



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/HIDasp

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

Counter: 9677, today: 14, yesterday: 64

- [===== はじめに =====](#)
 - [HIDaspとは\(2008年9月\)](#)
- [HIDaspリンク集 ---\[関連ページへの移動にご利用ください\]](#)
- [HIDasp\(USB接続のAVRライタ\)の紹介](#)
 - [HIDaspの回路図と製作例](#)
- [最新のHIDasp用アーカイブ ----- \[Download\]](#)
 - [開発に協力していただける方へ\(開発環境の補足\)](#)
 - [関連プロジェクト\(ATmega88/168用\)](#)
- [ファームウェア\(ATtiny2313\)のFUSE設定](#)
- [hidsp, hidmonなどのコマンドについて](#)
- [動作確認済みのAVRリストと動作速度の目安](#)
- [===== HIDaspに関連するTips =====](#)
 - [hidspの使い方](#)
 - [開発環境との統合\(AVR studio, BASCOM-AVR\)](#)
 - [AVRマイコンのFUSEデータ確認方法](#)
 - [「-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](#)
 - [USB-HUBの利用で高速化](#)
- [HIDaspとの出会い\(コラム\)](#)

===== はじめに ===== ↑

HIDaspに興味を持った方でも、このページに含まれる技術的な話の多さに驚かれる方が多いようです。その為、質問や感想を書くページを分離しました。技術的な内容やQ&Aは別のページ[AVR/HID_QA00](#)に移しましたので、詳細はそちらを参照してください。

また、「概要を知りたい方」は、簡潔に解説されている [kumanさんのページ](#) をご覧ください。

↑

HIDaspとは(2008年9月) ↑

(2008-09-02 (火) 09:14:33)

どんな組み込みマイコンの開発を行う場合でも、特殊なドライバや開発用ソフトウェアのインストールなどが必要です。しかし、これが許可されない環境(特に教育現場)も少なからずあることに気付きました。特に、教育現場では、「ドライバやソフトウェアのインストールは禁止」という環境は珍しくはありません。

一方、現在ではWinAVRなどの開発環境はCD-ROMやUSBメモリに格納しても利用可能ですから、ドライバ不要のAVRライタがあれば、インストール不要のAVRマイコン開発環境を実現できます。

ドライバのインストールが出来ない環境でもマイコン関連の学習を可能にする為、HIDaspやHIDspなどのライタやAVRUSBのプロジェクトを参考に、[irukaさん](#)の協力を得て、HIDaspをベースに独自に安定化と高速化などの改良したのが**HIDasp**であり、これはハードウェアとファームウェア一式を意味します。

HIDaspXをAVRライターとして利用するには、PC用プログラムが必要になります。これには私が保守しているavrspXを元に、HIDaspXをサポート機能を追加した**hidspX** コマンドを開発しました。なお、HIDaspX対応のコードには、瓶詰堂さんの作成されたコード(hidasp.cの改造版)を含んでいます。

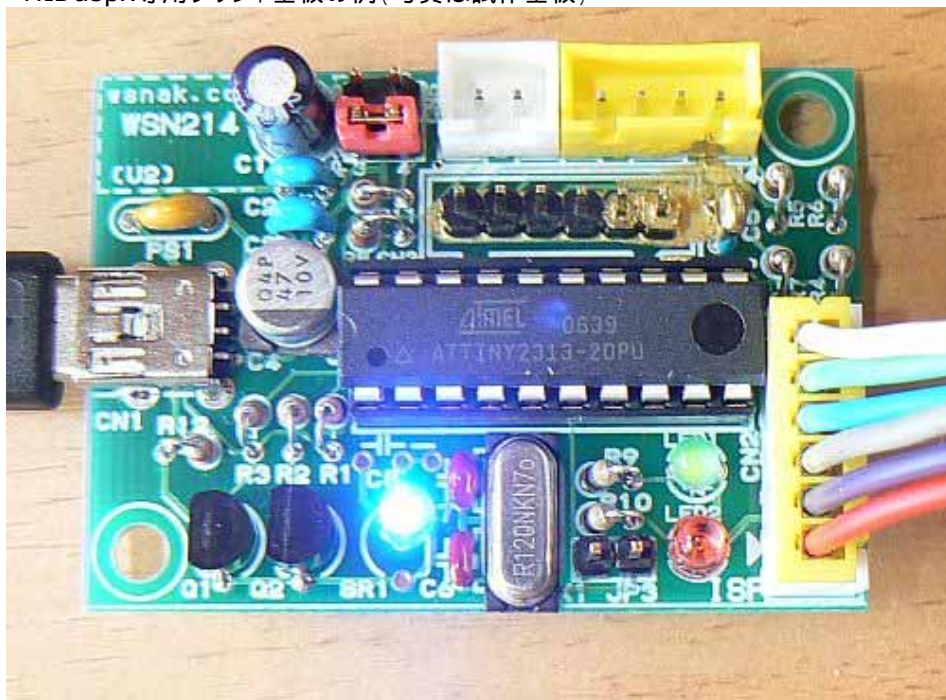
2008年10月以降、[hidmon](#)が 利用可能です。hidmonコマンドにてHIDaspXの各種I/Oを操作でき、USB接続のIOとして 利用できます。また、ATtiny2313のI/Oや各種のレジスタを直接制御できるため、マイコン学習教材としても有用です。

以下が、私が製作したHIDaspXです。+3.3Vの安定化回路を実装し、この「ライター」だけで、3.3V(乾電池2本分)の回路実験が可能です。COMポートが付いていないノートパソコンによるマイコン開発用に小型ケースに組み込んでみました。同様の目的には、USBaspも使えますが、USBaspではドライバのインストールが必要です。HIDaspXはドライバのインストールが不要という大きな特徴があります。

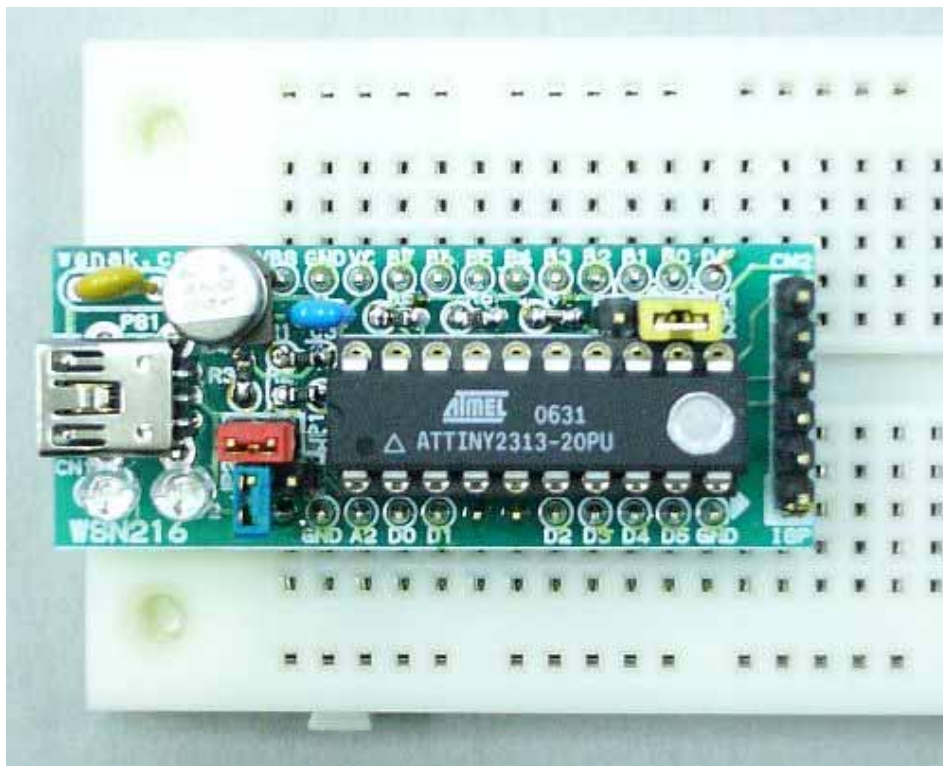
○ユニバーサル基板による製作例



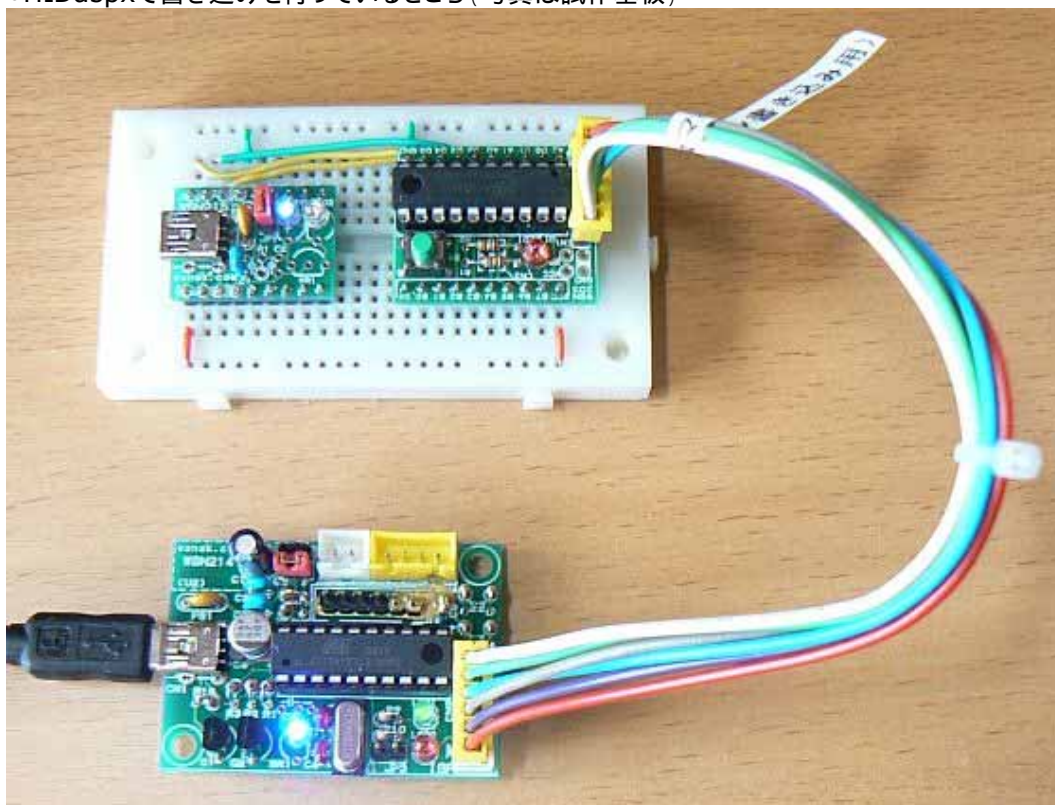
○HIDaspX専用プリント基板の例(写真は試作基板)



○USB-IO用ブレッドボード対応モジュール(写真は試作基板)



○HIDaspxで書き込みを行っているところ (写真は試作基板)



■ HIDaspxの特徴

- どのUSBコネクタに繋いでも動作する (ドライバのインストールも不要)
 - COMポートの無いPCでも利用可能 (今はこれが一般的です)
 - Windows 2000/XP/Vistaで動作を確認 (Windows95/98では動作しません)
 - MacOSやLinuxへの移植版も登場 (まだ動作確認が出来た段階ですが)
- 使い勝手に優れた **hidspcx**(**avrspcx**の別名) コマンドが利用できる
 - インストール不要 (CD-ROMやUSBメモリからの実行も可能)
 - ターゲットのAVRマイコンを自動認識 (新・旧のAVRマイコンをサポート)

- ごく一部の旧タイプを除き、CHIP名を指定する必要なし
 - BASCOM-AVRなどの開発環境にも組み込みできます。
 - hidspixのショートカットに、HEXファイルをD&Dすることでも書き込めます
- ISPコネクタを外すことなく開発が可能(ターゲット実行時の影響は最小限)
- 電源の投入順序も気にする必要なし(ライタのリセット時は自動で再接続)
- 製作コストは最小限(安価(秋月電子では100円!)なATtiny2313のみで実現)
 - 材料代は500円程度?(専用プリント基板を計画中)
- 小型に製作可能(USBメモリの形状で製作も可能)
- 十分な速度で動作(USBaspとほぼ同じ)
 - HUBを利用すると更に動作速度が向上(USBaspでは速度は低下する)
- USB-IOとして利用できる(hidmonを利用)
 - コマンドによる対話方式で内蔵レジスタを操作可能
 - スクリプトによる一括実行も可能
 - DLLを使って、各種のプログラム言語(C言語、Visual BASICなど)から操作可能
 - **ruby**を使っての制御が可能になりました(2009年1月2日)
 - 詳細は[AVR/HIDmon03](#)をご覧ください

HIDaspとHIDAspxは同一のハードウェアですが、ソフトウェアの互換性はありませんので、混同しないようにしてください。なお、HIDAspxは、**エイチ・アイ・ディー アスベックス**とお読みください。



HIDAspxリンク集 ---【関連ページへの移動にご利用ください】[↑]

- HIDAspxに関する情報 [タイトル一覧](#) [AVR/HIDAspx00](#)
- HIDAspx [更新履歴](#) [AVR/HIDAspx_news](#)
- HIDAspxに関するQ&A [質問はこちら](#) [AVR/HID_QA00](#)
- HIDAspxの製作レポート [利用者の声](#) [AVR/HID_reports](#)
- HIDAspxによる教材開発 [AVR/HID_kyouzai](#)
- AVRライタに関するFAQ集 [AVR/writer_FAQ](#)
- 掲示板にも関連情報があります [AVR/news_contents_all](#)
- 「**AVR-USB**」(このソフトがHIDAspxを支えています)
 - <http://www.obdev.at/products/avrusb/index.html>
- 2008-10-29時点のこのページの内容 [HIDAspx-1029.pdf](#)
- HIDAspxをUSB-IOとして利用する例
 - HIDmonについて [AVR/HIDmon00](#)
- HIDAspxのハードウェアをusb-RS232のブリッジとして利用 [AVR/usbRS232](#)

ページの簡素化を図りました。以前のページの内容は↑のPDFをご覧ください。

関連するページへのリンク

AVR開発ツールのリンク(外部)

- [ドライバをインストールする例\(マイコン講座/環境の構築\)](#)
 - avrspの使い方は、hidspixにもそのまま利用可能
- ポータブルWinAVR <http://www.chip45.com/index.pl?page=PortableWinAVR&lang=en>
- 瓶詰堂さん(HIDaspの開発元) http://www.binzume.net/library/avr_hidasp.html
 - このページにもリンクしていただきました
- irukaさんのサイト(瓶詰堂さんもお薦め)
 - <http://hp.vector.co.jp/authors/VA000177/html/A3C8A3C9A3C4A3E1A3F3A3F0.html>
- kumanさんの回路図を含む、実践レポート(一読をお勧めします)
 - <http://www.geocities.jp/kuman2600/n6programmer.html#13> (10/12追記あり)

- hidsp-1012b.zipの感想が書かれています
 - [HIDasp kuman流の使い方](#) (10/19追加)
 - 「工研Wiki」...電通大の学生さんによるHIDaspの解説
 - <http://delegate.uec.ac.jp:8081/club/koken/wiki/?AVR%2FHIDasp>
 - AVRライターで有名なTADさんのページにもHIDaspが登場
 - <http://homepage2.nifty.com/denshiken/AVW021.html>
 - 「Fight with life」(RAINさんのBlog)..HIDasp(x)を4台も作成されています
 -  <http://amenotiyukizora.blog76.fc2.com/>
 - HIDaspの使い方(マウス操作での利用法)
 - [AVR/HIDasp/使い方](#)
 - 中学生向けの学習教材にHIDaspを適用した例
 - <http://www.ne.jp/asahi/ja/asd/gijutu/HIDapio/>
-

↑

HIDasp(USB接続のAVRライター)の紹介 [↑]

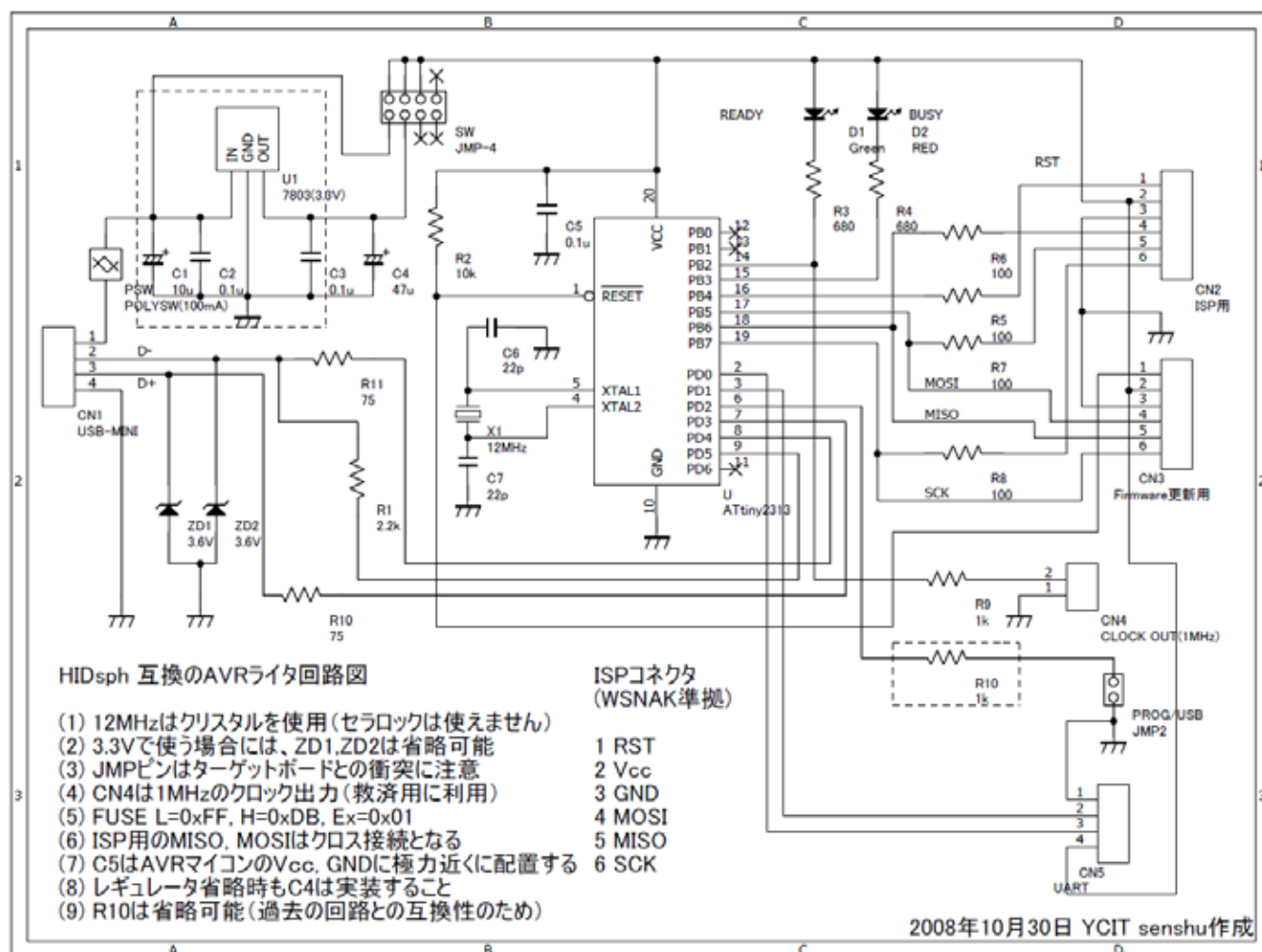
↑

HIDaspの回路図と製作例 [↑]

- HIDaspの回路例(HIDspと同じ回路です。ISPコネクタ配置はWSNAKボードに準拠)

クリックで拡大可能(後半で紹介しているHIDspと同じものです)

三端子レギュレータの入出力に接続している47uFのコンデンサは、USB規格としては 大きすぎるようです。発振止めとしてはこの程度が必要だと思いますが、規格内に 収めるなら、10uF程度に留める必要があります。(メーカーさんではどうやっているのでしょうか)



なお、UARTコネクタはusbRS232での利用を想定しています。AVRライタとして使う だけなら、このコネクタを実装する必要はありません。

水晶発振子は必須です

ある方から、HIDaspではセラロックで動作するのにHIDaspXはセラミック振動子では動作 しないとの報告がありました。HIDaspXでは高速化を実現する為にパケット長を32バイト に拡大しており、発振周波数の誤差に敏感です。回路図にも明記していますが、セラミック振動子(商品名:セラロックなど)では、動作しないと考えてください。

↑

最新のHIDaspX用アーカイブ ----- [Download] [↑]

ここで公開するものは、オリジナルのHIDaspではなく、高速化といくつかの安定化対策を施した**HIDaspX**に関するものです。HIDaspXは、HIDspH互換の回路で利用できます。

ぜひ**利用しての感想**をお寄せください。なお、これらは**評価用**に公開するものであり、予告無く変更することがあります。

- hidspXとHIDaspXファームウェア(ソースを含む) ... [hidspX-2009-0109.zip](#)
- hidmon(実行ファイル, ソースコード, スクリプト) ... [hidmon-1125.zip](#)
 - RC発振器でUSB-IOの利用が可能(RC発振モードは評価目的に公開)
- usbRS232(ファームウェア, ソースコード) ... [usbRS232-1109.zip](#)
 - HIDaspXのハードウェアでCDCクラスを実現した例(開発段階)

↑

開発に協力していただける方へ(開発環境の補足) [↑]

上記のFirmwareは、WinAVR-20060421をインストールした後に、AVR-Wikiから入手できる「機能追加版 2006-04-22」を上書きした環境で作成しています。

<http://avrwiki.jpn.ph/wiki.cgi?page=WinAVR%A5%D0%A5%B0%BE%F0%CA%F3>

ATtiny2313用のファームウェアを作成したい方は、このavr-gccを利用してください。
これ以外のコンパイラでは、ATtiny2313用のFLASHサイズ2048バイトに収めることは困難です。

```
2006/04/22 19:14      369,664 avr-ar.exe
2006/04/22 19:14      514,560 avr-as.exe
2006/04/22 21:23       91,136 avr-c++.exe
2006/04/22 19:14      399,360 avr-c++filt.exe
2006/04/22 21:23       90,624 avr-cpp.exe
2006/04/22 21:23       91,136 avr-g++.exe
2006/04/22 21:23      89,088 avr-gcc.exe
2006/04/22 21:23      26,112 avr-gcov.exe
```


バージョンの確認は、CMD窓にて、以下の方法で確認できます。

```
>avr-gcc -v
Reading specs from C:/WinAVR/lib/gcc/avr/3.4.6/specs
Configured with: ./configure --target=avr --prefix=/c/WinAVR
--enable-languages=c,c++ --with-dwarf2 --enable-win32-registry=WinAVR
--disable-nls --enable-c-mbchar
Thread model: single
gcc version 3.4.6
```

--disable-nls --enable-c-mbcharの表示があるのを確認ください。実行ファイル群の日付を確認するを忘れずに行ってください。

```
>which avr-gcc
```

でコンパイルに利用するコンパイラの実体(フルパス名)を確認できます。

- 開発者用のWinAVRアーカイブ(15MB)  [WinAVR-0604.7z](#)
 - ファームウェアをコンパイルしたい方だけ、ダウンロードしてください
 - 展開したファイル群をWinAVR20060421のディレクトリに上書きします
 - いつでも戻せるように、事前のバックアップを忘れずに行ってください


パソコン用の実行ファイルは、MinGWとBorlabd C++ でコンパイルしています。

また、今まで公開してきたアーカイブは、irukaさんのサイトから入手可能です。
(大量のファイルを保管していただき、感謝いたします。)

<http://psp.dip.jp/web/upload/filelist.cgi>

ここにあるアーカイブに不具合を発見した場合には、旧版もお試ください。

関連プロジェクト(ATmega88/168用) [↑]

- bootloader + hidmon88
 - bootmon(ファーム・実行ファイル, ソースコード) ...  [bootmon-1127.zip](#)
 - RC発振器でbootloadHIDの動作が可能(RC発振モードは評価目的に公開)
 - クリスタル発振子(12/20MHz)で動作確認済み

↑

↑

ファームウェア(ATtiny2313)のFUSE設定[†]

クリスタル発振器の場合(1028版以降の設定)

```

Low: 11111111 (0xFF)
    ||| |+++- CKSEL [3:0] システムクロック選択
    ||++- SUT [1:0] 起動時間
    |+- CKOUT (0:PD2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: 11-11011 (0xDB)
    ||| | | | +- RSTDISBL (RESETピン 1:有効, 0:無効(PA2))
    ||| |+++- BODLEVEL [2:0] (111:Off, 110:1.8, 101:2.7, 100:4.3)
    ||| +- WDTON (WDT 0:常時ON, 1:通常)
    ||+- SPIEN (1:ISP禁止, 0:ISP許可) ※Parallel時のみ
    |+- EESAVE (消去でEEPROMを 1:消去, 0:保持)
    +- DWEN (On-Chipデバッグ 1:無効, 0:有効)

Ext: -----1 (0xFF)
    +- SPMEN (SPM命令 1:無効, 0:有効)

```

1028版以降では、PD2端子は12MHzのクロック出力ではなく、AVR programmer/USB-IO の識別用に利用することになりました。

回路図とおりに製作している場合には、1kΩの抵抗があるので、従来のファームを 使っている場合でも悪影響はありません。この変更は、一つのハードウェアをAVR programmer/USB-IOとして共用する場合に重要な仕様変更です。オープンでProgrammer、GNDに接続時、USB-IOとして利用します。このモード変更は、HIDAspxモジュールがRESET時に一度だけ判断します。動作後にPINを変更してもモードは変わりません。hidspcで利用する時には、Programmerモード、hidmonから利用する時には、USB-IOモードに設定してください。特に、Programmerモードでは、PB2には1MHzのクロック信号が出力されます。この状態では、PB2をUSB-IOとしては使えませんが、ご注意ください。HIDAspxをAVR programmerとしてのみ使う場合には、従来(以下の設定)のままでも利用可能です。

PD2からの外部クロック出力を有効にする場合(従来の設定)

```

Low: 10111111 (0xBF)
    ||| |+++- CKSEL [3:0] システムクロック選択
    ||++- SUT [1:0] 起動時間
    |+- CKOUT (0:PD2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)

```



hidspc, hidmonなどのコマンドについて[†]

hidspc.exe, hidmon.exeなどのコマンドは、GUIツールから呼び出すことも可能ですが、**基本的にはコマンドプロンプト上で利用するツール**です。

一度はコマンドプロンプトでご利用いただき、CUIの便利さを堪能ください。

HIDAspxに関連する用語について、特徴や相互の関係を整理してみます。

(1) HIDAspx 解説ページ [AVR/HIDAspx](#)

1. HIDクラスを採用しATtiny2313一個で実現した「AVRライタ兼USB-IO」
(hidaspを参考にスタートしましたが、今は互換性はありません)
2. シンプルでローコストに製作できるハードウェア
3. HIDクラスの採用により、ドライバのインストールは不要
4. USB接続で高速に動作する(特にUSB-HUB経由の利用時)

(2) hidspcコマンド

1. HIDAspxをAVRライタとして使う場合に使用するプログラム

2. SETUPコマンドで、簡単に導入できる
3. AVRマイコンを自動認識し、旧タイプのAVRマイコンもサポート
4. BASCOM-AVRやAVRstudioとの統合利用も可能

(3) hidmonコマンド [AVR/HIDmon00](#)

1. HIDAspxをUSB-IOとして利用する時に使用する
2. HIDAspxのみでAVRマイコンの学習が可能
3. hidmon.dllを使えば、一般的な言語(C言語やVB, VBAなど)から制御可能

(4) bootmon [irukaさんによる解説](#)

1. シンプルでローコストに製作できるハードウェア
2. ATmega88/168用(12/16/20MHzのクリスタル発振子、あるいは内蔵RC発振器で動作)
3. ブートローダ&簡易AVRマイコン用モニターです(現時点ではAVRライタ機能はありません)
 - i. bootloadHID互換のbootloader機能
 - ii. AVRマイコンモニター(簡易デバッガ)

(5) bootloadHIDコマンド

1. bootmon用の書き込み用ツール
2. 任意の開発ツールで作成したHEXファイルを書き込み、実行できる

(6)hidmon88コマンド

1. AVRマイコン内のRAMやI/Oの読み書き、EEPROMやFLASHの読み出しが可能
2. 対話的(一括実行も可能)な操作が可能
3. シンボリック(レジスタ名は名前で指定)に操作可能

bootmonの名前が示すとおり、現時点ではAVRライタ機能はありません。
HIDAspxの完成度が高いので、AVRライタとしてはHIDAspxを使ってください。
HIDAspxが複数台あっても、HIDAspxは多機能で汎用性があり無駄にはなりません。
シリアル情報を設定すれば、一台のPCに複数台のHIDAspxを接続して利用できます。

このように、AVRマイコンを使って学習を考えている方には魅力的な仕様です。
どちらも、irukaさんと共同で開発を行ない、多くの方からの助言を元に、現在に至っています。

(irukaさんの作成されたbootmonに対し、シリアル情報のサポート、RC発振対応やmega168対応、各周波数のHEXファイルの一括生成やsetup機能を追加しました)

なお、bootmonではRC発振モードを利用できますが、hidmon88用のコードが入りません。12や20MHzのクリスタルでの利用をお勧めします。

↑

動作確認済みのAVRリストと動作速度の目安 [†]

MCU名	Device Signature	FLASH	page	EEPROM
AT90S2313	1E-91-01	2048		128
ATtiny2313	1E-91-0A	2048	32 x 64	128
ATtiny26L	1E-91-09	2048	32 x 64	128
ATtiny45	1E-93-06	4096	64 x 64	256
ATtiny85	1E-93-0B	8192	64 x 128	512
AT90S4433	1E-92-03	4096		256
AT90S8515	1E-93-01	8192		512
ATmega8515	1E-93-06	8192	64 x 128	512
ATmega48	1E-92-05	4096	64 x 64	256

ATmega8	1E-93-07	8192	64 x 128	512
ATmega88	1E-93-0A	8192	64 x 128	512
ATmega168	1E-94-06	16384	128 x 128	512
ATmega64	1E-96-02	65536	256 x 256	2048
ATmega644P	1E-96-0A	65536	256 x 256	2048
ATmega128	1E-97-02	131072	256 x 512	4096

これらのAVRマイコンは、8MHz以上のクロックで動作していれば -d1で Read/Write可能です。その時の読み取りは 4kB/秒程度です。書込み速度は ベリファイが必要なため、この1/2程度になります。

この時間は、hidmon benchテストで15kB/秒の場合です。

HIDaspXの利用者から、「ATmega8, ATmega64, ATtiny45でも使える」との 報告がありました。1014a版を使い、ATtiny2313で2kBの全領域をWrite/Verify時間は、8MHzの場合(-d1)で約1秒、1MHzの場合(-d4)で約3秒です。

```
>timeit hidspX -d0 2kB.hex
Detected device is ATtiny2313.
Erase Flash memory.
Write Flash: 2048/2048 B
Verify Flash: 2048/2048 B
Passed.

Elapsed Time:      0:00:00.953
```

↑

===== HIDaspXに関連するTips =====[↑]

↑

hidspXの使い方[↑]

1. HIDaspXライターとターゲットボードを6Pケーブルで接続します。
 - i. 電源はターゲットボードから供給が基本です。
- コマンドプロンプトを起動し、CDコマンドでHEXファイルのあるディレクトリに移動します。
 - [そのフォルダ内で直接コンソールを開く方法](#)を利用すると便利です。

コマン ド	オプション	機能	備考
hidspX	無し	オプション書式とライタの種類が表示	(-?で詳細表示)
hidspX	-pu?	USBのサーチ結果を表示	DLLバージョンも同時に表示
hidspX	-pcN	com writerを指定	Nは接続COMポート番号
hidspX	-pbN	com spi Bridgeを指定	Nは接続COMポート番号
hidspX	-phN	HIDaspXを指定	シリアル番号の指定も可能
hidspX	-?	サポートデバイス一覧を表示	
hidspX	-r	Device Read(デバイス名とデバイス仕様を 表示)	
hidspX	-rf	Read Fuse	
hidspX	-e	Erase	
hidspX	XXX.hex	Flash Write(複数ファイル指定可能)	

hidspX	XXX.eep	EEPROM Write(Flashとの同時指定も可能)	
hidspX	-v- XXX.hex	Verify無しのWrite(複数ファイル指定可能)	
hidspX	-v XXX.hex	Verify(複数ファイル指定可能)	
hidspX	-rp	Read Program	>出力ファイル名でファイルに出力可能
hidspX	-fL<hex>	Fuse Lowの設定	(FUSE CALCが便利)
hidspX	-fH<hex>	Fuse Highの設定	(FUSE CALCが便利)
hidspX	-fX<hex>	Fuse eXtendの設定	(FUSE CALCが便利)
hidspX	-l<hex>	メモリロックビットの設定	
hidspX	-dNN	Delay設定	
hidspX	-w	Wait	処理完了後、キー入力を待つ
hidspX	-wN	N秒Wait	処理完了後、N(2以上)秒間、動作を停止する

<hex>は、0x1b のように書くことができます。00011101のように 2進数で書くこともできます。詳細は、hidspXに同梱のavrx-tool.txtをご覧ください。

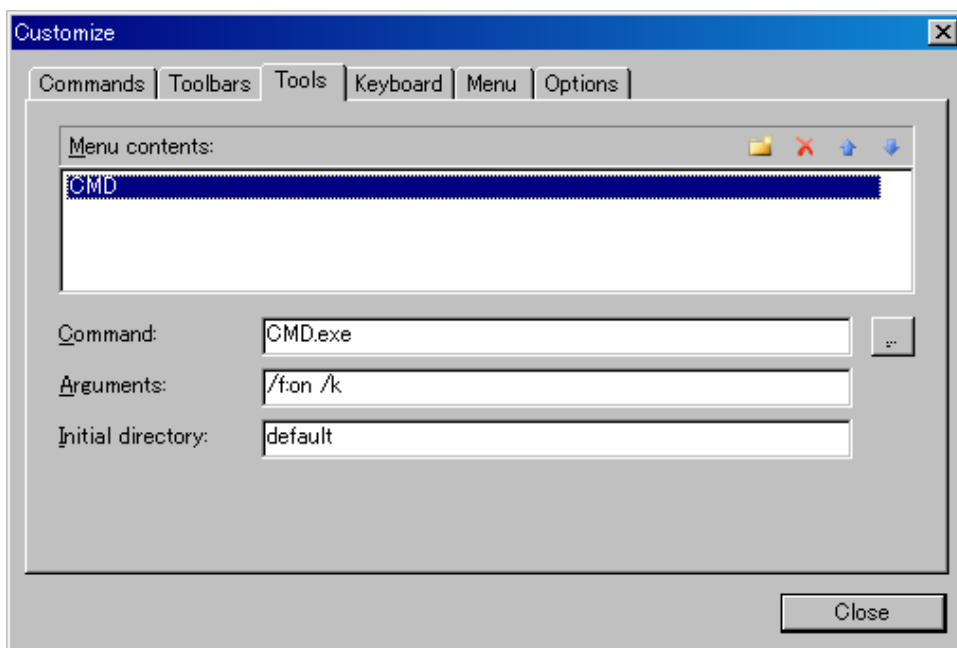


開発環境との統合 (AVR studio, BASCOM-AVR) [↑]

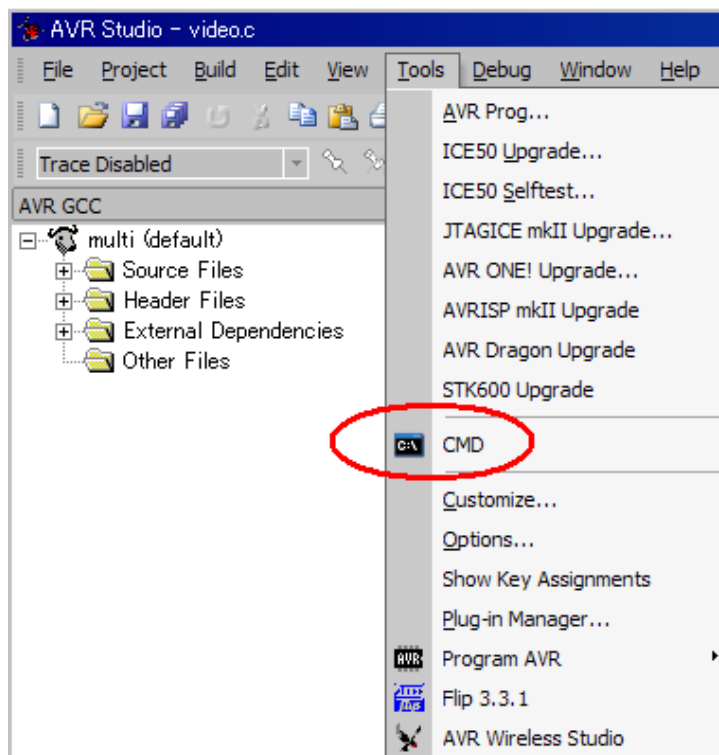
各種の開発環境から、HIDaspXを利用する事ができます。

例1: Atmel社の開発環境(AVR studio)から利用する

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。

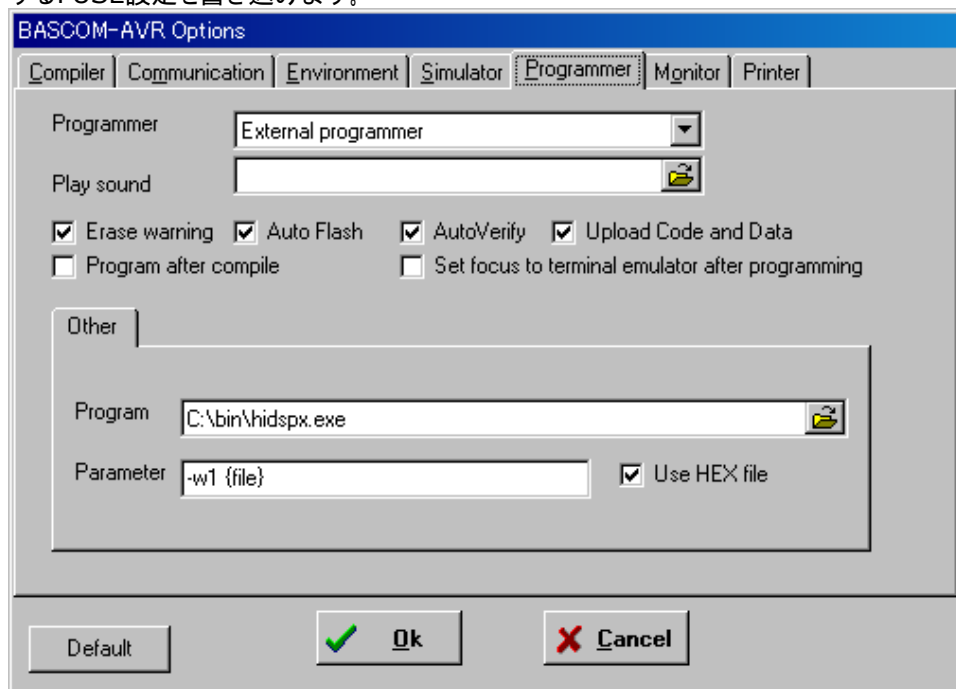


CMDが表示されれば、その中で**hidspx *.hex**を入力し、書き込むことができます。
Commandに **hidspx.exe**、Argumentsの部分に、***.hex**を登録することも不可能ではありませんが、FUSE設定や「-d」指定ができないため、お奨めはできません。

例2: BASCOM-AVRから利用する

BASCOM-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。

BASCOM-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) **hidspx.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、**hidspx**で書き込みできます。
なおこの場合、BASCOM-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 **hidspx**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、書き込み完了後に 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、書き込み結果を確認できます。

↑

AVRマイコンのFUSEデータ確認方法 [↑]

AVRマイコンでは、複雑なFUSE設定が可能なため、マニュアルとの格闘が必要になります。

そこで、hidspixではfuse.txtをhidspix.exeと同一の場所にコピーしておくことにより「-rf」(Read Fuse)で、現在設定された値が人間にもわかる形式で表示できます。

```
> hidspix -rf<Enter>
Low: 11100100
    |||+---- CKSEL[3:0] システムクロック選択
    ||+---- SUT[1:0] 起動時間
    |+---- CKOUT (0:PD2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:11-11111
    |||||+---- RSTDISBL (RESETピン 1:有効, 0:無効(PA2))
    |||+---- BODLEVEL[2:0] (111:0ff, 110:1.8, 101:2.7, 100:4.3)
    ||+---- WDTON (WDT 0:常時ON, 1:通常)
    |+---- SPIEN (1:ISP禁止, 0:ISP許可) ※Parallel時のみ
    +---- EESAVE (消去でEEPROMを 1:消去, 0:保持)
    +---- DWEN (On-Chipデバッグ 1:無効, 0:有効)

Ext: -----1
    +---- SPMEN (SPM命令 1:無効, 0:有効)

Cal: 84 85
```

FUSE設定を誤ると、AVRマイコンを書き込み不能にすることもあります。詳しくは、**hidspix付属の説明書(特にavrx-tool.txt)**を熟読の上、FUSE設定を行ってください。(実例を示していますので、それほど難しくはありません)

しかし、どういう設定を行えば希望する動作が出来るかは不明です。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。

その場合、以下のサイトが用意しているツールを利用すれば、比較的容易にFUSEデータを得る(または設定の意味の確認)ことができます。

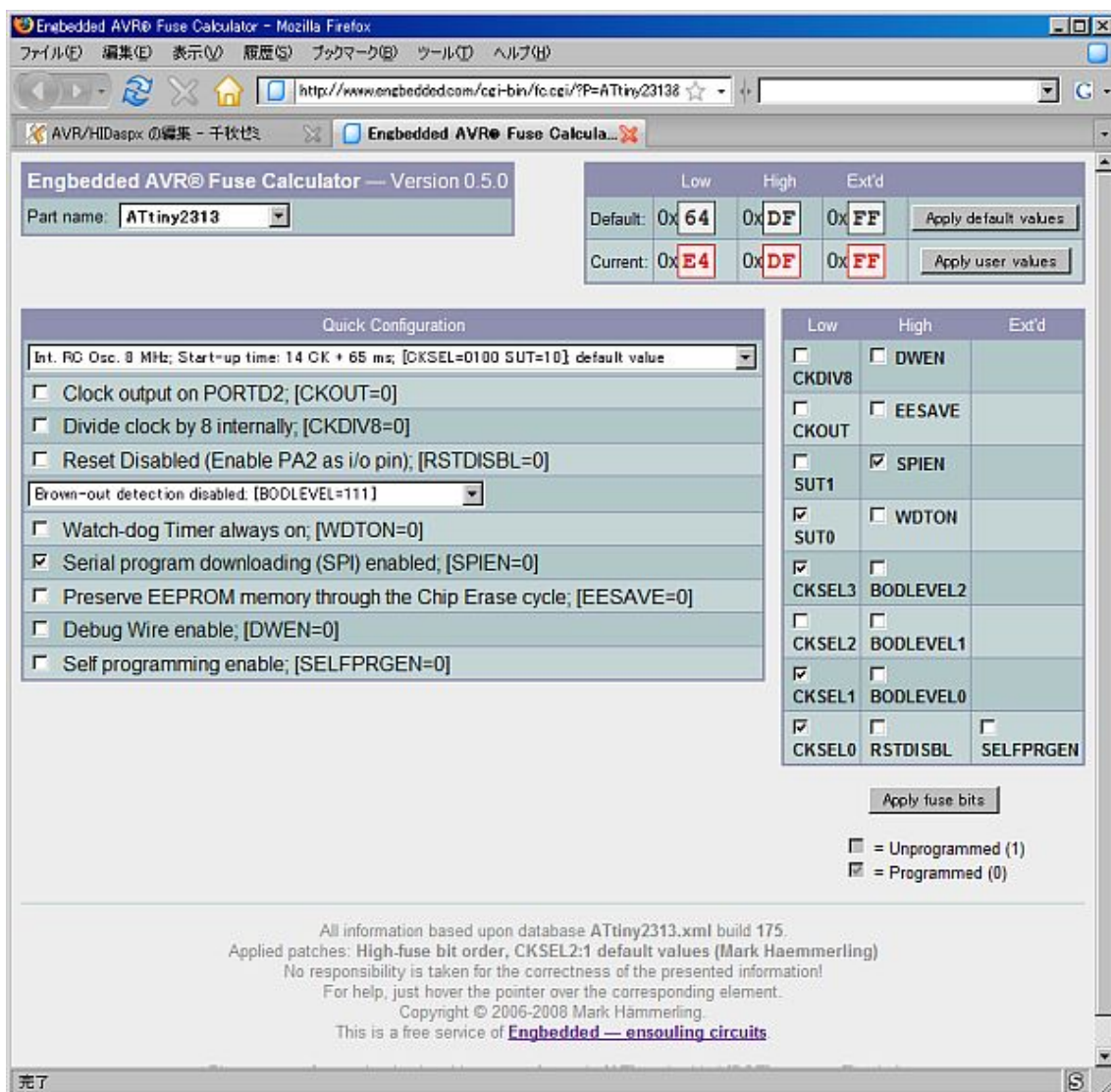
AVRマイコンのFUSE情報が確認できるサイト <http://www.engbedded.com/cgi-bin/fc.cgi/>

私が利用するAVRは全てサポートされているので、非常に重宝しています。
以下の機能があります。

1. プルダウンメニューから設定してFUSEを得る
2. FUSEの値から設定値を確認する
3. 工場出荷時の値の表示

2009年1月のhidspixに追加された機能

2009年1月10日以降のhidspixでは、-riオプションを指定で、「FUSE情報URLで設定内容を確認」できます。以下の図は、電子オルゴールのファームウェアを「hidspix -ri」と指定した時の表示です。

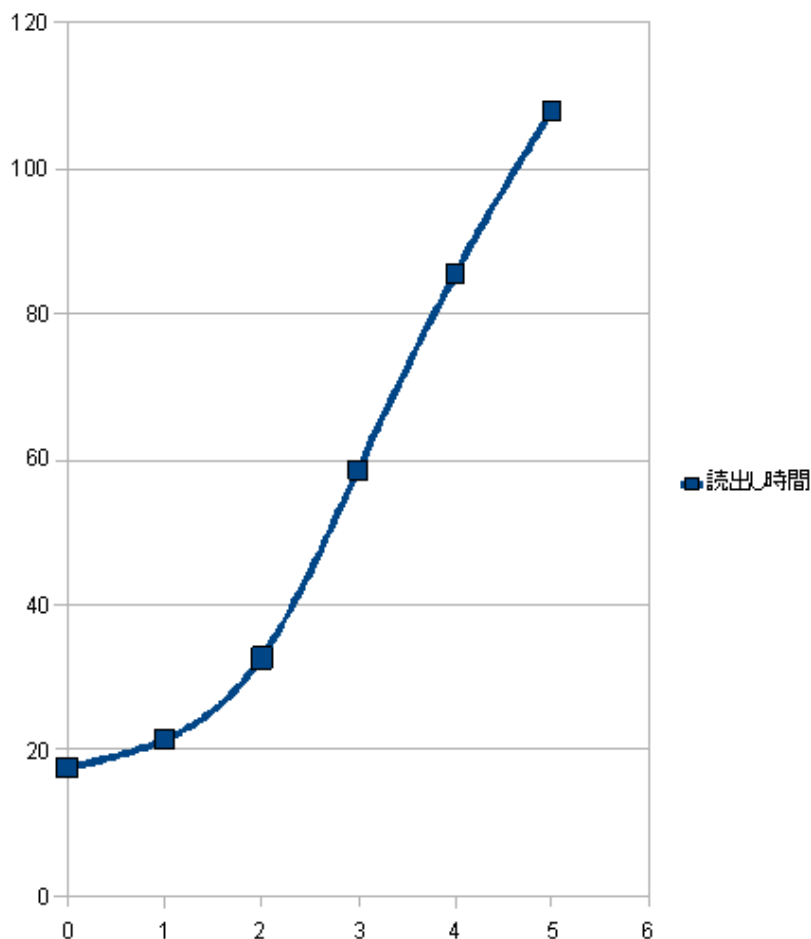


表示された内容を編集することも可能ですので、AVRマイコンのヒューズ内容の確認や設定が容易になりました。ぜひ、ご活用ください。

↑

「-dオプション」と読み出し時間の関係 ↑

HIDaspx (AVRライタ全般に共通です) 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 [†]

動作報告には、実行時間の報告をお願いしています。しかし、実行時間は どうすれば計測できるのでしょうか。簡単には時計があれば計測ことができますが、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 hidspx -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。()で括弧することで、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。



USB-HUBの利用で高速化 [†]

HIDaspixの転送速度を向上させるには、OHCIのUSB I/Fを使うほかに、UHCIでもUSB 2.0 規格のHUBを介すること

で、OHCI以上の速度が得られ、格段に使用感は向上します。

HUB経由の利用を行っていない方は、hidmonのbenchコマンドで転送速度を検証してください。このテストが正常であれば、HIDAspxはHUB経由で利用可能です。

関連情報 [hidmonの紹介](#)

```
>hidmon
~~~~~

■ UHCIのポートに接続
AVR> bench
hid write start
hid write end 38000 10953 s 3469 byte/s
AVR> q
Bye.

■ UHCIのポートからHUB経由で接続
AVR> bench
hid write start
hid write end 38000 2250 s 16888 byte/s
AVR> q
Bye.
```

このように4～5倍程度の違いが生じます。17kB/sの転送速度は、RS232C(115.2kbps)の速度を上回るものです。

USB 2.0規格のHUB経由で利用すれば、UHCI規格のPCでも転送速度が上がるため、HIDAspx全体の処理速度が向上し、結果としてUSBAspxに迫る性能が得られます。

↑

HIDaspとの出会い(コラム) ↑

先の問題を改善する為、ネット上で公開されている[瓶詰堂さん](#)開発の「HIDを使ったAVRライタ(HIDasp)」を試作してみました。

HIDaspは、ATtiny2313のみで構成され、安価に製作でき、ドライバのインストールが不要です。非常に魅力的な仕様を持っていますが、製作して使ってみると、

- Windows 2000ではエラーになり使えない(Windows Vistaでは動作しない)
- かなりの頻度でISP移行時にエラー(AVRマイコンを認識できない)になる
- HIDaspの電源ON直後の通信でエラーになることが多い
- 類似の構成のUSBAspxに比べ、処理に時間がかかる
- ISP用の信号がHi-Zになっておらず、回路構成によって不具合が生じる

などの問題を確認しました。改善したいと細かな修正を試みましたが、なかなかうまく行きません。

追記

この件を瓶詰堂さんに報告したところ、HIDasp(≠HIDAspx)の修正版を公開していただきました。以下の問題点が改善されました(私は動作を確認しておりません)。

- Windows 2000で使えない問題
- 電源ON時にリセット状態