SANYO

パーソナルコンピュータ　拡張BASIC説明書 MSX2+

WAVY シリーズ

WAVY

はじめに

この拡張ＢＡＳＩＣ説明書は、ＭＳＸおよびＭＳＸ2をさらに機能アップしたＭＳＸ2+およびＭＳＸ－ＭＵＳＩＣ、ＢＡＳＩＣコンパイラのＢＡＳＩＣ命令の説明書です。
ＭＳＸ2+またはＭＳＸ－ＭＵＳＩＣ、ＢＡＳＩＣコンパイラのどの機能が装備されているかは、お買い求めのパソコンによって異なります。別冊の取扱説明書をお読みいただき、各機能があるかを確かめてください。

この説明書をご使用になる前に、つなぎ方、基本的な操作方法について、別冊の取扱説明書をよくお読みください。
また、ＢＡＳＩＣコンパイラを使う前にＢＡＳＩＣ命令の使い方とプログラムの作り方を覚えておいてください。プログラムの作り方については別冊の「ＭＳＸプログラミング入門」をお読みください。

目　次

この拡張ＢＡＳＩＣマニュアルは、次の４章に分けて使い方を説明しています。目的に合わせてお読みください。

ＭＳＸ２＋（プラス）の使い方

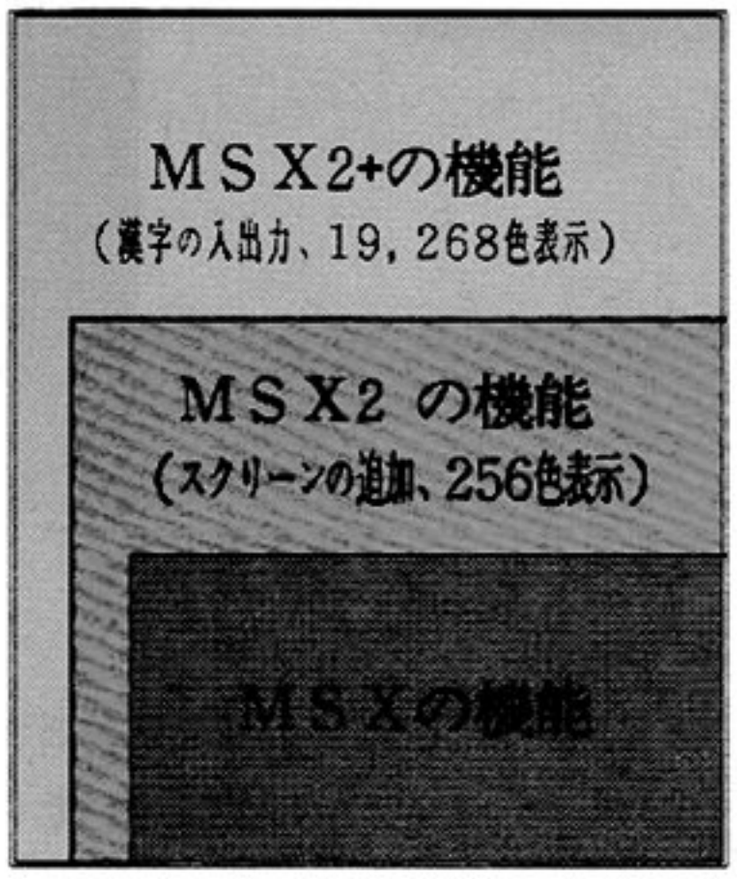
この章では、ＭＳＸおよびＭＳＸ2に対して追加･変更されたＭＳＸ ＢＡＳＩＣ Ｖｅｒ．３の拡張機能について説明します。
漢字入力、多彩なカラー表示など、ＭＳＸ2+パソコンの機能をフルに使うためにＢＡＳＩＣ命令の使い方を覚えてください。

ＭＳＸ２＋（プラス）ってなに？　－こんなことができます－

別冊の「ＭＳＸプログラミング入門」ではＭＳＸ　ＢＡＳＩＣの命令の基本的な使い方を説明しています。この章ではＭＳＸ、ＭＳＸ2 に追加・変更されたＭＳＸ2+ の機能を説明します.

ＭＳＸ2+ は、ＭＳＸ2 に漢字の入出力機能や 19,268色のカラー表示機能などを追加した新しいＢＡＳＩＣです。

主に次の機能が追加されています。



● 漢字の入出力ができます。
ＪＩＳ第１水準の漢字ＲＯＭを標準で備えています。単漢字変換による漢字混じりのプログラム入力や、ＰＲＩＮＴ命令による漢字の表示が簡単にできます。

● 多彩なカラー表示ができます。
スクリーンモードの追加により、最高19,268色の表示ができます。自然画の表示もよりリアルに行えます。

● 表示画面を自由にスクロールできます。
画面のスクロール（移動）が、縦方向に加えて横方向にもできます。ゲームなどで縦、横、斜め自由に移動できます。

このほかにも、漢字対応プリンタに、漢字混りのプログラムリストを出したりすることもできます。
また、ＭＳＸ2+ のオプション機能（標準ではなく、パソコンごとに追加できる機能）として、ＪＩＳ第２水準の漢字ＲＯＭへの対応や、ＭＳＸ－ＭＵＳＩＣのＢＡＳＩＣ命令にも対応できます。

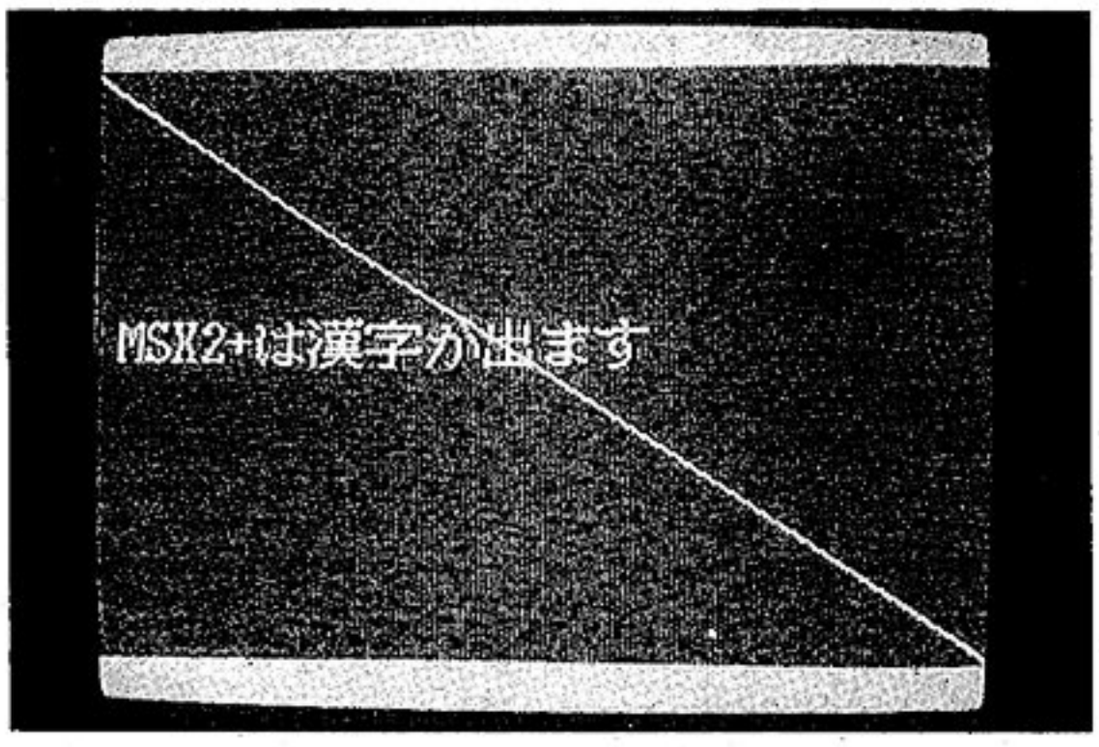
こんなことができます

漢字の入出力

● 漢字混りのプログラム

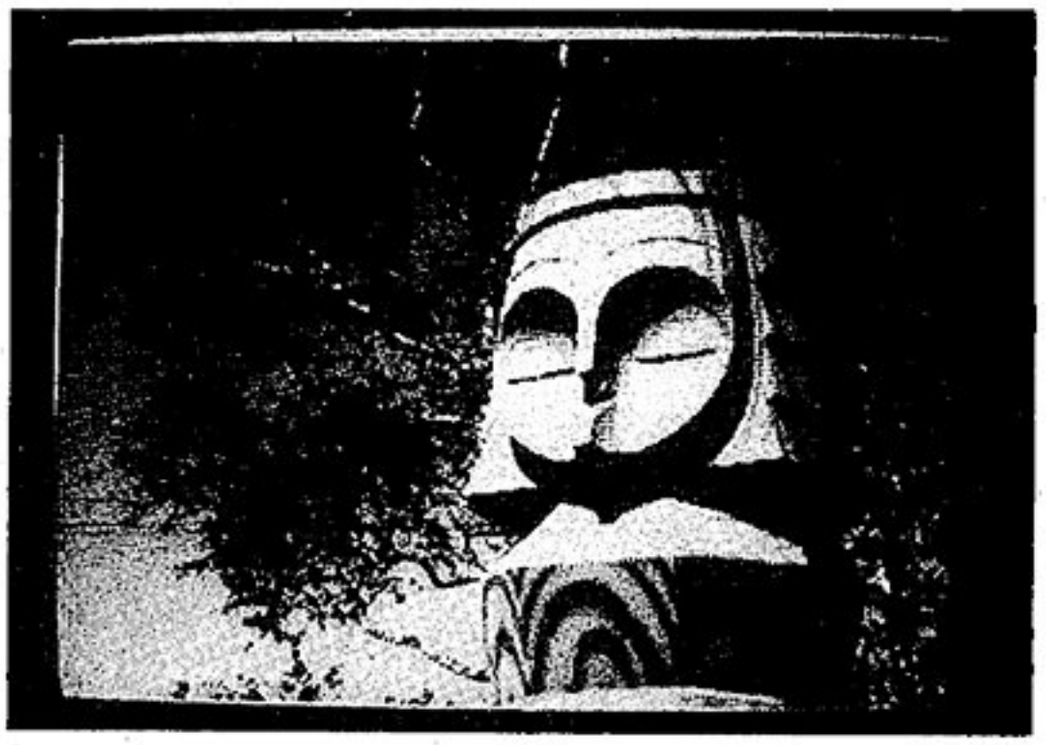
漢字のプログラム入力、プリンタへの出力ができます。

● 漢字とグラフィックの混在



漢字をグラフィック画面に重ねて表示できます。

19,268色の表示



自然画をリアルに再現できます。

画面の水平スクロール

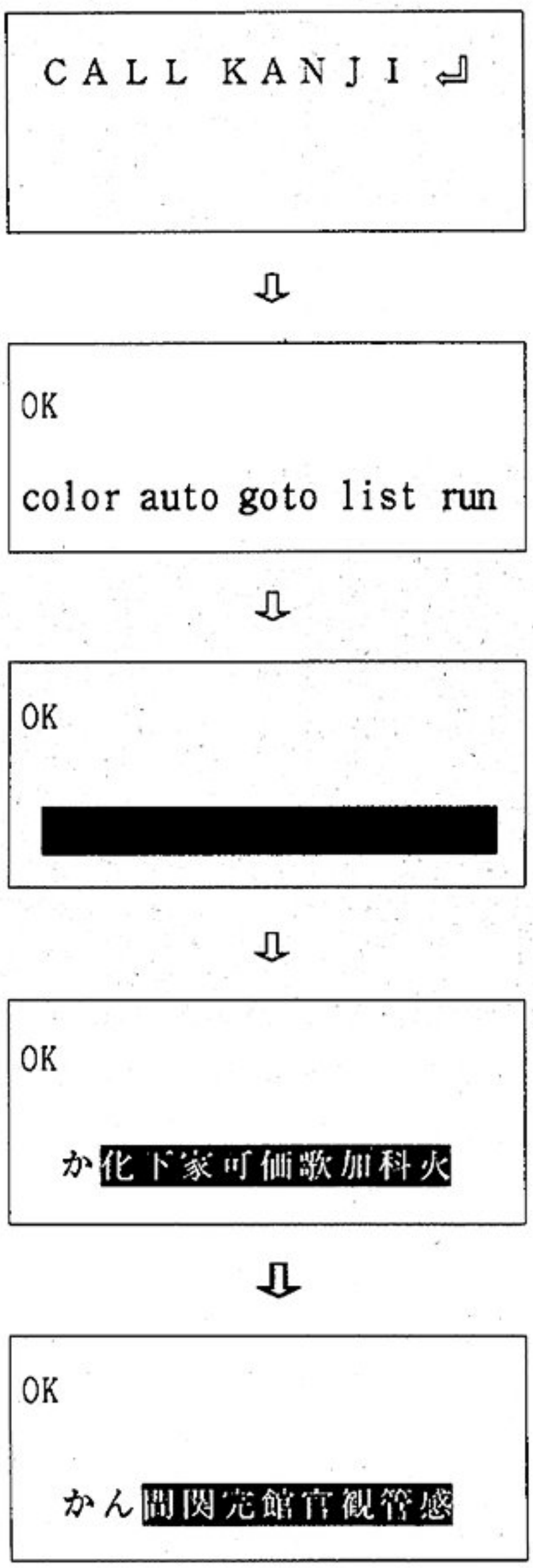
縦、横、斜めに画面を移動できます。

漢字の表示

ＭＳＸ2+パソコンでは、ＪＩＳ第一水準の漢字ＲＯＭを標準で備えています。テキスト画面で単漢字変換(１つの漢字の音読みを入力して漢字にする)を使った漢字の入力ができ、漢字を使ったプログラムの作成ができます。また、漢字対応のプリンタに漢字出力ができます。(第二水準の漢字ＲＯＭは、標準装備ではありません。ＭＳＸ2+のオプション装備です。機器の仕様を確かめてください)。

漢字入力のしかた －「漢字」と入力するとき－

1 ＣＡＬＬ（コール） ＫＡＮＪＩ（カンジ） ↵(リターンキー)と入力します。(漢字モード)
画面がクリアされ、"ＯＫ"という文字が漢字モードで大きく表示されます。
"ＣＡＬＬ"の代わりに"_(アンダーバー)"でもできます。
(_ＫＡＮＪＩ ↵)



2 CTRL（コントロール）キーを押しながらスペースキーを押します。(単漢字変換モードになります)
画面の下に変換テーブルと呼ぶ白色の帯が表示されます。(文字の表示色(COLORの色)を変えるとこのテーブルの色も変わります)。

3 かなキーを押して、かな入力の状態にします。(ローマ字カナ入力もできます)。

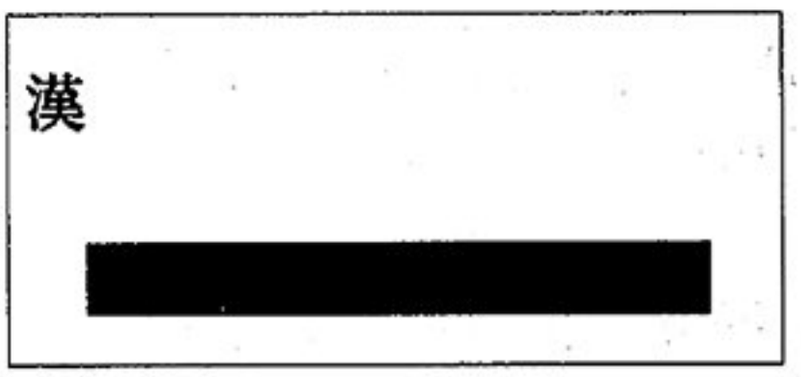
4 次に、「か」と入力します。
画面の下にひらがなの「か」と、「か」で始まる漢字が表示されます。音読みが「か」の漢字が反転して表示されます。

5 「ん」と入力します。
画面の下にひらがなの「かん」と、「かん」で始まる漢字が表示されます。音読みが「かん」の漢字が反転して表示されます。

6 変換テーブル内のカーソルをカーソルキーを押して、目的の文字に移動します。
もし、目的の漢字がないときは、カーソルキー⇧または⇩を押すと別の変換テーブルが表れます。（次ページの「ちょっと一言」を参照）

⇩

7 ↵(リターンキー)を押すと、「漢」が画面に表示されます。（変換テーブル内の文字はクリアされます）。

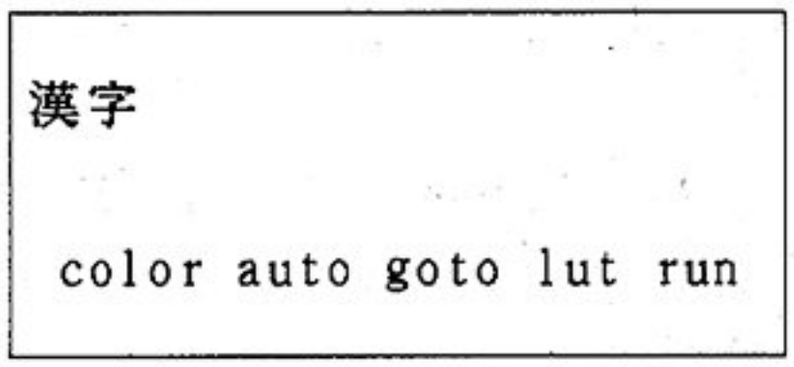


⇩

8 同様にして手順**4**～**7**の要領で「じ」を入力して、漢字の「字」を表示させます。

⇩

9 CTRLキーを押しながらスペースキーを押します。
変換テーブルが消え、漢字入力を終了します。



ちょっと一言

■単漢字変換とは

ワープロやパソコンでは直接漢字を入力できないので、漢字の読み仮名を入力して漢字に置き換え(変換)します。単漢字変換とは、漢字を単一(一文字づつ)入力して、置き換える方法のことです。

■漢字を表示させる命令

漢字を表示させる命令として、次の命令があります(２６ページを参照)。

BASIC命令	CALL KANJI CALL KANJI0	CALL KANJI1	CALL KANJI2	CALL KANJI3
ドット数	16X16ドット	12X16ドット	16X16ドット	12X16ドット
画面表示	1画面表示 (ノンインターレス)	1画面表示 (ノンインターレス)	2画面交互表示 (インターレス)	2画面交互表示 (インターレス)

記号入力のしかた －☆を入力するとき－

"●"や"★"などの記号には読み方がないので、漢字コードで指定します。
漢字コードは別冊の「MSXプログラミング入門」の「5.資料」にある、ＪＩＳ漢字コード表の番号（４ケタ）です。

1 ＣＡＬＬ ＫＡＮＪＩ ↵(リターンキー)と入力します。

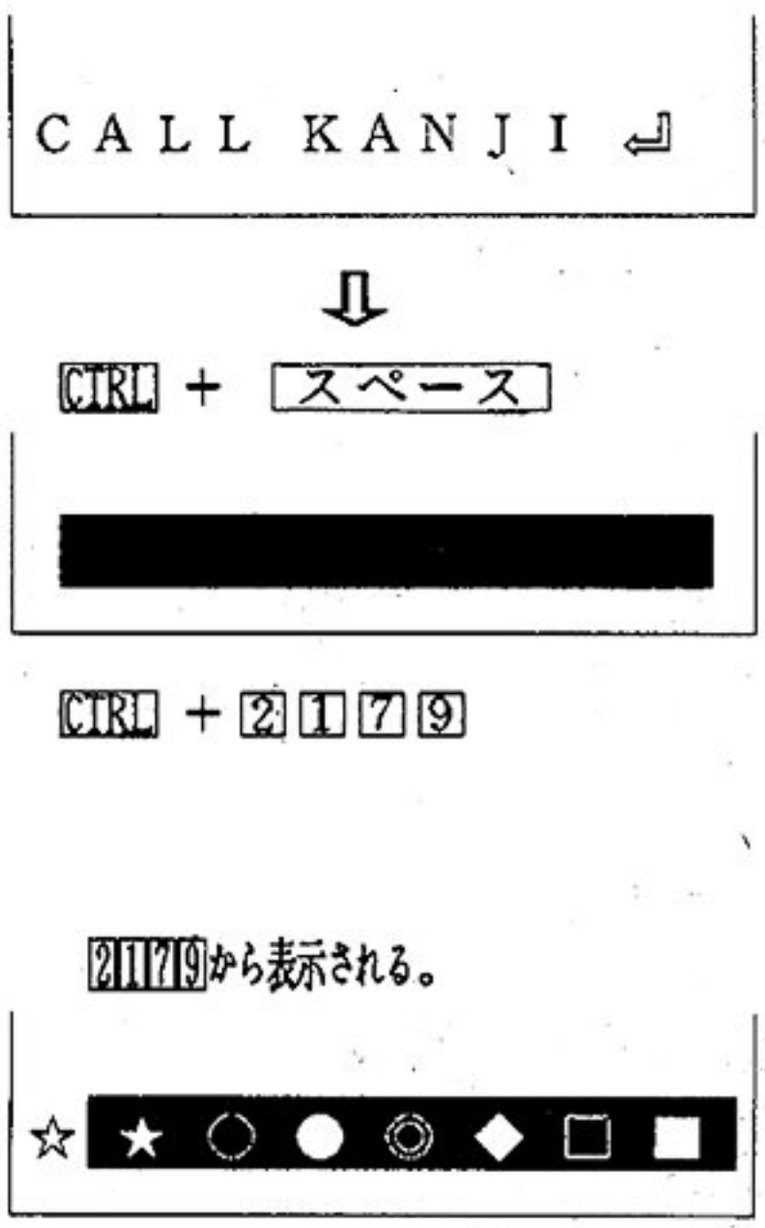
2 CTRLキーを押しながらスペースキーを押します。（単漢字変換モードになります）

3 CTRLキーを押しながら2179と押します。（漢字コードの入力）
画面の下に図のように記号が表示されます。

4 変換テーブル内のカーソルをカーソルキーで移動させ、目的の記号に合わせます。

5 ↵(リターンキー)を押すと、記号が画面に表示されます。（変換テーブル内の記号はクリアされます）

6 CTRLキーを押しながらスペースキーを押します。



ちょっと一言

■変換テーブルに表示される文字数

漢字の読みを入力したときや記号の漢字コードを入力したとき、変換テーブルに漢字が表示されます。一度に表示される文字数は、画面の表示文字数で決まり、CALL KANJI3、SCREEN 0でWIDTH 80のときが最大で、１５文字です。
また、漢字コードを入力したときなど、変換テーブルに表示しきれない文字はバッファ（数値やデータを一次的に記憶しているところ）に貯えられ、カーソルキーを押してカーソルを移動させると順に表示されます。一つの読み(コード)に対して変換テーブルに表示される文字の合計数は最高１２８文字です。

こんなこともできます

■ 文字の読み仮名

単漢字変換は、漢字の音読みを入力して漢字に変換しますが、一部の文字については訓読みにも対応しています。
（例：「川」は、「かわ」「せん」どちらでも変換できます。「木」、「髪」も同じ。）

■ 単漢字変換モードの開始

単漢字変換モードは CTRL ＋ スペース のほかに、 GRAPH ＋ SELECT でも開始・終了できます。

■ 読み仮名を入れ直すとき

BS キーで１文字分、読み仮名の入力を削除できます。 ESC キーを押すと、読み仮名を含めて変換テーブルがクリアされます。読み仮名を入力してください。

■ 漢字プリンタへの出力

漢字モード(CALL KANJI↵)でLlist命令などを実行すると、漢字で出力できます。

■ 入力できる文字の種類

文字の種類	大きさ	キー・ランプの状態
ひらがな	全角(かな)	かな ランプのみが点灯。
カタカナ	全角(カナ)	かな 、CAPS のランプが点灯。単漢字変換モード。
	半角(ｶﾀｶﾅ)	かな 、CAPS のランプが点灯。
記号	全角(●☆)	単漢字変換モードで、CTRL ＋漢字コードを入力。
英数字	全角(Ａ５)	かな ランプが消灯。単漢字変換モード。
	半角(A5B4)	かな ランプが消灯。
漢字	全角(二弐)	６ページからの方法で入力します。第二水準の漢字については、漢字ＲＯＭを備えていない場合、市販の漢字ＲＯＭカートリッジが必要です。

ひらがな、カナ、ローマ字カナ変換については、取扱説明書をご覧ください。

プログラム中の漢字モード

ＣＡＬＬ　ＫＡＮＪＩ命令は、プログラムの中でも実行できます。また、漢字の入力(INPUT命令)も、テキストモードで行うことができます。

サンプルプログラム

```
10 CALL KANJI
20 INPUT "名前は";A$
30 PRINT A$
40 FOR T=0 TO 200:NEXT T
50 SCREEN 5
60 LOCATE 10,10
70 PRINT A$
80 GOTO 80
```

止めるときは CTRL ＋ STOP キー

上のプログラムを実行すると、"名前は？"と入力待ちの状態となります。
漢字モードとなっていますので、CTRL キーを押しながらスペースキーを押して、単漢字変換モードにします。次に、漢字で名前を入力してください。
テキスト画面に表示した後、グラフィック画面に名前を表示します。

プログラムを作るときの注意

漢字を使ってＢＡＳＩＣプログラムを作るときの注意点を説明します。

■ プログラムの始めにＣＡＬＬ（コール） ＫＡＮＪＩ（カンジ）

ＣＡＬＬ ＫＡＮＪＩ命令は、プログラム中に何回でも実行できます(文字の種類を変えたり、漢字モードを終わることもできます)。最初に使うＣＡＬＬ ＫＡＮＪＩ命令は、できるだけプログラムの初めの行に入れてください。これは、ＣＡＬＬ ＫＡＮＪＩ命令の実行により、漢字変換用のワークエリア(漢字変換をするために必要なメモリ)を確保するためです。このため、プログラムや変数に使用できるメモリが少し減ります(約550バイト)。また、最初のＣＡＬＬ ＫＡＮＪＩ命令の実行で、変数、FOR~NEXT、GOSUB 命令用のメモリもクリアします。つまり、最初のＣＡＬＬ ＫＡＮＪＩ命令を実行する前に、変数、FOR~NEXT文、GOSUB 命令を使うプログラムを作ってはいけません。
一度ワークエリアを確保すると、２回目以降のＣＡＬＬ ＫＡＮＪＩ命令の実行では再び確保する必要がないのでメモリのクリアはされません。

■ 漢字モードの終了

ＣＡＬＬ ＡＮＫ（アンク） ↵(リターンキー)とキー入力すると、通常の英数カナ表示になります。ただし、ワークエリアのメモリは確保した状態です。
プログラム中でも実行できます。
(ＡＮＫ：英数カナの意味。Alphabet(英字)、Numeric(数)、Kanaの頭文字をとった用語です)

■ ＭＳＸ，ＭＳＸ2 のＢＡＳＩＣ命令との違い

一部のＢＡＳＩＣ命令が漢字モードで使えない、または働きが異なります。
また、追加された命令もあります。

命令について詳しくは、20ページからの説明をご覧ください。

ＢＡＳＩＣ命令	働　　き
CALL AKCNV	文字列の中の文字を全角文字列にします。
CALL ANK	漢字モードを終了します。
CALL CLS	画面をクリアします。
CALL JIS	文字列のＪＩＳコードを文字列として求めます。
CALL KACNV	文字列の中の文字を半角文字列にします。
CALL KANJI	漢字モードを起動する。
CALL KEXT	文字列の中から半角(全角)文字だけを抜き出します。
CALL KINSTR	文字列の中から指定した文字列を探します。
CALL KLEN	文字列の中の全角(半角)の文字数を得ます。
CALL KMID	文字列の中から指定された位置の文字を抜き出します。
CALL KNJ	文字列で指定された漢字コードの漢字を得ます。
CALL KTYPE	文字列の中の指定された文字のタイプ(全角/半角)を得ます。
CALL PALETTE	カラーパレットの初期化または設定を行ないます。
CALL SJIS	文字のシフトＪＩＳコードを得ます。
CLS	画面をクリアします。
COLOR	カラーパレットの初期化または設定を行ないます。
PUT KANJI	ＳＣＲＥＥＮ５～１２で漢字を表示します。
WIDTH	１行に表示する文字数を設定します。

多彩なカラー表示

ＭＳＸ2+では、ＭＳＸ2に次の画面表示の機能が追加されています。

- 最大19,268色のカラー表示できる画面モード
- 画面の水平・垂直スクロール

画面の水平・垂直スクロールについては、３６ページからのコマンド説明を参照してください。この項では、カラー表示について説明します。

人物や風景などの自然画の再現には、厳密には無限の色数が必要です。このためパソコンで自然画の再現をするとき、表示できる色の数が多いほどリアルな画像となります。
ＭＳＸ2 では最大２５６色の表示でしたが、ＭＳＸ2+ では最大１９,２６８色の再現が可能です。

スクリーンの種類

多彩なカラー表現ができる画面モードは、ＭＳＸ2+ で追加された次の３つのモードです(ＳＣＲＥＥＮ ９はありません)。

ＳＣＲＥＥＮ １０：ＹＪＫ方式による12,499色の表示と、ＲＧＢ方式の16色(512色中)を同時に表示できます。カラーコードに０～１５が指定できます。

ＳＣＲＥＥＮ １１：ＹＪＫ方式による12,499色の表示と、ＲＧＢ方式の16色(512色中)を同時に表示できます。ＳＣＲＥＥＮ １０との違いは、カラーコードに 0～255 が指定でき、ＲＧＢ方式とＹＪＫ方式のデータを扱うことができます。

ＳＣＲＥＥＮ １２：ＹＪＫ方式による19,268色を同時に表示できます。

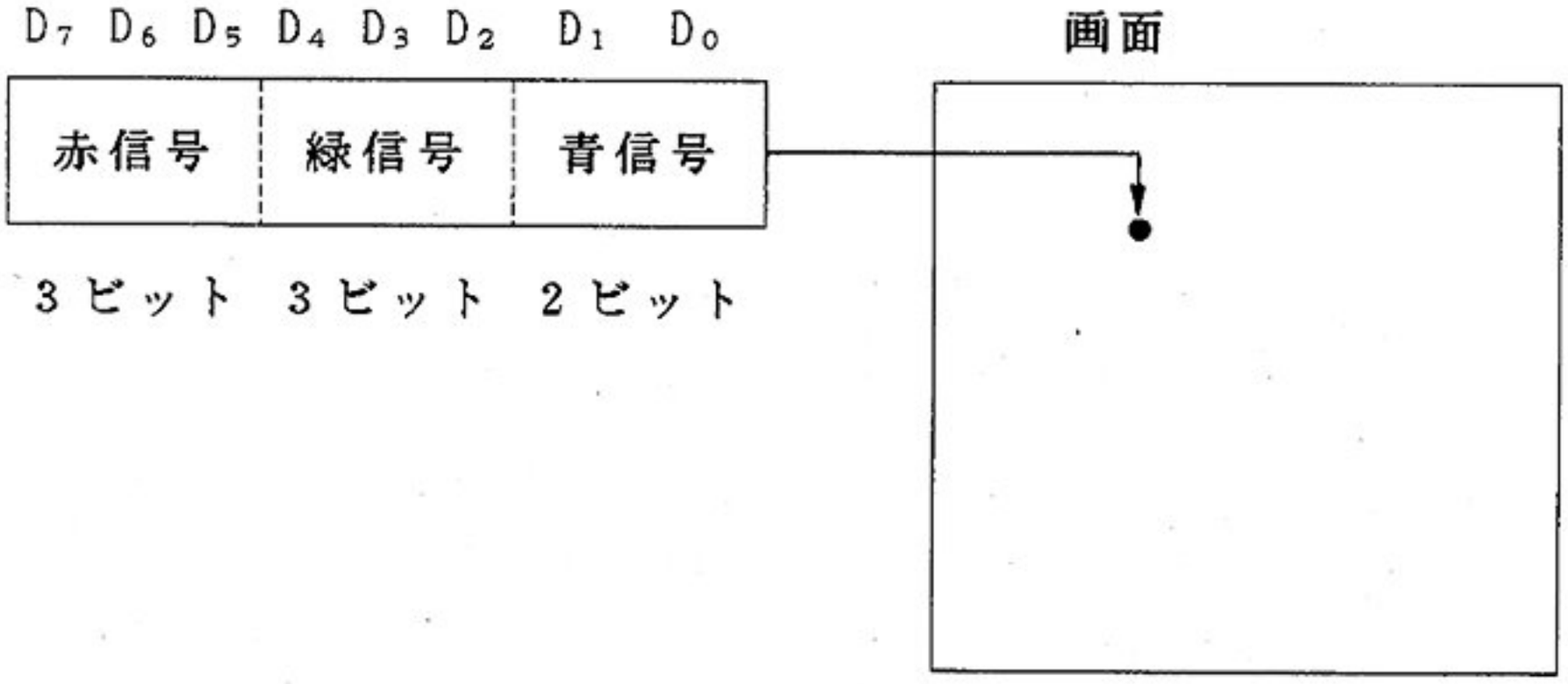
ＹＪＫ方式については、次ページからの説明をご覧ください。
ＳＣＲＥＥＮ １０～ＳＣＲＥＥＮ １２には、ＳＣＲＥＥＮ ８と同じ２５６×２１２ドットで画面が構成されます。

色の再現方法　－ＹＪＫ方式とは－

色を再現する方法には、赤・緑・青の色を混合して再現するＲＧＢ方式と、明るさ・色あい・色の濃さによって再現する色差方式があります。色差方式は、一般のテレビやビデオで使われている信号の方法です。ＭＳＸ2+では、パソコン用の色差方式としてＹＪＫ方式で色を表現することができます。

ＲＧＢ方式について

ＲＧＢ方式を、ＳＣＲＥＥＮ　８を例に説明します。
ＳＣＲＥＥＮ　８の画面では、画面上の１点(１ドット)がパソコンの１バイトのメモリ(ＶＲＡＭ)に対応します。１バイトは、赤信号３ビット・緑信号３ビット・青信号２ビットの８ビットデータで構成されています。
このため再現できる色の種類は、$2^3 \times 2^3 \times 2^2 = 256$色 となります。



ＲＧＢ方式は、赤･緑･青の明るさを変えることで色を再現します。
画面上の１点ごとに直接色をコントロールできるので、プログラムを作るのには便利ですが、多くの色を再現しようとすると、パソコンのメモリ(ＶＲＡＭ)が多く必要となります。

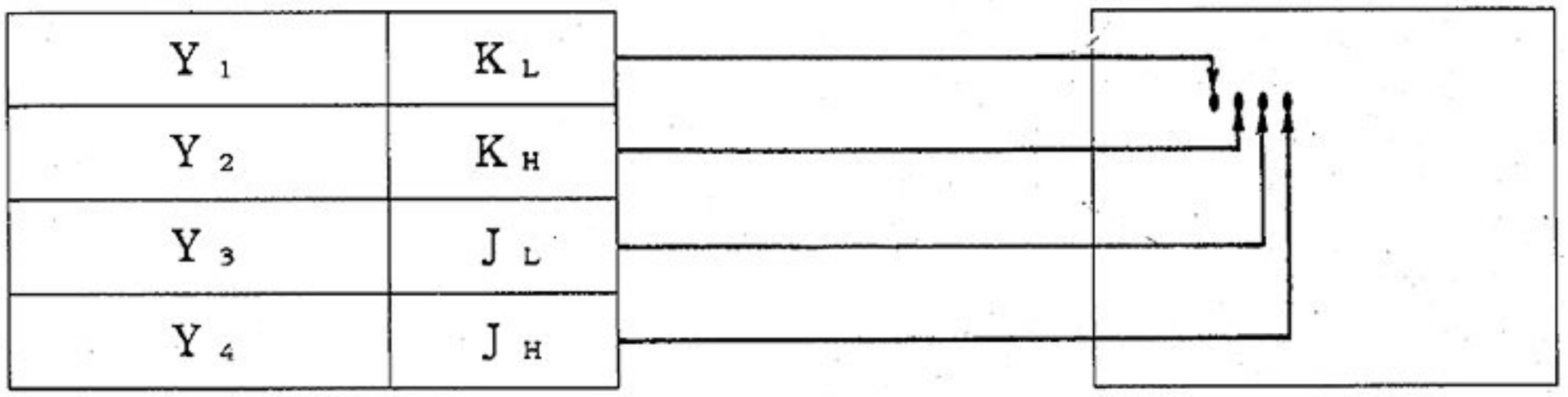
ＹＪＫ方式について

ＹＪＫ方式は、画面上の１点ごとに明るさのデータを持ちながら横４ドットの色を１つのデータとして扱います。これは、人間の目の性質として、明るさの違いについては敏感ですが、色の違いについては感度が低いことを利用しています。つまり、色の付け方はおおまかでも、明るさが細かく表現できればきれいに見えることになります。
これにより、画面の解像度はＳＣＲＥＥＮ　８と同じ２５６×２１２ドットでありながら、原理的に約１３万色($2^5 \times 2^6 \times 2^6 \fallingdotseq 13$万)を表示することができます(実際には、目に見える色の違いとして19,268色)。

ＹＪＫ方式のデータ（ＳＣＲＥＥＮ　１２の場合）

ＹＪＫ方式は横256ドット×縦212ドットの画面モードで、横方向に連続した４ドットを一組として色を再現します。

画面上の連続した４ドットのデータの内容は、次のように構成されています。



（SCREEN 12のとき）

Ｙは明るさ（輝度）のデータです。Ｋ、Ｊは色あいのデータです。K_LはＫの下位３ビット、K_HはＫの上位３ビットです。Ｊも同じです。

各ドットごとに独立してＹのデータを５ビット持ち、横４ドット共通としてＫのデータを６ビット、Ｊのデータを６ビット持っています。これにより、一つの点の色はＹ（５ビット）×Ｋ（６ビット）×Ｊ（６ビット）の組合わせで再現されます。

各点のＹＪＫのデータから画面に再現されるＲＧＢの色への変換は、パソコンのＶＤＰの内部で次の処理を行います。

$$R = Y + J$$

$$G = Y + K$$

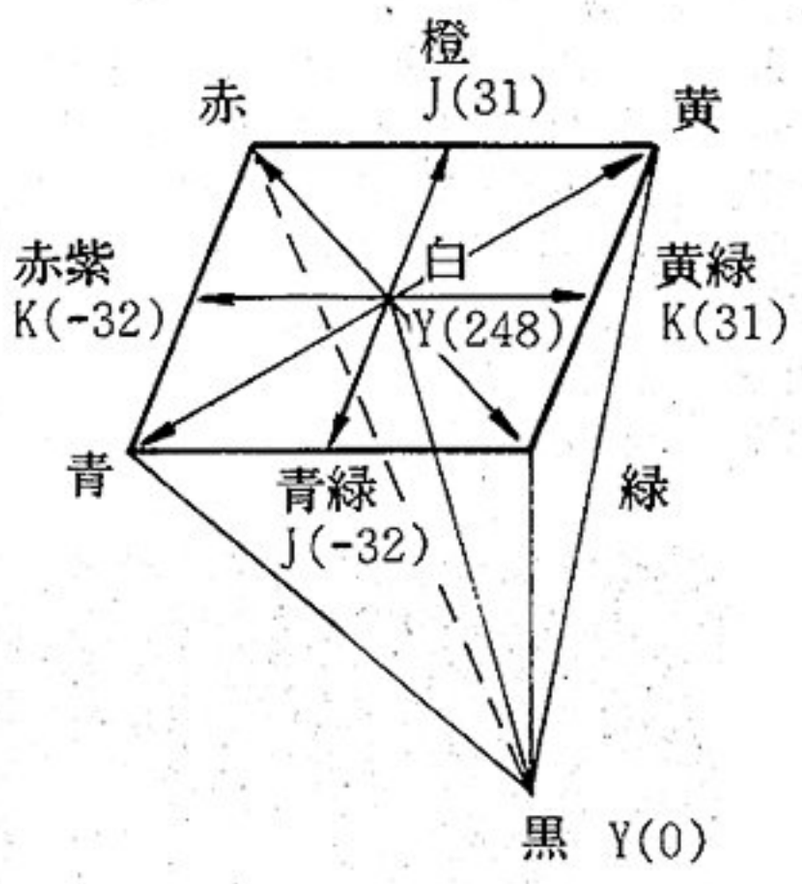
$$B = \frac{5}{4}Y - \frac{1}{2}J - \frac{1}{4}K$$

Ｙの値は０～３１（実際はD_3からのデータだから3ビットシフトして、8の倍数である0，8，16，‥，248の値）です。

ＪおよびＫは符号付きで処理され、値は－３２～＋３１です。（実際にLINE命令などで指定するときは0~63です。）

＜ＹＪＫの色の概念図＞

注意：この図はイメージをつかむための図です。実際の色とは異なります。



YJK方式のしくみ

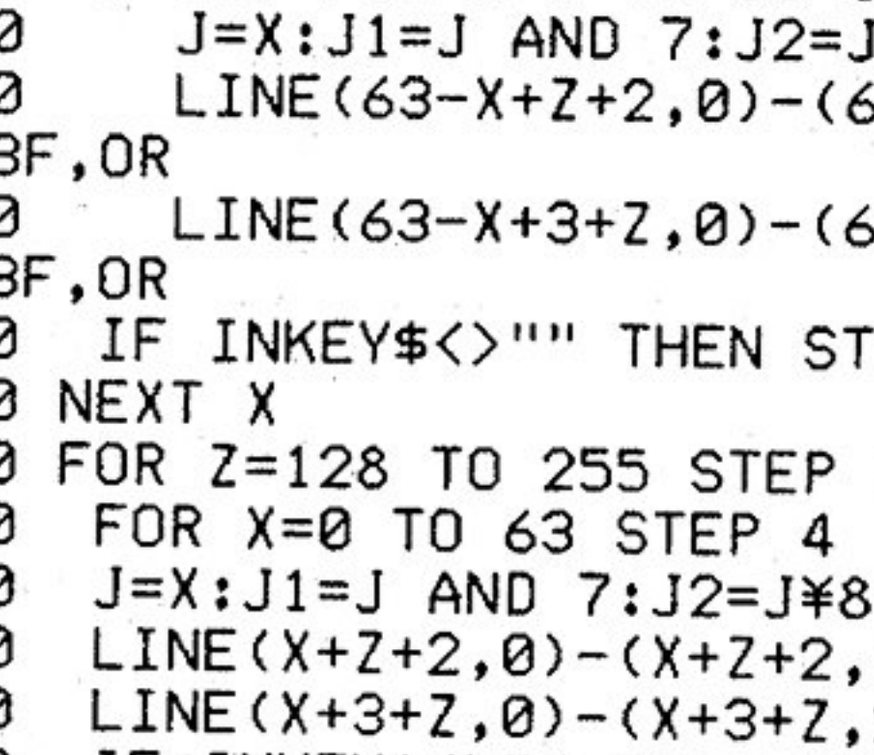
次のプログラムを入力してください。
ＳＣＲＥＥＮ １２の１９，２６８色を全て表示させることは簡単にはできませんが、ＹＪＫ方式によるカラーパターンを再現しています。

```
100 COLOR ,1,15:SCREEN 12
110 '********** Y Set ******************
120 FOR Y=0 TO 106 STEP 3
130  CC=VAL("&B"+BIN$(Y/3.3)+"000")
140  LINE(0,Y)-(255,Y+2),CC,BF
150  LINE(0,211-Y)-(255,209-Y),CC,BF
160 NEXT Y
170 '********** K Set ******************
180 FOR Z=0 TO 128 STEP 64
190  FOR X=0 TO 63 STEP 4
200  K=X:K1=K AND 7:K2=K¥8
210  LINE(X+Z,0)-(X+Z,211),K1,BF,OR
220  LINE(X+1+Z,0)-(X+1+Z,211),K2,BF,OR
230  IF INKEY$<>"" THEN STOP
240 NEXT X,Z
250 '********** J Set ******************
260  Z=61: FOR X=0 TO 63 STEP 4
270    J=X:J1=J AND 7:J2=J¥8
280    LINE(63-X+Z+2,0)-(63-X+Z+2,211),J
1,BF,OR
290    LINE(63-X+3+Z,0)-(63-X+3+Z,211),J
2,BF,OR
300  IF INKEY$<>"" THEN STOP
310 NEXT X
320 FOR Z=128 TO 255 STEP 64
330  FOR X=0 TO 63 STEP 4
340  J=X:J1=J AND 7:J2=J¥8
350  LINE(X+Z+2,0)-(X+Z+2,211),J1,BF,OR
360  LINE(X+3+Z,0)-(X+3+Z,211),J2,BF,OR
370  IF INKEY$<>"" THEN STOP
380 NEXT X,Z
390 GOTO 390
```

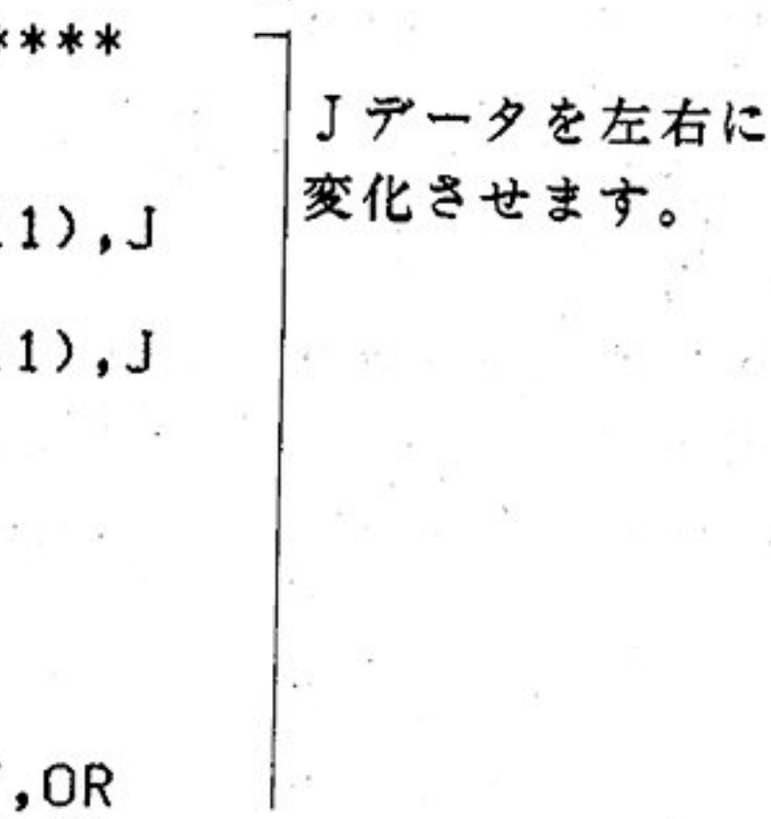
Yデータを上下に変化させます。

Kデータを左から変化させます。

Jデータを左右に変化させます。



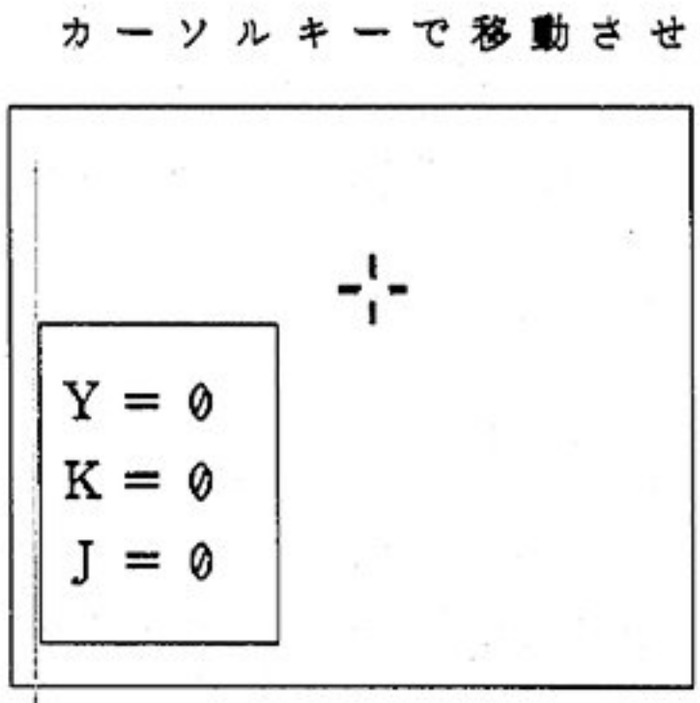
〈実行画面〉



次のプログラムは、前ページのプログラムで描いたパターンの指定した位置のＹＪＫデータを表示させます。２つのプログラムを組み合わせて使用するとＹＪＫの値と表示される色の関係がわかります。
１７ページのプログラムの３９０行からを入力し直してください(このプログラムの３８０行は単独でも動作するためです。組み合わせて使用するときは必要ありません)。

```
380 SCREEN 12
390 X=125:Y=106
400 OPEN "GRP:" AS #1
410 GOSUB 620:GOSUB 480
420 '** YJK ｹｲｻﾝ **
430 C1=POINT(X,Y):YD=C1¥8:XX=(X¥4)*4
440 K1=POINT(XX,Y) MOD 8:K2=POINT(XX+1,Y
) MOD 8
450 J1=POINT(XX+2,Y) MOD 8:J2=POINT(XX+3
,Y) MOD 8
460 K=K1+K2*8:J=J1+J2*8
470 GOTO 410
480 '** INPUT check & Sprite **
490 SPRITE$(0)=STRING$(2,16)+CHR$(0)+CHR
$(198)+CHR$(0)+STRING$(2,16)
500 A=STICK(0)
510 IF A=1 THEN Y=Y-1
520 IF A=2 THEN Y=Y-1:X=X+1
530 IF A=3 THEN X=X+1
540 IF A=4 THEN Y=Y+1:X=X+1
550 IF A=5 THEN Y=Y+1
560 IF A=6 THEN Y=Y+1:X=X-1
570 IF A=7 THEN X=X-1
580 IF A=8 THEN Y=Y-1:X=X-1
590 PUT SPRITE 0,(X-3,Y-3),8
600 IF STRIG(0)=0 THEN 500
610 RETURN
620 '** TABLE  ﾋｮｳｼﾞ **
630 LINE(12,168)-(79,200),0,BF,PSET:COLO
R 248,0
640 PSET (20,169),0:PRINT #1,USING "Y=##
";YD
650 PSET (20,179),0:PRINT #1,USING "K=##
";K
660 PSET (20,189),0:PRINT #1,USING "J=##
";J
670 RETURN
680 '** END ***
690 SCREEN 0:END
```

１６、１７ページのプログラムを組合わせて実行すると、右の図のような画面となります。



カーソルキーで"+"のスプライトを表示させたい色の位置に移動させ、スペースキーを押してください。画面の左下に、ＹＪＫの値が表示されます

ちょっと一言

■Y成分による色の変化

１６ページのプログラムを実行して、写真のように画面が表示されましたか？

画面を見て、縦の方向で色が違って見えますね？

画面の縦方向では、Y成分つまり明るさしか変わらないはずなのに、色が違って見えます。例えば右側の赤の色でも、明るければ白っぽく(黄色に)見え、暗ければ黒っぽく見えます。

これは１４ページにあるように、人の目が明るさに対して、色合いについては感度が低いことを意味します。

ＹＪＫ方式のデータ(ＳＣＲＥＥＮ　１０、１１の場合)

ＳＣＲＥＥＮ　１０と１１ではＹのデータを４ビットにし、８バイト中の１ビット(D_3)を属性(アトリビュート)ビットとしています。
この方式では、属性ビットが０の場合、その点はＹＪＫ方式で表示されます。属性ビットが１の場合、その点はＹの４ビットのデータがカラーコードとして扱われます。このカラーコードはカラーパレット機能を使って５１２色中１６色をＲＧＢのデータとして表示することができます。

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Y_1				A_1			K_L
Y_2				A_2			K_H
Y_3				A_3			J_L
Y_4				A_4			J_H

(SCREEN 10,11のとき)

Ｙは明るさ(輝度)のデータです。Ｋ、Ｊは色あいのデータです。K_LはＫの下位３ビット、K_HはＫの上位３ビットです。Ｊも同じです。
Ｙの値は０～１５(実際は4ビットシフトして、0,16,32,‥,240の値)です。ＪおよびＫは符号付きで処理され、値は－３２～＋３１です。(実際にLINE命令などで指定するときは0~63です。)

ＳＣＲＥＥＮ　１０の場合、ＹＪＫ方式の画面にＲＧＢ方式で図を描くとき、ＬＩＮＥ命令などの論理演算子の指定により、Ｙの値を意識しないで０～１５のカラーコードで描画色を指定できます。カラーコードの値はＹ成分にしか影響しません。また、描画した場合、属性ビットが１にセットされます。

ＳＣＲＥＥＮ　１１の場合、ＹＪＫ方式の画面にＲＧＢ方式で図を描くとき、カラーコードとして０～２５５が指定できます。Ｙ成分に加えてＪＫ成分の値も変えることができます。ＹＪＫ方式の画面の色を変えたくないときは、下位３ビットを考慮してカラーコードを指定してください。

ＳＣＲＥＥＮ　１０、１１は、ＲＧＢ方式で描画した場合、描画したカラーコードに対してカラーパレット機能が使用できます。

ＭＳＸ$_2$＋のコマンド －ＭＳＸ,ＭＳＸ$_2$との違い－

ＭＳＸ2+で使う命令や関数などを、アルファベット順に説明しています。またＭＳＸ、ＭＳＸ2で使っている命令でも、ＭＳＸ2+の機能を使ったときに使い方が変わる命令については、相違点のみ説明しています。
ＭＳＸ、ＭＳＸ2の命令については、別冊の「ＭＳＸプログラミング入門」をご覧ください。

機能別索引

＊印のコマンドは、ＭＳＸ、ＭＳＸ2から機能が変わります。

画面の制御・関数

関数・アドレス

画面の制御

色の制御

漢字の出力・制御

漢字表示の制御

漢字モードを終了します　CALL ANK ・・・23

漢字モードを起動します　CALL KANJI ・・・・・26

漢字の表示位置を指定する　LOCATE* ・・・33

漢字を表示します　PUT KANJI* ・・・33

文字列の制御

文字列を全角に変える　CALL AKCNV ・・・・・22

文字のＪＩＳコードを求める　CALL JIS ・・・24

文字列を半角に変える　CALL KACNV ・・・・・25

文字列の半角／全角文字だけを抜き出す　CALL KEXT ・・・・・28

文字列の中の指定した文字の位置を知る　CALL KINSTR ・・・・28

文字列の長さを知る　CALL KLEN ・・・・・29

文字列の指定した位置から文字列を取り出す　CALL KMID ・・・・29

文字列で指定された漢字コードの漢字を得る　CALL KNJ ・・30

文字列の指定した位置の文字の半角／全角を知る　CALL KTYPE ・・・・30

文字列のシフトＪＩＳコードを得る　CALL SJIS ・・・・31

BASE*

Base(ベース:基本)

働　き　画面出力に関連するテーブル(ＶＤＰレジスタの各テーブル)の先頭アドレスを得ます。

書き方　ＢＡＳＥ(<数式>)

例　ＰＲＩＮＴ ＢＡＳＥ(１０)
６１４４

説　明　画面表示を行うＶＤＰレジスタ内の各テーブルの先頭アドレスを得ます。指定できる数値は０～４４および５０～６４です。
このシステム変数に値を代入することはできません。

参　照　ＶＤＰ (37ページ)

CALL AKCNV

ANK Kanji Conversion(アンク・カンジ・コンバーション:ANK文字を漢字に変換する)

働　き　文字列の中の文字を全角文字列にします。

書き方　ＣＡＬＬ ＡＫＣＮＶ(<文字変数>,<文字列>)

例　ＣＡＬＬ ＡＫＣＮＶ(Ａ$,"ABC漢字ｲﾛﾊ"):ＰＲＩＮＴ Ａ$
ＡＢＣ漢字イロハ

説　明　<文字列>中の全ての文字を全角文字に変換して<文字変数>に代入します。
ＡＮＫ文字(Alphabet-Numeric-Kana文字)を半角文字と言います。
半角文字と漢字(全角文字)の混じった文字列を全て全角文字に置き換えることができます。

参　照　ＣＡＬＬ ＫＡＣＮＶ (25ページ)

CALL ANK

Alphabet-Numeric-Kana(アンク：英数カナ文字。半角で表すことのできる文字)

働　き　漢字モードを終了します。

書き方　ＣＡＬＬ　ＡＮＫ

説　明　漢字表示を終了します。ただし漢字処理用に確保したシステムのメモリは開放されません。

漢字表示を終了すると、漢字で作成されたプログラムなどは漢字表示されなくなります また、プリンタへのリスト出力なども漢字対応でなくなります。

参　照　ＣＡＬＬ　ＫＡＮＪＩ　(26ページ)

CALL CLS

Clean Screen(クリーン・スクリーン：画面のクリア)

働　き　画面をクリアします。

書き方　ＣＡＬＬ　ＣＬＳ

説　明　ＣＡＬＬ　ＫＡＮＪＩ命令により画面を漢字モードにしたとき、画面をクリアするために使用します。漢字モードでないときでも使用できます。

漢字モードでＣＬＳ命令を実行すると、"Illegal function call"エラーとなります。ただし、グラフィック画面(ＳＣＲＥＥＮ２～１２)では、ＣＬＳ命令も使用できます。

参　照　ＣＬＳ　(32ページ)

ＣＡＬＬ　ＪＩＳ

ＪＩＳ(ジス：ＪＩＳ規格の文字コード)

働　き　文字列のＪＩＳコードを文字列として求めます。

書き方　ＣＡＬＬ　ＪＩＳ(〈文字変数〉,〈文字列〉)

例　ＣＡＬＬ　ＪＩＳ(Ａ$,"漢字"):ＰＲＩＮＴ　Ａ$
３４４１

説　明　〈文字列〉の最初の２バイトを１６進４桁のＪＩＳコードに変換して〈文字変数〉に代入します。

参　考　文字の表し方には、ＪＩＳコードとシフトＪＩＳコードがあります。

ＪＩＳコードは日本工業規格(Japanese-Industrial-Standards)のルールです。

ＪＩＳコードは、漢字のデータの始まりに制御命令(ＳＩコード)を入れ、データの終りにも制御命令(ＳＯコード)を入れます。このため、漢字とＡＮＫ文字が混じると、メモリが多く必要となります。

これに対して、シフトＪＩＳコードは、ＪＩＳコードで決められていないコードに漢字制御と漢字の意味を持たせ、２バイトで漢字の表示を行います。メモリは少なくてすみますが、パソコン独自のグラフィック記号(●や♣など)は使えなくなります。

１６ビットのパソコンなどには、シフトＪＩＳコードで動作するソフトウェアもあり、データを交換するときに変換が必要となります。

参　照　ＣＡＬＬ　ＳＪＩＳ　(31ページ)

CALL KACNV

Kanji ANK Conversion(カンジ・アンク・コンバーション：漢字をANK文字に変換する)

働　き　文字列の中の文字を半角文字列にします。

書き方　ＣＡＬＬ　ＫＡＣＮＶ(〈文字変数〉,〈文字列〉)

例　ＣＡＬＬ　ＫＡＣＮＶ(Ａ$,"私はプログラマーです"):ＰＲＩＮＴ　Ａ$

私ハプログラマーデス

説　明　〈文字列〉中の半角文字に変換できる文字(英数字、カタカナ、ひらがな)を全て半角文字に変換して〈文字変数〉に代入します。

参　照　ＣＡＬＬ　ＡＫＣＮＶ　(22ページ)

CALL KANJI

Kanji(カンジ:漢字モードの開始)

働　き　漢字モードを起動する。

書き方　CALL KANJI〔<漢字モード>〕

例　CALL KANJI

CALL KANJI 1

説　明　漢字表示モードは、0，1，2，3のどれか1文字で、表示される漢字の大きさ(表示方法)を指定します。省略すると0と見なされます。6ページからの説明もご覧ください。

- CALL KANJI(またはCALL KANJI 0)
  漢字の文字パターン(フォントデータ)を漢字ROMから取り出して表示します。文字サイズは全角で横16ドット×縦16ドットです(カナ、英数字は半角で8×16ドット表示)。

- CALL KANJI 1
  文字パターン(フォントデータ)を漢字ROMから取り出し、横16ドットを12ドットに圧縮して表示します(12ドットの漢字ROMが差し込まれている場合は、このROMから表示します)。文字サイズは横12ドット×縦16ドットです。

- CALL KANJI 2
  CALL KANJI 0と同じですが、画面表示がインターレースモードとなり、縦方向の表示文字数(行数)が2倍となります。文字パターンは16×16ドットですが、見た目には16×8ドットとなります。

- CALL KANJI 3
  CALL KANJI 1と同じですが、画面表示がインターレースモードとなり、縦方向の表示文字数(行数)が増します。

BASICのコマンド待ちの状態(テキストモード)では、漢字の入力・表示が可能です。SCREEN2～12のグラフィックモードではプログラムによる漢字の出力のみが可能です。

日本語処理機能(MSX－JE)を持つカートリッジが差し込まれているときは、CALL KANJI命令の実行時にシステムに組み込まれます。このときは、連文節変換が可能となります。

プログラム中で最初のＣＡＬＬ ＫＡＮＪＩ命令を実行するときに、漢字変換用のワークエリア(漢字変換をするために必要なメモリ)を確保します。このため、プログラムや変数に使用できるメモリが少し減ります(約550バイト)。また変数、FOR~NEXT、GOSUB 命令用のメモリもクリアされます。一度ワークエリアを確保すると、２回目以降のＣＡＬＬ ＫＡＮＪＩ命令の実行ではメモリのクリアはされません。

なお、漢字モードを終了(ＣＡＬＬ ＡＮＫ命令を実行)してもメモリは確保されたままの状態です(リセットされるまで解放しません)。

参　考　漢字モードのときの表示幅

ＣＡＬＬ ＫＡＮＪＩ実行後の表示幅は、命令実行前の表示幅(ＡＮＫモードの表示幅)から次のように決定されます。

● ＳＣＲＥＥＮ ０を使用していたとき

・ＫＡＮＪＩ０、ＫＡＮＪＩ２命令のとき
ＡＮＫモードの表示幅の$\frac{4}{5}$が漢字モードの表示幅になります。

・ＫＡＮＪＩ１、ＫＡＮＪＩ３命令のとき
ＡＮＫモードの表示幅がそのまま漢字モードでの表示幅になります。

● ＳＣＲＥＥＮ １を使用していたとき

・ＫＡＮＪＩ０、ＫＡＮＪＩ２命令のとき
ＡＮＫモードの表示幅がそのまま漢字モードでの表示幅になります。

・ＫＡＮＪＩ１、ＫＡＮＪＩ３命令のとき
ＡＮＫモードの表示幅の$\frac{5}{4}$が漢字モードの表示幅になります。

ただし、計算によって決まった表示幅が２０未満の時は２０に設定されます。また、グラフィックモードにしたときは、テキストモードの表示幅に関係なく、常に表示できる最大幅に設定されます。

参　照 ☞　ＷＩＤＴＨ (38ページ)

CALL KEXT

Kanji Extract(カンジ・イクストラクト：漢字を抜き取る)

働　き　文字列の中から半角(全角)文字だけを抜き出します。

書き方　CALL KEXT(<文字変数>,<文字列>,<機能>)

例

```
CALL KEXT(A$,"今日は良イ天気。",0):PRINT A$
イ。
CALL KEXT(A$,"今日は良イ天気。",1):PRINT A$
今日は良天気
```

説　明　<機能>が0なら<文字列>中の半角文字(ANK文字)だけを、1なら全角文字だけを抜き出し<文字変数>に代入します。

CALL KINSTR

Kanji in String(カンジ・イン・ストリング：文字列の中の漢字)

働　き　文字列の中から指定した文字列を探します。

書き方　CALL KINSTR(<数値変数>,〔<数式>〕,<文字列1>,<文字列2>)

例

```
CALL INSTR(A,"A亜B","B"):PRINT A
3
```

説　明　<文字列1>の中から<文字列2>を探しだし、見つかれば発見した位置を、見つからなければ0を<数値変数>に代入する。
<数式>は探し始める位置を文字数で指定します。省略されると1とみなされる。
探し出す文字列は、全角/半角を区別して探します。
また全角文字の1文字も1文字として扱われます。

CALL KLEN

Kanji Length(カンジ・レングス:漢字の長さ)

働　き　文字列の中の全角(半角)の文字数を得ます。

書き方　CALL KLEN(<数値変数>,<文字列>,〔<機能>〕)

例

```
CALL KLEN(A,"今日ハ良イ天気です"):PRINT A
9
CALL KLEN(A,"今日ハ良イ天気です",1):PRINT A
2
CALL KLEN(A,"今日ハ良イ天気です",2):PRINT A
7
```

説　明　機能が0(もしくは省略)の時は<文字列>の全体の文字数を、1の時は<文字列>中の半角文字の文字数を、2の時は<文字列>中の全角文字の文字数を<数値変数>に代入します。

CALL KMID

Kanji Middle(カンジ・ミドル:漢字文字列の中)

働　き　文字列の中から指定された位置の文字を抜き出します。

書き方　CALL KMID(<文字変数>,<文字列>,<式1>,〔<式2>〕)

例

```
CALL KMID(A$,"今、何時？",3,2):PRINT A$
何時
```

説　明　<文字列>中の<式1>番目の文字から<式2>文字分だけ抜き出して<文字変数>に代入します。<式2>が省略された場合は<式1>番目の文字から終わりまで全ての文字が代入されます。

CALL KNJ

Kanji(カンジ:漢字)

働　き　文字列で指定された漢字コードの漢字を得ます。

書き方　CALL KNJ(<文字変数>,<文字列>)

例　CALL KNJ(A$,"3441"):PRINT A$

漢

説　明　<文字列>で指定される4桁の漢字コードに相当する漢字1文字を<文字変数>に代入します。漢字コードが&H8000未満の時はJISコード、以上の時はシフトJISコードとみなされます。

参照　CALL JIS (24ページ)

CALL KTYPE

Kanji Type(カンジ・タイプ:漢字の形)

働　き　文字列の中の指定された文字のタイプ(全角/半角)を得ます。

書き方　CALL KTYPE(<数値変数>,<文字列>,<数式>)

例　CALL KTYPE(A,"今日ハ",2):PRINT A

1

説　明　<文字列>中の<数式>番目の文字のタイプ(半角なら0、全角なら1)を<数値変数>に代入します。

<数式>は省略できません。<数式>を省略したり、文字列の文字数より大きい数を指定するとエラーとなります。

例).

```
10 A$="今日ハ良イ天気です"
20 CALL KLEN(L,A$)
30 FOR I=1 TO L
40 CALL KTYPE(T,A$,I):PRINT T;
50 NEXT I
RUN
1  1  0  1  0  1  1  1  1
```

CALL PALETTE

Palette(パレット:調色板、色彩)

働き　カラーパレットの初期化または設定を行ないます。

書き方　CALL PALETTE〔(〈カラーコード〉,〈赤レベル〉,〈緑レベル〉,〈青レベル〉)〕

例　CALL PALETTE(15,4,4,4)

説明　CALL KANJI命令により画面を漢字モードにしたときにパレットを変更するために使用します。漢字モードでないときでも使用できます。

漢字モードでは、COLOR命令で色レベルを変更しようとすると、"Illegal function call"エラーとなります。

なお、グラフィック画面(SCREEN2～12)では、COLOR命令で色レベルを変更することもできます。

パラメータをすべて省略した場合は、パレットを初期状態にします。

パラメータが指定された場合、〈カラーコード〉で指定された色を〈赤レベル〉,〈緑レベル〉,〈青レベル〉の色に設定します。なお、カラーコードや各レベルを省略することはできません。

CALL SJIS

Shift JIS(シフト・ジス:移動JISコード)

働き　文字のシフトJISコードを得ます。

書き方　CALL SJIS(〈文字変数〉,〈文字列〉)

例　CALL SJIS(A$,"漢字"):PRINT A$

8ABF

説明　〈文字列〉の最初の2バイトを16進4桁のシフトJISコードに変換して〈文字変数〉に代入します。

参照　CALL JIS (24ページ)

CLS*

Clean Screen(クリーン・スクリーン：スクリーンのクリア)

働　き　画面をクリアします。

書き方　ＣＬＳ

説　明　ＣＬＳ命令は、ＣＡＬＬ　ＫＡＮＪＩ命令により画面を漢字モードにすると、"Illegal function call"となり使用できません。
漢字モードの時はＣＡＬＬ　ＣＬＳ命令を使用してください。ただし、グラフィックモード(ＳＣＲＥＥＮ２～１２)では使用できます。

参　照　ＣＡＬＬ　ＣＬＳ　(23ページ)

COLOR*

Color(カラー：色)

働　き　カラーパレットの初期化または設定を行ないます。

書き方　ＣＯＬＯＲ
ＣＯＬＯＲ＝ＮＥＷ
ＣＯＬＯＲ＝ＲＥＳＴＯＲＥ
ＣＯＬＯＲ＝(〈カラーコード〉,〈赤レベル〉,〈緑レベル〉,〈青レベル〉)

説　明　ＣＡＬＬ　ＫＡＮＪＩ命令により漢字モードにすると、ＣＯＬＯＲ命令のカラーパレット機能は、"Illegal function call"となり使用できません。
漢字モードの時はＣＡＬＬ　ＰＡＬＥＴＴＥ命令を使用してください。ただし、グラフィックモード(ＳＣＲＥＥＮ２～１２)では使用できます。

参　照　ＣＡＬＬ　ＰＡＬＥＴＴＥ　(31ページ)

LOCATE*

Locate(ロケイト:置く、設定する)

働　き　テキスト画面のカーソルの位置および働きを指定します。

書き方　ＬＯＣＡＴＥ〔<Ｘ座標>〕,〔<Ｙ座標>〕,〔<カーソルスイッチ>〕

例　ＬＯＣＡＴＥ１５，１０

説　明　カーソルをテキスト画面の指定した位置(Ｘ,Ｙ)へ移動します。この命令は、ＰＲＩＮＴ文で表示する文字やＩＮＰＵＴ文で表示するプロンプト文の開始位置を指定するときなどに用います。(入力待ちのときの？の位置を指定できます。)

<Ｘ座標>は、画面の右端を０とする水平座標を表す数で、３２×２４テキストモードのとき０～３１、４０×２４テキストモードのとき０～３９です。省略した場合は０とみなされます。

ＣＡＬＬ　ＫＡＮＪＩ命令で画面を漢字モードにした後、グラフィックモード(ＳＣＲＥＥＮ２～１２)の画面に、ＰＲＩＮＴ命令で文字を表示させることができます。このとき、表示位置をＬＯＣＡＴＥ命令で指定することができます。ただし、カーソルスイッチは指定できません。

<Ｘ座標>の値はテキストモードの表示幅に関係なく、グラフィック画面で表示できる最大文字数までの値となります。

例．ＳＣＲＥＥＮ５、ＣＡＬＬ　ＫＡＮＪＩのとき、
　　<Ｘ座標>は０～３２です。

参　照　ＣＡＬＬ　ＫＡＮＪＩ　(26ページ)

PUT KANJI*

Put Kanji(プット　カンジ:漢字を出力する)

働　き　ＳＣＲＥＥＮ　５から１２で漢字を表示します。

書き方　ＰＵＴ　ＫＡＮＪＩ〔(<Ｘ>,<Ｙ>)〕,<漢字コード>〔,<カラーコード>〕〔,<ロジカルオペレーション>〕〔,<モード>〕

例　ＰＵＴ　ＫＡＮＪＩ(１００,１００),&Ｈ３０２１

説　明　ＭＳＸ2+では、漢字コードがＪＩＳ第二水準まで指定できるようになりました。ただし、表示するためにはＪＩＳ第二水準漢字ＲＯＭが必要となります(ＭＳＸ2+では、第二水準の漢字ＲＯＭはオプション装備です)。

SCREEN*

Screen(スクリーン:画面)

働　き　画面やスプライトの大きさなどについての設定を行ないます。

書き方　SCREEN〔<画面モード>〕〔,<スプライトサイズ>〕〔,<キークリックスイッチ>〕〔,<カセットボーレート>〕〔,<プリンタオプション>〕〔,<インターレースモード>〕

例　　SCREEN 12

説　明　MSX2+にはMSX2に、SCREEN10～12の画面モードが追加されています。なお、SCREEN9はありません。

<画面モード>以外の使用方法は、MSX2と同じです。<画面モード>以外については、別冊の「MSXプログラミング入門」をご覧ください。また、YJK方式については、13ページの説明もご覧ください。

- SCREEN 10

  属性(アトリビュート)混在のYJKモードを選択します。このモードで指定できるカラーコードの範囲は0～15です。

  描画のときの処理は、どんな論理演算子を指定したかにより変わります。

  ・論理演算子がPSETまたは省略されたとき

  PSETやLINE命令で描画するとき、描いた点の属性に1がセットされ、上位の4ビット(Y成分のデータ)が、指定されたカラーコードとなります。カラーコードはCOLOR命令などのパレット機能により表示する色を512色中16色で指定できます。

  ・論理演算子にPSET以外が指定されたとき

  指定されたカラーコードと、描画する場所のY成分とのあいだで論理演算が実行されます。カラーコードは、属性が1にセットされている点に対してはパレット機能を持つカラーコードとして指定していることになり、属性がセットされていない点に対してはY成分として指定することになります。

- SCREEN 11

  属性混在のYJKモードを選択します。このモードで指定できるカラーコードの範囲は0～255です。

  SCREEN 10と違い、描画のときに指定されたカラーコードは、そのまま描画する点のYJK成分と論理演算されます。このため、Y成分のほかにJK成分も変わり、描画した点の色が変わることもあります。描画した結果、属性が1にセットされた点のカラーコードに対して、パレット機能で表示する色を変えることもできます。

● SCREEN 12

属性なしのＹＪＫモードを選択します。このモードで指定できるカラーコードの範囲は０～２５５です。
描画のときに指定されたカラーコードは、そのまま描画する点のＹＪＫ成分と論理演算されます。

ＭＳＸ、ＭＳＸ2のＳＣＲＥＥＮ文では、〈画面モード〉が変わると表示を初期化(画面のクリア)しましたが、ＳＣＲＥＥＮ10,11,12の間での切換えでは次のようになります。

● 画面表示がクリアされない画面モード切換え
  - ＳＣＲＥＥＮ １０ ⇒ ＳＣＲＥＥＮ １１
  - ＳＣＲＥＥＮ １０ ⇒ ＳＣＲＥＥＮ １２
  - ＳＣＲＥＥＮ １１ ⇒ ＳＣＲＥＥＮ １０
  - ＳＣＲＥＥＮ １１ ⇒ ＳＣＲＥＥＮ １２

● 画面表示はクリアされませんが、属性が０にされる画面モード切換え（Ｙ成分が５ビットから４ビットになるため、階調が下がります）
  - ＳＣＲＥＥＮ １２ ⇒ ＳＣＲＥＥＮ １０
  - ＳＣＲＥＥＮ １２ ⇒ ＳＣＲＥＥＮ １１

サンプルプログラム

ＳＣＲＥＥＮ １０～１２の画面に、ＬＩＮＥ命令で線を描きます。１５０行の画面モードと、１６０行の論理演算子(ＯＲ,ＡＮＤ,ＰＳＥＴ)を変えてみてください。

```
100 COLOR ,1,15:SCREEN 12
110 FOR Y=0 TO 211 STEP 3
120  CC=VAL("&B"+BIN$(Y/7)+"000")
130  LINE(0,Y)-(255,Y+2),CC,BF
140 NEXT Y
150 SCREEN 11
160 LINE(0,0)-(255,211),8,,OR
170 GOTO 170
```

SET SCROLL

Set Scroll(セット スクロール:巻き物、巻く)

働　き　表示画面を水平及び垂直方向にスクロールします。

書き方　SET SCROLL〔<X>〕〔,<Y>〕〔,<マスク>〕〔,<2ページ>〕

例　SET SCROLL 128,100,0,0

説　明　表示画面を水平および垂直方向にスクロールします。<X>で水平方向(左へ)の、<Y>で垂直方向(上へ)のスクロール量を数式で指定します。<X>の範囲は0～511、<Y>の範囲は0～255です。

<X>、<Y>の値は、始めの位置からいくらスクロールするかの値です。現在の位置からのスクロール値ではありません。

<マスク>は表示画面の左端8ドットを周辺色で隠すかどうかを指定します。これは、水平スクロールのとき、画面の左端に画面の切れが見えるためです。<マスク>が0の時は隠しません。<マスク>が1の時は隠します。

<2ページ>はスクロールをしたときに、表示されている画面(ページ)の右端または左端に次のページを表示するかを数で指定します。<2ページ>が0ならばスクロールをしたときに同じページ内で表示が繰り返されます。<2ページ>が1ならば2ページ連続のスクロールになります。

例えば、表示しているページが左へ移動すると、移動した量だけ次ページの画面の内容が、右端に表示されます。また、この時はあらかじめSET PAGE命令で表示ページを奇数ページにする必要があります。

注　意　このSET SCROLL命令はSCREEN 0の時は動作が異なります。

表示されている文字の位置は変更されず、文字のフォントの表示開始位置が変わります(文字の形がくずれます)。

水平スクロール量は<X>の下位3ビットのみが有効となり、1を指定すると7ドット右にスクロールします。2の時は6ドット、7の時は1ドットとなります。7ドット以上はスクロールしません。

垂直スクロール量は<Y>の下位3ビットのみが有効となり、1を指定すると1ドット上にスクロールします。2の時は2ドット、7の時は7ドットとなります。7ドット以上はスクロールしません。

サンプルプログラム

３５ページのプログラムを変更して、水平･垂直スクロールを行います。

```
100 COLOR ,1,15:SCREEN 12
110 FOR Y=0 TO 211 STEP 3
120  CC=VAL("&B"+BIN$(Y/7)+"000")
130  LINE(0,Y)-(255,Y+2),CC,BF
140 NEXT Y
150 SCREEN 11
160 LINE(0,0)-(255,211),8,,OR
170 COPY (0,0)-(255,50) TO (0,212)
180 FOR X=0 TO 255
190 SET SCROLL X,X,1
200 NEXT X:GOTO 180
```

VDP*

VDP(ブイ･デー･ピー)

働　き　ＶＤＰのレジスタへデータを書き込んだり、レジスタからデータを読みだしたりします。

書き方　ＶＤＰ(<レジスタ番号>)

例　ＶＤＰ(２６)＝ＶＤＰ(２６)ＯＲ&Ｈ８

説　明　ＶＤＰ(ビデオ･ディスプレイ･プロセッサ)のレジスタへデータを書き込んだり、レジスタからデータを読みだしたりします。
指定できる<レジスタ番号>は－９～－１、０～２８および３３～４７です。－９～－１がＶＤＰのステータスレジスタ９～１へ、０～７がレジスタ０～７へ、８がステータスレジスタ０へ、９～２８がレジスタ８～２７へ、３３～４７がレジスタ３２～４６へ対応しています。
この変数は、ＶＤＰについての知識を得てから使用してください。

ＷＩＤＴＨ*

Width(ウィズス:広さ、幅)

働　き　画面に表示する文字数を設定します。

書き方　ＷＩＤＴＨ〈桁数〉

例　ＷＩＤＴＨ　２８

説　明　ＷＩＤＴＨ命令はＣＡＬＬ　ＫＡＮＪＩ命令を実行したときには次のように動作します。

- テキストモードの場合(ＳＣＲＥＥＮ　０と１)
  - ＫＡＮＪＩ０かＫＡＮＪＩ２の場合
    〈桁数〉の指定は２０から６４が有効です。
  - ＫＡＮＪＩ１かＫＡＮＪＩ３の場合
    〈桁数〉の指定は２０から８０が有効です。

  上記以外の値が指定されると"Illegal function call"となります。
- グラフィックモードの場合(ＳＣＲＥＥＮ　２～１２)
  常に"Illegal function call"となります。

漢字表示を行なわないときは、ＭＳＸ2と同じ働きをします。

ＭＳＸ－ＭＵＳＩＣ（ミュージック）の使い方

この章では、ＭＳＸ2+のオプション機能であるＭＳＸ－ＭＵＳＩＣの機能について説明します。

ＭＳＸ－ＭＵＳＩＣにより、ＦＭ音源を使ってピアノ、ギター、オルガンなど、６４種類の音色を出すことができます。また、８オクターブ９重和音の演奏もできます。

音楽の演奏は、ＭＳＸ ＢＡＳＩＣの機能を拡張したＢＡＳＩＣ命令により簡単にできます。あなたも、コンピュータミュージックに挑戦してください。

注意：ＭＳＸ－ＭＵＳＩＣが使えるパソコン

パソコン本体に♪マークがついていると、ＭＳＸ－ＭＵＳＩＣの機能が使えます。

ＦＭ音源とは？

別冊の「ＭＳＸプログラミング入門 」では、ＰＬＡＹ命令とＳＯＵＮＤ命令を使った音の出し方が説明してあります。ＰＬＡＹ命令とＳＯＵＮＤ命令は、ＭＳＸ、ＭＳＸ2 標準のＰＳＧ(プログラマブル・サウンド・ゼネレータ)を使って音を出します。

この章で説明するＭＳＸ－ＭＵＳＩＣは、ＭＳＸ2+ のオプション機能として追加されています。ＭＳＸ－ＭＵＳＩＣは、電子楽器(シンセサイザー)などに使われているＦＭ音源と同じ原理で音を出すことができ、ＰＳＧを使った音に加えて８オクターブ９重和音(または６重和音＋５リズム音)を発生できます。また、♪マークのついたゲームを使うと、ゲームの音が迫力ある音になります。

サンプルプログラム

♪マークのついたゲームを持っていないとき、また簡単に音の違いを知りたいとき、次のプログラムを実行してみましょう。３重和音のＦＭ音源でオルゴールの音を出します。

120行、150行のＰＬＡＹ命令の「＃２」を「＃０」にするとＰＳＧの音(３重和音)となります。

また、120行の「Ｔ９０」を「Ｔ９０＠ｎ」(ｎは０～６３の数字)に変えると、音の種類を変えることができます。

```
100 CALL MUSIC(0,0,1,1,1)
110 CALL TRANSPOSE(2400)
120 PLAY #2,"T90","T90","T90"
130 READ A$:IF A$="END" THEN RESTORE:GOT
O 130
140 READ B$,C$
150 PLAY #2,A$,B$,C$
160 GOTO 130
170 DATA L16,L16,L16
180 DATA O5C8DC04B05CA4F8
190 DATA O4F4F8O5C404A8
200 DATA O3A8B-AG+AO4F4C8
210 DATA O5E4D8L4G.L16
220 DATA A4G8L4B-.L16
230 DATA O3B-4B-8O4L4D.L16
240 DATA L8O4B-O5C.D16EF.G16
250 DATA L8O4DF.F16GB-.B-16
260 DATA L8O3GA.B-16O4CD.E16
270 DATA END
```

ちょっと一言

■音楽用ＬＳＩについて

ＭＳＸパソコンで音を出すには、次の３つの方法があります。

- ＭＳＸ標準のＰＳＧ(プログラマブル・サウンド・ゼネレータ)を使う。
- ＭＳＸ2+ のＭＳＸ－ＭＵＳＩＣ(オプション装備)を使う。
- ＭＳＸ－ＡＵＤＩＯ(カートリッジなど)を使う。

ＰＳＧでは、ノイズ発生器で音の高さに合わせて一定の周波数の信号を作り、この信号の出力する大きさと、出力する間隔を変えることで、ピアノやオルガンに似た音を作っています。しかし、実際の楽器の音は何種類もの(周波数の)音の合成でできているので、一定の信号ではリアルな音になりません。

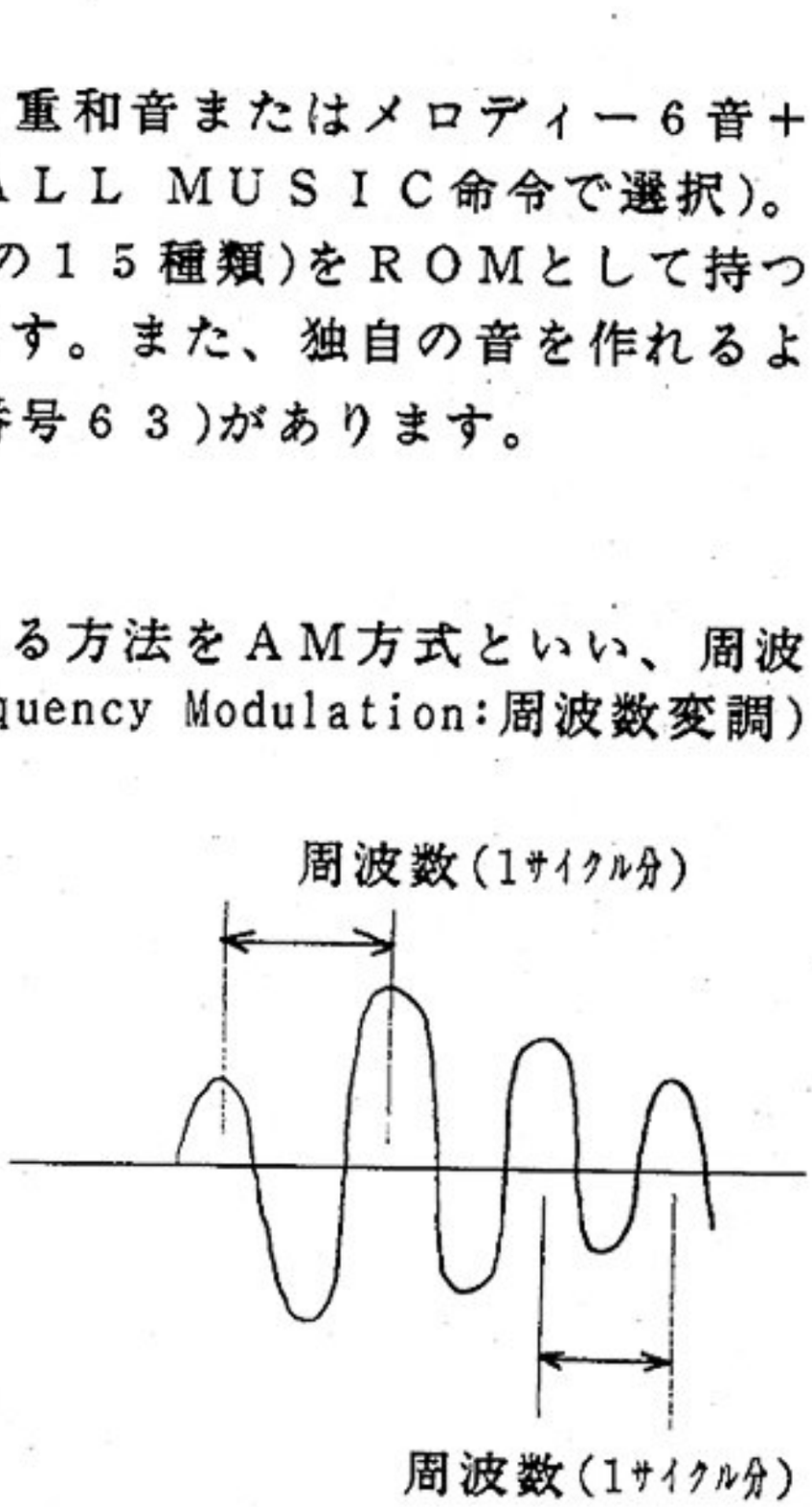
ＭＳＸ2+ とＭＳＸ－ＡＵＤＩＯでは、ＦＭ音源(ＦＭ方式の音出力)を採用しています。ＰＳＧと異なり、出力する信号の周波数を変えることができます。
ＭＳＸ2+ の音楽用ＬＳＩは、ＭＳＸ－ＡＵＤＩＯの演奏方式をパソコンに内蔵できるように小型化し、機能の一部を変更したものです。次の特長があります。

- ＦＭ音源を使用しています。
- 演奏のモードを選択することで９重和音またはメロディー６音＋リズム５音の演奏ができます(ＣＡＬＬ ＭＵＳＩＣ命令で選択)。
- 音色データ(音色番号０～６３中の１５種類)をＲＯＭとして持つので、音色の変更が簡単にできます。また、独自の音を作れるように、１音分のデータ領域(音色番号６３)があります。

■ＦＭ方式とは

音の信号を一定の信号(ノイズ)で作る方法をＡＭ方式といい、周波数を変えて作る方法はＦＭ方式(Frequency Modulation:周波数変調)といいます。

ＦＭ方式は、信号の周波数を変えることができるので、自然楽器の合成音から電子音までの音を作ることができます。



音楽命令の使い方 －コマンド一覧－

機能別索引

＊印のコマンドは、ＭＳＸ、ＭＳＸ2から機能が変わります。

関数･レジスタ制御

演奏の制御

音の制御

CALL AUDREG

Audio Register(オーディオレジスタ：音楽LSI用の登録)

働　き　音楽ＬＳＩのレジスタに書き込みを行います。

書き方　ＣＡＬＬ　ＡＵＤＲＥＧ(〈レジスタ番号〉,〈値〉[,〈チャンネル番号〉])

例　ＣＡＬＬ　ＡＵＤＲＥＧ(＆Ｈ２０,０)

説　明　ＦＭ音源で演奏する音楽ＬＳＩのレジスタに対して書き込みを行います。
システムソフトウェアが割り込みなどでひんぱんに書き込んでいるレジスタには、効果がないときがあり、このときはシステムの再立ち上げが必要な場合があります。
〈チャンネル番号〉は、０または省略しなければなりません。これは、ＭＳＸ－ＡＵＤＩＯとの互換性をとるために用意されているもので、ＭＳＸ－ＭＵＳＩＣでは意味を持ちません。

レジスタへの書き込みは、音楽ＬＳＩについての知識を得てから行ってください。

CALL BGM

BGM(ビージーエム:Back Ground Music の略)

働　き　バックグラウンド処理を行うかどうかを指定します。

書き方　CALL BGM(<変数>)

例　CALL BGM(0)

説　明　<変数>は0または1の値をとり、次のような意味を持ちます。

- 0：PLAY文のバックグラウンド処理を行わない。
- 1：PLAY文のバックグラウンド処理を行う。

CALL MUSIC命令による初期化ではバックグラウンド処理(1)が指定されています。

バックグラウンド処理とは、PLAY命令により演奏を開始した後、演奏が終了する前に他の命令を実行することができる方法です。

サンプルプログラム

40ページのプログラムを変更して、演奏するデータを表示させます。

CALL BGM(1)を0に変えると、演奏が途切れます。これは、演奏を終了してからデータを表示し、その後、演奏を再開するためです。

```
100 CALL MUSIC(0,0,1,1,1)
105 CALL BGM(0)
110 CALL TRANSPOSE(2400)
120 PLAY #2,"T90","T90","T90"
130 READ A$:IF A$="END" THEN RESTORE:GOT
O 130
140 READ B$,C$
150 PLAY #2,A$,B$,C$
155 PRINT A$,B$,C$
160 GOTO 130
170 DATA L16,L16,L16
180 DATA O5C8DC04BO5CA4F8
190 DATA O4F4F8O5C4O4A8
200 DATA O3A8B-AG+AO4F4C8
210 DATA O5E4D8L4G.L16
220 DATA A4G8L4B-.L16
230 DATA O3B-4B-8O4L4D.L16
240 DATA L8O4B-O5C.D16EF.G16
250 DATA L8O4DF.F16GB-.B-16
260 DATA L8O3GA.B-16O4CD.E16
270 DATA END
```

CALL MUSIC

Music(ミュージック:音楽)

働　き　MSX-MUSICのシステムを初期化します。

書き方　CALL MUSIC[([<モード>][,[0][,<PLAY文第1文字列のチャンネル数>[,<PLAY文第2文字列のチャンネル数>[,・・・[,<PLAY文第9文字列のチャンネル数>]]]]]]]]]])

例　CALL MUSIC　初期化します。
CALL MUSIC(0,0,1,1,1,1,1,1,1,1,1)
1チャンネルずつPLAY文の文字列に割り当てます。

説　明　音楽LSIの初期化とともに9個のFM音源のチャンネルをどのように使用するかを指定します。CALL MUSIC命令により初期化を行うまでは、全てのMSX-MUSICの命令は使用できません。

<モード>は0か1で、0はリズム音を使用しないモードです。
1はリズム音を使用するモードです。
リズム音を使用する場合にはチャンネル7、8、9を使用しますので、通常の和音に使えるのは残り6チャンネルになります。したがって、PLAY文で指定できる文字列はリズム使用時には6以下、リズムを使わないときは9以下である必要があります。
[0]は、MSX-AUDIOとの互換性のためで、必ず0です。
チャンネルの使用割り当ては、PLAY文では、チャンネル番号の小さい方から(1,2,3…を)割り当てます。

悪い例)
PLAY文の文字列へのチャンネル数を、0に設定したり省略したりすることはできません。次の例を参照してください。

CALL MUSIC(0,0,0,5,0)
↑　↑
0は設定できません
(Illegal function callになります)

CALL MUSIC(0,0,1,,,2)
↑
省略できません(Syntax errorになります)

ＣＡＬＬ　ＭＵＳＩＣ

パラメータなしでＣＡＬＬ　ＭＵＳＩＣと使われた時は、ＣＡＬＬ　ＭＵＳＩＣ
（１，０，１，１，１）と同じになります。
つまり、

（１，０，１，１，１）

- ＦＭ音源のチャンネル１をＰＬＡＹ文の最初の文字列に割り当てる。
- ＦＭ音源のチャンネル２をＰＬＡＹ文の２番目の文字列に割り当てる。
- ＦＭ音源のチャンネル３をＰＬＡＹ文の３番目の文字列に割り当てる。
- リズム音をＰＬＡＹ文の４番目の文字列に割り当て使用する。
- ＰＬＡＹ文の５番目～７番目の文字列はＰＳＧ音源の制御に割り当てる。

という意味になります。

注意！

ＣＡＬＬ　ＭＵＳＩＣ命令を実行すると、音楽演奏用のワークエリア（約810バイト）を確保します。
このため、それ以前に使われていた変数、配列などの内容およびＦＯＲ～ＮＥＸＴ、ＧＯＳＵＢの情報がクリアされます。ＣＡＬＬ　ＭＵＳＩＣ命令は、プログラムの初めに実行してください。

CALL PITCH

Pitch(ピッチ：調子、音の高低)

働　き　FM音源の音の高さ(ピッチ)を与えます。

書き方　CALL PITCH(〈ピッチ１〉[,〈ピッチ２〉])

例　CALL PITCH(４４０)

説　明　FM音源で発生する全体の音の高さを指定します。〈ピッチ１〉の範囲は４１０～４５９で単位は[Ｈｚ]です。中央C(ハ長調)のすぐ上のA音(ａ２)の周波数を基準に音の高さを表します。トランスポーズ(移調)とは独立に設定でき、初期値は４４０です。ピッチを変えるとFM音源の音の高さが変化します(リズム音や音程をもたない音は除く)。また、この命令は、PSG音源の音には作用しません。

〈ピッチ２〉はMSX-AUDIOとの互換性のために用意されているもので、MSX-MUSICでは指定しても意味を持ちません。

トランスポーズについては、CALL TRANSPOSE文の項を参照してください。

参　照　CALL TRANSPOSE (50ページ)

注　意　CALL PITCH命令を実行すると、実行した次点で演奏中のピッチが変化します。つまり、PLAY命令の実行により演奏データが一旦、バッファに貯えられ、順次、演奏されますが、演奏終了前にCALL PITCH命令が実行されると、それ以降のデータは変更されたピッチで演奏されます(次のプログラムの１４０行を止めると良く判ります)。

```
100 CALL MUSIC
120 CALL PITCH(410)
130 PLAY #2,"cc"
140 CALL PLAY(0,A):IF A<>0 THEN 140
150 CALL PITCH(459)
160 PLAY #2,"cc"
```

CALL PLAY

PLAY(プレイ:演奏している)

働　き　ＰＬＡＹ文による音楽を演奏中かどうかを返します。

書き方　ＣＡＬＬ　ＰＬＡＹ(<ＰＬＡＹ文の文字列番号>,<変数名>)

例　ＣＡＬＬ　ＰＬＡＹ(０,Ａ):ＰＲＩＮＴ　Ａ

説　明　ＦＭ音源およびＰＳＧの状態を調べ、指定されたチャンネルが音楽を演奏中かどうかを判断します。演奏中であれば－１、そうでなければ０の値を変数に代入します。<文字列番号>は、ＰＬＡＹ文で演奏を指定したデータ(文字列)の番号です。(例．ＰＬＡＹ＃２,"ＡＢ","ＣＤ","ＥＦ"とすると"ＡＢ"が「１」、"ＣＤ"が「２」、"ＥＦ"が「３」です。)<文字列番号>として０が与えられた場合は、ＦＭ音源およびＰＳＧ音源が演奏中であれば－１を、そうでなければ０を返します。

ＣＡＬＬ　ＰＬＡＹ文の文字列番号は、ＭＵＳＩＣ文で指定したチャンネル数＋３まで使えます。つまり、ＭＵＳＩＣ文で指定したＦＭ音源に加え３チャンネルのＰＳＧ音源について有効です。

CALL STOPM

Stop Music(ストップ　ミュージック:音楽を止める)

働　き　バックグラウンドで実行中のＰＬＡＹ文の演奏を停止します。

書き方　ＣＡＬＬ　ＳＴＯＰＭ

説　明　バックグラウンドで実行中のＰＬＡＹ文の音楽の演奏を停止します。

参　照　ＣＡＬＬ　ＢＧＭ (44ページ)

CALL TEMPER

Temperament(テンペラメント:音律)

働　き　音律を設定します。

書き方　CALL TEMPER(<音律番号>)　　音律番号:0～21

例　CALL TEMPER(0)

説　明　音律を与えるステートメントで、FM音源の音の高さに影響を与えます。

音律は1オクターブをどのような比率で12音に分割するかを決めるもので、古典音楽には古典音律が適していると言われます。

初期値は9番の完全平均律です。

番号	音　律
0	ピタゴラス
1	ミーントーン
2	ヴェルクマイスター
3	ヴェルクマイスター(修正)
4	ヴェルクマイスター(別)
5	キルンベルガー
6	キルンベルガー(修正)
7	ヴァロッティ・ヤング
8	ラモー
9	完全平均律(初期値)
10	純正律c　メジャー　(aマイナー)
11	純正律cis　メジャー　(bマイナー)
12	純正律d　メジャー　(hマイナー)
13	純正律es　メジャー　(cマイナー)
14	純正律e　メジャー　(cisマイナー)
15	純正律f　メジャー　(dマイナー)
16	純正律fis　メジャー　(esマイナー)
17	純正律g　メジャー　(eマイナー)
18	純正律gis　メジャー　(fマイナー)
19	純正律a　メジャー　(fisマイナー)
20	純正律b　メジャー　(gマイナー)
21	純正律h　メジャー　(gisマイナー)

CALL TRANSPOSE

Transpose(トランスポーズ：置きかえる。移調する)

働　き　FM音源の音に対してセント単位で移調を与えます。

書き方　CALL TRANSPOSE(<トランスポーズ値1>[,<トランスポーズ値2>])

例　CALL TRANSPOSE(0)

説　明　セント(半音の1/100)単位で、移調を行います。
たとえば、1オクターブ上げるには、＋1200を与えます。
<トランスポーズ値>は、－12799～＋12799です。ただし、音色によってはある高さの範囲以外は制限されることがあります。
トランスポーズはピッチとは独立して設定できます。
初期値は0です。
<トランスポーズ値2>はMSX－AUDIOとの互換性をとるために用意されているもので、指定しても無視されます。

参　照　CALL PITCH (47ページ)

CALL VOICE

Voice(ボイス:声、音声)

働　き　FM音源の各チャンネルに音色を直接設定します。

書き方　CALL VOICE([<チャンネル1用ボイスの数式>],[<チャンネル2用ボイスの数式>],・・[<チャンネル9用ボイスの数式>])

例　CALL VOICE(@0,@0,@0,,,,@7,@7,@7)

説　明　音楽用LSIの9チャンネルあるFM音源のそれぞれに音色を設定します。

音色の設定方法には2つあります。

あらかじめ設定されている音色を使って設定する場合、0～63の音色番号(次ページ参照)を、数式または、変数名により指定します。数式で表すときは、式の前に@記号をつけて、配列変数名と区別します。

例. CALL VOICE(@14,@14,@14)

プログラムにより音色データ(55ページ参照)を与えて設定する場合には、配列変数に音色データを入れて、その配列変数名を指定します。音色データの送り方は、CALL VOICE COPY文を参照してください。

パラメータを省略したチャンネルの音色は変更されません。

音色番号のうち＊印が付いていない音色や、配列変数で設定する場合は同時に1音色しか設定できません。複数設定したときは、パラメータ列のいちばん右側のパラメータか、最後に実行したCALL VOICE文の設定のみが有効となります。

例.
- CALL VOICE(@26,@27)
  チャンネル1の音色は2と同じ27番の音色になります
- CALL VOICE(@20,@10)
  CALL VOICE(,,@21)
  上記のように続けて命令すると、チャンネル1の音色は3と同じ21番となります。
- CALL VOICE(A,,B)
  チャンネル1の音色は3と同じBという配列変数で設定される音色となります。

音色一覧表

（ＶＯＩＣＥ文およびＭＭＬ（ミュージック・マクロ・ランゲージ、５７ページ参照）の＠ｎで使用できる音色の一覧表です。）

音色番号	音色名
0 *	ピアノ 1
1	ピアノ 2
2 *	バイオリン
3 *	フルート 1
4 *	クラリネット
5 *	オーボエ
6 *	トランペット
7	パイプオルガン 1
8	シロフォン
9 *	オルガン
10 *	ギター
11	サンツール 1
12 *	エレキベース
13	クラビコード 1
14 *	ハープシコード 1
15	ハープシコード 2
16 *	ビブラフォン
17	琴 1
18	太鼓
19	エンジン 1
20	ＵＦＯ
21	シンセサイザベル
22	チャイム
23 *	シンセ・ベース
24 *	シンセサイザー
25	シンセ・ドラム
26	シンセ・リズム
27	ハーモ・ドラム
28	カウベル
29	ハイハット
30	スネア・ドラム
31	バス・ドラム

音色番号	音色名
32	ピアノ 3
33 *	ウッドベース
34	サンツール 2
35	ブラス
36	フルート 2
37	クラビコード 2
38	クラビコード 3
39	琴 2
40	パイプオルガン 2
41	ＰｏｈｄｓＰＬＡ
42	ＰｏｈｄｓＰＲＡ
43	チャーチオルガンＬ
44	チャーチオルガンＲ
45	シンセ・バイオリン
46	シンセ・オルガン
47	シンセ・ブラス
48 *	ホルン
49	三味線
50	マジカル
51	フワワ
52	ワンダーフラット
53	ハードロック
54	マシーン
55	マシーン Ｖ
56	コミック
57	ＳＥーコミック
58	ＳＥーレーザー
59	ＳＥーノイズ
60	ＳＥー星 1
61	ＳＥー星 2
62	エンジン 2
63	無音

※音色名は参考のために付けたもので音色によっては、実際の楽器の音と異なることがあります。

＊印はＦＭ音源が内蔵している音色で、＊印のないものはプログラムが自動的に合成し、作り出す音色です。

CALL VOICE COPY

Voice Copy(ボイスコピー：音色を転送する)

働　き　音色データの内容をコピーします。

書き方　CALL VOICE COPY(〈パラメータ１〉〈パラメータ２〉)

例　CALL VOICE COPY(@１７,@６３)

１７番の音色データを６３番に転送する。

DIM A%(16):CALL VOICE COPY(@２８,A%)

２８番の音色データを配列変数A%に転送する。

CALL VOICE COPY(A%,@６３)

配列変数A%の音色データを63番に転送する。

説　明　配列と音色番号(０～６３)の間で、音色データの転送を行います。〈パラメータ１〉の音色データを〈パラメータ２〉に転送します。@と数式が指定された時は、その数式の結果で指定される音色番号の音色データが対象となります。

〈パラメータ１〉に指定できる音色番号は０～６３のうち＊印の付いていない音色の番号です。〈パラメータ１〉に＊印の付いている音色番号を指定すると"Illegalfunction call"となります。

〈パラメータ２〉に指定できる音色番号は６３に限ります。

@記号がない場合の変数名は配列変数とみなされ、その内容が転送の対象になります。

１つの音色データは３２バイトの長さがあります。

サンプルプログラム

```
100 CALL MUSIC
110 CLEAR:DIM A%(15)
120 CALL VOICE COPY (@25,A%)
130 A%(8)=2200
140 CALL VOICE COPY (A%,@63)
150 PLAY #2,"@25cdef"
160 PLAY #2,"@63cdef"
```

参考

音色パラメータデータのフォーマット

音色パラメータは次の表のように一音色あたり３２バイトのデータを使います。

オペレータ０とオペレータ１のデータは同じものの繰り返しになっています。

音色データ	内　　容
	ヘッダ
０～７	音色名
８～９	ボイス移調
１０	bbbbbbii　０～３：ルゴーピDDM＊ tttttttt　０～７：アフ固定MDロード可能＊ iiiiiiii　　：ルイ定MDD＊ tttttttt　　：PMD＊ bbbbbbbb　　：AMD＊
１１～１５	（予約）
	オペレータ　０
１６	bit　０～３：MULT bit　４　　：KSR bit　５　　：EG bit　６　　：PM bbbbbbbb　５～７：ベスイイレセンシビリティ（０～８）＊ iiiiiiii　　：レーレレンィ tttttttt　　：ルキイクスィテ tttttttt　　：タルケッーテシ tttttttt　４～７：ルケトトトベル
１７	
１８	
１９	
２０	
２１～２３	（予約）
	オペレータ　１
２４	bit　０～３：MULT bit　４　　：KSR bit　５　　：EG bit　６　　：PM bbbbbbbb　５～７：ベスイイレセンシビリティ（０～８）＊ iiiiiiii　　：レーレレンィ tttttttt　　：ルキイクスィテ tttttttt　　：タルケッーテシ tttttttt　４～７：ルケトトトベル
２５	
２６	
２７	
２８	
２９～３１	（予約）

＊印はＭＳＸ－ＡＵＤＩＯとの互換性のためにあるものです。ＭＳＸ－ＭＵＳＩＣでは意味を持ちません（無視されます）。

ＰＬＡＹ ＊

Play(プレイ：演奏する)

働　き　音楽をＭＭＬにしたがって演奏します。

書き方　ＰＬＡＹ〔＃〈モード〉,〕〈文字列１〉〔,〈文字列２〉〕〔,〈文字列３〉〕・・〔,〈文字列１３〉〕

例　ＰＬＡＹ ＃２,"ＣＤ","ＥＦ","ＧＡ"

説　明　ＰＬＡＹ文は音楽を演奏するもので、ＦＭ音源９音、従来のＰＳＧ音源３音の最大１２音まで同時演奏が可能です。

〈文字列〉に書かれたミュージック・マクロ・ランゲージ(ＭＭＬ)にしたがって演奏します。

他の拡張命令と異なりＣＡＬＬ文は必要ありません。

〈モード〉は０～３の値で、ＰＬＡＹ文の音源や動作モードを次のように設定します。

- ０(または省略時)はＰＳＧのみが音源となり、文字列は最大３つまでとなります。ＭＳＸ、ＭＳＸ２のＰＬＡＹ文と互換性があります。
- １の時、"Illegal function call"となります。これはＭＳＸ－ＡＵＤＩＯとの互換性をとるために用意されているものでＭＳＸ－ＭＵＳＩＣではエラーとなります。
- ２又は３の時、ＦＭ音源、リズム音、ＰＳＧ音源を使用できます(２の時と３の時で動作に違いはありません)。

〈文字列〉と音源との関係は初めから順に、

〈ＦＭ音源用文字列１〉,…,〈ＦＭ音源用文字列ｎ〉,〈リズム音用文字列〉,〈ＰＳＧ音源用文字列１〉,〈ＰＳＧ音源用文字列２〉,〈ＰＳＧ音源用文字列３〉

となります。ｎはＣＡＬＬ ＭＵＳＩＣ文で設定されたチャンネル数です。

ＣＡＬＬ ＭＵＳＩＣ文でリズム音を使用しないモードに設定した場合は、リズム音用文字列をカンマと共に省略しなければいけません。

例としてパラメータ省略時のＣＡＬＬ ＭＵＳＩＣ文に対する文字列の配列をあげると次のようになります。

〈ＦＭ音源用文字列１〉,〈ＦＭ音源用文字列２〉,〈ＦＭ音源用文字列３〉,〈リズム音用文字列〉,〈ＰＳＧ音源用文字列１〉,〈ＰＳＧ音源用文字列２〉,〈ＰＳＧ音源用文字列３〉

ＰＬＡＹ文で使用できるＭＭＬ(ミュージックマクロランゲージ)の一覧表

文　字	意　味	値の範囲	初期値＊１
Ｍｎ	エンベロープ周期の設定＊２	１≦ｎ≦６５５３５	Ｍ２５５
Ｓｎ	エンベロープ形状の設定＊２	０≦ｎ≦１５	Ｓ０
Ｖｎ	音量の設定	０≦ｎ≦１５	Ｖ８
Ｌｎ	長さの設定	１≦ｎ≦６４	Ｌ４
Ｑｎ	音の長さの割合	１≦ｎ≦８	Ｑ８
Ｏｎ	オクターブの設定	１≦ｎ≦８	Ｏ４
＞	オクターブを１つ上げる		
＜	オクターブを１つ下げる		
Ｔｎ	テンポの設定	３２≦ｎ≦２５５	Ｔ１２０
Ｎｎ	ｎで指定された高さの音を発生する	０≦ｎ≦９６	
Ｒｎ	休符の設定	１≦ｎ≦６４	Ｒ４
Ａ～Ｇ	音程の発生		
＋，＃	音を半音上げる		
－	音を半音下げる		
. (ピリオド)	音符や休符の長さを1.5倍にする(符点)		
ＸＡ＄；	文字変数Ａ＄に入っているＭＭＬを演奏する＊３		
＝ｘ；	パラメータｎを変数ｘで設定する	＊４	
＆	タイ、前後の音をつなぐ		
{｜ｎ	連符、ｎ分音符を {｜ 中の音程の個数で等分にした音を発生する	１≦ｎ≦６４	Ｌｎで設定された値
＠ｎ	ｎ番の音色に切り替える	０≦ｎ≦６３	
＠Ｖｎ	音量を細かく設定する	０≦ｎ≦１２７	
＠Ｗｎ	ｎで指定された長さだけ状態を維持する	１≦ｎ≦６４	Ｌｎで設定された値
Ｙｒ，ｄ	音楽用ＬＳＩのレジスタｒにｄを書き込む		

＊１：最初に設定されている値(ＣＡＬＬ　ＭＵＳＩＣを使ったときに設定される値)。
＊２：ＰＳＧ音源専用です。
＊３：「ＸＡ＄；」の後ろにＭＭＬを続けるとエラーになります。
＊４：値のとる範囲は直前のＭＭＬにより決定されるが、32767を越えることはできません。

リズム音用MML一覧

リズム音の場合、5種類のリズム楽器の音のうち3種類までを同時に発生できます。このため、リズム音では、まず同時に鳴らしたい楽器名を並べ、その後に次の音を発生するまでの(または終了するまでの)待ち時間を続けます。

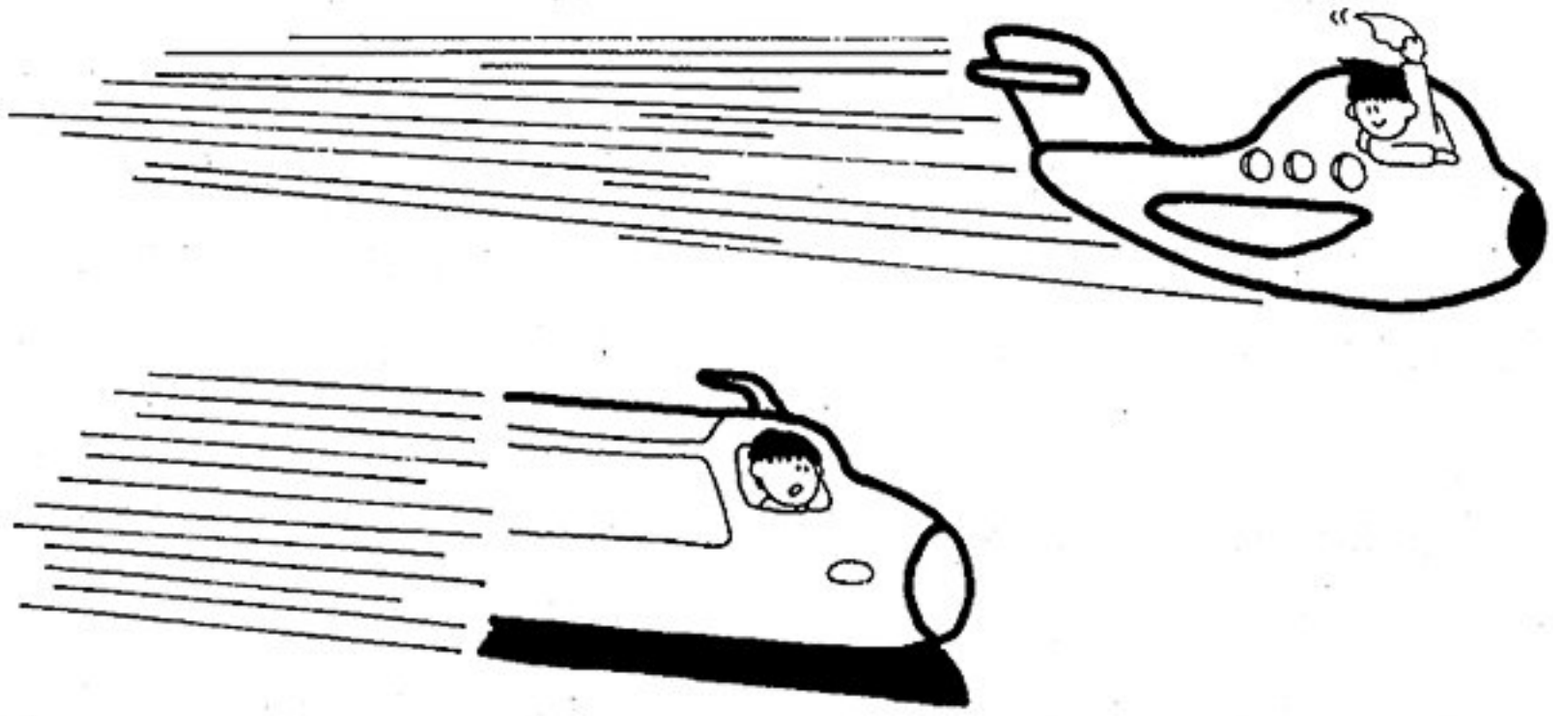
文　字	意　味	値のとる範囲	初期値
B	バスドラム音を発生		
S	スネアドラム音を発生		
M	タムタム音を発生		
C	シンバル音を発生		
H	ハイハット音を発生		
n（数値）	直前までに書かれた音楽を発生し、n分音符分待つ	1≦n≦64	
Vn	音量を設定する	0≦n≦15	8
!	直前に書かれた音楽の音量をアクセントボリュームにする		
@An	アクセントのついている音楽の音量を設定する	0≦n≦15	

「Tn」,「@Vn」,「Rn」,「XA$;」,「=x;」,「.（ピリオド)」は前ページのMMLと同じです。

例：PLAY#2,"","","","BSH8H8S!H!8H8"

- バス、スネア、ハイハットを鳴らし、8分音符分待ちます。
- ハイハットを鳴らし、8分音符分待ちます。
- スネア、ハイハットをアクセントつきで鳴らし、8分音符分待ちます。
- ハイハットを鳴らし、8分音符分待ちます。

ＢＡＳＩＣコンパイラの使い方



この「ＢＡＳＩＣコンパイラの使い方」では、ＭＳＸ ＢＡＳＩＣで作ったプログラムを高速で実行するＢＡＳＩＣコンパイラについて説明します。
ＢＡＳＩＣコンパイラは、パソコンのメモリ上にあるＢＡＳＩＣプログラムをその場で機械語プログラムに変換して実行します。ＢＡＳＩＣでリアルタイムのゲームやコンピュータグラフィックスのプログラムを作ることができます。

コンパイラとは？ －コンパイラの特長－

パソコンのプログラム言語として良く知られているものにＢＡＳＩＣがありますが、この他にＦＯＲＴＲＡＮ、ＣＯＢＯＬ、Ｃ言語などもあります。
しかし、実際にはパソコンはこれらのプログラム言語を理解できる訳ではなく、機械語というパソコン用の言語しか判りません。そこで人の作ったＢＡＳＩＣなどのプログラムを機械語に変換してあげる必要があります。

ＢＡＳＩＣとこれらのプログラム言語の大きな違いの一つに、プログラムの変換(実行)の仕方があります。ＢＡＳＩＣではプログラム命令を一つ一つ実行するインタプリタ方式で動作します。これに対して、ＦＯＲＴＲＡＮなどはプログラム命令をいったん機械語に変換して、いっきに命令を実行するコンパイラ方式で動作します。

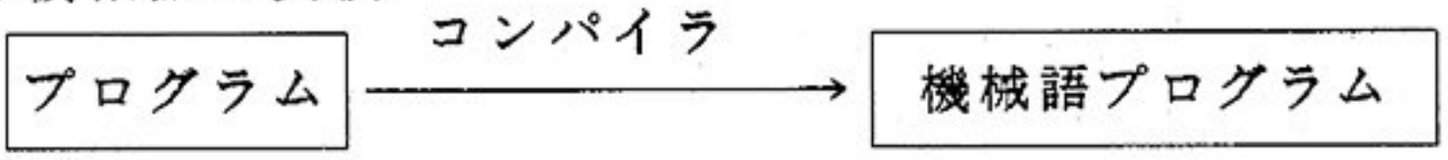
インタプリタ方式の場合は命令を一つ一つ実行するので、プログラム作成が途中でもエラーが発生するところまでの結果を見ることができます。しかし、命令を実行するときに一つずつＢＡＳＩＣ命令から機械語に変換するので、実行速度が遅くなります。

コンパイラ方式の場合はプログラムを全て機械語に変換して、変換したプログラム(機械語)で実行するので、実行速度が高速になります。しかし、プログラムがある程度完成していないとエラーなどで実行できず、途中までの結果も見ることができません。

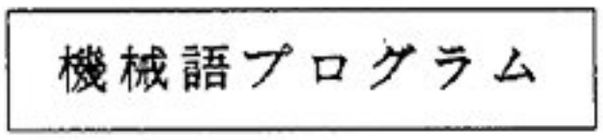
●インタプリタの場合

●コンパイラの場合

①機械語に変換



②命令の実行



ちょっと一言

■コンパイラとインタプリタ

コンパイラ(Compiler)は翻訳という意味で、インタプリタ(Interpreter)は通訳という意味です。
つまり、プログラムを機械語に翻訳(全部を変換)して実行するか、通訳(一つずつ変換)して実行するかの違いがあります。

ＢＡＳＩＣコンパイラの特長

ＢＡＳＩＣコンパイラは、メモリ上のＢＡＳＩＣプログラムをその場で機械語プログラムに変換し、機械語のプログラムとして高速で実行します。入出力命令などの一部のものを除き、普通のＭＳＸ ＢＡＳＩＣで書かれたプログラムをほとんど変更することなく機械語に変換できます。

使用方法も簡単で、従来のコンパイラ(変換ソフト)のように、

①ソースプログラム(ＢＡＳＩＣプログラム)の作成

②コンパイル→オブジェクトファイル(機械語プログラムに変換)の作成

③オブジェクトファイルの実行

といったステップは不要です。

プログラムの実行速度がＢＡＳＩＣの１５～２０倍となり、リアルタイムのゲーム、膨大な計算を必要とするコンピュータ・グラフィックス、デモプログラムなどに使用できます。

特長としては、

- 実数および文字変数が使用できます。
- 割り込みが使用できます。
- 配列の次元は自由で、メモリ容量によってのみ制限されます。
- ＭＳＸ ＢＡＳＩＣと基本的に文法が同じため、ＭＳＸ ＢＡＳＩＣでプログラムの作成・修正をして、ＢＡＳＩＣコンパイラで実行させることもできます。
- プログラムの一部だけをコンパイルして実行させることもできます。
  ＢＡＳＩＣコンパイラがサポートしていない命令・関数はＭＳＸ ＢＡＳＩＣで実行して、高速化が必要な部分をコンパイルして実行することができます。

制限としては、

- 実数の精度が低い（有効数字約４桁。２進法で言えば４．５桁）。
- ファイル入出力関係の命令・関数がサポートされていません。
- コンパイルした機械語プログラムのＳＡＶＥ、ＬＯＡＤができません。
- ＭＳＸ ＢＡＳＩＣと少し文法の違う命令があります。

[ご注意]　すでに作られているＢＡＳＩＣプログラムの全てがコンパイルして実行できる訳ではないので注意してください。お使いになる前に、この章の説明・注意をよく読んでください。

ちょっと一言

■ソースとオブジェクト

いきなりカタカナ用語が出ていますが、パソコン用の用語なのでこのまま覚えてください。ソース(Source:元の)は変換する前のプログラムの意味です。オブジェクト(Object:目的の)は実行するためのプログラムの意味です。

コンパイラの使い方

パソコンの電源を"入"にした後、キーボードからＣＡＬＬ ＢＣ↵ と入力します。画面に"Ｏｋ"と表示されたら、パソコンに内蔵されているＢＡＳＩＣコンパイラが使用可能となります(ＣＡＬＬ ＢＣはプログラム中でも実行できます)。

コンパイラの使い方

ＣＡＬＬ ＢＣ命令を実行したら、次のプログラムを実行してコンパイラの速さを確かめてみましょう。

```
10  TIME=0
20  DEFINT A-Z
30  FOR A=0 TO 250
40  B=B+A
50  NEXT A
60  PRINT B, TIME
70  END
```

ＣＡＬＬ ＲＵＮ↵ とキー入力するとプログラムがコンパイル・実行され、０から２５０の合計と計算に掛かった時間(1/60秒の単位)が表示されます。
普通に ＲＵＮ↵ で実行したときとの差を見てください。

ＣＡＬＬ ＲＵＮ↵ とキー入力したとき、ＢＡＳＩＣコンパイラで機械語に変換(コンパイル)できない命令やプログラムにエラーがあれば、変換中または実行中にエラーメッセージを出して実行を中断します。

次のプログラムではＰＬＡＹ命令がコンパイルできないので、プログラムを一部だけコンパイルするＴＵＲＢＯ命令を使います。コンパイルできない命令をＴＵＲＢＯ命令の外側に入れておきます。

```
10  SCREEN 5
20  CALL TURBO ON
30  CIRCLE (126, 106), C*10, C
40  C=C+1
50  IF INKEY$<>"" THEN END
60  IF C<15 THEN 30
70  CALL TURBO OFF
80  PLAY "CDEFG":GOTO 10
90  END
```

このプログラムを ＲＵＮ↵ すると２０～７０行の間だけがコンパイルされ、ＰＬＡＹ命令のみ普通のＢＡＳＩＣと同様に実行されます。
一つのプログラムの中にＴＵＲＢＯ ＯＮ／ＯＦＦの組は、いくつできてもかまいませんが、プログラムを ＣＡＬＬ ＲＵＮ することはできなくなります。

ＲＵＮ↵ または ＲＵＮ "ファイル名"↵ で実行できます。

コンパイラ用の命令

ＢＡＳＩＣコンパイラ用の命令は、次の４つだけです。（拡張命令については、６８ページを参照してください。）また、ＣＡＬＬは"_（アンダーバー）"でも代用できます。

ＣＡＬＬ　ＢＣ

BC（BACICコンパイラ）

働き　ＢＡＳＩＣコンパイラを使用可能にします。

書き方　ＣＡＬＬ　ＢＣ

説明　この命令により、ＢＡＳＩＣコンパイラが使用可能な状態となります。一度実行すると、リセットしたり電源を切らない限り、ＢＡＳＩＣコンパイラの機能が使用できます。

この命令は、ＢＡＳＩＣコンパイラをシステムに登録するのみです。メモリの確保はしません。

ＢＡＳＩＣコンパイラは基本的には、ＭＳＸ　ＢＡＳＩＣのみをサポートしています。ＭＳＸ2、ＭＳＸ2+特有の命令は使用できません。ただし、ＳＣＲＥＥＮ命令に関しては１２まで指定できます。また、漢字モードの状態でＢＡＳＩＣコンパイラの使用はできますが、逆(BASICコンパイラでCALL KANJI)はできません。

ＭＳＸ-ＤＯＳおよびＲＡＭディスク機能、メモリ・マッパー機能とは同時に使えません。

ＭＳＸ-ＤＯＳなどの機能を終了(解除)したときに、CALL BC 命令の再実行が必要です。

ＣＡＬＬ　ＲＵＮ

RUN（ラン:実行）

働き　ＢＡＳＩＣプログラムをすべてコンパイル(機械語に変換)して、プログラムを実行します。

書き方　ＣＡＬＬ　ＲＵＮ

説明　パソコンのメモリ上にあるＢＡＳＩＣプログラムを、全て機械語に変換し、実行します。

この命令を実行する前に、ＣＡＬＬ　ＢＣ命令を実行しておく必要があります。また、この命令はプログラム中で実行することはできません。

CALL TURBO ON
CALL TURBO OFF

TURBO ON/OFF（ターボ・オン/オフ）

働　き　BASICプログラム中で、プログラムの一部をコンパイルします。

書き方　CALL TURBO ON〔(<変数>)〔,<変数>〕･･･〔,<配列>)〕
CALL TURBO OFF

説　明　プログラム中にBASICコンパイラでコンパイルできない命令や関数があるとき、プログラムの一部をコンパイルして機械語として実行します。
TURBO ONがコンパイルの開始を、TURBO OFFがコンパイルの終わりを示します。
CALL TURBO ON／OFFはプログラム中でのみ使用でき、ダイレクトモードでは実行できません。

CALL TURBO ON、CALL TURBO OFF命令はマルチステートメント中には置けません。必ず独立した1行の中におさめてください。

悪い例）
```
100 IF A=0 THEN CALL TURBO
    OFF
```
良い例）
```
100 IF A=0 THEN 110 ELSE ····
110 CALL TURBO OFF
```

CALL TURBO ON、CALL TURBO OFF命令は文章中に何個でも指定できますが、入れ子の状態にしては行けません。また、GOTO命令などでTURBO命令で囲まれた中から、範囲の外へ実行を移すことはできません。

悪い例）
```
10 SCREEN 5
20 CALL TURBO ON
30 CIRCLE (126, 106), C*10, C
40 C=C+1
50 IF INKEY$<>"" THEN 90
60 IF C<15 THEN 30
70 CALL TURBO OFF
80 PLAY "CDEFG":GOTO 10
90 END
```

上の例では50行のTHEN 90により、TURBO OFF命令のある70行でUndefined line number エラーとなります。50行の90をENDに修正すると、動作します。

ＴＵＲＢＯ ＯＮ／ＯＦＦ命令を使うとき、変数や配列の使い方に次の制限があります。

普通のＢＡＳＩＣの部分と、ＴＵＲＢＯ命令で囲まれた部分では、変数や配列は別々に定義されます。同じ変数名で指定しても異なった値を持ちます。
また、ＴＵＲＢＯ ＯＦＦで１つのブロックが終わると、その中で使われていた変数や配列はすべて無効となります。
ただし整数型の変数と配列に限って、共通の変数として宣言することができます。このときはＣＡＬＬ ＴＵＲＢＯ ＯＮの後に必要な変数、配列名をかっこでくくって並べます。
ただし、ＴＵＲＢＯ命令の前に宣言文などの実行文で使用された変数、配列のみです。

例） ＣＡＬＬ ＴＵＲＢＯ ＯＮ（Ａ，Ｂ％，Ｃ（），ＤＥ％（））

ＢＡＳＩＣコンパイラで扱われる実数型の変数は普通のＢＡＳＩＣとは少し違っています。
型は"！"マークで表す単精度の変数も"＃"の倍精度の変数も、精度は約４桁で、範囲は２．９３９Ｅ－３９～１．７０Ｅ＋３８です。
また、１０，０００以上の数は「指数形式」で表示されます。精度が多少低いので、普通のＢＡＳＩＣとは違った計算結果となることもあります。

配列の宣言を省略することはできません。
配列はプログラムの一番初めの行(ＣＡＬＬ ＴＵＲＢＯ ＯＮのときはその直後)に宣言してください。宣言していない配列は扱うことができません。配列は何次元でもメモリの許す限り使えます。

無限ループを作る(１０ ＧＯＴＯ １０のような)と、CTRL＋STOP でもプログラムを止めることができなくなります。必要な場合はキー入力を受けつける部分を作ってください。

例） １０ ＩＦ ＩＮＫＥＹ＄ ＝"" ＴＨＥＮ １０

使える命令と使えない命令

ＢＡＳＩＣコンパイラは、基本的にＭＳＸ ＢＡＳＩＣの命令と関数をサポートしていますが、コンパイルできない(機械語に変換できない)命令、関数があります。次の命令を含んだ状態でコンパイルするとエラーとなります。

使えない命令

A

AUTO

B

BASE、BLOAD、BSAVE

C

CALL、CDBL、CINT、
CLEAR、CLOAD、CLOAD?、
CLOSE、CONT、CSAVE、
CSNG、CVD、CVI、CVS

D

DEFFN、DELETE、DRAW、
DSKF

E

EOF、EQV、ERASE、ERL、
ERR、ERROR

F

FIELD、FILES、FPOS、
FRE

G

GET

I

IMP、INPUT#

K

KEY LIST、KILL

L

LFILES、LINE INPUT#、
LIST、LLIST、LOAD、LOC
LOF、LPRINT USING、
LSET

M

MAXFILES、MERGE、
MKD$、MKI$、MKS$、
MOTOR

N

NEW、NAME

O

ON ERROR GOTO、OPEN

P

PLAY、PRINT#、PRINT#
USING、PRINT USING、
PUT KANJI

R

RENUM、RESUME、RSET

S

SAVE、SPC、

T

TAB、TRON、TROFF

W

WIDTH

使い方が変わる命令

つぎの命令、関数は使用方法が限られます。

ＣＩＲＣＬＥ	開始、終了の角度および比率は指定できません。
ＣＯＰＹ	グラフィック画面上でのＣＯＰＹ命令以外は使えません。（ディスクや配列へのコピーはできません。）
ＤＥＦＤＢＬ	ＤＥＦＳＮＧと同じです。Ａ！とＡ＃は同じ変数を意味します。（計算を速く行いたいときは、ＤＥＦＩＮＴ 命令で整数宣言した変数を使ってください。）
ＤＩＭ	すべての実行文に先だって宣言します。添字は整数の定数(変数でない)で指定します。ＤＩＭ文の前に置けるのは ＤＥＦＩＮＴ、ＤＥＦＳＮＧ、ＤＥＦＤＢＬ、ＤＡＴＡ、ＤＩＭ、コメント文だけです。
ＩＮＰＵＴ	一度に代入できる変数は１つだけです。
ＫＥＹ	ＯＮ ＫＥＹ ＧＯＳＵＢ 、ＫＥＹ（Ｎ） ＯＮなど以外では使えません(例．ファンクションキーに命令は設定できません)。
ＬＯＣＡＴＥ	Ｘ、Ｙ座標は必ずセットで指定し、カーソルスイッチは指定できません。
ＮＥＸＴ	ＮＥＸＴの後の変数名は省略できません。
ＯＮ	ＯＮ ＳＴＯＰ ＧＯＳＵＢ命令は使えません。
ＰＲＩＮＴ	カンマの働きが違います。普通のＢＡＳＩＣではカンマは有効数字の最大桁数に相当するスペースを出力しますが、ＢＡＳＩＣコンパイラではタブコードを出力するだけです。また、数値を出力するとき画面上のラップは行われません。
ＰＵＴ	ＰＵＴ ＳＰＲＩＴＥ命令以外は使えません。
ＲＵＮ	変数の初期化は行われません。プログラム中にあると、一番初めの行に実行が移りますが、変数の初期化はしません。
ＳＣＲＥＥＮ	画面モード、スプライトサイズ以外は指定できません。画面モードは、ＭＳＸ2+のＳＣＲＥＥＮ １２まで指定できます。
ＳＥＴ	ＳＥＴ ＰＡＧＥ命令とＳＥＴ ＳＣＲＯＬＬ命令以外は使えません。
ＳＴＯＰ	ＥＮＤ文と同じ意味になります。
ＵＳＲ	引数として使える値は整数のみです。
ＶＲＰＴＲ	ファイル番号を引数として指定することはできません。

これ以外の命令、関数は普通のＢＡＳＩＣと同じように扱うことができます。

こんなこともできます

ＢＡＳＩＣコンパイラにはＣＡＬＬ ＢＣとＣＡＬＬ ＲＵＮ、ＣＡＬＬ ＴＵＲＢＯ ＯＮ／ＯＦＦという拡張命令がありますが、このほかに次の拡張命令があります。

- Ｙ座標のクリッピングをＯＮ/ＯＦＦする …… ＃Ｃ
- ＮＥＸＴ文のオーバーフローをチェックする …… ＃Ｎ
- プログラム中に機械語プログラムを入れる …… ＃Ｉ

ＢＡＳＩＣコンパイラの拡張命令は、コメント文を「＃」で始めることで指定されます。（コメント文は「ＲＥＭ」で始めても「'」でもかまいません。）また、「＃」の後にくるキャラクタは大文字でも小文字でもかまいません。

＃Ｃ

Clip（クリップ：止める）

働　き　Ｙ座標指定のクリッピングを行うかどうかを指定します。

書き方　'＃Ｃ＋ …… クリッピングＯＮ（ＭＳＸ ＢＡＳＩＣと同じ）
　　'＃Ｃ－ …… クリッピングＯＦＦ

例　　１００ '＃Ｃ＋

説　明　クリッピングとは、ＬＩＮＥ命令などを実行するとき、Ｙ座標の指定値が表示範囲を越えるときに、値を制限して命令を行う機能です。
「＃Ｃ＋」でクリッピングをＯＮにし(ＭＳＸ ＢＡＳＩＣと同じ)、「＃Ｃ－」でクリッピングをＯＦＦにします。ＢＡＳＩＣコンパイラの起動時にはＯＮが設定されています。

例).

```
１０ ＳＣＲＥＥＮ ８
２０ '＃Ｃ－
３０ ＬＩＮＥ（０，０）－（２５５，２５５）
４０ ＩＦ ＩＮＫＥＹ＄＝"" ＴＨＥＮ ４０
５０ '＃Ｃ＋
６０ ＬＩＮＥ（０，０）－（２５５，２５５）
７０ ＩＦ ＩＮＫＥＹ＄＝"" ＴＨＥＮ ７０
```

注　意　ＢＡＳＩＣコンパイラでは、ＳＣＲＥＥＮ０～３の画面でのグラフィック処理、ＰＡＩＮＴ命令、ＣＩＲＣＬＥ命令の実行には、ＭＳＸ ＢＡＳＩＣと同じ処理を行うためクリッピングの指定は意味を持ちません。

＃Ｎ

Next（ネクスト:次の）

働　き　NEXT文の実行時に変数のオーバーフローをチェックします。

書き方　'＃Ｎ＋ ····· オーバーフローをチェックします。
'＃Ｎ－ ····· オーバーフローをチェックしません。

例　　１０ '＃Ｎ＋

説　明　ＦＯＲ～ＮＥＸＴ文のコンパイル時にオーバーフローをチェックするコードを作成するかどうかを決めます。

「＃Ｎ＋」とすると、それ以降のＮＥＸＴ文のコンパイルにおいてオーバーフローをチェックし、オーバーフローならループから脱出するプログラムを作成します。
「＃Ｎ－」とすると、オーバーフローをチェックしないプログラムを作成します。ＢＡＳＩＣコンパイラ起動時には「＃Ｎ＋」が設定されています。

例).

```
１０　ＦＯＲ　Ｉ％＝０　ＴＯ　＆Ｈ７ＦＦＦ
２０　ＮＥＸＴ　Ｉ％
```

このプログラムをＭＳＸ　ＢＡＳＩＣで実行させると、ループの終了直前にエラーが起きます。これは、＆Ｈ７ＦＦＦに１をたした結果、値が整数の範囲を越えたことによって起こります。
ＢＡＳＩＣコンパイラで実行させると、無限ループになります。
＆Ｈ７ＦＦＦより大きい整数はコンパイラでは存在しないからです。
このような状態を回避したい時にこの命令を用いてください。

注　意　「＃Ｎ＋」とすると、コンパイルするプログラムのサイズ、実行時間共に増加します。
また、この命令を使ったプログラムはＭＳＸ　ＢＡＳＩＣと異なった動作となります。

＃ I

Inline(インライン:文中に入る)

働　き　プログラム中に機械語プログラムを記述します。

書き方　'＃Ｉ ····· この行に機械語のプログラムを記述します。

例　　１００ '＃Ｉ ＆Ｈ１Ｂ

説　明　数字を書くとその値(バイト)がメモリに置かれます。変数名(整数型のみ)を書くと、コンパイル時にその変数の数値格納アドレスがプログラムに置かれます。

"@" が頭に付いた１０進数を書くと、その数で示されるＢＡＳＩＣプログラムの行番号に実行が移ることを意味します。 この行番号はコメント行にあるため、ＲＥＮＵＭ命令では行番号の変更がされないので注意してください。

例).グラフィック画面用文字出力ルーチン

```
１０ ＳＣＲＥＥＮ ５：Ｘ％＝０：Ｙ％＝０
２０ ＰＲＥＳＥＴ（０，０）
３０ ＦＯＲ Ｃ％＝０ ＴＯ ２５５
４０ ＧＯＳＵＢ １００
５０ ＮＥＸＴ Ｃ％
６０ ＩＦ ＩＮＫＥＹ＄＝"" ＴＨＥＮ ４０
７０ ＥＮＤ
１００ '＃Ｉ ＆Ｈ２Ａ，Ｃ％，＆Ｈ２９，＆Ｈ２９，＆Ｈ２９
，＆Ｈ１１，＆ＨＢＦ，＆Ｈ１Ｂ，＆Ｈ１９
１１０ '＃Ｉ ＆Ｈ０６，＆Ｈ０８，＆Ｈ３Ａ，＆ＨＢ７，＆
ＨＦＣ，＆Ｈ３２，Ｘ％，＆Ｈ３Ａ，＆ＨＢ９，＆ＨＦＣ，＆Ｈ
３２，Ｙ％
１２０ '＃Ｉ ＆Ｈ７Ｅ，＆Ｈ２３，＆ＨＣ５，＆Ｈ０６，＆
Ｈ０８
１３０ '＃Ｉ ＆Ｈ１７，＆ＨＦ５，＆ＨＣ５，＆ＨＥ５，＆
ＨＤＣ，＠１７０，＆ＨＥ１，＆ＨＣ１，＆Ｈ３Ａ，Ｘ％，＆Ｈ
３Ｃ，＆Ｈ３２，Ｘ％，＆ＨＦ１
１４０ '＃Ｉ ＆Ｈ１０，＆ＨＥＤ，＆Ｈ３Ａ，Ｘ％，＆ＨＤ
６，＆Ｈ０８，＆Ｈ３２，Ｘ％，＆Ｈ３Ａ，Ｙ％，＆Ｈ３Ｃ，＆
Ｈ３２，Ｙ％，＆ＨＣ１，＆Ｈ１０，＆ＨＤ６
１５０ '＃Ｉ ＆Ｈ３Ａ，Ｘ％，＆ＨＣ６，＆Ｈ０８，＆Ｈ３
２，＆ＨＢ７，＆ＨＦＣ，＆Ｈ３Ａ，Ｙ％，＆Ｈ３８，２，＆Ｈ
Ｄ６，＆Ｈ０８
１６０ '＃Ｉ ＆Ｈ３２，＆ＨＢ９，＆ＨＦＣ，＆ＨＣ９
１７０ ＰＳＥＴ（Ｘ％，Ｙ％）：ＲＥＴＵＲＮ
```

前ページのプログラムは、ＢＡＳＩＣコンパイラを使うとき、ＯＰＥＮ命令、ＰＲＩＮＴ＃命令が使えないので、それを補うためのグラフィック画面に文字を出力するサブルーチン用プログラムです。
２０行のＰＲＥＳＥＴ文で出力位置を決め、３０～５０行のＦＯＲ～ＮＥＸＴループでＣ％に文字のキャラクタコードを代入しています。次に４０行で機械語サブルーチンを呼び出しています。１００～１６０行までが機械語サブルーチンですが、その中からさらに１７０行のＢＡＳＩＣサブルーチンを呼び出す形式になっているので、自作のプログラムに組み込む際には１７０行を忘れないでください。
この例の場合には、１３０行中の"@１７０"です。この"@１７０"はＲＥＮＵＭ命令で修正されないので注意してください。

注意点

- Ｘ％、Ｙ％は整数型であれば別の変数名でもかまいません。
- このプログラムでは、Ｘ座標は０～２５５までです。
- 重なったグラフイック画面は、ＯＲで表示します。
- "月火水木金"などのグラフィックキャラクタはＭＳＸ ＢＡＳＩＣと異なり、直接１０進数のキャラクタコードを代入できます。

プログラム作成時のご注意

ＢＡＳＩＣコンパイラはＭＳＸ ＢＡＳＩＣと互換性があるように設計されていますが、多少の相違点があります。
コンパイルしたときにエラーが発生する。コンパイルすると動作が異なる。このようなとき、次の点をチェックしてください。

変数関係

● ＢＡＳＩＣの部分とＣＡＬＬ ＴＵＲＢＯ ＯＮの部分で変数、配列を共用する場合、あらかじめ定義された変数、配列のみが使用できます。
もし、定義していない変数や配列を共用しようとすると "Illegal function call" のエラーが起こります。変数や配列の定義が正しくされているか確認してください。

● 各命令の引数や添字の範囲のチェックは行われません（ただし配列の次元はコンパイル時にチェックされます）。
ＭＳＸ2、ＭＳＸ2+ 特有の命令をコンパイルした場合の動作は保証されません。ＢＡＳＩＣコンパイラは動作の高速化のためにエラーチェックが簡略化されています。ＭＳＸ ＢＡＳＩＣで正常に動作するのを確認してから、コンパイルしてください。

● ＦＯＲ～ＮＥＸＴ文を正しく対応させてください。また、変数の省略はできません。

● ＭＳＸ ＢＡＳＩＣでは、要素数が１０までの配列は宣言しなくても使えますが、ＢＡＳＩＣコンパイラでは、配列は必ずＤＩＭ文を使って宣言しなければいけません。

● ＢＡＳＩＣコンパイラでの宣言(ＤＥＦＩＮＴ、ＤＥＦＳＮＧ、ＤＥＦＤＢＬ、コメント中の＃Ｎ)は、コンパイル中に効果を発生するのではなく、実行時に有効となります。
このため、プログラム中に現れる順番に処理されます。例えば次のプログラムではＭＳＸ ＢＡＳＩＣとコンパイラでは実行結果が異なります。

```
１０  ＧＯＳＵＢ ５０
２０  Ａ＝１００／３
３０  ＰＲＩＮＴ Ａ
４０  ＥＮＤ
５０  ＤＥＦＩＮＴ Ａ
６０  ＲＥＴＵＲＮ
```

● 文字列および文字型の変数、配列、関数、演算は普通のＢＡＳＩＣ同様使えますが、一つの変数（または配列の一要素）ごとに２５６バイトを使用しますので、同じ変数名を繰り返し使うなど、メモリの使用に注意が必要です。

演算関係

● ＭＳＸ ＢＡＳＩＣでは、１０段の文字列のスタック(演算処理用のメモリ)を持っていて、これによって複雑な文字列演算を実行できますが、コンパイラでは文字列のスタックが１段しかありません。このため、ＭＳＸ ＢＡＳＩＣで実行できた文字列の演算がコンパイルできないこともあります。

＊文字列のスタックを使う命令とは、文字列の連結(たし算)と文字列の比較(大小、長さ)です。

```
１０　Ｂ＄＝” ４１”
２０　Ａ＄＝Ａ＄＋Ｃ＄＋ＣＨＲ＄（ＶＡＬ（” ＆Ｈ” ＋Ｂ＄））
```

上の例ではコンパイルを行うと、"String formula too complex in 20" というエラーが発生します。このエラーをさけるためには、文字列の演算をできるだけ分割するようにしてください。
上のプログラム例では次のように変更すると、正常に実行できます。

```
１０　Ｂ＄＝” ４１”
２０　Ｃ＄＝ＣＨＲ＄（ＶＡＬ（” ＆Ｈ” ＋Ｂ＄））
３０　Ａ＄＝Ａ＄＋Ｃ＄
```

● ＢＡＳＩＣコンパイラでは整数同士の演算結果は、割り算、べき乗の演算と特別の場合を除き、必ず整数になると仮定しています。
例えばＡ％が２００の時、Ａ％＊Ａ％はＭＳＸ ＢＡＳＩＣでは、浮動少数の４００００になりますが、コンパイラでは－２５５３６になります（オーバフローを無視しているため）。
演算の前に単精度実数の型宣言を入れ、１！＊Ａ％＊Ａ％とするとＭＳＸ ＢＡＳＩＣと同一の結果となります。
「特別の場合」とは、「式の演算結果が浮動少数になるということが、前もって分かっている場合」です。
例えば、上の例のように実数との演算、浮動少数変数への代入、浮動少数関数の引数の評価などを行うときです。ただし、この演算方法は完璧ではありません。ＬＯＧ（１＋Ａ％＊Ａ％）のように優先順位が高い演算が右にくる場合は、ＭＳＸ ＢＡＳＩＣと同じ結果になりません。

ＵＳＲ関数・プログラム制御

● ＵＳＲ関数の実行時のメモリマップはＭＳＸ ＢＡＳＩＣとＢＡＳＩＣコンパイラで結果が違います。
ＭＳＸ ＢＡＳＩＣの時は、ページ１のメモリ上(&Ｈ４０００～&Ｈ７ＦＦＦ)にＭＳＸ ＢＡＳＩＣが存在しますが、コンパイラの時はコンパイラが存在します。

● ＵＳＲ関数に渡したり受け取ったりする値は、整数に限ります。

● 割り込みを使用するとプログラムサイズ、実行時間共に増加します。

速いプログラムを作るには

● 変数はできるだけ整数型にしてください。
プログラムの初めにＤＥＦＩＮＴ　Ａ－Ｚ文などを設定してください。

● プログラム中の演算子には、「／」、「＾」をなるべく使わず、「¥」、「＊」を使うようにします。

● 整数の割り算の時は、なるべく２のＮ乗の数(２，４，８，･･･)で割るようにしてください。

● 整数の掛け算の時は、なるべく２のＮ乗の数か、３，５，６，７，９，１０，２０，２５，４０，５０，８０，１００，２００，２５６，２５７で掛けるようにしてください。

● 配列の要素数は、なるべく「上記の数－１」の値になるようにしてください。
２５５がメモリの使用効率の点から、一番効率が良くなります。
また多次元の配列を宣言する場合、この値の添字がなるべく左にくるように宣言してください。

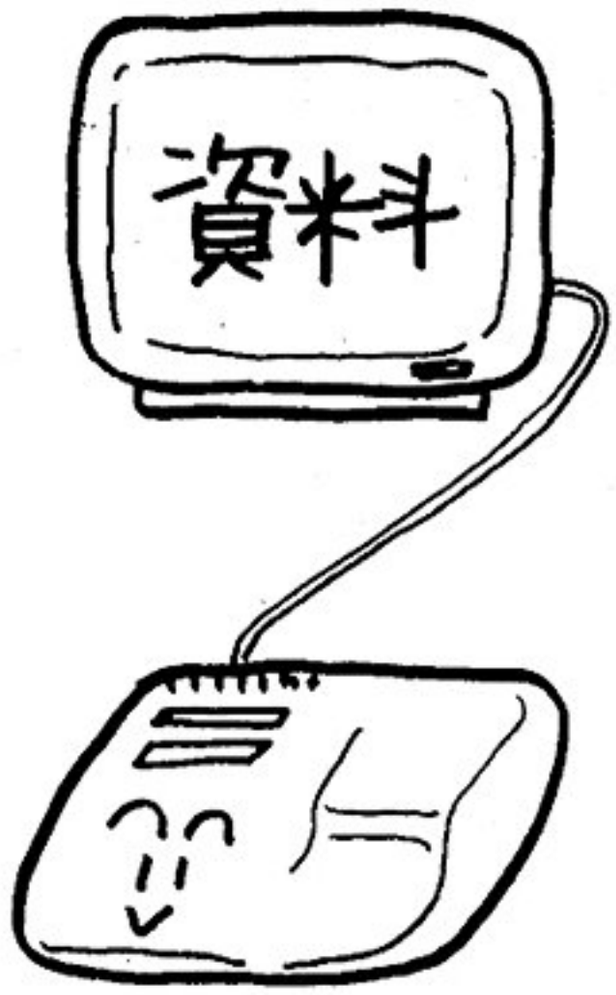
メモリの制限

● ＢＡＳＩＣコンパイラはコンパイルされる(ＢＡＳＩＣの)プログラムと、コンパイルされたプログラムが同じＲＡＭ(３２Ｋバイト)上にあるので、メモリの使用できる制限がきびしくなります。
大体の目安として１０Ｋバイトぐらいのプログラムは、コンパイルして実行することができますが、次のような場合は動作しなくなることがあります。

- ＣＡＬＬ　ＫＡＮＪＩ命令、ディスクドライブなどによってメモリのフリーエリアの上限が限られている場合。
- ＣＬＥＡＲ、ＭＡＸＦＩＬＥＳ命令などで余分にメモリが使われている場合。
- すでに非常に圧縮したＢＡＳＩＣプログラムで、余分なスペースなどがほとんど入っていない場合。
- 割り込み処理を使ったプログラムの場合。
- 多くの文字列変数や、大きな配列を使っている場合。

以上のこと以外で Out of memory エラーとなるときは、できるだけ整数型の変数を使い、同じ処理をする部分はサブルーチン化し、余分なスペースはつめ、できる限りマルチステートメントを使って行数を減らすなどして、メモリを節約する必要があります。

資　料



この章では、ＭＳＸ－ＭＵＳＩＣ、ＢＡＳＩＣコンパイラを使ったサンプルプログラムを記載しています。
各機能を使ったプログラムの作成の参考にしてください。
また、索引では、機能およびコマンドの説明してあるページを示しています。

サンプルプログラム

MSX-MUSIC

FM音源のリズム音を使って、繰り返し演奏するプログラムです。

```
100 CALL MUSIC(1,0,1,1,1,1,1,1)
110 CALL BGM(1)
120 CALL VOICE(@48,@48,@6 ,@6 ,@14,@14)
130 T$="T150"
140 PLAY #2,T$,T$,T$,T$,T$,T$,T$
150 READ A$
160 IF A$="end" THEN RESTORE:GOTO 150
170 READ B$,C$,D$,E$,F$,G$
180 PLAY #2,A$,B$,C$,D$,E$,F$,G$
190 GOTO 150
200 DATA V15R1L8
210 DATA V15R1L8
220 DATA V15R1L8
230 DATA V15R1L8
240 DATA V12L1605D8DDD8DDD8DDD8DD
250 DATA V15L803DDDDDDD
260 DATA v10CBH8H16H16SH8BH16H16BH8H16H1
6SH8BH16H16
270 DATA R805GR8GR8F4R8
280 DATA R805DR8DR8C4R8
290 DATA R804AR8AR8G4R8
300 DATA R1
310 DATA O5D8DDD8DDD8DDD8DD
320 DATA DO4CDO3D404CDO3D
330 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8BH16H16
340 DATA R1
350 DATA R1
360 DATA R1
370 DATA R1
380 DATA D8DDD8DDD8DDD8DD
390 DATA DDDDDDDD
400 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8BH16H16
410 DATA R805GR8GR8F4R8
420 DATA R805DR8DR8C4R8
430 DATA R804AR8AR8G4R8
440 DATA R1
450 DATA O5D8DDD8DDD8DDD8DD
460 DATA DO4CDO3D404CDO3D
470 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8BH16H16
```

```
480 DATA R1
490 DATA R1
500 DATA R1
510 DATA R1
520 DATA D8DDD8DDD8DDD8DD
530 DATA DDDDDDDD
540 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8BH16H16
550 DATA R805GR8GR8F8R8
560 DATA R805DR8DR8C8R8
570 DATA R804AR8AR8G8R8
580 DATA R2R4R8
590 DATA O5D8DDD8DDD8DDD8
600 DATA DO4CDO3D404CD
610 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8
620 DATA D1R8
630 DATA O4A1R8
640 DATA F1R8
650 DATA R8R1
660 DATA DDD8DDD8DDD8DDD8DD
670 DATA O2B-4B-B-B-B-B-B-B-
680 DATA CBH4H16H16SH8BH16H16BH8H16H16SH
8BH16H16
690 DATA F1
700 DATA O5D1
710 DATA A1
720 DATA R1
730 DATA D8DDD8DDD8DDD8DD
740 DATA B-B-B-B-B-B-B-B-
750 DATA CBH8H16H16SH8BH16H16BH8H16H16SH
8BH16H16
760 DATA R1L2O5E-.C8D8
770 DATA R2R404B-1O5C8D8
780 DATA R2L2O4F.FC8D8
790 DATA R4L1O4C.C8D8
800 DATA O3G1R1
810 DATA O3C1R1
820 DATA R1R1
830 DATA end
```

BASICコンパイラ

ＳＣＲＥＥＮ　２で惑星(スプライト)を回転させます。
プログラムを実行すると、数字(表示のステップ)の入力を求めてきますので、
１００～３００を指定してください．最大９個の惑星が回転します。

```
100 DEFINT A-Z:DEFSNG A,B,H,I,K,M,P
110 DIM C(14),X(300,10),Y(300,10)
120 COLOR 15,0,0:SCREEN 0
130 INPUT "NUMBER(100-300)";L
140 FOR J=20 TO L
150  IF L MOD J=0 THEN C(N)=J:N=N+1
160 NEXT J
170 PRINT N;"PLANETS"
180 C=N-1:H=90/N:N=0:D=L
190 '
200 P=ATN(1)*4
210 PRINT "WAIT"
220 FOR J=0 TO C
230  K=RND(1)*P/2:L=C(J)/2:M=H*(J+1)
240  FOR I=0 TO P*2 STEP P/L
250   PRINTC-J;P*2-I
260   A=COS(I+K):B=SIN(I)
270   X(N,J)=A*M*1.5+128
280   Y(N,J)=B*M+96
290   IF X(N,J)>255 OR X(N,J)<0 THEN X(N
,J)=-2:Y(N,J)=218
300   N=N+1
310  NEXT I
320  N=0
330 NEXT J
340 SCREEN 2,0
350 CIRCLE(128,96),4:PAINT(128,96)
360 SPRITE$(0)=CHR$(192)+CHR$(192)
370 IF INKEY$="" THEN 380 ELSE END
380 FOR J=0 TO C
390  PUT SPRITE J,(X(N MOD C(J),J),Y(N M
OD C(J),J)),15-J*2,0
400 NEXT J
410 N=(N+1) MOD D
420 GOTO 370
```

漢字モード

ＳＣＲＥＥＮ １と５を使ってバイオリズムを表示します。
プログラムを実行するまえに、ＳＥＴ　ＤＡＴＥ命令で今日の日付けに合わせてください。
プログラムを実行すると、あなたの誕生日を、きいてきますので入力してください。

```
100 '** ﾊﾞｲｵﾘｽﾞﾑ ｹｲｻﾝ ***
110 CALL KANJI:CALL BC
120 SCREEN 1:WIDTH 60:DEFINT A-Z
130 GET DATE D$
140 TY=VAL("19"+LEFT$(D$,2))
150 TM=VAL(MID$(D$,4,2))
160 TD=VAL(RIGHT$(D$,2))
170 INPUT "あなたの誕生年は";YY
180 IF YY<1900 THEN YY=YY+1925
190 INPUT "              月は";YM
200 INPUT "              日は";YD
210 YT!=INT(365.25*YY)+INT(30.6*YM)+1+YD
220 TT!=INT(365.25*TY)+INT(30.6*TM)+1+TD
230 DS!=TT!-YT!
240 SCREEN 5:CALL KANJI1
250 COLOR 2:LOCATE 4,1:PRINT "---身体"
260 COLOR 8:LOCATE 14,1:PRINT "---感情"
270 COLOR 11:LOCATE 24,1:PRINT "---知性"
280 LINE(0,120)-(255,120),15
290 LINE(40,40)-(40,180),15
300 COLOR 15:LOCATE 4,2:PRINT"今日"
310 FOR X=0 TO 255 STEP 4:IF(XMOD40)<>0 THEN L=14 ELSE L=15
320 T3=31:IF TM=2 THEN T3=28:IF (TYMOD4)=0 THEN T3=29
330 IF TM=4 OR TM=6 OR TM=9 OR TM=11 THEN T3=30
340 IF X<40 THEN 360 ELSE T2=TD+X/4-10-1
350 IF(T2MODT3)=0THEN380
360 LINE(X,118)-(X,122),L
370 NEXT X:GOTO 420
380 TM=TM+1:TD=TD-T3:IF TM>12THEN TM=1
390 LINE(X,68)-(X,122),7:COLOR7
400 IF X>220 GOTO 370
410 LOCATE X/6,3:PRINT TM;"月":GOTO 370
420 R!=-2*3.1415926535898#
430 A=2:S=23:PSET(-5,120):GOSUB 500
440 A=8:S=28:PSET(-5,120):GOSUB 500
450 A=11:S=33:PSET(-5,120):GOSUB 500
460 LOCATE 2,4:PRINT"活発"
470 LOCATE 1,8:PRINT"不安定"
480 LOCATE 2,12:PRINT"低調";
490 A$=INKEY$:IF A$="" THEN 490 ELSE 540
500 FOR X=0 TO 260 STEP 4
510 P=45*SIN(R!*(DS!+X/4-10)/S)+120
520 LINE -(X,P),A
530 NEXT X:RETURN
540 END
```

索　引

三洋電機株式会社

2CA6P1A0072--Ⓑ(Y3FL)