電子おもちゃ

武藤佳恭

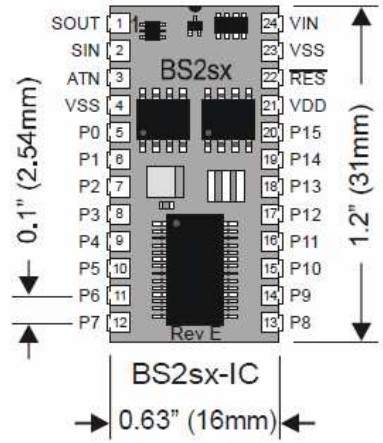
電子おもちゃ支援情報

- 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

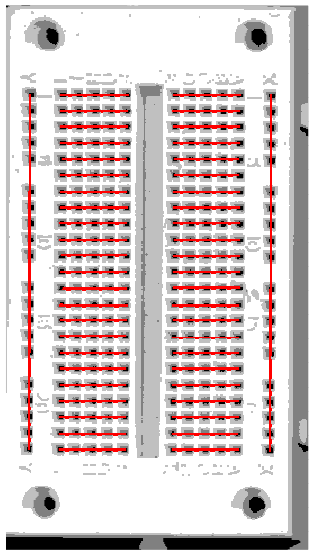
- Basic Stamp for Windows, Mac, Linux
- USB-RS232c ケーブル　1200円 （秋月）

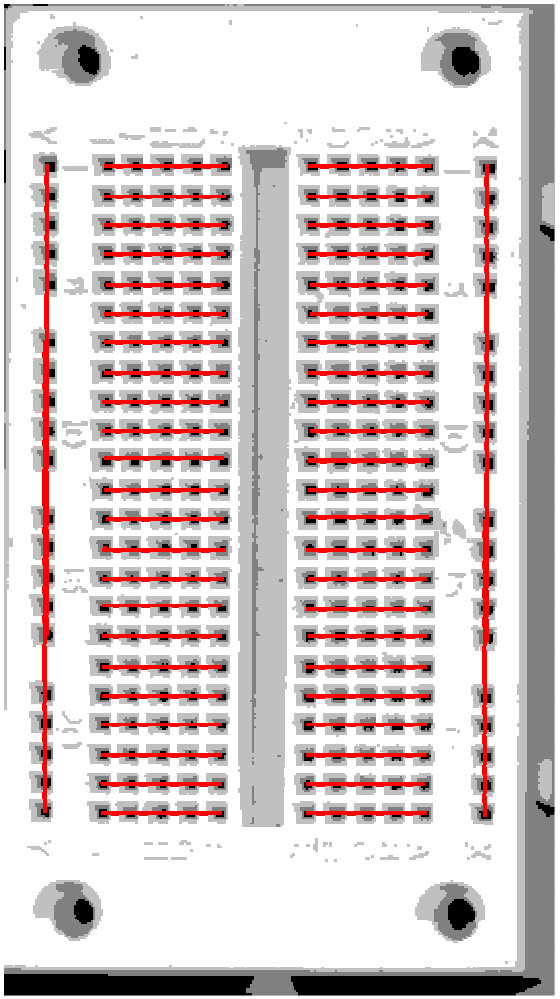
- BS2 3900円 （秋月）
- BS2SX　4700円 （秋月）

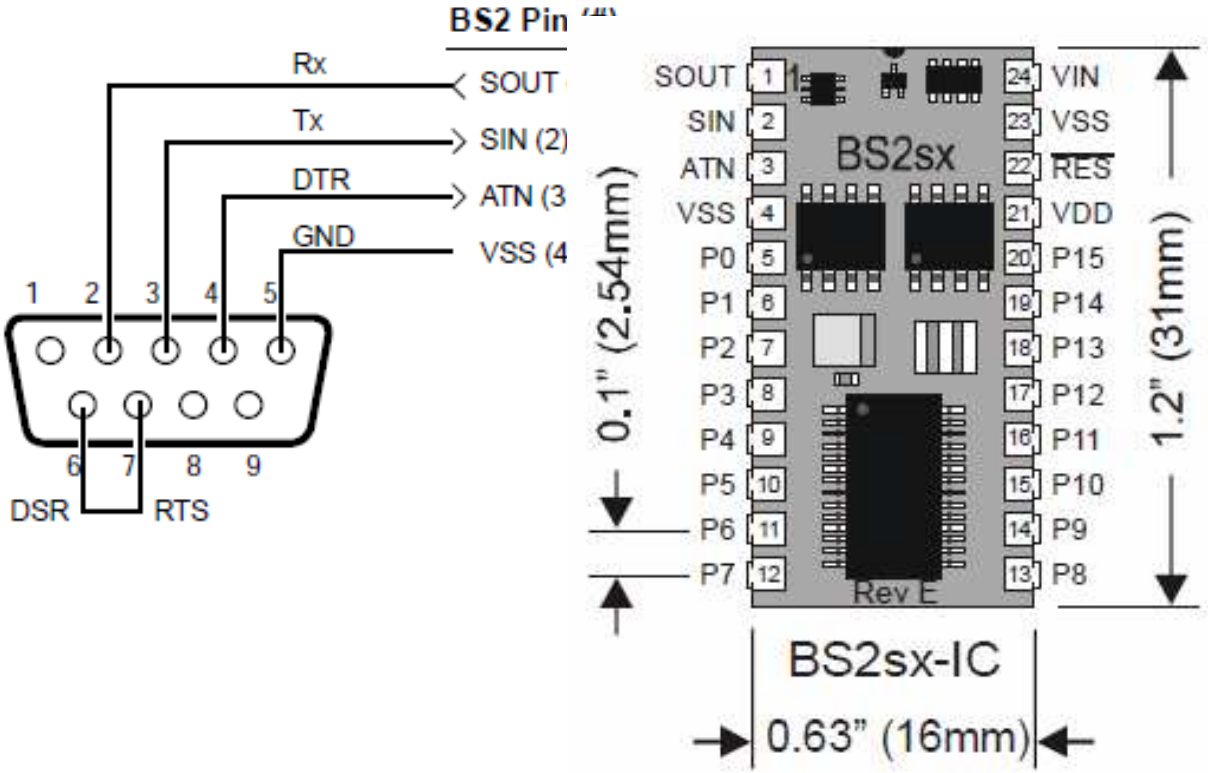


USB-RS232c

ブレッドボード






BS2 Pin
Rx
SOUT
Tx
SIN (2)
DTR
ATN (3
GND
VSS (4
1 2 3 4 5
6 7 8 9
DSR
RTS
SOUT 1
SIN 2
ATN 3
VSS 4
P0 5
P1 6
P2 7
P3 8
P4 9
P5 10
P6 11
P7 12
24 VIN
23 VSS
22 RES
21 VDD
20 P15
19 P14
18 P13
17 P12
16 P11
15 P10
14 P9
13 P8
BS2sx
Rev E
0.1" (2.54mm)
1.2" (31mm)
BS2sx-IC
0.63" (16mm)

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

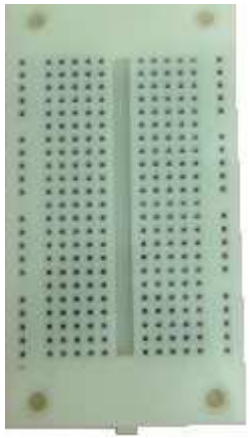
開発環境

- USB program writer
- RS232c program writer

main.hex

USB

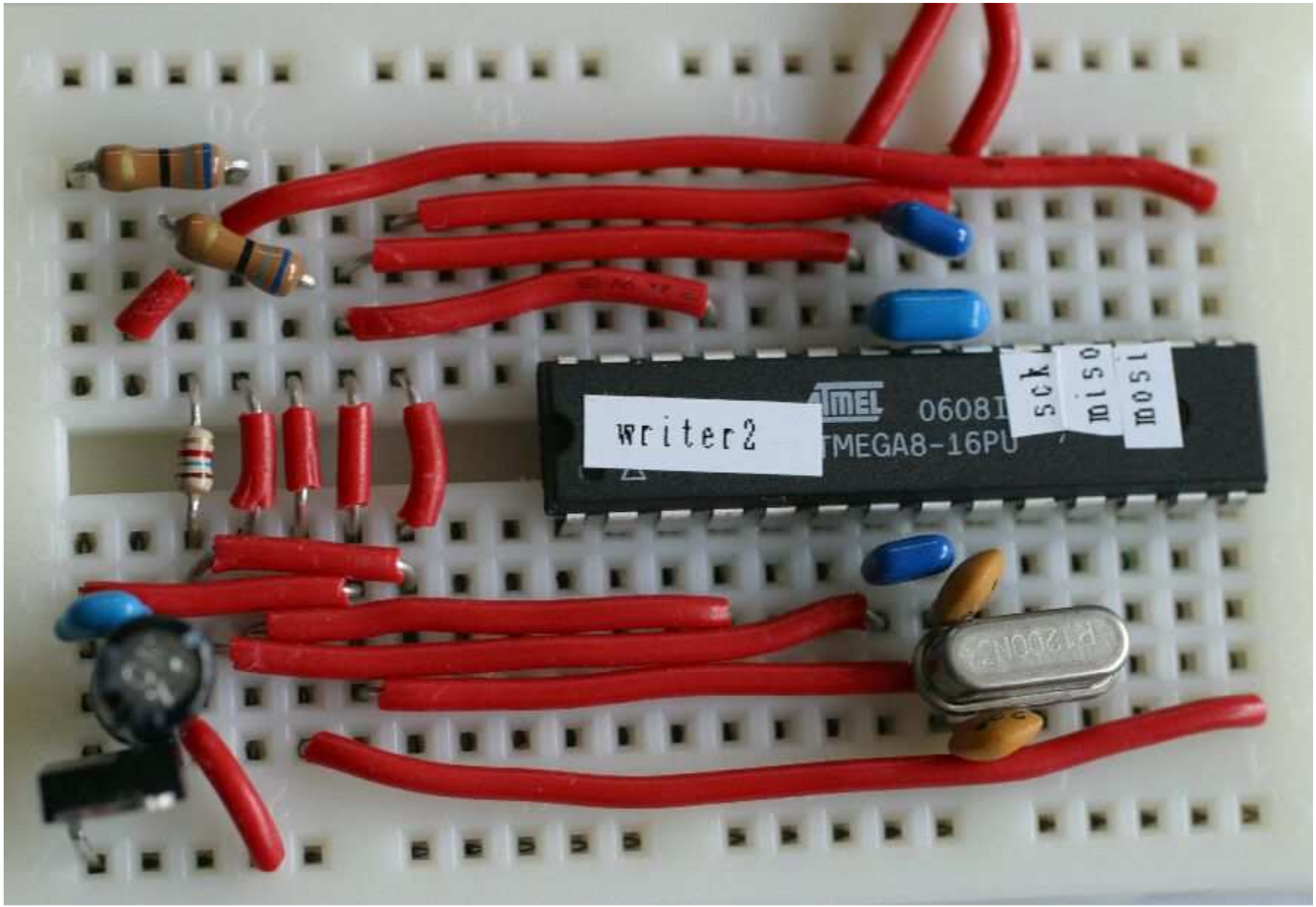
USB program writer



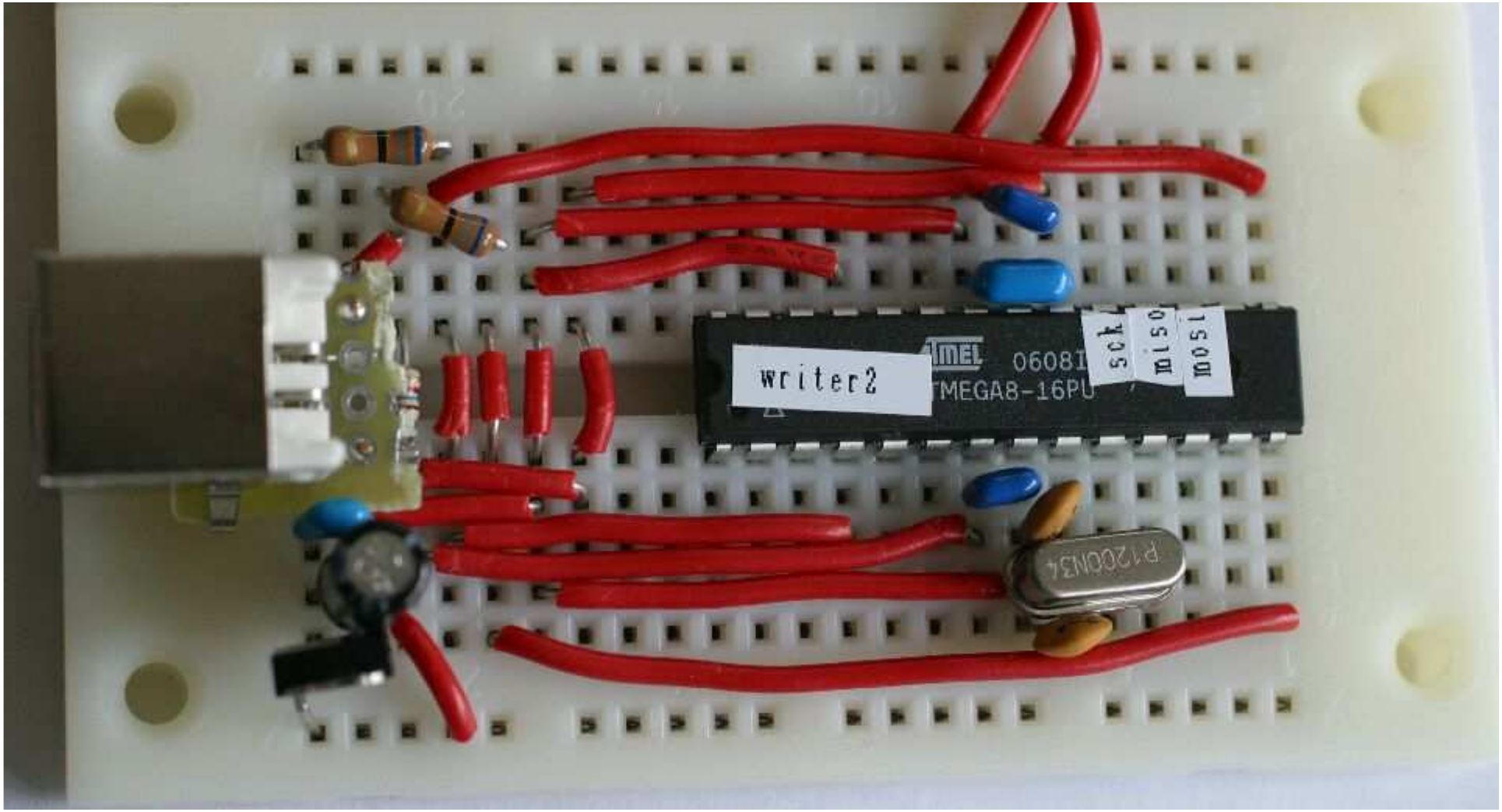
USB

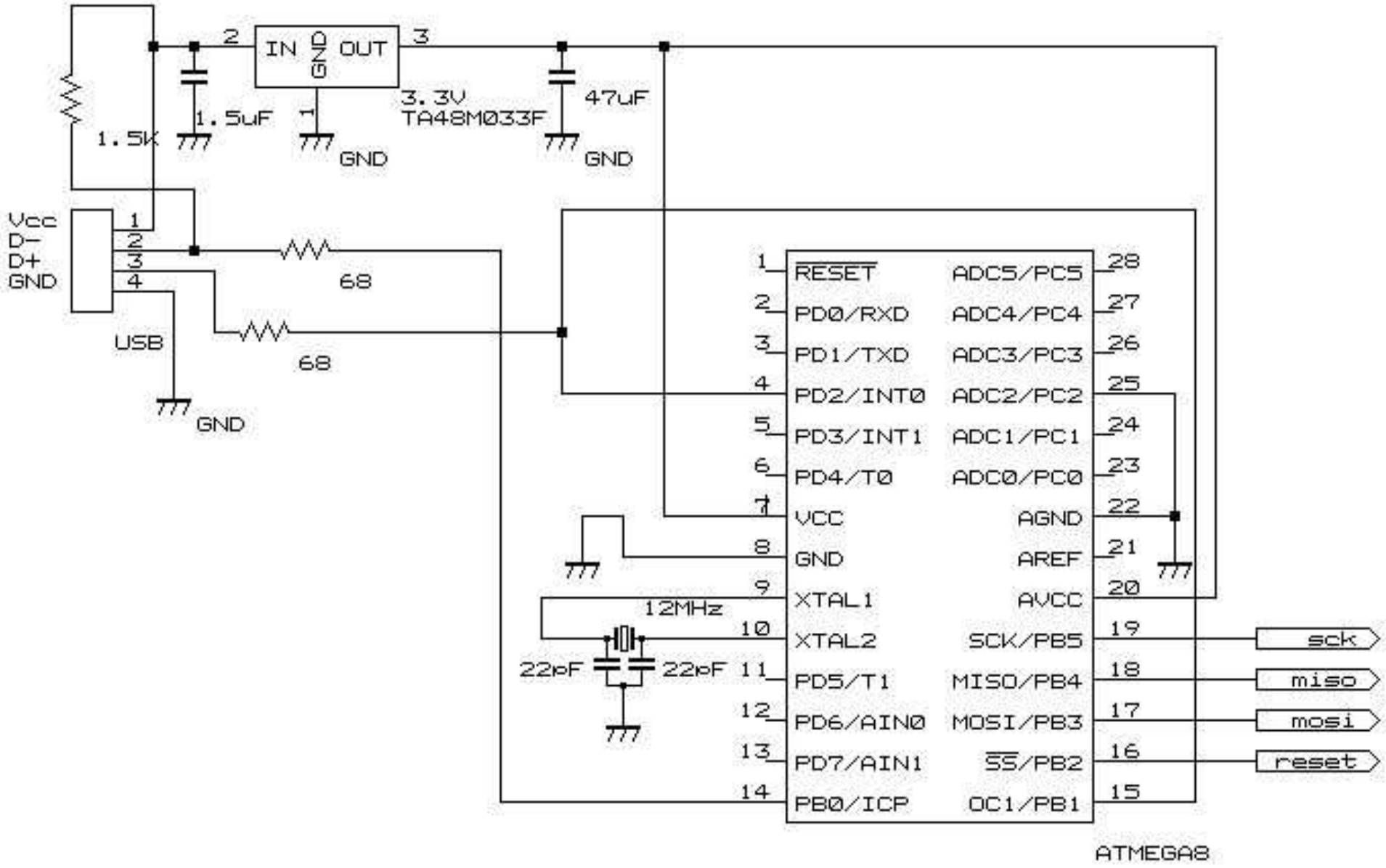
RS232c program writer

USB program writer



USB program writer





ATmega8 USB program writer by Takefuji

Program writerに必要な部品

- Atmega8
- ブレッドボード
- 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

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

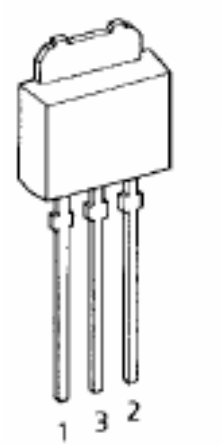
USB_CFG_DPLUS_BITはint0に接続する必要がある。

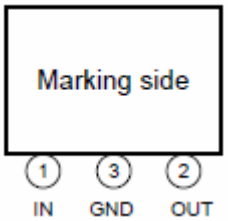
Device driver（PC側に常駐する）

- usbasp(プログラムライタ）のdevice driverは、

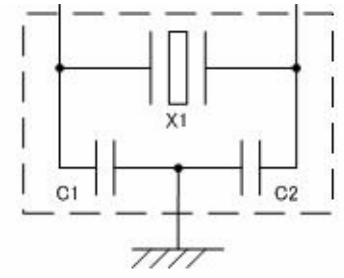
http://www.fischl.de/usbasp/から最新版のusbasp.tar.gzを解凍し、/bin/win-driverのドライバをインストールします。

- 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。

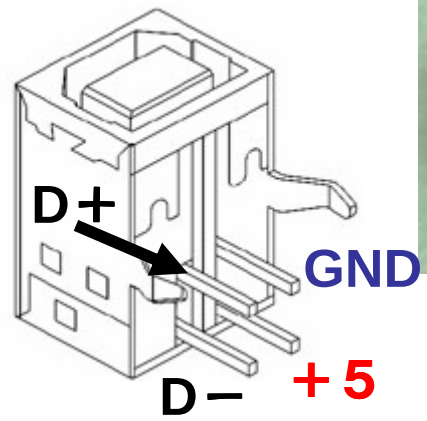




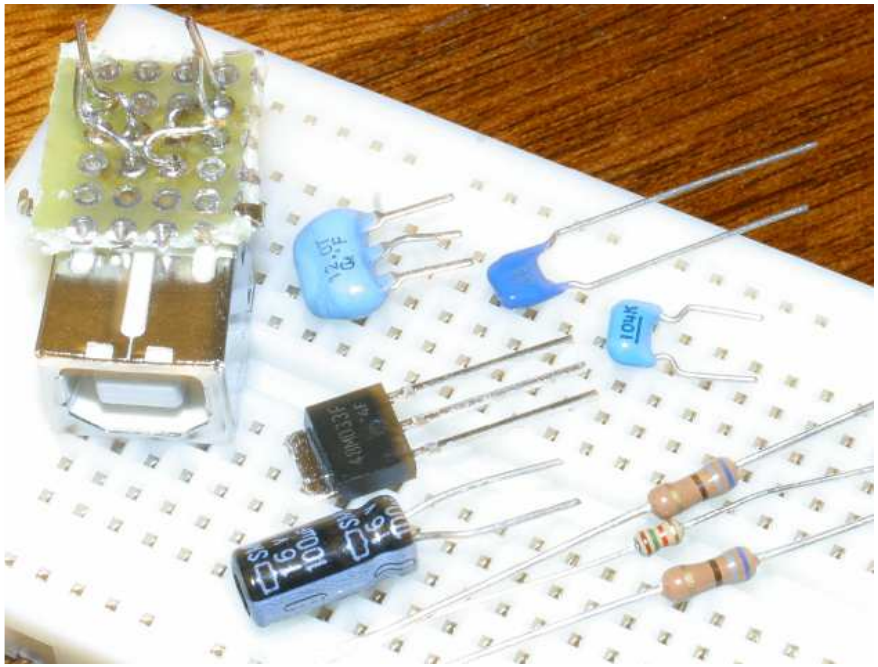
TA48M033F

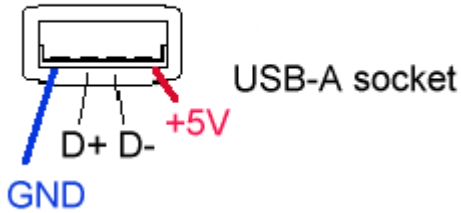


Ceralock



USB B Type PCB Female





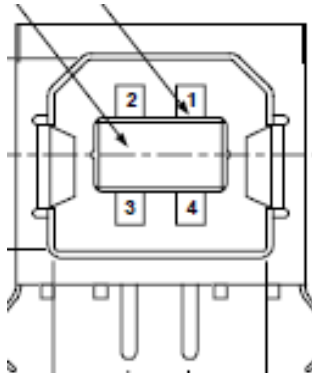
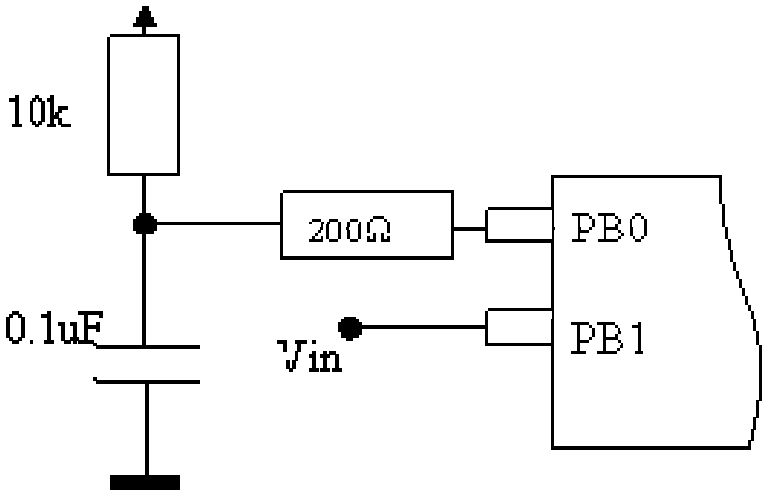
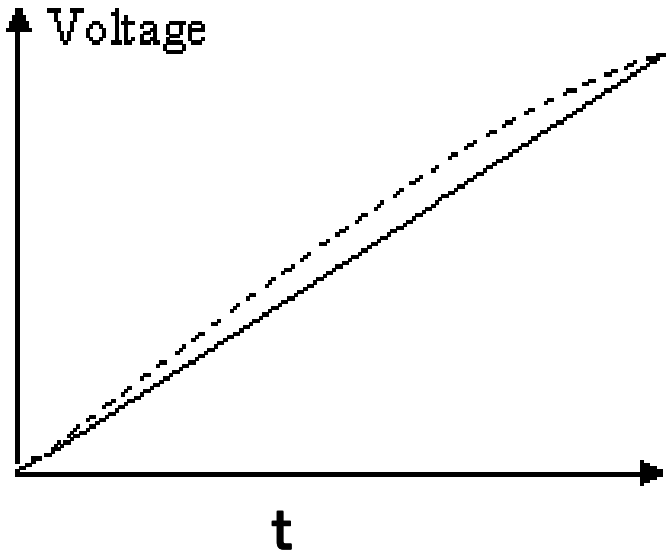


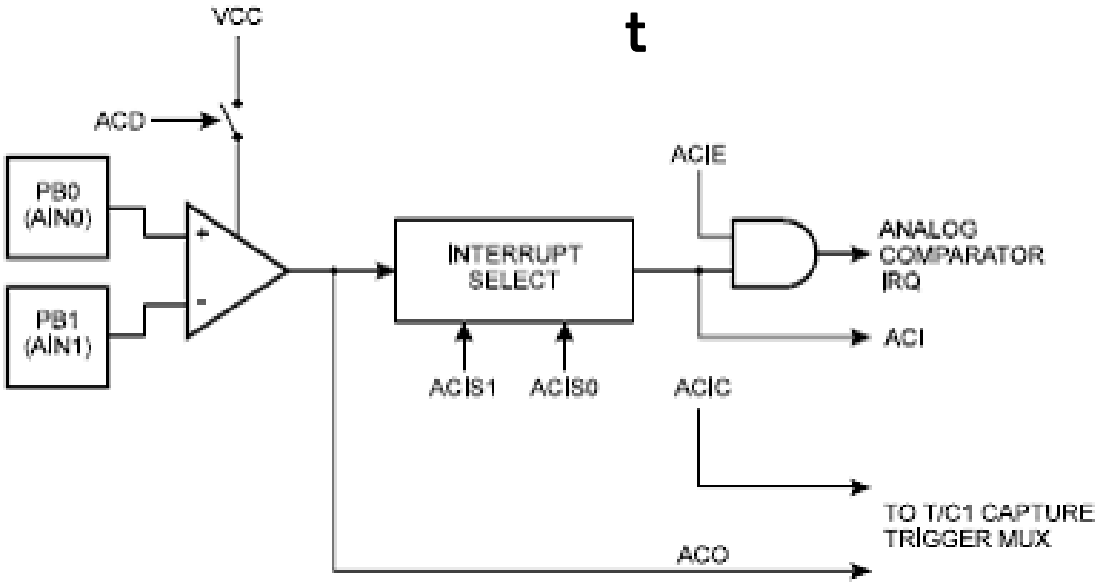
Table 6-1. USB Connector Termination Assignment

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

Analogデータのディジタルへの変換







積分回路

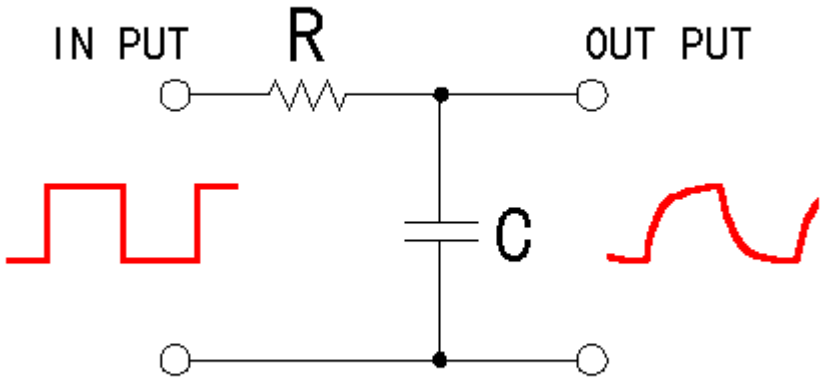
$$V = iR + Vc = RdQ/dt + Vc = Rd(CVc)/dt + Vc$$

$$Q = CVc$$

$$Ic = dQ/dt = i$$

$$V = RCdVc/dt + Vc$$

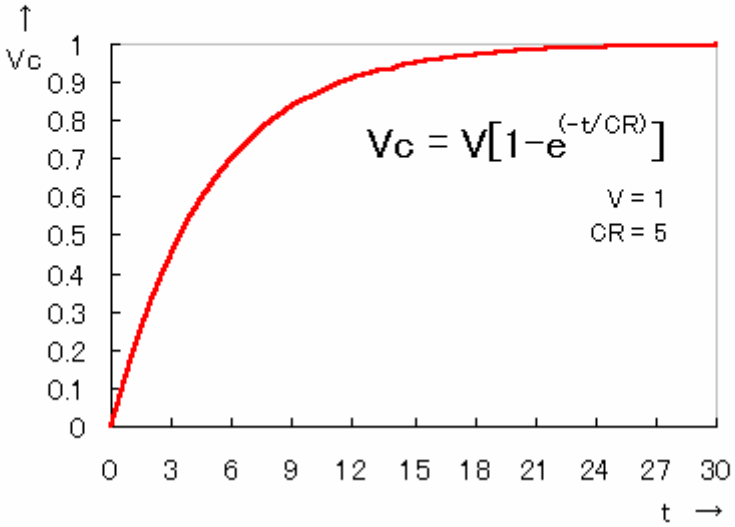
ode2('diff(v,t)+v=5, v, t);

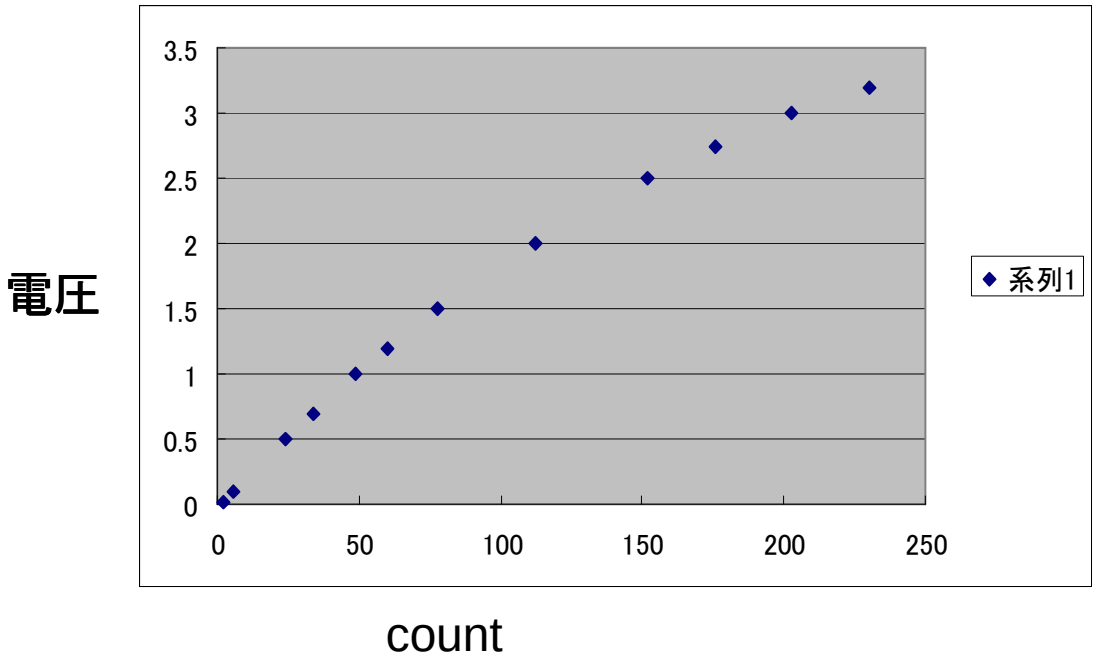


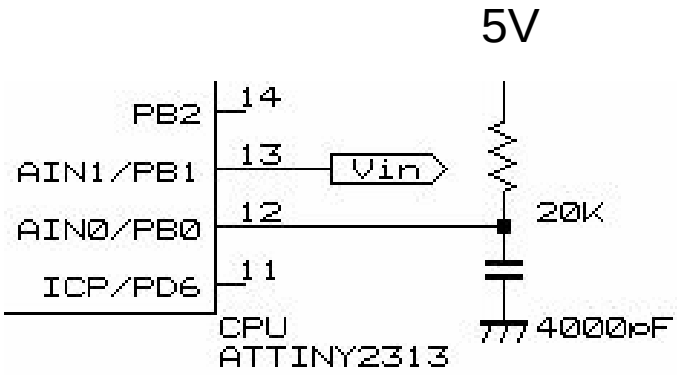
コンデンサ(C)の両端に加わる電圧(Vc)の変化は以下の式になります。

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

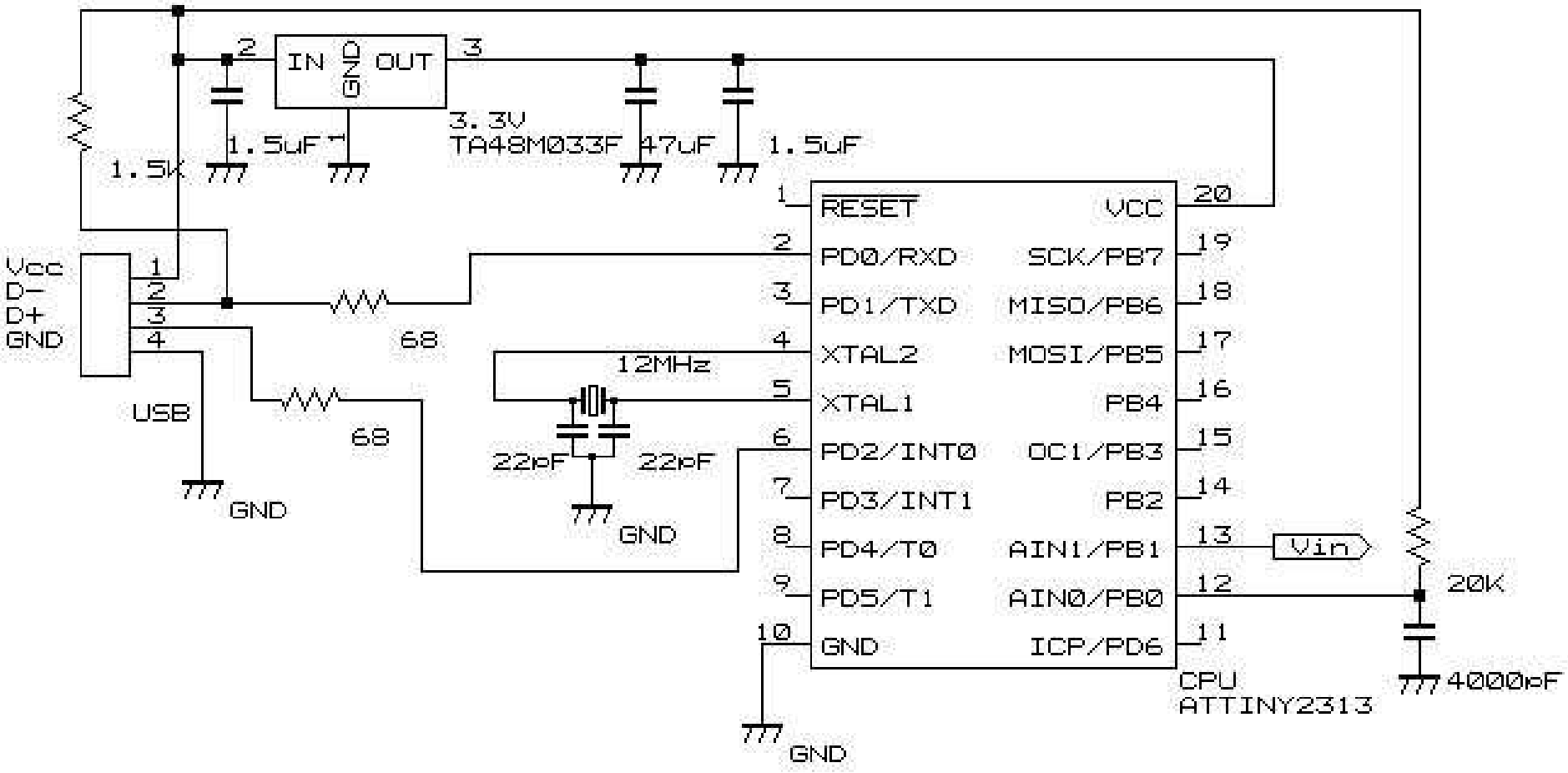
上の式をグラフで表すと以下のようになります。





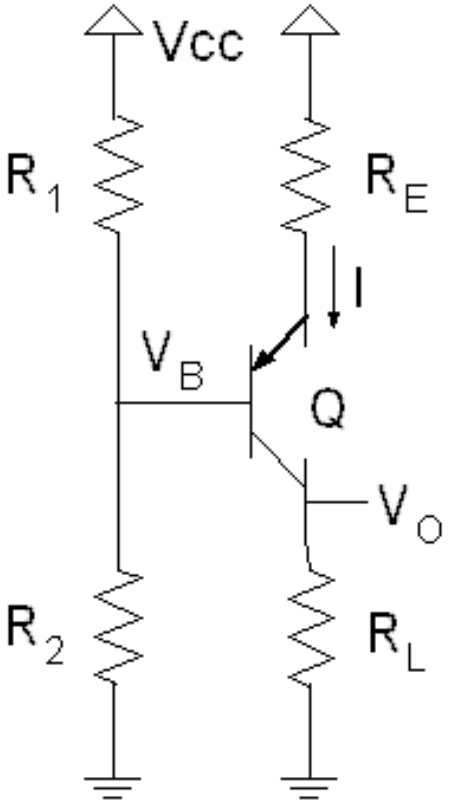


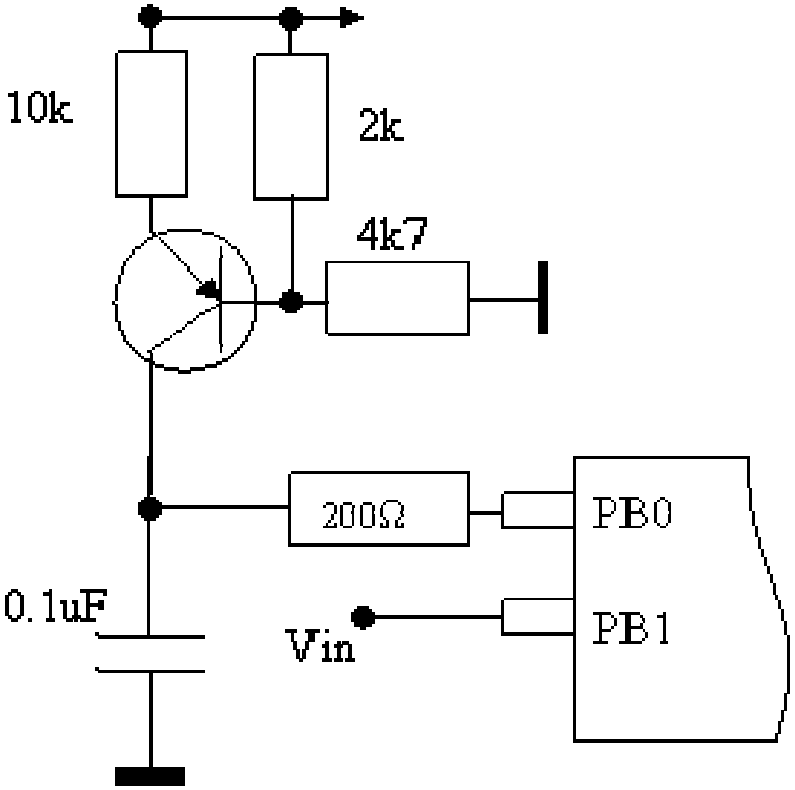
AT90s2313 アナログ電圧の精度



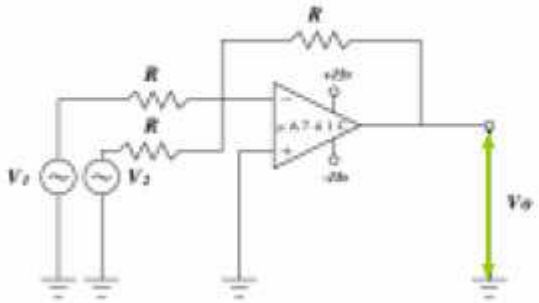
2313 analog入力回路

定電流回路





加算回路



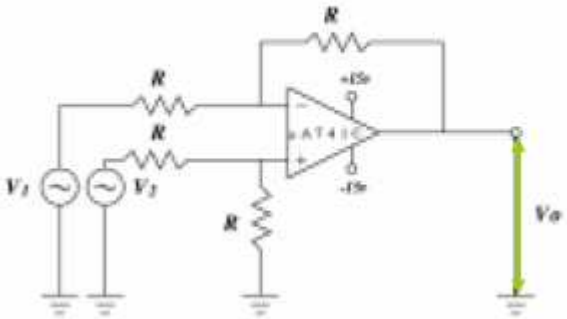
加算回路の解析

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



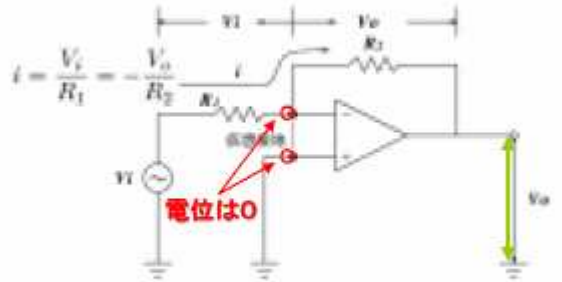
減算回路の解析

一入力端子における電流の関係：

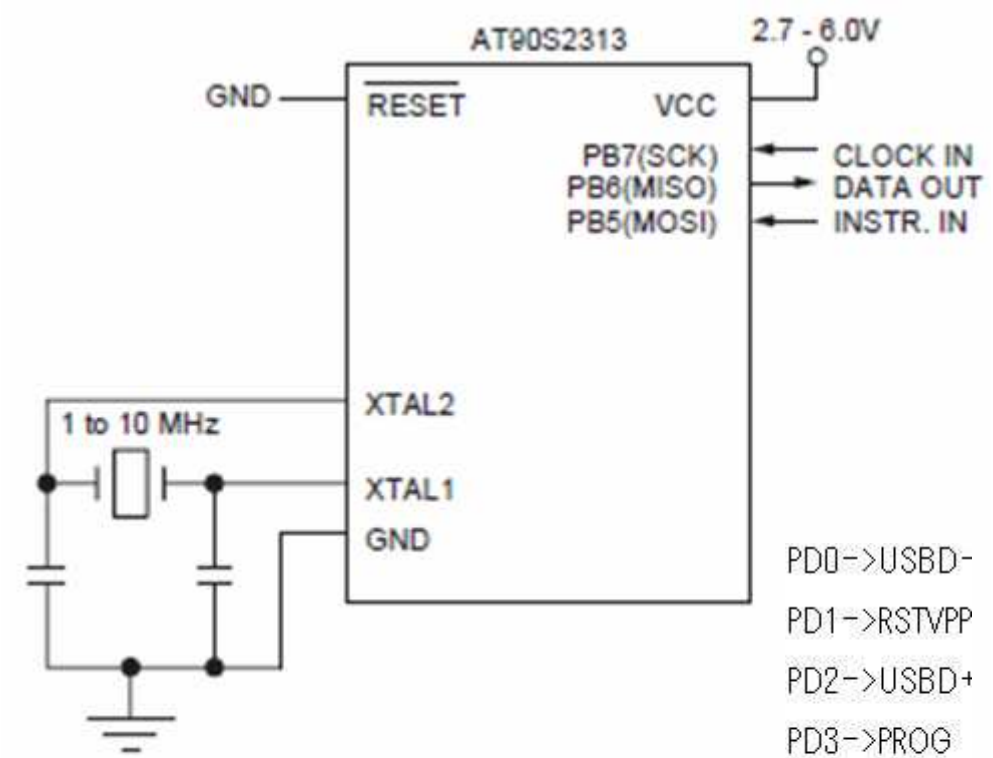
$$\frac{1}{R}\left(V_1-\frac{V_2}{2}\right)=\frac{1}{R}\left(\frac{V_2}{2}-V_o\right)$$

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

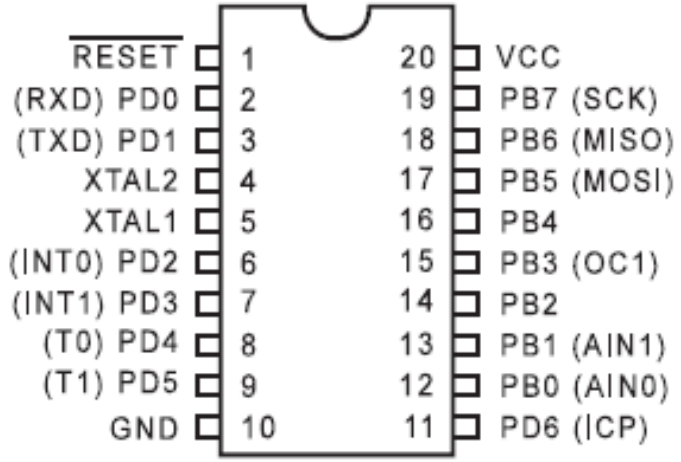
反転増幅回路

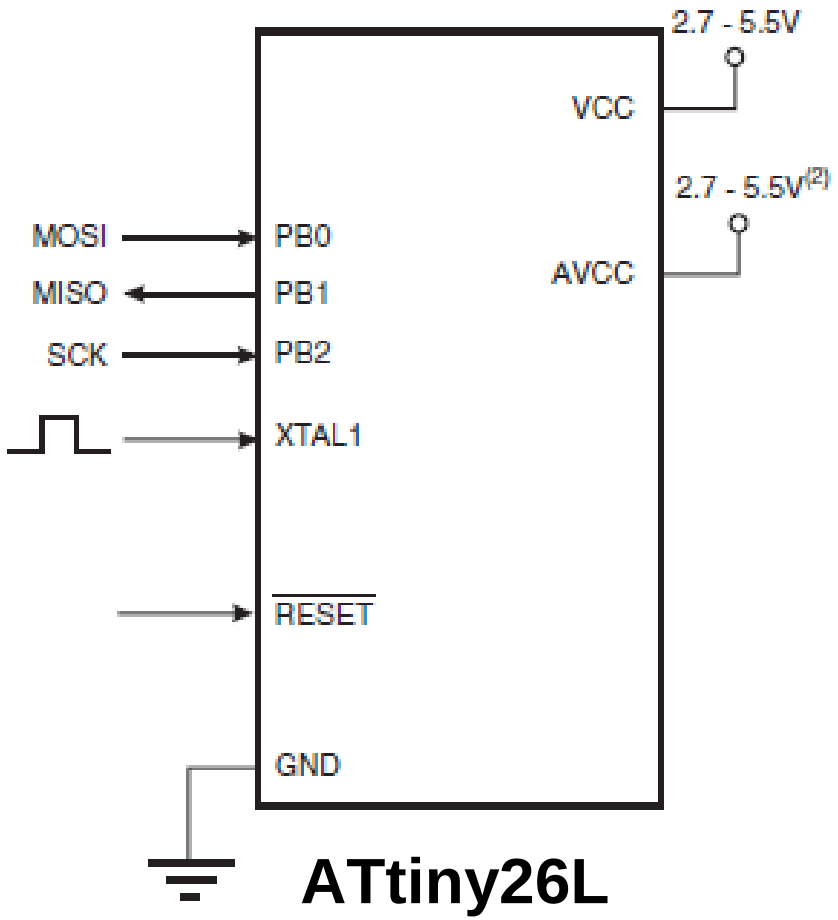


$$A=\frac{V_o}{V_i}=-\frac{R_2}{R_1}$$ ：位相が反転

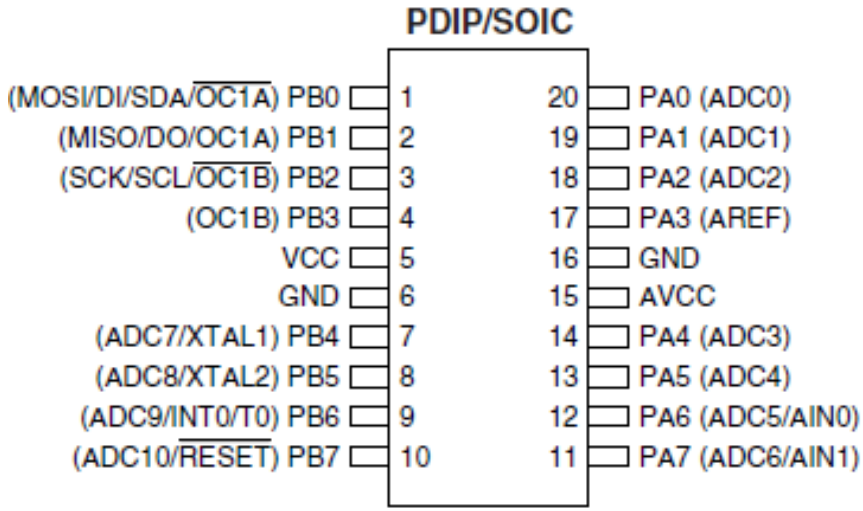


AT90S2313

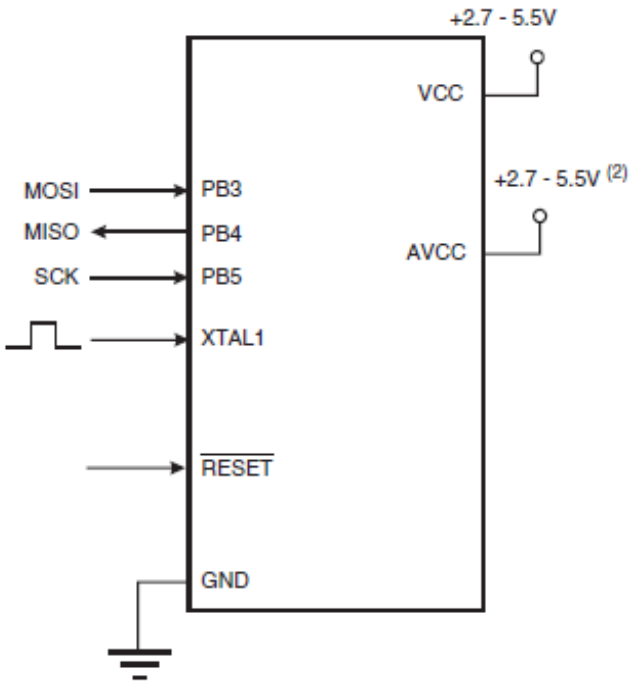




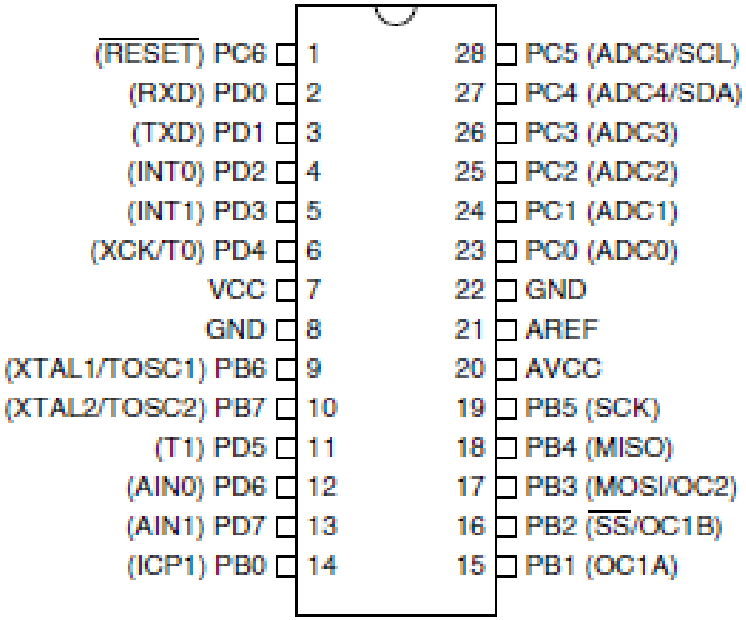
ATtiny26L



ATMEGA8

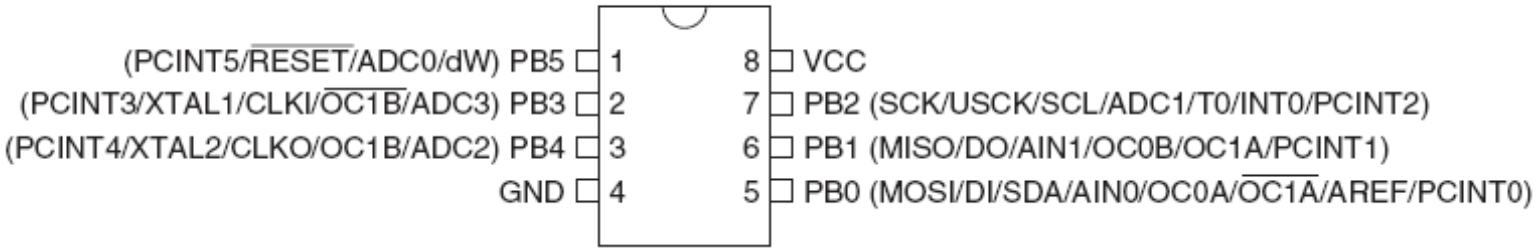


PDIP



TINY45

PDIP/SOIC



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
```

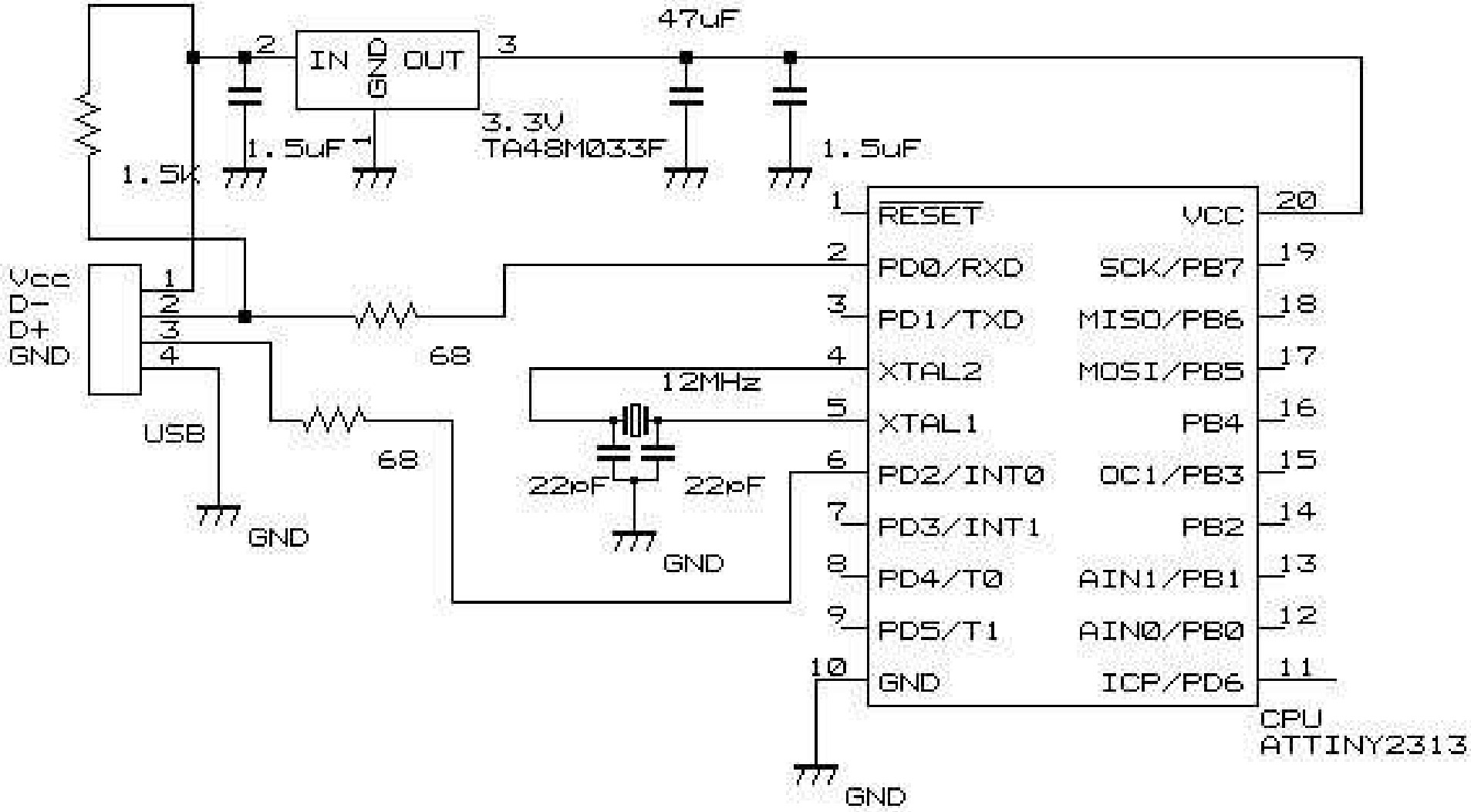
where:

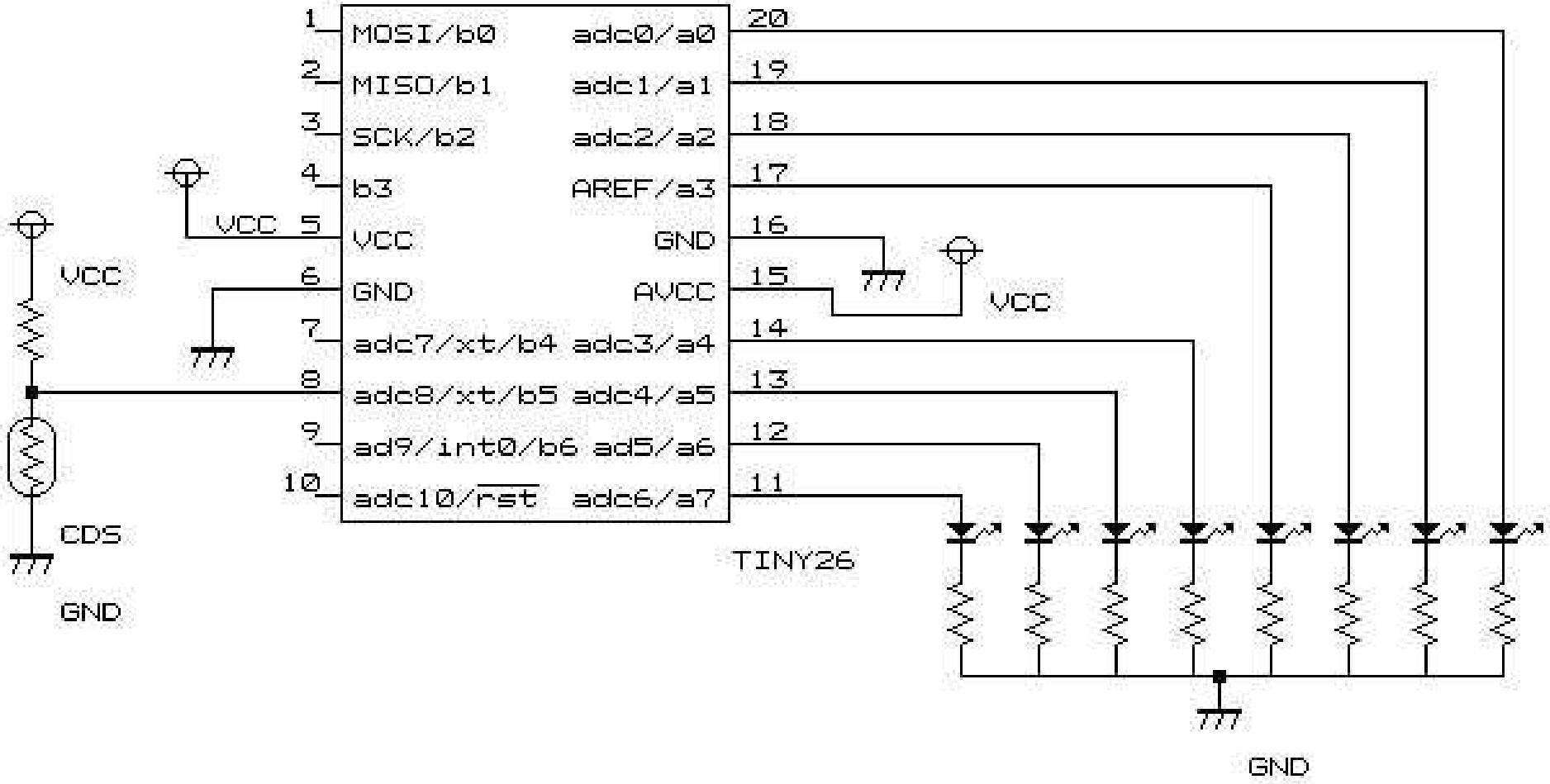
- **10** is the number of data bytes in the record.
- **2462** is the address where the data are to be located in memory
- **00** is the record type 00 (a data record).
- **464C...464C** is the data.
- **33** is the checksum of the record.

```
:00000001FF
```

where:

- **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
2
IN
GND
OUT
3
3.3V
TA48M033F
1.5uF
1.5uF
1.5K
Vcc
D-
D+
GND
USB
68
68
12MHz
22pF
22pF
GND
GND
GND
1
RESET
2
PD0/RXD
3
PD1/TXD
4
XTAL2
5
XTAL1
6
PD2/INT0
7
PD3/INT1
8
PD4/T0
9
PD5/T1
10
GND
20
VCC
19
SCK/PB7
18
MISO/PB6
17
MOSI/PB5
16
PB4
15
OC1/PB3
14
PB2
13
AIN1/PB1
12
AIN0/PB0
11
ICP/PD6
CPU
ATTINY2313


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

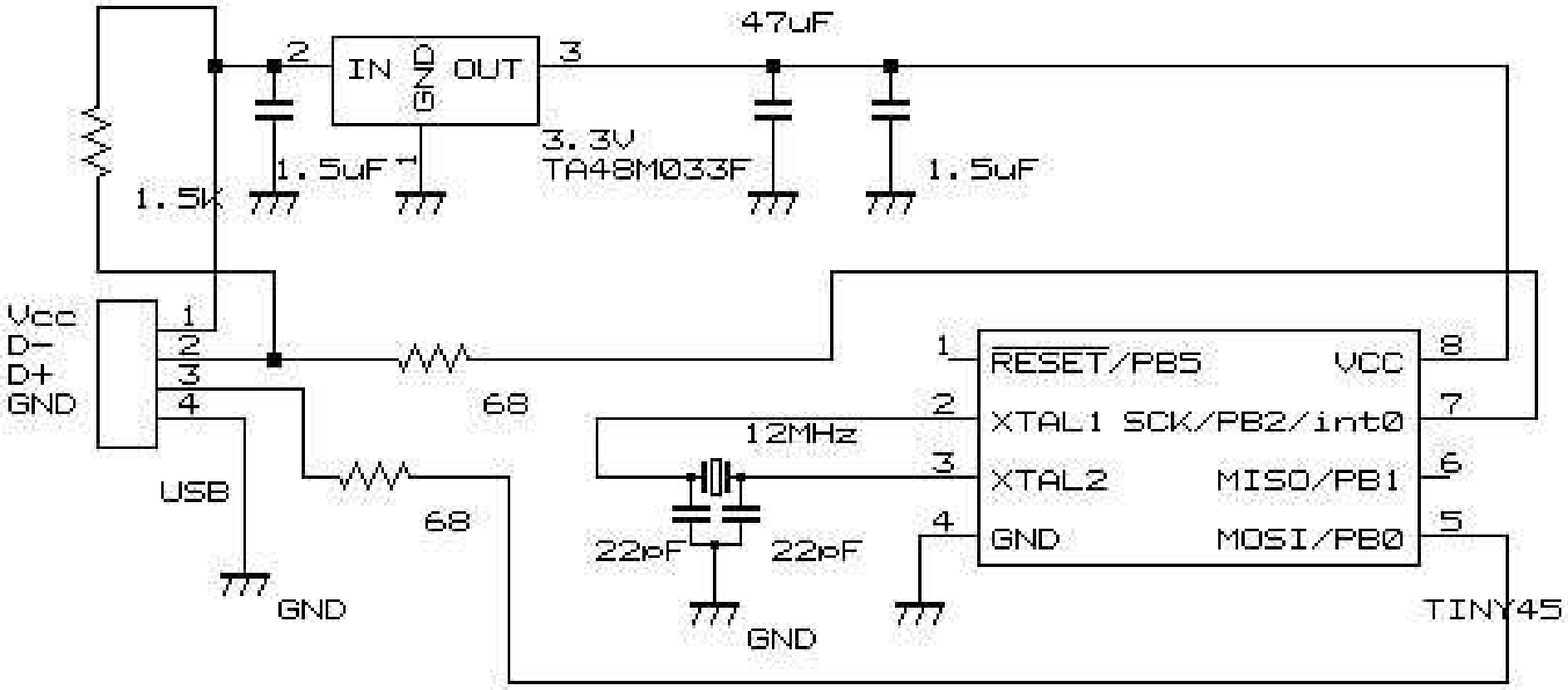
ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **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 with ADC by takefuji