最終更新時間：2025 年 04 月 22 日 13 時 04 分 25 秒

マイコン制御学習用プログラム言語の設計と実装

・マイクロコントローラ

はじめに

バスコンピは「ヒダピオ」の「簡易プログラミング編」で「マイコン制御学習」を行うために「簡単なＢＡＳＩＣのプログラムを機械語にコンパイルするソフトウェア」です。

「もちろんインストール不要のソフトウェアとして」実装しました。

今は、「ヒダピオ」に同梱されています。

使い方

アーカイブを展開すると「bascompi」フォルダが出てくるので、ヒダピオの「bin」フォルダに移動してください。「バスコンピ」ショートカットを作成するために、最初に 1 度だけ「mkshortcut.vbs」をダブルクリックしてください。

コンパイルの仕方は簡単で、ソースプログラムの名前が「bascompi.bas」なら「バスコンピ」ショートカットをダブルクリックするだけです。それ以外の名前ならば、「バスコンピ」ショートカットにドラッグ＆ドロップします。このときは生成するファイル名が「bascom.hex」になるので注意してください。

出来た HEX ファイルは「HAG_mini」でマイコンに書き込みます。

ヒダピオの新しい版では「コンパイル」ボタンが使えるかもしれません。

文法

■基本命令文

ddrB=

ヒダピオではプログラムの最初に必ず入出力の設定をします。

PortB=

書式 portB= 出力値 , 時間 (ミリ秒)

ポート B にデータを出力した後、指定した時間待ちます。

End

プログラムの終了。プログラムの最後で必ず実行させるようにしてください。

プログラム例

```
ddrB=255
PortB=1,300
PortB=2,300
End
```

■繰り返し命令文

Do 〜 Loop

Do から Loop の間を繰り返します。Exit Do によって抜け出すことが出来ます。

プログラム例

```
ddrB=255
Do
  PortB=1,300
  PortB=2,300
Loop
End
```

Exit Do

Do 〜 Loop から抜け出します。

For 〜 Next (中級モード)

書式 For i=1 To 繰り返し回数

For から Next の間を指定された回数だけ繰り返します。

プログラム例

```
ddrB=255
For i=1 To 3
  PortB=1,300
  PortB=2,300
Next i
End
```

portB=2^i,300 (中級モード)

変数 j も使えます。

portB=2^(7-i),300 (中級モード)

変数 j も使えます。

■条件分岐命令文

If 〜 Else 〜 End If (上級モード)

書式

条件が成立するときは If から Else の間にある文を実行します。そうでないときは Else から End If の間にある文を実行します

使用できる条件式

・pinD.0=1

・pinB=255
・a=0

使用できる比較演算子

=,>,<

プログラム例

```
ddrB=255
Do
  If pinD.0=1 Then
    PortB=0,300
    PortB=255,300
  Else
    PortB=85,300
    PortB=170,300
  End If
Loop
End
```

■変数 (上級モード)

変数として a,i,j,k,wt1,wt2 が使えます。型は 16 ビット符号付整数型ですが､符号なし数として扱われることもあります。また、ポート B への出力は 8 ビットしかないことから上位 8 ビットの処理は省略されることがあります。

■代入文 (上級モード)

変数に値を代入する代入文として次の形式のものが使えます。

・a=0
・a=i
・a=a+1
・a=a-1
・a=255-a
・a=a*2
・a=2^a
・a=int(a/2)
・a=pinB
・a=portB

(以下は j,k についても同様)

・i=0
・i=a
・i=i+1
・i=i-1
・i=255-i
・i=i*2
・i=2^i
・i=int(i/2)
・i=pinB
・i=portB

ポート B への出力に変数が使えます。

・portB=a,300

待ち時間の指定に変数が使えます。

・wt1=300
・portB=0,wt1
・portB=a,wt2

■サブルーチン

Sub (上級モード)
　サブルーチンのラベルを指定します。

書式 Sub ラベル

Return (上級モード)
　サブルーチンから戻ります。

書式 Return

Gosub (上級モード)
　サブルーチンに制御を移します。

書式 Gosub ラベル

■ LCD 制御命令

Cls (LCD モード)
　画面に表示されている文字を全て消します。

書式 Cls

LCD を使うときは最初の表示の前に必ず実行してください。

Lcd (LCD モード)
　LCD に指定された文字列を表示します。

書式 Lcd 文字列

文字列は "" で括ります。

プログラム例

```
Cls
Lcd "BASCOMPI"
```

```
End
```

Cursor Off (LCD モード)
　カーソルを非表示にします。
Cursor On (LCD モード)
　カーソルを表示するようにします。
Display Off (LCD モード)
　LCD を非表示にします。
Display On (LCD モード)
　LCD を表示するようにします。カーソルは非表示にします。
Upperline (LCD モード)
　上の行に表示します。
Lowerline (LCD モード)
　下の行に表示します。
Shiftlcd Left (LCD モード)
　表示されている文字列を左にずらします。
Shiftlcd Right (LCD モード)
　表示されている文字列を右にずらします。
Locate (LCD モード)
　表示位置を指定します。

書式 Locate 行 , 列

Waitms (LCD モード)
　指定された時間待ちます (ミリ秒単位)。

書式 Waitms 時間

Deflcdchar (LCD モード)
　指定された字形を指定された文字番号で登録します。

書式 Deflcdchar 文字番号 , データ 1, データ 2, データ 3, データ 4, データ 5, データ 6, データ 7, データ 8 指定できる文字番号は 0 から 7 までです。

Lcd chr() (LCD モード)
　指定された文字番号に対応する文字を LCD に表示します。

書式 Lcd chr(文字番号)

Lcd 変数名 (LCD モード)
　指定された変数の値を LCD に表示します。

書式 Lcd 変数名

■ Sound 出力命令

Sound (Sound モード)
　指定された周波数の音を指定された時間出力します。

書式 Sound 出力ポート , 長さ , 高さ

マイコンの動作周波数によって音の高さに指定する数値が変わります。

■その他の命令 1

ヒダピオの簡易プログラミングには出てきませんが電光掲示板 (電子掲示板) や電子オルゴールの中で使っています。

Data
　プログラム中にデータを置きます。

書式 Data データ 1, データ 2,...

Data 文の前にはラベルを置いてください。

Read
　Data 文で置いたデータを変数に読み出します。

書式 Read 変数

Restore
　データを読み出す位置を指定します。

書式 Restore ラベル

Read 文で最初のデータを読み込む前には必ず Restore 文を実行してください。

■その他の命令 2

電子オルゴールを実現するために BASCOM-AVR との互換性を高める目的で追加された命令です。しかし、BASCOM-AVR との完全な互換性を目指すものではありません。また、現在の電子オルゴールではこれらの命令は使われていません。

a=Lookup(i,label)
　指定されたデータ行から、指定された位置の数値データを変数に読み出します。

a="a"
　変数に文字 (のコード) を代入します。

If a="a" Then
　変数と文字 (のコード) を比較します。

■その他の命令 3

より一般的な応用が実現できるようにするために追加された命令です。

PortD=
　ポート D にデータを出力します。

a=Getadc(0)
　A/D 変換された値を変数に読み出します。

参考 ELM - シンプルな A-D コンバータ

■ BASCOM-AVR モード

バスコンピの文法ははヒダピオの簡易プログラミングで使用する BASIC 言語に準拠しています。ヒダピオの簡易プログラミングで作成したプログラムをそのまま受理し AVR のアセンブリ言語へ翻訳します。

しかし、ヒダピオが「コンパイラへ」ボタンによって出力するのはは BASCOM-AVR の文法に変換されたソースプログラムです。また、電光掲示板や電子オルゴールでは BASCOM-AVR 用のプログラムを出力します。これは、もともとヒダピオシステムが BASCOM-AVR を前提としているからです。

そこで、バスコンピは BASCOM-AVR のサブセットもコンパイルできるようにしてあります。

ヒダピオと BASCOM-AVR ではポート B の論理が逆になっています。

ヒダピオシステムでは回路としては負論理ですが、中学生が対象ということもありヒダピオ上は LED が '1' で点灯し、'0' で消灯するようなっています。

つまり､ヒダピオの BASIC 言語としてコンパイルするときと BASCOM-AVR としてコンパイルするときではポート B に出力するデータが反転しているのです。

バスコンピでは 2 つのモードを持っていてこれらを切り替えています。

モードを切り替えるには 2 つの方法があり、ひとつはコンパイラに対するオプションで指示する方法です。もうひとつは、プログラム中に以下の文があると BASCOM-AVR モードに切り替わります。

・$REGFILE
・$CRYSTAL
・CONFIG
・DIM

$REGFILE
　*.dat ファイルは読まずに、CPU の判別に使用しています。値として "ATtiny2313.DAT" と "m168Pdef.dat" が有効ですが、現在は ATtiny2313 のみの対応となっています。

$CRYSTAL
　クロックの値を読み取っています。値として 1000000 と 12000000 のみが有効です。

CONFIG

CONFIG PORTB、CONFIG PORTD は機能します。CONFIG LCD は単にスキップします。それ以外はスキップします。

DIM

宣言された情報は記号表に登録されますが、バスコンピには型がありませんので利用されません。例外として、電光掲示板では Read 文で変数に Byte 型のデータを読むところがあり、そこで記号表を参照しています。

サンプルプログラム

■ LED の点滅

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。

■ LED の明滅

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。尚、は 0.53 以降のコンパイラでコンパイルすることができます。

Fade.bas

```
ledPin Alias PortB.3
brightness Alias OCR1A

Config ledPin=Output

TCCR1B=&h01
TCCR1A=&h81

Do
  For i=0 To 254
    brightness=i
    Waitms 2
  Next i
  For i=255 To 0 Step -1
    brightness=i
    Waitms 2
  Next i
Loop
End
```

■スイッチでオンオフ制御

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。尚、は 0.49 以降のコンパイラでコンパイルすることができます。

Sw.bas

```
swPin Alias PinD.6
ledPin Alias PortB.3
Config ledPin=Output
Config swPin=Input
Const HIGH=1
Const LOW=0

Do
  If swPin=HIGH Then
    ledPin=HIGH
```

```
  Else
    ledPin=LOW
  End If
Loop
```

■楽譜の演奏 1

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。尚、は 0.54 以降のコンパイラでコンパイルすることができます。

Playnote.bas

```
bellPin Alias PortB.3
freq Alias k
note Alias k

Const C=262%
Const D=294%
Const E=330%
Const F=349%
Const G=392%
Const A=440%
Const B=494%
Const c=523%
Const d=587%
Const e=659%
Const f=698%
Const g=784%
Const R=1%

Dim ocr As Long
Dim timer1_toggle_count As Long
Dim duration As Long

On COMPARE1A TIMER1_COMPA_vect

Config bellPin=Output

TCCR1A=0
TCCR1B=Bits(WGM12,CS10)
Enable Interrupts

Do
  Restore NOTES
  Do
    Read note
    If note=0 Then
      Exit Do
    Else
      If note=R Then
      Else
        Gosub tone
      End If
      Waitms 330
    End If
  Loop
  Waitms 2400
Loop
End

tone:
  ocr=_XTAL/freq
  ocr=Int(ocr/2)
  ocr=ocr-1
  If ocr>&hffff Then
    ocr=_XTAL/freq
    ocr=Int(ocr/2)
    ocr=ocr/64
    ocr=ocr-1
    TCCR1B=Bits(WGM12,CS11,CS10)
  Else
    TCCR1B=Bits(WGM12,CS10)
  End If
  timer1_toggle_count=freq*2
```

```
  timer1_toggle_count=timer1_toggle_count*300
  timer1_toggle_count=timer1_toggle_count/1000
  OCR1A=ocr
  TIMSK.OCIE1A=1
Return

TIMER1_COMPA_vect:
If timer1_toggle_count=0 Then
  TIMSK=0
  bellPin=0
Else
  PINB.3=1
  timer1_toggle_count=timer1_toggle_count-1
End If
Return

NOTES:
Data R,D,D,B,A,G,D,R, R,D,D,B,A,G,E,R, R,E,E,c,B,A,F,R, R,d,d,d,c,A,B,R
Data R,D,D,B,A,G,D,R, R,D,D,B,A,G,E,R, R,E,E,c,B,A,d,d, d,d,e,d,c,A,G,R, d,R
Data B,B,B,R, B,B,B,R, B,d,G,A,B,R,R,R, c,c,c,c,c,c,B,B,B, B,A,A,B,A,R, d,R
Data B,B,B,R, B,B,B,R, B,d,G,A,B,R,R,R, c,c,c,c,c,c,B,B,B, d,d,c,A,G,R, g,R,0%
```

■カウントアップ

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。

Countup.bas

```
Config PortB=252
Config PortD=64

For i=0 To 9
  a=Lookup(i,LED)
  PortD=a
  a=a*2
  a=a*2
  PortB=a,500
Next i
End

LED:
Data 63%,6%,91%,79%,102%,109%,125%,39%,127%,111%
```

■アナログデジタル変換

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。(2011.10.13 追記) マイコンの動作周波数が 1MHz のときにはコンデンサの静電容量を 0.1 μ F にしてください。サンプルプログラム () にあった動作周波数の指定は不要のため削除しました。

Adc.bas

```
Config PortB=252
Config PortD=64

Do
  a=Getadc(0)
  a=a/103
  a=Lookup(a,LED)
  PortD=a
  a=a*2
  a=a*2
  PortB=a
Loop
```

```
End

LED:
Data 63%,6%,91%,79%,102%,109%,125%,39%,127%,111%
```

■サーボモータの制御

■押しボタン信号機

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。尚、は 0.55 以降のコンパイラでコンパイルすることができるようにする予定でます。

Signal.bas

```
RPin Alias PortB.7
YPin Alias PortB.6
GPin Alias PortB.5
RRPin Alias PortB.1
GGPin Alias PortB.0
swPin Alias PinD.6
Config PortB=&B11100011
Config swPin=Input
Const HIGH=1
Const LOW=0
Dim H As Integer
Dim L As Integer
Dim swMemo As Integer
Declare Sub carsig(g as integer,y as integer,r as integer)
Declare Sub mansig(r as integer,g as integer)
swMemo=LOW
H=HIGH
L=LOW
Call carsig(H,L,L)
Call mansig(H,L)
Do
  If swPin=HIGH Then
    If swMemo=LOW Then
      Waitms 4000
      Call carsig(L,H,L)
      Waitms 4000
      Call carsig(L,L,H)
      Call mansig(L,H)
      Waitms 16000
      For i=0 To 7
        Call mansig(L,H)
        Waitms 500
        Call mansig(L,L)
        Waitms 500
      Next i
      Call mansig(H,L)
      Waitms 4000
      Call carsig(H,L,L)
    End If
    swMemo=HIGH
  Else
    swMemo=LOW
  End If
Loop
End

Sub carsig
  GPin=g
  YPin=y
  RPin=r
  Return
End Sub

Sub mansig
  RRPin=r
  GGPin=g
  Return
```

```
End Sub
```

■楽譜の演奏 2 (Midi 楽器の制御)

小山先生の回路図を参考にして回路を組みます。メモ帳でを作成し、そのファイルアイコンをバスコンピのショートカットにドラッグ＆ドロップします。出来上がった bascom.hex を HAG_mini 等でマイコンに書き込みます。尚、は 0.55 以降のコンパイラでコンパイルすることができるようにする予定でます。

Midi.bas

```
$baud=31250
bellPin Alias PortD.3
note Alias k

Config bellPin=Output

Dim voice As Integer
Dim _H70 As Integer
Dim _H00 As Integer

Const C=60%
Const D=62%
Const E=64%
Const F=66%
Const G=67%
Const A=69%
Const B=71%
Const c=72%
Const d=74%
Const e=76%
Const f=78%
Const g=79%
Const Z=125%
Const z=126%
Const R=127%

Const HIGH=1
Const LOW=0

Declare Sub noteOn(data1 As Integer,data2 As Integer)

_H70=&H70
_H00=&H00

Do
  Print Chr(&HC0);Chr(voice);
  Restore NOTES
  For i=0 To 139
    Read note
    If note=Z Then
      bellPin=HIGH
    Else
      If note=z Then
        bellPin=LOW
      Else
        If note=R Then
          Waitms 300
        Else
          Call noteOn(note,_H70)
          Waitms 300
          Call noteOn(note,_H00)
        End If
      End If
    End If
  Next i
  Waitms 2400
  voice=voice+1
  If voice>16 Then
    voice=11
  End If
Loop
End
Sub noteOn
```

```
  Print Chr(&H90);Chr(data1);Chr(data2);
End Sub
NOTES:
Data R,D,D,B,A,G,D,R, R,D,D,B,A,G,E,R, R,E,E,c,B,A,F,R, R,d,d,d,c,A,B,R
Data R,D,D,B,A,G,D,R, R,D,D,B,A,G,E,R, R,E,E,c,B,A,d,d, d,d,e,d,c,A,G,R, d,R
Data Z,B,B,B,z,R, Z,B,B,B,z,R, B,d,G,A,B,R,R,R, c,c,c,c,c,B,B,B, B,A,A,B,A,R, d,R
Data Z,B,B,B,z,R, Z,B,B,B,z,R, B,d,G,A,B,R,R,R, c,c,c,c,c,B,B,B, d,d,c,A,G,R, Z,g,z,R
```

おまけ：BASCOM-AVR をインストールせずに使う方法

拙作バスコンピができたのはインストールせずに使える AVR 用の BASIC コンパイラが欲しいという要望に応えるためでした。実は BASCM-AVR には bascomp.exe というコマンドラインからコンパイラを使うためのユーティリティが用意されています。MCS Electronics 社のページからダウンロードできますが DEMO バージョンにも付属していて、BASCOM-AVR のインストールディレクトリの中にあります。バージョン 2.0.5.0 まではソース中に $REGFILE や $CRYSTAL だけでなく $BAUD まで書かなければいけなかったのですが、最新のバージョン 2.0.7.1 からは、これらを書かなくても良くなりました。

■必要なファイルのコピー

Windows XP では通常「C:\Program Files\MCS Electronics\BASCOM-AVR」にインストールされます。ここから必要なファイルだけ取り出します。ATtiny2313 用にコンパイルするために最低限必要なのは「BASC-AVR.DLL」、「bascomp.exe」、「ATtiny2313 .DAT」、「LIB\mcs.lbx」、「LIB\Double_Trig.LBX」、「LIB\Double.LBX」、「LIB\FP_Trig.LBX」です。

■バッチコマンドの作成

次のようなバッチファイルを作成します。

bascomp.bat

```
bascomp.exe %1 CHIP=31
```

■ショートカットの作成

作成したバッチコマンドのショートカットを作成します。

■コンパイル

作成したショートカットにソースファイルをドラッグ＆ドロップするとコンパイルできます。エラーメッセージは、*.err ファイルに出力されます。

参考ページ

■ヒダピオ

- JA 制御　ヒダピオ システム (中学校 , 技術・家庭科 , コンピュータ制御 ,USB-IO, プログラミング)
- 「プログラミングと計測・制御」の教材 HIDapio | 「ものづくり」は基本です。「ORCA 研究所」http://bl.orca-labo.com
- 新 ORCA 研究所ブログ｜「プログラミングと計測・制御」の教材 HIDapio
- http://www.kokyokyo.kofu-ymn.ed.jp/bukai/gijutu.html
- 教材開発アイディア - むっちーのブログ - Yahoo! ブログ
- 中学校の技術の授業で、ＪＡ制御ヒダピオというものを使って、実習をし ... - Yahoo! 知恵袋
- ArduBlock（ヒダピオシステム向けバージョン）を使おう | Gijyutu.com | Gijyutu.com
- パソコン制御用ソフトで計測・制御の学習をしよう !! - 弘前大学 ...

・AVR マイコンという、これまでまったく無縁だった世界に魅了され｜人生に出会う 7 WAYS ＋α

・PDF 版（生徒用） - JA 制御 ヒダピオ システム
・ヒダピオ制御カー | Manualzz
・中 3 の技術です。 - Clearnote
・ヒダピオ学習回路 付属説明書
・私家版 Ardublock
・Arduino/ リンク集 - 講義のページ
・NRS+　特集・小ネタ　Arduino で電子工作 2　- サンプルスケッチで L チカと簡単電子オルゴール -
・産教連通信第 199 号
・産教連通信
・産教連通信第 187 号ホームページ用
・色々 | It wishes strongly and it lives.
・計測・制御教育のための教材開発とその教材を活用 した授業設計及び実践
・未来社会・生活をつくる力を育てる技術・家庭科教育
・「ヒダピオ」tag 一覧｜人生に出会う 7WAYS ＋α
・授業のエキスパート養成事業（第３学年技術・家庭科）授業実践記録
・ArduBlock（ヒダピオシステム向けバージョン）を使おう | Gijyutu.com　技術の面白教材集
・Km2Net - おにたま (オニオンソフト) のおぼえがき
・体験と言葉をつなぐ教育
・Arduino プログラミングでものづくり (ATtiny4313 版)
・特別支援教育で参考になる教材情報

■簡易プログラミングでの文法について

・JA ヒダピオの使用方法（基本）
・JA ヒダピオの使用方法（基本）
・ヒダピオ学習回路 付属説明書
・ヒダピオ学習回路 付属説明書

■ AWK

・AWK ちょいメモ #AWK - Qiita
・AWK FAQ (from comp.computer-lang.awk)
・The GNU Awk User's Guide - gawk をインストールする
・awk++ - object oriented AWK
・Introduction to Script language
・awklang.org – Ｔ he site for things related to the awk language
・AWK++ によるオブジェクト指向入門 - Wikibooks
・timm/lawker: Automatically exported from code.google.com/p/lawker
・valdas/lawker: Automatically exported from code.google.com/p/lawker
・Google Code Archive - Long-term storage for Google Code Project Hosting.
・awk 実装の markdown 処理系 - Bye Bye Moore

・gawk-gd

・awk が新しくなる！？ 本家 Awk が Unicode (UTF-8) と CSV 対応に！ - Qiita
・開発から 45 年、プログラミング言語 AWK に Unicode サポートが追加 | スラド デベロッパー
　・GitHub - onetrueawk/awk: One true awk
・世界のプログラミング言語 (40) ハッカー御用達の言語 AWK はテキスト処理で今日も活躍中 | TECH+（テックプラス）
・[awk 入門 _2022 講義ノート]

・awklang.org – Ｔ he site for things related to the awk language
・[AWK「グフフ……」女騎士「くッ……！」アドベントカレンダー 2019 7 日目]| 東京工業大学デジタル創作同好会 traP
・Use Gawk to process CSV files - In the shadow of a kind tiger
・www.mogami.com/unix/awk-02.html(IA)
・myzsyweb ’s gists · GitHub
・awk 組み込み変数の活用でさまざまな形式のテキストファイルを扱う
・<code>%</code> の入った文字列を awk で <code>printf $0</code> するとエラーに | 上田ブログ
・gawk で先頭が空の CSV データを加工すると面白いことになる #AWK - Qiita
・[gawk4.1.4+MPFR でフィボナッチ数列を F[0] から F998] まで出力してみる。 - labunix's blog

・例題で学ぶ AWK プログラミング (入門編)
・AWK プログラミング (応用編)
・AWK 入門
・awkgui - jackparke
・GAWK 4.2.X DLL を作成する

・gawk · GitHub Topics · GitHub
・GitHub - freznicek/awesome-awk: A curated list of awesome AWK frameworks, libraries, software and resources
・GitHub - PacktPublishing/Learning-AWK-Programming: Learning AWK Programming, published by Packt
・comp.lang.awk - Google グループ
・comp.lang.awk

・AWK Users JP :: 日本の AWK ユーザのためのハブサイト (IA)
・Front Page - Japan GNU AWK Users Club(IA)
・AWK - Wikipedia(日本語)
・AWK - Wikipedia(英語)
・The GNU Awk User's Guide
・『コーパス日本語学ガイドブック』付録 CD 収録の jgawk について

・AwkChannelWiki: Home Page Japanese
・AwkChannelWiki: Awk On Windows
・awk.info » Awk.info(IA)
  ・Wrapper for AWK providing modules download | SourceForge.net
  ・cheusov/runawk: Powerful wrapper for AWK interpreter
  ・Quik Trim
  ・Jawk - AWK for Java
  ・Google Code Archive - Long-term storage for Google Code Project Hosting.
  ・spawkinfo
  ・MAWK – Freecode
  ・mawk – pattern scanning and text processing language
  ・takubo/inet

・「Gawky Auk」(Index of Gawky Auk)

・awk mania!
・Release(IA)
・Release
・AWK
・AWK

- AWK Hacks
- takubo (田窪守雄 (Takubo Morio)) · GitHub
  - takubo/awkGL: OpenGL extension for GNU AWK(GAWK).
  - takubo/OpenCV-AWK: OpenCV extension for GNU AWK(GAWK).
  - takubo/cacc: 究極の黒魔術
  - takubo/awk-posix-signal: GNU AWK(GAWK) で POSIX Signal 関連のシステムコールを使うための拡張機能。
  - takubo/awk_lib: Library for AWK.
  - takubo/cawk: ffi extension for gawk
  - takubo/open-usp-tukubai-by-awk: AWK による open-usp-tukubai の実装。
  - takubo/slides
  - takubo/awk-posix-select: GNU AWK(GAWK) で POSIX slect システムコールを使うための拡張機能。
  - takubo/awkmine: AWK で実装したマインスィーパー風ゲーム。ターミナル上で動作する。
  - takubo/awk_scripts: My awk scripts.

- takubo/sedtris: tetris in sed

- GAWK 文字列をバイト単位で数える方法 (IA)

- awk 製 3D シューティングゲーム awkaster で遊んでみた – matoken's meme
  - GitHub - TheMozg/awk-raycaster: Pseudo-3D shooter written completely in gawk using raycasting technique
- AWK 製 FPS ゲームの「TERMMAZE3D」 – matoken's meme
- Debian/Ubuntu で awk を切り替える – matoken's meme
- mikkun/AWKTC: AWKTC is Workable Klutzy Time-wasting Command

- (GNU)AWK で簡易 Web フレームワークを作る - Qiita

- awk をプログラミング言語として使う時の技術 - Qiita
- AWK 処理系の比較 – YOSBITS

- [SG ソフトウェア開発ブログ : awk]gawk と mawk
- mawk MBCS (32bit 版) の詳細情報 : Vector ソフトを探す！

- 例題で学ぶ AWK プログラミング (入門編)
- AWK プログラミング (応用編)
- プログラミングをする

■ AWK 本

- O'Reilly Japan - プログラミング言語 AWK 第 2 版
- プログラミング言語 AWK
- AWK を 256 倍使うための本
- sed & awk プログラミング 改訂版 : Dale Dougherty, Arnold Robbins, 福崎 俊博 : 本
- sed & awk プログラミング（改訂版）| Ohmsha
- O'Reilly Japan - sed & awk プログラミング 改訂版
- awk でプログラミング
- Amazon.co.jp： awk でプログラミング : 植村 富士夫 , 富永 浩之 : 本
- AWK 実践入門：書籍案内｜技術評論社
- 「シェル芸」に効く！ AWK 処方箋（斉藤 博文）｜翔泳社の本
- Software Design 2013 年 9 月号｜技術評論社
- The AWK Programming Language, Second Edition

■ GAWK for Windows

- corbie のブログ :Windows 版 gawk 選び
- WILLs' trash can (PROGRAM) (IA)
- gawk.exe 3.1.6 for Windows(IA)

- Gawk on Windows(IA)

- GitHub - mbuilov/gawk-windows: Fork of the GNU Awk (https://www.gnu.org/software/gawk/)
- Downloads - gnu-on-windows - compile GNU programs for win32 - Google Project Hosting
- ezwinports - Browse Files at SourceForge.net

- Homepage Volker Kiefel
  - Freie und "Open Source"-Software compiled for Windows systems
- Freeware Downloads

- Byte code version of 3.1.8 now available - comp.lang.awk
- Experimental gawk version, *with debugger*, now available
- DJGPP
- Index of /pub/djgpp/current/v2gnu
  - http://mirrors.fe.up.pt/pub/djgpp/current/v2gnu/gwk500b.zip
  - http://mirrors.fe.up.pt/pub/djgpp/current/v2gnu/gwk500d.zip
  - http://mirrors.fe.up.pt/pub/djgpp/current/v2gnu/gwk500s.zip

- GAWK Windows で GAWK を使用するために

■ fatal: print to "standard output" failed (No space left on device)

```
fatal: print to "standard output" failed (No space left on device)
```

- AWK 座敷
- gawk「cmd. line:1: (FILENAME=- FNR=1) fatal: print to "standard output" failed (No space left on device)」で死亡 - いろきゅう .jp 〜 Programmable maiden 〜 Tech side
- いろきゅう .jp 〜 Programmable maiden 〜 Tech side
- GAWK サクラエディタで快適に使う
- GAWK サクラエディタで快適に

■ warning: regexp escape sequence `\"' is not a known regexp operator

```
warning: regexp escape sequence `\"' is not a known regexp operator
```

- 1790227 – regexp escape sequence `\"' is not a known regexp operator
- warning: regexp escape sequence `\"' is not a known regexp operator

■ AWK 言語処理系の実装

- gawk.git - gawk
- [bug-gawk] Win32 Gawk 4.0.0 bugs
- Difference Between awk, nawk, gawk and mawk | Baeldung on Linux

- GitHub - cc2qe/zawk: make awk better

- GitHub - ThomasDickey/mawk-20121129: mawk 1.3.4 20121129
- GitHub - ploxiln/mawk-2: Mike Brennan's new development of his original "mawk" - https://github.com/mikebrennan000/mawk-2
- GitHub - snddns/mawk-notes: Going through the awk programming language with mawk
- GitHub - d4tocchini/spawk: forked from fast scripting phenom mawk (v2 beta), focused on adding parallelism, memory-efficient persistence, embeddability and healthy dose of sugar while maintaining performance

- GitHub - gvlx/gawk: forked from git://git.savannah.gnu.org/gawk.git
- GitHub - e36freak/awk-libs: GNU awk libraries

- noyesno/awka: Revive awka - Awk to C Compiler
- awkc -- compile AWK programs into executables

- GitHub - andychu/bwk: Kernighan's Awk a.k.a. "One True Awk"

- awk4j - Home
- awk4j - Home
  - awk4j - AWK for Java Platform プロジェクト日本語トップページ - OSDN
- TAWK Compiler
- awkium: an awk interpreter
- awkium: an awk interpreter
  - ダウンロードファイル一覧 - awkium - 拡張された awk インタプリタ - OSDN
  - morilib home(IA)
  - morilib.net
  - morilib.net（@SchluesselLang）さん / Twitter

- GitHub - benhoyt/goawk: A POSIX-compliant AWK interpreter written in Go
- GoAWK, an AWK interpreter written in Go
- 私的 goawk リポート - 標準愚痴出力

- κ een さんは Twitter を使っています : 「Rust 製の AWK（っぽい言語）の実装。型推論して JIT したり並列化したり SIMD で高速化頑張ってたりするらしい ezrosent/frawk: an efficient awk-like language https://t.co/LbWgvlTsOj」 / Twitter
- GitHub - ezrosent/frawk: an efficient awk-like language
- malloc/free まとめ - きゅうり。

- GitHub - wenley/rust-awk: Rust Implementation of Awk

- webawk/README.md at master · agordon/webawk · GitHub
- WebAWK - client-side awk in Javascript
- Javascript で awk が動く。魔界ライブラリ WebAWK - Bye Bye Moore

- AWK live editor
- A web based AWK interpreter
- Online Awk Compiler - Online Awk Editor - Online Awk IDE - Awk Coding Online - Practice Awk Online - Execute Awk Online - Compile Awk Online - Run Awk Online
- Online AWK Compiler - Online AWK Editor - Run AWK Online - Online AWK Runner
- Online AWK (gawk) compiler and IDE - API provided by Sphere Engine - Ideone.com

■ GPU

- philass/gpawk: Awk on the GPU

■ AWK SQL

・Home | Wim's pages

・XML Processing with GNU Awk
・XML Processing with GNU Awk
・gawk libraries for XML, PostgreSQL,... | Free Development software downloads at SourceForge.net

・dbohdan/sqawk: Like Awk but with SQL and table joins
・Sqawk
・SPAWK
・SPAWK
・awklang.org – SPAWK - SQL Powered AWK

■ AWK による言語処理系の実装

・AWK（gawk）でシンプルな自作言語のコンパイラを書いた

・AWK による SPARC V8 逆アセンブラ 竹下世界塔の計算機よもやま話 / ウェブリブログ → AWK による SPARC V8 逆アセンブラ : 竹下世界塔の計算機よもやま話
・ホワット・ア・ワンダフル・ワールド もう AWK なんて使わないなんて言わないよ絶対〜 (IA)
・aaa - The Amazing Awk Assembler by Henry Spencer
・しまりすくらぶ　梵天丸の PIC 改造工作／まきものの応用から，ものづくりメカトロニクスへ
・しまりすくらぶ　梵天丸の PIC 改造工作／まきものの応用から，ものづくりメカトロニクスへ
・Shudo's Notes (2007/9)
・PIC マイコンのお部屋 (IA)
・Mitzyuki's Blog :: awk を利用した構文解析ツール
・紫ログ :Grass on AWK - AWK でちょっと草を - livedoor Blog（ブログ）
・myzsyweb ’s gists

・darius/awklisp: A Lisp interpreter written in Awk.
・A LISP interpreter in Awk - ChaSen.org
・CLiki: AWK Lisp
  ・Index of /afs/cs/project/ai-repository/ai/lang/lisp/impl/awk
・CLiki: CLAWK
・[Announce] CL-AWK (Common Lisp AWK library)
・GitHub - sharplispers/clawk: CLAWK is an AWK implementation embedded into Common Lisp, by Michael Parker. This version contains some patches to make it compile again.
  ・http://www.foldr.org/~michaelw/lisp/clawk/(IA)
・Solene'% : Common LISP awk macro for easy text file operations
・シェルスクリプトで LISP 処理系を作ってみた

・I wrote a compiler in awk! To bytecode; I wanted to use the awk-based compiler a... | Hacker News
・Mercat

・楽天 Kobo 電子書籍ストア : Tcl/Tk 8.6 の世界 - { awk lisp } の世界へようこそ - 橋本ただし - 1230001284034

・GitHub - siraben/awk-vm: A virtual machine and assembler written in AWK.
・GitHub - LBPHacker/R216: Stuff one needs to play around with my R216 computers (see README)

・ａｗｋ言語で作った簡易クロス・アセンブラ
・AWK でアッセンブラーを作る (2)　アッセンブラースクリプト編 | Asm Sphere
・assembly - Is there a reference for the assembler-like output produced by `awk -W dump`? - Unix & Linux Stack Exchange

・GitHub - prolog8/awkprolog: Prolog in AWK

・GitHub - Syuparn/webawk: simple (mock) web server framework written in awk
・(GNU)AWK で簡易 Web フレームワークを作る - Qiita

・Tinyproxy でさくっとリバースプロキシを立ててみる - Qiita
・webawk

・GitHub - kevin-albert/awkserver: A minimal HTTP server that runs in GAWK

・タグシステム（抽象機械）の各言語等実装例まとめ - Qiita

■ AVR

・えるむ /ChaN

■ BASCOM-AVR

・BASCOM-AVR - MCS Electronics
・BASCOM-AVR 操作方法 マニュアル

■ LCD **関連**

・LCD の制御→ LCD の制御

■学習指導要領

・第 2 章　各教科　第 8 節　技術・家庭：文部科学省

・中学校学習指導要領　第 8 節　技術・家庭：文部科学省
・高等学校学習指導要領　第 10 節　情報：文部科学省
・高等学校学習指導要領　第 7 節　情報：文部科学省

■教科書

・[平成 24 年度版 中学校 教科書のご紹介新しい技術・家庭／技術分野]：東京書籍
・平成 24 年度用中学校技術・家庭科　技術分野教科書のご案内 / 開隆堂出版株式会社
・教育図書株式会社
・【東京書籍】 教科書教材　技術・家庭
・技術分野教科書 - 教育図書教育図書

■実態調査

・平成 22 年度 長野県中学校技術・家庭科 実態調査のまとめ

■その他の教材

・Arduino 風 パソコン制御とマイコン制御
・Arduino 風 パソコン制御とマイコン制御
・Arduino プログラミングでものづくり (ATtiny4313 版)

(2011.9.8) 今頃気づきましたが小山先生のページに「バスコンピ」の文字がありました。

■ BASCOM?

・TORO's Software library(PC-E500)
・TORO's Software library(PC-E500)

■ BASCOMP

・blghnbngl/BasComp: Basic Computer Behavioral Description

■高校生

・AVR マイコンを用いた交通信号機制御 (IA) 岡山県立岡山工業高等学校 平成 24 年度　情報技術科　課題研究
・Microsoft Word - AVR マイコンを用いた交通信号機制御 .docx

既知の問題

改訂の履歴とファイルのダウンロード

注意 ! 0.36 から 0.43 までは、Windows 7/Vista で動作しない場合があります。

自己展開形式（.exe）のファイルと zip ファイルの中身は全く同じものです。

2012.3.13 0.57

(1) [本体] PortB.x=0 の形式が直接使えるようにした。

(2) [本体] CALL 命令を使用せずにサブルーチンを呼び出す書式も使用できるようにした。

2012.3.5 0.56

(1) [本体] 名前付き定数の 2 重定義チェックが間違っているのを直した。

(2) [本体] Data 文中の定数式を加算以外も計算するようにした。ついでに再帰的に計算するようにした。循環参照は検出するようにしたつもり。

(3) [本体] 別名を使用したときに a=a+1 のパターンが認識できなかったのを直した。

(4) [本体] Waitms 文で名前付き定数が使えるようにした。

(5) [本体] 定数 _XTAL を設定するタイミングが間違っていたので直した。

(6) [本体] X=2^I、X=2^J が処理されていなかったのを直した。

(7) [本体] Call 文が正しく処理されていないのを直した。

(8) [vavrdisasm.exe] 同梱する逆アセンブラを最新のものにした。

2012.2.1 0.55

(1) [本体] シリアル通信の出力に対応した。$BAUD、Print 文、Printbin 文が使えるようにした。入力は未だ出来ない。

(2) [本体] Call 文でサブルーチンに引数を渡せるようにした。定義のための Declare Sub 文、Sub ～ End Sub 文が使えるようにした。

(3) [本体] Dim 文で String 型の変数と 1 次元の配列が宣言できるようにした。SRAM 上の領域も確保されるようにした。

(4) [本体] PortB.0=a のような書き方が出来るようにした。

(5) [本体] 配列要素への単純変数、配列変数および 10 進定数の代入が出来るようにした。(6) [本体] 代入文の右辺に 16 進定数と 2 進定数が使えるようにした。

(7) [本体] 代入文の右辺で配列の要素が使えるようにした。ただし、演算は出来ない。

(8) [本体] Data 文の中で定数同士の加算が出来るようになった。

(9) [本体] ソースコードからデバッグ用のメッセージ出力やコメントアウトした不要な行を削除した。

2012.1.6 0.54

(1) [本体] Bits/Nbits 関数が使えるようにした。

(2) [本体] If 文で Long 型変数と定数の比較が行なえるようにした。

(3) [本体] 初期化ルーチンを生成するタイミングが間違っているのを直した。

(4) [本体] Long 型変数のインクリメント / デクリメントができるようにした。

(5) [本体] If 文で 16 進定数が使えるようにした。

(6) [本体] 16 ビットの乗算ができるようにした。

(7) [本体] Data 文でカンマの前後に空白があると記号定数が認識できないのを直した。

(8) [本体] If 文で変数名にアンダースコアが使えないのを直した。

(9) [本体] If 文での変数と 0 との比較を高速化した。

(10) [本体] If 文で定数値に % が付いていても良いようにした。

2011.12.28 0.53

(1) [本体] 宣言した Long 型の変数に値を代入できるようにした。

(2) [本体] 32bit の除算ができるようにした。加算、減算、乗算についてはまだできない。

(3) [本体] 定数 _XTAL を定義しプログラム中で動作クロック数 ($Crystal で指定した値) を読み出せるようにした。

(4) [本体] HIGHW 関数により Long 型変数の上位 16bit を取り出すことができるようにした。

(5) [本体] $Baud でボーレートを設定できるようにした。シリアル通信はまだできない。

(6) [本体] Enable/Disable 文で個別の割り込みを許可 / 禁止できるようにした。

(7) [本体] On 割り込み 文で割り込みベクタを設定できるようにした。

(8) [本体] On 割り込み文で指定したサブルーチンを割り込みルーチンとして認識しレジスタの退避と復帰を行うようにした。

(9) [本体] インラインアセンブラの開始と終了に $asm と $end asm が使えるようにした。

(10) [本体] Portb 文で変数の値を代入する場合のコードを改善した。

(11) [本体] I/O レジスタへの代入及び値の参照ができるようになった。

(12) [本体] &Hxx による 16 進定数が使えるようにした。

(13) [本体] Stop TIMER0/COUNTER0/TIMER1/COUNTER1/WATCHDOG 文により各デバイスを停

止できるようにした。

(14) [本体] Start WATCHDOG 文によりウォッチドッグタイマを起動できるようにした。

(15) [本体] 引数なしの Stop 文によりプログラムの停止が行なえるようにした。End 文との違いは割り込みの禁止を行わないことである。

2011.12.7 0.52

(1) [本体] LCD に変数の値を表示できるようにした。ただし、負の数には対応していない。

(2) [mkbachfile.vbs] アセンブルエラーが出たときは空の hex ファイルを生成するようにした。

2011.11.26 0.51

(1) [本体] 簡易プログラミングで Sound 文がコンパイルできなくなる間違いを直した。

2011.11.22 0.50

(1) [本体] Enable Interrupts/Disable Interrupts で割り込みの許可と禁止が行えるようにした。

(2) [本体] ASM_TEXT/ASM_END でインラインアセンブラが使えるようにした。

(3) [Orgel.awk] 入力に対応した。

2011.11.8 0.49

(1) [本体] If 文で a>0 と a<0 の処理が間違っていたのを直した。

(2) [本体] Config Portd.0 の処理が間違っていたのを直した。

(3) [本体] If 文で文字比較の処理が間違っていたのを直した。

(4) [本体] If 文で名前付定数が使えるようにした。

(5) [本体] Config pinB、Config pinB.0、Config pinD、Config pinD.0 が使えるようにした。

2011.11.7 0.48

(1) [本体] Const が使えるようにした。

(2) [本体] Alias が使えるようにした。

(3) [本体] Locate 文で予約されていない変数を使ったときの処理が間違っていたので直した。

(4) [本体] BASCOM-AVR モードで余計な初期化コードが出力されないようにした。

(5) [本体] ATmega168(P) に対応させた。ただし、テストは一切行っていない。

(6) [本体] Config PortX で以下の形式が使えるようにした。

・Config PortB=Input
・Config PortD=Output
・Config PortX.0=Input
・Config PortX.0=Output
・Config PortX=&Bbbbbbbbb

(7) [本体] アセンブリ言語のオペランドで定数を 0b00000000 のように常に 8 桁になるようにした。

(8) [本体] (7) の変更によりオペランドが 0 のときに間違っていたのが直った。

(9) [本体] $regfile が正しく認識され無い場合があったのを直した。

(10) [本体] For 文で Step が使えるようになった (ただし指定できるのは -63 〜 -1、1 〜 63 の定数のみ)。

(11) [その他] 「m168def.inc」を添付した。

2011.10.6 0.47

(1) [本体] ATtiny2313 で a=getadc(0) によって A/D 変換ができるようにした。

ChaN 氏のページ

ELM - シンプルな A-D コンバータ (http://elm-chan.org/docs/avr/avradc_j.html)

からコードを引用した。ただし、クロックが 1MHz のときはあまりうまくは動かないようである。

(2) [本体] a=a*2 の処理が間違っていたのを直した。

(3) [本体] a=int(a/2) も間違っていたので直した。

(4) [本体] PortD=a ができるようにした。

2011.9.22 0.46

(1) [本体] Display On/Off のコードが間違っていたのを直した。

2011.9.15 0.45

(1) [本体] バージョンナンバーと日付以外は特に変更無し。

(2) [Orgel.awk] 12MHz に対応した。

(3) [Orgel.awk] 電子オルゴールのプログラムでないときは「bascompi_orz.bas」を出力しないようにした。

(4) [Orgel.awk] 各音符 (休符) の前に 30ms の待ち時間を入れた。

2011.9.13 0.44

(1) [本体] 再び Windows 7/Vista で動作しなくなっていることが発覚したので対処をした。

(2) [本体] ラベルの処理に関して余計なメッセージが出ていたのを直した。

(3) [本体] Config PortB、Config PortD が処理されていなかったのを直した。

(4) [Orgel.awk] #,b の処理がやっぱりなされていなかったのを直した。

2011.9.12 0.43

(1) [本体] Sound 文のランタイムルーチンでループのラベル位置が間違っていたのを直した。こ

れにより、テンポが遅くなったり、音程が低くなるのが改善したと思われる。

(2) [Orgel.awk] #,b の処理がなされていなかったのを直した。

2011-9-8 0.42

(0) 0.41 の日付が間違っていた。何か良い管理法は無いものか。

(1) [本体] Sound 文の引数でカンマの前後に空白があっても良くなった。

(2) [本体] ラベルの扱いが間違っていたのを直した。「-」を含むラベルも使えるようになった。

(3) [Orgel.awk] 判定法が甘く、Sound 文を含むプログラムを全て誤認していたのを Sound 文を含みかつ Data 文を含んでいて、そのデータが全て電子オルゴールの音符データであるときに「電子オルゴール」のプログラムと看做すことにした。

2011-9-7 0.41

(1) [本体] ソースを少し弄ったが、特に機能的な変更はない。

(2) [Orgel.awk] 電子オルゴールの出力したプログラムをバスコンピのプログラム「bascompi_orz.bas」に変換するスクリプト「Orgel.awk」を作成した。

(3) [mkbatchfile.vbs] 上の「Orgel.awk」のために大幅に書き換えた。

2011-9-6 0.40

(0) 0.39 の日付が間違っていた。

(1) Waitms 文で変数が使えるようになった。

(2) CONFIG、$CRYSTAL でも BASCOM-AVR モードになるようにした。

８ＬＥＤ点滅回路 http://hidapio.jp/m8led.htm

にある、プログラム例の「ヒダピオ」、「BASCOM」の両方がコンパイルできるようになった。

(3) If 文が壊れていたのを直した。

(4) 乗算ルーチン、除算ルーチンのコメントがコメントになっていなかったのを直した。

(5) 代入文の右辺に I、J、K、A 以外の変数も使えるようになった。

(6) If 文で I、J、K、A 以外の変数も使えるようになった。

(7) Sound 文で変数を使った場合のコード生成が間違っていたのを直した。

(8) 入力ファイルの名前が「bascompi*.bas」とき出力ファイル名を「bascompi.asm」にするようにした。

2011-9-5 0.39

(1) 「ddrb 255」の記法を使えるようにした。

(2) Lcd Chr(0) ; "Hello" ; Chr(0)

chr 関数と文字列定数をセミコロンで区切って並べられるようにした。セミコロンの前後に空白があっても良い。

(3) LCDdefchar 文でカンマの前後に空白があっても良くなった。

(4) LCDdefchar 文のコード生成が間違っていたのを直した。

(5) Locate 文の引数でカンマの前後に空白があっても良くなった。

(6) BASCOM-AVR モードでは Next 文の変数を無視するようにした。

(7) Dim 文で BASCOM-AVR モードになるようにした。

(8) For 文で「=」の前後に空白があっても良くなった。

(9) LCD 文で文字列定数の中にカナがあるとアセンブル時に警告がでるが実害はない。

これらによって、

ＡＶＲマイコンのプログラム（ＬＣＤ編）http://hidapio.jp/program6.htm

にあるサンプルがコンパイルできるようになった。

注意 ! If 文が壊れています。

2011-9-2 0.38

・Data &Hxx

一旦、バッファに記録し、ラベルを見つけるかプログラムの最後に到達したときに 2 バイト単位で .db 擬似命令を出力する。これにより本来バイトであるデータがワードになってしまいコードが膨らんでいたのを防ぐことができる。ただし、偶数行の連続した Data 文の後ろにラベル文を置く必要があることに注意する。

・Read a

Dim 文で型情報を記録していたのを使うようにし、Read 文で変数が Byte 型である場合は 1 バイトだけ読み出し変数の下位バイトに格納するようにした。

・これらにより、電子掲示板の全てのパターンがコンパイルできるようになった。

履歴以前の古い版

exe ファイルと zip ファイルの中身は全く同じものです。上の方にあるもの程、新しい版です。

・
・
・
・
・
・
・
・