

MSX-BACON(仮)

ユーザーズマニュアル

目次

はじめに.....	3
MSX-BASIC によるプログラミング.....	4
制御命令.....	9

はじめに

MSX-BACON は、MSX-BASIC の文法そのままの記述のプログラムをコンパイルして高速化する BASIC コンパイラです。

MSX-BASIC を「分かりやすい」「とっつきやすい」と感じる方々が多い中、一方で「動作が遅い」という声も多いです。MSX 誕生から 40 年が経過した 2023 年、MSX0 の登場により、再び MSX-BASIC に実用の可能性が出てきたわけですが、「動作が遅い」という部分をもう少し改善したいと思い、BASIC コンパイラの設計に取りかかりました。そのため、MSX0 から追加された IoT-BASIC もサポートしています。ある程度組み上がってきたところで、MSX の創造主である西和彦氏から「今後の MSX0/MSX3 だけでなく、これまでの MSX/MSX2/MSX2+/ MSXturboR を見捨てないようにしたい。MegaRAM カートリッジを製造するので、それに対応させて欲しい。」という話が舞い込んできました。もちろん、私は快諾いたしました。

このような経緯で生まれた MSX-BACON ですので、MSX/MSX2/MSX2+/MSXturboR/MSX0/MSX3 に対応します。MSX3 に関しては、MSXturboR までと互換性のある MSXOS 管理下の部分で動作するもののみ対応です。TaOX 管理下のプログラムは、Python/Cython/C/C++などをご利用下さい。

技術力に自信の無い方、技術にはあまり興味は無いが日常を少し便利にしたい方、技術力はあるがちょっとしたプログラムをササッと作りたい方、昔作った遅いプログラムを速くしたい方、等が主なターゲットとなります。プログラミングに自信の無い方は、まずは MSX-BASIC で組んで動くようになってから、MSX-BACON で高速化して快適に動くように調整する、といった組み方も出来ます。

一方で、ぬるぬる動くアクションゲームを作りたいとか、MSXとは思えない動きのゲームを作りたい等、MSX の限界を突き詰めるような使い方には向きません。そういうケースは、アセンブラでサイクルベースの最適化が必要ですので、アセンブラを使って下さい。MSX-BACON は万能ではありません。

MSX-BASIC の遅さを、ほんの少し改善。よく使う処理は簡単な記述で短時間で書ける。そんなプログラミング環境が、MSX-BACON です。

MSX-BACON には、「MSX-BASIC にこんな命令があったら良かったのに」と思う命令を、独自の命令として追加してあります。それらを使うと、MSX-BASIC では動作させることが出来なくなりますが、ある要件において簡潔に高速に動作する記述ができます。必要に応じてご利用下さい。

MSX/MSX2/MSX2+/MSXturboR/MSX0/MSX3 と全ての MSX に対応するソフトウェアには、特別なロゴの貼り付けが許可されるそうです。MSX-BACON では、その対応のための独自命令（機種判別など）を搭載して、全機種対応ソフトウェアをサポートします。

MSX-BASIC によるプログラミング

1980 年代後半～1990 年代前半くらいのころに MSX を使っていて、MSX-BASIC の画面を見たことがない人はいないとは思いますが、MSX0 をきっかけに始めて MSX に触れる方や、昔の事なんてすっかり忘れてしまった方もおられると思いますので、まずは簡単に MSX-BASIC の使い方について簡単に触れておきます。

MSX-BASIC には、BASIC プログラムを入力するための専用のエディタが内蔵されています。本体の電源を入れると MSX-BASIC の入力待ちの状態になると思います。（※機種によっては、専用のメニューが表示される場合があります。MSXturboR であれば、内蔵ソフトウェアスイッチを OFF にして電源を入れれば OK。その他の機種でも、メニューの中から BASIC を起動する選択をすれば MSX-BASIC の入力待ちの状態になります。）MSX-BASIC の入力待ちの画面は、Fig.1 入力待ち画面のよう表示になります。

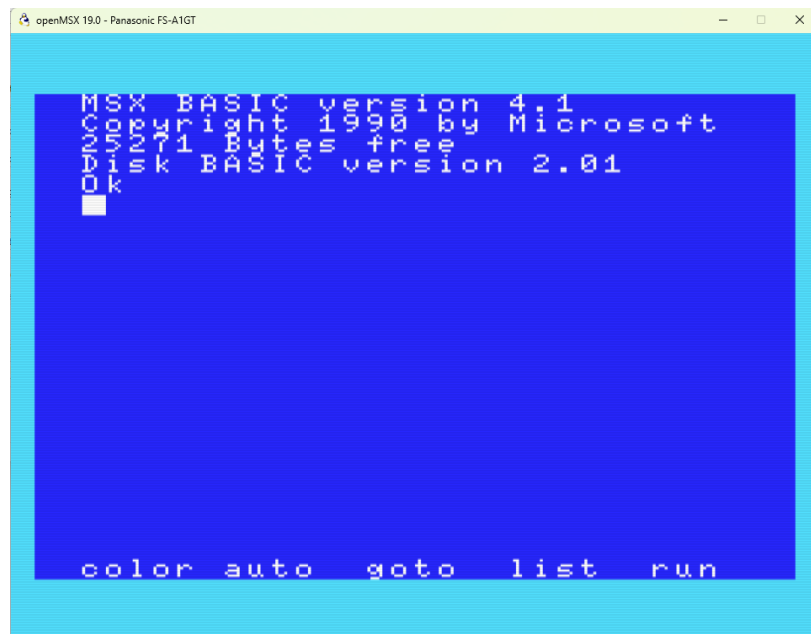


Fig.1 入力待ち画面

ここで PRINT “HELLO!” とタイプして [RETURN]キー (エミュレーターの場合、[Enter]キー)を押してください。Fig.2 HELLO!のような表示が出たと思います。

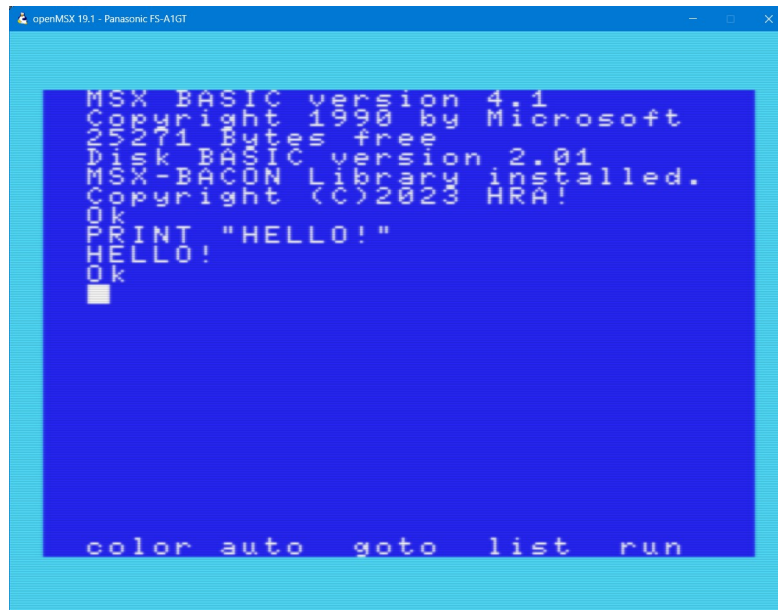


Fig.2 HELLO!

入力した PRINT が、「画面に指定の文字列を表示する命令」になります。BASIC では、この命令を並べていくことでプログラムを記述していきます。”HELLO!” が PRINT の第 1 引数で、「指定の文字列」になります。BASIC では、“ “ (ダブルクォート) で囲んだ部分が文字列と解釈されます。

PRINT “HELLO!” の HELLO! を書き換えて [RETURN]キーを押すと、すぐ次の行に表示される内容も、書き換えた内容に変わることがわかんと思います。

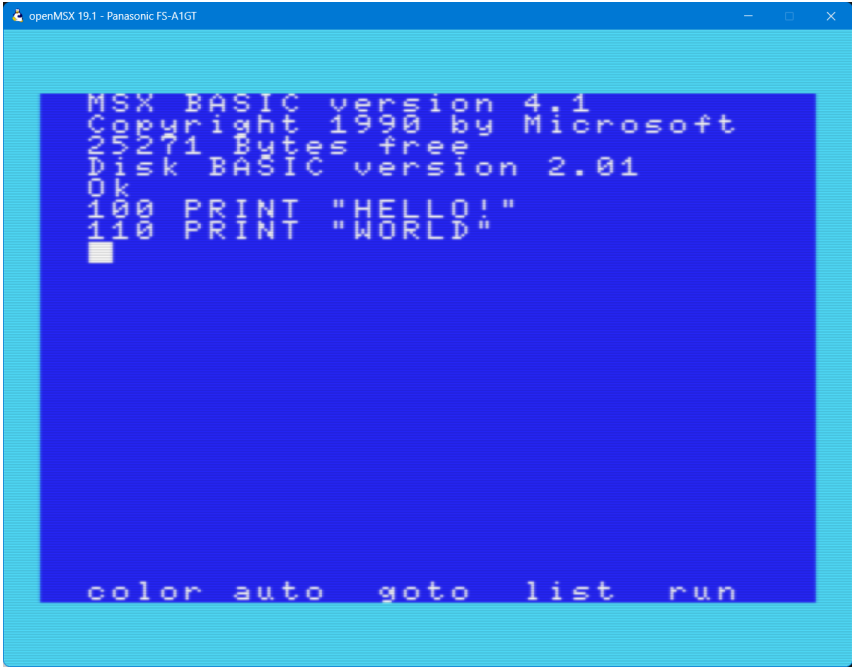
さて、[RETURN]キーを押すとすぐに実行されてしまいますが、どのように命令を羅列してプログラムに組み上げていくかを疑問に思った方もいるかもしれません。[RETURN]キーを押して即時に実行されるのを、今後「即時実行モード」と呼ぶことにします。それに対して、命令を並べてメモリにプログラムとして記憶させることを「プログラムする」と呼ぶことにします。

プログラムする方法ですが、先頭に行番号を記入します。試しに下記のように入力してください。1 行入力するたびに、その行で[RETURN]キーを押してください。その行為でその行が記憶されます。

100 PRINT “HELLO!”

110 PRINT “WORLD”

画面は、Fig.3 プログラムを入力になっています。

A screenshot of the MSX BASIC version 4.1 interface. The window title is "openMSX 19.1 - Panasonic FS-A1GT". The main display area has a blue background with white text. It shows the version information: "MSX BASIC version 4.1", "Copyright 1990 by Microsoft", "25271 Bytes free", and "Disk BASIC version 2.01". Below this, the program being entered is shown: "Ok", "100 PRINT 'HELLO!'", and "110 PRINT 'WORLD'". A white cursor is on the line following "110 PRINT 'WORLD'". At the bottom, a menu bar contains the options "color auto goto list run".

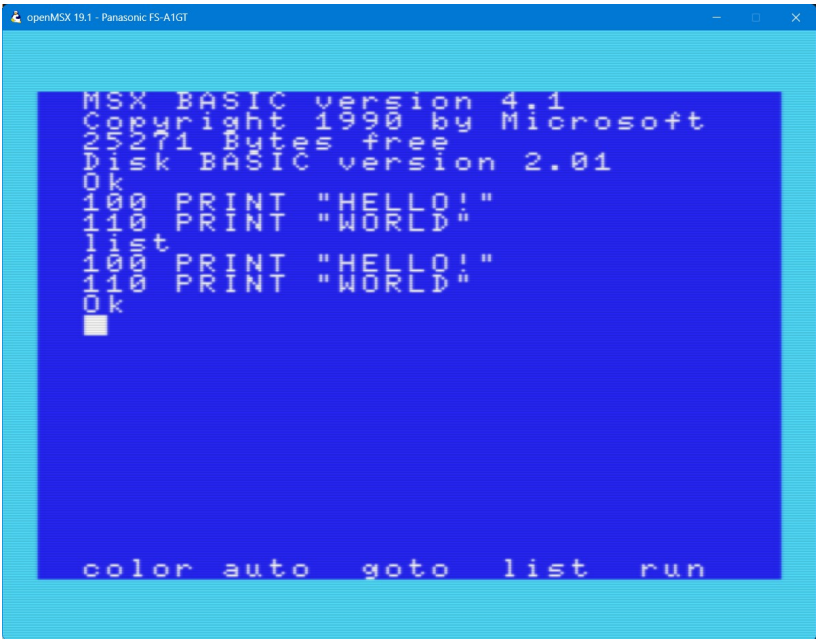
```
MSX BASIC version 4.1
Copyright 1990 by Microsoft
25271 Bytes free
Disk BASIC version 2.01
Ok
100 PRINT "HELLO!"
110 PRINT "WORLD"

```

color auto goto list run

Fig.3 プログラムを入力

入力したプログラムが、正しく記憶されているか確認してみましょう。LIST と入力して [RETURN]キーを押してください。Fig.4 LIST 表示のように記憶されているプログラムが行番号昇順で表示されます。

A screenshot of the MSX BASIC version 4.1 interface after the LIST command has been entered. The window title is "openMSX 19.1 - Panasonic FS-A1GT". The main display area has a blue background with white text. It shows the same version information as Fig.3. Below that, the program is listed in ascending order of line numbers: "Ok", "list", "100 PRINT 'HELLO!'", "110 PRINT 'WORLD'", and "Ok". A white cursor is on the line following the second "Ok". At the bottom, the same menu bar "color auto goto list run" is visible.

```
MSX BASIC version 4.1
Copyright 1990 by Microsoft
25271 Bytes free
Disk BASIC version 2.01
Ok
list
100 PRINT "HELLO!"
110 PRINT "WORLD"
Ok

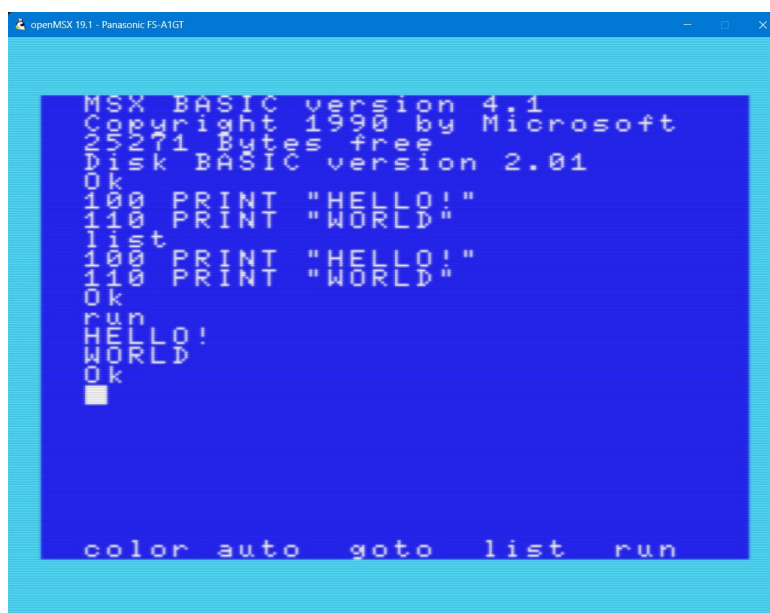
```

color auto goto list run

Fig.4 LIST 表示

LIST は、記憶したプログラムを表示する命令です。BASIC の命令は、アルファベットの大小に区別がありません。LIST のかわりに list でも、List でも、LiSt でも同じ動作をします。もちろん、PRINT も Print でも pRiNt でも同じ動作をします。ただし、プログラムとして記憶するのは、大文字に成形されます。例えば、100 print と入力しても、100 PRINT に置き換わります。

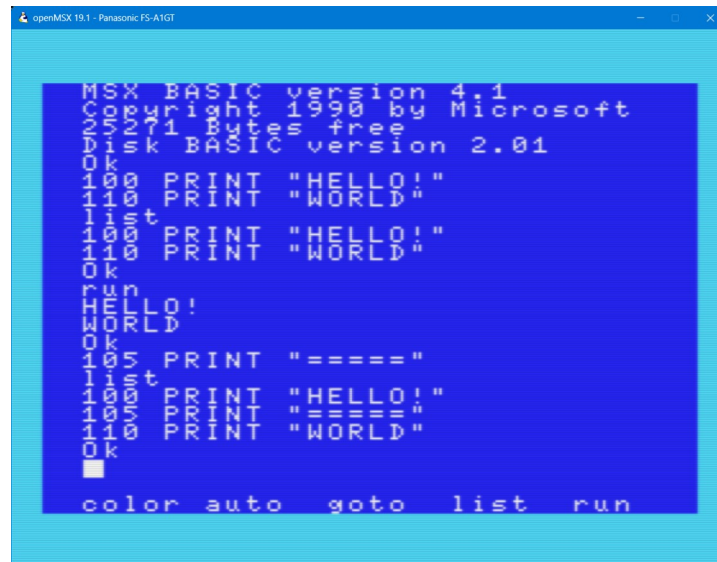
では、入力したプログラムを実行してみましょう。実行には、RUN 命令を即時実行モードで実行すると、そこから記憶されているプログラムが実行されます。では、RUN と入力して [RETURN] キーを押してください。



```
MSX BASIC version 4.1
Copyright 1990 by Microsoft
255271 Bytes free
Disk BASIC version 2.01
Ok
100 PRINT "HELLO!"
110 PRINT "WORLD"
list
100 PRINT "HELLO!"
110 PRINT "WORLD"
Ok
run
HELLO!
WORLD
Ok
color auto goto list run
```

Fig.5 実行

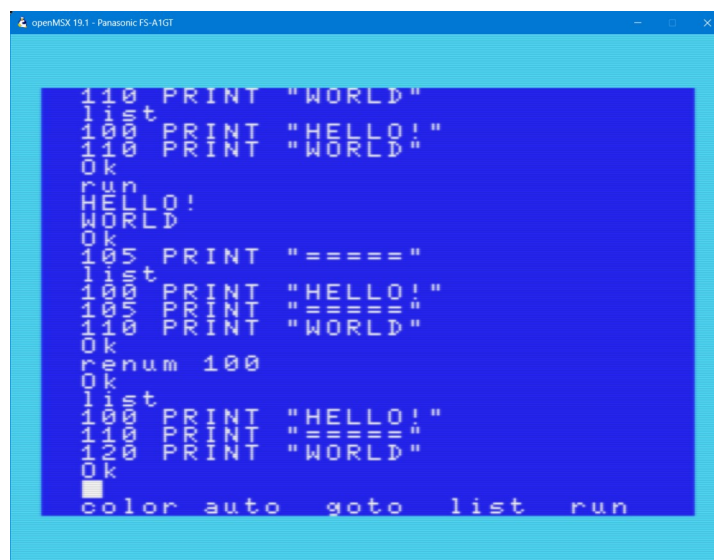
Fig.5 実行のように表示されたと思います。プログラムは、行番号の昇順に並べられて順番に実行されます。先ほどの例では、行 100 で HELLO! と表示し、行 110 で WORLD と表示します。PRINT 命令は、通常改行をつけるので、HELLO! を表示して改行した次の行に WORLD が表示されます。間の行番号を入力すると、自動的に行番号昇順になるように並べ替えられます。試しに、105 PRINT "====" と入力して [RETURN] キーを押してみてください。LIST すると、今入力した行 105 が、Fig.6 行の挿入のように行 100 と行 110 の間に入っているのを確認できます。このように、あとから追加しやすいように、行番号は 10 程度の間隔をあけて振っておくことをお奨めします。



```
MSX BASIC version 4.1
Copyright 1990 by Microsoft
25271 Bytes free
Disk BASIC version 2.01
Ok
100 PRINT "HELLO!"
110 PRINT "WORLD"
list
105 PRINT "HELLO!"
115 PRINT "WORLD"
Ok
run
HELLO!
WORLD
Ok
105 PRINT "===="
list
100 PRINT "HELLO!"
105 PRINT "===="
110 PRINT "WORLD"
Ok
color auto goto list run
```

Fig.6 行の挿入

挿入を繰り返して、行番号の並びが詰まり過ぎて、挿入するのに困ってしまった場合。RENUM 100 を実行してください。Fig.7 RENUM 100 の実行結果のように、行番号が振り直されます。この場合、自動的に 10 間隔になります。



```
110 PRINT "WORLD"
list
100 PRINT "HELLO!"
110 PRINT "WORLD"
Ok
run
HELLO!
WORLD
Ok
105 PRINT "===="
list
100 PRINT "HELLO!"
105 PRINT "===="
110 PRINT "WORLD"
Ok
renum 100
Ok
list
1100 PRINT "HELLO!"
1150 PRINT "===="
1200 PRINT "WORLD"
Ok
color auto goto list run
```

Fig.7 RENUM 100 の実行結果

このように、行番号のついた命令を追加・修正することによって、プログラムを入力していきます。

あとは、どのような命令があるのか、覚えていく必要がありますが、すべての命令を覚える必要はありません。自分のやりたいことに必要な命令を優先的に覚えていけばいいのです。その中でも、プログラムの流れを制御する命令は、どのようなプログラムを作るにせよ必要になる基本的な命令となりますので、まずはその制御命令を覚えてください。次章では、基本的な制御命令について説明します。

制御命令

BASIC の制御命令はたくさんありますが、特に下記の制御命令はよく使う基本的な制御命令となります。これを覚えておけば、とりあえずどんなプログラムも組めます。ここにはない制御命令は、ここにある制御命令の組み合わせでも実現できますので、無理して覚える必要はありません。余力があれば覚えておくと「簡潔に記述できる」程度のもので、覚える命令を少なく済ませたい場合は、まずここに記載の命令を覚えてください。

- ・ GOTO
- ・ IF 条件式 THEN 文1 ELSE 文2
- ・ GOSUB
- ・ RETURN
- ・ END

では、以下で順に説明していきます。