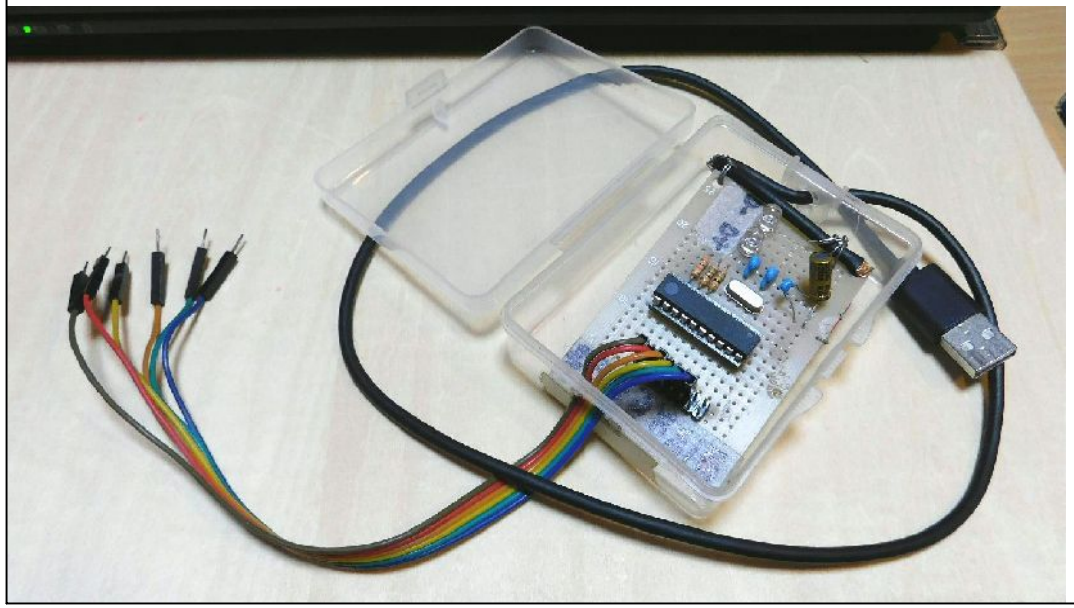
今回の　AVRライター hidspx に　ついて

　画像の物です。Uさんが、いくつか 作ってい て、そのうちの 一つを　私の所に 送ってくれま した。 ホスト側インタフェースは　USBで、ター ゲットマイコン側は　6本の信号線を　接続する 事になります。 ライター上のマイコンは、 ATtiny2313-20PUです。水晶は 12Mhzです。 USB通信の関係でしょう。



　ターゲットインタフェースの末端は、ブレッド ボードでの使用を　想定して作られたようです。 で、どの色のリード線が　どの信号線に対応する のか、把握しておく必要が　あります。

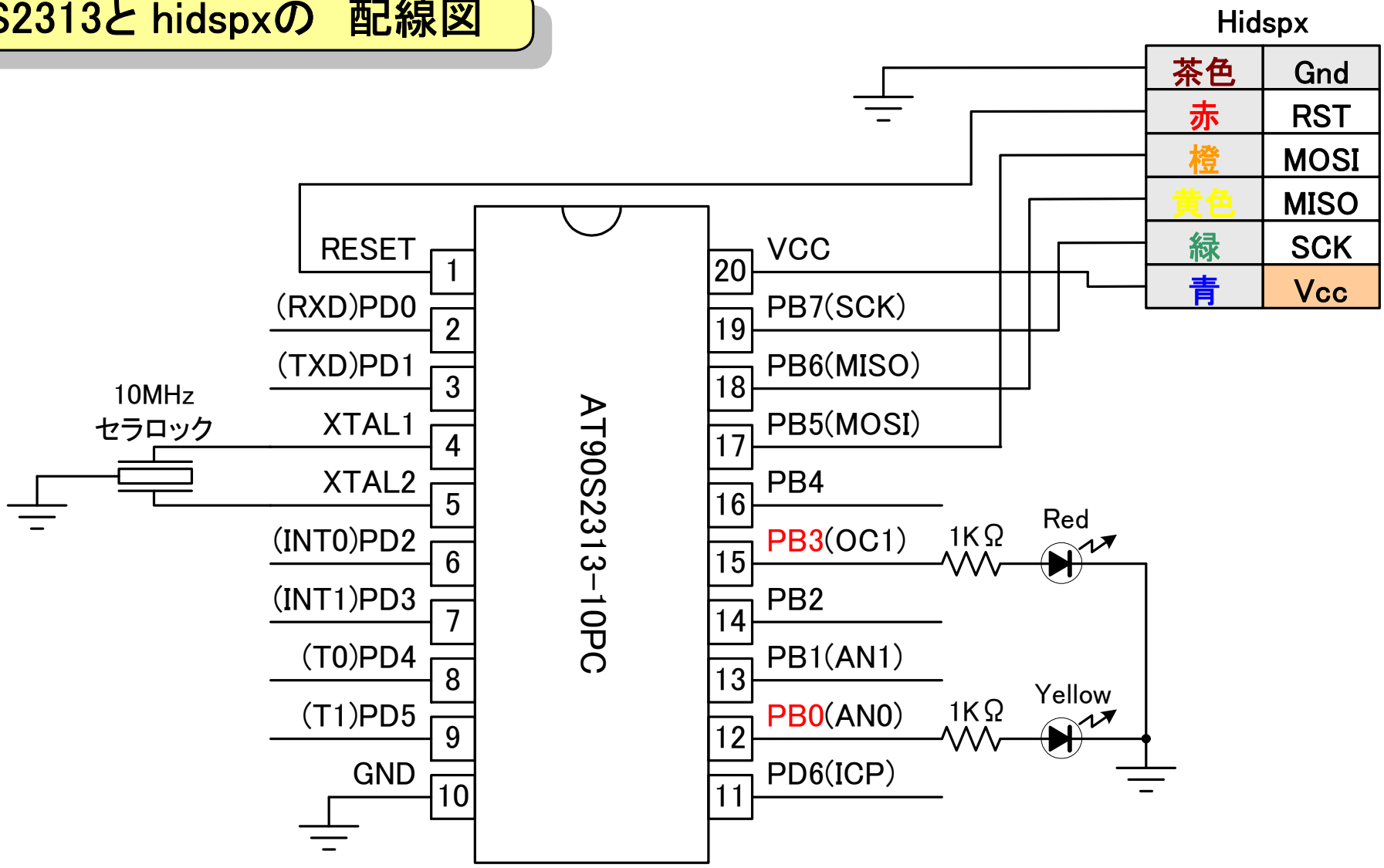
茶色	Gnd
赤	RST
橙	MOSI
黄色	MISO
緑	SCK
青	Vcc

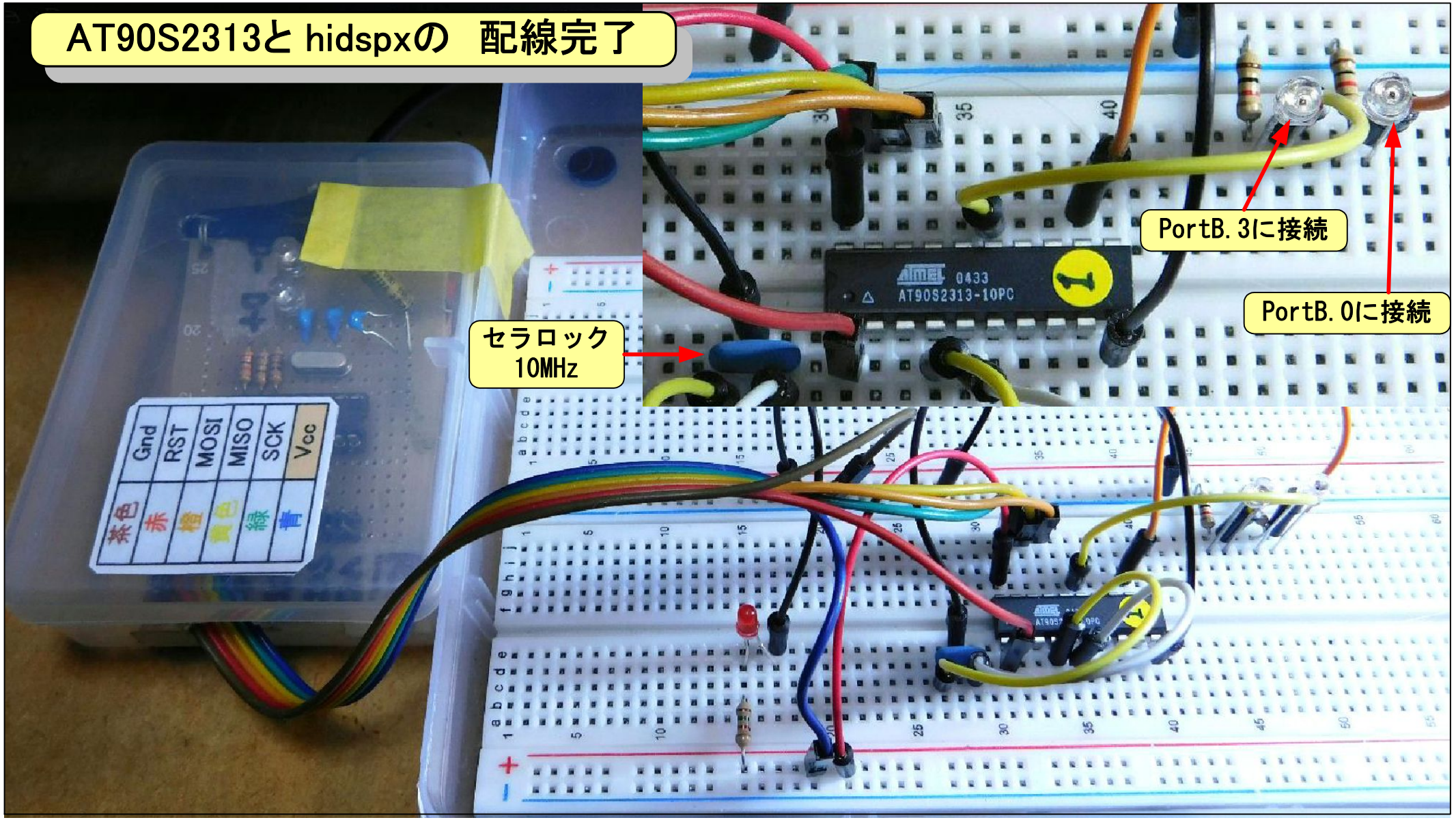
　接続に関しては、電源 Vcc と、Gndは　絶対 間違えない ように　ターゲットマイコンに 配線します。 その他の信号 線は、一時的に間違えても すぐに壊れる事は、無いと思 います。

　では、今回の目標である、AT90S2313に どのように接続するかを、次のページで 回路図で　示します。

　ついでに、実行時に必要となる動作確認用 の LED＋電流制限抵抗と　10MHzのセラロッ クも　接続する回路図とします。 LEDの点灯論理は　正論理とします。

AT90S2313と hidspxの　配線図




AT90S2313と hidspxの　配線完了
セラロック
10MHz
PortB.3に接続
PortB.0に接続
Gnd
RST
MOSI
MISO
SCK
Vcc
黒色
赤
橙
黄色
緑
青
AT90S2313-10PC

hidspx使って AT90S2313へ　書き込み

皆様　大変お待たせしました。
やっと　AT90S2313へ　プログラムの書き込みを行います。
プログラムは、前回　Miraさんの サイトのコメント欄から　コピペした LEDblink.BASを BACOM-AVRで　ビルドして作成した LEDblink.hex を　書き込みます。

で、ちょっと面倒ですが、BASCOM-AVRは Windows上で動作する開発環境です。
AVR書き込み器 hidspx は　Linux(Lubuntu) の　ターミナル窓で 実行します。

```
$ hidspx LEDblink.hex
```

を　行います。

右に　hexファイルを 指定するだけで、すぐに書き込めるので、扱いやすいです。

hidspx は　hexファイルの書き込みだけでなく -r の　オプションを付ければ、ATマイコンの　型式名、そしてプログラム用、データ用のフラッシュROMの 容量も表示してくれます。 -e で　フラッシュROMの 消去だけも　してくれます。

では、
動画で 書き込みの実験を 見てみましょう。

```
'--------------------------------------------------------------------
'LEDblink_3.BAS
'--------------------------------------------------------------------
$Regfile="2313def.dat"
$Crystal=10000000
$hwstack=32
$swstack=8
$framesize=24
'--------------------------------------------------------------------
Config Portb = Output
Dim i as Integer
i=0
Do
    Portb.3 = 1         ' LED on
    i = i + 1
    Waitms 500          ' 500 ms
    Portb.3 = 0         ' LED off
    Waitms 500          ' 500 ms
    If i >= 10  Then
        PORTB.0 = 1         ' LED2 on
        i = 0
        Waitms  200         ' 200 ms
        PORTB.0 = 0         ' LED2 off
    End If
Loop
```

LEDblink_3.BAS

余分な、コーディングを消したり
Waitms の 引数というか
値を 変更しました。

これで、正常なコーディングに
なったと 思います。

次は gccを使って プログラムをビルド

次は gcc-avrを 使って AVRマイコンに書き込む hexファイルを 生成します。
まずは、gcc-avrを メイク、インストールします。
Linuxの ターミナルウィンドウで、以下の コマンドを 実行します。

```
$ sudo apt install gcc-avr binutils-avr avr-libc make
```

を 行います。

正常終了すれば、AVRマイコン用の gccコンパイラが インストールされてます。

次は テストプログラムを 生成します。
さしあたり、右の サンプルソースを
BASCOMのプログラムと 同じクロック周波数
同じ LED接続の I/Oポートに 変更して
使用します。

```
#define F_CPU 10000000UL
#include <avr/io.h>
#include <util/delay.h>

int main( void )
{
    int i;

   DDRB = 0x09; // Set Dir
    while( 1 )
    {
       PORTB = 0x08;
      _delay_ms( 300 );
       PORTB = 0x00;
      _delay_ms( 100 );

       PORTB = 0x01;
      _delay_ms( 100 );
       PORTB = 0x00;
      _delay_ms( 100 );
       PORTB = 0x01;
      _delay_ms( 100 );
       PORTB = 0x00;
      _delay_ms( 100 );
    }
}
```

次は ビルドの方法です。

コマンドが 3行あって、面倒に思えますが、最初の1回目 ビルドのコマンドをターミナルに打ち込むと コマンド履歴を憶えていてくれますので、2回目以降は [↑]キーを 押す事で コマンドの履歴を出せます。 よって、2回目以降は 楽です。

以下の 3行で ビルドを 行います。 は 英大文字 O です。

```
$ avr-gcc -g -O2 -mmcu=at90s2313 -c -o main_2.o main_2.c
$ avr-gcc -g -O2 -mmcu=at90s2313 -o main_2.elf main_2.o
$ avr-objcopy -j .text -j .data -O ihex main_2.elf main_2.hex
```

を 行います。 これを見ると

main_2.c -> main_2.o -> main_2.elf -> main_2.hex と順に 変換されるようです。 正常に ビルド出来たら

```
$ hidspx main_2.hex
```

で、書き込みを 行って下さい。