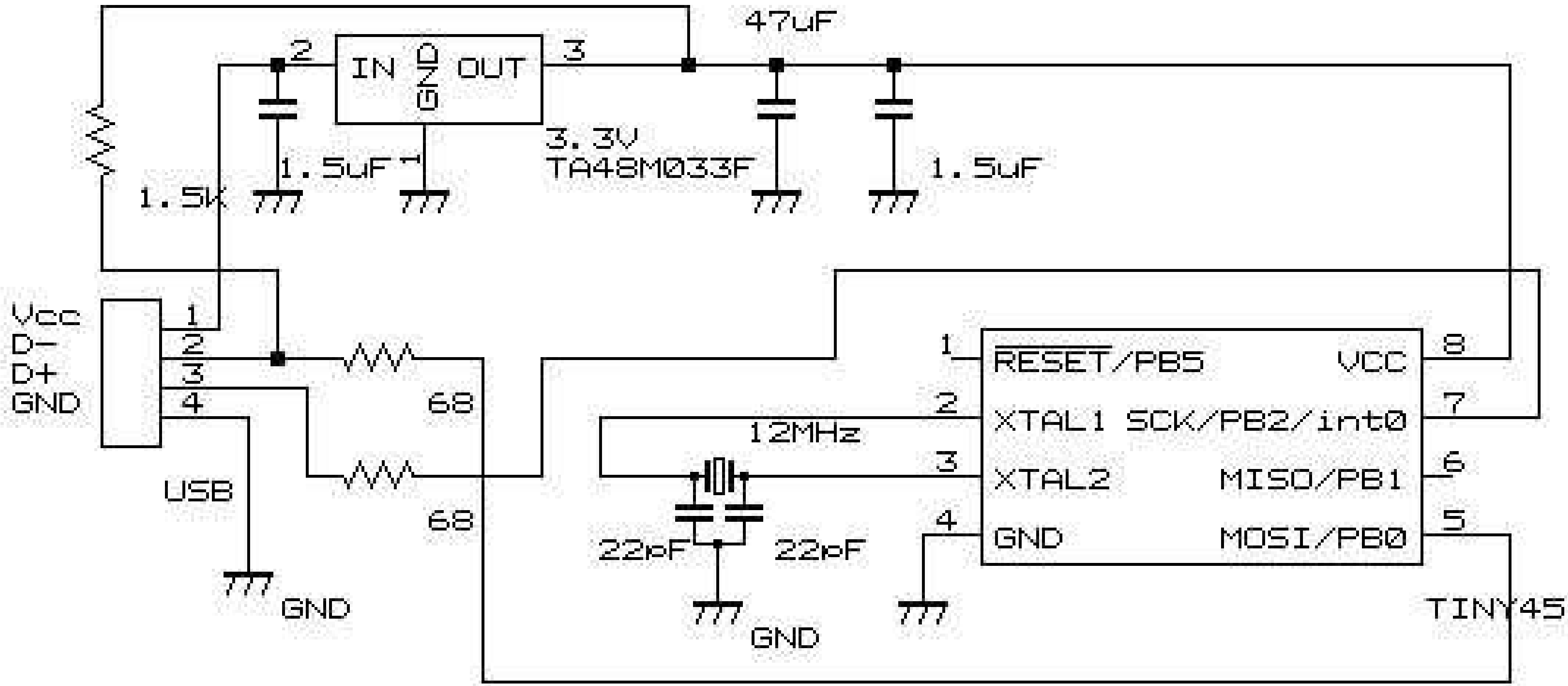
ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

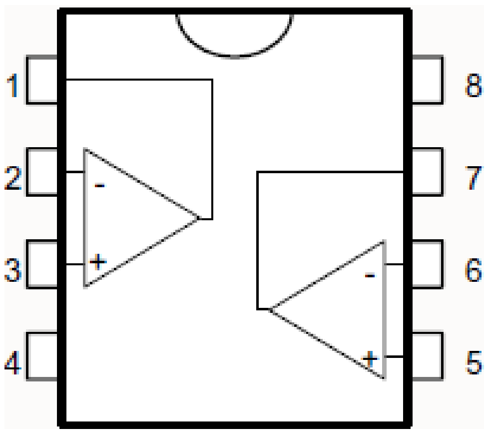
When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

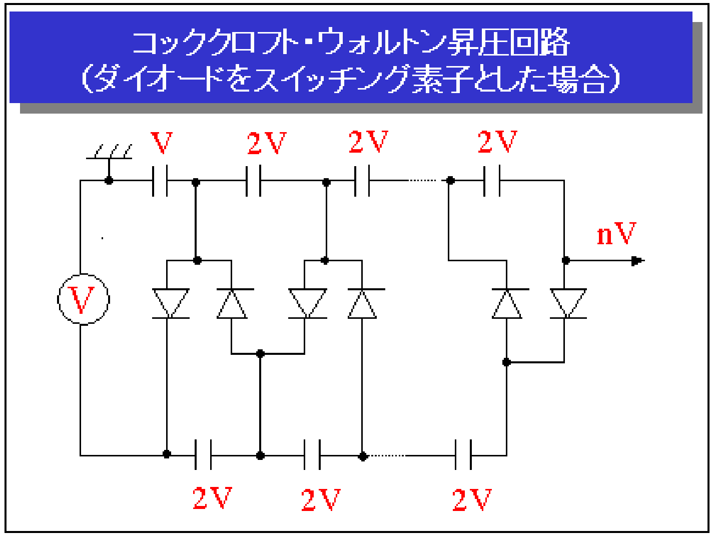
This bit is set when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is



TINY45 usb by takefuji


1
2
3
4
8
7
6
5
-
+
-
+

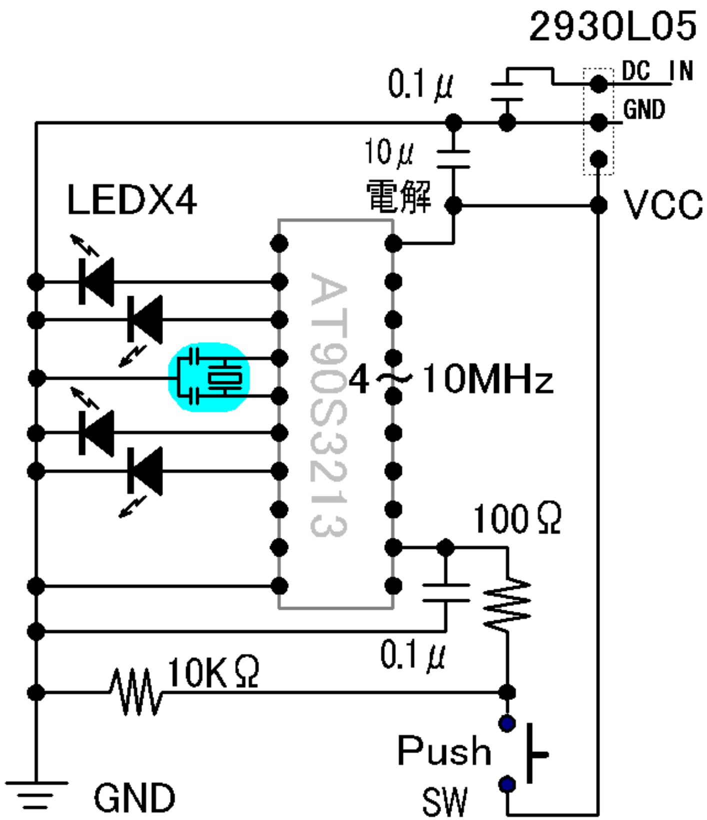
2904D


コッククロフト・ウォルトン昇圧回路
(ダイオードをスイッチング素子とした場合)
V
2V
2V
2V
nV
V
2V
2V
2V

昇圧回路

Avr programming

```
#include <avr/io.h>
typedef unsigned char  io8bit;

Int  main( void )
      {
      io8bit nowled ;

      DDRD=0xff;                         /* PD0-PD6 PortD for output */
      DDRB=0xfe;                         /* PB0 for input  PB1-7 for output */

      nowled = 0x00;                     /* LED init all Low         */

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

                        PORTD=nowled;                   /* LED out         */
                }
      }
```



USB firmware 1/2

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/wdt.h>
#include "usbdrv.h"

void delay(unsigned int p)
{  unsigned char i;
  unsigned char j;              //one loop is 0.0038225ms with 12MHz
  for(i=0;i<p;i++)
    for(j=0;j<10;j++);      }

uchar usbFunctionSetup(uchar data[1])
{
static uchar count;
static uchar replybuf[1];  /* how many bytes to be read */
    usbMsgPtr = replybuf;
    if(data[1] == 0){   /* PORTD */
                    count=0;
                    PORTB=0;                            // Vc=0
                    delay(5);
                    DDRB=0xfc;                          //PB0 and PB1 are set as inputs.
                    while ((ACSR&0x20) == 0) {count++;}
                    if (count < 47)             //Vin < 1
                    {PORTB = 0x00;                      //PB7=off
                    PORTD = 0xff;}                      //PD3=on
                    else if ( 46 < count && count < 107)     // 1<Vin <2
                    {PORTB = 0xff;                      //PB7=off
                    PORTD = 0x00;}
                    else                                              //Vin >2
                    {PORTB = 0xff;                      //PB7=off
                    PORTD = 0xff;}
                     replybuf[0] = count;
                    DDRB=0xfd;
        return 1;         }
return 0;      }
```

USB firmware 2/2

```
uchar usbFunctionRead(uchar *data, uchar len)
{    return 0;  }

uchar usbFunctionWrite(uchar *data, uchar len)
{   return 0;   }


int main(void)
{
  PORTD = 0;
  PORTB = 0;                    /* no pullups on USB and ISP pins */
  DDRD = 0xFA;                  /* all outputs except PD2 = INT0 and PD0*/
  DDRB = 0xFD;      /* all output except PB1*/
  ACSR= 0x00;                   //analog comparator enabled


  usbInit();
  sei();
  for(;;){          /* main event loop */
             usbPoll();
               }
  return 0;
}
```

USB usbconfig.h

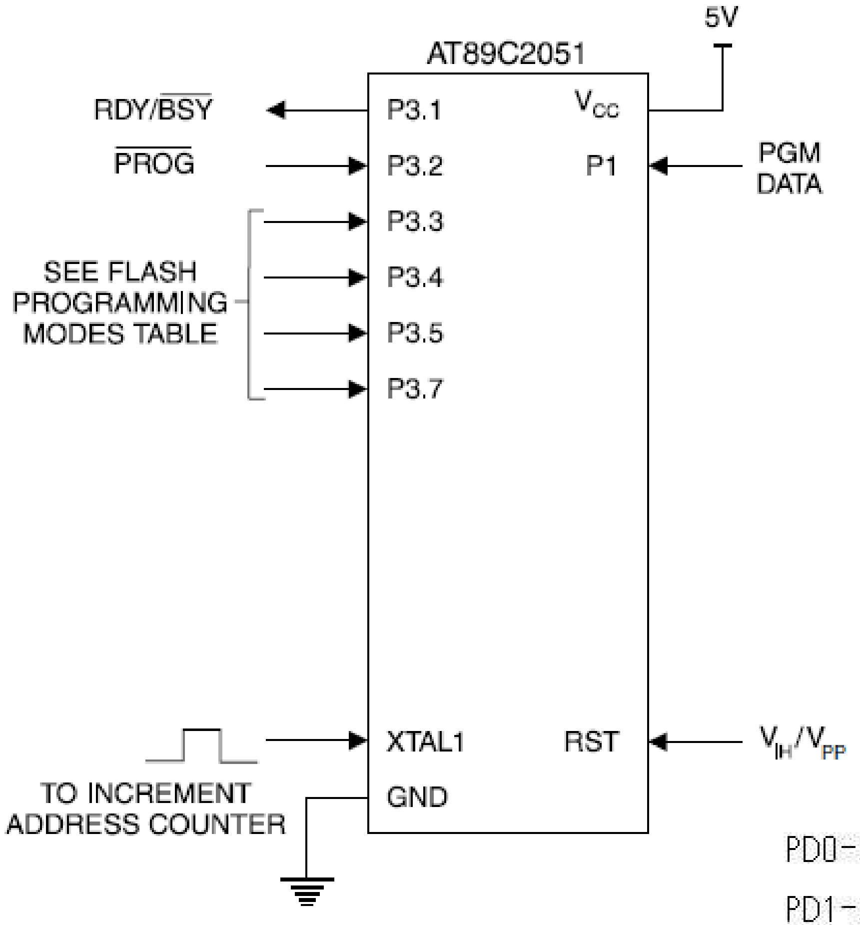
```
#define    USB_CFG_IOPORT                     PORTD
#define    USB_CFG_DMINUS_BIT                 0
#define    USB_CFG_DPLUS_BIT                  2
/* --------------------------- Functional Range ---------------------------- */
#define    USB_CFG_HAVE_INTRIN_ENDPOINT 1
#define    USB_CFG_INTR_POLL_INTERVAL         4
#define    USB_CFG_IS_SELF_POWERED            0
#define    USB_CFG_MAX_BUS_POWER              100
#define    USB_CFG_SAMPLE_EXACT               1
#define USB_CFG_IMPLEMENT_FN_WRITE            1
#define USB_CFG_IMPLEMENT_FN_READ             0
/* -------------------------- Device Description --------------------------- */
#define    USB_CFG_VENDOR_ID                  0xFE, 0x0B
#define    USB_CFG_DEVICE_ID                  0x03, 0x10
#define    USB_CFG_DEVICE_VERSION             0x00, 0x01
#define    USB_CFG_VENDOR_NAME                'M', 'o', 'r', 'p', 'h', 'y', ' ', 'P', 'l', 'a', 'n', 'n', 'i', 'g'
#define    USB_CFG_VENDOR_NAME_LEN            14
#define    USB_CFG_DEVICE_NAME                'U', 'S', 'B', '-', 'I', 'O'
#define    USB_CFG_DEVICE_NAME_LEN            6
#define    USB_CFG_DEVICE_CLASS               0
#define    USB_CFG_DEVICE_SUBCLASS            0
#define    USB_CFG_INTERFACE_CLASS            3
#define    USB_CFG_INTERFACE_SUBCLASS         0
#define    USB_CFG_INTERFACE_PROTOCOL         0
```

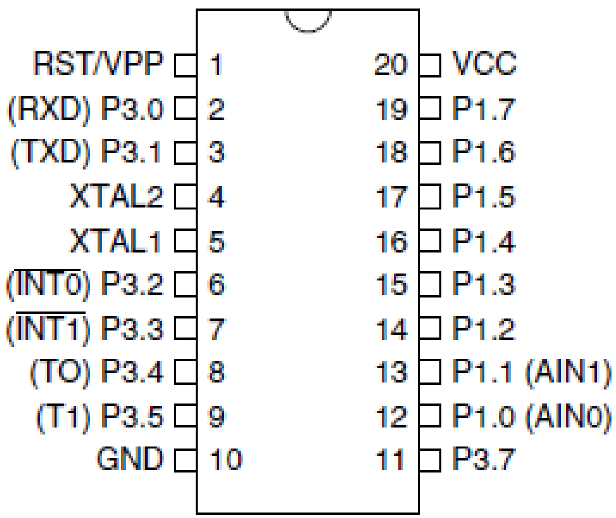
USB application 1/2

```
#include <usb.h>
#include <stdio.h>
#include <string.h>
unsigned short IDVendor=   0x1384;
unsigned short IDProduct=  0x8888;

static int usbOpenDevice(usb_dev_handle **device, int idvendor, int idproduct)
{
        struct usb_bus *bus;
        struct usb_device *dev;
        usb_dev_handle *udh=NULL;
        int retp, retm,errors;
        char string[256];

        usb_init();
        usb_find_busses();
        usb_find_devices();
        for (bus = usb_busses; bus; bus = bus->next)
        {
                for (dev = bus->devices; dev; dev = dev->next)
                {
                        udh=usb_open(dev);
                        retp = usb_get_string_simple(udh, dev->descriptor.iProduct, string, sizeof(string));
                        retm=usb_get_string_simple(udh, dev->descriptor.iManufacturer, string, sizeof(string));
                        if (retp > 0 && retm > 0)
                                if (idvendor==dev->descriptor.idVendor && idproduct==dev-
>descriptor.idProduct){ *device=udh;return errors=0;}
                                else { usb_close(udh);return errors=1;}
                }
        }
}
```

USB application 2/2

```
int main (int argc, char **argv)
{
        usb_dev_handle *d=NULL;
        unsigned char buffer[3];
        int i, mode, ret;
        char string[256];

        if(argc <2)
        {
                printf("tragi data¥n");
                exit(1);
        }
        i=atoi(argv[1]);
        mode=0;
        usb_init();
        ret=usbOpenDevice(&d, IDVendor,IDProduct);
        if(ret!=0){printf("usbOpenDevice failed¥n"); return 0;}

        ret=usb_control_msg(d, USB_TYPE_VENDOR | USB_RECIP_DEVICE |
USB_ENDPOINT_IN, mode, i, 0, (char *)buffer, sizeof(buffer), 5000);
        printf("buffer %d ¥n", buffer[0]);
        return 0;
}
```





PD0->USBD-

PD1->RSTVPP

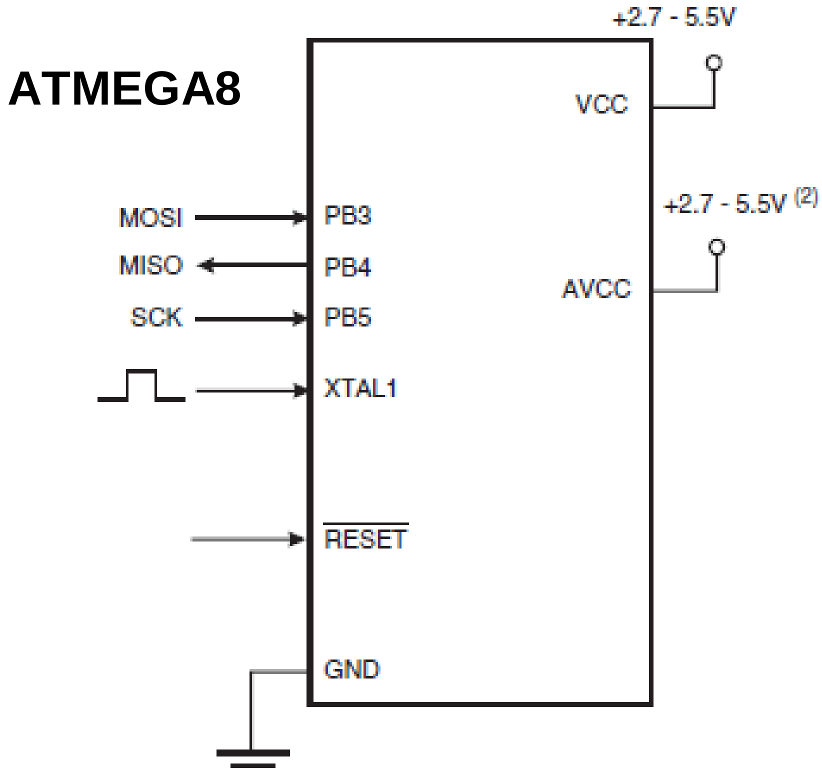
PD2->USBD+

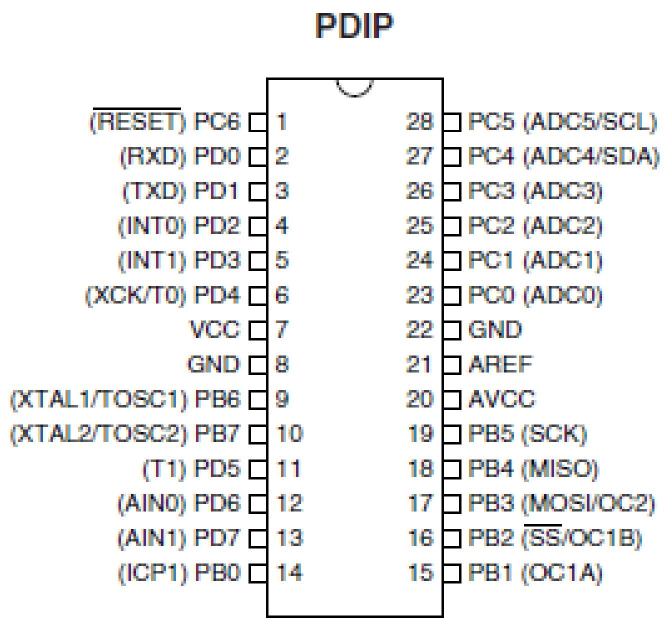
PD3->PROG

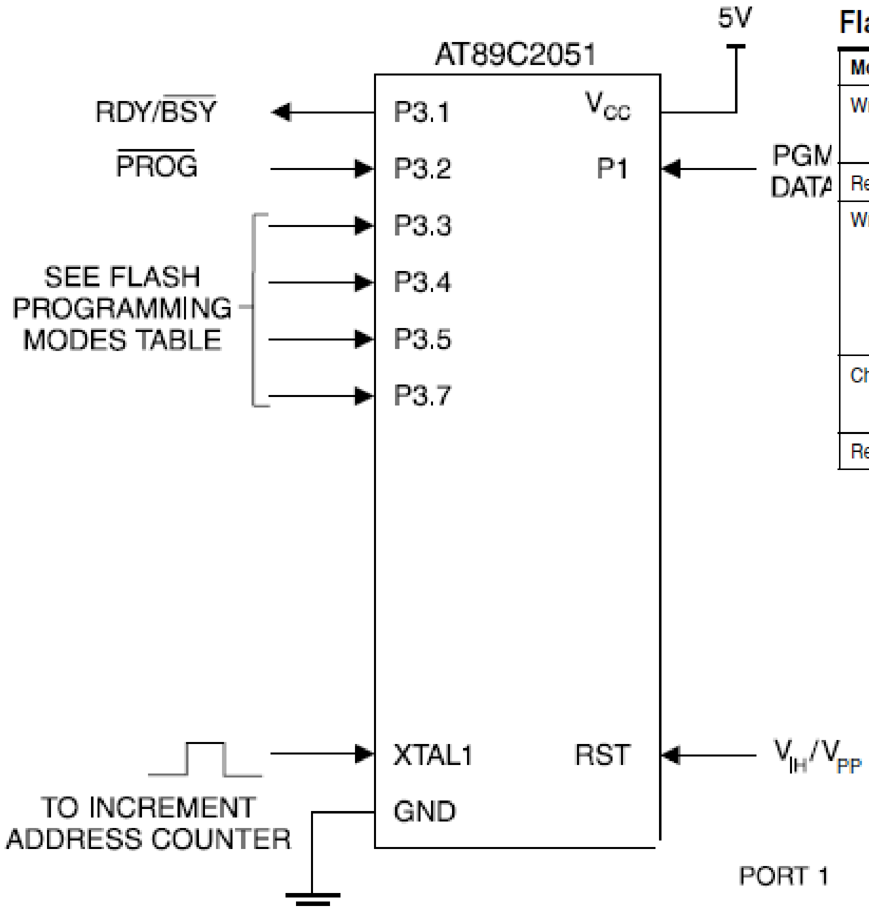
PD4->P33

PD5->P34

PD6->XTAL1

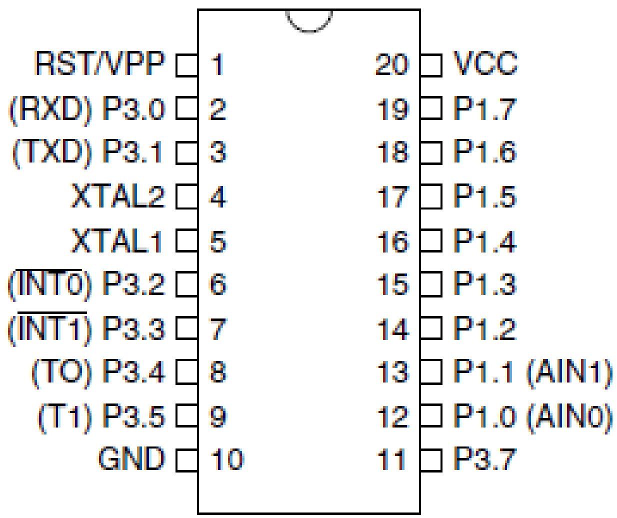


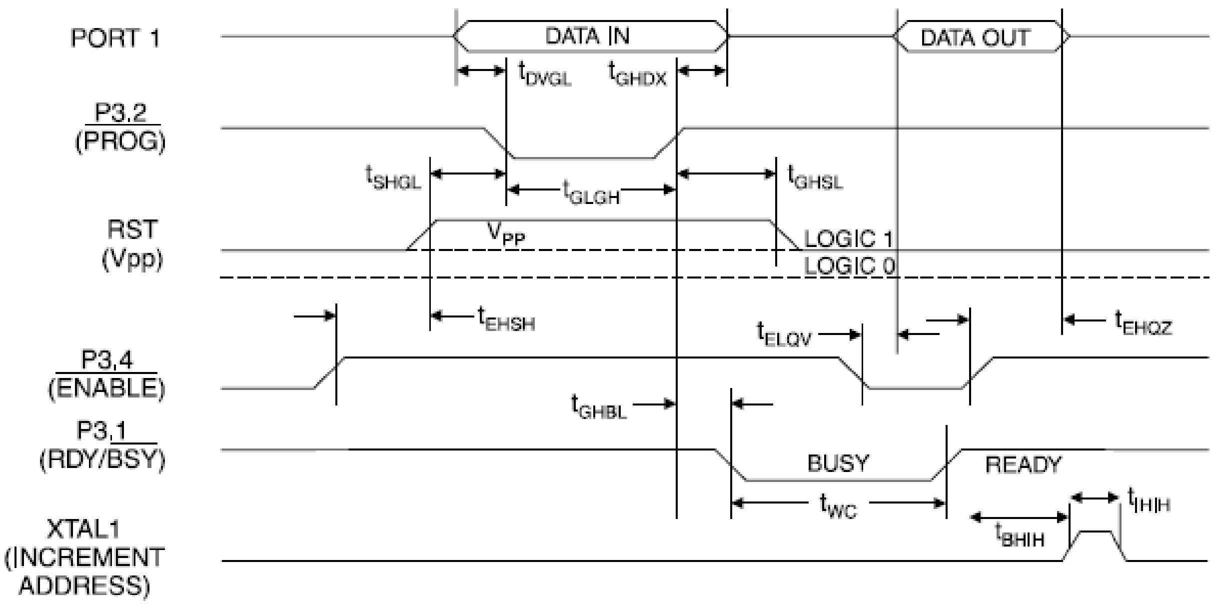


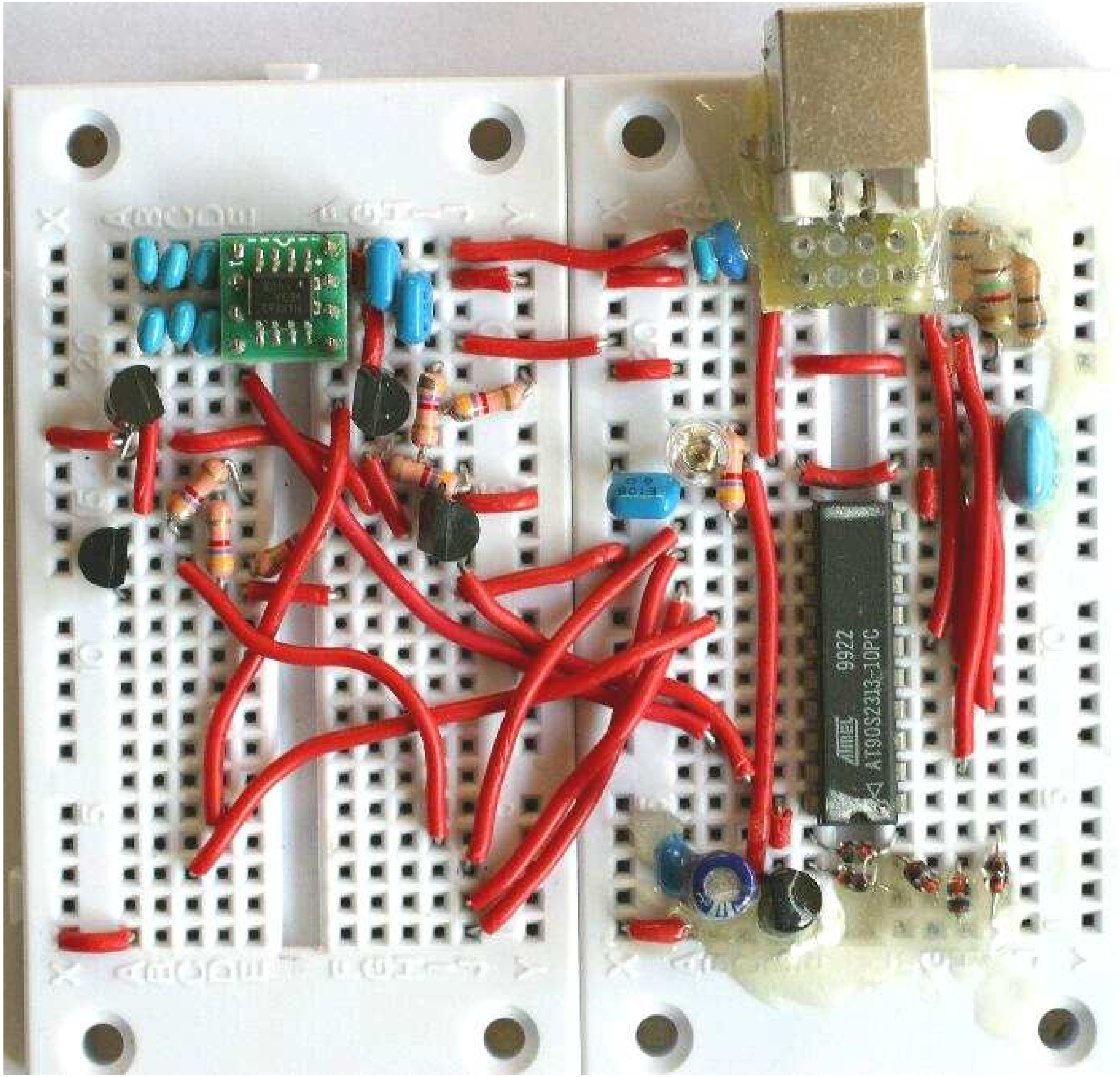


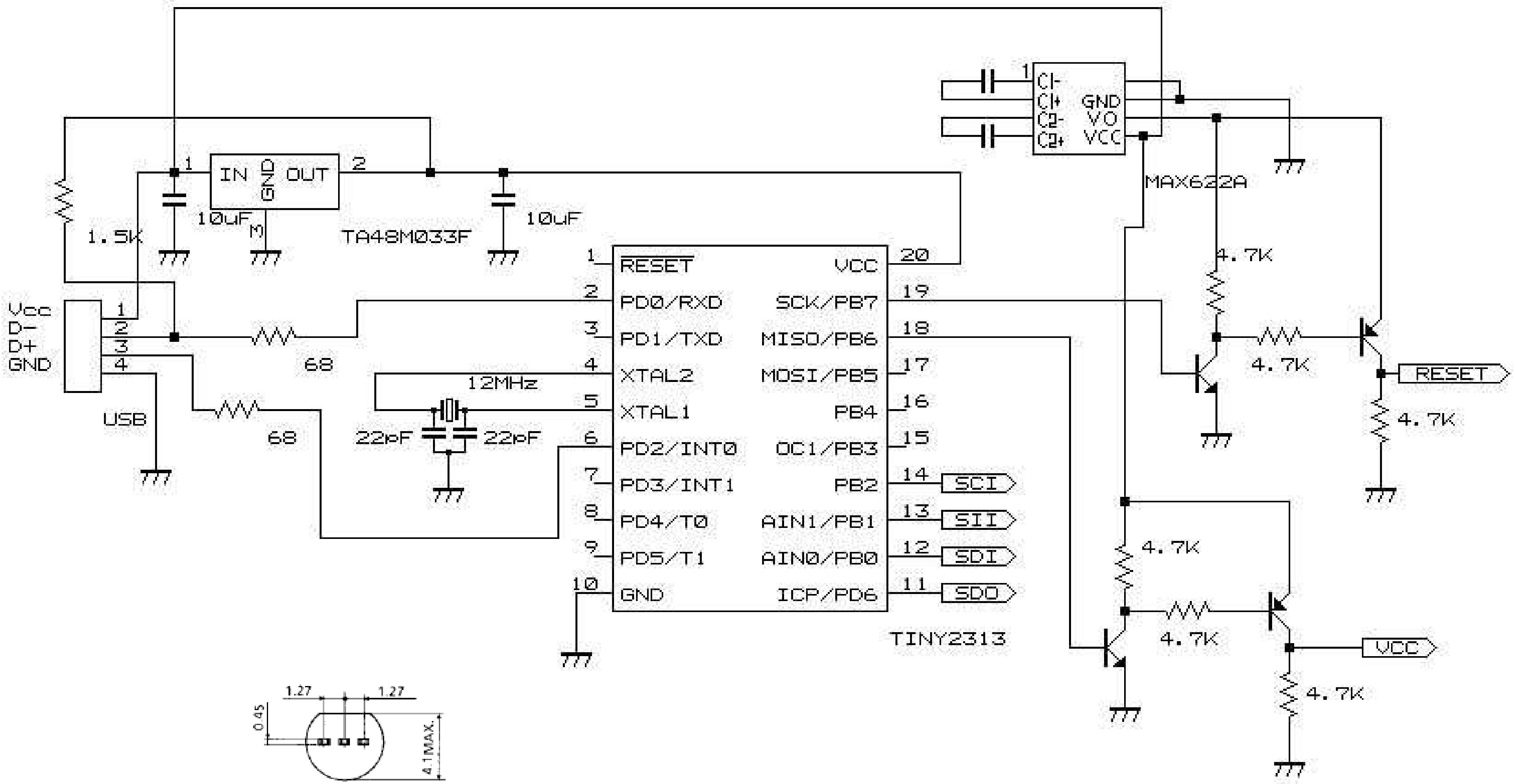
Flash Programming Modes

Mode		RST/VPP	P3.2/PROG	P3.3	P3.4	P3.5	P3.7
Write Code Data[1][3]		12V		L	H	H	H
Read Code Data[1]		H	H	L	L	H	H
Write Lock	Bit - 1	12V		H	H	H	H
	Bit - 2	12V		H	H	L	L
Chip Erase		12V	(2)	H	L	L	L
Read Signature Byte		H	H	L	L	L	L





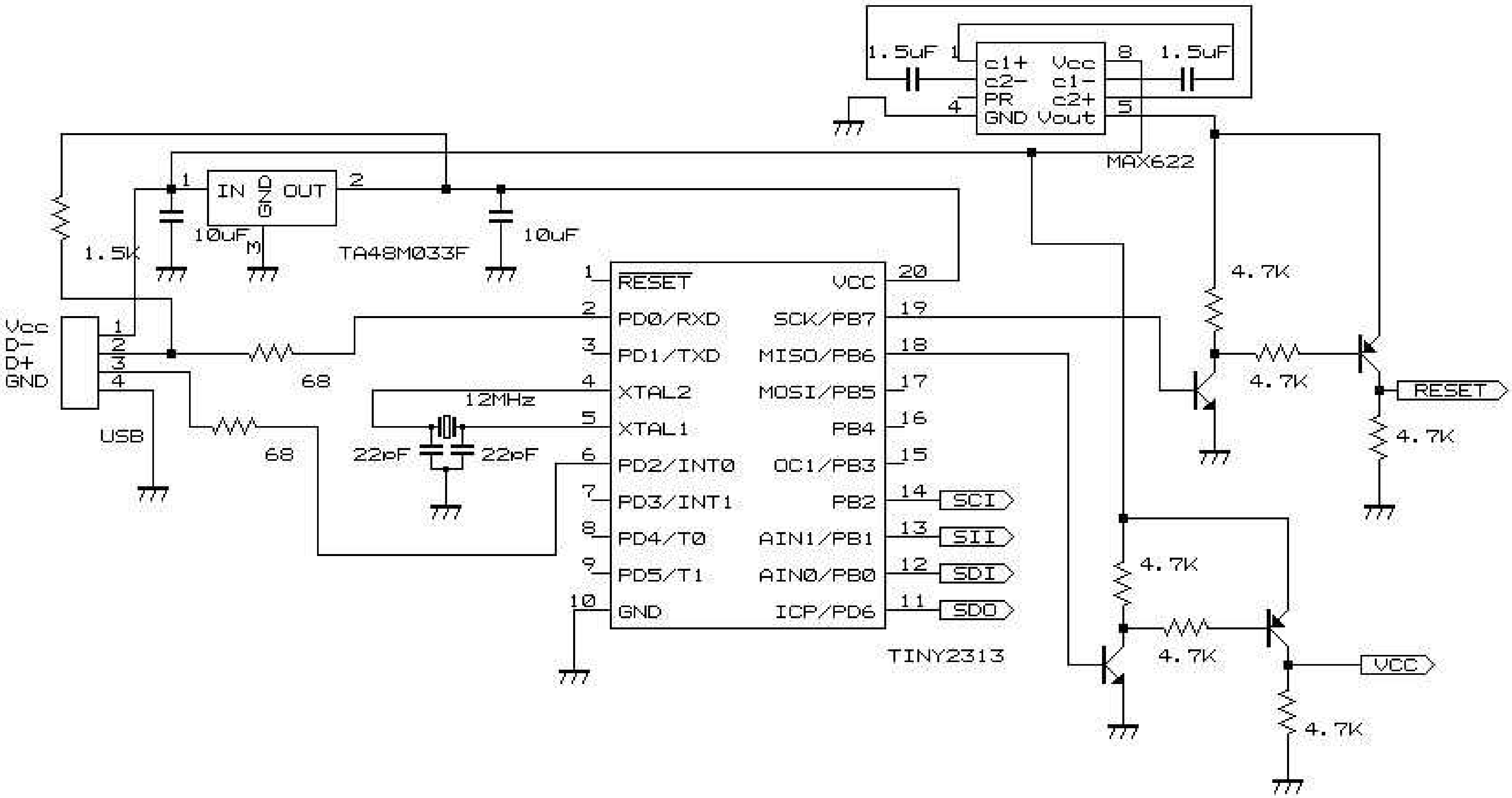

9922
AT90S2313-10PC



USB high voltage writer for tiny25/45/85 by takefuji

1. エミッタ
2. コレクタ
3. ベース

2SA1015
2SC1815

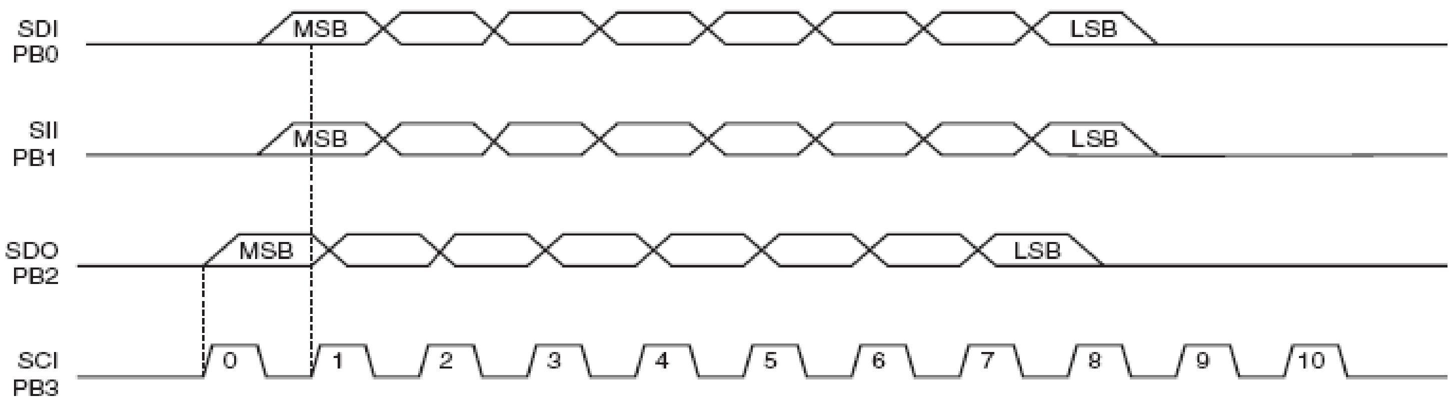


USB high voltage writer for tiny25/45/85 by takefuji

Table 22-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External Reset disable	1 (unprogrammed)
DWEN(2)	6	DebugWIRE Enable	1 (unprogrammed)
SPIEN(3)	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON(4)	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(5)	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1(5)	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0(5)	0	Brown-out Detector trigger level	1 (unprogrammed)

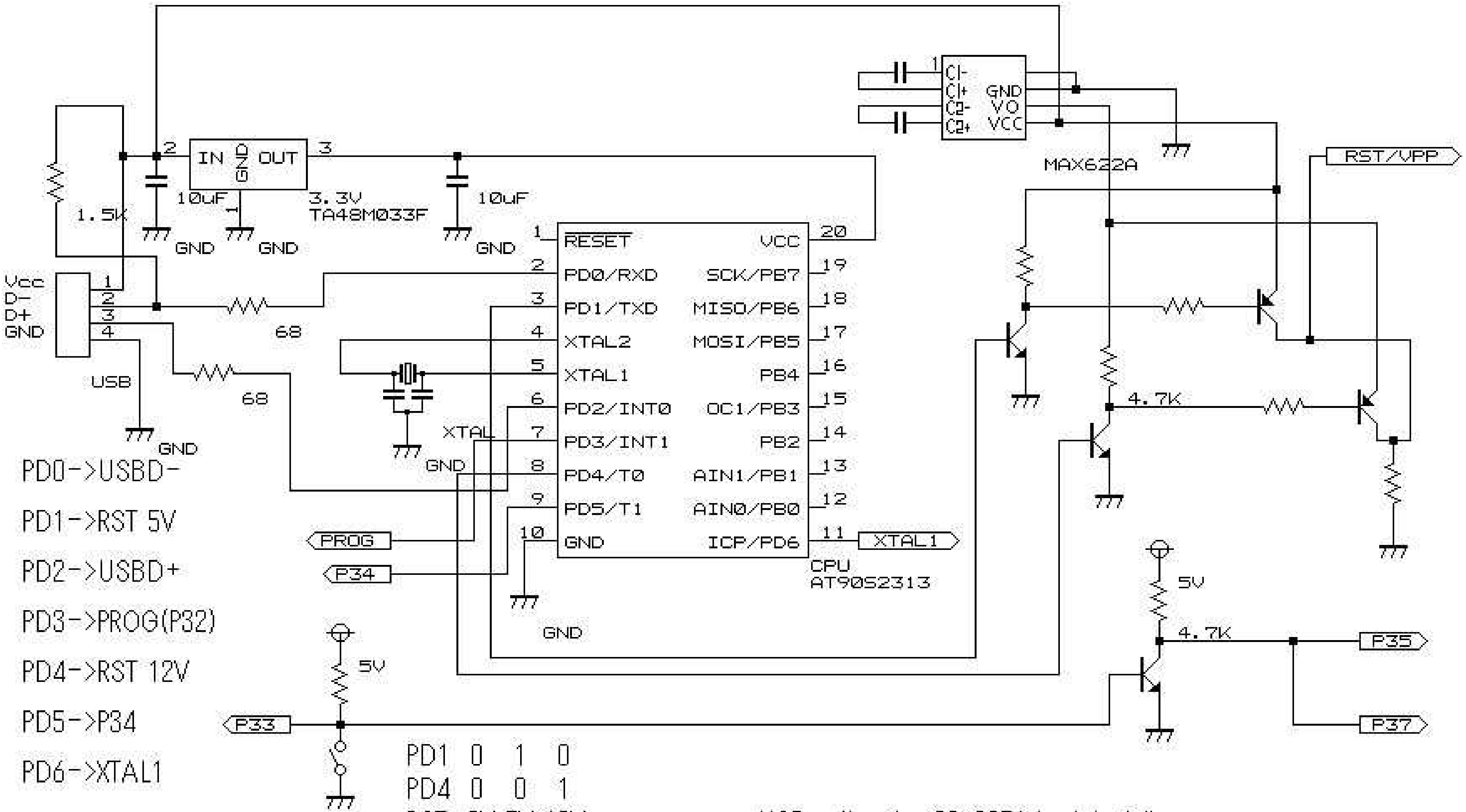
Figure 22-5. High-voltage Serial Programming Waveforms



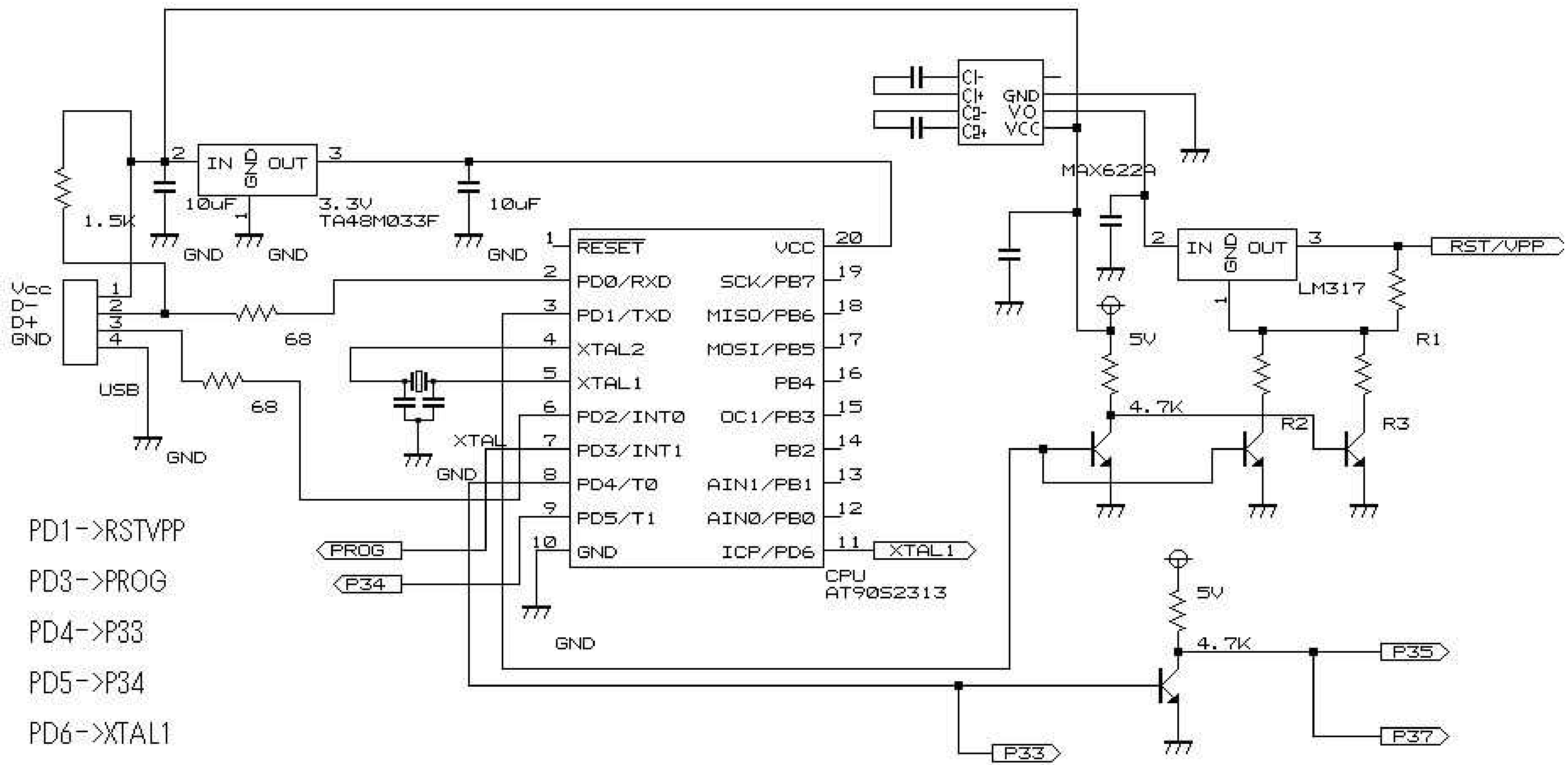
B = RSTDISBL Fuse, **C** = BODLEVEL0 Fuse, **D**= BODLEVEL1 Fuse, **E** = MONEN Fuse, **F** = SPMEN Fuse

0001 1 1 1 1
000F EDCB

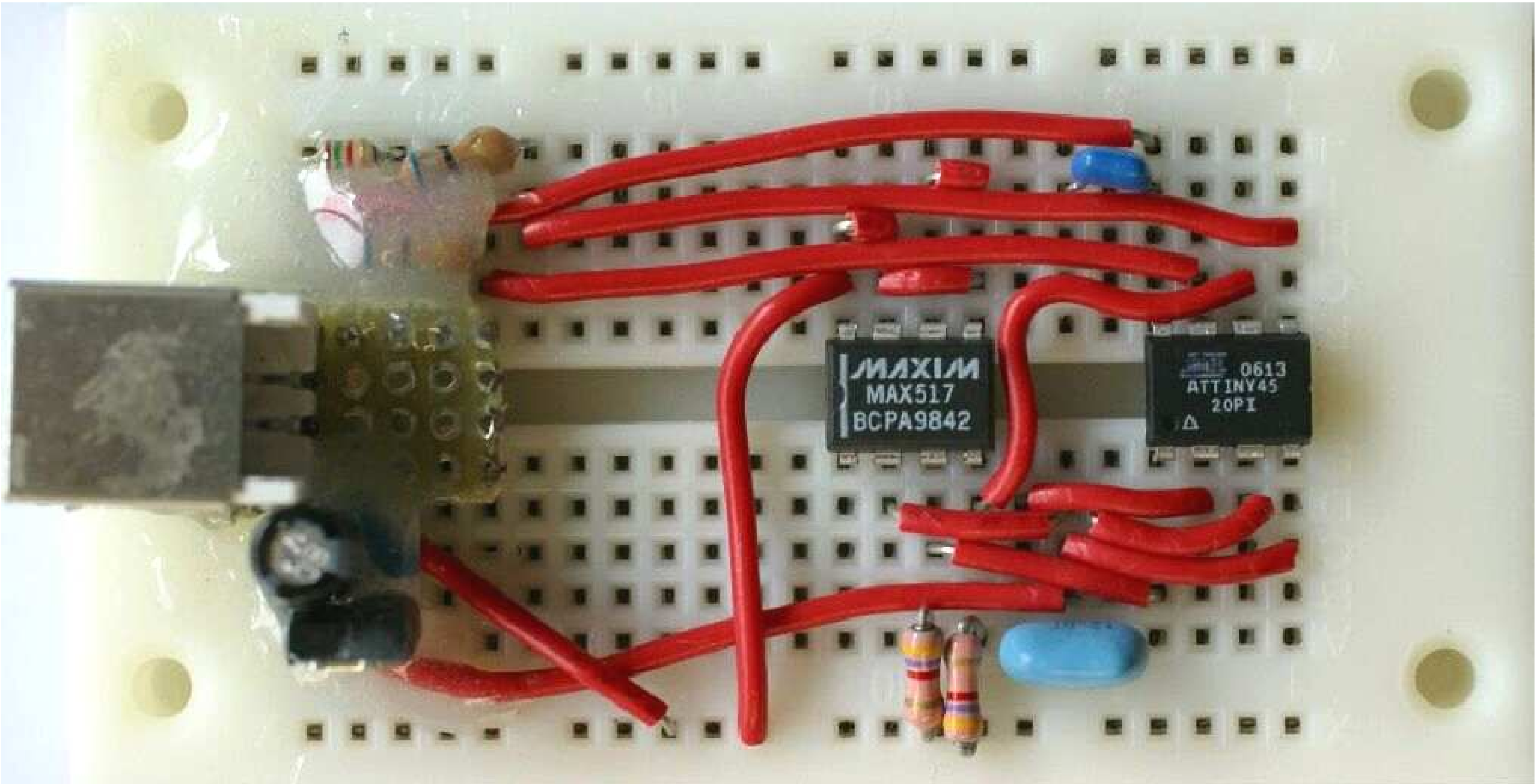
Instruction		Instruction Format: Instr.1/5	Instr.2/6	Instr.3	Instr.4	Operation Remarks
Write Fuse High Bits	SDI	0_0100_0000_00	0_000**F_EDCB**_00	0_0000_0000_00	0_0000_0000_00	Wait after Instr. 4 until SDO goes high. Write **F** - **B** = "0" to program the Fuse bit.
	SII	0_0100_1100_00	0_0010_1100_00	0_0111_0100_00	0_0111_1100_00	
	SDO	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	



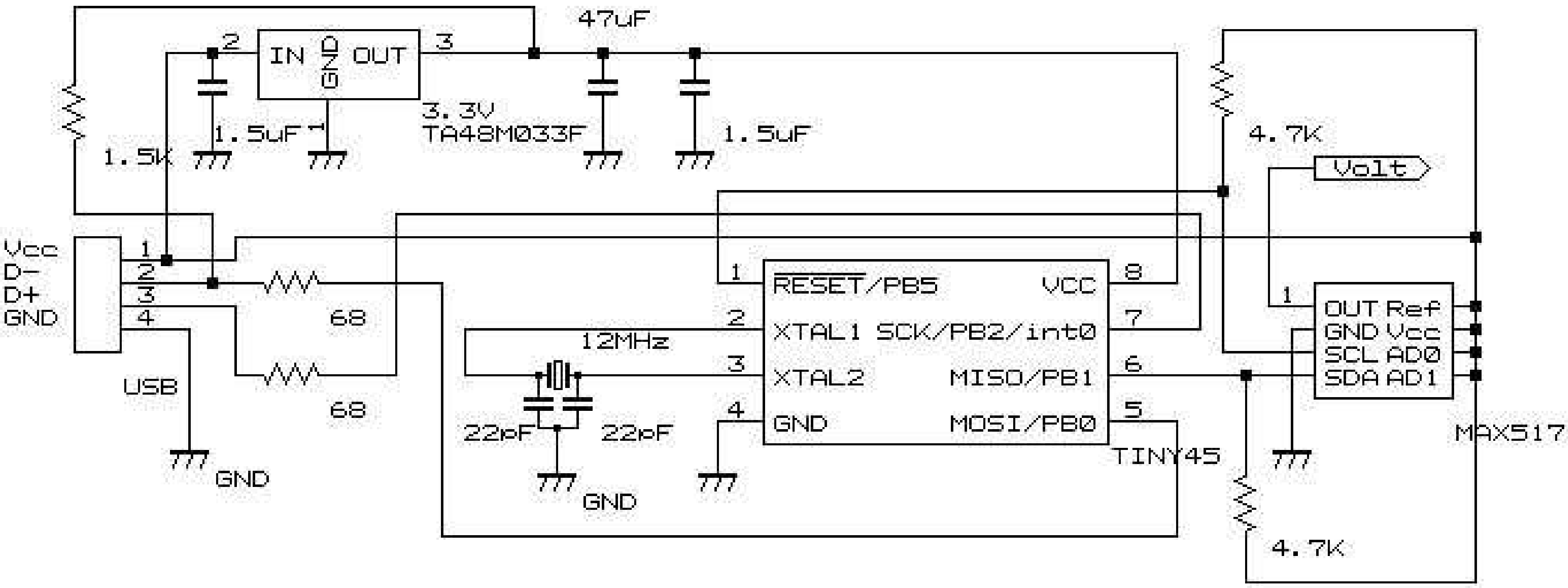
USB writer for 89c2051 by takefuji



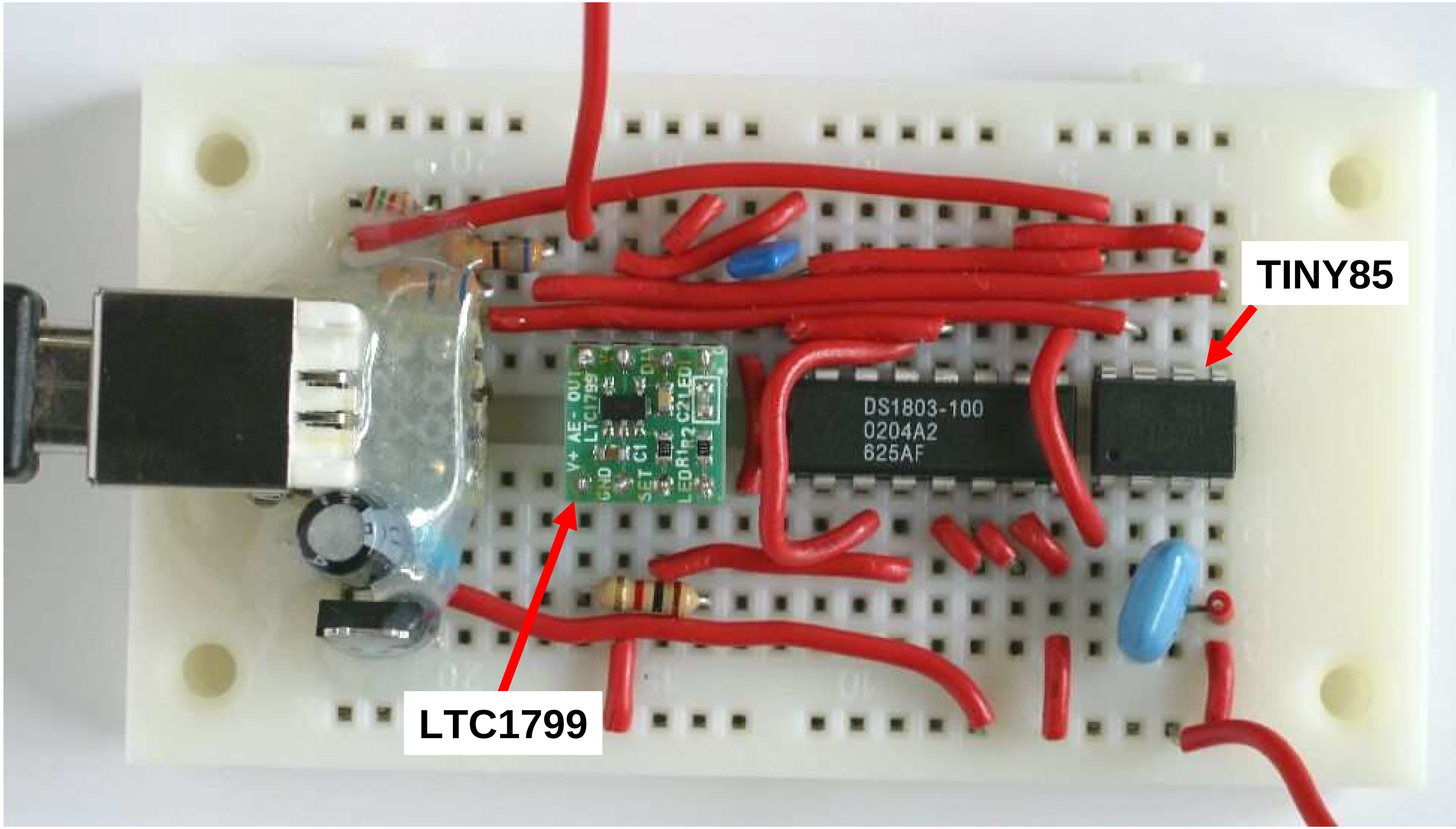
USB writer for 89c2051 by takefuji



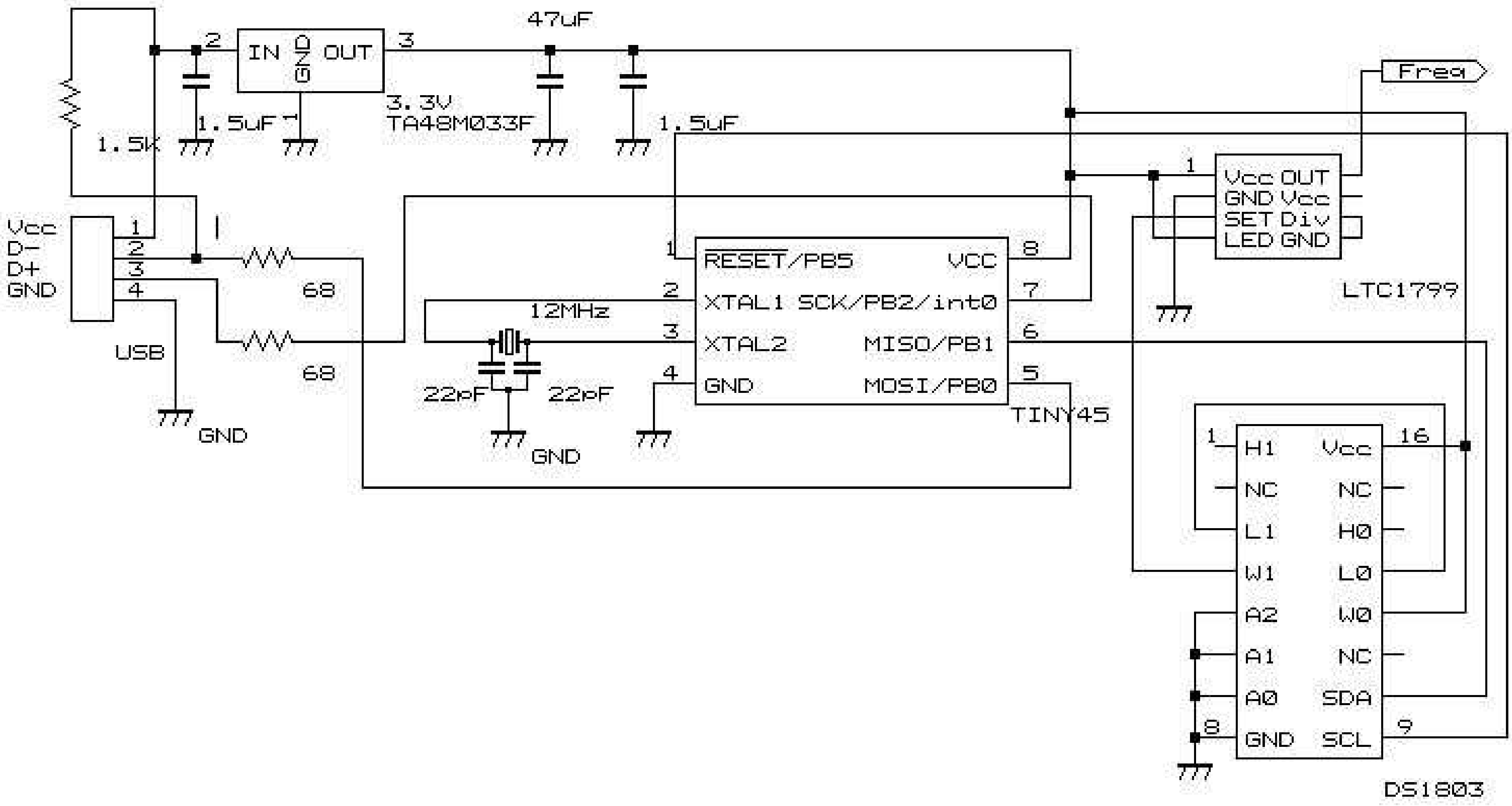
USB-DATiny45



USB-DAtiny45



USB programmable oscillator



USB programmable oscillator
From 500KHz to 20MHz

Table 7-3. Crystal Oscillator Operating Modes

CKSEL3:1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100[1]	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 -	12 - 22

Notes: 1. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 7-4.

Table 7-4. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1:0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	14CK + 4.ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	14CK + 64 ms	Ceramic resonator, slowly rising power
0	10	1K (1024) CK[2]	14CK	Ceramic resonator, BOD enabled
0	11	1K (1024)CK[2]	14CK + 4 ms	Ceramic resonator, fast rising power
1	00	1K (1024)CK[2]	14CK + 64 ms	Ceramic resonator, slowly rising power
1	01	16K (16384) CK	14CK	Crystal Oscillator, BOD enabled
1	10	16K (16384) CK	14CK + 4.ms	Crystal Oscillator, fast rising power
1	11	16K (16384) CK	14CK + 64 ms	Crystal Oscillator, slowly rising power

Table 7-1. Device Clocking Options Select[1]

Device Clocking Option	CKSEL3:0
External Low-Frequency Crystal	0110
External Crystal/Ceramic Resonator	1000-1111
Reserved	0101, 0111

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

2. This setting will select ATtiny15 Compatibility Mode, where the system clock is divided by fo resulting in a 1.6 MHz clock frequency.

Table 22-4. Fuse High Byte

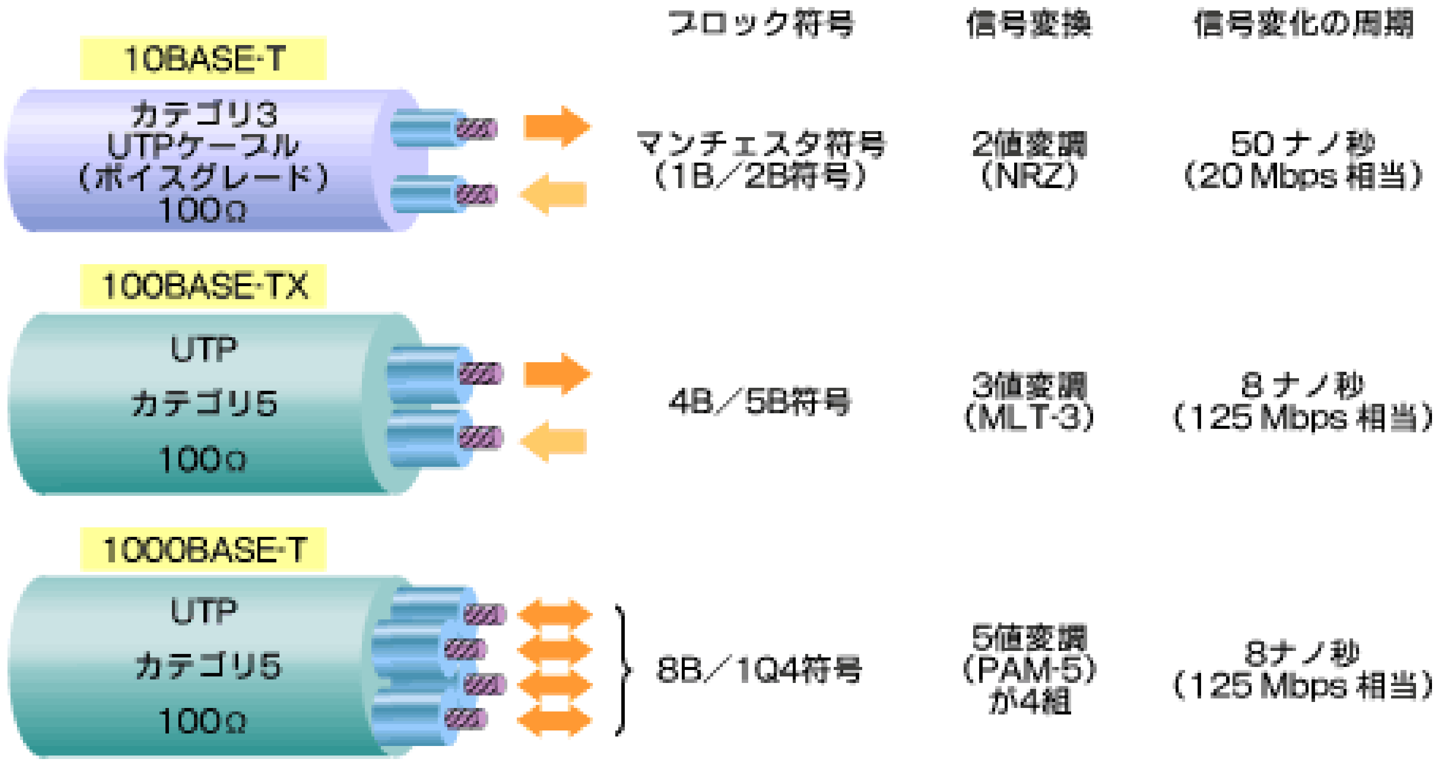
Fuse High Byte	Bit No	Description	Default Value
RSTDISBL[1]	7	External Reset disable	1 (unprogrammed)
DWEN[2]	6	DebugWIRE Enable	1 (unprogrammed)
SPIEN[3]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON[4]	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2[5]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1[5]	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[5]	0	Brown-out Detector trigger level	1 (unprogrammed)

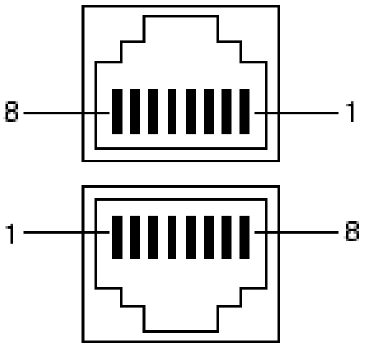
Table 22-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8[1]	7	Divide clock by 8	0 (programmed)
CKOUT[2]	6	Clock Output Enable	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[3]
SUT0	4	Select start-up time	0 (programmed)[3]
CKSEL3	3	Select Clock source	0 (programmed)[4]
CKSEL2	2	Select Clock source	0 (programmed)[4]
CKSEL1	1	Select Clock source	1 (unprogrammed)[4]
CKSEL0	0	Select Clock source	0 (programmed)[4]

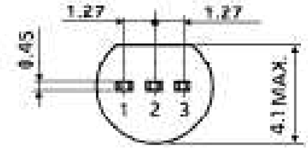
Table 7-1. Device Clocking Options Select[1]

Device Clocking Option	CKSEL3:0
External Clock	0000
PLL Clock	0001
Calibrated Internal RC Oscillator 8.0 MHz	0010
Calibrated Internal RC Oscillator 6.4 MHz[2]	0011
Watchdog Oscillator 128 kHz	0100

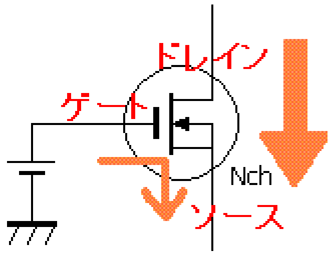


Pin Assignment	10BaseT Pin Number	MDI-X ports	100BaseTX and 1000BaseT Pin Number	MDI-X ports
	1	RD+	1	RD+
	2	RD-	2	RD-
	3	TD+	3	TD+
	4	Not used	4	CMT
	5	Not used	5	CMT
	6	TD-	6	TD-
	7	Not used	7	CMT
	8	Not used	8	CMT

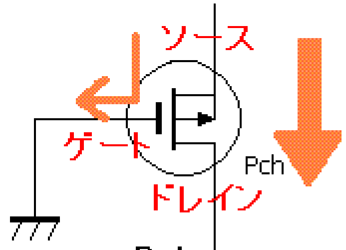
2SK30A



1. SOURCE
2. GATE
3. DRAIN

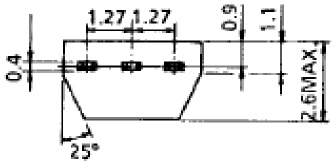


Nch
MOSFET



Pch
MOSFET

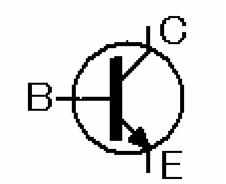
2SK241



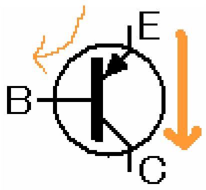
1. DRAIN
2. SOURCE
3. GATE

2SK

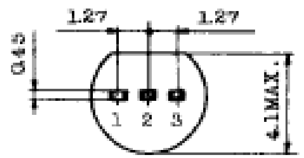
2SJ



NPN



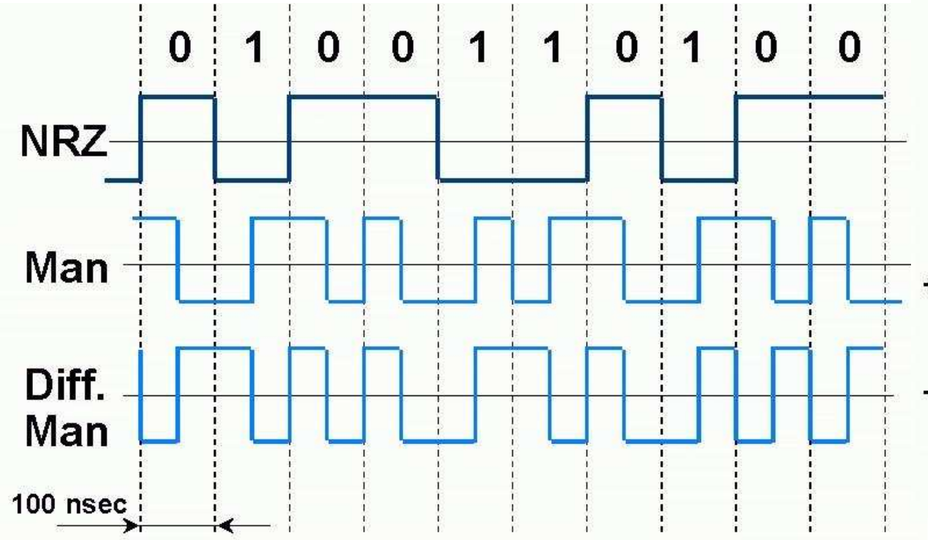
PNP

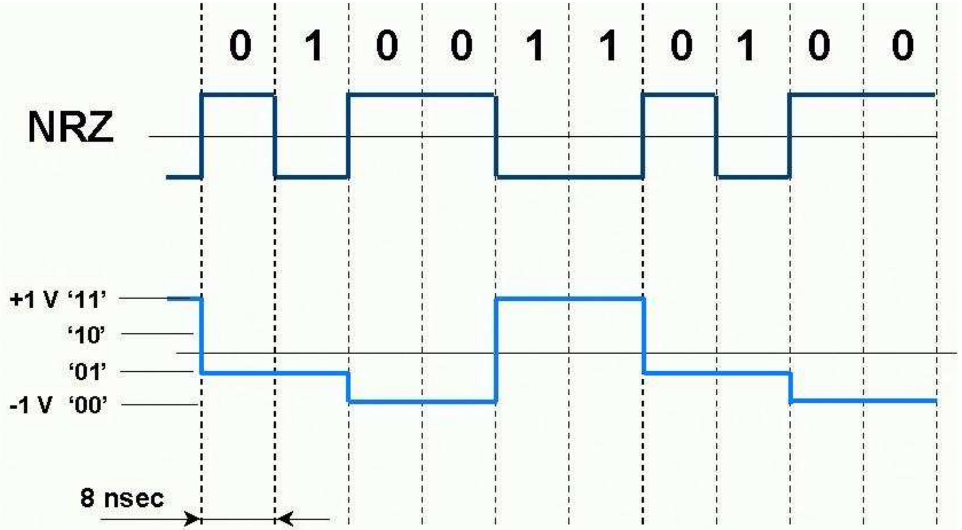


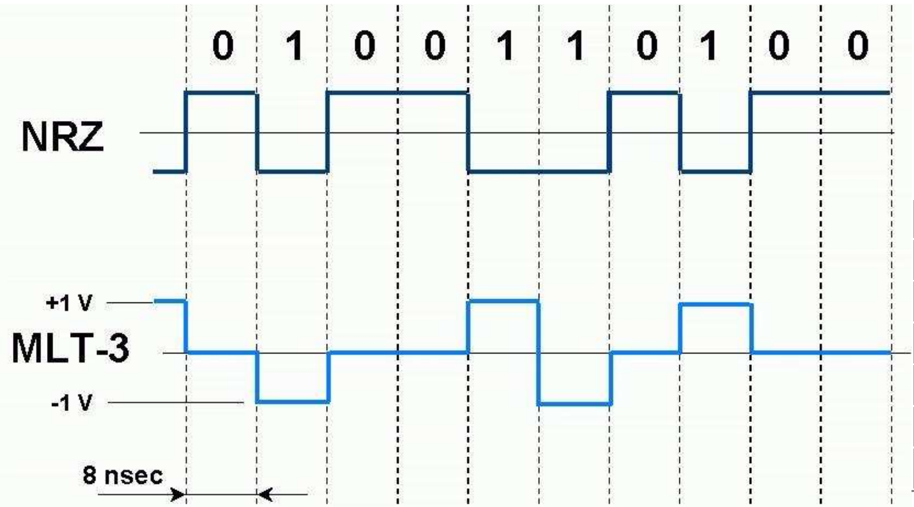
1. エミッタ
2. コレクタ
3. ベース

2SC

2SA







Type	Data Rate	Pairs Used	Frequency
10BaseT	10Mbps	2	10MHz
100BaseT4	100Mbps	4	15MHz
100BaseTX	100Mbps	2	80MHz
100VG-AnyLAN	100Mps	4	15MHz
ATM155	155Mbps	2	100MHz
1000BaseT	1000Mbps	4	100MHz

EEPROM

#include <avr/io.h>

#include <avr/eeprom.h>

int main(void) { uint8_t result;

eeprom_busy_wait(); /* 読み書き可能になるまで待つ */

eeprom_write_byte(0x00, 0xAA);　/* 値0xAAをEEPROMの0番地に書き込む */

eeprom_busy_wait(); /* 読み書き可能になるまで待つ */
result = eeprom_read_byte(0x00); /* EEPROMの0番地の値を読み出し変数val2に納める */

DDRB = 0xff;

PORTB = result;　/* PORTBに出力、LEDなどで表示 */

for (;;) {} }