2年前、何故 AT90S2313に プログラムを 書き込めなかったのか。？

古い話で恐縮ですが、2年前に 067の動画にて AT90S2313に プログラムを書き込む事が出来ませんでした。 同じ内容のプログラムを ATMEL STUDIO 7 で ターゲットCPUを ATMEGA328に変更して ビルドして、ATMEGA328に 書き込んだら無事、書き込むことが出来て、プログラムも所定の動きをしてくれました。

で、何故 AT90S2313には 書き込む事が出来なくて、ATMEGA328には 書き込みが 出来たのか。 という事に関して 2年前 私は よろしく無い事を 1ヶ所 していたのでした。

古いCPUなので、昔の ATMELの開発環境を入手したかったのですが、手に入らず 2年前は ATMEL STUDIO 7を パソコンにインストールしました。比較的新しい開発環境です。 この開発環境で 新しいプロジェクトを 生成する時、最初に ターゲットで使用する CPUの 型式を リストから選択しないといけないのですが AT90S2313 は リストに無かったのです。 メーカーの新しい開発環境なので、もう古いディスコンの CPUには、対応しないという事だったのでしょう。で、AT90S2313の 代わりに ATtiny2313 を 選択してビルドしたのでした。 これが 私が やってしまったよろしく無い事だったのです。後で、分かったのですが やはり AT90S2313 と ATtiny2313 は CPUの チップ IDが 異なるようで、ビルドして作成したHEXファイルに書き込まれたチップ IDと、ターゲットCPUのチップIDが 異なることにより 書き込み器のプログラムが 拒否していた可能性が あります。

つまり、書き込み器に問題が有った訳では無くて、ビルドに使用した開発環境が 新しすぎて古いCPUに 対応して無かった。 という事になります。

Device Selection

Device Family: All

Name	App./Boot Memory (Kbytes)	Data Me
ATtiny214	2	128
ATtiny2313	2	128
ATtiny2313A	2	128
ATtiny24	2	128
ATtiny24A	2	128
ATtiny25	2	128

メーカー純正の開発環境が 古いCPUチップの　対応を打ち切っていたとしても、gccや、サードパーティの 開発環境であれば　対応しているかもしれません。

で、ネットで　AT90S2313の書き込みに関わる記事を探していたら、「AT90S2313に　書き込みができるようになった」を　見つけました。 2023/0202 多分　年月日と思いますが　比較的新しい記事です。

Mira

左の女の子 Miraさんの　イラストがあるので　すぐ分かります。

ずっとうまくいかなかったAVRマイコン「AT90S2313??」への書き込みが　ついに ついに できるようになりました

この人も　AT90S2313で 長期に渡り悩んでいたようです。 女の子という事もあり、救いの手を差し伸べた方々が　おられたようです。

Miraさんは、今まで Arduino IDEで やっておられたようですが、Arduino IDEでは AT90S2313には 対応出来なくて、前に進めなかった様です。
で、どのように 書き込むプログラムをビルドしたかというと、BASCOM-AVR という AVRマイコン用開発環境を 教えてもらったようです。

BASCOMは BASIC言語の コンパイラです。まさか、AVRマイコンに BASCOMが あるとは思いませんでしたね。私が知ってる BASCOMは 30数年前 初期のMS-DOSの時代に MS-BASCOMという Microsoft社が作成したBASICコンパイラでした。
まだ、C言語を使い始める前で、パソコンに付いている BASICインタプリタでは 遅くて使い物にならない時代に、高速に動いてくれるので非常に重宝しました。

但し、私が 昔やっていたBASCOMとは、言語仕様が 少し異なるようです。 もしかしたら進化した という事なのでしょうか。 で、BASCOM-AVRは、無償のデモ盤がダウンロード出来ます。

但し、デモ盤は ビルドした プログラムサイズに制限が あるようです。 私の古い XPノートパソコンに インストールしました。

BASCOM-AVRの ダウンロード先は、Miraさんのサイトを 参照して下さい。
「AT90S2313への書き込み」でサーチすると出てきます。
今回は、さしあたり AT90S2313で Lチカプログラムが 作れればいいので、これも Miraさんのサイトから コピペしました。 短いプログラムです。 因みに そのプログラムの先頭に $Regfile="2313def.dat" が あります。 これが、AT90S2313の チップIDを 設定する記述と思われます。

LEDblink.BAS (1/2)

```
'-----------------------------------------------------------------
'LEDblink.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 100 '500 ms
Portb.3 = 0 'LED off
```

LEDblink.BAS (2/2)

```
$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
Waitms 100 '500 ms
If i>=60 Then
PORTB.0=1
i=0
Waitms 50
PORTB.0=0
'Waitms 10
End If
'$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
'$PROG &H00,&H00,&H00,&H00' generated. Take care that the chip supports all fuse bytes.
Loop
```

分からない部分が、上の fuse bytes. ですね。
&H00が 4つを 4行書いてあります。
下の2行は コメント化してあります。

で、前ページまでの　LEDblink.BAS を　ビルドして　正常にビルド出来ました。
そしていくつかのファイルが　生成されました。

下は、書き込み器に渡す　HEXファイルです。

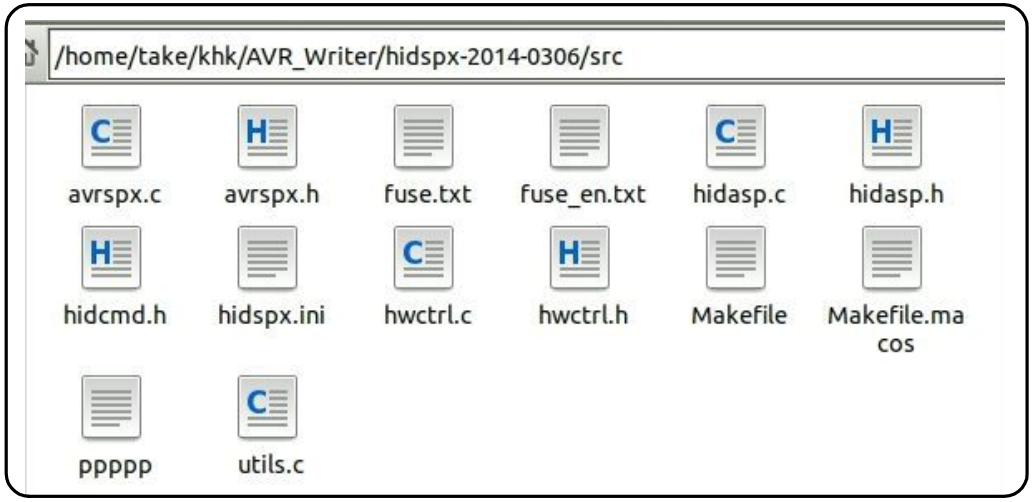
LEDblink.HEX

```
:100000000AC0189518951895189518951895189518956B
:1000100018951895189518958FED8DBFC0ECE8EB4E2E16
:10002000DD275D2EEEE7F0E0A0E6B0E088278D93B7
:1000300031197E9F766248FEF87BB80E090E0A0E678
:10004000D939C93C39AA0E60D911C910F5F1F4F57
:1000500A0E60D931C9384E690E021D0C39884E63B
:10006000090E01DD0A0E60D911C910C3350E01507D7
:10007000CF001C00AC0C09A80E090E0A0E68D9329
:10008009C9382E390E00BD0C098DCCF3197F1F7DE
:1009000095689462F80895E89462F80895EF93DB
:1000A000FF93EE27E82BE92B31F0E4ECF9E03197F0
:0C00B000F1F70197D1F7FF91EF9108954F
:00000001FF
```

プログラムの書き込み器

プログラムの書き込み器は Uさんから頂いた hidasp(下の画像)を 使用します。
Linux上で 使用する物のようで、環境を構築する必要があります。前回 中古ノートPCに インストールした 32bit Lubuntu (Liteな ubuntu という事です。) に hidaspを インストールします。 下の画像の 左の CD-Rに 必要なファイルが 格納されています。

CD-Rの中を見ると いくつかのバージョンの hidspxが 格納されていました。 フォルダ名が「私が使っているHIDaspx用hidxpx」の バージョンを 採用しました。 hidspx-2014-0306.tar.gz のアーカイブファイルなので 解凍というか展開しました。すると中身は README.TXTファイルと srcフォルダでした。 srcフォルダの中身を見るとCのソースファイル群でした。



という事で、まずは gccの開発環境が 必要となります。

Lubuntuは Ubuntu に近いと思われますので
Linuxの ターミナル窓で
sudo apt-get install make gcc
を 行います。 無事 インストール出来ました。

その後の インストール方法に関しては、
README.TXTに 操作方法を 書いてあります。
とはいえ、私も ちょっと 悩みました。

まず、srcディレクトリに 移動します。
$ cd src
そして、メイクを 行います。
$ make
$ sudo make install
で、インストール出来ます。
と書いてあります。
私は、最初の make にて エラーに ぶつかりました。 エラーは hidasp.c 内の
#include <usb.h> にて usb.hが 無いです。

usb.h が 無い事に対する対応は
$ sudu apt-get install libusb-dev
を 実行します。

その後に
$ make
$ sudo make install
を実行する事で、インストール出来ました。

こう書くと すんなりインストール出来た様に思われますが、試行錯誤で いろいろやっていたので 結構 悩みました。

hidspx を　単独で 起動すると、以下のような　オプションの一覧が　出てきました。
（ 画像で キャプチャして持って来たので　文字が小さくて荒いのは　ごめんなさい。）

```
take@take-dynabook-Satellite-B552-F:~/khk/AVR_Writer/hidspx-2014-0306/src$ hidspx
hidspx (b11.4) by t.k & senshu, GCC-7.5.0, Dec  8 2024
----
AVRSP - AVR Serial Programming tool R0.44 (C)ChaN, 2008  http://elm-chan.org/
Write code and/or data  : <hex file> [<hex file>] ...
Verify code and/or data : -v <hex file> [<hex file>] ...
Read code, data or fuse : -r{p|e|f|F|l} [-o<out hex file>]
HEX dump (Flash/Eeprom) : -r{p|e}h      [-o<HEX DUMP>]
Get AVR Information(Web): -r{I|i|d}
Write fuse byte         : -f{l|h|x}<bin>
Lock device             : -l[<bin>]
Copy calibration bytes  : -c
Erase device            : -e
Control port [-ph]      : -p{h|hu}
SPI control delay [-d3] : -d<n>
Help (Detail)           : -? or -h or --help

Supported Adapter:
HIDasp (USB -ph, -phu)
take@take-dynabook-Satellite-B552-F:~/khk/AVR_Writer/hidspx-2014-0306/src$
```