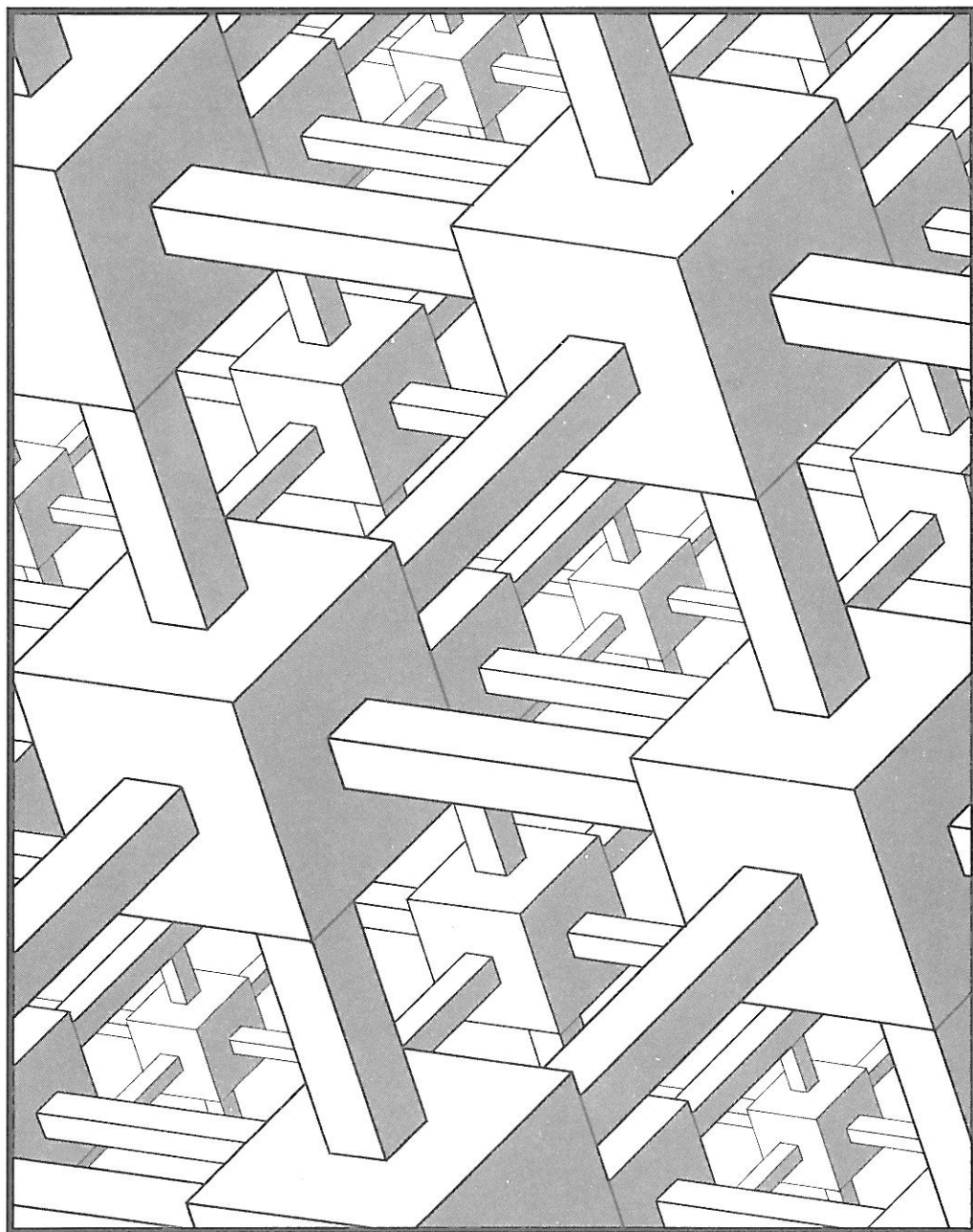


MSX-Datapack

エムエスエックス・データパック

Volume **2**



ASCII

このマニュアルの読み方

このマニュアルの構成

このマニュアルは以下のように構成されています。

Volume 1

第1部 ハードウェア

第2部 システムソフトウェア

第3部 MSX-DOS

第4部 VDP

Volume 2

第5部 スロットとカートリッジ

スロットの切り換えなど、ソフトウェアから見たスロットやカートリッジについて解説します。

第6部 標準的な周辺装置のアクセス

MSX に標準で実装されている周辺装置 (PSG、カセットインターフェイス、キーボードなど) の使用法を解説します。

第7部 オプションの周辺装置のアクセス

MSX の仕様ではオプションの周辺装置 (RS-232C、MSX MODEM、MSX-MUSIC など) の使用法を解説します。

APPENDIX

BIOS やワークエリアなどの一覧表です。

このマニュアルの表記

1. MSX のバージョン

MSX1	MSX BASIC version 1.0 を搭載した MSX
MSX2	MSX BASIC version 2.0 を搭載した MSX
MSX2+	MSX BASIC version 3.0 を搭載した MSX
MSX	MSX1、MSX2、MSX2+ の総称

2. キー

2つのキーが「+」で結ばれているときは、左側のキーを押しながら、右側のキーを押します。例えば、**CTRL** + **STOP** は、**CTRL** キーを押しながら **STOP** キーを押すことを示します。

3. BIOS エントリ、内部ルーチン

MSX2、MSX2+では BIOS が MAIN ROM と SUB ROM に分割されています。これを区別するため、以下のように表記します。

CHGET(009FH/MAIN)	MAIN ROM の 009FH 番地
REDCLK(01F5H/SUB)	SUB ROM の 01F5H 番地

4. ワークエリア、フック

【VALTYP(F663H, 1)】	F663H 番地の 1 バイトを使用
【BUF(F55EH, 258)】	F55EH 番地以降の 258 バイトを使用

5. その他、必要に応じて解説します。

目次

このマニュアルの読み方 iii

第5部 スロットとカートリッジ

1章	スロット	3
1.1	基本スロットと拡張スロット	3
1.2	スロットの選択	5
2章	インタースロットコール	7
2.1	インタースロットコールの動作	7
2.2	インタースロットコールの使用法	8
2.3	スロットの状態を知るためのワークエリア	17
3章	カートリッジソフトの作成法	19
3.1	カートリッジヘッダ	19
3.2	カートリッジ用ソフトの作成に関する諸注意	25

第6部 標準的な周辺装置のアクセス

1章	PSGと音声出力	31
1.1	PSGの機能	31
1.2	PSGのアクセス	37
1.3	1ビットサウンドポートによる音声発生機能	40
1.4	1ビットサウンドポートのアクセス	41
2章	カセットインターフェイス	42
2.1	ボーレート	42
2.2	1ビットの構成	43

2.3	1 バイトの構成	43
2.4	ヘッダの構成	44
2.5	ファイルのフォーマット	44
2.6	カセットファイルのアクセス	47
3 章	キーボードインターフェイス	52
3.1	キー配列	52
3.2	キースキャン	52
3.3	文字の入力	54
3.4	ファンクションキー	60
3.5	割り込み中の STOP キー	61
4 章	プリンターフェイス	63
4.1	プリンターインターフェイスの概要	63
4.2	MSX 仕様のプリンタへの出力	64
4.3	MSX 仕様以外のプリンタへの出力	65
4.4	プリンタのアクセス	65
5 章	汎用入出力インターフェイス	68
5.1	ポートの機能	69
5.2	ジョイスティックの使用法	70
5.3	パドルの使用法	72
5.4	タッチパネル、ライトペン、マウス、トラックボールの使用法	73
6 章	CLOCK と バッテリバックアップメモリ	75
6.1	CLOCK-IC の機能	75
6.2	CLOCK-IC の構造	76
6.3	MODE レジスタの機能	77
6.4	TEST レジスタの機能	78
6.5	RESET レジスタの機能	78
6.6	クロックおよびアラームの設定	79
6.7	バッテリバックアップメモリの内容	81
6.8	CLOCK-IC のアクセス	83

第7部 オプションの周辺機器

1章 MSX RS-232C	87
1.1 ハードウェア	87
1.2 拡張 BASIC	99
1.3 拡張 BIOS	130
1.4 サンプルプログラム	145
2章 MSX MODEM	146
2.1 ハードウェア	146
2.2 NCU の構成	147
2.3 メモリの構成	148
2.4 I/O の構成	148
2.5 拡張 BASIC	149
2.6 拡張 BIOS	199
3章 MSX-MUSIC	221
3.1 ハードウェア	221
3.2 拡張 BASIC	225
3.3 FM BIOS	244
3.4 YM2413 (OPLL)	255
4章 MSX-AUDIO	277
4.1 ハードウェア	277
4.2 拡張 BASIC	283
4.3 拡張 BIOS	324
4.4 MBIOS	348
4.5 Y8950 (MSX-AUDIO)	430
5章 MSX-JE	473
5.1 概要	473
5.2 MSX-JE の使用規定	475

5.3	ファンクションコールの方法	476
5.4	仮想端末入力インターフェイス	481
5.5	辞書インターフェイス	498
5.6	VJE-80 および VJE-80A 使用上の注意	519
6 章	24ドット漢字プリンタ	523
6.1	基本仕様	523
6.2	コントロールコード	524
6.3	グラフィック印字	558
6.4	漢字コード	565
6.5	縦拡大文字の網掛け処理	565
6.6	サンプルプログラム	565
7 章	MSX 拡張 BIOS 仕様	566
7.1	概要	566
7.2	拡張 BIOS におけるデバイス	567
7.3	拡張 BIOS の呼び出し	567
7.4	拡張 BIOS の実現方法	568
7.5	拡張 BIOS の機能	570

APPENDIX

A.1	BIOS 一覧	582
A.2	ワークエリア	645
A.3	VRAM マップ	660
A.4	I/O マップ	669
A.5	拡張 I/O ポート	672
A.6	スーパーインポーズ	675
A.7	サンプルプログラム	681
A.8	エスケープシーケンス	682
A.9	コントロールコード	683
A.10	キャラクタコード表	684
A.11	グラフィックキャラクタコード表	686

A.12 トークン順の中間コード一覧 687

A.13 ファンクションコール一覧 688

索引	693
参考文献	702
お問い合わせについて	703

<MSX-Datapak Vol.1 目次>

第 1 部 ハードウェア

- 1 章 概略仕様
- 2 章 主要ユニット
- 3 章 インターフェイス
- 4 章 カートリッジ
- 5 章 システムを拡張する際の注意
- 6 章 アドレスマップ
- 7 章 MSX2+回路例

第 2 部 システムソフトウェア

- 1 章 ブートシーケンス
- 2 章 割り込み
- 3 章 BASIC
- 4 章 BASIC の内部構造
- 5 章 マシン語とのリンク
- 6 章 内部ルーチン
- 7 章 プログラム開発の諸注意

第 3 部 MSX-DOS

- 1 章 概要
- 2 章 MSX-DOS の操作
- 3 章 MSX-DOS の構造
- 4 章 ファンクションコール

第 4 部 VDP

- 1 章 V9938 の構成
- 2 章 基本入出力
- 3 章 レジスタの機能
- 4 章 V9938 の画面モード
- 5 章 V9938 のコマンド
- 6 章 スプライト
- 7 章 その他の機能
- 8 章 V9958 の構成
- 9 章 YJK 方式
- 10 章 V9958 の画面モード

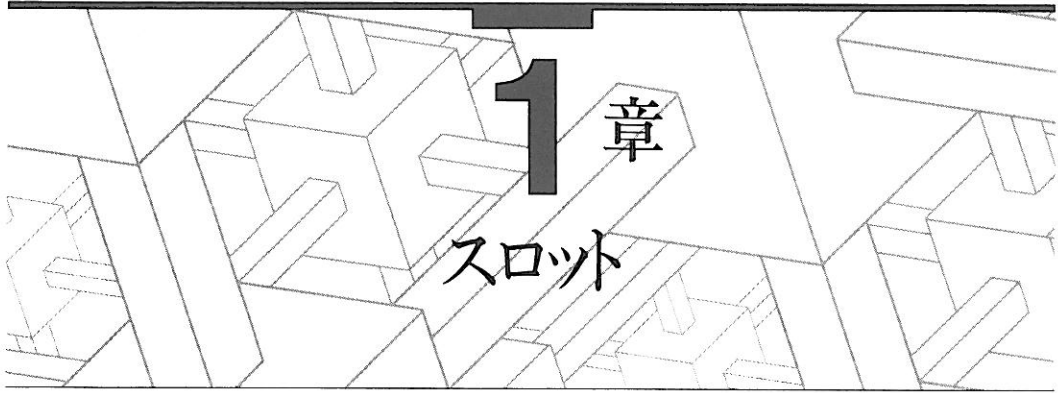


第5部

スロットとカートリッジ

MSX で使用されている CPU (Z80) は、通常 64K バイト (0000H~FFFFH) のアドレス空間しかアクセスできません。しかし、MSX は 1M (メガ) バイトに相当する空間を自由にアクセスすることができます。これは MSX が「スロット」の概念を採用し、同一のアドレスに複数のメモリあるいはデバイスを割り当てる機能を備えているからです。

ここでは、このスロットの使用方法、およびスロットを介して MSX にカートリッジソフトや新しいデバイスを接続するために必要となる情報を紹介します。



スロットは大量のアドレス空間を確保するためのアドレス空間の切り換えをするもので、MSXのアドレスバスに接続されるメモリやデバイスは、すべてスロットを介して実装されています。それは、本体内にある BASIC の ROM であろうと MSX-DOS モード時の RAM であろうと例外ではありません。カートリッジソフトを接続する場所も、1つのスロットです。ここでは、スロットに接続されたソフトウェアやデバイスの取り扱い方を説明します。

1.1 基本スロットと拡張スロット

スロットには基本スロットと拡張スロットの2種類があります。「基本スロット」とは、図 5.1 に示すように CPU のアドレスバスに直結するスロットを指し、MSX の仕様では最大 4 個持つことができます。基本スロットは、スロット拡張ボックスを接続することにより（本体内で拡張されていることもある）、最大 4 個のスロットに拡張でき、このときのスロットを「拡張スロット」と呼びます。4 個の基本スロットをそれぞれ 4 つの拡張スロットに拡張した場合、スロットの数は最大の 16 個となり、 $16 \text{ スロット} \times 64\text{K バイト} = 1\text{M バイト}$ のアドレス空間をアクセスすることが可能です。

なお、拡張スロットにさらに拡張ボックスを差し込んだ場合はシステム自体が起動できなくなりますので、このようなことは行わないで下さい。MSX 標準のカートリッジ用スロットは必ず基本スロットですが、各機種種のオプションハードウェア専用コネクタは、拡張スロットに接続されていることがあります。

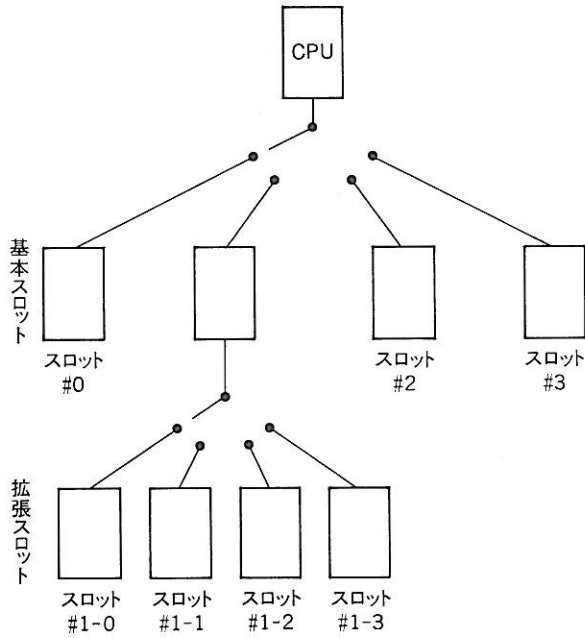


図 5.1 基本スロットと拡張スロット

各スロットは 0000H～FFFFH までの 64K バイトのアドレス空間を持ちますが、MSX ではそれを 16K バイトずつ 4 つに分け「ページ」として管理しています。CPU はページごとに任意のスロットを選択してアクセスでき、図 5.2 のように、いくつかのスロットから必要な部分だけを選んで組み合わせることが可能です。ただし、ある番号のページを異なる番号のページに割り当てることはできません。

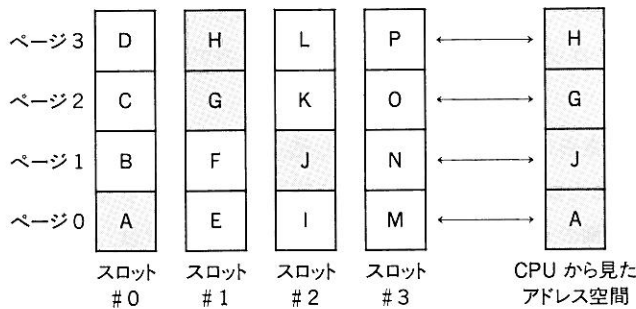


図 5.2 ページ選択の例

1.2 スロットの選択

スロットの選択方法は、基本スロットと拡張スロットでは異なります。基本スロットの場合は A8H 番地の I/O ポートによって行い(図 5.3)、拡張スロットの場合は実装された拡張カートリッジの「拡張スロット選択レジスタ (FFFFH)」によって行います(図 5.4)。しかし、それらを直接変更することはたいへん危険ですから、スロットの切り換えはスロットの構造を理解した上で、BIOS の ENASLT (0024H / MAIN) を呼び出して切り換えて下さい。他のスロットにあるプログラムを呼び出したい場合は、2章で説明するインタースロットコールを使用して下さい。

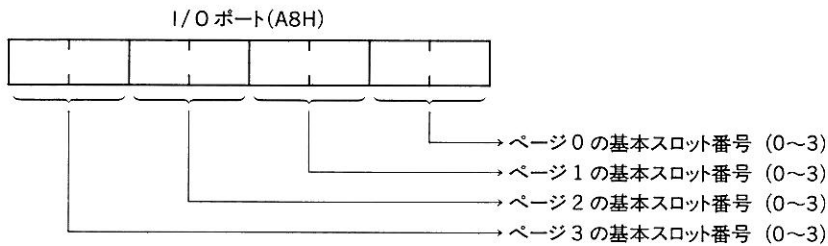
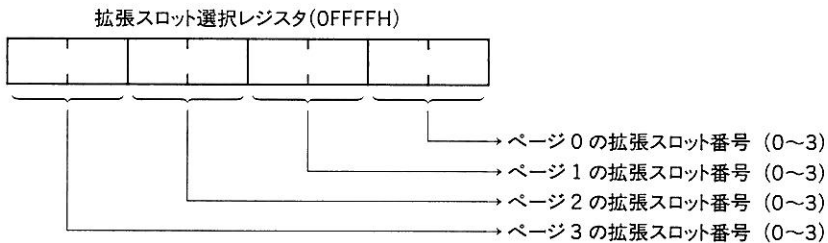


図 5.3 基本スロットの選択



注) このスロットが基本スロットか、拡張スロットかが判るように、拡張スロットの場合は値を書き込んだ後に読み出すと、書き込んだ値が反転したものが値として読み出されます。

このレジスタは同一基本スロット内であれば、どの拡張スロットでも同じ値です。

図 5.4 拡張スロットの選択

どこのスロットに MAIN ROM や RAM が実装されているか、またカートリッジ用のスロットが何番のスロットであるかは機種によって異なります。もし手持ちの MSX がどのようにスロットを使用しているかを知りたい人はマニュアルなどで調べて下さい。しかし、MSX はどのスロットに何があろうと正常な動作ができるように仕様が決められていますから、その仕様準拠に限り、通常はスロットの使用状況を気にする必要はありません。

しかし、場合によっては特定のソフトウェアがどこのスロット上に置かれているのかを知る必要が生じることもあります。例えば、BASICのMAIN ROMは基本スロット#0、または基本スロット#0を拡張した拡張スロット#0-0に置くという仕様ですが、MSX1にMSX2の機能を追加するMSX2バージョンアップアダプタでは、MAIN ROMがスロット#0やスロット#0-0以外の上に置かれることとなります。また、MSX2のSUB ROMが入っているスロットは機種によってまちまちで、このような場合、下記のワークエリアを参照することにより、BASICインタプリタのROMが置かれているスロットを知ることが可能です。スロット情報は図5.5のフォーマットで得られます。DOSからBIOSを呼び出すときも、この方法を使用してMAIN ROMのスロットを調べて下さい。

【EXPTBL(0FCC1H,1)】 MAIN ROMのスロット

【EXBRSA(FAF8H,1)】 SUB ROMのスロット (MSX1では0)

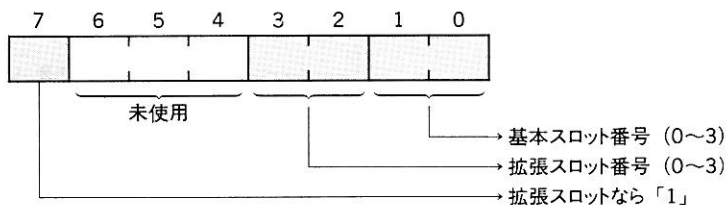


図5.5 スロットのフォーマット

また、あるルーチンがページ1とページ2(4000H~BFFFH)にまたがっている場合、このルーチンの中で1ページから2ページへ分岐が起こるときには、ページ2にページ1と同じスロットが選択されていなければなりません。そのためには、ページ1でまず自分のいるスロットを調べ、ページ2をそのスロットに切り換えるという作業が必要です。現在自分がどのスロットにいるのか、という情報を知るためには、添付のフロッピーディスクに入っている「WHEREAMI.MAC」を使用します。

2章

インタースロットコール

前述のように、MSX ではプログラムが異なったスロットに分かれているため、現在選択されているスロット上にはないプログラムが必要になることもあります。これは主に以下のような場合が考えられます。

1. MSX-DOS の環境から、MAIN ROM にある BIOS を呼び出す。
2. BASIC の環境から、SUB ROM にある BIOS を呼び出す (MSX2 以降)。
3. カートリッジソフトから、MAIN ROM あるいは SUB ROM の BIOS を呼び出す。

これらの作業を行う際、スロット切り換えが簡単かつ安全に行えるように、インタースロットコールという一群の BIOS ルーチンが用意されていて、どのスロットに存在するルーチンでも呼び出せるようになっています。この章では、このインタースロットコールの使用法を説明します。

2.1 インタースロットコールの動作

例えば、MSX-DOS から MAIN ROM 上の BIOS を呼び出す場合、スロットの状態の遷移は以下に示すとおりです。

1. 初めは MSX-DOS 環境なので、64K のアドレス空間すべてに RAM が選択されている。このままでは BASIC-ROM をアクセスできない (図 5.6-1)。
2. ROM の BIOS を呼び出すために、ページ 0 を BASIC の MAIN ROM に切り換え、アクセス可能な状態にする。そして BIOS を呼び出す (図 5.6-2)。
3. BIOS の処理が終わったら再び元の状態に戻し、初めに呼ばれたアドレスにリターンする。

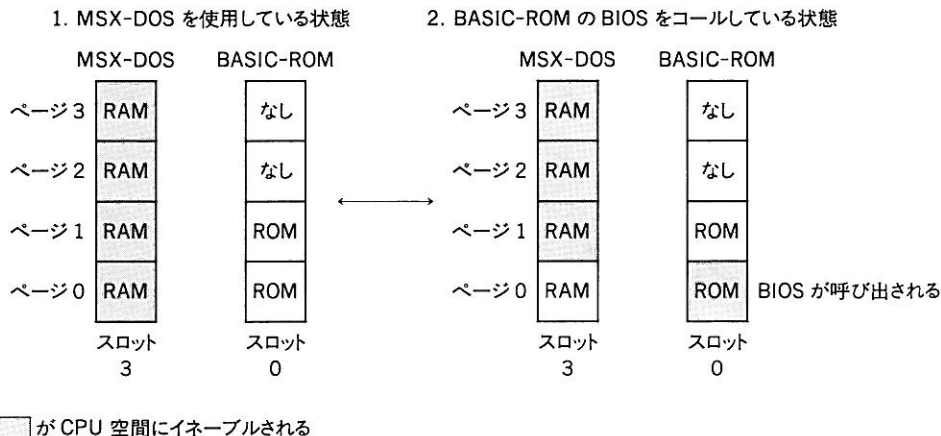


図 5.6 インタースロットコールのされ方

このとき、MSX-DOS 側のプログラムが、ページ 0 以外に存在するならば簡単なのですが、呼び出す側のプログラムが呼び出される側の BIOS と同じページ 0 に存在しているときは、非常に複雑になります。呼び出す方のプログラムがページ 0 を切り換えたときに自分自身がなくなって暴走してしまう、ということのないように配慮しなければいけません。

インタースロットコールでは、いったんページ 3 へ分岐してから実際のスロット切り換えを行うという方法で、この問題を解決しています。

このような複雑な状況をインタースロットコールはサポートするので、ユーザーは簡単に他のスロットのプログラムを呼び出すことができます。

2.2 インタースロットコールの使用法

インタースロットコールは、以下に述べるいくつかの方法で行うことができます。これらは BIOS として MAIN ROM 内に含まれているものですが、その中のいくつかは MSX-DOS の環境下においてもまったく同一のものが用意され、MSX-DOS 使用時のインタースロットコールを可能としています（「2.2.2 MSX-DOS のインタースロットコール」参照）。

2.2.1 BIOS のインタースロットコールルーチン

RDSLT(000CH / MAIN)

機能

指定スロットの指定アドレスから値を読み出します。

コール手順

A スロット指定 (図 5.5 と同じフォーマット)
HL 読み出すアドレス

戻り値

A 読み出した値

変更レジスタ

AF、BC、DE

解説

指定したスロットの指定したアドレスの内容を読み出し、Aレジスタに格納します。スロットの指定はAレジスタによって図5.5の形式で行います。このとき、目的のスロットが基本スロットならば、上位6ビットはすべて「0」に設定し、下位2ビットでスロット#0~#3を決めます。もし拡張スロットを指定するならば、同様にビット0とビット1で基本スロットを指定し、その基本スロットに接続されるどの拡張スロットかということをもビット2とビット3で指定、さらにビット7を「1」にします。

WRSLT(0014H / MAIN)

機能

指定スロットの指定アドレスに値を書き込みます。

コール手順

A スロット指定 (図 5.5 と同じフォーマット)
HL 書き込むアドレス

E 書き込む値

戻り値

なし

変更レジスタ

AF、BC、D

解説

Aレジスタで指定したスロット(指定の形式は図5.5と同じ)のHLレジスタで指定したアドレスに、Eレジスタの値を書き込みます。

CALSLT(001CH / MAIN)

機能

指定したスロットの指定アドレスをコールします。

コール手順

IY	の上位8ビット	スロット (図5.5と同じフォーマット)
IX		コールするアドレス

戻り値

呼び出し先プログラムの実行結果により異なる

変更レジスタ

呼び出し先プログラムの実行結果により異なる

解説

IYレジスタの上位8ビットで指定したスロット(指定の形式は図5.5と同じ)の、IXレジスタで指定するアドレスに存在するルーチンをコールします。

ENASLT(0024H / MAIN)

機能

スロットを切り換えます。

コール手順

- A スロット (図 5.5 と同じフォーマット)
HL 上位2ビットでスロット切り換えを行うページを指定する

戻り値

なし

変更レジスタ

すべて

解説

HLレジスタの上位2ビットで指定したページを、Aレジスタで指定したスロットに切り換えます。

CALLF(0030H / MAIN)

機能

指定スロットの指定アドレスをコールします。

コール手順

インラインパラメータ形式でスロットとアドレスを指定する

戻り値

呼び出し先のプログラムの実行結果により異なる

変更レジスタ

呼び出し先のプログラムの実行結果により異なる

解説

指定したスロットの指定したアドレスをコールしますが、前述の CALSLT と異なり、ス

ロットおよびアドレスの指定は次に示すように、インラインパラメータ形式で行います。すなわち、この CALLF を呼び出す命令の直後にスロットを指定する値 1 バイト (RDSLTL と同じフォーマット) を置き、その次にアドレスを指定する数値 2 バイトを置くという形でパラメータを渡します。「CALL 0030H」の代わりに「RST 30H」という RST (リスタート) 命令を使ってもかまいません。その場合、4 バイトでインタースロットコールが実現可能になります。

リスト 5.1 インタースロットコールの実行例

```
rst      30h          ; interslot call
db       00000000b   ; select slot # 0
dw       006ch       ; call address = 006ch
```

RSLREG(0138H / MAIN)

機 能

基本スロット選択レジスタの内容を読み出します。

コール手順

なし

戻り値

A 読み込んだ値 (図 5.5 と同じフォーマット)

変更レジスタ

A

解 説

基本スロット選択レジスタの内容を読み出し、A レジスタに入れます。

WSLREG(013BH / MAIN)

機 能

基本スロット選択レジスタへ値を書き込みます。

コール手順

A 書き込む値 (図 5.5 と同じフォーマット)

戻り値

なし

変更レジスタ

なし

解説

基本ロット選択レジスタに A レジスタの値を書き込み、ロットを選択します。

SUBROM(015CH / MAIN)

機能

SUB ROM の指定したアドレスをコールします。

コール手順

IX コールするアドレス、PUSH IX (「A.1.2 SUB ROM」参照)

戻り値

呼び出し先のプログラムの実行結果により異なる

変更レジスタ

裏レジスタと IX、IY レジスタは保存される

解説

BASIC の SUB ROM を呼び出すためのルーチンです。SUB ROM の存在するロットは BIOS が自動的に調べます。普通は、次の EXTROM を使います。

EXTROM(015FH / MAIN)

機能

SUB ROM の指定したアドレスをコールします。

コール手順

IX コールするアドレス

戻り値

呼び出し先のプログラムの実行結果により異なる

変更レジスタ

裏レジスタと IY レジスタは保存される

解説

BASICのSUB ROMを呼び出すためのルーチンです。上記のSUB ROMとこのEXTROMとは、IXレジスタの値をpushするか否かという点だけが異なります。

2.2.2 MSX-DOS のインタースロットコール

MSX-DOSでは、以下に示す5種類のインタースロットコールが用意され、MSX-DOSジャンプベクタにそのエントリアドレスが定義してあります。これらはBIOS内の同名のルーチンとまったく同じものですから、機能や使用方法に関しては、前述のBIOSをご覧ください。

なお、MSX-DOSからSUB ROM内のルーチンを呼び出す場合は、これらのルーチンを使用せず、「2.2.3 MSX-DOSからのSUB ROMコール」の方法を使います。

RDSLTL(000CH)	指定スロットの指定アドレスから値を読み出す
WRSLTL(0014H)	指定スロットの指定アドレスに値を書き込む
CALSLTL(001CH)	指定スロットの指定アドレスを呼び出す
ENASLTL(0024H)	指定スロットを使用可能な状態にする
CALLFL(0030H)	指定スロットの指定アドレスを呼び出す

リスト 5.2 MSX-DOS から BIOS を呼び出す

```

calslt equ 001ch ; inter slot call
exptbl equ 0fcc1h ; slot address of MAIN ROM

ld iy,(exptbl-1) ; load slot address of MAIN ROM
; in high byte of IY
ld ix,address of the BIOS jump table
call calslt

```

2.2.3 MSX-DOS からの SUB ROM コール

MSX-DOS の環境から SUB ROM を呼ぶ場合は、以下のような特別な配慮が必要です

1. CALSUB のエントリ条件は、IX にアドレスを入れるだけです。
2. スタックはページ 0 以外でなければなりません。
3. 割り込み禁止は必要ありません。
4. 終了後、NMI フックの解除は必要ありません。

ここで問題となるのは、EXTROM に必要な IX レジスタの値が渡せないことです。したがって、BIOS ROM が呼び出された後に IX がセットされ、EXTROM にジャンプする必要があります。

具体的には、スタック上に次のようなルーチンを用意し、NMI のフックからジャンプさせるようにしておきます。次に、BIOS の NMI エントリ (フックは用意されているが使用されない) をインタースロットコールします。リスト 5.3 を参照して下さい。

リスト 5.3 DOS からの SUB ROM コール

```

;      +0      inc      sp
;      +1      inc      sp
;      +2      ld        ix,<sub-ROM entry>
;      +6      nop
;      +7      jp        extrom
calslt  equ      001ch
extrom  equ      015fh
nmi     equ      0066h      ; Non-maskable interrupt
h.nmi   equ      0fdd6h      ; Hook for NMI
exptbl  equ      0fcc1h
;
_calsub::
      exx                ; Save argument registers over setup
      ex      af,af'
      ld      hl,extrom
      push   hl
      ld      hl,0c300h    ; jp xxxx, nop
      push   hl
      push   ix            ; SUB ROM entry
      ld      hl,021ddh    ; ld ix,xxxx
      push   hl
      ld      hl,03333h    ; inc sp, inc sp
      push   hl
      ld      hl,0
      add    hl,sp
      ld      a,0c3h
      ld      (h.nmi),a
      ld      (h.nmi+1),hl
      ex     af,af'        ; Restore registers
      exx
;
      ld      ix,nmi
      ld      iy,(exptbl-1)
      call   calslt
      ei
;
      ld      hl,10        ; Throw away the interface routine
      add    hl,sp
      ld      sp,hl
      ret
end

```

2.3 スロットの状態を知るためのワークエリア

スロットに関するワークエリアには、以下のものがあります。

【EXBRSA(0FAF8H,1)】 SUB ROM のスロットアドレス

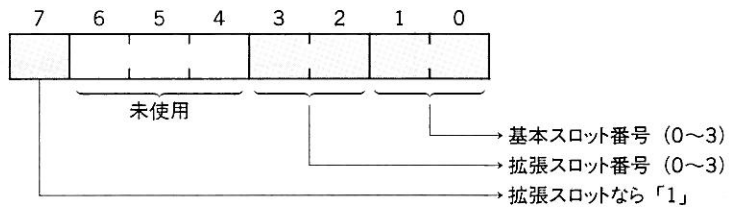


図 5.7 SUB ROM のスロットアドレス

【EXPTBL(0FCC1H,4)】 基本スロットの拡張の有無

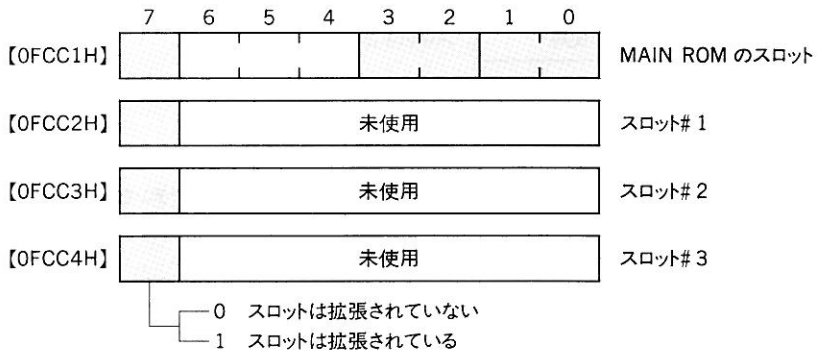


図 5.8 基本スロットの選択

【SLTTBL(0FCC5H,4)】 拡張スロット選択レジスタ値の保存エリア

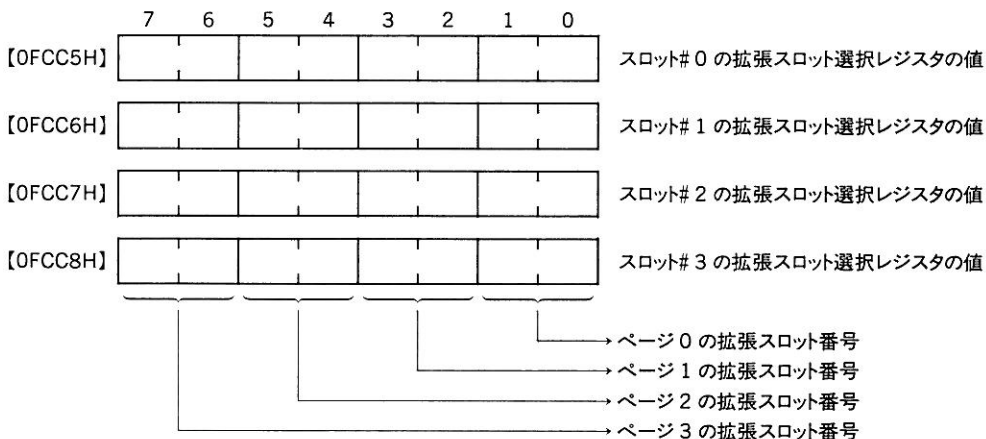


図 5.9 拡張スロットの選択

【SLTATR(0FCC9H,64)】 各スロット、ページにおけるアプリケーションの有無

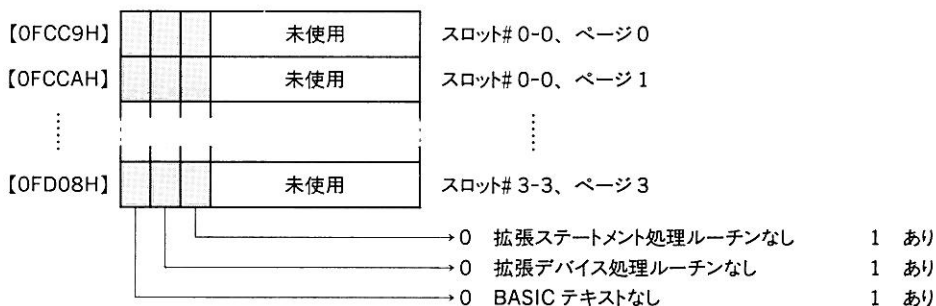


図 5.10 アプリケーションの有無

【SLTWRK(0FD09H,128)】 アプリケーション用ワークエリア

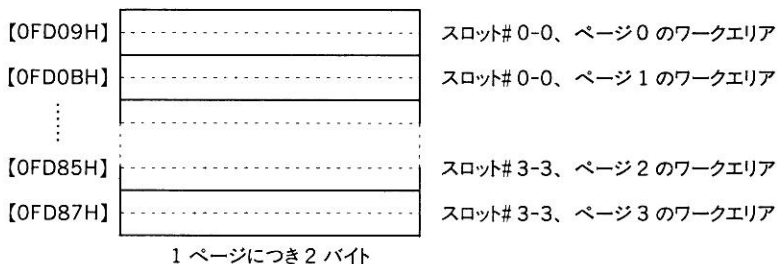


図 5.11 アプリケーション用ワークエリア

3章

カートリッジソフトの作成法

MSXには最低1つのスロットが表に出ており、ここに接続されるハードウェアを「カートリッジ」といいます。カートリッジには、

1. アプリケーションプログラムやゲームなどのROMカートリッジ
2. フロッピーディスクインターフェイスやRS-232Cインターフェイスなどの入出力装置カートリッジ
3. RAMを増やすための増設RAMカートリッジ
4. スロットを増やすためのスロット拡張器

などいろいろと接続でき、これによりMSXは機能の拡張を簡単に行うことができます。また、BASICやマシン語プログラムをROM化（カートリッジ化）することも簡単です。この章では、ソフトウェアをカートリッジ化する方法を説明します。

3.1 カートリッジヘッダ

MSXのカートリッジは、16バイトの共通ヘッダを持っており、システムがリセットされると、このヘッダに記述された情報によりカートリッジの初期設定が行われます。BASICやマシン語のプログラムをROM化したカートリッジの場合は、このヘッダに書き込む情報によってオートスタートさせることも可能です。カートリッジヘッダの構成を図5.12に示します。

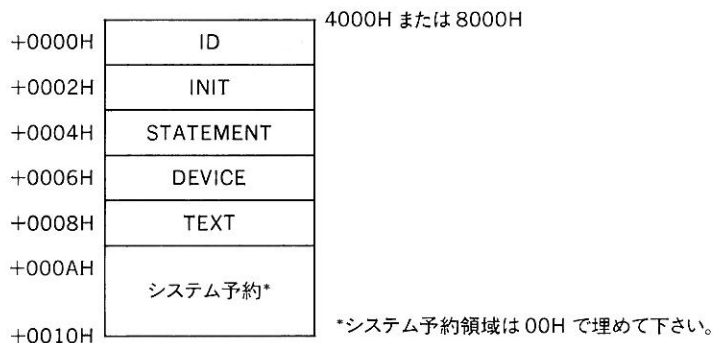


図 5.12 プログラムカートリッジのヘッダ

■ ID

ROM カートリッジであれば、この 2 バイトに「AB」(41H、42H) というコードを入れておきます。ちなみに SUB ROM の ID は「CD」となっています。

■ INIT

この 2 バイトには、カートリッジがワークエリアや I/O などの初期化を行うのであれば、その初期化ルーチンのアドレスを書き込み、行わなければ 0000H としておきます。初期化ルーチンの中でワークエリアの確保など必要な処理を行ったら、「RET」で終了させて下さい。このとき SP 以外のレジスタは内容を破壊してもかまいません。なお、ゲームのようにカートリッジ内でループしていればよいマシン語プログラムの場合は、ここからそのまま目的のプログラムを実行することが可能です。

■ STATEMENT

この 2 バイトには、カートリッジが CALL 文の拡張を行うのであれば、ステートメント拡張ルーチンのアドレスを書き込み、行わなければ 0000H としておきます。CALL 文の拡張を行う場合、ステートメント拡張ルーチンは 4000H～7FFFH になければいけません。

CALL 命令は以下の書式で記述されます。

CALL<拡張ステートメント名>[(<引数>[, <引数>…])]

拡張ステートメント名は 15 文字以下でなければいけません。CALL の省略として「_」(アンダースコア) も使用できます。

BASIC インタープリタは CALL 文を見つけると、ワークエリアの【PROCNM(0FD89H,16)】に拡張ステートメント名を入れ、ヘッダの STATEMENT の内容が 0 以外のカートリッジにス

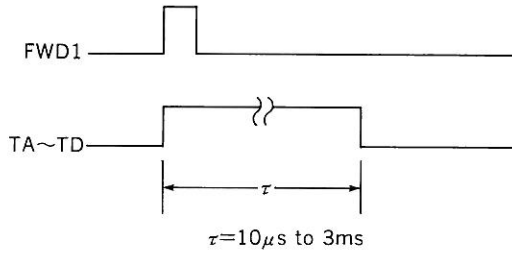


図 1.14 パドルのタイミング図

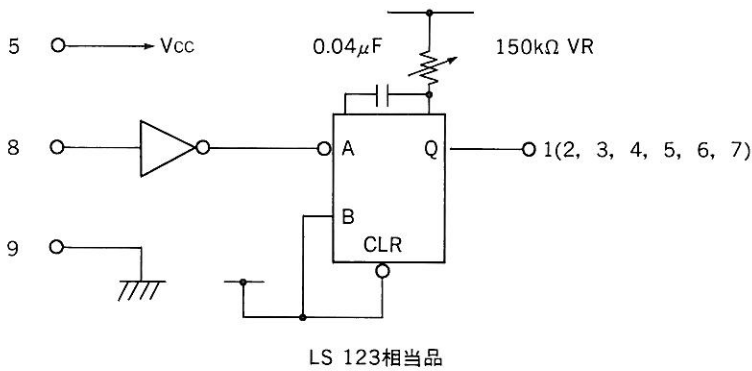


図 1.15 パドルの回路接続図

3.7 マウス

3.7.1 概要

MSX 用のマウスは本体側の負担が軽く、接続が簡単であるように汎用入出力インターフェイスに接続します。また、マウスをサポートしないソフトウェアでもマウスが使用できるように、カウンタモード（マウスモード）とジョイスティックモードの2つのモードを持ちます。

表 1.3 マウスの信号線一覧

ピン番号	信号名	ジョイスティック
1	データビット 1	前
2	データビット 2	後
3	データビット 3	左
4	データビット 4	右
5	+5V	
6	左ボタン	トリガ 1
7	右ボタン	トリガ 2
8	STROBE	GND
9	GND	

3.7.2 カウンタモード

カウンタモードはマウス本来のモードです。マウスは X・Y 方向の移動を検出し、それを 8 ビットの内部カウンタに保存します。そして、CPU からの要求があれば、MSX 本体にデータを転送します。データ転送シーケンスは以下のとおりです。

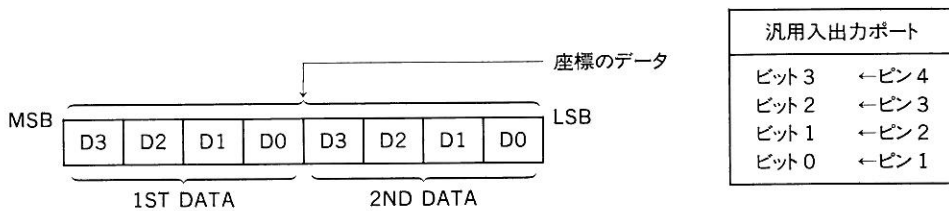


図 1.16 座標のデータ型式

マウスの座標の読み出しは、BIOS の GETPAD で行います。BIOS は以下のタイミングでマウスのデータを獲得しているので、ハードウェアはこのタイミングでデータを送ります。

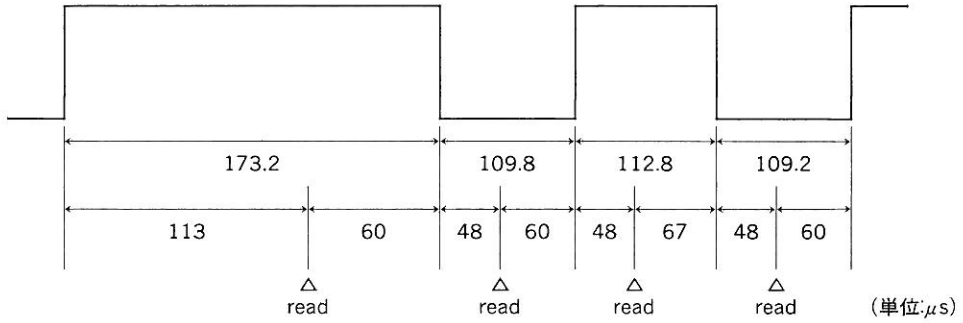


図 1.17 マウスのタイミング図

3.7.3 ジョイスティックモード

MSX のパワー-ON 時に、マウスの左ボタンを押しているとき、ジョイスティックモードになります。このモードではマウスをジョイスティックとして使用できます。

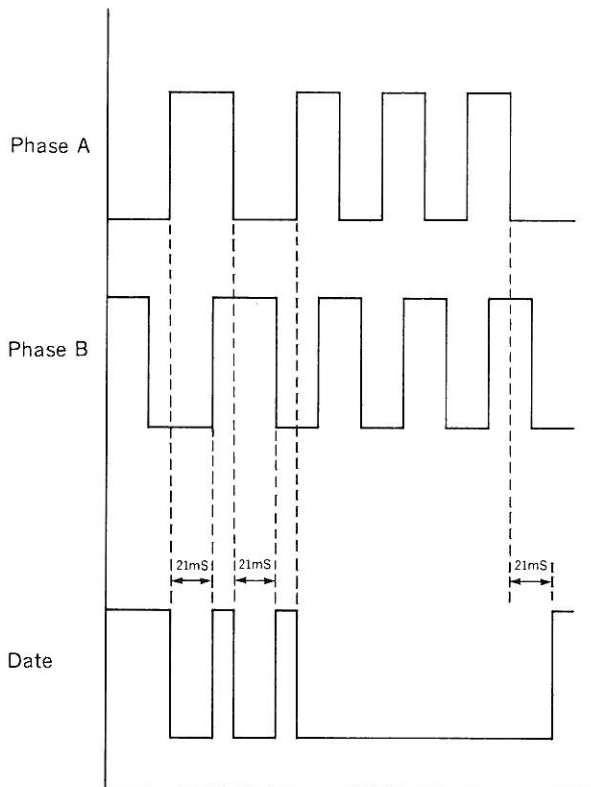
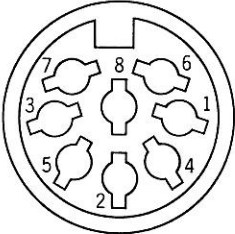


図 1.18 ジョイスティックモードのタイミング図

3.8 コネクタ

端子名	仕様
ビデオ出力	
● RGB 信号	決まっていないが、MSX2+では DIN 8 ピンコネクタ (DIN-45326) が一般的
● 複合映像信号 (ビデオ信号)	DIN 5 ピンコネクタ RCA 2 ピンコネクタ
● RF 変調信号	RCA 2 ピンコネクタ
カセット	DIN 8 ピンコネクタ DIN-45326
汎用入出力ポート	AMP 9 ピンコネクタ相当品
プリンタ	アンフェノール 14 ピンコネクタ相当品
カートリッジバス	2.54 ピッチ、50 ピンコネクタ
オーディオ出力	RCA 2 ピンコネクタ

接続端子	信号名	ピンコネクション
1	GND	
2	AUDIO	
3	AV	
4	SYNC	
5	YS	
6	R	
7	G	
8	B	

本体背面より見た図

図 1.19 RGB 信号 DIN 8 ピンコネクタの信号線

注意 これは、MSX2+、MSX2 のほとんどの機種で採用されている RGB 信号の信号線表であり、MSX の仕様として決められているわけではありません。

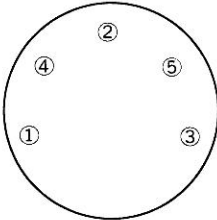
接続端子	信号名	ピンコネクション
1	+5V	
2	GND	
3	AUDIO	
4	MONITOR VIDEO	
5	RF VIDEO	

図 1.20 複合映像信号 DIN 5 ピンコネクタの信号線

注意

これは、MSX2の一部の機種で採用されている複合映像信号の信号線表であり、MSXの仕様として決められているわけではありません。

3.9 スロット

3.9.1 スロットの概念

この章で使われる「スロット」は、ひとつの64Kバイトのメモリ空間を構成するという点においては、「メモリバンク」という概念に近いものですが、CPUが番号を指定して選択することができるという点で、カートリッジを挿入するスロットに近いと考えられます。また、カートリッジバス上のこのスロットを選択する信号は、スロットセレクト信号と呼ぶのが最も自然であるので「スロット」と呼ぶことになりました。

この「スロット」の概念はソフトウェアの観点からとらえたものなので、実際のハードウェア(50ピンコネクタ)を示しているわけではありません(実際のハードウェアは、「カートリッジバス」もしくは「カートリッジスロット」と呼びます)。

3.9.2 「スロット」構造をとることの利点

従来の共通バス構造で複数のメモリバンクを持つ場合には、バスにつながる各デバイスに与えられる選択信号(スロットセレクト信号)に区別がないので、同じメモリ空間を占めるデバイスが複数個バスに接続された場合は、信号の衝突が起きてしまいシステムが正しく動作せず、場合によっては、ハードウェアの劣化をまねく恐れがあります。しかし、スロットセレクト信号によ

り、各デバイスが個別に選択される場合には、その心配はありません。さらに、同一アドレスを占めるプログラムが同時に存在できるので、システムの柔軟性・拡張性が高くなるという利点があります。

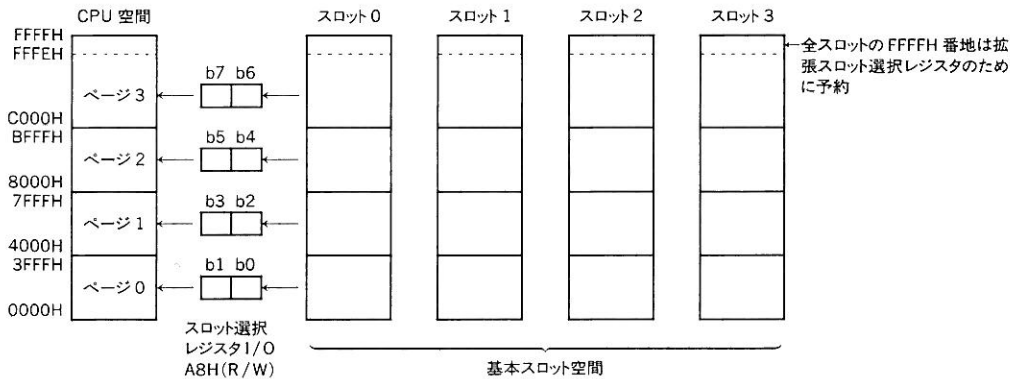


図 1.21 基本スロット

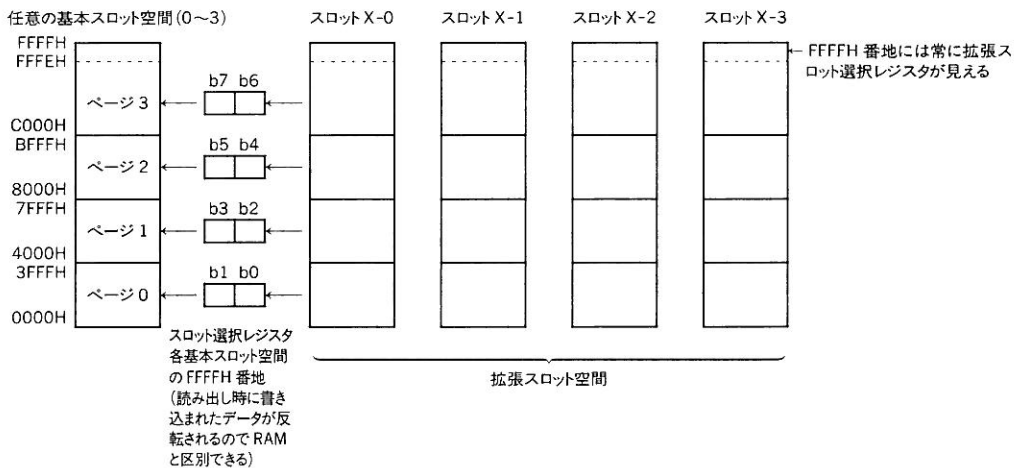


図 1.22 拡張スロット（オプション）

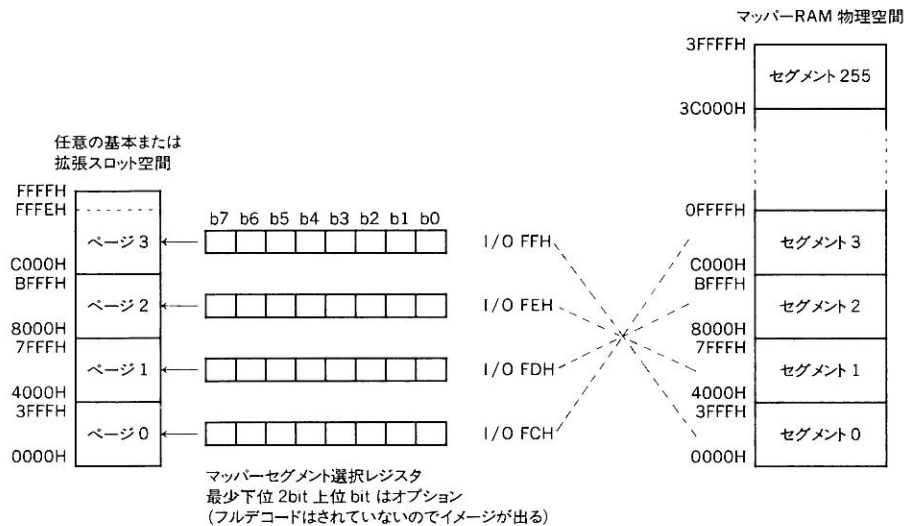


図 1.23 メモリマッパー (オプション)

3.9.3 スロットのタイミング

スロットは、以下のタイミングで動作します。

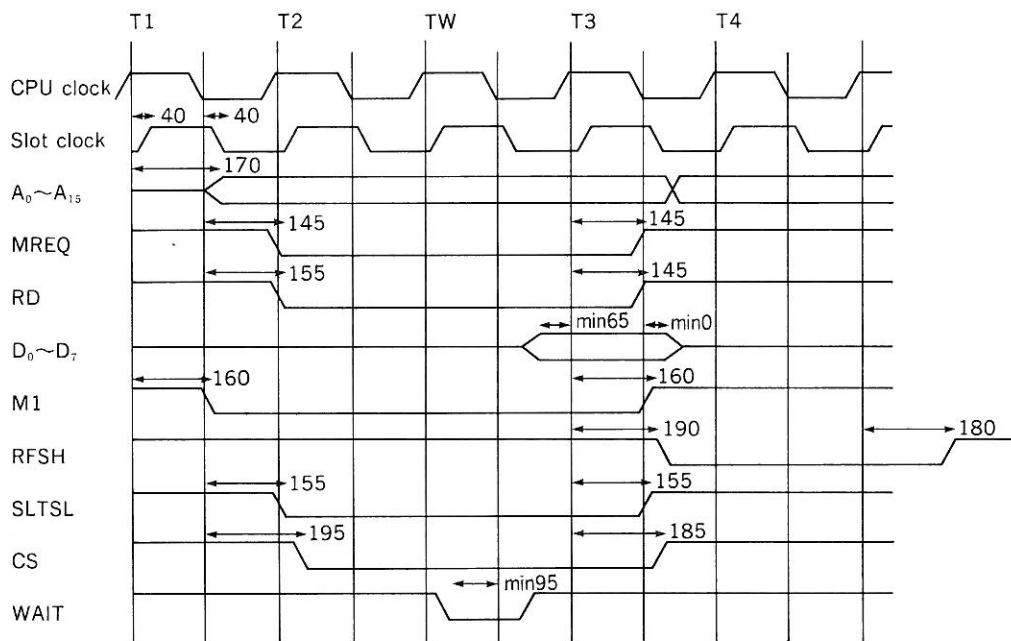


図 1.24 M1 サイクル (基本スロット)

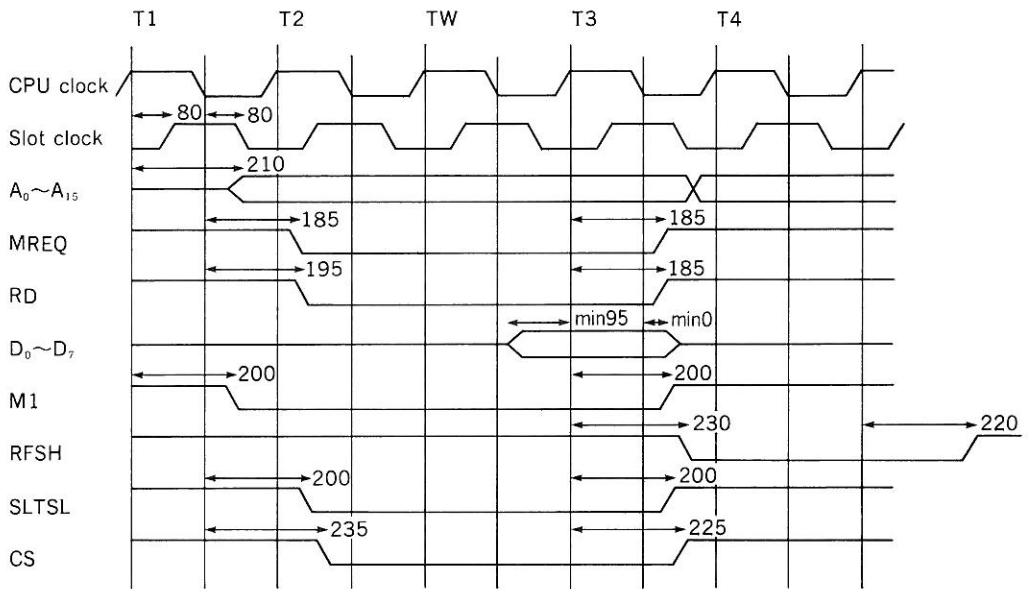


図 1.25 M1 サイクル (拡張スロット)

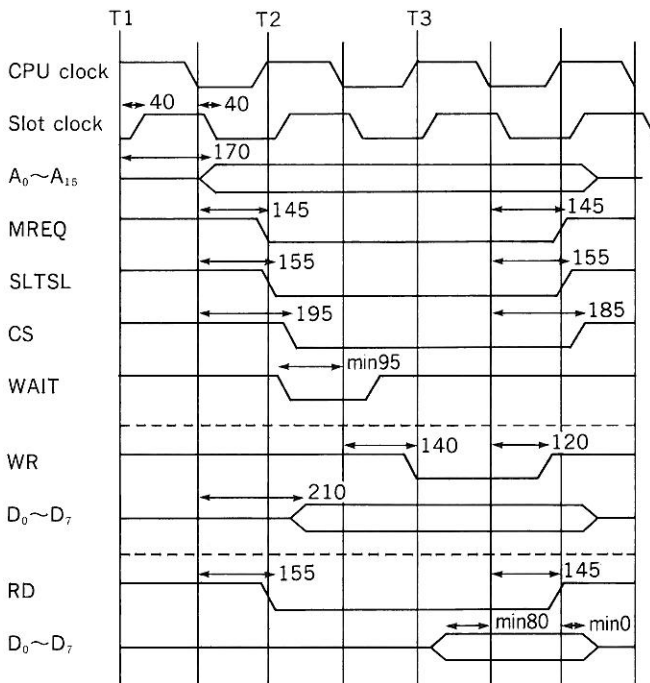


図 1.26 メモリサイクル (基本スロット)

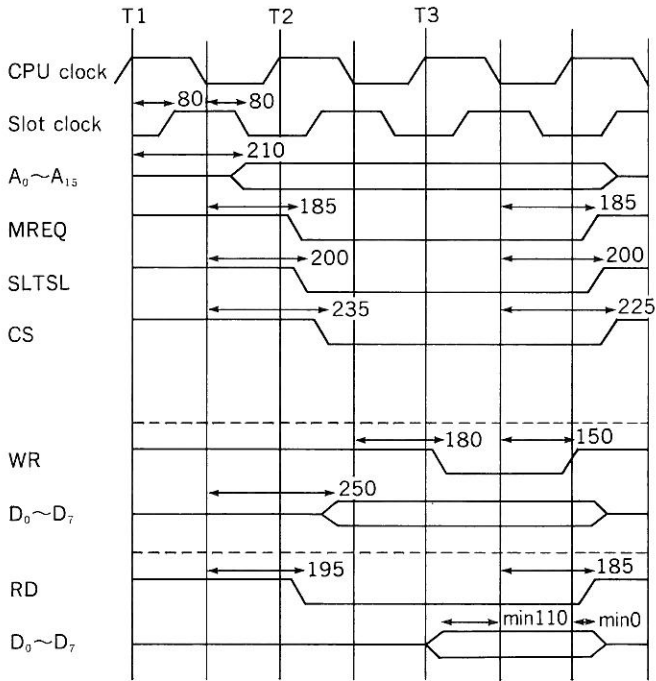


図 1.27 メモリサイクル (拡張スロット)

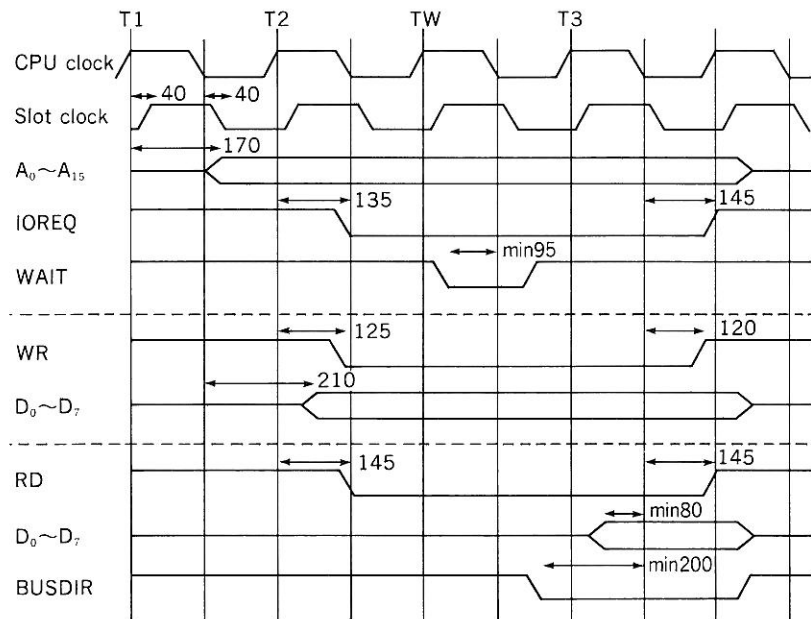


図 1.28 I/O サイクル (基本スロット)

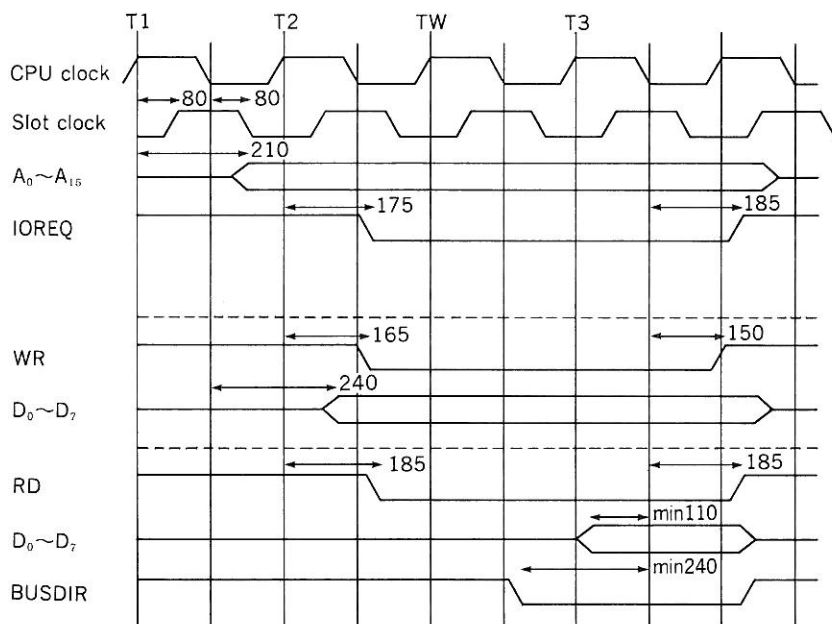
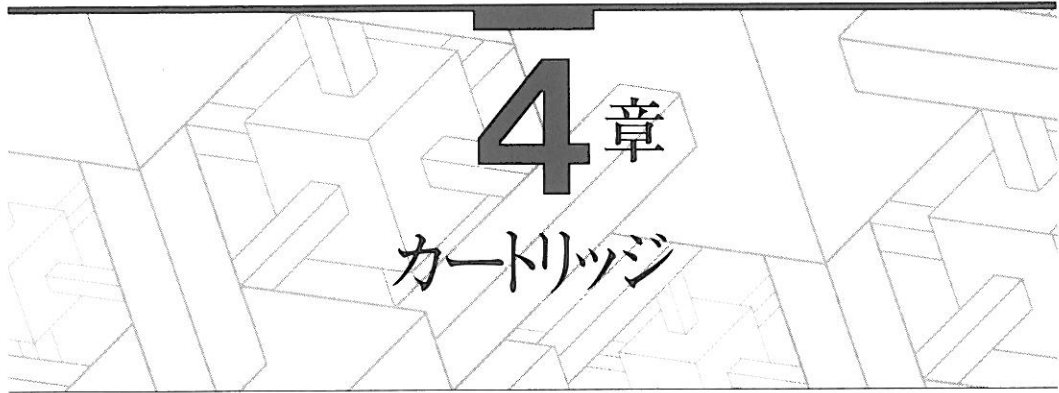


図 1.29 I/O サイクル (拡張スロット)



4.1 カートリッジ仕様

MSX のカートリッジは次のような仕様になっています。

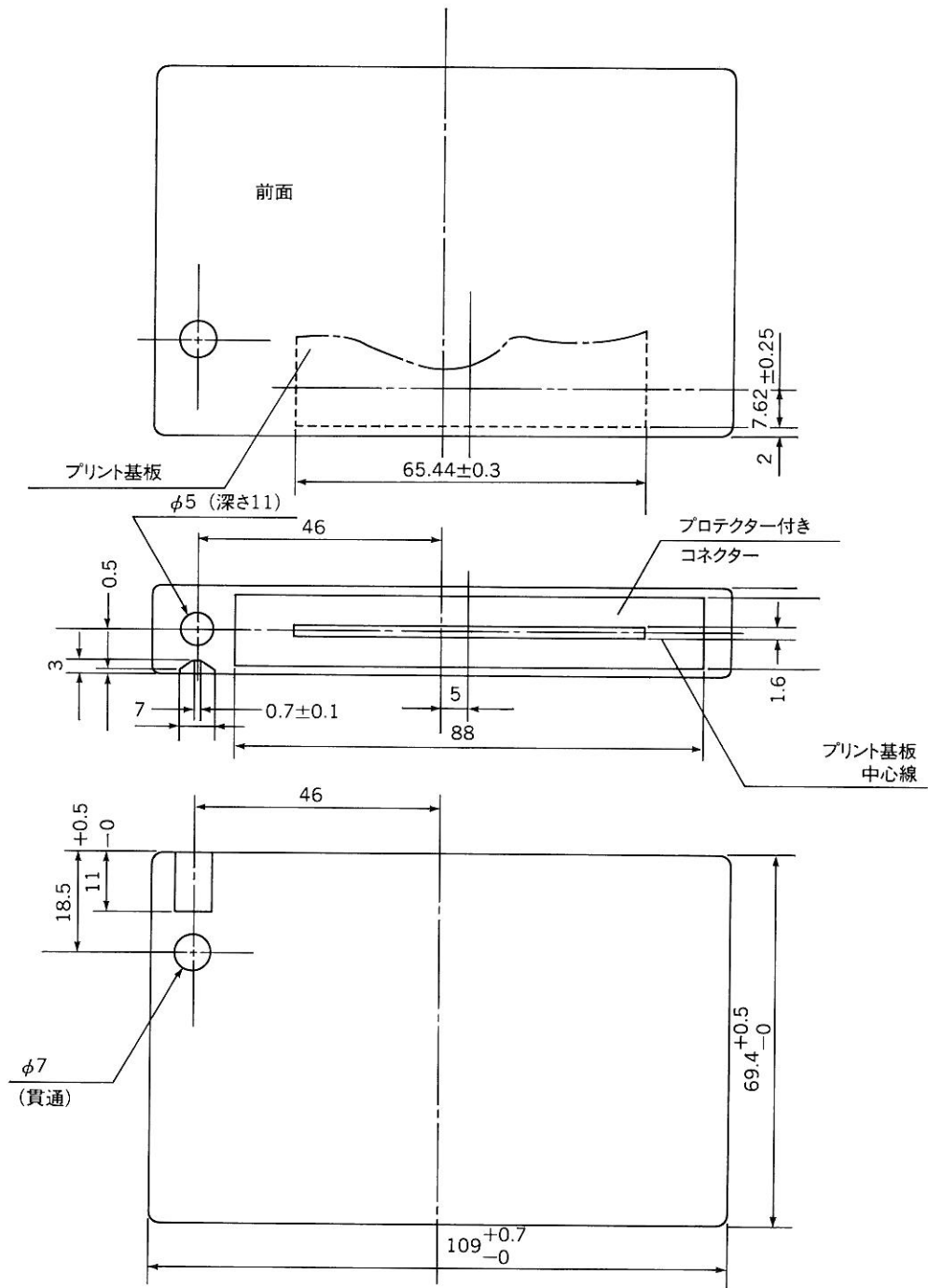


図 1.30 MSX カートリッジ標準仕様

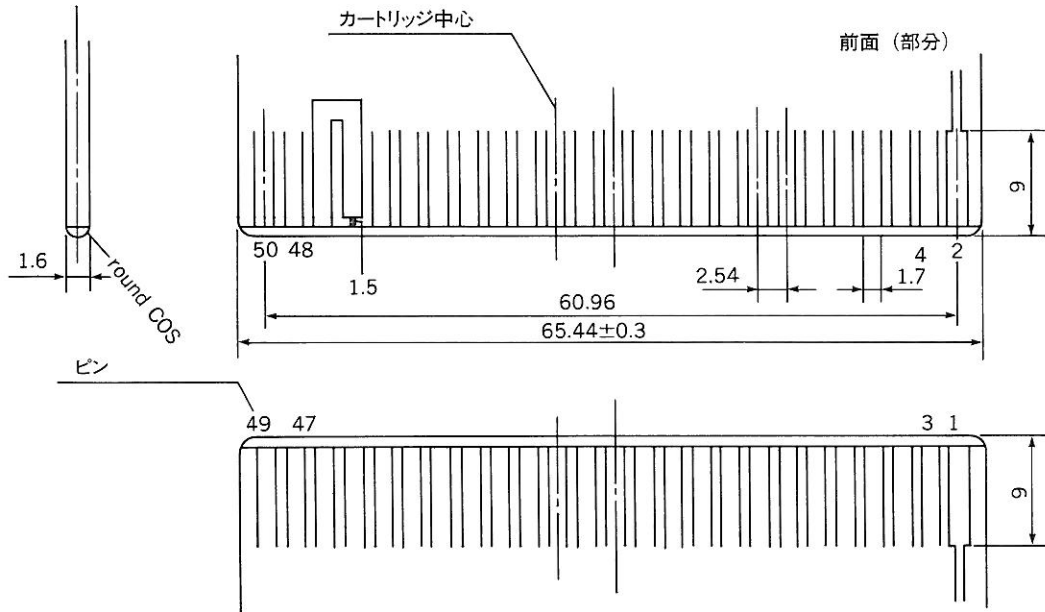
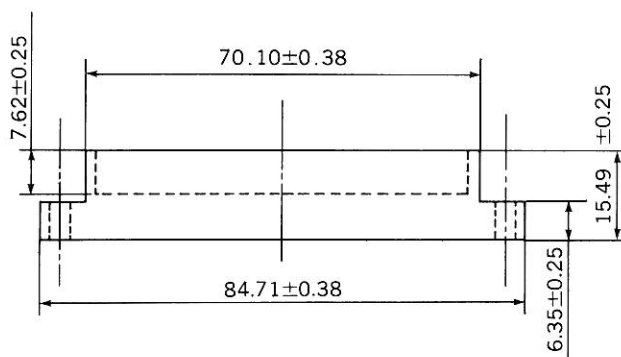
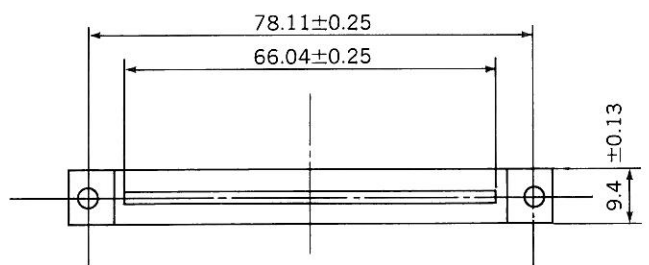


図 1.31 プリント基板のパターン



TI(corp)H4211***-25
selectable

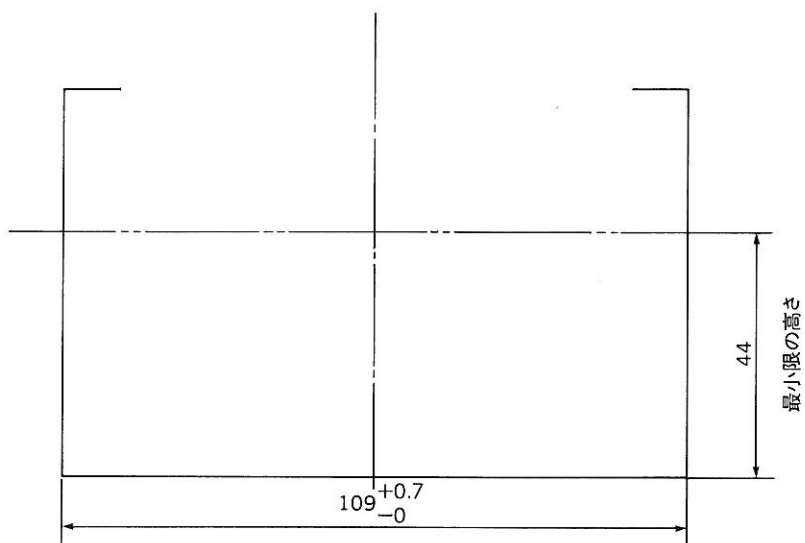


図 1.32 コネクタ例

注意

カートリッジのコネクタ側の端から、44mm 以内の部分に突起（スイッチ、コネクタなど）をつけてはいけません。

4.2 カートリッジバス

4.2.1 信号一覧

表 1.4 カートリッジバスの信号線一覧

ピン番号	名称	入出力*1	ピン番号	名称	入出力*1
1	$\overline{\text{CS1}}$	O	2	$\overline{\text{CS2}}$	O
3	$\overline{\text{CS12}}$	O	4	$\overline{\text{SLTSL}}$	O
5	予約*2	—	6	$\overline{\text{RFSH}}$	O
7	$\overline{\text{WAIT}}^{\ast 3}$	I	8	$\overline{\text{INT}}^{\ast 3}$	I
9	$\overline{\text{M1}}$	O	10	$\overline{\text{BUSDIR}}$	I
11	$\overline{\text{IORQ}}$	O	12	$\overline{\text{MERQ}}$	O
13	$\overline{\text{WR}}$	O	14	$\overline{\text{RD}}$	O
15	$\overline{\text{RESET}}$	O	16	予約*2	—
17	A9	O	18	A15	O
19	A11	O	20	A10	O
21	A7	O	22	A6	O
23	A12	O	24	A8	O
25	A14	O	26	A13	O
27	A1	O	28	A0	O
29	A3	O	30	A2	O
31	A5	O	32	A4	O
23	D1	I/O	34	D0	I/O
23	D3	I/O	34 ³⁶	D2	I/O
23	D5	I/O	34 ³⁸	D4	I/O
23	D7	I/O	34 ⁴⁰	D6	I/O
41	GND	—	42	CLOCK	O
43	GND	—	44	SW1	—
45	+5V	—	46	SW2	—
47	+5V	—	48	+12V	—
49	SUNDIN	I	50	-12V	—

注 意

*1 本体を基準にした入出力の区別

*2 予約は使用禁止端子

*3 オープンコレクタ出力

4.2.2 信号線の説明

表 1.5 カートリッジバス信号の説明一覧

ピン番号	名称	内容
1	$\overline{CS1}$	ROM 4000H~7FFFH 番地セレクト信号
2	$\overline{CS2}$	ROM 8000H~BFFFH 番地セレクト信号
3	$\overline{CS12}$	ROM 4000H~BFFFH 番地セレクト信号 (256K ROM 用)
4	\overline{SLTSL}	スロットセレクト信号 各スロット毎のそのスロット固有のセレクト信号
5	予約	将来のための予約 (使用禁止)
6	\overline{RFSH}	リフレッシュサイクル信号
7	\overline{WAIT}	CPU への WAIT 要求信号
8	\overline{INT}	CPU への割り込み要求信号
9	$\overline{M1}$	CPU のフェッチサイクルを表す信号
10	\overline{BUSDIR}	外部データバスバッファの方向を制御する信号 カートリッジがセレクトされ、データを送出するタイミングでメモリを除く各カートリッジより L レベルを出力する。 (I/O リードとアドレスをデコードした信号を OR した信号を入れる。詳細は7章を参照)
11	\overline{IORQ}	I/O リクエスト信号
12	\overline{MERQ}	メモリリクエスト信号
13	\overline{WR}	ライトタイミング信号
14	\overline{RD}	リードタイミング信号
15	\overline{RESET}	システムリセット信号
16	予約	将来のための予約 (使用禁止)
17~32	A0~A15	アドレスバス信号
33~40	D0~D7	データバス信号
41	GND	電源および信号グランド
42	CLOCK	CPU クロック 3.579545MHz
43	GND	信号グランド
44、46	SW1、SW2	抜き差しプロテクト用
45、47	+5V	+5V 電源
48	+12V	+12V 電源
49	SUNDIN	サウンド入力信号 (-5dBm)
50	-12V	-12V 電源

4.2.3 カートリッジバス接続条件

1. ファンイン、ファンアウト (LS TTL)

○ファンイン、ファンアウト(LS TTL)
データ・アドレスバス

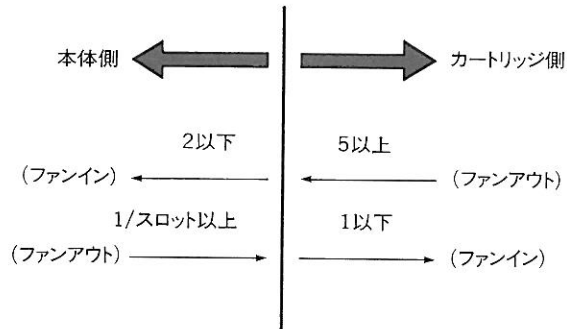


図 1.33 データ・アドレスバスのファンイン、ファンアウト

○コントロール信号

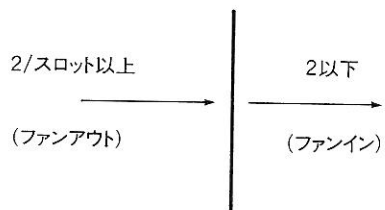


図 1.34 コントロール信号のファンイン、ファンアウト

2. 電圧レベル

TTL レベル

4.3 電源容量

+5V	300mA/各スロット
+12V	50mA/スロット合計
-12V	50mA/スロット合計

4.4 拡張スロットセレクト信号

拡張スロットセレクト信号の回路は、「7章 MSX2+回路例」を参照して下さい。

5章

システムを拡張する際の注意

5.1 スロットの拡張

基本構成の4スロットを基本スロットと呼びます。基本スロット以上のスロットをシステムに追加する場合には、基本スロットから拡張します。すなわち、拡張されたスロットを選択するには、まず拡張スロットが接続されている基本スロットを選択し、さらに、その中で追加されている拡張スロットを選択する構造を取るものとします。

この拡張されたカートリッジバスの双方向データバスバッファの向きを制御するのが BUSDIR 信号です。

メモリ (RAM や ROM など) だけのカートリッジ (ゲームカートリッジなど) は、この BUSDIR 信号を操作する必要がないので、カートリッジのコストを低くすることができます。しかし、CPU に対し信号を送る I/O デバイスを持ったカートリッジは、CPU にデータを送るタイミングに合わせて BUSDIR 信号を L レベルにして、バッファの向きを CPU 側に変えなくてはなりません。

拡張スロットのスロットセレクト信号を作成するためのスロットセレクトレジスタは、拡張スロットのメモリアドレス FFFFH 番地に置きます。そして、この拡張スロットセレクトレジスタを RAM と区別するため、このレジスタを読み出すとレジスタの内容のコンプリメントが読み出されるものとします。こうすることにより、基本スロットの1から3のどのスロットからでも、また、2つ以上のスロットからでも拡張スロットを設けることができます。

5.2 I/O の拡張

MSX システムに予め規定されていない I/O 装置を追加する場合は、その I/O 装置のアドレスをどこに割り当てるかが問題となります。I/O アドレスの 40H 番地から FFH 番地までは、

MSX システムの標準装置のための領域なので、その装置が標準装置として認められない限り使うことはできません。00H 番地から 3FH 番地は規定されていないので使用することはできませんが、複数の装置が競合を起こし、システムが動作しない事態を引き起こす可能性があります。

これを避けるために、原則として I/O 装置もメモリ空間に割り当てることにします。

6章 アドレスマップ

6.1 メモリマップ例

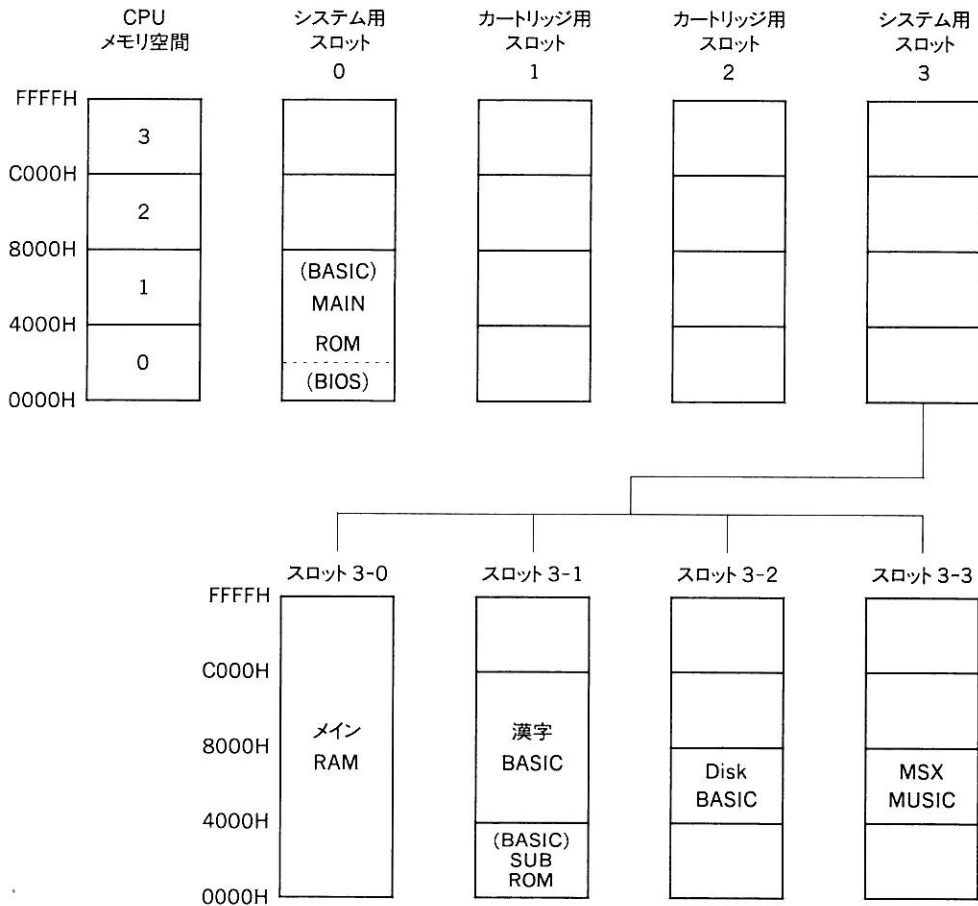
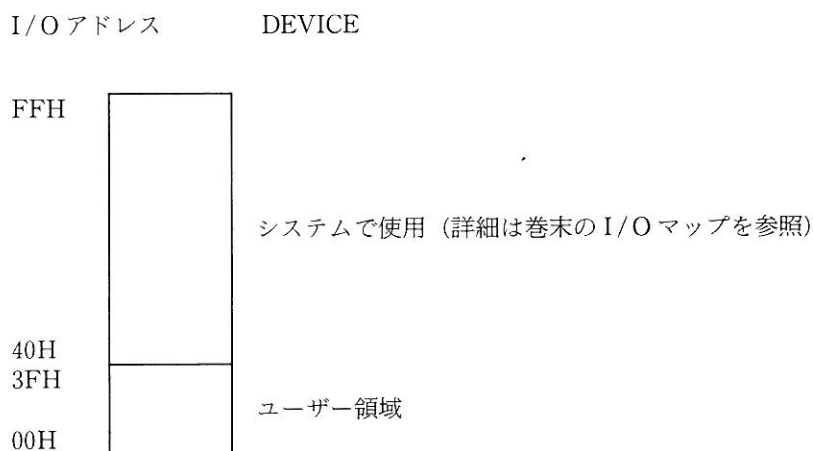


図 1.35 メモリマップ例

6.2 I/O マップ



6.3 I/O アドレスの割り当てについての注意

I/O アドレスの 40H～FFH はシステム用として使用します。この中で、現在使用していないアドレスは、将来のシステム拡張用として予約されています。詳細は巻末の「I/O マップ」を参照してください。

I/O ポートを直接操作するようなソフトウェアの作成は、ソフトウェアの互換性を損なう可能性が高くなります。そのため、ハードウェアに依存しないソフトウェアを作成するために、I/O デバイスは常に BIOS を通してアクセスするようにしてください。

ただし、VDP のアクセスに関しては、高速な画面操作を行うために I/O ポートを直接アクセスしてかまいません。しかし、その場合も、MAIN ROM の 6、7 番地を調べて、ポートアドレスを確認してください。詳しくは、「第4部 VDP」を参照して下さい。

I/O ポートの 00H～3FH はユーザーに解放されていますが、同一アドレスに複数の周辺装置が実装されることを防ぐために、できるだけメモリマップド I/O にしてください。

フロッピーディスクコントローラ (FDC) は I/O エリアに実装することもできますが、その場合には回路は外部からの信号によって遮断される機能を持たなければなりません。そして、FDC がアクセスされるときに限って I/O に接続されます。これによって、システムが複数の異なるフロッピーディスクインターフェイスを扱うことが可能になります。

市販の I/O 装置のために、どうしても I/O アドレスが必要になった場合は、株式会社アスキーシステム事業部で割り当てを行います。I/O 装置の製造前に必ず連絡して、I/O アドレスの割り当てを受けて下さい。

6.4 PPI ビット割り当て

表 1.6 PPI ビット割り当て一覧

ポート	ビット	I/O	信号名	内 容
A	0 1	↑ 出	CS0L CS0H	0000H~3FFFH 番地の スロット指定信号
	2 3		CS1L CS1H	4000H~7FFFH 番地の スロット指定信号
	4 5	↓ 力	CS2L CS2H	8000H~BFFFH 番地の スロット指定信号
	6 7		CS3L CS3H	C000H~FFFFH 番地の スロット指定番号
B	0 7	↑ 入力 ↓		キーボードリターン信号
C	0 1 2 3	↑ 出	KB0 KB1 KB2 KB3	キーボードスキャン信号
	4		CASON	カセットコントロール (L-ON) (カセットレコーダーの ON・OFF)
	5	↓ 力	CASW	カセット書き込み信号
	6		CAPS	CAPS ランプ信号 (L で点灯)
	7		SOUND	ソフトによるサウンド出力 (キーボードのクリック音)

6.5 PSGビット割り当て

表 1.7 PSGビット割り当て

ポート	ビット	I/O	接続コネクタ(ピン番号)	ジョイスティック使用時の信号
A	0	↑ 入 カ ↓	J3-1 ピン *1	FWD1
	1		J4-1 ピン *2	FWD2
			J3-2 ピン *1	BACK1
	2		J4-2 ピン *2	BACK2
			J3-3 ピン *1	LEFT1
	3		J4-3 ピン *2	LEFT2
			J3-4 ピン *1	RIGHT1
	4		J4-4 ピン *2	RIGHT2
J3-6 ピン *1		TRGA1		
5	J4-6 ピン *2	TRGA2		
	J3-7 ピン *1	TRGB1		
6	J4-7 ピン *2	TRGB2		
	キー配列指定入力 *4	H/L レベル(JIS・アイウエオ)		
7	CSAR (カセットテープリード)			
B	0	↑ 出 カ ↓	J3-6 ピン *3	H レベル
	1		J3-7 ピン *3	
	2		J4-6 ピン *3	
	3		J4-7 ピン *3	
	4		J3-8 ピン *3	
	5		J4-8 ピン *3	
	6		ポート A 入力セレクト (ジョイスティック 1、2 の選択)	
	7		KLAMP (カナランプ信号 「L」で点灯)	

注意

- *1 ポート B のビット 6 が「L」レベル時に有効。ジョイスティック 1 用。
- *2 ポート B のビット 6 が「H」レベル時に有効。ジョイスティック 2 用。
- *3 出力ポートとして使用しないときは「H」レベルにすること。
オープンコレクタバッファを介して出力する。
- *4 JIS 配列 —— 「H」レベル、アイウエオ配列 —— 「L」レベル

備考

J3、J4 の上記以外のピン端子信号

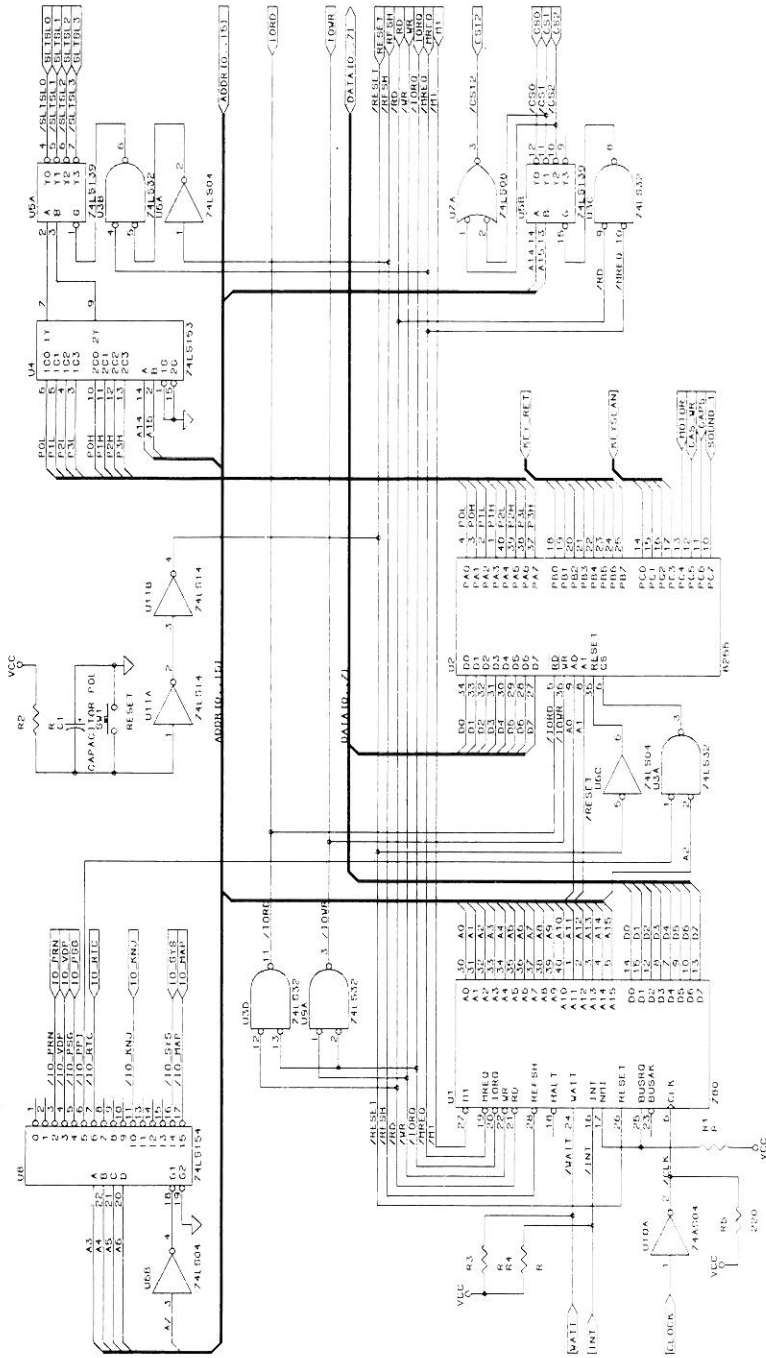
- 5 ピン +5V
- 9 ピン GND

7 章

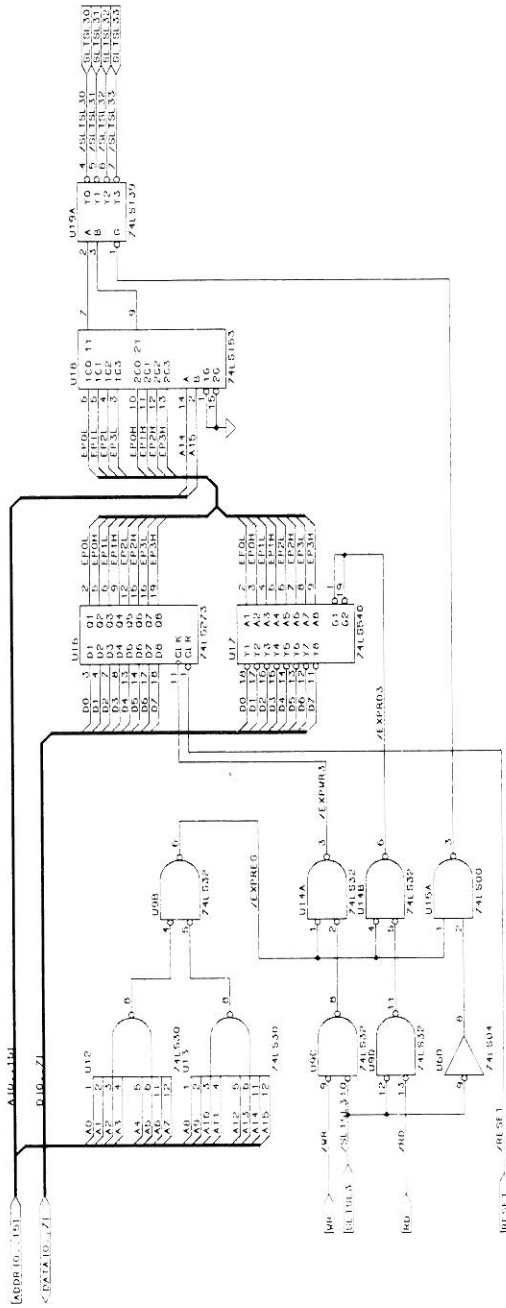
MSX2+回路例

この章では、MSX2+の回路例を示します。この回路例は、MSX2+の動作論理を理解するために参照して下さい。実際の MSX2+では、周辺回路をまとめた LSI を使用していますが、この回路例では動作論理の理解という目的のために、あえてシステムロジック LSI を使用しない構成になっています。

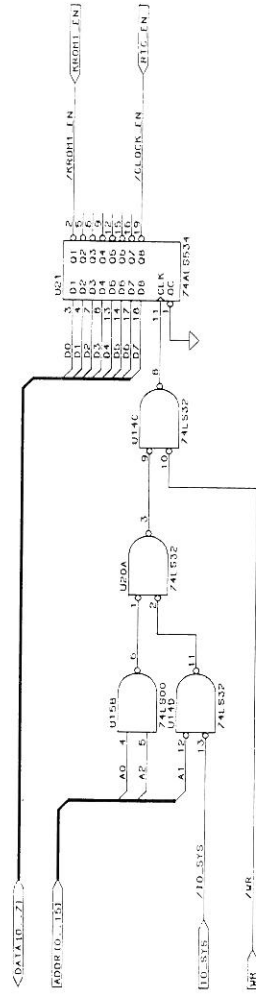
7.1 CPUと基本スロットセレクト信号



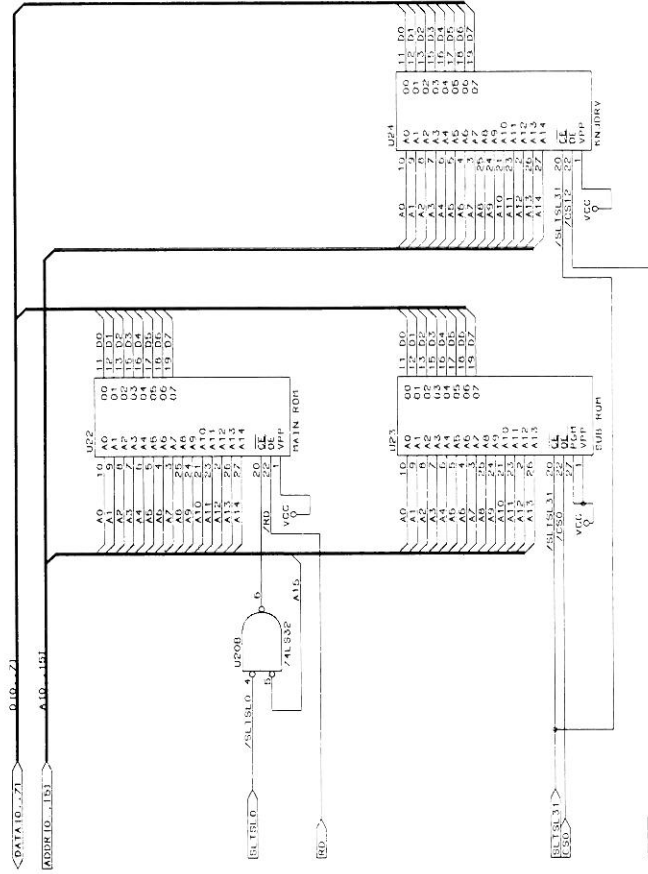
7.2 拡張スロットセレクト信号



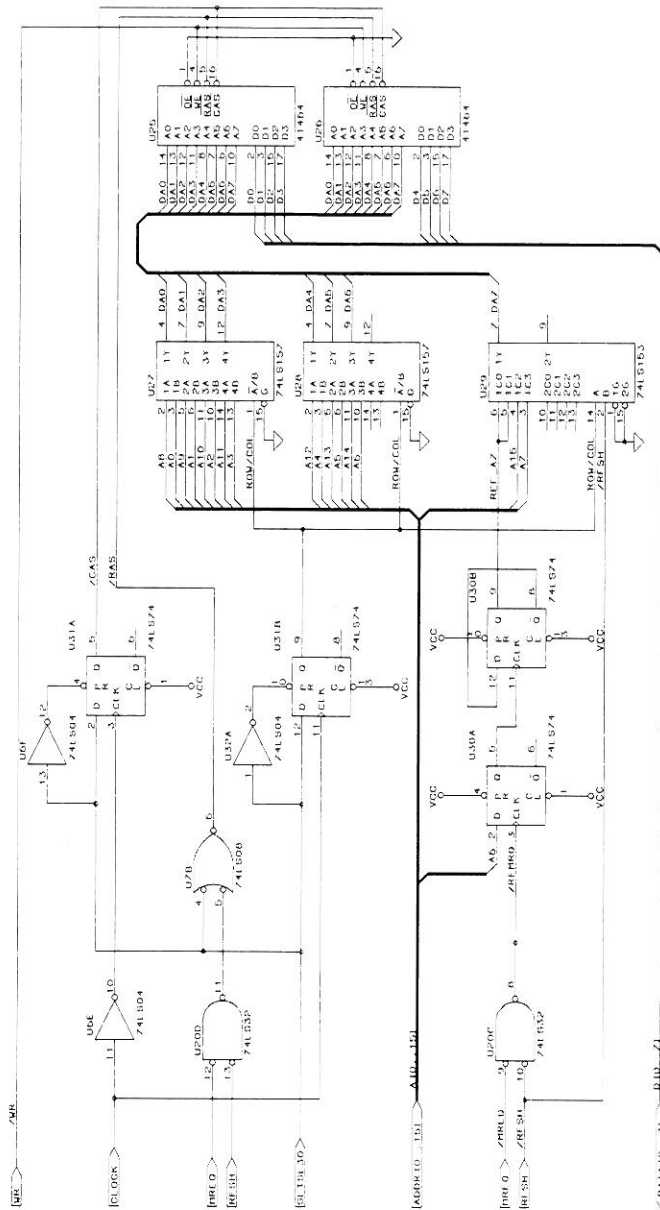
7.3 システムコントロールポート



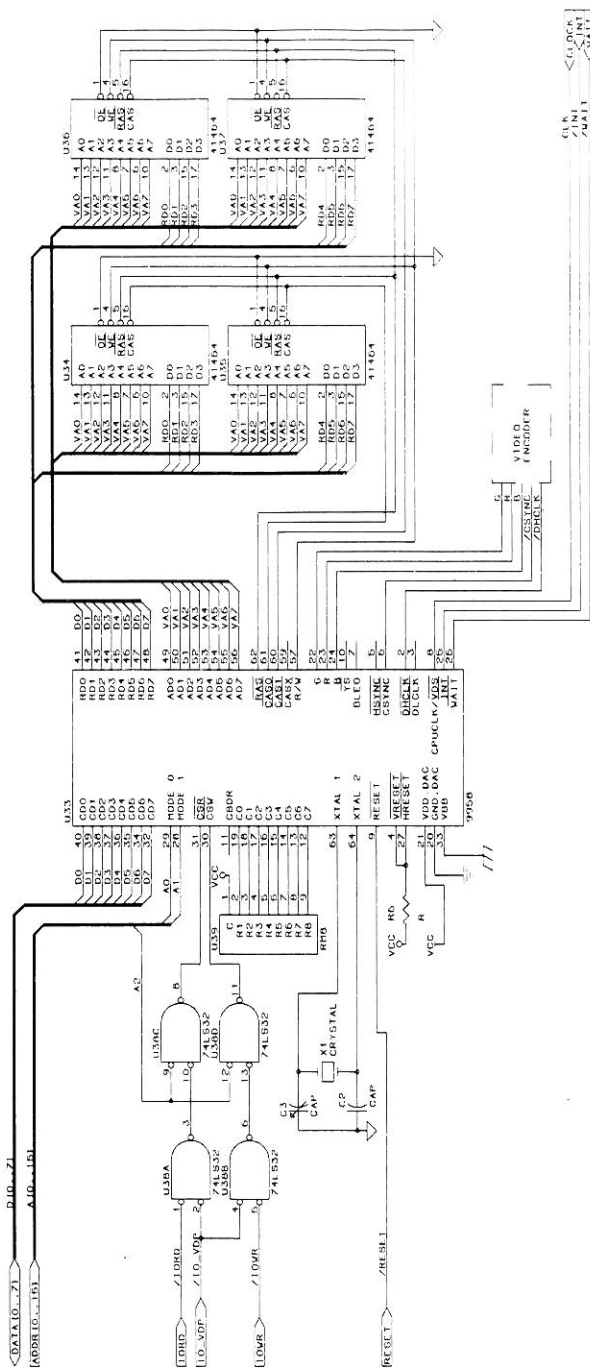
7.4 システム ROM



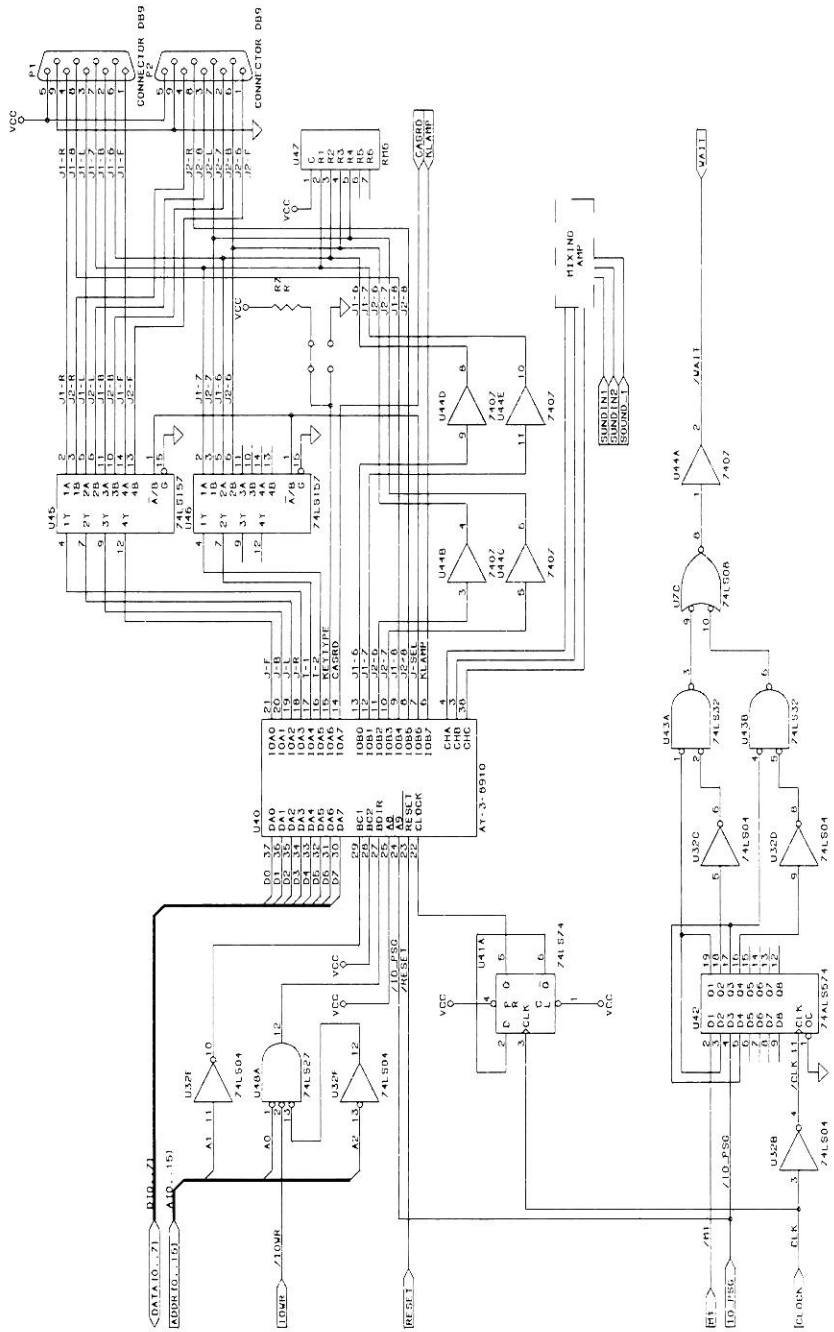
7.5 メイン RAM



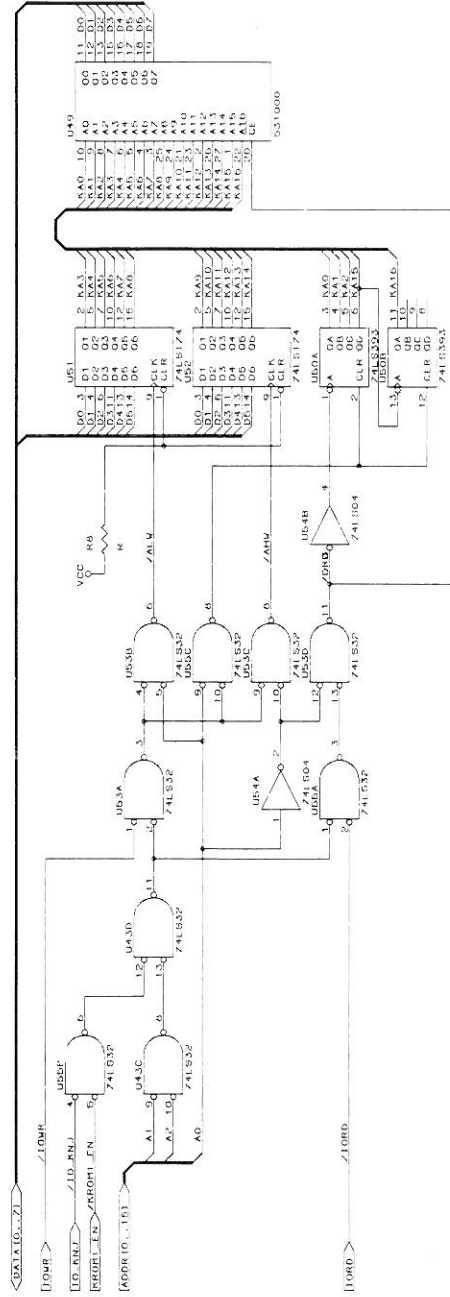
7.6 VDP



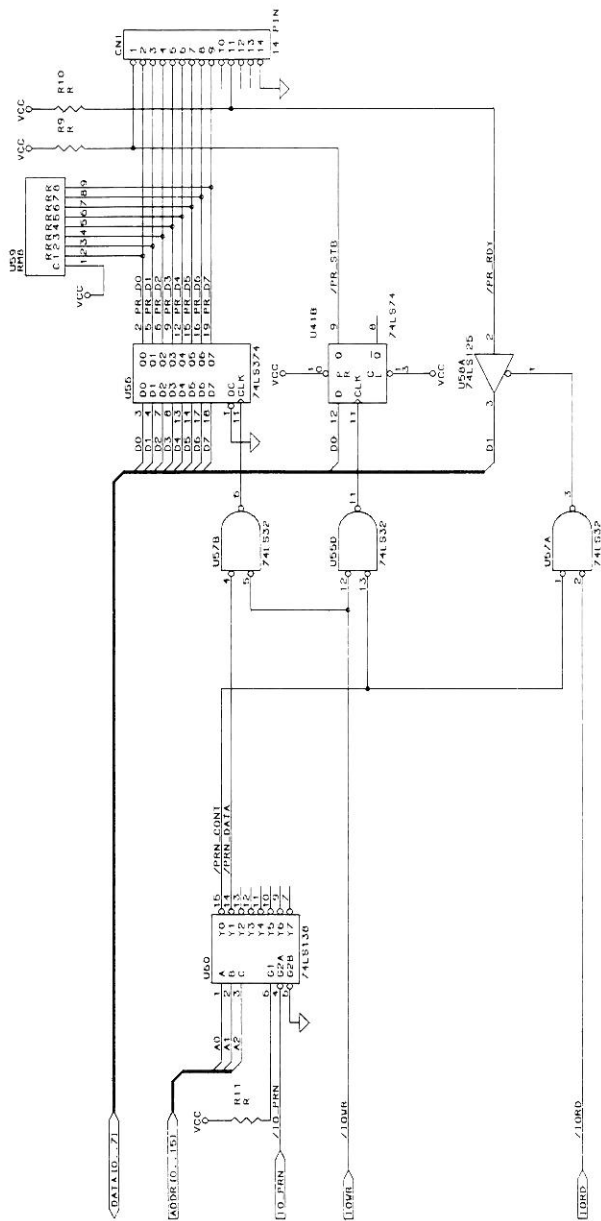
7.7 PSG とシステムウェイト



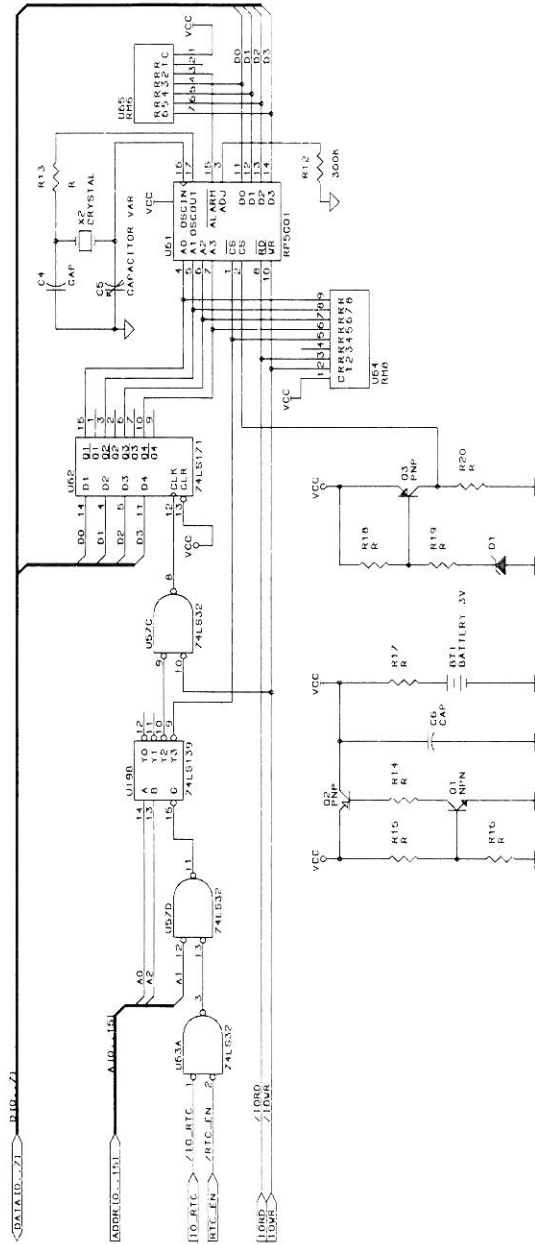
7.8 漢字 ROM



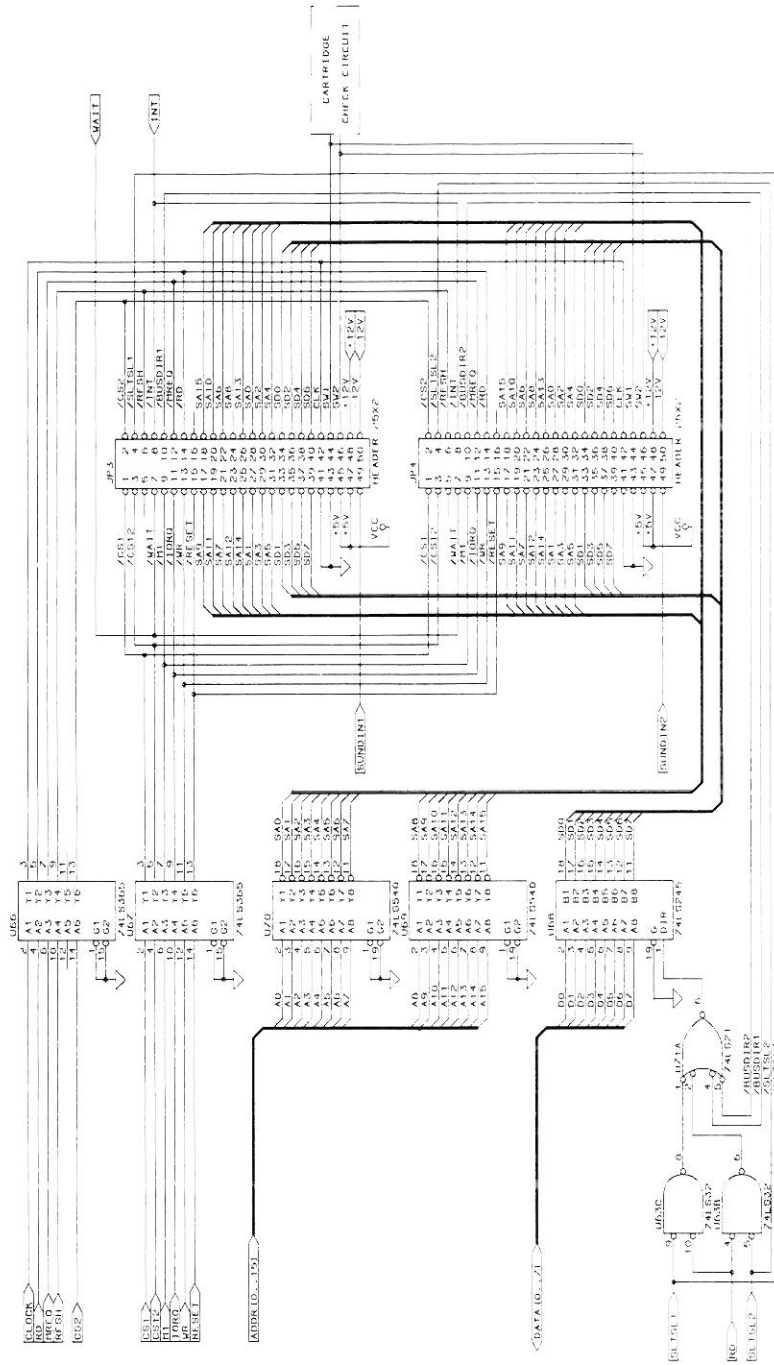
7.9 プリンターフェイス



7.10 リアルタイムクロック



7.11 カートリッジスロット



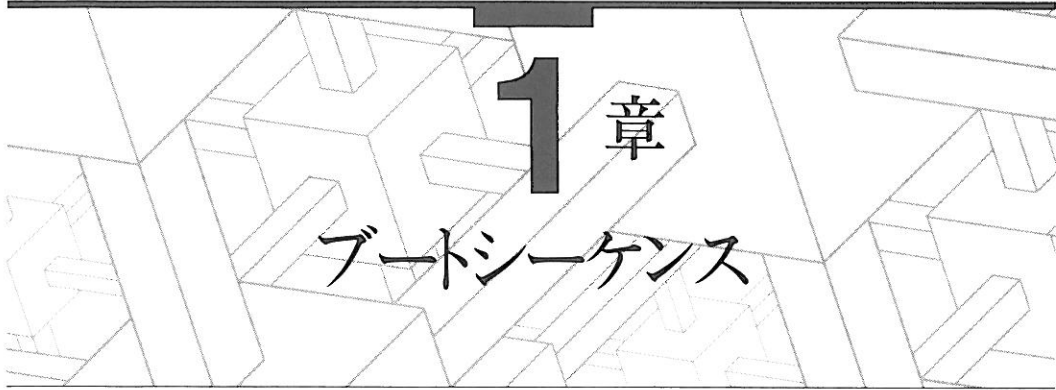


第2部

システムソフトウェア

MSX では電源投入後に、さまざまな内部的な初期化の処理を行っています。この処理により、MSX-DOS や MSX BASIC が使用できる状態になります。この処理は、すべてシステムソフトウェアと呼ばれるプログラムが行います。

第 2 部では、このシステムソフトウェアについて説明します。



MSX2 と MSX2+ では、システムソフトウェアは MAIN ROM と SUB ROM に分かれています。また、ディスクを搭載したシステムの場合は、DISK ROM もシステムソフトウェアとして内蔵されています。

この章ではこれらのシステムソフトウェアが、実際に行う初期化の手順について解説します。

1.1 MAIN ROM での初期化

MAIN ROM は下記の手順で初期化を行います。

1. マッパーおよび PPI の初期化
2. RAM のサーチ
3. システムワークエリアの初期化
4. SUB ROM のサーチ
5. ROM のサーチ
6. スクリーンの設定

1.1.1 マッパーおよび PPI の初期化

表 2.1 マッパーI/O ポートの初期値

I/O ポート	データ
0FFH	00H
0FEH	01H
0FDH	02H
0FCH	03H
0ABH	82H
0A8H	00H

ポート 0FCH から 0FFH はそれぞれページ 0 から 3 に現れるマッパーセグメントを指定するレジスタです。つまり、

ページ 3	セグメント 0
ページ 2	セグメント 1
ページ 1	セグメント 2
ページ 0	セグメント 3

が現れるように設定されます。

ポート 0ABH は PPI (i8255) のモード設定レジスタで、ポート 0A8H が出力、ポート 0A9H が入力、ポート 0AAH が出力に設定されます。

ポート 0A8H は基本スロットを選択するレジスタで、00H を出力することにより、ページ 0、1 に MAIN ROM が現れることとなります。

1.1.2 RAM のサーチ

MSX では RAM の存在するスロットを特に規定していないので、RAM のあるスロットを捜さなければなりません。RAM はページ 2、3 の順番で捜します。

ページ 2 の RAM は全てのスロットに対し、以下の順番でチェックします。

0BF00H から 0BFFFH まで1バイトずつ上位アドレス方向に
 0BE00H から 0BEFFH まで1バイトずつ上位アドレス方向に

⋮

08000H から 080FFH まで1バイトずつ上位アドレス方向に

チェックは、

1. 1バイト読み出す
2. 1の補数を取る
3. その結果を書き込む
4. それを比較する
5. データを元に戻す

という手順で行いますので、元の内容が破壊されることはありません。途中でRAMが実装されていない（正しく比較されなかった）と判断された時点で、そのスロットに対するチェックは中断します。

これを全てのスロットに対して繰り返し、最大の容量を持つスロット（同じ容量のスロットが存在する場合は、スロット番号の若いもの）が選択され、ページ2に現れます。

基本スロットが拡張されたスロットであるかどうかの判断も、ページ2のRAMチェックと同時にを行います。拡張スロットセレクトレジスタは、「書き込んだ値の1の補数が読み込まれる」と規定されていることを利用して拡張されているかどうかのチェックをしています。

ページ3のRAMも同じ様な手順で捜しますが、アドレスが以下ようになります。

0FE00H から 0FEFFH まで1バイトずつ上位アドレス方向に
 0FD00H から 0FDFFH まで1バイトずつ上位アドレス方向に

⋮

0C000H から 0C0FFH まで1バイトずつ上位アドレス方向に

このようにして見つかったページ3のRAMがシステムのワークエリアとして使用されます。

1.1.3 システムワークエリアの初期化

ワークエリアの初期化はまず 0F380H から 0FFF7H までを 0 でクリアすることから始まりま
す。その後、「1.1.2. RAM のサーチ」で得られた拡張スロットに関する情報を【EXPTBL(0FCC1H
~0FCC4H)】と【SLTTBL(0FCC5H~0FCC8H)】に書き込みます。その他の初期値はテーブ
ルからブロック転送するかプログラムにより設定します。

次に、ページ 2 と 3 で連続して実装されている RAM 容量のチェックを行い、【BOTTOM
(0FC48H)】という変数を設定します。RAM チェックの順番は以下の通りです。このときも元
の内容は破壊されません。

0FE00H から 0FEFFH まで 1 バイトずつ上位アドレス方向に
0FD00H から 0DFDFH まで 1 バイトずつ上位アドレス方向に
⋮
08000H から 080FFH まで 1 バイトずつ上位アドレス方向に

1.1.4 SUB ROM のサーチ

MSX2 では SUB ROM の存在するスロットは規定されていないので、SUB ROM を捜さなけ
ればなりません。SUB ROM は、

アドレスがページ 0 で、ROM の ID が「CD」の特殊な ROM カートリッジ

です。普通の ROM カートリッジと同様に、INIT エントリ、STATEMENT エントリ、DEVICE
エントリを持ちます。TEXT エントリは持つことができません。

SUB ROM での初期化については、「1.2 SUB ROM での初期化」で解説します。

1.1.5 ROM のサーチ

次に、MSX は ROM カートリッジを捜しますが、MSX2+ (BASIC version 3.0) の場合はそ
の前に MAIN ROM の 1472H 番地を経由します。ここには標準的には 4 バイトの NOP 命令が
書いてあるだけです。ハードウェアメーカーが他のスロットに優先して立ち上がるような内蔵ソ
フトを作りたいときには、この NOP を適当なインタースロットコール命令に置き換えれば実現

できることとなります。

ROM はスロット番号の若い順に、4000H、8000H 番地の順で、ROM の ID (「AB」) が書いてあるかどうかを調べます。

ROM ID が見つければ、INIT エントリを調べ、0 以外ならその番地をインタースロットコールします。もし、この ROM がゲームのような、ROM 内だけでループして動くソフトであったら、システムに制御が帰ってくることはありません。

次に、STATEMENT エントリ、DEVICE エントリ、TEXT エントリを調べ、0 以外であったらそのスロットに相当する【SLTATR (0FCC9H~0FD08H)】のそれぞれ BIT5、6、7 をセットします。

これを全てのスロットに対して繰り返します。

1.1.6 スクリーンの設定

クロック IC にはバッテリーでバックアップされた RAM が内蔵されていて、MSX2 ではデフォルトのスクリーンモードなどが設定されています。ここではその RAM の内容にしたがって各種のスクリーンモードを設定します。

1.2 SUB ROM での初期化

SUB ROMの初期化プログラムはINITエントリから起動されます。その内容は以下の通りです。

1. システムワークエリアの初期化
2. システムコントロールポートの設定
3. クロック IC の初期化
4. タイトルの表示

1.2.1 システムワークエリアの初期化

MAIN ROM の初期化では不十分な部分を初期化します。

1.2.2 システムコントロールポートの設定

MSX2 では、複数デバイスの競合による不都合を避けるため、標準的な内蔵デバイスに対して、その機能を禁止する機能があり、ポートの 0F5H がそれにあたります。内蔵デバイスはこのポートの対応するビットに 1 が書かれたときだけ機能するように設計されています。

表 2.2 システムコントロールポートの割り当て

ビット 0	第 1 水準漢字 ROM のためのビットです。
ビット 1	MSX2 では使用されていませんが、MSX2+では第 2 水準漢字 ROM のためのビットに割り当てられています。
ビット 2	MSX-AUDIO のためのビットです。ポート 0C0H が 0FFH 以外であったら MSX-AUDIO が存在すると判定されます。
ビット 3	AV コントロールのためのビットです。ポート 0F7H と 77H を AND したものが 77H 以外だったら存在すると判定されます。
ビット 4	MSX INTERFACE のためのビットです。ポート 0C8H が 0FFH 以外であったら MSX INTERFACE が存在すると判定されます。MSX INTERFACE は汎用の通信 LSI ですが、現在のところ使用されていません。
ビット 5	I/O マップされた RS-232C インターフェイスのためのビットです。i8251 をインターナルリセットした結果、ステータスレジスタが xx000101 (2 進数) になったら、存在すると判定されます。
ビット 6	ライトペンのためのビットです。ポート 0BBH が 0FFH 以外であったら存在すると判定されます。
ビット 7	クロック IC のためのビットですが、実際には使用されていません。

このように、漢字 ROM 以外のデバイスの判定は、MSX のバスがブルアップされている、つまり存在しないポートを読むと 0FFH が返って来ることを前提にしています。

1.2.3 クロック IC の初期化

クロック IC 内の RAM は初期化されていない場合もありますので、ブロック 2 (RAM-1) の最初の 1 ニブル (4 ビット) が 0AH でないか、WIDTH 情報が不正 (スクリーン 0 なのに 1 から 80 の範囲でない、またはスクリーン 1 なのに 1 から 32 の範囲でない) である場合は、RAM をデフォルトの内容で初期化します。デフォルトの内容は以下のとおりです。

表 2.3 クロック IC 内 RAM の初期値

ADJUST X	0
ADJUST Y	0
スクリーンモード	1
WIDTH の値	29
前景色	15
背景色	4
周辺色	7
カセットスピード	1200 ボー
プリンタモード	MSX
キークリック	あり
ファンクションキー表示	あり

1.2.4 タイトル表示

タイトル表示に関しては MSX2 と MSX2+ では大きく異なりますので、それぞれの場合で解説します。

1. MSX2 の場合

タイトルを表示して(垂直スクロールを使用)、VRAM のサイズを調べます。サイズは VRAM の 1FFFFH 番地が読み書きできるかどうかで判定します。読み書きできたら 128KVRAM で、できなかつたら 64KVRAM です。その後、VRAM サイズの表示、タイトル(MSX のロゴマーク)の表示、パスワードの表示と入力などを行います。

2. MSX2+ の場合

BIOS の RDRES を呼んでハードウェアリセットであるかどうかを調べます。ハードウェアリセットでなければ、何もしません。

ハードウェアリセットなら、メイン RAM の総容量を調べ、RAM の ID を消します(これについては後で述べます)。MSX2+ では VRAM 容量は 128K に決まっているので調べません。その後、タイトルを表示して(水平スクロールを使用)、メイン RAM の総容量の表示、タイトルの表示、パスワードの表示と入力などを行います。

メイン RAM の総容量は以下のようにして求めます。

リスト 2.1 メイン RAM 容量の算出シーケンス

```

総容量=0
全てのスロットに対して{
    ページ1にそのスロットを出す
    セグメント0の第1バイトに0AAHを書く
    セグメント1の第1バイトに055Hを書く
    if セグメント0の第1バイト=0AAH {
        マッパー容量=0
        全てのセグメントに対して
            第1バイトに0AAHを書く
        全てのセグメントに対して{
            if 第1バイト<>0AAH
                break
            第1バイトに055Hを書く
            if 第1バイト<>055H
                break
            マッパー容量=マッパー容量+1
        }
        総容量=総容量+マッパー容量
    }
}

```

上記の方法で、マッパー上に存在する全ての RAM の総セグメント数が計算できます。表示されるメイン RAM 総容量は、総セグメント数の 16 倍ですが、0 の場合は特別に 64K とみなして表示します。これはマッパー回路に接続されていない 64K の RAM が存在する場合を考慮したためです。

RAM の ID の消去は全てのスロットの 4000H、8000H 番地と、選択されている RAM の 0C010H 番地に 0 を書くことにより行われます。これはページ 1 の RAM に ROM のイメージを転送して動作するようなプログラムが、電源を ON・OFF しても動いてしまうことを防止するためです。DRAM の中には、電源を切っただけでは内容がなかなか消えないものがあるためこのような方法を取っています。また、0C010H に 0 を書くのは、ここをフラグとして使用している ROM のプログラムがあったためです。

1.3 FDD ROM での初期化

ここでは、ディスク環境での立ち上がりシーケンスについて解説します。ただし、ここで解説するのは DOS1 の初期化についてだけです。DOS2 は限られた紙面で解説するにはあまりにも複雑で、マスターカートリッジ（後述）を乗っ取ってマスターになるなど本質的なこと以外の部分が多く、理解の妨げになるためです。

また、この章の内容は、DOS のバージョンアップに伴って変更される可能性がありますので(実

際に DOS2 では違う)、あくまでも内部動作の理解のために使用して下さい。商用のアプリケーションソフトウェアでは、この内部情報は使用しないで下さい。

ディスクカートリッジは、MSX にとって特別な地位を約束されているわけではなく、一般のカートリッジと同等の扱いを受けます。ただし、システムに深く密着したソフトウェアですので、MSX で規定されていない RAM のワークエリアなどを使用しています。

以下にディスクカートリッジの INIT エントリで実行される初期化シーケンスを記述します。ここで INIHRD、DRIVES、INIENV はディスクカートリッジ内のディスクドライブのルーチンです。alloc は指定されたバイト数だけ上位アドレスから割り当てそこへのポインタを返す内部ルーチン、DISKID は 0FD99H 番地の内容で初期値は 0、p はポインタ変数、f はフラグ変数です。

DEFDPB はドライブ内部にある定数テーブルで、少なくともそのテーブルによって、ドライブがサポートしているすべてのメディアのブートセクタおよび FAT の第 1 セクタが読めることが保証されています。【\$DPBLIST (0F355H)】は DPB へのポインタを要素とする配列です。

リスト 2.2 DISK ROM の初期化シーケンス

```

INIHRD()                /* ディスクハードウェアの初期化 */
if DISKID >= 80H
    return
if DISKID = 0 {
    if SHIFT キーが押されている {
        ビープ
        DISKID = OFFH
        return
    }
    p = alloc (固定ワークエリアサイズ + ドライバワークエリアサイズ)
    固定ワークエリアを初期化
    f = CONTROL キーが押されている
    ビープ
    H.RUNC の設定
} else
    p = alloc (ドライバワークエリアサイズ)
p を SLTWRK にリンク
n = DRIVES (f)          /* ドライブ数を返す */
DRV TBL に設定
p = alloc (n * 21)
そのカートリッジ内の全てのドライブに対して {
    DEFDPB を p に転送
    p を $DPBLIST にリンク
    p = p + 21
}
INIENV()                /* ディスクドライブの環境初期化 */
DISKID = DISKID + 1

```

DISKID はディスク環境のディスエーブル、および最初のディスクカートリッジであるかどうかの判定に使われます。最初のディスクカートリッジはマスターカートリッジと呼ばれ、ディスク機能の大部分を受け持ちます。他のディスクカートリッジはスレーブカートリッジと呼ばれ、ディスクドライバの部分だけ使用されます。

マスターカートリッジは【H.RUNC (0FECBH)】というフックを設定します。MSX が全てのカートリッジのスキャンを終えて、BASIC を起動する際に必ず呼ばれるフックです。H.RUNC は以下に示すようなルーチンに接続されます。

【H.STKE (0FEDAH)】はディスクの機能を使用する ROM カートリッジが設定するフックです。

リスト 2.3 H.RUNC が接続されているルーチン

```

変数の初期化
ドライブ用セクタバッファ          = alloc (最大セクタサイズ)
DOS 汎用セクタバッファ            = alloc (最大セクタサイズ)
DOS ディレクトリ用セクタバッファ = alloc (最大セクタサイズ)
全てのドライブについて {
    p = alloc (セクタサイズ * FAT サイズ + 1)
    p + 1 をそのドライブの DPB 内の FAT ポインタにリンク
}
スクリーンの設定
ページ 0、1 の RAM を探す
SP = 0C200H
if H.STKE が設定されている
    BASIC ROM ヘジャンプ
if TEXT エントリを持ったカートリッジがある
    BASIC ROM ヘジャンプ
ブートセクタを読み最初の 256 バイトを 0C000H に転送する
if ブート不可能
    DISK BASIC を起動する
0C01EH をキャリーフラグをリセットして呼ぶ
if RAM が 64K ない
    DISK BASIC を起動する
ページ 0 の RAM を表に出す
ページ 0 を初期化する
スロットハンドラを設定する
割り込みハンドラを設定する
ブートセクタを読み最初の 256 バイトを 0C000H に転送する
if ブート不可能
    DISK BASIC を起動する
0C01EH をキャリーフラグをセットして呼ぶ

```

リスト 2.3 の 6 行目で alloc されるメモリが、FAT に必要とされるより 1 バイト多く割り当てられていることに注意して下さい。FAT の先頭の 1 バイト前はメモリ内の FAT とディスク上の FAT が一致しているかどうかのフラグです。ディスク上の FAT がまだメモリ内に読み出されていないときには 255 が、メモリ内の FAT のみを変更して、まだディスクに書き出していないときには 1 が、一致しているときには 0 が書かれています。

ディスク環境では全てのページの RAM のスロットアドレスがページ 0 から順に 0F341H、0F342H、0F343H、0F344H に格納されます。ページ 2、3 の RAM は MAIN ROM によって既に捜されていますから、それをスロットアドレスに変換して格納します。ページ 0、1 の RAM はマスターカートリッジが捜します。

ページ 0 の RAM は全てのスロットに対して、

0010H から 0001H まで 1 バイトずつ下位アドレス方向に

0410H から 0401H まで 1 バイトずつ下位アドレス方向に

⋮

3C10H から 3C01H まで 1 バイトずつ下位アドレス方向に

ページ 1 の RAM は全てのスロットに対して、

4010H から 4001H まで 1 バイトずつ下位アドレス方向に

4410H から 4401H まで 1 バイトずつ下位アドレス方向に

⋮

7C10H から 7C01H まで 1 バイトずつ下位アドレス方向に

の順でチェックされます。これも、MAIN ROM でのチェックと同様、元の内容を破壊することはありません。1K バイトおきに 15 バイトだけがチェックされることに注意して下さい。RAM が実装されていないと判断された時点でそのスロットに関するチェックは終了し、16K フルに実装されているスロットが見つかった時点で全てのチェックが終了します。

INIT エントリおよびマスターカートリッジによる初期化が終了した時点で、RAM の内容は以下のようになっています。ここではディスクカートリッジが 2 つあって、最初のカートリッジは 2 台、次のカートリッジは 1 台のディスクをサポートしているとします。

0F380H	ディスクシステム用の固定ワークエリア
0F1C9H	カートリッジ 1 のドライブ用ワークエリア
	ドライブ B の DPB (21 バイト)
	ドライブ A の DPB (21 バイト)
	カートリッジ 2 のドライブ用ワークエリア
	ドライブ C の DPB (21 バイト)
	ドライブ用セクタバッファ
\$SECBUF	DOS 汎用セクタバッファ
\$BUFFER	DOS ディレクトリ用セクタバッファ
\$DIRBUF	ドライブ A の FAT
	ドライブ B の FAT
	ドライブ C の FAT
HIMSAV	

図 2.1 マスターカートリッジ初期化終了時のワークエリア

【HIMSAV (0F349H)】はディスクシステムの核 (カーネル) が動作するのに最低限必要なエリアの底へのポインタです。【\$SECBUF (0F34DH)】、【\$BUFFER (0F34FH)】、【\$DIRBUF (0F351H)】は 3 種類のセクタバッファに対するポインタです。

2章

割り込み

この章では、MSXの割り込みについて解説します。

MSXはZ80の「モード1割り込み」と呼ばれる割り込みを使用しています。これは、割り込みがかかると必ず38H番地をコールするという割り込みモードです。MSXのMAIN ROM内の割り込みルーチンで処理していることを、以下に解説します。

リスト 2.4 割り込みシーケンス

```
全てのレジスタ (裏レジスタ、IX、IY 共) をセーブ
H.KEYI を呼ぶ
ライトペンの処理
VDP のステータスレジスタを読む
if 垂直帰線割り込み {
    H.TIMI を呼ぶ
    割り込みの許可
    STATFL の設定          /* 【STATFL (0F3E7H)】 */
    ON SPRITE の処理
    ON INTERVAL の処理
    JIFFY = JIFFY + 1      /* 【JIFFY (0FC9EH)】 */
    PLAY 文の処理
    SCNCNT = SCNCNT - 1   /* 【SCNCNT (0F3F6H)】 */
    if SCNCNT = 0 {
        SCNCNT = 2
        ON STRIG の処理
        キーボードのスキャン
    }
}
全てのレジスタ (裏レジスタ、IX、IY 共) をリストア
```

MSXの割り込み源のうち、システムが管理するのは垂直帰線割り込みだけです。周期はNTSC版の場合1/60秒、PAL/SECAM版の場合1/50秒です。日本版はNTSCですから、1/60秒になります。

また、VDPのステータスレジスタを読む際には、単にポート099Hを読むだけですから、ステータスレジスタポインタが0以外だと、割り込みがかかりっぱなしになり、MSXは暴走することに

なります。ステータスレジスタポインタを0以外に変更するときには必ず割り込みを禁止してください。そして、0に戻してから割り込みを許可してください。

【JIFFY (0FC9EH)】は垂直帰線割り込み毎にインクリメントされる1ワードの変数ですので、時間をカウントするのに使用することができます。

【SCNCNT (0F3F6H)】はキーボードおよびジョイスティックのスキャンを間引くためのものです。MSXでは2回の割り込みで1回スキャンされることになります。

ON STRIGの処理を行うためジョイスティックがスキャンされますが、このときジョイスティックポート1、2ともに、6、7番ピンがHIGHに、8番ピンがLOWになります。ジョイスティックポートに接続するための周辺機器を設計して、これらのピンを出力ピン（MSXから見ても）として使用する際、このことに充分注意してください。

具体的には6、7番ピンはHIGHが、8番ピンはLOWが定常状態となるように設計して、アクセスするときは割り込みを禁止して行う設計にしてください。

【H.KEYI(0FD9AH)】は割り込みがかかると必ず呼ばれるフックです。垂直帰線割り込み以外の割り込みを処理するようなルーチン（RS-232Cや水平帰線割り込みなどのハンドラ）は、このフックを使用します。【H.TIMI(0FD9FH)】は垂直帰線割り込みがかかると呼ばれるフックです。割り込みフックの使用法には次の2種類があります。

1. JP 命令を書く

ゲームなどのプログラムでは、一番効率の良い方法です。

前提条件としては、いつ割り込みがかかっても割り込みハンドラがCPU空間に出ていなければならないことです。つまり割り込みハンドラが「スロットが切り換わる可能性のあるページ」にはならないことになります。

割り込みハンドラからリターンするときには、

- 1) そのまま「RET」命令を実行する
- 2) スタックを1段ポップして、すべてのレジスタをリストアして「RET」命令を実行する
- 3) セーブしてある、以前のフックにジャンプする

などの方法があります。

1)では、MAIN ROMの割り込み処理が続行しますので、上で述べた処理（キーボードのスキャンなど）が必要なときは、この方法を採用します。ただし、割り込みフックを設定する前に、他のプログラムが既にこのフックを使用していたときは、それは無視されます。

2)では、MAIN ROMの割り込み処理は完全に無視されます。最高の効率が必要な場合はこの方法を採用します。ただし、MAIN ROMの割り込み処理が前提となっているような機能（BIOSのCHSNS、CHGETなど）のサービスは受けられません。また、以前の割り込みフックは無視されます。

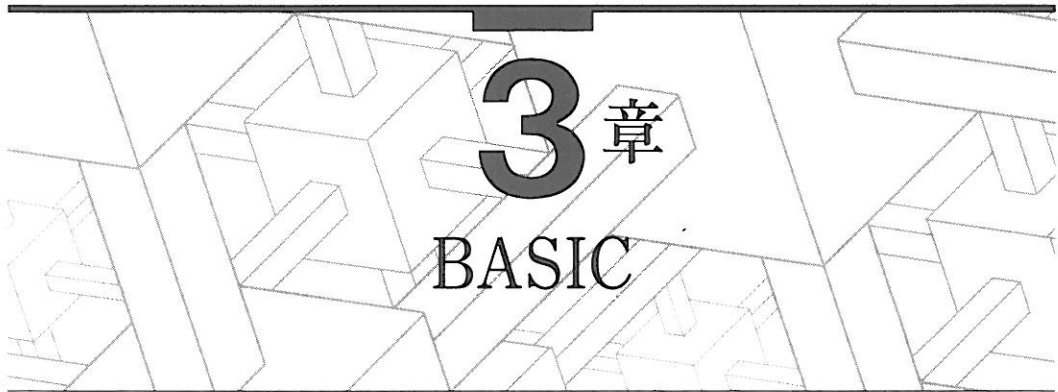
3)では、MAIN ROMの割り込み処理、以前の割り込みフックともに、実行されます。具体的には、フックを設定する前に、5バイトのセーブエリアにフックの内容をコピーして、それからフックを設定することが必要です。割り込みハンドラが終了したら、このセーブエリアにジャンプします。これを割り込みのチェーンと呼びます。

2. RST 30H (FAR CALL) 命令を書く

システムの一部として組み込まれる機能（例えば、RS-232C）を処理する割り込みハンドラは必ずこのようにしなければなりません。システムとして動いているときは、スロットが頻繁に切り換わるためです。

また、割り込みのチェーンも完全に行わなければなりません。よく、「ディスクのアクセスランプが光ったままになってしまう」という現象がありますが、これは割り込みのチェーンを行っていないプログラムが動いているからです。ほとんどのディスクドライバは垂直帰線割り込みを使用して、モーターのON・OFF（アクセスランプのON・OFF）を行っています。

添付のフロッピーディスクに、「INT.MAC」という名前のサンプルプログラムが入っています。



3.1 MSX BASIC

MSX のシステムに標準で搭載される言語は BASIC です。何度か拡張、改良され現在次のようなバージョンがあります。

MSX BASIC ver. 1.0	MSX1 の BASIC
MSX BASIC ver. 2.0	MSX2 の BASIC
MSX BASIC ver. 3.0	MSX2+ の BASIC
MSX Disk BASIC ver. 1.0	MSX-DOS1 下の Disk BASIC

MSX BASIC ver 2.0 では MSX2 で改良されたグラフィックス、バッテリーバックアップされたメモリの利用などのために機能が追加、拡張されています。

MSX BASIC ver 3.0 では MSX2+ で拡張されたグラフィックや漢字処理機能をサポートするための機能が追加、拡張されています。

3.2 BASIC の動作モード

MSX-DOS がない MSX のシステムでは、直接アプリケーションを実行する場合を除き BASIC、あるいは Disk BASIC が起動します。MSX-DOS のプロンプトから「BASIC」と入力して起動することもできます。BASIC は、ダイレクトモードとプログラムモードの2つの動作モードで実行されます。

3.2.1 ダイレクトモード

BASIC の命令文だけをキー入力すると、その文はリターンキーを押した後すぐに実行されます。これをダイレクトモードでの実行といいます。ダイレクトモードで入力された命令はメモリには記憶されません。

3.2.2 プログラムモード

BASIC の命令文の前に番号をつけてキー入力すると、その行はプログラム内の行として、番号とともにコンピュータのメモリに記憶されます。この番号のことを行番号といい、行番号の付いた命令文の集まりをプログラムといいます。プログラムは RUN 命令や GOTO 命令などを使って実行させることができます。これをプログラムモードでの実行といいます。

1. プログラムの作成

プログラムとは、行の集まりです。行番号は 0~65529 の整数で、できたプログラム行は行番号の小さい順に実行されます。BASIC の命令はアルファベットの大文字で入力しても小文字で入力しても、自動的に大文字に変換されて処理されます。ただし、引用符 (") で囲まれた文字や、REM やアポストロフィー (') に続くコメント文は、そのまま記憶されます。

2. プログラムの修正

プログラムの修正は、以下のようにします。

■ 行の追加

追加したい内容に、追加したい行番号をつけて入力します。入力と同時に行は番号順に並べ換えられて記憶されます。

■ 行の削除

削除したい行の行番号だけを入力すると、その行は削除されます。まとめて削除するときには、DELETE 命令を使います。

■ 行の置き換え

訂正したい行の行番号と新しい内容をキー入力してリターンキーを押せば、古い行の内容は自動的に新しいものになります。また、LIST 命令を使ってプログラムを画面に表示し、カーソルを直したい所に移動して直接書き換えてからリターンキーを押しても、その行は訂正されます。

MERGE 命令によって、ファイルから新しい行を読み込むこともできます。

■ 行番号のつけかえ

行番号をつけかえたいときには、RENUM 命令を使います。

3.3 BASIC の文法

3.3.1 BASIC で使用できる文字と特殊記号

MSX BASIC で使用できる文字は、英文字 (大文字・小文字)、数字、カナ文字、ひらがな、特殊記号、そしてグラフィック文字です。文字についての詳細は、巻末の「キャラクタコード表」を参照してください。また、漢字 BASIC のモードではシフト JIS コードによって全角漢字を処理することができますが、この場合は半角ひらがなは扱いません。

1. 特殊記号

使用できる文字の中でも、特殊記号と呼ばれる文字はそれぞれ特別な意味を持っています。特殊記号のうち、演算子については、「式と演算」で説明します。

- ・ (ピリオド) 現在 BASIC が着目している行番号を表します。
例) プログラム実行中にエラーが発生したとき、
LIST.
と入力すると、エラーの発生した行が表示される。

- － (マイナス) 数値の範囲を指定するときに使います。
例) LIST 命令で表示する行の範囲を指定するとき、
LIST 100-200
とすれば、100 行から 200 行までが表示される。

- ： (コロン) マルチステートメントの区切りとして使います。
例) A = B+C : PRINT A

- ， (カンマ) データを複数個指定するときに、その区切りとして使います。PRINT 命令で使われた場合には、偶数番目のデータは左から15桁目に表示されます。
例) INPUT A,B,C

- ;(セミコロン) データを複数個指定するときに、その区切りとして使います。PRINT 命令で使われた場合には、データは前のデータに続いて表示されます。
例) PRINT A;B;C
- '(アポストロフィ) REM 命令の代わりに用いて、注釈文を作ります。
例) 'コノキ ヨウハ コメントテ ス
- ?(疑問符) PRINT 命令の代わりに用います。
例) ? 25*12
- (空白) プログラムを見やすくするために、文中に好きなだけ空白を入れることができます。ただし、予約語(巻末の「予約語表」参照)の中には空白を入れることはできません。また、文字定数(「定数」参照)の中の空白は文字としての意味を持っています。文中の空白はプログラムの実行には影響を与えませんが、メモリには空白も記憶されます。

3.3.2 BASIC で扱うデータ

BASIC で扱うデータは、次のように分類することができます。

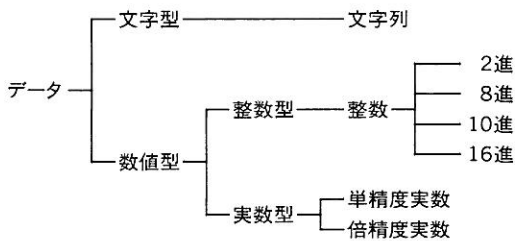


図 2.2 BASIC で扱うデータ

1. 文字列

255 文字以内の文字をつないだデータです。文字列には、BASIC で使用できる文字ならばどれでも使うことができますが、文字列を算術演算に使うことはできません。

2. 数値

数値は算術演算を行うことができるデータです。数値には整数型と実数型とがあり、数値の取る値はそれぞれで異なります。

3. 整数

-32768 から+32767 までの整数です。整数型の数値を使う演算は実行時間が短くて済みます。

4. 実数

実数には有効桁数によって、単精度実数と倍精度実数とがあります。MSX BASIC では、実数を使った演算はすべて倍精度実数に変換して行われ、結果も倍精度実数で得られますが、単精度実数として記憶したり、結果を単精度実数で受け取ることもできます。

5. 単精度実数

有効桁 6 桁の実数です。単精度実数で表せる数値は-9.99999E+62 から+9.99999E+62 で、指数の範囲は E+62~E-64 までです。

6. 倍精度実数

有効桁 14 桁の実数です。倍精度実数で表せる数値は-9.99999999999999E+62 から+9.99999999999999E+62 で、指数の範囲は E+62~E-64 までです。

また、数値型データは必要に応じて、その型を他の型に変換することができます。型の変換は、次の規則にしたがって行われます。

- 数値を異なる型の数値変数に代入する場合、数値は代入先の変数の型に変換される。
- 精度の異なる数値を使った演算では、精度の低い方が高い方の型に変換されて演算が行われる。
- 論理演算の場合、扱われる数値はすべて整数型に変換され、結果は整数で得られる。
- 実数が整数に変換される場合は、小数点以下は切り捨てられる。このとき、切り捨てた結果が整数型で扱える範囲（-32768~32767）を越えていればエラーとなる。
- 倍精度実数型変数を単精度実数型変数に代入すると、変数の値は有効数字 6 桁に直されたものになる。もとの倍精度実数型の 7 桁目以降は、四捨五入される。
- 文字型と数値型のデータの変換は、VAL、STR\$関数を使う。

3.4 定数

定数とは、それぞれに固有の値を持ったデータのことです。定数の書き方は、先に述べたデータの型によって異なります。

3.4.1 文字定数

文字定数は、文字列を引用符 (") で囲んで表します。文字列の長さは255文字以内でなければなりません。また、数字を引用符で囲むと文字列とみなされ、算術演算に用いることはできません。

例

"THANK YOU"

"3.14159265"

3.4.2 整数型定数

整数型の定数には、以下の4つの形式があります。

1. 10進形式

-32768～+32767 までの整数です。負の整数の場合、数値の前に必ずマイナス符号 (-) をつけなければなりません。プラス符号は省略できます。また、-32768～+32767 までの実数の後ろに % をつけると、小数点以下は切り捨てられて整数型の定数とみなされます。

例

0

-123

3875.6%

2. 8進形式

先頭に &O をつけて 8進数 (0～7) を並べたものです。範囲は &O0～&O177777 までですが、&O177777～&O100000 までは負の数で、-1 から -32768 に対応します。

例

&O012345 10進数では5349
 &O177777 10進数では-1

3. 16進形式

先頭に&Hをつけて16進数(0~9、A~F)を並べたものです。範囲は&HO~&HFFFFまでですが、&HFFFF~&H8000までは負の数で、-1から-32768に対応します。

例

&H100 10進数では256
 &HFFFF 10進数では-1

4. 2進形式

先頭に&Bをつけて0と1を並べたものです。範囲は&B0~&B1111111111111111までですが、&B1111111111111111~&B1000000000000000までは負の数で、-1から-32768に対応します。

例

&B01100100 10進数では100
 &B1010101010101010 10進数では-21846

8進形式、16進形式、2進形式で入力された数値は、PRINTやLPRINT命令で出力すると、10進形式で表示されます。10進以外の形式で出力するときは、それぞれOCT\$, HEX\$, BIN\$を使って文字列として出力します。

3.4.3 単精度実数型

単精度実数型の定数には、以下の3つの形式があります。

1. 6桁以内の実数。
2. 6桁以内の実数あるいは整数と、Eを使った指数表示の組み合わせ(Eの範囲はE-64からE+62)。
3. 最後に!をつけた数値(数値が6桁を越えていると、7桁目が四捨五入される)。

例

```
1.23
-7.06E+06
3.14!
```

3.4.4 倍精度実数型

倍精度実数型の定数には、以下の3つの形式があります。

1. 7桁以上の実数
2. 7桁以上の実数あるいは整数と、Eを使った指数表示の組み合わせ（Eの範囲はE-64からE+62）Eの代わりにDを使うこともできる。
3. 最後に#をつけた数値（数値が14桁を越えていると、15桁目が四捨五入される）

例

```
12345678
-1.5670D-12
246.876 #
```

3.5 変数

変数は、プログラム中で使うデータを格納するものでメモリ中に用意され、英字1文字あるいは英数字2文字からなる変数名がついています。変数に値を与える場合、変数の型は、それに代入されるデータの型と一致していなければなりません。変数に値を代入する前は、数値変数は0、文字変数はヌル（文字が何も入力されていない状態）であるものと見なされます。

3.5.1 変数の型

代入するデータの型に応じた変数の型は、変数名の最後に型宣言文字をつけることによって区別されます。型宣言文字の種類は以下の4つです。

% 整数型

!	単精度実数型
#	倍精度実数型
\$	文字型

この型宣言文字を用いることにより、変数名が同じであっても別の変数として扱うことができます(例えば、A\$, A! など)。なお、型宣言文字を省略すると、# (倍精度実数型) がついているとみなされます。

変数の型はプログラム中で、DEFINT・SNG・DBL・STR 命令を使用してグローバルに宣言することも可能です。この場合でも型宣言文字による指定は優先します。

3.5.2 配列変数

配列変数は、同じ性質を持つ複数個のデータの集まりで、X(1)、X(2)のように、変数名の後にカッコで番号をつけた形で用います。カッコの番号は添字といい、配列の中の何番目の要素であるかを表します。添字は常に0から始まります。

配列を使うには、あらかじめ DIM 命令によって配列を宣言し、変数の名前とそこに入る要素の最大数を決めます。DIM 命令を省略して配列を使用すると、添字の最大値は自動的に10に設定されます。

複数の添字を持つ配列変数を使うこともできます。複数の添字を使った場合、配列は添字の個数に応じた次元になり、例えば、X(0, 1, 5)のように用います。配列の次元は、いくつ添字を指定するかによって決まります。次元は、理論的には255次元までとれることになっていますが、実際には1行の長さで指定できないので、もっと少なくなります。

3.5.3 システム変数

MSX BASIC には BASIC 自身が専用の目的のために持っている変数(システム変数)があります。システム変数は以下の9つです。

- TIME 1/60秒ごとに値が増える変数で、時間を計ることができる。この変数には0~65535の値を代入することもできる。ただし、割り込みが禁止されている期間中は値が変化しないので、フロッピーディスクのアクセスなどがあるときは注意が必要。
- ERL エラーが発生したとき、エラーが発生した行番号を値として持つ。この変数に値を代入することはできない。

- ERR エラーが発生したとき、エラーの原因を表すエラーコードを値として持つ。この変数に値を代入することはできない。
- BASE(n) 画面出力に関連するテーブルのアドレスを値として持つ。
- SPRITE\$(n) 定義したスプライトパターンを値として持つ。
- VDP(n) VDPレジスタのデータを値として持つ。
- CSRLIN 画面上のカーソルの垂直方向の位置を値として持つ。
- LPOS(n) プリンタのヘッドの水平位置を値として持つ。
- POS(n) 画面上のカーソルの水平方向の位置を値として持つ。

以下に、各変数が使用するメモリの大きさを示します。各変数の領域がメモリ上に確保されるのは、変数に値を代入した時点、あるいは DIM 命令で配列を宣言したときだけです。

1. 変数

- 整数型 5 バイト
- 単精度実数型 7 バイト
- 倍精度実数型 11 バイト
- 文字型 6 + (文字数) バイト

2. 配列変数

- 整数型 $5 + 2 \times (\text{要素数}) + 2 \times (\text{次元数}) + 1$ バイト
- 単精度実数型 $5 + 4 \times (\text{要素数}) + 2 \times (\text{次元数}) + 1$ バイト
- 倍精度実数型 $5 + 8 \times (\text{要素数}) + 2 \times (\text{次元数}) + 1$ バイト
- 文字型 $5 + 3 \times (\text{要素数}) + 2 \times (\text{次元数}) + 1 + (\text{各要素に含まれる文字列の総文字数})$ バイト

3.6 式と演算

式とは、演算（計算）の手順を示すものであり、定数や変数を演算子（計算に使う特殊記号のこと）で結んだものです。式の演算結果は1個の数値または文字列になり、単に数値や文字だけのものも式と呼びます。

例

”BASIC”

$$10+3/5$$

$$2$$

$$A+B/C-D$$

MSX BASIC の演算は、次の5つに分けられます。

- 算術演算
- 関係演算
- 論理演算
- 関数
- 文字列演算

以下にこれらの演算について説明します。

3.6.1 算術演算

算術演算子には次のようなものがあります。なお、算術式の中に文字定数や文字変数が入ってはいけません。

表 2.4 算術演算

演算子	演算	例
^	指数演算	X^Y
-	マイナス符号	$-X$
*, /	かけ算、わり算	$X*Y, X/Y$
+, -	足し算、引き算	$X+Y, X-Y$

演算の優先順序も、この順番になります。順序を変更したいときにはカッコ (()) を使えば、カッコ内の演算子は他の演算子より先に実行されます。

1. ¥とMOD

整数のわり算は/のかわりに¥によって行えば、割り切れなかった場合に答の小数点以下が切り捨てられます。扱う数値が実数の場合は、演算が実行される前に小数点以下が切り捨てられます。また、余りを求めたいときにはMODを用います。この場合も、扱う数値が実数の場合は、演算が実行される前に小数点以下が切り捨てられます。

例

PRINT 10¥3

表示 3 (10/3 = 3 余り 1)

PRINT 10¥2.5

表示 5 (10/2 = 5 余り 0)

PRINT 13.3 MOD 4

表示 1 (13/4 = 3 余り 1)

2. 0でのわり算

数値を0で割ったときには、「Division by zero」というエラーメッセージが表示され、プログラムを中断します。0に対して負のべき乗を行った場合も同様です。

3. オーバーフロー(桁あふれ)

代入や演算の結果が、その変数の型の中で扱える範囲を越えた場合、桁あふれが occurs。この場合、「Overflow」というエラーメッセージが表示され、プログラムを中断します。

3.6.2 関係演算

関係演算子は、2つの数値または2つの文字列を比較するときに用います。関係演算の結果は、真(-1)、偽(0)で出され、IF命令などでプログラムの流れを変えるのに用いられます。以下に、関係演算子の意味と例をあげます。

表 2.5 関係演算

関係演算子	内 容	例
=	等しい	X = Y
<>, ><	等しくない	X <> Y, X > < Y
<	小さい	X < Y
>	大きい	X > Y
<=, =<	小さいか等しい	X <= Y, X =< Y
>=, =>	大きいか等しい	X >= Y, X => Y

例

IF X = 0 THEN 100

(もし X が 0 だったら、100 行にジャンプせよ)

IF A+B <> 0 THEN X = A+B

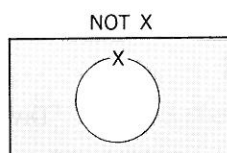
(もし A+B が 0 でなかったら、X に A+B の値を代入せよ)

3.6.3 論理演算

論理演算子は複数の条件を調べたり、ビット操作やブール演算を行ったりするのに使います。論理演算の結果は、ビットごとに0または1で返されます。各論理演算の内容を以下に示します。

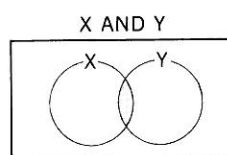
NOT (否定)

X	NOT X
1	0
0	1



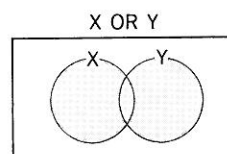
AND (論理積)

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0



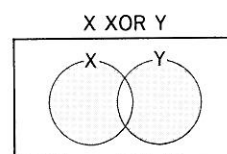
OR (論理和)

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0



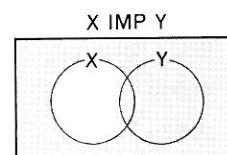
XOR (排他的論理和)

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0



IMP (包含)

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1



EQV (同値)

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

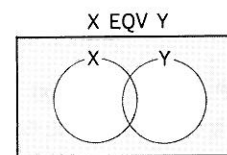


図 2.3 論理演算

IF 命令では、これらの論理演算子を使って、複数の条件を判断することができます。

例

```
IF X<0 OR 99<X THEN 100
  (X が負または 99 より大きければ、100 行にジャンプする)
IF 0<X AND X<100 THEN X = 0
  (X が正でしかも 100 より小さければ、X に 0 を代入する)
IF NOT(A = 0) THEN 20
  (A が 0 でなければ、20 行へジャンプする)
```

3.6.4 関数

関数とは、指定された数値に対して、ある決まった演算を行うもので、その演算の結果を返すものです。

MSX BASIC には、関数として、SIN や SQR などの数値を扱うものや、CHR\$や LEFT\$などの文字列を扱うものなどがあります。また、最初から定義されたものではなく、ユーザーが自由に定義できる関数 (DEFFN、DEFUSR) もあります。

3.6.5 文字列演算

1. 文字列の連結

文字列は、演算子「+」を使ってつなげることができます。

例

```
A$="ABC": B$="DEF": C$= A$+B$: ? C$
```

表示	ABCDEF
----	--------

2. 文字列の比較

文字列も、数値の比較に使われるものと全く同じ関係演算子 (=、<、>、<>、><、<=、=<、=>、>=) を用いて比較することができます。

文字列の場合、それぞれの文字列の先頭から 1 文字ずつ文字を比較します。両者が全く同じ文字列の場合は、その 2 つの文字列は等しくなりますが、1 文字でも違った場合は、その文字のキャラクタコードの大きい方の文字列が大きくなります。また、文字列の一方が短くて比較が途中で

終わった場合は、短い文字列の方が小さくなります。

例

"AB"<"BC"

"BC">"AAA"

"CAT"<"CATS"

"PEN ">"PEN"

"cm">"CM"

"BASIC"="BASIC"

文字列を比較することによって、文字列の内容を調べたり、文字をアルファベット順に並べかえたりすることができます。

3.6.6 演算の優先順位

演算は次の順位によって行われます。

1. カッコで囲まれたもの
2. 関数
3. 指数 (べき乗)
4. マイナス符号 (－)
5. *、/
6. %
7. MOD
8. +、－
9. 関係演算子 (=、<、>など)
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

3.7 エラーメッセージ

プログラムやダイレクトモードでの命令を実行しているときに、エラーが発生した場合、実行は中断され、エラーメッセージが画面に表示されます。

エラーメッセージは、ダイレクトモードでは、

XX.....

プログラムモードでは、

XX.....in YYYY

という形で表示されます。このとき XX.....はエラーメッセージで、YYYY はエラーが見つかった行番号です（必ずしもエラーの原因となる行番号ではありませんので注意してください）。エラーメッセージの内容については、「3.14 エラーメッセージ一覧」を参照してください。

3.8 画面モード

MSX の画面は次のものがあります。

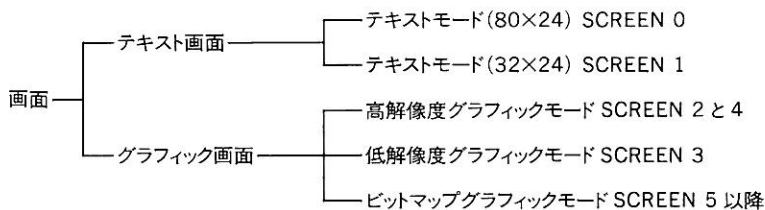


図 2.4 画面モード

テキスト画面は主に文字や数を表示するための画面で、リストやメッセージを表示することができます。テキスト画面には最大横 80×縦 24 文字表示できるモードと、最大横 32×縦 24 文字表示できるモードとがあります。

グラフィック画面は点や線を描くための画面で、細かい点や線が描ける高解像度グラフィックモードと、4×4 ドットのブロック単位で絵を描く低解像度グラフィックモードと、画面の 1 ドット

ト毎に色を付けることができるビットマップグラフィックモードがあります。

MSX BASIC ではこれらの 12 の画面のどれを使うかを、SCREEN 命令を使って指定します。BASIC 起動時には、自動的に 32×24 文字のテキストモードが選択されますが、MSX BASIC ver. 2.0 以降ではどのモードで立ち上がるかを SET SCREEN 命令でバッテリーバックアップされたメモリに記憶しておくことができます。各画面の色は COLOR 命令を使って変えることができます。使用できる画面は MSX BASIC ver. 1.0 では 0~3 (ただし SCREEN 0 は横 40×縦 24 キャラクタ)、ver. 2.0 では 0~8、ver. 3.0 では 0~12 (ただし 9 は使用できない) です。

1. テキスト画面

文字を表示するための画面です。1 行の文字数は WIDTH 命令を使って変えることができます。ファンクションキーを表示しているときは、画面の一番下の行は使えません (KEY ON / OFF 参照)。

LOCATE 命令での位置指定は、横 (X 軸方向) が 0~79 (32×24 の画面では、0~31)、縦 (Y 軸方向) が 0~23 の値で指定できます。

例

LOCATE 10,10

■ テキストモード (縦 80×横 24)

最大 80×24 文字を表示できます。1 文字の大きさは、横 6×縦 8 ドットです。この画面では、ひらがなの右側が欠けて表示されます。WIDTH 40 以下のときは文字の大きさが 2 倍になります。

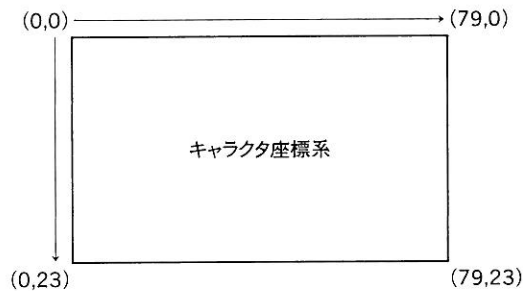


図 2.5 テキストモード (80×24)

■ テキストモード (縦 32×横 24)

最大 32×24 文字を表示できます。1 文字の大きさは、横 8×縦 8 ドットです。この画面では、スプライト機能を使うことができます。

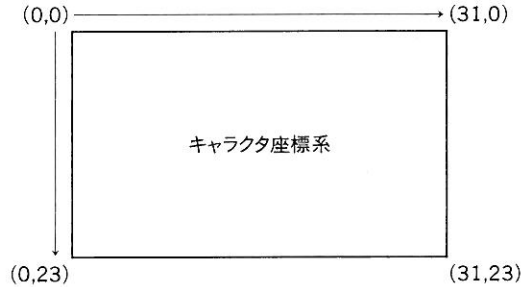


図 2.6 テキストモード (32×24)

2. グラフィック画面

点や線、絵を描くための画面です。スプライト機能を使うこともできます。グラフィック画面はプログラム中でだけ使うことができます。プログラムの実行が終了と、画面は自動的にテキスト画面に戻ります (SCREEN 参照)。

グラフィック画面で文字を表示するには、OPEN 命令で"GRP:"を指定してから、PRINT #命令を使って行いますが漢字グラフィックモードでは普通の PRINT 文でも可能です。

グラフィック画面でどの位置に点を表示するかなどの指定は、横 (X 軸方向) が 0~511、縦 (Y 軸方向) が 0~211 までの値で行います。

例

```
PSET(100,100)
```

■ 高解像度グラフィックモード (SCREEN 2 と 4)

横 256 個×縦 192 個のドットを使うことができる画面です。細かい点や線を描くことができます。横方向の 8 ドットごとに、2 色までの色を使えます。

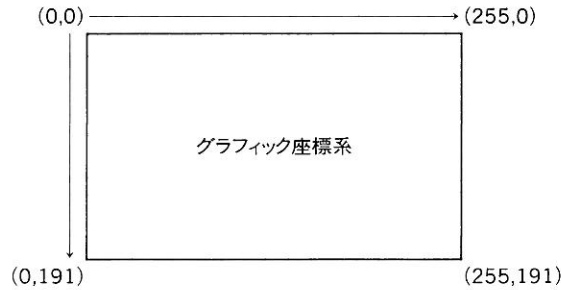


図 2.7 高解像度グラフィックモード

■ 低解像度グラフィックモード (SCREEN 3)

横 64×縦 48 ブロックのグラフィック画面で、点を描く単位が 4×4 ドットになっています。この 1 ブロックごとに色を付けることができます。

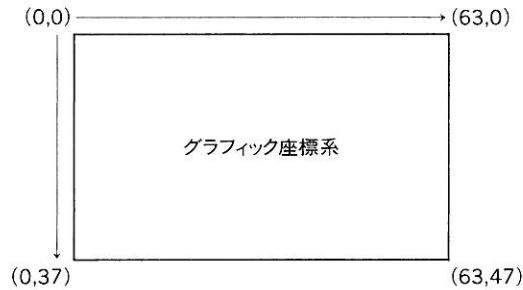


図 2.8 低解像度グラフィックモード

■ ビットマップグラフィックモード (SCREEN 5 以降)

横 256 個 (SCREEN 6 と 7 では 512) × 縦 212 個のドットを使うことができる画面です。細かい点や線を描くことができます。1 ドット毎に 16 のパレットから選んだ色が使えます。ただし SCREEN 8 では各ドットに 256 の固定色から選んだ色を使います。

3.9 カラー番号

MSX BASIC では、基本的には次の 16 色の色を使うようになっています。色の指定は、それぞれのカラー番号で行います。指定の方法は、それぞれの命令を参照してください。

表 2.6 カラー番号

カラー番号	色	カラー番号	色
0	透明（周辺色と同じ色）	8	赤
1	黒	9	明るい赤
2	緑	10	黄
3	明るい緑	11	明るい黄
4	暗い青	12	暗い緑
5	明るい青	13	紫
6	暗い赤	14	灰
7	水色	15	白

各カラー番号がこのように決まっているとき、「カラーパレットが初期状態にある」と言います。カラーパレットを上記以外の色にするときは、COLOR = を使ってパレットを変更します。例えば、

```
COLOR = (8,7,7,0)
```

とすることによってパレット 8 は、明るい黄色になります。SCREEN 6 では、0~3、それ以外では 0~15 のパレットが使えます。ただし SCREEN 8 では、パレットは使いません。カラー番号は次のように固定した色を意味しています。

```
&B00000000   カラー番号を 2 進数で表したもの
GGGRRRBB     対応する緑、赤、青の明るさ
```

例えば、&B11100000なら明るい緑になります。

3.10 スプライト機能

MSX BASIC には、解説した 4 つの画面のほかにスプライト面というものがあります。スプライト面は、0 番から 31 番までの 32 枚があり、1 枚の面に 1 個のスプライトパターンを表示することができます。

スプライト面は、普通の画面の前に重ねてある面と考えることができます。スプライト面にパターンを表示すると、そのパターンが元の画面の絵や文字の上に重なって見えます。また、2 つ以上のスプライトパターンを重ねると、スプライト面番号の小さいほうのパターンが手前に見え、立体的なパターンようになります。

スプライト機能は、SCREEN 0 以外のモードを使っているときに使うことができます。

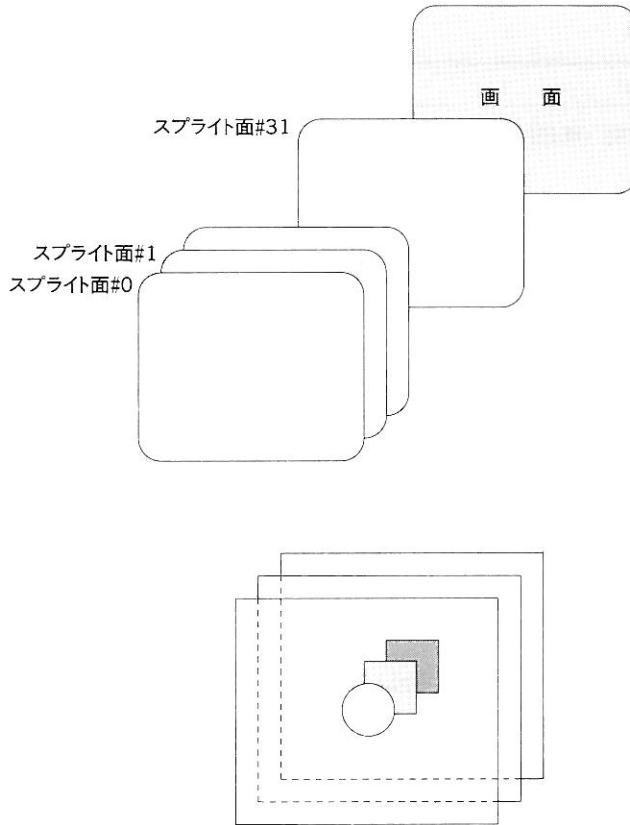


図 2.9 スプライト

パターンの大きさは、 8×8 ドット、あるいは 16×16 ドットです。 8×8 のときは、256 個、 16×16 のときは 64 個までのパターンを定義しておくことができます。また、パターンは同時に 32 個まで表示できますが、横方向に 5 個 (SCREEN 4 以降では 9 個) 以上重なると、スプライト番号の小さいパターン 4 つ (SCREEN 4 以降では 8 つ) だけが表示されます。

スプライトを表示するには、以下の手順で行います。

1. SCREEN でスプライトパターンの大きさを決める。
2. SPRITE\$ でスプライトパターンを定義する。
3. SCREEN 4 以降であれば COLOR SPRITE で各プレーンの色を定義する。
4. PUT SPRITE で、スプライト面にパターンを表示する。

このほかに、スプライトに関する命令として、以下のものがあります。

ON SPRITE GOSUB	パターンが重なったときに、指定した行にジャンプする。
SPRITE ON / OFF / STOP	ON SPRITE GOSUB で指定された行へジャンプするかどうかを決める。

3.11 ファイル

ファイルとは、意味を持つ情報の集まりです。カセットテープに保存したプログラムやデータは、ファイルとして扱われます。また他の周辺装置への出力も、まとまったデータを送るという意味で、ファイルというとならえかたで扱うことができます。

1. ファイル番号

プログラム中でファイルを扱うときには、まず OPEN 命令でファイルの使用開始を宣言し、ファイルに対して番号を割り当て、また、番号と対応した入出力用の領域を確保しなければなりません。この番号をファイル番号と呼びます。その後のファイルへの入出力は、すべてこのファイル番号によって指定されたファイルに行われます。

2. ファイルを扱う場所

ファイル进行操作する場所（ファイルの入出力の対象となる場所）には、以下の 5 つがあります。

ファイルを扱う所	指定の方法	なされる操作
カセットレコーダ	"CAS:ファイル名"	入出力を行う
メモリディスク	"MEM:ファイル名"	入出力を行う
テキスト画面	"CRT:"	出力のみ
グラフィック画面	"GRP:"	出力のみ
プリンタ	"LPT:"	出力のみ

3. ファイル名

ファイル名とは、ファイルにつける名前のことです。ファイル名を必要とするのは、カセットテープやメモリディスクへの入出力を行う場合で、その他の場合は省略できます。カセットテープで使えるファイル名は、6 文字で、メモリディスクでは 8 文字です。6（または 8）文字より小さい場合は、あいた部分には空白があてはめられます。6（または 8）文字より大きい場合は、先頭から 6（または 8）文字がファイル名とみなされ、それ以降の文字は無視されます。また、ファイル名の中に、コロン（:）を用いてはいけません。

3.12 割り込み

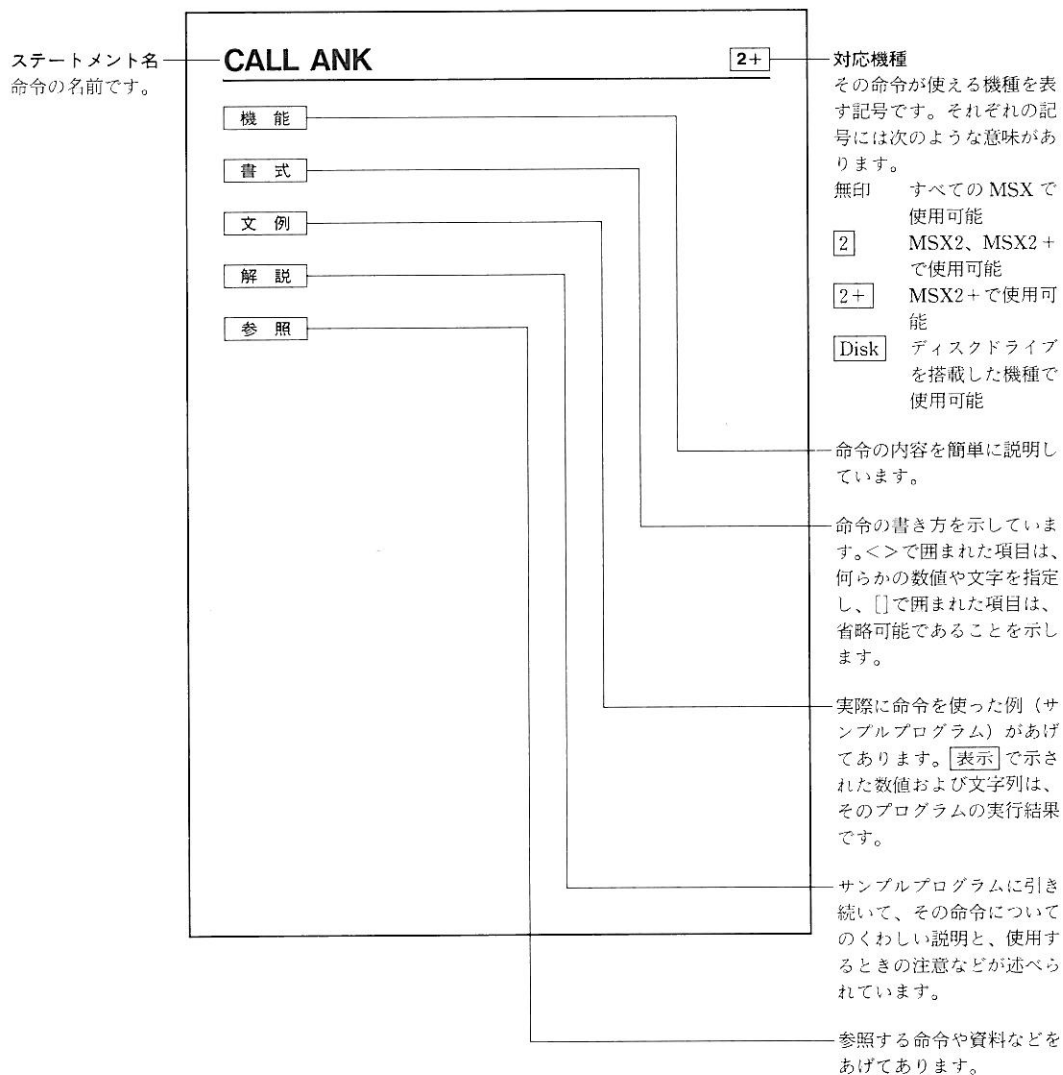
プログラムの実行中に、何か特別な事が起きたとき、現在実行中の処理を一時中止して、その特別な事柄に対応した処理を先に実行させることを、割り込みといいます。

例えば、ファンクションキーが押されたときに対応する処理を実行するようあらかじめ指定しておく、プログラムの実行中にファンクションキーが押された時点で割り込みがかかり、そのとき実行していた処理を中断して指定された処理を優先的に実行したあと、中断されていた処理が再開されます。

以下に、MSX BASIC が用意している割り込みを、優先順に示します。

- | | |
|---------------------|-------------------|
| 1. ファンクションキー割り込み | ON KEY GOSUB |
| 2. ストップキー割り込み | ON STOP GOSUB |
| 3. スプライトの衝突割り込み | ON SPRITE GOSUB |
| 4. ジョイスティックのトリガ割り込み | ON STRIG GOSUB |
| 5. インターバルタイマ割り込み | ON INTERVAL GOSUB |

3.13 ステートメント



ABS

機能

<数式>の絶対値を返します。

書式

ABS(<数式>)

文例

```
PRINT ABS(-1.2)
```

表示 1.2

ASC

機能

<文字式>のキャラクタコードを返します。

書式

ASC(<文字式>)

文例

```
PRINT ASC('a')
```

表示 97

解説

指定した<文字式>がいくつかの文字からなる文字列の場合は、一番最初の文字のキャラクタコードが返されます。文字とキャラクタコードの対応については、キャラクタコード表を参照してください。

参照

CHR\$, キャラクタコード表

ATN

機 能

<数式>の逆正接（アークタンジェント）を計算します。

書 式

ATN(<数式>)

文 例

```
PRINT ATN(1) * 4
```

表示 3.1415926535898

AUTO

機 能

プログラム行の先頭に行番号を自動的に発生させます。

書 式

AUTO[<行番号>][,<増分>]

文 例

```
AUTO 1000,100
```

解 説

最初の行番号も番号の増加分も省略できますが、この場合は自動的に10が採用されます。

例えば、

AUTO

行番号は 10 を先頭に 20、30・・・と付けられます。

AUTO 100

行番号は 100 を先頭に 110、120・・・と付けられます。

AUTO ,5

行番号は 10 を先頭に 15、20・・・と付けられます。

プログラム中に既に存在する行と同じ行番号が発生した場合には、行番号の後にアスタリスク（*）が表示されます。このとき、何も入力しないでリターンキーを押せば、その行は前の内容のままです。

AUTO 機能を中止したいときは、**CTRL** + **C** キーまたは **CTRL** + **STOP** キーを押します。

BASE

機能

画面出力に関連する VRAM 上のテーブルのアドレスを返します。

書式

BASE(<数式>)

文例

PRINT BASE(10)

解説

カッコ内の数値で指定された VRAM 上のテーブルのアドレスを返します。MSX1 では、VDP として TMS9918、MSX2 では V9938、MSX2+では V9958 を使っています。

参照

VDP

BEEP

機能

スピーカを鳴らします。

書式

BEEP

解説

SET BEEP 命令で設定した音色と大きさで、スピーカを鳴らします。

次のように、PRINT 命令で CHR\$(7)を実行しても、BEEP と同じ結果が得られます。

PRINT CHR\$(7)

参 照

コントロールコード表

BIN\$

機 能

<数式>を2進数の文字列に変換します。

書 式

BIN\$(<数式>)

文 例

PRINT BIN\$(50)

表示 110010

参 照

VAL、OCT\$、HEX\$

BLOAD

機 能

機械語プログラム、または画面データファイルを読み込みます (ロードする)。

書 式

BLOAD<ファイルスペック>[[,R]]|[,S]][,<オフセット>]

文 例

BLOAD "A: DEMO",R

解 説

<ファイルスペック>で指定した機械語プログラムファイルをメモリ上にロードします。カセットファイルに対してはファイル名を省略することができますが、ディスクファイルに対しては省略することはできません。<ファイルスペック>の中で<デバイス名>を省略すると現在選択されているディスクドライブ (カレントドライブ) とみなさ

れます。カセットファイルを対象とする場合は"CAS:"とします。

<オフセット>を省略すると機械語プログラムは BSAVE でセーブするときに指定した<開始アドレス>からロードしますが、<オフセット>を指定した場合には、セーブの際に指定された開始アドレスに、<オフセット>を加えた番地からロードします。したがって、<オフセット>を付けてロードするプログラムは、リロケート可能な性質をもっていないければなりません。

R オプションを指定すると、プログラムをロード後、BSAVE で指定しておいた実行開始アドレスから、ただちに実行されます。このとき、既に開かれているファイルがあればファイルは開かれたまま機械語プログラムが実行されます。

S オプションを指定すると、ディスクファイルの内容が VRAM にロードされます。<オフセット>を省略する場合にはカンマ (,) も省略してください。

参 照

BSAVE

BSAVE

機 能

機械語プログラム、または画面データを保存（セーブ）します。

書 式

BSAVE<ファイルスペック>,<開始アドレス>,<終了アドレス>[[,<実行開始アドレス>| [,S]]

文 例

BSAVE "B: DEMO",&HA100,&HA2FF

解 説

<開始アドレス>から<終了アドレス>までのメモリ上に置かれている機械語プログラムを、<ファイルスペック>で指定したファイルにセーブします。

<ファイルスペック>の中の<デバイス名>を省略した場合、カレントドライブにセーブされます。カセットファイルの場合は"CAS:"としてください。

<実行開始アドレス>を指定しておくと、BLOAD で R オプションを指定して、プログラムのロード後に自動的に<実行開始アドレス>から実行させることができます。

S オプションを指定すると、画面データをディスクファイルとしてセーブすることができます。

CALL

機 能

拡張ステートメントを呼び出します。

書 式

CALL<拡張ステートメント名>[(<引数>[, <引数>・・・])]

解 説

拡張 ROM カートリッジなどで供給される拡張ステートメントを呼び出します。CALL は_ (アンダースコア) で代用することができます。

CALL AKCNV

2+

機 能

<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。

書 式

CALL AKCNV(<文字変数>, <文字列>)

文 例

```
10 CALL AKCNV(A$, "ABC 漢字イロハ")
20 PRINT A$
```

表示 A B C 漢字イロハ

解 説

半角文字（カタカナ、英数字等）を全角の文字に変換します。もともと全角の文字はそのままです。

CALL ANK

2+

機 能

漢字ドライバを終了します。

書式

CALL ANK

文例

CALL ANK

解説

漢字モードを終了し、ANKモードとなります。ただし、漢字ドライバ用に確保されたメモリは解放されません。

CALL CLS

2+

機能

漢字モードの画面をクリアします。

書式

CALL CLS

文例

CALL CLS

解説

漢字モードでは通常のCLSは使えません。漢字モード以外でもCALL CLSは有効です。

CALL FORMAT

Disk

機能

フロッピーディスクをフォーマット（初期化）します。

書式

CALL FORMAT

文 例

CALL FORMAT

解 説

Disk BASIC や MSX-DOS で使用できるように、ディスクを初期化します。実行するとドライブ名をきいてくるので、指定します。間違って既に使用しているディスクをフォーマットしてしまうと、その中のデータは完全に失われてしまいます。十分注意してください。

CALL JIS

2+

機 能

<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。

書 式

CALL JIS(<文字変数>, <文字列>)

文 例

10 CALL JIS(A\$,"漢字")

20 PRINT A\$

表示 3441

CALL KACNV

2+

機 能

<文字列>中の半角文字に変換できる文字を全て半角文字に変換して<文字変数>に代入します。

書 式

CALL KACNV(<文字変数>, <文字列>)

文 例

10 CALL KACNV(A\$,"10人のプログラマー")

20 PRINT A\$

表示 10人ノブ' ログ' ラマー

CALL KANJI

2+

機能

漢字ドライバを起動します。

書式

CALL KANJI[n]

文例

CALL KANJI3

解説

nは、0、1、2、3のうちどれか1文字で、漢字フォントと SCREEN のインタレースモードを指定します。省略すると0と見なされます。

モード	文字の大きさ (全角 X×Y)	表示文字数 (最大横×縦)
KANJI0	16×16	32×13
KANJI1	12×16	40×13
KANJI2	16×16	32×24
KANJI3	12×16	40×24

CALL KANJI (または CALL KANJI0) はフォントを MSX の標準漢字 ROM から取って表示します。

CALL KANJI1 はフォントを MSX の標準漢字 ROM からとり、横 16 ドットを 12 ドットに圧縮して表示します。ただし、松下電器の仕様による 12 ドットフォント ROM がシステムに存在する場合はこれを表示します。

CALL KANJI2 は CALL KANJI (または CALL KANJI0) と同じですが、インタレースモードが使用され縦方向の表示文字数が増加します。

CALL KANJI3 は CALL KANJI1 と同じですが、インタレースモードが使用され縦方向の表示文字数が増加します。

MSX 標準の日本語フロントエンドプロセッサ (以下、MSX-JE) があれば、このステー

トメントの実行時またはシステムの立ち上げ時に組み込まれます。直接入力モード (ANK) と間接入力モード (漢字) は、**CTRL** + **SPACE** もしくは **GRAPH** + **SELECT** によって切り換えられます。MSX-JE が存在しないときは、単漢字変換機能が使用できます。

MSX 起動後の一番最初の CALL KANJI の実行時には注意が必要です。その時点での HIMEM (CLEAR 文による) の設定はキャンセルされ、全ての変数およびソフトウェアスタック (FOR/NEXT、GOSUB/RETURN 用) がクリアされます。これは漢字ドライバのワークエリア (もしあれば MSX-JE も) を確保する処理が行われるからです。2 回目以降の実行は問題なく行われます。

CALL KANJI 命令などを実行し、漢字表示ができる状態を「漢字モード」と呼びます。さらに漢字モードでかつ SCREEN 文で 0 もしくは 1 が設定されている状態を「漢字テキストモード」と呼び、漢字モードでかつ SCREEN 文で 2 以上の画面モードが設定されている状態を「漢字グラフィックモード」と呼びます。

BASIC のコマンド待ちの状態では漢字テキストモードになっており、漢字の入出力が可能です。漢字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。

漢字モードにおける WIDTH

■ 漢字テキストモード

CALL KANJI 実行後の WIDTH は ANK モードでの WIDTH をもとに、以下のよう
に決定されます。CALL KANJI 実行後に直接 SCREEN 文や WIDTH 文で指定する
こともできます。

1. ANK モードで SCREEN 0 を使用していた場合。

■ KANJI0 か KANJI2

ANK モードでの WIDTH の値の 4/5 が漢字テキストモードでの WIDTH になります。

■ KANJI1 か KANJI3

ANK モードでの WIDTH の値がそのまま漢字テキストモードでの WIDTH になります。

2. ANK モードで SCREEN 1 を使用していた場合。

■ KANJI0 か KANJI2

ANK モードでの WIDTH の値がそのまま漢字テキストモードでの WIDTH になります。

■ KANJI1 か KANJI3

ANK モードでの WIDTH の値の 5/4 が漢字テキストモードでの WIDTH になります。

■ 漢字グラフィックモード

常に表示できる最大に設定されます。

WIDTH による SCREEN モードの選択

■ KANJI0 か KANJI2

WIDTH が 26 から 32 の時は 256 ドットモード (VDP の GRAPHIC 4 モード) が、33 から 64 の時は 512 ドットモード (VDP の GRAPHIC 6 モード) が選択されます。

■ KANJI1 か KANJI3

WIDTH が 26 から 40 の時は 256 ドットモード (VDP の GRAPHIC 4 モード) が、41 から 80 の時は 512 ドットモード (VDP の GRAPHIC 6) が選択されます。

CALL KEXT

2+

機能

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

書式

CALL KEXT(<文字変数>, <文字列>, <機能>)

文例

```
10 CALL KEXT(A$, "今日ハ良イ天気デ ス", 0)
```

```
20 PRINT A$
```

表示 ハイデ ス

CALL KINSTR

2+

機能

漢字を含む文字列を検索します。

書式

CALL KINSTR(<数値変数> [, <数式>], <文字列 1>, <文字列 2>)

文例

```
10 CALL KINSTR(A, "A 亜 B", "B")
```

20 PRINT A

表示 3

解説

<文字列 1>の中から<文字列 2>を探しだし、見つければその位置を、見つからなければ0を<数値変数>に代入します。<数式>は探し始める位置を文字数を単位として指定するもので、省略した場合は1とみなされます。

CALL KLEN

2+

機能

漢字を含む文字列の文字数を返します。

書式

CALL KLEN(<数値変数>, <文字列> [, <機能>])

文例

```
10 CALL KLEN(A,"今日ハ良い天気です")
```

```
20 PRINT A
```

表示 9

解説

<機能>が0 (もしくは省略) のときは<文字列>の全体の文字数を、1 のときは<文字列>中の半角文字数を、2 のときは<文字列>中の全角文字数を<数値変数>に代入します。

CALL KMID

2+

機能

漢字を含む文字列の一部を取り出します。

書式

CALL KMID(<文字変数>, <文字列>, <数式 1> [, <数式 2>])

文 例

```
10 CALL KMID(A$,'今日はヨい天気です',3,3)
20 PRINT A$
```

表示 はヨい

解 説

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

CALL KNJ

2+

機 能

漢字コードに相当する漢字を獲得します。

書 式

CALL KNJ(<文字変数>, <文字列>)

文 例

```
10 CALL KNJ(A$,"3441")
20 PRINT A$
```

表示 漢

解 説

<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8000H未満であればJIS、それ以上の場合はシフトJISとみなします。

CALL KTYPE

2+

機 能

文字のタイプを調べます。

書 式

CALL KTYPE(<数値変数>, <文字列>, <数式>)

文 例

```
10 CALL KTYPE(T,'コレハ漢字',4)
20 PRINT T
```

表示 1

解 説

<文字列>中の<数式>番目の文字のタイプを<数値変数>に代入します。タイプは、0=半角、1=全角です。

CALL MEMINI

2**機 能**

メモリディスクで使うメモリの上限を指定します。

書 式

CALL MEMINI(<メモリディスクの上限>)

文 例

```
CALL MEMINI(&H7000)
```

表示 27904 bytes allocated

解 説

範囲は、0~&H7FFF です。初期状態では、&H7FFF が指定されています。

CALL MEMINI 命令が実行されると、メモリディスクに割り当てられたメモリの量が表示されます。

CALL MFILES

2**機 能**

メモリディスク中のファイル名を表示します。

書式

CALL MFILES

文例

CALL MFILES

CALL MKILL

2

機能

メモリディスク中のファイルを削除します。

書式

CALL MKILL(<ファイル名>)

文例

CALL MKILL("SAMPLE")

CALL MNAME

2

機能

メモリディスク中のファイル名を付け替えます。

書式

CALL MNAME(<新ファイル名>AS<旧ファイル名>)

文例

CALL MNAME("TEST" AS "TEST.AA")

CALL PALETTE

2+

機能

カラーパレットの初期化または設定を行ないます。

書式

CALL PALETTE[(**<パレット番号>**,**<赤輝度>**,**<緑輝度>**,**<青輝度>**)]

文例

CALL PALETTE(15,4,4,4)

解説

CALL KANJI ステートメントにより画面を漢字テキスト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。

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、**<パレット番号>**で指定されたパレットを**<赤輝度>**、**<緑輝度>**、**<青輝度>**の色に設定します。なお、パレット番号や各輝度を省略することはできません。

CALL SJIS

2+**機能**

<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。

書式

CALL SJIS(**<文字変数>**,**<文字列>**)

文例

10 CALL SJIS(A\$,"漢字")

20 PRINT A\$

表示 3441

CALL SYSTEM

Disk**機能**

MSX-DOS を起動します。

書式

CALL SYSTEM

文例

CALL SYSTEM

解説

SYSTEM コマンドを実行すると、DOS に戻る前にすべてのファイルは閉じられ、メモリ中のプログラムやデータは失われます。

この命令が使用できるのは、MSX-DOS から Disk BASIC を起動した場合だけです。そうでない場合は、「Illegal function call」エラーになります。

CDBL

機能

<数式>の値を倍精度実数に変換します。

書式

CDBL(<数式>)

文例

PRINT CDBL(1/3)

表示 .3333333333333333

CHR\$

機能

<数式>のキャラクタコードを持つ文字を返します。

書式

CHR\$(<数式>)

文例

PRINT CHR\$(65)

表示 A

数値は 0～255 の範囲でなければなりません。文字とキャラクタコードの対応については、キャラクタコード表を参照してください。

参照

ASC、キャラクタコード表

CINT

機能

<数式>の小数点以下を切り捨てて、実数に変換します。

書式

CINT(<数式>)

文例

```
PRINT CINT(123.45678)
```

表示 123

解説

<数式>の値が-32768～32767の範囲外のときは、「Overflow」エラーになります。

CIRCLE

機能

グラフィック画面に円や楕円を描きます。

書式

CIRCLE[STEP](X,Y),<半径>[,<カラーコード>][,<開始角度>][,<終了角度>][,<比率>]

文例

```
CIRCLE(128,96),50,1
```

解 説

SCREEN 2以上の画面で円を描きます。中心座標は、X軸方向に255 (SCREEN 6、7では511)、Y軸方向に191 (SCREEN 5以降では211) ありますので、その範囲内の数値を指定できます。また、STEPを用いて中心を指定することもできます (STEPを用いた指定の方法についてはLINE命令を参照)。

色を指定すると、その色で円が描かれます。色の指定を省略すると、COLORで指定された前景色が用いられます。

■ 弧と扇形

`CIRCLE(128,96),50,1,3/4*3.14,1/4*3.14`

のように、円の開始角度 ($3/4 * 3.14$) と終了角度 ($1/4 * 3.14$) を指定すれば、その範囲の弧が描かれます。それぞれにマイナス記号をつけると、扇形が描かれます。開始角度、終了角度の指定は、 -2π から 2π の範囲内で行います。

■ 楕円

`CIRCLE(128,96),50,1,,0.8`

のように、最後に比率 (0.8) を指定すれば、楕円が描かれます。比率は「垂直方向の半径 / 水平方向の半径」の値で指定します。値が1より小さいと横に平べったい楕円となり、1より大きいと縦に細長い楕円となります。

参 照

SCREEN、COLOR

CLEAR

機 能

変数の初期化と使用するメモリ領域の大きさを指定します。

書 式

`CLEAR[<文字列領域の大きさ>[,<メモリの上限>]]`

文 例

`CLEAR 200,&HD000`

解 説

CLEARを実行すると、すべての変数が初期化され、DEF命令 (DEF FN、DEFINT / SNG / DBL / STR) の定義は無効になります。初期化されると、数値変数は0に、文字変数はヌル (何も文字が代入されていない状態) になります。

また、

```
CLEAR 300,&HD000
```

のような指定では、変数の初期化のほかに、文字変数の内容をしまっておく文字列領域の大きさ（この場合、300）と、BASIC が使用するメモリの上限（この場合、&HD000）を設定することができます。文字列領域の大きさは、メモリの許す限りの値まで指定できます。省略したときは 200 になります。また、メモリの上限は &H831F～&HF380 の範囲内で指定できます。しかし、Disk BASIC などのワークエリアを壊さないように注意しなければなりません。メモリの上限を設定するときは、文字列領域の大きさの指定を省略できません。メモリの上限については、「4.2 ユーザーエリアの詳細」を参照して下さい。

参 照

FRE

CLOAD

機 能

カセットテープのプログラムをロードします。

書 式

```
CLOAD[<ファイルスペック>]
```

文 例

```
CLOAD "TEST"
```

解 説

カセットテープに CSAVE でセーブされたプログラムをロードします。

CLOAD が実行されると、<ファイルスペック>で指定されたファイル名のプログラムをカセットテープ上で探します。探している最中に別のプログラムを見つけると「Skip: ファイル名」と画面に表示され、指定されたプログラムが見つかったら「Found: ファイル名」と表示してロードを開始します。

CLOAD の後ろに指定するファイル名を省略することもできます。省略したときは、CLOAD が実行されて最初にみつけたプログラム (CSAVE でセーブされたもの) をロードします。CLOAD 命令を実行すると、そのときメモリ上にあったプログラムは無くなってしまいますので注意してください。また、CLOAD で、SAVE 命令によってセーブされた

プログラムをロードすることはできません。

参 照

CSAVE、LOAD

CLOAD ?

機 能

カセットテープのプログラムとメモリ上のプログラムを比較します。

書 式

CLOAD?[<ファイルスペック>]

文 例

CLOAD? "TEST"

解 説

メモリ上にあるプログラム (LIST 命令で画面に表示されるもの) と、カセットテープにセーブされたプログラムが同じかどうか調べます。同じであれば「Ok」と表示され、違っていると「Verify error」と表示されます。CLOAD?命令は、普通 CSAVE 命令を実行した後に、セーブが正しく行われたかどうかを確認するために使われます。

CLOAD?の後ろに指定するファイル名を省略することもできます。省略したときは、CLOAD?が実行されて最初にみつけたプログラム (CSAVE でセーブされたもの) を比較します。

参 照

CSAVE、LOAD

CLOSE

機 能

ファイルを閉じます。

書 式

CLOSE [[#]<ファイル番号>[, [#]<ファイル番号>]…]

文 例

CLOSE # 1

解 説

<ファイル番号>に対応するファイルを閉じます。指定した<ファイル番号>は、以後の OPEN 文で指定することができるようになります。また、いったん閉じたファイルをもう一度開くときには、他のファイルが使っていない<ファイル番号>ならば、どの<ファイル番号>でも指定することができます。

CLOSE では、<ファイル番号>を複数指定することにより、一度に複数のファイルを閉じることができます。また<ファイル番号>を省略すると、そのとき開いているファイルをすべて閉じます。

閉じたファイルに対しては、再び OPEN 文で開くまで入出力を行うことはできません。CLOSE 文は、ファイルが出力用にオープンされていた場合には、バッファに残っていたデータの掃き出しを行うので、ファイルへの出力処理を正しく終了するためには、必ず CLOSE 文を実行しなくてはなりません。

END 文、NEW 文を実行すると、自動的にすべてのファイルが閉じられますが、STOP 文はファイルを閉じないので注意する必要があります。

参 照

OPEN、END、NEW

CLS

機 能

画面に表示されている文字や絵を消します。

書式

CLS

文例

CLS

解説

CLS は、KEY ON 指定で表示されているファンクションキーの内容を除いて、文字、点、線、絵など画面に表示されているものをすべて消します。

また、CLS が実行されると、テキスト画面の場合、カーソルは画面一番上の左端に移動しますがグラフィック画面では LP (最終参照点) は変化しません。

CLS は漢字モードでは使えません (Illegal function call になる)。漢字モードで画面を消すときは CALL CLS 命令を使います。

参照

COLOR、CALL CLS

COLOR

機能

画面の色を指定します。書式 1 以外は全て MSX2 以降専用です。

書式 1

COLOR[<前景色>][,<背景色>][,<周辺色>]

書式 2

COLOR=(<パレット番号>,<Rコード>,<Gコード>,<Bコード>)

書式 3

COLOR = RESTORE

書式 4

COLOR[= NEW]

文例 1

COLOR 1,5,8

解説 1

色は、それぞれ以下の 0~15 (SCREEN 6 では 0~3) までの値で指定します。

0 透明 (周辺色と同じ色)	8 赤
1 黒	9 明るい赤
2 緑	10 黄
3 明るい緑	11 明るい黄
4 暗い青	12 暗い緑
5 明るい青	13 紫
6 暗い赤	14 灰
7 水色	15 白

電源投入時には、COLOR は「COLOR 15,4,7」に設定され、SCREEN 文を実行したときには、カラーパレットは上記のように設定されています。

SCREEN 8 以外では、書式 2 の COLOR = を使ってこれらの色を 512 色の中から選んで自由に設定することができます。

なお、プログラム中でグラフィック画面を指定すると、それまでテキストモードで使われていた色がそのまま使われます。

SCREEN 8 の色は、0~255 が使えます。

00000000 8ビットの色コード
GGGRRRBB Green、Red、Blue の各ビット

例えば、明るい緑は&B11100000 (= 224)、暗い紫なら&B00000101 (= 5) などとなります。

SCREEN 10 では 0~15、SCREEN 11、12 では 0~255 が使えます。SCREEN 10 では YJK/RGB 混在モードの RGB 部分のみの指定なので、SCREEN 7 以下のカラーパレットの指定と全く同じと考えることができます。SCREEN 11 以降ではカラーコードは VRAM の値と考えることができますが、YJK 方式の性格上これを単純に解釈することはできません。詳しくは V9958 の YJK モードに関する記述を参照してください。

■ 前景色

テキスト画面 (SCREEN 0 および 1 のテキストモード) の文字の色や、グラフィック画面の点や線の色です。ただし、グラフィック画面で点や線などを描くときには、各命令で色の指定を行うことができます。各命令で色を指定すると、その指定の方が優先されます。前景色だけを変えるには、例えば、次のように指定します。

COLOR 9

■背景色

テキスト画面やグラフィック画面の地の色のことです。テキスト画面では、命令実行直後に指定した背景色に変えられます。グラフィック画面では、SCREEN か CLS を実行した後に背景色が変わります。背景色だけを変えるには、例えば、次のように指定します。

COLOR ,4

■周辺色

画面上下の BASIC で使えない部分のことです。周辺色だけを変えるには、例えば、次のように指定します。

COLOR ,,13

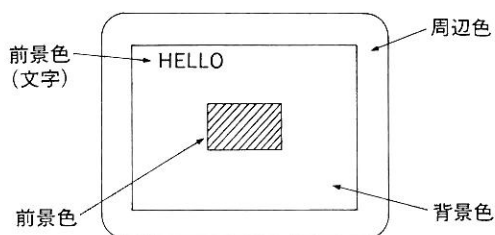


図 2.10 画面色

文 例

```
10 SCREEN 5
20 COLOR = (0,0,7,7)
```

解 説

SCREEN 6 では 0~3、SCREEN 8 以外では 0~15 のパレットに色を割り当てます。上の例ではパレット 0 に 0、7、7、つまり

```
&B000 &B111 &B111
```

RRR GGG BBB (RGB 各 3 ビット、0~7 の値)

の結果として明るい水色を割り当てています。割り当てた後は、書式 1 にしたがって色を指定します。

初期状態では、書式 1 で説明したカラー番号とパレット番号が同じになるように設定してあります。

文 例

```
COLOR = RESTORE
```

解 説

VRAM上のパレットテーブルの情報にしたがってパレットを設定します。

文 例

```
COLOR = NEW
```

解 説

パレットを初期状態にセットします。= NEW は省略してもかまいません。

参 照

CALL KANJI、CLS、SCREEN、VRAM マップ、V9958

COLOR SPRITE

2

機 能

スプライトの1ライン毎の色を決めます。

書 式 1

```
COLOR SPRITES(<プレーン番号>)=<文字式>
```

文 例 1

```
COLOR SPRITE$(0)=CHR$(1)+CHR$(7)
```

解 説

スプライトの色も実際には、パレット番号で指定します。この例ではプレーン0の1ライン目を黒に、2ライン目を水色にしています。スプライトの1ライン毎に色を付けられるのは、SCREEN モード4以降のときだけです。色コードについてはCOLORを参照してください。なお色コードが15を越える場合、上位4ビットは次のような意味を持ちます。

- bit7 32ドット左シフト表示。
- bit6 PUT SPRITE を実行したとき連続したプレーンを同時に動かす。このとき重なった部分の色コードはORされて表示され、衝突は検出されない*。

- bit5 衝突を検出しない。
- bit4 未使用。

*bit6 が ON のときは表示が複雑になります。詳しくは「第4部 VDP」を参照して下さい。

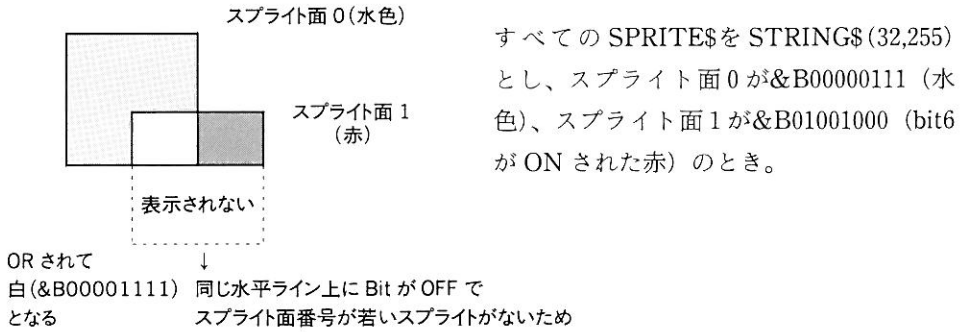


図 2.11 bit6 が ON の時のスプライト

bit6 が ON のスプライト面が連続していると、1つの PUT SPRITE 命令でそれらを動かすことができます。

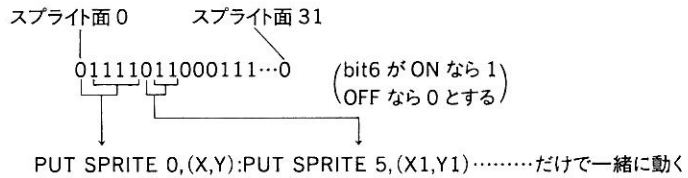


図 2.12 bit6 が ON の時の PUT SPRITE 命令

SCREEN 8 のスプライトだけは、次のような 16 色に固定されます。

表 2.7 SCREEN 8 のスプライト色

カラー番号	0	1	2	3	4	5	6	7
色	黒	暗い青	暗い赤	暗い紫	暗い緑	暗い水色	暗い黄色	灰色
カラー番号	8	9	10	11	12	13	14	15
色	肌色	青	赤	紫	緑	水色	黄	白

書式 2

COLOR SPRITE(<プレーン番号>)=<式>

文例 2

```
COLOR SPRITE(1)= 15
```

プレーン 1 全体を白にします。

参照

COLOR、SPRITE

CONT

機能

停止したプログラムの実行を再開します。

書式

```
CONT
```

文例

```
CONT
```

解説

プログラムの実行停止後、「CONT」と入力すると、停止した次の文から実行が再開されます。プログラムの実行停止は、**CTRL** + **STOP** キー入力やプログラム文中の STOP 文や END 文によるものでなければなりません。また INPUT\$関数のキー入力待ちを **CTRL** + **STOP** キーで止めた場合は、CONT はできません。

実行停止中に LIST 命令や PRINT 命令などを実行することはできますが、プログラムそのものの変更を行ったときは、CONT による実行の再開はできません。

参照

STOP、END

COPY

2 **Disk**

機能

ファイルや VRAM の内容、配列をコピーします。書式 1 以外は全て SCREEN モードが 5 以降で有効なので MSX2 以降専用です。パラメータに <ファイルスペック> を含む書

式は全て Disk BASIC でのみ有効です。論理演算子としては、AND、OR、XOR、PSET、PRESET が指定可能です。

書式 1

COPY <ファイルスペック 1> [TO <ファイルスペック 2>]

書式 2

COPY (<ソース座標 1>)-(<ソース座標 2>)[, <ソースページ>] TO (<デスティネーション座標>)[, <デスティネーションページ>][, <論理演算子>]

書式 3

COPY (<ソース座標 1>)-(<ソース座標 2>)[, <ソースページ>] TO <ファイルスペックまたは配列名>

書式 4

COPY <ファイルスペックまたは配列名>[, <方向>] TO (<デスティネーション座標>)[, <デスティネーションページ>][, <論理演算子>]

書式 5

COPY <配列名> TO <ファイルスペック>

書式 6

COPY<ファイルスペック> TO <配列名>

書式 7

COPY SCREEN [<モード>]

文例 1

COPY 'A:SAMPLE.DAT' TO 'SAMPLE.BAK'

解説 1

<ファイルスペック 1>を<ファイルスペック 2>で指定したドライブへ、指定したファイル名でコピーします。この COPY コマンドはディスク装置間のみで有効です。

指定の方法は4種類あります。

- <ファイルスペック 1>だけを指定した場合

COPY 'B:SAMPLE.DAT'

この場合、SAMPLE.DAT は、カレントドライブに同じ名前(SAMPLE.DAT)でコピーされます。

- <ファイルスペック 2>でドライブ名だけを指定した場合
COPY "A:SAMPLE.DAT" TO "B:"
この場合、SAMPLE.DAT は、ドライブ B に同じ名前で作成されます。
- <ファイルスペック 2>でファイル名だけを指定した場合
COPY "B:SAMPLE.DAT" TO "SAMPLE.BAK"
この場合、SAMPLE.DAT は、カレントドライブに SAMPLE.BAK という名前で作成されます。
- <ファイルスペック 2> でドライブ名とファイル名とを指定した場合
COPY "A:SAMPLE.DAT" TO "B:SAMPLE.BAK"
この場合、SAMPLE.DAT は、ドライブ B に SAMPLE.BAK という名前で作成されます。

文例 2

```
10 SCREEN 5
20 COPY(10,10)-(100,120),1 TO (30,30),0,XOR
```

解説 2

VRAM 上の長方形の領域を他の領域にコピーします。ページを省略した場合はそれぞれアクティブページとみなされます。論理演算子を省略した場合は PSET とみなされません。

COPY 命令で画面範囲外にコピーした場合、スプライトが表示されてしまうことがあります。例えば、

```
COPY(0,0)-(255,39) TO (0,211)
```

を実行すると画面データをスプライトなどのデータエリアに書いてしまうので、スプライトのゴミが表示されてしまいます。VDP (V9938、V9958) はハードウェアスクロールの機能を持っており、COPY 文では、その機能を十分生かすために画面表示範囲外に画面データをコピーできるようになっています。したがって、画面表示範囲外に画面データをコピーするときには、スプライトを VDP 命令で消して下さい。VDP レジスタの R #8 の bit 1 を 1 にすると、スプライトの表示が禁止されます。具体的には、以下の命令を実行します。

```
VDP(9)=VDP(9) OR 2
```

文例 3

```
10 SCREEN 8
20 DIM B%(327)
30 COPY(100,100)-(120,130),0 TO B%
```

解説 3

VRAM上の長方形の領域の内容を配列やファイルにコピーします。配列の場合は必要な大きさ以上のものを確保しておかなければなりません。必要なバイト数Nは次のように計算されます。

$$N = \text{INT}(X * Y / K + 5)$$

ただしXは横、Yは縦のドット数、Kは定数で、

SCREEN 5と7では2

SCREEN 6では4

SCREEN 8以降では1

として求め、次に配列の型によって

整数型 (N+1)¥2

単精度型 (N+3)¥4

倍精度型 (N+7)¥8

として最低限必要な要素数が得られます。文例3のケースでは

$$N = \text{INT}((120-100+1) * (130-100+1) / 1 + 5); N = 656$$

$$(N+1)¥2 = 328$$

からDIM B%(327)となります。

文例 4

10 SCREEN 7

20 COPY "TEST",1 TO (100,100),1,AND

解説 4

ファイルまたは配列の内容を画面に転送します。方向はデータを書き込む向きで、

- 0 左上から右下
- 1 右上から左下
- 2 左下から右上
- 3 右下から左上

のどれかです。

文例 5

```
10 DIM A%(100)
20 COPY A% TO "TEST"
```

解説 5

配列の内容をファイルにセーブします。

文例 6

```
10 DIM A%(100)
20 COPY "TEST" TO A%
```

解説 6

ファイルから配列に内容をコピーしてきます。ファイルはもともと書式3や書式5で作ったものでなければならず、配列は十分な大きさでなければなりません。

文例 7

```
10 SCREEN 8
20 COPY SCREEN
```

解説 7

外部ビデオ信号をデジタル化し、VRAM に送ります。〈モード〉は0なら通常で、1ならインタレースモードです。この機能は画像取り込み装置がつながっている場合にのみ有効です。どのスクリーンでどのモードで取り込みができるかはその取り込み装置の性能に依存します。

参照

SET PAGE、LINE

COS

機能

〈数式〉の余弦（コサイン）を計算します。

書式

COS(〈数式〉)

文 例

```
PRINT COS(3.14159)
```

表示 -.999999999999649

CSAVE

機 能

メモリ上にあるプログラムをカセットテープにセーブします。

書 式

CSAVE<ファイル名>[, <ボーレート>]

文 例

```
CSAVE "TEST"
```

解 説

メモリ上にあるプログラム (LIST 命令で画面に表示されるもの) を、カセットテープに保存します。

ファイル名の後に次のように指定すれば、セーブするときの速さを指定することができます。指定できるのは1か2です。

```
CSAVE "TEST",1
```

2と指定すると1の倍の速さでセーブされますが、カセットレコーダの性能によってはうまくセーブできないことがあります。このセーブの速さは、SCREEN 命令でも指定でき、CSAVE で省略されたときには、SCREEN 命令の指定が使われます。

うまくセーブされたかどうかは、CLOAD?命令で確認してください。セーブされたプログラムは、CLOAD 命令を使ってロードすることができます。

CSAVE と SAVE との違いについては、SAVE 命令を参照してください。

参 照

CLOAD、CLOAD?、SAVE

CSNG

機能

<数式>の値を単精度実数に変換します。

書式

CSNG(<数式>)

文例

```
PRINT CSNG(123.45678)
```

表示 123.456

CSRLIN

機能

テキスト画面上のカーソルの行位置を返します。

書式

CSRLIN

文例

```
A = CSRLIN
```

解説

CSRLIN は、現在カーソルが何行目にあるのかを、一番上の行を 0 とした値で調べます。

CSRLIN は、テキスト画面でのみ使うことができます。

参照

POS

CVI、CVS、CVD

Disk

機能

文字列を数値データに変換した値を返します。

書式

CVI(<2バイト文字列>)

CVS(<4バイト文字列>)

CVD(<8バイト文字列>)

文例

```
PRINT CVI(A$)
```

解説

ディスク上のランダムファイルから読み込んだ数値データは文字型になっているため、これらの関数を使って数値型に変換しなければなりません。CVIは2文字(2バイト=16ビット)の文字列を整数値に、CVSは4文字の文字列を単精度実数値に、CVDは8文字の文字列を倍精度実数値にそれぞれ変換します。

この変換は、例えば、CVIの場合、次のように行われます。

A\$="90"の場合、すなわち A\$=CHR\$(&H39)+CHR\$(&H30)の場合には、CVI(A\$)の値は、キャラクタコードを数値データとみなし、&H3039 (=12345) となります。

CVS、CVDの場合は、各キャラクタコードを正規化された浮動小数点形式の数値データとみなします。

DATA

機能

READ文で読み込むためのデータをプログラム中に用意します。

書式

```
DATA<定数>[, <定数>・・・]
```

文例

```
10 READ A,B  
20 DATA 1,2
```

解説

<定数>は文字型か数値型で、READ 文でそのデータを読み込む変数の型と一致していなければなりません。この例では数値型の変数 A および B に、それぞれ 1 と 2 という数値が代入されます。

DATA 文中のデータは、1 行に収まる範囲内でいくつでもカンマで区切って記述することができます。DATA 文中のコロン(:)は引用符に囲まれていない限りマルチステートメントの区切りと解釈されます。DATA 文は、READ 文によって読み込まれない限り何の働きもしません。

参照

READ、RESTORE

DEF FN

機能

計算式を定義します。

書式

<名前> [(<引数> [, <引数> . . .])] = <関数の定義式>

文例

```
10 DEF FN A(X)=X*X*3.14
20 FOR I= 10 TO 20 STEP 5
30 PRINT FN A(I)
40 NEXT
```

表示 314
706.5
1256

解説

10 行で円の面積を出す計算式を FN A(X)として定義し、その式を 30 行で I という変数を使って値を変えながら呼び出しています。

DEF FN は、名前 (上の場合 A) で、計算式を定義しておくことができます。名前は、英字で始まる 2 文字以内の英数字で付けます。名前の次にカッコで囲んだ変数 (上の場合 X) を指定して、計算式 (上の場合 X*X*3.14) を定義します。カッコ内の変数と計算式の中の変数は対応しています。

定義した計算式は「FN 名前 (変数)」という形で呼び出せます。定義を行ったときの変数は仮のものなので、呼び出すときにはカッコ内でそのときどきで必要な変数（上の場合 I）に変えて指定することができます。

DEFINT / SNG / DBL / STR

機能

変数の型を宣言します。

書式

```
DEFINT <文字の範囲> [, <文字の範囲> . . . ]
DEFSNG <文字の範囲> [, <文字の範囲> . . . ]
DEFDBL <文字の範囲> [, <文字の範囲> . . . ]
DEFSTR <文字の範囲> [, <文字の範囲> . . . ]
```

文例

```
10 DEFINT A,B
20 DEFSTR C-E
30 A = 3.12:B = 0.33
40 C = '3.12':D = '0.33':E = 'ABC'
50 PRINT A,B
60 PRINT C,D,E
表示 3      0
      3.12  0.33  ABC
```

解説

この例では変数 A、B を整数型に、変数 C、D、E を文字型に宣言しています。変数の指定は、英字 1 字 (例 A) あるいは英字 1 字-英字 1 字 (例 A-Z) という形で行います。型は以下のように設定されます。

```
DEFINT    整数型
DEFSNG    単精度実数型
DEFDBL    倍精度実数型
DEFSTR    文字型
```

型宣言が行われていない変数は、すべて倍精度実数とみなされます。また、DEF 命令での型宣言よりも、型宣言文字での指定の方が優先されます。

参 照

CLEAR

DEF USR

機 能

機械語プログラムの実行開始番地を指定します。

書 式

```
DEF USR[<番号>]=<開始番地>
```

文 例

```
DEF USR2 =&HE000
```

解 説

USR を使って呼び出す機械語プログラムの、実行開始番地 (&HE000) を指定します。
DEF USR の次には、複数の機械語プログラムを使うときに区別するための番号 (2) を付けます。指定できるのは、0 から 9 までの値です。

参 照

CLEAR、USR、「5 章 マシン語とのリンク」

DELETE

機 能

指定されたプログラム行を削除します。

書 式

```
DELETE<行番号の範囲>
```

文 例

```
DELETE 50-100
```

解説

行番号の指定の方法によって、さまざまな形の削除が行えます。以下に、その例を示します。

DELETE 10

行番号 10 の行を削除します。

DELETE -100

プログラムの先頭から行番号 100 までの行を削除します。

参照

LIST、MERGE、NEW、RENUM

DIM

機能

配列変数を使うことを宣言し、それをメモリ領域に割り当てます。

書式

DIM <変数名> (<添字の最大値> [, <添字の最大値> . . .])

文例

DIM A(11),B%(100),C\$(2,3)

解説

DIM は、使用する配列変数の最大値を設定し、同時にメモリ上にその配列の領域を確保します。設定できる最小の値は 0、最大の値はメモリの容量が許す範囲です。設定を行わない場合は、値は 10（要素数は添え字 0～10 で 11 個）とみなされます。

不用になった配列変数は、ERASE 命令で削除できます。また、宣言してある配列変数は、CLEAR 命令か ERASE 命令を実行しないと再宣言できません。

参照

ERASE、CLEAR

DRAW

機能

グラフィック画面に図形を描きます。

書式

DRAW <文字式>

文例

```
10 SCREEN 2
20 A$="A0C15S4BM150,80L40D12R8U6R24D6R8U12"
30 B$="BM134,86F12D8L32U8E12"
40 C$="C8BM132,92L4G4D4F4R4E4U4H4"
50 DRAW "XA$;":DRAW "XB$;"
60 DRAW XC$
70 GOTO 70
```

表示 

解説

各文字変数に代入された文字式が、50 行の指定で実行されて、画面に電話器が描かれます。

DRAW での指定は次のような形で行います。

```
DRAW "文字式"
DRAW 文字変数
DRAW "X 文字変数;"
```

文字式は、線を描くための情報で記号と数値でできていて、DRAW に続けて引用符で囲んで指定します。文字変数を使って指定するときは、あらかじめその文字変数に、線を描くための情報 (文字式) を代入しておきます。文字変数を用いるときには、"X 文字変数;"と指定しても (50 行)、単に文字変数だけ指定しても (60 行)、結果は同じです。指定する記号には、移動、回転、色、サイズの 4 種類のものがあります。

表 2.8 DRAW 文の指定一覧

	記号	数値	意味	例	備考
移動命令	U	距離	上へ移動	U10	距離は画面の1点を単位とする(単位はサイズ命令で変更可能)斜め方向と上下左右方向で同じ距離を指定しても、画面上での実際の移動距離は異なる。
	D	〃	下 〃	D5	
	L	〃	左 〃	L8	
	R	〃	右 〃	R12	
	E	〃	右上 〃	E7	
	F	〃	右下 〃	F20	
	G	〃	左下 〃	G15	
	H	〃	左上 〃	H40	
	M	座標 (X,Y)	(X,Y)まで描画	M100,100	Xの前に+か-をつけると相対座標指定
	B	なし	移動のみ	BM50,50	U~Mまでの記号に付けて用いる。
	N	なし	LP 移動しない	NL10	U~Mまでの記号に付けて用いる。
回転命令	A	0~3	90度単位で回転	A1U12L4D12	0~270度の回転。指定し直すまで有効。
色命令	C	0~255	色指定	C8D10	色コードの範囲は画面モードによる。
サイズ命令	S	1~255	移動距離の単位	S20U12L4D12	省略値は4で、これが画面1ドット分。

■ 数値変数の指定

なお、次の形式で数値変数も指定できます。

DRAW "=変数名;"

例えば、

10 SCREEN 2

20 A = 100:B = 120

30 DRAW "BM = A,;= B;U20"

40 GOTO 40

というプログラムは(100,120)の位置まで線を引かずに移動し、そこから上に20点分の線を引くというものです。

参照

COLOR

DSKF

Disk

機能

フロッピーディスクの残り容量をクラスタ単位で返します。

書式

DSKF (<ドライブ番号>)

文例

PRINT DSKF(1)

解説

<ドライブ番号>で指定されたフロッピーディスクの残り容量をクラスタ単位で返します。1クラスタは以下のとおりです。

メディア	サイズ
フロッピーディスク	1024 バイト
RAM ディスク*	512 バイト (2M バイトまで) 1024 バイト (2M~4M バイト)
ハードディスク	可変 [ハードディスクの初期化 (領域確保) のとき、指定する領域の大きさにより 1K バイト、2K バイト、4K バイト、8K バイト、16K バイト、32K バイト、64K バイトから選択する。]

* MSX-DOS2 の RAMDISK コマンドで設定するもの。

END

機能

プログラムを終了します。

書 式

END

文 例

```
10 PRINT "ハルカ"
20 END
30 PRINT "コナイ"
40 END
```

解 説

END 命令はプログラムを終了させます。プログラム内にいくつ置いておかまいません。また、プログラムの最後の END（この場合、40 行）は省略してもかまいません。上の例では、20 行に END があるので、そのあとの行は実行されず、「コナイ」は表示されません。

EOF

機 能

ファイルが終了したかどうかを調べます。

書 式

EOF (<ファイル番号>)

文 例

```
IF EOF(1) THEN CLOSE1
```

解 説

EOF は、ファイル番号で指定されたファイルが終ったかどうか調べるもので、終わっていれば -1 を、まだ途中であれば 0 を結果として返します。EOF に続くカッコ内の数値は、指定したファイルのファイル番号で、OPEN 命令で指定されているものと対応しています。調べるファイルは、OPEN 命令の INPUT モードで開かれていなければなりません。

参 照

OPEN、INPUT #

ERASE

機能

配列変数を削除します。

書式

ERASE <配列変数名> [, <配列変数名> . . .]

文例

ERASE A

解説

DIM で指定した配列変数を削除し、割り当てられていたメモリを使えるようにします。

参照

DIM

ERL、ERR

機能

エラーコードやエラーの起こった行番号を調べます。

書式

ERR

ERL

文例

```
10 ON ERROR GOTO 40
20 A%= A%+10
30 GOTO 20
40 IF ERR <> 6 THEN 80
50 PRINT ERL ; "キ ョウデ エラーハッセイ"
60 A%= 0:PRINT "ショリシマシタ"
70 RESUME
80 ON ERROR GOTO 0
```

解説

40行で発生したエラーコードが6かどうかの判定を行い、6なら50行でエラーの発生した行番号を表示し、60行でエラーに対する処理を行っています。

このように、ERLやERRを用いると、エラーが発生したとき、エラーの起こった行番号(ERL)やエラーコード(ERR)を知ることができます。

ERLやERRは普通、ON ERROR GOTO命令でジャンプするエラー処理の操作で使われます。ダイレクトモードでエラーが発生したときには、ERLの値は65535となります。

参照

ON ERROR GOTO、ERROR

ERROR

機能

プログラムをエラー状態にします。

書式

ERROR<エラーコード>

文例

ERROR 23

解説

ERRORで0から255までの値を指定することによって、プログラムをエラー状態にすることができます。メッセージが用意されているエラーコードを指定すると、画面に対応するエラーメッセージが表示されます。

エラーコード23、26~49、60~255にはメッセージが定義されていないので、ON ERROR GOTO文と併せて独自のエラー処理を行うこともできます。

参照

ON ERROR GOTO、エラーメッセージ表

EXP

機能

自然対数の底「e」の<数式>乗を計算します。

書式

EXP(<数式>)

文例

PRINT EXP(1)

表示 2.7182818284588

FIELD

Disk

機能

ランダム入出力バッファ中に変数領域を割り当てます。

書式

FIELD[#]<ファイル番号>, <フィールド幅>AS<文字変数>[, <フィールド幅>
AS<文字変数>] . . .

文例

FIELD # 1, 15 AS A\$, 10 AS B\$

解説

<ファイル番号>とは OPEN 文で開いたファイルにつけた番号のことで<フィールド幅>とは<文字変数>に割り当てられる文字数のことです。また<文字変数>はランダムファイルをアクセスするとき使われる変数名です。

FIELD 文は、GET 文によるランダム入出力用バッファへのデータの読み込み、あるいは PUT 文によるランダム入出力用バッファからのデータの書き出しを可能にするために GET、PUT 文の前に実行されなければなりません。

例えば、

FIELD1, 15 AS A\$, 10 AS B\$

では、ランダム入出力用バッファ中の最初の 15 文字 (バイト) を A\$という文字変数に

割り当て、次の10文字がB\$に割り当てられます。

FIELD文は、実際にはランダム入出力用バッファにデータを置くことはせず、これはLSET、RSET文で行われます。

FIELD文で割り当てられる総バイト数は、OPEN時に決めたレコード長を越えることはできません。万一、越えた場合には、「Field overflow」エラーが生じます。

なお、1つのファイル番号に対して、何回かに分けてFIELD文を実行してもかまいません。このとき、以前に実行されたFIELD文も、その効力を保持します。FIELD文は、実行するたびに最初の1文字からバッファを定義するため、同一データに対し複数の領域を定義する効果をもたらします。

FIELD文で定義した文字変数は、ランダム入出力用バッファ中の対応するデータを指していますが、FIELD文の実行後に入力関係の文、または代入文の左辺で使用すると、その変数は文字列の記憶領域に移り、ランダム入出力用バッファからは消えてしまうので注意する必要があります。

FILES、LFILES

Disk

機能

フロッピーディスクの中のファイル名を表示します。

書式

[L]FILES [<ファイルスペック>]

文例

```
FILES '* .BAS'
```

```
LFILES 'B:'
```

解説

<ファイルスペック>で指定されたファイルのファイル名を表示します。指定ファイルがフロッピーディスク中になければ、「File not found」エラーが生じます。

<ファイルスペック>が省略された場合は、現在選択されているディスクドライブ中の全ファイル名を表示します。

指定するファイル名には、疑問符(?)を含むことができ、ファイル名や、拡張子中の1文字の代わりをすることができます。また、ファイル名、拡張子の文字中にアスタリスク(*)を用いると、任意のファイル名、または拡張子の代わりとすることができます。<ファイルスペック>で、ディスクドライブが指定されていれば、そのディスクドライブ中の指定ファイルが出力され、そうでないときは、現在選択されているディスクドラ

イブ（カレントドライブ）が使用されます。
LFILES は出力先がプリンタであることを除き、FILES とまったく同じです。

FIX

機能

<数式>の値の小数点以下を取り去った値を計算します。

書式

FIX(<数式>)

文例

```
PRINT FIX(-1.2)
```

表示 -1

解説

<数式>の値が正のときは INT と同じですが、負のときは以下ようになります。

式	結果	意味
FIX (-1.2)	-1	小数点以下切り捨て
INT (-1.2)	-2	-2.3 を超えてない最大の整数 <small>1.2</small>

FOR~NEXT

機能

FOR から NEXT までの間で指定された操作を、指定された回数だけ繰り返します。

書式

FOR<変数名>=<初期値>TO<終値>[STEP<増分>]

文例

```
10 FOR I = 0 TO 10 STEP 2
20 PRINT I
```

```
30 PRINT "ループ"  
40 NEXT I  
50 END
```

解説

FOR で繰り返しの回数を指定し、FOR と NEXT の間の操作をその回数だけ繰り返して行います。FOR の次に「I = 0」の形で、繰り返しの初期値 (0) を設定します。TO の次で繰り返しの最終値 (10) を設定します。最後に STEP に続けて初期値と最終値との間の増分 (2) を指定します。

プログラムの実行が NEXT (40 行) まで来ると、変数 I が増分だけ増やされます。そして、最終値を超えていなければ FOR (10 行) に戻ります。したがって、初期値が最終値よりも大きく設定されているときも、FOR 文から NEXT 文までのプログラムは 1 度実行されます。

このプログラムでは、10 行の指定で、I が 0、2、4、6、8、10 と増えますので、20 行と 30 行は 6 回実行され、「ループ」は変数 I の値とともに 6 回表示されます。I が最終値 10 と同じになったところでプログラムの実行は FOR~NEXT から出て、NEXT の次の行に移ります。

FOR~NEXT の STEP 以下は省略することができます。省略したときは増分は +1 になります。

FRE

機能

メモリの未使用領域の大きさを返します。

書式

FRE(<引数>)

文例 1

```
PRINT FRE(0)
```

解説 1

引数が数値の場合は、BASIC が使っていないメモリの大きさ(ユーザーエリアの未使用領域)がバイト数で返されます。引数は、数値であれば何でもかまいません。

文例 2

```
PRINT FRE('A')
```

解説 2

引数が文字列の場合は、まず文字領域の整理が行われ、それから未使用の文字領域の大きさがバイト数で出されます。この引数は、文字列であれば何でも（空文字でも）かまいません。

参照

CLEAR、メモリマップ

GET

Disk

機能

ランダムファイルから、ランダム入出力用バッファに1レコードを読み込みます。

書式

```
GET [#]<ファイル番号>[,レコード番号>]
```

文例

```
GET # 1,1
```

解説

<ファイル番号>は、そのファイルを OPEN 文によって開いたときに指定した番号です。<レコード番号>が省略された場合には、直前の GET、PUT 文で参照されたレコードの次のレコードがバッファに読み込まれます。レコード番号として指定可能な最大の数は、4,294,967,295 です。

GET DATE

2

機能

<文字変数>に現在の日付を代入します。

書式

GET DATE <文字変数>[, A]

文例

GET DATE A\$

解説

現在の日付を指定した<文字変数>に代入します。オプションの A を付けると、アラームの設定された日付が代入されます。日付は、「85/04/03」のように年月日がスラッシュで区切られ、月日が1桁の場合は04のように2桁となります。

GET TIME

2

機能

<文字変数>に現在の時刻を代入します。

書式

GET TIME <文字変数>[, A]

文例

GET TIME T\$

解説

現在の時刻を指定した<文字変数>に代入します。オプションの A を付けると、アラームの設定された時刻が代入されます。時刻は、「11:20:03」のように時分秒がコロン(:)で区切られ、時刻が1桁の場合は03のように2桁となります。

GOSUB

機能

指定したサブルーチンを呼び出します。

書式

GOSUB<行番号>

文 例

```

10 GOSUB 30
20 END
30 PRINT "ココカラサブ ルーチン"
40 PLAY "CDE"
50 PRINT "GOSUB ノツキ ノキ ヨウヘモト ル
60 RETURN

```

解 説

GOSUB 命令で指定した行で始まるサブルーチンを呼び出します。呼び出されたサブルーチンは、RETURN 命令によって GOSUB の次の文または行に戻ります。サブルーチンとは、GOSUB で指定する行から、RETURN の行までの独立した1つのプログラムです。

参 照

RETURN、GOTO

GOTO

機 能

指定した行にジャンプします。

書 式

GOTO<行番号>

文 例

```

10 PRINT "10 キ ヨウテ ス":GOTO 30
20 PRINT "20 キ ヨウテ ス":END
30 PRINT "30 キ ヨウテ ス"
40 GOTO 20

```

表示 10 キ ヨウテ ス
30 キ ヨウテ ス
20 キ ヨウテ ス

解説

10行で、30行に飛ぶように指定しているので、20行のEND命令を飛び越して30行が実行されます。そのあと、40行から20行にジャンプして、プログラムは終了します。

参照

ON GOTO、ON GOSUB、GOSUB

HEX\$

機能

<数式>の数值を16進数の文字列に変換します。

書式

HEX\$(<数式>)

文例

PRINT HEX\$(12)

表示 C

参照

VAL、BIN\$、OCT\$、STR\$

IF~THEN~ELSE、IF~GOTO~ELSE

機能

条件判断を行い、その結果に応じた操作を行います。

書式

IF<条件>THEN<文または行番号>[ELSE<文または行番号>]

IF<条件>GOTO<行番号>[ELSE<文または行番号>]

文例

10 A = A + 1

20 PRINT A


```
30 IF A = 20 THEN PRINT "ハタチ" ELSE 10
40 END
```

解説

IF に続けて条件を指定します。その条件が満たされた（0以外の値を取る）ときは THEN 以下の指定が実行され、条件が満たされない（0を値として取る）ときは ELSE 以下の指定が実行されます。

ELSE 以下の指定は省略することができます。その場合、条件が満たされなかったときは次の行が実行されます。

THEN、ELSE の後ろの指定は、文でも行番号でもかまいません。THEN の後ろに行番号を指定する場合、THEN の代わりに GOTO を使って、IF～GOTO～ELSE という形にすることもできます。

条件には、以下のような比較演算子と論理演算子とが使えます。論理演算子は、複数の条件を指定するときに用います。

比較演算は条件が成立したときには「-1」、成立しないときには「0」を値として返します。

表 2.9 比較演算

比較演算子	意味	例
式1=式2	式1が式2と等しい	IF A\$="y"
式1<>式2	式1が式2と等しくない	IF A<>0
式1<式2	式1が式2より小さい	IF A<10
式1>式2	式1が式2より大きい	IF A>B
式1<=式2 (式1=<式2)	式1が式2と等しいか、 小さい	IF A<= B+1
式1>=式2 (式1=>式2)	式1が式2と等しいか、 式2より大きい	IF A\$>="Z"

表 2.10 論理演算

論理演算子	例	意味
NOT	IF NOT (A = 1)	A が 1 でなければ THEN 以下を実行
AND	IF A = 0 AND B = 3	A が 0 で B が 3 のときのみ THEN 以下を実行
OR	IF A < 0 OR B = 0	A が 0 より小さいか、B が 0 のときに THEN 以下を実行

INKEY\$

機能

キーが押されていればその文字を、押されていなければ空文字（ヌル）を返します。

書式

INKEY\$

文例

```
10 A$= INKEY$
20 IF A$="" THEN 10
30 PRINT A$;"=";HEX$(ASC(A$))
40 GOTO 10
```

解説

このプログラムを実行すると、キーボードから打ち込まれた文字と、その文字のアスキーコードが画面に表示されます。

INKEY\$は、**CTRL** + **C** と **CTRL** + **STOP** 以外のコントロール文字も読み取ります。

参照

INPUT、LINE INPUT、INPUT\$、キャラクタコード表

INP

機能

指定した入力ポートからデータを獲得します。

書式

INP(<ポート番号>)

文例

```
A = INP(15)
```

解説

カッコ内のポート番号 (15) で指定された入力ポートから 1 バイトのデータを読み取り、その値を A に代入します。ポート番号に関しては、巻末の「I/O マップ」を参照して下さい。

INP 関数は、ハードウェアと非常に密着した関数で、他の MSX 機種では返す値が違う場合もあります。MSX の互換性を維持するため、広く公表するプログラムには使用しないでください。

参照

OUT、WAIT

INPUT

機能

キーボードから入力されるデータを指定した変数へ代入します。

書式

INPUT["<プロンプト文>"];<変数名>[, <変数名>・・・]

文例

```
10 INPUT "データ";A
20 PRINT A * A:GOTO 10
```

解説

INPUT 命令を実行すると、疑問符 (?) が画面に表示され、プログラムはキーボードからのデータの入力待ちになります。何か数値を入力するとその数値は?に続いて画面に表示され、その値が指定された変数 A に代入されます。

引用符 (") で囲まれた文字列 (上の例の場合"データ") は、プロンプト文と呼ばれるもので、これを指定していれば、?の前にその文字列が表示されて、メッセージの役割を果たします。

変数は、カンマ (,) で区切って複数個指定することもできます。この場合、入力するデータもカンマで区切って、変数の数だけ入力しなければなりません。入力するデータの個数が足りないと、「??」と表示されて入力待ちとなり、個数が多いと「?Extra ignored」と表示され、余分に入力されたデータは無視されます。また、対応する変数の型とデータの型は、一致していなければなりません。型が違っていた場合には、「Redo from start」と表示されて再び入力待ちとなります。

INPUT

機能

ファイルからデータを読み込みます。

書式

INPUT #<ファイル番号>, <変数> [, <変数> ...]

文例

INPUT # 1,A,B

解説

あらかじめ OPEN 文で INPUT モードを指定して開いておいたファイルからデータを読み込み、変数にセットします。INPUT #文はデータを読み込む対象がファイルであることを除けば、INPUT 文とほぼ同じです。

<ファイル番号>は、OPEN 文で指定した番号です。

INPUT #文で読み込むデータは、PRINT #文で書き出した各データと対応した順に読み込まれます。したがって<変数>の型はファイルから入力するデータの型と対応していなくてはなりません。

ファイル中のデータのうち、先に続くスペース、ラインフィードは無視され、スペース、ラインフィード以外の最初の文字がデータの始まりとされます。データは、スペース、キャリッジリターン、ラインフィード、カンマによって区切られていなければなりません。ファイル中のデータが文字型の場合には、最初の文字が引用符 (") であると、その文字型データは、最初の引用から次の引用符までの間に読み込まれた文字で構成されているとみなされます。したがって、引用符で囲まれた文字列中には、引用符を文字として入れることはできません。最初の文字が引用符以外の場合は、文字列は、囲まれていないものとして扱われ、カンマ、キャリッジリターン、ラインフィード、または 255 文字の読み込み終了によって区切られます。数値、または文字型データの読み込み中に、ファイルの終わり (EOF) に達したときは、そのデータ項目はそこで区切られます。

参照

INPUT、OPEN、PRINT #

INPUT\$

機 能

指定されたファイルから指定された長さの文字を獲得します。

書 式

INPUT\$(<文字列>[, [#]<ファイル番号>])

文 例

WD\$= INPUT\$(6,# 2)

解 説

<ファイル番号>で指定されたファイルから<文字数>分の文字列を読み出します。INPUT\$は、**CTRL** + **STOP**、**CTRL** + **C**を除くすべての文字をそのまま読み出すので、INPUT文やLINE INPUT文では入力することのできないラインフィードコードやリターンコードも入力することができます。

#のあとの数値は、読み出すファイルのファイル番号で、OPEN命令で指定されたものと対応しています。ファイル番号が省略された場合には、キーボードから入力されたデータを読み出しますが、INPUT命令と異なり、入力された文字は画面に表示されません。

CTRL + **STOP**でこの入力を中断した場合、CONTは実行できません。

カッコ内の最初の数値は、読み出す文字数を指定しています。キーボードからの入力を読み出す場合、INPUT\$は指定された文字数の文字が入力されるのを待ち続けますが、すでにキーボードバッファに入力済みのデータがあるときには、バッファの中の文字から先に読み出します。

参 照

OPEN、INPUT、LINE INPUT

INSTR

機 能

ある文字列の中から指定された文字列を捜し、見つかった位置を返します。

書 式

INSTR([<数式>,<文字列 1>,<文字列 2>)

文 例

```
10 A$='SUPER MSX'  
20 PRINT INSTR(A$,'M')  
30 END  
表示 7
```

解 説

引数の最初の文字列 (A\$) の中から、2 番目の文字列 (「M」) をさがします。この例の場合、「M」は A\$ の頭から数えて 7 番目にあるので、結果は 7 になります。指定された文字列が見つからなかったときは、結果は 0 になります。また、INSTR (6, A\$,「M」) のように、最初に数値を入れて、さがし始める位置を指定することもできます。

INT

機 能

<数式>以下の最大の整数を返します。

書 式

INT(<数式>)

文 例

```
PRINT INT(-1.2)  
表示 -2
```

解 説

<数式>の値が正のときは FIX と同じですが、負のときは異なります。FIX を参照して下さい。

INTERVAL ON / OFF / STOP

機 能

ON INTERVAL GOSUB 命令の設定を実行するかどうか決定します。

書式

```
INTERVAL ON
INTERVAL OFF
INTERVAL STOP
```

文例

```
ON INTERVAL = 100 GOSUB 100:INTERVAL ON
```

解説

INTERVAL ON は、ON INTERVAL GOSUB の設定を実行するように指定し、以後、指定した時間ごとにサブルーチンにジャンプします。INTERVAL OFF は、設定によるジャンプを禁止します。INTERVAL STOP は、設定によるジャンプを一時止め、後で INTERVAL ON が指定されると直ちにジャンプを実行します。

参照

```
ON INTERVAL GOSUB
```

KEY

機能

ファンクションキーの内容を定義します。

書式

```
KEY<キー番号>,<文字列>
```

文例

```
KEY 1,'LOAD'
```

解説

数値はファンクションキーの番号に対応させるため、1～10 までの値でなければなりません。定義できる文字列は、最大 15 文字までの文字およびコントロール文字です(コントロール文字は、CHR\$を使って表現します)。一度定義したファンクションキーの内容は、あらたに定義し直すか、一旦電源を切って再び電源を入れるまで変わりませんので注意してください。

参 照

CHR\$, KEY LIST、KEY ON/OFF

KEY LIST

機 能

ファンクションキーの内容を画面に表示します。

書 式

KEY LIST

文 例

KEY LIST

解 説

F1～**F10**までの全てのファンクションキーの内容を画面に表示します。
電源投入時には、ファンクションキーの内容は以下のように設定されています。

F1	"color"
F2	"auto"
F3	"goto"
F4	"list"
F5	"run"+CHR\$(13)
F6	"color 15,4,7"+CHR\$(13)
F7	"cload"+CHR\$(34) (MSX2+では"load"+CHR\$(34))
F8	"cont"+CHR\$(13)
F9	"list."+CHR\$(13)+CHR\$(31)+CHR\$(31)
F10	CHR\$(12)+"run"+CHR\$(13)

参 照

KEY、KEY ON/OFF

KEY(n) ON / OFF / STOP

機能

ファンクションキーが押されたときに特別な処理を行うかどうかを決定します。

書式

```
KEY(<キー番号>)ON  
KEY(<キー番号>)OFF  
KEY(<キー番号>)STOP
```

文例

```
10 ON KEY GOSUB 40,60  
20 KEY(1) ON:KEY(2)ON  
30 GOTO 30  
40 PRINT "F1 が オサレマシタ"  
50 KEY(1) STOP:RETURN  
60 PRINT "F2 が オサレマシタ"  
70 RETURN
```

解説

この命令は、ON KEY GOSUB によって定義された処理のサブルーチンの実行を許可するか (ON)、禁止するか (OFF)、一時停止するか (STOP) を決定します。

カッコ内の数値は、ファンクションキーのキー番号に対応しています。例えば、KEY(1) ON を実行すると、**F1** を押すごとに、ON KEY GOSUB により定義されている処理のサブルーチンが呼び出されます。サブルーチンが実行がされている間は、KEY(1) STOP が自動的に実行されるので、サブルーチンの実行が終わるまでは、多重にサブルーチンが呼び出されることはありません。KEY(2) OFF を実行すると、**F2** を押しても、用意された処理のサブルーチンは実行されません。KEY(3) STOP を実行すると、**F3** を押しても、用意された処理のサブルーチンは実行されませんが、再び KEY(3) ON を実行することによって、処理が実行されます。この命令は、KEY(3) ON の状態でのみ有効です。

上のプログラムでは、**F1** に関して、一旦 KEY ON していますが(20行)、その処理のサブルーチンで KEY STOP しているので(50行)、一度 **F1** キーが押されてしまうと、2度目からは「F1 が オサレマシタ」というメッセージは表示されません。

参照

ON KEY GOSUB

KEY ON / OFF

機能

ファンクションキーの内容を画面の下部に表示するかしないかを設定します。

書式

KEY ON
KEY OFF

文例

KEY ON
KEY OFF

解説

KEY ON を実行すると、ファンクションキーの内容が画面の最下行に表示されます。普通の状態では **F1** ～ **F5** までが、シフトキーを押しているときは **F6** ～ **F10** までが表示されます。

KEY OFF を実行すると、ファンクションキーの表示がとりやめられます。KEY ON / OFF は SET SCREEN 命令により、初期状態を登録することができます。

参照

KEY、KEY LIST

KILL

Disk

機能

ファイルを消去します。

書式

KILL <ファイルスペック>

文例

KILL "A:SAMPLE.BAK"

解説

<ファイルスペック>で指定したファイルを消去します。ファイルスペックの指定には、ワイルドカード(?や*)を使うことができますが予想外のファイルまで消去してしまわないよう十分注意が必要です。また、ファイルスペック中で、ドライブ名が指定されていない場合は、自動的に現在選択されているドライブ(カレントドライブ)とみなされます。開いているファイルに対して KILL コマンドを実行すると、「File still open」エラーとなります。

LEFT\$

機能

<文字式>の左側から指定された数の文字列を返します。

書式

LEFT\$(<文字式>, <数式>)

文例

```
10 A$="SUPER HOME COMPUTER"  
20 B$= LEFT$(A$,5)  
30 PRINT B$:END
```

表示 SUPER

解説

与えられた文字列(A\$)の左側から、数値(5)で指定された数の文字を取り出します。指定した数が、文字列の総文字数より大きいときは、文字列がそのまま取り出されます。

LEN

機能

<文字式>で指定された文字列の総文字数を返します。

書式

LEN(<文字式>)

文 例

```
10 A$='SUPER HOME COMPUTER'  
20 PRINT LEN(A$)  
30 END
```

表示 19

解 説

LEN はコントロールコードや空白も文字として数えます。

LET

機 能

変数に値を代入します。

書 式

[LET]<変数名>=<式>

文 例

```
10 LET A = 10  
20 B = 20  
30 PRINT A;B
```

表示 10 20

解 説

代入するものは、文字列でも数値でもかまいません。LET は省略できるので、普通は使いません。

LINE

機 能

グラフィック画面に直線や四角を描きます。

書 式

LINE[[[STEP](X1,Y1)]-[STEP](X2,Y2)[,<カラーコード>][,B[F]]][,<論理演算子>]

文 例

```

10 COLOR 15,4,7:SCREEN 2
20 LINE(88,56)-(168,136),6,BF
30 LINE(128,76)-(108,116),15
40 LINE -STEP(40,-25),15
50 LINE -STEP(-40,0),15
60 LINE -STEP(40,25),15
70 LINE -(128,76),15
80 GOTO 80

```

解 説

10 行でグラフィック画面を指定してから、20 行で赤く塗りつぶされた四角を描き、30 行から 70 行でその四角の中に星を描いています。

LINE は、(X1,Y1) で指定した始点と (X2,Y2) で指定した終点とを結ぶ直線を引きます。X 軸方向には、SCREEN 2~5 および 8 以降では 255、SCREEN 6 と 7 は 511 まで、Y 軸方向には、SCREEN 2~4 では 191、SCREEN 5 以降では 211 までありますので、0 からこの値の範囲の点を指定すると、画面に線が描かれます。

色を指定すると、指定した色で線が引かれます。省略した場合は、COLOR で指定している前景色が使われます。色と指定する値との対応については、COLOR を参照してください。

<カラーコード>の後に B を指定すると、(X1,Y1) から (X2,Y2) を結ぶ線を対角線にした四角が描かれます。BF を指定したときには、四角の中が指定された色で塗りつぶされます(20 行参照)。LINE の最後に<論理演算子>を指定すると、LINE の色を次のように操作することができます。ただしこの指定ができるのは、SCREEN 5 以降に限られます。

C を指定色、SC を LINE の下になる色としたときは、

AND	C AND SC
OR	C OR SC
XOR	NOT(C) AND SC
PSET	C
PRESET	NOT(C)
TAND	C AND SC(C=0 なら C=SC)
TOR	C OR SC(C=0 なら C=SC)
TXOR	NOT(C) AND SC(C=0 なら C=SC)
TPSET	C(C=0 なら C=SC)
TPRESET	NOT(C) (C=0 なら C=SC)

が実際の色となります。

(X1,Y1)を省略して-(X2,Y2)だけが指定されると、最終参照点から(X2,Y2)までを結んだ線が引かれます(70行参照)。最終参照点というのは、線や点などを描くのに1番最後に指定された点のことです。

もう1つのLINEの指定は、STEPを用いたものです。STEPを付けて指定された点は、最終参照点からカッコの中で指定した値の分だけ移動した点を示します。例えば、最終参照点が(100,100)であったとすると、STEP(20,-30)というのは、(120,70)の点を指すことになります。したがって、STEPを用いて引かれる線は、最終参照点とそこからの距離によって指定される点とを結んだものになります(40、50、60行参照)。

80行で80行自身にジャンプしているのは、グラフィック画面では、プログラムの実行が終ると直ちにテキスト画面に移ってしまうからです。これについてはSCREENを参照してください。

参 照

SCREEN、COLOR

LINE INPUT

機 能

キーボードから入力される1行全体の文字列をそのまま文字変数に代入します。

書 式

LINE INPUT["<プロンプト文>"];<文字型変数名>

文 例

```
10 LINE INPUT "デ`ータ";A$
20 PRINT A$
30 GOTO 10
```

表示 デ`ータ

解 説

LINE INPUT 命令を実行すると、プログラムはキーボードからの入力待ちの状態になります。このとき、疑問符(?)は表示されません。以降、何かデータを入力すると、リターンキーを押すまでに入力された文字データが画面に表示され、それがそのまま文字型変数に代入されます。キーボードから入力できる文字列の長さは、254文字以内です。また、LINE INPUT 命令では、カンマ(,)や引用符(")もデータとして入力できます。

なお、INPUT 命令同様、引用符(“)を用いてプロンプト文を指定することができます。

参 照

INPUT

LINE INPUT

機 能

1行単位のデータをファイルから文字変数へ読み込みます。

書 式

LINE INPUT #<ファイル番号>, <文字型変数名>

文 例

```
LINE INPUT #1,A$
```

解 説

あらかじめ OPEN 文で INPUT モードを指定して開いておいたファイルからキャリッジリターンコード+ラインフィードコードで区切られている 1 行単位 (255 文字以内) のデータを指定した文字変数に読み込みます。

SAVE 文でアスキー形式のプログラムファイルを作成した場合、一行一行はキャリッジリターンコード (CHR\$(13)) とラインフィードコード (CHR\$(10)) とで区切られて書き出されています。また、PRINT #文で作成したファイルには PRINT #文を実行するとに数値や文字列の並びがキャリッジリターンコード+ラインフィードコードで区切られて書き出されています。LINE INPUT #文はこれらのファイルからキャリッジリターン+ラインフィードコードには含まれているデータを読み込むときに使います。

<ファイル番号>は OPEN 文で指定した番号です。

参 照

OPEN、CLOSE、PRINT #、LINE INPUT

LIST、LLIST

機能

メモリにあるプログラムを画面に表示、プリンタに印刷します。

書式

[L]LIST[<行番号1>][-[<行番号2>]]

文例

LIST 100-200

解説

行番号1と同じか行番号1以上で最も近い番号の行から、行番号2または行番号2以下で最も近い番号の行までのプログラムを画面に表示(LLISTはプリンタに印刷)します。

LIST

プログラムの先頭から最後までが表示されます。

LIST 50

行番号50の行のみ表示されます。

LIST -100

プログラムの先頭から100行目までが表示されます。

LIST 200-

プログラムの200行目から最後までが表示されます。

LISTの機能を中断したいときは、**STOP**キーを押してください。**STOP**キーをもう一度押せば表示を再開します。また、LISTの機能を終了したいときは、**CTRL** + **STOP**キーを押して下さい。

LISTの代わりにLLISTを用いれば、プログラムをプリンタに印刷することができます。

LOAD

機能

プログラムをメモリ上に読み込みます(ロードする)。

書式

LOAD<ファイルスペック>[,R]

文 例

```
LOAD "A:TEST"
```

解 説

LOAD コマンドは<ファイルスペック>で指定したプログラムをメモリ上にロードします。LOAD 文を実行すると、LOAD 文実行前にメモリ上にあったプログラムは消去されます。また、すべての開いているファイルは閉じられ、変数の値は初期化されます。<ファイルスペック>のデバイス名を省略すると、カレントドライブからロードします。また、ファイル名はカセットファイルに対して用いる時のみ省略でき、この場合はLOAD "CAS:"とします。ディスクファイルの場合はプログラムのセーブ形式が、バイナリでもアスキーでもかまいませんが、カセットファイルではアスキー形式でセーブしたものに限られます。カセット上のバイナリ形式のプログラム (CSAVE したもの) はCLOAD でのみロードすることができます。

R オプションをつけるとファイルは開いたままで、プログラムをロード後、ただちに実行を開始します。

LOAD は、指定されたファイルを見つけてプログラムのロードを開始するまでは、メモリ上のプログラムを保存しています。

参 照

SAVE、CLOAD

LOC

Disk

機 能

ファイル中の現在の位置を返します。

書 式

LOC (<ファイル番号>)

文 例

```
IF LOC (1) > 50 THEN STOP
```

解 説

ランダムファイルを指定した時は、LOC 関数は、最後に読んだり書いたりしたレコード番号を値として返します。ただし、オープン直後は LOC 関数の値は 0 になっています。シーケンシャルファイルを指定したときは、そのファイルがオープンされてから読み出

されたり書き込まれたデータのバイト数を返します（ただし256バイトを1単位とする）。ファイルをシーケンシャル入力モードでオープンした場合は、BASICは最初のセクタを自動的に読むので、そのファイルを読む前でもLOC関数の値は256となります。

LOCATE

機能

テキスト画面のカーソル位置を指定します。

書式

LOCATE[<X座標>][,<Y座標>][,<カーソルスイッチ>]

文例

LOCATE 10,12

解説

カーソルを、指定する位置へ移動します。(10,12)なら、画面の左側から11桁目、上から13行目の位置を意味しています(0から指定するため)。LOCATE命令でカーソル位置を指定してから、PRINTやINPUTを実行すると、その位置に文字や?が表示されます。指定された位置が画面の範囲を越えていると、横、縦とも、それを越えない最大の値に設定し直されます。

<カーソルスイッチ>は、0なら入力待ちのときのみカーソルを表示、0以外なら常に表示となります。LOCATEは、テキスト画面および漢字グラフィックモードでのみ使うことができます。

LOF

Disk

機能

指定されたファイルの大きさを返します。

書式

LOF (<ファイル番号>)

文 例

```
IF NM% > LOF (1) THEN PRINT "INVALID ENTRY"
```

解 説

指定されたファイルの実際の大きさをバイト単位で返します。

LOG

機 能

<数式>の自然対数を返します。

書 式

LOG(<数式>)

文 例

```
PRINT LOG(10)
```

表示 2.302585092994

LPOS

機 能

プリンタのヘッドの位置を調べます。

書 式

LPOS(<式>)

文 例

```
A = LPOS(0)
```

解 説

現在のプリンタのヘッド位置を調べ、その結果を A に代入します。LPOS によって得られる値は、プリンタバッファ上のプリンタヘッドの位置なので、必ずしも実際の（物理的な）プリンタヘッドの位置とは限りません。カッコ内の数値は形式的なものなので、何であっても構いません。

参 照

LPRINT、WIDTH

LPRINT

機 能

プリンタに文字列や数値を印刷します。

書 式

LPRINT[<式>・・・]

文 例

```
10 A$="ABC":A = 123
20 LPRINT A$,A
30 LPRINT "DEF"
```

解 説

LPRINT に続けて指定された数値（この場合、45）や文字列（DEF）をプリンタに出力します。また、数値変数（A）や文字変数（A\$）が指定されると、それらの変数に代入されている数値や文字列がプリンタに出力されます。

LPRINT 命令は、出力先がプリンタであるということを除けば、その機能や使い方は PRINT 命令と全く同じですので、そちらを参照してください。

参 照

PRINT、PRINT USING、LPRINT USING

LPRINT USING

機 能

文字列や数値を指定した書式でプリンタに印刷します。

書 式

LPRINT USING<書式>;<式>・・・

解 説

LPRINT USING を実行すると、引用符 (") で囲まれた書式にしたがって、指定された文字列や数値が編集され、その結果がプリンタに出力されます。

LPRINT USING 命令は、出力先がプリンタであるということを除けば、その機能や使い方は PRINT USING 命令と全く同じですので、そちらを参照してください。

参 照

PRINT USING、LPRINT

LSET、RSET

Disk**機 能**

ランダム入出力用バッファにデータを転送します (PUT コマンドのための準備)。

書 式

LSET<文字型変数>=<文字式>

RSET<文字型変数>=<文字式>

文 例

LSET D\$=MKI\$(D)

解 説

<文字式>の文字が、FIELD 文で指定された文字より短い場合は、LSET 文では左詰め、RSET 文では右詰めでフィールド内を埋め、余分な所は空白で満たされます。逆に FIELD 文での割り当てより長い場合は、LSET 文、RSET 文両方とも文字列の右側が失われます。また、数値データを扱うときは、あらかじめそれらを文字型データに変換しておかねばなりません。それらについては、MKI\$、MKS\$、MKD\$の各関数を参照して下さい。

LSET 文、RSET 文は FIELD 文で定義されていない文字型変数についても、そのフィールド内で左詰めないし右詰めをすることができます。これは、プリント出力の構成に際して利用することができます。

MAXFILES

機能

使用するファイルの数を定義します。

書式

MAXFILES = <ファイル数>

文例

MAXFILES = 3

解説

OPEN 文で開いて使用するファイルの数を定義します。以降、指定したファイルを同時に開いて使用することができます。定義しないときは、MAXFILES = 1 に設定されています。<ファイル数>は 0~15 の範囲を持つ式で、ファイル 1 個について 267 バイトのファイルコントロールブロック領域が確保されます。MAXFILES 文を実行すると、変数がすべて初期化され、現在オープンしているファイルはクローズされます。また、以前の変数の型宣言などは無効になります。

参照

CLEAR、OPEN

MERGE

機能

アスキー形式で保存されているプログラムをメモリ上のプログラムにマージ（混合）します。

書式

MERGE <ファイルスペック>

文例

MERGE "CAS:TEST"

解説

メモリ上のプログラムに<ファイルスペック>で指定したプログラムファイルを混合して1つのプログラムにし、メモリ上に置きます。<ファイルスペック>で指定するファイルはSAVE””,A文を使ってアスキー形式でセーブされていなければなりません。そうでない場合には、ドライブ名を指定していればエラーとなり、CAS:を指定していればテープが回り続けます。<ファイルスペック>のデバイス名を省略すると、カレントドライブが使用されます。ファイル中のプログラムと、メモリ中のプログラムに同一行番号があった場合には、ファイル中の行でメモリ中の行を置き換えます。MERGE コマンドは実行を終了すると、コマンドレベルにもどります。

参照

SAVE、LOAD

MID\$ ①

機能

文字列の中から指定された数の文字列を取り出します。

書式

MID\$1(<文字列>,<式1>[,<式2>])

文例

```
10 A$='SUPER HOME COMPUTER'
20 PRINT MID$(A$,7,4)
30 END
```

表示 HOME

解説

文字列 (A\$) の左側から数えて7番目の文字から、4つの文字を取り出します。取り出したい文字数(この場合は4)が省略されたときや、文字列の残りの文字数よりも大きいときは、指定した位置以降のすべての文字列が取り出されます。

参照

LEFT\$, RIGHT\$, MID\$ ②

MID\$ ②

機能

文字変数の一部を他の文字列と置き換えます。

書式

MID\$(<文字変数>, <式1> [, <式2>]) = <文字列>

文例

```
10 A$='SUPER HOME COMPUTER'
```

```
20 MID$(A$,7,4)='ホーム'
```

```
30 PRINT A$:END
```

表示 SUPER ホーム E COMPUTER

解説

文字列 (A\$) の左から数えて7番目から4文字を、右辺の文字列 ("ホーム") と置き換えます。この代入文を実行しても、もとの文字列の長さは変わりません。また、文字数 (4) を省略すると<文字列>の文字数と見なされます。

参照

MID\$ ①

MKI\$, MKS\$, MKD\$

Disk

機能

各数値を内部表現に対応したキャラクタコードに変換します。

書式

MKI\$ (<整数表記>)

MKS\$ (<単精度表記>)

MKD\$ (<倍精度表記>)

文例

```
RSET A$= MKI$(123)
```


解説

ランダムファイルでは、文字データしか扱えないので、数値データは文字型データに変換してディスクに書き込み、読み出す際は逆に文字型データを数値型データに変換します。MKI\$は整数値を2バイトの文字列に、MKSS\$は単精度数値を4バイトの文字列に、そして MKD\$は倍精度数値を8バイトの文字列にそれぞれ変換します。

数値から文字への変換は数値が持つ内部表現(2進数表現)の値をそのままそれに対応するキャラクタコードにすることによって行われます。この逆の動作をする関数として CVI、CVS、CVD 関数が用意されています。

MOTOR

機能

カセットレコーダのモーターを ON、OFF します。

書式

MOTOR[ON または OFF]

文例

MOTOR ON

解説

リモート端子のあるカセットレコーダを接続したときに、モーターの ON、OFF を行って、カセットテープの早送り、巻き戻しができるようにします。MOTOR ON と指定するとモーターが ON になり、MOTOR OFF とすると OFF になります。MOTOR とだけ指定すると、現在モーターが ON なら OFF に、OFF なら ON になります。

NAME

Disk**機能**

ファイルの名前を変更します。

書式

NAME <ファイルスペック> AS <ファイル名>

文 例

NAME "TEST.BAS" AS "TEMP.BAS"

解 説

変更しようとするファイルは存在しているものでなければなりません。また、付けようとする名前と同じファイルは存在してはいけません。ドライブ名を省略すると、カレントドライブ（現在選択されているドライブ）が選択されます。NAME コマンドが実行されると、そのファイルの名前は変更されますが、ファイルの存在する位置や、占有する大きさなどは変化しません。

NEW

機 能

メモリにあるプログラムを削除し、すべての変数を初期化します。

書 式

NEW

文 例

NEW

解 説

NEW を実行すると、メモリにあるプログラムはすべて削除され、定義されている変数はすべて初期化され、開かれているファイルはすべて閉じられます。

参 照

DELETE、ERASE

OCT\$

機 能

<数式>の数値を8進数の文字列に変換します。

書式

OCT\$(<数式>)

文例

PRINT OCT\$(100)

表示 144**参照**

BIN\$, HEX\$

ON ERROR GOTO

機能

エラーが起こったときにジャンプする行を指定します。

書式

ON ERROR GOTO <行番号>

文例

```

10 ON ERROR GOTO 60
20 A$= INKEY$
30 IF A$='Y' THEN 80
40 A%= A%+100
50 GOTO 20
60 PRINT "シヨリ":A%=0
70 RESUME
80 ON ERROR GOTO 0

```

解説

10行で、エラーが起こったときに60行にジャンプするように指定してあるので、40行のA%が32767を越えて「Overflow」エラーが起きると60行にジャンプし、A%に0を代入します。

あらかじめ、エラーを処理する部分をプログラム中に入力しておき、プログラムの最初にON ERROR GOTOでその部分の最初の行を指定しておけば、エラーが起きたときに、エラー処理用の指定が実行されます。エラー処理用の指定の最後にはRESUME命令を置くと、エラー処理を終了させ、元のプログラムを再開させることができます(70

行参照)。

ON ERROR GOTO 命令は、他の ON ERROR GOTO 命令、RUN、CLEAR、NEW が実行されるまで有効ですので、そのままプログラムを終了すると、ダイレクトモードでタイプミスなどをしても、再びエラー処理用の行へジャンプし、プログラムが実行されてしまいます。したがってプログラムの終了直前には、ON ERROR GOTO の指定を無効にする、ON ERROR GOTO 0 を実行するようにしてください (80 行参照)。

参 照

RESUME、ERL、ERR、ERROR

ON GOTO、ON GOSUB

機 能

<式>で指定されたいずれかの行にジャンプします。

書 式

ON <式>GOSUB <行番号> [, <行番号> . . .]

ON <式>GOTO <行番号> [, <行番号> . . .]

文 例 1

```
10 INPUT "1 カ 2 ラ オシテクタ サイ";A
20 ON A GOTO 100,200
30 GOTO 10
100 PRINT "100 キ ヨウニシ ャンプ シマシタ":END
200 PRINT "200 キ ヨウニシ ャンプ シマシタ":END
```

文 例 2

```
10 INPUT "1 カ 2 カ オシテクタ サイ";A
20 ON A GOSUB 100,200
30 GOTO 10
100 PRINT "100 キ ヨウニシ ャンプ シマシタ":RETURN
200 PRINT "200 キ ヨウニシ ャンプ シマシタ":RETURN
```

解 説

ON と GOTO、あるいは ON と GOSUB の間の値 (A) に応じて、GOTO、GOSUB の後ろで指定した行にジャンプします。ON GOTO は指定した行にジャンプし、ON

GOSUB は指定した行で始まるサブルーチンにジャンプします (GOSUB 参照)。
 GOTO、GOSUB の後ろの指定は、左から 1、2、3・・・という数に対応していて、例えば、上の例では、1 が押されれば 100 行に、2 が押されれば 200 行にジャンプします。
 対応する行番号がないとき (上の例では 0 や 3 以上の数値を入力した場合) は、次の命令が実行されます。負の数が指定されたときは、Illegal function call エラーとなります。

参 照

GOTO、GOSUB

ON INTERVAL GOSUB

機 能

指定した時間ごとにサブルーチンの呼び出しを行うように設定します。

書 式

ON INTERVAL = <時間>GOSUB <行番号>

文 例

```
10 ON INTERVAL = 600 GOSUB 40
20 INTERVAL ON
30 GOTO 30
40 PRINT"*";
50 RETURN
```

解 説

INTERVAL で指定する時間は、1 から 65535 までの数値で、単位は 1/60 秒です。ON INTERVAL GOSUB 命令は、設定を行うだけです。設定後に INTERVAL ON を指定して初めて、指定した時間が経つとサブルーチンにジャンプします。サブルーチンから戻るには、RETURN 命令を使います。また、RETURN に戻り先の行番号を指定することもできます。

RETURN 命令が実行されるまでは、INTERVAL STOP が実行されたのと同じ状態になっているので、多重にインターバル割り込みがかかることはありません。

上のプログラムは 10 秒ごとに「*」を表示するものです。10 行で、ON INTERVAL で指定した時間 (600) ごとに、GOSUB で指定された行 (40 行) にジャンプするように設定しています。

参 照

INTERVAL ON/OFF/STOP

ON KEY GOSUB

機 能

ファンクションキーが押されたときにジャンプする行を指定します。

書 式

ON KEY GOSUB[<行番号>][,<行番号>・・・]

文 例

```
10 ON KEY GOSUB 40,60
20 KEY(1)ON:KEY(2)ON
30 GOTO 30
40 PRINT "F1 カ オサレマシタ"
50 RETURN
60 PRINT "F2 カ オサレマシタ"
70 RETURN
```

解 説

ファンクションキーが押されたときに、呼び出すサブルーチンの行番号を指定します。ON KEY GOSUB に続く行番号 (40 と 60) が、実際に処理を行うサブルーチンの先頭行で、指定されたファンクションキー (**F1** と **F2**) が押されたときには、プログラムの実行はここに移ります。

行番号の並びの順序は、ファンクションキーのキー番号に対応しています。ファンクションキーは **F1** から **F10** まであるので、最大 10 個の行番号が並べられます。

呼び出されたサブルーチンは、GOSUB 命令同様、RETURN 命令によってジャンプした次の行に戻ります。また、RETURN に戻り先の行番号を指定することもできます。ただし、この命令は、ファンクションキーが押されたときの処理の定義を行うだけで、これだけでは処理は実行されません。上のプログラムでも、20 行が実行されて、初めて 40 行と 60 行が実行されています。詳しくは、KEY(n) ON/OFF/STOP を参照してください。

参 照

KEY(n) ON/OFF/STOP

ON SPRITE GOSUB

機 能

スプライトパターンが重なったときにジャンプする行を指定します。

書 式

ON SPRITE GOSUB<行番号>

文 例

```
10 SCREEN 1,1
20 SPRITE$(0) = STRING$(8,CHR$(&B11111111))
30 ON SPRITE GOSUB 90
40 PUT SPRITE 0,(100,100),1,0
50 SPRITE ON
60 X = 255
70 PUT SPRITE 1,(X,100),6,0
80 X = X-1:GOTO 70
90 REM SUB
100 X = 255
110 RETURN
```

解 説

30行で、スプライトのパターンが重なったときには、90行にジャンプするように指定しています。

ON SPRITE GOSUB を実行しておき、SPRITE ON を実行すると、画面上でスプライトパターンが重なったときに、ON SPRITE GOSUB で指定した行 (90) へジャンプします。ジャンプした行以降で、スプライトが重なったときの処理を行ったあとは、RETURN で元の行へ戻ります。また、RETURN に戻り先の行番号を指定することもできます。

なお、SCREEN 4以降のスプライトで特殊なカラーコードを持つものは衝突しても割り込みを起こしません。詳しくは、COLOR SPRITE 文を参照してください。

参 照

SPRITE ON/OFF/STOP

ON STOP GOSUB

機 能

`CTRL` + `STOP` キーが押されたときにジャンプする行を指定します。

書 式

ON STOP GOSUB<行番号>

文 例

ON STOP GOSUB 100

解 説

`CTRL` + `STOP` キーが押されると、プログラムの実行は 100 行以降に移ります。ON STOP GOSUB は、`CTRL` + `STOP` キーが押されたときに、分岐するサブルーチンの行番号を指定するものです。ON KEY GOSUB に続く行番号（この場合、100）が、実際に処理を行うサブルーチンの先頭行になります。呼び出されたサブルーチンは、GOSUB 命令同様、RETURN 命令によってジャンプした次の行に戻ります。また、RETURN に戻り先の行番号を指定することもできます。ON KEY GOSUB の説明を参照して下さい。

参 照

ON KEY GOSUB、STOP ON/OFF/STOP

ON STRIG GOSUB

機 能

ジョイスティックのトリガボタンが押されたときにジャンプする行を指定します。

書 式

ON STRIG GOSUB<行番号>[, <行番号>・・・]

文 例

ON STRIG GOSUB,100,200

解 説

ポート 1 に接続されているジョイスティックのトリガボタン 1 が押されたらプログラムの実行を 100 行以降に移し、ポート 2 に接続されているジョイスティックのトリガボタン 1 が押されたらプログラムの実行を 200 行以降に移します。

この命令は、ジョイスティックのトリガボタンが押されたときに、分岐するサブルーチンの行番号を指定するものです。ON STRIG GOSUB に続く行番号（この場合、100 と 200）が、実際に処理を行うサブルーチンの先頭行で、指定されたトリガボタンが押されたときには、プログラムの実行はここに移ります。

行番号は最大 5 個まで指定することができ、左側から順に以下のトリガボタンと対応しています。

行番号	意 味
1 番目	キーボードのスペースキー
2 番目	ポート 1 に接続されているジョイスティックのトリガボタン 1
3 番目	ポート 2 に接続されているジョイスティックのトリガボタン 1
4 番目	ポート 1 に接続されているジョイスティックのトリガボタン 2
5 番目	ポート 2 に接続されているジョイスティックのトリガボタン 2

ただし、ジョイスティックによっては、トリガボタン 2 のないものもあります。

呼び出されたサブルーチンは、GOSUB 命令同様、RETURN 命令によってジャンプした次の行に戻ります。また、RETURN に戻り先の行番号を指定することもできます。ただし、この命令は、ジョイスティックのトリガボタンが押されたときの処理の定義を行うだけで、これだけでは処理は実行されません。STRIG(n) ON/OFF/STOP を参照して下さい。

参 照

STRIG(n) ON/OFF/STOP、ON KEY GOSUB

OPEN

機 能

ファイルを開きます。

書式

OPEN <ファイルスペック> [FOR <モード>] AS # <ファイル番号> [LEN = <レコード長>]

文例

OPEN "SAMPLE.DAT" FOR OUTPUT AS # 1

解説

OPEN 文は、ファイルを使用するにあたってバッファエリアを割り当て、アクセスモードを決定します。一度ファイルをオープンすると、以後は、<ファイル番号>でデータのやりとりを行います。以下の命令や関数を使用する前には、あらかじめ OPEN 文を実行しておく必要があります。

EOF	FIELD
GET	INPUT #
INPUT\$(n,#m)	LINE INPUT #
LOC	LOF
PRINT #	PRINT # USING
PUT	VARPTR(#n)

<ファイルスペック>でドライブ名を省略したときは、現在のディスクドライブが使用されます。

<モード>はファイルに対するアクセスのモードを決定するもので、次の4種があります。

FOR INPUT	既存のシーケンシャルファイルから入力を行う。
FOR OUTPUT	新しくシーケンシャルファイルを作り出力を行う。
FOR APPEND	既存のシーケンシャルファイルに新たなデータを追加出力する。
<モード>省略	ランダムファイルに対して入出力を行う。

<ファイル番号>は、1 から 15 までの値を用いることができますが、MAXFILES 文で指定した数を越えて指定することはできません。また、既にオープンされているファイルが使用している番号を用いることはできません。

シーケンシャル入力モードおよび追加モードでは、指定されたファイルが存在しないと、「File not found」エラーとなります。シーケンシャル出力モードでは、常に指定された名前のファイルを新しく作り、同一名のファイルがあった場合にはそのファイルは削除されます。ランダム入出力モードでファイルが存在しない場合には新たに作られます。<レコード長>は、ランダム入出力モードでファイルをオープンしたときの、レコード長を設定するものです。これは 1 から 256 の範囲で、省略した場合は 256 に設定されます。

シーケンシャル入力、あるいはランダム入出力モードを指定する場合には、1つのファイルを複数の OPEN 文により開くことができます。この場合、異なるファイル番号により指定することになりますが、シーケンシャル出力モードを指定する場合には、1つのファイルを複数の OPEN 文により開くことはできません。

■ ファイルの指定

OPEN に続く〈ファイルスペック〉で、そのファイル进行操作する場所とファイル名を指定します。ディスクファイル以外では次の7つがあります。

カセットレコーダ	"CAS:ファイル名 (6文字以内)"と指定する
メモリディスク	"MEM:ファイル名"と指定する
ディスプレイのテキスト画面	"CRT:"と指定する
ディスプレイのグラフィック画面	"GRP:"と指定する
プリンタ	"LPT:"と指定する
コンソールデバイス	"CON"と指定する
ヌルデバイス	"NUL"と指定する

コンソールデバイスはキーボードからの読み込み、画面への書き込みに使用します。例えば、

```
COPY "TEST.DOC" TO "CON"
```

とすると、TEST.DOCの内容を画面に表示します。

ヌルデバイスは実際には何もしません。出力文字は無視され、入力は常にエンドオブファイル (**CTRL** + **Z**) となります。

ファイル名が必要なのは、ディスクファイルおよび、CAS:や MEM:だけです。

参 照

CLOSE、INPUT #、PRINT #、MAXFILES、INPUT\$, EOF

OUT

機 能

指定した出力ポートにデータを送ります。

書式

OUT<ポート番号>,<式>

文例

OUT &H22,&H80

解説

&H22 のポート番号で指定された出力ポートへ、&H80 というデータを送ります。ポート番号に関しては、I/O マップを参照してください。

OUT は、ハードウェアと非常に密着した命令で、各々の MSX で異なる動作をする場合があります。MSX の互換性を維持するため、広く公表するプログラムには使用しないでください。

参照

INP

PAD

機能

タッチパネルの状態を調べます。

書式

PAD(<数式>)

文例

X = PAD(1)

解説

ポート 1 に接続されたタッチパネルが押されていたら、その点の X 座標を変数 X に代入します。

PAD に続くカッコ内には、0 から 19 まで値が入ります。PAD 命令を実行すると、指定によって、以下の結果が返されます。

数値	意味	出される値
0、4	タブレットのデータが有効かどうかを調べる	有効=-1 無効=0
1、5	タブレットが押されている点の X 座標を調べる	X 座標
2、6	タブレットが押されている点の Y 座標を調べる	Y 座標
3、7	タブレット上のスイッチが押されているかどうか調べる	押されている=-1 押されていない=0
*8	ライトペンのデータが有効かどうかを調べる	有効=-1 無効=0
*9	ライトペンの X 座標を調べる	X 座標
*10	ライトペンの Y 座標を調べる	Y 座標
*11	ライトペンのスイッチが押されているかどうかを調べる	押されている=-1 押されていない=0
*12、*16	マウスまたはトラックボールのデータを与える	常に-1
*13、*17	マウスまたはトラックボールの X 座標を調べる	X 座標
*14、*18	マウスまたはトラックボールの Y 座標を調べる	Y 座標
*15、*19	マウスまたはトラックボールのデータを与える	常に0

注意

0～3または12～15はポート1に接続された装置を、4～7または16～19はポート2に接続された装置を調べます。ただし、*のついた装置(8～19)はMSX BASIC ver 2.0以降で有効です。

どの周辺装置も座標を調べる前にはそのデータが有効かどうかを確認しなければなりません。それぞれ PAD(0)または(4)、PAD(8)、PAD(12)または PAD(16)を実行して、その値が-1であったら他のデータを調べて下さい。

参照

STRIG、PDL

PAINT

機能

グラフィック画面の、指定された境界色でかこまれた部分を指定された色で塗りつぶします。

書式

PAINT[STEP](X,Y)[,<領域色>][,<境界色>]

文 例

```
10 SCREEN 3
20 CIRCLE(128,96),50,11
30 PAINT(128,96),6,11
40 GOTO 40
```

解 説

(128,96) を中心点にして、黄色で描かれた円の中を赤で塗ります。
PAINT の指定は次の形で行います。

PAINT(X,Y)、領域色、境界色

PAINT は、(X,Y) で指定した位置を含む境界色で囲まれた領域を、領域色で塗りつぶします。色の指定については、COLOR を参照してください。

(X,Y) の指定は STEP 使って行うこともできます。STEP については、LINE を参照してください。

境界色は SCREEN 3 および 5 以上でだけ使えます。SCREEN 2 や 4 では、領域色と同じ色で囲まれた領域が、領域色で指定した色で塗られます。例えば、上の例を SCREEN 2 で行くと次のようになります。

```
10 SCREEN 2
20 CIRCLE(128,96),50,11
30 PAINT(128,96),11
40 GOTO 40
```

領域色の指定を省略すると、COLOR で指定している前景色が使われます。境界色の指定を省略すると、領域色で指定したのと同じ色が使われます。PAINT を実行したときに、指定した境界色 (SCREEN 2 の場合には領域色) で囲まれた領域が無いと、画面一面が指定した領域色で塗りつぶされます。

参 照

SCREEN、COLOR

PDL

機 能

パドルの状態を調べます。

書 式

PDL(<パドル番号>)

文 例

X = PDL(2)

解 説

例ではポート 2 に接続されたパドルの状態を調べ、その結果を変数 X に代入します。

PDL に続くカッコ内には、1 から 12 までの値が入ります。数値が奇数(1、3、5、7、9、11) ならば、ポート 1 に接続されたパドルが指定され、偶数 (2、4、6、8、10、12) ならば、ポート 2 に接続されたパドルが指定されます。

PDL 命令を実行すると、パドルの状態がその回転の度合によって、0~255 の間の値で得られます。

参 照

PAD、STRIG

PEEK

機 能

メモリ上の指定された番地の内容を読み出します。

書 式

PEEK(<番地>)

文 例

A = PEEK(&HA000)

解 説

例ではカッコ内で指定された番地 (&HA000) のメモリの内容が、数値として A に代入されます。読み込み番地は、-32768~65535 の範囲の数値で指定します(数値が負の場合は、その値に 65536 を加えれば実際の番地を知ることができます)。読み込まれるデータは、0~255 の値を持つ数値です。

参 照

POKE、メモリマップ

PLAY ①

機能

音楽を演奏します。

書式

PLAY<文字式1>[,<文字式2>][,<文字式3>]

文例

```
10 PLAY "O4L4T180V8"
20 PLAY "GAGAGGE2"
30 PLAY "EFEFEECR"
40 PLAY "E2CDE2CD"
50 PLAY "EFGGDEDR"
60 PLAY "GAGAGGE2"
70 PLAY "EFEFEECR"
80 PLAY "A2GEGGEC"
90 PLAY "EEDDC2"
```

解説

指定された記号にしたがって、音楽の演奏をします。10行では、演奏のためのテンポ(はやさ)や音量、音階の音域などを指定しています。

文字式は、音楽を演奏するための記号を組み合わせたもので引用符で囲みます。文字変数で指定するときは、その文字変数に演奏用の記号を代入しておきます("X文字変数;"と指定しても、単に文字変数だけを指定しても、結果は変わりません)。記号には、音の高さ、音の長さ、速さ(テンポ)、音の大きさ、音色の指定の5種類があります。

PLAYでは、3つのチャンネルを使うことができ、最大3重和音まで出すことができます。文字式1、文字式2、文字式3はそれぞれチャンネル1、2、3に対応しています。文字式に何も指定しなければ、そのチャンネルからは音がでません。

以下で、各記号についての解説を行います。

例 PLAY "C";"E";"G"

■ 音の高さ

A~G 音の高さを音階名で表現する。A~Gの後ろに#、+を付けると半音上がり、-を付けると半音下がる。

O1~8 音階の音域を1~8の数値で指定する。初期値はO4。例えば、O5は

O4 より1オクターブ上の音階を指す。

N0~96 音の高さを0~96の数値で表現する。N1はO1のC#と同じ高さ、N96はO8のBと同じ。N0はR(休符)と同じ。

The diagram illustrates the mapping of musical notes to numerical values. The top staff shows a scale of notes with accidentals. Below it is a piano keyboard diagram with notes labeled C, D, E, F, G, A, B, C. The bottom staff shows a scale of notes with numerical labels 01 through 08 above them, and N1, N36, N47, N95, N96 below them. Dashed lines connect the notes to their numerical labels.

図 2.13 音の高さ

例 `PLAY "CDE"`
`PLAY "O3CDEO4CDEO5CDE"`
`PLAY "N36N38N40"`

■ 音の長さ

L1~64 音の長さを1/数値音にする。L1は全音符(4拍)、L2は2分音符(2拍)、L64は64分音符(1/16拍)。1音だけ長さを変えるときは、音階名のうしろに数値だけをつける。

. (ピリオド) 音階名、休符の後ろに付けて音の長さを1.5倍する。

R 数値 休符を指定する。数値の指定の方法は、Lのときと同じ。数値を省略するとR4とみなされる。

例 `PLAY "L2CEGL4CEGL8CEG"`
 `PLAY "L4CEGCE8G"`
 `PLAY "L4CD.E..F..."`
 `PLAY "CR4DR2E"`

■ 速さ (テンポ)

T32~255 テンポを指定する。数値で1分間に演奏する4分音符の数を指定する。初期値は T120

例 `PLAY "T64CEGT128CEG"`

■ 音の大きさ

V0~15 音量を指定する。数が大きいほど音も大きくなる。

例 `PLAY "V2CV4DV6EV8FV10G"`

■ 音色

S0~15 音量変化の波を指定する。V とは一緒に指定できない。数値に対応する波形は次の図の通り。

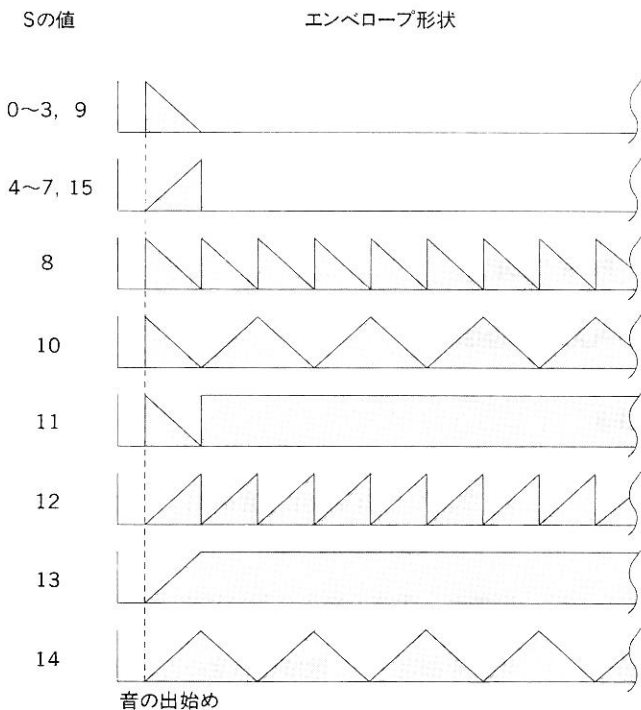


図 2.14 エンベロープ形状

```

例 10 FOR J = 8 TO 14
    20 PLAY "S = J;M1000CDE"
    30 NEXT

```

M1~65535 音量変化の波の周期を指定する。

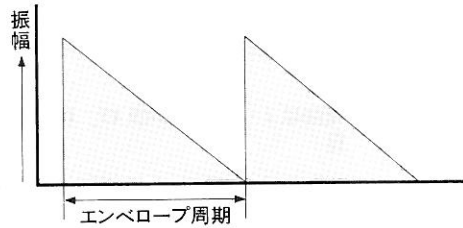


図 2.15 エンベロープ周期

```

例 10 FOR J = 100 TO 3000 STEP 100
    20 PLAY "S8M = J;CDE"
    30 NEXT

```

O、L、T、V の指定は、次に別の値で指定が行われるまで有効です。また V を指定すると、そのチャンネルの S や M の指定はキャンセルされます。

■ 数値変数での指定

なお、次の形式で数値変数も指定できます。

<記号>=数値変数；

```

例 10 FOR J = 1 TO 64
    20 PLAY "L = J;CDE"
    30 NEXT

```

表 2.11 マクロ一覧

	記号	意味	備考	例
音程	A~G	ラ~ソ	音階名	PLAY"CDE"
	#,+	シャープ	半音上げる	PLAY"CD#"
	-	フラット	半音下げる	PLAY"CD-"
	O 数値	オクターブ	音域指定。範囲は1~8 (省略値4)	PLAY"O5C"
	N 数値	音程	数値の範囲は0~96で半音づつ上がる。0は休符	PLAY"N36"
音長	L 数値	音符長	範囲は1~64 (省略値4)。再指定までの音符長	PLAY"L4C"
	数値	音符長	音符の後の数値は、その音の長さだけ指定	PLAY"C32"
	.	符点	A~G、N、Rの後に付けて長さを1.5倍する	PLAY"C4."
	R 数値	休符長	Lと同じ指定だが休符は別に管理される	PLAY"CR8"
速さ	T 数値	テンポ	範囲は32~255 (省略値120)	PLAY"T64"
音量	V 数値	音量	範囲は0~15	PLAY"V8C"
音色	S 数値	波形	範囲は0~15。Vと一緒に指定できない	PLAY"S10"
	M 数値	周波数	範囲は1~65535。Sと共に用いる	PLAY"M10"

PLAY ②

機能

音楽を演奏中かどうかを調べる関数です。

書式

PLAY(<ボイスチャンネル番号>)

文例

```

10 A$='O4L4T120V4'
20 PLAY A$,A$,A$
30 PLAY "C","E","G"
40 PLAY "C","F","A"
50 PLAY "B","A","G"
60 IF PLAY(0)=0 THEN 30
70 GOTO 60

```

解説

PLAY (数値) は、各チャンネルで、指定した音楽を演奏中かどうかを調べます。演奏中であると-1、そうでなければ0が与えられます。

上のプログラムはまず10行で音階の音域、音の長さなどの設定を行って20行でそれを3つのチャンネルで実行させています。30行から50行で音を出し、60行で演奏が終わったかどうかを調べて、終わっていたら30行にジャンプします。70行で60行にジャンプしているのは、PLAYの値が0になるのを待つためです。プログラムでは、50行に続いて60行がすぐ実行されるのですが、演奏は続いているので、1度だけ60行で調べてもPLAYの値は-1で、30行へのジャンプが行われなくなってしまうからです。

PLAYの指定は次の形で行います。

PLAY (チャンネル)

チャンネルは0~3の値で指定します。

0	チャンネル1、2、3 (どれかが演奏中であれば-1となる)
1	チャンネル1
2	チャンネル2
3	チャンネル3

POINT

機能

グラフィック画面上の指定した点の色を調べます。

書式

POINT(X,Y)

文例

```

10 R = RND(-TIME)
20 C = INT(RND(1) * 15 + 1)
30 COLOR ,C,C
40 SCREEN 2
50 A = POINT(128,96)
60 SCREEN 1
70 PRINT A
80 END

```

解説

POINT は、指定した座標の色を調べて、カラー番号で返します。ただし、スプライトの色は調べることができません。

上のプログラムは乱数を使って画面の色を変え、50行でその色を調べています。そのあと60行で画面を切り替え、70行でカラー番号を表示しています。

参照

COLOR

POKE

機能

メモリ上の指定された番地にデータを書き込みます。

書式

POKE <番地>, <データ>

文例

POKE &HA000,57

解説

書き込み番地は、-32768~65535の範囲の数値で指定します(数値が負の場合は、その値に65536を加えれば実際の番地を知ることができます)。書き込むデータは、0~255の値を持つ数値です。

この命令は、現在のメモリ内容を書きかえてしまうため、不用意に使うとエラーやシステムの暴走の原因になることがあります。使うときには、メモリマップなどで使用できる領域かどうかを確認してください。

上の例では、メモリ上の&HA000番地に57というデータを書き込みます。

参照

CLEAR、PEEK、メモリマップ

POS

機 能

テキスト画面上のカーソルの水平位置を調べます。

書 式

POS(<式>)

文 例

```
10 FOR I = 10 TO 99
20 A = POS(0)
30 IF A > 12 THEN PRINT
40 PRINT I;
50 NEXT
```

解 説

POS はカーソルが左から何桁目にあるかを、一番左の位置を 0 とした値で調べます。引き数は仮のものなので、どんな数値でもかまいません。POS は、テキスト画面でのみ使うことができます。

上のプログラムは、単に 10 から 99 までの値を画面に表示するものですが、20 行で POS を使ってカーソルの水平位置を調べ、30 行で、その値が 12 を越えたら改行を行うように指示しています。

参 照

CSRLIN

PRINT

機 能

テキスト画面に文字列や数値を表示します。

書 式

PRINT [<式> . . .]

文 例

```

10 A$='ABC':A = 123
20 PRINT A$,A
30 PRINT A$;A
40 PRINT
50 PRINT 456 'DEF';
60 PRINT 'IIII'

```

解 説

PRINT に続けて数値 (456) や文字列 (DEF、IIII) を指定すると、その数値、文字列が画面に表示されます。また、数値変数 (A) や文字変数 (A\$) が指定されると、それらの変数に代入されている数値 (123) や文字列 (ABC) が画面に表示されます。PRINT だけを指定すると (40 行) 改行が行われます。

数値を表示する場合、その後ろに必ず空白が1つ入ります。また、数値の前には符号用の桁が用意されて、プラスのときには空白、マイナスのときにはマイナス符号が表示されます。

複数のデータを表示する場合、それらの区切りとして、カンマ (,)、セミコロン (;)、空白が使えます。カンマを使うと (20 行)、PRINT の後ろの奇数番目のデータは画面の一番左の列に表示され、偶数番目のデータは 15 桁目に表示されます。ただし上の例の場合、2 番目のデータが数値なので見かけ上は 16 桁目に表示されます。

区切りにセミコロンあるいは空白を使うと (30 行、50 行)、PRINT の後ろのデータがつながって画面に表示されます。一番最後のデータの後ろにセミコロンを置くと、次の PRINT 文のデータが前のデータにつながって表示されます (50 行と 60 行)。

PRINT の省略形として? (疑問符) を使うことができます。

PRINT

機 能

シーケンシャルファイルにデータを出力します。

書 式

PRINT #<ファイル番号>,[<式>…]

文 例

```
PRINT # 1,A$
```


解説

<ファイル番号>は、そのシーケンシャルファイルが出力モードとしてオープンされたときに指定された番号です。

<式>は、ファイルに書き込まれる数値式または文字式です。

PRINT #文は画面に出力するのと同じものをファイルに出力します。そのため、データを読むときに正しく入力できるように、書き込むデータを適切に区切る必要があります。シーケンシャルファイルに対するデータの出力にも、PRINT # USING 文が用意されています。カンマ(,)は省略できません。

参照

OPEN、CLOSE、INPUT #、LINE INPUT #、INPUT\$、PRINT # USING、PRINT、LPRINT

PRINT USING

機能

文字列や数値を指定した書式でテキスト画面に表示します。

書式

PRINT USING <書式> ; <式> . . .

文例

```

10 A$='ABCDE':B$='ZYXWV'
20 A = 123.45:B = -98.765
30 'モ シレツ
40 PRINT USING '!';A$,B$
50 PRINT USING '& &';A$
60 PRINT USING '@ MNL @';A$,B$
70 'スウチ
80 PRINT USING '+####.###';A,B
90 PRINT USING '####.###-';B
100 PRINT USING '*###.###';B
110 PRINT USING '¥¥###.###';B
120 PRINT USING '*###.###';A
130 PRINT USING '####.###';A
140 PRINT USING '####.###^^^'; A

```

```
150 PRINT USING "##";A
160 PRINT USING "コ'ウケイ###.###";B
```

表示 A Z
 A B C
 A B C D E M N L Z Y X W V
 + 1 2 3 . 4 5 0 0 - 9 8 . 7 6 5 0
 9 8 . 7 6 5 -
 * - 9 8 . 7 6 5
 - ¥ 9 8 . 7 6 5
 * 1 2 3 . 4 5 0
 1 2 3 . 4 5 0
 1 2 3 . 4 5 0 E + 0 0
 % 1 2 3
 コ'ウケイ - 9 8 . 7 6 5

解 説

PRINT USING の後ろに指定した、書式文字列でセミコロン (;) 以下の文字列や数値を編集して表示します。

複数のデータを表示するときには、カンマ (,) かセミコロンで区切ります。カンマで区切ってもセミコロンで区切っても同じです。

書式には、文字列を編集するものと数値を編集するものがあります。以下に書式として使う記号を説明します。

■ 文字列を編集するもの

- ! 文字列の左側 1 字を表示する。
- &空白& 文字列の左側から、指定した空白の数+2 文字を表示する。
- @ @を指定した文字列で置き換える。

■ 数値を編集するもの

- # 数値を#で指定した桁数だけ表示する。
- . 小数点の位置を指定する。
- + - 数値に正負符号を付ける。書式の左端に付けると、数値の前に、右端に付けると数値の後ろに正負符号を付ける。
- ** 書式の左端を**にしておくと、数値の整数部の桁数が書式の指定より小さいときに、小さい分だけ*が表示される。
- ¥¥ 書式の左端を¥¥にしておくと、数値の直前に¥が付けられる。¥¥は指数形式の書式指定をしているときには使えない。
- **¥ 書式の左端を**¥にしておくと、数値の直前に¥が付けられ、整数部の桁数が書式の指定より小さいときに、小さい分だけ*が表示さ

れる。* * ¥は指数形式の書式指定をしているときには使えない。

, ,を整数部の#と#の間、あるいは小数点の左側に置いたときには、整数部が3桁ごとにカンマで区切られて表示される。小数点より右に置いたときには、数値の右にカンマが付けられる。カンマ(,)は指数形式の書式指定をしているときには使えない。

^^^^ ^^^^^を#の後ろに付けると、数値が指数形式で表示される。書式に+または-を指定していない場合、数値が正なら数値の前に空白が1つ、負なら数値の前に-が表示される。

% 表示しようとする編集済みの数値が書式で指定した桁数より大きいと、数値の前に%が付けられる。

書式の中にこれらの記号以外の文字を置くと文字の位置に応じて数値の前や後ろにその文字が表示されます(160行参照)。

PRINT # USING

機能

指定されたファイルに指定した書式でデータを書き出します。

書式

PRINT # <ファイル番号>, USING <書式> ; <式> . . .

文例

```
10 OPEN "CRT:" FOR OUTPUT AS # 1
20 A = 123:B = -45
30 A$ = "SUPER"
40 PRINT # 1, USING "####.##-"; A; B
50 PRINT # 1, USING "@ MSX"; A$
60 CLOSE
```

解説

PRINT # USING を実行すると、書式文字列にしたがって、指定された文字列や数値が編集され、その結果がファイルに書き出されます。PRINT # USING は、データを指定された書式にしたがって出力することを除けば、その機能は PRINT # と同じです。また、書式を使った編集方法は、PRINT USING と同じですので、そちらを参照してください。

参 照

PRINT #、PRINT USING

PSET、PRESET

機 能

グラフィック画面上に点を描いたり (PSET)、描いてある点を消したり (PRESET) します。

書 式

PSET[STEP](X,Y)[,<カラーコード>][,<論理演算子>]

PRESET[STEP](X,Y)[,<カラーコード>][,<論理演算子>]

文 例

```
10 SCREEN 3
20 X = INT(RND(1) * 255)
30 Y = INT(RND(1) * 192)
40 C = INT(RND(1) * 15 + 1)
50 PSET(X,Y),C
60 FOR I = 0 TO 100:NEXT
70 PRESET(X,Y)
80 GOTO 20
```

解 説

PSET は、(X,Y) で指定した位置に C で指定された色の点を描きます (色の指定を省略すると、COLOR 命令で指定した前景色が使われます)。

PRESET で色の指定を省略すると、指定した位置に描かれている点を消します (このとき背景色以外の色を指定すると、その色で点が描かれてしまいますので注意してください)。これらの指定には STEP を使うこともできます。STEP については LINE を参照してください。また、色コードの後には <論理演算子> を指定できる場合があります。これについても LINE を参照してください。

参 照

COLOR、LINE

PUT

Disk

機 能

ランダム入出力バッファ中のデータをランダムファイルへ出力します。

書 式

PUT [#]<ファイル番号>[,<レコード番号>]

文 例

PUT # 1,1

解 説

<ファイル番号>で指定されたファイルに対するバッファの内容を書き出します。指定されたファイルは、ランダムアクセスのモードでオープンされていなければなりません。<レコード番号>は、1 から 4,294,967,295 までが有効で、バッファ中のデータが指定されたレコードに書き込まれます。<レコード番号>が省略された場合には、直前の GET 文、PUT 文で参照されたレコードの次のレコードに書き込まれます。

なお、書き出すデータはあらかじめ FIELD 文、LSET 文、RSET 文により準備しておかねばなりません。

バッファの最後を越えてリードまたはライトしようとするとき「Field overflow」エラーとなります。

PUT KANJI

2

機 能

グラフィック画面に漢字を表示します。

書 式

PUT KANJI[(X,Y)],<漢字コード>[,<カラーコード>[,<論理演算子>[,<モード>]]]

文 例

```

10 SCREEN 5:X = 0:Y = 0:I = &H2121
20 PUT KANJI(X,Y),I
30 X = X + 16:IF X = 256 THEN X = 0 ELSE 50
40 Y = Y + 16:IF Y = 208 THEN A$ = INPUT$(1):CLS:Y = 0
50 I = I + 1
60 IF (I AND 255 OR 128) = 128 THEN I = I + 160
70 IF I = &H2820 THEN I = &H3020
80 IF I < &H4F53 THEN 20
90 END

```

解 説

<漢字コード>は JIS 漢字コードで、MSX2 では &H2121 から &H4F53、MSX2+ では &H2121 から &H737C の値を取ります。<論理演算子>については LINE を参照してください。<モード>は 0~2 で次のような意味を持ちます。

- | | |
|---|--------------------|
| 0 | 16×16 ノーマル (省略値) |
| 1 | 偶数ラインだけを縦 8 ドットで表示 |
| 2 | 奇数ラインだけを縦 8 ドットで表示 |

モード 1 と 2 はインタレースモードで使います。漢字 ROM がシステムにない場合は何も表示しません。PUT KANJI は SCREEN 5 以降でのみ有効です。

この例では、漢字 ROM に入っている文字を順に画面に表示します。途中、余分な空白部分は飛ばしています。

参 照

LINE、SCREEN

PUT SPRITE

機 能

スプライトパターンを画面に表示します。

書 式

PUT SPRITE<スプライト面番号>[, [STEP](X,Y)][, <カラーコード>][, <スプライトパターン番号>]

文 例

```

10 SCREEN 1,2
20 SPRITE$(0)=STRING$(32,CHR$(&B11111111))
30 PUT SPRITE0,(100,100),1,0
40 END

```

解 説

このプログラムは SPRITE\$ 命令によってスプライトパターン番号 0 に定義されている四角い模様を、スプライト面 0 番を使って、画面の (100,100) の位置に、黒で表示します。なお、プログラムが終わってもスプライトは画面から消えません。スプライトを消すには、Y 座標に 209 か 208 (SCREEN 4 以降では 217 か 216) を指定して PUT SPRITE を実行するか、SCREEN 命令の第 2 パラメータで再度スプライトの大きさを指定します。スプライト面番号は 0~31 の値で指定します。1 つのスプライト面には、1 つのパターンだけしか表示できません。すでにパターンが表示してある面を指定すると、先に表示されていたパターンは消されます。

スプライト面の 0~31 は同時に表示できますが、横方向に 5 つ以上 (SCREEN 4 以降では 9 つ以上) のパターンが並ぶと、5 つ目以降 (SCREEN 4 以降では 9 つ目以降) のパターンは画面から消えます。

(X,Y) で、パターンを表示する位置を指定します。X は -32~255 の値を、Y は -32~191 の値 (SCREEN 4 以降では ~211) を指定します。STEP をつけて位置を指定することもできますが、これについては LINE を参照してください。(X,Y) を省略すると、そのスプライト面で使われている値が使われます。この (X,Y) はスプライトパターンの左上隅の位置を指しています。

パターンが同じ位置に重なると、スプライト面番号の小さい方のパターンが手前に重なって表示されます。

Y に 208 (SCREEN 4 以降では 216) を指定すると、そのスプライト面以降のパターンが画面から消されます。Y に 209 (SCREEN 4 以降では 217) を指定すると、指定したスプライト面に表示されていたパターンが消されます。

色は、表示するパターンの色で 0~15 の値で指定します。省略すると COLOR 文で指定している前景色となります。カラー番号と色との対応については、COLOR を参照してください。また SCREEN 4 以降の各スプライト面では、1 ライン毎に色を付けることができます。これについては、COLOR SPRITE を参照してください。

スプライト番号は、SPRITE\$ で指定した番号です。

参 照

COLOR、COLOR SPRITE、SPRITE\$、SCREEN

READ

機能

DATA で用意した数値や文字のデータを読み込み、変数に代入します。

書式

READ<変数>[, <変数>・・・]

文例

```
10 READ A$,A,B$,B
20 PRINT B$,B,A$,A
30 DATA アカクラ,3.5
40 DATA サ オウ,2.3
```

解説

30行、40行のデータを READ で読み込み、「アカクラ」を A\$、3.5 を A、「サ オウ」を B\$、2.3 を B にそれぞれ代入しています。

READ は、行番号の小さい DATA のデータから順に読み込み、指定した変数に代入していきます。DATA は、READ で読み出される数値や、文字を用意します。これらの数値や文字のことをデータといいます。データは、1行では255文字以内で、カンマ(,)で区切って指定します。データを複数行指定することもできますが、その場合もそれらのデータは連続したものと見なされます。データは、整数、単精度実数、倍精度実数、文字定数のどれでもかまいませんが、READ 文で指定する変数の型とそれぞれ対応していません。データとして、文字列の先頭や最後に空白、ピリオド(.)、カンマ(,)があるときには、その文字列を引用符(")で囲みます。

参照

DATA、RESTORE

REM

機能

プログラム中に注釈を入れます。

書 式

```
REM[<注釈文>]
```

文 例

```
REM MSX
```

解 説

REM に続く文字列は、プログラム中の注釈になるだけで、プログラムの実行には全く関係しません。プログラムを見やすくする覚え書きとして使用してください。REM の代わりにアポストロフィ (') を用いることもできます。例えば、上の例を、

```
'MSX
```

としても、かまいません。

RENUM

機 能

プログラムの行番号を付け直します。

書 式

```
RENUM[<新行番号>][,<旧行番号>][,<増分>]
```

文 例

```
RENUM 200,100,20
```

解 説

旧行番号が 100 のところから行番号の付けかえが行われます。新しい行番号の最初の番号は 200 で、以後 20 ずつ増えた行番号が付けられます。新行番号の最初の番号(この場合、200)、新行番号の増加分(この場合、20)とも、省略すると 10 が採用されます。行番号の付けかえを始めるところ(この場合、100)を指定しないと、そのプログラムの先頭から新しい行番号に変わります。

例えば、

```
RENUM
```

プログラム全体の行番号を付けかえます。新しい行番号は 10 から始まり、20、30・・・と付けられます。

RENUM 100,50

行番号 50 に新しい行番号として 100 が付けられ、以降 110、120・・・と付け直されます。

RENUM 100,,100

プログラム全体の行番号を付けかえます。新しい行番号は 100 から始まり、200、300・・・と付けられます。

RENUM は、GOTO、GOSUB、THEN、ELSE、ON～GOTO、ON～GOSUB、ERL などでは使われている行番号も、新しい行番号に合わせて変更します。

RESTORE

機能

READ で読む DATA を指定します。

書式

RESTORE[<行番号>]

文例

```
10 RESTORE 50
20 READ A$,A
30 PRINT A$,A
40 DATA アカクラ,3.5
50 DATA ザ オウ,2.3
```

解説

RESTORE で行番号 (50) を指定し、READ でどの DATA から読み始めるかを決めます。これにより、A\$には「ザオウ」、A には 2.3 が代入されます。

行番号が省略されると、READ はプログラムの最初の DATA から読み始めます。

参照

READ、DATA

RESUME

機能

エラー処理を終了し、元のプログラムの実行を再開します。

書式

RESUME[0]または NEXT または<行番号>

文例

```
RESUME 100
```

解説

ON ERROR GOTO でジャンプしたエラー処理の操作を終了し、元のプログラムの 100 行から再び実行を再開します。RESUME 0 と指定すると、エラーが発生した行からプログラムが再開されます。この場合、0 は省略できます。RESUME NEXT と指定すると、エラーが発生した次の行からプログラムが再開されます。

参照

ON ERROR GOTO

RETURN

機能

サブルーチンから戻ります。

書式

RETURN[<行番号>]

文例

```
10 GOSUB 40
20 GOGUB 60
30 END
40 PRINT "40 キ ョウ"
50 RETURN
60 PRINT "60 キ ョウ"
```

70 RETURN 30

解説

サブルーチンを終了し、そのサブルーチンを呼び出した次の行に戻ります。また、上の70行のように、RETURNの後ろに行番号を指定して、復帰する行を指定できます。上のプログラムが実行される順番は、10、40、50、20、60、70、30行となります。

なお、サブルーチンの中でCLEAR命令やMAXFILES命令等を使うと、RETURNによる復帰ができなくなりますので注意してください。

参照

GOSUB、ON GOSUB

RIGHT\$

機能

文字列の右側から指定された数の文字列を取り出します。

書式

RIGHT\$(<文字列> , <数式>)

文例

```
10 A$="SUPER HOME COMPUTER"
```

```
20 PRINT RIGHT(A$,8)
```

```
30 END
```

表示 COMPUTER

解説

カッコ内の文字列(A\$)の右側から、数値(8)で指定された数の文字列を取り出します。指定した数が文字列の総文字数より大きいときは、文字列がそのまま取り出されます。

参照

LEFT\$, MID\$

RND

機 能

<数式>によって指定された0以上1未満の乱数を発生します。

書 式

RND(<数式>)

文 例

```
PRINT RND(1)
```

表示 .748299301545582

解 説

<数式>が正のときは、同じ乱数の並びの新しい乱数を返します。

<数式>が0のときは、同じ乱数の並びの前回使った値を返します。

<数式>が負のときは、その値に対応した固有の乱数の並びを作ります。

RNDの乱数の並びは、プログラムが実行されるたびに同じものが使われますが、RND(-TIME)を1度先頭に用いれば、毎回別の乱数の並びを作ることができます。

RUN

機 能

メモリにあるプログラムの実行を開始します。あるいはフロッピーディスクからファイルをメモリにロードし、そのプログラムを実行します。

書 式

RUN [<行番号>]またはRUN "<ファイルスペック>"[,R]

文 例

```
RUN "NEWFILE"
```

解 説

<行番号>を指定すると、その行から実行が始まります。指定のない場合には、最も若い行番号から実行が始まります。

<ファイルスペック>が指定されると、指定されたプログラムをロード後、実行を開始

します。〈ファイルスペック〉中でデバイス名を省略すると、カレントドライブからロードされます。RUN コマンドを実行すると、すべての開いているファイルは閉じ、目的のプログラムをロードする前にメモリの内容がクリアされます。R オプションを付けた場合には、すべてのデータファイルは開いたままになります。

SAVE

機能

メモリの BASIC プログラムを保存（セーブ）します。

書式

SAVE 〈ファイルスペック〉[,A]

文例

SAVE "A:COM2",A

解説

〈ファイルスペック〉で指定されるファイルにプログラムを書き込みます。デバイス名を省略すると、現在使用しているディスクドライブ（カレントドライブ）に出力されます。フロッピーディスクに保存する場合は、ファイル名は8文字以下で、拡張子は3文字以内で指定します。もし指定したファイル名がすでにディスク上に存在する場合は、古いファイルは消去され、新しいものに変更されます。

A オプションが指定された場合には、プログラムはアスキー形式で保存され、オプションの指定がない場合には、内部表現形式（バイナリ形式）に圧縮されてプログラムの保存が行われます。アスキー形式の保存は、内部表現形式よりも、多くのディスクスペースを必要としますが、保存されたプログラムのファイルにロード以外の操作を加える場合には、アスキー形式で保存されていることが必要です。例えば、MERGE コマンドは、アスキー形式のファイルでなければ実行できません。また、アスキー形式で保存されたファイルは、データファイルとして読み出すこともできます。

カセットファイルに対しての保存では、内部表現（バイナリ）形式では CSAVE、アスキー形式では SAVE と使い分けられますが、ディスクに対しては SAVE" "で内部表現形式、SAVE" ",A でアスキー形式というように、オプションによって使い分けられます。

参照

LOAD、CSAVE、MERGE

SCREEN

機能

画面やスプライトの大きさなどについての設定を行います。

書式

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>][,<インタレースモード>]

文例

SCREEN 1,2

解説

使用する画面を 32×24 文字のテキストモードに設定し、スプライト模様の高さを 16×16 に設定します。最初の値 (1) が画面モードを、カンマに続く次の値 (2) がスプライトの高さを設定しています。

設定を行うには、以下の項目で説明している値を SCREEN に続けて指定します。

■画面モード

12 の画面モードのうちのどれを使うかを指定します。画面の指定をして SCREEN を実行すると、今まで表示されていた画面が消えて、新しい設定となります。ただし、SCREEN 10 以上のモードでは互いに切り替えても画面の初期化はしません。また SCREEN 9 は日本語バージョンでは使えません。BASIC をスタートしたときには、32×24 文字の画面が自動的に選ばれています。

モード	意味
0	40×24 (または 80×26) 文字のテキストモード
1	32×24 文字のテキストモード
2	256×192 点の高解像度グラフィックモード
3	64×48 ブロックの低解像度グラフィックモード
4	256×192 点の高解像度グラフィックモード
5	256×212 点のビットマップグラフィックモード
6	512×212 点のビットマップグラフィックモード
7	512×212 点のビットマップグラフィックモード
8	256×212 点のビットマップグラフィックモード
9	日本語バージョンにはない
10	256×212点のRGB、YJK混在モード、グラフィックルーチン処理あり
11	256×212点のRGB、YJK混在モード、グラフィックルーチン処理なし
12	256×212 点の YJK モード

0、1のテキストモードではプログラムのリストを表示させたりできますが、線や絵を描くことはできません。2以降ではこれとは逆になります。0のテキストモードでは、スプライトを表示できません。また、2以降の画面を指定しても、プログラムが終了するとすぐにテキストモードに戻ってしまいます。2以降の画面を表示し続けたいときには、例えば、次のように指定します。

10 SCREEN 2

20 GOTO 20

このプログラムを実行すると、2の画面が表示され続けます。

それぞれのモードで使える色は、次のようになります。

0	16パレット中の2色
1	16パレット中の3色
2~5、7	16パレット
6	4パレット
8	256色固定
10,11	0~15のパレットおよびYJK
12	YJK

1つのパレットには、512色の中から1色を選んで割り当てることができます。パレットの割り当てについては、COLOR =を参照してください。

MSX2+ではスクリーン10、11、12が追加されています。10、11はRGB、YJK混在モードですが、10の場合はLINE文等のグラフィック命令を実行する場合に、RGB上の命令として処理し、アトリビュートビットのON/OFF等が考慮されます。11、12ではそのような処理はまったくありません。YJK方式についてはVDPの表示モードに関する記述を参照してください。

また、スプライトはSCREEN 4以降では1ライン毎に色を付けることができ、水平線上に8つまで並べて表示することができます (COLOR SPRITE 参照)。

■ スプライトの大きさ

スプライトを画面に表示するときの大きさを設定します。また、SPRITE\$でスプライトを作るときには、ここでの設定と同じ大きさにします (SPRITE\$参照)。BASICをスタートさせた時点では、自動的に0が設定されています。

0	8×8の点でスプライトを作る
1	8×8の点のスプライトを縦横2倍にして表示する
2	16×16の点でスプライトを作る
3	16×16の点のスプライトを縦横2倍にして表示する

画面のモードを変えずに、スプライトの大きさだけを設定するには、例えば、

次のように指定します。このときスプライトの座標データ等は初期化されます。

SCREEN ,2

SCREEN では、ほかに 3 番目、4 番目、5 番目、6 番目の値を指定することによって、以下のような設定を行うことができます。

表 2.12 SCREEN 文の 3~6 番目のパラメータ

	意 味	指定する値	備 考
3 番目	キーを押したときに音を出すか出さないかを設定する	0 =音を出さない 1 =音を出す	BASIC スタート時には 1 に設定されている
4 番目	プログラムをカセットにセーブする時のボーレートの設定	1 = 1200bps 2 = 2400bps	BASIC スタート時には 1 に設定されている
5 番目	使っているプリンタが MSX 標準のものかどうかの設定	0 = MSX 標準 0 以外 = MSX 標準以外	0 以外を指定すると、グラフィック文字は空白、ひらがなはカタカナに置き換えて印字される。
6 番目	インタレースモードの設定	0 =通常 1 =インタレース 2 = Even/Odd 交互 3 = Even/Odd インタレース	MSX2 以降のみ可能。モード 1 は同じページの内容をインタレース表示。モード 2 は 2 つのページの内容を同一ライン上に交互に表示。モード 3 は 2 つのページの内容をインタレース表示。モード 2 と 3 はディスプレイページが奇数でなくてはならない。

参 照

COLOR、COLOR SPRITE、CSAVE、SPRITES

SET ADJUST

2

機 能

画面の表示位置を上下左右に調整します。

書 式

SET ADJUST(<X 座標>, <Y 座標>)

文 例

SET ADJUST(4,4)

解 説

調整した値はバッテリーバックアップされた SRAM に記憶されます。X、Y はそれぞれ-8~+7 の値です。

SET BEEP

2

機 能

BEEP 音を設定します。

書 式

SET BEEP <音色>, <大きさ>

文 例

SET BEEP 2,2

解 説

設定した値はバッテリーバックアップされた SRAM に記憶されます。<音色>、<大きさ>はそれぞれ0~3の値です。

1~4

SET DATE

2

機 能

<日付>を現在の日付として設定します。

書 式

SET DATE <日付>[,A]

文 例

SET DATE "90/01/01"

解 説

設定した値はバッテリーバックアップされた SRAM に記憶されます。オプションの A を付けると、この日付にアラームが設定されます。<日付>は、「85/04/03」のように年月日をスラッシュで区切ります。月日が1桁の場合は04のように2桁とします。アラームの設定日時になったときに、何が起るかは各機種によって異なります。

SET PAGE

2

機能

VRAM上でページングを行います。

書式

SET PAGE <ディスプレイページ>,<アクティブページ>

文例

SET PAGE 0,1

解説

<ディスプレイページ>とは実際にモニタに表示されるページで、<アクティブページ>はデータが書き込まれるページです。ページを切り替えてもその内容は初期化されません。ページは SCREEN 5 と 6 では 0~3、SCREEN 7 以上では 0~1 が使用できません。

SET PASSWORD

2

機能

パスワードを設定します。

書式

SET PASSWORD <パスワード>

文例

SET PASSWORD "sesame"

解説

電源 ON のとき<パスワード>を入力しなければそのシステムが使えないようにします。<パスワード>は 6 文字以内です。なお設定したパスワードを忘れてしまったときには、**GRAPH** キーと **STOP** キーを押しながりセットします。この設定は SRAM に記憶されますが、SET PROMPT や SET TITLE で設定するのと同じ場所を使っているためこれらの設定とは共存できません。最後に設定したものだけが有効となります。設定したパスワードを無効にするときは、

SET TITLE'' ''

を実行します。詳しくは SET TITLE を参照して下さい。

SET PROMPT

2

機能

プロンプトを設定します。

書式

SET PROMPT <プロンプト>

文例

SET PROMPT "Ready"

解説

BASIC のプロンプト「Ok」を好きなように変更します。<プロンプト>は6文字以内です。この設定は SRAM に記憶されますが、SET PASSWORD や SET TITLE で設定するのと同じ場所を使っているのでこれらの設定とは共存できません。最後に設定したものだけが有効となります。プロンプトの設定が無効になった場合は、プロンプトは「Ok」となります。

SET SCREEN

2

機能

現在の WIDTH、SCREEN の各パラメータを初期値として登録します。

書式

SET SCREEN

文例

SET SCREEN

SET SCROLL

2+

機能

ハードウェアスクロールを行います。

書式

SET SCROLL [<X座標>][,<Y座標>][,<左端マスク>][,<2ページ連続>]

文例

SET SCROLL 100,100,1,1

解説

MSX2+専用命令で、V9958のハードスクロールをサポートします。<X座標>は0～511、<Y座標>は0～255です。<左端マスク>が1なら画面の左端8ドット分は表示せず、0なら表示します。<2ページ連続>が1なら連続する2ページ(例えば、0、1)を横につながった1ページとして処理し、0なら単独ページ内のスクロールだけとします。2ページ連続の表示を行うときは表示ページが奇数ページでなくてはなりません。ハードスクロールはSCREEN 1以降、2ページ連続の表示はSCREEN 5以降でのみ可能です。

SET TIME

2

機能

<時刻>を現在の時刻として設定します。

書式

SET TIME <時刻> [,A]

文例

SET TIME "14:50:30"

解説

設定した値はバッテリーバックアップされたSRAMに記憶されます。オプションのAを付けると、この時刻にアラームが設定されます。<時刻>は、「11:20:03」のように時分秒をコロンで区切ります。時刻が1桁の場合は03のように2桁にします。

アラームの設定日時になったときに、何が起こるかは各機種によって異なります。

SET TITLE

2

機能

初期画面のタイトルと色を指定します。

書式

SET TITLE<タイトル>[,<色番号>]

文例

SET TITLE "MSX2+!"

解説

<タイトル>は6文字以内で、ちょうど6文字のときは何かキーを押すまでタイトルのままになります。<色番号>は1~4です。

番号	色
1	青
2	緑
3	赤
4	オレンジ

この設定はSRAMに記憶されますが、SET PASSWORDやSET PROMPTで設定するのと同じ場所を使っているのでこれらの設定とは共存できません。最後に設定したものが有効となります。

PASSWORD、PROMPT、TITLEの機能を使わないときは、

```
SET TITLE""
```

を実行して下さい。

SET VIDEO

2

機能

スーパーインポーズのモードを設定します。

書式

```
SET VIDEO <モード> [, <YM> [, <CB> ]]
```

文例

```
SET VIDEO 2
```

解説

<モード>は

- | | |
|---|----------------|
| 0 | 内部同期、コンピュータ画面 |
| 1 | 外部同期、コンピュータ画面 |
| 2 | 外部同期、スーパーインポーズ |
| 3 | 外部同期、外部ビデオ画面 |

となっています。

<YM>はモニタの画面密度で、0がノーマル、1がハーフトーンです。<CB>はVDPのカラーバスのモードで、0が出力、1が入力モードとなります。

この命令は、スーパーインポーズ機能のない機種では無意味です。

SGN

機能

<数式>の符号を調べ、正の数なら1、0なら0、負の数なら-1を返します。

書式

```
SGN(<数式>)
```

文例

```
PRINT SGN(20)
```

```
表示 1
```

SIN

機能

<数式>の正弦（サイン）を計算します。

書式

SIN(<数式>)

文例

PRINT SIN(3.1415927/2)

表示 1

SOUND

機能

PSG に直接指定して音を出します。

書式

SOUND<レジスタ番号>,<データ>

文例

10 'チャンネル A
20 SOUND 0,254
30 SOUND 1,0
40 SOUND 8,8
50 'チャンネル B
60 SOUND 2,250
70 SOUND 3,0
80 SOUND 9,16
90 'チャンネル C
100 SOUND 6,4
110 SOUND 10,4
120 'キョウツウ
130 SOUND 7,&B011100
140 SOUND 11,172
150 SOUND 12,13
160 SOUND 13,12

解 説

このプログラムは、チャンネル A、B から音を出し、チャンネル C からはノイズを出すものです。チャンネル A と C の音量は一定ですが、チャンネル B では波形によって音量が変わるように指定しています。音を止めるには、**CTRL** キーを押しながら **STOP** キーを押すか、次のように打ち込んでそれぞれリターンキーを押します。

SOUND 8,0 チャンネル A の音が止まる。
 SOUND 9,0 チャンネル B の音が止まる。
 SOUND 10,0 チャンネル C の音が止まる。

SOUND では、A、B、C の三つのチャンネルごとに細かな指定をして、各自あるいは同時に音を出すことができます。指定は次の形で行います。

SOUND レジスタ番号, データ

レジスタ (SOUND 用の特別のメモリ) は 0~13 まであり、番号に応じてそれぞれの機能を持っているので、SOUND では行わせたい機能に応じてレジスタを選択します。データは、それぞれのレジスタの情報となるものです。データはレジスタごとに 1 つずつ指定します。

3 つのチャンネルで使うレジスタを機能別に示します。

表 2.13 SOUND 命令のレジスタ番号

チャンネル	周波数	ノイズ周波数	チャンネル設定	音量	波形周期	波形形状
A	0	6	7	8	11	13
	1				12	
B	2	6	7	9	11	13
	3				12	
C	4	6	7	10	11	13
	5				12	

SOUND で音を出すときに必ず指定しなくてはならないのは、チャンネル設定、音量、それに周波数またはノイズ周波数です。

次に、各レジスタの機能をまとめてみます。

表 2.14 SOUND 命令のレジスタの機能

レジスタ	機能
0、1	チャンネル A の周波数を 2 つのレジスタを使って指定する
2、3	チャンネル B の周波数を 2 つのレジスタを使って指定する
4、5	チャンネル C の周波数を 2 つのレジスタを使って指定する
6	ノイズ周波数を指定する
7	どのチャンネルから音あるいはノイズを出すかを指定する
8、9、10	音量の指定と、波形を使うかどうかの指定を行う
11、12	波形の周期を 2 つのレジスタを使って指定する
13	波形の形状を指定する

■ 周波数の設定

各チャンネルの周波数 f_T に対応するレジスタの値 (T_P) は、次のようにして求めます。

$$T_P = f_{\text{clock}} / 16 \times f_T$$

f_{clock} 1.78977MHz (1.78977×10^6)
 f_T 周波数

例えば、440Hz の周波数を出したいときは、先の式の f_T に 440 をあてはめます。

$$T_P = 1.78977 \times 10^6 / 16 \times 440$$

$$T_P = \text{約 } 254$$

この T_P の値を 256 で割った答がレジスタ 1、3、5 の値、余りがレジスタ 0、2、4 の値になります。

$$254 / 256 \quad \dots \text{答 (この場合、0)} \quad \text{レジスタ 1、3、5 への値}$$

$$\dots \text{余り (この場合、254)} \quad \text{レジスタ 0、2、4 への値}$$

余りを求めるには BASIC の MOD を使って次のように指定すると簡単です。

```
PRINT 254 MOD 256
```

また、 T_P の値が 32767 を越える場合には、次のように指定します (例えば、32768)。

```
PRINT 32768-INT(32768/256)*256
```

チャンネル A にこれらの周波数を指定するときは、次のようにします。

```
SOUND 0,254
```

SOUND 1,0

データが0のときにも、前に指定した値が残っていることがあるので、必ず指定を行うようにします。

■ ノイズ周波数の設定

ノイズ周波数 f_N に対応するレジスタの値 (N_P) は、次のようにして求めます。

$$N_P = f_{\text{clock}} / 16 \times f_N$$

f_{clock} 1.78977MHz (1.78977×10^6)
 f_N ノイズ周波数 (Hz)
 N_P レジスタ 6 に指定する値

30KHz のノイズを出力するには、上の式の f_T に 30 をあてはめて、

$$N_P = 1.78977 \times 10^6 / 16 \times (30 \times 10^3)$$

$$N_P = \text{約 } 4$$

レジスタ 6 へ入れる値は、4 となります。SOUND で指定するには次のように行います。

SOUND 6,4

レジスタ 6 は、三つのチャンネルで共通です。

■ チャンネルの設定

どのチャンネルから音あるいはノイズを出すのかを指定します。指定はチャンネルに対応するビットを 0 にします。

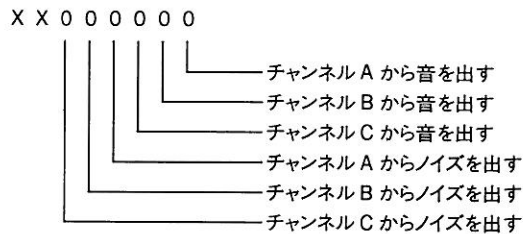


図 2.16 SOUND 文のチャンネル設定

例えば、チャンネル A から音とノイズを出し、チャンネル B から音を出すという指定をするときの値は &B110100 となります。

SOUND で指定すると、次のようになります。

SOUND 7,&B110100

レジスタ 7 のチャンネル指定の場合、このように 2 進数で指定した方がわかりやすく簡単です。

■ 音量設定

音量の設定では、音の大きさと音量を波形に応じて変化させるかどうかの指定を行います。

音量は、0~15 の値で指定します。15 のとき最大音量となり、0 のときには、音が出ません。

音量を波形で変化させるときには 16 を指定します。

チャンネル B で波形変化を行うには、次のように行います。

SOUND 9,16

■ 波形周期の設定

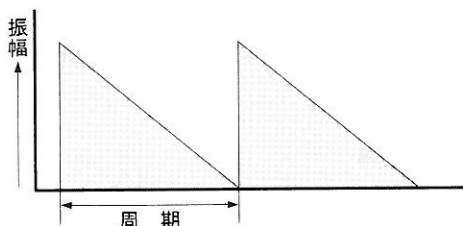


図 2.17 波形周期の設定

波形の周期 E_P は次のようにして求めます。

$$E_P = f_{\text{clock}} \times T_E / 256$$

$$f_{\text{clock}} \quad 1.78977\text{MHz} \quad (1.78977 \times 10^6)$$

$$T_E \quad \text{音が増加・減少する周期 (秒)}$$

$$E_P \quad \text{周期}$$

例えば、音の波が減少して 0 になるまでの時間を 2 秒にするとします。そのときに指定する値は次のようになります。

$$E_P = 1.78977 \times 10^6 \times 2 / 256$$

$$E_P = \text{約 } 14000$$

この値を 256 で割ったときの答がレジスタ 12 への値、余りがレジスタ 11 への値です。

$$14000 / 256 \quad \dots \text{答 (この場合、54)} \quad \text{レジスタ 12 への値}$$

$$\dots \text{余り (この場合、176)} \quad \text{レジスタ 11 への値}$$

余りを求めるには BASIC の MOD を使って次のように指定すると簡単です。

```
PRINT 14000 MOD 256
```

また、TP の値が 32767 を越える場合には、次のように指定します (例えば、32768)。

```
PRINT 32768-INT(32768/256)×256
```

この値を SOUND で指定するには、次のように行います。

```
SOUND 11,176
```

```
SOUND 12,54
```

波形の周波数は三つのチャンネルで共通のレジスタに指定します。

■ 波形の形状の設定

波形の形には次の図のようなものがあります。

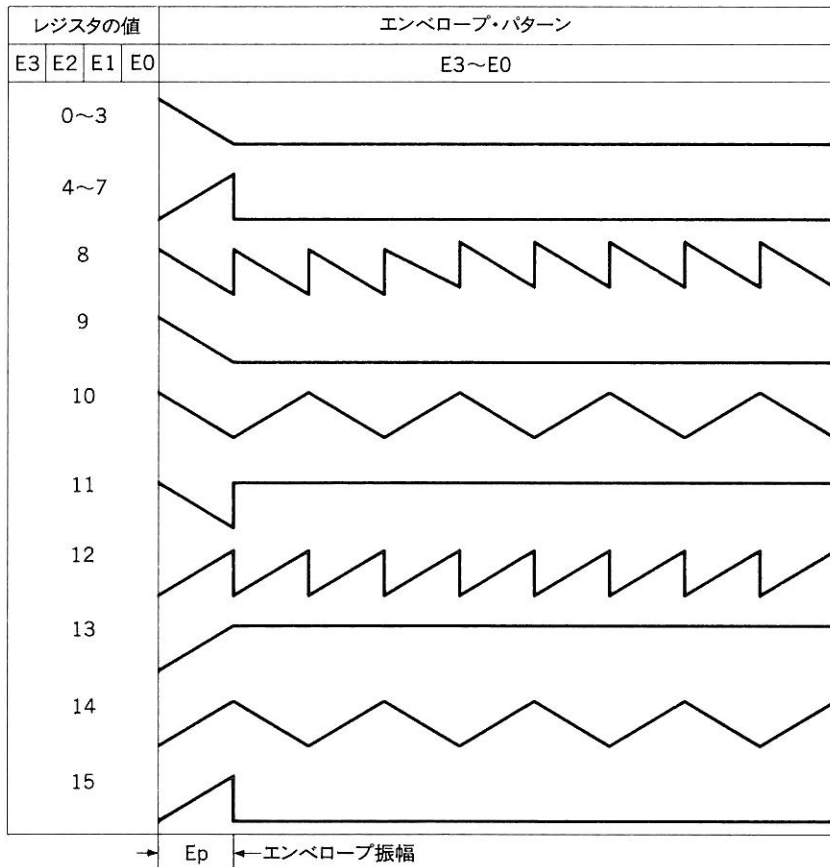


図 2.18 波形形状の設定

この中から選んで値を指定します。値12の波形を指定するには、次のように行います。

SOUND 13,12

波形の形は三つのチャンネルで共通のレジスタに指定します。

SPACE\$

機能

指定した数の空白からなる文字列を作ります。

書式

SPACE\$(<数式>)

文例

```
10 A$='MSX'  
20 B$= SPACE$(5)  
30 PRINT A$+B$+A$:END
```

表示 MSX MSX

SPC

機能

指定した数だけ空白を表示します。

書式

SPC(<数式>)

文例

```
PRINT SPC(10);"TEST"
```

解説

TABの場合は画面の左端から数えて空白を表示しますが、SPCは指定された位置から空白を表示します。SPCで指定できる値は、0から255です。

参 照

SPACE\$, TAB

SPRITE\$

機 能

スプライトで表示するパターンを定義します。

書 式

SPRITE\$(<スプライトパターン番号>)

文 例 1

```

10 SCREEN 1,0
20 FOR I= 1 TO 8
30 READ A$:B$= B$+CHR$(VAL('&B'+A$))
40 NEXT I
50 SPRITE$(0)= B$
60 PUT SPRITE0,(128,96),3,0
100 DATA 00010000
110 DATA 00111000
120 DATA 01111100
130 DATA 11111110
140 DATA 11111110
150 DATA 00010000
160 DATA 00010000
170 DATA 00010000

```

解 説

SPRITE\$の指定は次の形で行います。

SPRITE\$(スプライト番号)=文字列データ

スプライト番号は、SCREEN でスプライトサイズを0か1に指定しているときには0～255、スプライトサイズを2か3に指定しているときには0～63の値を付けます。

スプライトのパターンは、SCREEN のスプライトサイズの指定を0か1にしているときは8×8の点、2か3にしているときには16×16の点で作ります。例えば、上の例では、10行でSCREEN 1,0と指定しているので、スプライトは、8×8の大きさになります。

スプライトのパターンを定義するときには、上の例のように0と1とで表したDATAを使うとわかりやすいものになります。1のところは画面に表示される部分で、0のところは表示されません。パターンの定義は、1バイトずつ行われます。

100行から170行は、1行が1バイト分になっているので、上から順番に文字変数に代入していき、8つとも代入されると、その文字変数の値はスプライトのパターンを表すものとなります。

上の例では、30行でデータ文からA\$に値を読み込み、さらにA\$の値が表すキャラクタコードの文字をB\$に入れています。この行でA\$を、"&B"+A\$として2進数として扱うのは、100行から170行で指定しているデータが2進数に相当するものだからです。20行の指定でB\$には100行から170行の8列のデータが入れられます。このB\$の値がスプライトのパターンを表すものです。50行で、スプライト番号0番としてB\$の値を定義しています。60行のスプライトの表示の方法については、PUT SPRITEをみてください。

上の例の10行をSCREEN 1,1のように変えると、表示されるスプライトが倍の大きさになります。

文例 2

```

10 SCREEN 1,3
20 FOR I= 1 TO 16
30 READ D$
40 A$= A$+CHR$(VAL("&B"+LEFT$(D$,8)))
50 B$= B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
60 NEXT I
70 SPRITE$(0)= A$+B$
80 PUT SPRITE0,(128,96),8
100 DATA 00000000000011110
110 DATA 0000100000101001
120 DATA 0001011111101101
130 DATA 0001111111111001
140 DATA 0011111011111111
150 DATA 0001111111111000
160 DATA 0000001111111000
170 DATA 0000011111110000
180 DATA 0000001111100010
190 DATA 0000000111100100
200 DATA 1100001111100100
210 DATA 0011111111110010
220 DATA 0000001111110010

```



```
230 DATA 0000001111110010
240 DATA 0000000111111100
250 DATA 0000011111110000
```

このように、SCREEN のスプライトサイズの指定を2か3とすると、16×16の点でスプライトのパターンを作ることができます。このとき注意しなくてはならないのが、データを代入していく順番です。

データは、まず左側の8つ(1バイト)が上から順番に、一番下まで定義されます。この時点で16個のデータ(1バイト単位で)が定義されたことになります。次に、今度は右がわの8つが上から順番に、一番下まで定義されます。これで合計32個のデータが定義され、16×16のパターンを定義したことになります。そのため文字変数は、左側用のもの(A\$)と右側用(B\$)のものと2つ用意されています。70行でこれらのデータを合わせて、1つのパターンをスプライトとして定義しています。

SCREEN 4以降のSPRITEでは、1ライン毎に色を付けることができます。これについてはCOLOR SPRITEを参照してください。

参 照

COLOR SPRITE、PUT SPRITE

SPRITE ON / OFF / STOP

機 能

スプライトのパターンが重なったときにジャンプするかどうかを決定します。

書 式

```
SPRITE ON
SPRITE OFF
SPRITE STOP
```

文 例

```
10 SCREEN 1,1
20 SPRITE$(0) = STRING$(8,CHR$(&B11111111))
30 ON SPRITE GOSUB 90
40 PUT SPRITE 0,(100,100),1,0
50 SPRITE ON
60 X = 255
```

```
70 PUT SPRITE 1,(X,100),6,0
80 X = X-1:GOTO 70
90 REM SUB
100 X = 255:SPRITE OFF
110 RETURN
```

解説

50 行で SPRITE ON が指定してあるので、最初にパターンが重なったときには、30 行の指定によって 90 行へジャンプします。100 行で SPRITE OFF が指定してあるので、以後ジャンプは行われません。このプログラムを終了するには、**CTRL** キーを押しながら **STOP** キーを押します（スプライトパターンを消すには、画面モードかスプライトモードを指定した SCREEN 命令を実行してください）。

SPRITE ON を指定すると、以後スプライトのパターンが重なったときには、ON SPRITE GOSUB で指定した行へジャンプします。

SPRITE OFF を指定すると、以後スプライトのパターンが重なっても、ジャンプは起こりません。

SPRITE STOP を指定すると、以後スプライトのパターンが重なったときのジャンプを一時中止し、SPRITE ON が指定されると直ちにジャンプします。

SCREEN 4以降の SPRITE で、COLOR SPRITE によって特殊なカラー番号を指定しておく、その色の部分は割り込みを起こさないようにすることができます。このやり方については、COLOR SPRITE を参照してください。

参照

COLOR SPRITE、ON SPRITE GOSUB

SQR

機能

<数式>の平方根（ルート）を返します。

書式

SQR(<数式>)

文例

```
PRINT SQR(2)
```

表示 1.414213562373

STICK

機能

ジョイスティックがどの方向に押されているかを調べます。

書式

STICK(<ジョイスティック番号>)

文例

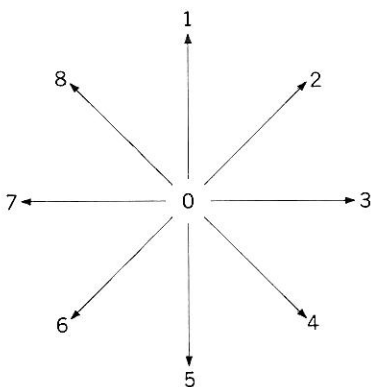
```
10 PRINT STICK(0):GOTO 10
```

解説

STICK に続くカッコ内の数値には、0 か 1 か 2 が入り、それぞれ次のように、何の方向を調べるかを指定します。

番号	意味
0	カーソルキー
1	ポート 1 に接続されたジョイスティック
2	ポート 2 に接続されたジョイスティック

STICK 命令の実行によって得られる数値は、0 から 8 までの 9 つで、それぞれ以下の図に示された方向と対応しています。



0 は、ジョイスティックがどの方向にも向いていない（またはカーソルキーが何も押されていない）ことを示しています。

図 2.19 STICK 命令で得られる値

STOP

機能

プログラムの実行を停止します。

書式

STOP

文例

```
10 PRINT "ハルカ"  
20 STOP  
30 PRINT "キタ"  
40 END
```

解説

プログラムの実行を停止して、直接命令が入力できる状態にします。STOP はプログラムのどこにおいてもかまいません。STOP 命令が実行されると、「Break in xxxx」(xxxx は STOP された行番号) と画面に表示されます。END 命令と異なり、オープンされたファイルがあってもクローズはされません。プログラムを再開するには CONT 命令を実行します。

参照

CONT

STOP ON / OFF / STOP

機能

[CTRL] + [STOP] キーが押されたときに ON STOP GOSUB 文で指定された行に分岐するかどうかを決定します。

書式

```
STOP ON  
STOP OFF  
STOP STOP
```

文 例

STOP ON

解 説

この命令は、ON STOP GOSUB によって定義された処理のサブルーチンの実行を許可するか (ON)、禁止するか (OFF)、一時停止するか (STOP) を決定します。

STOP ON を実行すると、**CTRL** + **STOP** キーを押すごとに、ON STOP GOSUB により定義されている処理のサブルーチンが呼び出されます。

STOP OFF を実行すると、**CTRL** + **STOP** キーを押しても、用意された処理のサブルーチンは実行されず、プログラムの実行が停止します。

STOP STOP を実行すると、**CTRL** + **STOP** キーを押しても、用意された処理のサブルーチンは実行されませんが、再び STOP ON を実行することによって、処理が実行されます。

この命令は ON KEY(n) GOSUB 命令と KEY(n) ON/OFF/STOP 命令の関係と同じです。

STOP ON/STOP の状態では、プログラムによっては実行を停止できなくなるので注意して下さい。

参 照

ON STOP GOSUB、KEY(n) ON/OFF/STOP

STR\$

機 能

数値を文字列に変換します。

書 式

STR\$(<数式>)

文 例

```
10 A = 123:B = -45
20 C = A+B
30 C$ = STR$(A) + STR$(B)
40 PRINT C,C$:END
```

表示 78 123-45

解説

<数式>の数値を10進数の文字列に変換します。文字列に変換された「123」と「-45」は、もう数値ではないので、+によって加算されても「78」にはならず、「123-45」という文字列になります。

参照

VAL、STRING\$

STRIG

機能

ジョイスティックのトリガボタンが押されたかどうか調べます。

書式

STRIG(<ジョイスティック番号>)

文例

```
10 PRINT "スペースキーラオシテクタ'サイ"
20 A = STRIG(0)
30 IF A THEN BEEP
40 GOTO 20
```

解説

スペースキーが押されると、-1が変数Aに代入されて30行が実行され、ビープ音が鳴ります。

STRIG命令は、指定したジョイスティックのトリガボタンまたはスペースキーが押されたかどうかを調べ、押されていれば-1を、押されていなければ0を返します。

STRIGに続くカッコ内には、0から4までの数値が入り、それぞれ以下に示すトリガボタンと対応しています。

番号	意味
0	トリガボタンのかわりにスペースキーを使う
1	ポート1に接続されたジョイスティックのトリガボタン1
2	ポート2に接続されたジョイスティックのトリガボタン1
3	ポート1に接続されたジョイスティックのトリガボタン2
4	ポート2に接続されたジョイスティックのトリガボタン2

ただし、ジョイスティックによってはトリガボタン2のないものもあります。

参 照

ON STRIG GOSUB、STRIG ON/OFF/STOP

STRIG(n) ON / OFF / STOP

機 能

ジョイスティックのトリガボタンが押されたときに、ON STRIG GOSUB 文によって指定された行に分岐するかどうかを決定します。

書 式

STRIG(<ジョイスティック番号>)ON
 STRIG(<ジョイスティック番号>)OFF
 STRIG(<ジョイスティック番号>)STOP

文 例

STRIG(0) ON

解 説

キーボードのスペースキーが押されると、ON STRING GOSUB によって指定された行番号以下の処理を実行します。この命令は、ON STRIG GOSUB によって定義された処理のサブルーチンの実行を許可するか (ON)、禁止するか (OFF)、一時停止するか (STOP) を決定します。

カッコ内の数値は、以下のようにジョイスティックのトリガボタンに対応しています。

番号	意 味
0	キーボードのスペースキー
1	ポート 1 に接続されているジョイスティックのトリガボタン 1
2	ポート 2 に接続されているジョイスティックのトリガボタン 1
3	ポート 1 に接続されているジョイスティックのトリガボタン 2
4	ポート 2 に接続されているジョイスティックのトリガボタン 2

ただし、ジョイスティックによっては、トリガボタン2のないものもあります。

STRIG ON を実行すると、指定されたトリガボタンが押されるごとに、ON STRIG GOSUB により定義されている処理のサブルーチンが呼び出されます。

STRIG OFF を実行すると、指定されたトリガボタンが押されても、用意された処理のサブルーチンは実行されません。

STRIG STOP を実行すると、指定されたトリガボタンが押されても、用意された処理のサブルーチンは実行されませんが、再び STRIG ON を実行することによって、処理が実行されます。

参 照

ON STRIG GOSUB、KEY ON/OFF/STOP

STRING\$

機 能

<式2>で指定された文字だけでできた文字列を作ります。

書 式

STRING\$(<式1>,<式2>)

文 例

```
10 A$= STRING$(10,"*")
20 PRINT A$:END
```

表示 *****

解 説

<式2>で指定した文字を<式1>指定した数だけ連ねた文字列を作ります。<式2>の文字は0~255までの数値でも構いません。この場合は、対応するキャラクタコードが使用されます。

SWAP

機 能

2つの変数の値を交換します。

書 式

SWAP<変数>,<変数>

文 例

```

10 A = 12:B = 34
20 PRINT "A=";A;"B=";B
30 SWAP A,B
40 PRINT "A=";A;"B=";B
50 END

```

表示 A = 12	B = 34
A = 34	B = 12

解 説

行番号 30 の実行により、変数 A と変数 B の値が入れ替わります。変数の型は、文字型でも数値型でも構いませんが、2 つの変数の型は同じでなければいけません。

TAB

機 能

指定された桁まで空白を表示します。

書 式

TAB(<数式>)

文 例

```
PRINT TAB(10);"TEST"
```

解 説

指定した桁まで空白を出力します。指定した桁から表示を行いたいときに、PRINT や LPRINT などの命令の中で使います。指定できるのは、0 から 255 までの値です。現在の出力位置がすでに指定された桁位置よりも大きいときは、何も出力しません。

参 照

SPC

TAN

機能

<数式>の正接（タンジェント）を計算します。

書式

TAN(<数式>)

文例

```
PRINT TAN(3.14159/4)
```

表示 .99999867320603

TIME

機能

時間を計ります。

書式

TIME[=<数式>]

文例

```
10 CLS:TIME = 0
20 LOCATE 0,12:T%= TIME / 60:PRINT T%
30 IF T%= 60 THEN PRINT "1 フォンケイカ" ELSE 20
40 END
```

解説

TIME は初期値を与えると、1/60 秒ごとにその値を1つずつ増やしていきます。そのため、プログラム内の適当な場所に TIME = 0 を置いて、最後に TIME の内容を調べれば、その間のプログラムの実行時間を計ることができます。TIME に与える初期値は、0 から 65535 までの値であれば、0 でなくてもかまいません。また、フロッピーディスクを使用している間などは TIME の値は更新されません。

上のプログラムでは、実行後の経過秒数が画面に表示されます。60 秒経つと「1 フォンケイカ」と表示されて、プログラムは終了します。

なお、上のプログラムの 20 行で T%としているのは、TIME の値を 60 で割ったときの

余りを切り捨てるためです。

バッテリーバックアップされている時計の時刻や日付については、GET DATE、GET TIME、SET DATE、SET TIME を参照してください。

参 照

GET DATE、GET TIME、ON INTERVAL GOSUB、SET DATE、SET TIME

TRON、TROFF

機 能

プログラムの実行状態を追います。

書 式

TRON
TROFF

解 説

TRON を実行させてからプログラムを RUN させると、以後実行されたプログラムの行番号がカギカッコ ([]) 付きで画面に表示されます。もとの状態に戻すときは、TROFF を用います。

USR

機 能

機械語で作られたプログラムを呼び出します。

書 式

USR[<番号>](<引数>)

文 例

I = USR3(J)

解 説

ユーザーの作成した機械語プログラムを呼び出します。機械語プログラムは、あらかじめ

めメモリ中に準備されていなければならず、また DEF USR 命令により、その実行開始番地が指定されていなければなりません。

<番号>は DEF USR 命令により定義された番号に対応しており、0 から 9 が使えます。省略した場合は 0 とみなされます。カッコ内の引数は、BASIC から機械語プログラムへ値を引き渡すためのものです。

参 照

DEF USR

VAL

機 能

文字列を数値に変換します。

書 式

VAL(<文字式>)

文 例

```
10 A$='123':B$='-45'  
20 C$= A$+B$  
30 A = VAL(A$):B = VAL(B$)  
40 C = A+B  
50 PRINT C$,C:END
```

表示 123-45 78

解 説

カッコ内の文字列を数値に変換しています。A と B はそれぞれ「123」と「-45」が数値に変わったものなので、加算によって C は 78 という数になります。

参 照

STR\$

VARPTR

機 能

変数の格納されているメモリ番地、ファイルに割り当てられているファイルコントロールブロックの開始番地を求めます。

書 式

VARPTR (<変数名>)
VARPTR (#<ファイル番号>)

文 例 1

```
PRINT HEX$(VARPTR(A))
```

解 説 1

<変数名>で指定した変数のデータが格納されている変数領域のメモリ番地を求めます。このとき指定されている変数には値が代入されていなければなりません。

文 例 2

```
BVAD = VARPTR(# 1)
```

解 説 2

指定した<ファイル番号>に割り当てられているファイルコントロールブロックの開始番地を得ます。

参 照

メモリマップ

VDP

機 能

VDP (TMS9918、V9938、V9958) のレジスタヘデータを書き込んだり、レジスタからデータを読み出したりします。

書 式

VDP(<レジスタ番号>)

文 例

```
VDP(1) = &HE1
```

解 説

VDP (ビデオディスプレイプロセッサ) のレジスタ 1 に、&HE1 というデータを書き込みます。

読み書きできる範囲は次のとおりです。

	レジスタ 0~7(R/W)	ステータスレジスタ 0(R)	レジスタ 8~(R/W)
MSX	0~7	8	
MSX2	0~7	8	9~24(R8~R23)
MSX2+	0~7	8	9~24、26~28(R8~R27)

参 照

BASE

VPEEK

機 能

VRAM 上の指定した番地の内容を読み出します。

書 式

VPEEK(<番地>)

文 例

```
PRINT VPEEK(0)
```

解 説

読み出すデータを指定する番地は、SET PAGE 命令で設定したページからのオフセットで、0~&hFFFF(65535)の値です。返される値は0~255です。詳しくは VRAM マップを参照して下さい。

参 照

SET PAGE、VPOKE

VPOKE

機能

VRAM 上の指定した番地にデータを書き込みます。

書式

VPOKE<番地>, <データ>

文例

VPOKE 0,65

解説

データを書き込む番地は、SET PAGE命令で設定したページからのオフセットで、0～&HFFFF (65535) の値です。書き込む値は0～255です。詳しくは VRAM マップを参照して下さい。

参照

SET PAGE、VPEEK

WAIT

機能

入力ポートに指定した値が入るまで、プログラムの実行を中断します。

書式

WAIT<ポート番号>, <式1>[, <式2>]

文例

WAIT 1,&H22,&H22

解説

<ポート番号>で指定した入力ポートから読み込んだデータと<式2>との XOR の結果を、<式1>で AND した結果が真になるまで、プログラムの実行を中断します。<式2>が省略されたときは、「0」が適用されます。ポート番号に関しては、I/O マップを参照してください。

WAIT は、ハードウェアと非常に密着した命令で、各々の MSX では異なる動作をする場合があります。MSX の互換性を維持するため、広く公表するプログラムには使用しないでください。

WIDTH

機能

1 行に表示する文字数を設定します。

書式

WIDTH <桁数>

文例

WIDTH 30

解説

テキストモードで画面に表示する、1 行の文字数を 30 文字に設定します。40×24 のモードでは 1~40、32×24 のモードでは 1~32、80×26 のモードでは 1~80、までの値を指定できます。

文字数の設定は、各テキストモードでそれぞれ行います。例えば、32×24 のテキストモードで、上の例のように 30 文字の指定を行っても、40×24 のテキストモードでの設定は変わりません。

BASIC をスタートした時点では、40×24 のテキストモードで 39 文字、32×24 のテキストモードで 29 文字が自動的に設定されます。

WIDTH ステートメントは漢字モードの時には以下のように動作します。

漢字テキストモードの場合

■ KANJI0 か KANJI2 の場合

<桁数>の指定は 26 から 64 が有効です。

■ KANJI1 か KANJI3 の場合

<桁数>の指定は 26 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。

漢字グラフィックモードの場合

常に「Illegal function call」となります。

参 照

CALL KANJI、SCREEN、SET SCREEN

3.14 エラーメッセージ一覧

メッセージ	エラーコード	説 明
Bad drive name	62	ドライブ名の指定に誤りがある。
Bad FAT	60	ディスクのフォーマットが正常でない。
Bad file mode	61	OPEN されたモードに対して、正常な入出力を行っていない。
Bad file name	56	ファイル名が適当でない。
Bad file number	52	オープンしていないファイル番号を指定した。指定したファイル番号が MAXFILES 文で指定した数を越えている。
Bad sector number	63	セクタ番号に誤りがある。
Can't CONTINUE	17	CONT コマンドでプログラムの実行を再開できない。エラーが発生して終了したプログラムは再開できない。プログラムの一部を書き換えたり、CLEAR 文などを実行すると、次に実行すべき箇所へのポインタが壊れて再開ができない。INPUT\$関数を中断した場合再開できない。
Device I/O error	19	カセットレコーダ、プリンタなどへの入出力中にエラーが発生した。
Direct statement in file	57	アスキー形式のプログラムファイルをロードあるいはマージ中にプログラム以外のデータ（行番号がないもの）があった。
Disk full	66	ディスクの空き領域がない。
Disk I/O error	69	ディスクのデータ入出力中になんらかの障害が起きた。
Disk offline	70	ディスクが入っていない。
Disk write protected	68	ディスクが書き込み禁止状態なのに書き込もうとした。
Division by zero	11	0 による除算を行おうとした。0 に対して負のべき乗を行った。
FIELD overflow	50	FIELD 文で定義したフィールドサイズの合計が 256 バイトを越えた。

メッセージ	エラーコード	説明
File already exists	65	NAME 文で指定したファイル名が、既にディスク上に存在している。
File already open	54	OPEN 文で指定したファイル番号が、他のファイルで使用されている。
File not found	53	指定されたファイルが見つからない。
File not OPEN	59	OPEN 文で開いていないファイルに対して入出力を行おうとした。
File still open	64	ファイルがまだクローズされていない。
Illegal direct	12	ダイレクトモードでは実行できないステートメント (実際には MSX BASIC にはない) をダイレクトモードで実行しようとした。
Illegal function call	5	ステートメントや関数の使い方が誤っている。引き数等が指定可能な範囲を越えている。
Input past end	55	ファイル中のデータを全て読み込んだ後、更に入力文を実行した。読み終わっているかどうかは EOF 関数で調べる。
Internal error	51	BASIC 内部でエラーが発生した。
Line buffer overflow	25	入力されたデータが許される文字数を越えた。
Missing operand	24	文中に必要なパラメータが欠けている。
NEXT without FOR	1	NEXT 文に対応する FOR 文がない。
No RESUME	21	エラー処理ルーチンに RESUME 文がない。
Out of DATA	4	READ 文で読み出せるデータがない。
Out of memory	7	プログラムが大き過ぎる、配列が大き過ぎるなどの原因でメモリが足りなくなった。
Out of string space	14	文字型変数用のメモリが足りなくなった。CLEAR 文で対応する。
Overflow	6	演算結果や入力された数値が、型に応じた許容範囲を越えた。
Redimensioned array	10	配列変数を 2 重に定義しようとした。
Rename across disk	71	NAME を異なるディスク間で行おうとした。
RESUME without error	22	エラー処理ルーチン以外で RESUME 文を使用した。
RETURN without GOSUB	3	GOSUB 文と RETURN 文が正しく対応していない。
Sequential I/O only	58	シーケンシャルファイルに対してランダム入出力を行おうとした。
String formula too complex	16	指定した文字式が複雑過ぎる。式を分割する。
String too long	15	文字列が 255 文字より長い。

メッセージ	エラーコード	説明
Subscript out of range	9	指定した配列変数の添字の値が、DIM 文で定義した配列変数の添字の最大値より大きい。
Syntax error	2	文の書き方を誤っている。カッコが対応していない、綴りが違う、記号が足りない等。
Too many files	67	ファイル数が 112 を越えた。
Type mismatch	13	数値型と文字型のデータを混用している。
Undefined line number	8	GOTO 文、GOSUB 文等で指定した行番号がない。削除しようとした行がない。
Undefined user function	18	DEF FN 文で定義していないユーザー定義関数を使用した。
Unprintable error	23,26~49,72~255	メッセージが定義されていないエラー。
Verify error	20	カセットにセーブされたプログラムがメモリ上のプログラムと内容が異なる。

4章

BASICの内部構造

BASICをより高度に利用するためには、BASICインタプリタの内部構造に関する知識が不可欠です。この章では、BASICの内部構造について説明します。

4.1 ユーザーエリア

ユーザーエリアの下限アドレスは、MSX1ではRAM容量が8K、16K、32K、64Kといろいろだったため異なっていましたが、MSX2の場合、RAM容量は最低で64Kなので、ユーザーエリアの下限アドレスは常に8000H番地です。なお、これは【BOTTOM(0FC48H)】の内容から知ることができます。

ユーザーエリアの上限アドレスは、ディスクが実装されていない場合、F380H番地ですが、ディスクが実装されている場合（Disk BASIC 使用時）はドライブ数やディスク容量などにより変わります。これは電源投入後、CLEAR文を実行する前の【HIMEM(0FC4AH)】の内容から知ることができます。

MSXを起動した時のメモリの状態を図2.20に示します。

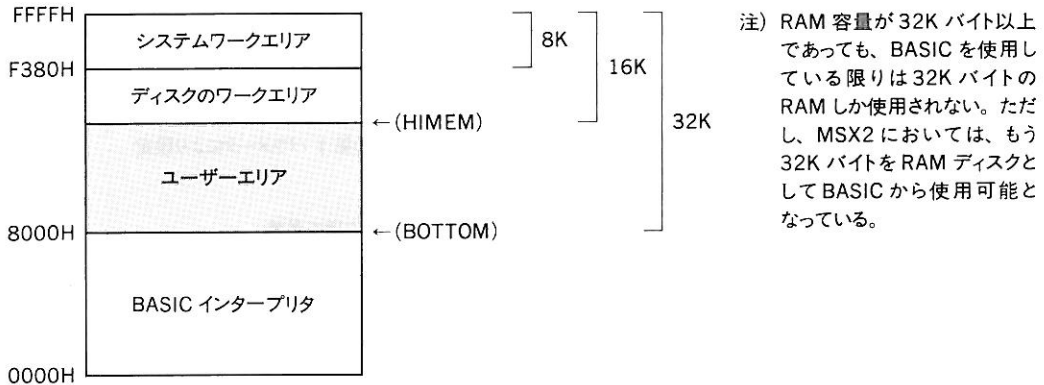


図 2.20 BASIC モードでのメモリの状態

MSX のアプリケーションプログラムを作成する場合、必ず HIMEM の内容を確認、最悪の場合でも暴走しないように対処してください。対処の方法としては、次のようなものが考えられます。

1. ワークエリアをリロケータブルにする
2. ワークエリアを BOTTOM から確保する
3. ドライブの数を減らすように指示して停止する

MSX はディスクが実装されている場合も、**SHIFT** キーを押しながらリセットすることにより、ディスクを切り離すことができます。また、ディスクが 1 ドライブだけ実装されている機種を起動すると、2 ドライブのシミュレータ機能が働き、2 ドライブ分のワークエリアが確保されます。このような時は **CTRL** キーを押しながらリセットすると、2 ドライブシミュレータを無効にして、起動することができます。これにより、約 1.5K バイトのユーザーエリアを確保できます。

4.2 ユーザーエリアの詳細

BASIC 使用時のユーザーエリアの状況を図 2.21 に、それらのエリアの開始番地へのポインタ情報を持つワークエリアを表 2.15 に示します。なお、このワークエリアは読み出し専用ですから、内容を書き換えた場合の動作に関しては保証されません。

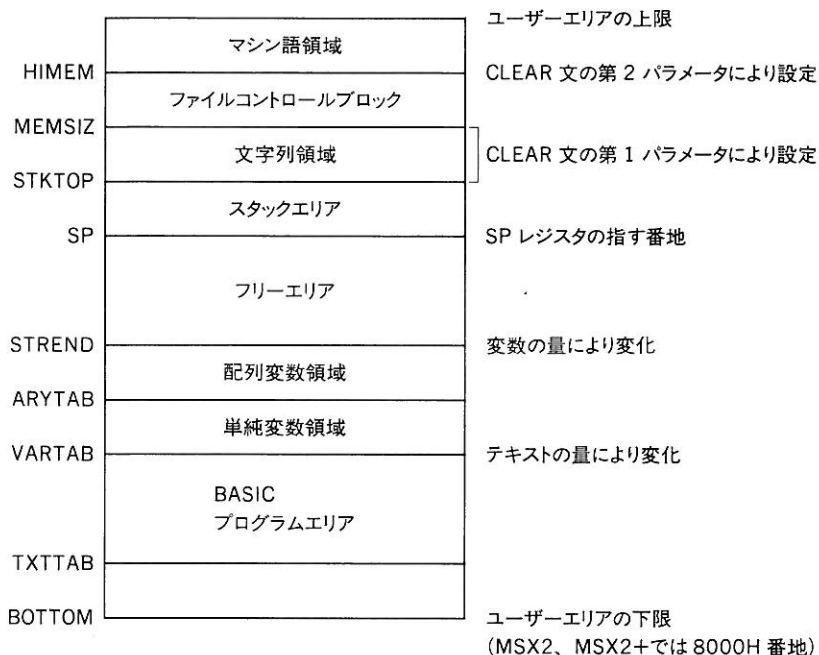


図 2.21 ユーザーエリアの状態

表 2.15 各エリアの開始と終了アドレスを持つワークエリア

エリア名	開始アドレス	終了アドレス
マシン語領域	【HIMEM(0FC4AH)】	ユーザーエリアの終わりまで
ファイルコントロールブロック	【MEMSIZ(0F672H)】	【HIMEM(0FC4AH)】-1
文字列領域 (文字列領域の空きエリアの先頭)	【STKTOP(0F674H)】 【FRETOP(0F69BH)】	【MEMSIZ(0F672H)】-1
スタックエリア	【SP レジスタ】	【STKTOP(0F674H)】-1
フリーエリア	【STREND(0F6C6H)】	【SP レジスタ】-1
配列変数エリア	【ARYTAB(0F6C4H)】	【STREND(0F6C6H)】-1
単純変数エリア	【VARTAB(0F6C2H)】	【ARYTAB(0F6C4H)】-1
プログラムエリア	【TXTTAB(0F676H)】	【VARTAB(0F6C2H)】-1
ユーザーエリア	【BOTTOM(0FC48H)】	リセット時の【HIMEM(0FC4AH)】-1

各ユーザーエリアの役割を以下に示します。

■ BASIC のプログラムエリア

BASIC で作成したプログラムは、ユーザーエリアの下限アドレス (MSX2 では 8000H 番地) から格納され、その大きさはプログラムの量とともに変化します。

■ 変数領域

変数領域は BASIC プログラムエリアのすぐ後ろに位置し、プログラム実行時に使用された変数の名前や値の保存のため確保されます。この時の変数の格納形式を図 2.22 (単純変数の場合)、図 2.23 (配列変数の場合) に示します。なお、配列変数を DIM 文で宣言せずに使用すると添字が 10 の配列としてエリアが確保されます。ただし、4 次元以上の配列はかならず宣言する必要があります。

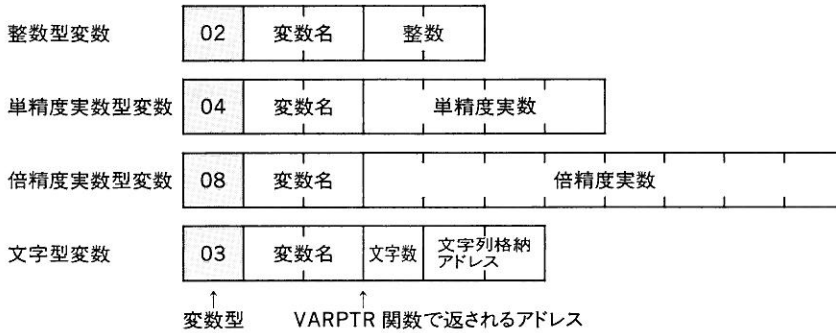
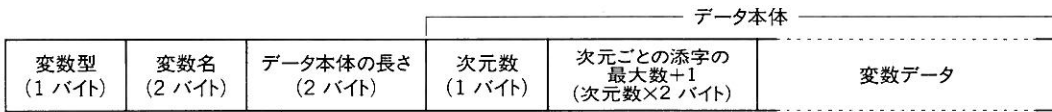
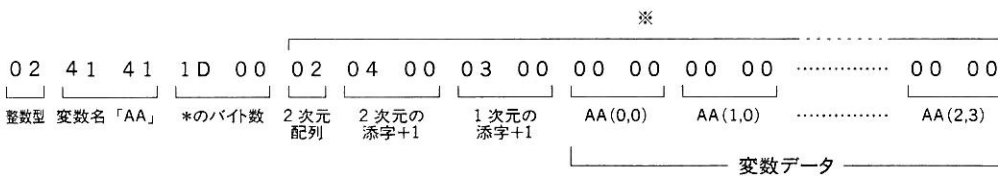


図 2.22 単純変数の格納形式



例 DEFINT A: DIM AA (2,3)



注意 変数データの形式は単純変数の格納形式と同じ。2 バイトの数値は上位と下位バイトが逆に格納される。

図 2.23 配列変数の格納形式

■ フリーエリア

プログラムエリアや変数領域が大きくなったり、スタックが多くなって、フリーエリアがなくなると「Out of memory」エラーになります。なお、BASIC の FRE 関数で PRINT FRE (0) を

評価することにより、フリーエリアの大きさを知ることができます。

■ スタックエリア

BASIC が使用するスタックエリアです。GOSUB 命令や FOR 命令などの実行時に上位アドレス側から順に使用されます。

■ 文字列領域

文字列変数の内容を保存するために使用される領域で、上位アドレス側から順に使用されます。この領域の大きさは BASIC の CLEAR 文の第1パラメータで指定することができ、省略時は200バイトの大きさで確保されます。このエリアがすべて使われてしまうと「Out of string space」となります。BASIC の FRE 関数で PRINT FRE ("")) を評価することにより、文字列領域の残りの大きさを知ることができます。

■ ファイルコントロールブロック

ファイルの情報を保存する領域で、1ファイルごとに 10BH (267) バイト分の領域が確保されます。ファイルいくつ分の領域を確保するかは BASIC の MAXFILES 文で指定でき、リセット時はファイル1つ分 (MAXFILES = 1) の領域が取られています。しかし、それとは別に SAVE、LOAD 命令のために、常時かならず1つ分の領域が確保されているので、実際にはこれも含めて2つ分の領域が確保されていることになります。ファイルコントロールブロックの形式を表 2.16 に示します。

表 2.16 ファイルコントロールブロック (FCB) の形式

変位	ラベル	意味
+0	FL.MOD	ファイルがオープンされたモード
+1	FL.FCA	BDOS 用 FCB へのポインタ (Low)
+2	FL.LCA	BDOS 用 FCB へのポインタ (High)
+3	FL.LSA	バックアップキャラクタ
+4	FL.DSK	デバイスナンバー
+5	FL.SLB	インタープリタが内部で使用
+6	FL.BPS	FL.BUF の位置
+7	FL.FLG	いろいろな情報を持つフラグ
+8	FL.OPS	仮想的なヘッドの位置
+9~	FL.BUF	ファイルバッファ (256 バイト)

■ マシン語領域

マシン語プログラムを扱ったり、メモリを直接操作する場合の領域です。これらを行う場合は、CLEAR 命令によって、あらかじめこの領域を確保しておく必要があります。

■ ディスクのワークエリア

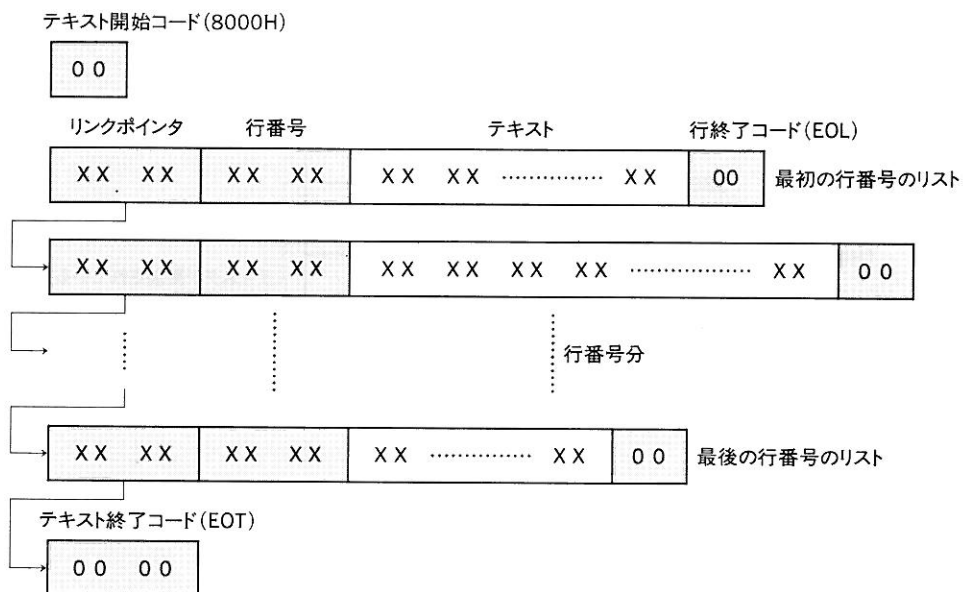
ディスク実装時に確保されるワークエリアを図 2.24 に示します。この領域はディスクが実装されていない場合は存在しませんので注意してください。なお、図中左のラベルは、そこにこのアドレスの情報があることを意味します。



図 2.24 ディスクのワークエリア

4.3 BASIC プログラムの格納形式

BASIC プログラムはメモリ上に図 2.25 のような形式で記憶されており、その内容には以下のような意味があります。



注意 リンクポインタ、行番号は上位と下位バイトが逆に格納されています。

図 2.25 テキスト格納形式

■ リンクポインタ

次の行へのテキストポインタが絶対アドレスで書かれています。

■ 行番号

文字どおりプログラムの行番号が格納されており、通常は 0~65529 (0000H~FFF9H) の値が入ります。メモリを直接操作することにより 65530 以上の行番号を作ることも可能ですが、LIST コマンドはその行以降を表示しません。

■ テキスト

ここにプログラム本体が中間コード形式で格納されています。中間コードに変換されるものは、

予約語（キーワード）、演算子、数値で、その他のもの（変数名や文字列定数など）はキャラクタコードの形で格納されます。中間コードを表 2.17 に、テキスト中の数値形式を図 2.26 に示します。

キャラクタコードについては、巻末の付録を参照してください。なお、グラフィックキャラクタは「CHR\$(1)+(グラフィックキャラクタコード+64)」の2バイト（2文字分）で格納されますので、グラフィック文字を取り扱う時には注意が必要です。

表 2.17 中間コード一覧表

>	EE	CSNG	FF9F	GOTO	89	NAME	D3	SGN	FF84
=	EF	CSRLIN	E8	HEX\$	FF9B	NEW	94	SIN	FF89
<	F0	CVD	FFAA	IF	8B	NEXT	83	SOUND	C4
+	F1	CVI	FFA8	IMP	FA	NOT	E0	SPACES\$	FF99
-	F2	CVS	FFA9	INKEY\$	EC	OCT\$	FF9A	SPC (DF
*	F3	DATA	84	INP	FF90	OFF	EB	SPRITE	C7
/	F4	DEF	97	INPUT	85	ON	95	SQR	FF87
^	F5	DEFDBL	AE	INSTR	E5	OPEN	B0	STEP	DC
¥	FC	DEFINT	AC	INT	FF85	OR	F7	STICK	FFA2
ABS	FF86	DEFSNG	AD	IPL	D5	OUT	9C	STOP	90
AND	F6	DEFSTR	AB	KEY	CC	PAD	FFA5	STR\$	FF93
ASC	FF95	DELETE	A8	KILL	D4	PAINT	BF	STRIG	FFA3
ATN	FF8E	DIM	86	LEFT\$	FF81	PDL	FFA4	STRING\$	E3
ATTR\$	E9	DRAW	BE	LEN	FF92	PEEK	FF97	SWAP	A4
AUTO	A9	DSKF	FFA6	LET	88	PLAY	C1	TAB (BD
BASE	C9	DSKI\$	EA	LFILES	BB	POINT	ED	TAN	FF8D
BEEP	C0	DSKO\$	D1	LINE	AF	POKE	98	THEN	DA
BIN\$	FF9D	ELSE	3AA1	LIST	93	POS	FF91	TIME	CB
BLOAD	CF	END	81	LLIST	9E	PRESET	C3	TO	D9
BSAVE	D0	EOF	FFAB	LOAD	B5	PRINT	91	TROFF	A3
CALL	CA	EQV	F9	LOC	FFAC	PSET	C2	TRON	A2
CDBL	FFA0	ERASE	A5	LOCATE	D8	PUT	B3	USING	E4
CHR\$	FF96	ERL	E1	LOF	FFAD	READ	87	USR	DD
CINT	FF9E	ERR	E2	LOG	FF8A	REM	8F	VAL	FF94
CIRCLE	BC	ERROR	A6	LPOS	FF9C	RENUM	AA	VARPTR	E7
CLEAR	92	EXP	FF8B	LPRINT	9D	RESTORE	8C	VDP	C8
CLOAD	9B	FIELD	B1	LSET	B8	RESUME	A7	VPEEK	FF98
CLOSE	B4	FILES	B7	MAX	CD	RETURN	8E	VPOKE	C6
CLS	9F	FIX	FFA1	MERGE	B6	RIGHT\$	FF82	WAIT	96
CMD	D7	FN	DE	MID\$	FF83	RND	FF88	WIDTH	A0
COLOR	BD	FOR	82	MKD\$	FFB0	RSET	B9	XOR	F8
CONT	99	FPOS	FFA7	MKI\$	FFAE	RUN	8A	'	3A8FE6
COPY	D6	FRE	FF8F	MKS\$	FFAF	SAVE	BA		
COS	FF8C	GET	B2	MOD	FB	SCREEN	C5		
CSAVE	9A	GOSUB	8D	MOTOR	CE	SET	D2		

8進数(&O)	0B	XX	XX					
16進数(&H)	0C	XX	XX					
行番号(RUN後)	0D	XX	XX	分岐命令の飛び先となる行のメモリ上の絶対アドレス				
行番号(RUN前)	0E	XX	XX	分岐命令の飛び先の行番号。一度RUNされると識別コードがODHとなり絶対アドレスとなる。				
10~255の整数(%)	0F	XX						
0~9の整数(%)	11~1A							
256~32767の整数(%)	1C	XX	XX					
単精度実数(!)	1D	XX	XX	XX	XX			
倍精度実数(#)	1F	XX	XX	XX	XX	XX	XX	XX
2進数(&B)	&	B	「&B」に続く0や1のキャラクタ					

図 2.26 テキスト中の数値形式

数値は予約語や変数名との区別をつけるために「識別コード」と呼ばれる番号が割り付けられており、それを見ることで後に続く値がどのようなものかわかるようになっています。

2バイトの数値の場合は、上位と下位バイトが逆に格納されます。符号付き数値については、識別コードの前に+または-の中間コードが置かれるのみであり、数値自体はかならず正の数で記憶されます。浮動小数点表記法についてはおおむね 6.3 のマスパック (Mathematical Package) の説明と同じですが、上記のように数値はかならず正の数で記憶される点に注意してください。2進数(&B)については、識別コードは割り当てられておらず、アスキーコードそのままの形で記憶されます。

5章

マシン語とのリンク

これまで説明したように、MSX BASIC は強力な機能を備えています。さらに実行速度を上げたり、MSX2 のハードウェア機能を限界まで引き出す場合は、マシン語を使用しなければなりません。この章では、その必要が生じた時のために、BASIC からマシン語プログラムを呼び出す方法、およびその際に知っておくべき情報について説明します。

5.1 USR 関数

BASIC からのマシン語の呼び出しは、以下の手順で行います。USR 関数のカッコの中の値は、引数としてマシン語に渡すためのものです。引数は数式、文字式のいずれでもかまいません。

1. DEF USR 文で、マシン語プログラムの実行開始アドレスを指定する
2. USR 関数でマシン語を呼び出す
3. マシン語から BASIC に戻る場合は、RET 命令 (C9H) を実行する

例

C000H 番地が実行開始アドレスであるマシン語プログラムを呼び出す。

```
DEF USR =&HC000  
A = USR(0)
```

5.2 USR 関数の引数と戻り値によるデータの受け渡し

BASIC からマシン語に引数を渡す場合、マシン語側ではレジスタ A の内容でその型を判断します(表 2.18)。目的の値は引数の型により、図 2.27 のような形式で格納されていますから、それに応じて値を受け取ります。例として、文字列型の引数を受け取るプログラムをリスト 2.5 に示します。

表 2.18 レジスタ A に代入される引数の型

2	2 バイト整数型
3	文字列型
4	単精度実数型
8	倍精度実数型

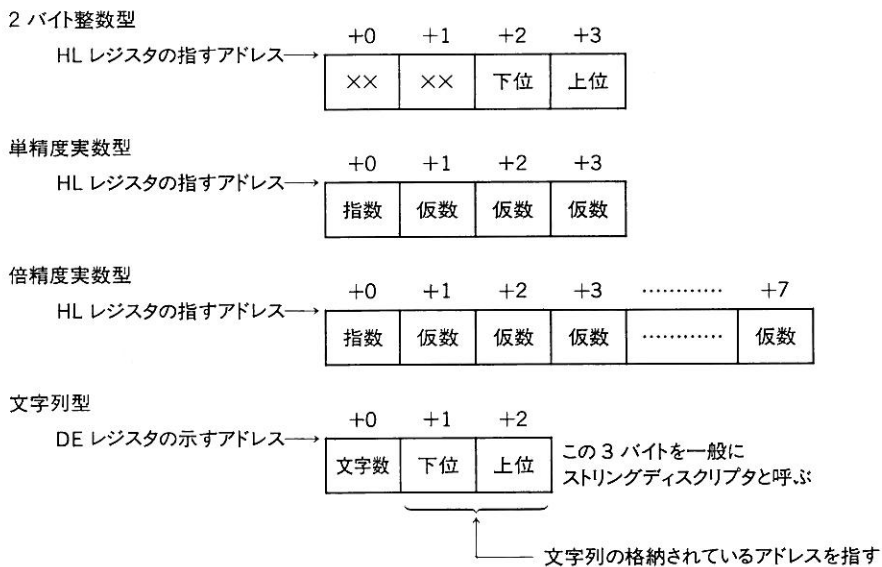


図 2.27 引数による値のわたされ方

リスト 2.5 文字型の引数の使用例

```

:
:      USR 関数で渡した文字列をそのまま表示する
:      使用法：DEF USR =&Hxxxx : A$=USR("STRING")
:
chput equ    00a2h          ; character output
rdarg::
  cp      3
  ret     nz                ; parameter is not string

  push    de
  pop     ix                ; IX := string descriptor
  ld      a,(ix+0)          ; get string length
  ld      l,(ix+1)          ; get string pointer (low)
  ld      h,(ix+2)          ; get string pointer (high)
  or      a
  ret     z                 ; if length = 0
rd1:
  push    af
  ld      a,(hl)            ; get a charcter
  call    chput             ; get a charcter
  pop     af
  dec     a
  ret     z
  inc     hl

  jr      rd1

end

```

逆に、引数として渡された値をマシン語の中で処理することによって、USR 関数の値として BASIC に引き渡すこともできます。この場合、【VALTYP(0F663H)】を変更することにより、戻り値を BASIC からの引数の型以外のものに変更することも可能です。ただし、文字列型の場合、文字数を変えることはできません。

5.3 命令の増設

MSX では現在「CMD」と「IPL」という予約語が未使用となっており、この予約語のフック（それぞれ FE0DH、FE03H）を自分で作成した任意のマシン語ルーチンにジャンプするよう書き換えるだけで、新しい命令を作成することができます。簡単な例をリスト 2.6 に示します。

リスト 2.6 CMD 命令の増設

```

;
;
;      CMD コマンドの使用法(CAPS の ON/OFF)
;      コマンドの初期化： DEF USR =&Hxxxx： A = USR(0)
;      コマンドの使用法： CMD
;
chgcap equ    0132h          ; CAPS LAMP on/off
capst  equ    0fcabh        ; CAPS LOCK status
hcmd   equ    0fe0dh        ; CMD HOOK

; CMD コマンドの増設

start::
    ld      bc, 5           ; NEW HOOK SET
    ld      de,hcmd
    ld      hl,hdat
    ldir
    ret

; HOOK に書き込むデータ

hdat:
    pop    af
    jp    capkey
    nop

; CMD コマンドの実体

capkey:
    call   chgcap
    ld     a,(capst)
    cpl
    ld     (capst),a
    ret

end

```


フックに書き込む最初の「pop af」は、「CMD」の実行時にスタックに積まれるエラー処理アドレスを捨てるためのものです。これを入れておかないと、「RET」命令で BASIC に戻るはずが、エラー処理ルーチンにジャンプしてしまいます。ユーザールーチン内でエラーを出したい場合は、このアドレスを取っておくのも1つの方法です。

なお、これらのフックは本来は将来の拡張用にリザーブしてある場所なので、商用のアプリケーションプログラムで使用してはいけません。

5.4 BASIC の内部ルーチン

より複雑なステートメントの拡張を行うには、CMD 命令にも引数を渡せると便利です。マシン語を呼び出した際、HL レジスタが BASIC テキスト中の「CMD」の次の位置を指していますが、これを利用して後に続く文字列の評価を行えば目的は果たせます。これらを行うために有効な内部ルーチンを以下で説明します。

CHRGTR (4666H / MAIN)

機能

テキストから1文字取り出します。

コール手順

HL テキストを指すアドレス

戻り値

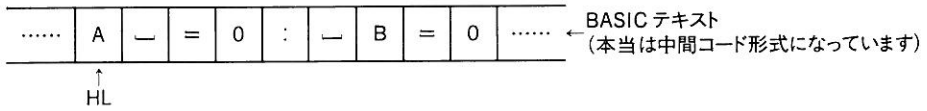
HL 取り出した文字のアドレス
 A 取り出した文字
 Z フラグ 文末（「:」または00H）ならば ON
 CY フラグ 0~9 ならば ON

変更レジスタ

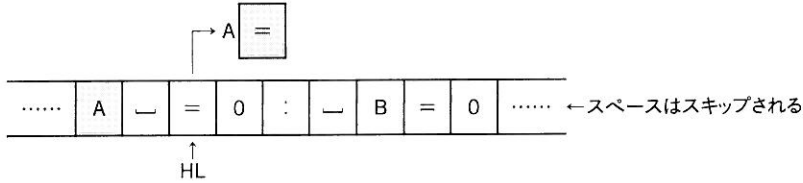
解説

(HL+1) の場所のテキストから1文字取り出します。スペースはスキップされます。

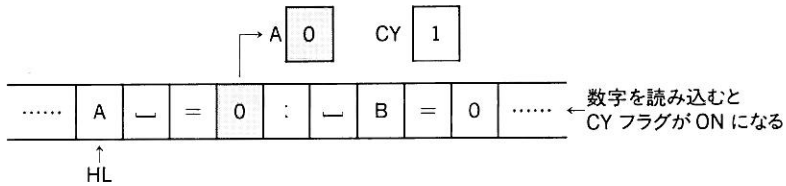
コール前



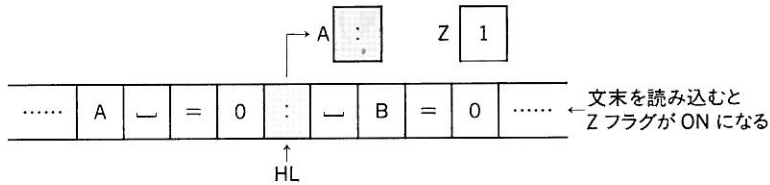
1 度目の実行



2 度目の実行



3 度目の実行



FRMEVL (4C64H / MAIN)

機能

テキスト中の式を評価します。

コール手順

HL テキスト中の式の見出しアドレス

戻り値

HL 式の次のアドレス

【VALTYP(0F663H)】 型に応じて 2、3、4、8 の値が入る

【DAC(0F7F6H)】 式の評価結果

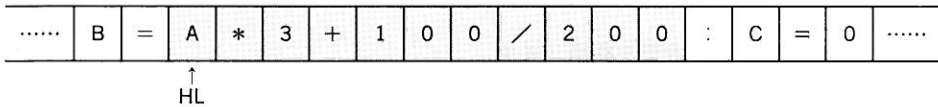
変更レジスタ

すべて

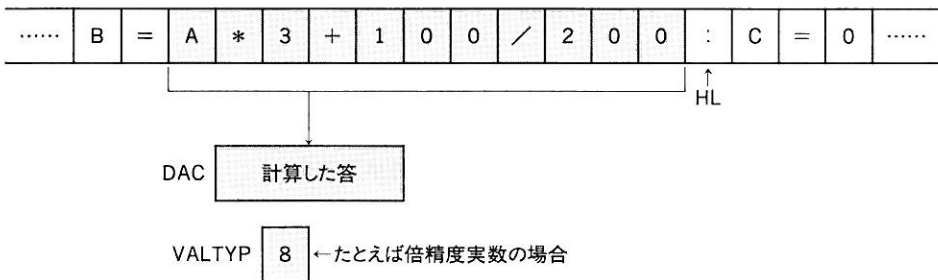
解説

式を評価し、型に応じて出力します「5.2 USR 関数の引数と戻り値によるデータの受け渡し」参照。

コール前



コール後



FRMQNT (542FH / MAIN)

機能

式を2バイト整数型で評価します。

コール手順

HL テキスト中の式の見出しアドレス

戻り値

HL 式の次のアドレス

DE 式の評価結果

変更レジスタ

すべて

解 説

式を評価し、整数型 (INT) で出力します。結果が2バイト整数型の範囲に納まらない場合には、「Overflow」エラーを発生し、BASIC コマンドレベルに戻ります。

GETBYT (521CH / MAIN)

機 能

式を1バイト整数型で評価します。

コール手順

HL テキスト中の式の見出しアドレス

戻り値

HL 式の次のアドレス

A、E 式の評価結果 (A と E には同じ値が入る)

変更レジスタ

すべて

解 説

式を評価し、1バイト型で出力します。結果が1バイト整数型の範囲に収まらない場合には「Illegal function call」エラーを発生し、BASIC コマンドレベルに戻ります。

FRESTR (67DOH / MAIN)

機 能

文字列を登録します。

コール手順

【VALTYP(0F663H)】 型 (文字型でないとエラーになる)

【DAC(0F7F6H)】 スtringディスクリプタへのポインタ

戻り値

HL スtringディスクリプタへのポインタ

変更レジスタ

すべて

解 説

FRMEVL で得た文字列型の結果を文字列領域に登録し、そのSTRINGディスクリプタを得ます。文字列を評価した場合は、FRMEVL と組み合わせて以下のように使用するのが一般的です。

```

      ⋮
call  frmevl
push  hl          ; テキストポインタをセーブする
call  FRESTR
ex    de,hl      ; STRINGディスクリプタを得る
pop   hl
ld    a,(de)    ; 入力された文字列の長さを得る
      ⋮

```

PTRGET (5EA4H / MAIN)

機 能

変数の格納アドレスを獲得します。

コール手順

HL テキスト中の変数名の先頭アドレス
【SUBFLG(0F6A5H)】 0 = 単変数、0 以外 = 配列変数

戻り値

HL 変数名の次のアドレス
DE 目的の変数の内容が格納されているアドレス

変更レジスタ

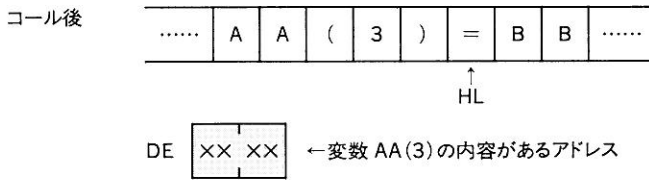
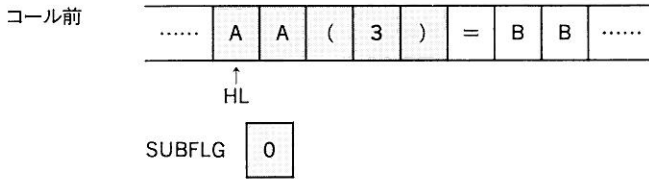
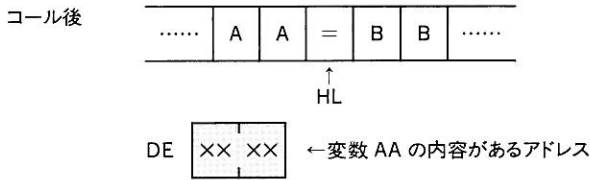
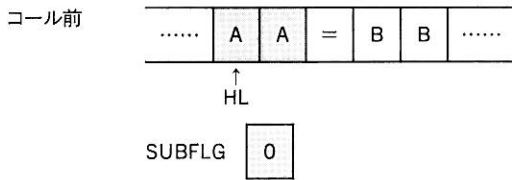
すべて

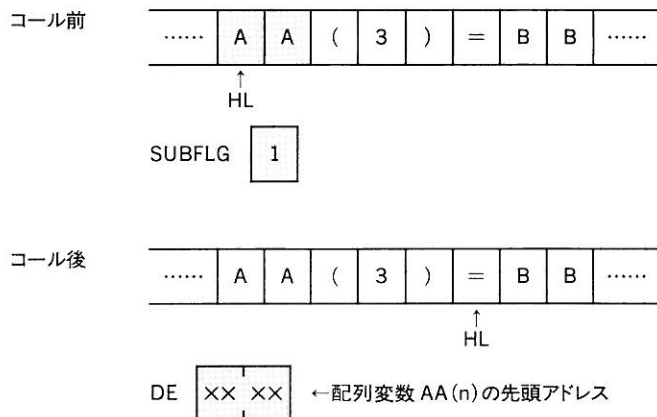
解説

変数（または配列変数）の格納アドレスを得ます。目的の変数の領域が確保されていなかった場合は、領域の確保も行われます。【SUBFLG(0F6A5H)】の値を0以外に設定しておくとは配列の個々の要素ではなく、その配列の先頭アドレスが得られます。

BASICで、USR関数により実行中の機械語プログラムから、BASICの変数(ここではS%)に直接結果を返すには、以下のようにします。

```
vars :
    defd 'S%',0
    ld hl,vars
    call ptrget
```





CRUNCH (42B2H / MAIN)

機能

BASIC のステートメントを中間コードに変換します。

コール手順

HL 00H で終わる ASCII 形式のテキストの先頭アドレス

戻り値

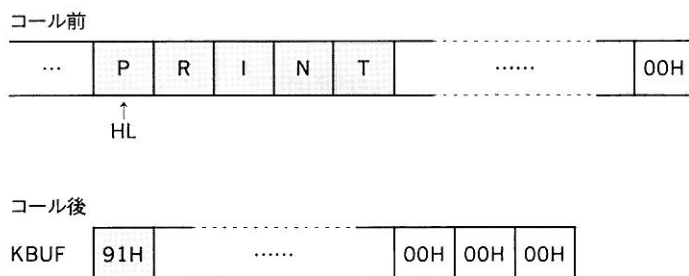
【KBUF(OF41FH)】 中間コード

変更レジスタ

すべて

解説

BASIC のステートメントを中間コードに変換します (表 2.17 参照)。



NEWSTT (4601H / MAIN)

機能

任意のテキストを実行します。

コール手順

HL 実行するテキストのアドレス

戻り値

なし

変更レジスタ

すべて

解説

テキストの実行を行います。テキストは下図の状態となっている必要があります。

実際は、BASIC の内部表現のコードが入る



これらの内部ルーチンは、BASIC テキストを対象とするものであるため、エラーが発生した場合は、BASIC と同様のエラー処理が行われます。その場合、【HERRO (0FFB1H)】の書き換えにより、ユーザーがエラー処理を行うこともできます。その場合は、Eレジスタにエラー番号が入ります。リスト 2.7 を参照してください。

リスト 2.7 エラー処理ルーチンの変更

```

;
;
;      エラー処理ルーチンの変更
;      用法 : DEF USER =&Hxxxx : A = USR(0)
;
herr   equ    0ffb1h      ; error HOOK
synerr equ    2          ; syntax error code
chput  equ    0a2h       ; character output BIOS
newstt equ    04601h     ; run
readyr equ    0409bh

;      エラー処理ルーチンの変更

```



```

start::
    ld    bc,5          ; set new HOOK
    ld    de,herr
    ld    hl,hdat
    ldir
    ret

hdat:
    jp    error
    nop
    nop

; エラー処理ルーチンの実体

error:
    ld    a,e          ; when in error, E holds error code
    cp    synerr       ; syntax error?
    ret   nz           ; no

loop:
    ld    hl,data1     ; yes
    ld    a,(hl)       ; put new error message
    cp    '$'
    jr   z,exit
    push hl
    call chput
    pop  hl
    inc  hl
    jr   loop

exit:
    jp    readyr

data1: defm 'マチガッテマス' ; new error start
       defb 07h,07h,07h,'$

end

```

5.5 拡張ステートメント

4.4 で紹介したのは、「CMD」という未使用の予約語を使って、BASIC の機能を拡張する方法でした。しかし、この方法は予約語を使っている関係で、市販のアプリケーションソフトウェアでは使用できません。ここで紹介する拡張ステートメントは、ROM カートリッジでの使用という条件がつきますが、将来に渡って互換性が保証されている方法ですので、市販のソフトウェアで採用するのに最適です。

拡張ステートメントは CALL 命令で始まり、以下の書式で使います。

```
CALL <拡張ステートメント名>[( <引数> [, <引数> ... ] )]
```

拡張ステートメントは、MSXのカートリッジヘッダーのSTATMENTに拡張ステートメントルーチンのアドレスを書き込むことにより実行できます。その実現方法などについては、「xxx カートリッジヘッダー」をご参照ください。

添付のフロッピーディスクに、BASICの拡張ステートメントのサンプルプログラムが入っています。ファイル名は「`FOO=BAR.MAC`」です。このソースプログラムを、

```
M80 = foo=bar
      EXBASIC
```

でアセンブルして、

```
L80 /P:4000,foo=bar,foo=bar/n/x/e
      EXBASIC
```

でリンクすると、`FOO=BAR.HEX` というファイルができます。

このファイルをPROMライターに転送してROMに書き込み、カートリッジにすると「FOO」と「BAR」という拡張ステートメントが使用できるようになります。「FOO」と「BAR」は以下の機能を持っています。

CALL FOO

書式

CALL FOO(<文字変数>, <文字列>)

機能

<文字列>を大文字に変換して、<文字変数>に代入します。

文例

```
CALL FOO(A$, "This is an example")
```

```
PRINT A$
```

```
表示 THIS IS AN EXAMPLE
```

CALL BAR

書式

CALL BAR(<数値変数>, <数式1>, <数式2>)

機 能

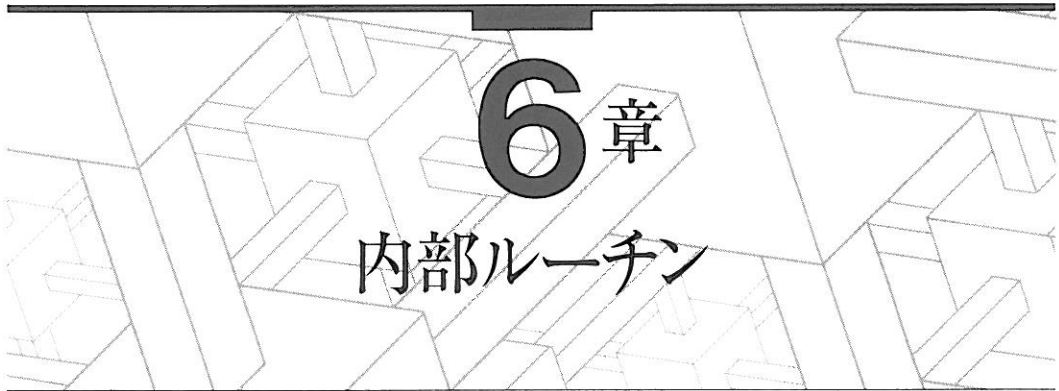
<数式 1>と<数式 2>を足し算して、その結果を<数値変数>に代入します。

文 例

```
CALL BAR(A,100,23)
```

```
PRINT A
```

```
表示 123
```



6.1 PLAY 文 BIOS

6.1.1 概要

MSX は PSG の音楽を割り込みルーチン内で演奏しています。BASIC では PLAY 文が MML (Music Macro Language) を割り込みルーチンが解釈できる形に変換してキューに書き込み、あるフラグをセットします。割り込みルーチンはそのフラグを見て、キューにデータが書かれたことを知り、キューからデータを取り出して演奏します。

演奏はキューから取り出したデータが END MARK のときに終了します。このキューは各チャンネルごとに 1 本ずつで、合計 3 本あります。

6.1.2 キューの構造

まず、【QUEUES(0F3F3H,2)】というワークエリアがあり、これが QCB(Queue Control Block) という構造体の配列の先頭を指しています。QCB 構造体の配列は、通常【QUETAB(0F959H,24)】にあり、要素数は 4 つです。しかし、最後の 1 つは PLAY 文では使用しません。1 つの QCB は次のような構造をしています。

表 2.19 QCB の構造

開始番地 からの オフセット	用途
0	PUT オフセット キューにデータを書き込むときのオフセット
1	GET オフセット キューからデータを読み出すときのオフセット
2	バックアップキャラクタ キューにデータを書き込むときのに一時的に使われる
3	キューの長さ 実際のキューの長さ-1。取り得る値は2のn乗-1(nは2から8)。単位はバイト。通常は127
4、5	キューの開始番地 実際のキューの開始番地。通常、以下のキューを指している チャンネル1 【VOICAQ(0F975H,128)】 チャンネル2 【VOICBQ(0F9F5H,128)】 チャンネル3 【VOICCQ(0FA75H,128)】

6.1.3 キューデータの解説

キューには1回の割り込み処理中に設定されるデータが可変長で入っています。最初の2バイトの上位3ビットはその割り込み処理中に処理されるデータが何バイトであるかを指定します。指定できる範囲は1から5です。残りの13ビットは次のデータの処理をするまでの時間をタイマー割り込みの回数を指定します。この2バイトのデータは上位バイトが先になります。終了はこの2バイトの代わりに、1バイトのFFHが入ります。残りのデータは次の4種類のうちのどれかが入っているかで異なります。

1. 音程を設定するデータ (MMLのAからG、Nに相当する)
2. 音程を設定するデータ (Vに相当する)
3. エンベロープの形状を設定するデータ (Sに相当する)
4. エンベロープの周期を設定するデータ (Mに相当する)

各々の詳細な形式を以下で説明します。PSGに設定するデータの詳細は、「第xx部 x章 PSG」を参照してください。

1. 音程を設定するデータ

音程を設定するデータは2バイトで構成されます。上位2ビットは音程を設定するデータであ

ることを示すために、「00」とします。残りの14ビットでPSGに設定する音色をそのまま指定しますが、PSGでは下位12ビットのみ有効です。上位バイトが先になります。

2. 音量を設定するデータ

音量を設定するデータは1バイトで構成されます。上位4ビットは音量を設定するデータであることを示すために、「1000」とします。残りの4ビットでPSGに設定する音量をそのまま指定します。

3. エンベロープの形状を設定するデータ

エンベロープの形状を設定するデータは1バイトで構成されます。上位4ビットはエンベロープの形状を設定するデータであることを示すために、「1001」とします。残りの4ビットでPSGに設定するエンベロープの形状をそのまま指定します。

4. エンベロープの周期を設定するデータ

エンベロープの周期を設定するデータは3バイトで構成されます。最初の1バイトはエンベロープの周期を設定するデータであることを示すために「40H」とします。残りの2バイトでPSGに設定するエンベロープの周期をそのまま指定します。この2バイトは上位バイトが先になります。

5. エンベロープの形状と周期を同時に設定するデータ

このデータは3バイトで構成されます。最初の1バイトのうち、上位4ビットは「1101」とします。下位4ビットはPSGに設定するエンベロープの形状をそのまま指定します。残りの2バイトでPSGに設定するエンベロープの周期をそのまま指定します。この2バイトは上位バイトが先になります。

6.1.4 データの具体例

MMLで”T120L4V8O4CDE”とした場合（以下16進数）

60,1E	3バイトのデータがあり、0.5秒間続ける
88	音量は8
01,AC	音程は01ACH
60,1E	3バイトのデータがあり、0.5秒間続ける
88	音量は8

01,7D	音程は 017DH
60,1E	3 バイトのデータがあり、0.5 秒間続ける
88	音量は 8
01,53	音程は 0153H
FF	終了

6.1.5 BIOS

BIOS は STRTMS (0099H / MAIN) という名前のルーチンがあります。これと呼ぶことにより、演奏を始めることができます。以下のサンプルプログラムを参照して下さい。

リスト 2.8 STRTMS の使用例

```

        strtms    equ    0099h
        musicf    equ    0fb3fh    ; 音楽演奏用の割り込みフラグ
        plycnt    equ    0fb40h    ; キューされている PLAY 文の数
        quetab    equ    0f959h    ; キューテーブル
        qcblen    equ    6         ; QCB の長さ
        cseg

music:
    ld    a,(musicf)    ; 演奏中?
    or    a
    jr    nz,music     ; 演奏中なら、終わるまで待つ
    di
    ld    de,quetab
    ld    hl,queini
    ld    bc,qcblen * 3
    ldir                                ; QCB の設定
    ld    a,1
    ld    (plycnt),a
    call strtms        ; 演奏開始
    ei
    ret

queini:
    db    0ffh
    db    0
    db    0
    db    0ffh
    dw    queue1
    db    0ffh
    db    0
    db    0
    db    0ffh
    dw    queue2
    db    0ffh
    db    0
    db    0
    db    0ffh

```

```
        dw      queue3
;
;
; "T120L4V804CDE"
;
queue1:
db      0          ; a dummy
db      60h,1eh,88h,01h,0ach ; T120L4V8C
db      60h,1eh,88h,01h,7dh  ; T120L4V8D
db      60h,1eh,88h,01h,53h  ; T120L4V8E
db      0ffh        ; end mark
queue2:
db      0
db      0ffh
queue3:
db      0
db      0ffh

end
```


6.2 ビットブロックトランスファ

ビットブロックトランスファとは、RAM、VRAM、ディスク間でデータを転送する SUB ROM 内ルーチンです。本来、MSX2 で拡張された BASIC の COPY 命令用のルーチンですが、完全なサブルーチンの形式になっているので、BIOS のようにユーザープログラムから利用できます。

なお、ビットブロックトランスファルーチンが使用可能な画像データは、BASIC の COPY 命令で作成されたデータです。BSAVE "ファイル名",&Hxxxx,&Hxxxx,S で作成されたデータは使用できません。

X 方向の サイズ	Y 方向の サイズ	VRAM データ
--------------	--------------	----------

図 2.28 COPY 命令で作成した画像ファイルのヘッダ情報

例

COPY (0,0)-(10,20) TO "TEST.PIC"

で作成した TEST.PIC は、0BH、00H、15H、00H、.... となる。

6.2.1 VRAM 間の転送

BLTVV(0191H / SUB)

機能

VRAM 領域内で転送を行います。

コール手順

HL 0F562H

さらに以下のパラメータを設定する。

SX(0F562H,2)	;	転送元基準点 X 座標
SY(0F564H,2)	;	転送元基準点 Y 座標
DX(0F566H,2)	;	転送先基準点 X 座標
DY(0F568H,2)	;	転送先基準点 Y 座標
NX(0F56AH,2)	;	X 方向転送ドット数
NY(0F56CH,2)	;	Y 方向転送ドット数
CDUMMY(0F56EH,1)	;	ダミー (セットする必要はない)
ARG(0F56FH,1)	;	方向と拡張 RAM の選択 (VDP の R # 45 と同じ)
LOGOP(0F570H,1)	;	ロジカルオペレーションコード

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

; COPY (0,0)-(255,99) TO (0,100)

```
bltvv equ 0191h
extrom equ 015fh
ld hl,0
ld (sx),hl ; 転送元基準点は (0,0)
ld (sy),hl
ld (dx),hl ; 転送先基準点は (0,100)
ld l,100
ld (dy),hl
ld hl,255-0+1 ; Y 方向転送ドット数の設定
```

```

ld      (nx),hl
ld      hl,99-0+1      ; X 方向転送ドット数の設定
ld      (ny),hl
xor     a
ld      (arg),a
ld      (Lop),a
        ; operation is 'replace'
        ;
ld      hl,0f562h
ld      ix,bltvv
call   extrom
        ;
        ;

```

6.2.2 RAM と VRAM 間のデータ転送

以下のルーチンを使う場合、グラフィックのデータ領域として、スクリーンモードによって次のような大きさの領域を確保しておく必要があります。

表 2.20 スクリーンモード別確保領域の算出法

スクリーンモード	領域の大きさ
6	X 方向のドット数×Y 方向のドット数÷4+4
5、7	X 方向のドット数×Y 方向のドット数÷2+4
8	X 方向のドット数×Y 方向のドット数+4

注意 割り算の端数は切り上げて下さい。

なお、ディスクや RAM では配列データと同様にサイズを示すデータが付加されます。このデータは先頭から 2 バイトが X 方向のドット数、次の 2 バイトが Y 方向のドット数を示しています。

BLTVM(0195H / SUB)

機能

配列を VRAM に転送します。

コール手順

HL 0F562H

さらに以下のパラメータを設定する。

DPTR(0F562H,2)	;	メモリの転送元アドレス
DUMMY(0F564H,2)	;	ダミー (セットする必要はない。)
DX(0F566H,2)	;	転送先基準点 X 座標
DY(0F568H,2)	;	転送先基準点 Y 座標
NX(0F56AH,2)	;	X 方向転送ドット数 (セットする必要はない)
NY(0F56CH,2)	;	Y 方向転送ドット数 (セットする必要はない)
CDUMMY(0F56EH,1)	;	ダミー (セットする必要はない)
ARG(0F56FH,1)	;	方向と拡張 RAM の選択 (VDP の R # 45 と同じ)
LOGOP(0F570H,1)	;	ロジカルオペレーションコード

戻り値

転送するデータの個数がおかしければ CY フラグをセット

変更レジスタ

すべて

例

; COPY <B000H> TO (5,6)

```

bltvm   equ   0195h
extrom  equ   015fh
        ld    hl,0b000h
        ld    (dptr),hl      ; データは B000H にある
        ld    hl,5
        ld    (dx),hl        ; 転送先基準点は (5,6)
        inc  hl
        ld    (dy),hl
        xor  a                ; 方向の設定
        ld    (arg),a
        ld    (l_op),a       ; operation is 'replace'
        ;
        ld    hl,0f562h
        ld    ix,bltvm
        call extrom
        ;
        ;
        ;

```

BLTMV(0199H / SUB)

機能

VRAM から配列に転送します。

コール手順

HL 0F562H

さらに以下のパラメータを設定する

SX(0F562H,2)	;	転送元基準点 X座標
SY(0F564H,2)	;	転送元基準点 Y座標
DPTR(0F566H,2)	;	メモリの転送元アドレス
DUMMY(0F568H,2)	;	ダミー (セットする必要はない)
NX(0F56AH,2)	;	X方向転送ドット数
NY(0F56CH,2)	;	Y方向転送ドット数
CDUMMY(0F56EH,1)	;	ダミー (セットする必要はない)
ARG(0F56FH,1)	;	方向と拡張RAMの選択 (VDPのR#45と同じ)

戻り値

CYフラグをリセット

変更レジスタ

すべて

例

; COPY (10,20)-(50,25) TO <B000H>

```
bltmv equ 0199h
extrom equ 015fh
ld hl,0b000h
ld (dptr),hl ; データは B000H にある
ld hl,10
ld (sx),hl ; 転送基準点は (10,20)
ld hl,20
ld (sy),hl
ld hl,50-10+1 ; X方向転送ドット数の設定
ld (nx),hl
ld hl,25-20+1 ; Y方向転送ドット数の設定
ld (ny),hl
xor a ; 方向の設定
ld (arg),a
;
ld hl,0f562h
ld ix,bltmv
```

```

call    extrom
      ⋮

```

6.2.3 ディスクとRAM、VRAM 間の転送

ディスクを使う場合、まずファイル名を設定する必要があります(ファイル名は BASIC と同様に指定)。例を次に示します。

```

      ⋮
ld     hl, fname      ; ファイル名へのポインタの獲得
ld     (fnptr), hl   ; パラメータエリアに設定
      ⋮

```

```
fname : db 22h,'b:test.pic', 22h, 0 ; "TEST.PIC", end mark
```

ここで、何かエラーが起きれば、BASIC インタープリタのエラーハンドラーへジャンプします。そのため、自分のプログラム内でエラーをハンドリングしたい場合や MSX-DOS、ROM カートリッジでこのルーチンを呼ぶ場合、フックをセットする必要があります。このフックは【H. ERRO(0FFB1H)】です。

BLTVD(019DH / SUB)

機能

ディスクから VRAM へ転送します。

コール手順

HL 0F562H

さらに以下のパラメータを設定する

FNPTR(0F562H,2)	; ファイル名のあるアドレス
DUMMY(0F564H,2)	; ダミー (セットする必要はない)
DX(0F566H,2)	; 転送先基準点 X 座標
DY(0F568H,2)	; 転送先基準点 Y 座標
NX(0F56AH,2)	; X 方向転送ドット数 (セットする必要はない)
NY(0F56CH,2)	; Y 方向転送ドット数 (セットする必要はない)

CDUMMY(0F56EH,1) ; ダミー (セットする必要はない)
 ARG(0F56FH,1) ; 方向と拡張 RAM の選択 (VDP の R # 45 と同じ)

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

; COPY "TEST.PIC" TO (3,2)

```
bltvd equ 019dh
extrom equ 015fh
ld hl,filename ; ファイル名へのポインタの設定
ld (fnptr),hl
ld hl,3
ld (dx),hl ; 転送先基準点は(3,2)
dec hl
ld (dy),hl
xor a ; 方向の設定
ld (arg),a
ld (l_op),a ; operation is 'replace'
;
ld hl,0f562h
ld ix,bltvd
call extrom
;
;
filename: db 22h,'test.pic',22h,0
;
;
```

BLTDV(01A1H / SUB)

機能

VRAM からディスクへ転送します。

コール手順

HL 0F562H

さらに以下のパラメータを設定する

FNPTR(0F562H,2) ; ファイル名のあるアドレス

DUMMY(0F564H,2) ; ダミー (セットする必要はない)
 DX(0F566H,2) ; 転送先基準点 X座標
 DY(0F568H,2) ; 転送先基準点 Y座標
 NX(0F56AH,2) ; X方向転送ドット数
 NY(0F56CH,2) ; Y方向転送ドット数
 CDUMMY(0F56EH,1) ; ダミー (セットする必要はない)
 ARG(0F56FH,1) ; 方向と拡張RAMの選択 (VDPのR#45と同じ)

戻り値

CYフラグをリセット

変更レジスタ

すべて

例

; COPY (10,20)-(50,25) TO 'A:TEST.PIC'

```

bltdv equ 01a1h
extrom equ 015fh
ld hl,filename
ld (fnptr),hl ; ファイル名へのポインタの設定
ld hl,10
ld (sx),hl ; 転送先基準点は(10,20)
ld hl,20
ld (sy),hl
ld hl,50-10+1 ; X方向転送ドット数の設定
ld (nx),hl
ld hl,25-20+1 ; Y方向転送ドット数の設定
ld (ny),hl
xor a ; 方向の設定
ld (arg),a
;
ld hl,0f562h
ld ix,bltdv
call extrom
;
;
filename:db 22h,'a:test.pic',22h,0
;
;
    
```


BLTMD(01A5H / SUB)

機能

ディスクから配列データをロードします。

コール手順

HL 0F562H

さらに以下のパラメータを設定する

FNPTR(0F562H,2)	;	ファイル名のあるアドレス
SY(0F564H,2)	;	ダミー (セットする必要はない)
SPTR(0F566H,2)	;	ロードする先頭アドレス
EPTR(0F568H,2)	;	ロードする最終アドレス

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

```

bltmd equ 01a5h
extrom equ 015fh
      ld hl,filename ; ファイル名へのポインタの設定
      ld (fnptr),hl
      ld hl,8000h ; 先頭アドレスの設定
      ld (sptr),hl
      ld hl,0affh ; 最終アドレスの設定
      ld (eptr),hl
      ;
      ld hl,0f562h
      ld ix,bltmd
      call extrom
      ;
      ;
filename:db 22h,'test.bin',22h,0
      ;
      ;

```

BLTDM(01A9H / SUB)

機能

ディスクへ配列データをセーブします。

コール手順

HL 0F562H

さらに以下のパラメータを設定する

- SPTR(0F562H,2) ; セーブする先頭アドレス
- EPTR(0F564H,2) ; セーブする最終アドレス
- FNPTR(0F566H,2) ; ファイル名のあるアドレス

戻り値

CY フラグをリセット

変更レジスタ

すべて

例

```

bltdm   equ    01a9h
extrom  equ    015fh
        ld     hl,filename      ; ファイル名へのポインタの設定
        ld     (fnptr),hl
        ld     hl,8000h        ; 先頭アドレスの設定
        ld     (sptr),hl
        ld     hl,0affh        ; 最終アドレスの設定
        ld     (eptr),hl
        ;
        ld     hl,0f562h
        ld     ix,bltdm
        call  extrom
        ;
        ;
filename:db 22h,'a:test.bin',22h,0
        ;
        ;
    
```

6.2.4 ビットブロックトランスファの利用法

1. BASIC の環境から利用する場合

定められたワークエリアに値を入れて、HL レジスタが 0F562H を指すようにセットし、SUB ROM 内のアドレスをコールします。

以下のサンプルプログラムは、「MSX-DOS TOOLS」を使用し、BASIC の環境からマシン語のプログラムをメモリ上にロードし、実行するものです。

リスト 2.9 BASIC 環境でのビットブロックトランスファ使用例

```

:
:
:          BLTVV.MAC  sample program
:
bltvv    equ    0191h
sx       equ    0f562h
sy       equ    0f564h
dx       equ    0f566h
dy       equ    0f568h
nx       equ    0f56ah
ny       equ    0f56ch
cdummy  equ    0f56eh
arg      equ    0f56fh
logop    equ    0f570h
extrom   equ    015fh
start::

        ld      de,0
        ld      (sx),de      ; 転送元基準点は (0,0)
        ld      (sy),de
        ld      (dx),de      ; 転送先基準点は (0,100)
        ld      de,100
        ld      (dy),de
        ld      (nx),de      ; X 方向転送ドット数を 100 に設定
        ld      (ny),de      ; Y 方向転送ドット数を 100 に設定
        ld      a,00000000b   ; VRAM に対してコピーする
        ld      (arg),a      ; 方向は、X 方向が右、Y 方向が下
        ld      (logop),a    ; ロジカルオペレーションは IMP
        ld      hl,sx        ; hl レジスタに F562H をセット
        ld      ix,bltvv
        call   extrom        ; SUB ROM コール
        ret

        end

```

以下のバッチファイルは、上記のアセンブラプログラムを BASIC から BLOAD するバイナリプログラムにするものです。

```
m80 = bltvv
l80 /p:c000,bltvv,bltvv/n/x/e:start
bsave bltvv.hex > bltvv.bin
```

こうして作成したバイナリファイルを実行するには、以下の BASIC プログラムを使用します。

```
10 CLEAR 300,&HBFFF
20 BLOAD"bltvv.bin"
30 SCREEN 8:COPY "sample.pic" TO (0,0)
40 DEFUSRO =&HC000
50 A = USR(0)
60 A$= INKEY$:IF A$="" THEN GOTO 60
```

2. MSX-DOS の環境から利用する場合

ビットブロックトランスファルーチンは、本来 BASIC の COPY 命令で使用するためのものですから、BASIC 環境での動作を前提に作られています。したがって、MSX-DOS のプログラムからビットブロックトランスファを利用する場合には、BASIC 環境からアクセスする手順に加えて、以下の手続きが必要になります。

- 1) ビットブロックトランスファルーチンは、SP と【STREND(0F6C6H)】が指す番地までをディスクアクセス時のバッファとして使用するので、STREND に値を設定しなければならない。
- 2) ディスクをアクセスするルーチン使用時に設定する FNAME の内容は、ページ 2 かページ 3 になければならない。

リスト 2.10 DOS 環境でのビットブロックトランスファ使用例

```

;
;
; BLTVD sample program
;
strend equ 0f6c6h
bltvd equ 019dh
calslt equ 001ch
exptbl equ 0fcc1h
exbrsa equ 0faf8h
fnptr equ 0f562h
dx equ 0f566h
dy equ 0f568h
arg equ 0f56fh
l_op equ 0f570h
chgmod equ 005fh
extrn _calsub
;
start:
ld sp,(6) ; スタックを設定
```

```

ld    hl,0c000h    ; STREND を C000H に設定
ld    (strend),hl
ld    bc,11-fnamelen ; FNAME の内容を B000H からにブロック転送
ld    de,0b000h
ld    hl,fname
ldir
ld    a,8          ; スクリーン 8
ld    iy,(exptbl-1)
ld    ix,chgmod
call  calslt
ld    hl,0b000h    ; ブロック転送したアドレスを設定
ld    (fnptr),hl
ld    de,0
ld    (dx),de
ld    (dy),de
xor   a

ld    (arg),a
ld    (L_op),a
ld    hl,0f562h
ld    ix,bltvd
call  _calsub

ld    a,0          ; スクリーンをテキストモードに戻す
ld    iy,(exptbl-1)
ld    ix,chgmod
call  calslt
jp    0

;
fname: db    22h,'test.pic',22h,0
-fnamelen equ $-fname
end

```

この方法は互換性を損なう可能性が大きいので
 高品質アプリケーションプログラムでは使用しない方がいい

6.3 Math-Pack

Math-Pack とは、MSX BASIC の数値演算ルーチンの中核をなすもので、これらをマシン語から呼び出すことにより、簡単に浮動小数点演算や三角関数ルーチンを利用することができます。

Math-Pack での実数演算はすべて BCD (Binary Coded Decimal = 2 進化 10 進法) 表現によって行われます。また、実数表現には、「単精度」と「倍精度」の 2 種類があり、単精度 (6 桁) は 4 バイト、倍精度 (14 桁) は 8 バイトで表現されます。(図 2.29、図 2.30)

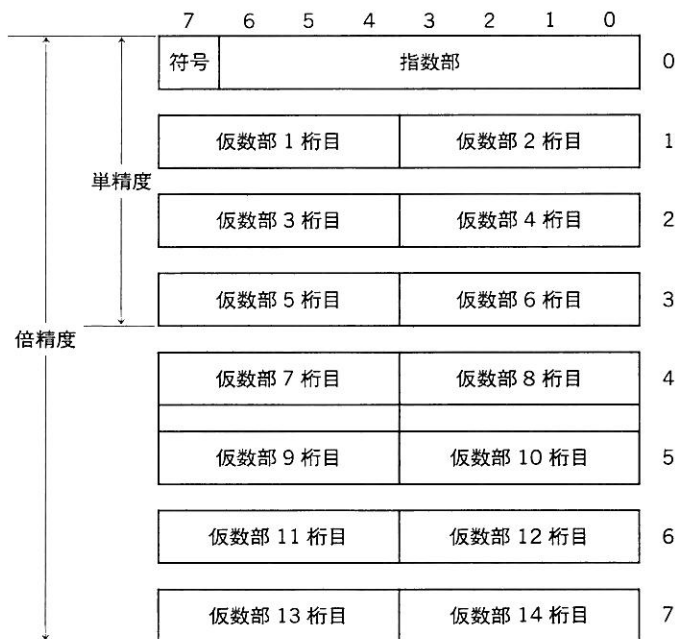


図 2.29 BCD による実数表現のフォーマット

単精度表現の例

123456 → 0.123456 E+6

	0	1	2	3
DAC	46	12	34	56

倍精度表現の例

123456.78901234 → 0.12345678901234 E+6

	0	1	2	3	4	5	6	7
DAC	46	12	34	56	78	90	12	34

図 2.30 実数の表現例

1つの実数は、符号部、指数部、仮数部から成り立っています。符号部とはこの仮数部の符号を表し、0ならば正、1ならば負となります。指数部はバイナリ表現であり、+63乗～-63乗まで表すことができます（図 2.31）。倍精度実数の有効範囲を図 2.32 に示します。

符号	指数部							意味
0	0	0	0	0	0	0	0 0
1	0	0	0	0	0	0	0 未定義(-0?)
×	0	0	0	0	0	0	1 10の-63乗
×	1	0	0	0	0	0	0 10の0乗
×	1	1	1	1	1	1	1 10の+63乗

注意 「×」はその値が1または0のどちらであっても関係がないことを表します。

図 2.31 指数部フォーマット

	0	1	2	3	4	5	6	7	(バイト)
DAC	FF	99	99	99	99	99	99	99	-0.9999999999999999 E+63
				⋮					⋮
	81	10	00	00	00	00	00	00	-0.1000000000000000 E-63
	00	×	×	×	×	×	×	×	0
	01	10	00	00	00	00	00	00	+0.1000000000000000 E-63
				⋮					⋮
DAC	7F	99	99	99	99	99	99	99	+0.9999999999999999 E+63

図 2.32 倍精度実数表現の有効範囲

Math-Pack では、演算の対象となるメモリがあらかじめ決められています。このメモリエリアを【DAC(Decimal ACcumlator - 0F7F6H)】と呼び、DAC との演算対象となる数値を格納するエリアを【ARG(0F847H)】といいます。例えば、乗算ルーチンの場合、DAC に入っている数と ARG に入っている数の積が計算され、その結果が DAC に入ることになります。

DAC には、単精度実数、倍精度実数、2 バイト整数を格納することができますが、これらのうちのどれが入っているのかを示すために【VALTYP(0F663H)】が使われ、それぞれの場合について 4、8、2 の値が入ります。

単精度実数と倍精度実数は、いずれも DAC の先頭から数値を格納しなければなりません。ただし、2 バイト整数の場合だけは、下位、上位を DAC+2、+3 に格納してください。

演算の対象が明記されていない演算は【VALTYP(0F663H)】によって値の型を参照するので、呼び出す前に正しい型を設定して下さい。

Math-Pack は BASIC の内部ルーチンであるため、エラーが起きた場合 (0 除算やオーバーフローなど) はそれに応じたエラールーチンへ自動的に分岐し、その後 BASIC のコマンドレベルに戻ります。それを避けたいときは、【H.ERR0(0FFB1H)】を書き換えて、アプリケーションプログラム側のエラー処理ルーチンへジャンプするようにして下さい。

6.3.1 Math-Pack ワークエリア

ラベル	アドレス	サイズ	意味
VALTYP	F663H	1	DACに入っている数の形式を表す
DAC	F7F6H	16	BCD形式の浮動小数点アキュムレータ
ARG	F847H	16	DACの演算対象

6.3.2 Math-Pack エントリ

1. 基本演算

ラベル	アドレス	意味	変化するレジスタ
DECSUB	268CH	DAC ← DAC-ARG	すべて
DECADD	269AH	DAC ← DAC+ARG	すべて
DECNRM	26FAH	DACを正規化する*1	すべて
DECROU*2	273CH	DACを四捨五入する	すべて
DECMUL	27E6H	DAC ← DAC×ARG	すべて
DECDIV	289FH	DAC ← DAC÷ARG	すべて

注意

これらは、いずれも DAC、ARG にある数を倍精度実数として扱う。

*1 仮数部の不要な 0 を取り去ること (0.00123 → 0.123E-2)

*2 DECROU は倍精度の値の最終桁+1 (15 桁目) に対して演算を行う。

<演算の例>

1.23456789012345 → DECROU → 1.2345678901235

1.11111111111114 → DECROU → 1.1111111111111

1.11111111111115 → DECROU → 1.1111111111112

1.99999999999999 → DECROU → 2.0

2. 関数 1

ラベル	アドレス	意 味	変化するレジスタ
COS	2993H	DAC ← COS(DAC)	すべて
SIN	29ACH	DAC ← SIN(DAC)	すべて
TAN	29FBH	DAC ← TAN(DAC)	すべて
ATN	2A14H	DAC ← ATN(DAC)	すべて
LOG	2A72H	DAC ← LOG(DAC)	すべて
SQR	2AFFH	DAC ← SQR(DAC)	すべて
EXP	2B4AH	DAC ← EXP(DAC)	すべて
RND	2BDFH	DAC ← RND(DAC)	すべて
INT	30CFH	DAC ← INT(DAC)	すべて

注 意

すべて、BASIC における同名の関数の処理ルーチン。
 変化するレジスタすべてとは、A、B、C、D、E、H、L の各レジスタを指す。

3. 関数 2

ラベル	アドレス	意 味	変化するレジスタ
SIGN	2E71H	A ← DAC の符号	A
ABSFN	2E82H	DAC ← ABS(DAC)	すべて
NEG	2E8DH	DAC ← NEG(DAC)	A、H、L
SGN	2E97H	DAC ← SGN(DAC)	A、H、L

4. 移動

ラベル	アドレス	意 味	対 象	変化するレジスタ
MAF	2C4DH	ARG ← DAC	倍精度	A、B、D、E、H、L
MAM	2C50H	ARG ← [HL]	倍精度	A、B、D、E、H、L
MOV8DH	2C53H	[DE] ← [HL]	倍精度	A、B、D、E、H、L
MFA	2C59H	DAC ← ARG	倍精度	A、B、D、E、H、L
MFM	2C5CH	DAC ← [HL]	倍精度	A、B、D、E、H、L
MMF	2C67H	[HL] ← DAC	倍精度	A、B、D、E、H、L
MOV8HD	2C6AH	[HL] ← [DE]	倍精度	A、B、D、E、H、L
XTF	2C6FH	[SP] ↔ DAC	倍精度	A、B、D、E、H、L
PHA	2CC7H	ARG → [SP]	倍精度	A、B、D、E、H、L

ラベル	アドレス	意味	対象	変化するレジスタ
PHF	2CCCH	DAC → [SP]	倍精度	A、B、D、E、H、L
PPA	2CDCH	[SP] → ARG	倍精度	A、B、D、E、H、L
PPF	2CE1H	[SP] → DAC	倍精度	A、B、D、E、H、L
PUSHF	2EB1H	DAC → [SP]	単精度	D、E
MOVFM	2EBEH	DAC ← [HL]	単精度	B、C、D、E、H、L
MOVFR	2EC1H	DAC ← (CBED)	単精度	D、E、
MOVRF	2ECCH	(CBED) ← DAC	単精度	B、C、D、E、H、L
MOVRMI	2ED6H	(CBED) ← [HL]	単精度	B、C、D、E、H、L
MOVVM	2EDFH	(BCDE) ← [HL]	単精度	B、C、D、E、H、L
MOVVMF	2EE8H	[HL] ← DAC	単精度	A、B、D、E、H、L
MOVE	2EEBH	[HL] ← [DE]	単精度	B、C、D、E、H、L
VMOVAM	2EEFH	ARG ← [HL]	VALTYP	B、C、D、E、H、L
MOVVFM	2EF2H	[DE] ← [HL]	VALTYP	B、C、D、E、H、L
VMOVE	2EF3H	[HL] ← [DE]	VALTYP	B、C、D、E、H、L
VMOVFA	2F05H	DAC ← ARG	VALTYP	B、C、D、E、H、L
VMOVFM	2F08H	DAC ← [HL]	VALTYP	B、C、D、E、H、L
VMOVAF	2F0DH	ARG ← DAC	VALTYP	B、C、D、E、H、L
VMOVFMF	2F10H	[HL] ← DAC	VALTYP	B、C、D、E、H、L

注意

[HL]、[DE]は、それぞれ HL、DE レジスタが指したメモリ上の数値を示す。

() 内の 4 レジスタ名は、左から (符号+指数部)、(仮数部 1、2 桁)、(仮数部 3、4 桁)、(仮数部 5、6 桁) を示す単精度実数である。

移動の対象が VALTYP となっているものは、【VALTYP(F663H)】に示されている型に応じた移動 (2、4、8 バイト) を行う。

5. 比較

ラベル	アドレス	対象	左辺	右辺	変化するレジスタ
FCOMP	2F21H	単精度実数	CBED	DAC	HL
ICOMP	2F4DH	2 バイト整数	DE	HL	HL
XDCOMP	2F5CH	倍精度実数	ARG	DAC	すべて

注意

結果はすべて A レジスタにはいる。レジスタの意味は、

A = 1 → 左辺 < 右辺

A = 0 → 左辺 = 右辺

A = -1 → 左辺 > 右辺

単精度の比較で、左辺の CBED とは各レジスタに、それぞれ単精度の(符号+指数部)、(仮数部 1、2 桁)、(仮数部 3、4 桁)、(仮数部 5、6 桁)が入ることを意味する。

6. 浮動小数点入出力

ラベル	アドレス	機能
FIN	3299H	浮動小数点を表す文字列を実数表現に変換して、DAC に格納する。

コール手順

HL 文字列の先頭アドレス

戻り値

DAC 実数
 C FFH =小数点なし、0 =小数点あり
 B 小数点以降の桁数
 D 総桁数

ラベル	アドレス	機能
FOUT	3225H	DAC に入っている実数表現の数を文字列に変換する(フォーマット指定なし)
PUFOUT	3426H	DAC に入っている実数表現の数を文字列に変換する(フォーマット指定あり)

コール手順

A 変換のフォーマット
 bit7 0 =フォーマット指定なし、1 =フォーマット指定あり
 bit6 0 =カンマ (,) による区切りなし、1 =3 ごとにカンマ (,) で区切る
 bit5 0 =意味なし、1 =先頭のスペースを「*」で埋める
 bit4 0 =意味なし、1 =数値の前に「\$」をつける
 bit3 0 =意味なし、1 =正の場合も符号「+」をつける
 bit2 0 =意味なし、1 =数値の後ろに符号をつける
 bit1 未使用
 bit0 0 =固定小数点、1 =浮動小数点
 B 小数点を含まない、小数点以前の桁数
 C 小数点を含む、小数点以降の桁数

戻り値

HL 文字列の先頭アドレス

ラベル	アドレス	機能
FOUTB	371AH	DAC+2,+3に入っている2バイト整数を2進表現の文字列に変換する
FOUTO	371EH	DAC+2,+3に入っている2バイト整数を8進表現の文字列に変換する
FOUTH	3722H	DAC+2,+3に入っている2バイト整数を16進表現の文字列に変換する

コール手順

DAC+2 2バイト整数

VALTYP 2

戻り値

HL 文字列の先頭アドレス

注意

レジスタはいずれも保存されない。

出力ルーチンにおける文字列先頭アドレスは、通常、【FBUFFR(F7C5H~)】内となるが、場合により先頭アドレスは前後する。

FOUTB、FOUTO、FOUTHに限らずDAC+2に入れた整数を扱うときには、【VALTYP(F663H番地)】に必ず「2」を入れなければならない。

7. 型変換

ラベル	アドレス	機能
FRCINT	2F8AH	DACを2バイト整数にする (DAC+2,+3)
FRCSNG	2FB2H	DACを単精度実数にする
FRCDL	303AH	DACを倍精度実数にする
FIXER	30BEH	$DAC \leftarrow SGN(DAC) \times INT(ABS(DAC))$

注意

実行後、【VALTYP(F663H)】にはDACの型を表す数(2、4、8)が入る。

レジスタはいずれも保存されない。

8. 整数演算

ラベル	アドレス	機能	変化するレジスタ
UMULT	314AH	DE ← BC×DE	A、B、C、D、E
ISUB	3167H	HL ← DE-HL	すべて
IADD	3172H	HL ← DE+HL	すべて
IMULT	3193H	HL ← DE×HL	すべて
IDIV	31E6H	HL ← DE÷HL	すべて
IMOD	323AH	HL ← DE mod HL (DE ← DE/HL)	すべて

注意 ISUB、IADD、IMULT、IDIV、IMOD については、演算結果は DAC+2 にも入る。

9. べき乗

ラベル	アドレス	機能	基数	指数	結果
SNGEXP	37C8H	単精度実数のべき乗を求める	DAC	ARG	DAC
DBLEXP	37D7H	倍精度実数のべき乗を求める	DAC	ARG	DAC
INTEXP	383FH	2バイト整数のべき乗を求める	DE	HL	DAC

注意 レジスタはいずれも保存されない。

6.3.3 Math-Pack を MSX-DOS 上のプログラムから使用方法

Math-Pack はその本体の大部分は Page 0 にありますが、一部のルーチンが Page 1 にあるために、MSX-DOS からコールした場合に正しい計算ができません。

それを回避するためには、以下のようにします。

リスト 2.11 Math-Pack を DOS 環境で使用方法

```

406fh、4666h、4eb8h、5439h にそれぞれ以下のルーチンを置く。
calbas    equ    0159h
:
:         at 406fh
:
:         ld ix,406f
:         jp calbas
:
:         at 4666h

```

```

;
;
;       ld ix,4666
;       jp calbas
;
;
;       at 4eb8h
;
;
;       ld ix,4eb8
;       jp calbas
;
;
;       at 5439h
;
;
;       ld ix,5439h
;       jp calbas

```

5597h に以下のルーチンを置く。

```

;
;
;       at 5597h
;
;
; getypr:
;       ld     a,(valtyp)
;       cp     8
;       jr     nc,double
;       sub    3
;       or     a
;       scf
;       ret
;
;
; double:
;       sub    3
;       or     a
;       ret

```

66a7h に以下のルーチンを置く。

```

;
;
;       at 66a7h
;
;
; ppswrt:
;       pop   af
;       ret

```

6.4 MSX 漢字ドライバ

MSX 漢字ドライバは、BASIC や DOS で統一的に日本語入出力を行うためのシステムソフトウェアです。

6.4.1 特長

MSX 漢字ドライバには以下のような特長があります。

1. BIOS を使って文字の入出力をしているソフトウェアであれば、プログラムを変更しないで漢字の入出力ができる。
2. テキスト画面で、文字ごとに色を変えることができる。
3. JIS 第二水準の漢字をサポートしている。
4. 漢字プリンタへの漢字出力をサポートしている。
5. MSX-JE (MSX 日本語フロントエンドプロセッサ) がなくても、単漢字変換機能で漢字の入力ができる。
6. グラフィック画面に対し、LOCATE 文や PRINT 文で文字の表示ができる。

6.4.2 注意点

1. PRINT 文

MSX 漢字ドライバでは、グラフィック画面に対して、PRINT 文を使って文字を表示することができますが、以下の点を注意しなければなりません。

1. LOCATE 文で指定するカーソルポジションは、半角の文字単位です。
2. カーソルポジションを保存しているワークエリアはひとつしかないので、SET PAGE 命令でアクティブページを変更しながら PRINT すると、他のページで変更したカーソルポジションの続きに文字が表示されます。
3. インターレスモード (KANJI2、KANJI3) で文字を表示する際には、SCREEN 文で EVNE/ODD のインターレスモード (SCREEN ,,,,3) に設定して、SET PAGE 文で表示ページを奇数ページに、アクティブページを表示ページ-1 に設定しなければなりません。

2. SCREEN 文

従来は、SCREEN モードを変更しただけでは、スプライト表示禁止ビット (VDP レジスタ 8) やインタレースモード (VDP レジスタ 9) は変化しませんでした。漢字モードでは両方ともクリアされます。ただし、インタレースモードは SCREEN 文の第 6 パラメータを同時に指定した場合は、正しく設定されます。

3. ユーザーエリア

MSX-JE が実装されているシステムで、CALL KANJI 命令を実行すると、MSX-JE 用のワークエリアが確保されます。一度確保された MSX-JE 用のワークエリアはたとえ、CALL ANK 命令を実行した場合でも解放することはできません。

そのため、アプリケーションプログラムによっては、メモリ不足のために漢字モードからは起動できないものがあるかもしれません。その場合は、一度リセットして ANK モードで起動してください。

なお、確保されるワークエリアの大きさは、フロントエンドプロセッサによって異なります。

6.4.3 拡張 BIOS

漢字ドライバは、アプリケーションソフトウェアからその機能の一部を使用できるように、拡張 BIOS を準備しています。MSX 漢字ドライバの拡張 BIOS は、エントリアドレステーブルを持たずに 0FFCAH 番地を呼び出します。拡張 BIOS では、現在の画面モードを返したり、画面を希望するモードに変更できます。

漢字は一文字出力の BDOS コールや、BIOS の CHPUT をコールして表示します。

1. 拡張 BIOS の使用法

MSX 漢字ドライバの拡張 BIOS は以下の手順で使用します。

スタックポインタは 0C000H 番地以上を指すようにしてください。

1. D レジスタに MSX 漢字ドライバのデバイス番号である 11H を、E レジスタに拡張 BIOS ファンクション番号を入れる。
2. E レジスタにセットした拡張 BIOS ファンクションに必要なレジスタの設定を行う。
3. 0FFCAH をコールする。

6.4.4 拡張 BIOS ファンクション

モードの取得

機能

現在の画面モードを数値で返します。

入力

E 0 (モードの取得)
A インストール検出 (下記参照)

出力

A モード
 0 = ANK
 1 = KANJI0
 2 = KANJI1
 3 = KANJI2
 4 = KANJI3

解説

漢字ドライバがインストールされていない場合は[A]が不変で返ってきます。これを利用して、[A]を0FFHにして呼ぶことにより、漢字ドライバがそもそもインストールされているのか否かが判ります。単にモードが知りたいだけなら、[A]を0にして呼んでください。

モードの設定

機能

画面モードを設定します。

入力

E 1 (モードの設定)
A モード
 0 = ANK

- 1 = KANJI0
- 2 = KANJI1
- 3 = KANJI2
- 4 = KANJI3

出力

なし

解説

スクリーンを指定されたモードに設定します。漢字ドライバがインストールされていないと無視されます。

6.5 漢字 ROM

6.5.1 漢字 ROM の I/O ポート

漢字 ROM は以下の I/O ポートに接続されています。

表 2.21 漢字 ROM と I/O ポート

アドレス	用途	IN	OUT
第1水準			
0D8H	第1水準漢字 ROM	フォントデータ	漢字番号の下位6ビット
0D9H			漢字番号の上位6ビット
第2水準			
0DAH	第2水準漢字 ROM	フォントデータ	漢字番号の下位6ビット
0DBH			漢字番号の上位6ビット

6.5.2 フォントデータの読み出し

MSX の漢字 ROM は、読み出したい文字の漢字番号を1回設定した後、32回 I/O ポートを読み出すとデータを獲得できます。内部にアドレスカウンタを持っているので、番号を変えてフォ

ントデータを読み込む必要はありません。

1. 漢字番号の計算方法

	区	漢字番号
第一水準	～15	(区×96+点)
	16～	(区×96+点-512)
第二水準	((区-48)×96+点)	

フォントイメージの並び方は8バイトごとに、左上、右上、左下、右下、の順です。

また、MSXの漢字ROM仕様では、フォントの形は決まっていません。フォントパターンは各メーカーによって異なっています。しかし、MSXの漢字ROM仕様では、ある特定のフォントのパターンを規定しており、そのパターンの有無をチェックすることにより、漢字ROMがシステムに実装されているかどうかの判断を行います。

JISの区点コードで規定されていない部分は統一されていません。

2. 第一水準漢字ROMのチェック

第一水準では、JISコード2140H(1区32点)の左上部分が、次のようなフォントになっています。

```

○ ○ ○ ○ ○ ○ ○ ○
○ ● ○ ○ ○ ○ ○ ○
○ ○ ● ○ ○ ○ ○ ○
○ ○ ○ ● ○ ○ ○ ○
○ ○ ○ ○ ● ○ ○ ○
○ ○ ○ ○ ○ ● ○ ○
○ ○ ○ ○ ○ ○ ● ○
○ ○ ○ ○ ○ ○ ○ ●
○ ○ ○ ○ ○ ○ ○ ○

```

3. 第二水準漢字ROMのチェック

第二水準では、JISコード737EH(83区94点)の漢字(●)の始め8バイトのデータのチェックサムが95Hになるようになっています。

例 01 02 0C 37 C0 3B 2A 2A

リスト 2.12 第二水準漢字ROM実装のチェックプログラム

```

;
; 第二水準漢字ROMが実装されているかどうかを調べるルーチン

```

```

;
; .Z80
;
; followings are definitions of Kanji ROM port addresses
;
kan2w0 defl 0dah ; Kanji ROM port address of lower 6bit
kan2w1 defl 0dbh ; Kanji ROM port address of higher 6bit
kan2rd defl 0dbh ; Kanji ROM port address of data read
;
; このプログラムは、JIS コードの 737EH (83 区 94 点) のフォントデータのうち、
; 最初の 8 バイトを読んで、チェックサムを計算します。そして、その値が 95H
; であれば、Z フラグをセットしてリターンします。
;
kancod defl 737eh
kanval defl 95h
;
; Calculate Kanji ROM address
;
romtmp defl kancod-5000h ; Subtract offset of 1st level Kanji code
romadd defl (high romtmp) * 96 + (low romtmp) - 20h
romlog defl romadd mod 64
romhi defl romadd / 64
;
kanchk: ld a,romlow ; Low order byte
; out (kan2w0),a ; Set Kanji ROM lower 6 bit address
; ld a,romhi ; High order byte
; out (kan2w1),a ; Set Kanji ROM higher 6 bit address
; ld c,0 ; Clear check sum accumulator
; ld b,8 ; Loop count
chklop: in a,(kan2rd) ; Read the character data
; add a,c ; Calculate check sum
; ld c,a ; Update the check sum
; djnz chklop ; loop till all done
; cp kanval ; Set zero flag if Kanji ROM found
; ret
;
end

```

6.5.3 デバイスイネーブル

漢字 ROM を内蔵する MSX システムは、外部スロットから漢字 ROM が挿入された場合のバスの競合を避けるために、システムソフトウェアにより内蔵 ROM のアクセスを制御しなくてはなりません。つまり、I/O アドレス F5H 番地のビット 1 に 0 が書き込まれた場合は、外部に漢字 ROM が存在することを示しているわけですから、内蔵 ROM のアクセスを禁止しなければなりません。

6.5.4 JIS から区点コードへの変換例

リスト 2.13 JIS から区点コードへの変換例

```

10 DEFINT A-Z
20 DIM P(31)
30 'INPUT JIS CODE IN HEX(4 DIGITS) THEN CONVERT TO ROM CODE
40 INPUT N$:L = VAL("&H"+RIGHT$(N$,2)):H = VAL("&H"+LEFT$(N$,2))
50 N = (H-32) * 96 + L - 32:IF H =>&H30 THEN N = N-512
60 L = N AND 63:H = N¥64
70 'ROM CODE OUT
80 OUT &HD8,L
90 OUT &HD9,H
100 PRINT "LOW =";L,"HIGH =";H
110 'GET PATTERN
120 FOR I = 0 TO 31
130 P(I) = INP(&HD9)
140 NEXT I
150 'PRINT OUT PATTERN
160 FOR I = 0 TO 7
170 P$ = RIGHT$("0000000"+BIN$(P(I)),8):GOSUB 250
180 P$ = RIGHT$("0000000"+BIN$(P(I+8)),8):GOSUB 250:PRINT
190 NEXT I
200 FOR I = 16 TO 23
210 P$ = RIGHT$("0000000"+BIN$(P(I)),8):GOSUB 250
220 P$ = RIGHT$("0000000"+BIN$(P(I+8)),8):GOSUB 250:PRINT
230 NEXT I
240 GOTO 40
250 FOR J = 1 TO 8
260 IF MID$(P$,J,1) = "0" THEN PRINT " "; ELSE PRINT "*" ;
270 NEXT J
280 RETURN

```

6.5.5 シフト JIS から JIS への変換例

ファイル上で上位 (変数 H)、下位 (変数 L) の順番に 2 バイトずつデータが入っている場合、

リスト 2.14 シフト JIS から JIS への変換例

```

100 IF &H80 <= H AND H < &HA0 THEN H = H - &H70:GOTO 130
110 IF &HE0 <= H AND H < &HF0 THEN H = H - &HB0:GOTO 130
120 GOTO 160
130 H = H + H:IF L >= &H80 THEN L = L - 1
140 IF L < &H9E THEN H = H - 1 ELSE L = L - &H5E
150 L = L - &H1F
160 REN End conversion

```

の方法を取り、H と L を 16 進 4 桁の上位、下位に変換する。

6.5.6 サンプルプログラム

添付のフロッピーディスクに、漢字 ROM をアクセスするためのサンプルプログラム(ファイル名「KANJIROM.MAC」)が入っています。

7章

プログラム開発の諸注意

7.1 BIOS

7.1.1 BIOS

アプリケーションプログラムが入出力を行う時には、必ず BIOS を使って下さい。BIOS を使う目的は、ハードウェアとソフトウェアを分離し、ハードウェアが変更されてもソフトウェアをそのまま使えるようにする事です。

BIOS は ROM の 0000H 番地から始まるジャンプテーブルを通して呼び出されます。ジャンプテーブルの飛び先、つまり BIOS の内容はハードウェアの変更や機能の拡張のために変更されますので、アプリケーションプログラムがそこを直接に（ジャンプテーブルを通さずに）参照してはいけません。MSX2 の拡張 ROM にも、同様のジャンプテーブルがあり、拡張機能の呼び出しに使われます。

7.1.2 拡張 BIOS コールの方法

MSX では、新たに追加されたデバイスの持つドライバールーチン（拡張 BIOS）をアプリケーションソフトウェアから利用できます。その方法はデバイスによって異なりますので、詳細については各デバイスの仕様書をご覧ください。

以下では、RS-232C を例にとりて、拡張 BIOS コールの手順を説明します。

1. 8000H 番地以上に拡張 BIOS のエントリーアドレスを入れるためのワークエリア（64 バイト）をとる。

2. HL レジスタに、1. で取ったワークエリアの先頭アドレスを入れる。
3. B レジスタに、1. で取ったワークエリアが存在する RAM のスロットアドレスを入れる。RAM のスロットアドレスについては、「7.3.5 RAM のあるスロット」参照。
4. D レジスタにデバイス番号、E レジスタに拡張 BIOS のファンクション番号 (0) を入れる。
5. 【EXPBIO(0FFCAH)】をコールする。
6. その結果、1. で取ったワークエリアに、以下の順番でデータが入ってくる。

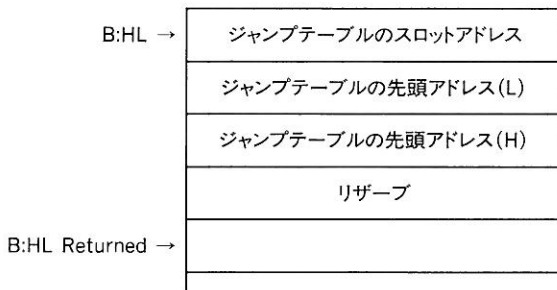


図 2.33 RETURN 情報の形式

ジャンプテーブルのスロットアドレスとは、拡張 BIOS を持った ROM の存在するスロットアドレスを、ジャンプテーブルの先頭アドレスとは拡張 BIOS へのジャンプテーブルのアドレス (EXBTBL) を指します。例えば、RS-232C の場合、ポートを OPEN する場合には、EXBTBL + 6 バイトをインタースロットコールします。拡張 BIOS の詳細は、各デバイスの拡張 BIOS コール仕様書をご覧ください。

拡張 BIOS コールを使用するデバイスには、RS-232C、MSX MODEM、MSX-AUDIO、MSX-JE などがあります。

7.1.3 拡張 BIOS のワークエリア

拡張 BIOS を使うときには、スロットが切り換えられても CPU が RAM を参照できるように、スタックを C000H 番地よりも上位に置いて下さい。

同様の理由で、RS-232C などの FCB は 8000H 番地よりも上位に置いて下さい。

7.2 I/O ポート

7.2.1 I/O ポート

MSX では互換性と拡張性を維持するために、ハードウェアについては基本的な仕様を決めるにとどめ、ハードウェアの違いをソフトウェアによって吸収するように定めています。したがって、市販を目的としたソフトウェアでは、その内容を問わず決して I/O ポートを直接アクセスしないで下さい。I/O ポートを直接アクセスするプログラムは各システムでの互換性を確実に失ってしまいます。ただし、VDP に関してはこの限りではありません。

7.2.2 ユーザー定義用 I/O ポート

I/O の 00H~3FH はユーザー用に解放されています。専用システムの構築などの目的で、ハードウェアを接続するときは、ここに割り当て下さい。

40H~7FH は周辺機器用にリザーブされています。市販を目的としたハードウェアはなるべくメモリマップド I/O 方式で本体と接続して下さい。スピード等の問題で、どうしても I/O ポートに接続することを希望する場合は、あらかじめ株式会社アスキー MSX 推進部(連絡先は巻末を参照)までご連絡下さい。当社で I/O の割り当て調整を行います。

7.3 スロット

7.3.1 拡張スロット

過去に、

1. 拡張スロットに差し込まれた場合
2. RAM が拡張スロットにある場合

などに、動作しないソフトウェアがいくつかありました。MSX2 の多くの機種では本体内部で拡

張スロットを使っていますので、拡張スロットでの動作不良は致命的な問題になります。ソフトウェアを発売するまえに、そのソフトウェアが拡張スロットに入れられた場合と、RAMが拡張スロットにある場合について、必ず動作試験を行って下さい。

特に、FFFFH番地には拡張スロットレジスタが置かれますので、そこをRAMのつもりで参照してはいけません。拡張スロットで暴走するソフトウェアの多くが、

```
LD    SP,0
```

を行っていました。このようなソフトウェアを作らないようにご注意ください。

7.3.2 プログラムのスロットアドレス

プログラムが1ページと2ページ(4000H番地からBFFFH番地)までの32KバイトのROMに入っている場合には、1ページのROMのINITルーチン(データブックのスロット管理メカニズム参照)が呼び出された時に、2ページがそのROMの続きではなく、別のスロットになっています。1ページのROMの内部で自分がいるスロットのスロットアドレスを調べ、2ページをそのスロットに切り換える必要があります。

添付のフロッピーディスクにサンプルプログラムが入っています。ファイル名は「WHERE.AMI.MAC」です。

他のプログラムから参照されるような場合にも、同じ方法で求めた自分のスロットアドレスを利用して下さい。

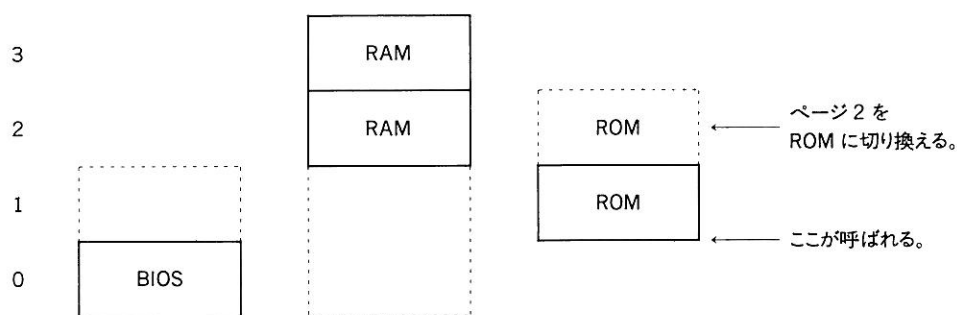


図 2.34 INIT ルーチン呼び出し時のスロット状況例

7.3.3 BASIC の ROM のスロットアドレス

従来は、「BASIC の ROM は基本スロットの 0、または基本スロットの 0 を拡張した 0-0 に置く。」という仕様がありました。しかし、本体外に V9938 と ROM を増設して MSX2 の機能を持たせる場合には、BASIC の ROM が他のスロットにあります。また、MSX2 の拡張 ROM が入っているスロットは、機種によって異なります。

ゆえに、下記のシステムワークエリアの内容を参照して、BASIC の ROM のスロットアドレスを調べて下さい。

DOS から BIOS を呼ぶ場合にこの方法を使えます（「5部 スロットとカートリッジ」参照）。

表 2.22 BASIC の ROM のスロットアドレス

アドレス	ラベル	内容
FCC1H	EXPTBL	MAIN-ROM のスロットアドレス
FAF8H	EXBRSA	SUB-ROM のスロットアドレス

注意

BASIC 1.0 では、EXBRSA の内容は 00H です。

7.3.4 スロット切り替え時の注意

インタースロットコールを行うと、IX、IY および裏レジスタの内容が破壊され、リターン時に割り込みが禁止されています。

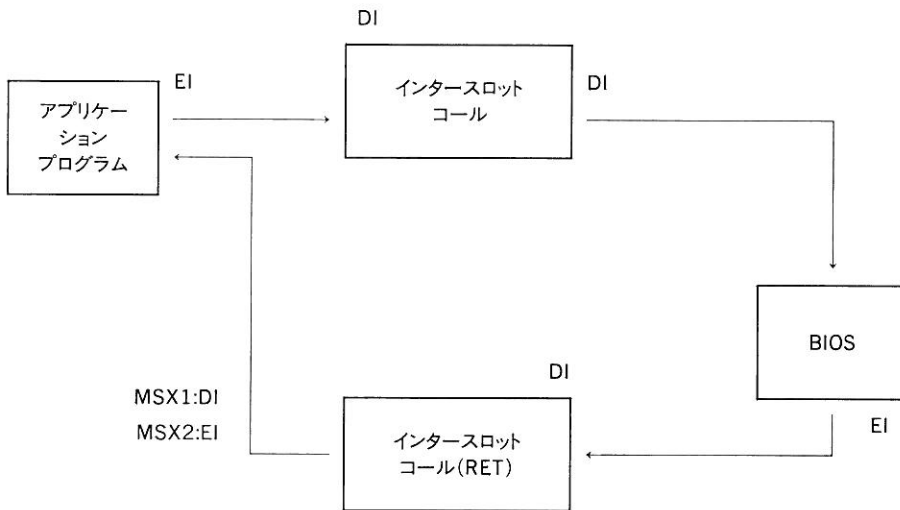


図 2.35 インタースロットコール時の割り込み

Z80 の特性により、MSX2 でもごく希なケースとして、DI 状態でリターンすることがあります。割り込み状態が EI の必要があるプログラムでは、インタースロットコール直後に EI することをお勧めします。

7.3.5 RAM のあるスロット

フロッピーディスクドライブを実装しているシステムでは、各ページの RAM のあるスロットを以下のワークエリアに保存しています。フロッピーディスクドライブが実装された MSX で、RAM の実装されたスロットアドレスを知りたい場合には、必ずこのワークエリアを参照して下さい。

例えば、ページ 1 の RAM をデータエリアとして使用するためにスロットを切り換えるといった場合に、このワークエリアの値を参照します。

表 2.23 RAM のスロットアドレスを保存したワークエリア

RAM のスロット	ラベル	アドレス
ページ 0	RAMAD0	0F341H
ページ 1	RAMAD1	0F342H
ページ 2	RAMAD2	0F343H
ページ 3	RAMAD3	0F344H

各ビットは以下の意味を持ちます。

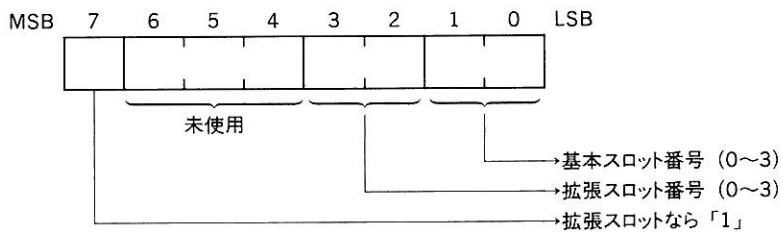


図 2.36 RAMAD0~3 のビットの意味

フロッピーディスクドライブを実装しないシステムの場合は、RAMのあるスロットを自分で捜さなければなりません。添付のフロッピーディスクに、「RAMSRCH.MAC」という名前のサンプルプログラムが入っています。

このプログラムは BASIC プログラムから DEFUSR 命令で呼ばれるようになっていました。フロッピーディスクドライブを実装した機種では、上記のワークエリア【RAMAD0(0F341H)】～【RAMAD3(0F344H)】を参照し、このサンプルプログラムによる RAM サーチは行わないで下さい。さもなければ、システムに矛盾をきたし、暴走の原因となる場合があります。

7.4 ワークエリア

7.4.1 BASIC のワークエリア

メイン RAM の F380H 番地から FFFE H 番地までは、BIOS と BASIC インタープリタのワークエリアですから、不用意に使わないで下さい。ワークエリア内の使われていない場所は、将来

の機能拡張のために予約されています。

7.4.2 ユーザーエリア

BASIC、Disk BASIC の環境で、ユーザーの使用できる RAM エリアは 8000H 番地～0FC4AH と 0FC4BH の内容が示すアドレスです。0FC4AH のラベル名は HIMEM です。

MSX-DOS では、CP/M と同じく、100H 番地～6、7H 番地の内容が示すアドレスまでが TPA となります。

アプリケーションプログラムは必ず HIMEM (DOS の場合は 6、7H 番地) を確かめて、最悪の場合でも暴走しないように対処する必要があります。対処の方法としては、

1. ワークエリアをリロケータブルにする。
2. ワークエリアを BOTTOM から確保する。
3. ドライブの数を減らすように指示して停止する。

などがあります。

7.4.3 RAM とスタックポインタの初期化

電源投入時の RAM の内容は不定で、システムのワークエリア以外の領域は初期化されません。自分のワークエリアを初期化する事は、アプリケーションプログラムの責任です。

RAM の内容が 00H である事を期待しているために、特定の本体との組み合わせでしか動かないソフトウェアがあり、問題になっています。

また、ROM カートリッジの INIT ルーチンが呼ばれた時のスタックポインタの値は不定で、ディスクインターフェイスが先に初期化されたような場合にはそうでない場合よりもスタックポインタの値が小さくなっています。このため、スタックポインタを初期化しないプログラムが暴走するという問題が起きました。INIT ルーチンで起動し、そのまま走り続けるプログラムは、必ずスタックポインタを初期化して下さい。

7.4.4 デバイスドライバなどのワークエリア

デバイスドライバや BASIC から呼ばれるサブルーチンのように、他のプログラムと同時にメモリに存在するプログラムは、ワークエリアの確保に特に注意を要します。

カートリッジの INIT ルーチンが HIMEM (または BOTTOM) を書き換え、古い HIMEM と新しい HIMEM の間を自分のワークエリアとして予約します。そのワークエリアのアドレスを SLTWRK というスロット別に割り当てられた 2 バイトの領域に記録します。

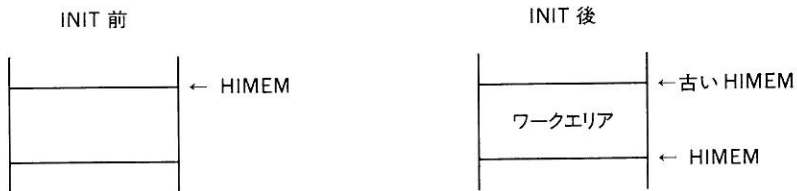


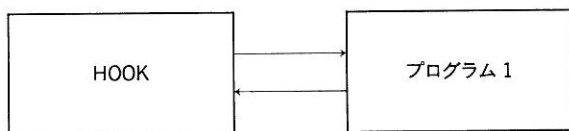
図 2.37 デバイスドライバのワークエリア

7.4.5 フック

例えば、RS-232C カートリッジが割り込み処理のためにフックを書き替えたとします。次のカートリッジが同じフックを書き替えると、RS-232C カートリッジは割り込みフックを使えなくなります。

このような事を防ぐためにフックを書き替える場合には、以前のフックの内容 (例では、RS-232C カートリッジの割り込み処理ルーチンへのインタースロットコール命令) を記憶します。そして、自分がフックで呼ばれたら以前のフックの内容を呼び出し、そのフックを使おうとしている全てのカートリッジに制御が渡るようにして下さい。

プログラム 1 の初期化時



プログラム 2 の初期化時

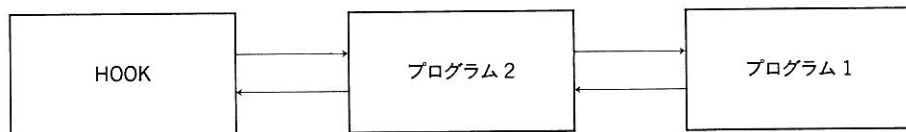


図 2.38 フック

7.5 VDP

7.5.1 VDP の I/O アドレス

MAIN ROM の 6 番地と 7 番地には VDP のリードアドレスとライトアドレスが書き込まれています。I/O ポートを使って VDP を直接アクセスする場合は、ROM の内容から I/O ポートのアドレスを調べ、その I/O アドレスをアクセスして下さい。TMS9918 を使っている機種 (MSX1) では VDP が 98H 番地に置かれていましたので、その番地を直接参照しても実害はありませんでしたが、V9938 を外付けして MSX2 に拡張したハードウェア (MSX2 アダプター) の場合には VDP のアドレスが変わりますので、必ず以上の事を守って下さい。

表 2.24 VDP の I/O アドレス

VDP のポート	I/O アドレス
データリード	MAIN ROM の 6 番地の内容
データライト	MAIN ROM の 7 番地の内容
ステータスリード	データリードアドレス+1
コマンドライト	データライトアドレス+1
パレットライト	データライトアドレス+2
インダイレクトアクセス	データライトアドレス+3

7.5.2 レジスタの保存

VDP のいくつかのレジスタは書き込み専用であり、直接には現在のレジスタの内容を知ることができません。そこで、システムワークエリアの【RG0SAV (0F3DFH)】から【RG7SAV (0F3E6H)】と【RG8SAV (0FFE7H)】から【RG23SA (0FFF6H)】 (RG8SAV 以降は V9938 使用機種のみ) に、レジスタの内容を保存してあります。BIOS 通さずに VDP のレジスタを書き替える場合には、これらのワークエリアの内容も書き替えて下さい。特に、機械語プログラムから BASIC や DOS に制御が戻った時にレジスタの内容とワークエリアの内容が異なると、画面表示が正しく行われません。

また、VDP のレジスタの一部のビットを書き替える場合には、ワークエリアからレジスタの内容を読んで、必要に応じてビットを変え、そのデータをレジスタとワークエリアの両方に書き込

んで下さい。

7.5.3 VRAM の初期化

アプリケーションプログラムが動き始める時の VRAM の内容は不定です。アプリケーションプログラムは、必ず VRAM を初期化して下さい。

VRAM を初期化していないために、ハードウェアの特性で VRAM の内容が 0 になっている機種でのみ画面が正しく表示されるというソフトウェアがありました。

7.5.4 VRAM の容量

FAFCH 番地のビット 2 とビット 1 の内容で、VRAM の容量が判ります。

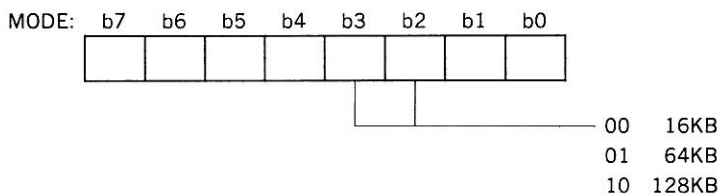


図 2.39 【MODE(0FAFCH)】のビット

7.6 キー入力

7.6.1 キー入力の方法

BIOS によるキー入力に 2 つの方法があります。目的に応じて使い分けて下さい。特に、ヨーロッパ用などの異なる種類のキーボードに対応するソフトウェアを作る場合に注意が必要です。

1. SNSMAT ルーチンを使って、リアルタイムにキーの物理的な状態を得ます。
2. CHSNS と CHGET を使い、バッファリングされた文字コードを得ます。また、KILBUF ルーチンでキーバッファの内容を捨てることができます。この方法では、

VDP によるタイマ割り込み時にキースキャンが行われますので、割り込みが許可されている必要があります。

7.6.2 キークリック

システムワークエリアの【CLIKSW(0F3DBH)】という 1 バイトの内容が 0 でなければキー入力時にクリック音が発生し、0 であればクリック音が発生しません。

7.7 プリンタ

7.7.1 MSX 用でないプリンタへの対応

【NTMSXP(0F416H)】というワークエリアの内容が 0 でなければ、MSX 用でないプリンタのために BIOS が文字を変換します。その場合、ひらがながカタカナに変換され(日本版のみ)、グラフィック文字が空白に変換されます。

7.7.2 RAW モード印字

【RAWPRT(0F418H)】の内容が 0 でなければ、RAW モード印字となり、タブ (09H) の空白への変換と前項の文字の変換が行われません。ビットイメージプリントを行う時には、必ず RAW モードを設定して下さい。

ただし、RAW モード印字は BIOS の OUTDO(0018H / MAIN)を使用しているプログラムのみ有効で、LPTOUT(00A5H / MAIN)では無効です。したがって、BASIC の PRINT 命令などでは RAW モード印字されますが、MSX-DOS のプリンタ出力 (ファンクションコール 05H) などでは何も起こりません。

MSX2+や MSX-DOS2 の漢字 BASIC でプリンタにエスケープシーケンスなどを送る場合も、RAW モード印字にして下さい。

7.8 BASICの種類

7.8.1 BASICのバージョン番号

アプリケーションプログラムが BASIC のバージョン番号を知るためには、下記の方法を使います。

MAIN ROM の 2DH 番地に BASIC のバージョン番号が入っています。MSX BASIC 1.0 では 2DH 番地の内容が 00H、MSX BASIC 2.0 では 01H、MSX BASIC 3.0 では 02H です。

7.8.2 インターナショナル MSX

MSX は世界各国で使われており、その種類は主なもので 10 を越えています。下記の項目が、国によって大きく異なります。

1. キーボード配列
2. 文字セット
3. PRINT USING の書式
4. タイマ割り込み周波数

ID バイトという ROM 内の情報を読むと本体の種類が分かり、各国の MSX に対応できます。

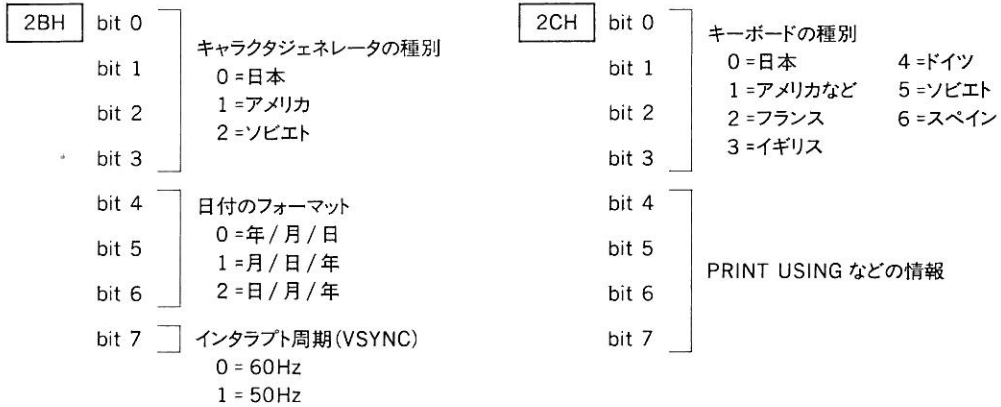


図 2.40 ID バイトの内容

表 2.25 国別の MSX 仕様

国名	TV セット	日付 フォーマット	初期画面 モード	PRINT USING		
				文字列の長さ指定	文字の置き換え	通貨シンボル
日本	NTSC (60Hz)	年 / 月 / 日	SCREEN 1	%	@	¥(エン)
イギリス	PAL (50Hz)	日 / 月 / 年	SCREEN 0	%	&	£(ポンド)
インター ナショナル	PAL (50Hz)	月 / 日 / 年	SCREEN 0	%	&	\$(ドル)
アメリカ	NTSC (60Hz)	月 / 日 / 年	SCREEN 0	%	&	\$(ドル)
フランス	SECAM (50Hz)	日 / 月 / 年	SCREEN 0	%	&	\$(ドル)
ドイツ	PAL (50Hz)	日 / 月 / 年	SCREEN 0	%	&	\$(ドル)
ソビエト	NTSC (60Hz)	月 / 日 / 年	SCREEN 0	%	&	\$(ドル)
スペイン	PAL (50Hz)	月 / 日 / 年	SCREEN 0	%	&	\$(ドル)

7.9 ディスク

フロッピーディスクが使用するワークエリアの情報は、原則的には公開していません。この章で公開しているアドレスと使用法以外の目的で、ワークエリアを参照したり、書き換えたりすると動作は保証できませんので、十分ご注意ください。また、この章で紹介しているワークエリア以外は、将来のバージョンアップの際に、アドレスや使用法が変わる可能性があります。

7.9.1 ディスクを使わない方法

リセット（または電源投入）してからビーブ音が鳴るまでシフトキーを押していると、ディスクインターフェイスが無視され、ROM BASIC が起動します。ディスク内蔵機でディスクとワークエリアが衝突するソフトウェアを動かす場合には、この方法を使って下さい。

同様に、コントロールキーを押していると、2ドライブシミュレーション機能(1ドライブのシステムでソフト的に2ドライブをシミュレートする機能)が無効になり、ディスクワークエリアが少し小さくなります。

アプリケーションプログラムから見てドライブが2台接続されているシステムで、それが2ドライブシミュレーション機能なのか、それとも物理的にディスクが接続されているかを知ることができません。また、ソフトウェアから2ドライブシミュレーション機能を無効にすることも不可能です。

7.9.2 ドライブの有無と数を知る方法

【H.PHYDIO(0FFA7H)】というフックの内容を読んで下さい。そこに RET 命令が書かれていれば、ディスクは接続されていません。RET でなければ、ディスクが接続されています。

ディスクが存在する場合は、DOS のシステムコール (BDOS コール) の 18H (ログインベクトルの獲得) を使用して下さい。DOS のシステムコールは Disk BASIC から利用できます。詳しくは「第3部 4章 システムコール」をご覧ください。

7.7.1参照

7.9.3 エラー処理

MSX-DOS には、ディスク入出力のエラーを検出して適切な回復手続きを行う機能が用意されています。

1. MSX-DOS の標準エラー処理手順

システムコールのディスク入出力でエラーが発生すると、エラー処理ルーチンが呼ばれます。標準処理手順では、エラーメッセージと「Abort Retry Ignore?」というプロンプトが表示されて、キー入力を待ちます。ここで、A、R、I を入力しますが、これらはそれぞれ以下のような意味があります。

- A システムのウォームブートが起こり、DOS のコマンド入力状態になります。
 R 処理を再び試みます。
 I システムコールをキャンセルして、アプリケーションには正常終了を返します。

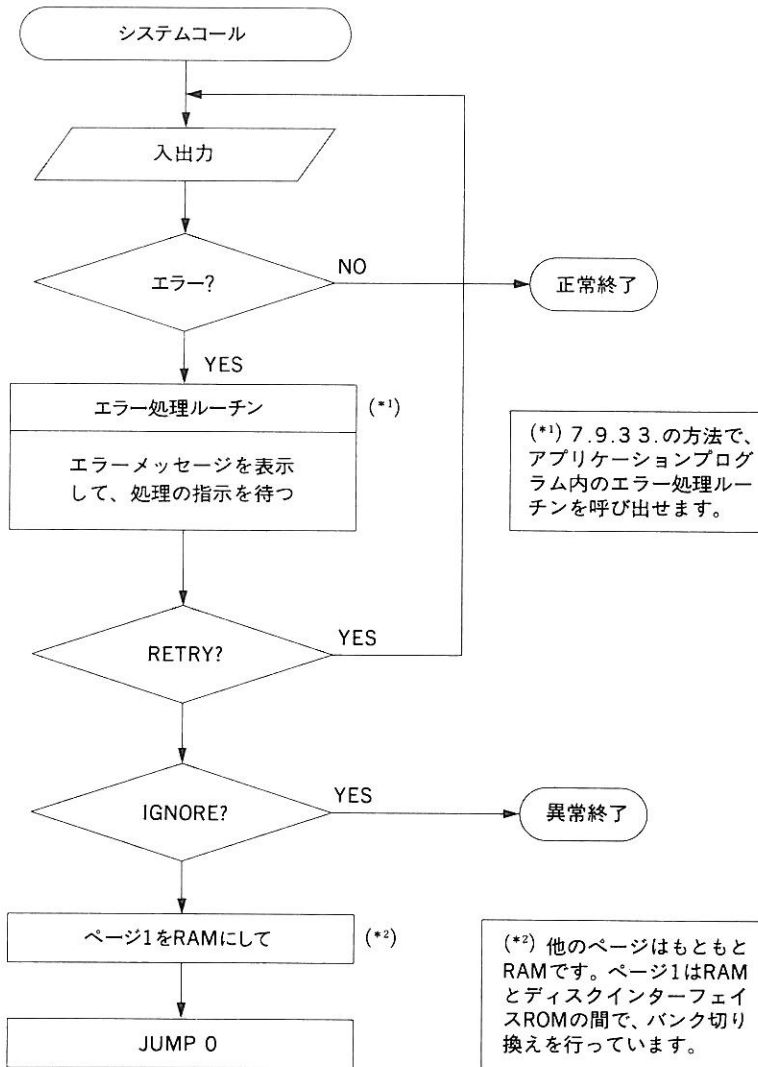


図 2.41 MSX-DOS の標準エラー処理手順

2. Disk BASIC の標準エラー処理

Disk BASIC の環境でディスクエラーが起こった時には、エラーメッセージを表示して、BASIC の入力待ち状態になります。

3. アプリケーションプログラムがエラーを処理する方法

【DISKVE(0F323H)】は、ディスクエラー処理ルーチンへのポインタへのポインタ（エラー処理ルーチンの開始アドレスを記憶してあるアドレス）です。そこを書き替えることにより、エラー発生時にアプリケーションプログラム内のエラー処理ルーチンを呼び出すことができます。

図 2.41 の *1 は、エラー処理ルーチンの入口を示します。ここで、

A レジスタにはエラーを起こしたドライブの番号 (A = 0、B = 1...H = 7)

C レジスタにはエラーの種類

が入っています。C レジスタの各ビットの意味は以下の通りです。例えば、書込み時に、「NOT READY」エラーが起ると、C レジスタの内容は 03H になります。

表 2.26 エラーの種類

bit0(LSB)	エラー時の状態
0	読み込み時
1	書き込み時

(MSB)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	エラーの種類
1	0	0	0	0	0	0	bad FAT
0	0	0	0	0	0	0	write protect
0	0	0	0	0	0	1	not ready
0	0	0	0	0	1	0	CRC error
0	0	0	0	0	1	1	seek error
0	0	0	0	1	0	0	record not found
0	0	0	0	1	0	1	write fault
0	0	0	0	1	1	0	other error

図 2.41 の *2 は、エラー処理ルーチンの出口を示します。ここから DOS の標準エラー処理に戻る場合には、C レジスタの内容で処理を指定し、リターンします。ここで、AF、B、DE、HL レジスタには何が入っていても構いません。

C レジスタ	事後処理
2	abort (処理を中断)
1	retry (入出力を再度試みる)
0	ignore (エラーを無視して先へ進む)

以下の場合には、エラー処理ルーチンは必ず2 (abort) または1 (retry) を返すようにして下さい。さもなければ、ディスクが壊れることがあります。

1. エラーコードのビット7が1 (bad FAT) だった場合
2. エラーコードが1010 (Unsupported media type) だった場合

Disk BASIC の環境下では、エラーが発生したときには、Disk ROM がイネーブルされており、スロット切り替えは行われません。したがって、ページ1にあるディスクエラー処理ルーチンは制御を受け取ることができません。しかし、MSX-DOS の環境では、ページ1のRAMがイネーブルされてから、コントロールが渡ってきます。

DOS から帰ってくるエラーは「directory entry not found」や「no more free sectors to write」のようなロジカルエラーです。そして、ディスクエラーハンドラが返すのはハードウェアのエラーです。この2つは異なります。

MSX-DOS 上で動作するアプリケーションプログラムが特別なエラー処理を行いたい場合は、下記のようにします。

1. DISKVE の内容をセーブする。
2. DISKVE にエラー処理ルーチンへのポインタへのポインタを設定する。
3. 01H 番地と 02H 番地の内容をセーブする。
4. 1、3.でセーブした値を RESTORE するルーチンへのエントリを 01H 番地、02H 番地
に書く。1、3.の値を RESTORE するルーチンとは、
 - a. DISKVE を元の値に戻す
 - b. 01H 番地と 02H 番地を元の値に戻す
 - c. 0H 番地へジャンプする
 を実行するもの。
5. システムコールの実行前に、スタックポインタを保存する。
6. システムコールを行う。
7. エラー処理ルーチンが呼び出されたら、スタックポインタを元に戻し、適当な処理を続ける。

前述の【DISKVE】の内容は、システムのコールドスタート時、およびDOSとBASICの切り換え時に初期化されます。このため、あるプログラムが【DISKVE】の内容を書き換えたまま終了し、DOSのコマンドや次のプログラムがエラーを起こすと暴走します。

ゆえに、DISKVEを書き替えたプログラムはそのままコマンドレベルに戻らずに、01H番地と02H番地の内容(WOOM BOOT)を書き替えて、CTRL + C でプログラムが止まらないようにして下さい。

7.9.4 BDOS コール

MSX Disk BASIC の環境下でも、DOS ファンクションコールが可能です。F37DH が Disk BASIC の場合でのファンクションコールエントリで、MSX-DOS の場合の 5 番地コールに相当する機能を提供します。詳しくは「第3部 4章 システムコール」を参照して下さい。

ファンクション

7.9.5 FDC

MSX では FDC の種類や割り当てられるアドレスは定められていません。したがって、いかなる理由があろうとも、アプリケーションプログラムは直接 FDC をアクセスしてはいけません。ディスクのアクセスには必ずファンクションコール (BDOS コール) を使用してください。

7.9.6 DISK ROM が入っているスロットを知る方法

ワークエリアの FB21H から FB28H までのドライブテーブルという領域に、ディスクインターフェイスに関する情報が記録されています。各カートリッジに 2 バイトが割り当てられており、その内容は以下のようになっています。カートリッジが 3 個以下であれば空いているテーブルに 00H が入ります。

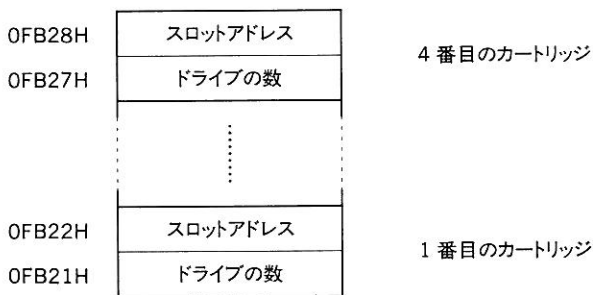


図 2.42 DISK ROM の情報

7.9.7 FDD のモーターを止める方法

以下の方法で、アプリケーションプログラムが FDD のモーターを止めることができます。ただし、ディスクドライブによっては、モーター-OFF のエントリを持っていないものがあるので、そのエントリがあるかどうかを事前にチェックしてください。また、ディスクドライブによっては、何もしないでリターンするものもあります。

なお、このルーチンはアスキーから発売しているハードディスクインターフェイスで実行すると、シャットダウン（ハードディスクのヘッドを安全地帯へ退避させること）の効果があります。ユーザープログラムがシャットダウンを行うときは、ユーザープログラムから、@_SHUTDOWN をコールして下さい。

リスト 2.15 FDD のモーター-OFF

```

;
;      moter off Subroutine
;
;      .z80
;      entry          @_SHUTDOWN
;
rdslt          equ      000ch
calslt         equ      001ch
motor_off_entry equ      4029h
master_slot    equ      0f348h
;
@_SHUTDOWN:
    ld      a,(master_slot)          ; motor off entry present?
    ld      hl,motor_off_entry
    call   rdslt
    and    a
    ret    z                          ; no, no way....
    ld      iy,(master_slot-1)       ; we have it! call it now
    ld      ix,motor_off_entry
    jp     calslt
;
;      注意
;      マスターカートリッジの'motor_off_entry'はシステムにつながっている
;      すべてのディスクドライブを調べ、モーターを止めます。
;      しかし、旧版のマスターカートリッジはモーターを止めることができません。
;      この場合は、モーターを止める手だてはありません。
;      たとえ、マスターカートリッジがその機能を持っていても、
;      スレブカートリッジがその機能を持っていなければ、
;      スレブカートリッジが管理するディスクドライブのモーターは止まりません。
;
end

```

C コンパイラからは、_shutdown() ; でコールできます。

C コンパイラのテストプログラムです。

```
#include <stdio.h>
VOID _shutdown();
main()
{
    _shutdown();
}
```

7.9.8 マスターカートリッジのロットアドレス

ディスク機能の大部分を受け持つディスクカートリッジは、「マスターカートリッジ」と呼びます。マスターカートリッジのロットアドレスは、【MASTER_SLOT(0F348H)】に保存されています。

7.9.9 ディスクの交換

2ドライブシミュレーション機能が働いているときは、別のドライブにアクセスすると、

```
Insert disk for drive x:
and strike a key when ready
```

というメッセージが表示されます。そのため、グラフィックを使ったアプリケーションソフトウェアでは不都合が生じます。

アプリケーションソフトウェアが、自分でディスクの交換メッセージを出すためには、【H.PROMPT(0F24FH, 3)】をアプリケーションプログラム側のディスク交換ルーチンへのジャンプ命令に書き換えます。

【H.PROMPT(0F24FH, 3)】がコールされたときは、Aレジスタにドライブ名(A、B・・・)がキャラクタで入っています。また、スタックトップが標準のプロンプトルーチンへのリターンアドレスで、スタックトップ+2がシステムへのリターンアドレスになっています。したがって、標準プロンプトルーチンをバイパスするためには、リターンする前にスタックを1レベル捨てなければなりません。なお、レジスタを保存する必要はありません。

注意

交換ルーチンはページ1には置けません。

交換ルーチンからは、アプリケーションプログラムにジャンプしてはいけません。必ずリターンして下さい。

7.10 便利な機能

7.10.1 エスケープシーケンス

MSX には別表のようなエスケープシーケンス機能があります。この機能は BASIC の PRINT 文、BIOS および BDOS コールのコンソール出力で利用できます。シーケンスの機能は、DEC 製 VT52 ターミナルおよびヒースキット製 H19 ターミナルのサブセットになっています。

7.10.2 BASIC に戻る方法

BASIC ROM のスロットをイネーブ爾した後、MAIN ROM の 409BH 番地へジャンプして下さい。BASIC のワークエリアが壊されていないければ、BASIC のコマンド入力の状態になり、OK (MSX2 でプロンプトの設定を行ってればその文字列)が表示されます。ジャンプする時のレジスターやスタックの内容は一切無視されます。

また、内部ルーチンの NEWSTT (「4.4 BASIC の内部ルーチン」参照) で次の命令を実行しても、BASIC に戻ることができます。



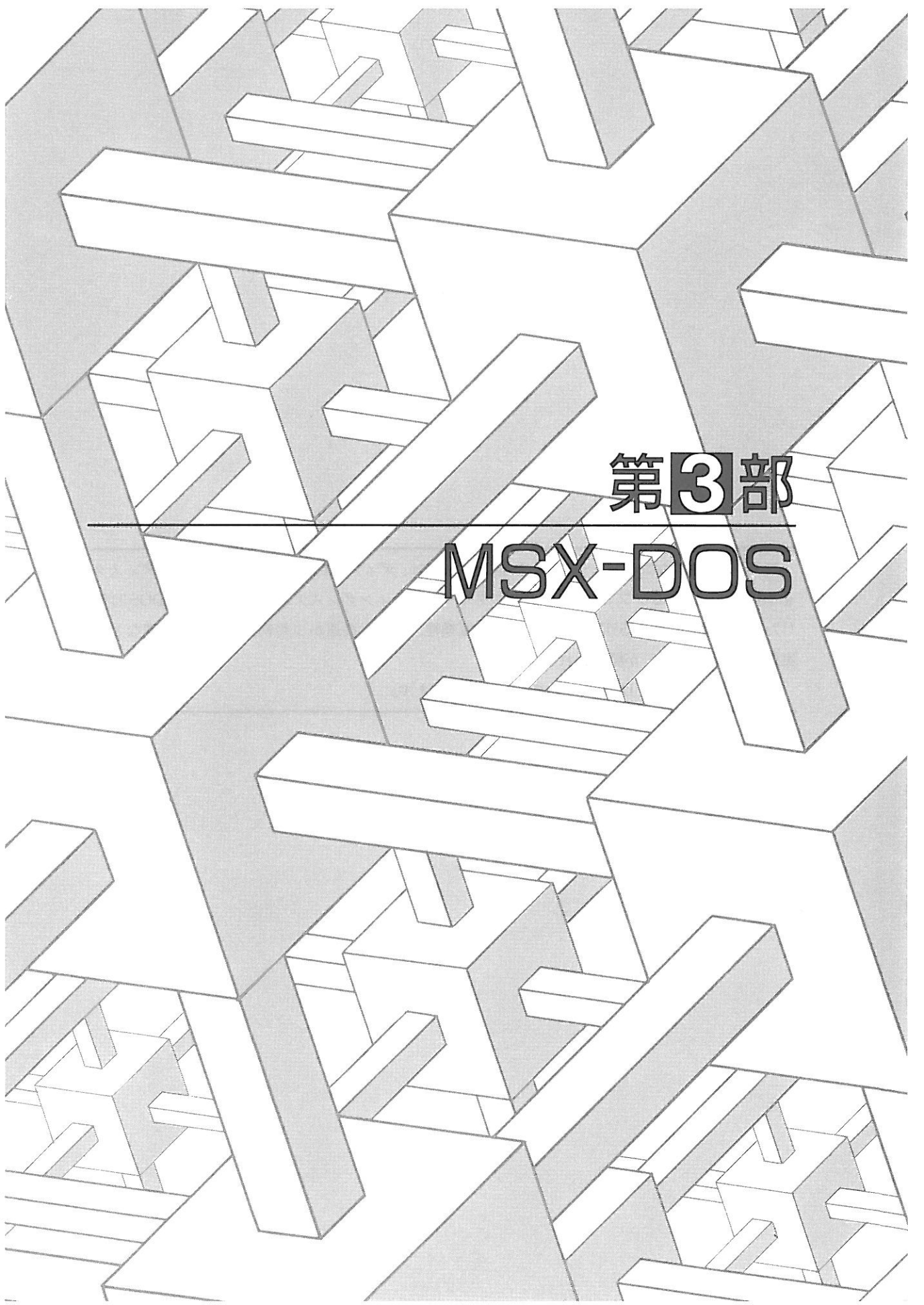
図 2.43 ウォームスタートのための NEWSTT の設定

7.10.3 オートスタート

単純なゲームカートリッジ等では、ROM のヘッダの INIT の場所にプログラムの実行開始アドレスを書くことでそのプログラムを起動できます。しかし、この方法を使うと他のカートリッジの初期設定が行われなため、ディスク等を使えなくなります。

ROM カートリッジのソフトウェアでディスクを使う場合は、カートリッジの INIT ルーチン実行時に、起動させたいプログラムへのインタースロットコール命令を【H.STKE(0FEDAH)】というフックに書き、リターン命令でシステムへ戻って下さい。すると、全てのカートリッジの初期設定が終わり、ディスクがあれば Disk BASIC の環境が整ってから、そのフックがコールされ、目的のプログラムを起動できます。

この方法は、ディスクが無い場合でも有効です。



第3部

MSX-DOS

MSX を実用的なコンピュータとして活用するには、ディスクが不可欠です。そこで、ディスクをコントロールするソフトウェア（ディスクオペレーティングシステム）として、MS-DOS に準じた高度な操作性や MS-DOS とのファイル互換性、強力で高速かつ柔軟なファイル処理などを実現した MSX-DOS が開発されました。

第3部では、MSX-DOS について詳しく説明します。



1.1 MSX-DOS の特徴

MSX-DOS は、MSX 用のディスクオペレーティングシステムとして、数々の優れた特徴があります。以下で、その特徴について解説します。

1.1.1 MS-DOS に準じた高度な操作性

MSX-DOS は、16 ビットマイクロプロセッサ用の汎用 OS である MS-DOS をもとに開発されました。そのため、MSX-DOS を知ることで、MS-DOS の操作にも慣れることができます。

1. MSX-DOS のコマンドの名前と機能は MS-DOS と共通です。そのため、MS-DOS を使用するときも操作の違いで困ることはほとんどありません。
2. ファイルを修正すると、その日付と時刻が自動的にディスクに記録されるので、いつファイルが修正されたかわかります。
3. コマンド行の編集機能を備えています。そのため入力ミスを容易に修正することができ、同じコマンドを続けて入力したり、似たコマンドを入力したりするのが簡単です。
4. 実行するプログラム名を「autoexec.bat」というファイルに書き込んでおくだけで、起動時に自動的に実行する機能を備えています。

1.1.2 MS-DOS とのファイルの互換性

フロッピーディスクのフォーマットが MS-DOS と同じなので、MS-DOS で作成したデータファイルを MSX-DOS で読み書きすることも、その逆も可能です。

1.1.3 強力、高速かつ柔軟なファイル処理

MSX-DOS は以下のような優れたファイル処理を実現しています。

1. MS-DOS と同様に、最大1 ギガバイトという大容量のファイルを取り扱うことができます。
2. FAT やディレクトリのバッファリングにより、高速なディスクの入出力を実現しています。また、ブロック入出力を利用したプログラムでは、さらに高速な入出力が可能です。
3. ファイルの処理に論理レコードと呼ばれる手法を取り入れて、プログラムの開発を容易にし、ディスクの有効利用を図っています。
4. 周辺機器をディスクファイルと同じように取り扱うことができます。例えば、ファイルからの入力と同じ方法でキーボードから入力したり、ファイルに出力するようにしてプリンタに出力したりすることができます。

1.1.4 Disk BASIC と DOS 間の関係

MSX Disk BASIC は MSX-DOS を利用して動作しています。そのため、BASIC の画面編集機能や画面制御命令などの便利な機能を、MSX-DOS から利用することができます。

1. ファイル処理に同じ方法を用いているので、Disk BASIC で作成したファイルを MSX-DOS で扱うことも、その逆も可能です。
2. MSX-DOS には Disk BASIC に移行するコマンドが、また Disk BASIC には MSX-DOS に移行するコマンドが用意されているので、相互に往き来することができます。

1.1.5 CP / M からのプログラムの移植が容易

MSX-DOS のファンクションコールの大部分は、同じ 8 ビットマイクロプロセッサ用の OS である CP / M のファンクションコールと互換性を持っています。したがって、CP / M 上の多くのプログラムを移植してそのまま利用することができます。

1.2 この章の表記法

[] (角型カッコ) コマンド、ファイル名などが角型カッコに囲まれているときは、それが省略可能な項目であることを示します。

< > (山型カッコ) 山型カッコは、その中で指示されている項目を入力することを示します。

例

```
copy <ファイルスペック 1> <ファイルスペック 2>
rem <コメント>
```

{ } (大カッコ) 大カッコに囲まれているものは、その中にあるもののうちの 1 つを必ず入力しなければならないことを示します。

| (縦線) 縦線で区切られた項目から 1 つを選んで入力します。

例

```
VERIFY {ON | OFF} (ON か OFF のどちらか選ぶ)
```

... (省略記号) 直前の項目が必要に応じて何回も繰り返されることを示します。

CAPS (大文字) 大文字は、そのつづりどおりに入力しなければならないコマンドまたはパラメータの一部分であることを示します。

スペース (区切り記号)

「 」はコマンドとパラメータ、またはパラメータとパラメータとの間の区切りを示します。区切り記号にはスペース、タブ、カンマが使用できます。



2 章

MSX-DOS の操作

2.1 MSX-DOS の起動と終了

2.1.1 MSX-DOS で使用するディスクの種類

MSX-DOS では、次の 2 つのタイプのディスクを使用します。

1. 3.5 インチマイクロフロッピーディスク (1DD タイプ)

1DD とは、片面倍密度倍トラック (Single sided、Double density、Double track) の意味で、フォーマット時の記憶容量は 360K バイトです。

2. 3.5 インチマイクロフロッピーディスク (2DD タイプ)

2DD とは、両面倍密度倍トラック (Double sided、Double density、Double track) の意味で、フォーマット時の記憶容量は 720K バイトです。

2.1.2 MSX-DOS の起動

MSX-DOS を起動するには、以下の手順で行います。

1. MSX コンピュータとディスプレイ、ディスクドライブ、プリンタ (必要に応じて装備) などの周辺機器が正しく接続されていることを確認します。
2. 周辺機器の電源を入れます。
3. システムディスクをドライブ A に挿入します。ここで「ドライブ A」とは、次のような

ドライブを指します。

- 接続しているドライブが1台のときは、そのドライブ。
- 接続しているドライブが2台のときは、ディスクインターフェイスカートリッジに接続されているドライブ。
- なお本体内蔵型のドライブを使用している方は、「2.1.4 4. 複数のドライブを用いるときのドライブ番号」を参照して下さい。

4. MSX コンピュータの電源を入れます。

これで、MSX-DOS がディスクからメモリに読み込まれ、ディスプレイに次のように表示されます。

MSXDOS.SYS のバージョン番号を表す。



MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11



COMMAND.COM のバージョン番号を表す。

時計機能を装備していない MSX コンピュータの場合、ディスプレイには続いて次のように表示されます。

Current date is Sun 84-01-01
Enter new date : █



これがカーソル

ここで日付を入力して下さい。入力を間違えたときは、BS(バックスペース)キーを押してカーソルをその場所まで戻し、入力しなおします。

Enter new date: 85-9-2

A>

時計機能を装備した MSX コンピュータでは、内蔵した時計の時刻と日付が自動的に参照されるので、日付の入力は必要ありません。

MSX-DOS version 1.03
Copyright 1984 by Microsoft

COMMAND version 1.11
A>

MSX-DOS が起動すると、ディスプレイに「A>」という記号が表示されます。この記号は MSX-DOS の入力促進記号(プロンプト)で、これが表示されている時はコンピュータが入力を受けられる状態であることを示します。この状態を「MSX-DOS のコマンド入力待ち状態にある」といい、プロンプトに続くコマンド行に様々なコマンド名を入力し、プログラムを実行することができます。

プロンプトにある「A」は、現在 MSX-DOS が優先して使用するドライブ (デフォルトドライブ) の名前前で、「B>」ならばデフォルトドライブがドライブ B であることを示します。ここで「d:」(d はドライブ名) を入力すると、デフォルトドライブを変更することができます。この結果プロンプトのドライブ名は指定したものに変わります。たとえばデフォルトドライブを A から B に変更するとき、次のように入力します。

```
A>B:  
B>
```

なお、MSX-DOS を起動した時点でのプロンプトは「A>」です。

2.1.3 MSX-DOS の終了

MSX-DOS での処理が終わったら、次のような手順で MSX-DOS を終了します。

1. 最後に実行したコマンドまたはプログラムの処理が終了していることを確認します。処理が終わっていれば、ディスプレイにはプロンプトが表示されています。
2. ディスクドライブからディスクを取り出します。
3. MSX コンピュータの電源を切ります。
4. 各周辺機器の電源を切ります。

最後に実行したコマンドまたはプログラムの処理が終了していないうちにコンピュータの電源を切ると、ディスクに記録されていたファイルが正しく読み出せなくなってしまう可能性があるので注意して下さい。

2.1.4 ハードウェア構成による使用上の違い

MSX-DOSは、時計機能の有無、MSX1 かそれとも MSX2 (MSX2+) か、ディスクドライブが何台接続されているかなど、コンピュータを構成するハードウェアの違いによってその使用法が異なります。

1. 時計機能

時計機能は MSX1 ではオプションで、MSX2 と MSX2+ では標準装備です。

時計機能を装備した MSX は、日付、時刻を自動的に更新し、閏年、月の大小、曜日なども管理します。また、MSX-DOS の起動時には、時計が持つ日付と時刻が自動的に参照されるので、日付の入力は必要ありません。ただし、長時間電源を切ったままにしておくとも時計の動作が停滞り、正しい時刻と日付を示さなくなることがあります。そのようなときは、「date」コマンドと「time」コマンドを使って、正しい日付と時刻に修正して下さい。なお「date」コマンド、「time」コマンドについてはコマンドの解説を参照して下さい。

2. MSX と MSX2 (MSX2+)

MSX と MSX2 では、ディスプレイの最大表示文字幅が異なり、MSX では 40 字、MSX2 と MSX2+ では 80 字になっています。なお、MSX2 と MSX2+ の機能として、文字の表示幅、画面の前景色、背景色、周辺色、ビーブ音などを、起動時に自動的に設定することができます。これらは、BASIC で好みの値に設定し「SET SCREEN」命令、および「SET BEEP」命令でバッテリバックアップした SRAM 内に記憶され、起動時にはこの値が用いられます。

3. 仮想ドライブ機能

MSX-DOS は、1 台のドライブでも使用できます。これは MSX-DOS が、1 台のドライブをあたかも 2 台のドライブが接続されているかのごとく動作させる、仮想ドライブ機能(2 ドライブシミュレータ) を備えているためです。

表 3.1 に、内蔵型のディスクを持たず、ディスクインターフェイスカートリッジが 1 台のときの、仮想ドライブ機能の有無を示します。

表 3.1 仮想ドライブ機能の有無

ディスクインターフェイスカートリッジ	ドライブ数	仮想ドライブ機能
1 台	1 台	有
1 台	2 台	無

■仮想ドライブ機能が動作しているときの操作方法

MSX-DOS は、ディスクインターフェイスカートリッジ1つに1台のドライブしかつながっていないときに、標準で仮想ドライブ機能が動作します。

MSX-DOS は仮想ドライブ（ここではドライブ B）を扱うコマンドを受け取ると、そのコマンドの処理中に次のようなメッセージを表示します。

Insert diskette for drive B:
and strike a key when ready

（ドライブ B 用のディスクを入れ、どれかキーを押して下さい）

ここで、その処理に必要なもう1枚のフロッピーディスクと入れ換えます。すると、MSX-DOS はそのフロッピーディスクをもう1台の仮想ドライブとみなして処理を行います。その処理が終って、もとのディスクに対する処理が必要になると、再びディスプレイに次のようなメッセージを表示します。

Insert diskette for drive A:
and strike a key when ready

（ドライブ A 用のディスクを入れ、どれかキーを押して下さい）

このメッセージにしたがって処理を行えば、2台のドライブを必要とする処理を、1台のドライブで行うことができます。このように仮想ドライブ機能は、MSX-DOS を1台のドライブで運用する上で、重要な役割を果たしています。

■仮想ドライブ機能を動作させないとき

2台のディスクインターフェイスカートリッジを用いて、それぞれに1台のドライブを接続したときなど、仮想ドライブ機能を用いるとこえて操作が複雑になることがあります。そのときには、起動時にビーブ音が鳴るまで **CTRL** キーを押し続けて、仮想ドライブ機能を動作させないようにして下さい。

4. 複数のドライブを用いるときのドライブ番号

図 3.1～3.6 に、複数のドライブを接続するとき、どのようにドライブ番号が割り当てられるかを示します。カッコ内は仮想ドライブ機能が動作している場合です。起動後のドライブ番号がこの解説と異なるときは、使用しているハードウェアのマニュアルを参照して下さい。

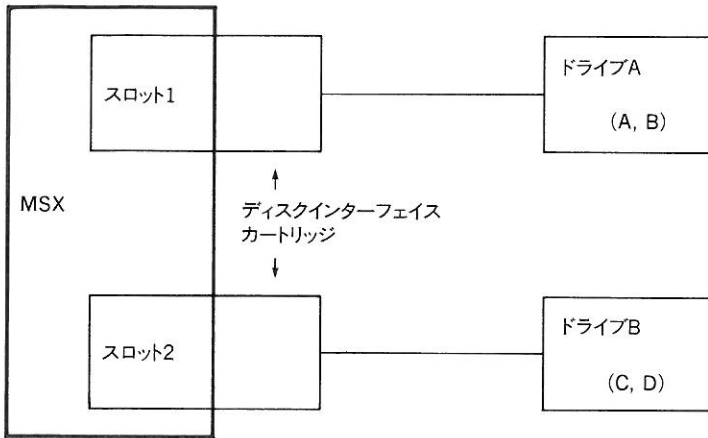


図 3.1 2 台のディスクインターフェイスにドライブを 1 台ずつ接続

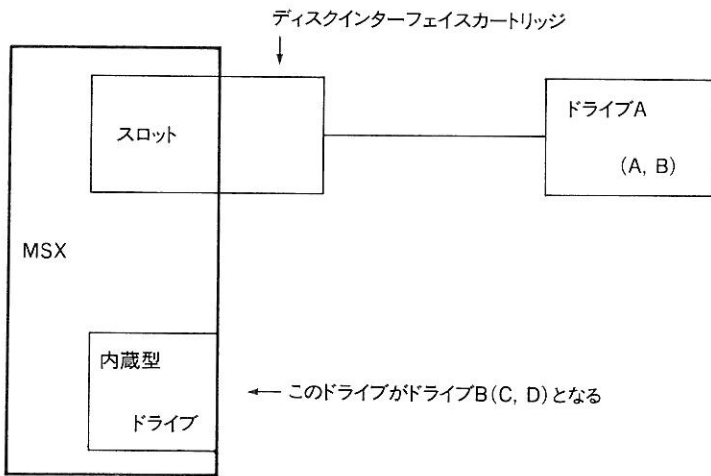


図 3.2 内蔵型ドライブ 1 台+ディスクインターフェイスにドライブを 1 台接続

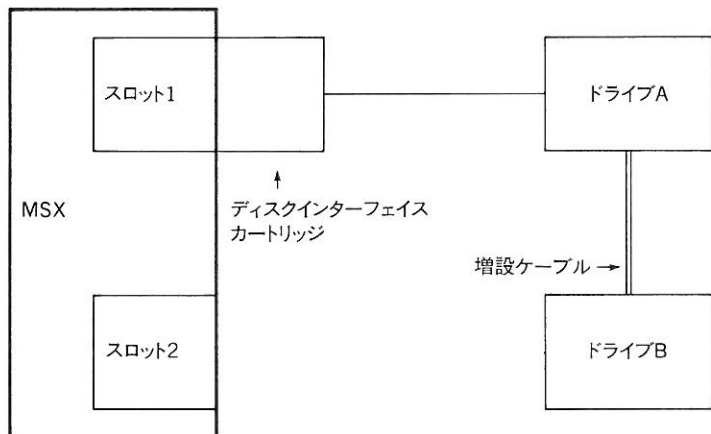


図 3.3 1 台のディスクインターフェイスに外付け型ドライブを 2 台接続

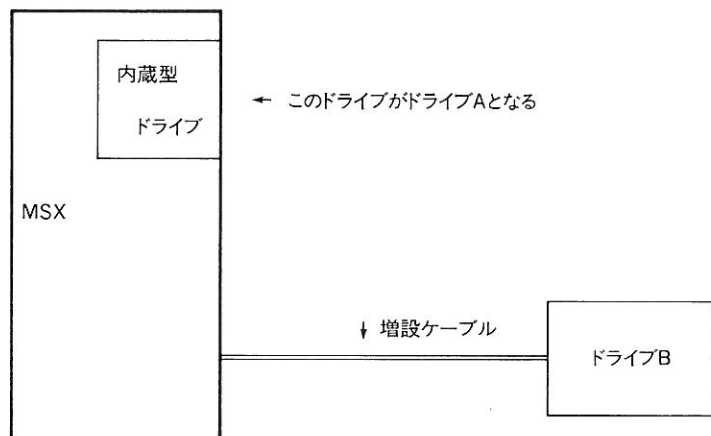


図 3.4 内蔵型ドライブ 1 台 + 増設ケーブルにより外付け型ドライブを 1 台接続

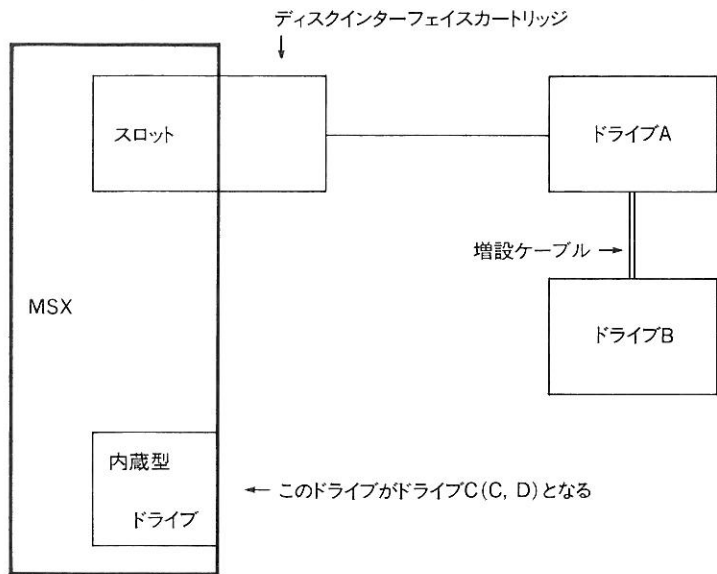


図 3.5 内蔵型ドライブ 1 台+ディスクインターフェイスに外付け型ドライブを 2 台接続

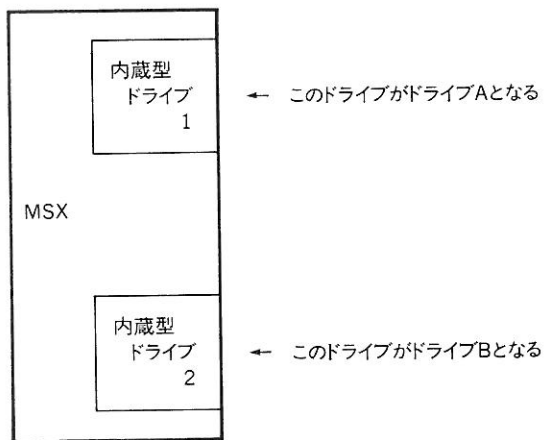


図 3.6 2 台の内蔵型ドライブ

2.2 ファイル

2.2.1 ファイルの概要

コンピュータは、外部記憶装置としてフロッピーディスクドライブ、クイックディスク、データレコーダ（テープレコーダ）や紙テープ装置などを使用します。これらの外部記憶装置に記録されたプログラムやデータを総称してファイルといいます。

DOSは、このようなファイルの登録、既存のファイルへの追加、修正、ファイルの削除などの処理を行う機能を持っています。このような機能（それを実現するプログラム）を、ファイルマネージメントシステム、または単にファイルシステムと呼んでいます。

DOSが扱うファイルとは、普通ディスクに記録されるディスクファイルです。しかし、MSX-DOSでは、プリンタなどの周辺機器をもファイルとして管理しています。このため、周辺機器を扱うためのプログラムをわざわざ開発する必要がなく、周辺機器の取り扱いが簡単になります。各周辺機器には特殊なファイル名が割り当てられており、このファイル名はディスクファイル名と同様にコマンド行で使用します。

2.2.2 ディスクファイル

MSX-DOSではディスクファイルの管理を、それぞれのディスクに用意されているFAT(File Allocation Table)とディレクトリ(Directory)を用いて行います。FATとは、ディスクでのファイルの配置状況を記録した地図のようなものです。また、ディレクトリは、ファイル名とそのファイルに関する情報を記録した索引です。DOSを利用するときはこのうちディレクトリにあるファイル名だけを意識していればよく、実際のファイル管理はすべてDOSが行います。

以下のように「dir」コマンドを用いてディレクトリを見ると、そのディスクにどんなファイルがあり、そのファイルの大きさといつ修正されたか、そしてディスクの残り容量がわかります。

```
A>dir
MSXDOS   SYS      2432 85-08-23  9:29p
COMMAND  COM      6656 85-09-02 10:10p
          2 files 352256 bytes free
A>
```

1. ファイル名

ファイル管理の中心となるものは、そのファイルに付けられたファイル名です。ファイル名は、1～8文字のファイル名と1～3文字の拡張子からなっていますが、拡張子は必ずしも必要ではありません。また、ファイル名と拡張子の間はピリオド「.」で区切ります。ファイル名として使用できる文字は次のとおりです。

A～Z 0～9 \$ & # @ ! % ' ()
 - { } ~ ひらがな カタカナ

なお、アプリケーションプログラムによっては、ファイル名として一部の文字が使えないことがあります。そのときにはA～Z、0～9を使用して下さい。

上記の文字は、ファイル名や拡張子のどの部分にも使うことができます。その際、アルファベットの小文字は、MSX-DOSによって大文字に変換されます。

拡張子は、ファイルの種類を区別しやすくするために使います。拡張子には、MSX-DOSでその使用が決っているものや、アプリケーションによっては習慣的に特定の拡張子を用いるものがあります。そのようなときには、必ずその拡張子を使って下さい。

■ MSX-DOS でその使用が決っている拡張子

拡張子	ファイルの種類	例
.BAT	バッチファイル	AUTOEXEC.BAT
.COM	コマンドファイル	COMMAND.COM
.SYS	システムファイル	MSXDOS.SYS

■ 習慣的に使用する拡張子

拡張子	ファイルの種類
.ASM	アセンブラソースファイル
.BAK	バックアップファイル
.BAS	ベーシックプログラムファイル
.C	C ソースファイル
.COB	COBOL ソースファイル
.CRF	クロスリファレンスファイル
.DAT	データファイル
.FIF	FORTH ソースファイル
.FOR	FORTRAN ソースファイル
.HEX	インテル HEX オブジェクトファイル
.LIB	アセンブラライブラリファイル
.M80	アセンブラライブラリファイル
.MAC	アセンブラソースファイル
.PAS	PASCAL ソースファイル
.PLI	PL/I ソースファイル
.PRN	リスティングファイル
.REL	リロケートブルオブジェクトファイル

なお、アプリケーションによっては、使う拡張子が決っているものがあります。

2. ワイルドカード文字

ワイルドカード文字とは、ファイル名を指定する際に、任意の1字または文字列に対応して、その文字または文字列の代りに用いることができる省略記号のことです。ワイルドカード文字を用いると、ファイルを指定する際にいくつかのファイルをまとめて指定することができます。ワイルドカード文字には、任意の1字に対応する「?」（クエスチョンマーク）と、任意の文字列に対応する「*」（アスタリスク）の2種類があります。

「dir」コマンドを例に、ワイルドカード文字の使い方を解説します。「dir」コマンドにワイルドカード文字を用いれば、限定されたファイルについての情報を表示させることができます。

ここでは、ドライブ A に次のようなディスクが入っていたとして述べます。

```
A>dir
ABCDE   EXT   1664 85-07-17  9:18a
ABXDE   EXT    256 85-07-26  6:56p
ABYDE   EXT   1280 85-07-31  3:48p
ABCZDE  COM   9984 85-08-09 12:08p
XYZ     COM   8320 85-08-25  3:39a
      5 files 337920 bytes free
A>
```

■「?」(クエスチョンマーク)

「dir」のパラメータに「ab?de.ext」というファイル名を指定したとします。この場合、「dir」は以下に示すようにファイル名が全体で5文字でABで始まりDEで終わり、拡張子が「.EXT」のすべてのファイルを表示します。

```
A>dir ab?de.ext
ABCDE   EXT   1664 85-07-17  9:18a
ABXDE   EXT   256 85-07-26  6:56p
ABYDE   EXT   1280 85-07-31  3:48p
        3 files  337920 bytes free
A>
```

なお、「?」は、拡張子の中でも同様に使うことができます。

■「*」(アスタリスク)

ファイル名または拡張子の中に「*」があると、「*」はその位置以降の文字列を「?」に置き換えられます。つまり、「*」は連続した「?」の略記と同じです。次の2つのコマンドは全く同じ意味で、そのディスクに記録されているすべてのファイルを表示します。

```
A>dir *.*
A>dir ???????.???
```

次のコマンドは、拡張子が「.COM」の全てのファイルを表示します。

```
A>dir *.com                               ← 「dir ???????.com」と同じ
ABCZDE  COM   9984 85-08-09 12:08p
XYZ      COM   8320 85-08-25  3:39a
        2 files  337920 bytes free
A>
```

次のコマンドは、ファイル名がABCで始まる全てのファイルを表示します。

```
A>dir abc *.*                               ← 「dir abc?????.???'と同じ
ABCDE   EXT   1664 85-07-17  9:18a
ABCZDE  COM   8320 85-08-25 12:08p
        2 files  337920 bytes free
A>
```

3. ドライブ指定

MSX-DOS はファイルをディスクドライブごとに管理しています。そのため、違うドライブに同じ名前前のファイルがあっても、問題はありません。

MSX-DOS が管理するドライブの数は最大 8 台で、それぞれ A、B、C～H のアルファベットのドライブ名を持っています。ドライブを指定するときには、この文字の後ろに「:」（コロン）を付け、「d:」というように指定します。この文字とコロンの組み合わせを、「ドライブ指定」といいます。

また、ドライブ指定、ファイル名、拡張子をこの順で続けたものを、特に「ファイルスペック」といいます。ファイルスペックの書式は以下のとおりです。

[<ドライブ指定>]<ファイル名>[.<拡張子>]

例

A:BCDEFGHI.BAS

なお、デフォルトドライブにあるファイルを指定するときは、ドライブ指定は省略することができます。

2.2.3 特殊なファイル名

MSX-DOS では周辺機器に特殊なファイル名を割り当て、各周辺機器をディスクファイルと同じようにコマンド行で取り扱えるようにしています。この特殊なファイル名を「デバイスファイル名」といいます。デバイスファイル名には次のものがあります。

デバイスファイル名	機能
AUX	データレコーダなどの補助入出力装置との入出力や他のコンピュータとの通信のために用意してあります。起動時は、NUL と同じ機能に設定してあります。
CON	キーボードからの入力や、ディスプレイへ出力するときに使用します。
NUL	コマンドがその構文上、入出力のファイル名を要求していても、特にファイルを作らないときに使用します。
PRN	プリンタに出力するときに使用します。LST もこれと同じです。

デバイスファイル名を用いた例を以下に 2 つあげます。

```
A>copy con autoexec.bat
```

キーボードからの入力を「autoexec.bat」というファイル名でデフォルトドライブ上に作ります。


```
A>copy test.mac prn
```

「test.mac」というファイルの内容をプリンタに打ち出します。

なお、デバイスファイルを使用するとき、たとえドライブ指定や拡張子が指定されていても、デバイスファイル名が優先します。したがって、CON.LSTはファイルにはなりません。例えば、下のようなコマンドを入力しても、CON.LSTというディスクファイルは作られず、SAMPLE.MACの内容がディスプレイに表示されます。

```
A>copy sample.mac con. 1st
```

2.3 コマンドの概要

2.3.1 コマンドとは

1. コマンドの概念

BASICなどのプログラミング言語では、コマンドは1つのプログラムを構成する部分品でした。しかしMSX-DOSでいうコマンドとは、プログラミング言語のコマンドとは違い、プログラムそのものです。コマンドは、プロンプトに続くコマンド行にその名前を入力することで実行されます。

MSX-DOSで実際にコマンドの実行を受け持つのはシステムコマンドファイルである「COMMAND.COM」です。すべてのコマンド（プログラム）は、「COMMAND.COM」が解析し、実行されます。

2. コマンドの書式

コマンドは、コマンド名とそれに続くパラメータからなっています。コマンドによって、パラメータが必要なものとそうでないものがあります。また、コマンド名とパラメータ、パラメータ相互の間は、スペース、タブまたはカンマによって区切ります。コマンドの書式は以下のようになります。

```
A>[<ドライブ指定>]<コマンド名>[ <パラメータ>]...
```

これは、<ドライブ指定>で指定されたドライブにある<コマンド名>というコマンドを実行せよ、ということです。

2.3.2 コマンドの種類

コマンドには、MSX-DOS の内部にあらかじめ用意されている内部コマンドと、ディスクにファイルとして記録されていて、コマンド名が入力されるたびにメモリに読み込まれて実行される外部コマンド、MSX-DOS のバッチ処理機能により自動的に複数の内部コマンド、外部コマンドを実行するバッチコマンドの3種類があります。

1. 内部コマンド

MSX-DOS には、以下の13種類の内部コマンドがあります。それぞれのコマンドの詳細については、「2.4 コマンド一覧」を参照して下さい。

BASIC	DIR	REN	VERIFY
COPY	FORMAT	REM	
DATE	MODE	TIME	
DEL	PAUSE	TYPE	

2. 外部コマンド

アセンブラやコンパイラなどを利用して作成した「.COM」という拡張子を持つファイルが外部コマンドです。システムディスクの「COMMAND.COM」もこれにあたります。

外部コマンドはファイル名をコマンド行から入力すると、ディスクから読み込まれて実行されます。なお、コマンドファイルがデフォルトドライブにないときは、コマンド名にドライブ指定をつけるか、デフォルトドライブをコマンドファイルがあるドライブに変更しなければなりません。

以下に外部コマンドとデフォルトドライブの関係を示します。例えば、ドライブ A、B に、それぞれ次のようなディスクが入っているとします。

```
A>dir
TEST      COM      384 85-07-25  6:16p
NEWDISK   BAT      78 85-08-03 10:02p
          2 files 360448 bytes free

A>dir b:
CAT       COM     8320 85-08-20  3:04p
          1 files 353280 bytes free

A>test
```

← 「test」を入力

TEST というコマンド（プログラム）を実行

```
A>cat
Bad command or file name
```

←「cat」を入力
←CAT.COMがデフォルトドライブであるドライブ A 上になかったので、エラーメッセージが表示された。

```
A>b:cat
```

CAT を実行

```
A>b:
B>cat
```

←デフォルトドライブをドライブBに変更
←「cat」を入力

CAT を実行

```
B>
```

3. バッチコマンド

バッチコマンドとは、拡張子が「.BAT」であるバッチファイルに記録した一連のコマンドで、バッチファイル名をコマンド行から入力すれば、記録されているコマンドが次々と実行されます。バッチコマンドを利用することで、個々のコマンドをいちいち入力することなく、一連の処理を自動的に行うことができます。このため、アセンブルやコンパイルなど、同じコマンドを使って何回も処理を行うときには、非常に便利な機能です。

バッチファイルがデフォルトドライブにないとき、バッチファイル名の前にドライブ指定をつけるか、デフォルトドライブを変更しなければなりません。これは、外部コマンドのときと同じです。

バッチファイル「NEWDISK.BAT」が次のような内容であるとしします。ファイルの内容をディスプレイに表示させるには、「type」コマンドを使います。

```
A>type newdisk.bat
rem This is file NEWDISK.BAT
pause Insert new disk in drive b:
format
dir b:
```

```
A>
```

このバッチファイル名を入力すれば、バッチファイルの内容が次々と実行されます。

```
A>newdisk
A>rem This is file NEWDISK.BAT
A>pause Insert disk in drive B:
Strike a key when ready
```

←ドライブ B に新しいディスクを入れたらどれかキーを押す

```
A>format
Drive name? (A, B) b
Strike a key when ready
Format complete
```

←ドライブ B のディスクをフォーマット
←どれかキーを押す
←フォーマットが終了すると表示される

```
A>dir b:
File not found
A>
```

←ドライブ B のディスクにはファイルがない

2.4 コマンド一覧

ここでは MSX-DOS の内部コマンドについて解説します。コマンドはアルファベット順になっています。

コマンド名	機能	ページ
BASIC	Disk BASIC に移行します。	355
COPY	ファイルをコピーまたは連結します。	356
DATE	日付を表示、変更します。	360
DEL (ERASE)	ファイルを削除します。	362
DIR	ディレクトリを表示します。	363
FORMAT	ディスクを初期化します。	364
MODE	文字表示幅を設定します。	366
PAUSE	バッチコマンドの実行を一時停止します。	367
REM	バッチファイルにコメントを書き込みます。	368
REN (RENAME)	ファイル名を変更します。	368 (369)
TIME	時刻を表示、変更します。	369
TYPE	ファイルの内容を表示します。	370
VERIFY	書き込みの際の検査の有無を設定します。	371

BASIC

機能

Disk BASIC を起動します。

書式

BASIC [<ファイルスペック>]

解説

MSX-DOS が起動されている状態から Disk BASIC を起動したいときは、「basic」コマンドを用います。<ファイルスペック>によって Disk BASIC のプログラムファイルを指定すると、Disk BASIC を起動した後、そのプログラムを読み込んで実行します。

Disk BASIC から MSX-DOS へ移行するには「_system (または call system)」命令を実行します。

例

Disk BASIC を起動し、アスキーセーブされた BASIC プログラムの「FUNCKEYS.INI」を実行します。このプログラムは、Disk BASIC でファンクションキーを設定して、再び MSX-DOS に戻るものです。

```
A>type funckeys.ini
100 KEY 1, "dir "
110 KEY 2, "ren "
120 KEY 3, "copy "
130 KEY 4, "del "
140 KEY 5, "type "
150 KEY 6, "time "+CHR$(13)
160 KEY 7, "date "+CHR$(13)
170 KEY 8, "format "+CHR$(13)
180 KEY 9, "verify "
190 KEY 10, "basic "
200 CLS:CALL SYSTEM
```

```
A>basic funckeys.ini
```

注意

MSX-DOS と Disk BASIC ではメモリ構成が異なり、「basic」コマンドの実行によってメモリが切り換えられます。このため、メモリを介して MSX-DOS と Disk BASIC との間でデータをやりとりすることはできません。

COPY

機能

ファイルを複写（コピー）、または連結（付加）します。

書式 1

```
COPY [/A | /B] <ファイルスペック 1> [/A | /B] [<ファイルスペック 2> [/A  
| /B]]
```

書式 2

```
COPY [/A | /B] <ファイルスペック 1> [/A | /B] + <ファイルスペック 2> [/A  
| /B] [••• + <ファイルスペック n> [/A | /B]] [<ファイルスペック> [/A | /B]]
```

解説

<ファイルスペック>の指定にはワイルドカード文字を使うことができます。普通、「copy」コマンドは、ファイルを複写するために使用します。

■同じディスクへのファイルの複写

以下は、「MSXDOS.SYS」というファイルを「MSXDOS.BAK」というファイル名で同じディスクに複写するときのコマンド行の例です。

```
A>copy msxdos.sys msxdos.bak
```

ここで<ファイルスペック 1>と<ファイルスペック 2>は、必ず違うファイル名でなければなりません。同じファイル名を指定すると、「copy」コマンドは中断され（ファイルをそれ自身にコピーすることは許されません）、次のようなエラーメッセージが表示されます。

```
A>copy msxdos.sys msxdos.sys  
File cannot be copied onto itself  
0 files copied  
A>
```

■違うディスクへのファイルの複写

このとき、<ファイルスペック 2>には4つの形があります。

1. 省略されたとき

コピーはデフォルトドライブに対して行われ、コピー後のファイルには<ファイルス

ベック 1>と同じ名前がつけられます。次の例では、ドライブ B の「msxdos.sys」がデフォルトドライブであるドライブ A にコピーされます。

```
A>copy b:msxdos.sys
```

<ファイルスペック 1>がデフォルトドライブ上のもので、<ファイルスペック 2>が省略されていると、「copy」コマンドは中断され(ファイルをそれ自身にコピーすることは許されません)、次のようなエラーメッセージが表示されます。

```
A>copy msxdos.sys
File cannot be copied onto itself
0 files copied
A>
```

2. ドライブ指定のみのとき

指定されたドライブに同じ名前でもコピーします。

```
A>copy a:msxdos.sys b:
```

3. ファイル名だけのとき

指定されたファイル名でデフォルトドライブにコピーします。

```
A>copy b:msxdos.sys msxdos.bak
```

4. 完全なファイルスペックを指定したとき

指定されたドライブへ指定された名前でもファイルをコピーします。

```
A>copy msxdos.sys b:msxdos.sys
```

■「copy」コマンドのスイッチ

/A スイッチと/B スイッチでコピーモードを指定します。

/A アスキーモード (Ascii mode) の意味で、処理しているファイルの中にエンドオブファイル文字 (EOF、ファイルの最後に付けられる) があると、そこでファイルが終っているとみなして処理を終了します。

/B バイナリモード (Binary mode) の意味で、EOF があっても処理を続けます。

<ファイルスペック>にスイッチまたはがついているとき、そのスイッチは直前の<ファイルスペック>から次のスイッチが指定されるまで有効です。/A が指定されている間に読み取られるファイルは、EOF 以降がコピーされません。

コピー先のファイル名につけられるスイッチは、EOF がファイルの終りに置かれるかどうかを決定します。/A スイッチが指定されていると、ファイルの書き込みでは EOF が

1つだけつけられます。したがって、アスキーファイルを

```
A>copy ab.mac/B cd.mac/A
```

というコマンドでコピーすると、

- もとのファイルの EOF は /B スイッチのため除去されず、
- コピー先のファイルの /A スイッチにより余分な EOF が付加されるので、

結果として2つの EOF が付いたファイルができることになります。

次はファイルを連結するときにスイッチがついた例で、実行ファイル「PROG.COM」に、アスキーファイルである定数データ「ERRS.TXT」を連結するときのもです。結果として、必要なのがバイナリファイルとすると、終りの EOF は必要ないのでスイッチをこのように用います。

```
A>copy prog.com/b+errs.txt/a newprog.com/b
```

スイッチを指定しないで複写すると、バイナリモードでコピーされます。

```
A>copy /a ab.txt cd.txt
```

このとき、「AB.TXT」の途中に EOF が含まれていると、その時点でコピーは終了するので、「CD.TXT」の大きさは「AB.TXT」より小さくなることがあります。なお、このとき「CD.TXT」には、EOF が最後の文字としてついています。

■ファイルの連結

「copy」コマンドではファイルを連結することもできます。連結は、「copy」コマンドのパラメータとして<ファイルスペック>を「+」でつなげて指定します。

以下の例では、「AB.TXT」、「CD.TXT」、「EF.TXT」を連結し、その結果を「GH.CRP」という名前でドライブ A に書き込みます。

連結は、通常アスキーファイルで行います。バイナリファイルを連結するときは、ファイルの中に EOF が含まれていると、その時点でそのファイルのコピーを打ち切り、次のファイルを続けて連結コピーします。そのため、バイナリファイルを連結するときは /B スイッチによりバイナリモードに切り換えて行います。

```
A>copy ab.txt+cd.txt+ef.txt gh.txt
```

以下の例では、「AB.COM」に「CD.COM」を連結し、「EF.COM」というファイルを作ります。

```
A>copy /b ab.com+cd.com ef.com
```

次の例は、ドライブ B にある「TEST1.TXT」に、デフォルトドライブにある「TEST2.TXT」を付加するときのコマンド行です。


```
A>copy b:test1.txt+test2.txt b:test1.txt
```

ただし、もともになるファイルと付加するファイルが双方ともデフォルトドライブ上にあるときは、最後のパラメータを省略することができます。

■ワイルドカード文字の使用例

1. 複写（コピー）

ワイルドカード文字を利用すれば、ディスクのバックアップや特定のファイル名または拡張子を持つファイルのみをコピーすることができます。

次の例では、デフォルトドライブであるドライブ A のすべてのファイルをドライブ B にコピーします。

```
A>copy *.* b:
```

次の例では、デフォルトドライブにある拡張子が「.COM」であるファイルのすべてを、ドライブ B にコピーします。

```
A>copy *.com b:
```

次の例では、ドライブ B にあるファイル名が3字以下であるファイルのすべてを、デフォルトドライブ（ドライブ A）にコピーします。

```
A>copy b:???.*
```

2. 連結（付加）

ワイルドカード文字を利用すれば、特定のファイル名または拡張子を持つファイルをすべて連結し、1つのファイルを作るなどの処理ができます。

次の例では、拡張子が「.LST」であるすべてのファイルを連結して、「CONBIN.PRN」という名前のファイルを作ります。

```
A>copy *.lst conbin.prn
```

また、以下のようにいくつかのファイルを個々に連結したり、または1つのファイルに連結したりすることもできます。例えば、ドライブ A に次のようなファイルがあるとします。

```
A>dir
FILE1      LST      .....
ABCDE     LST      .....
FILE1      REF      .....
ABCDE     REF      .....
          :
```

このとき、次のようなコマンドを入力すると、「FILE1.LST」と「FILE1.REF」が連結されて「FILE1.PRN」という名前のファイルが作成され、「ABCDE.LST」と「ABCDE.REF」が連結されて「ABCDE.PRN」という名前のファイルができます。

```
A>copy *.lst+*.ref *.prn
FILE1      LST
ABCDE      LST
          2 files copied

A>dir
FILE1      LST      .....
ABCDE      LST      .....
FILE1      REF      .....
ABCDE      REF      .....
FILE1      PRN      .....
ABCDE      PRN      .....
          .
          .
```

次の例では、「*.LST」に該当するすべてのファイルを連結してから、「*.REF」に該当する全ファイルを連結し、「COMBIN.PRN」というファイルを作ります。

```
A>copy *.lst+*.ref combin.prn
```

注意

連結した結果作られるファイル名として、すでにディスクにあるファイルの名前を指定すると、そのファイルに連結するファイルの内容が重ね書きされます。結果としてもとのファイルの内容は書き換えられてしまうので、MSX-DOSは警告メッセージを表示します。

```
A>copy *.lst all.lst
Content of destination lost before copy
          1 file copied

A>
```

次の例では、拡張子が「.LST」である全ファイルを、「ALL.PRN」というファイルに付加します。

```
A>copy all.prn+*.lst
```

DATE

機能

日付の表示、変更を行います。

書 式

DATE [`<年>-<月>-<日>`]

解 説

パラメータなしで入力したときは次のように表示されます。

```
A>date
Current date is www yy-mm-dd
Enter new date:
```

www、yy、mm、dd は内蔵の時計が示す日付をもとに表示されます。

www 曜日を示します。曜日は日付から自動的に計算されます。表示されるのは次のうちの1つです。

Sun、Mon、Tue、Wed、Thu、Fri、Sat

yy 西暦年を示します。下2桁が表示されます。

mm 月を示します。2桁の数字です。

dd 日を示します。2桁の数字です。

日付を変更する必要がないときは、ここでリターンキーを押します。

パラメータとして日付を入力すると、そのまま新しい日付がセットされます。このときは、メッセージは表示されません。

```
A>date 85-9-2
```

日付の入力には数字だけが使用できます。文字は使えません。指定できるパラメータの範囲は次のとおりです。

```
年      1980~2079
        または 80~99 (1980~1999 と解釈される)
        または 00~79 (2000~2079 と解釈される)
月      1~12 (01~12)
日      1~31 (01~31)
```

年月日の入力、ハイフン (-) かスラッシュ (/) で区切って下さい。

日付は時刻とともに MSX-DOS が管理するので、時刻が24時になると自動的に更新されます。また、月の大小や閏年もチェックされ、正しく変更されます。ただし、時計機能がない MSX のときは更新はされません。

パラメータや区切り記号が誤っているときは、次のメッセージが表示されるので、もう一度正しく入力して下さい。

```
Invalid date      ←日付の指定が違います
Enter new date    ←日付を入力して下さい
```

注意

本体の ROM BIOS が日本版以外のものでは、日付の表示と入力の形式が次のように変わります。

インターナショナル版	mm-dd-yyyy
ヨーロッパ版	dd-mm-yyyy

DEL

機能

<ファイルスペック>で指定されたファイルを削除します。

書式

DEL <ファイルスペック>

解説

<ファイルスペック>で指定されたファイルを削除します。

次の例は、デフォルトドライブからファイル「TEST.MAC」を削除します。

```
A>del test.mac
```

「del」コマンドでは、削除するファイルの指定にワイルドカード文字が使用できます。例えば、拡張子が「.TXT」のファイルをすべて削除するときは、次のように入力します。

```
A>del *.txt
```

ワイルドカード文字の詳細については、「2.2.1 ファイルの概要」を参照して下さい。ファイルの指定に「*.*」を用いると、ディスクにあるすべてのファイルを削除することになります。MSX-DOS は実行してもよいかどうかを確認するため、次のようなメッセージを表示します。全ファイルを削除してもよいときは、ここで「y」を入力します。

```
A>del *.*  
Are you sure (Y/N)?
```

なお、「del」と同等の機能を持つコマンドとして「erase」があります。

DIR

機 能

ディスクに記録されているファイルについての情報を表示します。

書 式

DIR [<ファイルスペック>][/P][/W]

DIR [/P][/W] [<ファイルスペック>]

解 説

「dir」コマンドはディレクトリに記録されているファイルの名前、大きさ、最後に修正された日付、時刻および表示したファイル数、ディスクの残り容量などの情報をディスプレイに表示します。

「dir」コマンドでは「/P」と「/W」の2つのスイッチが使用できます。

/P ページモードを意味し、ディスプレイいっぱいに表示されたところで表示を中断します。表示を再開するには、任意のキーを押します。

/W ワイドディスプレイモードを意味し、ファイル名のみを1行に表示できるだけ表示します。1行の表示幅は「mode」コマンドで設定した値によります。

「dir」コマンドにパラメータがないとき、デフォルトドライブ上にあるディスクのディレクトリの内容を表示します。デフォルトドライブ A にあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>dir
```

パラメータとしてドライブ指定のみがあるとき、指定されたドライブにあるディスクのディレクトリの内容を表示します。ドライブ B にあるディスクのディレクトリの内容を表示させるには、次のように入力します。

```
A>dir b:
```

パラメータとして拡張子なしのファイル名だけがあるとき（ドライブ指定はあってもよい）、そのドライブ上にあるディスクのディレクトリを検索して、指定されたファイル名を持つすべてのファイルの情報を表示します。次の例では、ドライブ B にあり、ファイル名に「TEST1」を持つすべてのファイルの情報を表示します。

```
A>dir b:test1
TEST1   ASM   .....
TEST1   REL   .....
```

```
TEST1      COM      .....
           3 files  .....
```

A>

完全なくファイルスペック>を指定したとき、指定されたドライブにあるディスクのディレクトリからそのファイルを検索し、その情報を表示します。

```
A>dir b:test1.asm
TEST1      ASM      .....
           1 file   .....
```

A>

ファイル名のパラメータとしてワイルドカード文字を使用することができます。ワイルドカード文字については、「2.2.1 ファイルの概要」を参照して下さい。なお、「dir」コマンドの呼び出して同等の動作をするものを以下に上げます。

コマンド	同等のコマンド形式
dir	dir *.*
dir file	dir file.*
dir .ext	dir *.ext
dir .	dir *.*

注意

表示幅が 36 桁未満のときは、情報の一部が表示されないことがあります。このようなときは「mode」コマンドを用いて、表示幅を 36 桁以上に設定して下さい。

ERASE

機能

DEL コマンドと同じです。DEL コマンドを参照して下さい。

FORMAT

機能

新しいディスクを使用できるように初期化します。

書式

FORMAT

解説

「format」コマンドの処理はメッセージに従って対話方式で進められます。標準的なメッセージは以下のとおりです。

■ドライブが1DDタイプ的时候

```
A>format
Drive name? (A, B)      ←フォーマットするディスクが入っているドライブ名
                           を入力
Strike a key when ready ←どれかキーを押す
Format complete
A>
```

■ドライブが2DDタイプ的时候

```
A>format
Drive name? (A, B)      ←ドライブ名を入力
1 - 1 side, double track
2 - 2 sides, double track
?                        ←数字を入力する(標準フォーマットはこの例では1)
Strike a key when ready ←どれかキーを押す
Format complete
A>
```

対話を中断するときは、**CTRL** + **C** を入力します。このとき、ディスプレイには次のようなメッセージが表示されます。

```
Aborted                ←中止した
```

注意

MSX-DOS で使用できるディスクのフォーマット (様式) を表 3.2 にあげます。

表 3.2 MSX-DOS で使用できるディスクのフォーマット

フォーマット	1DD	2DD	1DD	2DD
記録可能な総ファイル数	112	112	112	112
FAT が占めるセクタ数	2	3	1	2
1トラックのセクタ数	9	9	8	8
サイド数	1	2	1	2
サイドあたりのトラック数	80	80	80	80
1セクタのバイト数	512	512	512	512
使用可能なセクタ数	708	1426	630	1268

フォーマット	1DD	2DD	1DD	2DD
使用可能なセクタ数	708	1426	630	1268
使用可能な総バイト数	362496	730112	322560	649216
物理セクタ番号	1~9	1~9	1~8	1~8
トラック番号	0~79	0~79	0~79	0~79
サイド番号	0	0、1	0	0、1

いったんフォーマットが始まると中断することはできません。フォーマット中にエラーが発生すると、その内容によって次のようなメッセージが表示されます。

Write protected
Not ready
Disk error

エラーの内容は、「2.7 MSX-DOS メッセージ一覧」を参照して下さい。

3.5 インチ型の MSX 用ディスクドライブ装置では、全ての機種で片面 80 トラック (1DD) フォーマットを読み書きできるようになっています。他の装置で使う可能性のあるディスクをフォーマットするときは、このフォーマットを選ぶことが望ましいでしょう。

MODE

機能

ディスプレイの表示文字幅を設定します。

書式

MODE <表示文字幅>

解説

設定できる表示文字幅は、MSX1 では 1~40 です。41 以上の値を設定するとエラーになります。MSX2、MSX2+ のときは 1~80 です。設定した値が 1~32 のときはスクリーンモード 1 (BASIC の SCREEN 1) に、33~40 (MSX2、MSX2+ では 33~80) のときはスクリーンモード 0 (BASIC の SCREEN 0) に設定されます。

「mode」コマンドを実行するとディスプレイは消去されます。

PAUSE

機 能

バッチコマンドの実行を一時停止します。

書 式

PAUSE [<コメント>]

解 説

バッチコマンドの実行を一時停止させるときに使います。バッチファイルにこのコマンドを書き込んでおくと、バッチコマンドの実行が「pause」のところで一時停止します。このときディスプレイには以下のようなメッセージが表示されます。

Strike any key when ready

(準備ができたらどれかキーを押して下さい)

バッチコマンドの実行は、**CTRL** + **C** 以外の任意のキーを押すと再開されます。

CTRL + **C** を入力すると、続いて次のようなメッセージが表示されます。

Terminate batch file (Y/N)?

(バッチ処理を中止しますか?)

ここで「Y」を入力するとバッチコマンドの実行は中止され、MSX-DOS のコマンド入力待ちに戻ります。「N」を入力したときはバッチコマンドの実行が再開されます。

「pause」コマンドのパラメータにコメントを与えると、それをディスプレイに表示させることができます。例えば、下の例のように、使用者に指示を与えることもできます。

```
A>newdisk
A>rem This is NEWDISK.BAT
A>pause Insert disk in drive B:
Strike a key when ready
A>format
Drive name? (A, B) b
Strike a key when ready
Format complete
```

```
A>
```

REM

機能

何もしません。ただし結果として、「rem」のあとに続くパラメータがコメントとしてディスプレイに表示されます。

書式

REM [<コメント>]

解説

「rem」コマンドは上記の動作以外に、他に何の影響も与えません。「pause」コマンドの例を参照して下さい。

REN

機能

第1パラメータで指定したファイル名1を、第2パラメータのファイル名2に変更します。

書式

REN <ファイル名1> <ファイル名2>

または

RENAME <ファイル名1> <ファイル名2>

解説

デフォルトドライブ以外のファイル名を変更するときは、ファイル名1を指定する際にドライブ指定が必要です。「ren」コマンドはファイル名を変更するものなので、ファイル名2でドライブを指定しても意味を持ちません。また「ren」コマンドで他のディスクにファイルを移動させることはできません。

「ren」コマンドではワイルドカード文字を使用することができます。このとき、ファイル名1で指定したファイルとファイル名2で指定したファイルとの間で、各文字が1対1に対応して処理されます。例えば次の例では、「.LST」拡張子を持つ全てのファイルの拡張子を「.PRN」に変えます。

```
A>ren *.lst *.prn
```

次の例では、ドライブ B 上のファイル ABCDE を ADCBE という名前に変更することになります。

```
A>ren b:abcde ?d?b?
```

なお、「rename」は「ren」と同等のコマンドです。

RENAME

機能

REN コマンドと同じです。REN コマンドを参照して下さい。

TIME

機能

時刻の表示、変更を行います。

書式

```
TIME [<時>[:<分>[:<秒>]][{A | P}]
```

解説

パラメータなしで入力したときは次のように表示されます。

```
A>time
Current time is hh-mm-ss.tt {a | p}
Enter new time:
```

hh、mm、ss.tt、a または p は内蔵の時計が示す時刻です。

hh 1～12 の数字で時を示します。

mm 0～59 の数字で分を示します。

ss.tt ss は 0～59 の数字で秒を示します。また、tt は 1/100 秒を示しますが、表示される数字は「00」です。

a または p 午前 (a) か午後 (p) かを示します。

表示された時刻を変更する必要がないときはここでリターンキーを押します。

パラメータとして時刻を入力すると、メッセージは表示されずにそのまま時刻の変更が行われます。

時刻の入力には数字を使います。「.tt」を入力する必要はありません。また、最後の数字の後に「a」または「p」をつけて、午前 (am) と午後 (pm) を指定することができます。「a」や「p」をつけないと、24 時間制で時刻を指定したとみなされます。したがって、次の2つの例では、最初のは午前10時10分30秒に変更し、次のものは時刻を午後10時10分30秒に変更します。

```
A>time 10:10:30
A>time 10:10:30p
```

時刻の入力に使用できるパラメータの範囲は次のとおりです。

時	0～23	24 時間制で入力したと見なされる
	1～12 {a p}	a または p を省略すると午前と見なされる
分	0～59	
秒	0～59	

時：分：秒の入力は、コロン (:) で区切ります。パラメータや区切り記号が誤っていると、次のメッセージが表示されるので、もう一度正しく入力して下さい。

```
Invalid time      ←時刻の指定が違います
Enter new time:  ←時刻を入力して下さい
```

TYPE

機能

<ファイルスペック>で指定したファイルの内容をディスプレイに表示します。

書式

TYPE <ファイルスペック>

解説

「type」コマンドはアスキーファイルの内容を表示させるために用います。<ファイルスペック>の指定にはワイルドカード文字を使用できますが、ディレクトリを捜して最初に指定に該当したファイルの内容のみを表示します。

バイナリファイルの内容を表示させると、ベルコード、フォームフィードコード、およびエスケープシーケンスを含むコントロールシーケンスがディスプレイに送られ、正しい表示が行われません。

表示させるファイルの名前を捜すには「dir」コマンドを使います。

VERIFY

機 能

ディスクに書き込む際に、読み込みチェックをするかどうかを設定します。

書 式

VERIFY {ON | OFF}

解 説

「verify」コマンドは、書き込みに際してファイルが正しく書き込まれたことを読み出してチェックするかするかどうかを設定します。「verify」を「on」にすると、ディスクに書き込むごとにチェックを行い、「off」にするとチェックは行いません。

「verify on」のときは書き込みに時間がかかるので、通常は「off」にします。MSX-DOSの起動時には、「verify off」に設定されています。重要なファイルをコピーするとき、例えばマスターディスクの予備を作るときなどには、「verify on」にすると良いでしょう。

ベリファイ機能はオプションであり、ドライブによってはその機能がないものもあります。

2.5 特殊キーの機能

2.5.1 特殊キーとは

MSX-DOS で使用するキーには、文字キーの他に特別な機能を持つ「特殊キー」があります。これらのキーを使うことで、MSX-DOS の操作がより簡単になります。

特殊キーには、**CTRL**、**DEL**、**HOME**、**INS**、**SELECT** などがあります。特殊キーのほとんどは単独で使用されますが、**CTRL** キーだけは、それを押しながら他の文字キーを押して使います。

注意

CTRL + 文字は、**CTRL** キーを押しながら文字のキーを押すことを表します。

2.5.2 コマンド行の編集機能

1. 編集機能

MSX-DOS には、直前に入力されたコマンド行を記憶し、それを再び実行したり、修正して実行する機能があります。この機能を利用すれば、同じようなコマンドを繰り返し入力する手間を減らすことができます。この記憶されたコマンド行を「テンプレート」といいます。

図 3.7 は入力されたコマンド行がどのように記憶され、呼び出されるかを表したものです。

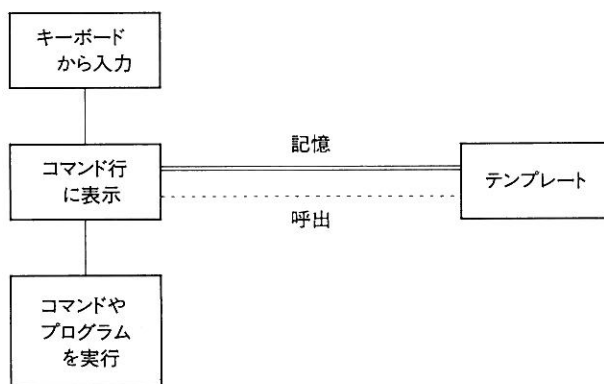


図 3.7 コマンド行とテンプレート

コマンド行の編集に使われる特殊キーの機能を表 3.3 に示します。1 つの機能に複数のキーが割り当てられているときは、どのキーを用いても同じ効果があります。

表 3.3 特殊キーの機能

特殊キー	機能	内容
→ CTRL + ¥	1字コピー	テンプレートからコマンド行へ1文字コピーします。
SELECT CTRL + X	指定コピー	テンプレートからコマンド行へ、指定された文字の直前までの文字をコピーします。
↓ CTRL + _	1行コピー	テンプレート内に残っている全ての文字をコマンド行へコピーします。
DEL	1字スキップ	テンプレート内の1文字を飛び越します(コピーしません)。
CLS CTRL + L	指定スキップ	テンプレート内の文字を指定された文字の直前まで飛び越します(コピーしません)。
↑ ESC CTRL + ^ CTRL + [CTRL + U	取消	現在のコマンド行を取り消します。テンプレートの内容は変更されません。
INS CTRL + R	挿入	挿入モードに入ったり抜け出たりします。
← BS CTRL + H CTRL +]	後退	コマンド行から最後の1文字を除去します。
HOME CTRL + K	新しい行を作る	コマンド行をテンプレートにコピーします(新しいテンプレートを作ります)。
CTRL + J	行換え	ディスプレイの表示のみ改行します。

2. 編集機能の使用例

ここで実際に例を上げてコマンド行の編集機能を解説します。なお、テンプレートに関するキー操作はディスプレイには表示されません。ここではわかりやすくするために、特殊キーを《 》で囲んで示します。

以下のコマンド行は、ファイル「PROG.COM」のファイル情報を表示するものです。このコマンドを入力すると「dir」コマンドの実行とともに、コマンド行がテンプレートに記憶されます。

A>dir prog.com

もう一度このコマンド行を繰り返すには、↓とリターンキーを使います。↓を押すと、テンプレートがコマンド行にコピーされます。そして、リターンキーを押すとコマンドとして実行されます。

A>《↓》dir prog.com

次に、ファイル「PROG.MAC」のファイル情報を表示し、続けてその内容を表示しましょう。

現在、テンプレートには「dir prog.com」と記憶されています。まず、**SELECT** を押し、続いて **C** と押すと、「c」の直前までの文字がコマンド行にコピーされます。これに続けて「mac」と押すと、目的のコマンド行ができあがります。

A>《SELECT》《C》dir prog.
A>dir prog.mac

ここでリターンキーを押すと、このコマンド行が実行され、今度は「dir prog.mac」がテンプレートに記憶されます。では「type prog.mac」というコマンド行を作ってみます。それには次のようにキーを押します。

A>type 《INS》 《↓》

新しく入力された文字は直接コマンド行に入力されます。「dir」とそれに続くスペースの4文字が「type」に変換され、そしてコマンドとパラメータの間に必要なスペースを入れるために **INS** とスペースを入力します。そして、最後にテンプレートの残りの「prog.mac」をコマンド行にコピーします。すると、コマンド行は次のようになります。

A>type prog.mac

次に、コマンド行の編集機能を用いてテンプレートを修正する手順を説明します。ここでは、「type」とするべきところを「byte」と入力してしまったとします。これを修正するには **HOME** キーを使います。**HOME** には現在のコマンド行をテンプレートにコピーし、新しいテンプレートを作る機能があります。**HOME** を押すと新しいテンプレートを作った目印として、コマンド行の末尾に「@」が表示され、カーソルは次の行に移ります。

A>byte prog.mac 《HOME》 @

ここで、**→** と **↓** を使いテンプレートの内容を修正しながらテンプレートをコマンド行にコ

ピーします。[→] はテンプレートから1文字をコマンド行にコピーする機能を持っています。間違えた「b」を「t」に、さらに「t」を「p」に置き換え、残りを↓でコピーします。

```
A>byte prog.mac@
t <→> yp <↓> e prog.mac
```

また、同じことは以下のように [DEL] キーと [INS] キーを用いても可能です。

```
A>byte prog.mac@
<DEL> <DEL> <→> t <INS> yp <↓> e prog.mac
```

これを分解すると次のようになります。

タイプ	コマンド行	意味
		1文字目をスキップ
		2文字目をスキップ
<→>	t	3文字目をコピー
<INS> yp	typ	ypを挿入
<↓>	type prog.mac	残りをコピー

2.5.3 その他の機能

MSX-DOSにはコマンド行の編集に用いる特殊キーの他に、表3.4のような特殊キーがあります。

表 3.4 その他の特殊キー

機能	特殊キー	内容
中止	[CTRL] + [C]	現在動作中のコマンドやプログラムの実行を中止します。
印刷	[CTRL] + [P]	ディスプレイへの表示をプリンタへも出力します。
印刷取消	[CTRL] + [N]	「[CTRL] + [P]」の動作を取消します。
表示停止	[CTRL] + [S]	ディスプレイへの表示を一時停止します。任意のキーを押すと表示は再開します。

2.6 バッチ処理

2.6.1 バッチファイルの作成

バッチコマンドを利用するには、まずディスク上にバッチファイルを作成しなければなりません。バッチファイルは、「copy」コマンドを用いて次のような手順で作成します。バッチファイルはエディタでも作成できます。

- キーボードからの入力を「AUTOEXEC.BAT」というファイル名でドライブ A のディスクにコピーする

```
A>copy con autoexec.bat
rem This is AUTO EXECUTE FILE
basic funckey.ini
^Z
1 file copied
A>
```

この入力が入力がファイルに記録される
←ファイルの終りには、必ず **CTRL** + **Z** を入力する

「^Z」はファイルがそこで終ることを示す記号で、エンドオブファイルマーク (EOF) といい、ファイルの最後には必ず書き込まれなければなりません。

上の例の「AUTOEXEC.BAT」がディスク上に作成されたかどうか、「type」コマンドで表示させると以下ようになります。

```
A>type autoexec.bat
rem This is AUTO EXECUTE FILE
basic funckey.ini
A>
```

2.6.2 パラメータの使用例

バッチファイルでは、「%」のあとに1桁の数字を用いた仮パラメータを使用することができます。この仮パラメータは、バッチコマンド実行時にコマンド行に書かれたファイルスベックやスイッチ、パラメータと置き換えて処理されます。これを利用すれば、1つのバッチファイルで様々なファイルに対する処理を行うことができます。

仮パラメータは、%0 から%9 までの10個が使用できます。%0 はコマンド名自身と置き換えら

れ、%1 から%9 までの 9 個が実パラメータと置き換えられます。次の例では、仮パラメータを使用したバッチファイルで、アセンブラを用いてコマンドファイルを作っています。

```
A>type asm.bat
rem This is ASM.BAT
rem START BATCH FILE
dir
m80=%1.asm
m80=%2.asm
l80 %1, %2, %1/n/e
dir

A>
```

このバッチファイルを使って、test1.asm と test2.asm をアセンブルし、コマンドファイルを作成します。

```
A>asm test1 test2
A>rem This is ASM.BAT
A>rem START BATCH FILE
A>dir
MSXDOS      SYS      2432 85-08-23  9:29p
COMMAND    COM      6656 85-09-02 10:10p
M80        COM     20224 85-04-01 10:27p
L80        COM     10725 85-04-01 10:25p
ASM        BAT       117 85-07-17 12:24p
TEST1     ASM       256 85-07-18 12:20a
TEST2     ASM       256 85-07-18 12:23a
          7 files  317440 bytes free
A>m80 = test1.asm

No Fatal error(s)

A>m80 = test2.asm

No Fatal error(s)

A>l80 test1,test2,test1/n/e

MSX.L-80  1.00 01-Apr-85   (C) 1981, 1985 Microsoft

Data  0100  0141  < 65 >

43504 Bytes Free
[0000  0141  1]
```

```
A>dir
MSXDOS      SYS      2432 85-08-23  9:29p
COMMAND    COM      6656 85-09-02 10:10p
M80        COM     20224 85-04-01 10:27p
L80        COM     10725 85-04-01 10:25p
ASM        BAT       117 85-07-17 11:24p
TEST1     ASM       256 85-07-18 12:20a
TEST2     ASM       256 85-07-18 12:23a
TEST1     REL       128 85-07-18 12:24a
```

```
TEST2      REL      128 85-07-18 12:25a
TEST1      COM      128 85-07-18 12:25a
           10 files  314368 bytes free
A>
A>
```

このように、%1、%2 という2つの仮パラメータが、コマンド行から入力した順に test1、test2 と置き換って処理されます。

2.6.3 MSX-DOS 起動時の自動実行機能

「AUTOEXEC.BAT」というバッチファイル名は、MSX-DOS が使用します。「AUTOEXEC」とは自動実行(AUTO EXECution)という意味で、MSX-DOS 起動時に自動的に実行されるバッチファイルです。

MSX-DOS が起動すると、MSX はドライブ A に入っているディスクから「AUTOEXEC.BAT」というバッチファイルを捜し、見つければバッチコマンドとして実行します。「AUTOEXEC.BAT」がないときは通常どおりに起動します。

なお、時計機能がない MSX で、「AUTOEXEC.BAT」が実行されると、起動時に必要な日付を入力する手続は省略されますが MSX-DOS の動作に影響はありません。

2.7 MSX-DOS メッセージ一覧

MSX-DOS は、動作中にいろいろなメッセージを表示します。このメッセージには、使用者に対してなんらかの処置を求める入力待ちメッセージと、エラーが起こったことを伝えるエラーメッセージとがあります。

解説はメッセージのアルファベット順に、次のような構成で行います。なお、ディスプレイには対処法が表示されないため、そのメッセージに対する対処についても解説を加えます。

表示されるメッセージ

コマンド

そのメッセージが表示される可能性があるコマンドなど。

意味

メッセージの意味の説明。

内容

メッセージが表示された原因、内容などの説明。

対処方法

そのメッセージへの対処方法。

なお、コマンドの項に「COMMAND」とあるものは、そのメッセージが「COMMAND.COM」によって管理される外部コマンドやバッチコマンドにおいて表示される可能性があることを示します。

2.7.1 入力待ちメッセージ

Are you sure(Y / N)?

コマンド

DEL

意味

よろしいですか <Y/N>?

内容

「del」コマンドでファイル名の指定に「*.*」を用いると（全部のファイルを削除という意味）表示されます。

対処方法

全ファイルを削除してもよいのなら「Y」を押します。「N」を押すと、削除せずにコマンド入力待ちに戻ります。

Boot error Press any key for retry

コマンド

MSX-DOS 起動時

意味

MSX-DOS を起動できません。
任意のキーを押し、もう一度起動して下さい。

内容

ドライブ A に挿入されているディスクに「MSXDOS.SYS」が入っていないため、MSX-DOS を起動することができません。

対処方法

「MSXDOS.SYS」が記録されているディスクをドライブ A に挿入したのち、任意のキーを押してもう一度起動して下さい。

Disk error reading drive <ドライブ番号> Abort, Retry, Ignore?

Disk error writing drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

ディスクアクセス中 (COMMAND、COPY、DEL、DIR、REN、TYPE)

意味

ディスクアクセス中にエラーが発生しました。
中止<A>、再試行<R>、無視<I>?

内容

ディスクにキズがあるなどの物理的な事故があったとき、またはドライブに故障があるときにこのエラーが発生します。

対処方法

何度か「R」を入力し、それでもエラーが繰り返されるようであれば、そのディスクは使用不能です。「A」を入力してコマンドを中止してから、新しいディスクでその処理を行って下さい。それでもエラーが発生したらドライブの故障と考えられます。

Insert disk with batch file and strike any key when ready

コマンド

バッチコマンド

意味

バッチファイルのあるディスクをドライブに入れて下さい。
準備ができたら任意のキーを押して下さい。

内容

バッチコマンド実行中に、そのバッチファイルがあるディスクを引き抜いてしまったときに表示されます。

対処方法

実行中のバッチファイルがあるディスクをもとのドライブに挿入してから任意のキーを押します。

Insert diskette for drive <ドライブ番号> and strike a key when ready

コマンド

COMMAND、COPY、DEL、DIR、FORMAT、REN、TYPE

意味

ディスクをドライブ<ドライブ番号>に入れて下さい。
準備ができたら任意のキーを押して下さい。

内 容

仮想ドライブ機能が動作しているとき、複数のドライブに対して処理を行うコマンドを実行すると表示されます。

対処方法

「2.1.4 3.仮想ドライブ機能」を参照して下さい。

Insert DOS disk in default drive and strike any key when ready

コマンド

すべてのコマンド

意 味

COMMAND.COMのあるディスクをデフォルトドライブに入れて下さい。
準備ができたなら任意のキーを押して下さい。

内 容

外部コマンド終了時に、デフォルトドライブに挿入されているディスクに「COMMAND.COM」が存在しないと、このメッセージが表示されることがあります。
また、MSX-DOS 起動時に、ドライブ A に挿入されているディスクに「MSXDOS.SYS」だけがあり、「COMMAND.COM」がないときにもこのメッセージが表示されます。

対処方法

「COMMAND.COM」が入ったディスクをデフォルトドライブに挿入してから任意のキーを押します。

Invalid date Enter new date

コマンド

DATE

意味

日付の指定が違います。
新しい日付を入力して下さい。

内容

日付の指定に許されていない値を入力すると表示されます。

対処方法

正しい値をもう一度入力します。

Invalid time Enter new time

コマンド

TIME

意味

時刻の指定が違います。
新しい時刻を入力して下さい。

内容

時刻の指定に許されていない値や区切り記号を入力すると表示されます。

対処方法

正しい値をもう一度入力します。

Not ready error reading drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、DIR、REN、TYPE

意味

ドライブ<ドライブ番号>は読み込む準備ができていません。
中止<A>、再試行<R>、無視<I>?

内容

ディスクが正しくドライブに挿入されていないときや、ドライブに故障があったときに表示されます。

対処方法

ディスクが正しくドライブに挿入されているかどうかを確認し、「R」を押します。

Not ready error writing drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、REN

意味

ドライブ<ドライブ番号>に書き込む準備ができていません。
中止<A>、再試行<R>、無視<I>?

内容

ディスクが正しくドライブに挿入されていないときや、ドライブに故障があったときに表示されます。

対処方法

ディスクを引き抜いてしまったときは正しいディスクを挿入し「R」を入力します。
ドライブの故障のときは修理して下さい。

Strike a key when ready

コマンド

DIR/P、FORMAT、PAUSE

意味

準備ができたなら任意のキーを押して下さい。

内容

一時停止した画面の表示またはバッチコマンドの実行を再開するかどうか指示を得なければならないときに表示されます。

「format」コマンドでは、ディスクの準備のために表示されます。

対処方法

必要な処置をとってから任意のキーを押します。

Terminate batch file (Y/N)?

コマンド

バッチコマンド

意味

バッチ処理を中止しますか (Y/N) ?

内容

バッチ処理中に **CTRL** + **C** を押すと表示されます。

対処方法

バッチコマンドを中止するときは「Y」を、続行するときは「N」を入力します。「N」を入力押したとき、バッチコマンドは次のコマンド行から続行されます。

Write protect error writing drive <ドライブ番号> Abort, Retry, Ignore?

コマンド

COMMAND、COPY、DEL、REN

意味

ドライブ<ドライブ番号>に対して書き込みができません。
中止<A>、再試行<R>、無視<I>?

内容

書き込みの対象となっているディスクが書き込み禁止になっている（ライトプロテクトノッチが書き込み不可側にセットされている）ときに発生します。

対処方法

ライトプロテクトノッチを書き込み可にセットしなおしてから「R」を入力します。
ファイルのコピー中などに、ディスクを入れ間違えたときは、「A」を入力して処理を中止し、もう一度最初から処理を行います。

注意

「R」を入力するときには、ディスクを入れ替えないで下さい。ディスクを入れ替えると、そのディスクが壊れ、データも正しく書き込まれません。

以下のメッセージも入力待ちメッセージですが、ここでは解説を省略します。「2.4 コマンド一覧」のそれぞれのコマンドの項を参照して下さい。

Current date is www yy-mm-dd
Enter new date:

コマンド

DATE

Current time is hh:mm:ss.tt {a | p}
Enter new time:

コマンド

TIME

2.7.2 エラーメッセージ

Bad command or file name

コマンド

COMMAND

意味

コマンドまたはファイル名が違います。

内容

入力した外部コマンド名またはバッチファイル名が指定したドライブ上にないときや、コマンド名、ファイル名の入力を間違えたときに発生します。

対処方法

- ドライブ指定で、そのファイルが記録されているディスクが挿入されているドライブを指定します。
- そのファイルが記録されているディスクをドライブに挿入し、そのドライブ名を指定します。
- 正しいファイル名を入力します。

Bad FAT, drive <ドライブ番号>

コマンド

COMMAND、COPY、DEL、DIR、REN、TYPE

意味

FAT の内容が破壊されています。

内容

FAT の内容に異常があったときに発生します。

対処方法

致命的なエラーで、そのディスクに記録されているファイルの内容は保証できません。

このエラーが出たときは、読み出せるファイルがあれば（「type」コマンド、「copy」コマンドを使って確認する）、別のディスクにコピーして救済して下さい。エラーが起こったディスクはフォーマットしなおす必要があります。

Bad parameter

コマンド

FORMAT

意味

パラメータが違います。

内容

「format」コマンドがフォーマットの様式を選択するようになっていたときに、間違った値を入力するとこのエラーが発生します。

対処方法

「format」コマンドをもう一度起動し、正しい値を入力して下さい。

Disk error

コマンド

FORMAT

意味

ディスクにエラーが発生しました。

内容

「format」コマンド実行中にディスク入出力関係のなんらかのエラーが発生しました。

対処方法

他のコマンドを実行して具体的にどのようなエラーなのかを調べ、それに対処します。

File cannot be copied onto itself

コマンド

COPY

意味

ファイルはそれ自身にはコピーできません。

内容

ファイルをそれ自身が記録されているディスクにコピーしようとする、このエラーが発生します。例えば、次のようなコマンドを実行したときなどです。

```
A>copy file.ext  
A>copy file.ext a:  
A>copy *.* a:
```

対処方法

- 同じディスクにコピーしたいときは違うファイル名でコピーします。
- 違うディスクにコピーします。

File creation error

コマンド

COPY

意味

ファイルを作れません。

内容

すでに記録可能な総ファイル数まで使用されているディスク上に、新しくファイルを作成しようとする、このエラーが発生します。

対処方法

新しいディスクを使用して下さい。

File not found

コマンド

DIR、TYPE、DEL、COPY

意味

ファイルが見つかりません。

内容

<ファイルスペック>で指定したファイルが、指定したドライブに存在しないときや、ファイル名の入力を間違えたときに発生します。

対処方法

- そのファイルが記録されているドライブを指定します。
- そのファイルが記録されているディスクをドライブに入れます。
- 正しいファイル名を入力します。

Insufficient disk space

コマンド

COPY

意味

ディスクの容量が足りません。

内容

ファイルを作ったりコピーするために十分な容量がディスクに残っていないときに発生します。

対処方法

新しいディスクまたは十分に空きのあるディスクを用いてその処理を行います。

Invalid drive specification

コマンド

COMMAND、DIR、TYPE、REN、DEL、COPY

意味

ドライブの指定が違います。

内容

ドライブ指定で、使用することができないドライブを指定したときに発生します。例えば、ドライブ B までしか使用できないのにドライブ C を指定したときなどです。

対処方法

正しいドライブ指定を行います。

Invalid parameter

コマンド

MODE、VERIFY

意味

パラメータが違います。

内容

「mode」コマンドで許されている値の範囲を越えて指定したとき、「verify」コマンドでは「on」、「off」以外のものを指定したときに発生します。

対処方法

正しいパラメータを入力します。

No enough memory

コマンド

FORMAT、MSX-DOS 起動時

意味

メモリが足りません。

内容

MSX-DOS が起動するために必要なメイン RAM (64KB) がないときや、「format」コマンドを実行するために十分なメモリがないときに発生します。

対処方法

ドライブの台数が多すぎるのが考えられます。そのときは、**CTRL** キーを押しながら立ちあげて、2ドライブシミュレータ機能を切り離すなどの方法でドライブ数を少なくします。

Not ready

コマンド

FORMAT

意味

ディスクが入っていません。

内容

フォーマットの対象となっているドライブにディスクが入っていないときに表示されます。

対処方法

ドライブにディスクを挿入します。

Program too big to fit in memory

コマンド

COMMAND

意味

プログラムが大き過ぎて、メモリに入りません。

内 容

使用できるメモリより大きいプログラムを実行しようとしたときに発生します。

対処方法

- 接続しているディスクの数を減らして、使用できるメモリを増やします。
- 大きさが小さくなるように、プログラムを変更します。

Rename error

コマンド

REN

意 味

このファイル名変更は間違っています。

内 容

ファイル名を変更する際に、新しいファイル名としてすでに存在するファイル名を指定したときに発生します。

対処方法

dir コマンドでどのようなファイルが記録されているかを確認し、そのディスクにはないファイル名を指定します。

Unsupported media type error reading drive <ドライブ番号>

コマンド

ディスクアクセス中

意 味

<ドライブ番号>のディスクはこのドライブでは扱えません。

内 容

そのドライブで扱うことができない種類のディスクに対して入出力しようとしたときに発生します。このとき自動的に Abort します。

対処方法

そのドライブで扱えるタイプのディスク以外は使用しないで下さい。ただし、そのディスクがフォーマットされていない可能性もあります。

Write error

コマンド

COPY

意味

書き込みができません。

内容

何らかの理由で出力先のファイルに書き込めなくなったときに発生します。

対処方法

もう一度最初からやり直します。

Write protected

コマンド

FORMAT

意味

ディスクが書き込み保護されています。

内容

フォーマットの対象となっているディスクが書き込み禁止になっている（ライトプロテクトノッチが書き込み不可側にセットされている）ときに発生します。

対処方法

ライトプロテクトノッチを書き込み可にセットしなおしてから、もう一度「format」コマンドを実行します。

3章

MSX-DOS の構造

3.1 MSX-DOS の起動

MSX-DOS は以下の手順で起動します。

1. MSX をリセットすると、始めにすべてのスロットを調べ、調べたスロットの先頭に「41H、42H」の 2 バイトが書き込まれていれば、そのスロットには何らかの ROM プログラムが接続されていると判断し、ROM のヘッダ部分にアドレスを設定された INIT (初期化) ルーチンを実行します。ディスクインターフェイス ROM の INIT ルーチンの場合は、まずそのインターフェイスに接続されているドライブのために、ワークエリアを確保します。「第 2 部 1 章 ブートシーケンス」を参照して下さい。
2. すべてのスロットを調べ終わったら、【H.STKE(FEDAH)】を参照します。このアドレスの内容が C9H でなければ、Disk BASIC の環境を設定し、H.STKE にジャンプします。
3. もし、上記の調査で、H.STKE の内容が C9H であれば、TEXT エントリを持つカートリッジを各スロットで探し、あれば Disk BASIC の環境を設定した後、そのカートリッジの BASIC プログラムを実行します。
4. ブートセクタ (論理セクタ 0) の内容を C000H~C0FFH へ転送します。このとき、「DRIVE NOT READY」か「READ ERROR」が発生したり、転送されたセクタの先頭が「FBH」か「E9H」ではなかったときは、Disk BASIC が起動します。
5. C01EH に転送されたルーチンが CY フラグをリセットした状態でコールされます。通常は、このアドレスには「RET NC」のコードが書き込まれているため、何も実行しないでリターンします。

このとき、スロットの状態は以下のようになっています。しかし、BIOS コールやファンクションコールによって、ページ 1 を RAM に切り換えることはできないので、ユーザープログラムの起動には適当ではありません。

ページ	内 容
0	BIOS ROM
1	DISK ROM
2	RAM
3	RAM

- RAM の容量を調べます。このとき、RAM の内容は壊れません。もし、RAM が 64KB 未満だったら、Disk BASIC が起動します。
- MSX-DOS の環境を初期化した後、CY フラグをセットした状態で C01EH がコールされます。通常は、このアドレスには「RET NC」の後ろに標準のブートプログラムが書き込まれています。この部分に任意の機械語プログラムのブートプログラムを書き込んでおけば、そのプログラムが自動的に起動します。
このとき、スロットの状態は以下のようになっています。

ページ	内 容
0	RAM
1	DISK ROM
2	RAM
3	RAM

また、レジスタには、以下の情報が入っています。

レジスタ	内 容
A	0 ならば POWER ON 直後を示す
DE	この内容をコールすると、ページ 1 の FDD ROM が RAM に切り換わる
HL	ディスクエラー処理ルーチンへのポインタへのポインタ（「第2部 7.9.3 エラー処理」参照）

標準のブートプログラムは、MSXDOS.SYS を 100H からのアドレスにロードし、100H にジャンプします。MSXDOS.SYS がないときは、Disk BASIC を起動します。

- MSXDOS.SYS は自分自身を高位アドレスに転送した後、COMMAND.COM を 100H からのアドレスにロードし、その先頭へジャンプします。COMMAND.COM がないときは、「INSERT A DISKETTE」のメッセージが表示されて、正しいディスクがドライブにセットされるのを待ちます。

9. COMMAND.COM も自分自身を MSXDOS.SYS のすぐ下の高位アドレスに転送し、COMMAND.COM 自身が実行を始めます。
10. MSX-DOS が最初に起動したとき、「AUTOEXEC.BAT」という名前のファイルがあれば、それをバッチファイルとして実行します。MSX-DOS が起動せず、Disk BASIC が立ち上がったときは、「AUTOEXEC.BAS」という名前の BASIC プログラムがあれば、それを実行します。

3.2 プログラムの起動から終了まで

MSX-DOS は外部コマンドの形でコマンドを追加・変更できる拡張性の高いオペレーティングシステムです。ユーザーの作ったプログラムも外部コマンドとして簡単に実行することができます。

外部コマンドは、アセンブラやコンパイラなどを利用して作られた、「COM」という拡張子のついたファイルです。コマンドプロセッサは指定されたコマンド名+「COM」という名前のファイルをディスク上に見つけると、それをメモリ上にロードして実行します。このファイルの内容はメモリの 100H 番地からロードして、そのまま実行できるような形式になった機械語のプログラムです。

3.2.1 プログラムの起動

MSX-DOS の基本的な動作は、コマンド行を入力し実行させる、という処理の繰り返しです。この時、コマンド行の入力からコマンドの解釈実行までのユーザーインターフェイスを受け持つのがコマンドプロセッサである「COMMAND.COM」というプログラムです。入力したコマンドが外部コマンドのとき、コマンドプロセッサは、それを以下のようにして実行に移します。

1. コマンド行のパラメータについて、その長さをシステムスクラッチエリアの 80H 番地に、実際の文字列を 81H 番地以降に格納する。さらに最初の 2 つのパラメータをファイル名とみなし、それを FCB の形式でそれぞれシステムスクラッチエリアの 5CH 番地および 6CH 番地以降に格納する。
2. 外部コマンドを 100H 番地以降に読み込む。
3. 100H 番地にジャンプする。

MSX-DOS では、システムとプログラムとの間でデータなどを受け渡すために、メモリの 0

～FFH 番地を使っています。これをシステムクラッチエリアといいます。

コマンドプロセッサはまず、コマンド行のパラメータを渡すために、システムクラッチエリアの設定を行います。外部コマンドは、システムクラッチエリアを参照することにより、渡されたパラメータを知ることができます。このために2つの方法が使われます。

1. 5CH 番地または 6CH 番地はそのままファイルのアクセスに使える形式になっているので、ファイル名をパラメータとするようなプログラムではこの方法が便利です。ただし、この方法では両 FCB(5CH、6CH)の先頭アドレスが 16 バイトしか離れていないので、完全な FCB として使用できるのはどちらか片方だけになります。FCB については、「3.3.2 ファイルの入出力」の項で詳しく説明します。
2. 80H 番地には全パラメータがそのまま入っているので、ファイル名以外をパラメータとしたり、3 つ以上のファイル名を扱うようなプログラムではこの方法を使用します。

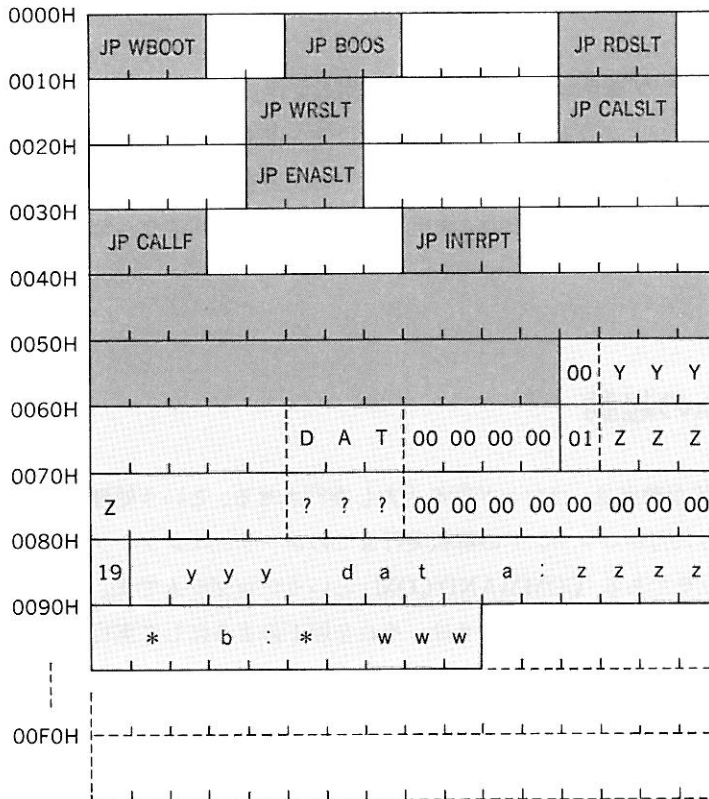


図 3.8 システムクラッチエリア

図 3.8 はシステムスクラッチエリアの内容です。これは、

```
A>test yyy.dat a:zzzz.* b:*.www
```

というコマンドを入力して、外部コマンド「test」を実行したときの例です。80H 番地以降にはこのコマンドに渡すパラメータとその長さが入っているのがわかります。また、前述したように、5CH 番地と 6CH 番地に最初の 2 つのパラメータが形式を整えられて入っています。先頭 1 バイトには、ドライブが無指定のときは「0」、A:であれば「1」、B:なら「2」・・・という値が入ります。また、小文字は大文字に変換され、「*」は複数の「?」に展開されます。システムスクラッチエリアの 5CH～FFH 番地は、このようにパラメータを渡すために使いますが、プログラムが実行を開始した後は、ワークエリアとして使うこともできます。

図 3.9 はプログラムが実行を開始したときのメモリマップです。00H～FFH 番地はシステムスクラッチエリアで、プログラムはその直後の 100H 番地からロードされます。100H 番地から始まり、06H～07H 番地の内容で示されているアドレスの 1 バイト手前までの領域は、TPA (Transient Program Area =一時プログラム領域) と呼ばれ、プログラムで自由に利用してよい領域です。

システムスクラッチエリアの先頭の方には、いくつかのジャンプ命令が置かれています (図 3.8 参照)。プログラムはこれらの決められたアドレスをコールすることで、MSX システムの持つ各種の機能が実行できます。00H 番地にあるのがウォームスタートのためのエントリ、05H 番地にあるのが 4 章で解説するファンクションコールのためのエントリです。この 2 つは MSX-DOS が持っている機能をプログラムから利用するために用意されています。また RDSLTL、WRSLTL、CALSLTL、ENASLTL、CALLF および INTRPT は、MSX BIOS にある同名のファンクションと同じ機能を持っています。INTRPT は割り込み時に使われますが、その他のエントリは MSX BIOS と同様に利用できます。MSX-DOS 自身もこれらのエントリを使用しているので、0H～5BH 番地 (□ の部分) を破壊するとシステムダウンにつながります。

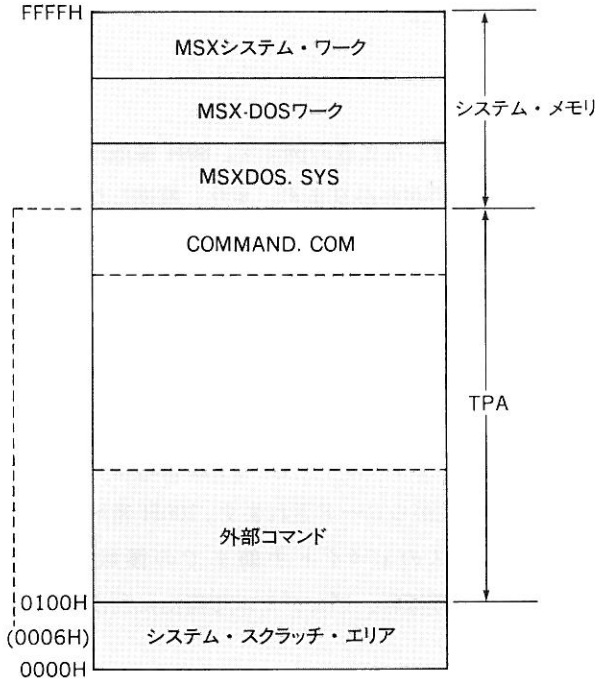


図 3.9 外部コマンド実行時のメモリマップ

05H 番地はファンクションコールのためのジャンプ命令です。06H～07H 番地はジャンプ先のアドレスで、これが TPA の上限を示しているということは、ジャンプ先が MSXDOS.SYS の先頭になっているということです。つまり、05H～07H 番地はジャンプ命令であると同時に TPA の上限を示すという 2 重の働きを持っています。

3.2.2 プログラムの終了

プログラムは実行が終わると、次の 3 つの方法のいずれかで MSX-DOS のコマンドレベルに戻ることができます。

1. スタックポインタを変更していない場合、「RET」を実行する。
2. 後述のファンクションコールを使って、「システムリセット」を行う。
3. 00H 番地（ウォームスタートのためのエントリ）にジャンプする。

コマンドプロセッサはスタックの先頭に 0 を置いて、外部プログラムを呼び出すので、1. は 2. と同じ結果になります。ただし、ソフトウェアデバッガ（MSX-SBUG など）を使って、デバッグ作業を行うときは、スタックがこのように設定されていないことがあるので、注意が必要です。

コマンドプロセッサの実体は「COMMAND.COM」というプログラムで、TPAの内側にあります。コマンドプロセッサが必要なのは、プログラムの実行が終わって、次のコマンドを入力するときで、プログラムの実行中は、コマンドプロセッサの機能は必要ないので、プログラムでコマンドプロセッサがロードされている領域を使用（破壊）してもかまわないわけです。このため、ウォームスタートの際には、コマンドプロセッサがプログラムによって壊されている可能性があるため、いったんMSXDOS.SYSが制御を受け取ります。MSXDOS.SYSは、チェックサムを用いてCOMMAND.COMが壊されているかどうかを調べてから、コマンドプロセッサに制御を渡します。COMMAND.COMが破壊されたときは、再びディスクからロードしますが、破壊されていないときは、ディスクからのロードは行われないので、コマンドレベルに戻るのが早くなります。

こうして、再びコマンドプロセッサが次のコマンド行の入力待ちになります。

3.3 MSX-DOS の入出力

3.3.1 周辺装置との入出力

MSX-DOSの持つ機能は、ディスク上のファイルを扱うものだけではありません。それだけでは、オペレーティングシステムを持つ重要な機能の1つであるユーザーインターフェイスの機能が限られてしまいます。そのため、MSX-DOSでは、コンソールスクリーン、キーボード)やプリンタなどの周辺装置の制御を行う機能を豊富に用意しています。これらの装置は、基本的には1文字単位で入出力を行うものですが、コンソールに関しては、各種のオプションを持った複数の入出力機能に加えて、行入力や行出力の機能も用意しています。

3.3.2 ファイルの入出力

ファイルの入出力（アクセス）の特徴は、データの位置をディスク上のアドレスのような具体的な数値で表現するのではなく、「名前」を用いて指定できるという点にあります。MSX-DOSはディスク上に書かれたディレクトリなどの情報を元に、ファイルのデータがどこにあるかということを常に把握しています。目的のデータがディスク上のどんなアドレスに存在しているか、ということはすべてMSX-DOSにまかせ、プログラムはただファイル名を指示するだけで、ファイルのアクセスができます。

1. FCB (ファイルコントロールブロック)

どのファイルにアクセスするか、ということを示すために、MSX-DOSではFCB(ファイルコントロールブロック)を使います。

FCBは、ファイルを開く際に必要となる情報を格納しておく領域で、1つのファイルを開くごとに1つ、図3.10や図3.11のようなメモリエリアをプログラムで用意します。

FCBはプログラムで許された範囲でメモリ上のどこに置いても構いませんが、MSX-DOSの機能を活かすため、しばしば5CH番地が使われます。

FCB 先頭からのバイト数 ↓	0	ドライブ番号 ファイルの存在するディスクドライブを示します(0 = デフォルトドライブ、1 = A、2 = B、 ~7 = H)。
	1 : 8	ファイル名 最大 8 文字まで指定できます。 8 文字に満たない部分はスペース (20H) で埋めます。
	9 : 11	拡張子 最大 3 文字まで指定できます。 3 文字に満たない部分はスペース (20H) で埋めます。
	12	カレントブロック シーケンシャルアクセスの際、 参照中のブロック番号を示します(ファンクション 14H、15H 参照)。
	13*	
	14	レコードサイズ ブロック中のレコード数が設定されます (ファンクション 14H、15H 参照)。
	15*	
	16 : 19	ファイルサイズ ファイルの大きさがバイト単位で設定されます。
	20 21	日付 最後にファイルに書き込みを行った日付が設定されます。
	22 23	時刻 最後にファイルに書き込みを行った時刻が設定されます。
	24	デバイス ID 周辺装置・ディスクファイルの区別が設定されます。
	25	ディレクトリロケーション 何番めのディレクトリエントリに該当するファイルであるかが設定されます。
	26 27	先頭クラスタ ファイルの先頭のクラスタ番号が設定されます。
	28 29	最終クラスタ 最後にアクセスされたクラスタ番号が設定されます。
	30 31	相対位置 最後にアクセスされたクラスタの、先頭クラスタからのファイル内での相対位置が番 号で設定されます。
	32	カレントレコード シーケンシャルアクセスの際、参照中のレコード番号を示します(ファンクション 14H、 15H 参照)。
	32*	
	33 : 35	ランダムレコード ランダムアクセスの際、アクセスしたいレコードを指定します。(ファンクション 21H、 22H、28H 参照)。

(*システムで使用)

図 3.10 FCB として使うメモリエリア (CP/M 互換)

FCB 先頭からのバイト数 ↓	0	ドライブ番号 ファイルの存在するディスクドライブを示します(0 = デフォルトドライブ、1 = A、2 = B、 ~7 = H)。
	1 : 8	ファイル名 最大 8 文字まで指定できます。 8 文字に満たない部分はスペース (20H) で埋めます。
	9 : 11	拡張子 最大 3 文字まで指定できます。 3 文字に満たない部分はスペース (20H) で埋めます。
	12 : 13*	
	14 15	レコードサイズ レコードサイズがバイト数で設定されます (ファンクション 26H、27H 参照)。
	16 : 19	ファイルサイズ ファイルの大きさをバイト単位で設定します。
	20 21	日付 最後にファイルに書き込みを行った日付が設定されます。
	22 23	時刻 最後にファイルに書き込みを行った時刻が設定されます。
	24	デバイス ID 周辺装置・ディスクファイルの区別が設定されます。
	25	ディレクトリロケーション 何番めのディレクトリエントリに該当するファイルであるかが設定されます。
	26 27	先頭クラスタ ファイルの先頭のクラスタ番号が設定されます。
	28 29	最終クラスタ 最後にアクセスされたクラスタ番号が設定されます。
	30 : 31	相対位置 最後にアクセスされたクラスタの、先頭クラスタからのファイル内での相対位置が番 号で設定されます。
	32*	
	33 : 36	ランダムレコード ランダムブロックアクセスの際、アクセスしたいレコードを指定します。レコードサイズ が 1~63 のときは+33~36 の 4 バイトを使い、レコードサイズが 64 以上のとき は+33~35 の 3 バイトを使用します (ファンクション 26H、27H 参照)。

(*システムで使用)

図 3.11 FCB として使うメモリエリア (MSX-DOS 専用)

2. ファイルのオープン

FCB を用いてファイルの入出力を行うには、最初にファイルを「オープン」する手続きが必要です。「オープン」とは、ファイル名フィールドだけが設定された不完全な FCB を、ディスク上に記された情報を用いて、完全な FCB に変換することを意味します。図 3.12 は「オープンされて
いない FCB」と「オープンされた FCB」の違いを表しています。

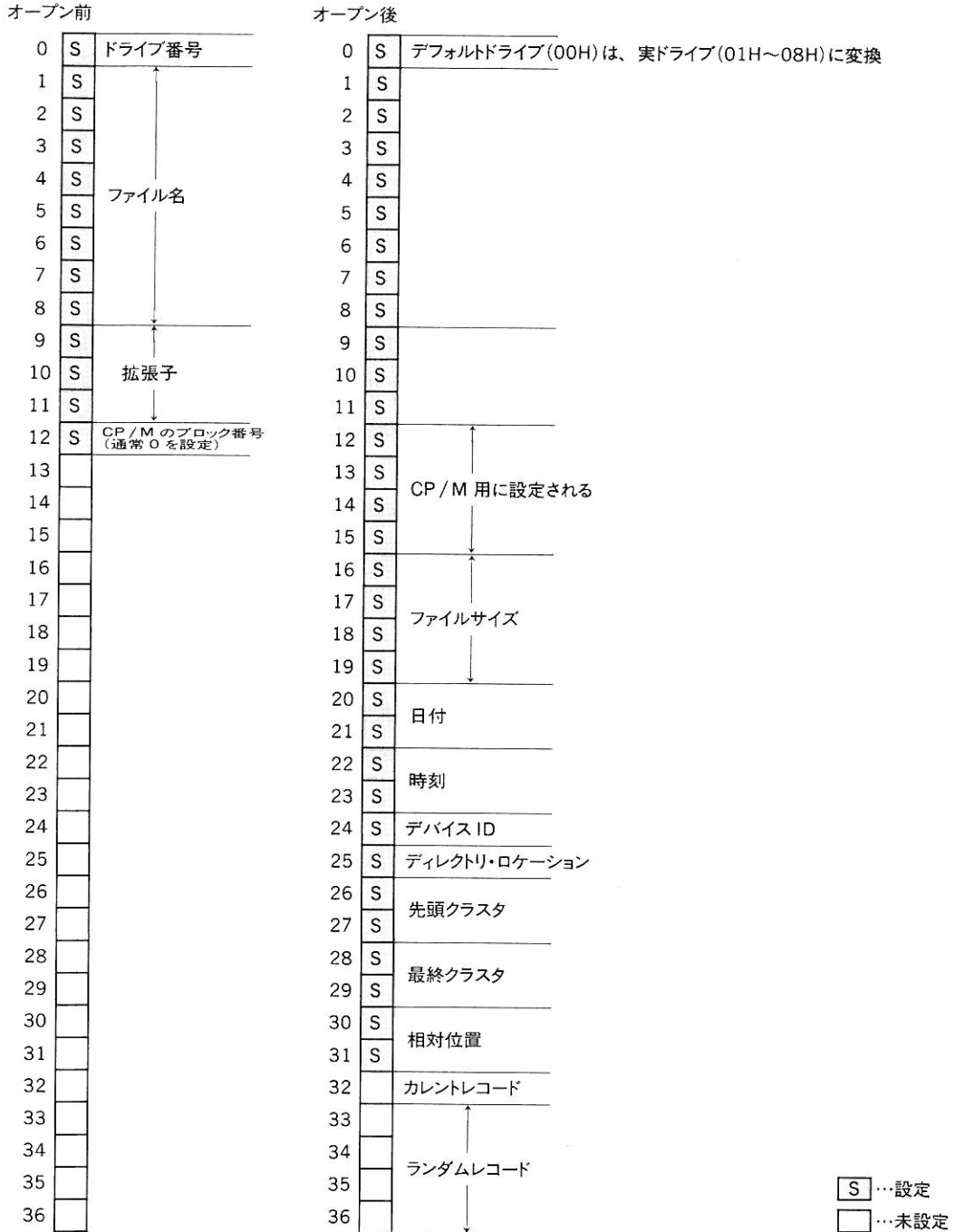


図 3.12 FCB オープンの前後

3. ファイルのクローズ

ファイルをオープンして書き込みを行ったとき、それに伴って、ファイルサイズを始めとするFCBの各フィールドの内容が変更されます。この更新されたFCBの情報をディレクトリ領域に戻しておかないと、次回ファイルをアクセスする際に、ディレクトリの情報と実際のファイルの内容が異なってしまいます。更新されたFCBの情報をディレクトリに戻すというこの操作が「ファイルのクローズ」です。

4. DTA (ディスク転送アドレス)

ディスクの入出力では、基本的には文字単位の入出力である周辺装置と違って、一度に複数のバイトを読み書きします。そのため、ディスクの入出力には「バッファ」が必要になります。MSX-DOSでは、このバッファとして使用されるメモリ領域の先頭アドレスをDTA (Disk Transfer Address = ディスク転送先アドレス) と呼びます。

ディスクの入出力では、DTAを先頭とするメモリ領域をバッファとして使用します。DTAの値は、初期設定では0080Hですが、任意のアドレスに設定し直すことができます。

5. レコード

ファイルの大きさは、ディスク容量の許す限りいくらでも(最大1Gバイト)大きくすることができますので、メモリに全体が読み込めることは期待できません。そのため、ファイルの入出力では、ファイルをレコードと呼ばれる適当な大きさのデータ単位に分割して、レコード単位で読み書きする方法がとられています。

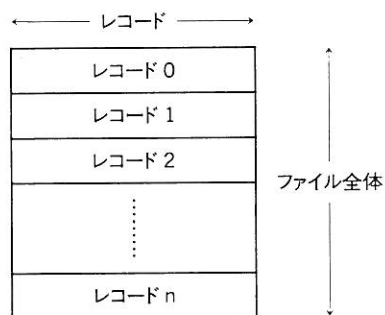


図 3.13 ファイルとレコード

したがって、ファイルのアクセスを行うためには、さらにどのレコードをアクセスするかという指示が必要になります。このレコードの指示方法はファンクションコールの種類により、大きく2つのグループに分けられます。1つのグループはCP/Mとの互換性を保つために用意された入出力で、もう1つのグループはMSX-DOSの独自のランダムブロック入出力です。この2つ

は、FCBの使用方法が異なっているので、同じFCBに対して同時に使用することはできません。

6. CP/M 互換のレコードアクセス

CP/Mとの互換性を保つため、MSX-DOSではCP/Mと同様のファイルアクセス方式をサポートしています。CP/Mでは、2つの全く異なるアクセス方式があります。また、レコードサイズは128バイトに固定されています。

■シーケンシャルアクセス（カレントレコード+カレントブロックによるレコード管理） CP/Mとの互換性を保つための1つめの方法が、「カレントレコード」と「カレントブロック」で管理されるシーケンシャルアクセスです。ファイルのアクセスは常に先頭から順（シーケンシャル）に行い、アクセスしたレコード数はFCBのカレントレコードフィールドでカウントされます。カレントレコードフィールドの値は128になると0にリセットされ、その桁上りがカレントブロックフィールドにカウントされます。このアクセス方式では、「ランダムレコード」は使いません。

■ランダムアクセス（ランダムレコードによるファイル管理）

CP/Mとの互換性を保つための2つめの方法が、「ランダムレコード（3バイト）」で管理されるランダムアクセスです。「ランダムレコード」を変更することで、任意の位置のレコードにアクセスすることができますが、別のレコードにアクセスするためには、毎回この操作が必要です。

7. ランダムブロックアクセス（論理レコードによるファイル管理）

MSX-DOSには、「ランダムブロックアクセス」という、大変に有能な入出力方式があります。これは、MSX-DOS独自のファンクションコールで、次のような特徴があります。

- レコードサイズを任意に設定できる。
- 複数のレコードを一度にアクセスできる。
- シーケンシャルアクセスとランダムアクセスを任意に切り換えられる。

ランダムブロックアクセスでは、FCBの「レコードサイズ」を用いて、レコードの大きさを指示し、レコード位置を「ランダムレコード」で指示します。レコードサイズの2バイトには任意の値を入れることができます（1～65535）。ファイル全体を1レコードとして扱うことも、1バイトを1レコードとして扱うことも、128バイトを1レコードとして扱うこと（CP/M方式）も可能です。

このアクセス方式では、ランダムレコードのフィールドがアクセスの終わったレコードの次のレコードを指すように、常に更新されます。したがって、ランダムレコードを1度セットするだけで、任意のレコードからレコードまでをシーケンシャルに読み書きすることが簡単にできます。

3.3.3 日付と時刻の設定

MSX-DOS は、ファイルの内容が変更されたときには、自動的にファイルの最終更新日付・時刻を現在の日付・時刻に書き換えます。これは、MSX-DOS 自身には何の意味もないのですが、ユーザーがファイルを管理する目的には、非常に有用です。このため、MSX-DOS には、日付・時刻の読み出しや変更を行うファンクションコールがあります。

3.4 ディスクファイルの構造

ファンクションコールを使用すれば、手軽にキメ細かくファイルを取り扱うことができます。ユーザーは、ディスク上でファイルがどのような方法で管理され、どのような形式で記録されているか、といった情報を詳しく知る必要はありません。しかし、場合によっては、ディスクの管理情報を取り出したり、それをもとにファイルとは無関係にディスクを直接アクセスしたりする手段が必要になることがあります。

MSX-DOS では、このような目的のために、ディスクの管理情報を得たり、「論理セクタ」を直接アクセスするファンクションコールを用意しています。

3.4.1 ディスク上のデータ構造

セクタを直接アクセスする場合には、どのセクタにどのような情報が書き込まれているか、という基本知識が必要になります。

1. 論理セクタ

MSX-DOS では、3.5 インチフロッピーディスクでもハードディスクでも、あるいはその他のドライブでも基本的にはアクセスすることができます。それぞれのドライブやメディアの種類によりセクタの大きさ、トラック毎のセクタ数、記録面の数などは違っていますが、これを統一的に管理するため、MSX-DOS では、ディスク上の物理的な境界にとらわれず、すべてのセクタに連続した通し番号をつけて、その番号でセクタを管理する方法を採用しています。これを「論理セクタ」と呼びます。

「論理セクタ」（以下、セクタ）の番号は0からそのディスクの総セクタ数-1（ディスクの種類によって異なる）までの一連の番号によって指定します。

MSX-DOS では、ディスクの中のセクタを表 3.5 に示す 4 つの領域に分けています。最初の 3 つの領域にはデータを管理するための情報が書き込まれ、ファイルデータの本体は「データ領域」の部分に書き込まれます。これらの位置関係は図 3.14 のとおりです。ブートセクタはかならずセクタ 0 にありますが、

- FAT
- ディレクトリ
- データ領域の開始セクタの位置

はメディアによって異なります。ただし、これらの情報は、ブートセクタ読み出せば得ることができます。

表 3.5 ディスクの領域

領域	内容
ブートセクタ	ディスク固有の情報と MSX-DOS の起動プログラム
FAT	ディスク上のファイルの位置情報
ディレクトリ	ディスク上のファイルの管理情報
データ領域	実際のファイルデータ

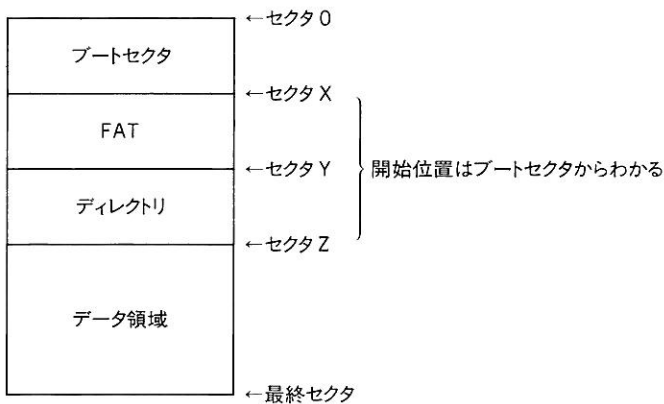


図 3.14 ディスク上の領域の位置関係

2. クラスタ

ディスクの入出力は、前述のとおりセクタが基本単位です。ただし、ファイルに対してディスク上のセクタを割り当てるときには、セクタではなく複数のセクタから成る「クラスタ」という単位が使われます。それぞれのファイルには、そのファイルサイズに応じて必要な数のクラスタ

が割り当てられます。1 クラスタよりも大きなサイズのファイルの場合、データは複数のクラスタにまたがって記録されます。1 クラスタ未満の部分については、たとえそれが1 バイトであっても1 クラスタ分のデータ領域が割り当てられます。クラスタは論理セクタと同様に連続した番号で指定されていますが、FAT の項で述べる理由で、2 から始まる通し番号になっており、データ領域の先頭がクラスタ#2 の位置に相当します。

3. ブートセクタと DPB (ドライブパラメータブロック)

MSX-DOS では、接続されている個々のドライブごとに「DPB」という領域がメモリ上のワークエリアに設けられ、各ドライブに固有の情報が記録されます。MSX-DOS はどのようなタイプのディスクドライブにも対応できますが、それはこの DPB を参照して個々のドライブに対応した処理を行うことによって、メディア間の差異が吸収できるからです。

DPB に書き込まれる情報は、ブートセクタに記録されているもので、それが MSX-DOS の起動時やメディアが交換される毎に変更されます。ただし、ブートセクタと DPB は、図 3.15 と図 3.16 に示すように、形式が異なります。

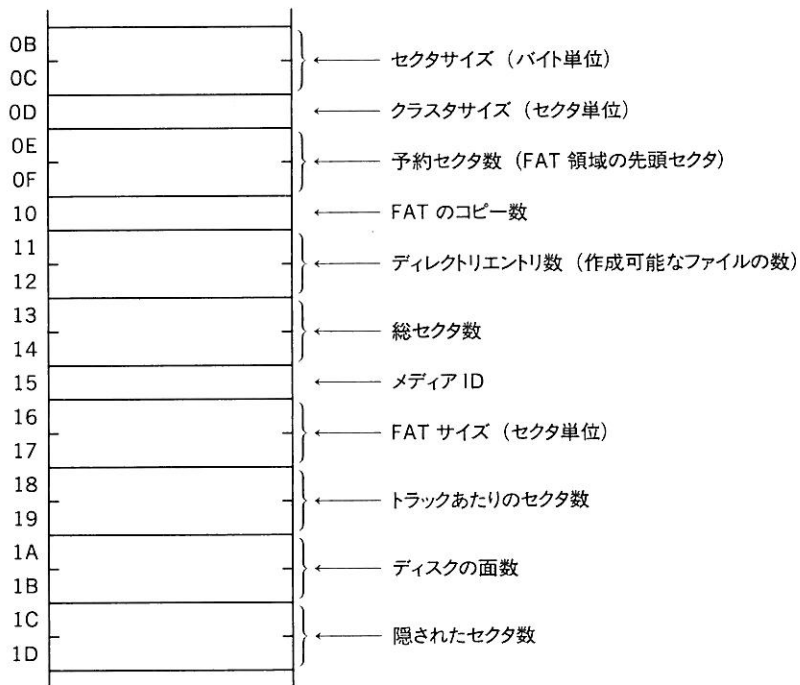


図 3.15 ブートセクタの情報

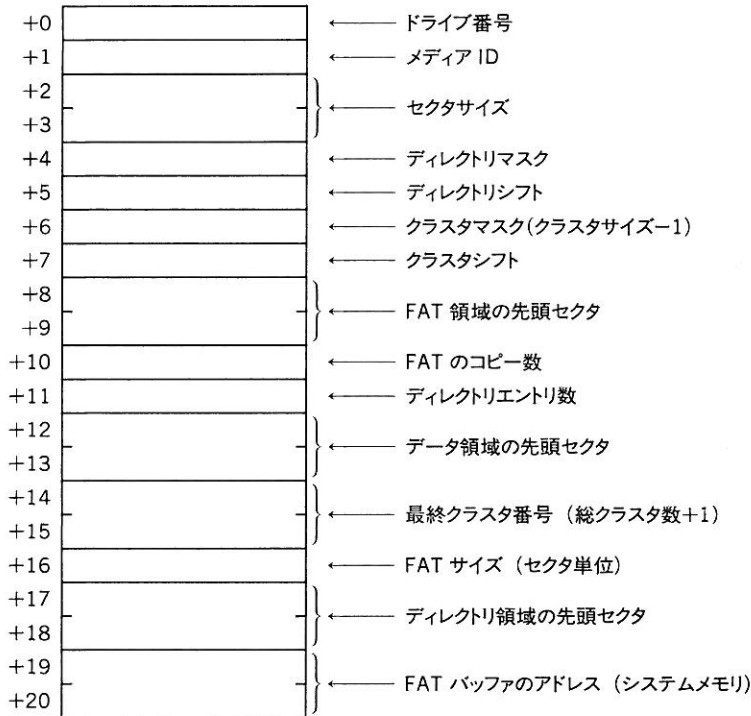


図 3.16 DPB の構造

4. FAT (ファイルアロケーションテーブル)

MSX-DOS では、1 クラスタよりも大きなサイズのファイルは、複数のクラスタにまたがって記録されますが、そのとき連続した番号のクラスタが使用されるとは限りません。特に、ファイルの作成・削除を何度も繰り返した後では、使われなくなったクラスタがディスク上のあちこちに散在した状態になります。この状態でサイズの大きなファイルを作成すると、データは飛び飛びのクラスタに分散して置かれます。そこで、「何番目のクラスタは何番目のクラスタに続いている」、というリンク情報を記録しておく場所が必要になります。それが FAT の役割です。また、未使用クラスタの位置や、不良クラスタが発見されたときに以後そこをアクセスしないように記録する目的にも FAT が利用されます。

FAT に記録されるこのようなクラスタのリンク情報や不良クラスタ情報は、ディスクファイルを管理するうえで不可欠なものであり、一部でも破損してしまうとディスク全体が使用できなくなる恐れがあります。そのため FAT は常に複数個のコピーが用意され、万一に備えています。

FAT の例を図 3.17 に示します。FAT には、

- 先頭の 1 バイトは「FAT ID」と呼ばれ、ディスクのメディアタイプを示す値
- 次の 2 バイトはダミー値の FFH
- その次から、1 クラスタにつき 12 ビットというフォーマットで実際のリンク情報を記録

している FAT エントリ

が記録されます。0番と1番に相当する3バイトが「FAT ID」に使われていますので、ファイルのデータに対応するFATエントリは2番から始まります。FATエントリの番号は、それに対応するクラスタの番号でもあります。FATエントリに記録された12ビットのリンク情報は、図3.18のように並んでいます。リンク情報は、次に続くクラスタ番号を示す値です。もし、FFFHとなっているときは、そのクラスタでファイルが終了したことを意味します。

図3.17の例では、クラスタ#2→クラスタ#3→クラスタ#4という3クラスタ分の大きさのファイルと、クラスタ#5→クラスタ#6の2クラスタ分のファイルが存在していることがわかります。なお、クラスタが番号の小さい順にリンクしているのは図を見やすくするために、実際には番号順であるとは限りません。

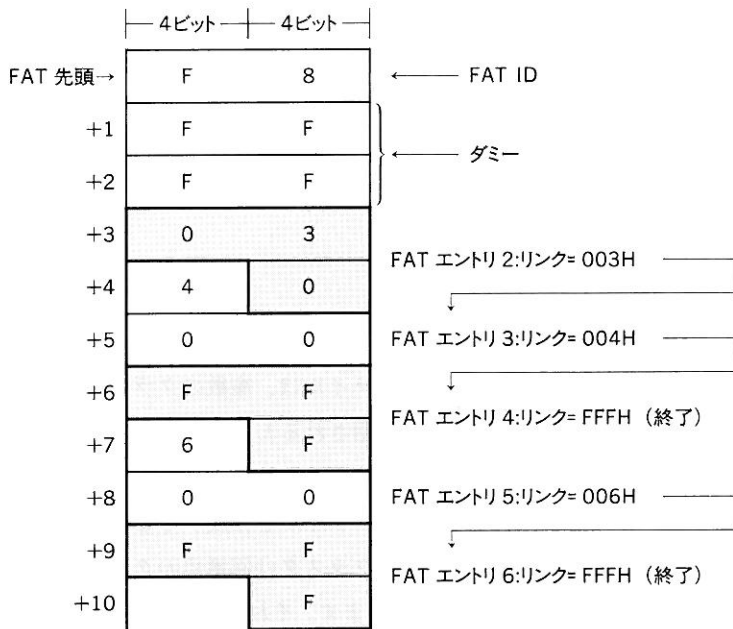


図 3.17 FAT の実例

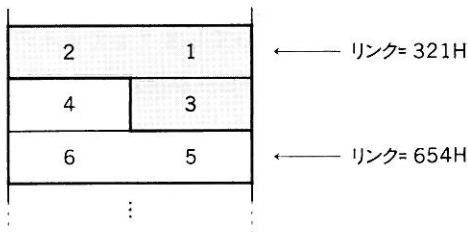


図 3.18 FAT の読み方

5. ディレクトリ

FAT はデータの位置関係などを表すものであり、ファイル自体に関する情報は含んでいません。したがって、そのファイルの名前やそれに付随する情報を知るには、FAT とは別の情報源が必要です。これが「ディレクトリ」です。

ディレクトリはディスク上のディレクトリ領域に記録されていて、図 3.19 のように 32 バイトごとにディレクトリエントリ（ディレクトリの格納場所）が並んでいます。ファイルを作成すると、使われていないディレクトリエントリの中で、いちばん番号が小さいところに目的のファイルのディレクトリが作られます。ファイルが削除されると、該当するディレクトリエントリの最初の 1 バイトに E5H が書き込まれ、そのディレクトリエントリが空いたことを示します。ディレクトリエントリがすべて使用されてしまうと、データ領域がいくら残っていても新しいファイルを作ることはできません。

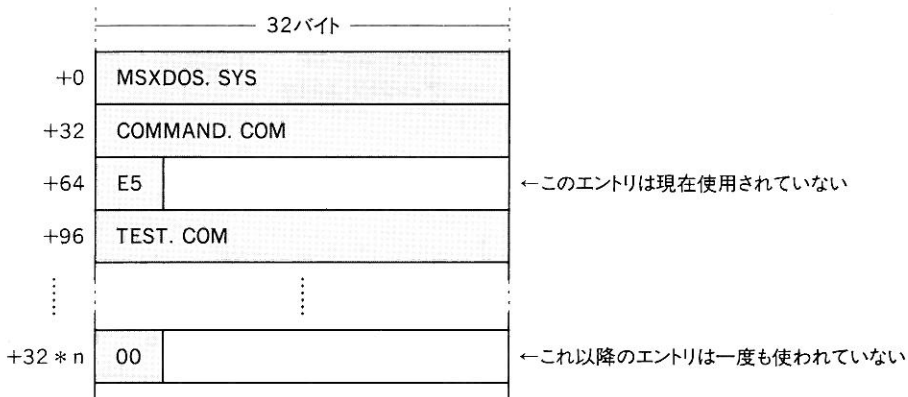


図 3.19 ディレクトリ領域の構造

ディレクトリエントリは図 3.20 のような構造で、それぞれファイル名、ファイル属性、作成・更新の日時、ファイルの先頭クラスタ番号、ファイルサイズの情報を記録しています。

ファイル属性は、ファイルに各種の属性を与えるものです（図 3.21 参照）。MSX-DOS では属性を持ったファイルを作ることはできません。ただし、既存のファイルに不可視・システム・ボリューム・ディレクトリのいずれかの属性が付けられていると、そのファイルはファンクションコールではアクセスできないようになります。読み出し専用（書き込み禁止）属性は無視されません。

日付と時刻は図 3.22 と図 3.23 のように、それぞれ 2 バイトの領域を 3 つのビットフィールドに分割して記録しています。「年」は 7 ビットに 0～99 の値を設定することで、西暦 1980 年～2079 年を表します。「秒」用のビットフィールドは 5 ビットで、時間の分解能は 2 秒です。

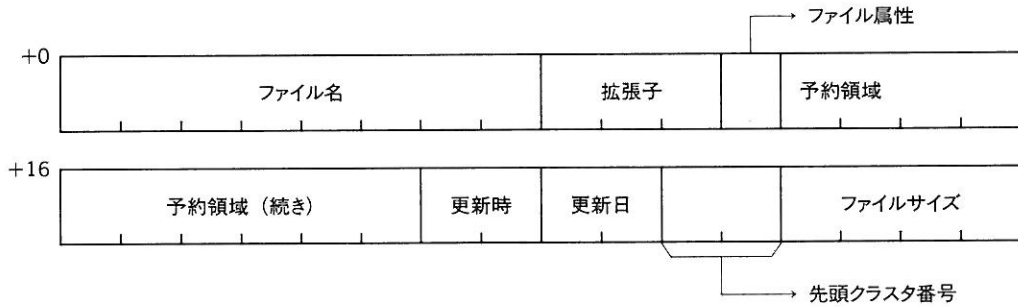


図 3.20 ディレクトリの構造

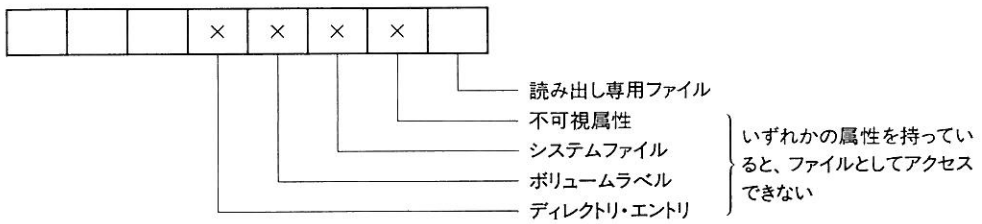


図 3.21 ファイル属性

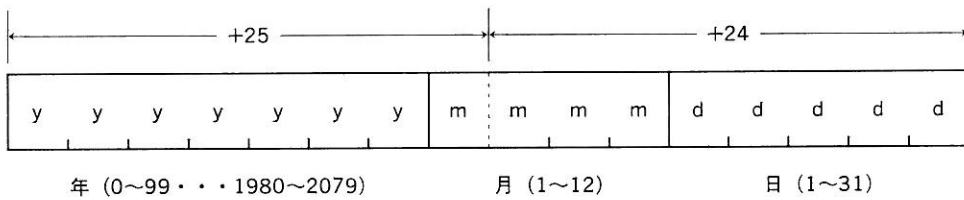


図 3.22 日付を表すビットフィールド

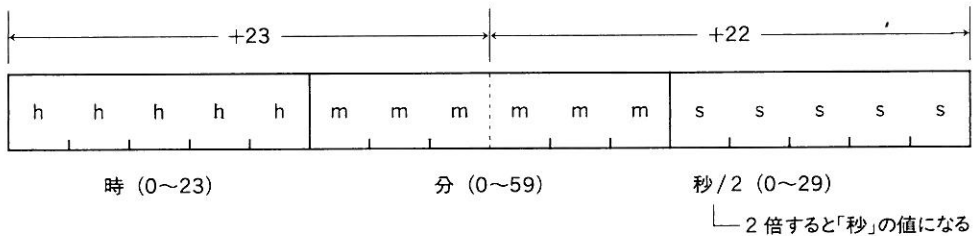


図 3.23 時刻を表すビットフィールド

6. クラスタからセクタへの換算

FATやディレクトリでは、ディスク上のデータの位置はクラスタ単位で表わされています。クラスタで示されたこれらのデータをファンクションコールでアクセスするためには、あるクラスタが何番のセクタに対応しているか、という関係を求めなければなりません。これは、データ領域がクラスタ#2から開始していることを元に、以下のように計算することができます。

1. 与えられたクラスタ番号をCとする。
2. データ領域の開始セクタを調べ、これをS0とする。
3. 1クラスタが何セクタに相当するか調べ、これをnとする。
4. 求めるセクタ番号Sは、 $S = S0 + (C-2) * n$ の計算で得られる。

4章

ファンクションコール

4.1 概要

MSX ディスクシステムの持つ入出力機能は BDOS というかたちで、汎用のサブルーチン形式で ROM にまとめられています。ファンクションコールとは、BDOS を呼び出すためにあらかじめ決められた手順のことで、ユーザープログラムはファンクションコールを実行することでファイルの入出力操作が簡単にできます。

ファンクションコールの役割は、大きく分けると2つがあります。

1. 基本的な機能をあらかじめ用意することによってプログラムの負担を少なくすること。
2. すべてのプログラムが共通の手順を使用することによって、移植性や汎用性を高めること。

ファンクションコールを自由自在に活用することによって、プログラム開発の期間は短縮され、できあがったプログラムは移植性の高いものになります。これは、OS を利用した時の最大のメリットの1つです。

ファンクションコールを実行するには、CPU の C レジスタに決められたファンクション番号を入れ、次のアドレスをコールします。

0005H MSX-DOS 上の外部コマンド
F37DH Disk BASIC 上のマシン語プログラム

例えば、ファンクション番号が 1FH であり、コール手順として A レジスタに 00H をセットすることになっているファンクションコールがあるとすると、このファンクションコールは次のようにして実行します。

```
ld      a, 00h
ld      c, 1fh
call   0005h          ; Disk BASIC のときは 0F37DH
```

CALL 文の後には、戻り値（処理の結果）を受け取ったり、退避していたレジスタを復帰させたりする事後処理が続きます。

Disk BASIC のプログラムからは、CLEAR 文で確保した領域にマシン語プログラムを置き、その開始アドレスを USR 関数で呼び出すことでファンクションコールを利用できます。

4.2 書式

ここでは、以下の書式によってファンクションコールの使用法を紹介します。

ファンクション

ファンクション番号とは、ファンクションコールの種類を判別するためのもので、それぞれのファンクションコールに対応する番号が決められています。ファンクションコールを実行するときには、このファンクション番号を C レジスタにセットします。

コール手順

ファンクションコールの前準備として、レジスタやメモリに必要な値をセットしなければならないことがあります。ここでは、これを「コール手順」の部分に示します。

戻り値

ファンクションコールの結果得られた値は、レジスタやメモリにセットされます。獲得した値が、どこにどのようにセットされるかを、「戻り値」の部分に示します。ただし、ファンクションコールでは、戻り値がセットされるレジスタ以外にも、ファンクション内での作業に使用され、内容が破壊されるレジスタがあります。ファンクションコールでは原則としてレジスタのもとの内容は保証していません。そこで、ファンクションコールを使用するときには、その前に、破壊されては困るレジスタの内容を、適当な場所（スタックなど）に退避して下さい。

解説

ファンクションコールの解説です。

表 3.6 ファンクションコール一覧

ファンクション番号	機能	CP/M*	ページ
00	システムリセット	○	419
01	コンソール1文字入力	○	419
02	コンソール1文字出力	○	420
03	補助入力装置1文字入力	○	421
04	補助出力装置1文字出力	○	421
05	プリンタ1文字出力	○	422
06	コンソール直接入力(入力時エコーバック、 コントロールコードチェック、入力待無)	○	422
07	コンソール直接入力(エコーバック、コント ロールコードチェック無)		423
08	コンソール1文字入力(エコーバック無)		423
09	コンソール文字列出力	○	424
0A	コンソール1行入力	○	424
0B	コンソール入力チェック	○	425
0C	バージョン番号の取り出し	○	425
0D	ディスクリセット	○	426
0E	デフォルトドライブの設定	○	426
0F	ファイルのオープン	○	427
10	ファイルのクローズ	○	427
11	ファイルの検索(ワイルドカードに一致する 最初のファイル)	○	428
12	ファイルの検索(ワイルドカードに一致する 後続のファイル)	○	429
13	ファイルの削除	○	429
14	シーケンシャル読み出し	○	430
15	シーケンシャル書き込み	○	430
16	ファイルの作成	○	431
17	ファイル名の変更	○	432
18	ログインベクトルの獲得	○	432
19	デフォルトドライブの獲得	○	433
1A	転送アドレスの設定	○	433
1B	ディスク情報の獲得		434
21	ランダム読み出し	○	434
22	ランダム書き込み	○	435
23	ファイルサイズの獲得	○	435
24	ランダムレコードの設定	○	436
26	ランダムブロック書き込み		436
27	ランダムブロック読み出し		437
28	ゼロ書き込みを伴うランダム書き込み	○	438
2A	日付の獲得		438
2B	日付の設定		439
2C	時刻の獲得		440
2D	時刻の設定		440
2E	ベリファイフラグの設定		441
2F	論理セクタの読み出し		441
30	論理セクタの書き込み		442

注意 *「○」は、CP/M 2.2 のファンクションとの互換性があるものを示している。

ファンクションコールのファンクション番号は 00H~30H ですが、この中には以下の無効なファンクション番号があります。

■ 1CH~20H、25H、29H

これらの無効なファンクションコールを実行したときには、AレジスタとBレジスタに 00H がセットされる以外は何も行いません。また、31H 以降のファンクション番号でも結果は同じです。

4.3 ファンクションコールの解説

システムリセット

ファンクション

00H

コール手順

なし

戻り値

なし

解説

MSX-DOS 上のプログラムからコールしたときには、MSX-DOS がウォームスタートし、DOS のコマンドレベルに戻ります。Disk BASIC 上のプログラムからコールしたときには、Disk BASIC がウォームスタートし、BASIC のコマンドレベルに戻ります。このとき、ロードされている BASIC のプログラムは破壊されません。

コンソール入力

ファンクション

01H

コール手順

なし

戻り値

A コンソールから入力した1文字

解説

入力がないとき（キーボードバッファが空のとき）には、入力を待ちます。入力された文字はコンソールにエコーバックされます。以下に示すコントロールキャラクタはファンクション内部で処理されるため、入力として扱いません。

CTRL + C	システムリセット
CTRL + P	プリンタへのエコー開始
CTRL + N	プリンタへのエコー停止

CTRL + **C** が入力されるとプログラムの実行を中断してMSX-DOSのコマンドレベルに戻ります。**CTRL** + **P** が入力されると、以降のコンソール出力はすべてプリンタにもエコーします。この状態は、**CTRL** + **N** を入力することによって解除されます。

コンソール出力

ファンクション

02H

コール手順

E 出力する文字コード

戻り値

なし

解説

Eレジスタで指定した文字を画面に表示します。この際、コンソール入力をチェックし、**CTRL** + **C**、**CTRL** + **P**、**CTRL** + **N**、**CTRL** + **S** の処理を行います。**CTRL** + **S** が入力されると、次に任意のキーを押すまでファンクション内部で入力待ちを行うので、プログラムの一時停止ができます。

補助入力

ファンクション

03H

コール手順

なし

戻り値

A 補助入力装置（AUX デバイス）から読み込んだ1文字

解 説

補助入力装置がセットされていないときは、EOF 文字 (1AH) を返します。コンソール入力は、ファンクション 02H と同様にチェックします。

補助出力

ファンクション

04H

コール手順

E 補助出力装置（AUX デバイス）に出力する文字コード

戻り値

なし

解 説

補助出力装置がセットされていないときは何もしません。出力は捨てられます。コンソール入力は、ファンクション 02H と同様にチェックします。

プリンタ出力

ファンクション

05H

コール手順

E プリンタに出力する文字コード

戻り値

なし

解 説

文字コードをプリンタに出力します。コンソール入力は、ファンクション 02H と同様にチェックします。

直接コンソール入出力

ファンクション

06H

コール手順

E FFH 入力
 FFH 以外 セットされた値を文字コードとみなしてコンソールに出力

戻り値

- E レジスタが FFH だったとき（入力）
 - A コンソール入力があるときはその文字コード
 - コンソール入力がないときは 00H
- E レジスタが FFH 以外だったとき（出力）
 - なし

解 説

入力するとき、エコーバック、コントロールキャラクタの処理は行いません（入力文字として扱われる）。また、出力するとき、プリンタへのエコーバックは行いません。

直接コンソール入力

ファンクション

07H

コール手順

なし

戻り値

A コンソールから入力した 1 文字

解 説

エコーバック、コントロールキャラクタの処理は行いません。
このファンクションコールは CP/M との互換性はありません。

コンソール入力

ファンクション

08H

コール手順

なし

戻り値

A コンソールから入力した 1 文字

解 説

エコーバックは行いません。コントロールキャラクタはファンクション 01H と同様に処理します。
このファンクションコールは CP/M との互換性はありません。

文字列出力

ファンクション

09H

コール手順

DE コンソールに出力する文字列があるメモリの先頭アドレス

戻り値

なし

解説

文字列の最後には、終端文字（ターミネータ）として「\$」（24H）を置きます。最初に現れる「\$」の直前までの文字列をコンソールに出力するので、「\$」文字を出力することはできません。

コントロールキャラクタはファンクション 02H と同様にチェックします。

文字列入力

ファンクション

0AH

コール手順

DE 行バッファの先頭アドレス
行バッファ+0 最大入力文字数（1～255）（図 3.22 参照）

戻り値

行バッファ+1 実際に入力された文字数
行バッファ+2～ コンソールから入力された文字列

解説

リターンキーの直前までをコンソールからの入力と見なします。最大入力文字数以上を入力することはできません。このファンクションコールによる文字列入力時には、テンプレートによる編集ができます。

このファンクションコールは、コントロールキャラクタをチェックします。

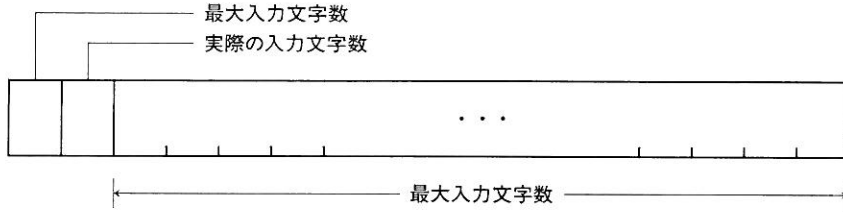


図 3.24 行バッファの構造

コンソール入力状態のチェック

ファンクション

0BH

コール手順

なし

戻り値

A FFH コンソール入力がある
 00H コンソール入力がない

解 説

コンソール入力の状態を調べます。入力文字があるときもその文字は取り出しません。ただし、コントロールキャラクタはファンクション 02H と同様にチェックします。入力文字は、ファンクション 01H か 08H で取り出すことができます。

バージョン番号の獲得

ファンクション

0CH

コール手順

なし

戻り値

HL 0022H

解説

このファンクションコールは、CP/Mのバージョン番号を獲得するためのものです。
MSX-DOSでは、必ず0022Hが戻ります。

ディスクリセット

ファンクション

0DH

コール手順

なし

戻り値

なし

解説

変更されて、まだディスクに書き込まれていないセクタがあれば、それをディスクに書き込んだ後、デフォルトドライブをAドライブにセットし、DTAを0080Hにセットします。

デフォルトドライブの設定

ファンクション

0EH

コール手順

E デフォルトドライブ番号 (A = 0、B = 1、・・・ H = 7)

戻り値

なし

解説

デフォルトドライブを変更します。FCBなどのドライブ番号に「0」が指定されたときに有効です。接続されていないドライブが指定されたときには、変更しません。

ファイルのオープン

ファンクション

0FH

コール手順

DE	オープンされていないFCBの先頭アドレス
FCBのカレントブロック	読み出すブロック番号

戻り値

■オープン成功のとき

A 0

FCBの各フィールドが設定される

■オープン失敗のとき

A 0FFH

解説

FCBがファイル入出力のファンクションコールで使用できるようになります。レコードサイズ、カレントレコード、ランダムレコードの各フィールドは使用するファンクションに応じてオープンの後、ユーザープログラムで設定しなければなりません。

ファイルのクローズ

ファンクション

10H

コール手順

DE オープンされたFCBの先頭アドレス

戻り値

A	00H	クローズが成功
	FFH	クローズが失敗

解説

現在の FCB の内容をディスク上の該当するディレクトリエリアに書き込むことによって、ファイルの更新に関する整合性を保ちます。ただし、ファイルの内容を変更していないときには、書き込みは行いません。クローズした FCB はオープンされていない FCB として再利用できます。また、オープンされた FCB としてさらに入出力を続けることもできます。

ファイルの検索（最初の一致）

ファンクション

11H

コール手順

DE オープンされていない FCB の先頭アドレス

戻り値

A	00H	ファイルが見つかった
	FFH	ファイルが見つからなかった

ファイルが見つかったときは、DTA で指示される領域にドライブ番号を、それに続く 32 バイトにそのファイルのディスク上のディレクトリエントリをセットする。

解説

ファイル名にはワイルドカード文字の「?」を使用することができます。例えば、ファイル名の名前部分に「???????」、拡張子部分に「MAC」を設定すると、「MAC」という拡張子をもつファイルのうち、最初にマッチしたファイルのディレクトリ情報を得ることができます。ファンクションコールレベルでは、ワイルドカード文字の「*」はサポートしていないので、適当な数の「?」に置き換えて下さい。一致するファイルすべてを検索したいときや、一致するファイルが1個だけかどうかを知りたいときには、ファンクション 12H を利用します。

DTA 領域に設定されたデータはオープンされていない FCB として使用できます。

ファイルの検索（後続の一致）

ファンクション

12H

コール手順

なし

戻り値

A	00H	ファイルが見つかった
	FFH	ファイルが見つからなかったとき

ファイルが見つかったときは、DTA で指示される領域にドライブ番号を、それに続く 32 バイトにそのファイルのディスク上のディレクトリエントリをセットする。

解説

このファンクションコールは、ファンクション 11H のワイルドカード文字によるファイル名の指定と一致した複数のファイルを検索するときに使います。単独で使用することはできません。このファンクションコールを繰り返し使用して、ファンクション 11H によって一致した複数のファイルのディレクトリ情報を、1 つずつ順番に獲得していくことができます。

DTA 領域に設定されたデータはオープンされていない FCB として使用できます。

ファイルの削除

ファンクション

13H

コール手順

DE	オープンされていない FCB の先頭アドレス
----	------------------------

戻り値

A	00H	削除が成功
	FFH	削除が失敗

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のカレントレコード	書き込むレコード番号
DTA 以降の 128 バイト	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解説

FCB のカレントブロックとカレントレコードは、書き込み後に次のレコードを指すように自動的に更新されます。したがって、連続して書き込むときには、カレントブロックとカレントレコードを再設定する必要はありません。

このファンクションコールが失敗するのは、ディスクの残り容量がなくなったときです。

ファイルの作成

ファンクション

16H

コール手順

DE	オープンされていない FCB の先頭アドレス
FCB のカレントブロック	読み出すブロック番号

戻り値

A	00H	ファイルの作成が成功
	FFH	ファイルの作成が失敗

解説

FCB 中のレコードサイズ、カレントレコード、ランダムレコードの各フィールドは、このファンクションコールの実行後に必要に応じて設定しなければなりません。

このファンクションコールが失敗するのは、ファイル名が正しくセットされていないか、ディレクトリに空きがないときです。

ファイル名の変更

ファンクション

17H

コール手順

DE オープンされていないFCBの先頭アドレス。
 FCB+0 旧ファイル名
 FCB+16 新ファイル名

戻り値

A 00H ファイル名の変更が成功
 FFH ファイル名の変更が失敗

解説

新旧ファイル名に、ワイルドカード文字を使用できます。例えば、旧ファイル名に「???????REL」を、新ファイル名に「???????LIB」を指定すれば、「REL」という拡張子のファイルがすべて、「LIB」という拡張子のファイルに変更されます。

新ファイル名のドライブ番号は無視されます。

ログインベクトルの獲得

ファンクション

18H

コール手順

なし

戻り値

HL オンラインドライブ情報

解説

MSXには最大8台のディスクドライブが接続できます。「オンラインドライブ」とは、MSXに実際に接続されているドライブを示します。このファンクションコールを実行すると、各ドライブがオンラインであるかどうかを調べ、結果を表3.7のようにHLレジ

スタに入れて返します。それぞれのビットが1ならば対応するドライブはオンラインであり、0ならば接続されていないことを示します。MSX-DOS のときは、H レジスタは常に「0」です。

表 3.7 ログインベクトル

レジスタ	H								L							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
ビット																
ドライブ	常に、0								H:	G:	F:	E:	D:	C:	B:	A:
オンライン・オフライン	各ビットが「1」ならばオンライン、「0」ならばオフライン															

デフォルトドライブ番号の獲得

ファンクション

19H

コール手順

なし

戻り値

A デフォルトドライブ番号 (A = 0、B = 1、・・・ H = 7)

解説

現在設定されているドライブ番号を獲得します。

転送先アドレス (DTA) の設定

ファンクション

1AH

コール手順

DE 設定する DTA アドレス

戻り値

なし

解 説

DTA アドレスは、システムリセット時に 0080H に初期化されますが、このファンクションコールによって、任意のアドレスに設定し直すことができます。

ディスク情報の獲得

ファンクション

1BH

コール手順

E 目的のディスクが入っているドライブ番号

戻り値

A 1 クラスタあたりの論理セクタ数
FFH 指定ドライブがオフライン
BC セクタのサイズ (バイト単位)
DE クラスタの総数
HL 未使用クラスタの総数
IX DPB の先頭アドレス
IY FAT バッファの先頭アドレス

解 説

指定ドライブのディスクの情報を獲得します。ドライブ番号に「0」を指定すると、デフォルトドライブの指定になります。それ以外は、Aドライブなら「1」、Bドライブなら「2」、・・・を指定します。

このファンクションコールは MSX-DOS 独自のものです、CP/M との互換性はありません。

ランダムな読み出し

ファンクション

21H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のランダムレコード	読み出すレコード番号

戻り値

A	00H	読み出しが成功
	01H	読み出しが失敗

読み出しが成功したときには、DTA 以降の 128 バイトに読み出したレコードをセットします。

解説

レコードの大きさは 128 バイト固定です。

ランダムな書き込み

ファンクション

22H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のランダムレコード	書き込むレコード番号
DTA 以降の 128 バイト	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解説

レコードの大きさは 128 バイト固定です。

ファイルサイズの獲得

ファンクション

23H

コール手順

DE オープンされた FCB の先頭アドレス

戻り値

A 00H ファイルサイズの獲得が成功
FFH ファイルサイズの獲得が失敗
獲得が成功すれば、FCB のランダムレコードフィールドに指定されたファイルの 128 バイト単位のサイズをセットします。

解説

ファイルのサイズは 128 バイト単位です。200 バイトのファイルであれば「2」が、257 バイトのファイルであれば「3」が設定されます。

ランダムレコードフィールドの設定

ファンクション

24H

コール手順

DE オープンされた FCB の先頭アドレス
FCB のカレントブロック 目的のブロック
FCB のカレントレコード 目的のレコード

戻り値

ランダムレコードフィールドに、指定された FCB のカレントブロックフィールドとカレントレコードフィールドから計算したカレントレコードポジションを設定します。

ランダムブロック書き込み

ファンクション

26H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のレコードサイズ	書き込むレコードサイズ
FCB のランダムレコード	書き込みを開始するレコード
HL	書き込むレコード数
DTA 以降のメモリ領域	書き込むデータ

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

機能

書き込みが終わると、ランダムレコードフィールドの値が自動的に更新されて、最後に書き込んだレコードの次のレコードを指します。1つのレコードの大きさはFCBのレコードサイズフィールドによって1バイトから65535バイトまで任意に設定できます。このファンクションコールはMSX-DOS独自のものです、CP/Mとの互換性はありません。

ランダムブロック読み出し

ファンクション

27H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のレコードサイズ	読み出すレコードサイズ
FCB のランダムレコード	読み出しを開始するレコード
HL	読み出すレコード数

戻り値

A	00H	読み出しが成功
	01H	読み出しが失敗
HL		実際に読み込んだレコードの個数

解 説

データの読み込みが終わると、ランダムレコードフィールドの値が自動的に更新されます。このファンクションコール実行後、HL レジスタには実際に読み込んだレコードの総数が設定されます。つまり、指定した数のレコードを読み込み終わる前にファイルの終りに達してしまったときには、それまでに読み込んだ実際のレコード数が HL レジスタに入ることになります。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

「0」の書き込みをともなうランダムな書き込み

ファンクション

28H

コール手順

DE	オープンされた FCB の先頭アドレス
FCB のランダムレコード	書き込むレコード番号
DTA 以降の 128 バイト	書き込むレコード

戻り値

A	00H	書き込みが成功
	01H	書き込みが失敗

解 説

レコードの大きさは 128 バイト固定です。

このファンクションコールでは、ファイルサイズより大きなレコード番号が指定されたとき、ファイルの終りの次のレコードから指定されたレコードの直前までのレコードを「0」で埋めた後、指定されたレコードに書き込みます。「0」で埋める点を除けば、ファンクション 22H（ランダムな書き込み）と同じです。

日付の獲得

ファンクション

2AH

コール手順

なし

戻り値

HL 年 (1980～2079)
 D 月 (1～12)
 E 日 (1～31)
 A 曜日 (0 =日曜、1 =月曜、・・・、6 =土曜)

機能

クロック IC に設定されている日付のデータを獲得します。

このファンクションコールは MSX-DOS 独自のものです、CP/M との互換性はありません。

日付の設定

ファンクション

2BH

コール手順

HL 年 (1980～2079)
 D 月 (1～12)
 E 日 (1～31)

戻り値

A 00H 日付の設定が成功
 FFH 日付の設定が失敗

機能

クロック IC に指定した日付を設定します。

このファンクションコールは MSX-DOS 独自のものです、CP/M との互換性はありません。

時刻の獲得

ファンクション

2CH

コール手順

なし

戻り値

H	時
L	分
D	秒
E	1/100 秒

機能

クロック IC に設定されている時刻のデータを獲得します。
このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

時刻の設定

ファンクション

2DH

コール手順

H	時
L	分
D	秒

戻り値

A	00H	時刻の設定が成功
	FFH	時刻の設定が失敗

機 能

クロック IC に時刻のデータを設定します。

このファンクションコールは MSX-DOS 独自のものです、CP/M との互換性はありません。

ベリファイフラグの設定

ファンクション

2EH

コール手順

E	00H	ベリファイフラグをリセット
	01H	ベリファイフラグをセット

戻り値

なし

機 能

ベリファイフラグをセットすると、以降のディスクに対する書き込みをベリファイ付きで行います。ベリファイとは、書き込んだあとでその内容をディスクから読み込み、書き込むべき内容と比較して等しいかどうかをチェックする機能です。

ベリファイ機能はオプションであり、ドライブによってはその機能がないものもあります。その場合、ベリファイフラグを設定しても意味はありません。

このファンクションコールは MSX-DOS 独自のものです、CP/M との互換性はありません。

論理セクタの読み出し

ファンクション

2FH

コール手順

DE	読み出す論理セクタの番号 (複数のときはその先頭の論理セクタ番号)
H	読み出す論理セクタの個数

L 読み出すディスクのドライブ番号 (A = 0、B = 1、・・・、H = 7)

戻り値

DTA 以降の (論理セクタサイズ×論理セクタの個数) バイトに読み込んだ内容を設定

機能

指定ドライブの指定論理セクタから指定された個数の論理セクタを読み出し、その内容を DTA 以降のメモリに設定します。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。

論理セクタの書き込み

ファンクション

30H

コール手順

DE 書き込む論理セクタの番号 (複数のときはその先頭の論理セクタ番号)

H 書き込む論理セクタの個数

L 書き込むディスクのドライブ番号 (A = 0、B = 1、・・・、H = 7)

DTA 以降の (論理セクタサイズ×論理セクタの個数) バイトに書き込むデータを設定

戻り値

なし

機能

DTA 以降に設定されたデータを指定ドライブの指定論理セクタへ、指定された論理セクタの個数だけ書き込みます。

このファンクションコールは MSX-DOS 独自のもので、CP/M との互換性はありません。



第4部
VDP

MSX は画面表示のために Video Display Processor (VDP) と呼ばれる専用の LSI を採用しています。この LSI を直接操作することにより、高速なグラフィック表示が可能となります。MSX1 では VDP として TMS9918A が、MSX2 では V9938 が、MSX2+では V9958 が使用され、高度なグラフィックシステムを提供しています。

第4部では、V9938 と V9958 について詳しく説明します。

1 章

V9938 の構成

「V9938」(MSX-VIDEO) は、MSX2 用の VDP (Video Display Processor) として開発された VLSI です。MSX2 の開発にあたっては、TMS9918A (MSX1 の VDP) とソフトウェア上位互換性があり、さらに機能強化されたビデオプロセッサが必要でした。V9938 はこの条件を満たす VDP として、株式会社アスキーとヤマハ株式会社によって共同開発され、以下の特徴があります。

- フルビットマップ表示モード
- テキスト 80 文字表示
- X、Y 座標アクセスを実現し、I/O ドライバの負荷軽減と画面モードからの独立が可能
- 基本コマンドのハードウェア実行により、I/O ドライバの処理の軽減と高性能を実現
- デジタイズと外部同期機能の実現
- カラーパレット (9 ビット×16 種) とアナログ RGB 出力をサポート
- 1 水平ライン上の表示スプライト数の増加
- ハードウェアによる垂直スクロール

1.1 レジスタ

V9938 では、その機能を利用するために内部に「VDP レジスタ」を備えています。特に断りのない場合、これを単にレジスタと呼びます。レジスタは機能によって、

1. コントロールレジスタ
2. ステータスレジスタ
3. パレットレジスタ

に分けられます。この内の、コントロールレジスタとステータスレジスタについては、BASIC の VDP (n) というシステム変数によって参照することができます。

1.1.1 コントロールレジスタ(R#0~R#23、R#32~R#46)

V9938 の動作を制御する書き込み専用の8ビットレジスタ群です。一般に R#n の記号で表します。R#0~R#23 は主に画面モードの設定に使用し、R#32~R#46 は VDP コマンドの実行時に使用します。V9938 には R#24~R#31 に相当するコントロールレジスタはありません。

1.1.2 ステータスレジスタ(S#0~S#9)

V9938 から得られる各種の情報を読みとるための、読み出し専用の8ビットレジスタ群です。一般に S#n という記号で表します。

1.1.3 パレットレジスタ(P#0~P#15)

カラーパレットを設定するための書き込み専用のレジスタ群です。一般に P#n という記号で表します。n はパレット番号を意味し、各パレットを 512 色中の 1 色に設定します。1 つのパレットレジスタは 9 ビット長になっており、RGB 各色につき、それぞれ 3 ビットずつ使用します。

カラーパレットは、リセット時には表 4.1 のように設定されます。「0」は輝度が 0 であることを意味し、「7」は輝度が最も高いことを意味しています。

ただし、GRAPHIC 7*モードではカラーパレットを使わないため、この表は意味を持ちません。また、GRAPHIC 5*モードではカラーコードが 0~3 の 4 種類しか存在しないため、表 4.1 の 4~F は意味を持ちません。

注意

* この GRAPHIC モードは、BASIC のスクリーンモードとは異なります。
詳しくは、「4章 V9938 の画面モード」を参照して下さい。

表 4.1 リセット時のカラーパレット設定値

カラーコード	G	R	B
0	0	0	0
1	0	0	0
2	6	1	1
3	7	3	3
4	1	1	7
5	3	2	7
6	1	5	1
7	6	2	7
8	1	7	1
9	3	7	3
A	6	6	1
B	6	6	4
C	4	1	1
D	2	6	5
E	5	5	5
F	7	7	7

1.2 VRAM

V9938 には 128K バイトの VRAM (VIDEO RAM) と 64K バイトの拡張 RAM を接続できます。V9938 はこの 128K バイトのアドレス空間をアクセスするために 17 ビットのアドレスカウンタを持っています。なお、このメモリは V9938 によって管理されるものですから、CPU から直接アクセスすることはできません。

拡張 RAM は VRAM のように、その内容を画面に表示することはできませんが、VDP コマンド実行時には、ワークエリアとして、VRAM と同等に扱うことができます。ただし、MSX2 では拡張 RAM は使用しません。

1.3 I/Oポート

V9938 は CPU とデータのやり取りを行うために、4つの I/O ポートを持っています。このポートは表 4.2 の機能を持ち、CPU の I/O アドレスを通して接続されています。表中 m、n で表されたアドレスは MAIN ROM の 6、7 番地に記録されています。アプリケーションプログラムは、この番地の内容を参照して VDP をアクセスして下さい。

表 4.2 V9938 のポート

ポ ー ト	ア ド レ ス	用 途
ポート#0 (READ)	m	VRAM からのデータを読み出し
ポート#0 (WRITE)	n	VRAM へのデータ書き込み
ポート#1 (READ)	m+1	ステータスレジスタの読み出し
ポート#1 (WRITE)	n+1	コントロールレジスタへの書き込み
ポート#2 (READ)	n+2	パレットレジスタへの書き込み
ポート#3 (WRITE)	n+3	間接指定されたレジスタへのデータ書き込み

注 意 m の値は MAIN ROM の 6 番地を参照して得る。
n の値は MAIN ROM の 7 番地を参照して得る。

MSX BASIC や BIOS では VDP の書き込み専用レジスタの値を参照する必要が生じたときのために、それらのレジスタに書き込む値をメインメモリのワークエリアや VRAM の空き領域に保存しています。

ユーザープログラムが、この保存されている値を更新しないで、書き込み専用レジスタの値を変更したときには、その後の BASIC や BIOS の動作は保証されません。

2章

基本入出力

この章では、VDP を操作するのに必要なコントロールレジスタ、パレットレジスタ、ステータスレジスタ、VDP にアクセスする方法などについて説明します。

2.1 コントロールレジスタのアクセス

V9938 のコントロールレジスタ (R #0~R #46) にデータをセットする方法は、直接指定と間接指定の 2 とおりあります。

2.1.1 直接指定

ポート#1 に、データ、レジスタ番号の順に出力する方法です。この順番は必ず守らねばならないので、割り込みルーチンなどで V9938 をアクセスしているときには注意が必要です。

	7	6	5	4	3	2	1	0	
Port # 1 1st byte	D7	D6	D5	D4	D3	D2	D1	D0	1. ポート#1 にデータを出力
Port # 1 2nd byte	1	0	R5	R4	R3	R2	R1	R0	2. ポート#1 にレジスタ番号を出力 (上位 2 ビットは常に「10」)

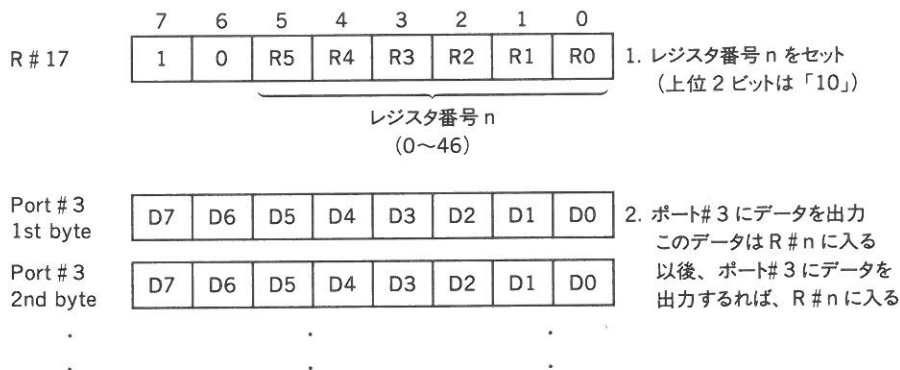
2.1.2 間接指定

コントロールレジスタ R #17 (Control register pointer) でレジスタ番号を指定する方法で、非オートインクリメントとオートインクリメントの 2 つのモードがあります。

1. 非オートインクリメントモード

直接指定で R # 17 にレジスタ番号をセットし、ポート#3 にデータを出力します。R # 17 の値は間接指定では変更できません。

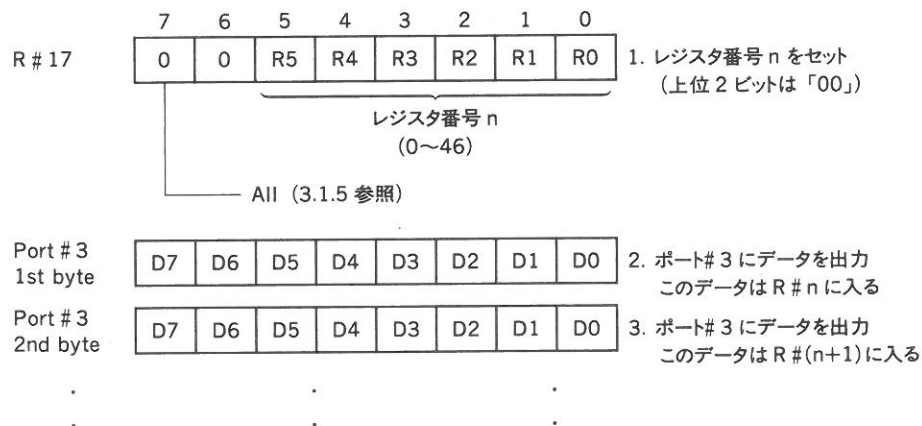
非オートインクリメントモードでは、R # 17 の内容は不変なので、再セットする必要はありません。このモードは VDP コマンドの実行時など、同じレジスタに連続してデータを送るときに使用します。



2. オートインクリメントモード

直接指定で R # 17 にレジスタ番号をセットし、ポート#3 にデータを出力します。このとき R # 17 の AII ビットに「0」をセットすると、ポート#3 にデータを出力するたび、R # 17 の内容はオートインクリメントされます。

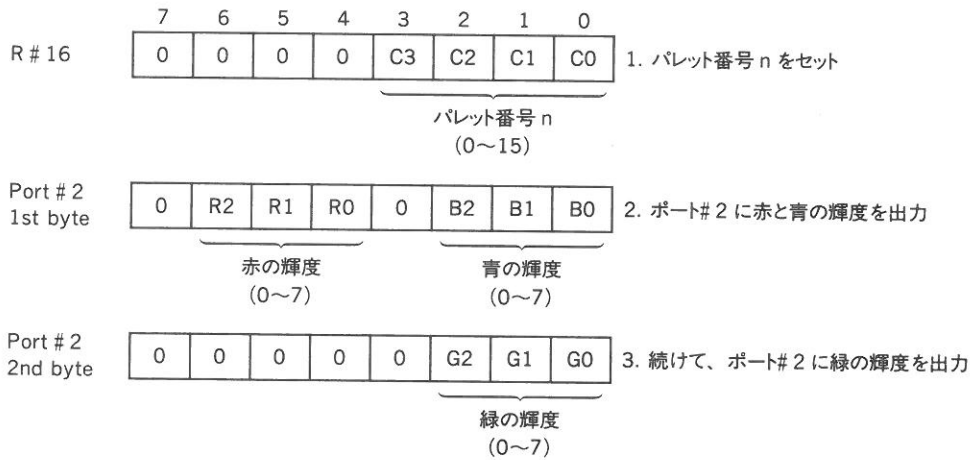
このモードを利用すると、スクリーンモードの変更時など、連続した多数のレジスタを一度に書き換える場合に便利です。



2.2 パレットレジスタのアクセス

パレットレジスタにデータをセットするためには、レジスタ R # 16 (Color palette address pointer) にパレット番号をセットします。そして、ポート # 2 にデータを 2 バイト連続で出力します。これは、V9938 のパレットレジスタ (P # 0 ~ P # 15) は 9 ビットの長さを持つため、最初に赤と青の輝度、次に緑の輝度とデータを 2 度に分けて出力する必要があるからです。

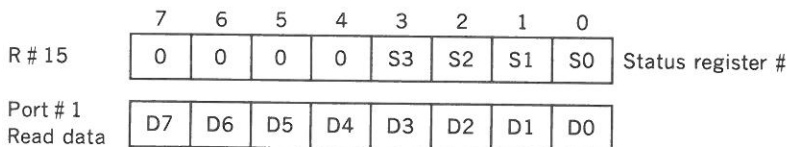
R # 16 は、ポート # 2 にデータを 2 バイト書き込むとオートインクリメントします。



2.3 ステータスレジスタのアクセス

V9938 のステータスレジスタ (S # 0 ~ S # 9) を読み出すには、レジスタ R # 15 (Status register pointer) にステータスレジスタ番号をセットし、ポート # 1 を読み出します。

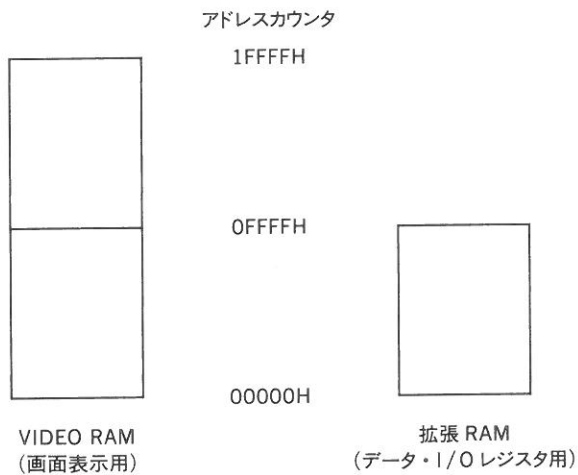
MSX の割り込み処理ルーチンは、R # 15 に「0」が設定されていることを前提にしています。したがって、ステータスレジスタをアクセスする前には、必ず割り込みを禁止し、目的の処理が終了した後に、R # 15 に「0」をセットしてから割り込みを解除して下さい。



2.4 VRAM(VIDEO RAM)のアクセス

V9938には128KバイトのVRAMと64Kバイトの拡張RAMを接続することができます。拡張RAMはVDPコマンド実行時にワークエリアとして使用することができます。ただし、MSX2では拡張RAMは使用しないので、R#45のビット6には常に「0」を設定して下さい。

メモリマップは以下のようになっています。



VRAM アクセスは次の手順で行います。

1. バンク切り換え (VRAM～拡張 RAH)
2. アドレスカウンタセット (A16～A14)
3. アドレスカウンタセット (A7～A0)
4. アドレスカウンタセット (A13～A8) 読み出し・書き込み指定
5. データの読み出し・書き込み

1. アドレスカウンタセット(A16~A14)

レジスタ R#14 (VRAM Access base address register) を使って、アドレスカウンタの上位3ビット (A16~A14) をセットします。

このレジスタは、TMS9918 互換モードのときには、オートインクリメントを指定してもカウントアップしません。

	7	6	5	4	3	2	1	0
R#14	0	0	0	0	0	A16	A15	A14

2. アドレスカウンタセット(A7~A0)

ポート#1 に、アドレスカウンタの下位8ビット (A7~A0) にセットするデータを出力します。

	7	6	5	4	3	2	1	0
Port #1	A7	A6	A5	A4	A3	A2	A1	A0

3. アドレスカウンタセット(A13~A8) 読み出し・書き込みの指定

ポート#1 に、アドレスカウンタの残り6ビット (A13~A8) にセットするデータと読み出し・書き込みの指定を出力します。

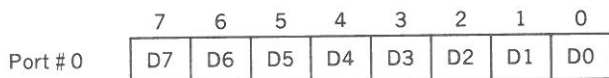
	7	6	5	4	3	2	1	0
Port #1	0	R/W	A13	A12	A11	A10	A9	A8

└─ 0 リード
 └─ 1 ライト

4. データの読み出し・書き込み

ポート#0 にデータを読み出し・書き込みをすると、アドレスカウンタが自動的にインクリメントされるので、連続してアクセスすることができます。

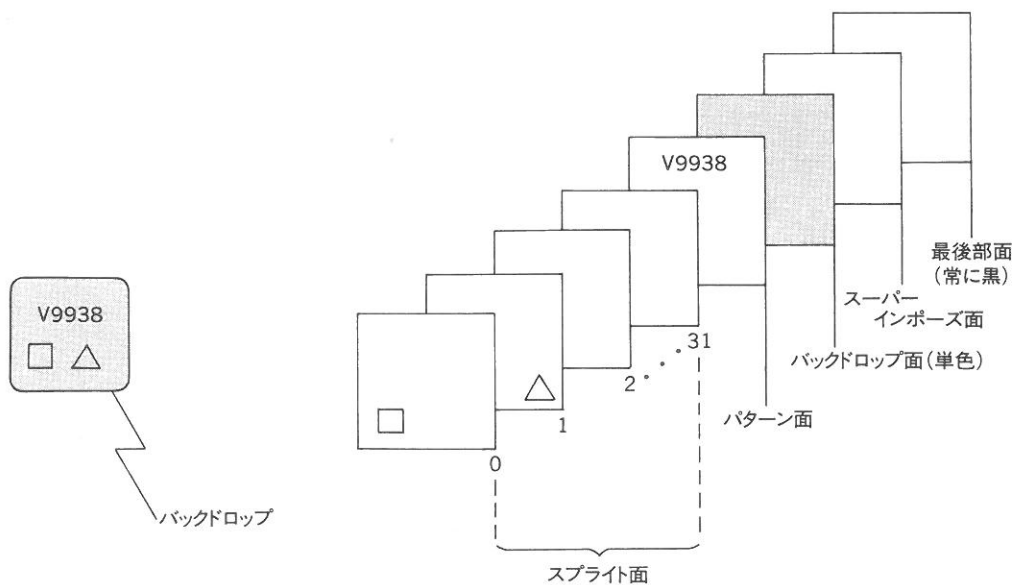
ただし、TMS9918 互換モードのときには、A16～A14 はインクリメントされません。



VRAM のアクセスには、VDP コマンドを使用する方法もあります。これについては5章で詳しく述べます。

2.5 V9938 の画面構成

V9938 の画面は以下のような構成になっています。



3章

レジスタの機能

この章では、VDP を制御するコントロールレジスタ、ステータスレジスタについて説明します。

3.1 コントロールレジスタ#0～#23、#32～#46(Write only)

3.1.1 モードレジスタ

モードレジスタは、各種動作モードを設定するレジスタで、コントロールレジスタ#0、#1、#8、#9に配置されています。

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	M5	M4	M3	0

- DG 1 のとき、カラーバスを入力モードにして、データを VRAM に取り込む
(デジタイズ機能を持った MSX2 でのみ使用可能)
- IE2 Interrupt Enable2(1 のとき、ライトペンによる割り込みを可能にする)
- IE1 Interrupt Enable1(1 のとき、水平帰線による割り込みを可能にする)
- M5 表示モードの設定に使用する
- M4 表示モードの設定に使用する
- M3 表示モードの設定に使用する

IE2はライトペン割り込み用のビットです。MSXではこの機能は使用していないので、常に「0」にして下さい。

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	M1	M2	0	SI	MAG

- BL 1=画面表示、0=画面非表示
- IE0 Interrupt Enable0(1のとき、垂直帰線による割り込みを可能にする)
- M1 表示モードの設定に使用する
- M2 表示モードの設定に使用する
- SI スプライトのサイズ 1=16×16、0=8×8
- MAG スプライトの拡大 1=拡大する、0=拡大しない

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

- MS 1=マウスを使用する(カラーバスは入力モード)、0=マウスを使用しない(カラーバスは出力モード)
- LP 1=ライトペンを使用する、0=ライトペンを使用しない
- TP カラーコード0の色をカラーパレットの色にする
- CB 1=カラーバスを入力モードにする、0=カラーバスを出力モードにする
- VR VRAMの種類を選択する
1=64K×1bit または 64K×4bit、0=16K×1bit または 16K×4bit
- SPD 1=スプライト非表示、0=スプライト表示
- BW 1=白黒32階調、0=カラー(Composit encoderにのみ有効)
BWはMSXでは使用していません。

MSはライトペン用の、LPはマウス用のレジスタです。MSXではこの機能は使用していないので、常に「0」にして下さい。

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	NT	DC

LN	1 = 縦 212 ドット表示、0 = 縦 192 ドット表示
S1	同期モード選択 (「7.4 同期モードの選択」参照)
S0	同期モード選択 (「7.4 同期モードの選択」参照)
IL	1 = Interlace (完全 NTSC タイミング)、0 = Non Interlace (不完全 NTSC タイミング)
EO	1 = Even field / Odd field で 2 枚の絵を交互に表示、0 = Even field / Odd field で同じ絵を表示
NT	1 = PAL (313line)、0 = NTSC (262line) RGB 出力のみ有効
DC	1 = DLCLK 端子を入力モードにする、0 = DLCLK 端子を出力モードにする

3.1.2 テーブルベースアドレスレジスタ

V9938 に対して VRAM 上の各テーブルの先頭アドレスを宣言するためのレジスタ群です。

表示モードによっては、設定できるデータの値に制限 (実際のアドレスと異なる場合) があるので注意して下さい。詳しくは、各表示モードの解説を参照して下さい。

Pattern name table base address register

	7	6	5	4	3	2	1	0
R # 2	0	A16	A15	A14	A13	A12	A11	A10

Color table base address register low

	7	6	5	4	3	2	1	0
R # 3	A13	A12	A11	A10	A9	A8	A7	A6

Color table base address register high

	7	6	5	4	3	2	1	0
R # 10	0	0	0	0	0	A16	A15	A14

Pattern generator table base address register

	7	6	5	4	3	2	1	0
R # 4	0	0	A16	A15	A14	A13	A12	A11

Sprite attribute table base address register low

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	1	1

Sprite attribute table base address register high

	7	6	5	4	3	2	1	0
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

3.1.3 カラーレジスタ

V9938 の表示色、ブリンクなどを制御するためのレジスタ群です。

Text color / Back drop color register

	7	6	5	4	3	2	1	0					
R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	(GRAPHIC 7 モード時以外)				
					BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0	(GRAPHIC 7 モード時)

TC3~TC0 TEXT 1、TEXT 2 モードにおけるテキストの色を指定

BD3~BD0 GRAPHIC 7 以外の表示モードにおけるバックドロップの色を指定

BD7~BD0 GRAPHIC 7 モードにおけるバックドロップの色を指定

Text color / Back color register

	7	6	5	4	3	2	1	0
R#12	T23	T22	T21	T20	BC3	BC2	BC1	BC0

TEXT 2 モードにおいてパターンにブリンクの属性がついているときは、このレジスタで指定された色と R#7 で指定された色が交互に表示されます。

T23~T20 パターンの 1 の部分の色を指定

BC3~BC0 パターンの 0 の部分の色を指定

Blinking period register

	7	6	5	4	3	2	1	0
R # 13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0

GRAPHIC 4~GRAPHIC 7 のビットマップモードと TEXT 2 モードで、2 ページの画面を交互に表示させる(ブリンクさせる)ためのレジスタです。このレジスタにデータをセットし、表示ページを奇数ページにセットするとブリンクを開始します。表 4.3 を参照して下さい。

ON3~ON0 偶数ページの表示時間 (ビットマップモード時)

R # 7 の色の表示時間 (TEXT 2 モード時)

OF3~OF0 奇数ページの表示時間 (ビットマップモード時)

R # 12 の色の表示時間 (TEXT 2 モード時)

Color burst register 1

	7	6	5	4	3	2	1	0
R # 20	0	0	0	0	0	0	0	0

Color burst register 2

	7	6	5	4	3	2	1	0
R # 21	0	0	1	1	1	0	1	1

Color burst register 3

	7	6	5	4	3	2	1	0
R # 22	0	0	0	0	0	1	0	1

カラーバーストレジスタには、それぞれ上記の値がパワーオン時にプリセットされます。この値をすべて 0 にするとコンポジットビデオ出力の色成分の信号を消すことができます。R # 20~R # 22 は MSX2 では使用しません。

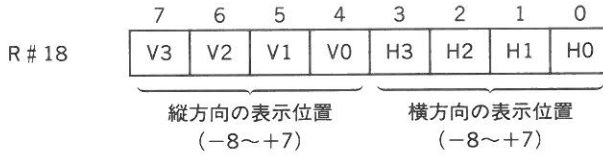
補 足

MSX のコンポジットビデオ出力は、多くの場合 V9938 の RGB ビデオ出力から外部回路により作成されます。したがって、コンポジット出力に影響のある機能は有効ではないとお考え下さい。

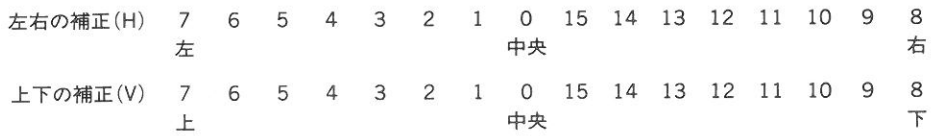
3.1.4 ディスプレイレジスタ

CRT 上の表示位置を制御するレジスタ群です。

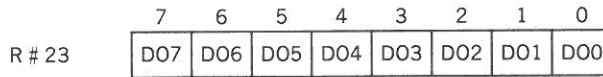
Display adjust register



CRT 上の表示位置を補正するためのレジスタです。



Display offset register



表示開始ラインをセットするためのレジスタです。このレジスタの値を変えることによって画面の縦スクロールを行うことができます。ただし、スクロールは256ライン単位で行われ、スプライトテーブルなどは別のページに置かなければなりません。

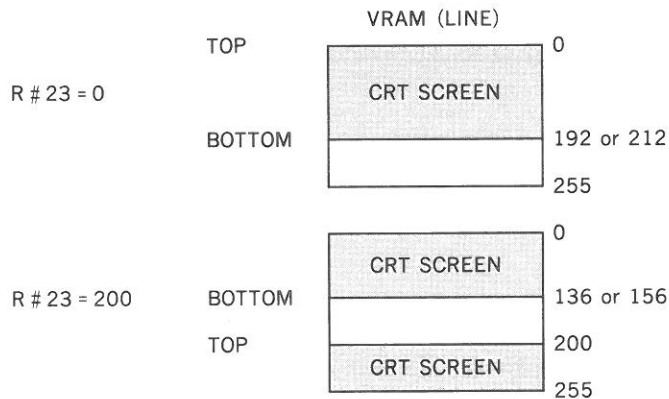


図 4.1 ディスプレイオフセットレジスタの設定例

Interrupt line register

	7	6	5	4	3	2	1	0
R # 19	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0

V9938 では、CRT が特定の走査線の表示を終えたときに割り込みを発生させることができます。このレジスタに割り込みを発生させる走査線の番号をセットし、R # 0 のビット 4 に「1」をセットします。

詳しくは、添付のフロッピーディスクの中に走査線割り込みのサンプルプログラムが入っていますので、参照して下さい。

3.1.5 アクセスレジスタ

V9938 のレジスタや VRAM をアクセスするときに使用するレジスタ群です。

VRAM Access base address register

	7	6	5	4	3	2	1	0
R # 14	0	0	0	0	0	A16	A15	A14

V9938 の VRAM をアクセスするときに、アドレスの上位 3 ビットをこのレジスタにセットします。

また、このレジスタの値は、VRAM をアクセスすると A13 からのキャリーを受けて自動的にインクリメントされます。ただし、GRAPHIC 1、GRAPHIC 2、MULTI COLOR、TEXT 1 モードではインクリメントしません。

Status register pointer

	7	6	5	4	3	2	1	0
R # 15	0	0	0	0	S3	S2	S1	S0

V9938 のステータスレジスタ (S # 0 ~ S # 9) を読み出す際、このレジスタにステータスレジスタの番号 (0 ~ 9) をセットします。

Color palette address register

	7	6	5	4	3	2	1	0
R # 16	0	0	0	0	C3	C2	C1	C0

V9938 のカラーパレットにアクセスする際、このレジスタにパレットの番号をセットします。

Control register pointer

	7	6	5	4	3	2	1	0
R # 17	All	0	R5	R4	R3	R2	R1	R0

レジスタ番号 n
(0~46)

0 オートインクリメント
1 オートインクリメント禁止

V9938 では、このレジスタの値をポインタとして他のレジスタをアクセスすることができます。また、All ビットの指定によって、内容を自動的にインクリメントさせることができます。

3.1.6 コマンドレジスタ

V9938 のコマンドを実行するときに使用するレジスタ群です。詳しくは5章を参照して下さい。

Source X low register

	7	6	5	4	3	2	1	0
R # 32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0

Source X high register

	7	6	5	4	3	2	1	0
R # 33	0	0	0	0	0	0	0	SX8

Source Y low register

	7	6	5	4	3	2	1	0
R # 34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0

Source Y high register

	7	6	5	4	3	2	1	0
R # 35	0	0	0	0	0	0	SY9	SY8

Destination X low register

	7	6	5	4	3	2	1	0
R # 36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0

Destination X high register

	7	6	5	4	3	2	1	0
R # 37	0	0	0	0	0	0	0	DX8

Destination Y low register

	7	6	5	4	3	2	1	0
R # 38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0

Destination Y high register

	7	6	5	4	3	2	1	0
R # 39	0	0	0	0	0	0	DY9	DY8

Number of dot X low register

	7	6	5	4	3	2	1	0
R # 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0

Number of dot X high register

	7	6	5	4	3	2	1	0
R # 41	0	0	0	0	0	0	0	NX8

Number of dot Y low register

	7	6	5	4	3	2	1	0
R # 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0

Number of dot Y high register

	7	6	5	4	3	2	1	0
R # 43	0	0	0	0	0	0	NY9	NY8

Color register

	7	6	5	4	3	2	1	0
R # 44	CH3	CH2	CH1	CH0	CL3	CL2	CL1	CL0

Argument register

	7	6	5	4	3	2	1	0
R # 45	0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ

Command register

	7	6	5	4	3	2	1	0
R # 46	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0

3.2 ステータスレジスタ#0～#9(Read only)

V9938 の状態を読み出すための、読み出し専用のレジスタ群です。

Status register 0

	7	6	5	4	3	2	1	0
S # 0	F	5S	C	5th sprite #				

F 垂直帰線割り込みフラグ

S # 0 を読み出すとリセットされる

5S 第5スプライトフラグ

1 水平線上にスプライトが5個 (GRAPHIC 3～GRAPHIC 7モードでは9個)

並ぶとセットされる

C 衝突フラグ

スプライトが衝突するとセットされる

5th sprite # 第5(第9)スプライトの番号がセットされる

Status register 1

	7	6	5	4	3	2	1	0
S # 1	FL	LPS	ID #				FH	

- FL ライトペンスイッチ（ライトペンフラグがセットされているとき）
 ライトペンが光を検出するとセットされる。このとき、IE2 がセットされていると割り込みを発生する。S # 1 を読み出すとリセットされる。
 マウススイッチ 2（マウスフラグがセットされているとき）
 マウスのスイッチ 2 が押されたらセットされる。S # 1 を読み出してもリセットされない。
- LPS ライトペンスイッチ（ライトペンフラグがセットされているとき）
 ライトペンのスイッチが押されるとセットされる。S # 1 を読み出してもリセットされない。
 マウススイッチ 1（マウスフラグがセットされているとき）
 マウスのスイッチ 1 が押されたらセットされる。S # 1 を読み出してもリセットされない。
- ID # V9938 の ID 番号
- FH 水平帰線割り込みフラグ
 水平帰線（R # 19 で指定）による割り込み（フラグ IE1 がセットされているとき）が発生するとセットされる。S # 1 を読み出すとリセットされる。

FL と LPS はライトペン用のレジスタです。MSX ではこのインターフェイスは使用していないので、ビット 6、7 は意味を持ちません。

Status register 2

	7	6	5	4	3	2	1	0
S # 2	TR	VR	HR	BD	1	1	EO	CE

- TR 転送レディフラグ
 CPU to VRAM、VRAM to CPU などのコマンドを実行するときは、CPU はこのフラグを見ながらデータを読み書き（転送）する。1 のとき転送可。
- VR 垂直帰線期間フラグ
 垂直帰線期間中は 1 になる。
- HR 水平帰線期間フラグ
 水平帰線期間中は 1 になる。

- BD 境界色発見フラグ
 サーチコマンドの実行で、境界色または非境界色を発見したら1になる。
- EO 表示フィールドフラグ
 0=第1フィールド、1=第2フィールド
- CE コマンド実行フラグ
 コマンドを実行中は1になる。

Column register low

	7	6	5	4	3	2	1	0
S#3	X7	X6	X5	X4	X3	X2	X1	X0

Column register high

	7	6	5	4	3	2	1	0
S#4	1	1	1	1	1	1	1	X8

Row register low

	7	6	5	4	3	2	1	0
S#5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Row register high

	7	6	5	4	3	2	1	0
S#6	1	1	1	1	1	1	EO	Y8

これらのレジスタには、スプライトの衝突座標などがセットされます。詳しくは、「6.2.8 スプライトの衝突」をご参照下さい。

Color register

	7	6	5	4	3	2	1	0
S#7	C7	C6	C5	C4	C3	C2	C1	C0

POINT、VRAM to CPU などのコマンドを実行すると、VRAM のデータがこのレジスタにセットされます。

Border X register low

	7	6	5	4	3	2	1	0
S#8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0

Border X register high

	7	6	5	4	3	2	1	0
S#9	1	1	1	1	1	1	1	BX8

サーチコマンドで発見した境界色または非境界色の X 座標が、このレジスタにセットされます。

4章

V9938 の画面モード

この章では、V9938 の各画面モードのレジスタの設定方法などについて説明します。

4.1 TEXT 1 (SCREEN 0、40 字モード)

4.1.1 特徴

- | | |
|----------------------|---------------------|
| ■ 1 パターンのサイズ | 横 6 ドット 縦 8 ドット |
| ■ パターンの数 | 256 種類 |
| ■ 画面上のパターン数 | 横 40 パターン 縦 24 パターン |
| ■ パターンの色 | 512 色中 2 色 (全画面) |
| ■ 1 画面表示に必要な VRAM 容量 | 4K バイト |

4.1.2 関係するレジスタと VRAM の領域

- | | |
|--------------------|---------------------|
| ■ パターンのフォント | VRAM パターンジェネレータテーブル |
| ■ 画面上のパターンの位置 | VRAM パターンネームテーブル |
| ■ パターンの 1 の部分の色コード | R # 7 上位 4 ビット |
| ■ パターンの 0 の部分の色コード | R # 7 下位 4 ビット |
| ■ バックドロップの色コード | R # 7 下位 4 ビット |

4.1.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	1	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を TEXT 1 モード (00001) にセットした例です。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00000000、01110000、00001000、00001000 の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルは、パターン(文字)のフォントを記憶させるエリアです。

パターンには#0~#255の番号がつけられ、パターンを画面に表示するときには、このパターン番号で指定します。

パターンジェネレータテーブルの先頭アドレスはレジスタ R#4 にセットします。指定できるのは先頭アドレスの上位6ビット (A16~A11) のみで、下位11ビット (A10~A0) は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは 00000H から 2K バイト単位の位置になります。

1個のパターンのフォントは8バイトで構成され、各バイトの下位2ビットは表示されません。

Pattern generator table base address register

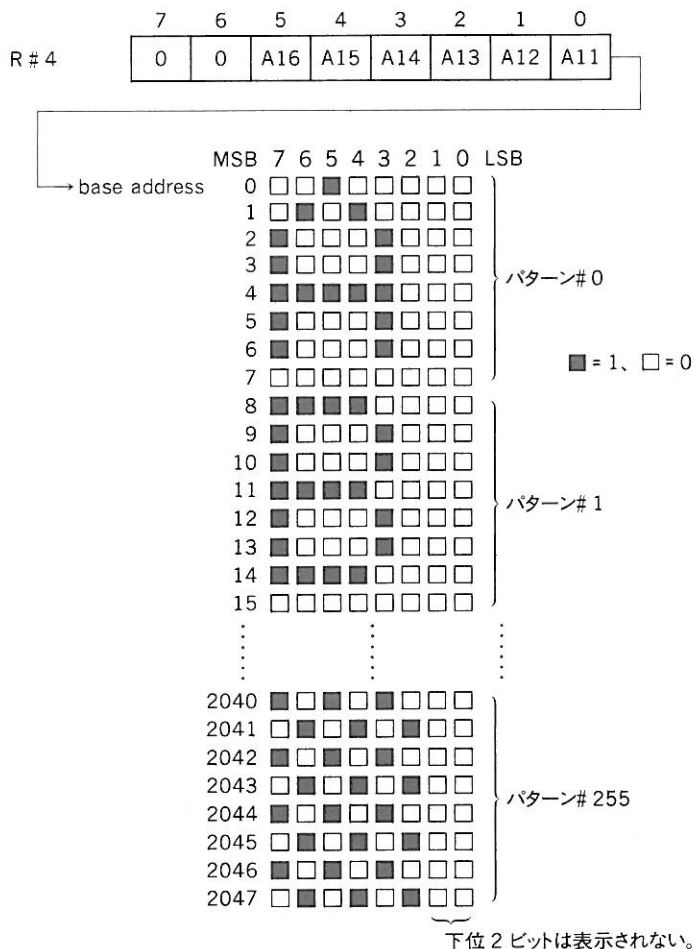


図 4.2 TEXT 1 モードのパターンジェネレータテーブル例

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0~255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは先頭アドレスの上位6ビット (A16~A11) のみで、下位11ビット (A10~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから1Kバイト単位の位置になります。

Pattern name table base address register

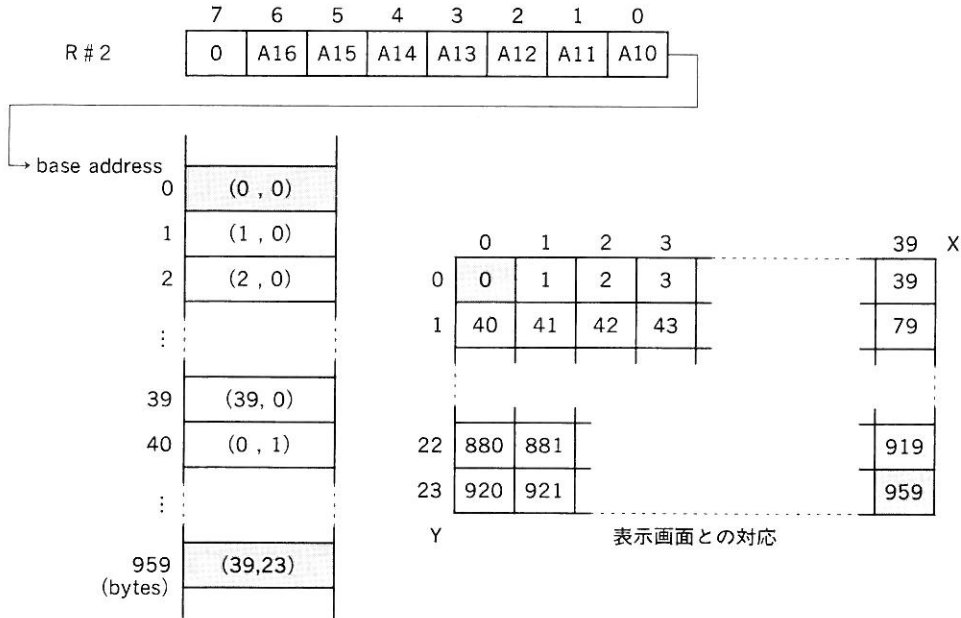
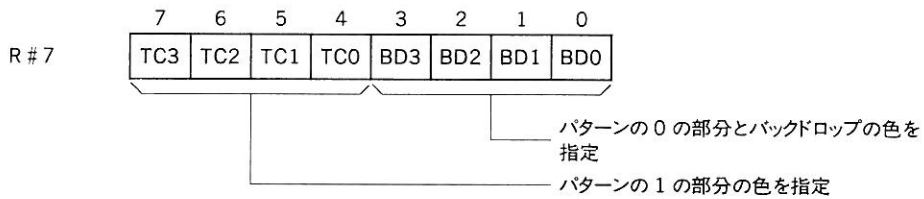


図 4.3 TEXT 1 モードのパターンネームテーブル

4. カラーレジスタの設定

画面の色は R#7 で指定します。背景色は R#7 の下位 4 ビット、前景色は R#7 の上位 4 ビットで指定されたパレットによって色が決まります。フォントパターン中の、「0」の部分は背景色、「1」の部分は前景色で表示されます。なお、TEXT1 モードでは画面のバックドロップの色を設定することはできず、背景色と同じになります。

Text color / Back drop color register



4.1.4 TEXT 1 (SCREEN 0、40 字)モードの VRAM マップ

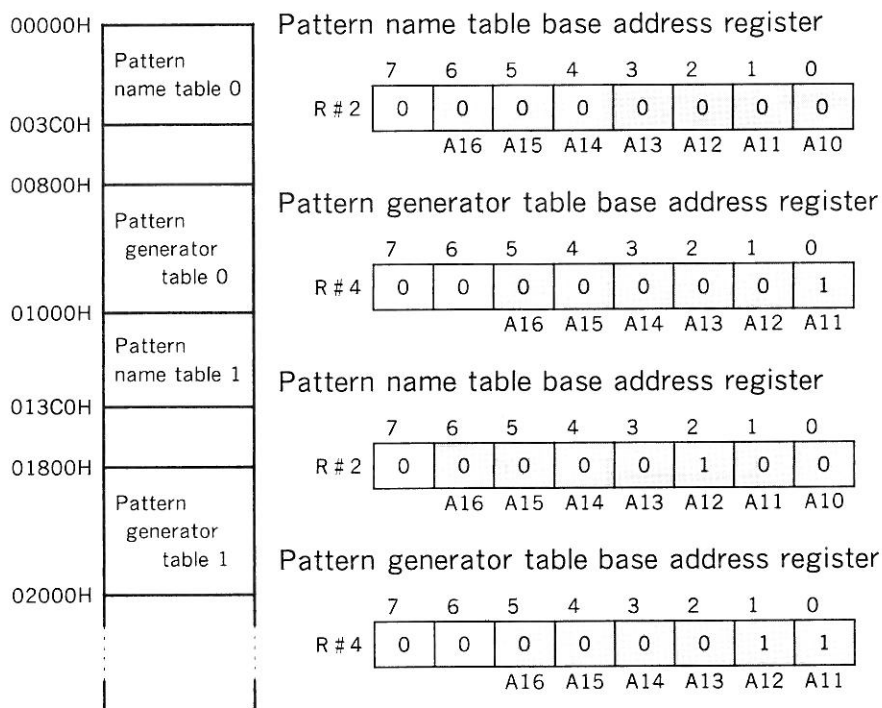


図 4.4 TEXT 1 モードの VRAM マップ

以下同様に、最高 32 ページまで割り付けが可能 (VRAM 128K 実装機種) です。ただし MSX-BASIC ではこのモードはページ 0 のみサポートしています。

パレット情報はパレットレジスタが書き込み専用のため、MSX の BIOS が管理し、VRAM の一部を使って記憶しています。この画面モードでは、0400H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

color 0 の情報は

	7	6	5	4	3	2	1	0
0400H	0	R	R	R	0	B	B	B
0401H	0	0	0	0	0	G	G	G
⋮								

のように連続 2 バイトで、RGB 各 3bit が格納されています。この情報は、パレット制御用の color 命令で参照されます。

4.2 TEXT 2 (SCREEN 0、80 字モード)

4.2.1 特徴

- | | |
|----------------------|--|
| ■ 1 パターンのサイズ | 横 6 ドット 縦 8 ドット |
| ■ パターンの数 | 256 種類 |
| ■ 画面上のパターン数 | 横 80 パターン 縦 24 パターン
横 80 パターン 縦 26.5 パターン |
| ■ パターンのブリンク | 1 文字ごとに指定可能 |
| ■ パターンの色 | 512 色中 2 色(全画面)
ブリンクを使用すると 4 色 |
| ■ 1 画面表示に必要な VRAM 容量 | 8K バイト |

4.2.2 関係するレジスタと VRAM の領域

- | | |
|--------------------|---------------------|
| ■ パターンのフォント | VRAM パターンジェネレータテーブル |
| ■ 画面上のパターンの位置 | VRAM パターンネームテーブル |
| ■ ブリンクの属性 | VRAM カラーテーブル |
| ■ パターンの 1 の部分の色コード | R # 7 上位 4 ビット |
| ■ パターンの 0 の部分の色コード | R # 7 下位 4 ビット |
| ■ バックドロップの色コード | R # 7 下位 4 ビット |
| ■ パターンの 1 の部分の色コード | R # 12 上位 4 ビット |
| ■ パターンの 0 の部分の色コード | R # 12 下位 4 ビット |
- ┌───┐ ─── ブリンク用

4.2.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	1	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	1	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を TEXT 2 モード (01001) にセットした例です。この表示モードでは、LN = 1 のとき 26.5 行、LN = 0 のとき 24 行になります。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00001000、01110000、00001000、00001000 の値で初期化します。

2. パターンジェネレータテーブルの設定

TEXT 1 モードと同じです。TEXT 1 モードのパターンジェネレータテーブルの項を参照して下さい。

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0~255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

画面上に表示されるパターンは LN = 0 のとき横 80、縦 24 パターンで、LN = 1 のときは横 80、縦 26.5 パターンです。縦の 27 個目のパターンは上半分のみ表示されます。ただし、BASIC では LN = 1 のモードはサポートしていません。

パターン名前テーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、パターン名前テーブルの先頭アドレスの上位 5 ビット (A16~A12) のみで、下位 12 ビット (A11~A0) は「0」とみなされます。したがって、パターン名前テーブルの先頭アドレスとして指定できるのは 00000H から 4K バイト単位の位置になります。

Pattern name table base address register

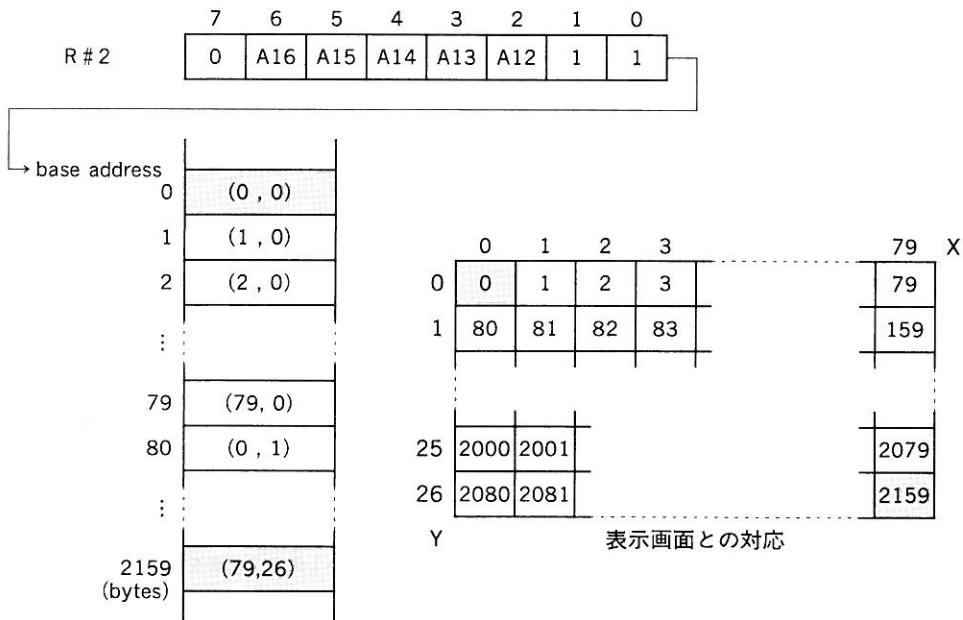


図 4.5 TEXT 2 モードのパターン名前テーブル

4. カラーテーブルの設定

TEXT 2 モードでは、各パターン (文字) に 1 ビットずつアトリビュートエリアがあり、このビットを 1 にすることによって、そのパターンにブリンクの属性を持たせることができます。

カラーテーブルの先頭アドレスはレジスタ R#3 と R#10 にセットします。カラーテーブルの先頭アドレスとして指定できるのは 00000H から 512 バイト単位の位置になります。

Color table base address register

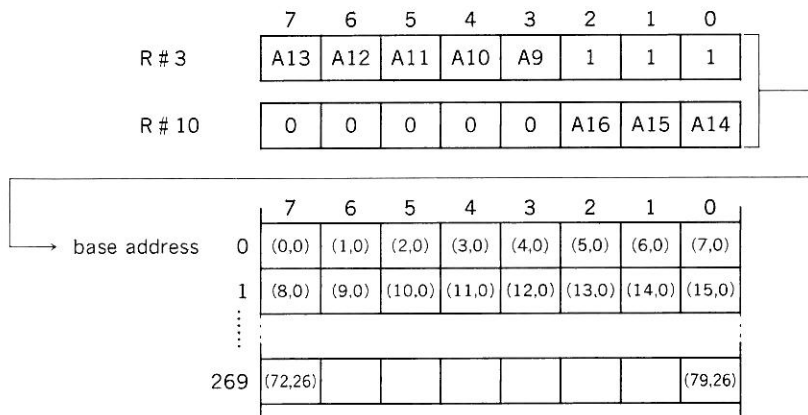
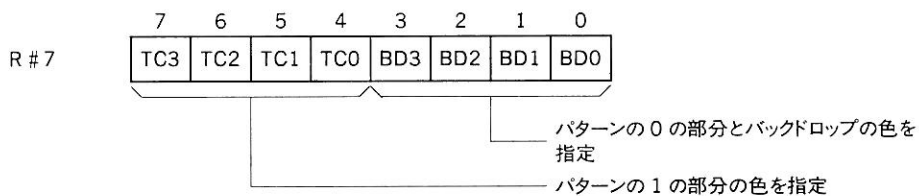


図 4.6 TEXT 2 モードのカラーテーブル

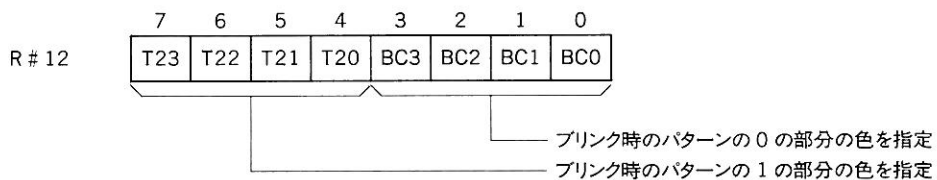
5. カラーレジスタの設定

ブリンクの属性を与えられたパターンは、レジスタ R # 7 とレジスタ R # 12 で指定された色コードを交互に表示します。

Text color / Back drop color register



Text color / Back color register



6. ブリンクレジスタの設定

ブリンクの属性を与えられたパターンは、レジスタ R # 7 とレジスタ R # 12 で指定された色コードを交互に表示しますが、この時間の間隔はレジスタ R # 13 で指定します。R # 13 の上位 4 ビットは本来の文字色が表示されている時間を、下位 4 ビットはブリンク色が表示されている時間を表 4.3 のように設定します。詳しくは表 4.3 を参照して下さい。

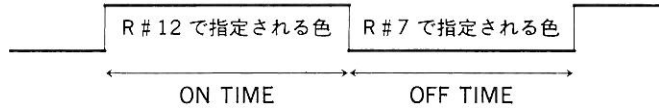


図 4.7 ブリンクの周期

Blinking period register



表 4.3 ブリンクのデータと時間一覧

ON TIME と OFF TIME のデータと時間の関係は、このようになります。このデータは、NTSC モードにおいて計算されたものです。

DATA (BIN)	TIME (ms)
0 0 0 0	0
0 0 0 1	166.9
0 0 1 0	333.8
0 0 1 1	500.6
0 1 0 0	667.5
0 1 0 1	834.4
0 1 1 0	1001.3
0 1 1 1	1168.2
1 0 0 0	1335.1
1 0 0 1	1501.9
1 0 1 0	1668.8
1 0 1 1	1835.7
1 1 0 0	2002.6
1 1 0 1	2169.5
1 1 1 0	2336.3
1 1 1 1	2503.2

4.2.4 TEXT 2 (SCREEN 0、80 字)モードの VRAM マップ

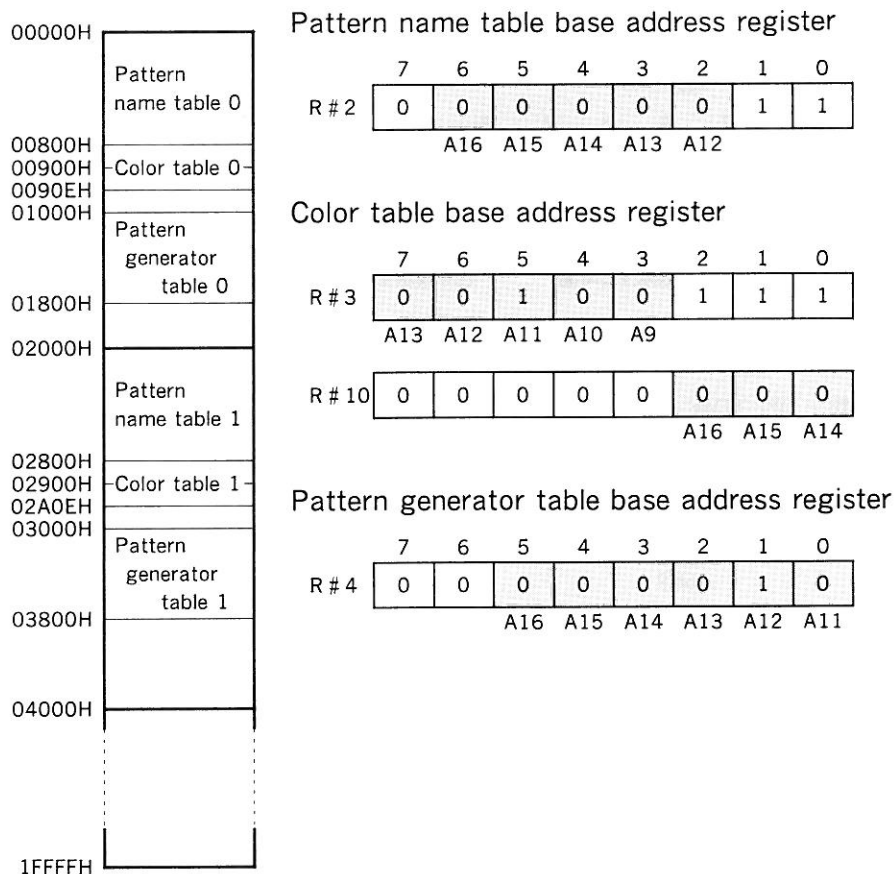


図 4.8 TEXT 2 モードの VRAM マップ

以下同様に、最高 16 ページまで割り付けが可能 (VRAM 128K 実装機種) です。ただし、BASIC ではこのモードはページ 0 のみサポートし、カラーテーブル、プリンク機能はサポートしていません。

この画面モードでは、0F00H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.3 MULTI COLOR (SCREEN 3)

4.3.1 特徴

■ 1 ブロックのサイズ	横 4 ドット 縦 4 ドット
■ 画面上のブロック数	横 64 ブロック 縦 48 ドット
■ ブロックの色	512 色中 16 色 (各ブロックごとに指定可)
■ スプライト	モード 1
■ 1 画面表示に必要な VRAM 容量	4K バイト

MULTI COLOR モードは他のグラフィックモードとは異なり、1 画面が横 64 個、縦 48 個のカラーブロックで構成されます。各カラーブロックは、ブロックごとに 512 色中から 16 色を指定できます。

4.3.2 関係するレジスタと VRAM の領域

■ カラーブロックの色コード	VRAM パターンジェネレータテーブル
■ カラーブロックの位置	VRAM パターンネームテーブル
■ バックドロップの色コード	R #7 下位 4 ビット
■ スプライト	VRAM スプライトアトリビュートテーブル VRAM スプライトパターンテーブル

4.3.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R #0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R # 1	0	BL	IE0	0	1	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R # 8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

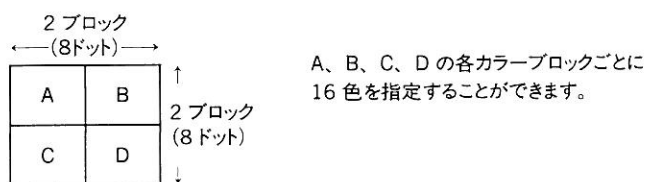
	7	6	5	4	3	2	1	0
R # 9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を MULTI COLOR モード (00010) にセットした例です。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00000000、01101000、00001000、00001000 の値で初期化します。

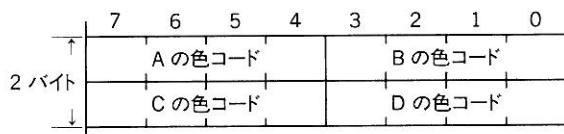
2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルはカラーブロックの色を記憶させるエリアです。

1 個のパターンは 4 個のカラーブロックで構成されます。このパターンは画面の表示領域のドット数を 256×192 とした場合、8×8 に相当する大きさです。

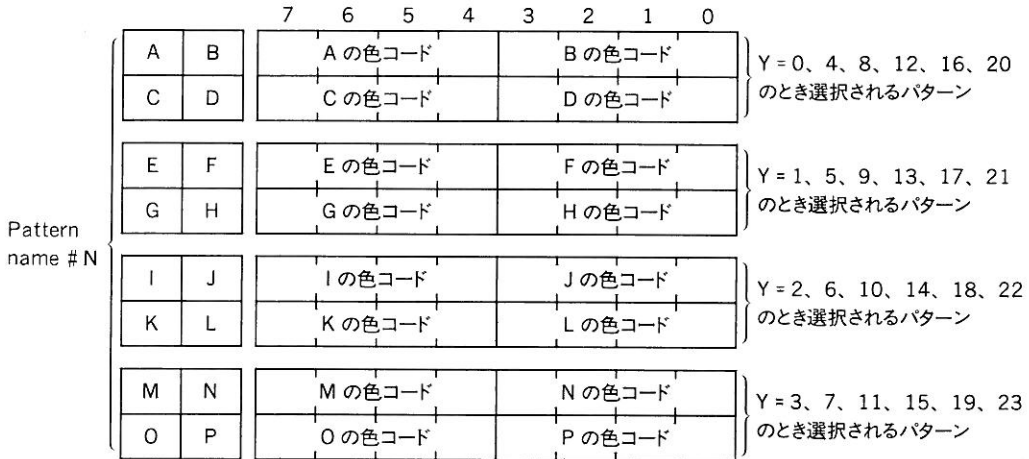


1 個のパターンに含まれるカラーブロックの色は下図のように 2 バイトで表現します。



1 個のパターン名に対して、表示の Y 座標によって自動的に選択される 4 個のパターンが対応します。

パターンジェネレータテーブルの先頭アドレスはレジスタ R#4 にセットします。指定できるのは、先頭アドレスの上位6ビット (A16~A11) のみで、下位11ビット (A10~A0) は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは 00000H から 2K バイト単位の位置になります。



Pattern generator table base address register

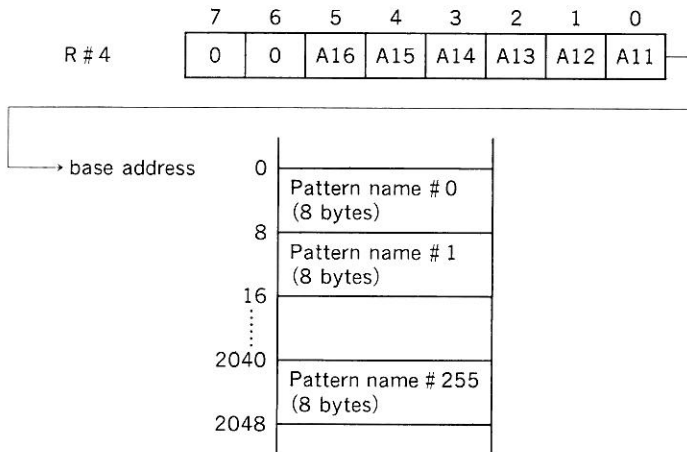


図 4.9 MULTI COLOR モードのパターンジェネレータテーブル

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0~255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、パターンネームテーブルの先頭アドレスの上位7ビット (A16~A10) のみで、下位10ビット (A9~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H から 1K バイト単位の位置になります。

Pattern name table base address register

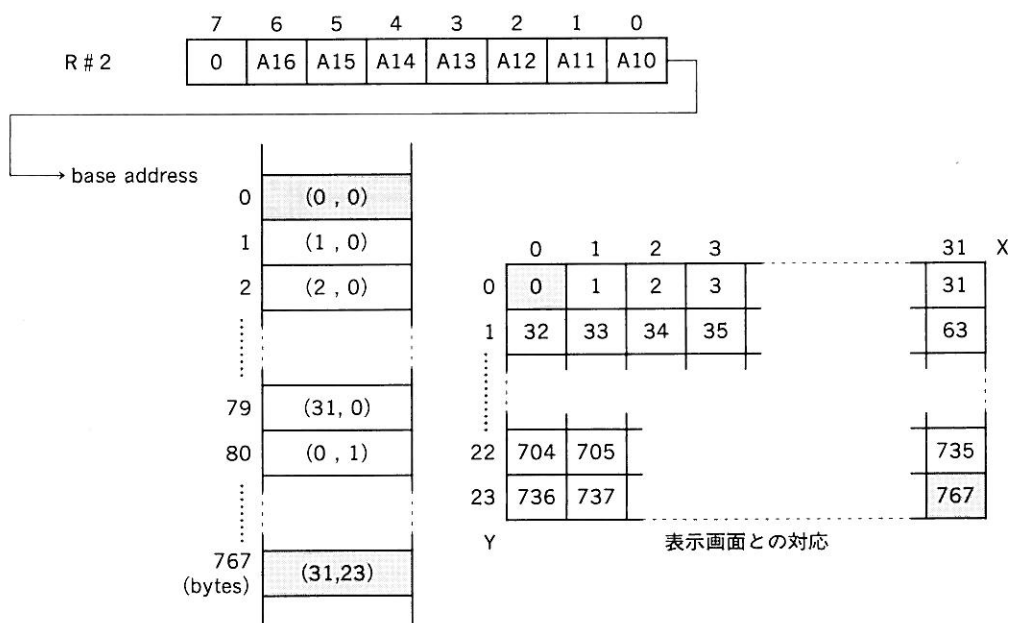
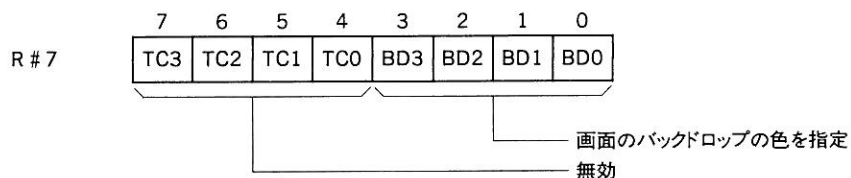


図 4.10 MULTI COLOR モードのパターンネームテーブル

4. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



5. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.1 スプライトモード 1」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.3.4 MULTI COLOR(SCREEN 3)モードの VRAM マップ

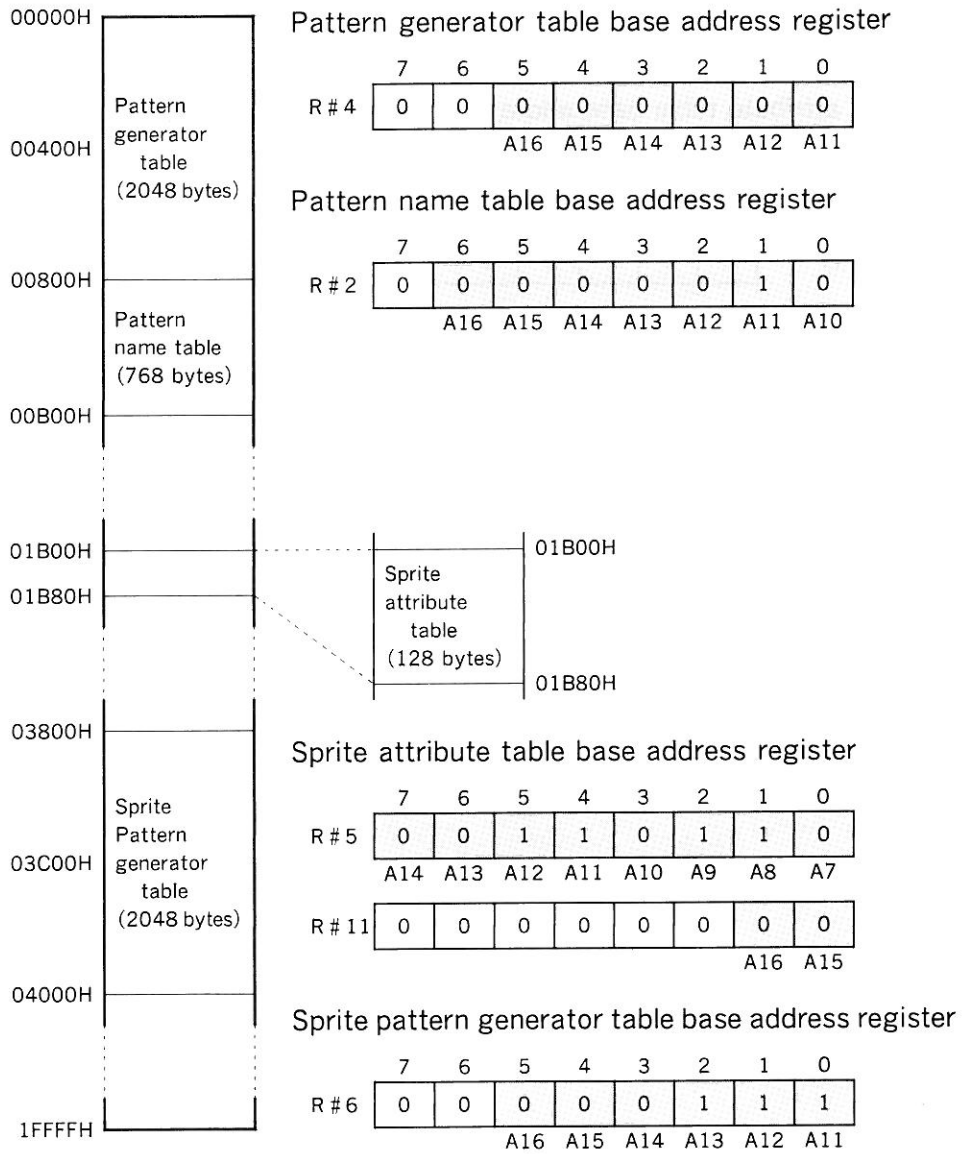


図 4.11 MULTI COLOR モードの VRAM マップ

以下同様に 8 ページまで割り付けが可能 (VRAM 128K 実装機種) です。配置によっては 16 ページ、スプライトジェネレータの節約で 32 ページ、64 ページなどの割り付けも可能ですが、BASIC ではサポートしていません。

この画面モードでは、02020H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.4 GRAPHIC 1 (SCREEN 1)

4.4.1 特徴

- | | |
|----------------------|---------------------|
| ■ 1 パターンのサイズ | 横 8 ドット 縦 8 ドット |
| ■ パターンの数 | 256 種類 |
| ■ 画面上のパターン数 | 横 32 パターン 縦 24 パターン |
| ■ パターンの色 | 512 色中 16 色(全画面) |
| ■ スプライト | モード 1 |
| ■ 1 画面表示に必要な VRAM 容量 | 4K バイト |

4.4.2 関係するレジスタと VRAM の領域

- | | |
|--------------------|---|
| ■ パターンのフォント | VRAM パターンジェネレータテーブル |
| ■ 画面上のパターンの位置 | VRAM パターンネームテーブル |
| ■ パターンの 1 の部分の色コード | VRAM カラーテーブル
8 パターンごとに 1 組指定可能 |
| ■ パターンの 0 の部分の色コード | |
| ■ バックドロップの色コード | R #7 下位 4 ビット |
| ■ スプライト | VRAM スプライトアトリビュートテーブル
VRAM スプライトパターンテーブル |

4.4.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を GRAPHIC 1 モード (00000) にセットした例です。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00000000、01100000、00001000、00001000 の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルはパターンのフォントを記憶させるエリアです。

パターンには#0～#255の番号がつけられ、パターンを画面に表示するときには、このパターン番号で指定します。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは、先頭アドレスの上位6ビット(A16～A11)のみで、下位11ビット(A10～A0)は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから2Kバイト単位の位置になります。

1個のパターンのフォントは8バイトで構成されます。

Pattern generator table base address register

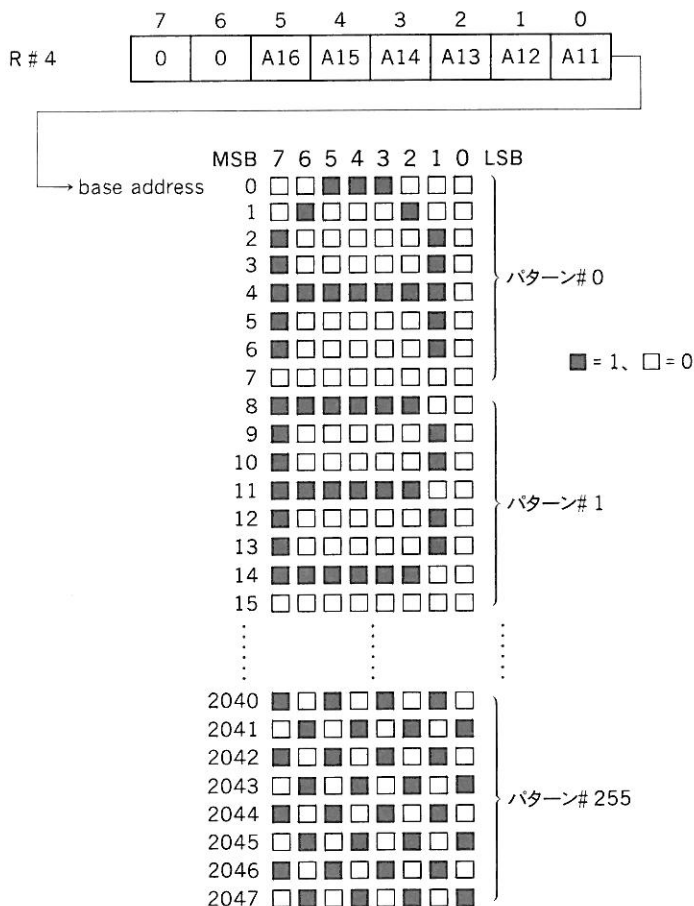


図 4.12 GRAPHIC 1 モードのパターンジェネレータテーブル例

3. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0~255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示されます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位7ビット (A16~A10) のみで、下位10ビット (A9~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから1Kバイト単位の位置になります。

Pattern name table base address register

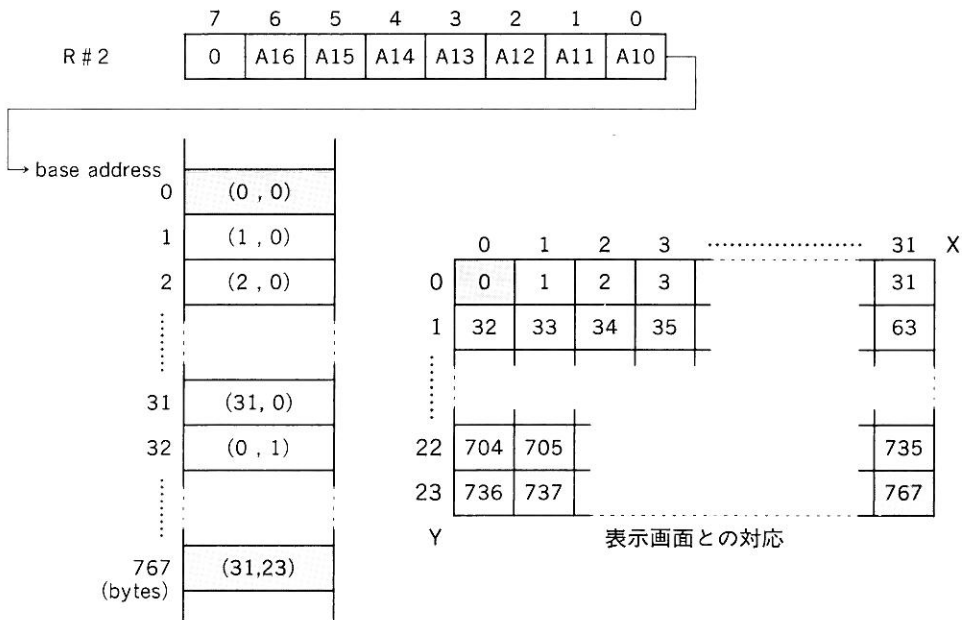


図 4.13 GRAPHIC 1 モードのパターンネームテーブル

4. カラーテーブルの設定

パターンの「1」の部分の色と「0」の部分の色を8パターンごとに1組設定します。

カラーテーブルの先頭アドレスはレジスタ R # 3 と R # 10 にセットします。カラーテーブルの先頭アドレスとして指定できるのは、00000H から 64 バイト単位の位置になります。

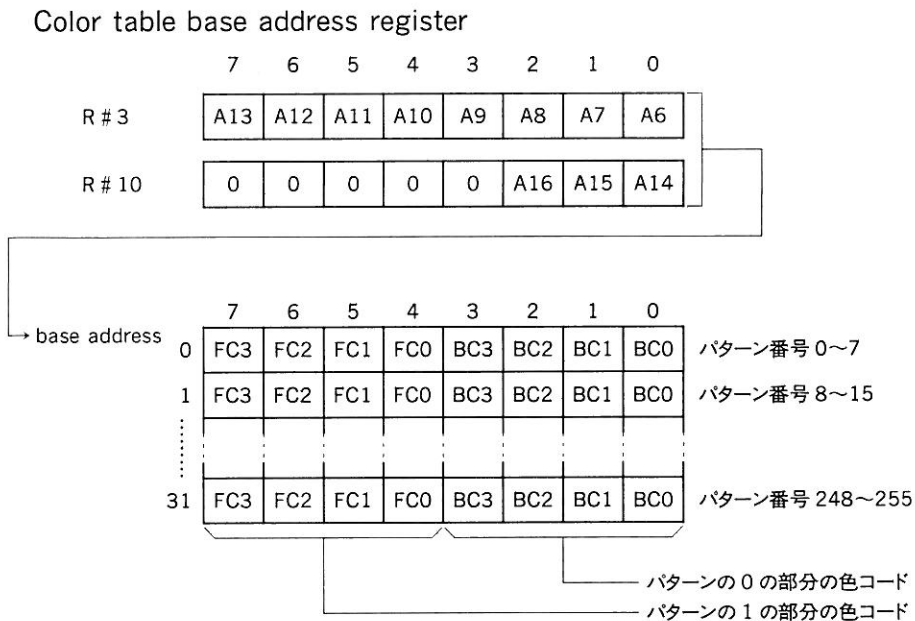
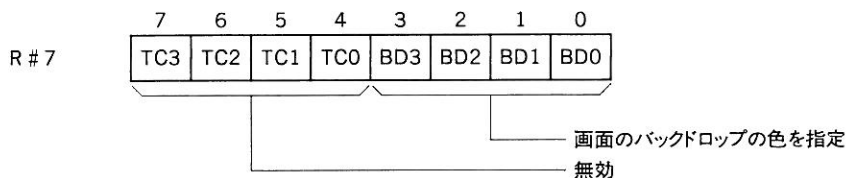


図 4.14 GRAPHIC 1 モードのカラーテーブル

5. カラーレジスタの設定

R # 7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



6. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.1 スプライトモード 1」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.4.4 GRAPHIC 1(SCREEN 1)モード VRAM マップ

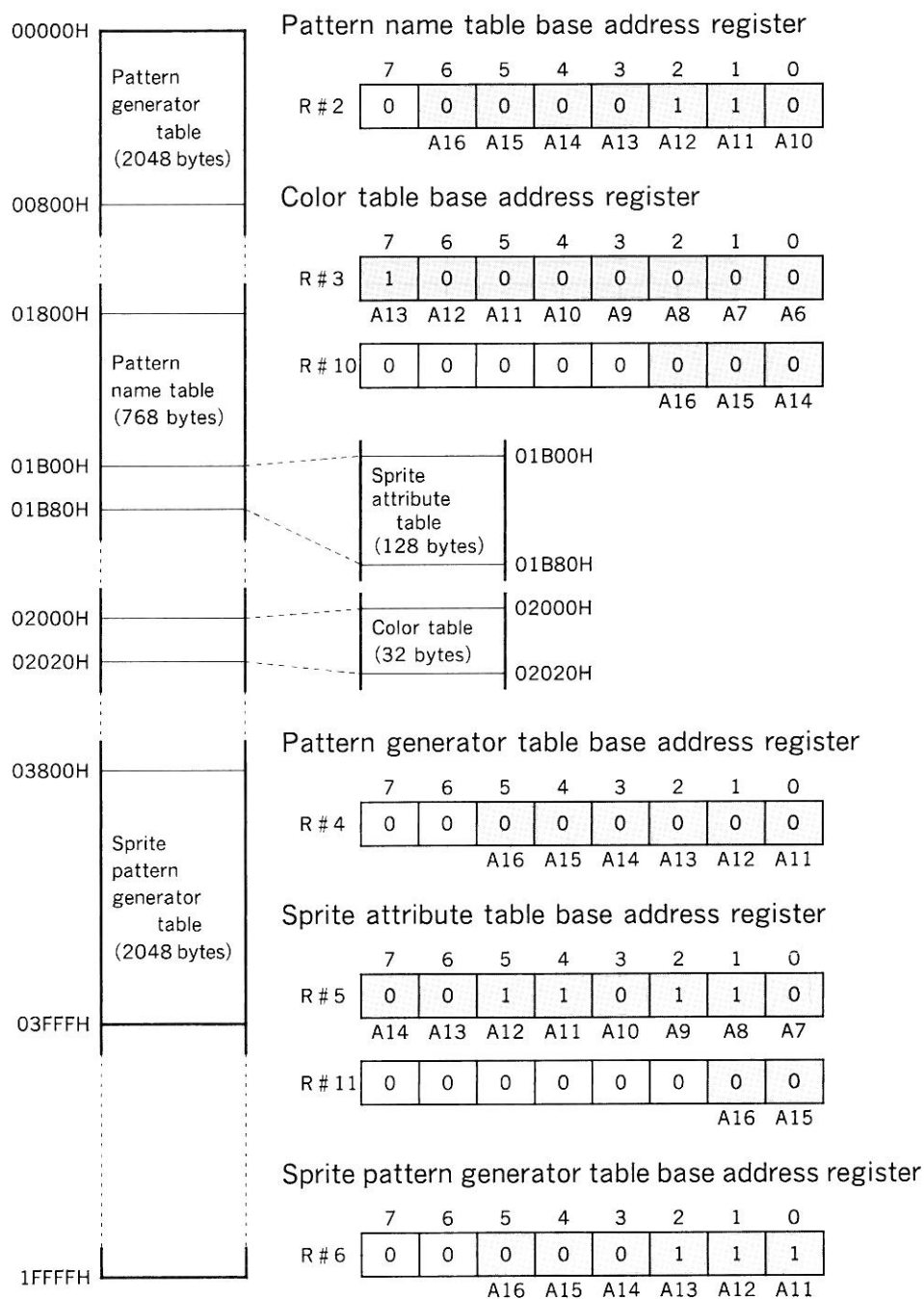


図 4.15 GRAPHIC 1 モードの VRAM マップ

以下同様に 8 ページまで割り付け可能 (VRAM 128K 実装機種) です。配置によっては 32 ページなどの割り付けも可能ですが、BASIC ではサポートしていません。

この画面モードでは、02020H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.5 GRAPHIC 2 (SCREEN 2)、 GRAPHIC 3 (SCREEN 4)

4.5.1 特徴

■ 1パターンのサイズ	横8ドット 縦8ドット
■ パターンの数	768種類
■ 画面上のパターン数	横32パターン 縦24パターン
■ パターンの色	512色中16色(全画面)
■ スプライト	モード1 (GRAPHIC 2) モード2 (GRAPHIC 3)
■ 1画面表示に必要なVRAM容量	16Kバイト

GRAPHIC 2 と GRAPHIC 3 の違いは、スプライトのモードだけです。

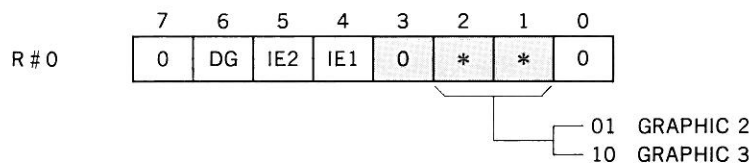
4.5.2 関係するレジスタとVRAMの領域

■ パターンのフォント	VRAM パターンジェネレータテーブル
■ 画面上のパターンの位置	VRAM パターンネームテーブル
■ パターンの1の部分の色コード	VRAM カラーテーブル 各パターンの1ラスタに1組指定可能
■ パターンの0の部分の色コード	
■ バックドロップの色コード	R#7 下位4ビット
■ スプライト	VRAM スプライトアトリビュートテーブル VRAM スプライトパターンテーブル

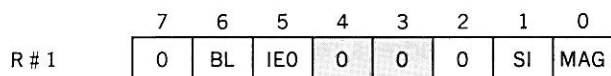
4.5.3 初期設定

1. モードレジスタの設定

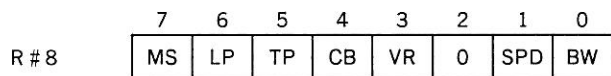
Mode register 0



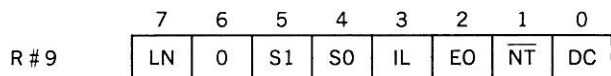
Mode register 1



Mode register 2



Mode register 3



□は表示モード設定用のビット (M5~M1) を GRAPHIC 2 (00100) または GRAPHIC 3 (01000) モードにセットした例です。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00000010 (GRAPHIC 3 のときは 00000100)、01100000、00001000、00001000 の値で初期化します。

2. パターンジェネレータテーブルの設定

パターンジェネレータテーブルはパターンを記憶させるエリアです。

パターンには#0~#255の番号(パターンネーム)がつけられています。画面を3分割して、それぞれのブロックごとに1つのパターンジェネレータテーブルを持つことにより、768種類のパターンを使用できます。これにより、画面に任意のパターンを表示することができるフルビットマップ表示の機能を実現しています。

1個のパターンは8バイトで構成されるので、合計6136バイトの領域を占めることになります。

パターンジェネレータテーブルの先頭アドレスはレジスタR#4にセットします。指定できるのは、先頭アドレスの上位4ビット(A16~A13)のみで、下位13ビット(A12~A0)は「0」とみなされます。したがって、パターンジェネレータテーブルの先頭アドレスとして指定できるのは00000Hから8Kバイト単位の位置になります。

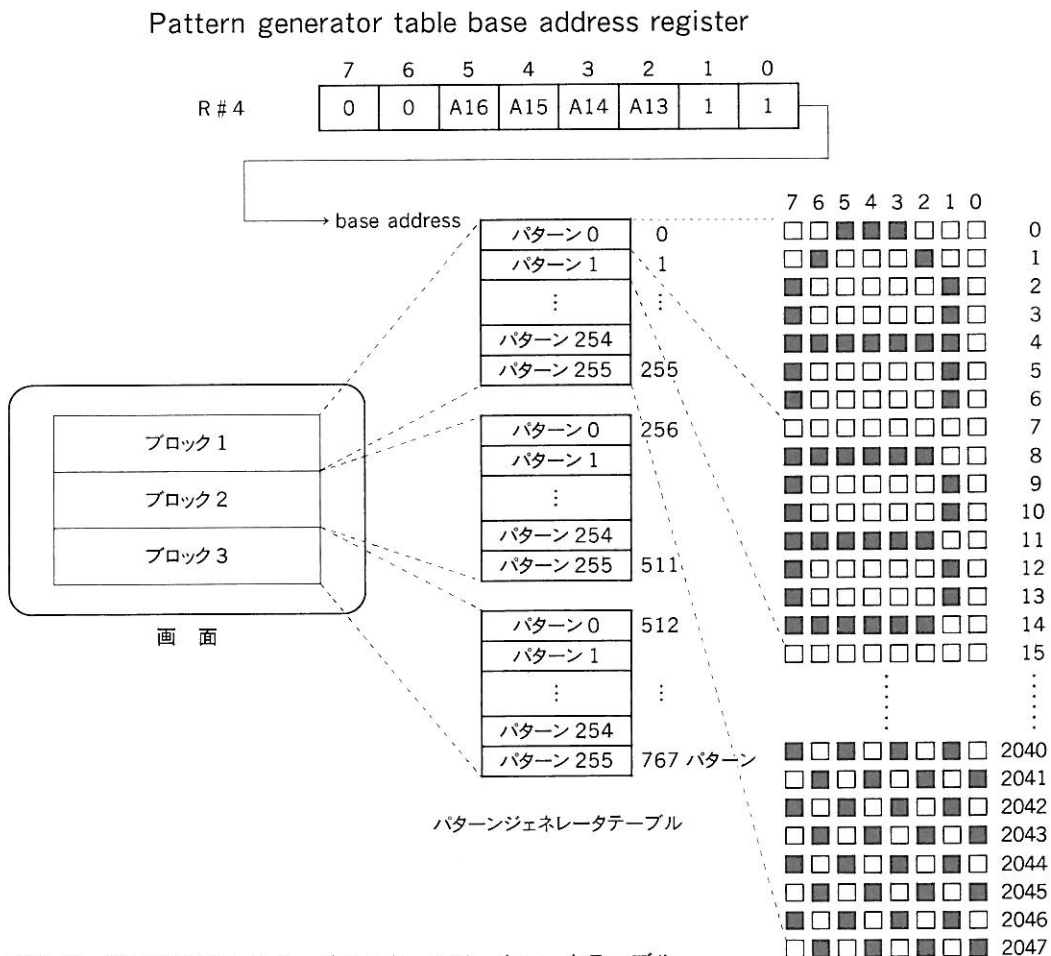


図 4.16 GRAPHIC 2、3 モードのパターンジェネレータテーブル

3. カラーテーブルの設定

カラーテーブルはパターンジェネレータテーブルと1対1に対応し、パターンの「1」の部分の色と「0」の部分の色を各パターンの1ラスタに対して1組指定します。したがって、1つのパターンに対して8バイトのデータとなり、合計でパターンジェネレータテーブルと同じ6136バイトの領域を占めます。

カラーテーブルの先頭アドレスはレジスタ R#3 と R#10 にセットします。カラーテーブルの先頭アドレスとして指定できるのは、00000H から 8K バイトの位置になります。

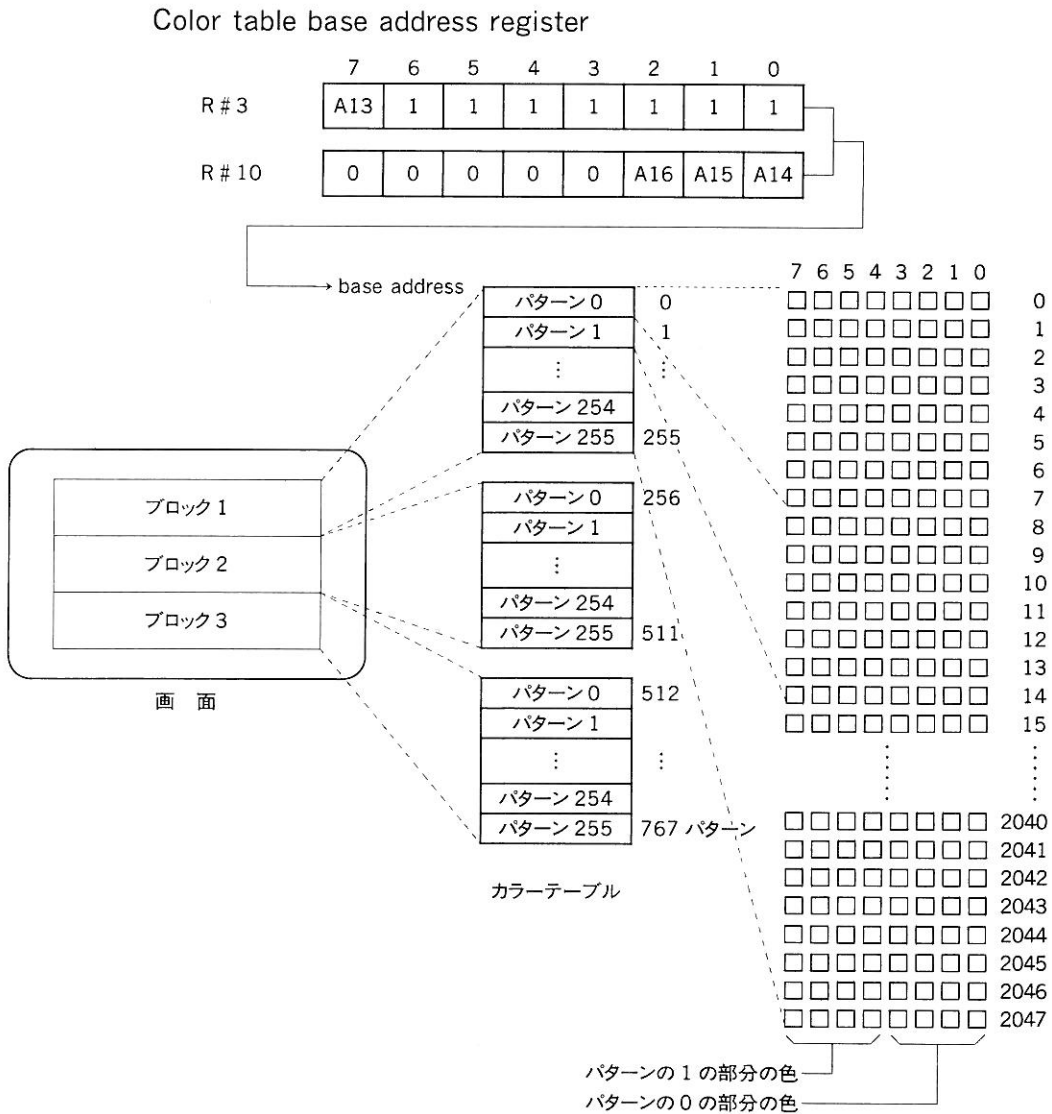
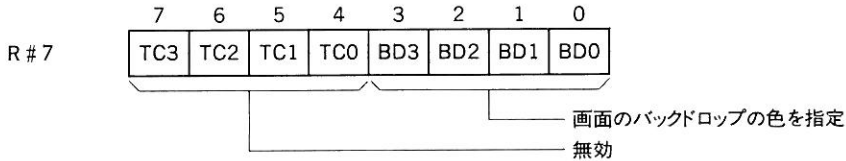


図 4.17 GRAPHIC 2、3 モードのカラーテーブル

4. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。

Text color / Back drop color register



5. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1パターンに対応しています。このテーブルに0~255のパターン番号を書き込むと、対応する画面上の位置に指定したパターンが表示され、画面を上、中、下と3分割して使うことにより、768種類のパターンを表示することができます。

パターンネームテーブルの先頭アドレスはレジスタR#2にセットします。指定できるのは、先頭アドレスの上位7ビット(A16~A10)のみで、下位10ビット(A9~A0)は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000Hから1Kバイト単位の位置になります。

Pattern name table base address register

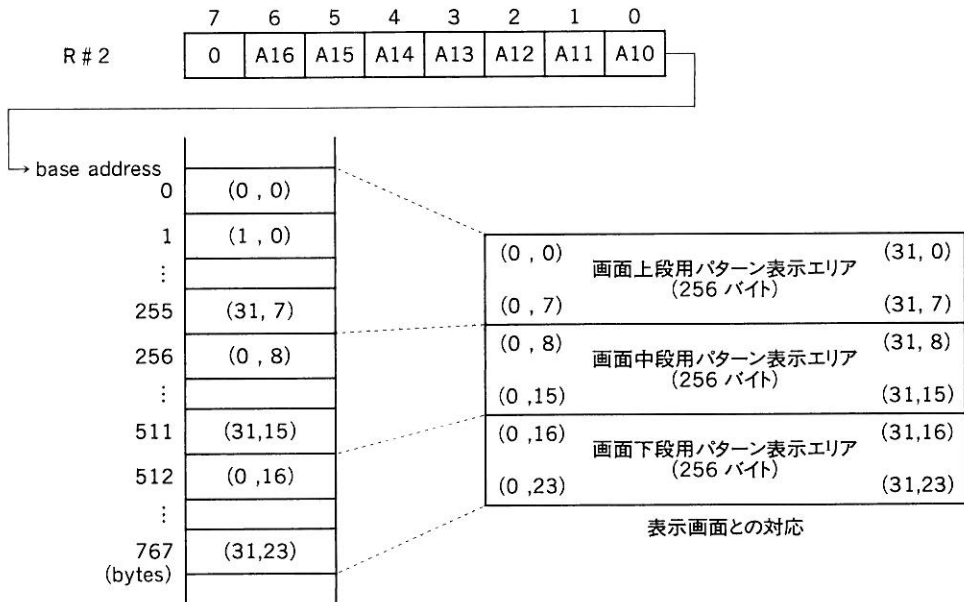


図 4.18 GRAPHIC 2、3 のパターンネームテーブル

6. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。GRAPHIC 2 モードのときは「6.1 スプライトモード 1」の項を、GRAPHIC 3 モードのときは「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.5.4 GRAPHIC 2 (SCREEN 2)モードの VRAM マップ

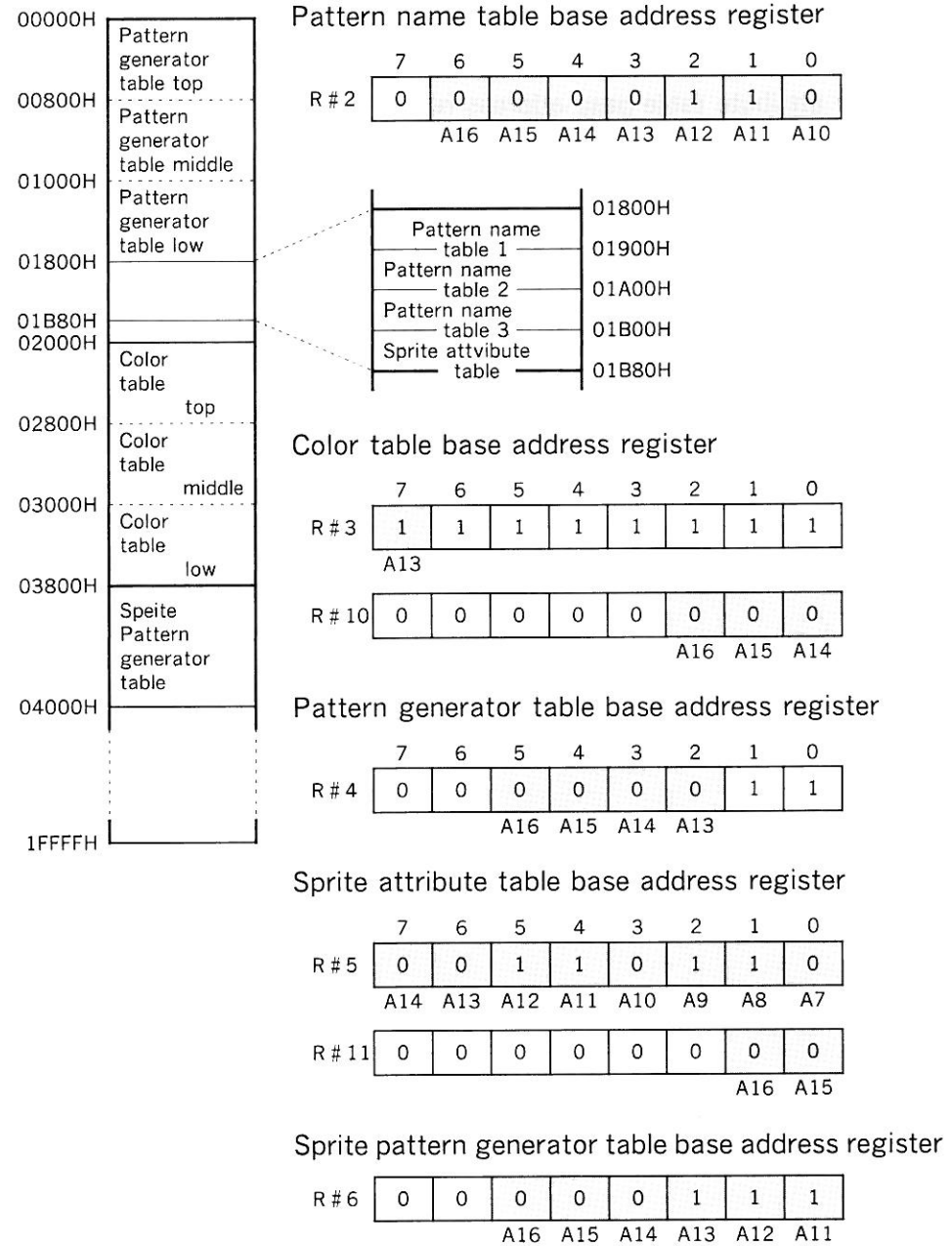


図 4.19 GRAPHIC 2 の VRAM マップ

以下同様に、最高 8 ページまで割り付け可能 (VRAM 128K 実装機種) です。しかし、BASIC ではサポートしていません。

この画面モードでは、01B80H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

表示エリアが 256×192 のとき図 4.19 のとおりですが、ハードウェアスクロールの利用によって表示エリアを 256×256 とする場合には、パターンネームテーブルは 768 バイトではなく 1024 バイト連続しており、1024 種類のパターンを用意できることとなります。また、この場合はパターンジェネレータテーブルとカラーテーブルもそれぞれ連続した 8K バイトとなり、スプライトジェネレータテーブル、パターンネームテーブルなどは別のページに移動しなければなりません。

4.5.5 GRAPHIC 3 (SCREEN 4)モードの VRAM マップ

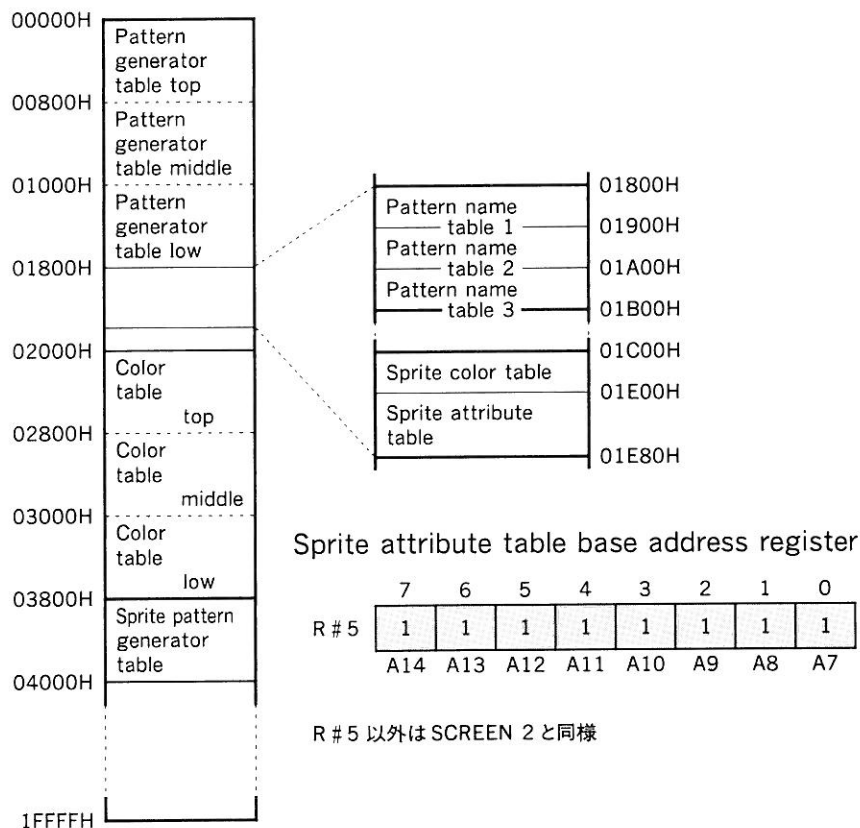


図 4.20 GRAPHIC 3 の VRAM マップ

以下同様に、最高 8 ページまで割り付けが可能 (VRAM 128K 実装機種) です。しかし、BASIC ではサポートしていません。

この画面モードでは、01B80H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.6 GRAPHIC 4 (SCREEN 5)

4.6.1 特徴

- 解像度 横 256 ドット×縦 212 ドット(または縦 192 ドット)のビットマップグラフィックモード
- 表示色 512 色中 16 色 (全画面)
- スプライト モード 2
- 1 画面表示に必要な VRAM 容量 32K バイト

4.6.2 関係するレジスタと VRAM の領域

- グラフィックス VRAM パターンネームテーブル
- バックドロップの色コード R#7 下位 4 ビット
- スプライト VRAM スプライトアトリビュートテーブル
VRAM スプライトパターンテーブル

4.6.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	0	1	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を GRAPHIC 4 (01100) モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドットに設定されます。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00000110、01100000、00001000、10001000 の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の2ドットに対応しており、1ドットごとにパレットにより選択された512色中の16色を選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、上位2ビット (A16~A15) のみで、下位15ビットは「0」とみなされます。したがって、パターンネームテーブルが設定可能なアドレスは、00000H、08000H、10000H、18000H の4箇所になります。

Pattern name table base address register

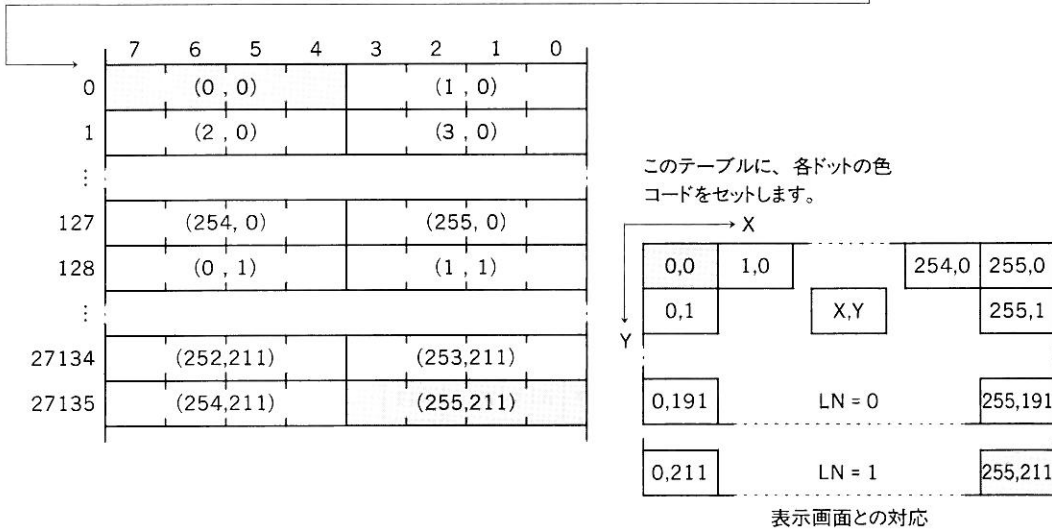
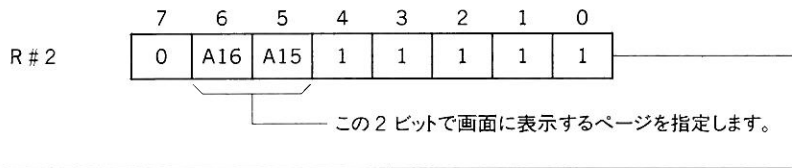
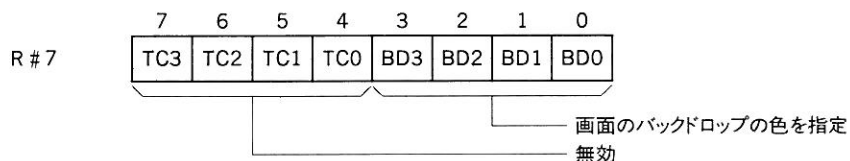


図 4.21 GRAPHIC 4 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7で画面のバックドロップの色が指定できます。

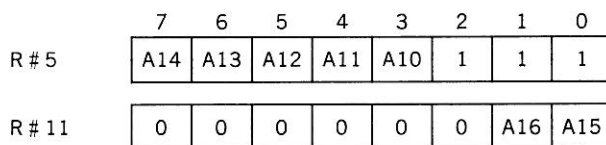
Text color / Back drop color register



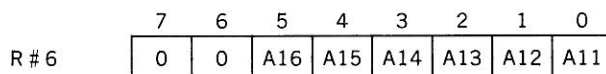
4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register



Sprite pattern generator table base address register



4.6.4 GRAPHIC 4 (SCREEN 5)モードの VRAM マップ

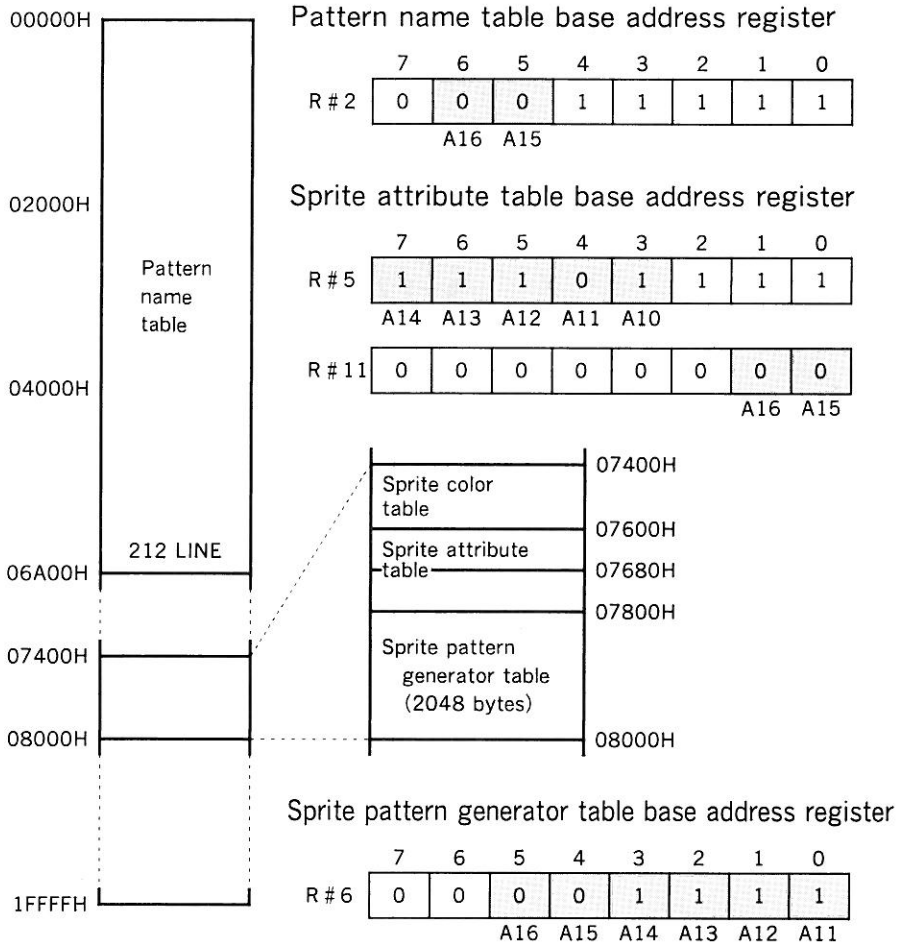


図 4.22 GRAPHIC 4 モードの VRAM マップ

以下同様に、4 ページまで割り付け可能 (VRAM 128K 実装機種) です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは、07680H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.7 GRAPHIC 5 (SCREEN 6)

4.7.1 特徴

- 解像度 横 512 ドット×縦 212 ドット(または縦 192 ドット)
のビットマップグラフィックモード
- 表示色 512 色中 4 色 (全画面)
- スプライト モード 2
- 1 画面表示に必要な VRAM 容量 32K バイト

4.7.2 関係するレジスタと VRAM の領域

- グラフィックス VRAM パターンネームテーブル
- バックドロップの色コード R#7 下位 4 ビット
- スプライト VRAM スプライトアトリビュートテーブル
VRAM スプライトパターンテーブル

4.7.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	1	0	0	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IE0	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R # 8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R # 9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5～M1) を GRAPHIC 5 (10000) モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドットに設定されます。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00001000、01100000、00001000、10001000 の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の4ドットに対応しており、1ドットごとにパレットにより選択された512色中の4色から選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位2ビット (A16~A15) のみで、下位15ビット (A14~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは00000H、08000H、10000H、18000Hの4箇所になります。

Pattern name table base address register

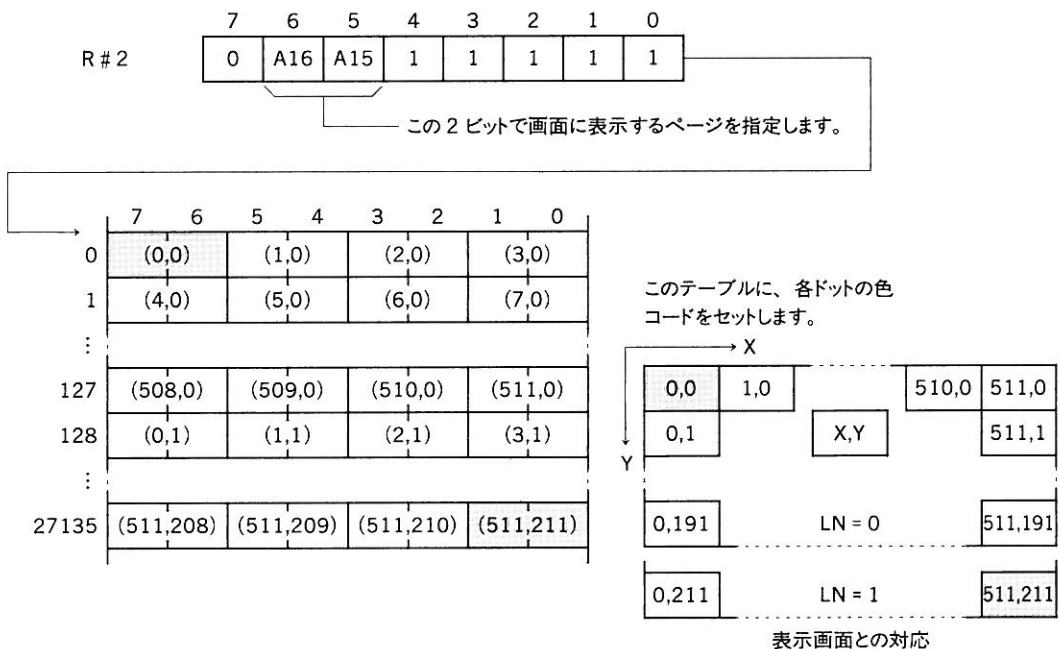
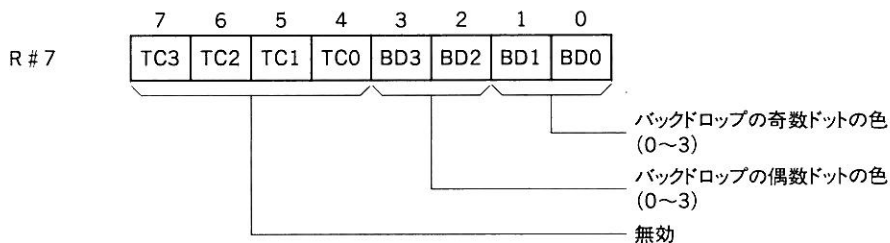


図 4.23 GRAPHIC 5 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1

R#11	0	0	0	0	0	0	A16	A15
------	---	---	---	---	---	---	-----	-----

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

5. ハードウェアタイリング機能

GRAPHIC 5 モードでは、スプライトとバックドロップの色についてハードウェアタイリングの機能が働きます。これらの色はパターンネームテーブルの場合と異なり4ビットで設定します。この4ビットの上位2ビットをX座標(0~511)の偶数ドットの色コードに、下位2ビットを奇数ドットの色コードに設定します。

GRAPHIC 5 モードでは、スプライトの1ドットの大きさはグラフィックのドットの大きさの2倍ですが、このタイリング機能によってスプライトの1ドットの左半分と右半分にそれぞれ異なる色を表示することができます。

バックドロップの色も同様に偶数ドットと奇数ドットに対して異なる色を指定できます。

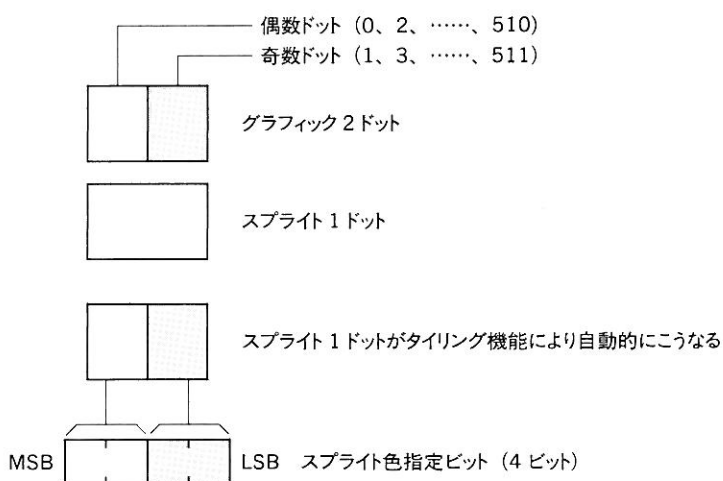


図 4.24 ハードウェアタイリングの設定

4.7.4 GRAPHIC 5 (SCREEN 6)モードの VRAM マップ

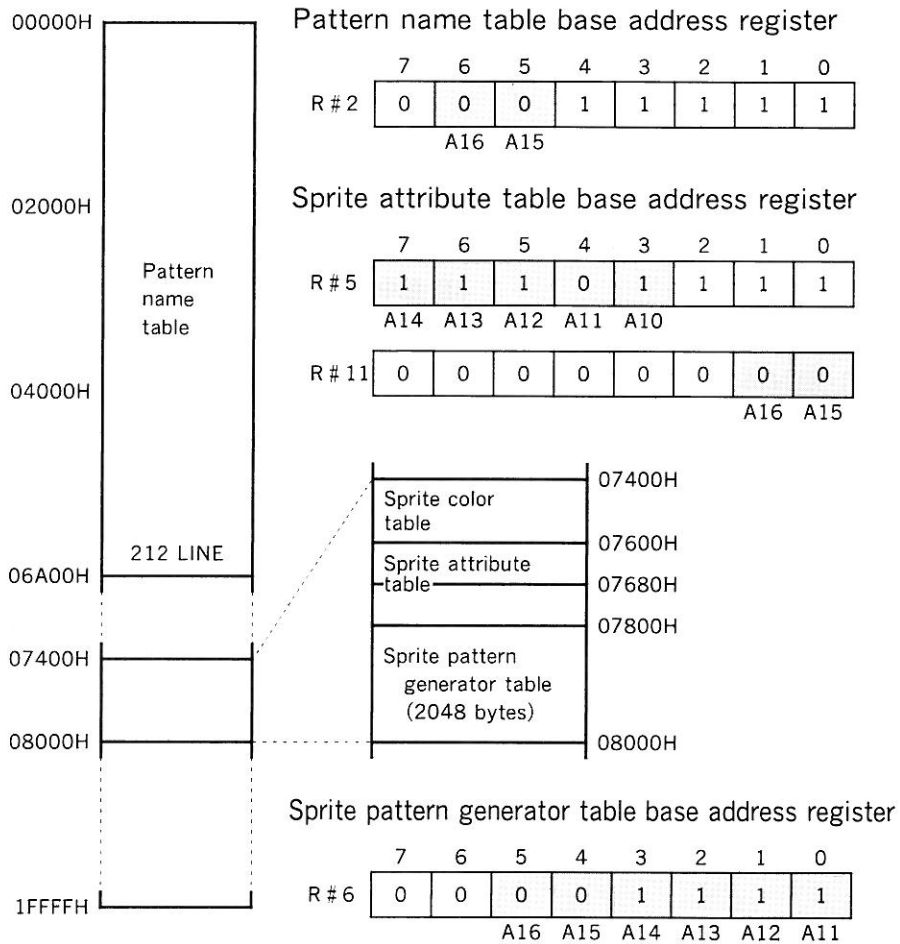


図 4.25 GRAPHIC 4 モードの VRAM マップ

以下同様に、4 ページまで割り付け可能 (VRAM 128K 実装機種) です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは、07680H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.8 GRAPHIC 6 (SCREEN 7)

4.8.1 特徴

- 解像度 横 512 ドット×縦 212 ドット(または縦 192 ドット)
のビットマップグラフィックモード
- 表示色 512 色中 16 色 (全画面)
- スプライト モード 2
- 1 画面表示に必要な VRAM 容量 128K バイト (2 画面)

このモードを使用するためには、VRAM が 128K バイト必要です。

4.8.2 関係するレジスタと VRAM の領域

- グラフィックス VRAM パターンネームテーブル
- バックドロップの色コード R #7 下位 4 ビット
- スプライト VRAM スプライトアトリビュートテーブル
VRAM スプライト パターンテーブル

4.8.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R #0	0	DG	IE2	IE1	1	0	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IEO	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を GRAPHIC 6 (10100) モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドットに設定されます。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 0001010、01100000、00001000、10001000 の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の2ドットに対応しており、1ドットごとにパレットにより選択された512色中の16色から選んで表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位1ビット (A16) のみで、下位16ビット (A15~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H と 10000H の位置になります。

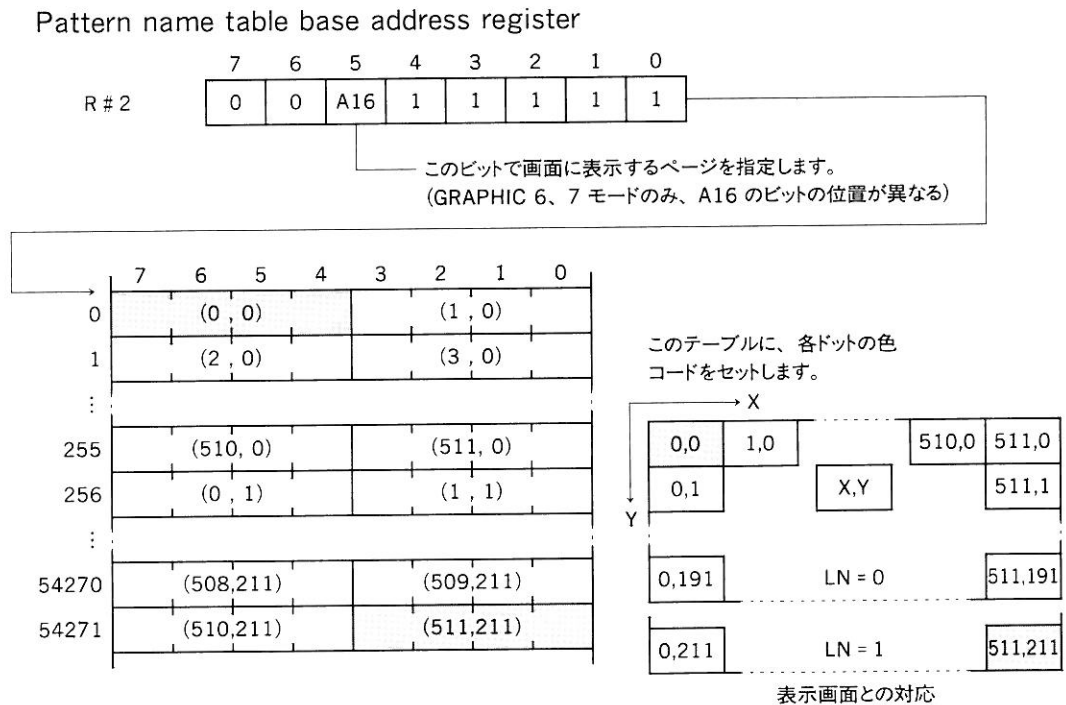
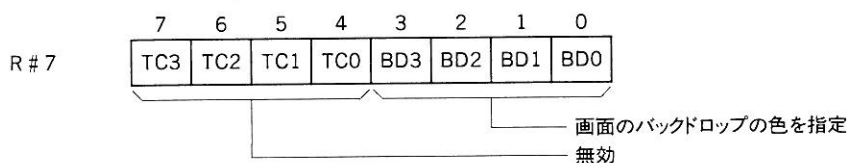


図 4.26 GRAPHIC 6 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.8.4 GRAPHIC 6 (SCREEN 7)モードの VRAM マップ

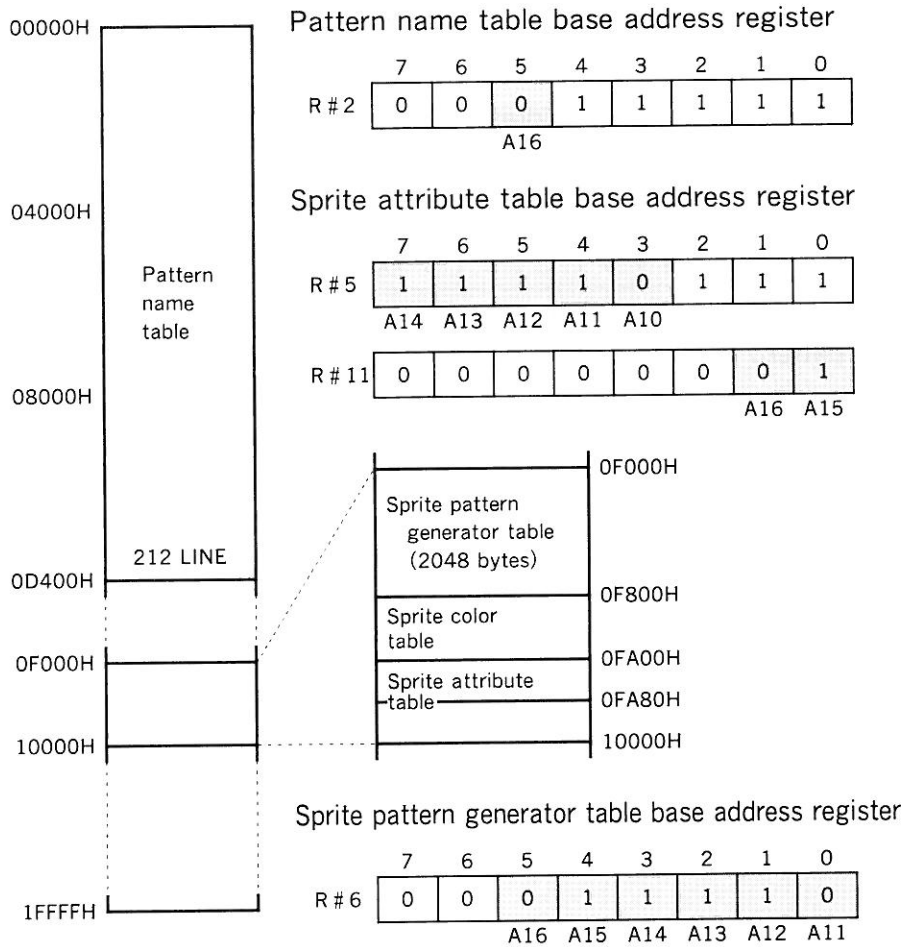


図 4.27 GRAPHIC 6 モードの VRAM マップ

以下同様に、2 ページまで割り付け可能 (VRAM 128K 実装機種) です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは、0FA80H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

4.9 GRAPHIC 7 (SCREEN 8)

4.9.1 特徴

- 解像度 横 256 ドット×縦 212 ドット(または縦 192 ドット)
のビットマップグラフィックモード
- 表示色 256 色同時 (全画面)
- スプライト モード 2
- 1 画面表示に必要な VRAM 容量 128K バイト (2 画面)

このモードを使用するためには、VRAM が 128K バイト必要です。

4.9.2 関係するレジスタと VRAM の領域

- グラフィックス VRAM パターンネームテーブル
- バックドロップの色コード R#7 下位 4 ビット
- スプライト VRAM スプライトアトリビュートテーブル
VRAM スプライトパターンテーブル

4.9.3 初期設定

1. モードレジスタの設定

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	1	1	1	0

Mode register 1

	7	6	5	4	3	2	1	0
R#1	0	BL	IEO	0	0	0	SI	MAG

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

□は表示モード設定用のビット (M5~M1) を GRAPHIC 7 (11100) モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドットに設定されます。その他のビットは任意に設定します。MSX のシステムソフトウェアはモードレジスタをそれぞれ 00001110、01100000、00001000、10001000 の値で初期化します。

2. パターンネームテーブルの設定

パターンネームテーブルは1バイトが画面上の1ドットに対応しており、1ドットごとに256色表示することができます。

パターンネームテーブルの先頭アドレスはレジスタ R#2 にセットします。指定できるのは、先頭アドレスの上位1ビット (A16) のみで、下位16ビット (A15~A0) は「0」とみなされます。したがって、パターンネームテーブルの先頭アドレスとして指定できるのは 00000H と 10000H の位置になります。

Pattern name table base address register

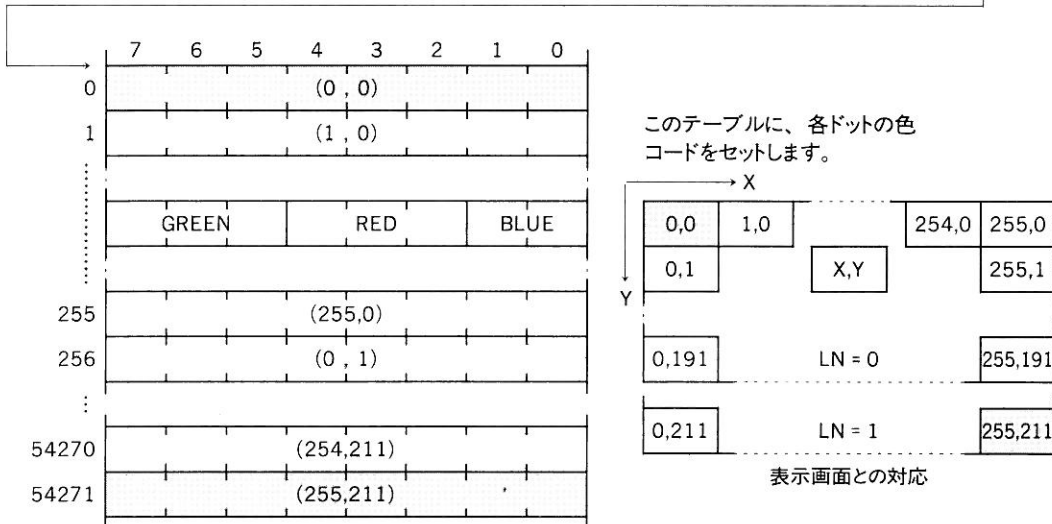
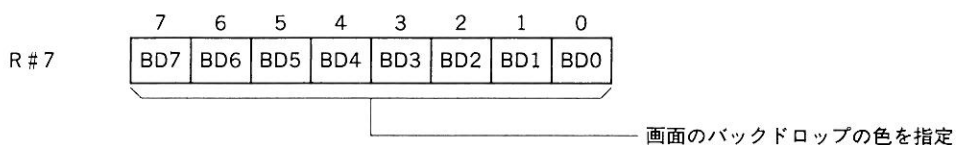


図 4.28 GRAPHIC 7 モードのパターンネームテーブル

3. カラーレジスタの設定

R#7 で画面のバックドロップの色が指定できます。

Text color / Back drop color register



4. スプライトの設定

スプライトアトリビュートテーブルの先頭アドレスをレジスタ R#5 と R#11 に、スプライトパターンジェネレータテーブルの先頭アドレスをレジスタ R#6 にセットします。詳しくは、「6.2 スプライトモード 2」の項を参照して下さい。

Sprite attribute table base address register

	7	6	5	4	3	2	1	0
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

	7	6	5	4	3	2	1	0
R#6	0	0	A16	A15	A14	A13	A12	A11

4.9.4 GRAPHIC 7 (SCREEN 8)モードの VRAM マップ

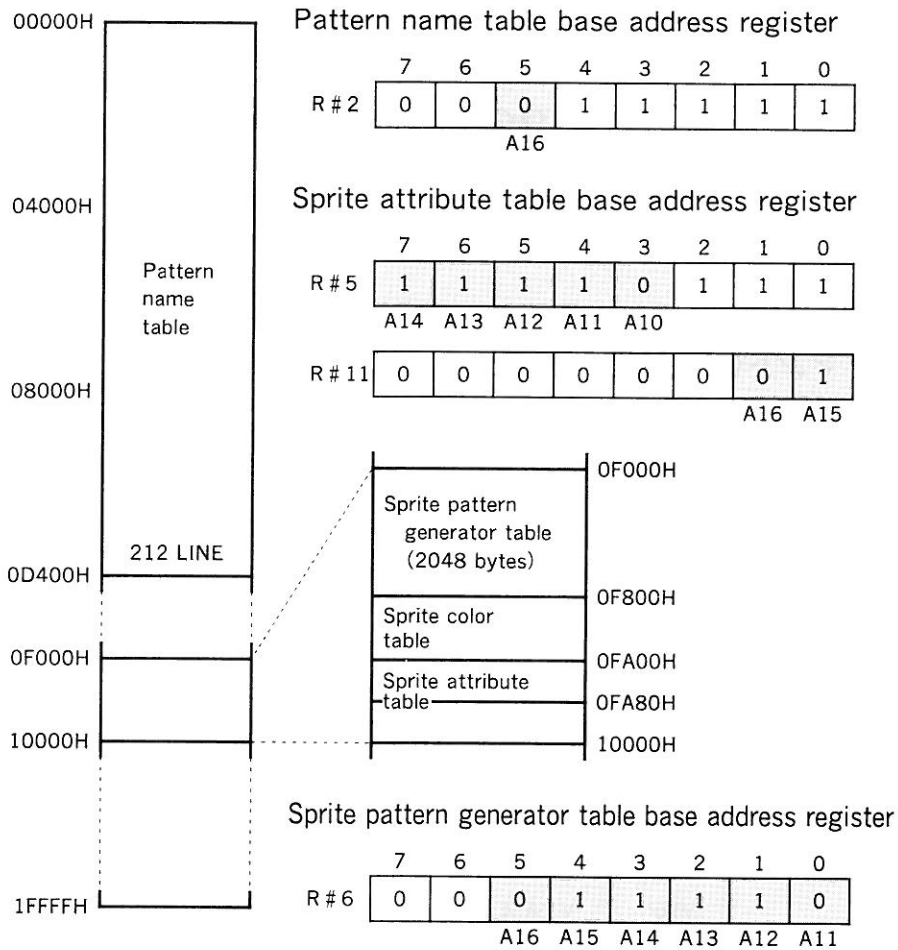


図 4.29 GRAPHIC 7 モードの VRAM マップ

以下同様に、2 ページまで割り付け可能 (VRAM 128K 実装機種) です。BASIC では SET PAGE 命令で切り換えて表示することができます。

この画面モードでは 0FA80H から 32 バイトがパレットテーブルとして使われます (MSX のシステムソフトウェアが設定する)。

5章

V9938 のコマンド

5.1 コマンドの種類

V9938 に備わっている以下のコマンドを使えば、描画 (LINE、PSET など) や画面の部分転送が簡単にできます。

VDP コマンドの使用例は添付のフロッピーディスクの中に入っていますので、そのサンプルプログラムを参照して下さい。

表 4.4 VDP のコマンド一覧表

コマンド名	転送元	転送先	転送単位	ニーモニック	R #46(上位4ビット)
High speed move (高速移動)	CPU	VRAM	バイト	HMMC	1 1 1 1
	VRAM	VRAM	バイト	YMMM	1 1 1 0
	VRAM	VRAM	バイト	HMMM	1 1 0 1
	VDP	VRAM	バイト	HMMV	1 1 0 0
Logical move (論理移動)	CPU	VRAM	ドット	LMMC	1 0 1 1
	VRAM	CPU	ドット	LMCM	1 0 1 0
	VRAM	VRAM	ドット	LMMM	1 0 0 1
	VDP	VRAM	ドット	LMMV	1 0 0 0
Line(描線)	VDP	VRAM	ドット	LINE	0 1 1 1
Search(探索)	VDP	VRAM	ドット	SRCH	0 1 1 0
Pset(描点)	VDP	VRAM	ドット	PSET	0 1 0 1
Point	VRAM	VDP	ドット	POINT	0 1 0 0
Invalid	——	——	——	——	0 0 1 1
Invalid	——	——	——	——	0 0 1 0
Invalid	——	——	——	——	0 0 0 1
Stop	——	——	——	——	0 0 0 0

V9938 はレジスタ R # 46 (Command register) にデータが書き込まれると、ステータスレジスタ S # 2 のビット 0 (CE/Command Execute) を 1 にセットしてからコマンドの実行を開始します。必要なパラメータはコマンド実行前にレジスタ R # 32~R # 45 にセットされていなければなりません。

コマンドの実行が終了すると CE は 0 になります。コマンドの実行を途中で中断するには、STOP コマンドを実行して下さい。

コマンドの動作は、ビットマップモード (SCREEN 5~8 あるいは GRAPHIC 4~7) のときしか保証されません。

5.2 ページの概念

V9938 のコマンドを使用する場合の位置パラメータは、すべて X、Y 座標です。つまり、V9938 内部のコマンドプロセッサは、VRAMの全エリアを表示モードに応じたX、Y座標でアクセスします。

画面に表示されるのは、同一ページ内の 212 ラインです (R # 23 で指定)。表示ページの選択は R # 2 で行います。コマンドの実行は、表示ページの選択とは独立して実行されます。

各表示モードにおける VRAM と座標との関係は図 4.30 のとおりです。

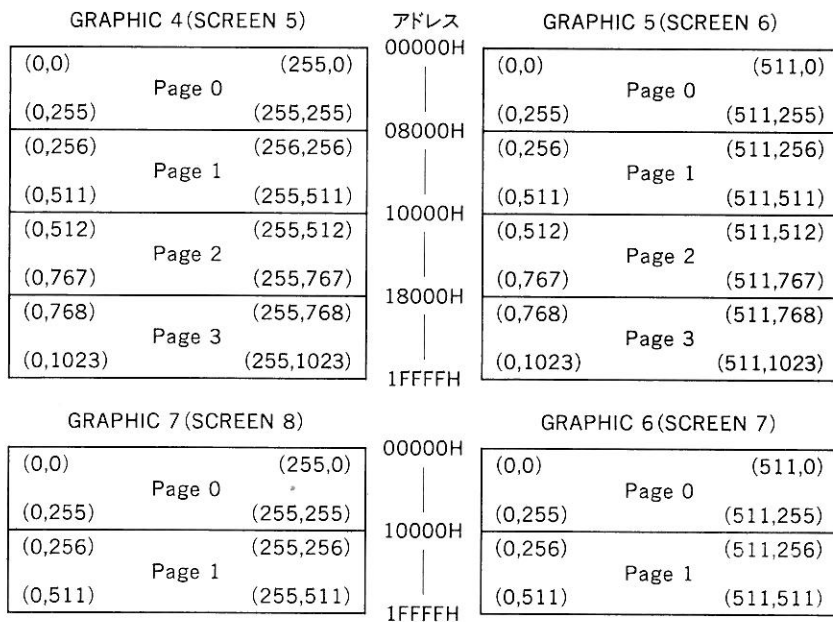


図 4.30 各モードの VRAM の座標

5.3 ロジカルオペレーション

V9938 では、LINE、PSET、LOGICAL MOVE などのコマンドを実行するときに、指定したデータと画面上の色を演算することができます。演算(ロジカルオペレーション)は、コマンドを指定するときに R # 46 (Command register) の下位 4 ビットに同時に書き込みます。

表 4.5 ロジカルオペレーションコード

演算名	Operation(演算動作)	R # 46(下位 4 ビット)
IMP	DC = SC	0 0 0 0
AND	DC = SC×DC	0 0 0 1
OR	DC = SC+DC	0 0 1 0
EOR	DC = SC×DC+SC×DC	0 0 1 1
NOT	DC = SC	0 1 0 0
—		0 1 0 1
—		0 1 1 0
—		0 1 1 1
TIMP	if SC = 0 then DC = DC else DC = SC	1 0 0 0
TAND	if SC = 0 then DC = DC else DC = SC×DC	1 0 0 1
TOR	if SC = 0 then DC = DC else DC = SC+DC	1 0 1 0
TEOR	if SC = 0 then DC = DC else DC = SC×DC+SC×DC	1 0 1 1
TNOT	if SC = 0 then DC = DC else DC = SC	1 1 0 0
—		1 1 0 1
—		1 1 1 0
—		1 1 1 1

* SC = Source Color code

* DC = Destination Color code

* EOR = Exclusive OR

5.4 各コマンドの解説

5.4.1 HMMC(CPU → VRAM 高速転送)

CPU から VRAM の矩形領域 (X、Y 座標上) へ、V9938 経由でデータを転送するコマンドです。データの転送は 1 バイト単位で行われるので、X 座標として指定できる最大値は、表示モードによって制限を受けます。

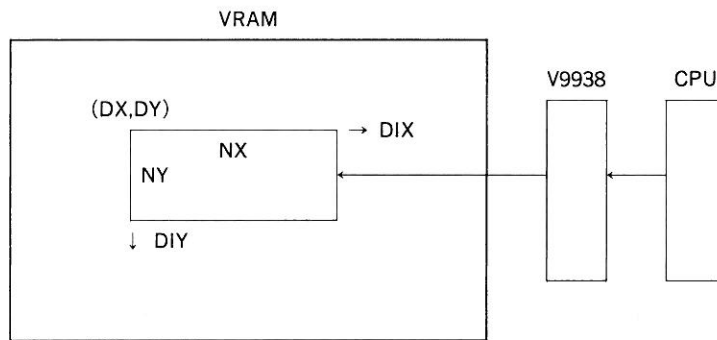


図 4.31 HMMC コマンドの動作

1. HMMC 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DX	転送先基準点 X 座標 (0~511)
DY	転送先基準点 Y 座標 (0~1023)
NX	X 方向転送ドット数 (1~512* ¹)
NY	Y 方向転送ドット数 (1~1024* ²)

DX、NX ともに、GRAPHIC 4 と 6 (SCREEN 5 と 7) モードのときは下位 1 ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位 2 ビットが無視されます。

*¹ 512 を指定するときは、NX には「0」を入れます。

*² 1024 を指定するときは、NY には「0」を入れます。

DIX	転送先基準点からの NX の方向 (0 = 右、1 = 左)
DIY	転送先基準点からの NY の方向 (0 = 下、1 = 上)

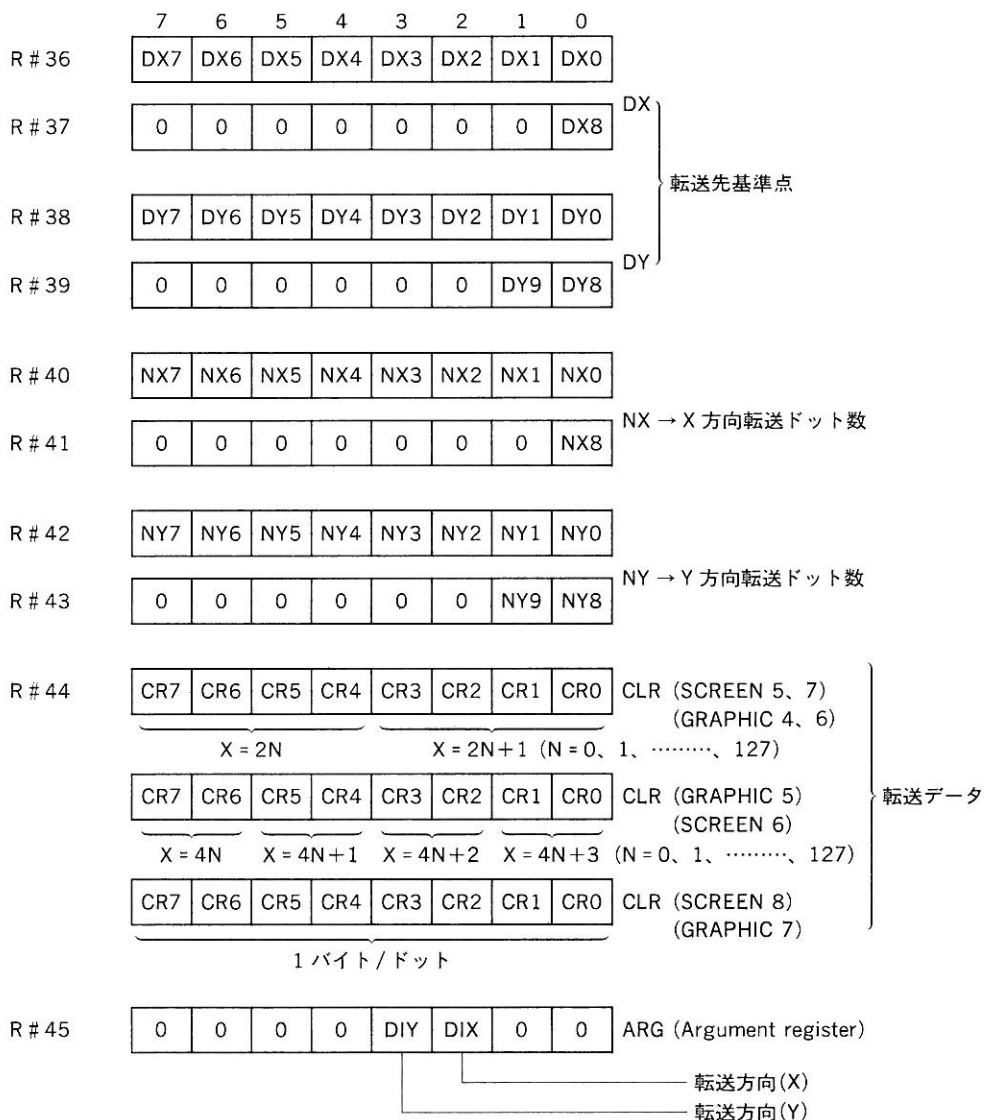
CLR (R#44 Color register) 転送データの第1バイト

2) 上記のデータをセットした後、コマンドを実行します。

CMR (R#46 Command register) に 11110000B を書き込む

3) ステータスレジスタ S#2 の TR と CE をチェックしながら、転送データの第2バイト以降を CLR に転送します。

2. HMMC レジスタセットアップ



3. HMMC コマンド実行

	7	6	5	4	3	2	1	0	
R # 46	1	1	1	1	0	0	0	0	CMR

4. HMMC 実行フローチャート

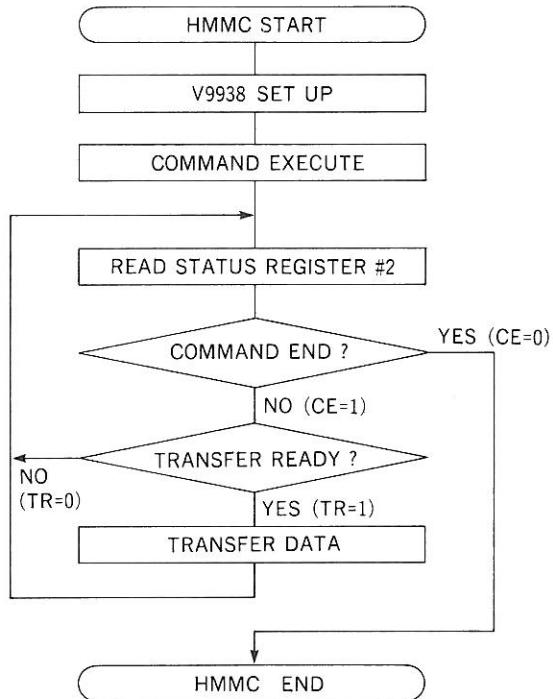


図 4.32 HMMC コマンド実行のフローチャート

5.4.2 YMMM(Y 方向の VRAM 間高速転送)

VRAM 上の DX、SY、NY、DIX、DIY と画面の右端(または左端)で指定される領域を Y 軸方向 (DY で指定) に転送するコマンドです。

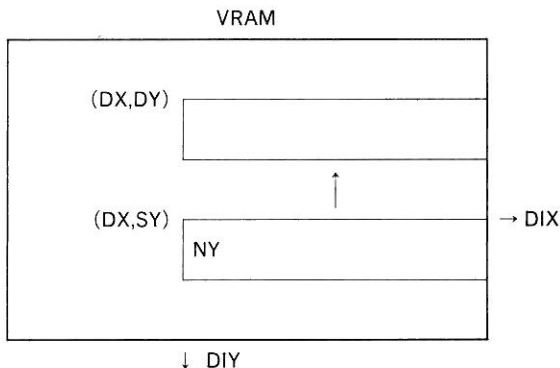


図 4.33 YMMM コマンドの動作

1. YMMM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DY	転送先基準点 Y 座標 (0~1023)
DX	転送元基準点 X 座標 (0~511)
SY	転送元基準点 Y 座標 (0~1023)
NY	Y 方向転送ドット数 (1~1024*)

DX は GRAPHIC4 と 6 (SCREEN 5 と 7) モードのときは下位 1 ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位 2 ビットが無視されます。

* 1024 を指定するときは、NY には「0」を入れます。

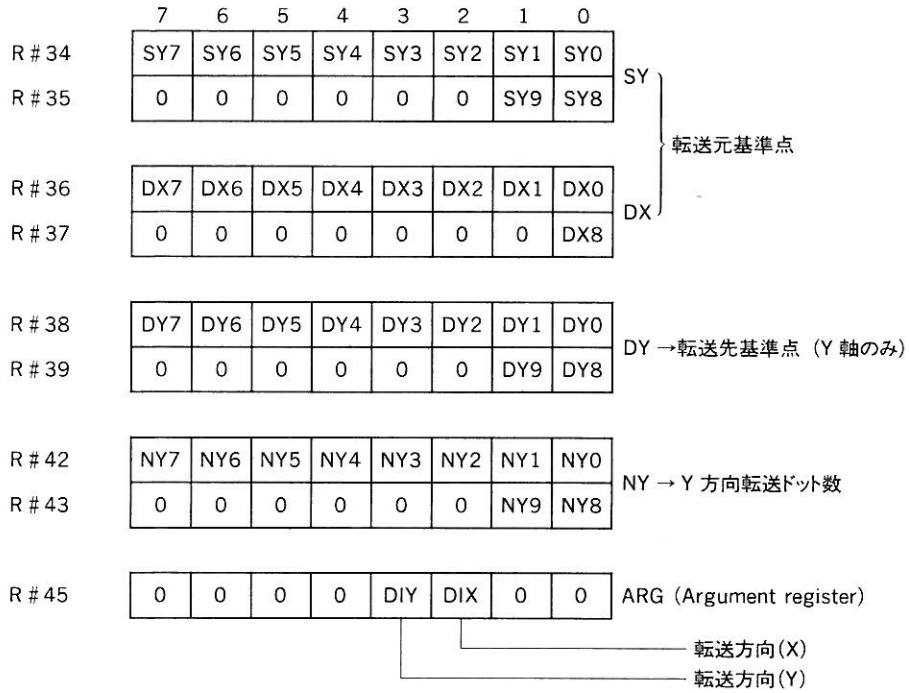
DIX	転送元基準点から左右どちらの画面端までを転送するのかを設定 (0 = 右、1 = 左)
DIY	転送元基準点から見た NY の方向 (0 = 下、1 = 上)

2) 上記のデータをセットした後コマンドを実行します。

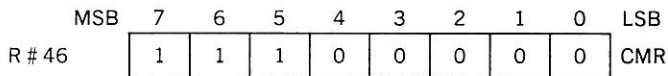
CMR (R # 46 Command register) に 11100000B を書き込む

3) 以上の操作で V9938 は YMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。

2. YMMM レジスタセットアップ



3. YMMM コマンド実行



5.4.3 HMMM(VRAM 間高速転送)

VRAM から VRAM へ矩形の領域を転送するコマンドです。データの転送は1バイト単位で行われるので、X 座標として指定できる最大値は表示モードによって制限を受けます。

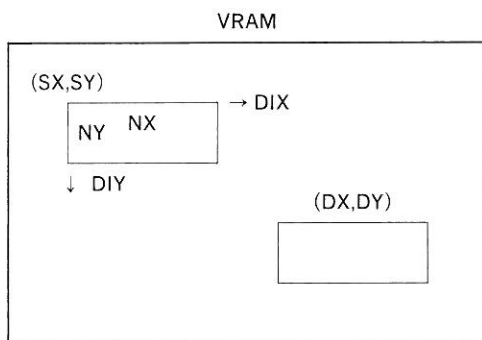


図 4.34 HMMM コマンドの動作

1. HMMM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点 X 座標 (0~511)
SY	転送元基準点 Y 座標 (0~1023)
NX	X 方向転送ドット数 (1~512* ¹)
NY	Y 方向転送ドット数 (1~1024* ²)
DIX	基準点からの NX の方向 (0 = 右、1 = 左)
DIY	基準点からの NY の方向 (0 = 下、1 = 上)
DX	転送先基準点 X 座標 (0~511)
DY	転送先基準点 Y 座標 (0~1023)

SX、DX、NX とともに、GRAPHIC 4 と 6 (SCREEN 5 と 7) のときは下位 1 ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位 2 ビットが無視されます。

*¹ 512 を指定するときは、NX には「0」を入れます。

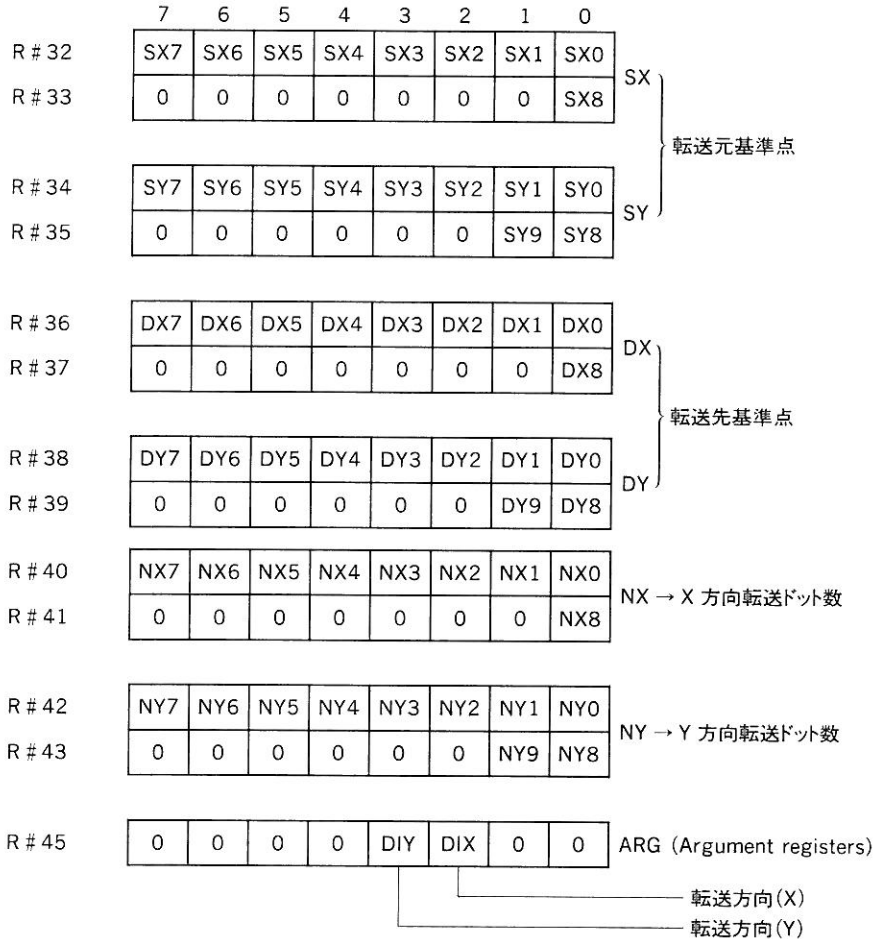
*² 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

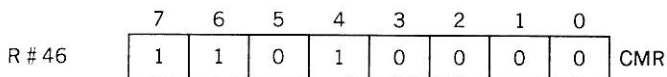
CMR (R # 46 Command register) に 11010000B を書き込む

3) 以上の操作で V9938 は HMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. HMMM レジスタセットアップ



3. HMMM コマンド実行



5.4.4 HMMV(長方形の高速塗りつぶし)

VRAMの矩形の領域をカラーコードで塗りつぶすコマンドです。データの転送は1バイト単位で行われるので、X座標として指定できる最大値は表示モードによって制限を受けます。

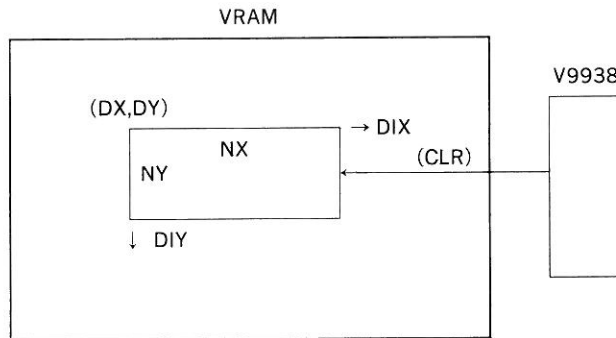


図 4.35 HMMV コマンドの動作

1. HMMV 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

NX	X 方向転送ドット数 (1~512* ¹)
NY	Y 方向転送ドット数 (1~1024* ²)
DIX	転送先基準点からの NX の方向 (0 = 右、1 = 左)
DIY	転送先基準点からの NY の方向 (0 = 下、1 = 上)
DX	転送先基準点 X 座標 (0~511)
DY	転送先基準点 Y 座標 (0~1023)

DX、NX ともに、GRAPHIC 4 と 6 (SCREEN 5 と 7) のときは下位 1 ビット、GRAPHIC 5 (SCREEN 6) モードのときは下位 2 ビットが無視されます。

*1 512 を指定するときは、NX には「0」を入れます。

*2 1024 を指定するときは、NY には「0」を入れます。

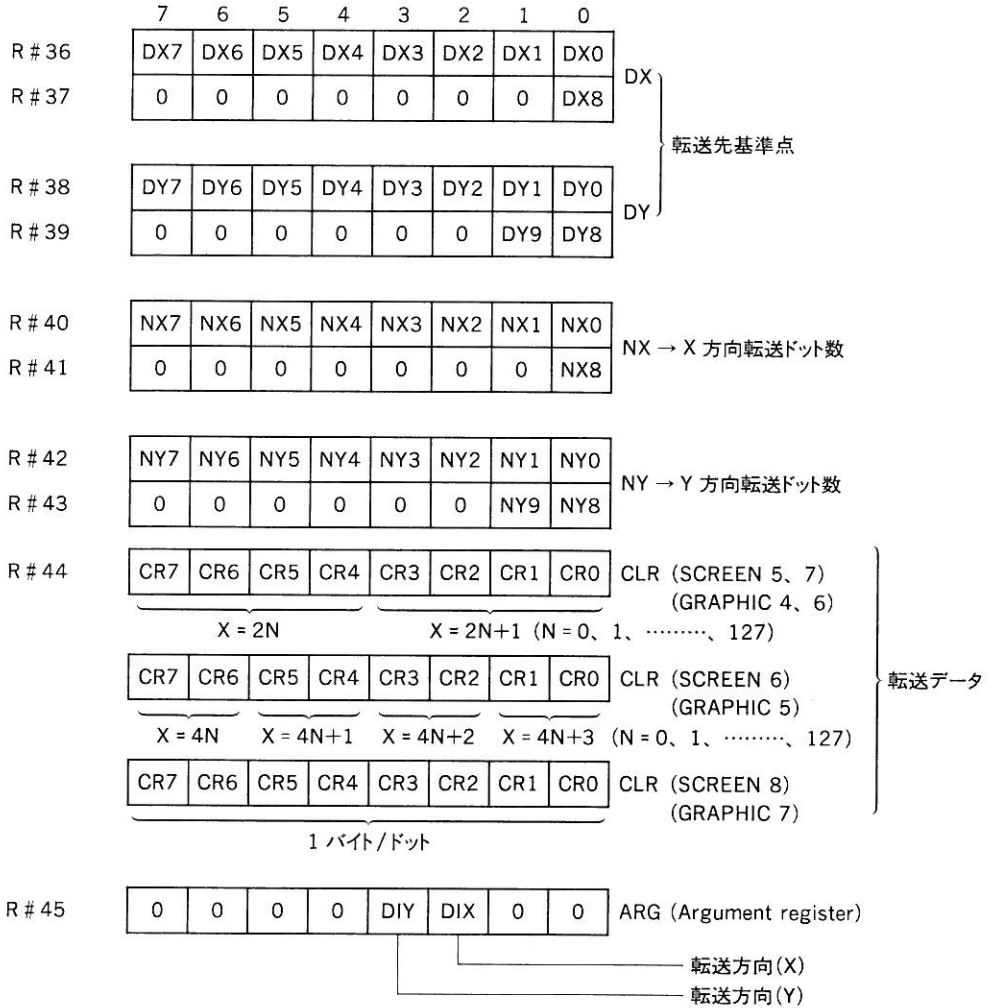
CLR (R # 44 Color register) 塗りつぶしデータ

2) 上記のデータをセットした後コマンドを実行します。

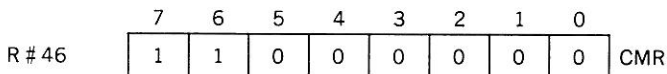
CMR (R # 46 Command register) に 11000000B を書き込む。

3) 以上の操作で V9938 は HMMV コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. HMMV レジスタセットアップ



3. HMMC コマンド実行



5.4.5 LMMC(CPU → VRAM 論理転送)

CPU から VRAM の矩形領域へ、V9938 経由でデータを転送するコマンドです。データの転送は1ドット単位で行われ、転送先のデータと演算することができます。

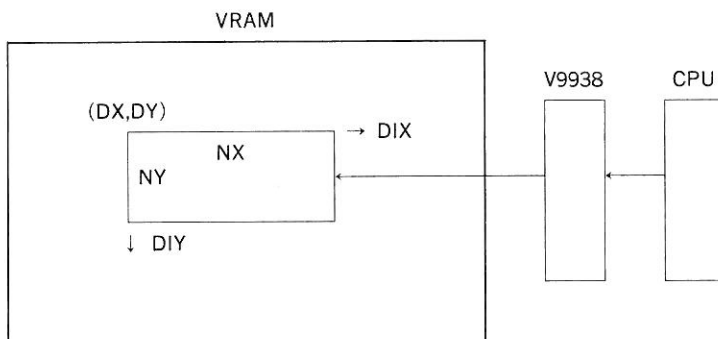


図 4.36 LMMC コマンドの動作

1. LMMC 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

DX	転送先基準点 X 座標 (1~512* ¹)
DY	転送先基準点 Y 座標 (1~1024* ²)
NX	X 方向転送ドット数 (0~511)
NY	Y 方向転送ドット数 (0~1023)
DIX	転送先基準点からの NX の方向 (0 = 右、1 = 左)
DIY	転送先基準点からの NY の方向 (0 = 下、1 = 上)
CLR	(R # 44 Color register) 転送データの第1バイト

*1 512 を指定するときは、NX には「0」を入れます。

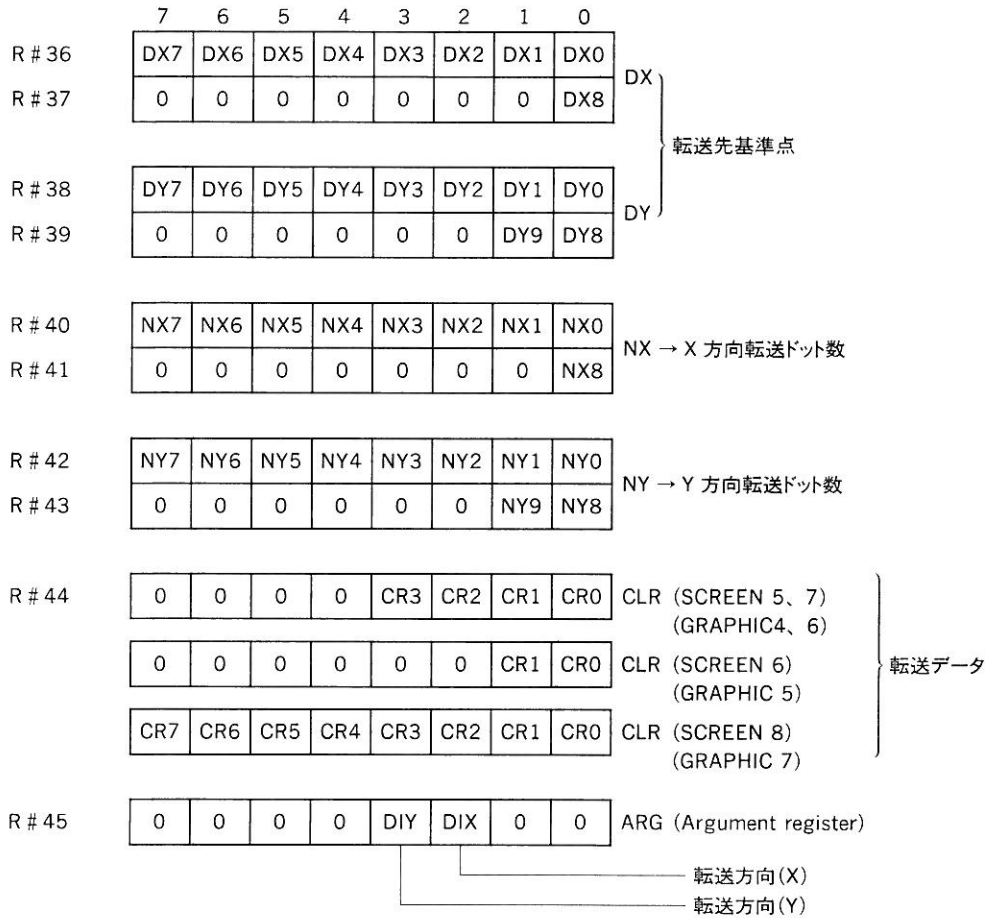
*2 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

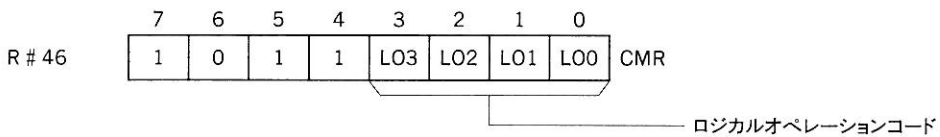
CMR (R # 46 Command register) の上位4ビットに1011Bを、下位4ビットにロジカルオペレーションコード(表4.5参照)を書き込む。

3) ステータスレジスタ S#2 の TR と CE をチェックしながら、転送データの第2バイト以降を CLR レジスタに転送します。

2. LMMC レジスタセットアップ



3. LMMC コマンド実行



4. LMMC 実行フローチャート

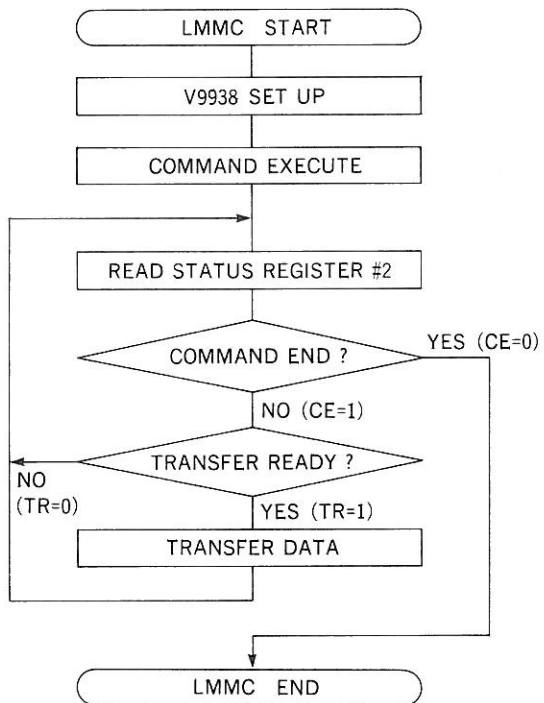


図 4.37 LMMC コマンド実行のフローチャート

5.4.6 LMCM(VRAM → CPU 論理転送)

VRAM の矩形領域 (X、Y 座標上) のデータを CPU に転送するコマンドです。データの転送は 1 ドット単位で行われます。

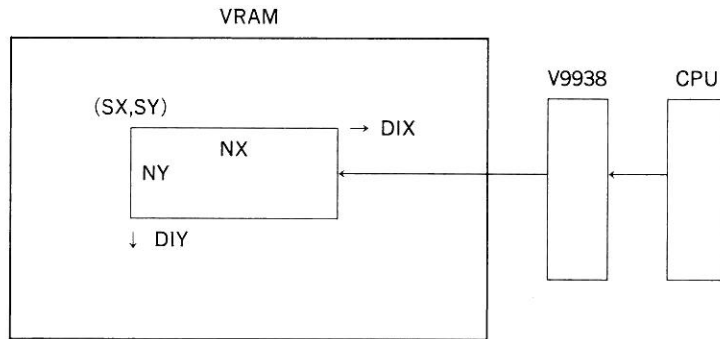


図 4.38 LMCM コマンドの動作

1. LMCM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点 X 座標 (0~511)
SY	転送元基準点 Y 座標 (0~1023)
NX	X 方向転送ドット数 (1~512* ¹)
NY	Y 方向転送ドット数 (1~1024* ²)
DIX	転送元基準点からの NX の方向 (0 = 右、1 = 左)
DIY	転送元基準点からの NY の方向 (0 = 下、1 = 上)

*¹ 512 を指定するときは、NX には「0」を入れます。

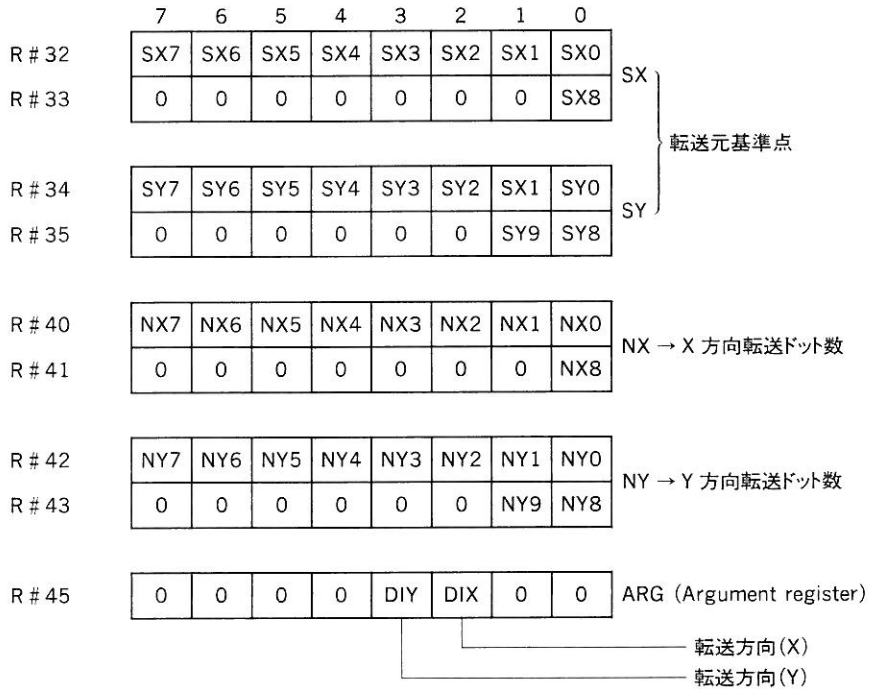
*² 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

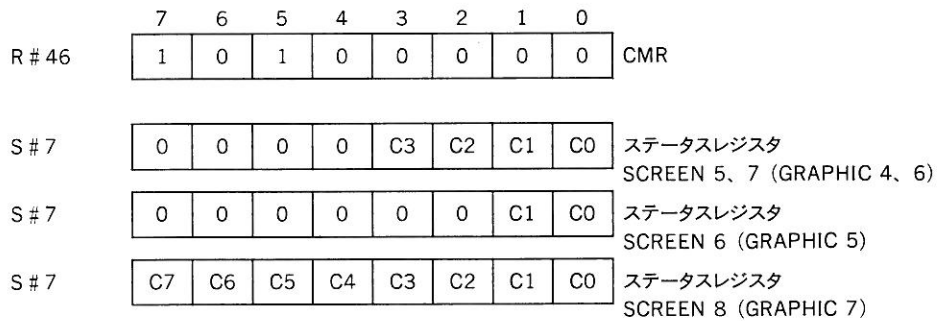
CMR (R # 46 Command register) に 10100000B を書き込む。

3) ステータスレジスタ S # 2 の TR と CE をチェックしながら、ステータスレジスタ S # 7 を読み出します。

2. LMCM レジスタセットアップ



3. LMCM コマンド実行



4. LMCM 実行フローチャート

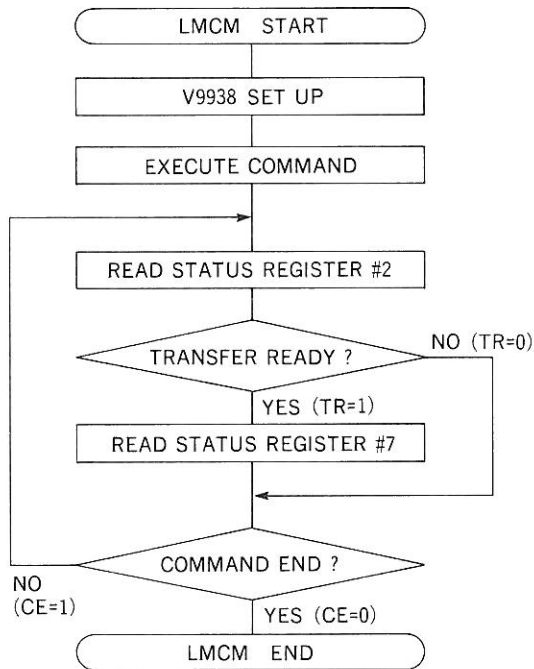


図 4.39 LMCM コマンド実行のフローチャート

注意

1. 「EXECUTE COMMAND」の前に TR をリセットする必要があるため、「V9938 SET UP」の中でステータスレジスタ S#7 をリードして下さい。
2. 最後のデータがステータスレジスタ S#7 にセットされて TR = 1 となっても、V9938 内部ではコマンドは終了し CE = 0 となります。

5.4.7 LMMM(VRAM 間論理転送)

VRAM から VRAM へ矩形の領域を転送するコマンドです。データの転送は1ドット単位で行われ、転送先のデータと論理演算をすることができます。

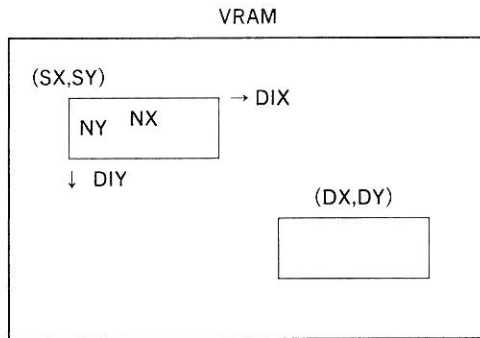


図 4.40 LMMM コマンドの動作

1. LMMM 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX	転送元基準点 X 座標 (0~511)
SY	転送元基準点 Y 座標 (0~1023)
NX	X 方向転送ドット数 (1~512* ¹)
NY	Y 方向転送ドット数 (1~1024* ²)
DIX	基準点からの NX の方向 (0 = 右、1 = 左)
DIY	基準点からの NY の方向 (0 = 下、1 = 上)
DX	転送先基準点 X 座標 (0~511)
DY	転送先基準点 Y 座標 (0~1023)

*¹ 512 を指定するときは、NX には「0」を入れます。

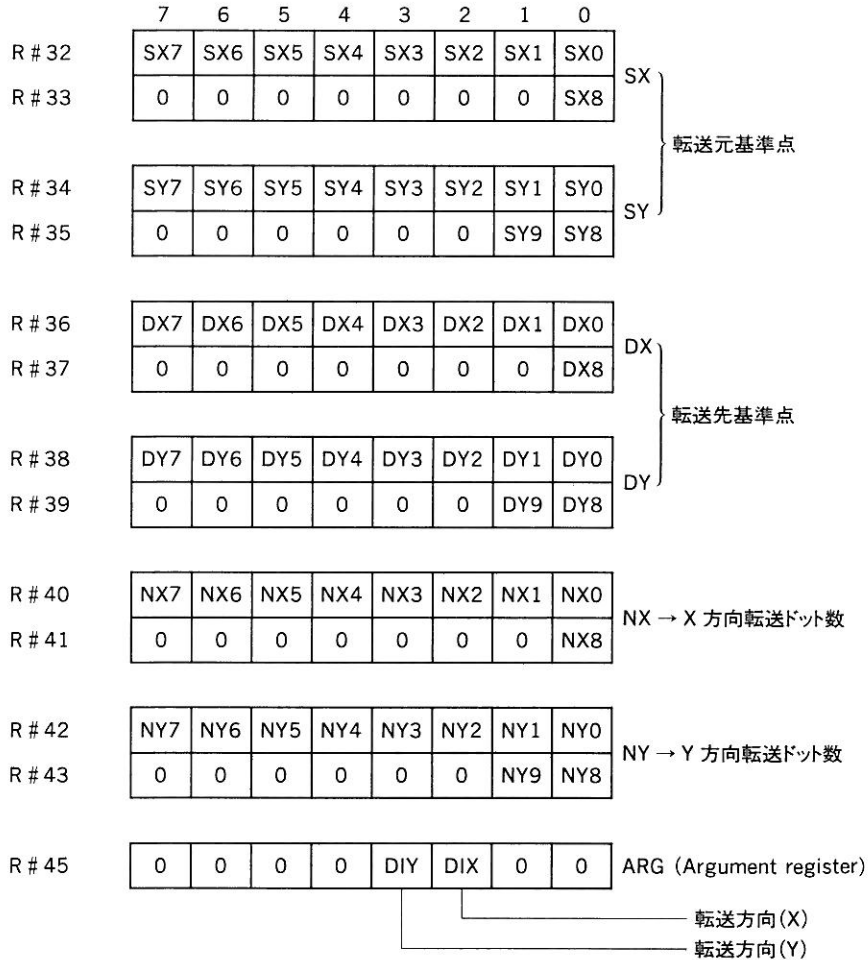
*² 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

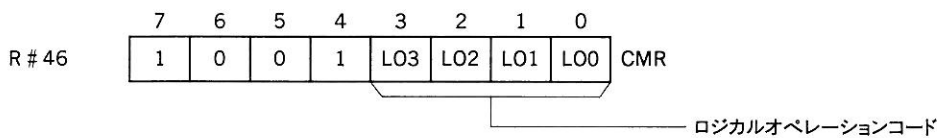
CMR (R # 46 Command register) の上位 4 ビットに 1001B を、下位 4 ビットにロジカルオペレーションコード (表 4.5 参照) を書き込む。

- 3) 以上の操作で V9938 は LMMM コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. LMMM レジスタセットアップ



3. LMMM コマンド実行



5.4.8 LMMV (VRAM 論理塗りつぶし)

VRAM の矩形の領域をカラーコードで塗りつぶすコマンドです。データの転送は1ドット単位で行われ、転送先のデータと論理演算をすることができます。

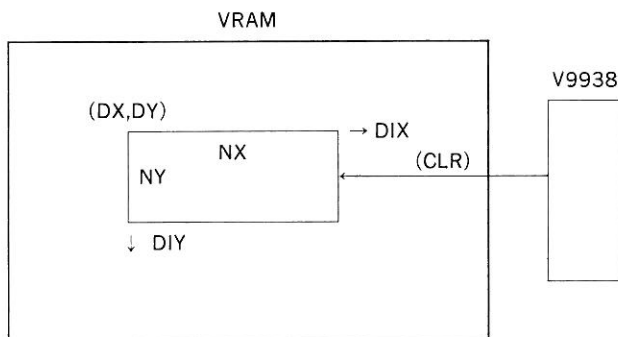


図 4.41 LMMV コマンドの動作

1. LMMV 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

NX	X 方向転送ドット数 (1~512 ^{*1})
NY	Y 方向転送ドット数 (1~1024 ^{*2})
DIX	転送先基準点からの NX の方向 (0 = 右、1 = 左)
DIY	転送先基準点からの NY の方向 (0 = 下、1 = 上)
DX	転送先基準点 X 座標 (0~511)
DY	転送先基準点 Y 座標 (0~1023)
CLR	(R # 44 Color register) 塗りつぶしデータ

*1 512 を指定するときは、NX には「0」を入れます。

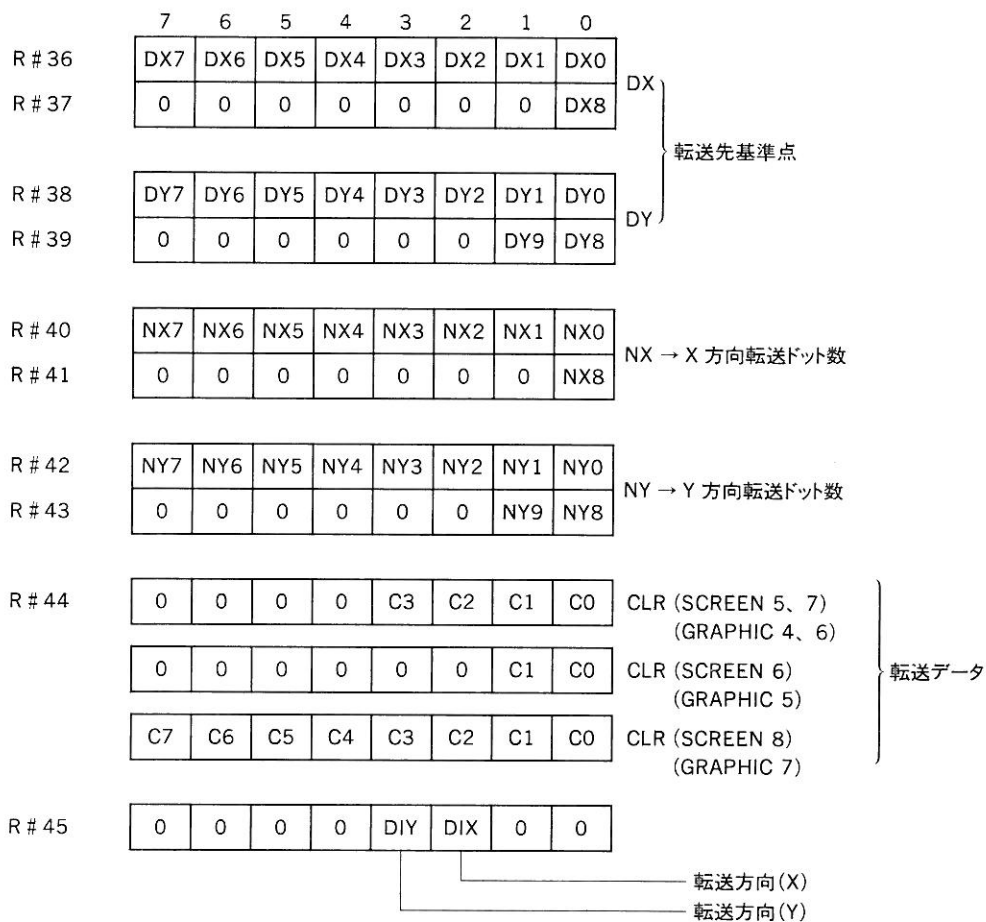
*2 1024 を指定するときは、NY には「0」を入れます。

2) 上記のデータをセットした後コマンドを実行します。

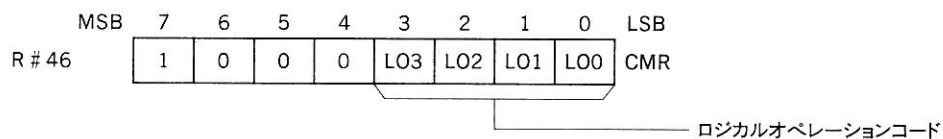
CMR (R # 46 Command register) の上位 4 ビットに 1000B を、下位 4 ビットにロジカルオペレーションコード (表 4.5 参照) を書き込む

3) 以上の操作で V9938 は LMMV コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。

2. LMMV レジスタセットアップ



3. LMMV コマンド実行



5.4.9 LINE(直線の描画)

直線を描画するコマンドです。基準点と長辺・短辺からなる長方形の対角線を描画します。データの転送は1ドット単位で行われ、転送先のデータと論理演算をすることができます。

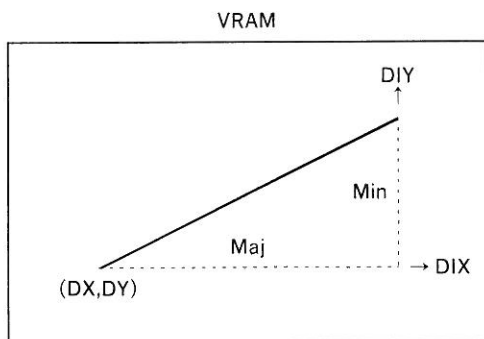


図 4.42 LINE コマンドの動作

1. LINE 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

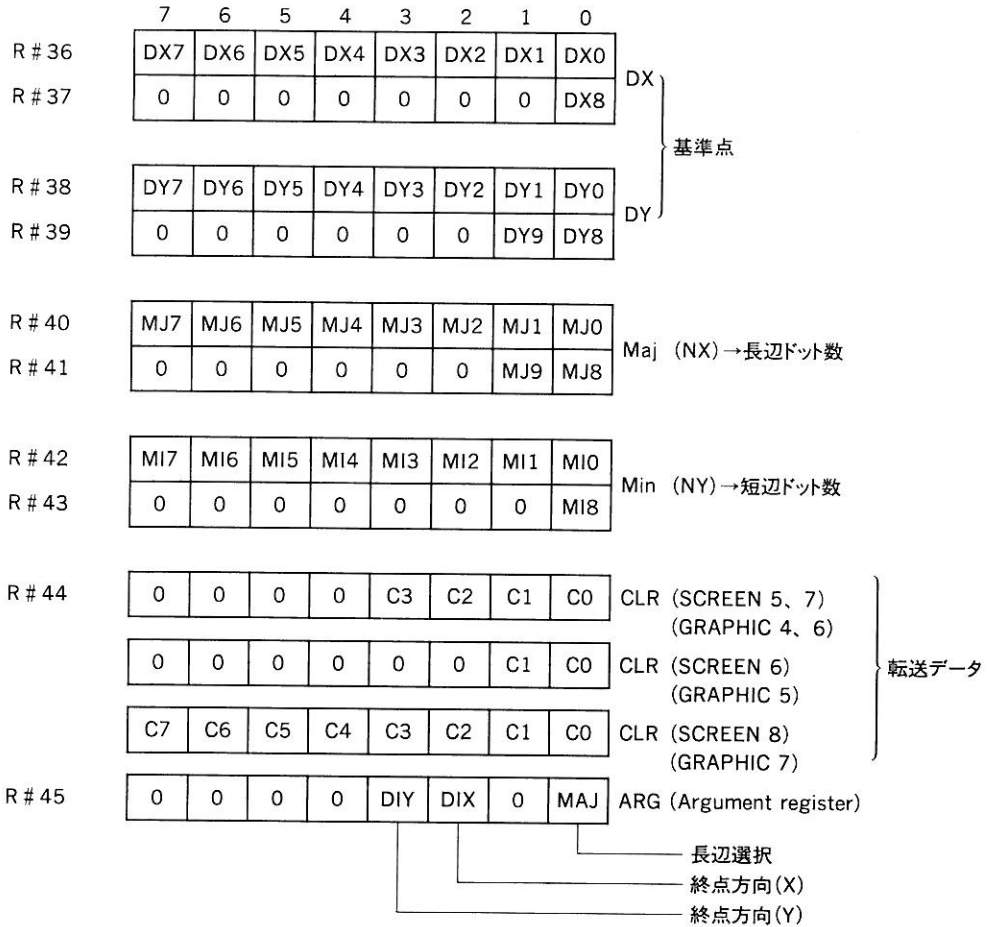
Maj	長辺ドット数 (0~1023)
Min	短辺ドット数 (0~511)
MAJ	0=長辺は X 軸と平行、1=長辺は Y 軸と平行 / または長辺=短辺
DIX	基準点からの終点の方向 (0=右、1=左)
DIY	基準点からの終点の方向 (0=下、1=上)
DX	基準点 X 座標 (0~511)
DY	基準点 Y 座標 (0~1023)

2) 上記のデータをセットした後コマンドを実行します。

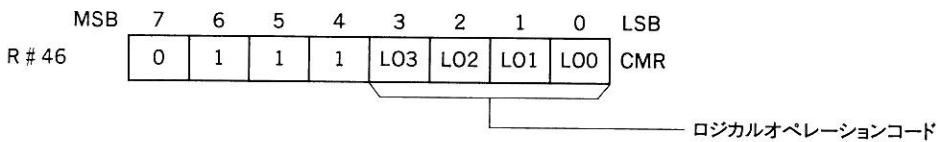
CMR (R # 46 Command register) の上位 4 ビットに 0111B を、下位 4 ビットにロジカルオペレーションコード (表 4.5 参照) を書き込む。

3) 以上の操作で V9938 は LINE コマンドを実行します。コマンドの実行中はステータスレジスタ S # 2 の CE ビットが「1」になり、終了すると「0」になります。

2. LINE レジスタセットアップ



3. LINE コマンド実行



5.4.10 SRCH(色コードのサーチ)

基準点から右(左)に境界色(または非境界色)をサーチするコマンドです。

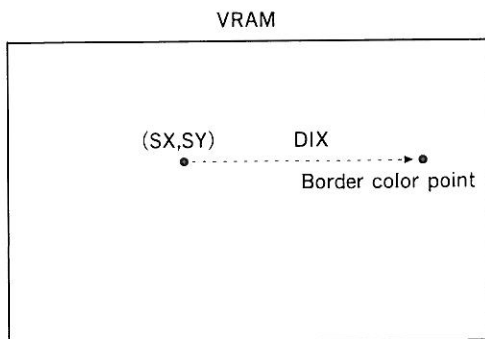


図 4.43 SRCH コマンドの動作

1. SRCH 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

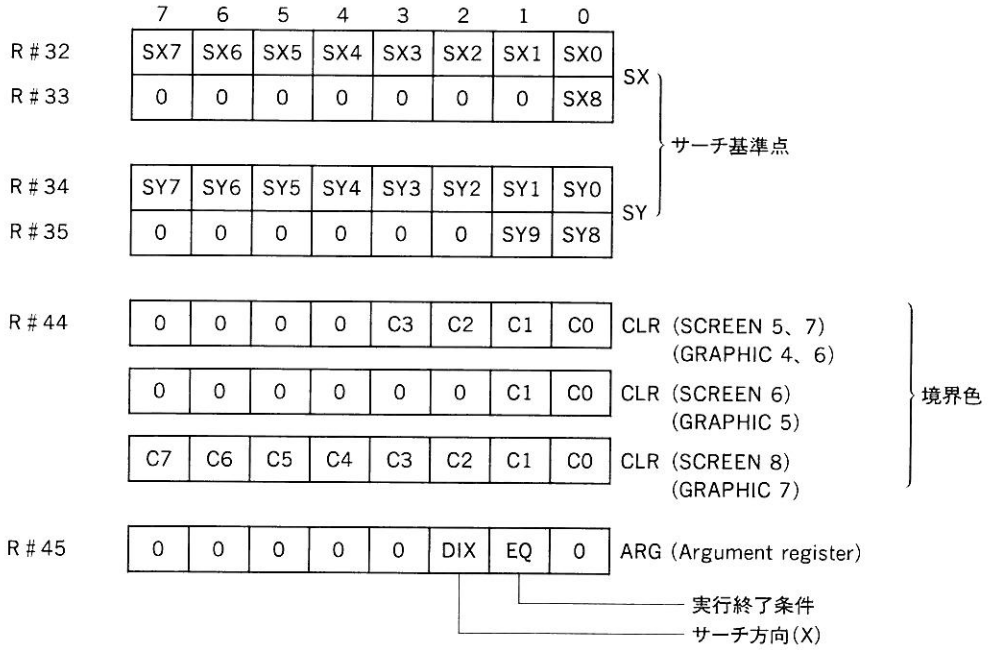
DIX	基準点からのサーチ方向 (0 = 右、1 = 左)
EQ	0 = 境界色を発見したときに実行を終了、1 = 境界色以外を発見したときに実行を終了。
SX	サーチ基準点 X 座標 (0~511)
SY	サーチ基準点 Y 座標 (0~1023)
CLR	(R # 44 Color register) 境界色

2) 上記のデータをセットした後コマンドを実行します。

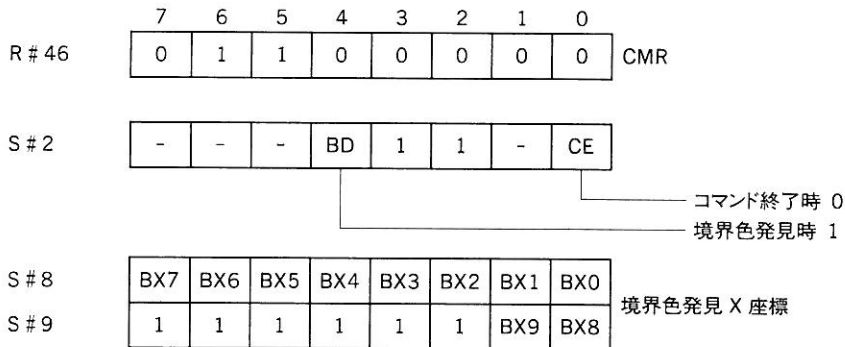
CMR (R # 46 Command register) に 01100000B を書き込む。

3) 以上の操作で V9938 は SRCH コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。また、画面の端に達するまでに境界色を発見した場合に BD ビットが「1」になります。

2. SRCH レジスタセットアップ



3. SRCH コマンド実行



4. SRCH 実行フローチャート

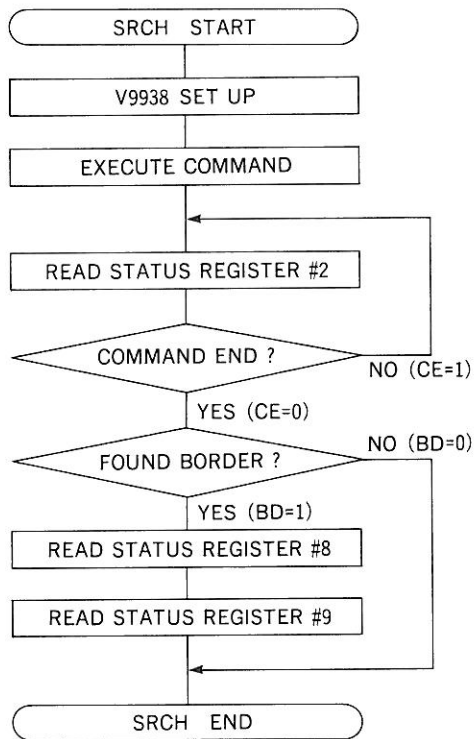


図 4.44 SRCH コマンド実行のフローチャート

5.4.11 PSET(点の描画)

点を描画するコマンドです。すでに表示されている点のデータと論理演算をすることができます。

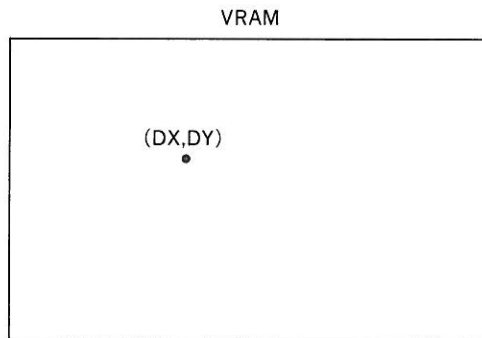


図 4.45 PSET コマンドの動作

1. PSET 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

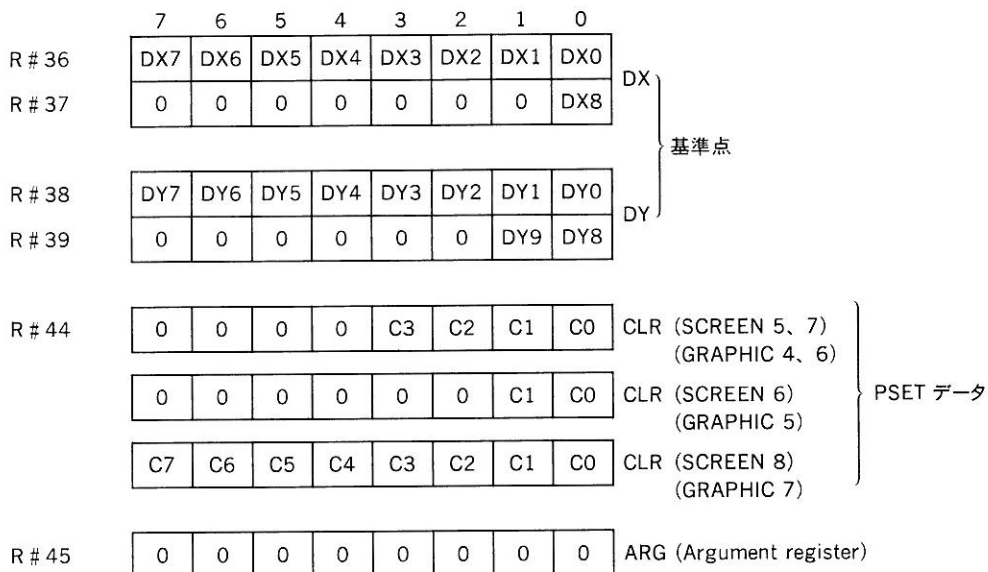
DX	基準点 X 座標 (0~511)
DY	基準点 Y 座標 (0~1023)

2) 上記のデータをセットした後コマンドを実行します。

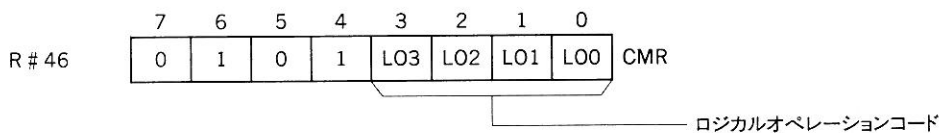
CMR	(R # 46 Command register) の上位 4 ビットに 0101B を、下位 4 ビットにロジカルオペレーションコード (表 4.5 参照) を書き込む。
-----	--

3) 以上の操作で V9938 は PSET コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。

2. PSET レジスタセットアップ



3. PSET コマンド実行



5.4.12 POINT(色コードの読み出し)

VRAM 上の基準点の色コードを読み出すコマンドです。

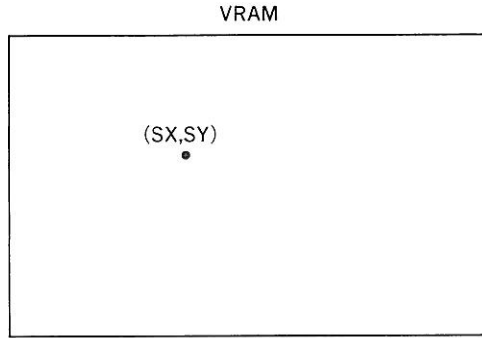


図 4.46 POINT コマンドの動作

1. POINT 実行手順

1) V9938 のコマンドレジスタに必要なパラメータをセットします。

SX 基準点 X 座標 (0~511)

SY 基準点 Y 座標 (0~1023)

2) 上記のデータをセットした後コマンドを実行します。

CMR (R # 46 Command register) に 01000000B を書き込む

3) 以上の操作で V9938 は POINT コマンドを実行します。コマンドの実行中はステータスレジスタ S#2 の CE ビットが「1」になり、終了すると「0」になります。また、読み出された色コードがステータスレジスタ S#7 にセットされます。

2. POINT レジスタセットアップ

	7	6	5	4	3	2	1	0	
R # 32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	SX
R # 33	0	0	0	0	0	0	0	SX8	
R # 34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	SY
R # 35	0	0	0	0	0	0	SY9	SY8	

} 基準点

3. POINT コマンド実行

	7	6	5	4	3	2	1	0	
R # 46	0	1	0	0	0	0	0	0	CMR
S # 2	-	-	-	-	1	1	-	CE	コマンド終了時 0
S # 7	0	0	0	0	C3	C2	C1	C0	カラーコード SCREEN 5、7 (GRAPHIC 4、6)
	0	0	0	0	0	0	C1	C0	カラーコード SCREEN 6 (GRAPHIC 5)
	C7	C6	C5	C4	C3	C2	C1	C0	カラーコード SCREEN 8 (GRAPHIC 7)

5.5 コマンドの高速化

次の操作をすると、コマンドの処理速度が上がります。

1. スプライトの表示を禁止

レジスタ R # 8 のビット 1 (SPD) を「1」にすると、スプライトの処理に使っていた時間をコマンドの実行に使用できるので、コマンドの処理速度が上がります。

2. 画面の表示を禁止

レジスタ R # 1 のビット 6 (BL) を「0」にすると、表示の処理に使っていた時間をコマンドの実行に使用できるので、コマンドの処理速度が上がります。

5.6 コマンド終了時のレジスタの状態

コマンド終了時のレジスタの状態は表 4.6 のようになります。

表 4.6 コマンド終了時のレジスタの状態

コマンド名	SX	SY	DX	DY	NX	NY	CLR	CMR H	CLR L	ARG
HMMC	—	—	—	*	—	#	—	0	—	—
YMMM	—	*	—	*	—	#	—	0	—	—
HMMM	—	*	—	*	—	#	—	0	—	—
HMMV	—	—	—	*	—	#	—	0	—	—
LMMC	—	—	—	*	—	#	—	0	—	—
LMCM	—	*	—	—	—	#	*	0	—	—
LMMM	—	*	—	*	—	#	—	0	—	—
LMMV	—	—	—	*	—	#	—	0	—	—
LINE	—	—	—	*	—	—	—	0	—	—
SRCH	—	—	—	—	—	—	—	0	—	—
PSET	—	—	—	—	—	—	—	0	—	—
PINT	—	—	—	—	—	—	*	0	—	—

— 変化なし

* コマンド終了時の座標 (SY*, DY*) とカラーコード

画面端を検出したときはそのときのカウンタ数 (NYB) となる

SY*、DY*、および NYB の値は Y 方向実行ドット数を N とすると、次のように計算されます。

$$SY* = SY + N, \quad DY* = DY + N \quad (DIY = 0)$$

$$SY* = SY - N, \quad DY* = DY - N \quad (DIY = 1)$$

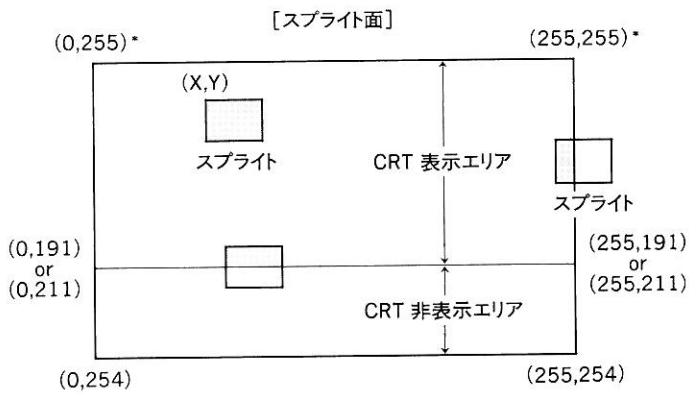
$$NYB = NY - N$$

注意 LINE で MAJ = 0 のときは、N = N - 1 となります。

6章 スプライト

V9938 は 32 個のスプライトを表示することができます。スプライトは 8×8 または 16×16 ドットの大きさです。また、横方向の 1 ドットの大きさは表示モードにかかわらず画面の $1/256$ で、画面上の任意の位置に表示することができます。

スプライトは独立した表示面を持っており、他の表示面のデータには影響を及ぼしません。



*スプライトの Y 座標の最上端の座標は 255 です。

図 4.47 スプライト表示面

V9938 には 2 種類のスプライト表示モードがあります。この 2 種類のモードは、画面表示モードによって自動的に選択されます。

- スプライトモード 1 SCREEN 1~3
 GRAPHIC 1、GRAPHIC 2、MULTI COLOR
- スプライトモード 2 SCREEN 4~8
 GRAPHIC 3~7

6.1 スプライトモード 1

スプライトモード 1 は TMS9918 のスプライトと同機能で、GRAPHIC 1、2、MULTI COLOR モード (SCREEN 1~3) で使うことができます。

6.1.1 スプライトモード 1 の特徴

32 個のスプライトには各々に #0~#31 までの番号がつけられており、番号の若い方が優先度が高くなります。CRT の 1 水平線上には優先度の高い順に 4 個表示され、5 番目以降のスプライトは表示されません。

スプライトモード 1 は TMS9918 のスプライトモードと互換性があります。

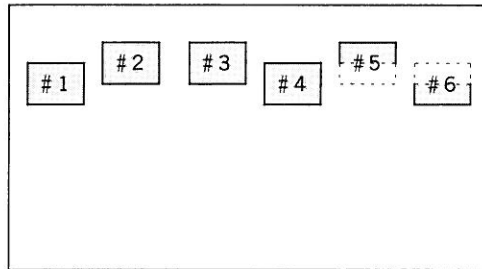


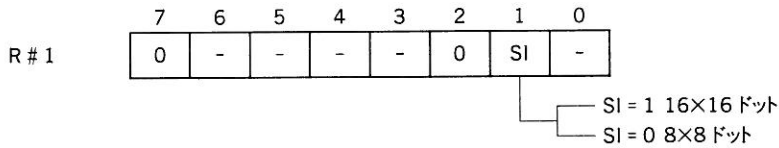
図 4.48 スプライトモード 1 の最大表示数

2 個のスプライトが衝突した (パターン の 1 の部分が重なった) ときは、ステータスレジスタ S #0 のビット 5 が「1」になるので、衝突の発生を知ることができます。

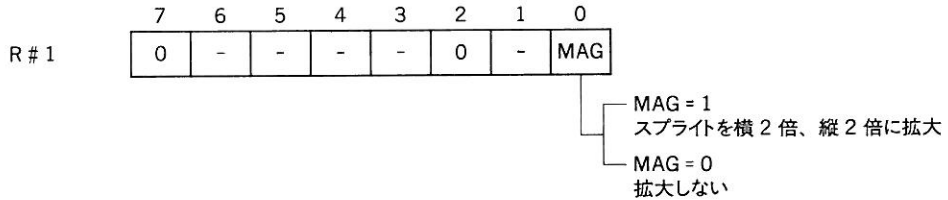
また、1 水平線上に 5 個以上のスプライトが並んだときには、ステータスレジスタ S #0 のビット 6 が「1」になり、S #0 の下位 5 ビットに 5 番目のスプライトの番号がセットされます。

6.1.2 スプライトモード 1 の表示

スプライトを表示するには、次の操作を行います。



1) スプライトの拡大……R#1のビット0



2) スプライトパターンジェネレータテーブルのセット

VRAM上のスプライトパターンジェネレータテーブルに、スプライトのパターンをセットします (#0~#255)。

3) スプライトアトリビュートテーブルのセット

VRAM上のスプライトアトリビュートテーブルに、スプライトの座標、スプライトのパターン番号と色をセットします (#0~#31)。

6.1.3 スプライトアトリビュートテーブル

32個あるスプライトの各々の表示位置(X、Y)、色、パターン番号などを指定するためのVRAMの領域です。この領域の先頭アドレスはR#5とR#11 (Sprite attribute table base address register) で指定します。スプライト1個につき4バイトのデータを書き込みます。

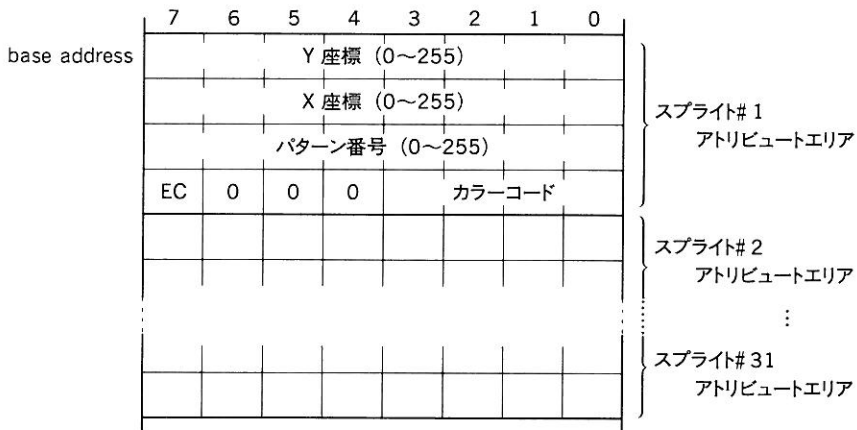


図 4.49 スプライトモード1のアトリビュートテーブル

1) Y座標 (0~255)

スプライトのY座標を指定します。

このY座標の値を208にすると、そのスプライトより優先度の低いスプライトは表示されません。例えば、#10のスプライトのY座標を208にすると、#10~#31のスプライトは表示されません。

2) X座標 (0~255)

スプライトのX座標を指定します。

3) パターン番号 (0~255)

スプライトパターンジェネレータテーブル上のパターン番号を指定します。スプライトのサイズが16×16のときは、4個のパターン番号を1個のスプライトに使用しますが、このうちのどれを指定してもかまいません。

スプライトのサイズを8×8に指定した場合は256個のパターンを、16×16に指定した場合は64個のパターンを指定することができます。

4) カラーコード (0~15)

スプライトパターンのビットが「1」の部分の色を指定します。「0」の部分は透明です。

5) EC (Early Clock)

このビットが「1」のとき、スプライトの表示は32ドット左にシフトされます。この機能を使うと、スプライトを画面の左端から1ドットずつ出現させることができます。

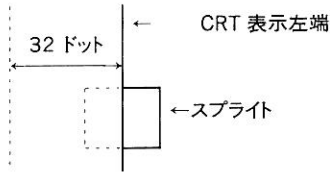


図 4.50 アトリビュートテーブルの EC ビット

6.1.4 スプライトパターンジェネレータテーブル

スプライトのパターン(形)を指定するための VRAM の領域です。この領域の先頭アドレスは R #6 (Sprite pattern generator table base address register) で指定します。

この領域には、8 バイトで 1 単位のパターンを最大 256 個書き込むことができます。これらのパターンには #0~#255 までの番号がつけられており、スプライトのサイズが 8×8 ドットのときはスプライト 1 個に対して 1 個のパターンが対応し、16×16 ドットのときはスプライト 1 個に対して 4 個のパターンが対応します。

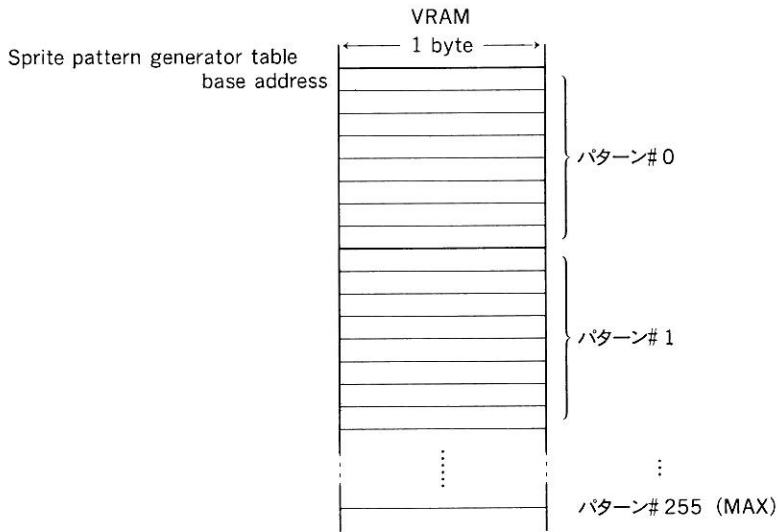


図 4.51 スプライトモード 1 のパターンジェネレータテーブル

6.1.5 スプライトパターンジェネレータテーブルデータセット例

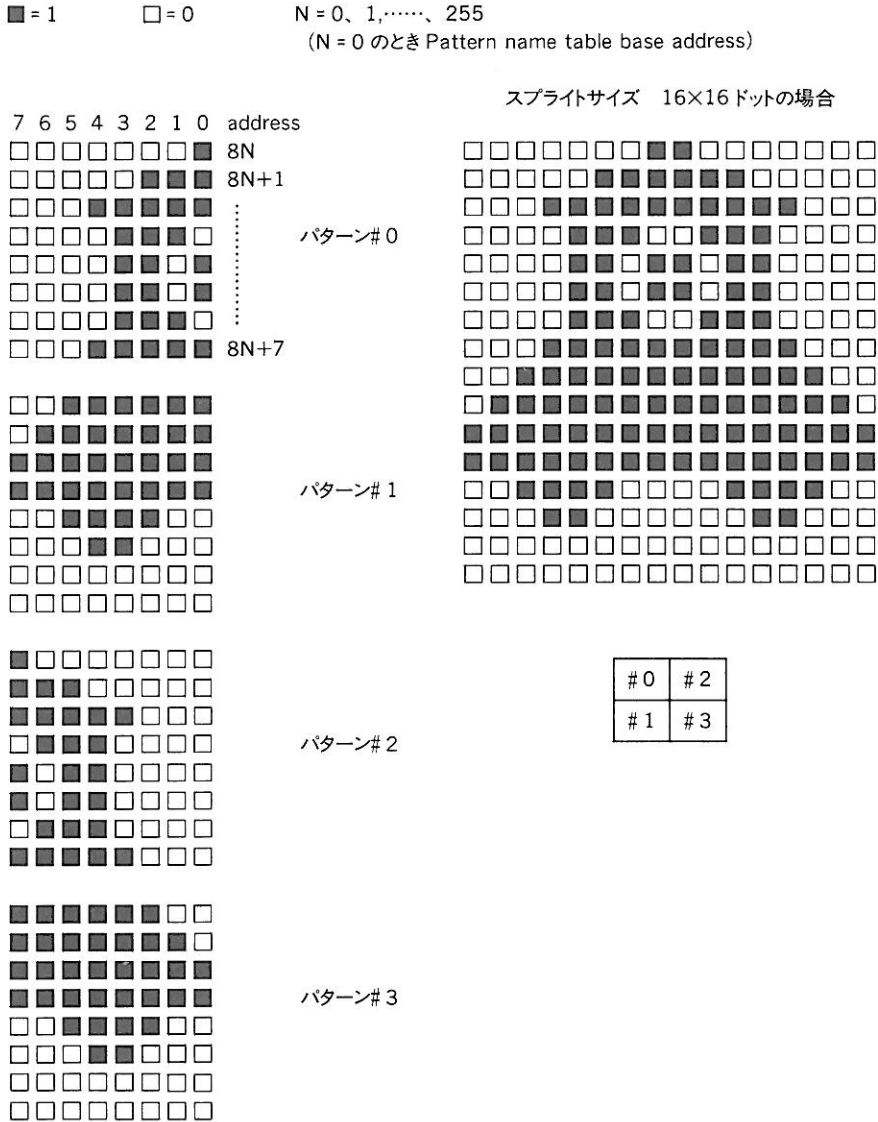


図 4.52 スプライトパターンジェネレータテーブルの例

スプライトのサイズを 16×16 とした場合、スプライトアトリビュートテーブルで指定するパターン番号は #0～#3 のどれでもかまいません。

6.2 スプライトモード2

スプライトモードは V9938 で新たに追加された機能で、GRAPHIC 3~7 モード (SCREEN 4 ~8) で使うことができます。

6.2.1 スプライトモード2の特徴

32個のスプライトには各々に#0~#31の番号がつけられており、番号の若い方が優先度が高くなります。CRTの1水平線上には優先度の高い順に8個表示され、9番目以降のスプライトの重なった部分は表示されません。

スプライトモード2は TMS9918 との互換性はありません。

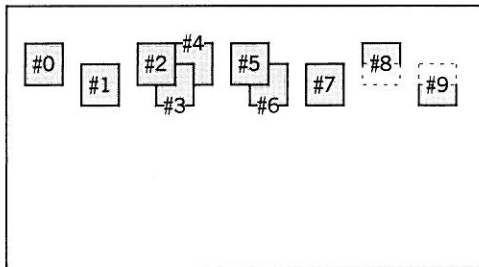


図 4.53 スプライトモード2の最大表示数

2個のスプライトが衝突した(透明でない部分が重なった)ときは、ステータスレジスタ S#0 のビット5が1になるので、衝突の発生を知ることができます。このときの衝突座標は、ステータスレジスタ#3~#5にセットされます。

1水平線上に9個以上のスプライトが並んだときは、ステータスレジスタ S#0 のビット6が1になり、S#0の下位5ビットに9番目のスプライトの番号がセットされます。

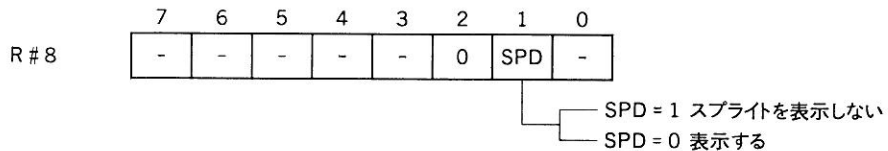
スプライトの色は1水平線ごとに指定できます。

スプライトカラーテーブルの CC ビットを指定することによって、スプライトを組み合わせさせて使えます。これにより横方向の色をより多く指定することができます。

6.2.2 スプライトモード2の表示

スプライトを表示するには、次の操作を行います。

- 1) スプライトのサイズ
スプライトモード1と同じです。
- 2) スプライトの拡大
スプライトモード1と同じです。
- 3) スプライトの表示



- 4) スプライトパターンジェネレータテーブルのセット
VRAM上のスプライトパターンジェネレータテーブルに、スプライトのパターンをセットします (#0~#255)。
- 5) スプライトカラーテーブルのセット
VRAM上のスプライトカラーテーブルに、スプライトの色、EC、CC、ICを1ラインずつセットします。
- 6) スプライトアトリビュートテーブルのセット
VRAM上のスプライトアトリビュートテーブルに、スプライトの座標、スプライトのパターン番号をセットします (#0~#31)。

6.2.3 VRAM 上の各テーブルの関係

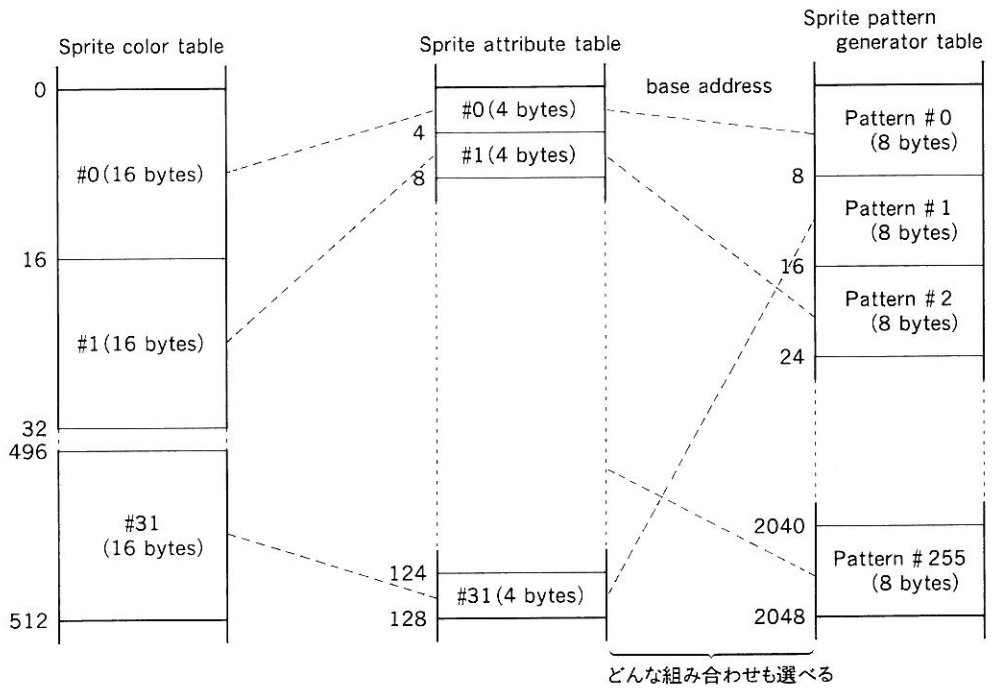


図 4.54 VRAM 上の各テーブルの関係

6.2.4 スプライトアトリビュートテーブル

32個あるスプライトの各々の表示位置 (X、Y)、パターン番号を指定するための VRAM の領域です。スプライト 1 個につき 3 バイトのデータを書き込みます。

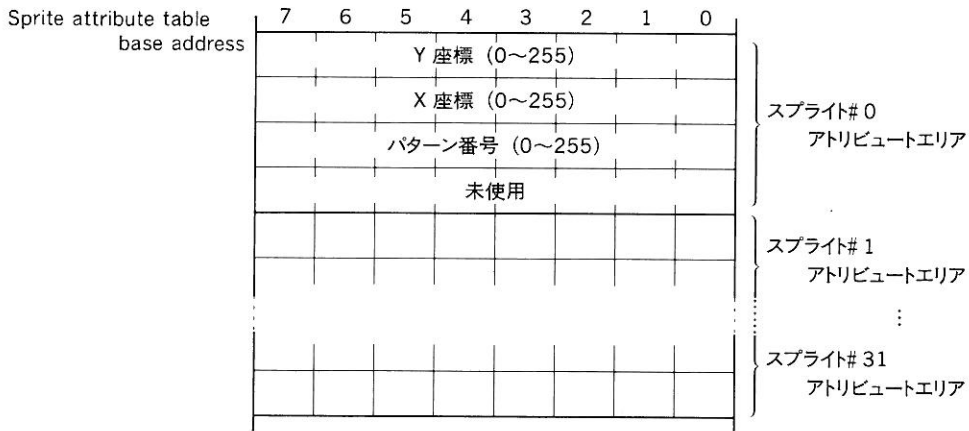


図 4.55 スプライトモード 2 のアトリビュートテーブル

1) Y 座標 (0~255)

スプライトの Y 座標を指定します。

この Y 座標の値を 216 にすると、そのスプライトより優先度の低いスプライトは表示されません。たとえば #10 のスプライトの Y 座標を 216 にすると、#10~#31 のスプライトは表示されません。

2) X 座標 (0~255)

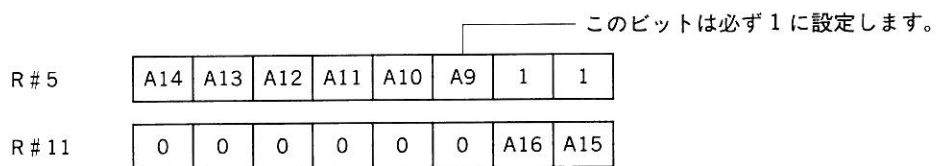
スプライトの X 座標を指定します。

3) パターン番号 (0~255)

スプライトパターンジェネレータテーブル上のパターン番号を指定します。スプライトのサイズが 16×16 のときは 4 個のパターン番号を 1 個のスプライトに使用しますが、このうちのどれを指定してもかまいません。

スプライトのサイズを 8×8 に指定した場合は 256 個のパターンを、 16×16 に指定した場合は 64 個のパターンを指定することができます。

Sprite attribute table base address register



6.2.5 スプライトパターングジェネレータテーブル

スプライトモード1と同じです。

6.2.6 スプライトカラーテーブル

スプライトモード2では、32個のスプライトの各ラインごとにパターンの1の部分の表示色(0の部分は透明に固定)、優先順位の有無、衝突検出の有無、EC (Early Clock) の指定をすることができます。

スプライトカラーテーブルの先頭アドレスは、スプライトアトリビュートテーブルの先頭アドレスの値から512を引いた値に自動的にセットされます。

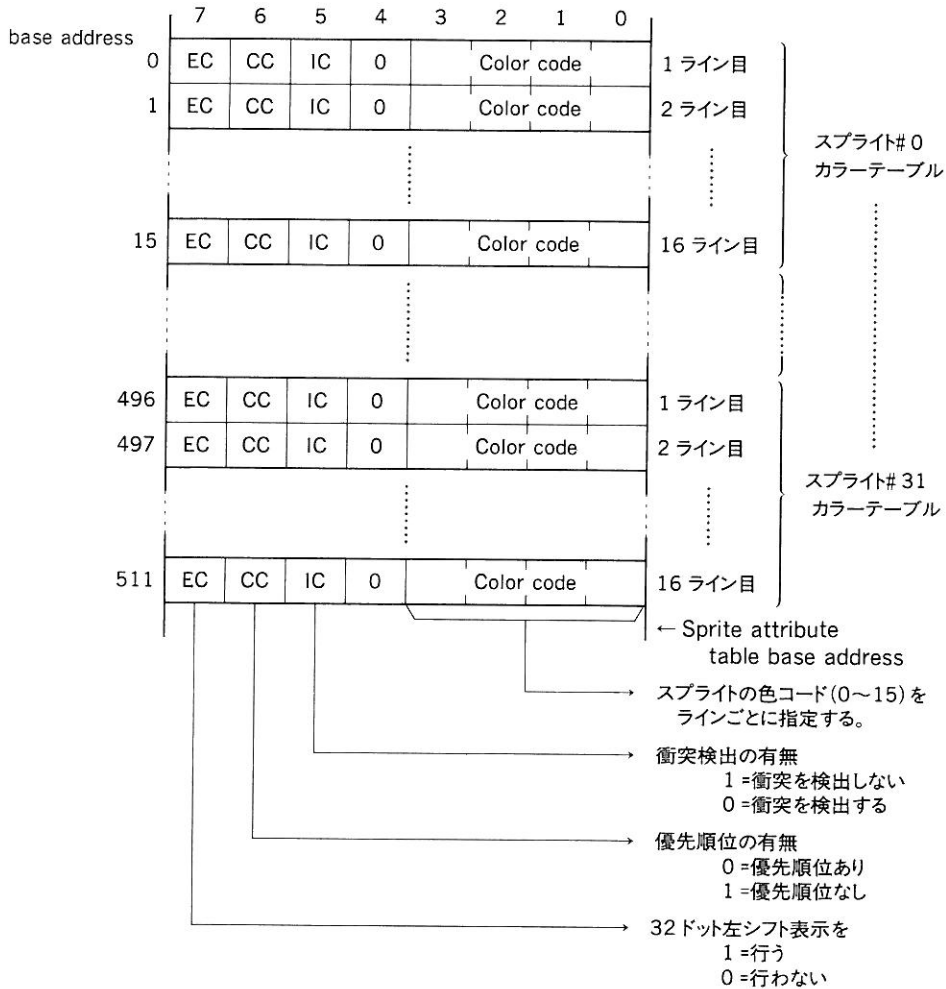


図 4.56 スプライトモード 2 のカラーテーブル

6.2.7 スプライトの組み合わせ

スプライトモード 2 において、カラーテーブルの CC ビットを「1」にすると、スプライトの優先順位をなくすことができます。

CC を「1」に指定した部分 (1 ラインごと) は、そのスプライトより若い番号で、かつ CC が「0」のスプライトの存在する水平線上にのみ表示されます。この様子を図示したものが図 4.57 です。

このときも、スプライトが 1 水平線上に 8 個を超えると、9 個目のスプライトは表示されません。

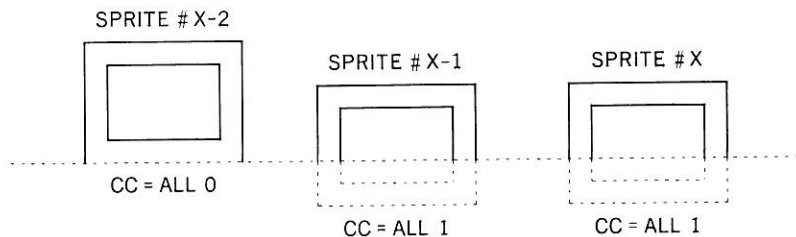
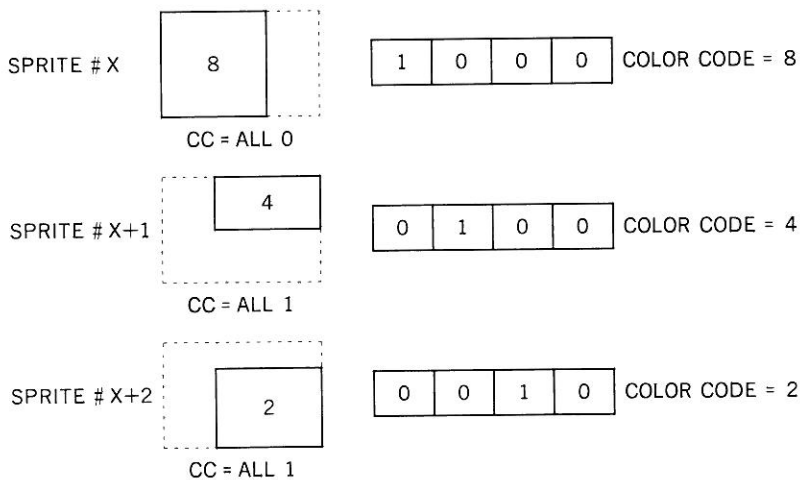


図 4.57 カラーテーブルの CC ビット

CC を「1」に指定した部分は、そのスプライトより若い番号で、かつ最もそのスプライトに番号が近いスプライトの CC = 0 の部分とは、たとえパターンが重なっても衝突の検出は行われません。また、重なった部分は各スプライトの色コードの OR をとって表示されます。

3 個のスプライトで 7 色を表示した例



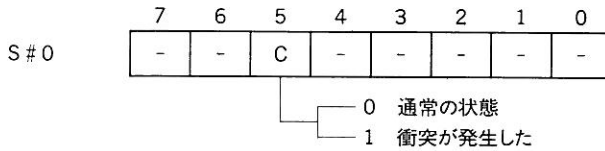
上記のスプライト 3 個を重ね合わせたもの



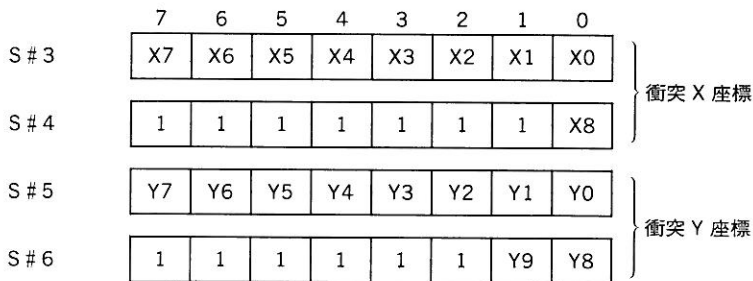
図 4.58 スプライトの重ね合わせ例

6.2.8 スプライトの衝突

スプライトの表示色が透明でなく、かつ CC が「0」の部分と重なると衝突が発生します。衝突が発生するとステータスレジスタ S#0 のビット 5 が「1」になります。このビットは S#0 を読み出すことによってリセットされます。



このとき、レジスタ R#8 のマウスフラグ (MS) またはライトペンフラグ (LP) がセットされていないならば (MSX ではセットされることはない)、ステータスレジスタ S#3~S#6 にスプライトの衝突座標がセットされます。



ステータスレジスタ S#3~S#6 は、S#5 を読み出したときリセットされます。また、ステータスレジスタ S#3~S#6 にセットされる値には、実際の衝突座標に対して次式のようなオフセットがつけられています。

実際の衝突座標 (XC, YC)

$$\begin{aligned}
 X &= XC + 12 & Y &= YC + 8 & X &= (S\#4, S\#3) \\
 & & & & Y &= (S\#6, S\#5)
 \end{aligned}$$

6.3 スプライトの色設定

GRAPHIC 7 (SCREEN 8)以外の表示モードでは、スプライトの色コードは各モードの色コードと共通です。つまり、スプライトの画面上での表示色はパレットレジスタの値で決定されます。

GRAPHIC 7 (SCREEN 8) モードにおいてはスプライトの表示色は固定されており、パレットレジスタの影響を受けません。GRAPHIC 7モードにおけるスプライトの色を表4.7に示します。

表 4.7 GRAPHIC 7 モード時のスプライトの色

COLOR CODE				GREEN			RED			BLUE		
C3	C2	C1	C0	G2	G1	G0	R2	R1	R0	B2	B1	B0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	1	0	1	0
0	1	0	0	0	1	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0	1	1	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0
1	0	0	0	1	0	0	1	1	1	0	1	0
1	0	0	1	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0
1	0	1	1	0	0	0	1	1	1	1	1	1
1	1	0	0	1	1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	0	0	0	1	1	1
1	1	1	0	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1

TP とスプライト

TP (レジスタ R#8 のビット 5) を操作することにより、カラーコード 0 は次のように定義され、スプライトもその影響を受けます。

- TP = 0 カラーコード 0 は透明として扱われる
 スプライトのカラーコード 0 で指定された部分は表示されず、他のスプライトと重なっても衝突は発生しません。
- TP = 1 カラーコード 0 はパレットレジスタで指定された色になる (GRAPHIC7 のときのみ、固定色 R = 0、G = 0、B = 0 にセットされる)
 スプライトのカラーコード 0 で指定された部分は表示され、他のスプライトと重なると衝突が発生します。

7章 その他の機能

7.1 グラフィック画面 2 ページの交互表示

V9938 では、次の方法でグラフィック画面 (GRAPHIC 4~GRAPHIC 7) 2 ページを交互に表示させることができます。交互に表示できるページの組み合わせは図 4.59 のとおりです。

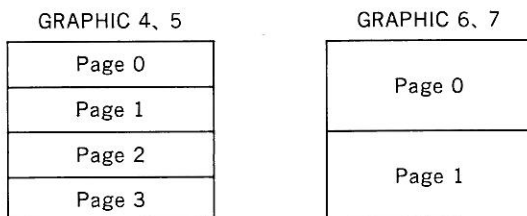


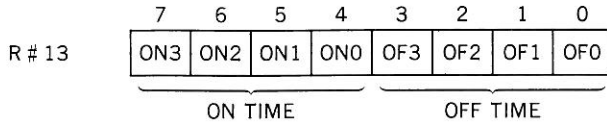
図 4.59 グラフィック画面の交互表示

7.1.1 レジスタ R#13 を使う方法

この方法によれば、各ページの表示時間をそれぞれ 166m 秒~2053m 秒の間で指定できます。

- 1) パターンネームテーブルベースアドレス (レジスタ R # 2) を奇数ページにセットする
- 2) レジスタ R # 13 に ONTIME (偶数ページの表示時間)、OFFTIME (奇数ページの表示時間) をセットする (セットする値については、TEXT 2 の項を参照して下さい。)

Blinking period register

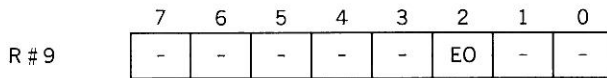


7.1.2 EOビットを使う方法

60Hz 周期で 2 ページを交互に表示させることができます。

- 1) パターンネームテーブルベースアドレス (レジスタ R#2) を、奇数ページにセットする
- 2) レジスタ R#9 のビット 2 (EO) を「1」にセットする

Mode register 3



7.2 インタレース表示

V9938 にはインタレース表示の機能があります。

7.2.1 第 1 フィールドと第 2 フィールドに同じページを表示させる場合

レジスタ R#9 の IL を「1」にセットする

7.2.2 第 1 フィールドに偶数ページ、第 2 フィールドに奇数ページを表示させる場合

1. レジスタ R#9 の IL を「1」に、R#9 の EO を「1」にセットする
2. パターンネームテーブルベースアドレス (レジスタ R#2) を奇数ページにセットする

7.3 GRAPHIC5 モードでのスプライト

GRAPHIC 5 モードの静止画は水平方向の解像度が512であり、カラーコードは2ビット/ピクセルです。

スプライトは静止画の1/2の解像度で表示されます。すなわちスプライトパターンは水平方向256ピクセルの解像度で、座標系はX=0~255となります。

図4.60のようなスプライトパターンは、図4.61のようにスプライトパターンジェネレータテーブルに書き込まれなければなりません。

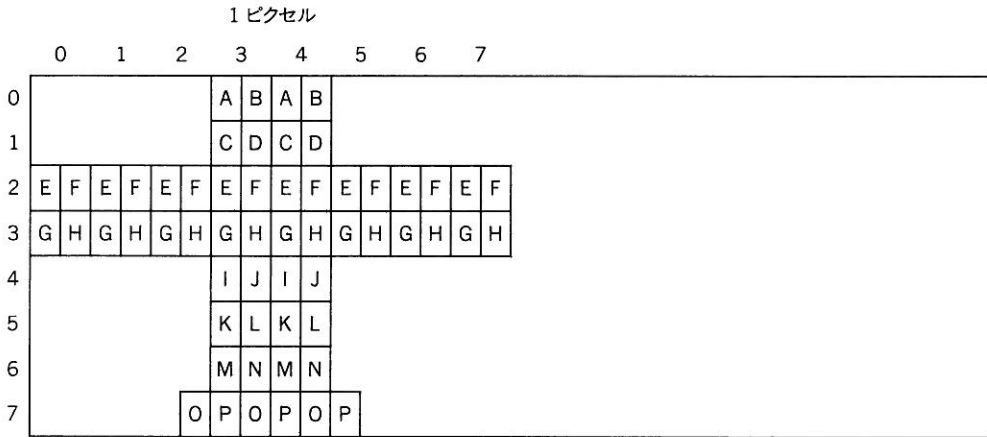


図4.60 GRAPHIC 5 モードでのスプライトパターン例

0	0	0	0	1	1	0	0	0
1	0	0	0	1	1	0	0	0
2	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1
4	0	0	0	1	1	0	0	0
5	0	0	0	1	1	0	0	0
6	0	0	0	1	1	0	0	0
7	0	0	1	1	1	1	0	0

図4.61 GRAPHIC 5 モードでのパターンジェネレータテーブル例

スプライトの色はスプライトカラーテーブルの内容によって決まります。カラーテーブルの上位4ビットは Early Clock などのコントロールとして使われます。下位4ビットは2ビットずつに分けられ、上位側の2ビットでスプライトの偶数ピクセルの色、下位側の2ビットで奇数ピクセルの色を表わします。このとき、スプライト色づけの解像度は512です。

図 4.62 のようにカラーテーブルに書き込まれている場合に、図 4.60 のように表示されます。

0		A	B
1		C	D
2		E	F
3		G	H
4		I	J
5		K	L
6		M	N
7		O	P

図 4.62 GRAPHIC 5 モードでのカラーテーブル例

7.4 同期モードの選択

V9938 では、同期モードはレジスタ R#9 の S1 と S0 の 2 ビットで指定します。

Mode register 3

	7	6	5	4	3	2	1	0
R#9	LN	0	S1	S0	IL	EO	\overline{NT}	DC

表 4.8 同期モード

S1	S0	同期モード	$\overline{Y_s}$	用途
0	0	パソコン同期	常に V9938 を選択 (0)	V9938 の画面を表示
0	1	標準同期	表示画面の透明部分で発生	スーパーインポーズ、デジタイズなど
1	0	標準同期	常に外部信号を選択	外部信号の表示
1	1	予約	予約	予約

8章

V9958 の構成

「V9958」は、MSX2+用の VDP (Video Display Processor) として開発された VLSI で、TMS9918A (MSX の VDP) および V9938 (MSX2 の VDP) とソフトウェア上位互換性があります。V9958 は株式会社アスキーとヤマハ株式会社によって共同開発され、V9938 に加えて以下の特徴があります。

- YJK 方式により最大 19,268 色を同時に表示可能
- ハードウェアによる水平スクロール
- 全表示モードでコマンド機能を使用可能

この章では V9938 に対して変更・追加された機能を解説します。この章で説明のない部分については V9938 と同じ仕様です。

8.1 レジスタ

8.1.1 追加されたレジスタ

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2
R # 26	0	0	HO8	HO7	HO6	HO5	HO4	HO3
R # 27	0	0	0	0	0	HO2	HO1	HO0

注意

空ビットには、必ず「0」を設定してください。

リセット時には、すべてのビットは「0」に設定され、機能的には V9938 と同等になります。

8.1.2 変更されたレジスタ

Status register 1 の ID 番号を「2」とし、V9958 であることを表します。V9938 では「0」が返ってきます。

	7	6	5	4	3	2	1	0
S # 1	FL	LPS	0	0	0	1	0	FH
	ID #							

8.2 V9958 の機能

V9958 では V9938 を拡張して、新たな機能を加えています。追加された機能には以下のようなものがあります。

8.2.1 水平スクロール

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2
R # 26	0	0	HO8	HO7	HO6	HO5	HO4	HO3
R # 27	0	0	0	0	0	HO2	HO1	HO0

R # 25 MSK 0 = 左側 8 ドットはマスクされない(初期値)
 1 = 左側 8 ドットをマスクし、ボーダーカラーを表示する
 R # 27 の値が「0」の場合は、マスクする必要はありません (GRAPHIC 5、6 では 16 ドットのマスクになる)。

SP2 0 = 水平方向画面サイズを 1 ページとする(初期値)
 スクロールは 1 ページ内で行われ、画面上には左側の隠れた部分が右側に表れます。
 1 = 水平方向画面サイズを 2 ページとする
 スクロールは 2 ページで行われ、スクロールによって裏のページが表れます。

第4部 VDP

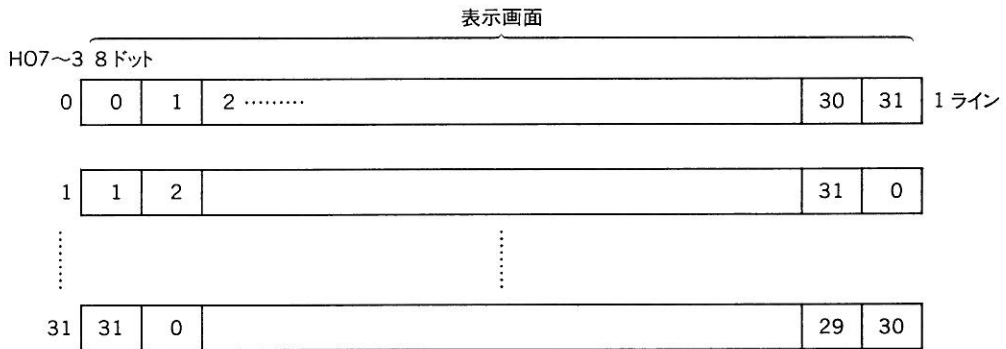
HO8~HO0 静止画の水平方向スクロール量をドット単位で設定する (GRAPHIC 5、6 は2ドット単位のスクロールになる)

水平スクロール時の VRAM アクセスは、8ドット単位で行われ、R#27の値が「0」以外の場合は、表示用のデータが不定になります。この不定のデータを表示させないためにマスクが必要になります。

HO8~HO3 に対する表示画面

左方向へ、8ドット単位 (GRAPHIC 5、6 は16ドット単位) で設定値だけシフトします。

SP2 = 0 のとき



注意 HO8は無視されます。

SP2 = 1 のとき

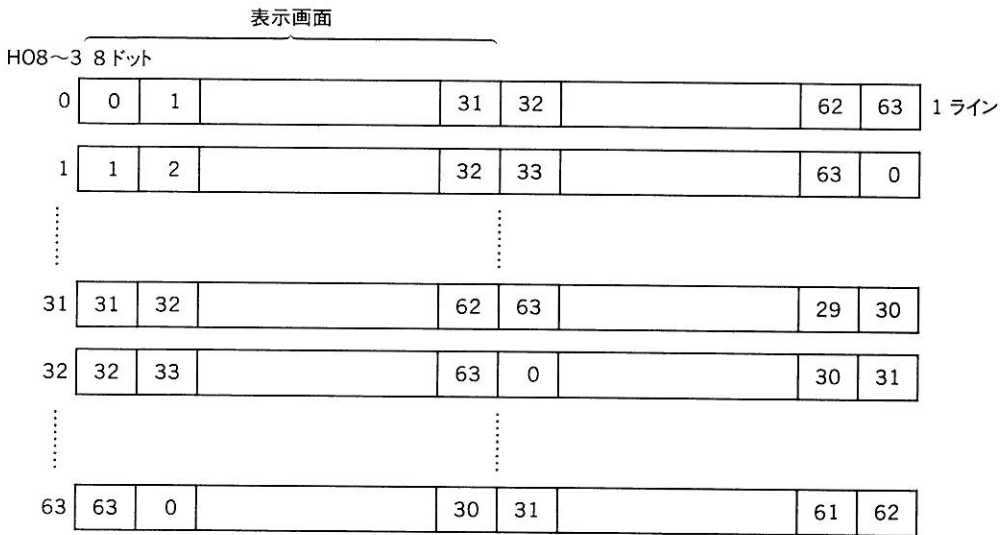


図 4.63 H08~H03 に対する表示画面

注意

SP2 = 1 のときは、パターンネームテーブルの先頭アドレスは、奇数ページが選択されるように設定します。

各テーブルのベースアドレスは次のようになります。

パターンネームテーブル(PNT)	0~31 偶数ページ
	32~63 奇数ページ
パターンジェネレータテーブル(PGT)	ベースアドレス設定値
	スクロールによって変化しない
カラーテーブル(CT)	ベースアドレス設定値
	スクロールによって変化しない

H02~H00 に対する表示画面

右方向へ、1ドット単位 (GRAPHIC 5、6 は、2ドット単位) で設定値だけシフトします。

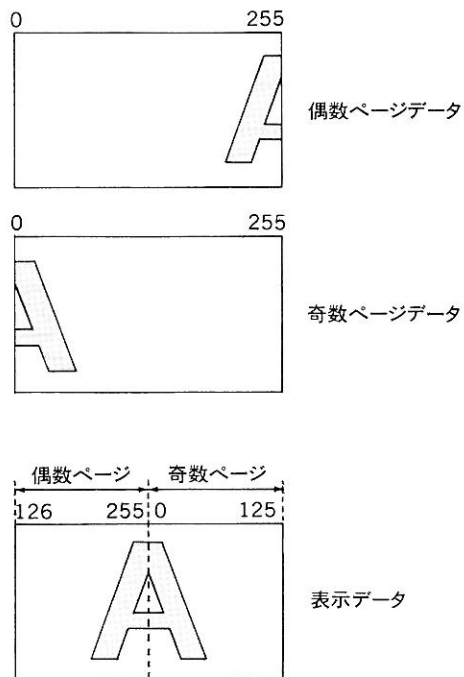


図 4.64 2画面水平スクロールの例

図 4.64 のように、左へ 126 ドットスクロールするときは、以下のようになります。

R # 26 = 128 = 8 × 16 (左へ 128 ドット)

R # 27 = 2 (右へ 2 ドット)

8.2.2 ウェイト機能

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

WTE 0 = ウェイト機能を無効にする (初期値)

V9938 と同様に機能します。

1 = ウェイト機能を有効にする

CPU が VRAM をアクセスした際に、V9958 の VRAM アクセスが完了するまで、全ての V9958 ポートへのアクセスに対してウェイト信号を発生します。

レジスタとカラーパレットへのアクセス未完了およびコマンドのデータレディによるウェイト機能はありません。

注意

ウェイト機能はMSXでは使っていません。

このビットを「1」にするときは、その前にダミーのVRAMアクセスを行って下さい。

8.2.3 コマンド機能

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

CMD 0 = GRAPHIC 4~7 の表示モードでのみコマンド機能を有効にする (初期値)
 1 = 全表示モードにおいてコマンド機能を有効にする
 GRAPHIC 4~7 モードでは、V9938 と同様に機能し、それ以外の表示モードでは、GRAPHIC 7 モードとして動作します。したがって、GRAPHIC 4~7 モード以外でコマンドを使用するときには、パラメータは GRAPHIC 7 モードの X、Y 座標系で設定します。

8.2.4 YJK 方式データの表示機能

	7	6	5	4	3	2	1	0
R#25	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2

YJK 0 = VRAM上のデータをRGB方式。(例えば、GRAPHIC 7のときは各3、3、2ビット)として扱う (初期値)。

スプライトの表示色は従来どおりです。

1 = VRAM上のデータをYJK方式とみなし、これをRGB信号 (各5ビット) に変換し、アナログ出力する。

スプライトの表示色にはパレットが有効になります。

YAE YJK 方式のデータフォーマットの選択

以下は、YJK方式のデータフォーマットです。連続した4ドットをグルーピングして表わします。

1) アトリビュートがない場合 (YAE = 0) (初期値)

	C7	C6	C5	C4	C3	C2	C1	C0	
1ドット			Y ₁				KL		□は1ドット分のカラー情報です。
1ドット			Y ₂				KH		
1ドット			Y ₃				JL		
1ドット			Y ₄				JH		

図 4.65 アトリビュートなしのYJK データフォーマット

YJK 方式のデータは連続した4ドット分の情報でグループ分けされており、

Y₁、KL、KH、JL、JH で先頭1ドットのカラー情報

Y₂、KL、KH、JL、JH で2ドット目のカラー情報

Y₃、KL、KH、JL、JH で3ドット目のカラー情報

Y₄、KL、KH、JL、JH で4ドット目のカラー情報

となります。

K と J の値は4ドットに共通で使われます。Y の値は1ドットごとに指定します。

2) アトリビュートがある場合 (YAE = 1)

	C7	C6	C5	C4	C3	C2	C1	C0	
1ドット			Y ₁		A	KL		A はアトリビュート	
1ドット			Y ₂		A	KH			
1ドット			Y ₃		A	JL			
1ドット			Y ₄		A	JH			

図 4.66 アトリビュートありのYJK データフォーマット

A = 0 YAE = 0 の場合と同じ (初期値)

Y、J、K は全て YJK 方式のデータとなります。□は1ドット分のカラー情報です。A のビットは無視されます。

A = 1 Y1~Y4 はそれぞれカラーコードとなり、カラーパレットを通して RGB 出力されます。J と K は YJK 方式のデータとなります。

3) YJK と JAE の組み合わせ

YJK	YAE	VRAM データ
0	×	従来のカラーパレット経由
1	0	YJK → RGB 変換テーブル経由
	1	A = 0 ……YJK → RGB 変換テーブル経由 A = 1 ……カラーパレット経由

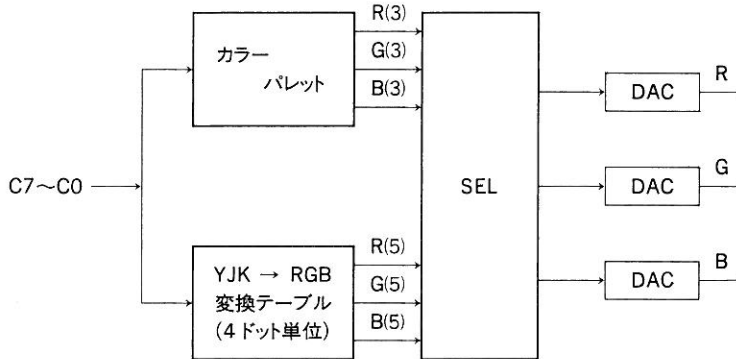


図 4.67 YJK と JAE の組み合わせ

8.2.5 V9958 では削除された機能

1. コンポジットビデオ出力
2. マウス・ライトペンインターフェイス

この変更に伴い、次の内部レジスタのビット（□の部分）が意味を持たなくなります。これらのレジスタに値を書き込む場合には、これらのビットに必ず「0」を書き込んで下さい。

Status Register 1

	7	6	5	4	3	2	1	0
S #1	FL	LPS	0	0	0	1	0	FH

FL ライトペン光検出フラグ

LPS ライトペン・マウススイッチ

Mode register 0

	7	6	5	4	3	2	1	0
R#0	0	DG	IE2	IE1	M5	M4	M3	0

IE2 ライトペンによる割込み
 (MSX2 は V9938 のライトペン機能は使用していない)

Mode register 2

	7	6	5	4	3	2	1	0
R#8	MS	LP	TP	CB	VR	0	SPD	BW

MS マウスで使用
 LP ライトペンで使用
 (MSX2 は V9938 のマウス・ライトペン機能は使用していない)

9章

YJK 方式

9.1 YJK 方式とは

MSX2 までは、色の情報を扱うために、色を赤、緑、青の信号に分解してその割合を数値で表現する方式 (RGB 方式) が採用されていました。これに対して、YJK 方式とは色情報を輝度成分 (Y) と色相成分 (J と K) に分解して VRAM 上に置き、表示するときに RGB 方式に変換するものです。

V9958 では VRAM の容量を V9938 と同じにするために、輝度情報については、画面 1 ドットに対して 5 ビット (0~31) 設定できるようになっていますが、色相情報については、画面 4 ドットに対して 6 ビット (-32~+31) しか設定できません (SCREEN 12 の場合)。

YJK 方式は、デジタイズ処理の際も 4 ドットごとの J、K の値をどうするかなどの点で難しい面がありますが、最大の特徴は VRAM 容量を変えずに色数が増えたことです。YJK 方式は、ほとんど共通の R と G を JK とし、これに B および明るさのグラデーションを付ける Y を加えて表示する方式なので、同じ色調のグラデーション (自然界に多い) や、白や灰色などの無彩色の背景上にある物体などを表現する能力が高く、真っ青な背景に斜めの明るい黄色や赤の細い線などの人工的な物体の表現能力はあまり高くないという性質があります。つまり、人工的なコンピュータグラフィックスなどには向かない場合もありますが、自然物の表示には相当の効果があります。

19,268 色という数字は、この YJK の取り得る全ての値で導かれる RGB の組み合わせを計算して、重複しているものを除いたものです。

9.2 YJK と RGB の変換

9.2.1 YJK から RGB への変換式

$$R = Y + J$$

$$G = Y + K$$

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

9.2.2 RGB から YJK への変換式

$$Y = \frac{1}{2}B + \frac{1}{4}R + \frac{1}{8}G$$

$$J = R - Y$$

$$K = G - Y$$



10章

V9958 の画面モード

MSX BASIC 3.0 では、YJK モードおよび YJK / RGB 混在モード (以下、YJK モードと YJK / RGB 混在モードを総称して、自然画モードと呼ぶ) として、SCREEN 10~12 を設けていますが、V9958 ではそれに相当する GRAPHIC モードはありません。

V9958 を自然画モードにするためには、表示モードを GRAPICH 7 にした上で、R # 25 の bit 3、4 を操作します。詳しくは、10.4 のサンプルプログラムをご参照下さい。

10.1 SCREEN 9

SCREEN 9 は韓国バージョンの MSX でハングル文字を表示するために使われています。したがって、日本製の MSX では使用しません。

10.2 SCREEN 10、11

10.2.1 特徴

- 解像度 横 256×縦 212 ドット (または縦 192 ドット) の YJK・RGB 混在モード
- 表示色 12,499 色同時 (全画面)
- スプライト モード 2 (パレットが使用可能)
- 1 画面に必要な VRAM 容量 128K バイト (2 画面)

10.2.2 関係するレジスタと VRAM の領域

GRAPHIC 7 (SCREEN 8) と同じです。BASIC では SCREEN 10 または SCREEN 11 として扱います。

10.2.3 初期設定

R # 25 以外は、GRAPHIC 7 (SCREEN 8) と同じです。

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	1	1	WTE	MSK	SP2

□は表示モード設定用のビットを YJK・RGB 混在モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドット (BASIC ではサポートしない) に設定されます。その他のビットは任意に設定します。

1) パターンネームテーブルの設定

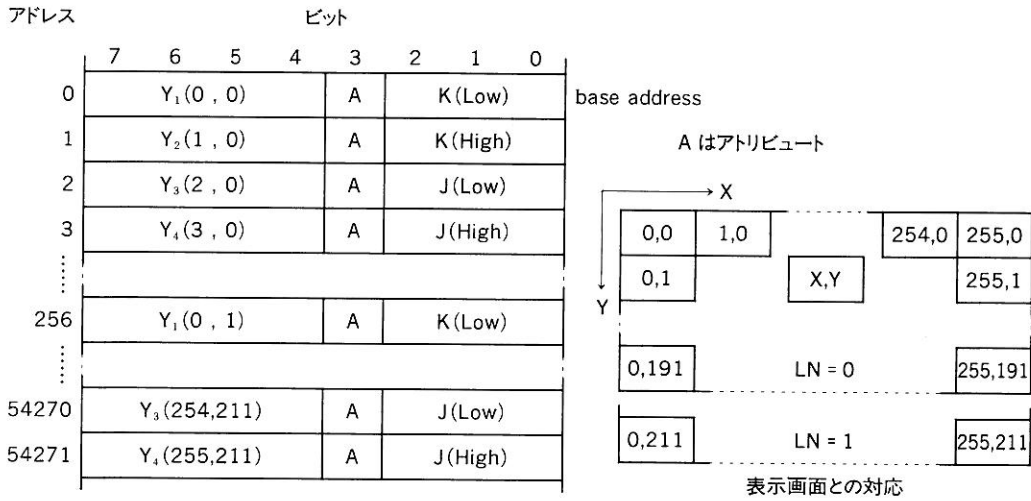


図 4.68 SCREEN 10、11 モードのパターンネームテーブル

10.2.4 SCREEN 10、11 モードの VRAM マップ

GRAPHIC 7 (SCREEN 8) と同じです。

10.2.5 SCREEN 10 と SCREEN 11 の違い

VDP 上では SCREEN 10 と 11 は何の違いもありません。アトリビュートが「0」の画素は YJK 方式で表示され、アトリビュートが「1」の画素は Y の 4 ビットが色コードとして扱われます。

MSX-BASIC 3.0 ではこの YJK / RGB 混在モードを更に 2 つのスクリーンモードとして扱います。

SCREEN 10 では、

- LINE、PSET などの描画コマンド
- PRINT #文による文字表示
- 漢字モードにおけるグラフィック画面への PRINT 文

などで色指定を行うときは、SCREEN 0 から 7 までで使用する通常の色コードである 0~15 だけが使用できます。これ以外の値は Illegal function call となります。この色指定で RGB 以外の部分、すなわち JK の成分 (下位 3 ビット) はそのままにして、描画されない画素の表示が影響され

第4部 VDP

ることを防ぎます。これによって、背景に自然画をロードしておいてその上に線を引くようなことが可能となります。

例えば、色コード 15 のドットはこの画面では Y 成分の部分（上位 4 ビット）およびアトリビュートビットを 1 にすることになり、VRAM 上では $\&HF8$ となりますが、SCREEN 10 のときは描画に際して、このような変換を意識する必要はありません。

一方、SCREEN 11 では、色コードの扱いは VRAM 上の値そのものとなり、色コードは 0~255 を指定できます。

10.3 SCREEN 12

10.3.1 特徴

- 解像度 横 256×縦 212 ドット（または縦 192 ドット）の YJK モード
- 表示色 19,268 色同時（全画面）
- スプライト モード 2（パレットが使用可能）
- 1 画面に必要な VRAM 容量 128K バイト（2 画面）

10.3.2 関係するレジスタと VRAM の領域

GRAPHIC 7（SCREEN 8）と同じです。BASIC では SCREEN 12 として扱います。

10.3.3 初期設定

R # 25 以外は、GRAPHIC 7（SCREEN 8）と同じです。

	7	6	5	4	3	2	1	0
R # 25	0	CMD	VDS	0	1	WTE	MSK	SP2

□は表示モード設定用のビットを YJK モードにセットした例です。この表示モードでは、LN = 1 のとき縦 212 ドット、LN = 0 のとき縦 192 ドット（BASIC ではサポートしない）に設定されます。その他のビットは任意に設定します。

1) パターンネームテーブルの設定

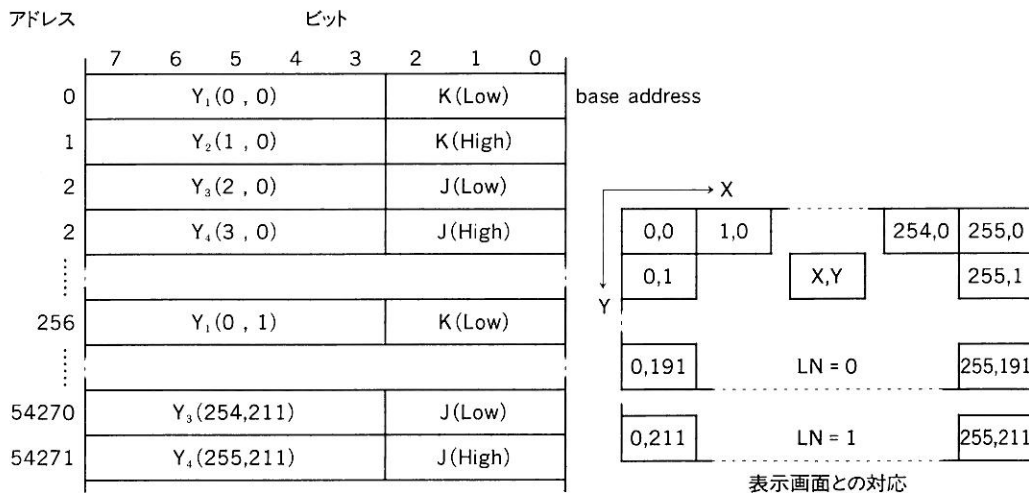


図 4.69 SCREEN 12 モードのパターンネームテーブル

10.3.4 SCREEN 12 モード VRAM マップ

GRAPHIC 7 (SCREEN 8) と同じです。

10.4 SCREEN 10、11、12 の設定方法

SCREEN 0～8 の切り換えは、モードレジスタに値を設定することによって行いますが、自然画モード (SCREEN 10～12) については、別の方法を使います。なぜならば、自然画モードとは V9958 にとって、GRAPHIC 7 モードで R # 25 の YEA ビットと YJK ビットを設定した状態だからです。

したがって、GRAPHIC モードが 7 の状態で以下のプログラムを実行すれば、SCREEN モードを 10、11、12 に設定することができます。

MODE 変数の bit 5 の SET、RESET は ROM 内のルーチン (GRPPRT、NVBXLN など) を使用する可能性がある場合にのみ必要で、アプリケーションが VDP を直接操作する場合には必要ありません。もちろん、そのような場合には、アトリビュートの操作はアプリケーションプログラムから行わなければなりません。

また、BASIC で行っているようなサービス (SCREEN12 から SCREEN10 に切り換えた時アトリビュートを落としている) は、以下の中には含まれていません。

リスト 4.1 SCREEN 10、11、12 の設定方法

```

mode      equ      0fafch
rg25sa    equ      0fffah          ; register # 25 save area
wrtvdp    equ      47h

screen10:
    ld      hl,mode
    res    5, (hl)
    ld      a, (rg25sa)
    or     00011000b
    ld      b,a
    ld      c,25
    jp     wrtvdp

screen11:
    ld      hl,mode
    set    5, (hl)
    ld      a, (rg25sa)
    or     00011000b
    ld      b,a
    ld      c,25
    jp     wrtvdp

screen12:
    ld      a, (rg25sa)
    and    11100111b
    or     00001000b
    ld      b,a

```

第 4 部 VDP

```
ld    c,25  
jp    wrtvd  
end
```



索引

(< > 内は Volume 2 のページ数を示します)

■ 記号

!	82
#	82
\$	82
%	81
'	77
*	349
,	76
-	76
.	76
:	76
;	77
<	85
<=、=<	85
<>、><	85
=	85
>	85
>=、=>	85
?	77, 349
¥	84
(空白)	77

■ 数字

0 でのわり算	85
1 ビットサウンドポート	<40>
24 ドット漢字プリンタ	<523>

■ A

ABS	98
ADPCM 音色データ	<305>
AND	86
ASC	98
ATN	99
AUTO	99

■ B

BASE	83, 100
BASIC	74, 355
ROM 化	<24>
エラーメッセージ	247
格納形式	256
BEEP	100
BIN\$	101
BIOS	310, <582>
MAIN ROM	<583>
SUB ROM	<627>
BLOAD	101

BLTDM	288
BLTDV	285
BLTMD	287
BLTMV	283
BLTVD	284
BLTVM	281
BLTVV	280
BREAKX	<61>
BSAVE	102

■ C

CALL	103
CALL AKCNV	103
CALL ANK	103
CALL APEEK	<286>
CALL APOKE	<286>
CALL APPEND MK	<320>
CALL AUDIO	<287>
CALL AUDREG	<228>, <289>
CALL BGM	<228>, <289>
CALL CLS	104
CALL COM GOSUB	<104>, <153>
CALL COMBREAK	<103>, <153>
CALL COMDTR	<104>
CALL COMHELP	<105>, <154>
CALL COMINI	<106>, <155>
CALL COMOFF	<109>, <158>
CALL COMON	<110>, <159>
CALL COMPROTocol	<159>
CALL COMSTAT	<111>, <161>
CALL COMSTOP	<113>, <163>
CALL COMTERM	<113>, <164>
CALL CONT MK	<320>
CALL CONVA	<303>
CALL CONV P	<304>
CALL COPY PCM	<304>
CALL DIAL	<169>
CALL DIALC	<171>
CALL DTMF	<172>
CALL FORMAT	104
CALL INMK	<314>
CALL JIS	105
CALL KACNV	105
CALL KANJI	106
CALL KEXT	108
CALL KEY ON/OFF	<315>
CALL KINSTR	108
CALL KLEN	109
CALL KMID	109

- CALL KNJ 110
 CALL KTYPE 110
 CALL LINESEL <173>
 CALL LOAD PCM <306>
 CALL MEMINI 111
 CALL MFILES 111
 CALL MKILL 112
 CALL MK PCM <315>
 CALL MK STAT <321>
 CALL MK TEMPO <316>
 CALL MK VEL <317>
 CALL MK VOICE <317>
 CALL MK VOL <318>
 CALL MNAME 112
 CALL MUSIC <229>
 CALL NET GOSUB <117>
 CALL NETCARRIER <174>
 CALL NETCONFIG <175>
 CALL NETHOOK <178>
 CALL NETINI <179>
 CALL NETMODEM <180>
 CALL NETOFF <181>
 CALL NETON <182>
 CALL NETSPK <183>
 CALL NETSTAT <183>
 CALL NETSTOP <185>
 CALL PALETTE 112
 CALL PCM FREQ <307>
 CALL PCM VOL <307>
 CALL PITCH <290>, <320>
 CALL PLAY <235>, <294>
 CALL PLAY MK <321>
 CALL PLAY PCM <308>
 CALL REC MK <322>
 CALL RECMOD <323>
 CALL REC PCM <309>
 CALL SAVE PCM <310>
 CALL SET PCM <311>
 CALL SJIS 113
 CALL STOPM <235>, <295>
 CALL SYNTH <295>
 CALL SYSTEM 113
 CALL TEMPER <236>, <296>
 CALL TRANSPOSE <237>, <297>
 CALL VOICE <237>, <298>
 CALL VOICE COPY <242>, <302>
 CALLF <11>
 CALSLT <10>
 CDBL 114
 CHGET <56>
 CHGSND <41>
 CHR\$ 114
 CHRGR 263
 CHSNS <55>
 CINT 115
 CIRCLE 115
 CLEAR 116
 CLOAD 117
 CLOAD? 118
 CLOCK-IC <75>
 MODE レジスタ <77>
 RESET レジスタ <78>
 TEST レジスタ <78>
 アクセス <79>
 時間の設定 <79>
 CLOSE 119, <120>, <189>
 CLS 119
 CNVCHR <57>
 COLOR 120
 COLOR SPRITE 123
 CONT 125
 COPY 125, 356
 COS 129
 CRUNCH 269
 CSAVE 130
 CSNG 131
 CSRLIN 83, 131
 CVD 132
 CVI 132
 CVS 132
■ D
 DATA 132
 DATE 360
 DEF FN 133
 DEFUSR 135
 DEFDBL 134
 DEFINT 134
 DEFSTR 134
 DEFSTR 134
 DEFSTR 134
 DEL 362
 DELETE 135
 DEVICE <22>
 DIM 136
 DIR 363
 DPB 410
 DRAW 137
 DSKF 139

DTA 406
 DVINFB <132>, <202>
 DVTYPE <133>, <202>

■ E

ENASLT <11>
 END 139
 EOF 140, <127>, <196>
 EQV 86
 ERASE 141, 364
 ERL 82, 141
 ERR 83, 141
 ERROR 142
 EXBRSA <17>
 EXP 143
 EXPTBL <17>
 EXTROM <14>

■ F

FAT 411
 FCB 402
 FDD ROM での初期化 66
 FIELD 143
 FILES 144
 FIX 145
 FORMAT 364
 FOR~NEXT 145
 FRE 146
 FRESTR 266
 FRMEVL 264
 FRMQNT 265

■ G

GET 147
 GET DATE 147
 GET TIME 148
 GETBYT 266
 GICINI <38>
 GOSUB 148
 GOTO 149
 GRAPHIC 1 モード 486
 GRAPHIC 2 モード 494
 GRAPHIC 3 モード 494
 GRAPHIC 4 モード 503
 GRAPHIC 5 モード 508
 GRAPHIC 6 モード 514
 GRAPHIC 7 モード 519
 GTPAD <73>
 GTPDL <72>

GTSTCK <70>
 GTTRIG <71>

■ H

HEX\$ 150
 HMMC 527
 HMMM 532
 HMMV 534

■ I

I/O ポート 312
 I/O マップ <669>
 ID <20>
 IF~GOTO~ELSE 150
 IF~THEN~ELSE 150
 IMP 86
 INIFNK <61>
 INIT <20>
 INKEY\$ 152
 INLINE <59>
 INP 152
 INPUT 153
 INPUT # 154, <120>, <190>
 INPUT\$ 155, <127>, <196>
 INSTR 155
 INT 10, 156
 INTERVAL 156

■ K

KEY 157, 159, 200
 KEY LIST 158
 KILBUF <57>
 KILL 160

■ L

LEFT\$ 161
 LEN 161
 LET 162
 LFILES 144
 LINE 162, 546
 LINE INPUT 164
 LINE INPUT # 165, <121>, <191>
 LIST 166
 LLIST 166
 LMCM 539
 LMMC 536
 LMMM 542
 LMMV 544
 LOAD 166, <117>, <186>

- LOC 167, <128>, <197>
 LOCATE 168
 LOF 168, <129>, <198>
 LOG 169
 LPOS 83, 169
 LPOUT <65>
 LPRINT 170
 LPRINT USING 170
 LPTSTT <66>
 LSET 171
- M**
 MAIN ROM 59
 Math-Pack 292
 MAXFILES 172
 MERGE 172, <117>, <187>
 MID\$ 173, 174
 MKD\$ 174
 MKI\$ 174
 MKSS 174
 MK 記録 <319>
 MML <232>, <292>
 MOD 84
 MODE 366
 MOTOR 175
 MSX-AUDIO 14, <277>
 LSI <430>
 MBIOS <348>
 ハードウェア <277>
 拡張 BASIC <283>
 拡張 BIOS <324>
 MSX-DOS 335
 エラーメッセージ 387
 起動 395
 特殊キー 372
 入出力 401
 ファンクションコール 416
 プログラムの起動 397
 プログラムの終了 400
 メッセージ 378
 MSX-JE <473>
 MSX MODEM <146>
 ハードウェア <146>
 拡張 BASIC <149>
 拡張 BIOS <199>
 MSX-MUSIC 14, <221>
 FM-BIOS <244>
 FM-BIOS のデータ構造 <250>
 FM-BIOS 使用上の注意 <253>
- 拡張 BASIC <225>
 ハードウェア <221>
 MSX-VIDEO 445
 MULTI COLOR モード 479
- N**
 NAME 175
 NEW 176
 NEWSTT 270
 NEXT 145
 NMI 10
 NOT 86
 NTMSXP <65>
- O**
 OCT\$ 176
 ON ERROR GOTO 177
 ON GOSUB 178
 ON GOTO 178
 ON INTERVAL GOSUB 179
 ON KEY GOSUB 180
 ON SPRITE GOSUB 181
 ON STOP GOSUB 182
 ON STRIG GOSUB 182
 OPEN 183, <122>, <191>
 OPLL <255>
 OR 86
 OUT 185
 OUTDLP <66>
- P**
 PAD 186
 PAINT 187
 PAUSE 367
 PDL 188
 PEEK 189
 PINLIN <58>
 PLAY 190, 194, <231>, <291>
 PLAY 文 BIOS 274
 POINT 195, 553
 POKE 196
 POS 83, 197
 PPI 43, 60
 PRESET 202
 PRINT 197
 PRINT USING 199
 PRINT # 198, <123>, <192>
 PRINT # USING 201, <125>, <193>
 PSET 202, 551

PSG 13, 44, <31>
 アクセス <37>
 レジスタ <32>
 PTRGET 267
 PUT 203
 PUT KANJI 203
 PUT SPRITE 204

■ R

RAWPRT <64>
 RDPSG <39>
 RDSLT <9>
 READ 206
 REDCLK <83>
 REM 206, 368
 REN 368
 RENAME 369
 RENUM 207
 RESTORE 208
 RESUME 209
 RETURN 209
 RGB 24
 RIGHTS\$ 210
 RND 211
 RS-232C <87>
 拡張 BASIC <99>
 拡張 BIOS <130>
 RSET 171
 RSLREG <12>
 RUN 211, <118>, <187>

■ S

SAVE 212, <119>, <188>
 SCREEN 213
 SCREEN 10 588
 SCREEN 11 588
 SCREEN 12 591
 SET ADJUST 215
 SET BEEP 216
 SET DATE 216
 SET PAGE 217
 SET PASSWORD 217
 SET PROMPT 218
 SET SCREEN 218
 SET SCROLL 219
 SET TIME 219
 SET TITLE 220
 SET VIDEO 220
 SGN 221

SHUTDOWN 329
 SIN 221
 SLTATR <18>
 SLTTBL <18>
 SLTWRK <18>
 SNSMAT <53>
 SOUND 222
 SPACES\$ 228
 SPC 228
 SPRITE 231
 SPRITE\$ 83, 229
 SQR 232
 SRCH 548
 STATEMENT <20>
 STB <485>
 STEP 145
 STICK 233
 STMOTR <50>
 STOP 234
 STOP キー <61>
 STR\$ 235
 STRIG 236, 237
 STRINGS\$ 238
 STRTMS <40>
 SUB ROM 63, <13>
 SWAP 238

■ T

TAB 239
 TAN 240
 TAPIN <47>
 TAPIOF <48>
 TAPION <47>
 TAPOOF <50>
 TAPOON <48>
 TAPOUT <49>
 TEXT <23>
 TEXT 1 モード 468
 TEXT 2 モード 473
 THEN 150
 TIME 82, 240, 369
 TO 126
 TROFF 241
 TRON 241
 TTB <487>
 TYPE 370
 TYPE A <523>
 TYPE B <523>

- U
 - USING 170, 199, 201
 - USR 241
 - USR 関数 259
- V
 - V9938 445
 - I/O ポート 448
 - コマンド 524
 - 画面構成 454
 - 画面モード 468
 - V9958 576
 - レジスタ 576
 - 画面モード 587
 - 削除された機能 583
 - VAL 242
 - VARPTR 243
 - VDP 83, 243, 319, 445
 - I/O アドレス 319
 - VERIFY 371
 - VJE-80 <519>
 - VPEEK 244
 - VPOKE 245
 - VRAM アクセス 452
 - VRAM マップ <660>
- W
 - WAIT 245
 - WIDTH 246
 - WRSLT <9>
 - WRTCLK <84>
 - WRTPSG <39>
 - WSLREG <12>
- X
 - XOR 86
- Y
 - Y8950 <430>
 - YJK 581, 585
 - YM2413 <255>
 - YMMM 530
- ア
 - アクセスレジスタ 461
 - イメージ印字 <555>
 - インタースロットコール <7>
 - インターフェイス 16
 - カセット 16
 - フロッピーディスク 17
 - プリンタ 18
 - 汎用入出力 19
 - インタレース表示 573
 - エスケープシーケンス <682>
 - 演算 83
 - 関係演算 85
 - 関数 87
 - 算術演算 84
 - 文字列演算 87
 - 優先順位 88
 - 論理演算 86
 - エンベロープパターン <36>
 - エンベロープ周期 <36>
 - オートインクリメントモード 450
 - オーバーフロー 85
 - オープン 404
 - 音色パラメータ <240>, <301>
 - 音色ライブラリ <239>, <298>
 - 音声出力 15
 - 音声ファイルの構造 <312>
 - 音程 <33>
 - 音量 <35>
- カ
 - カートリッジ 31, <19>
 - バス 35
 - ヘッド <19>
 - 拡張 BIOS <566>
 - 拡張 BIOS コール 310
 - 拡張 I/O ポート <672>
 - 拡張ステートメント 271, <20>
 - 拡張スロット 312, <3>
 - 拡張スロットセレクト信号 47
 - カセット <42>
 - 仮想端末入力インターフェイス <481>
 - カラーレジスタ 458
 - カラー番号 92
 - 漢字 ROM 305
 - 漢字ドライバ 302
 - 拡張 BIOS 303
 - 間接指定 449
 - キーボード 12, <52>
 - JIS 配列 13
 - キースキャン <52>
 - キー配列 <52>
 - バッファ <55>
 - マトリックス 12
 - マトリックス <54>

参考文献

- | | |
|-------------------------------|------|
| 「MSX テクニカルデータブック 1 増補改訂版」 | アスキー |
| 「MSX2 テクニカルハンドブック」 | アスキー |
| 「V9938 MSX-VIDEO テクニカルデータブック」 | アスキー |
| 「YM2413 アプリケーションマニュアル」 | ヤマハ |
| 「Y8950 アプリケーションマニュアル」 | ヤマハ |

お問い合わせについて

弊社では厳重に梱包した上、細心の注意を払って製品を発送しております。万一、輸送上のトラブルが起こった場合には、ご一報いただければ新しいものと交換いたします。

マニュアルの作成については、なるべく詳細な説明をするよう心がけたつもりですが、理解できないところは、実際にコンピュータと向き合って納得のいくまで確かめて下さい。また、他のページを参照するのほひとつの方法です。それでも疑問点が解決できないときは、封書にて、下記の要領でお問い合わせ下さい。恐れ入りますが、このパッケージの性格上、お電話でのお答えは不可能と存じますので、ご了承下さいますようお願い申し上げます。

記

1. 送付先 〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル
株式会社アスキー ユーザーサポート係

2. 必要項目

(1) お客様の氏名、郵便番号、住所、電話番号（市外局番も含む）

(2) 製品名、ユーザーID 番号、製品シリアル番号

(3) 機器構成

本体のメーカー名、機種名

接続している周辺機器のメーカー名、機種名（型番）

使用しているソフトウェアのメーカー名とその名前

3. お問い合わせ内容

お問い合わせの内容は、製品のマニュアルに記述されている用語を用いて、具体的かつ明確に記述して下さい。なお、障害と思われる現象については、その現象を再現可能な情報が必要です。当社で再現できないものは調査できません。その現象が発生するまでの操作手順、データを必ず添付して下さい。データディスクがある場合は、そのコピーも同封して下さい。

なお、お客様がプログラミングされたアプリケーションソフトウェアの設計、作成、運用、保守などについては、当社のサポート範囲外ですので、お問い合わせいただいても回答できません。例えば、「このプログラムリストはなぜ動かないのか」といった質問にはお答えできません。

また、MSX-MUSIC と MSX-AUDIO 用の FM 音源 LSI の技術資料は、ヤマハ株式会社のご好意により掲載させていただきましたが、この内容についての問い合わせも弊社で承ります。ただし、具体的な音作りなどに関するご質問にはお答えできませんので、ご了承下さいますようお願い申し上げます。

MSX-Datapack Volume 1

1991年2月1日 第1版第1刷

編 集 株式会社アスキー システム事業部

発 行 所 株式会社アスキー

担当 遠藤 祥、北浦 訓行

〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル

TEL (03)3486-7111(大代表)

制 作 株式会社ジャパックスインターナショナル