



サイト内検索:

☒ AND検索☐ OR検索管理メニュー: [トップ](#) [新規](#) [編集](#) [リロード](#) [添付](#) [凍結](#) [差分](#) [最終更新](#) [バックアップ](#) [MENU編集](#) [一覧](#)

AVR/hidspcx_tips

[Prev](#)[AVR](#)[Next](#)

Counter: 75, today: 7, yesterday: 31

- ===== hidspcxに関連するTips =====
 - AVRマイコン開発環境との統合(for Windows)
 - 「Atmel社の開発環境(AVR studio)」
 - AVR studioにhidspcx用の書き込みボタンを追加する
 - 「BASCOM-AVR」
 - 「MikroC PRO 2008 for AVR」
 - hidspcx用のGUIフロントエンド
 - FUSEデータ確認方法(Windowsの場合)
 - FUSEデータ確認方法(Linux, MacOSの場合)
 - w3mを利用する
 - lynxブラウザ
 - 文字ベースのWebブラウザ利用上の注意点
 - GUIブラウザ
 - リモートログイン時にWebブラウザで開く
 - 時間計測機能について
 - 「-dオプション」と読み出し時間の関係
 - 実行時間の計測方法

===== hidspcxに関連するTips =====[↑]

[AVR/HIDAspx](#)のページが長大になり、閲覧に時間がかかるようになってきました。

そのため、Tipsの追加も難しくなってきたので、別ページに分離しました。

今後は、各種のノウハウをこのページを充実させていきます。

なおココに記したものは、HIDAspxのハードを前提にするもの以外は、COM-SPIブリッジなどのhidspcxがサポートするAVRライターでも使えます。HIDAspx以外の利用者もご活用ください。

HIDAspx用の制御コマンドhidspcxは、Windows(2000/XP/Vista), Linux, MacOS Xで利用でき、共通の操作が可能です。

[↑](#)

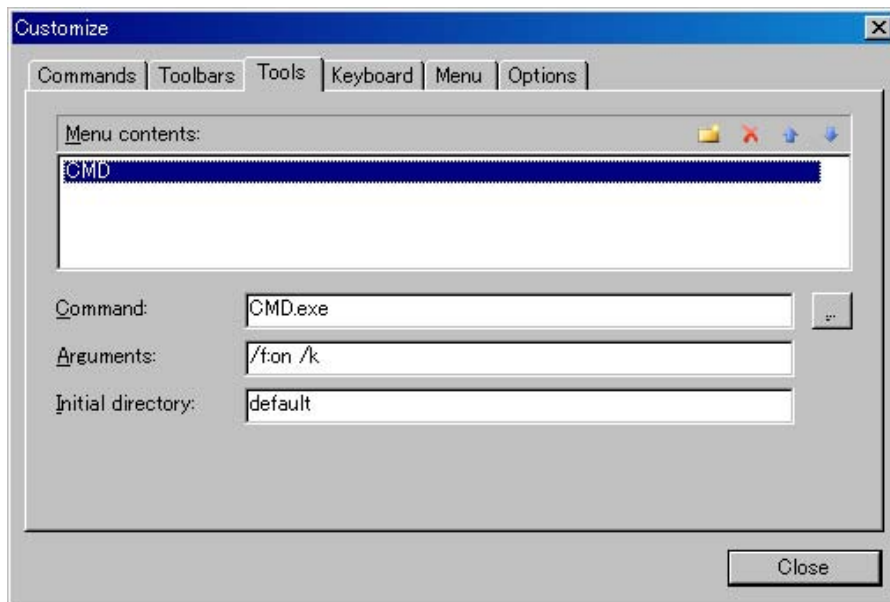
AVRマイコン開発環境との統合(for Windows)[↑]

各種の開発環境から、HIDAspxを利用する事ができます。

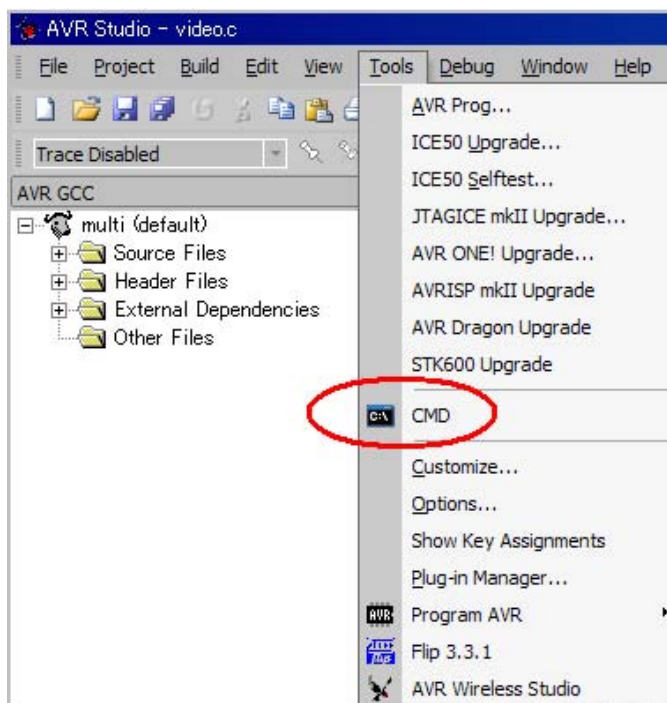
[↑](#)

「Atmel社の開発環境(AVR studio)」[↑]

AVR studioでは、外部ツールを呼び出す仕組みを持っています。「MENUのTools Custmize Tools」と選択し、以下の内容を登録します。



以下のメニューを選択すると、HEXファイルが生成される場所でCMDプロンプトを起動できます。



CMDが表示されれば、その中で**hidspix *.hex**を入力し、書き込むことができます。Commandに **hidspix.exe**、Argumentsの部分に、***.hex**を登録することも不可能ではありませんが、FUSE設定や「-d」指定ができないため、お奨めはできません。

AVR studioにhidspix用の書き込みボタンを追加する[↑]

audinさんが、「avrwan : One Click Programmer wrapper for AVR-Studio」のタイトルで興味深い取り組みを展開しています。

<http://avr.paslog.jp/article/1075132.html>

このアイデアを元に、GAWKで書いた例を紹介します。defaultフォルダでの利用を前提にしていますが、Makefileが見つからない場合には、defaultにあるMakefileを対象にするのがよいと思います(このコードは20分ほどで書いたので叩き台と思ってください)。HEXファイルを抽出するロジックは厳密なものではありませんが、AVR studioの生成するMakefileに限定すれば、これで問題はないと思います。

以下のコードはDEBUG用ですから、利用する場合には、コード内容をよく読み、systemの前の#とsleep関数を解除して使ってください。

```
# avrwrite.awk
# Written by senshu.
#
BEGIN {
    FILE = "Makefile";
    WRITE_CMD = "hidspk.exe";
    cmdline = WRITE_CMD;

    for (i=1; i<ARGC; i++) {
        cmdline = cmdline " " ARGV[i];
    }
#   print cmdline
    while (r = (getline line < FILE) > 0) {
        if (line ~ /all:/) {
#           print line;
            n = split(line, files, " ");
            for (i=1; i<=n; i++) {
                if ((files[i] ~ /[.]hex/) || (files[i] ~ /[.]EEP/)) {
                    cmdline = cmdline " " files[i];
                }
            }
        }
        if (r < 0) {
            printf("%s: File not exists.\n", FILE);
            system("sleep 5");
            exit 1;
        }
        print cmdline;
        system("sleep 5");
#       r = system(cmdline);
        if (r==0) {
            print "SUCCESS";
        } else {
            print "Error found";
        }
    }
}
```

この方法で欠点を挙げるなら、gawk.exeで表示されるアイコンがライタ用のボタンに見えないことです。上記の考え方で問題が無いのであれば、このロジックと同じ動作をするコードをC言語でリライトし、見栄えのするアイコンを付けると幸せになれるかもしれません。

現行のコードは、マクロ定義されたファイル名などを正しく扱うことができませんが、AVR studioから使うなら、この問題は生じないと考えます。

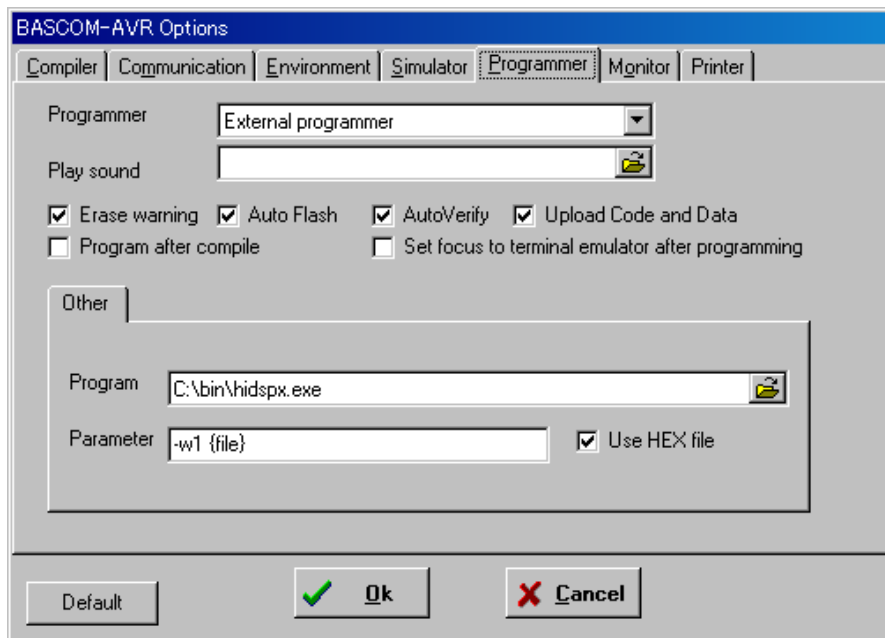


「BASCOS-AVR」[↑]

BASCOS-AVRは、一般的なAVR用プログラマを書き込み用プログラマとして設定できます。

BASCOS-AVRのメニューのOptions=>Programmerを選択し、以下のように(自分が利用している)hidspk.exeのフルパス名を指定します。これで、[F4キー]やメニューの書き込みボタンを操作すれば、hidspkで書き込みできます。

なおこの場合、BASCOS-AVRからはFUSE設定はできませんので、事前に CMDプロンプト上でhidspkを使って、希望するFUSE設定を書き込みます。



avrspxでも同じ設定で利用できます。

-w5 {file}

「-w1」は完了後も指示のあるまで表示を続けます。2以上の値(例えば**-w5**)は、書込み完了後にその指定した秒数だけCMD窓を表示することを意味します。これにより、書込み結果を確認できます。

↑

「MikroC PRO 2008 for AVR」↑

AVR freaks でmikroe社から、AVRマイコン用のC言語開発環境が2008年12月にリリースされました。

今までは、BASICとPascalだけでしたが、新たにC言語が追加され、ユーザーインターフェースもスマートなスタイルに一新しました。安定性その他は不明ですが、長らく待たれていたC言語用の開発環境です。

今まで使ってきたPICマイコン用と類似の関数を持ち、使い方もほぼ同様です。これが安定して利用できると、かなり便利になります。

WinAVRも十分便利なのですが、Atmel純正のシミュレータを利用するには、WinAVRとAVR studioを合計すると200MBを超えるdisk領域が必要ですが、この開発環境は32MBほどのサイズで利用でき、手持ちのAVRライタを開発環境に組み込むことも可能(HIDaspixライタもOK)です。

また、ANSI-C準拠度が高く、C言語を知っている方なら新たに学習する部分は極わずかです。

DEMO版でも、2kW(4kB)までのコードを作成できます。この程度のサイズまで利用できれば、かなり複雑なレベルまで利用できます。(ATtiny2313やATmega48なら制限なしです。)

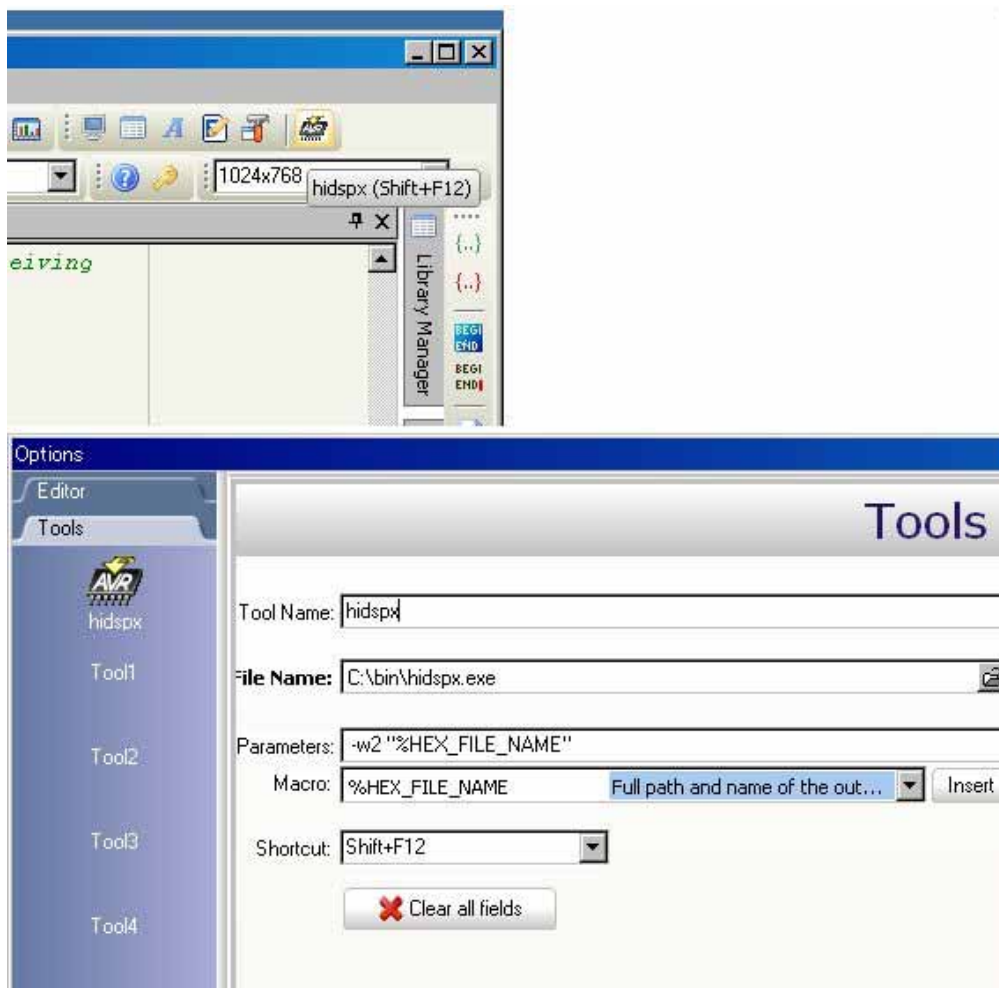
ただし、ツール自体のUIは非常に凝りすぎていてBUGが心配です。

<http://www.mikroe.com/en/compilers/mikroc/avr/>

また、AVR-GCCと異なり、const宣言したものはプログラムメモリ(Flash領域)に割り付けます。

実機が無くとも、内蔵のシミュレータでDEBUGが可能で、実行時間も詳細にチェックできます。また、豊富なライブラリが付属するので、各種のI/Oを短いコードで利用可能です。

以下のように外部ツールを設定すると、アイコンクリック(あるいはShif+F12)でHIDaspixを使って、AVRマイコンに書き込みが可能になります。ワンクリックとはいえませんが、ワンキーで便利に書き込みができます。



設定のポイントは、parametersに「-w2 "%HEX_FILE_NAME"」のように「"」で囲むことです。これは空白を含むファイル名の時に有効に機能します。

以下のプログラムでは問題なく動作しました。

```

/*
 * Project name:
 *   UART (Simple usage of UART module library functions)
 */

char uart_rd;

void main() {

    UART1_Init(9600);           // Initialize UART module at 9600 bps
    Delay_ms(100);              // Wait for UART module to stabilize

    while (1) {                 // Endless loop
        if (UART1_Data_Ready()) { // If data is received,
            uart_rd = UART1_Read(); // read the received data,
            UART1_Write(uart_rd);   // and send data via UART
        }
    }
}

```

- [senshu](#) 2009-01-27 (火) 18:01:58

こうしたツールを利用すると、RAINさん作のAVRライタのアイコンが映えます。
正に、どこかの製品の感じがします(気のせい?)

↑

hidspc用のGUIフロントエンド ↑

JA1WXYさんのページに、hidspix用の簡易GUI(D&Dに対応)が公開されています。

<http://www15.plala.or.jp/ja1wyx/seisaku/hidspix/hidspix1.html>

ソースコード公開なので改良は容易です。各自が希望する機能を追加してみましょう。



FUSEデータ確認方法(Windowsの場合) [↑]

avrx-tool.txtに書かれていない場合には、データシートを読むのが一番なのですが、必ずしも読みやすくはありません。どうい設定を行えば希望する動作が出来るかは、経験を必要とします。

そこで、hidspixにWebブラウザとの連携機能を追加しました。CMD窓で「start URL」と入力すると該当のページを開くことができますが、この機能を積極的に利用するわけです。

機能を拡張で考慮したことは、大幅な修正は行わない、比較的簡単に行える修正で利便性を増す、の2点です。そこで、avrライタに接続しているAVRマイコンに関する情報を適切に提供できる機能を追加するのが望ましいと考え、ライタから得られるFUSE情報を元に適切なページを開く機能を追加しました。

(1) 従来とおり「hidspix -rf」で確認出来ますが、詳細な情報が得られないため、希望する設定に変更するためには、データシートを参照する必要がありました。

```
Detected device is ATmega88.

Low: 11111111
|||++++ CKSEL[3:0] システムクロック選択
||++ SUT[1:0] 起動時間
|+ CKOUT (0:PBOにシステムクロックを出力)
+- CKDIV8 クロック分周初期値 (1:1/1, 0:1/8)

High:11-11101
|||++++ BODLEVEL[2:0] (111:無, 110:1.8V, 101:2.7V, 100:4.3V)
|||+- EESAVE (消去でEEPROMを 1:消去, 0:保持)
||+- WDTON (1:WDT通常動作, 0:WDT常時ON)
|+- SPIEN (1:ISP禁止, 0:ISP許可) ※Parallel時のみ
|+- DWEN (On-Chipデバッグ 1:無効, 0:有効)
+- RSTDISBL (RESETピン 1:有効, 0:無効(PC6))

Ext: ----000
||+- BOOTRST ※データシート参照
++- BOOTSZ[1:0] ※データシート参照

Cal: 170
```

そこで、この機能を拡張し「hidspix -ri」では、得られたFUSE情報を元に、以下のURLを開きます。このページを利用すれば、データシートが手元に無くとも(あったほうが良いですが)、FUSEの設定内容を確認し、変更内容を詳細に検討できます。

Engbedded AVR® Fuse Calculator — Version 0.5.0

Part name: **ATmega88**

	Low	High	Ext'd	
Default:	0x 62	0x DF	0x F9	Apply default values
Current:	0x FF	0x DD	0x F8	Apply user values

Quick Configuration

Ext. Crystal Osc.: Frequency 8.0~ MHz; Start-up time PWRDN/RESET: 16K CK/14 CK + 65 ms; [CKSEL=1111 SUT=11]

☐ Clock output on PORTB0; [CKOUT=0]

☐ Divide clock by 8 internally; [CKDIV8=0]

Brown-out detection level at VCC=2.7 V; [BODLEVEL=101]

☐ Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]

☐ Watch-dog Timer always on; [WDTON=0]

☒ Serial program downloading (SPI) enabled; [SPIEN=0]

☐ Debug Wire enable; [DWEN=0]

☐ Reset Disabled (Enable PC6 as i/o pin); [RSTDISBL=0]

☒ Boot Reset vector Enabled (default address=\$0000); [BOOTRST=0]

Boot Flash section size=1024 words Boot start address=\$0000; [BOOTSZ=00]; default value

	Low	High	Ext'd
<input type="checkbox"/> CKDIV8	<input type="checkbox"/> RSTDISBL		
<input type="checkbox"/> CKOUT	<input type="checkbox"/> DWEN		
<input type="checkbox"/> SUT1	<input checked="" type="checkbox"/> SPIEN		
<input type="checkbox"/> SUT0	<input type="checkbox"/> WDTON		
<input type="checkbox"/> CKSEL3	<input type="checkbox"/> EESAVE		
<input type="checkbox"/> CKSEL2	<input type="checkbox"/> BODLEVEL2	<input checked="" type="checkbox"/> BOOTSZ1	
<input type="checkbox"/> CKSEL1	<input checked="" type="checkbox"/> BODLEVEL1	<input checked="" type="checkbox"/> BOOTSZ0	
<input type="checkbox"/> CKSEL0	<input type="checkbox"/> BODLEVEL0	<input checked="" type="checkbox"/> BOOTRST	

Apply fuse bits

☐ = Unprogrammed (1)
☒ = Programmed (0)

(2) 「hidspc -rd」では、AVRマイコンのピン接続やデータシートのページを参照できます。このページでは、ピン接続やデータシートの入手、閲覧、類似機能を持つマイコンとの比較などが可能です。

AVR Freaks: AVR Freaks - Mozilla Firefox

http://www.avrfreaks.net/index.php?module=Freaks_Devices&func=displayDev&objectid=77

mySmartControl controller board with ATmega8 or ATmega168

DEVICES >>> AVR MICROCONTROLLERS

compare devices

atmega88

Availability: In production

Summary

The mega88 is a 8k variant of the mega48, which was allegedly designed to fill the need for an 1.8V mega8-like part. Its siblings are the mega48 and mega168; 4- and 16k flash variants of the same. The ATmega48, ATmega88 and ATmega168 differ only in memory sizes, boot loader support, and interrupt vector sizes. In ATmega88 and ATmega168, the Boot Loader Support provides a real Read-While-Write Self-Programming mechanism. NB! The PDIP-28 version has only 6 ADC channels, while the other packages has 8 channels.

Crossrelation

- pin and feature compatible with mega48, more memory.

Overview features

Flash	EEPROM	SRAM	Speed	Volts
8kB	512B	1024B	0 ~ 20MHz	2.7 ~ 5.5V

Files

- DS: Datasheet: mega48/88/168 [DS31.26 KB] (Jun 29, 2005)
- DS: ATmega48/88/168 Automotive Preliminary Complete [DS34.30 KB] (Aug 20, 2005)

Packages

General package info

	32-pin TQFP	32-lead MLF	28-pin PDIP
Package Lead Code	32	22	28
Carrier Type	TRAY		TUBE
Max Package Height	1.2	1	

「--atmel-avr」や「--avr-devices」もお試ください。教育現場の実習では、実習用の多数のPCにAVR関連のブックマークが登録されていることは期待できません。しかし、この機能があれば容易にAVR関連情報に到達できます。

これら機能の追加に要したコードは二百行未満だと思いますが、かなり使い勝手が向上していると考えます。(段階的に機能を追加したので、何度も公開ファイルを差し替えましたが)

私なりに、コンパイルを行わなくとも各種の連携が可能ないように、利用者が考えてコマンドやファイルとの連携を可能にする仕組みを追加しました。

基本はブックマーク登録機能ですが、各種のアプリケーションも起動できます。
特に、付属の説明書はかなりの時間を費やして書いていますが、なかなか読んでもらえません。--Helpや--hidspixで説明書を開く機能を追加したので、読んでくれる方が増えるのを期待します。

↑

FUSEデータ確認方法(Linux, MacOSの場合) [†]

--riオプションにより、オープンすべきURLが表示されます。

```
hidspix -ri ... 「`」は逆クォート(日本語キーボードでは「Shift+@」)です
```

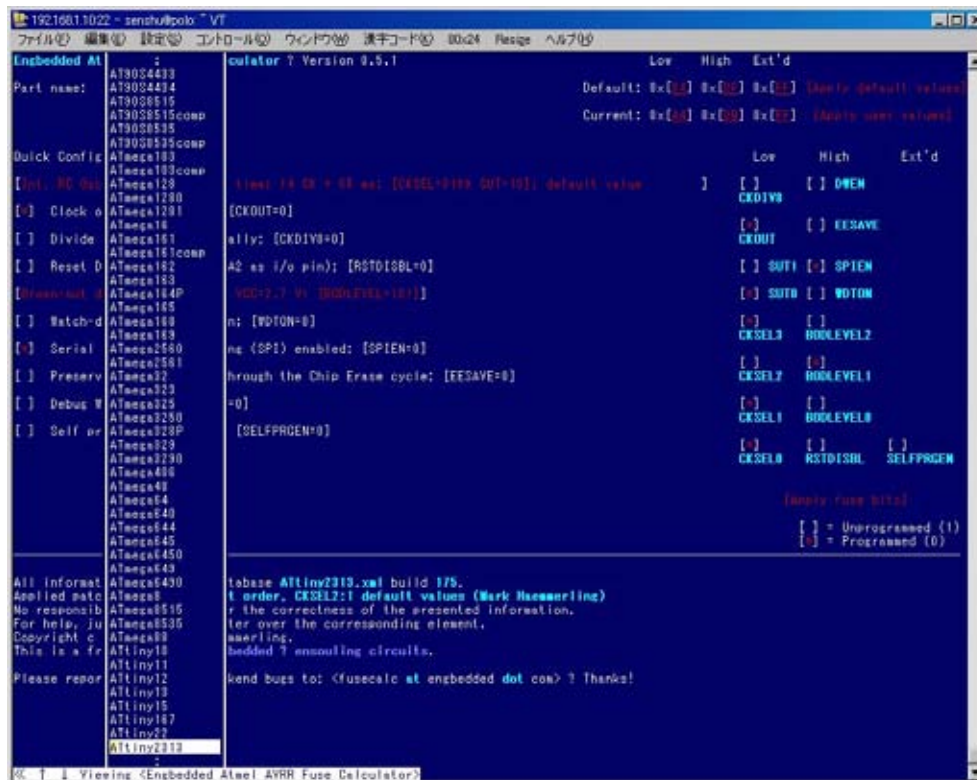
X環境で端末を使っている場合には、URL部分をクリックすれば、そのURLを開くことができます。マウス操作が面倒だったり、テキストブラウザを利用したい場合には、以下の方法を利用してください。

↑

w3mを利用する [†]

w3mをインストールしていない方は、インストール後に利用してください。
(w3mは、山形大学の伊藤さんの作です)

```
w3m `hidspix -ri` ... 「`」は逆クォート(日本語キーボードでは「Shift+@」)です
```



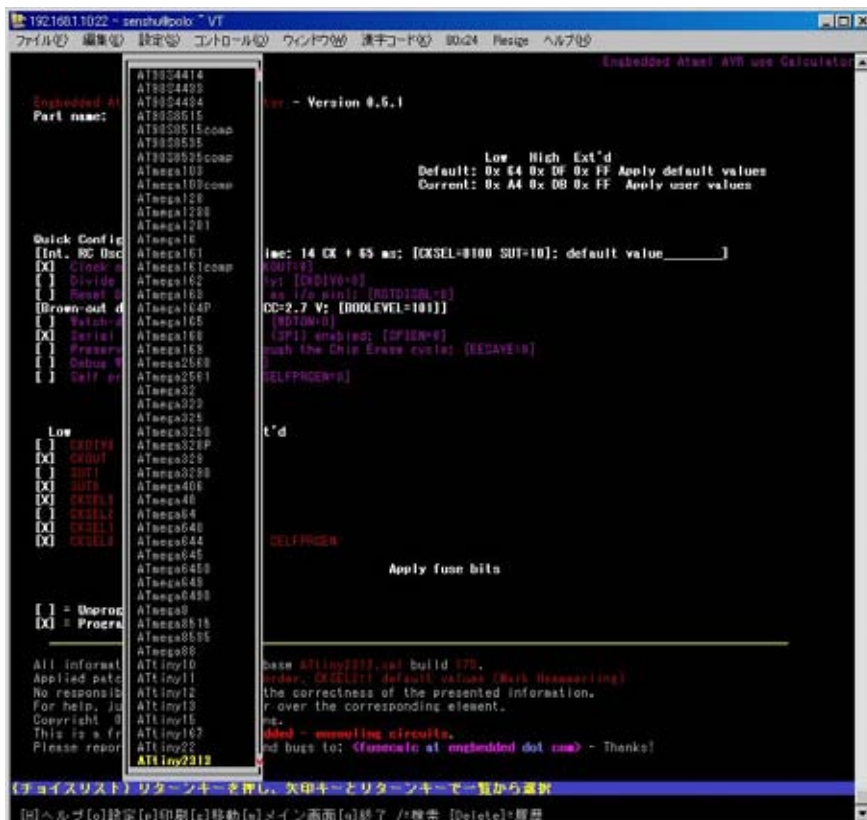
配色は各自の環境により異なります。横のサイズは情報量が多いので、120文字程度は必要です。

↑

lynxブラウザ [†]

長い歴史を持つlynxを使った例です。レイアウトはw3mに比べ見劣りがしますが、lynxの方が横の文字数は少なくとも操作が出来ます。

```
lynx `hidspix -ri`
```

文字ベースのWebブラウザ利用上の注意点 [↑]

慣れは必要ですが、どちらもTABとEnter、矢印キーで操作してください。qで終了できます。
(慣れれば、通常のWeb利用もこれでOKという方もいらっしゃると思います。lynxは数年間、Windowsの日本語版を私的にメンテしていました。)

GUIブラウザ [↑]

firefoxやkonquerorなどのGUIブラウザ使う場合も同様です。X-Window-systemを使っている場合に利用できます。リモートログインではこの方法は機能しませんので注意してください。

```
firefox `hidspc -ri`& ... (&のように、続けて&をつけて起動してください)
あるいは
konqueror `hidspc -ri`&
```

この後、firefoxが起動し、制御はそちらに移りますが、ターミナルに制御を戻し、Enterキーを押下すれば、操作を継続できます。

```
$ firefox `hidspc -rd`&
# hidspc -d4 -ph --new-mode -rd
[1] 6663
$ Detected device is ATtiny2313.    -> Enterキー押下

[1]+  Done                  firefox `hidspc -rd`
```

リモートログイン時にWebブラウザで開く [↑]

WindowsからTeraTerm経由でLinuxを使っている場合には、表示されたURLをマウスでダブルクリックすればWindows上の既定のブラウザが開きます。

時間計測機能について [↑]

Linux版では、当初は時間計測機能しませんでした。谷岡さんの協力により、以下のように

Windows版と同様の表示が可能になりました。

```
usl-5p:~/hidspc-2009-0126b/src# time ./hidspc -d1 /home/landisk/avrsttest.hex
# hidspc -d4 -ph --new-mode -d1 /home/landisk/avrsttest.hex
Detected device is ATtiny2313.
Erase Flash memory.
Flash memory...
Writing [#####] 1000, 0.64s
Verifying [#####] 1000, 0.36s
Passed.
Total read/write size = 2000 B / 1.37 s (1.43 kB/s)

real    0m1.382s
user    0m0.000s
sys     0m0.032s
```

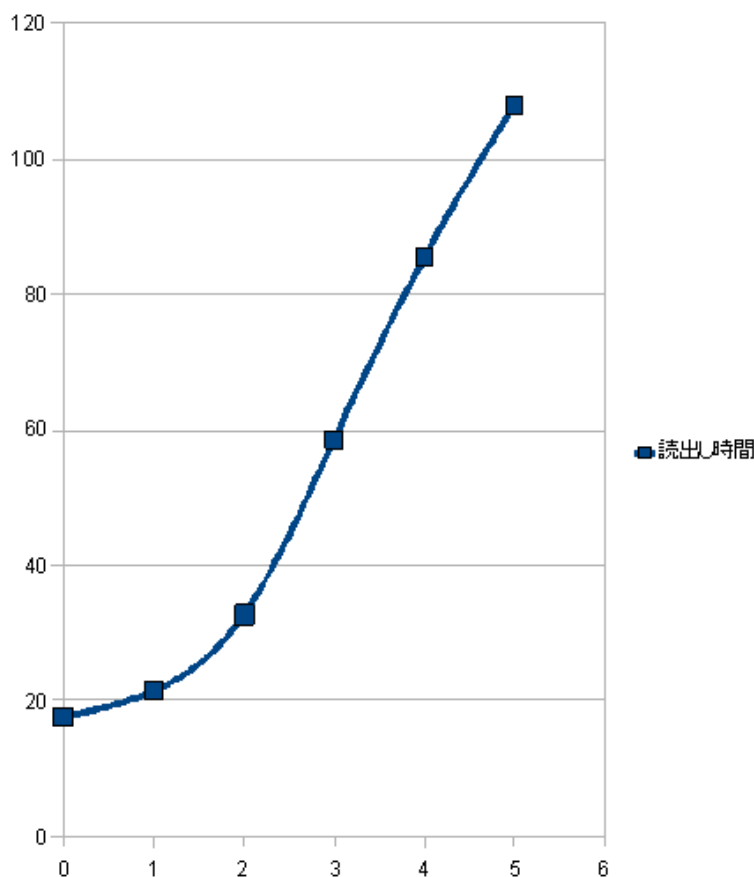
↑

「-dオプション」と読出し時間の関係 ↑

HIDaspX(AVRライター全般に共通です)では、-dの値によって大きく処理時間が変化します。

そこで、delay値と読み出し時間をグラフ(ディレイ値と時間[秒])を作成しました。

縦軸は、NECのチップによる増設ボードのUSBポートを使い、ATmega128の全メモリの読み出しに要する時間を表しています。



このグラフから、-d0と-d5では5倍もの違いがあることがわかります。hidspc.iniでは「-d4」を省略時の値にしています。これは、工場出荷時の値でもエラー無く処理できるということから、この値を設定しています。

書き込みと照合に必要な時間は、この時間の約2倍が必要と考えてください。

この特性を理解し、書き込み対象のマイコン速度に合った値を指定し、効率的な利用を行ってください。

指定の例(ATtiny2313の場合)

No	FUSE Low	-dの値	発振周波数	備考
0	----	-d0	18MHz以上	外部クリスタル/セラミック発振子
1	-fL0xe4	-d1	8MHz	14CK+65ms
2	-fL0xe2	-d2	4MHz	14CK+65ms
3	-fL0x64	-d4	1MHz	工場出荷値
4	-fL0x62	-d5	500kHz	14CK+65ms
5	-fL0xe6	-d17	128kHz	14CK+65ms
6	-fL0x66	-d120	16kHz	118, 119では不安定

↑

実行時間の計測方法 [↑]

現在公開中の**hidspix**には、実行時間を計測する機能があります。この機能は 2009年以降に追加した機能ですので、以前の版を利用している方は、次の項の説明を 参考になしてください。(特に理由がなければ、アップデートをお薦めします)

■ 2009年以降の版での動作例

```
G:\hidspix-2009-0125b\bin>hidspix.exe -d1 avrsttest.hex
Detected device is ATtiny2313.
Erase Flash memory.
Flash memory...
Writing [#####] 1000, 0.45s
Verifying [#####] 1000, 0.36s
Passed.
Total read/write size = 2000 B / 1.11 s (1.76 kB/s)
```

HIDaspixの動作報告では、実行時間の報告をお願いしています。

時計があれば計ることができますが、短い時間を正確に計測することはできません。

そこで、作成したプログラムの性能評価のためのストップウォッチ的なソフトウェアが 必要になります。

私は、動作時間の計測には、MS社の提供する無償ツール**timeit**コマンドを使い、1/1000秒の分解能(実際には1/100秒程度?)で計測を行っています。

以下が、timeitコマンドで、ATtiny2313の全メモリを書き出し・照合に要する時間を 計測した結果です。8MHzで動作している ATtiny2313の2kBの書き込み・照合を1.03秒で 完了していることがわかります。手動計測では、人間の反応時間を計っているようなものであり、この種のツールを利用しなければ計測は不可能です。

```
>timeit hidspix -d1 2313.hex
~~~~~
Detected device is ATtiny2313.
Erase Flash memory.
Verify Flash: 2048/2048 B

Passed.

Version Number: Windows NT 5.0 (Build 2195)
Exit Time: 10:40 am, Thursday, October 30 2008
Elapsed Time: 0:00:01.031
Process Time: 0:00:00.265
System Calls: 296309
Context Switches: 1945
Page Faults: 616
Bytes Read: 10106
Bytes Written: 4876
Bytes Other: 8326
```

詳細は、[@ITの紹介記事](#)を 参考になしてください。

コマンドプロンプトから利用しますが、計測の対象はWindows上で動作する全アプリケーション です。実に便利で、標準でOSに含めて欲しいツールです。各種のスイッチを持っていますが、非常にUNIXの香りのするツールです。

```
Usage: TIMEIT [-f filename] [-a] [-c] [-i] [-d] [-s] [-t] [-k keyname | -r keyname]
[-m mask] [commandline...]
where:
  -f specifies the name of the database file where TIMEIT
      keeps a history of previous timings. Default is .%timeit.dat
  -k specifies the keyname to use for this timing run
  -r specifies the keyname to remove from the database. If
      keyname is followed by a comma and a number then it will
      remove the slowest (positive number) or fastest (negative)
      times for that keyname.
  -a specifies that timeit should display average of all timings
      for the specified key.
  -i specifies to ignore non-zero return codes from program
  -d specifies to show detail for average
  -s specifies to suppress system wide counters
  -t specifies to tabular output
  -c specifies to force a resort of the data base
  -m specifies the processor affinity mask
```

実行結果は、STDERRに出力されますので、

```
(timeit 計測するコマンド) 2>&1 > 結果
      ^^^^
```

と指定すれば、ファイルに書き出すことが出来ます。○で括ることで、コマンドの
範囲を明確にします。このコマンドは、過去に実行したコマンドの統計情報を出力する
こともできる強力なツールです。

お名前:

コメントの挿入

siteDev extends PukiWiki 1.4.4 Copyright © 2001-2004 PukiWiki Developers Team. License is GPL.
Based on "PukiWiki" 1.3 by yu-ji customized by php spot.

