BASIC Ver 1.0 コマンドリファレンス

1. BASIC 命令文の記述について

   BASIC 命令文を記述する場合は、それぞれの単語は必ずスペース、または、タブで区切ってください。ただし、数式は 1 つの単語として扱いますので、1 つの数式はスペースやタブを入れないでください。

   悪い例)

   ```
   a=10
   if b<>c
   then a=20
   ```

   良い例)

   ```
   a = 10
   if b <> c
   then a = 20
   ```

2. アルファベット表記について

   この BASIC は、変数名やコマンド名などでアルファベットの大文字と小文字を区別しますので注意してください。

3. 変数について

   変数はアルファベット 1 文字 `a`〜`z`、`A`〜`F` で表される合計 32 個が使用できます。この BASIC で使用できる変数型は、整数型と文字型となります。

   整数型は 32 ビット長で `2147483647`〜`-2147483648` の範囲で使用できます。文字型は最大 40 文字までで、それ以上は代入されません。

   BASIC 起動時にはすべての変数が 32 ビット整数型となりますが、文字を代入すると文字型になります。

   ただし、文字型変数は 16 個までとなります。

   もし、17 個以上の変数に文字を代入すると”out of memory”でエラー終了しますので注意してください。

   文字型変数に整数を代入すると整数型変数になります。

4. I/O ポートや物理メモリに対する操作

このBASICでは、H8上のI/Oポートや物理メモリへの操作をサポートしています。

I/Oポートや物理メモリへのアクセスは、変数間接アドレッシングで行ない、あらかじめ、任意の変数にI/Oポートや物理メモリのアドレスを代入しておきます。

”@変数名” でI/Oポートや物理メモリの操作ができます。

5. 定数表記について

10進数の整数は、0〜9 までの数値で指定します。

16進数の整数は、0〜9、a〜f までの数値で指定し、先頭に &H をつけます。

文字列は、”abcdef”のように前後にダブルコーテーションをつけます。

6. 数式表記について

数式は、整数定数・整数型変数・演算子の組合せで表記します。

1つの数式はスペースを入れずに表記しなければなりません。

悪い例

```
29 * (20 + 5 + b) - a
```

良い例

```
29*(20+5+b)-a
```

数式の演算子は以下のものがサポートされています。

- 足し算　：　＋
- 引き算　：　－
- かけ算　：　＊
- 割り算　：　／
- 論理和　：　｜
- 論理積　：　＆

それぞれの演算子の優先順位は以下のようになります。

優先順位	演算子
大	( )
↑	&、|
↓	＊、／
小	＋、－

また、同じ優先順位どうしの演算子の処理順番は左から右となっています。

7. ラベル表記について

goto 文や gosub 文のジャンプ先は行番号で指定することはできません。

この BASIC では、ラベルによって goto 文や gosub 文のジャンプ先を指定します。

ラベル表記は、2〜8 文字定数で指定し、行の先頭から記述し、後尾に必ず”：”をつけなければいけません。

ラベル表記をする場合は単独の行で指定し、ラベル表記のあとに BASIC コマンドを記述してはいけません。

8. 代入文

書式

[変数] ＝ [代入する内容]

変数の型は、代入する内容で決まります。

代入する内容が文字列定数・文字列変数の場合は、代入される変数が文字列変数になります。

代入する内容が整数定数・整数変数、またはそれらを組み合わせた数式の場合は、代入される変数が整数変数になります。

変数間接アドレッシングでの I/O ポートや物理メモリは、8 ビット整数で代入され、代入する内容は、整数定数・整数変数、またはそれらを組み合わせた数式でなければなりません。

32 ビットの整数のうち下位 8 ビットだけが、I/O ポートや物理メモリに代入されます。

H8-3664 のポート 8 への出力例

```
a = &Hffdb
@a = &Hff
```

9. print 文 (pr 文)

   `print [ 表示対象 ]`
   表示対象が整数ならば 10 進数で標準出力に表示します。
   表示対象が文字列ならばその文字を標準出力に表示します。
   ただし、出力後に改行をします。

10. put 文

    `put [ 表示対象 ]`
    基本的には print 文と同じです。
    ただし、出力後に改行をしません。

11. prlong 文

    `prlong [ 数値 ]`
    数値を 32 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。
    ただし、出力後に改行はしません。

12. prword 文

    `prword [ 数値 ]`
    数値を 16 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。
    ただし、出力後に改行はしません。

13. prbyte 文

    `prbyte [ 数値 ]`
    数値を 8 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。

ただし、出力後に改行はしません。

14. putlong 文

    putlong [数値]
    数値を 32 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。
    ただし、出力後に改行はします。

15. putword 文

    putword [数値]
    数値を 16 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。
    ただし、出力後に改行はします。

16. putbyte 文

    putbyte [数値]
    数値を 8 ビットの 16 進数で表示します。
    文字列を指定するとエラーになります。
    ただし、出力後に改行はします。

17. input 文

    input [変数]
    標準入力から値を読み込んで変数にセットします。

18. wait 文

    wait [時間]
    時間待ちをします。
    時間は、0.1 ミリ秒単位で指定します。

19. if〜then〜else 文

`if [ 値 ] [ 条件演算子 ] [ 値 ]`

`then [ BASIC 処理文 ]`

`else [ BASIC 処理文 ]`

if 文と then 文と else 文は別々の行に記述します。

then 文、または、else 文はいずれも省略可能です。

if 文での条件判定は、次の if 文まで有効なので、then 文や else 文は必ずしも if 文の直後にある必要はありません。

条件演算子は以下が使用可能です。

条件演算子	動作	整数型	文字型
＝	一致するか？	○	○
＜＞	一致しないか？	○	○
＞＝	左の値が右以上か？	○	×
＞	左の値が右より大きいか？	○	×
＜＝	左の値が右以下か？	○	×
＜	左の値が右より小さいか？	○	×

使用例 (1)

```
if s = "basic"
then puts "match"
else puts "not match"
```

使用例 (2)

```
if a > 10
then puts "Large."
else puts "Small."
```

使用例 (3)

```
if a = 0
then b = &Hffdb
then @b = &Hff
else b = &Hffdb
else @b = 0
```

使用例 (4)

```
if a = 0
else goto ledoff
  b = &Hffdb
  @b = &Hff
  goto done
ledoff:
  b = &Hffdb
  @b = 0
done:
```

また、if 文の条件を複数指定する場合は以下のように指定します。
使用例 (5)—すべての条件を満たす場合

```
c = &Hffdb
if a = 0
and b = 0
then @c = &Hff
```

使用例 (6)—どれかの条件を満たす場合

```
c = &Hffdb
if a = 0
or b = 0
then @c = &Hff
```

20. goto 文

```
goto [ ラベル ]
```

指定したラベルの場所にジャンプします。

21. for〜next 文

```
for [ 整数型変数 ] = [ 初期値 ] to [ 終了値 ]
next [ 整数型変数 ]
```

for〜next の間にある処理を初期値から終了値の回数だけ繰り返します。
next 文の変数は、必ず、先の for 文で指定した変数でなければなりません。
使用例

```
for i = 1 to 10
  print i
  puts " count."
next i
```

22. gosub〜return 文

gosub [ラベル]

[ラベル] :

return

gosub 文で指定されたラベルのサブルーチンにジャンプします。

サブルーチンの最後で return 命令を実行することにより、呼び出された gosub 文の次に戻ります。

使用例

```
  gosub portout
halt:
  goto halt
portout:
  a = &Hffdb
  @a = &Hff
return
```

23. 右シフト演算

[整数型変数] >> [シフトビット数]

対象となる変数を指定されたビット数だけ右シフトさせます。

シフトビット数は、32 以下の数値を指定しなければエラーになります。

右シフト演算は、論理シフトで行ないます。

24. 左シフト演算

[整数型変数] << [シフトビット数]

対象となる変数を指定されたビット数だけ左シフトさせます。

シフトビット数は、32 以下の数値を指定しなければエラーになります。

25. format 文

    `format`
    AT24C1024 の SEEPROM をフォーマットします。
    SEEPROM は、AT24C1024 でなければなりません。
    SEEPROM をフォーマットしないとプログラムの保存、読みだしができません。

26. save 文

    `save [ ファイル名 ]`
    SEEPROM にプログラムを保存します。

27. romsave 文

    `romsave [ ファイル名 ]`
    H8 内蔵 ROM に書き込まれたプログラムを SEEPROM に保存します。

28. load 文

    `load [ ファイル名 ]`
    SEEPROM からプログラムを読みだします。

29. files 文

    `files`
    SEEPROM に保存されているファイル一覧を表示します。

30. page 文

    `page [ 表示行数 ]`
    表示行数の少ない液晶モジュールでのリスト表示などで、指定された行数ごとに内容を表示するときに指定します。
    表示を次に進める場合は、スペースキーを押します。

31. コメント

    コメントを記述する場合は、文の先頭に ”rem”、” //”、”;” のいずれかを記述します。

32. end 文

    コマンドライン上で実行したプログラムで end 文を実行するとコマンドラインに戻ります。

    ROM 化したプログラムでは意味がありません。

33. list 文 (コマンドライン操作用命令)

    コマンドライン上で編集したプログラムリストを表示します。

    ROM 化したプログラムでは意味がありません。

34. renum 文 (コマンドライン操作用命令)

    コマンドライン上で編集したプログラムの行番号を割り当て直します。

    プログラムの行番号は、先頭から 3 の倍数となります。

    ROM 化したプログラムでは意味がありません。

35. run 文 (コマンドライン操作用命令)

    コマンドライン上で編集したプログラムを実行します。

    実行しているプログラムを途中で止めたい場合は、”Control+C” で止めます。

    ROM 化したプログラムでは意味がありません。

36. romlist 文 (コマンドライン操作用命令)

    H8 内蔵 ROM に書き込まれたプログラムリストを表示します。

    H8-Tiny マイコンの場合は、6000H 番地からプログラムリストを書き込んでおかなければいけません。

    H8-300H マイコンの場合は、10000H 番地からプログラムリストを書き込んでおかなければいけません。

37. romrun 文 (コマンドライン操作用命令)

H8 内蔵 ROM に書き込まれたプログラムを実行します。

実行しているプログラムを途中で止めたい場合は、”Control+C”で止めます。

H8-Tiny マイコンの場合は、6000H 番地からプログラムリストを書き込んでおかなければいけません。

H8-300H マイコンの場合は、10000H 番地からプログラムリストを書き込んでおかなければいけません。

38. new 文 (コマンドライン操作用命令)

RAM 上のプログラムリストを全て消去します。

ROM 化したプログラムでは意味がありません。

39. プログラム編集 (コマンドライン操作用)

`[ 行番号 ] [ BASIC プログラム文 ]`

行番号は、1〜65534 までが使用可能です。

すでに存在する行番号で BASIC プログラム文を記述すると、その行の内容が更新されます。

行番号だけを指定すると、その行の BASIC プログラム文が削除されます。

1 つのプログラム領域の上限は、512 バイトとなっています。