





MSX クリエイターズ ガイドブック

G-NET クリガイプロジェクト

MSX,MSX2,MSX2+,MSXturboR はアスキーの商標です。

本書について

この資料は MSX ユーザーの資料不足を解消する事を第一の目的として、GENUINE NETWORK の有志によるプロジェクトにより作成されました。

本書は様々な制約の中で、とにかく少しでもユーザーの役に立つものを作るという観点から、MSX に関連する事項の中から特に使われる場合が多いと思われる項目を中心に構成しました。取り上げている事項についても枝葉末節は省いている所もあります。そういった部分が必要になった時には参考文献を、ということが許されない現状ではありますが、自分で補うなり人に聞くなりして切り抜けて頂きたいと思います。逆にこれまで一般の資料にはなかった記事を、筆者陣とご意見頂いた方々の知識及び多数の資料を基に収録する事ができたのは大きな成果でした。

本書の一部または全部は、非営利かつ本書の目的に沿う場合に限り、自由に複製・配布して構いません。

本書を購入したい場合は奥付の連絡先で通信販売を行っています。

凡例

本来 MSX1 という名称は存在しませんが、MSX2 以降の MSX と区別するために MSX BASIC Ver.1.0 を搭載した MSX を MSX1 と表記します。また MSXturboR を turboR と表記する事があります。単に MSX とある場合は MSX1、MSX2、MSX2+、MSXturboR の総称です。

BIOS 等のルーチンは原則的に次のように説明してあります。

● ラベル(アドレス/アドレス区分)	MSXバージョン
機能	
入力	
出力	
変更	
説明	

アドレス区分はそのルーチンのエントリが存在するスロットを示します。メイン RAM に存在する場合など書いていない場合もあります。

MAIN	MAIN ROM
SUB	SUB ROM
FM	FM BIOS ROM

機種はそのルーチンが使える機種の情報が書いてあります。また、この情報が書いてあっても、turboR で削除されたなど別に説明されている場合があります。オプション機器のルーチンなどでは書いていない場合があります。

MSX1	MSX1 からすべての MSX で使えます。
MSX2	MSX2 以降のバージョンで使えます。
MSX2+	MSX2+以降のバージョンで使えます。
MSXtR	MSXturboR 以降のバージョンで使えます。

入力は呼び出す前に設定すべきレジスタやワークエリア、出力はルーチンが設定して返すレジスタやワークエリアです。

変更はルーチンが変更するレジスタです。この項目が省略されてその章で説明している場合があります。

目次

第 1 部 基本システム 13

1 章 基本仕様	14
2 章 CPU とその周辺	15
1.2.1 CPU と動作モード	15
1.2.2 Z80	15
1.2.2.1 Z80 の概要	15
1.2.2.2 M1 サイクルのウェイト	15
1.2.2.3 I/O アクセスのウェイト	15
1.2.3 R800	16
1.2.3.1 R800	16
1.2.3.2 DRAM アクセス	16
1.2.3.3 内蔵 ROM のアクセス	16
1.2.3.4 外部メモリのアクセス	17
1.2.3.5 I/O ポートのアクセス	17
1.2.3.6 リフレッシュ	17
1.2.3.7 命令の実行時間	17
1.2.3.8 高速なプログラムを組むには	18
1.2.3.9 システムタイマー	18
1.2.4 命令実行クロック	18
3 章 メモリ管理機構	20
1.3.1 スロット	20
1.3.1.1 スロットとは	20
1.3.1.2 スロットの構造	20
1.3.1.3 スロットの操作	22
1.3.1.4 スロットを使用する上での注意事項	26
1.3.2 メモリマップ	27
1.3.2.1 メモリマップとは	27
1.3.2.2 メモリマップの構造	27
1.3.2.3 メモリマップの操作	28
1.3.2.4 メモリマップを使用する上での注意事項	33
4 章 I/O ポート	34
1.4.1 I/O ポート	34
1.4.2 拡張 I/O ポート	34
1.4.3 16 ビットアドレス I/O	34
5 章 割り込み	35
1.5.1 MSX の割り込みの概要	35
1.5.2 CPU の割り込み動作	35
1.5.3 システムの割り込み処理	35
1.5.4 ユーザーの割り込み処理	36
1.5.5 割り込みに関連した注意	36
6 章 フック	37
1.6.1 フック	37
1.6.2 フックの使用法	37

1.6.3	フック使用上の注意	37
7 章	BIOS.....	39
1.7.1	BIOS	39
1.7.2	拡張 BIOS	39
1.7.2.1	デバイス番号	39
1.7.2.2	拡張 BIOS の呼び出し	40
1.7.2.3	ブロードキャストコマンド	40
1.7.2.4	各デバイスに対するコマンド	41
1.7.2.5	システムエクスクルーシブ	42
1.7.2.6	拡張 BIOS の設定	42

第 2 部 V D P 45

1 章	VDP に関する基礎知識.....	46
2.1.1	VDP の種類	46
2.1.2	VDP のレジスタ	46
2.1.2.1	コントロールレジスタ (R#0 ~ R#23, R#25 ~ R#27, R#32 ~ R#46)	46
2.1.2.2	ステータスレジスタ (S#0 ~ S#9)	46
2.1.2.3	パレットレジスタ (P#0 ~ P#15)	47
2.1.3	VRAM	47
2.1.4	I/O ポート	47
2 章	基本入出力.....	48
2.2.1	コントロールレジスタのアクセス	48
2.2.1.1	直接指定	48
2.2.1.2	間接指定	48
2.2.2	ステータスレジスタのアクセス	49
2.2.3	パレットレジスタのアクセス	50
2.2.4	VRAM (VIDEO RAM) のアクセス	50
2.2.5	VDP の I/O アクセスに関する注意点	51
2.2.6	TMS9918A 互換モード	52
2.2.7	画面構成	52
2.2.8	VDP に関連するワークエリア	52
2.2.9	パレットテーブル	53
2.2.10	BASIC の VDP(n)関数	53
3 章	レジスタの機能.....	55
2.3.1	コントロールレジスタ R#0 ~ R#23, R#25 ~ R#27, R#32 ~ R#46 (Write only)	55
2.3.1.1	モードレジスタ	55
2.3.1.2	テーブルベースアドレスレジスタ	57
2.3.1.3	カラーレジスタ	57
2.3.1.4	ディスプレイレジスタ	59
2.3.1.5	アクセスレジスタ	60
2.3.1.6	コマンドレジスタ	61
2.3.2	ステータスレジスタ S#0 ~ S#9 (Read only)	61
4 章	VDP の画面モード.....	63
2.4.1	TEXT1 (SCREEN 0 ・ 40 字モード)	63
2.4.1.1	特徴	63
2.4.1.2	関係するレジスタと VRAM 領域	63
2.4.1.3	レジスタと VRAM の設定	64
2.4.1.4	VRAM マップ	65
2.4.2	TEXT2 (SCREEN 0 ・ 80 字モード)	66

2.4.2.1	特徴	66
2.4.2.2	関係するレジスタと VRAM 領域	66
2.4.2.3	レジスタと VRAM の設定	66
2.4.2.4	VRAM マップ	68
2.4.3	GRAPHIC1 (SCREEN 1)	69
2.4.3.1	特徴	69
2.4.3.2	関係するレジスタと VRAM 領域	69
2.4.3.3	レジスタと VRAM の設定	69
2.4.3.4	VRAM マップ	72
2.4.4	GRAPHIC2 (SCREEN 2)	72
2.4.4.1	特徴	72
2.4.4.2	関係するレジスタと VRAM 領域	72
2.4.4.3	レジスタと VRAM の設定	73
2.4.4.4	VRAM マップ	76
2.4.5	MULTI COLOR (SCREEN 3)	77
2.4.5.1	特徴	77
2.4.5.2	関係するレジスタと VRAM 領域	77
2.4.5.3	レジスタと VRAM の設定	77
2.4.5.4	VRAM マップ	80
2.4.6	GRAPHIC3 (SCREEN 4)	80
2.4.6.1	特徴	80
2.4.6.2	関係するレジスタと VRAM 領域	80
2.4.6.3	レジスタと VRAM の設定	81
2.4.6.4	VRAM マップ	82
2.4.7	GRAPHIC4 (SCREEN 5)	82
2.4.7.1	特徴	82
2.4.7.2	関係するレジスタと VRAM 領域	83
2.4.7.3	レジスタと VRAM の設定	83
2.4.7.4	VRAM マップ	84
2.4.8	GRAPHIC5 (SCREEN 6)	84
2.4.8.1	特徴	84
2.4.8.2	関係するレジスタと VRAM 領域	85
2.4.8.3	レジスタと VRAM の設定	85
2.4.8.4	VRAM マップ	87
2.4.9	GRAPHIC6 (SCREEN 7)	87
2.4.9.1	特徴	87
2.4.9.2	関係するレジスタと VRAM 領域	87
2.4.9.3	レジスタと VRAM の設定	88
2.4.9.4	VRAM マップ	89
2.4.10	GRAPHIC7 (SCREEN 8)	89
2.4.10.1	特徴	89
2.4.10.2	関係するレジスタと VRAM 領域	90
2.4.10.3	レジスタと VRAM の設定	90
2.4.10.4	VRAM マップ	91
2.4.11	YJK/RGB 混在 (SCREEN 10・11)	92
2.4.11.1	YJK 方式	92
2.4.11.2	特徴	92
2.4.11.3	関係するレジスタと VRAM 領域	92
2.4.11.4	レジスタと VRAM の設定	93
2.4.11.5	SCREEN 10 と SCREEN 11	94
2.4.11.6	VRAM マップ	95
2.4.12	YJK (SCREEN 12)	95
2.4.12.1	特徴	95
2.4.12.2	関係するレジスタと VRAM 領域	95
2.4.12.3	レジスタと VRAM の設定	95
2.4.12.4	VRAM マップ	98

5 章 スプライト.....99

2.5.1	スプライトに関する基礎知識	99
-------	---------------	----

2.5.2	スプライトモード 1	99
2.5.2.1	特徴	99
2.5.2.2	表示	100
2.5.2.3	スプライトパターンジェネレータテーブル	100
2.5.2.4	スプライトアトリビュートテーブル	102
2.5.3	スプライトモード 2	103
2.5.3.1	特徴	103
2.5.3.2	表示	103
2.5.3.3	スプライトパターンジェネレータテーブル	104
2.5.3.4	スプライトアトリビュートテーブル	104
2.5.3.5	スプライトカラーテーブル	105
2.5.3.6	スプライトの衝突	106
2.5.4	スプライトに対するハードウェアスクロールの影響	106
6 章	VDP コマンド	107
2.6.1	V9938・V9958 のコマンド	107
2.6.2	VDP コマンドにおける座標系	107
2.6.3	ロジカルオペレーション	108
2.6.4	各コマンドの説明	108
2.6.4.1	HMMC (CPU→VRAM 高速転送)	108
2.6.4.2	YMMM (Y 方向の VRAM 間高速転送)	109
2.6.4.3	HMMM (VRAM 間高速転送)	110
2.6.4.4	HMMV (矩形領域の高速塗り潰し)	111
2.6.4.5	LMCM (CPU→VRAM 論理転送)	112
2.6.4.6	LMCM (VRAM→CPU 論理転送)	114
2.6.4.7	LMMM (VRAM 間論理転送)	115
2.6.4.8	LMMV (矩形領域の論理塗り潰し)	116
2.6.4.9	LINE (直線の描画)	117
2.6.4.10	SRCH (カラーコードのサーチ)	118
2.6.4.11	PSET (点の描画)	119
2.6.4.12	POINT (カラーコードの読み出し)	120
2.6.4.13	STOP (VDP コマンドの中断)	120
2.6.5	コマンド終了時のレジスタの状態	121
2.6.6	コマンド処理の高速化	121
2.6.6.1	スプライトの表示を禁止する	121
2.6.6.2	画面の表示を禁止する	121
2.6.7	VDP コマンドの実行時間	122
2.6.8	VDP コマンド実行中の処理	122
7 章	その他	123
2.7.1	V9958 で削除された機能	123
2.7.2	コントロールレジスタ R#9 の IL・EO	123
2.7.3	水平帰線割り込み	124
2.7.4	PAL 方式	124
2.7.5	画面モードによる VRAM 構成の変化	125
2.7.6	表示タイミング	125
2.7.7	レジスタ更新の反映タイミング	126

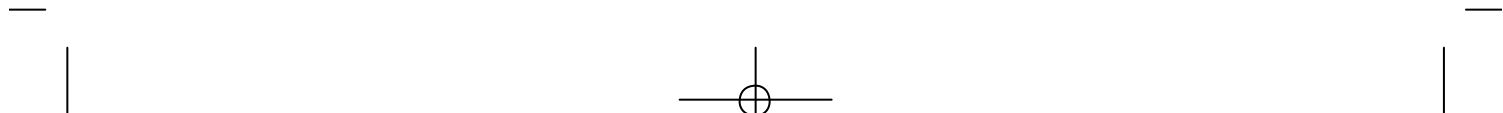
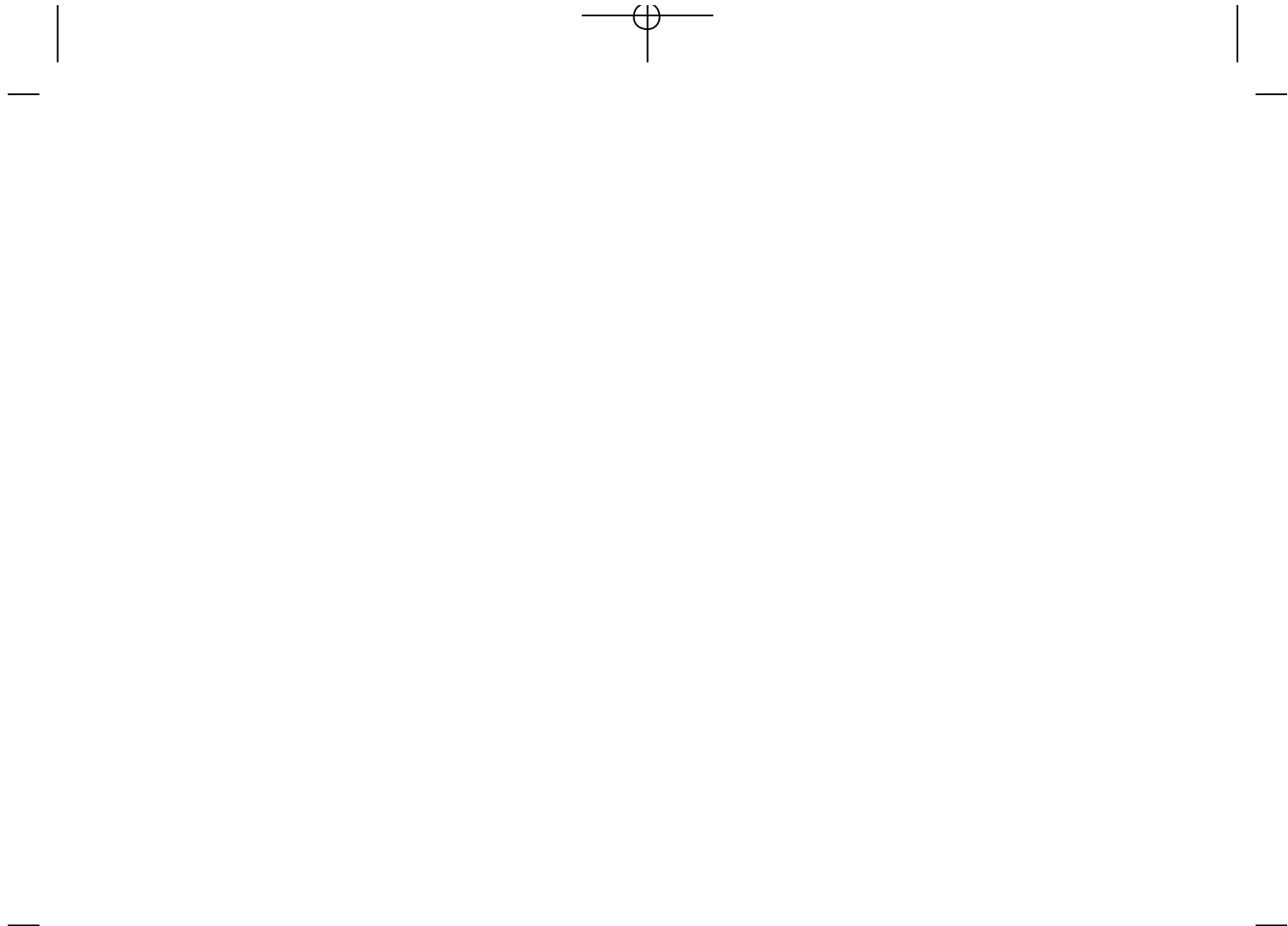
第 3 部 音源 129

1 章	PSG	130
3.1.1	PSG の概要	130
3.1.2	PSG の操作方法	130
3.1.3	PSG の内部レジスタ	131
3.1.4	PSG を使用する上での諸注意	134

2 章	OPLL.....	135
3.2.1	FM 音源(OPLL)の概要	135
3.2.2	FM 音源(OPLL)の操作方法	135
3.2.3	FM 音源(OPLL)の内部レジスタ	139
3.2.4	音色設定	142
3.2.5	FM 音源(OPLL)を使用する上での諸注意	145
3 章	SCC.....	146
3.3.1	SCC の概要	146
3.3.2	SCC の操作方法	146
3.3.3	SCC の内部レジスタ	147
3.3.4	SCC の波形メモリ	148
3.3.5	SCC を使用する上での諸注意	149
4 章	MIDI.....	150
3.4.1	MSXにおける MIDI の位置付け	150
3.4.2	MSX-MIDI	150
3.4.3	MSX-MIDI の認識	151
3.4.4	MSX-MIDI の操作	151
5 章	1 ビットサウンドポート.....	155
6 章	PCM.....	156
3.6.1	PCM とは	156
3.6.2	BIOS でのアクセス	156
3.6.3	I/O ポートでのアクセス	156
3.6.3.1	ハードウェア	156
3.6.3.2	再生手順	157
3.6.3.3	録音手順	157
3.6.4	その他	157
7 章	P A C (Pana Amusement Cartridge)	159
3.7.1	P A C について	159
3.7.2	P A C の検索	159
3.7.3	P A C へのアクセス	160
3.7.4	注意	160
8 章	その他.....	161
第 4 部 周辺装置		163

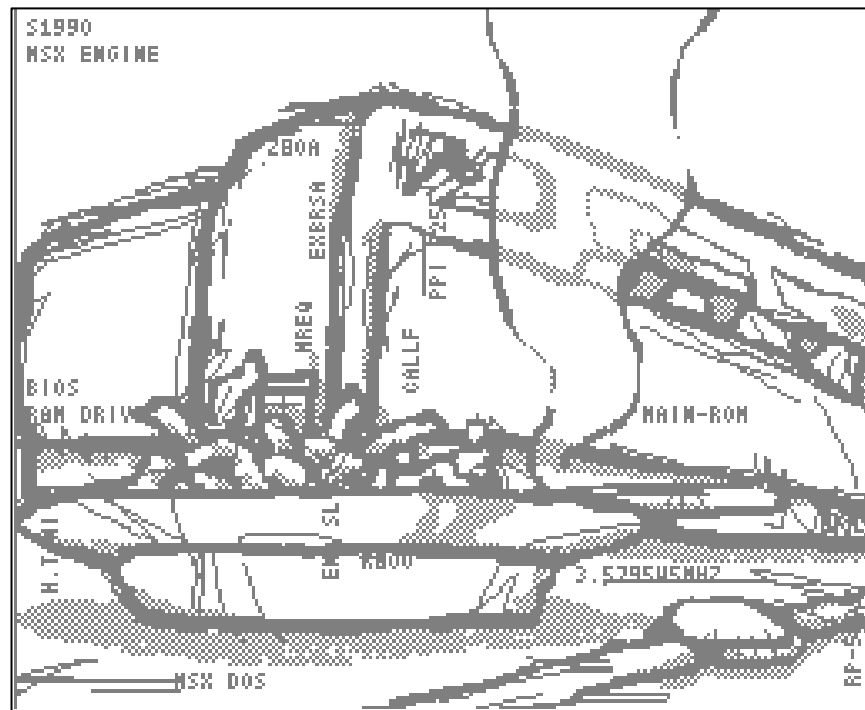
1 章	汎用入出力インターフェイス.....	164
4.1.1	インターフェイスの構造	164
4.1.2	ジョイスティック	165
4.1.3	マウス	165
4.1.4	インターフェイス使用上の注意	165
2 章	プリンタインターフェイス.....	166
4.2.1	プリンタインターフェイス	166
4.2.2	MSX仕様のプリンタ	166
4.2.3	プリンタへの出力	166

3 章	キーボード	167
4.3.1	キー配列	167
4.3.2	キーマトリクス	167
4.3.3	キースキャン	168
4.3.3.1	BIOS を用いる方法	168
4.3.3.2	I/O ポートを用いる方法	168
4.3.3.3	ワークエリアを読む方法	168
4.3.4	文字の入力	168
4.3.4.1	キーバッファ	168
4.3.4.2	BIOS	169
4.3.5	ファンクションキー	169
4.3.6	CTRL+STOP の判定	169
4 章	リアルタイムクロック	170
4.4.1	リアルタイムクロック	170
4.4.2	クロック IC	170
4.4.2.1	レジスタの構成	170
4.4.2.2	クロックとアラーム	170
4.4.2.3	バッテリーバックアップメモリ	170
4.4.2.4	MODE レジスタ (#13)	172
4.4.2.5	TEST レジスタ (#14)	172
4.4.2.6	RESET レジスタ (#15)	172
4.4.3	アクセス	173
5 章	漢字 ROM	174
4.5.1	漢字 ROM とは	174
4.5.2	漢字 ROM の I/O ポート	174
4.5.3	漢字 ROM のチェック	174
4.5.3.1	第 1 水準漢字 ROM のチェック	174
4.5.3.2	第 2 水準漢字 ROM のチェック	174
4.5.4	漢字コード変換法	175
4.5.5	半角文字のフォント	175
6 章	システムタイマー	176
第 5 部 資料		177
1 章	BIOS	178
2 章	システムワークエリア	188
3 章	I/O マップ	197
4 章	キャラクターコード	200
5 章	コントロールコード	201
6 章	エスケープシーケンス	202
7 章	メーカーコード	203



第1部

基本システム



1章 基本仕様

表 1.1 MSX シリーズの基本仕様

		MSX1	MSX2	MSX2+	MSXturboR
CPU		Z80A 相当品 3.579545MHz±1%			Z80A 相当品 3.579545MHz±1% R800 相当品 7.15909MHz
メモリ	ROM	32KB (MSX BASIC Ver.1.0)	48KB (MSX BASIC Ver.2.0)	80KB (MSX BASIC Ver.3.0)	160KB (MSX BASIC Ver.4.0 / 4.1)
		MAIN ROM 32KB	MAIN ROM 32KB SUB ROM 16KB	MAIN ROM 32KB SUB ROM 16KB 漢字ドライバ 16KB 単漢字変換 16KB	MAIN ROM 32KB SUB ROM 16KB 漢字ドライバ 16KB 単漢字変換 16KB MSX-DOS1 16KB MSX-DOS2 48KB MSX-MUSIC 16KB
	RAM	8KB 以上	64KB 以上	64KB 以上	256KB 以上
	VRAM	16KB	64KB または 128KB	128KB	
VDP		TMS9918A相当品	V9938 相当品	V9958 相当品	
	最大 解像度	256 × 192 (ノンインターレース)	512 × 212 (ノンインターレース) 512 × 424 (インターレース)		
	最大表 示色数	16 色	256 色	19268 色	
	ハードウェア スクロール		縦	縦・横	
CMTカセット インターフェイス		FSK 方式 1200・2400bps			
音源	PSG	AY-3-8910 相当品			
	FM 音源	MSX-AUDIO (オプション)			
		FM-PAC 対応		MSX-MUSIC (オプション)	MSX-MUSIC
	MIDI				MSX-MIDI (オプション)
PCM				8bit 15.75KHz	
キーボード		英数、ひらがな、カタカナ、グラフィック機能対応			
		JIS 配列、50 音配列対応			
フロッピーディスク		(オプション)			2DD (1DD はオプション)
プリンタ		(オプション)	8ビットパラレル (セントロニクスインターフェイス準拠)		
カートリッジスロット		1つ以上		2つ	
ジョイスティック		1または2	2		
漢字	ROM	第1水準 (オプション)		第1水準	
			第2水準(オプション)	第2水準	
	入力	アプリケーションで対応		単漢変換 (MSX-JE 対応)	
リアルタイム クロック IC		(オプション)	RP5C01 相当品		

2章 CPU とその周辺

1.2.1 CPU と動作モード

CPU には MSX2+までは Z80A 相当品が、MSXturboR では Z80 に加えて R800 相当品が使われています。動作クロックは Z80 が 3.579545MHz、R800 は 7.15909MHz です。R800 はクロック周波数が 2 倍になり命令のクロック数も減少したので、Z80 に比べて大幅に高速化しています。

turboR では Z80 モードと R800 モードをソフト的に切り換えて使用します。CPU の切り換えは BIOS を使用します。詳しくは第 5 部 1 章 BIOS (p.184 CHGCPU) を参照して下さい。Z80 上で組まれたプログラムを使用する時に R800 では速すぎてうまく動作しない時に Z80 を使います。"1"キー(テンキー不可)を押しながら起動することにより、Z80 モードで起動します。

turboR では R800 時には R800-ROM モードと R800-DRAM モードの 2 つのモードがあります。R800-DRAM モードではメイン ROM とサブ ROM、漢字 BASIC の 4 ページ分の ROM の内容を DRAM 上にコピーして実行するモードです。R800-ROM モードは普通どおり ROM で実行するモードです。

R800-DRAM モードではメイン ROM / サブ ROM / 漢字 BASIC の ROM があるスロットには DRAM が見えるようになります。この DRAM は ROM と同じように書き込みはできなくなります。R800 では DRAM 上のプログラムの方が高速に実行できるので R800-DRAM モードの方が動作が早くなります。R800-DRAM モードでは内蔵 DRAM の最後の 4 ページが ROM の代わりに使われるので RAM として使えるエリアが減ります。

HD64180 という Z80 上位互換 CPU を Z80 と切り換えて使用できる MSX2 がありましたが、HD64180 は MSX 規格外です。

1.2.2 Z80

1.2.2.1 Z80 の概要

Z80 の動作クロックは 3.579545MHz です。1 クロックサイクルは約 0.279 μ 秒になります。

1.2.2.2 M1 サイクルのウェイト

MSX では Z80 の M1 サイクルで 1 ウェイトが発生します。M1 サイクルは命令コードの読み込みサイクルで、命令により 1 ~ 2 回の M1 サイクルがあります。この M1 サイクルのウェイトは DRAM のリフレッシュに使われます。この M1 サイクルのウェイトは MSX 特有のもので、一般の Z80 のマニュアル等にあるクロック数を MSX にそのまま適用することはできません。1.2.4 命令実行クロックに M1 サイクルのウェイトを含めた Z80 のクロック数の表があります。

1.2.2.3 I/O アクセスのウェイト

一部の機種(Panasonic・SANYO の MSX2+と Panasonic の MSXturboR)では VDP アクセスに 1 ウェイトが発生します。

1.2.3 R800

1.2.3.1 R800

R800 の動作クロックは 7.15909MHz です。1 クロックサイクルは約 0.140 μ 秒になります。供給クロックは 28.63636MHz ですが、命令実行サイクルはその 1/4 のサイクルになります。

命令のフェッチと命令の実行はパイプライン化されているので、命令の実行中に次の命令のフェッチを行います。これにより、命令を最短で 1 クロック実行します。ただし、割り込みによりプリフェッチした命令を実行できなかった時には、割り込み終了後に再び実行出来なかった命令をフェッチします。

Z80 になかった命令の拡張も行われており、Z80 で動作する未定義命令として使われていたインデックスレジスタの 8 ビットアクセス命令が正式に使えるようになったり、掛け算命令が新設されたりしています。

R800 時に発生するウェイトはメモリアクセスのウェイト、I/O アクセスのウェイト、リフレッシュの為のウェイトがあります。この他、特にウェイトを発生する I/Oなどをアクセスした場合にウェイトが発生します。

R800 の DMA・内蔵マップ・モード 3 割り込み・高速 I/O アクセスモードは、ハード的に無効に設定されているので MSX では使えません。また、R レジスタが 8 ビット長(Z80 では 7 ビット)になっているので、乱数の種等につかうときは注意して下さい。

1.2.3.2 DRAM アクセス

R800 は内蔵 DRAM に対してはページモードでアクセスします。アドレスの上位バイトが同じ 256 バイトをページと言い、同じページのなかで連続してアクセスすることをページアクセスといいます。ページアクセスできない場合、ページブレークが発生したと言い、ページブレークが発生した場合の DRAM アクセスには 2 クロックが必要になります。

ページアクセスができずにページブレークの発生するのは、次の 2 つの条件のうち 1 つ以上が成立した時です。たとえばアドレスの上位バイトが変化しなくても、アドレスを指定するレジスタが変わればウェイトは発生します。

1. アドレスの上位バイトが変わった
2. アドレスを指定するレジスタが変わった

DRAM の上位バイトが変わる場合というのは、PC の上位バイトが変わった場合やページ境界の連続したメモリアクセスした場合 (LD HL,(00FFH)など) です。

アドレスを指定するレジスタは、プログラムのフェッチ時には PC ですが、レジスタ間接アドレッシング (LD A,(HL)など)、インデックスアドレッシング (LD A,(IX+n)など)、エクステンダドレッシング (LD A,(nn)など)、及び絶対ジャンプ命令で変化します。

エクステンダドレッシングや絶対ジャンプ命令はレジスタを使わないように見えるので、これに該当しないように思えるかもしれませんが、内部的には一度アドレスをテンポラリレジスタに記憶してメモリアクセスするようになっています。

PC から他のレジスタでアドレスを指定するレジスタが変わると、次の命令の読み込みで再び PC にレジスタが変わりますので、一度レジスタが変わると 2 回はウェイトが発生するのが普通です。

1.2.3.3 内蔵 ROM のアクセス

内蔵 ROM のアクセスには 2 ウェイトが発生します。

1.2.3.4 外部メモリのアクセス

スロットに接続された ROM や RAM をアクセスするときは互換性の為に Z80 のタイミングでのアクセスになり、ウェイトが発生します。外部スロットは Z80 のタイミングで動作しているので、Z80 に比べクロック周波数が 2 倍になっている R800 でアクセスすると命令の実行タイミングによりウェイトのサイクル数が異なってきます。

Z80 の動作クロックの後半にあたるタイミングでアクセスしようとする場合をタイミング A、前半にアクセスしようとする場合をタイミング B として説明します。

外部メモリアクセスでは、タイミング A でアクセスすると 3 クロック、タイミング B でアクセスすると 4 クロックのウェイトが発生します。ただし、外部のメモリをアクセスした直後に連続して外部のメモリをアクセスすると 5 ウェイト発生します (LD HL, (nn) でアドレス nn が外部のメモリだった場合など)。

実行する前にこのタイミングのどちらで実行されるかを CPU が知る方法はありませんが、タイミングによってウェイトの数が異なるので命令の実行直後に現在どちらのタイミングになっているかはわかります。ウェイトを含めると、メモリアクセスは必ずタイミング A で終わるので、メモリアクセスの終了後に次の命令はタイミング B から実行されるからです。

1.2.3.5 I/O ポートのアクセス

I/O ポートは互換性の為に Z80 のタイミングで動作するので、ウェイトにはアクセスのタイミング(1.2.3.4 外部メモリのアクセス参照)が関係します。

タイミング A のアクセスは 6 ウェイト、タイミング B のアクセスで 5 ウェイトが発生します。

例外として次のものがあります。

VDP	VDP は 8.66 μ 秒以内にアクセスしようとした場合にシステムがウェイトが発生
漢字 ROM	漢字 ROM の読みだしは内蔵 ROM のウェイトと同じ 2 クロックのウェイト
プリンタ	プリントデータの出力 (91H 出力), STROBE 信号の出力・解除 (90H 出力), BUSY 信号の読み出し (90H 入力) のどの組合せに対しても 4.48 μ 秒以内に連続して行った場合にウェイトが発生
キーボード	キーボードスキャン信号を出力 (91H 出力) してから 4.48 μ 秒以内にデータ読み出し (90H 入力) した場合にウェイトが発生

1.2.3.6 リフレッシュ

リフレッシュは 29.3 μ 秒毎に 16 もしくは 24 クロック掛けて行われます(内蔵 RAM 容量が 512KB では 24 クロック・256KB と 1024KB では 16 クロック)。リフレッシュに掛かる時間は 29.3 μ 秒の中に含まれます。リフレッシュ後にはその前にプリフェッチした命令を再度フェッチするために 2 クロック余計に時間が掛かり、18 もしくは 26 クロックのウェイトになります。

R800 は命令実行とは非同期で行われるので、このウェイトが発生するタイミングを知ることはできません。

1.2.3.7 命令の実行時間

様々なウェイトが頻繁に発生するために、R800 のノーウェイトのクロック数を使って実行時間を予測するのはかなり難しくなります。しかし、ユーザーが開発プログラムは DRAM 上で動作する場合がほとんどで、その場合に限定すれば命令コードの実行に伴って発生するアドレスを指定するレジスタ変更によるウェイトは、命令の種類によって予測可能です。そこで、内蔵 DRAM 上で動作するプログラムの場合には、DRAM アクセス時のレジスタ切り換えの為に発生するページブレークについては、実質的にそのウェイトを含めた時間が基本的な命令実行時間と考える事ができます。そのように DRAM アクセスのレジスタの切り換えでのウェイトを含めたクロック数は 1.2.4 命令実行クロックにまとめました。

1.2.3.8 高速なプログラムを組むには

特に高速なプログラムを組みたい場合は、ページ境界をまたがないようにするとよいでしょう。

また、ROM 上や外部の RAM 上のアクセスはウェイトが発生するので、ROM で供給されるプログラムは内蔵 DRAM 上に転送して実行したりする事が有効です。

1.2.3.9 システムタイマー

ウェイトの発生条件が複雑な上に命令実行と非同期のウェイトがあるので、R800 では正確な実行時間を知ることができません。その為にソフトタイマーでは一定の時間待ちを作る事ができません。一定の時間待ちが欲しい場合には MSXturboR で新しく装備されたシステムタイマーを使う必要があります。

システムタイマーについては6章 システムタイマーを参照して下さい。

1.2.4 命令実行クロック

Z80 と R800 について、各命令の動作に必要なクロックを表 1.2.4 にまとめました。

Z80 については、ノーウェイトでのクロック数と M1 サイクルで1ウェイト発生する場合のウェイトを含めたクロック数があります。MSX では後者を使います。

R800 については、ノーウェイトのクロック数と、DRAM 上で実行した場合にアドレスを指定するレジスタが変わる事によって発生するウェイトを含めたクロック数があります。後者はその命令自体は既にフェッチされているとして、その命令により発生したメモリアクセス及び次の命令コードのフェッチによってアドレスを指定するレジスタが変わったことによるウェイトを含めています。DRAM 上で実行され、DRAM 以外へのアクセスも行わない場合は、このクロック数でウェイトまで含めた実行時間が計算できます。その場合でも、PC がページ境界を超えた場合やリフレッシュの発生によるウェイトは例外となります。

表 1.2.1 動作クロック表

8ビット移動命令					
命令	B	Z80		R800	
LD r,r'	1	4	5	1	1
LD r,n	2	7	8	2	2
LD u,u'	2	-	-	2	2
LD u,n	3	-	-	3	3
LD r,(HL)	1	7	8	2	4
LD r,(IX+d)	3	19	21	5	7
LD (HL),r	1	7	8	2	4
LD (IX+d),r	3	19	21	5	7
LD (HL),n	2	10	11	3	5
LD (IX+d),n	4	19	21	5	7
LD A,(BC)	1	7	8	2	4
LD A,(DE)	1	7	8	2	4
LD A,(nn)	3	13	14	4	6
LD (BC),A	1	7	8	2	4
LD (DE),A	1	7	8	2	4
LD (nn),A	3	13	14	4	6
LD A,I	2	9	11	2	2
LD A,R	2	9	11	2	2
LD I,A	2	9	11	2	2
LD R,A	2	9	11	2	2
PUSH / POP 命令					
命令	B	Z80		R800	
PUSH qq	1	11	12	4	6
PUSH IX	2	15	17	5	7
POP qq	1	10	11	3	5
POP IX	2	14	16	4	6

16ビット移動命令					
命令	B	Z80		R800	
LD ss,nn	3	10	11	3	3
LD IX,nn	4	14	16	4	4
LD HL,(nn)	3	16	17	5	7
LD ss,(nn)	4	20	22	6	8
LD IX,(nn)	4	20	22	6	8
LD (nn),HL	3	16	17	5	7
LD (nn),ss	4	20	22	6	8
LD (nn),IX	4	20	22	6	8
LD SP,HL	1	6	7	1	1
LD SP,IX	2	10	12	2	2

交換命令					
命令	B	Z80		R800	
EX DE,HL	1	4	5	1	1
EX AF,AF'	1	4	5	1	1
EX (SP),HL	1	19	20	5	7
EX (SP),IX	2	23	25	6	8
EXX	1	4	5	1	1

16ビット演算命令					
命令	B	Z80		R800	
ADD HL,ss	1	11	12	1	1
ADD IX,pp	2	15	17	2	2
ADC HL,ss	2	15	17	2	2
SBC HL,ss	2	15	17	2	2
INC ss	1	6	7	1	1
INC IX	2	10	12	2	2
DEC s		INC と同じ			

8ビット演算命令

命令	B	Z80	R800
ADD A,r	1 4 5	1 1	
ADD A,p	2 - -	2 2	
ADD A,(HL)	1 7 8	2 4	
ADD A,(IX+d)	3 19 21	5 7	
ADD A,n	2 7 8	2 2	
ADC A,s	ADDと同じ		
SUB s	ADDと同じ		
SBC A,s	ADDと同じ		
OR s	ADDと同じ		
AND s	ADDと同じ		
XOR s	ADDと同じ		
CP s	ADDと同じ		
INC r	1 4 5	1 1	
INC p	2 - -	2 2	
INC (HL)	1 11 12	4 7	
INC (IX+d)	3 23 25	7 10	
DEC s	INCと同じ		
DAA	1 4 5	1 1	
CPL	1 4 5	1 1	
NEG	2 8 10	2 2	

乗算命令

命令	B	Z80	R800
MULUB A,r	2 - -	14 14	
MULUW HL,ss	2 - -	36 36	

ブロック転送命令

命令	B	Z80	R800
LDI	2 16 18	4 7	
LDIR (BC ≠ 0)	2 21 23	4 7	
	16 18	4 7	
LDD	2 16 18	4 7	
LDDR (BC ≠ 0)	2 21 23	4 7	
	16 18	4 7	
CPI	2 16 18	4 6	
CPIR (BC ≠ 0 又は A ≠ (HL))	2 21 23	5 7	
	16 18	5 7	
CPD	2 16 18	4 6	
CPDR (BC ≠ 0 又は A ≠ (HL))	2 21 23	5 7	
	16 18	5 7	

分岐命令

命令	B	Z80	R800
JP nn	3 10 11	3 5	
JP cc,nn (条件成立)	3 10 11	3 5	
JP (HL)	1 4 5	1 3	
JP (IX)	2 8 10	2 4	
JR e (ページブレーク)	2 12 13	3 3	
	- -	3 4	
JR cc,e (条件成立)	2 7 8	2 2	
	12 13	3 3	
	- -	3 4	
DJNZ e (ページブレーク)	2 13 14	3 3	
	- -	3 4	
	8 9	2 2	

コール命令

命令	B	Z80	R800
CALL nn	3 17 18	5 7	
CALL cc,nn (条件成立)	3 10 11	3 3	
	17 18	5 7	
RET	1 10 11	3 5	
RET cc (条件成立)	1 5 6	1 1	
	11 12	3 5	
RETI	2 14 16	5 7	
RETN	2 14 16	5 7	
RST k	1 11 12	4 6	

ビット循環命令

命令	B	Z80	R800
RLCA	1 4 5	1 1	
RLA	1 4 5	1 1	
RRCA	1 4 5	1 1	
RRA	1 4 5	1 1	
RLD	2 18 20	5 8	
RRD	2 18 20	5 8	
RLC r	2 8 10	2 2	
RLC (HL)	2 15 17	5 8	
RLC (IX+d)	4 23 25	7 10	
RL s	RLCと同じ		
RRC s	RLCと同じ		
RR s	RLCと同じ		
SLA s	RLCと同じ		
SRA s	RLCと同じ		
SRL s	RLCと同じ		

ビット操作命令

命令	B	Z80	R800
BIT b,r	2 8 10	2 2	
BIT b,(HL)	2 12 14	3 5	
BIT b,(IX+d)	4 20 22	5 7	
SET b,r	2 8 10	2 2	
SET b,(HL)	2 15 17	5 8	
SET b,(IX+d)	4 23 25	7 10	
RES b,s	SETと同じ		

入出力命令

命令	B	Z80	R800
IN A,(n)	2 11 12	3 4	
IN r,(C)	2 12 14	3 4	
IN F,(C)	2 - -	3 4	
INI	2 16 18	4 6	
INIR (B ≠ 0)	2 21 23	4 6	
	16 18	3 5	
IND	2 16 18	4 6	
INDR (B ≠ 0)	2 21 23	4 6	
	16 18	3 5	
OUT (n),A	2 11 12	3 4	
OUT (C),r	2 12 14	3 4	
OUTI	2 16 18	4 6	
OTIR (B ≠ 0)	2 21 23	4 6	
	16 18	3 5	
OUTD	2 16 18	4 6	
OTDR (B ≠ 0)	2 21 23	4 6	
	16 18	3 5	

CPU 制御命令

命令	B	Z80	R800
CCF	1 4 5	1 1	
SCF	1 4 5	1 1	
NOP	1 4 5	1 1	
HALT	1 4 5	2 2	
DI	1 4 5	2 2	
EI	1 4 5	1 1	
IM 0	2 8 10	3 3	
IM 1	2 8 10	3 3	
IM 2	2 8 10	3 3	

各欄の意味

B	命令長
Z80 左	ノーウェイト
右	M1サイクルで
	1ウェイト
R800 左	ノーウェイト
右	DRAM上で
	実行した場合

レジスタ等の記号

r,r'	B,C,D,E,H,L,A
u,u'	B,C,D,E,IXH,IXL,A
p	IXH,IXL
ss	BC,DE,HL,SP
pp	BC,DE,IX,SP
qq	BC,DE,HL,AF
cc	条件
k	RST 命令のコールアドレス
s	許されるすべてのオペランド
(IY に関する命令は IX の場合と同じです)	

3章 メモリ管理機構

MSX では Z80 の 64KB のメモリ空間をスロット及びメモリマップによって拡張しています。スロットは MSX の基本的なメモリ空間拡張機構、メモリマップは RAM 拡張の規格で、本来は別のものですがここではメモリ管理機構としてまとめました。

1.3.1 スロット

1.3.1.1 スロットとは

MSX に使用されている CPU「Z80(R800)」は、16 ビットのアドレスバスを持っています。そのため、基本的には最大で 64KB のメモリ空間しかアクセスすることができません。

このような CPU でもっと大きなメモリ空間を扱うために、MSX には最大 1MB のメモリ空間をコントロールすることができるようにするための機構「スロット」が装備されています。「スロット」に接続された最大で 1MB のメモリ空間を、ある程度の制限はあるもののアプリケーション側から自由に CPU に接続することができるようにするものです。

MSX 本体の外側に出ている各種カートリッジを接続するためのコネクタもスロット（カートリッジスロット）と呼ばれています。確かにこの章で取り扱う「スロット」との関係は非常に大きいのですが、あくまでそれは「スロット」の一側面に過ぎません。

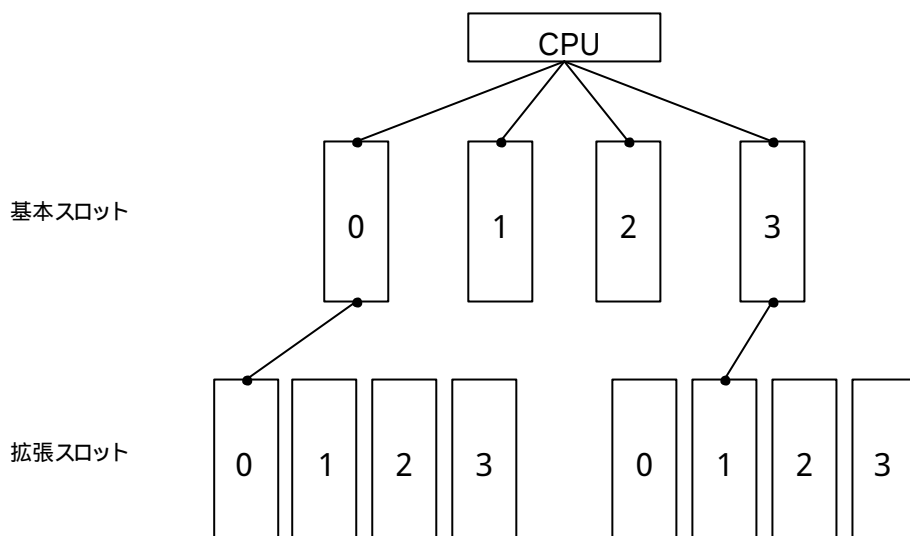
1.3.1.2 スロットの構造

「スロット」とは、要するに CPU からのアドレス・データの信号の切り換え装置みたいなものだと思います。各スロットも 64KB のメモリ空間を持ち、スイッチ（I/O ポートですが）1 つで特定のスロットと CPU が持つ 64KB のメモリ空間を接続してしまうというものです。メモリなどは、このスロットを通して CPU と接続されています。

スロットには、まず基本の 4 スロット（基本スロット（プライマリスロットともいう））が存在します。それぞれ 0 ～ 3 番の番号が付けられ、各種のデバイス（RAM・ROM 等）が接続されています。4 つの基本スロットは、そのうち 1 つを CPU に接続することができます。

基本スロットには、それぞれ更に 4 つずつのスロットを接続することができます（拡張スロット（セカンダリスロットともいう））。更に接続された 4 つのスロットは、いずれか一つを任意に基本スロットに接続することができるようになっていきます。要するに、この拡張スロットを使用することで一つの基本スロットを最大 4 つのスロットに増やすことができるというわけです。拡張スロットの扱いは基本的には基本スロットと変わりませんので、それぞれ各種デバイスを独立して接続することができます。但し、拡張スロットを更に拡張することはできません。

(例)



結局、基本・拡張それぞれのスロットを任意に CPU へ接続することができるようになりました。1つのスロットがそれぞれ 64KB のメモリ空間を持つことができるわけですから、これはすなわち、最大 $64 \times 4 \times 4 = 1024(\text{KB})$ のメモリ空間を扱うことができるようになったということです。

スロット番号の呼び方は、スロット a-b (a:基本スロット b:拡張スロット) となります。上記の例だと、CPU には、スロット 0-0 が見えています。また、拡張スロットは基本スロットごとに独立して設定できるので、基本スロット 3 にはスロット 3-1 が選択されています。

ちなみに、外部に突き出た「カートリッジスロット」は、このスロットのうちのいずれかがハードウェア的に表に出ているものです。この「スロット」に外部から任意のデバイス (ROM カートリッジ、増設 RAM など) を接続することで、MSX は新しいメモリ空間を手軽に (差し込むだけ!) 得ることができるというわけです。また、カートリッジスロットは、大抵基本スロットとなっていますので、外部に「拡張スロット (ハードウェア的に本当にカートリッジスロットを増やしてしまう)」を接続することもできるようになっています。

ところで、各スロットを選択する場合、CPU から見える 64KB のメモリ空間の全てを切り換えたのでは非常に使い勝手が悪くなります。切り換えた途端、切り換えたプログラム自身がいなくなってしまうし…。では、メモリ空間を一部分だけ切り換える機能があればどうでしょう。自分自身はそのまま、必要な部分だけスロット切り換えができるような…。そこで、CPU から見えている 64KB のアドレス空間を 16KB ごとに区切り、それらを「ページ」として独立して扱うようにしてみました。すなわち「ページ」ごとに独立してスロットを設定できる仕掛けです。これによって、プログラム自身がいる「ページ」はそのまま、別のページだけを任意のスロットに切り換えることができるようになりました。

これらの機能によって、CPU から見える 64KB のアドレス空間には、最大 1MB 分のメモリ空間を 16KB ごとに切り換えて置くことができるようになりました。たとえば、ページ 0 には BASIC-ROM、ページ 1 には MSX-MUSIC ドライバ、ページ 2・3 には MAIN-RAM、というように、必要に応じて各ページに独立したスロットを設定して（スロットを「出す」）、好きなようにメモリ空間を設定することができるのです。

但し、元々ページ 2 に接続されているメモリをページ 0 に出したい、ということは、残念ながらできません。

ページ スロット番号 メモリ上のアドレス

0	0-0	0000H
		3FFFH
1	3-2	4000H
		7FFFH
2	2	8000H
		BFFFH
3	3-0	C000H
		FFFFH

各ページには、独立して任意のスロットを「出す」ことができる

1.3.1.3 スロットの操作

アプリケーションプログラムからスロットを操作する方法としては、BIOS を使用する、あるいは直接 I/O を操作するということが挙げられます。

最初に、I/O を直接操作してスロットを切り換える方法を示します。

前節で触れたように、スロットは基本（プライマリ）スロット・拡張（セカンダリ）スロットで構成されています。任意のスロットはこの 2 つの組合せによって接続されていますので、必要なスロットを得るためにはこの 2 つを同時に操作する必要があります。

基本スロットの設定は I/O ポートに対するアクセスで行われます。基本スロットの切り換えポートは 0A8H に接続され、「基本スロット選択レジスタ」と呼ばれます。

0A8H

PAGE3	PAGE2	PAGE1	PAGE0
-------	-------	-------	-------

 各ページに対する基本スロットの設定

ページ 0 からページ 3 に設定する基本スロット番号をそれぞれ 2 ビット（0～3）で設定します。なお、このポートは読み書き可能です。

必要なスロットの基本スロットが表に出たら、次は拡張スロットを設定します。拡張スロットの切り換えポート、すなわち「拡張スロット選択レジスタ」はメモリマップド I/O（メモリの特定のアドレスに対する読み書きが I/O へのアクセスになる）となっていて、各スロットの 0FFFFH（同基本スロットであればどの拡張スロットでも共通）に存在します。

0FFFFH

PAGE3	PAGE2	PAGE1	PAGE0
-------	-------	-------	-------

 各ページに対する拡張スロットの設定

基本スロットと同様に各ページに設定する拡張スロット番号をそれぞれ 2 ビットで設定します。このレジスタも読み書きは可能ですが、スロットが拡張されているかどうかのチェックのため、このレジスタに書き込んだ値は次に読み出される時には全ビットが反転されています。すなわち、値を書き込んだ後読み込んで、そのデータが反転されていればそのスロットには拡張スロットが接続されているわけです。

このような構造になっていますので、任意のページを切り換えるためには、

任意のページとページ 3 を切り換えたいスロットの基本スロットに切り換える
FFFFH に切り換えたいスロットの拡張スロット番号を書き込む
もちろん、切り換えたいページ以外の値は保存しておく
ページ切り換え終了。必要な処理を行う

（以下元に戻す場合）

FFFFH に切り換える前の値を戻す(ビット反転に注意)
基本スロットレジスタに切り換える前の値を戻す

という流れになります。

次に、BIOS を使用してスロットに対する各種操作を行う方法を以下に示します。

● RDSLT (000CH/MAIN)

機能 指定スロットの指定アドレスから 1 バイトを読みます

入力 A スロット番号指定

HL 読み出すアドレス

出力 A 読み出したデータ

変更 AF BC DE

説明 A レジスタで指定したスロットの HL で指定されるアドレスからデータを 1 バイト読み込み、A レジスタに返します。

● WRSLT (0014H/MAIN)

機能 指定スロットの指定アドレスに 1 バイト書き込みます

入力 A スロット番号指定

HL 読み出すアドレス

E 書き込むデータ

出力 なし

変更 AF BC D

説明 A レジスタで指定したスロットの HL で指定されるアドレスに、E レジスタの値を書き込みます。

- CALSLT (001CH/MAIN)

機能	指定スロットの指定アドレスをコールします (インタースロットコール)
入力	IY スロット番号指定 (上位 8 ビット) IX コールするアドレス
出力	コールしたルーチンからの戻り値
変更	コールしたルーチンが使用するレジスタ、IX IY 裏レジスタ
説明	IY レジスタの上位 8 ビットで指定されるスロットの IX レジスタで指定されるアドレスをコールします。このルーチンをコールする前と後でスロットの状態は保存されます。

- ENASLT (0024H/MAIN)

機能	指定したページのスロットを切り換えます
入力	A スロット番号指定 HL 上位 2 ビットで切り換えるページを指定
出力	なし
変更	全て
説明	HL の上位 2 ビットで指定されるページを A レジスタで指定されるスロットに切り換えます。ページ番号の指定方法ですが、データ構造上任意のアドレスを HL に入れておけばそのアドレスを含むページが変更されるようになっています。

- CALLF (0030H/MAIN)

機能	指定スロットの指定アドレスをコールする
入力	インラインパラメータ形式でスロット・アドレスを指定
出力	コールしたルーチンからの戻り値
変更	コールしたルーチンが使用するレジスタ、IX IY 裏レジスタ
説明	指定スロットの指定アドレスをコールします。スロットおよびアドレスの指定方法ですが、「インラインパラメータ形式」といって、このコールを呼ぶ命令の後ろに直接スロット番号・アドレスを設定するようにします。すなわち、

CALL	CALLF	; RST	30H でも可
DEFB	slot_number		
DEFW	address		

(以下その後のプログラム)

という形になります。コールから返ってきた後はスタックが補正されていますので、上記の例では (以下その後...) の実行が始まるわけです。

なお、このコールを使用する場合には Z80 の命令「RST 30H」が使用できます。これを使用すると 4 バイトでインタースロットコールが実現できます。

- RSLREG (0138H/MAIN)

機能	基本スロット選択レジスタの値を読みます
入力	なし
出力	A 基本スロット選択レジスタの値
変更	なし
説明	基本スロット選択レジスタ (I/O の 0A8H) を読み出します。

- 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
入力	A 書き込む値
出力	なし
変更	なし
説明	基本スロット選択レジスタ (I/O の 0A8H) へ A レジスタの値を書き込みます。

- EXTROM (015FH/MAIN)

機能 SUB-ROM の指定したアドレスをコールします

入力 IX コールするアドレス

出力 コールしたルーチンからの戻り値

変更 コールしたルーチンが使用するレジスタ

説明 MSX2 以降で拡張された SUB-ROM 上に存在する BIOS などをコールします。

以上のルーチンは全て MAIN-ROM 上に存在します。なお、いくつかは MSX-DOS(2)環境にも用意されていて、全く同じアドレス・機能で使うことができるようになっています。使用できるルーチンは以下のルーチンです。

RDSLT	000CH
WRSLT	0014H
CALSLT	001CH
ENASLT	0024H
CALLF	0030H

ところで、以上のコールの中で再三に渡って使用された「スロット番号」ですが、基本・拡張を同時に指定できるように独特のフォーマットを持っています。そのフォーマットは、

EXP	0	0	0	Sec_num	Pri_num	Slot format
-----	---	---	---	---------	---------	-------------

です。「Pri_num」「Sec_num」にそれぞれ基本・拡張のスロット番号を入れ、拡張スロットまで有効な場合は「EXP」を1にします。そのスロットが拡張されていない場合は Pri_num のみを指定し、EXP を0にしてください。なおビット4～6は未使用ビットです。必ず0にしておいてください。

次に、スロットを扱う上で必要になるであろうワークエリアを示します。なお、以下で指定されるスロット番号も、上記のフォーマットに従います。

【EXPTBL(0FCC1H)】 MAIN-ROMのスロット番号

また、このアドレスから4バイトは、基本スロットの拡張の有無を示しています。0FCC1H から順に基本スロット0～3までの拡張の有無を示し、それぞれの最上位ビットが1であればそのスロットは拡張されています(拡張スロットが存在する)。

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

【SLTTBL(0FCC5H～0FCC8H)】 (4bytes)

拡張スロット選択レジスタの値を保存。BIOS コールで設定されたそれぞれのスロットに対する拡張スロット選択レジスタの値が、0FCC5H から基本スロット0～3の順で入っています。

【SLTATR(0FCC9H～)】 (64bytes)

BAS	DEV	STT	-	-	-	-	-
-----	-----	-----	---	---	---	---	---

各スロット・ページにおけるアプリケーションの有無。任意のスロット・ページに、「拡張ステートメント処理ルーチン(STT)」「拡張装置処理ルーチン(DEV)」「BASICテキスト(BAS)」のいずれかが存在するかどうかを設定されています(存在すれば各ビットが1)。データはスロット0-0のページ0、同ページ1...スロット3-3のページ3の順番で64バイト並んでいます。

【SLTWRK(0FD09H～)】(128bytes)

アプリケーション用のワークエリア。各スロット・ページに存在するアプリケーションが自由に使用することができる2バイトずつのワークエリアです。SLTATR 同様、スロット 0-0 のページ 0 から順番に並んでいます。

【RAMAD0(0F341H)】

【RAMAD1(0F342H)】

【RAMAD2(0F343H)】

【RAMAD3(0F344H)】

MAIN-RAM としてページ 0 からページ 3 にそれぞれ割り当てられているスロット。なお、このワークはディスクインターフェイスが接続されていない場合は存在しません。

1.3.1.4 スロットを使用する上での注意事項

直接 I/O を操作したり ENASLT を使用したりしてスロットを直接切り換える場合は、特にスタックや割り込み処理に関しては注意するようにしてください。スロットを切り換えたらスタックを置いていた RAM が見えなくなった、ページ 0 または 3 を任意のスロットに切り換えたまま割り込みを許可してしまった(割り込みルーチンが行方不明)、などという状況は、かなり発見の難しいバグとなる可能性があります。特に割り込み関係は、アプリケーション側が禁止していてもそれ以外の、たとえば BIOS などが許可している場合もありますので注意が必要です。

スロット関係の BIOS には、いくつか厳しい仕様(という名のバグ)があります。SUB-ROM のコールやページ 0 に対するインタースロットコールなどにあるのですが、ページ数の関係でここでは触れることができません。詳細・対策等はテクニカルガイドブック(発売元:ASCAT)に記載がありますのでそちらも参照してください。

BASIC から ENASLT を使用する場合、ページ 0 を切り換えることはできません。また、BASIC・DOS に拘らず、ページ 3 を切り換えることはできません。切り換えルーチン自体がそれらのページに置かれているためです。

MSX2 以降から、CALSLT・CALLF でインタースロットコールをする場合、その前後で割り込みを保存するようになりました。が、その機能を実現するために使用する Z80 の機能自体(LD A,I など)にバグがあり、実際にはうまく機能していないようです。安心した MSX 生活を送るためには、アプリケーション側で必要に応じて DI・EI を行った方がよいでしょう。

スロットの構成は MSX の機種ごとに大きく異なります。MAIN-RAM ですらページごとに別々のスロットに接続されている可能性があるのです。そのためか、「うちの機種では動くんだけど、別の機種では…」というバグの温床になる可能性が高いようです。MAIN-ROM・MAIN-RAM など、ワークからスロット番号が得られるものに関しては、なるべくワークエリアからスロット番号を確保するなどして機種依存性が生まれないようにする必要があります。こういったスロットの機種依存性に関する記載も、テクニカルガイドブックに詳しい記載がありますのでそちらを参照してください。

スロット構成はもともとはまったく制限はなかったのですが、次第に統一化されてきました。

MSX2+の場合

- ・外部スロットはスロット1と2の2つになる。
- ・スロット3を拡張して、そのどれか1つにRAMを、別のどれか1つにSUB-ROM、漢字ドライバを置く事になっている。
- ・スロット0は拡張されない場合と拡張される場合があり、MAIN-ROMはスロット0または拡張されたスロット0-0に置かれる。
- ・内蔵ディスクインターフェイスがある場合、DISK-ROMは必ずスロット3のいずれかの拡張スロットに置かれる。

turboR の場合

以下のように標準化されました。

0-0	MAIN-ROM
0-2	MSX-MUSIC
1 及び 2	外部スロット
3-0	MAIN-RAM
3-1	SUB-ROM、漢字ドライバ
3-2	DISK-ROM
3-3	内蔵ソフト

1.3.2 メモリマップ

1.3.2.1 メモリマップとは

前節で説明した「スロット」を利用することで、MSXでは最大1MBのメモリ空間を利用できるようになりました。しかし、それ以上のメモリを追加したい、あるいは、複数のスロットにメモリを載せるのは使うのが面倒、などという要求があったのかどうかは知りませんが、MSXには更にメモリマップという機能を使用して、1つのスロットにつき最大4MBのメモリ(RAM)を接続することができるようになっています。

「メモリマップ」とは、最大4MBのメモリ空間を16KBごとに区切って、接続されたスロットの任意のページに出す(CPUから見える状態にする)ことができるというものです。区切られた16KBの単位を「セグメント」と呼び、1つのスロットにつき最大256セグメント($256 \times 16\text{KB} = 4096\text{KB}$)が接続できます。

MAIN-RAMが64KBより多い機種(一部の64KB機種も含む)では必ず搭載されている機能です。また、MSX-DOS2ではシステム自体が64KB以上のRAMを扱う必要ができたため、必須の機能となっています。

1.3.2.2 メモリマップの構造

メモリマップは最大で4MBのメモリを扱うことができるわけですが、もちろんMSXに搭載されたCPU(Z80,R800)は最大で64KBのメモリ空間しか扱えません。もう1つのメモリ管理機構であるスロットではその空間をページごとに16個のスロットのいずれかに切り換えて1MBのメモリ空間を得ていましたが、メモリマップはまた独自の方法を取っています。まずそのスロット上に最大4MBのメモリをまとめて置いてしまうのです。当然そのままでは全てのRAMにアクセスすることはできませんので、登録されたメモリを16KBごとに「セグメント」というものに分割して、そのセグメントを任意にスロット上で切り換えるという方法を取っています。セグメントを切り換えることで、CPUからは同じスロット・メモリ空間上に何枚も違うメモリが乗っているように見えるわけです。

よって、CPUからマップ上のメモリを扱うためには、まずマップの置かれたスロットを表に出して、その上で任意のセグメントをそのスロット上に出してやることになります。

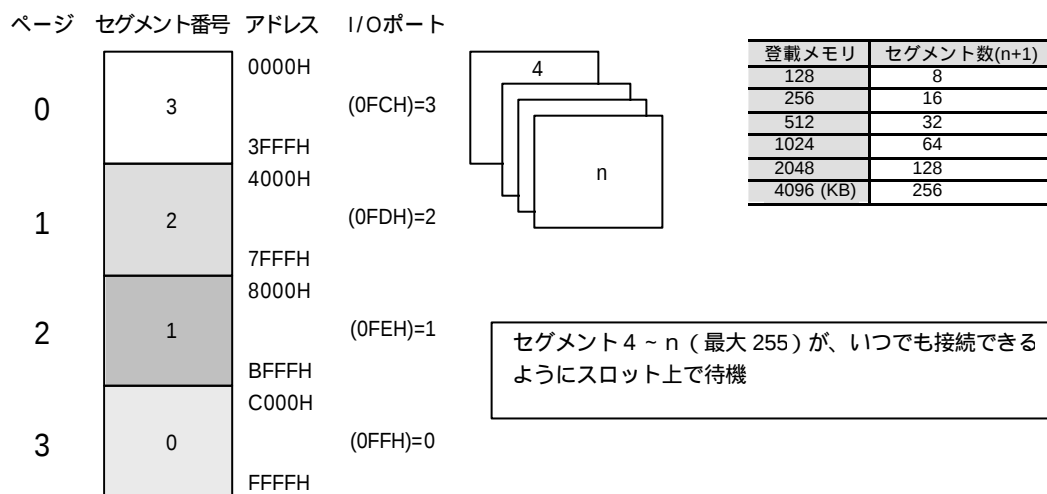
メモリマップ上の RAM は、先頭から順に 16KB ごとに区切られています。それぞれをセグメントと呼び、0～255（最大）までの番号が振られています。それぞれのセグメントは完全に独立していて、マップの存在するスロット上の 0～3 の各ページごとにこれらのセグメントのうちのいずれかを接続することができるようになっています。

各ページに接続するセグメントの番号は I/O ポートから指定するようになっています。ページ 0～3 に対してそれぞれ I/O ポートの 0FCH～0FFH でセグメント番号を指定します。任意のページを表に出す動作は、これらのポートに直接セグメント番号を書き込むことで行われます。

但し、この I/O アドレスは MSX 本体に接続された全てのマップに対して共通です。つまり、どのスロットに接続されたマップも、セグメントを切り換える場合には全て上記のポートをアクセスすることになっているのです。ということは、スロット 3-0 に既にマップが接続されていて、更にスロット 1 にマップを接続するような場合、ページ 0 のセグメントを変えようとして 0FCH に値を書き込むと、スロット 3-0・スロット 1 どちらのセグメントも指定セグメントに切り替わってしまいます。

また、回路的にはこれらの I/O から値を読むことができるようになってはいますが、このように 2 つ以上のマップが接続されていた場合にバスの競合が起こってしまい（両方のマップが値を返してくる）、誤動作する可能性があります。

しかし、読み書きするスロットとセグメント番号さえしっかり把握・管理しておけば（スロット a にあるマップのセグメント b へのアクセス、のように）、各スロットごとに独立して最大 256 枚のセグメントを用意できることが分かります。MSX の規格上では最大 16 のスロットにマップ機能を接続することができるようになっていますので、最大 64MB のメモリ空間が扱えることになります。



1.3.2.3 メモリマップの操作

アプリケーションプログラムからメモリマップを操作するためには、拡張 BIOS を使用します。上記のように直接 I/O を操作する方法もありますが、複数のマップが接続されていた場合に I/O の競合が発生するため、できるだけ避けてください。但し、古い機種ではマップは搭載されていないが拡張 BIOS が搭載されていない機種もありますので、その場合は I/O をアクセスするしかありません。

メモリマップの拡張 BIOS の操作方法は、D レジスタに 4（マップ BIOS のデバイス ID）、E レジスタに各種機能番号、その他のレジスタに必要なパラメータを入れて EXTBIOS(FECAH) をコールすることで行われ

ます。以下にその機能を紹介します。

● 機能番号 1

機能 マップテーブルの先頭アドレスを得る

入力 A 0

出力 A プライマリマップの-slot番号
HL マップテーブルの先頭アドレス

変更 IX、IY、裏レジスタ

説明 システムに認識されているメモリマップ（内蔵・外付け問わず）の情報を得るためのコールです。
A レジスタの値が変更されず、戻ってきた A レジスタの値が 0 だった場合には、メモリマップ拡張 BIOS は存在しません。

A レジスタに入ってくる値はプライマリマップの-slot番号です。プライマリマップとは、MAIN-RAM が確保されているメモリマップで、各種コールにおいても他のマップより優先的に扱われます。MSXturboR では必ず本体内蔵のマップ RAM が使用されます（外付け-slotへのアクセスはウェイトが掛かるため）。

HL レジスタに返される値は、MSX に認識されている全マップの情報を示すテーブルの先頭アドレスです。内容は以下の通り。

オフセット	内容
(HL+0)	マップの存在する-slot番号（プライマリマップ）
(HL+1)	その-slotに接続されたマップの総セグメント数(1～255)
(HL+2)	未使用セグメント数
(HL+3)	システムに割り当てられたセグメント(システムセグメント)数
(HL+4)	ユーザーに割り当てられたセグメント(ユーザーセグメント)数
(HL+5～7)	未使用
(HL+8)	マップの存在する-slot番号（プライマリマップ以外） 他の-slotにこれ以上マップがない場合、(HL+8×n) は 0 になり、テーブルはそこで終わり

● 機能番号 2

機能 マップサポートルーチンの先頭アドレスを得る

入力 A 0

出力 A プライマリマップの総セグメント数
B プライマリマップの-slot番号
C プライマリマップの未使用セグメント数
HL ジャンプテーブルの先頭アドレス

変更 IX IY 裏レジスタ

説明 拡張 BIOS エントリ(0FFCAH) を経由せず、直接マップ BIOS を操作するためのコール（マップサポートルーチン）のジャンプアドレスなどを得るためのコールです。更に A B C レジスタでプライマリマップに付いての各種情報が得られます。

HL に返ってくる値は、マップサポートルーチンへのジャンプテーブルの先頭アドレスです。内容は以下の通り。

オフセット	内容	機能
(HL+00H)	JP ALL_SEG	セグメントを割り当てる
(HL+03H)	JP FRE_SEG	セグメントを解放する
(HL+06H)	JP RD_SEG	セグメントの内容を読む
(HL+09H)	JP WR_SEG	セグメントに書き込む
(HL+0CH)	JP CAL_SEG	インターセグメントコール
(HL+0FH)	JP CALLS	インターセグメントコール
(HL+12H)	JP PUT_PH	セグメント切り換え
(HL+15H)	JP GET_PH	現在のセグメント番号を得る
(HL+18H)	JP PUT_P0	ページ 0 のセグメント切り換え
(HL+1BH)	JP GET_P0	ページ 0 の現在のセグメント番号を得る
(HL+1EH)	JP PUT_P1	ページ 1 のセグメント切り換え
(HL+21H)	JP GET_P1	ページ 1 の現在のセグメント番号を得る
(HL+24H)	JP PUT_P2	ページ 2 のセグメント切り換え
(HL+27H)	JP GET_P2	ページ 2 の現在のセグメント番号を得る
(HL+2AH)	JP PUT_P3	なにもしない
(HL+2DH)	JP GET_P3	ページ 3 の現在のセグメント番号を得る

たとえば、GET_P1 を使用してページ 1 に現在選択されているセグメント番号を得るためには、このコールで得られた HL レジスタの値を元にして、

LD	D,4	
LD	E,2	
XOR	A	
CALL	0FFCAH	
LD	(MAP_TBL),HL	; マップサポートルーチンの先頭アドレスを得る
LD	HL,(MAP_TBL)	; マップサポートルーチンの先頭アドレス
LD	DE,21H	
ADD	HL,DE	; GET_P1 へのジャンプテーブル
JP	(HL)	; JP (HL+21H) → JP GET_P1

のようになります。

次に、このマップサポートルーチンの内容を紹介します。MAP_TBL は上記のコールで得られるマップサポートルーチンの先頭アドレスとします。

- ALL_SEG (MAP_TBL+00H)

機能 セグメントを 1 つ割り当てる

入力 A 0 ユーザーセグメントとして割り付ける
1 システムセグメントとして割り付ける
B 0 プライマリマップに割り付ける
0 以外 B レジスタで指定するスロットに割り付ける
指定方法は下の通り

EXP	Allocate	Sec_num	Pri_mun	Slot_format
-----	----------	---------	---------	-------------

Allocate : 000 指定スロットのみで割り付け
001 指定スロット以外で割り付け
010 指定スロットで割り付けを行い、
失敗した場合は指定スロット以外で割り付ける
011 指定スロット以外で割り付けを行い、
失敗した場合は指定スロットで割り付ける
それ以外のビットは、通常のスロット番号の形式と同じ

出力 Cy 1 未使用セグメントがない (失敗)
0 割り付けに成功
A 割り付けられたセグメント番号
B 割り付けられたセグメントのスロット番号
(入力時 B=0 なら 0)

変更 全て ?

説明 アプリケーション用にセグメントを 1 つ割り付けます。アプリケーションがマップを使用する場合、このコールを使用してセグメントをシステムから割り付けてもらう必要があります。割り付けてもらったセグメントは、アプリケーションが自由に使うことができます。

セグメントを割り付ける場合、ユーザーセグメントとして割り付ける方法とシステムセグメントとして割り付ける方法があります。ユーザーセグメントとして割り当てられたセグメントは、アプリケーションプログラムが終了した時点で自動的に解放され、その後使用し続けることができません。システムセグメントとして割り付けられたセグメントは、そのアプリケーションが終了しても確保されていますので、後からでも参照したりすることができます。たとえば、MSX-DOS2 のシステムで使用されている RAM はシステムセグメントとして割り付けられています。

なお、システムセグメントとして割り付けられたセグメントを解放するためには、次の FRE_SEG を使用します。

このコールを使用する場合、スタックはページ1またはページ3においてください。

● FRE_SEG (MAP_TBL+03H)

機能	指定セグメントを1つ解放する		
入力	A	解放するセグメント番号	
	B	0	プライマリマップ
		0以外	指定スロット（普通のスロット番号）
出力	Cy	1	指定セグメントを解放できない（失敗）
		0	解放に成功
変更	全て？		
説明	アプリケーションが確保したセグメントを解放します。ここで指定するセグメント番号・スロット番号は、上の ALL_SEG で得られた値です。 このコールを使用する場合、スタックはページ1またはページ3においてください。		

以下の2つのコール（RD_SEG、WR_SEG）は、セグメントに対する読み書き動作です。但し、処理速度を稼ぐために指定セグメントの存在のチェック・ページの切り換えは行われません。これらのコールはページ2を介して読み書きを行うため、ページ2には操作するセグメントが存在しているスロットを出しておかなければなりません。なお、いずれも割り込みは禁止されて返ります。

● RD_SEG (MAP_TBL+06H)

機能	指定セグメントの指定アドレスから値を読む		
入力	A	読み出すセグメント番号	
	HL	読み出すアドレス（上位 2 ビットは無効）	
出力	A	読み出した値	
変更	F		
説明	指定したセグメントの指定アドレスの内容を読み出します。		

● WR_SEG (MAP_TBL+09H)

機能	指定セグメントの指定アドレスに値を書き込む		
入力	A	書き込むセグメント番号	
	HL	書き込むアドレス（上位 2 ビットは無効）	
	E	書き込む値	
出力	なし		
変更	AF		
説明	指定したセグメントの指定アドレスに値を書き込みます。		

以下の2つのコール（CAL_SEG、CALLS）は、インターセグメントコール（セグメント内のルーチン呼び出す）です。これらのコールも処理速度を稼ぐために指定セグメントの存在のチェック及びページの切り換えを行いません。指定ページにはそのマップが接続されているスロットを出しておかなければなりません。いずれも割り込みは禁止されて返ります。

● CAL_SEG (MAP_TBL+0CH)

機能	指定セグメントの指定アドレスをコールする		
入力	IX	呼び出すセグメントの番号（上位 8 ビットで指定）	
	IY	呼び出すアドレス	
	AF, BC, DE, HL	が呼び出し先のルーチンへ渡される	
出力	AF, BC, DE, HL, IX, IY が呼びだし先のルーチンから返される		
変更	裏レジスタ		
説明	指定したセグメントの指定アドレスをコールします。コールの前後でセグメントの状態は保存されま す。		

- CALLS (MAP_TBL+0FH)

機能	指定セグメントの指定アドレスをコールする（インラインパラメータ）
入力	（呼びだし方法）
	CALL CALLS
	DEFB 呼び出すセグメントの番号
	DEFW 呼び出すアドレス
	AF, BC, DE, HL が呼び出し先のルーチンへ渡される
出力	AF, BC, DE, HL, IX, IY が呼びだし先のルーチンから返される
変更	裏レジスタ
説明	指定したセグメントの指定アドレスをコールします。コールの前後でセグメントの状態は保存されません。

以下は直接指定ページのセグメントを切り換えるコール（ダイレクトページングルーチン）です。これらのコールもまた、指定セグメントが存在するかどうかのチェックは行われません。

PUT 関係のコールでは、実際にセグメントを切り換えると同時にそのセグメント番号をワークに保存します。それぞれの GET ルーチンは、そこで保存された値をワーク上から読み出しているだけです。

アプリケーションが直接セグメントを切り換える場合は、必ず切り換える前のセグメントの状態を保存して、利用が済んだ後はセグメントの状態を元に戻さなければなりません。

- PUT_PH (MAP_TBL+12H)

機能	指定ページのセグメントを切り換える
入力	H ページを指定（上位 2 ビット）
	A セグメント番号
出力	なし
変更	なし
説明	指定ページのセグメントを切り換えます。ページは H レジスタの上位 2 ビットで指定されます。

- GET_PH (MAP_TBL+15H)

機能	指定ページのセグメント番号を得る
入力	H ページを指定（上位 2 ビット）
出力	A セグメント番号
変更	なし
説明	指定ページのセグメント番号を得ます。ページは H レジスタの上位 2 ビットで指定されます。

- PUT_Pn (MAP_TBL+18H, +1EH, +24H)

機能	各ページのセグメントを切り換える
入力	A セグメント番号
出力	なし
変更	なし
説明	各ページのセグメントを切り換えます。n はページ番号を表します（0 ~ 2）。 PUT_P3 (MAP_TBL+2AH) も存在しますが、何もせずに返ります。

- GET_Pn (MAP_TBL+1BH, +21H, +27H, +2DH)

機能	各ページのセグメント番号を得る
入力	なし
出力	A セグメント番号
変更	なし
説明	各ページのセグメント番号を得ます。n はページ番号を表します（0 ~ 3）。

1.3.2.4 メモリマップを使用する上での注意事項

メモリマップサポートルーチンおよびマップ関連のワークはページ3に置かれます。そのため、ページ3に対しセグメントを切り換えることはできません。切り換えた途端に自分自身がなくなって暴走してしまうためです。

マップの存在を調べるためには、それぞれのスロットに対しマップレジスタを切り換えながら読み書きを行ってマップ RAM の存在をチェックするしかありません。但し、単にマップがあるかどうかを調べたい場合には、BASIC から OUT 255,1 (ページ3のセグメントを1にする) とすることで調べられます。暴走すればマップが存在します。

turboR では起動時にプライマリマップの後ろから4枚にシステム ROM を転送します。たとえば RAM256KB の機種では、セグメント番号15に MAIN-ROM のページ0、14に同ページ1、13に SUB-ROM、12に漢字ドライバが転送されます。そして、各 ROM へのアクセスが発生すると、システムは ROM の代わりに転送された RAM をアクセスするようになります。R800 では ROM へのアクセスより RAM へのアクセスの方が速いため、こうした方がシステム全体が速くなるのです。もちろん、この分(64KB) MAIN-RAM は減少しています。このとき CPU は R800-DRAM モードです。

一方、多少速度は落ちてもメモリが欲しい、という場合もありますので、その場合は CPU を R800-ROM モードとして (BIOS の CHG CPU を使用)、プライマリマップの後ろから4枚を FRE_SEG で解放してしまうことでメモリを 64KB 稼ぐことができます。但し、一旦このモードになった場合はリセットしない限り R800-DRAM モードに戻ることはできません (一旦解放されたセグメントは内容が保証されないため)。なお、Z80 では ROM でも RAM でも速度は変わりません。

DOS2 のカーネルは初期化時に 128KB 以上の容量を持つメモリマップが存在するかチェックし、存在すれば、最大の容量を持つメモリマップのうちスロット番号の一番小さいマップをプライマリマップとします。

ただし、turboR では外部スロットのメモリアクセスにウェイトが発生するためスロット 3-0 にある内蔵 RAM が常にプライマリマップになります。

4章 I/O ポート

1.4.1 I/O ポート

MSX では互換性と拡張性の為に、ハードウェアの差異を BIOS によって吸収するように定められています。そのため、市販用のソフトウェアでは特に定められている場合を除いて I/O ポートをアクセスしてはいけない事になっています。

例外として定められているものに VDP がありますが、他にも漢字 ROM など I/O ポートをアクセスせざるを得ないものも存在しますので、現実的にはかなり I/O ポートを使う機会もあると考えられます。

ユーザーが設計する機器ではユーザー用の I/O ポート 00H ~ 3FH を使用して下さい。それ以外のエリアは使用されていないポートもシステムで予約されています。

I/O ポートの割り当てについては第 5 部 3 章 I/O ポート割り当てをご覧ください。

1.4.2 拡張 I/O ポート

I/O ポートに接続できる機器の数を増やす為に、拡張 I/O ポートの仕様が定められています。I/O ポートの 40H ~ 4FH まだが拡張 I/O ポートの領域で、ここに接続された機器は 40H にデバイス ID を書き込んだ場合のみ接続されるようになっています。

デバイス ID は 1 ~ 254 が使用されます。デバイス ID の反転を読んでデバイスの有無を検出する時の都合で 0 と 255 は使用しません。1 ~ 127 は拡張 BIOS と同様にメーカー ID を使用します。128 ~ 254 はデバイスの種類に応じた番号を割り当てます。

原則として特定の機種で使われる内蔵デバイスにはメーカー ID を、複数の機種に接続可能なデバイスにはデバイスの種類を割り当てます。

将来の拡張が考えられる装置や、メーカー ID で接続されるデバイスでは、16 ビットアドレスでアクセスすることが推奨されています。

1.4.3 16 ビットアドレス I/O

Z80 や R800 の I/O は実際には 16 ビットアドレスを持っています。接続された装置がアドレスの上位バイトもデコードすれば 16 ビットのアドレスすべてを使用して接続する I/O 装置を増やせます。

16 ビットアドレス I/O は普通の I/O 命令によって行われます。この時アドレスの上位バイトの指定には、C レジスタで I/O アドレスを指定する命令では B レジスタ、イミディエイト値で指定する命令では A レジスタが使われます。

5章 割り込み

1.5.1 MSX の割り込みの概要

MSX は Z80 のモード 1 割り込みを使用しています。この割り込みモードは割り込み信号がアクティブになると 0038H がコールされる割り込みです。原則的に Z80 の他の割り込みモードは使えません。

CPU への割り込み信号を発生する割り込み源には、VDP や RS-232C、MSX-MIDI などがあります。

基本的なシステムでサポートしているのは VDP の垂直帰線割り込みだけです。この割り込みは 1 秒間に 60 回 (NTSC の場合、国外の PAL、SECAM 仕様の MSX では 50 回) 発生します。この割り込みはキーボードの処理や、BASIC の PLAY 文による演奏などに利用されています。

1.5.2 CPU の割り込み動作

割り込みは IFF (割り込み許可フリップフロップ) によってコントロールされており、IFF がリセットされている状態では割り込み発生は禁止され、IFF がセットされている状態では発生が許可されます。

DI 命令は割り込みを禁止 (IFF をリセット) し、EI 命令は割り込みを許可 (IFF をセット) します。

IFF の状態は LD A,I または LD A,R 命令によって P/V フラグにセットされます。しかし、この機能は一部の MSX で正しく動作しないので使えません。MSX エンジンのバグらしいです。

割り込み信号がアクティブになって、かつ割り込みが許可されている場合には割り込みが発生します。

割り込みが発生した場合の動作は次のようになります。

- ・再度割り込みが発生してしまうのを防ぐために IFF がリセットされ割り込みが禁止されます。
- ・現在の PC をスタックに積んで 0038H にジャンプします。

1.5.3 システムの割り込み処理

割り込みが発生して 0038H がコールされると、ページ 0 に BIOS ROM が出ている状態では BIOS ROM の割り込みルーチンが実行されます。

DOS 上ではページ 0 が RAM になっています。0038H からページ 3 の MSX-DOS のシステムにジャンプし、そこから MAIN ROM の割り込みルーチンがコールされます。

割り込みルーチンでは、【H.KEYI】と【H.TIMI】のフックが呼ばれます。【H.KEYI(FD9AH)】は割り込みルーチンの最初で呼び出されますので、すべての割り込みによって呼び出されます。その後割り込みがタイマ割り込みだった場合には【H.TIMI(FD9FH)】を呼び出します。

システムの割り込み処理では、キーボードやジョイスティックのスキャンなどを行っています。

【JIFFY(FC9EH,2)】は垂直帰線割り込み毎にインクリメントされますので、時間をカウントするのに使うことができます。

【SCNCNT(F3F6H)】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。これにより 2 回に 1 回の割り込みでスキャンしています。

1.5.4 ユーザーの割り込み処理

BASIC 上で割り込み処理を追加、変更したい場合にはフックを使って下さい。

DOS 上で割り込み処理を変更したい場合には、フックを使う他に、0038H のジャンプ命令を書き換えてユーザーの割り込みルーチンにジャンプさせることも可能です。注意が必要なのは、BIOS やシステムコールを使っている時にページ 0 のスロットやセグメントが切り替わっていると、もともとの BIOS ROM ルーチンにジャンプする可能性があるので注意して下さい。

垂直帰線割り込みを使う場合には、【H.TIMI(FD9FH)】を使います。垂直帰線割り込み以外の割り込みは【H.KEYI(FD9AH)】を使います。

フックを使用する場合の方法については 6 章 フックを参照して下さい。

フックを使うにしても 0038H からのジャンプ命令を書き換えるにしても、終了するソフトウェアでは必ず元の状態を保存して、終了する前に復元しないと暴走の原因になります。プログラムがアボートされた場合にも復元できるように注意して下さい。

割り込みの処理が終了したら EI 命令で割り込みを許可しないと次の割り込みが発生しなくなってしまいます。

1.5.5 割り込みに関連した注意

システムの割り込み処理では、VDP のステータスレジスタを見て VDP の垂直帰線割り込みを判断します。この時、VDP のポート #1 を読み出すだけなので、VDP のステータスレジスタポインタ (R#15) を 0 以外にする時は割り込みを禁止しなければなりません。そして割り込みを許可する前に 0 に戻して下さい。0 以外になっていると、VDP の割り込みがリセットされずに割り込みが掛かりっぱなしになり暴走します。

割り込みを長時間禁止すると、割り込み処理を使用するデバイス (RS-232C など) の割り込み処理に不都合が生じる場合があります。これを避ける為に、1 ミリ秒以上割り込みを禁止する場合には拡張 BIOS を利用して割り込みの禁止を宣言して下さい。これにより割り込み処理を使用するデバイスは割り込み禁止に備える事ができます。割り込みを許可する時には、再び拡張 BIOS で割り込みの許可を宣言して下さい。

割り込み処理中で VDP、特に VRAM をアクセスする場合には注意が必要です。システムでは VDP レジスタの設定をする間は割り込みを禁止しますが、VRAM のアクセス中は割り込みを禁止していません。VRAM をアクセスしている途中で発生した割り込み中で VRAM のアクセスポインタが変わってしまうと、割り込みが終了した後の VRAM アクセスは変更されたアクセスポインタによってアクセスされてしまいます。

特に高速な応答を必要とする割り込み処理を行う場合には、システムの割り込み処理を無効にする事も考えられます。BASIC 環境では、一番最初に呼ばれるフック【H.KEYI】から処理をユーザーのルーチンにジャンプさせ、割り込み処理終了後はスタックに積まれているリターンアドレスを捨て、システムが保存したレジスタを復帰させてリターンします。DOS 環境では 0038H からジャンプさせるだけです。システムの割り込みを無効にするとキーボードスキャンなどシステムが割り込み処理により提供する機能は使用できなくなるので注意して下さい。

タイマ割り込みのチェーンを行わないと、本体やディスクドライブの機種によって、モーターが止まらなくなります。これはディスクドライブタイマ割り込みをカウントしている為で、この症状を回避するにはモーターを停止させる為のルーチンを使います。これについては MSX テクニカルガイドブックディスク編(発売元: ASCAT)などを参照して下さい。

6章 フック

1.6.1 フック

システムのソフトウェアの動作を変更する為にフックとよばれる機構があります。ROM に書かれたプログラムは変更することができませんが、変更を想定したルーチンには RAM 上に置かれたフックアドレスを呼び出しているため、そのフックを書き換えることによってシステムの動作を変更することができます。

フックは5バイトまたは3バイトの領域で、最初は RET 命令が書き込まれています。もしその内容を書き換えたい場合には、そこにジャンプ命令かインタースロットコールを書くことによって新しい処理ルーチンに処理を移すことができます。

1.6.2 フックの使用法

フックを書き換えるには JP 命令を使う方法と、RST 30H によるインタースロットコールによる方法があります。

JP 命令による書き換えは処理も高速で効率が良いですが、フックが呼ばれる時に処理ルーチンのあるスロットが選択されている必要があります。フックには処理ルーチンへの JP 命令を書きます。

RST 30H (CALLF) によるインタースロットコールを利用する場合は、処理ルーチンが選択されていないスロットに存在しても呼び出すことができます。その場合はフックに RST 30H 命令に続いて、スロット番号と呼びだしアドレス、最後に RET 命令の5バイトを書きます。ROM で供給されるデバイスドライバや裏 RAM に置かれるプログラムに処理ルーチンがある場合にはこの方法を使います。

フックを書き換える時に、既にフックを書き換えているプログラムがある場合、フックを書き換えることによって、先にフックを書き換えたプログラムには処理が渡らなくなってしまいます。これによって支障が発生するのを防ぐためにフックのチェーンを行う必要があります。それには元のフックの内容を自分が確保したエリアに保存し、フックから処理が渡ってきて自分の処理を終えた後に保存したフックを実行します。JP 命令によるフックを行っているプログラムがあると、スロットの状態が問題になりますので注意が必要です。

フックを書き換えたり、復元したりする時、そのフックが割り込み処理で呼ばれる可能性がある場合には、フックの保存や書き換えが完了して整合した状態になるまでは割り込みを禁止しておく必要があります。これが正しく行われないと、フックを書き換えている途中で割り込みが発生すると暴走する事になります。

プログラム実行を終了する事のできるプログラムでは、フックを書き換える時に元のフックの内容を保存しておく必要があります。プログラムの終了時、あるいは常駐プログラムの常駐解除時に元のフックの内容を復元しないと暴走の原因になります。

1.6.3 フック使用上の注意

フックを使う場合、フックを呼ぶルーチンによりレジスタの内容が意味を持っている場合があるので、RET で戻る場合にはレジスタの内容の保存などに注意が必要です。

常駐プログラムなどの常駐解除を行う場合には、自分よりも後にフックを書き換えたプログラムがある可能性がありますので、不用意に元の内容を上書きすると暴走の原因になります。必ず、自分が書き換えた後

第1部

に更に他のプログラムによって書き換えられていないかどうかを確認してから、常駐解除処理を行って下さい。

7章 BIOS

1.7.1 BIOS

BIOS とは Basic Input Output System の略で、MSX の機能を簡単に使うことができるように、また互換性を確保する為に提供されているシステムのルーチン群です。

BIOS は MAIN ROM (BIOS ROM) 及び SUB ROM のページ 0 に存在します。ROM の先頭付近にジャンプテーブルが存在しており、そのジャンプテーブルをコールする事で機能が利用できます。

MSX ではバージョンや機種による差異を BIOS で吸収するために、VDP など一部を除いて周辺装置へのアクセスは I/O ポートを使用せずに BIOS を使って行うことが推奨されています。特に市販するアプリケーションでは MSX のバージョンの違いにより動作しなくなってしまうことがないように必ず BIOS を使用する必要があります。実際には部分によりますが、かなり I/O ポートによるアクセスも行われていますが、機種によって動作しないことがないように十分な配慮が必要です。

BIOS のエントリアドレスはジャンプテーブルになっていますが、ジャンプテーブルのジャンプ先はバージョンによって異なっているので、ジャンプ先を直接呼び出すようなことはしてはなりません。

詳しい BIOS の内容は、BIOS の機能に関連する各部に収録しています。また本文で扱っていない範囲については巻末に収録しています。

1.7.2 拡張 BIOS

MSX ではハードウェアを拡張した時には、そのスロットの ROM 上に拡張したハードウェアを制御するソフトウェアが実装されます。そのハードウェアやソフトウェアにアクセスするには、どのスロットに接続されているかわからなければなりません。

この問題を解決するために定められたのが拡張 BIOS という手法です。拡張 BIOS では、

- ・何の拡張ハードウェアがシステムに実装されているか
- ・その数はいくつか
- ・どのカートリッジスロットに接続されているか

などを調べることができます。

また拡張デバイスに対して、割り込みの禁止/許可の宣言を行う機能があります。

1.7.2.1 デバイス番号

拡張 BIOS では拡張ハードウェアとその制御用ソフトウェアの組み合わせをデバイスとしています。デバイスには 0 ~ 255 のデバイス番号を割り当てています。

デバイス番号は本来アスキーが登録・管理していたものですが、拡張 BIOS は構造的にはソフトウェアのみで拡張する事もできるので、常駐プログラムで使用される事もあります。そのような場合は 254 以下の番号が利用されるのが普通です。

デバイス番号 0 はすべてのデバイスに対する要求に使うデバイス番号です。

表 1.7.1 拡張 BIOS のデバイス番号

0	すべてのデバイス
1 ~ 3	未使用
4	MSX-DOS2 (メモリマッパーサポート)
5 ~ 7	未使用
8	RS232C、MSX-MODEM
9	未使用
10	MSX-AUDIO
11	MSX-MIDI
12 ~ 15	未使用
16	MSX-JE
17	漢字ドライバ
18 ~ 254	未使用
255	システムエクスクルーシブ (メーカー独自の拡張 BIOS)

1.7.2.2 拡張 BIOS の呼び出し

拡張 BIOS は MSX の基本仕様ではないので、まず拡張 BIOS が存在するかどうかが問題になります。拡張 BIOS が存在する場合、【HOKVLD(FB20H)】のビット 0 が 1 になっています。

ディスクシステムは拡張 BIOS エントリのセットアップを行うので、DOS のようにディスクが存在することが確実な環境では拡張 BIOS の存在の確認はしなくても構いません。

拡張 BIOS を使用するには、D レジスタにデバイス番号、E レジスタに機能番号を設定して EXTBI0(FFCAH) をコールします。

拡張 BIOS はインタースロットコールで呼び出され、しかも複数のデバイスがネストするため、デバイスが 1 つ呼び出されるたびに最低 16 バイトのスタックを必要とします。また、ページ 2 に拡張 BIOS のプログラムが配置されていることもあるのでスタックはページ 3 にある必要があります。

- EXTBI0 (FFCAH)

機能 拡張 BIOS のエントリ

入力 D デバイス番号

E 機能番号 (デバイスによる)

出力 拡張 BIOS のデバイスと機能による

変更

1.7.2.3 ブロードキャストコマンド

デバイス番号 0 によるすべてのデバイスに対する機能 (ブロードキャストコマンド) には次の物があります。

- 機能番号 0

機能 デバイス番号の取得

入力 B テーブルのスロット

HL テーブルのアドレス

出力 HL テーブルの終了アドレス + 1

指定したテーブルにデバイス番号のリストがセットされる

説明 システムにどのようなデバイスがあるか調べます。呼び出すプログラムはデバイス番号を格納するテーブルを用意して、そのスロットとアドレスを渡します。

システムに接続されたデバイスはそれぞれ自分のデバイス番号をテーブルに書き込んで HL レジスタをインクリメントします。

拡張 BIOS から帰ってきた HL アドレスと設定した HL アドレスの差の 1/2 がデバイスの数になります。これは、各デバイスがすべてのデバイス (デバイス番号 0) としても動作する為です。また、この方法では存在を調べられないデバイスあります (例: 漢字ドライバ)。

- 機能番号 1

機能 トラップ使用数の取得
 入力 A 0
 出力 A 使用するトラップの数
 説明 MSX BASIC のトラップは 26 個あり、イベント番号 18～23 が拡張デバイス用のトラップになっています。トラップを使用するデバイスは初期化時に他のデバイスがいくつのトラップを使用するかを調べ、18 + (戻り値) のトラップを使用します。

- 機能番号 2

機能 割り込み禁止の宣言
 入力 なし
 出力 なし
 説明 割り込みが長時間禁止されると不都合が生じるデバイスのために、割り込みを禁止する事を宣言します。割り込みを禁止する長さが 1 ミリ秒以上になる場合はこのエントリを呼び出して下さい。割り込みを許可する時は必ず機能番号 3 の割り込み許可の宣言を呼び出して下さい。

- 機能番号 3

機能 割り込み許可の宣言
 入力 なし
 出力 なし
 説明 機能番号 2 で禁止宣言されていた割り込みを許可します。

1.7.2.4 各デバイスに対するコマンド

各デバイスに対するコマンドは、機能番号 0 のみが共通に定められており、その他の機能はデバイスによって異なりますので各デバイスの解説を参照して下さい。

- 機能番号 0

機能 エントリアドレスの取得
 入力 B テーブルのスロット
 HL テーブルのアドレス
 出力 HL テーブルの終了アドレス + 1
 テーブルが設定される
 説明 各デバイスが持っている BIOS のジャンプテーブルのスロットと先頭アドレスをテーブルに設定します。テーブルは 1 つ 4 バイトで、同じデバイス番号のデバイスが複数ある場合には複数のテーブルのエントリが複数返されます。各エントリは 4 バイトなので、戻り値の HL の値と設定した HL の値の差はエントリ数の 4 倍になります。標準的なテーブルのエントリの内容は次の通りです。なお、これとは異なる形のテーブルを帰すデバイスも存在します。

+0	デバイスのスロットアドレス
+1	ジャンプテーブルのアドレス (下位)
+2	ジャンプテーブルのアドレス (上位)
+3	システム予約 (0)

1.7.2.5 システムエクスクルーシブ

デバイス番号 255 によるメーカー独自の拡張 BIOS に対するコマンドです。

● 機能番号 0

機能 エントリアドレスの取得

入力 B テーブルのスロット

HL テーブルのアドレス

出力 HL テーブルの終了アドレス + 1

テーブルが設定される

説明 各デバイスが持っているメーカー独自の拡張 BIOS のジャンプテーブルのスロットと先頭アドレス及びメーカーコードをテーブルに設定します。メーカーコードは第 5 部 7 章 メーカーコードを参照して下さい。

すべてのデバイスがテーブルを設定するので、特定のデバイスを選択する時は各デバイスに対するコマンドの機能番号 0 を併用します。

テーブルは 1 つ 5 バイトで、同じデバイス番号のデバイスが複数ある場合には複数のテーブルのエントリが複数返されます。各エントリは 5 バイトなので、戻り値の HL の値と設定した HL の値の差はエントリ数の 5 倍になります。

テーブルのエントリの内容は次の通りです。

+0	デバイスのスロットアドレス
+1	ジャンプテーブルのアドレス (下位)
+2	ジャンプテーブルのアドレス (上位)
+3	メーカーコード
+4	システム予約 (0)

1.7.2.6 拡張 BIOS の設定

ユーザーのプログラムが拡張 BIOS に機能を設定するには、次のようにします。

【HOKVLD】のビット 0 が 0 ならば 1 にして、【EXTBIO(FFCAH)】からの 29 バイトを C9H で埋めて初期化します。これは既に拡張 BIOS が存在している環境なら必要ありません。

先に【EXTBIO】を使っているプログラムのために、【EXTBIO】から 5 バイトを自分のワークエリアに保存します。自分の拡張 BIOS 処理の後に保存した内容を実行するようにします。

自分の拡張 BIOS へのインタースロットコール命令を【EXTBIO(FFCAH)】からに書きます。

DISINT, ENAINT のプログラムを設定します。その内容はリスト 1.7.1 の通りで、拡張 BIOS のブロードキャストコマンドを呼び出すものです。ブロードキャストコマンドの割り込みの禁止/許可の宣言を使わない拡張 BIOS の場合には設定しなくても平気でしょう。このルーチンの使われ方は良くわからないのですが、ユーザーが割り込み禁止宣言をする時は通常の拡張 BIOS のエントリを使うのが無難かもしれません。

拡張 BIOS はブロードキャストコマンドと各々のデバイスで決まっている呼び出しコマンドは必ずサポートして下さい。

拡張 BIOS の処理ルーチンでは、デバイス番号が 0 (ブロードキャストコマンド)の場合と自分のデバイス番号に一致する場合に、それに対する処理をして下さい。

リスト 1.7.1

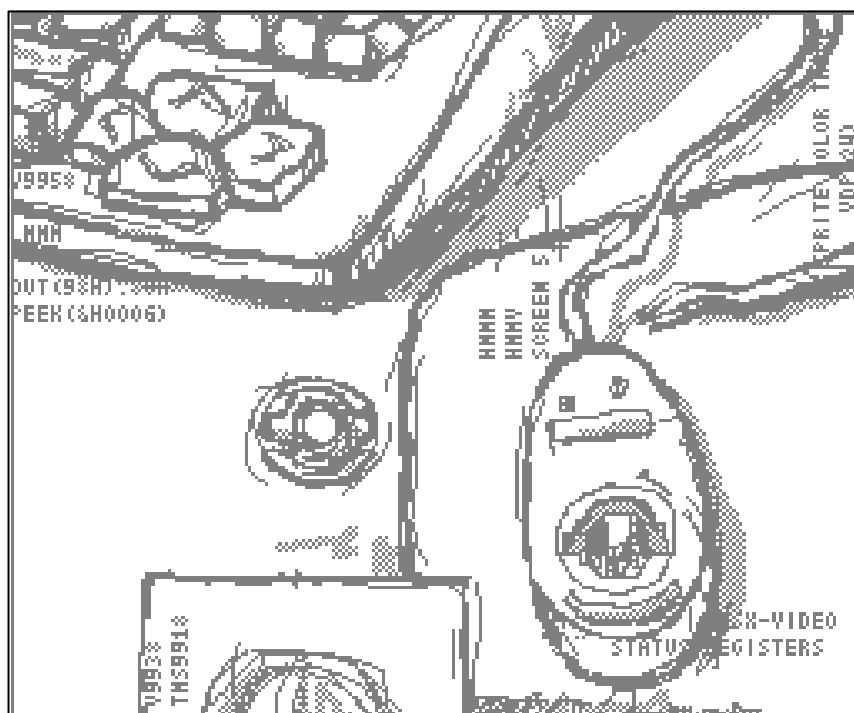
DINTPR:		
	PUSH	DE
	LD	E,2
	JR	INTPRG
EINTPR:		
	PUSH	DE
	LD	E,3
INTPRG:		
	LD	D,0
	PUSH	IX
	PUSH	IY
	CALL	EXTBIO
	EI	
	POP	IY
	POP	IX
	POP	DE
	RET	

第 2 部

第 2 部

VDP

MSX では画面表示を CPU が直接行うのではなく、VDP (Video Display Processor) と呼ばれる専用 LSI を介して行います。この部では、この LSI を BIOS を通さずに直接操作する方法を説明します。



1章 VDP に関する基礎知識

第2部

MSX では画面表示を CPU が直接行うのではなく、VDP (Video Display Processor) と呼ばれる専用 LSI を介して行います。この章では VDP を操作するに当たっての基本的な知識を示します。

2.1.1 VDP の種類

VDP には MSX のバージョンに応じて、いくつかの種類が存在します。MSX のバージョンと VDP の種類の対応は表 2.1.1 の通りです。

表 2.1.1 MSX のバージョンと VDP

機種	VDP
MSX1	TMS9918A
MSX2	V9938
MSX2+	V9958
MSXturboR	V9958

2.1.2 VDP のレジスタ

VDP は、その機能を外部から利用する為の「VDP レジスタ」を内部に持っています。VDP レジスタ (以下レジスタ) は、その機能によって、次の3つに大別されます。

- ・コントロールレジスタ
- ・ステータスレジスタ
- ・パレットレジスタ

2.1.2.1 コントロールレジスタ (R#0 ~ R#23, R#25 ~ R#27, R#32 ~ R#46)

VDP 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 という記号で表します。R#0 ~ R#23, R#25 ~ R#27 は主に画面モードの設定に使用し、R#32 ~ R#46 は VDP コマンドの実行に使用します。

VDP によっては存在しないレジスタもあるので注意が必要です。表 2.1.2 に記すのは、VDP と存在するレジスタ番号の関係です。

表 2.1.2 VDP とコントロールレジスタの対応

コントロールレジスタ番号	VDP の種類		
R# 0 ~ R# 7	TMS9918A	V9938	V9958
R# 8 ~ R#23		V9938	V9958
R#25 ~ R#27			V9958
R#32 ~ R#46		V9938	V9958

2.1.2.2 ステータスレジスタ (S#0 ~ S#9)

VDP から得られる各種情報を読み取る為に存在する、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。

TMS9918A には S#0 のみ存在します。

2.1.2.3 パレットレジスタ (P#0 ~ P#15)

カラーパレットを設定する為に存在する、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。一つのパレットレジスタは 9 ビットの長さを持ち、RGB 各色につき、それぞれ 3 ビットずつを割り当てています。

TMS9918A は、固定パレットであり、パレットレジスタは存在しません。また、カラーコード 0 は透明色として扱われます。

カラーパレットは、システムの初期化時に表 2.1.3 のように設定されます。この設定値は、TMS9918A の固定パレットと同等です。RGB の値はそれぞれの輝度を表し、0 が最小、7 が最大です。

GRAPHIC7 (SCREEN 8) では、カラーパレットを使用しません。また、GRAPHIC5 (SCREEN 6) では、P#0 ~ P#3 のみ使用します。

表 2.1.3 システム初期化時のカラーパレット

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
10	6	6	1
11	6	6	4
12	4	1	1
13	2	6	5
14	5	5	5
15	7	7	7

2.1.3 VRAM

TMS9918A には 16KB の VRAM を、V9938・V9958 には 128KB の VRAM と 64KB の拡張 RAM を接続出来ます。このメモリ空間にアクセスする為に、TMS9918A は 14 ビット、V9938・V9958 は 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 VDP が管理するものであり、CPU が直接アクセスする事は出来ません。

V9938・V9958 の拡張 RAM は、その内容を画面に表示することは出来ませんが、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出来ます。ただし、MSX 規格にこの拡張 RAM は入っていません。

2.1.4 I/O ポート

VDP は CPU とデータをやり取りする為に、2 つ、もしくは 4 つの I/O ポートを持っています。このポートは CPU の I/O 空間を通して接続されています。各ポートの機能を表 2.1.4 に記します。

なお、表中 m, n で表されたアドレスは、それぞれ MAIN ROM の 0006H, 0007H 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、必ずこの番地を参照して VDP へのアクセスを行ってください。

表 2.1.4 VDP のポート

ポート	アドレス	機能
ポート#0 (READ)	m	VRAM からデータを読み出す
ポート#0 (WRITE)	n	VRAM にデータを書き込む
ポート#1 (READ)	m+1	ステータスレジスタを読み出す
ポート#1 (WRITE)	n+1	コントロールレジスタにデータを書き込む
ポート#2 (WRITE)	n+2	パレットレジスタにデータを書き込む
ポート#3 (WRITE)	n+3	間接指定されたコントロールレジスタにデータを書き込む

注意 m の値は MAIN ROM の 0006H を参照して得る

n の値は MAIN ROM の 0007H を参照して得る

2章 基本入出力

第2部

この章では VDP の操作に必要な、コントロールレジスタ、ステータスレジスタ、パレットレジスタ、および VRAM にアクセスする方法等について説明します。

2.2.1 コントロールレジスタのアクセス

VDP のコントロールレジスタ (R#n) にデータをセットする方法には、TMS9918A では直接指定と呼ばれる方法が、V9938・V9958 では直接指定に加え間接指定と呼ばれる方法が存在します。

2.2.1.1 直接指定

ポート#1(WRITE)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注意が必要です。例えば、割り込みルーチンが VDP をアクセスしている場合には、割り込みを禁止してから出力を行います。

この方法は TMS9918A・V9938・V9958 で共通に使えます。

ポート#1 にデータを出力

ポート#1

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

 1st byte

ポート#1 にレジスタ番号を出力 (上位 2bit は常に 10)

ポート#1

1	0	R5	R4	R3	R2	R1	R0
---	---	----	----	----	----	----	----

 2nd byte
R5~R0 コントロールレジスタ番号 n (0~46)

2.2.1.2 間接指定

コントロールレジスタ R#17 (Control register pointer) に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モードとオートインクリメントモードの 2 つのモードがあります。

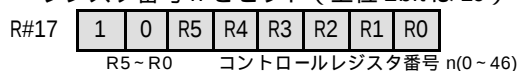
この方法は TMS9918A では使用出来ません。

2.2.1.2.1 非オートインクリメントモード

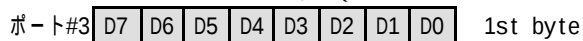
直接指定で R#17 に「レジスタ番号 OR 80H」をセットし、ポート#3 にデータを出力します。R#17 の値は、直接指定でのみ変更可能です。

非オートインクリメントモードでは R#17 の値は変化しないので、データを出力した後に再び同一レジスタにデータをセットしたい場合、R#17 の再セットを行う必要はありません。このモードは VDP コマンドの実行時等、同一レジスタに連続してデータを送る場合に使用します。

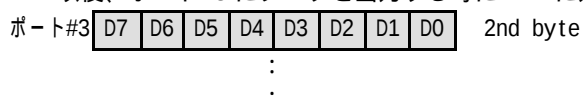
レジスタ番号 n をセット (上位 2bit は 10)



ポート#3 にデータを出力 (このデータは R#n に入る)



以後、ポート#3 にデータを出力する毎に R#n に入る

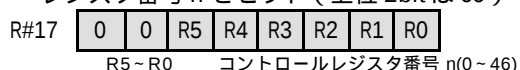


2.2.1.2.2 オートインクリメントモード

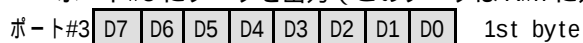
直接指定で R#17 に「レジスタ番号」をセットし、ポート#3 にデータを出力します。R#17 の値は、直接指定でのみ変更可能です。

オートインクリメントモードでは、ポート#3 にデータを出力する度に、R#17 の値がオートインクリメントされます。このモードは VDP コマンドのセット、スクリーンモードの変更等、連続した多数のレジスタを書き換える場合に使用します。

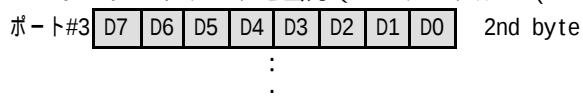
レジスタ番号 n をセット (上位 2bit は 00)



ポート#3 にデータを出力 (このデータは R#n に入る)



ポート#3 にデータを出力 (このデータは R#(n+1)に入る)



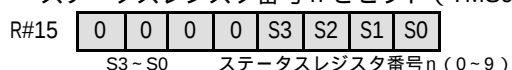
2.2.2 ステータスレジスタのアクセス

VDP のステータスレジスタ(S#n)を読み出すには、コントロールレジスタ 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 (READ) を読み出します。

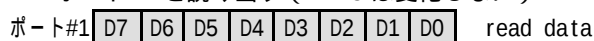
MSX の割り込みルーチンは、R#15 に 0 がセットされている事を前提としています。従って、R#15 を書き換える場合には、必ず割り込みを禁止し、目的の処理が終了した後、R#15 に 0 をセットしてから割り込みを許可して下さい。

なお、TMS9918A には、S#1 ~ S#9 及び R#15 は存在せず、ポート#1 (READ) を読み出す操作のみを行う事で、必ず S#0 の値が返ります。

ステータスレジスタ番号 n をセット (TMS9918A では不要)



ポート#1 を読み出す (R#15 は変化しない)



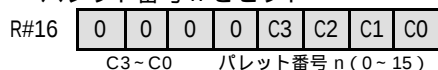
2.2.3 パレットレジスタのアクセス

V9938・V9958 のパレットレジスタ (P#n)にデータをセットする為には、コントロールレジスタ R#16(Color palette address pointer) を使用します。

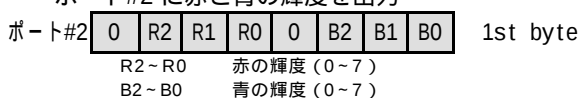
まず、R#16 にパレット番号をセットし、次にポート#2 にデータを 2 バイト連続で出力します。R#16 はポート#2 にデータを 2 バイト書き込むとオートインクリメントされます。

なお、TMS9918A にはパレット機能が無い為、パレットレジスタは存在しません。

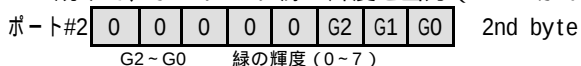
パレット番号 n をセット



ポート#2 に赤と青の輝度を出力



続けて、ポート#2 に緑の輝度を出力 (R#16 がインクリメントされる)



2.2.4 VRAM(VIDEO RAM)のアクセス

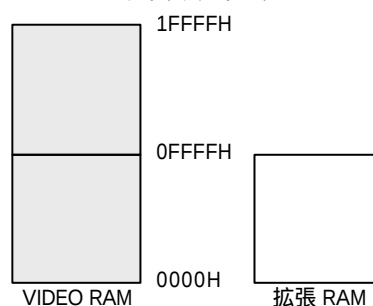
TMS9918A には 16KB の VRAM を、V9938・V9958 には 128KB の VRAM と 64KB の拡張 RAM を接続出来ます。V9938・V9958 の拡張 RAM は、その内容を画面に表示することは出来ませんが、VDP コマンド実行時にはワークエリアとして、VRAM と同等に扱うことが出来ます。ただし、MSX 規格にこの拡張 RAM は入っていないので、R#45 のビット 6 には常に 0 を設定して下さい。

VRAM アクセスは次の手順で行います。

バンク切り替え (VRAM / 拡張 RAM)
 アドレスカウンタセット (A16 ~ A14)
 アドレスカウンタセット (A7 ~ A0)
 アドレスカウンタセット (A13 ~ A8)
 読み出し・書き込み指定
 データの読み出し・書き込み

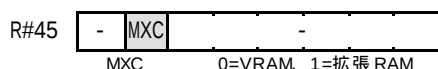
なお、TMS9918A を使用する場合は、V9938・V9958 でアドレスカウンタの上位 3 ビットが前回のアクセスと同じ場合は、手順 ~ のみを行います。

図 2.2.1 VDP のメモリマップ
アドレスカウンタ



バンク切り替え (VRAM ~ 拡張 RAM)

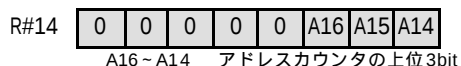
R#45 のビット 6 に、VRAM を使う場合は 0、拡張 RAM を使う場合は 1 をセットします。MSX では常に 0 をセットして下さい。



アドレスカウンタセット (A16～A14)

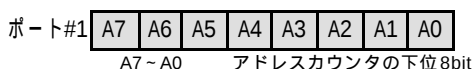
コントロールレジスタ R#14 (VRAM access base address register) を使って、アドレスカウンタの上位 3 ビット (A16～A14) をセットします。

このレジスタは、TMS9918A 互換モード (2.2.6 TMS9918A 互換モードを参照) の時には、オートインクリメントされません。



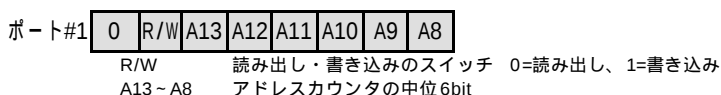
アドレスカウンタセット (A7～A0)

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。



アドレスカウンタセット (A13～A8) 読み出し・書き込み指定

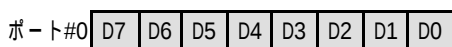
ポート#1 に、アドレスカウンタの中段 6 ビット (A13～A8) にセットするデータ、及び読み出し・書き込みの指定を出力します。ビット 6 が、読み出し・書き込みのスイッチになっており、0 を指定すると読み出し、1 を指定すると書き込みになります。



データの読み出し・書き込み

ポート#0 にアクセスする事で、VRAM の読み出し・書き込みが出来ます。ポート#0 にアクセスする度にアドレスカウンタがオートインクリメントされるので、連続して読み出し書き込みを行えます。

ただし、TMS9918A 互換モードの時には、A16～A14 がオートインクリメントされません。



V9938・V9958 で VRAM のアクセスを行うには、VDP コマンドを使用する方法もあります (これについては 6 章 VDP コマンドを参照)。

2.2.5 VDP の I/O アクセスに関する注意点

Z80 使用時に直接 I/O ポートを用いて VDP にアクセスする際、短い間隔でアクセスを行うと VDP が誤動作を起こす事があります。これを防ぐ為には十分な間隔を空けてアクセスしなければなりません。アクセス対象・画面モードによって必要な間隔は異なります。まず VRAM アクセスですが、SCREEN 0 では 23 クロック、SCREEN 1～12 では 18 クロックの間隔でアクセスすれば誤動作は起きません。VDP レジスタのアクセスでは、14 クロック間隔でアクセスすれば問題ありません (これらの値は筆者が独自に調査したもので、M1 サイクルのウェイトを含めた値です)。

具体的には、SCREEN 0 における VRAM の連続アクセスでは INIR, OTIR を用いるか、これらの命令以上の間隔を空けてアクセスを行って下さい。同様に SCREEN 1～12 における VRAM アクセスでは OUTI, INI を、VDP レジスタのアクセスでは IN r(C), OUT (C),r を使用して下さい。

V9958 には、誤動作を防ぐ為に VDP がウェイト信号を発生する機能がありますが、MSX2+, MSXturboR では使用出来ません (V9958 のウェイト出力が Z80 に入力されていない為)。また、MSXturboR で R800 を使用中は、8.66 μ 秒以内に連続してアクセスするとシステムが自動的にウェイトを発生します。

2.2.6 TMS9918A 互換モード

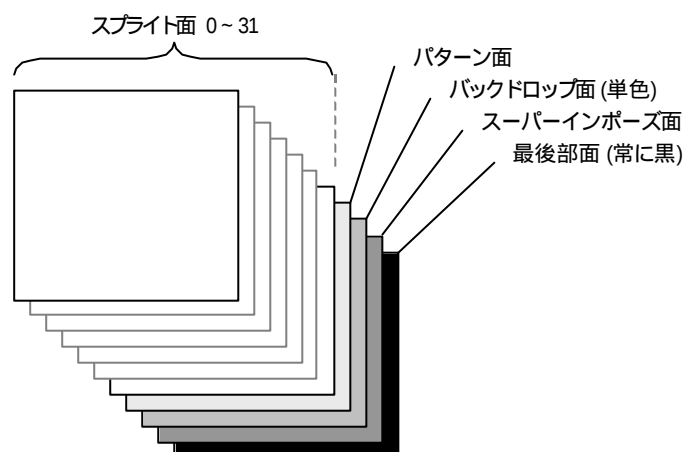
V9938・V9958 は TMS9918A との上位互換性を持っています。互換性は、TMS9918A 互換モードと呼ばれる画面モードを使用する事で確保されます。表 2.2.1 に、TMS9918A 互換モードに含まれる画面モードを示します。

表 2.2.1 TMS9918A 互換モード

VDP モード	BASIC の SCREEN モード
TEXT1	SCREEN 0 (40 字モード)
GRAPHIC1	SCREEN 1
GRAPHIC2	SCREEN 2
MULTI COLOR	SCREEN 3

2.2.7 画面構成

TMS9918A・V9938・V9958 の画面は、幾つかの独立した面が重なって構成されています。その構成を次に示します。



スプライト面	スプライトを表示する画面で、スプライトアトリビュートテーブル、スプライトパターンジェネレータテーブル、スプライトカラーテーブルによって表示内容が決定されます
パターン面	文字、画像を表示する面で、パターンネームテーブル、パターンジェネレータテーブルによって表示内容が決定されます
バックドロップ面	背景となる画面で、カラーレジスタによって面の色を指定します
スーパーインポーズ面	V9938, V9958 にのみ存在する画面で、外部信号 (テレビ等) が表示されます
最後部面	全ての画面の下に存在する黒い画面で、操作する事は出来ません

2.2.8 VDP に関連するワークエリア

VDP のレジスタは、ステータスレジスタを除いて全て書き込み専用です。その為、レジスタの特定のビットを書き換える場合等に備えて、これらの値を保存しておくワークエリアが存在します。ユーザーが VDP レジスタに値を書き込む場合、これらのワークエリアに同じ値を書き込まないと、それ以後の BIOS や BASIC の動作は保証されません。BIOS や BASIC を使用しない場合は、書き込みを行わなくても問題ありません。

VDP に関連する主なワークエリアを表 2.2.2 (章末) に挙げておきます。

2.2.9 パレットテーブル

パレットレジスタ (P#0 ~ P#15) は書き込み専用の為、システムによってその値を保持する領域 (パレットテーブル) が VRAM 上に確保されます。パレットテーブルは、1色2バイトの長さを持ち、全体で32バイトあります。型式は次の様になっています。

パレットを変更する BIOS や BASIC のステートメントを使用すると、副作用として VRAM の値が変化する事になるので、パレットテーブルの領域を表示など他の目的に使う場合には注意が必要です。

base address

+00	0	R2	R1	R0	0	B2	B1	B0	P#0
+01	0	0	0	0	0	G2	G1	G0	
:									
+30	0	R2	R1	R0	0	B2	B1	B0	P#15
+31	0	0	0	0	0	G2	G1	G0	

R2 ~ R0

赤の輝度 (0 ~ 7)

B2 ~ B0

青の輝度 (0 ~ 7)

G2 ~ G0

緑の輝度 (0 ~ 7)

パレットテーブルの先頭アドレスと画面モードの関係を表 2.2.3 に示します。

表 2.2.3 パレットのセーブエリア (MSX2 以降)

VDP 画面モード	BASIC スクリーンモード	パレットテーブル先頭アドレス
TEXT1	SCREEN α (40 字モード)	00400H
TEXT2	SCREEN α (80 字モード)	00F00H
GRAPHIC1	SCREEN 1	02020H
GRAPHIC2	SCREEN 2	01B80H
MULTI COLOR	SCREEN 3	02020H
GRAPHIC3	SCREEN 4	01B80H
GRAPHIC4	SCREEN 5	07680H
GRAPHIC5	SCREEN 6	07680H
GRAPHIC6	SCREEN 7	0FA80H
GRAPHIC7	SCREEN 8	0FA80H
YJK/RGB 混在	SCREEN 10	0FA80H
YJK/RGB 混在	SCREEN 11	0FA80H
YJK	SCREEN 12	0FA80H

2.2.10 BASIC の VDP(n)関数

BASIC で VDP レジスタにアクセスする方法の一つに、VDP(n) 関数があります。VDP(n) 関数と VDP レジスタの関係を表 2.2.4 に示します。

表 2.2.4 VDP(n)関数

n	VDP レジスタ
-1 ~ -9	S#1 ~ S#9
0 ~ 7	R#0 ~ R#7
8	S#0
9 ~ 24	R#8 ~ R#23
26 ~ 28	R#25 ~ R#27

VDP(n) 関数ではその値を参照出来ますが、これは 2.2.8 VDP に関連するワークエリアのワークエリアの内容を参照する事と同等です。

表 2.2.2 VDP に関連するワークエリア

アドレス	ラベル名	長さ	初期値	内容
F3B3	TXTNAM	2	0000H	SCREEN 0のパターンネームテーブル
F3B5	TXTCOL	2	0800H	SCREEN 0のカラーテーブル (MSX2以降)
F3B7	TXTCGP	2	0800H	SCREEN 0のパターンジェネレータテーブル
F3BD	T32NAM	2	1800H	SCREEN 1のパターンネームテーブル
F3BF	T32COL	2	2000H	SCREEN 1のカラーテーブル
F3C1	T32CGP	2	0000H	SCREEN 1のパターンジェネレータテーブル
F3C3	T32ATR	2	1800H	SCREEN 1のスプライトアトリビュートテーブル
F3C5	T32PAT	2	3800H	SCREEN 1のスプライトジェネレータテーブル
F3C7	GRPNAM	2	1800H	SCREEN 2のパターンネームテーブル
F3C9	GRPCOL	2	2000H	SCREEN 2のカラーテーブル
F3CB	GRPCGP	2	0000H	SCREEN 2のパターンジェネレータテーブル
F3CD	GRPATR	2	1800H	SCREEN 2のスプライトアトリビュートテーブル
F3CF	GRPPAT	2	3800H	SCREEN 2のスプライトジェネレータテーブル
F3D1	MLTNAM	2	0800H	SCREEN 3のパターンネームテーブル
F3D5	MLTCGP	2	0000H	SCREEN 3のパターンジェネレータテーブル
F3D7	MLTATR	2	1800H	SCREEN 3のスプライトアトリビュートテーブル
F3D9	MLTPAT	2	3800H	SCREEN 3のスプライトジェネレータテーブル
F3DF	RG0SAV	1		コントロールレジスタ R#0 の値
F3E0	RG1SAV	1		コントロールレジスタ R#1 の値
F3E1	RG2SAV	1		コントロールレジスタ R#2 の値
F3E2	RG3SAV	1		コントロールレジスタ R#3 の値
F3E3	RG4SAV	1		コントロールレジスタ R#4 の値
F3E4	RG5SAV	1		コントロールレジスタ R#5 の値
F3E5	RG6SAV	1		コントロールレジスタ R#6 の値
F3E6	RG7SAV	1		コントロールレジスタ R#7 の値
F3E7	STATFL	1		ステータスレジスタ S#0 の値
F3E9	FORCLR	1	15	前景色省略値
F3EA	BAKCLR	1	4	背景色省略値
F3EB	BDRCLR	1	7	周辺色省略値
F91F	CGPNT	3		フォント格納スロットアドレスとアドレス
F922	NAMBAS	2		現在のパターンネームテーブル
F924	CGPBAS	2		現在のパターンジェネレータテーブル
F926	PATBAS	2		現在のスプライトジェネレータテーブル
F928	ATRBAS	2		現在のスプライトアトリビュートテーブル
FAF5	DPPAGE	1		ディスプレイページ番号 (MSX2以降)
FAF6	ACPAGE	1		アクティブページ番号 (MSX2以降)
FAFC	MODE	1		VRAM サイズ等 (MSX2以降、VDPに関わる部分のみを記す) bit5 0=SCREEN 10、1=SCREEN 11 (MSX2+以降) bit3 BIOSを使ってVRAMアドレスを指定する時、SCREEN 3以下でVRAM アドレスを、1=マスクしない、0=マスクする (AND 3FFFHをとる) bit2~1 VRAMのサイズが、00=16KB、01=64KB、10=128KB
FCAF	SCRMOD	1		スクリーンモードの番号
FCB0	OLDSCR	1		スクリーンモード保存場所
FFE7	RG8SAV	1		コントロールレジスタ R#8 の値
FFE8	RG9SAV	1		コントロールレジスタ R#9 の値
FFE9	RG10SA	1		コントロールレジスタ R#10 の値
FFEA	RG11SA	1		コントロールレジスタ R#11 の値
FFEB	RG12SA	1		コントロールレジスタ R#12 の値
FFEC	RG13SA	1		コントロールレジスタ R#13 の値
FFED	RG14SA	1		コントロールレジスタ R#14 の値
FFEE	RG15SA	1		コントロールレジスタ R#15 の値
FFEF	RG16SA	1		コントロールレジスタ R#16 の値
FFF0	RG17SA	1		コントロールレジスタ R#17 の値
FFF1	RG18SA	1		コントロールレジスタ R#18 の値
FFF2	RG19SA	1		コントロールレジスタ R#19 の値
FFF3	RG20SA	1		コントロールレジスタ R#20 の値
FFF4	RG21SA	1		コントロールレジスタ R#21 の値
FFF5	RG22SA	1		コントロールレジスタ R#22 の値
FFF6	RG23SA	1		コントロールレジスタ R#23 の値
FFFA	RG25SA	1		コントロールレジスタ R#25 の値
FFFB	RG26SA	1		コントロールレジスタ R#26 の値
FFFC	RG27SA	1		コントロールレジスタ R#27 の値

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、VDP の状態を保持するステータスレジスタについて説明します。

2.3.1 コントロールレジスタ

R#0 ~ R#23, R#25 ~ R#27, R#32 ~ R#46 (Write only)

コントロールレジスタは VDP を制御する、書き込み専用のレジスタ群です。

2.3.1.1 モードレジスタ

VDP の各種動作モードを設定するレジスタ群です。

R#0	0	DG	IE2	IE1	M5	M4	M3	0	Mode register 0
	DG	カラーバスの設定							
		1=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, MSX2+, MSXturboR で使用可能) 0=カラーバスを使用しない							
	IE2	ライトペンによる割り込みを 1=使用する、0=使用しない (MSXでは、この機能は使用していないので常に0を指定する)							
	IE1	水平帰線による割り込みを 1=使用する、0=使用しない							
	M5 ~ M3	画面モードの設定に使用							

TMS9918Aでは、DG, IE2, IE1, M5, M4に0を設定して下さい。

R#1	4/16K	BL	IE0	M1	M2	0	SI	MAG	Mode register 1
	4/16K	TMS9918Aで、VRAMの種類を選択する 1=8Kbit 又は 16Kbit の DRAM を使用、0=4KbitDRAMを使用 (V9938, V9958では常に0をセットする)							
	BL	1=画面表示、0=画面非表示							
	IE0	垂直帰線による割り込みを 1=使用する、0=使用しない							
	M1 ~ M2	画面モードの設定に使用							
	SI	スプライトのサイズ 1=16×16 ドット、0=8×8 ドット							
	MAG	スプライトを 1=拡大する、0=拡大しない							

M5 ~ M1 の設定値と画面モードの対応は表 2.3.1 の通りです。

表 2.3.1 画面モードとモードレジスタ

M5 ~ M1	VDPの画面モード	BASICのスクリーンモード
0 0 0 0 1	TEXT1	SCREEN 0 (40 字モード)
0 1 0 0 1	TEXT2	SCREEN 0 (80 字モード)
0 0 0 0 0	GRAPHIC1	SCREEN 1
0 0 1 0 0	GRAPHIC2	SCREEN 2
0 0 0 1 0	MULTI COLOR	SCREEN 3
0 1 0 0 0	GRAPHIC3	SCREEN 4
0 1 1 0 0	GRAPHIC4	SCREEN 5
1 0 0 0 0	GRAPHIC5	SCREEN 6
1 0 1 0 0	GRAPHIC6	SCREEN 7
1 1 1 0 0	GRAPHIC7	SCREEN 8
1 1 1 0 0	YJK/RGB 混在	SCREEN 10(R#25 と併せて設定)
1 1 1 0 0	YJK/RGB 混在	SCREEN 11(R#25 と併せて設定)
1 1 1 0 0	YJK	SCREEN 12(R#25 と併せて設定)

R#8 MS LP TP CB VR 0 SPD BW Mode register 2

MS	マウスを 1=使用する (カラーバスは入力モード)、0=使用しない (カラーバスは出力モード) (MSX では、この機能は使用していないので常に 0 を指定する)
LP	ライトペンを 1=使用する、0=使用しない (MSX では、この機能は使用していないので常に 0 を指定する)
TP	カラーコード 0 の色を 1=カラーパレットの色にする、0=透明色として扱う SCREEN 0 では、前景色として用いた時は常に透明色として扱われ、背景色として用いた時は常にカラーパレットの色である
CB	カラーバスを 1=入力モードにする、0=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 システムは常に 1 をセットする
SPD	スプライトを 1=表示しない、0=表示する
BW	1=白黒 32 階層、0=カラー (Composite encoder にのみ有効) (MSX ではこの機能は使用していない)

R#9 LN 0 S1 S0 IL EO NT DC Mode register 3

LN	1=縦 212 ドット表示、0=縦 192 ドット表示
S1~S0	同期モード選択 (表 2.3.2 を参照)
IL	1=インターレース (完全 NTSC タイミング)、0=ノン・インターレース (不完全 NTSC タイミング)
EO	Even/Odd field で 1=2 枚の絵を 1/60 秒毎に交互に表示 (PAL の場合は 1/50 秒毎)、0=同じ絵を表示
NT	1=PAL (313line)、0=NTSC (262line) RGB 出力のみ有効
DC	DLCLK 端子を 1=入力モードにする、0=出力モードにする 普通のアプリケーションプログラムが書き換えてはならない システムは 0 をセットする

V9938, V9958 では、同期モードを S1, S0 の 2 ビットで指定します。

表 2.3.2 同期モード

S1	S0	同期モード	Ys	用途
0	0	パソコン同期	常に V9938, V9958 を選択(0)	V9938, V9958 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・デジタイズ等
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

R#25 0 CMD VDS YAE YJK WTE MSK SP2

CMD	コマンド機能を 0=GRAPHIC4~7 モードでのみ使用可能にする 1=全画面モードで使用可能にする (GRAPHIC4~7 では V9938 と同等に機能する。 それ以外の画面モードでは GRAPHIC7 として動作し、座標系は GRAPHIC7 モードの (X,Y) 座標系を用いる)
VDS	VDP の端子 8 を制御する (普通のアプリケーションプログラムが書き換えてはならない) システムは 0 をセットする
YAE	YJK 方式のデータフォーマットを選択 0=YJK、1=YJK/RGB 混在
YJK	VRAM 上のデータを 0=RGB 方式として扱う、1=YJK 方式と見なし、これを RGB 信号 (各 5bit) に変換しアナログ出力する (スプライトの表示色はパレットが有効) YJK を 1 にすると画面全体が右に 4 ドットずれる
WTE	ウェイト機能を 0=無効にする (TMS9918A, V9938 と同等) 1=有効にする (CPU が VRAM をアクセスした際、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) レジスタとパレットへのアクセス未完了及びコマンドのデータレディによるウェイト機能は存在しない
MSK	画面左端 8 ドットを、0=マスクしない、1=マスクしバックドロップの色を表示する。 GRAPHIC5, 6 では 16 ドットのマスク
SP2	ハードウェア横スクロールを使用する時に用いる ハードウェア横スクロール時、水平方向画面サイズを 0=1 ページとする (スクロールは 1 ページ内で行われ、隠れた部分が反対側に表れる) 1=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る) GRAPHIC4 未満では無効 (必ず 1 ページでスクロールする)

ウェイト機能は MSX では使用していません。WTE を 1 にする時は、その前にダミーの VRAM アクセスを行って下さい。

YAE、YJK の各ビットの詳細については、2.4.11 YJK/RGB 混在(SCREEN 10・11)と 2.4.12 YJK(SCREEN 12)を参照してください。

2.3.1.2 テーブルベースアドレスレジスタ

VDP に対して VRAM 上の各テーブルの先頭アドレスを宣言する為のレジスタ群です。

画面モードによって、設定出来るアドレスに制限があるので注意して下さい。詳細は、4 章 VDP の画面モードを参照して下さい。

TMS9918A では、A16～A14 には 0 をセットして下さい。

R# 2	0	A16	A15	A14	A13	A12	A11	A10	Pattern name table base address register
R# 4	0	0	A16	A15	A14	A13	A12	A11	Pattern generator table base address register
R# 3	A13	A12	A11	A10	A9	A8	A7	A6	Color table base address register low
R#10	0	0	0	0	0	A16	A15	A14	Color table base address register high
R# 5	A14	A13	A12	A11	A10	A9	A8	A7	Sprite attribute table base address register low
R#11	0	0	0	0	0	0	A16	A15	Sprite attribute table base address register high
R# 6	0	0	A16	A15	A14	A13	A12	A11	Sprite pattern generator table base address register

2.3.1.3 カラーレジスタ

VDP の表示色、ブリンク等を制御する為のレジスタ群です。

Text color/Back drop color register

R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	(GRAPHIC7 モード時以外)
R#7	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0	(GRAPHIC7 モード時)

画面モードによって機能が変わります。

TC3～TC0 TEXT1, TEXT2 モードにおけるテキストの色を指定

カラーコード 0 を指定した場合、TP (R#8 の bit5) の値にかかわらず必ず透明色になる

BD3～BD0 GRAPHIC7 以外の画面モードにおけるバックドロップの色を指定

TEXT1, TEXT2, YJK/RGB 混在モードでカラーコード 0 を指定した場合、TP (R#8 の bit5) の値にかかわらず必ずカラーパレットの色になる

BD7～BD0 GRAPHIC7 モードにおけるバックドロップの色を指定

Text color/Back color register

R#12	T23	T22	T21	T20	BC3	BC2	BC1	BC0
------	-----	-----	-----	-----	-----	-----	-----	-----

TEXT2 モードにおいて、ブリンク属性がセットされている時には、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

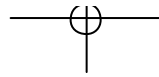
カラーコード 0 を指定した場合、TP (R#8 の bit5) の値にかかわらず必ず透明色になる
(パターンの 1 の部分の色はパターンの 0 の部分の色になる)

BC3～BC0 パターンの 0 の部分の色を指定

カラーコード 0 を指定した場合、TP (R#8 の bit5) の値にかかわらず必ずカラーパレットの色になる

Blinking period register

R#13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0
------	-----	-----	-----	-----	-----	-----	-----	-----



2.3.1.3.1 TEXT2 モードにおける動作

R#7、R#12 で指定された色を交互に表示します。

ON3～ON0	R#12 で指定された色の表示時間
OF3～OF0	R#7 で指定された色の表示時間

2.3.1.3.2 ビットマップモード (GRAPHIC4～7, YJK/RGB 混在, YJK) における動作

2 ページの画面を交互に表示します。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、交互表示を開始します。

ON3～ON0	偶数ページの表示時間
OF3～OF0	奇数ページの表示時間

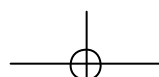
表示時間は 1/60 秒の倍数値で、ON3～ON0、OF3～OF0 の値によって表 2.3.3 の様に变化します (PAL では 1/50 秒の倍数値をとる)。

表 2.3.3 ブリンクのデータと時間 (NTSC モード)

ON3～ON0, OF3～OF0	時間 (ms)	フレーム数 60 フレーム / 秒
0 0 0 0	0.0	0
0 0 0 1	166.9	10
0 0 1 0	333.8	20
0 0 1 1	500.6	30
0 1 0 0	667.5	40
0 1 0 1	834.4	50
0 1 1 0	1001.3	60
0 1 1 1	1168.2	70
1 0 0 0	1335.1	80
1 0 0 1	1501.9	90
1 0 1 0	1668.8	100
1 0 1 1	1835.7	110
1 1 0 0	2002.6	120
1 1 0 1	2169.5	130
1 1 1 0	2336.3	140
1 1 1 1	2503.2	150

R#20	0	0	0	0	0	0	0	0	Color burst register 1
R#21	0	0	1	1	1	0	1	1	Color burst register 1
R#22	0	0	0	0	0	1	0	1	Color burst register 3

これらのレジスタには、上記の値が起動時にセットされます。この値を全て 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事が出来ます。しかし、MSX のコンポジットビデオ出力は多くの場合、VDP の RGB 出力から外部回路により合成される為、このレジスタの機能は使用されません。



2.3.1.4 ディスプレイレジスタ

ディスプレイ上の表示位置を制御するレジスタ群です。

R#19

IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
-----	-----	-----	-----	-----	-----	-----	-----

 Interrupt line register

IL7 ~ IL0

割り込みを発生させる走査線番号

実際に水平帰線割り込みが発生するのは、192ラインモードでは走査線番号0 ~ 234 (212ラインモードでは0 ~ 244) のみであり、それ以外の値を指定しても水平帰線割り込みは発生しない。

V9938, V9958 では、VDP が特定の走査線の出力を終えた時に、割り込みを発生させる事が出来ます。

割り込みを発生させるには、このレジスタに割り込みを発生させる走査線番号を設定し、R#0 のビット4に1をセットします。割り込みが発生すると、S#1 のビット0 がセットされ、割り込みベクタ (0038H) が割り込みを禁止されてコールされます。通常は、0038H からシステムの割り込みルーチンへ進みます (詳細は、2.7.3 水平帰線割り込みを参照)。

R#18

V3	V2	V1	V0	H3	H2	H1	H0
----	----	----	----	----	----	----	----

 Display adjust register

V3 ~ V0

縦方向の表示位置 (-7 ~ +8)

H3 ~ H0

横方向の表示位置 (-7 ~ +8)

ディスプレイ上の表示位置を補正する為のレジスタです。

縦方向の補正 (V3 ~ V0)	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
	上							中央								下
SET ADJUST 命令の値	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7	+8
横方向の補正 (H0 ~ H0)	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
	左							中央								右
SET ADJUST 命令の値	-7	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6	+7	+8

R#23

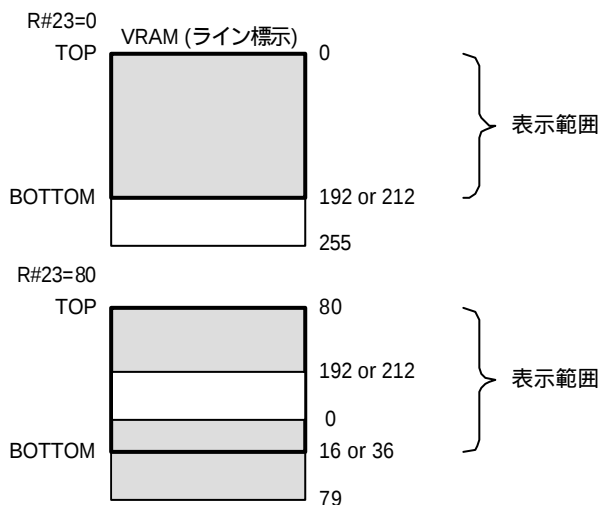
D07	D06	D05	D04	D03	D02	D01	D00
-----	-----	-----	-----	-----	-----	-----	-----

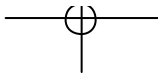
 Display offset register

D07 ~ D00 表示開始ライン (TEXT1, TEXT2 では D02 ~ D00 のみが有効です)

表示開始ラインをセットする為のレジスタです。このレジスタの値を連続的に変える事によって、ハードウェア縦スクロールを行えます。ただし、スクロールは256ライン目で上下がつながる形で行われる為、スプライトテーブル等は別のページに置く必要があります。

図 2.3.1 オフセットレジスタの設定例





R#26	0	0	H08	H07	H06	H05	H04	H03	Horizontal scroll high
R#27	0	0	0	0	0	H02	H01	H00	Horizontal scroll low

H08 ~ H00 横スクロール量をドット単位で指定 (GRAPHIC5, 6は2ドット単位)
H08 ~ H03の値に関して、左方向へ8ドット単位で設定値だけシフト (H08は1ページスクロールの時は無効)
H02 ~ H00の値に関して、右方向へ1ドット単位で設定値だけシフト
ただし、TEXT1, TEXT2ではH02 ~ H00のみ有効。

V9958によるハードウェア横スクロール時のVRAMアクセスは8ドット単位 (GRAPHIC5, 6は16ドット単位)で行われる為、R#27が0の時以外は画面左端部の表示用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為に、MSK (R#25のbit1)によるマスクが必要になります。

SP2 (R#25のbit0)に水平方向の画面サイズを設定出来ます。SP2=0で1画面の両端がつながってスクロールします。SP2=1では、パターンネームテーブルを奇数ページにセットすることで、偶数ページ、奇数ページがつながってスクロールします。

2.3.1.5 アクセスレジスタ

VDPのレジスタやVRAMをアクセスする時に使用するレジスタ群です。詳細は、2章 基本入出力を参照して下さい。

R#14	0	0	0	0	0	A16	A15	A14	VRAM Access base address register
A16 ~ A14 アドレスカウンタの上位3bit									

V9938, V9958のVRAMをアクセスする時に、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 (アドレスカウンタのbit13)からの繰り上がりを受けて自動的にインクリメントされます。ただし、TMS9918A 互換モードでは、このレジスタのインクリメントは行われません。

R#15	0	0	0	0	S3	S2	S1	S0	Status register pointer
S3 ~ S0 ステータスレジスタ番号 (0 ~ 9)									

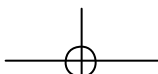
V9938, V9958のステータスレジスタ (S#0 ~ S#9)を読み出す際、このレジスタにステータスレジスタ番号 (0 ~ 9)をセットします。

R#16	0	0	0	0	P3	P2	P1	P0	Color palette address register
P3 ~ P0 パレット番号 (0 ~ 15)									

V9938, V9958のパレットレジスタ (P#0 ~ P#15)にアクセスする際、このレジスタにパレット番号 (0 ~ 15)をセットします。

R#17	A11	0	R5	R4	R3	R2	R1	R0	Control register pointer
A11 0=オートインクリメント、1=オートインクリメント禁止 R5 ~ R0 コントロールレジスタ番号 (0 ~ 46)									

V9938, V9958では、このレジスタをポインタとして他のレジスタをアクセスする事が出来ます。また、A11 (bit7)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出来ます。



2.3.1.6 コマンドレジスタ

V9938, V9958 のコマンドを実行する時に使用するレジスタ群です。詳細は6章 VDP コマンドを参照して下さい。

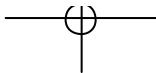
R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X low register
R#33	0	0	0	0	0	0	0	SX8	Source X high register
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y low register
R#35	0	0	0	0	0	0	SY9	SY8	Source Y high register
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X low register
R#37	0	0	0	0	0	0	0	DX8	Destination X high register
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y low register
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y high register
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X low register
R#41	0	0	0	0	0	0	0	NX8	Number of dot X high register
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of dot Y low register
R#43	0	0	0	0	0	0	NY9	NY8	Number of dot Y high register
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	Color register
R#45	0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ	Argument register
R#46	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0	Command register

2.3.2 ステータスレジスタ S#0 ~ S#9(Read only)

VDP の状態を得る為の、読み出し専用のレジスタ群です。

S#0	F	5S	C	5th sprite#	Status register 0
	F			垂直帰線割り込みフラグ	
				垂直帰線割り込み (R#0 の bit5 がセットされている時) が発生すると 1 になる。S#0 を読み出すと 0 になる。	
		5S		第 5 スプライトフラグ	
				F が 0 で、かつ 1 水平線上にスプライトが 5 個 (スプライトモード 2 では 9 個) 並ぶと 1 になり、同時に 5th sprite# がセットされる	
			C	衝突フラグ。スプライトが衝突すると 1 になる。S#0 を読み出すと 0 になる。	
			5th sprite#	1 水平線上にスプライトが 5 個 (スプライトモード 2 では 9 個) 並んだ時の、第 5 (第 9) スプライトの番号がセットされる。	

S#1	FL	LPS	ID#	FH	Status register 1
	FL				ライトペンスイッチ (R#8 の bit6 が 1 の時)
					ライトペンが光を検出すると 1 になる。この時、IE2 (R#0 の bit5) が 1 であれば割り込みが発生する。S#1 を読み出すと 0 になる。
					マウススイッチ 2 (R#8 の bit7 が 1 の時)
					マウスのスイッチ 2 が押されると 1 になる。S#1 を読み出しても 0 にならない。
		LPS			ライトペンスイッチ (R#8 の bit6 が 1 の時)
					ライトペンのスイッチが押されると 1 になる。S#1 を読み出しても 0 にならない。
					マウススイッチ 1 (R#8 の bit7 が 1 の時)
					マウスのスイッチ 1 が押されると 1 になる。S#1 を読み出しても 0 にならない。
			ID#		VDP の ID 番号。V9938 では「00000」。V9958 では「00010」。
			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 (R#0 の bit4 が 1 の時) が発生すると 1 になる。S#1 を読み出すと 0 になる。



FL (bit7) と LPS (bit6) はマウス・ライトペン用のレジスタです。MSX では、V9938 のマウス、ライトペン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。また、V9958 ではこれらの機能は削除されています。

S#2	TR	VR	HR	BD	1	1	E0	CE	Status register 2
	TR	転送レディフラグ。CPU to VRAM, VRAM to CPU 等のコマンドを実行する時は、CPU はこのフラグを見ながらデータの転送を行う。この bit が 1 の時、転送可能。							
	VR	垂直帰線期間 (画面を書き終わって次に書き始めるまでの期間) フラグ。垂直帰線期間中は 1 になる。							
	HR	水平帰線期間 (走査線を書き終わって次に書き始めるまでの期間) 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時 1 になる。							
	E0	表示フィールドフラグ (インターレースモード & 2 画面交互表示の時に意味を持つ)							
		現在の表示フィールドが 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。コマンド実行中は 1 になる。							

S#3	X7	X6	X5	X4	X3	X2	X1	X0	Column register low
S#4	1	1	1	1	1	1	1	X8	Column register high
S#5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Row register low
S#6	1	1	1	1	1	1	E0	Y8	Row register high

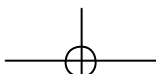
これらのレジスタには、スプライトの衝突座標がセットされます(詳しくは、2.5.3.5 スプライトの衝突を参照)。

S#7	C7	C6	C5	C4	C3	C2	C1	C0	Color register
-----	----	----	----	----	----	----	----	----	----------------

POINT, VRAM to CPU 等のコマンドを実行すると、読み込まれた VRAM のデータがこのレジスタにセットされます。

S#8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0	Border X register low
S#9	1	1	1	1	1	1	1	BX8	Border X register high

サーチコマンドで発見した境界色又は非境界色の X 座標がこのレジスタにセットされます。



4章 VDP の画面モード

この章では、VDP の各画面モードのレジスタの設定方法等について説明します。
VDP と存在する画面モードの関係は表 2.4.1 の様になっています。

表 2.4.1 VDP と画面モード

VDP の画面モード	BASIC のスクリーンモード	VDP の種類		
TEXT1	SCREEN 0 (40 字モード)	TMS9918A	V9938	V9958
TEXT2	SCREEN 0 (80 字モード)		V9938	V9958
GRAPHIC1	SCREEN 1	TMS9918A	V9938	V9958
GRAPHIC2	SCREEN 2	TMS9918A	V9938	V9958
MULTI COLOR	SCREEN 3	TMS9918A	V9938	V9958
GRAPHIC3	SCREEN 4		V9938	V9958
GRAPHIC4	SCREEN 5		V9938	V9958
GRAPHIC5	SCREEN 6		V9938	V9958
GRAPHIC6	SCREEN 7		V9938	V9958
GRAPHIC7	SCREEN 8		V9938	V9958
YJK/RGB 混在	SCREEN 10			V9958
YJK/RGB 混在	SCREEN 11			V9958
YJK	SCREEN 12			V9958

* GRAPHIC6 (SCREEN 7) 以降は VRAM を 128KB 実装した機種でのみ使用可能

2.4.1 TEXT1 (SCREEN 0・40 字モード)

6 × 8 ドットのパターンを要素として、画面を 40 × 24 に分割して表示する画面モードです。使用出来る色数は、画面全体で 2 色です。

TMS9918A, V9938, V9958 の全ての VDP で使用出来ます。

2.4.1.1 特徴

1 パターンのサイズ	横 6 ドット × 縦 8 ドット
画面上のパターン数	横 40 パターン × 縦 24 パターン
パターンの種類	256 種類
パターンの色	2 色 (画面全体)
1 画面に必要な VRAM 容量	3KB

2.4.1.2 関係するレジスタと VRAM 領域

パターンのフォント	パターンジェネレータテーブル (VRAM)
パターンの位置	パターンネームテーブル (VRAM)
パターンの 1 の部分のカラーコード	R#7 上位 4bit
パターンの 0 の部分のカラーコード	R#7 下位 4bit
バックドロップのカラーコード	R#7 下位 4bit

2.4.1.3 レジスタとVRAMの設定

2.4.1.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	0	0	0	0	Mode register 0
R# 1	0	BL	IE0	1	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958のみ)

M5～M1を00001、YAE・YJKを0に設定します。その他のビットは任意です。
システムは、Mode register 0～3をそれぞれ00000000、01110000、00001000、00000000に、R#25を00000000に初期化します。

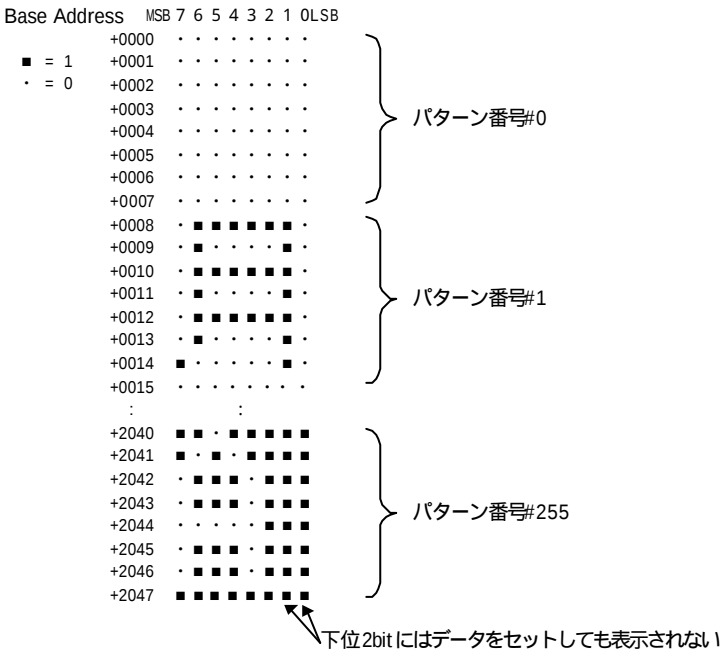
2.4.1.3.2 パターンジェネレータテーブル

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する領域です。
パターンには#0～#255の番号が付けられており、この番号を指定してパターンを画面に表示します。
パターン1個のフォントは8バイトで構成されており、1個のパターンジェネレータテーブルは2KBの大きさを持ちます。表示される時、フォントの各バイトの下位2ビットは無視されます。
パターンジェネレータテーブルの先頭アドレスは、R#4にセットします。指定出来るのは先頭アドレスの上位6ビット(A16～A11、TMS9918AではA13～A11)のみで、下位11ビット(A10～A0)は0と見なされます。よって、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出来るのは、0000Hから2KB単位の位置になります。

R# 4	0	0	A16	A15	A14	A13	A12	A11	Pattern generator table base address register
------	---	---	-----	-----	-----	-----	-----	-----	---

TMS9918Aでは、A16～A14に0をセットして下さい。

図 2.4.1 TEXT1 モードのパターンジェネレータテーブル設定例



2.4.1.3.3 パターンネームテーブル

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、指定され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、横40×縦24パターンです。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来るのは先頭アドレスの上位7ビット(A16~A10、TMS9918AではA13~A10)のみで、下位10ビット(A9~A0)は0と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから1KB単位の位置になります。

R#2

0	A16	A15	A14	A13	A12	A11	A10
---	-----	-----	-----	-----	-----	-----	-----

 Pattern name table base address register
TMS9918Aでは、A16~A14に0をセットして下さい。

図 2.4.2 TEXT1 モードのパターンネームテーブル

Base Address	+000	(0, 0)	0	0	1	2	・	・	38	39	X
	+001	(1, 0)	1	0	1	2	・	・	38	39	
	+002	(2, 0)	1	40	41	42	・	・	78	79	
	:	:	:	:	:	:	:	:	:	:	
	+039	(39, 0)	22	880	881	882	・	・	918	919	
	+040	(0, 1)	23	920	921	922	・	・	958	959	
	:	:	Y								
	+958	(38, 23)									
	+959	(39, 23)									

2.4.1.3.4 カラーレジスタ

画面の色はR#7で指定する2色が全てです。前景色はR#7の上位4ビット、背景色はR#7の下位4ビットで指定されたカラーコードで色が決まります。フォントパターン中の1の部分は前景色、0の部分は背景色で表示されます。また、バックドロップの色は背景色と同じになります。

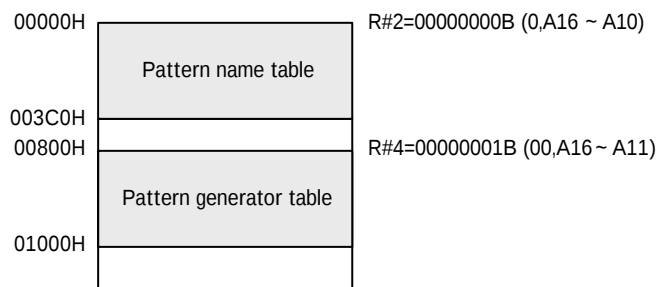
R#7

TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0
-----	-----	-----	-----	-----	-----	-----	-----

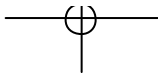
 Text color/Back drop color register
TC3~TC0 前景色(パターンの1の部分の色)を指定
カラーコード0を指定した場合、TP(R#8のbit5)の値にかかわらず必ず透明色になる
BD3~BD0 背景色(パターンの0の部分の色)とバックドロップの色を指定
カラーコード0を指定した場合、TP(R#8のbit5)の値にかかわらず必ずカラーパレットの色になる

2.4.1.4 VRAM マップ

図 2.4.3 TEXT1 モードのVRAM マップ



パレットテーブルは、0400Hに作成されます。



2.4.2 TEXT2 (SCREEN 0・80 字モード)

6 × 8 ドットのパターンを要素として、画面を 80 × 24 (26.5) に分割して表示する画面モードです。使用出来る色数は、画面全体で 2 色ですが、ブリンクを用いると 4 色です。

V9938, V9958 で使用出来ます。

2.4.2.1 特徴

1 パターンのサイズ	横 6 ドット × 縦 8 ドット
画面上のパターン数	横 80 パターン × 縦 24 パターン (26.5 パターン)
パターンの種類	256 種類
パターンの色	2 色 ブリンク使用で 4 色 (画面全体)
1 画面に必要な VRAM 容量	6KB

2.4.2.2 関係するレジスタと VRAM 領域

パターンのフォント	パターンジェネレータテーブル (VRAM)
パターンの位置	パターンネームテーブル (VRAM)
ブリンク属性	VRAM カラーテーブル
パターンの 1 の部分のカラーコード	R#7 上位 4bit
パターンの 0 の部分のカラーコード	R#7 下位 4bit
パターンの 1 の部分のカラーコード	R#12 上位 4bit (ブリンク用)
パターンの 0 の部分のカラーコード	R#12 下位 4bit (ブリンク用)
バックドロップのカラーコード	R#7 下位 4bit

2.4.2.3 レジスタと VRAM の設定

2.4.2.3.1 モードレジスタ

R#0	0	DG	IE2	IE1	0	1	0	0	Mode register 0
R#1	0	BL	IE0	1	0	0	SI	MAG	Mode register 1
R#8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938, V9958)
R#9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938, V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958 のみ)

M5 ~ M1 を 01001、YAE、YJK を 0 に設定します。TEXT2 モードでは、LN (R#9 の bit7) によって表示行数が変わります。LN=1 で 26.5 行、LN=0 で 24 行表示となります。その他のビットは任意です。

システムは、Mode register 0 ~ 3 をそれぞれ 00000100、01110000、00001000、00000000 に、R#25 を 00000000 に初期化します。

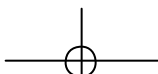
2.4.2.3.2 パターンジェネレータテーブル

TEXT1 モードと同じです。TEXT1 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。

2.4.2.3.3 パターンネームテーブル

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、指定され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 の時、横 80 × 縦 24 パターン、LN=1 の時、横 80 × 縦 26.5 パターンです。LN=1 での縦 27 個目のパターンは上部 4 ドットのみが表示されます。なお、縦 26.5 パターンのモードは BASIC ではサポートしていません。



パターンネームテーブルの先頭アドレスは、R#2 にセットします。指定出来るのは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 と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000H から 4KB 単位の位置になります。

R#2 0 A16 A15 A14 A13 A12 1 1 Pattern name table base address register

図 2.4.4 TEXT2 モードのパターンネームテーブル

Base Address

+0000	(0, 0)	0	0	1	2	...	78	79	X
+0001	(1, 0)	1	80	81	82	...	118	119	
+002	(2, 0)	:	:	:	:	:	:	:	
:	:	22	2000	2001	2002	...	2078	2079	
+0079	(79, 0)	23	2080	2081	2082	...	2158	2159	
+0080	(0, 1)	Y							
:	:								
+2158	(78, 23)								
+2159	(79, 23)								

2.4.2.3.4 カラーテーブル

TEXT2 モードでは、パターンネームテーブルの各バイトにそれぞれ 1 ビットのアトリビュートエリアが割り当てられており、アトリビュートを 1 にする事で各バイトにプリリンク属性を設定出来ます。このアトリビュートエリアをカラーテーブルと呼びます。

カラーテーブルの先頭アドレスは、R#3 と R#10 にセットします。指定出来るのは先頭アドレスの上位 8 ビット (A16 ~ A9) のみで、下位 9 ビット (A8 ~ A0) は 0 と見なされます。よって、カラーテーブルの先頭アドレスとして指定出来るのは、0000H から 512 バイト単位の位置になります。

Color table base address register

R# 3 A13 A12 A11 A10 A9 1 1 1
R#10 0 0 0 0 0 A16 A15 A14

図 2.4.5 TEXT2 モードのカラーテーブル

Base Address

	MSB	0	1	2	3	4	5	6	7	LSB
+000	0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0	7, 0		
+001	8, 0	9, 0	10, 0	11, 0	12, 0	13, 0	14, 0	15, 0		
:	:	:	:	:	:	:	:	:		
+009	72, 0	73, 0	74, 0	75, 0	76, 0	77, 0	78, 0	79, 0		
+010	0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1		
:	:	:	:	:	:	:	:	:		
+269	72, 26	73, 26	74, 26	75, 26	76, 26	77, 26	78, 26	79, 26		
Y										

2.4.2.3.5 カラーレジスタ

ブリンク属性を与えられたパターンは、R#7 と R#12 で指定されたカラーコードを交互に表示します。

R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
	TC3~TC0				BD3~BD0				
	前景色（パターンの1の部分の色）を指定				背景色（パターンの0の部分の色）とバックドロップの色を指定				
	カラーコード0を指定した場合、TP（R#8のbit5）の値にかかわらず必ず透明色になる				カラーコード0を指定した場合、TP（R#8のbit5）の値にかかわらず必ずカラーパレットの色になる				

R#12	T23	T22	T21	T20	BC3	BC2	BC1	BC0	Text color/Back color register
	T23~T20				BC3~BC0				
	ブリンク時のパターンの1の部分の色を指定				ブリンク時のパターンの0の部分の色を指定				
	カラーコード0を指定した場合、TP（R#8のbit5）の値にかかわらず必ず透明色になる（パターンの1の部分の色はパターンの0の部分の色になる）				カラーコード0を指定した場合、TP（R#8のbit5）の値にかかわらず必ずカラーパレットの色になる				

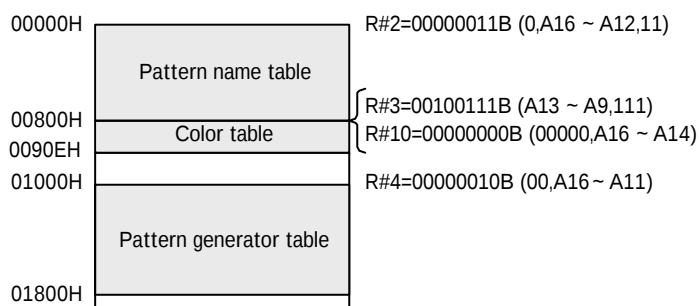
2.4.2.3.6 ブリンクレジスタ

ブリンク属性を与えられたパターンの、ブリンク間隔を R#13 で指定します。R#13 の上位4ビットには本来の色が表示される時間を、下位4ビットにはブリンク色が表示される時間をそれぞれ設定します。設定値と時間の関係は、2.3.1.3 カラーレジスタの表 2.3.3 を参照して下さい。

R#13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0	Blinking period register
	ON3~ON0				OF3~OF0				
	R#12 で指定された色の表示時間				R#7 で指定された色の表示時間				

2.4.2.4 VRAM マップ

図 2.4.6 TEXT2 モードの VRAM マップ



BASIC では 26.5 行表示、カラーテーブル、及びブリンク機能はサポートしていません。また、26.5 行表示を行う場合にはカラーテーブルを他の領域に移動させる必要があります。

パレットテーブルは、0F00H に作成されます。

2.4.3 GRAPHIC1 (SCREEN 1)

8 × 8 ドットのパターンを要素として、画面を 32 × 24 に分割して表示する画面モードです。使用出来る色数は、画面全体で 16 色です。また、スプライトの表示が出来ます。

TMS9918A, V9938, V9958 の全ての VDP で使用出来ます。

2.4.3.1 特徴

1パターンのサイズ	横 8 ドット × 縦 8 ドット
画面上のパターン数	横 32 パターン × 縦 24 パターン
パターンの種類	256 種類
パターンの色	8 パターンに 2 色 画面全体で 16 色
スプライト	モード 1
1 画面に必要な VRAM 容量	4KB

2.4.3.2 関係するレジスタと VRAM 領域

パターンのフォント	パターンジェネレータテーブル (VRAM)
パターンの位置	パターンネームテーブル (VRAM)
パターンの 1 の部分のカラーコード	VRAM カラーテーブル 8 パターン毎に 1 組
パターンの 0 の部分のカラーコード	
バックドロップのカラーコード	R#7 下位 4bit
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ジェネレータテーブル

2.4.3.3 レジスタと VRAM の設定

2.4.3.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	0	0	0	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938, V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938, V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958 のみ)

M5 ~ M1 を 00000、YAE・YJK を 0 に設定します。その他のビットは任意です。

システムは、Mode register 0 ~ 3 をそれぞれ 00000000、01100000、00001000、00000000 に、R#25 を 00000000 に初期化します。

2.4.3.3.2 パターンジェネレータテーブル

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する領域です。
 パターンには#0～#255の番号が付けられており、この番号を指定してパターンを画面に表示します。
 パターン1個のフォントは8バイトで構成されており、1個のパターンジェネレータテーブルは2KBの大きさを持ちます。
 パターンジェネレータテーブルの先頭アドレスは、R#4にセットします。指定出来るのは先頭アドレスの上位6ビット（A16～A11、TMS9918AではA13～A11）のみで、下位11ビット（A10～A0）は0と見なされます。よって、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出来るのは、0000Hから2KB単位の位置になります。

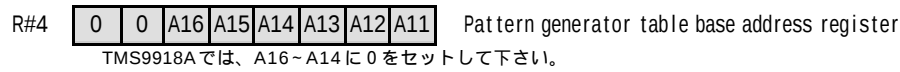
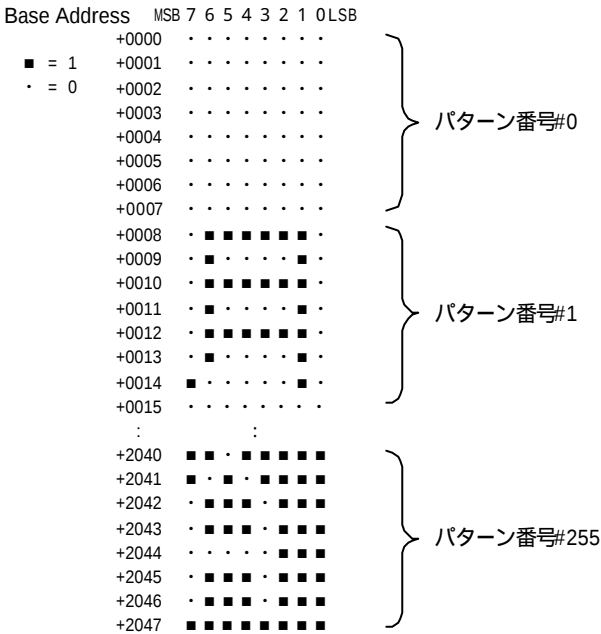


図 2.4.7 GRAPHIC1 モードのパターンジェネレータテーブル設定例



2.4.3.3.3 パターンネームテーブル

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、指定され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、横32×縦24パターンです。
 パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来るのは先頭アドレスの上位7ビット（A16～A10、TMS9918AではA13～A10）のみで、下位10ビット（A9～A0）は0と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから1KB単位の位置になります。



図 2.4.8 GRAPHIC1 モードのパターンネームテーブル

[illegible]

2.4.3.3.4 カラーテーブル

GRAPHIC1 モードでは、パターンの 1 の部分と 0 の部分の色を 8 パターン毎に 1 組設定します。

カラーテーブルの先頭アドレスは、R#3 と R#10 にセットします。指定出来るのは先頭アドレスの上位 11 ビット (A16~A6、TMS9918A では A13~A6) のみで、下位 6 ビット (A5~A0) は 0 と見なされます。よって、カラーテーブルの先頭アドレスとして指定出来るのは、0000H から 64 バイト単位の位置になります。

Color table base address register

R# 3	A13	A12	A11	A10	A9	A8	A7	A6
R#10	0	0	0	0	0	A16	A15	A14

図 2.4.9 GRAPHIC1 モードのカラーテーブル

+00	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	パターン番号 #0 ~ 7
+01	FC3	FC2	FC1	TC0	BC3	BC2	BC1	BC0	パターン番号 #8 ~ 16
:									
+31	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0	パターン番号 #248 ~ 255

FC3 ~ FC0 パターン 1 の部分のカラーコード
BC3 ~ BC0 パターン 0 の部分のカラーコード

2.4.3.3.5 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R#7

TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0
-----	-----	-----	-----	-----	-----	-----	-----

 Text color/Back drop color register

TC3~TC0 無効
BD3~BD0 バックドロップの色を指定

2.4.3.3.6 スプライト

スプライトモード 1 を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを 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 R#6 にセットします。詳細は、2.5.2 スプライトモード 1 を参照して下さい。

Sprite attribute table base address register

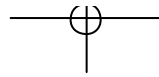
R# 5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

R# 6

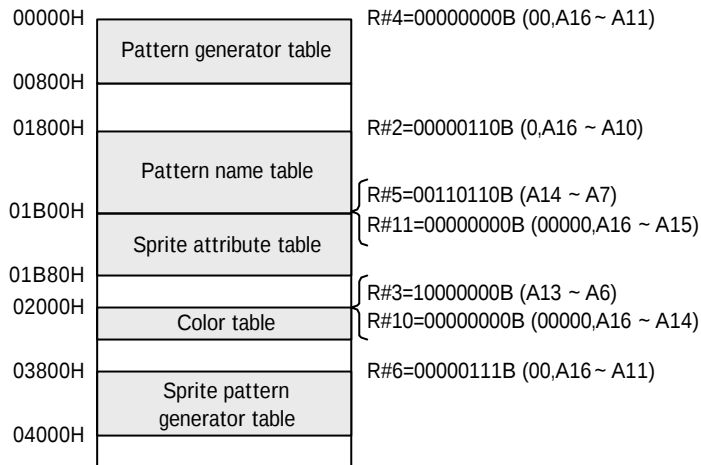
0	0	A16	A15	A14	A13	A12	A11
---	---	-----	-----	-----	-----	-----	-----

TMS9918Aでは、A16~A14に0をセットして下さい。



2.4.3.4 VRAM マップ

図 2.4.10 GRAPHIC1 モードの VRAM マップ



表示エリアが 256×192 の時は図 2.4.10 の VRAM マップの通りですが、V9938, V9958 でハードウェア縦スクロールによって表示エリアを 256×256 とする場合は、パターンネームテーブルの大きさは 768 バイトではなく 1024 バイトになります。その為、スプライトアトリビュートテーブルは別の領域に移動する必要があります。以上は、LN (R#9 の bit7) による 26.5 行表示に関しても同様です。

パレットテーブルは、02020H に作成されます。

2.4.4 GRAPHIC2 (SCREEN 2)

8×8 ドットのパターンを要素として、画面を 32×24 に分割して表示する画面モードです。使用出来る色数は、パターンの各ラインに 2 色、画面全体で 16 色です。また、スプライトの表示が出来ます。

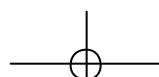
TMS9918A, V9938, V9958 の全ての VDP で使用出来ます。

2.4.4.1 特徴

1 パターンのサイズ	横 8 ドット×縦 8 ドット
画面上のパターン数	横 32 パターン×縦 24 パターン
パターンの種類	768 種類
パターンの色	パターンの各ラインに 2 色 画面全体で 16 色
スプライト	モード 1
1 画面に必要な VRAM 容量	16KB

2.4.4.2 関係するレジスタと VRAM 領域

パターンのフォント	パターンジェネレーターテーブル (VRAM)
パターンの位置	パターンネームテーブル (VRAM)
パターンの 1 の部分のカラーコード	VRAM カラーテーブルパターンの各ラインに 1 組
パターンの 0 の部分のカラーコード	
バックドロップのカラーコード	R#7 下位 4bit
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ジェネレーターテーブル



2.4.4.3 レジスタとVRAMの設定

2.4.4.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	0	0	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958のみ)

M5～M1を00100、YAE・YJKを0に設定します。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00000010、01100000、00001000、00000000に、R#25を00000000に初期化します。

2.4.4.3.2 パターンジェネレータテーブル

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する領域です。

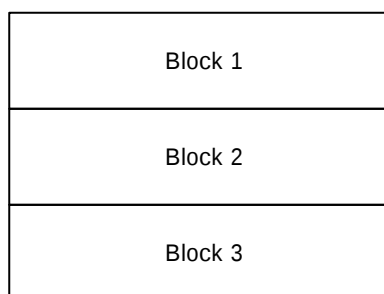
パターンには#0～#255の番号が付けられており、この番号を指定してパターンを画面に表示します。GRAPHIC2モードでは画面を3つのブロックに分割し、それぞれのブロックでパターンジェネレータテーブルを持つ事で、画面全体では768種類のパターンを使用出来ます。

パターン1個のフォントは8バイトで構成されており、パターンジェネレータテーブルは画面全体では6KBの大きさを持ちます。

パターンジェネレータテーブルの先頭アドレスは、R#4にセットします。指定出来るのは先頭アドレスの上位4ビット(A16～A13、TMS9918AではA13)のみで、下位13ビット(A12～A0)は0と見なされます。よって、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出来るのは、0000Hから8KB単位の位置になります。

R#4	0	0	A16	A15	A14	A13	1	1	Pattern generator table base address register
-----	---	---	-----	-----	-----	-----	---	---	---

TMS9918Aでは、A16～A14に0をセットして下さい。



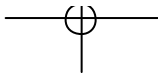
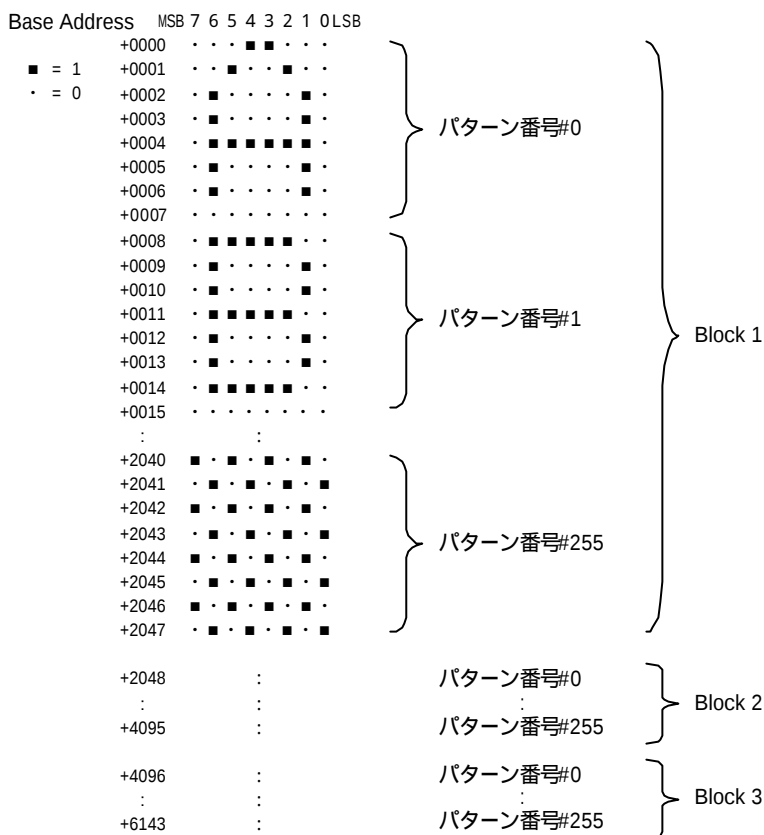


図 2.4.11 GRAPHIC2 モードのパターンジェネレータテーブル設定例



2.4.4.3.3 パターンネームテーブル

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、指定されたパターンが表示されます。GRAPHIC2モードでは画面を3つのブロックに分割し、それぞれのブロックでパターンネームテーブルを持つ事で、画面全体では768種類のパターンを表示出来ます。

1ブロックに表示されるパターンは横32×縦8パターンで、画面全体では横32×縦24パターンです。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来るのは先頭アドレスの上位7ビット（A16~A10、TMS9918AではA13~A10）のみで、下位10ビット（A9~A0）は0と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから1KB単位の位置になります。

R#2 0 A16 A15 A14 A13 A12 A11 A10 Pattern name table base address register
TMS9918Aでは、A16~A14に0をセットして下さい。

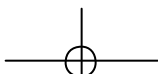


図 2.4.12 GRAPHIC2 モードのパターンネームテーブル

(0, 0)		(31, 0)
(0, 7)	Block 1 (256Bytes)	(31, 7)
(0, 8)		(31, 8)
(0,15)	Block 2 (256Bytes)	(31,15)
(0,16)		(31,16)
(0,23)	Block 3 (256Bytes)	(31,23)

2.4.4.3.4 カラーテーブル

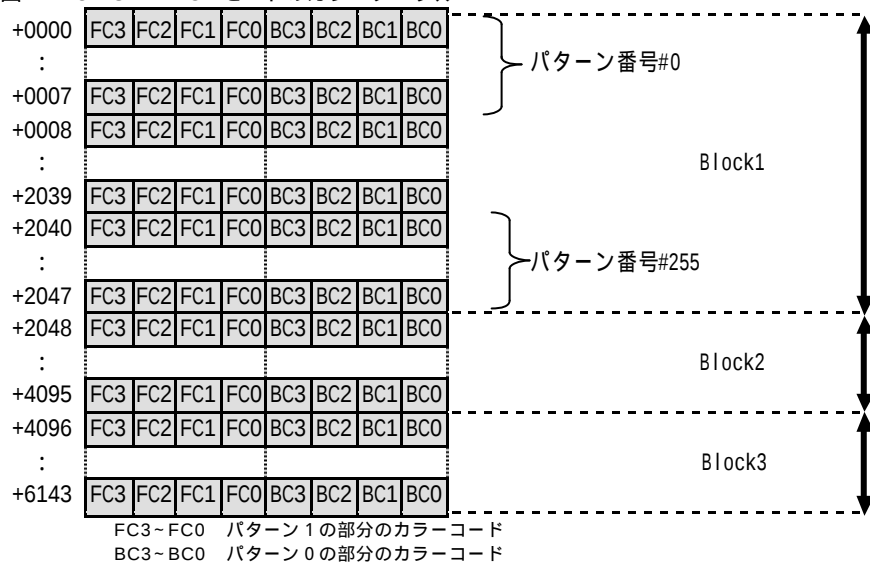
カラーテーブルはパターンジェネレータテーブルと1対1の対応をしており、パターンの1の部分と0の部分の色をパターンの各ラインに対して1組指定します。従って、1つのパターンに8バイトのカラーデータを持つ事になり、カラーテーブルは合計で6KBの大きさを持ちます。

カラーテーブルの先頭アドレスは、R#3 と R#10 にセットします。指定出来るのは先頭アドレスの上位4ビット (A16 ~ A13、TMS9918A ではA13) のみで、下位 13 ビット (A12 ~ A0) は0と見なされます。よって、カラーテーブルの先頭アドレスとして指定出来るのは、0000H から8KB 単位の位置になります。

Color table base address register

R# 3	A13	1	1	1	1	1	1	1
R#10	0	0	0	0	0	A16	A15	A14

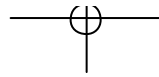
図 2.4.13 GRAPHIC2 モードのカラーテーブル



2.4.4.3.5 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
	TC3~TC0 無効				BD3~BD0 バックドロップの色を指定				



2.4.4.3.6 スプライト

スプライトモード1を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを 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 R#6 にセットします。詳細は、2.5.2 スプライトモード1を参照して下さい。

Sprite attribute table base address register

R# 5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

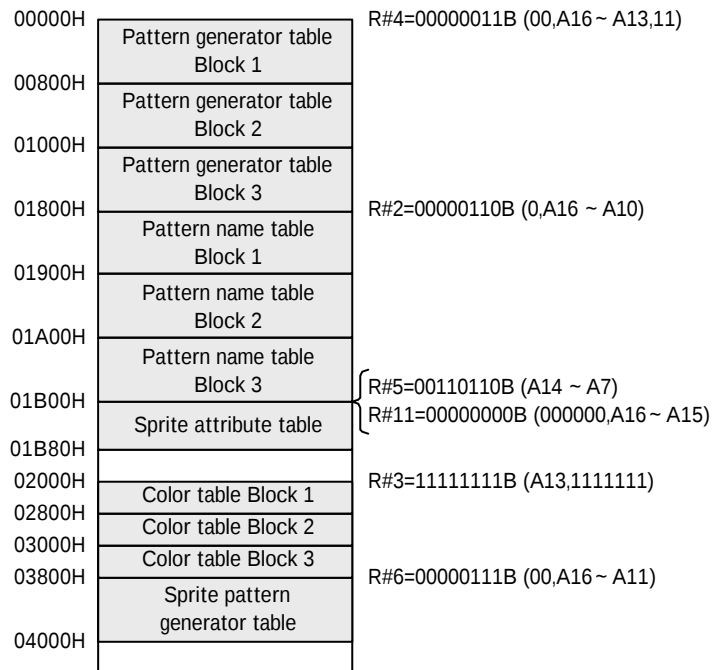
Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

TMS9918Aでは、A16～A14に0をセットして下さい。

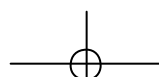
2.4.4.4 VRAM マップ

図 2.4.14 GRAPHIC2 モードのVRAMマップ



表示エリアが 256×192 の時は図 2.4.14 の VRAM マップの通りですが、V9938, V9958 でハードウェア縦スクロールによって表示エリアを 256×256 とする場合は、パターンネームテーブルの大きさは 768 バイトではなく連続した 1024 バイトになります。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 8KB となります。その為、パターンネームテーブル、スプライトパターンジェネレータテーブル、スプライトアトリビュートテーブルは別の領域に移動する必要があります。以上は、LN (R#9 の bit7) による 212 ライン表示に関しても同様です。

パレットテーブルは、01B80H に作成されます。



2.4.5 MULTI COLOR (SCREEN 3)

4 × 4 ドットのブロックを要素として、画面を 64 × 48 に分割して表示する画面モードです。使用出来る色数は、画面全体で 16 色です。また、スプライトの表示が出来ます。

TMS9918A, V9938, V9958 の全ての VDP で使用出来ます。

2.4.5.1 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 ブロック
ブロックの色	1 ブロックに 1 色 画面全体で 16 色
スプライト	モード 1
1 画面に必要な VRAM 容量	4KB

2.4.5.2 関係するレジスタと VRAM 領域

ブロックのカラーコード	パターンジェネレータテーブル (VRAM)
ブロックの位置	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
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ジェネレータテーブル

2.4.5.3 レジスタと VRAM の設定

2.4.5.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	0	0	0	0	Mode register 0
R# 1	0	BL	IE0	0	1	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938, V9958)
R# 9	LN	0	S1	S0	IL	E0	NT	DC	Mode register 3 (V9938, V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958 のみ)

M5 ~ M1 を 00010、YAE・YJK を 0 に設定します。その他のビットは任意です。

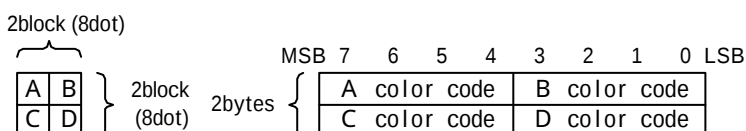
システムは、Mode register 0 ~ 3 をそれぞれ 00000000、01101000、00001000、00000000 に、R#25 を 00000000 に初期化します。

2.4.5.3.2 パターンジェネレータテーブル

パターンジェネレータテーブルは、ブロックの色を記憶する領域です。

パターン 1 個はブロック 4 個で構成されます。1 パターンに含まれるブロックの色は図 2.4.15 の様に指定します。

図 2.4.15 MULTI COLOR モードのパターンとブロック



パターンには#0～#255の番号が付けられており、この番号を指定してパターンを表示します。注意しなければならないのは、1個の番号に対して4つのパターンが割り当てられている事で、Y座標によって表示パターンが自動的に選択されます。パターン番号1つに8バイトが割り当てられ、1個のパターンジェネレータテーブルは2KBの大きさを持ちます。

パターンジェネレータテーブルの先頭アドレスは、R#4にセットします。指定出来るのは先頭アドレスの上位6ビット（A16～A11、TMS9918AではA13～A11）のみで、下位11ビット（A10～A0）は0と見なされます。よって、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出来るのは、0000Hから2KB単位の位置になります。

R#4

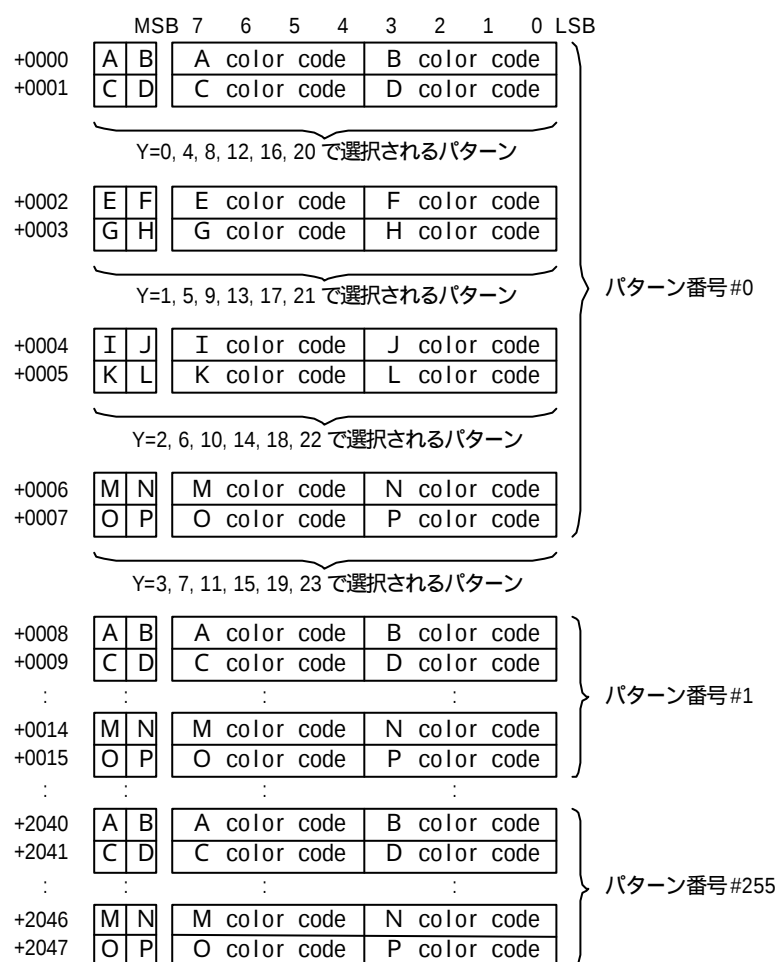
0	0	A16	A15	A14	A13	A12	A11
---	---	-----	-----	-----	-----	-----	-----

 Pattern generator table base address register

TMS9918Aでは、A16～A14に0をセットして下さい。

図 2.4.16 MULTI COLOR モードのパターンジェネレータテーブル

Base Address



2.4.5.3.3 パターンネームテーブル

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、指定されたパターンが表示されます。同じパターン番号を書き込んで、Y座標によって異なるパターンが表示され得る事に注意して下さい。

画面上に表示されるパターンは、横32×24パターンです。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来るのは先頭アドレスの上位7ビット(A16~A10、TMS9918AではA13~A10)のみで、下位10ビット(A9~A0)は0と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから1KB単位の位置になります。

R#2

0	A16	A15	A14	A13	A12	A11	A10
---	-----	-----	-----	-----	-----	-----	-----

 Pattern name table base address register
TMS9918Aでは、A16~A14に0をセットして下さい。

図 2.4.17 MULTI COLOR モードのパターンネームテーブル

Base Address	+000	(0, 0)	0	0	1	2	・	・	38	39	X
	+001	(1, 0)	1	32	33	34	・	・	62	63	
	+002	(2, 0)	:	:	:	:	:	:	:	:	
	:	:	22	704	705	706	・	・	734	735	
	+031	(31, 0)	23	736	737	738	・	・	766	767	
	+032	(0, 1)	Y								
	:	:									
	+766	(30, 23)									
	+767	(31, 23)									

2.4.5.3.4 カラーレジスタ

R#7でバックドロップの色を設定出来ます。

R# 7

TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0
-----	-----	-----	-----	-----	-----	-----	-----

 Text color/Back drop color register
TC3~TC0 無効
BD3~BD0 バックドロップの色を指定

2.4.5.3.5 スプライト

スプライトモード1を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを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R#6にセットします。詳細は、2.5.2 ブライトモード1を参照して下さい。

Sprite attribute table base address register

R# 5

A14	A13	A12	A11	A10	A9	A8	A7
-----	-----	-----	-----	-----	----	----	----

R#11

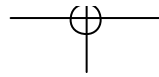
0	0	0	0	0	0	A16	A15
---	---	---	---	---	---	-----	-----

Sprite pattern generator table base address register

R# 6

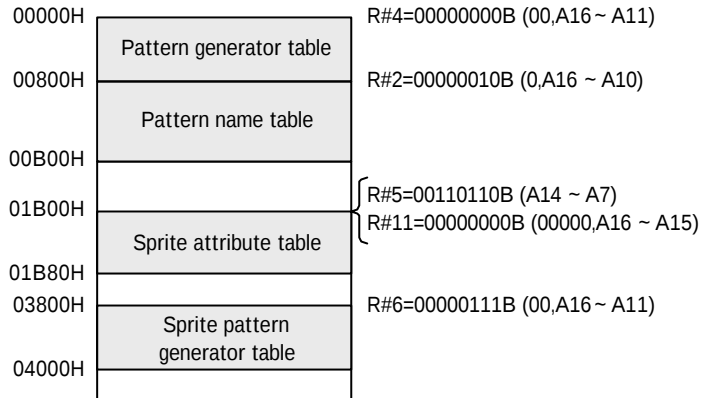
0	0	A16	A15	A14	A13	A12	A11
---	---	-----	-----	-----	-----	-----	-----

TMS9918Aでは、A16~A14に0をセットして下さい。



2.4.5.4 VRAM マップ

図 2.4.18 MULTI COLOR モードの VRAM マップ



表示エリアが 256 × 192 の時は図 2.4.18 の VRAM マップの通りですが、V9938, V9958 でハードウェア縦スクロール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の大きさは 768 バイトではなく 1024 バイトになります。この時の Y 座標による場合分けは、(0, 4, 8, 16, 20, 24, 28), (1, 5, 9, 13, 17, 21, 25, 29), (2, 6, 10, 14, 18, 22, 26, 30), (3, 7, 11, 15, 19, 23, 27, 31)の 4 つになります。以上は、LN (R#9 の bit7) による 212 ライン表示に関しても同様です。

パレットテーブルは、02020H に作成されます。

2.4.6 GRAPHIC3 (SCREEN 4)

8 × 8 ドットのパターンを要素として、画面を 32 × 24 に分割して表示する画面モードです。使用出来る色数は、パターンの各ラインに 2 色、画面全体で 16 色です。また、モード 2 のスプライトの表示が出来ます。

V9938・V9958 で使用出来ます。

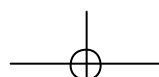
2.4.6.1 特徴

1 パターンのサイズ	横 8 ドット × 縦 8 ドット
画面上のパターン数	横 32 パターン × 縦 24 パターン
パターンの種類	768 種類
パターンの色	1 ラインに 2 色 画面全体で 16 色
スプライト	モード 2
1 画面に必要な VRAM 容量	16KB

スプライトの扱いがモード 2 である事以外は、GRAPHIC2 と同等です。

2.4.6.2 関係するレジスタと VRAM 領域

パターンのフォント	パターンジェネレータテーブル (VRAM)
パターンの位置	パターンネームテーブル (VRAM)
パターンの 1 の部分のカラーコード	VRAM カラーテーブルパターンの各ラインに 1 組
パターンの 0 の部分のカラーコード	
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAM スプライトアトリビュートテーブル
	VRAM スプライトカラーテーブル
	VRAM スプライトパターンジェネレータテーブル



2.4.6.3 レジスタとVRAMの設定

2.4.6.3.1 モードレジスタ

R#0	0	DG	IE2	IE1	0	1	0	0	Mode register 0
R#1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R#8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R#9	LN	0	S1	S0	IL	E0	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958のみ)

M5～M1を01000、YAE・YJKを0に設定します。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00000100、01100000、00001000、00000000に、R#25を00000000に初期化します。

2.4.6.3.2 パターンジェネレーターテーブル

GRAPHIC2モードと同じです。GRAPHIC2モードのパターンジェネレーターテーブルの項を参照して下さい。

2.4.6.3.3 パターンネームテーブル

GRAPHIC2モードと同じです。GRAPHIC2モードのパターンネームテーブルの項を参照して下さい。

2.4.6.3.4 カラーテーブル

GRAPHIC2モードと同じです。GRAPHIC2モードのカラーテーブルの項を参照して下さい。

2.4.6.3.5 カラーレジスタ

GRAPHIC2モードと同じです。GRAPHIC2モードのカラーレジスタの項を参照して下さい。

2.4.6.3.6 スプライト

スプライトモード2を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスをR#5とR#11に、スプライトパターンジェネレーターテーブルの先頭アドレスをR#6にセットします。詳細は、2.5.3 スプライトモード2を参照して下さい。

Sprite attribute table base address register

R# 5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15

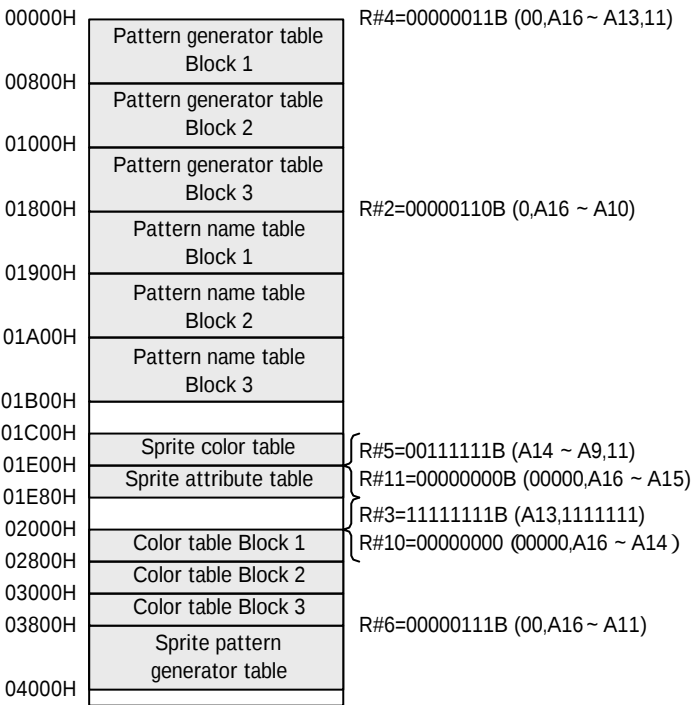
A9 必ず1をセット

Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

2.4.6.4 VRAM マップ

図 2.4.19 GRAPHIC3 モードの VRAM マップ



表示エリアが 256×192 の時は図 2.4.19 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 256×256 とする場合は、パターンネームテーブルの大きさは 768 バイトではなく連続した 1024 バイトになります。この場合はパターンジェネレーターテーブルとカラーテーブルもそれぞれ連続した 8KB となります。その為、パターンネームテーブル、スプライトパターンジェネレーターテーブル、スプライトアトリビュートテーブルは別の領域に移動する必要があります。以上は、LN (R#9 の bit7) による 212 ライン表示に関しても同様です。

パレットテーブルは、01B80H に作成されます。

2.4.7 GRAPHIC4 (SCREEN 5)

256×212(192) ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で 16 色です。また、モード 2 のスプライトの表示が出来ます。

V9938, V9958 で使用出来ます。

2.4.7.1 特徴

解像度	横 256 ドット×縦 212 ドット (192 ドット)
表示色	16 色 (画面全体)
スプライト	モード 2
1 画面に必要な VRAM 容量	32KB

2.4.7.2 関係するレジスタとVRAM 領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAMスプライトアトリビュートテーブル VRAMスプライトカラーテーブル VRAMスプライトパターンジェネレータテーブル

2.4.7.3 レジスタとVRAM の設定

2.4.7.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	0	1	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	E0	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958 のみ)

M5～M1 を 01100、YAE・YJK を 0 に設定します。GRAPHIC4 モードでは、LN (R#9 の bit7) によって縦方向の解像度が変わります。LN=1 で 212 ドット、LN=0 で 192 ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3 をそれぞれ 00000110、01100000、00001000、10000000 に、R#25 を 00000000 に初期化します。

2.4.7.3.2 パターンネームテーブル

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 ドット毎にパレットにより選択された 512 色中の 16 色から選んで表示出来ます。

パターンネームテーブルの先頭アドレスは、R#2 にセットします。指定出来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 と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000H から 32KB 単位の位置になります。

R# 2	0	A16	A15	1	1	1	1	1	Pattern name table base address register
------	---	-----	-----	---	---	---	---	---	--

図 2.4.20 GRAPHIC4 モードのパターンネームテーブル

Base Address						
+00000	(0, 0)	(1, 0)	(0, 0)	(1, 0)	(254, 0)	(255, 0)
+00001	(2, 0)	(3, 0)	(0, 1)			(255, 1)
:	:	:				
+00127	(254, 0)	(255, 0)				
+00128	(0, 1)	(1, 1)	(0,191)			(255,191)
:	:	:				
+27134	(252,211)	(253,211)				
+27135	(254,211)	(255,211)	(0,211)			(255,211)

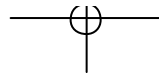
LN=0

LN=1

2.4.7.3.3 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R# 7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
	TC3～TC0 無効				BD3～BD0 バックドロップの色を指定				



2.4.7.3.4 スプライト

スプライトモード2を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを 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 R#6 にセットします。詳細は、2.5.3 スプライトモード2を参照して下さい。

Sprite attribute table base address register

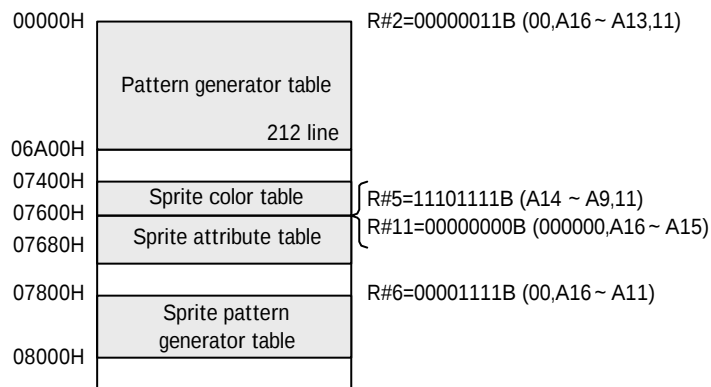
R#5	A14	A13	A12	A11	A10	1	1	1
R#11	0	0	0	0	0	0	A16	A15

Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

2.4.7.4 VRAM マップ

図 2.4.21 GRAPHIC4 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です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが 256 × 212 (192) の時は図 2.4.21 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 256 × 256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、07680H に作成されます。

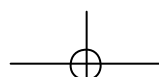
2.4.8 GRAPHIC5 (SCREEN 6)

512 × 212 (192) ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で4色です。また、モード2のスプライトの表示が出来ます。

V9938, V9958 で使用出来ます。

2.4.8.1 特徴

解像度	横 512 ドット × 縦 212 ドット (192 ドット)
表示色	4 色 (画面全体)
スプライト	モード 2
1 画面に必要なVRAM容量	32KB



2.4.8.2 関係するレジスタとVRAM 領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAMスプライトアトリビュートテーブル
	VRAMスプライトカラーテーブル
	VRAMスプライトパターンジェネレータテーブル

2.4.8.3 レジスタとVRAM の設定

2.4.8.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	1	0	0	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958 のみ)

M5～M1 を 10000、YAE・YJK を 0 に設定します。GRAPHIC5 モードでは、LN (R#9 の bit7) によって縦方向の解像度が変わります。LN=1 で 212 ドット、LN=0 で 192 ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3 をそれぞれ 00001000、01100000、00001000、10000000 に、R#25 を 00000000 に初期化します。

2.4.8.3.2 パターンネームテーブル

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 ドット毎にパレットにより選択された 512 色中の 4 色から選んで表示出来ます。

パターンネームテーブルの先頭アドレスは、R#2 にセットします。指定出来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 と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000H から 32KB 単位の位置になります。

R#2	0	A16	A15	1	1	1	1	1	Pattern name table base address register
-----	---	-----	-----	---	---	---	---	---	--

図 2.4.22 GRAPHIC5 モードのパターンネームテーブル

Base Address

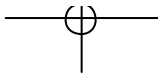
+00000	(0, 0)	(1, 0)	(2, 0)	(3, 0)
+00001	(4, 0)	(5, 0)	(6, 0)	(7, 0)
:	:	:	:	:
+00127	(508, 0)	(509, 0)	(510, 0)	(511, 0)
+00128	(0, 1)	(1, 1)	(2, 1)	(3, 1)
:	:	:	:	:
+27134	(504, 211)	(505, 211)	(506, 211)	(507, 211)
+27135	(508, 211)	(509, 211)	(510, 211)	(511, 211)

	(0, 0)	(1, 0)	(510, 0)	(511, 0)
	(0, 1)			(511, 1)
LN=0	(0,191)			(511,191)
LN=1	(0,211)			(511,211)

2.4.8.3.3 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R# 7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
	TC3～TC0 無効								
					BD3～BD2 バックドロップの偶数ドットの色を指定				
					BD1～BD0 バックドロップの奇数ドットの色を指定				



2.4.8.3.4 スプライト

スプライトモード2を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを R#5 と R#11 に、スプライトパターンジェネレーターテーブルの先頭アドレスを R#6 にセットします。詳細は、2.5.3 スプライトモード2を参照して下さい。

Sprite attribute table base address register

R#5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15

A9 必ず1をセット

Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

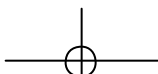
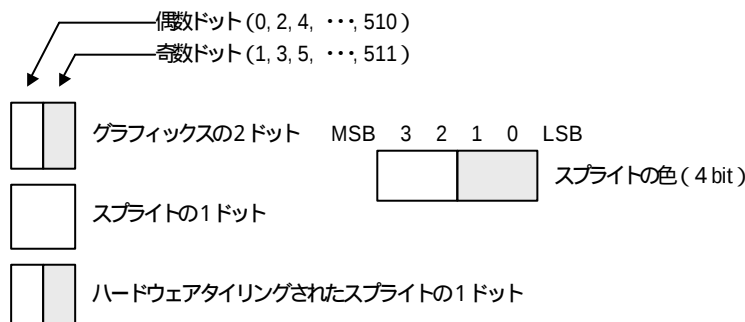
2.4.8.3.5 ハードウェアタイリング

GRAPHIC5 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が行われます。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ビットはX座標 (0~511) の偶数ドットのカラーコードとして、下位2ビットは奇数ドットのカラーコードとして扱われます。

GRAPHIC5 モードでは、スプライトの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ドットの左半分と右半分にそれぞれ異なる色を表示出来ます。

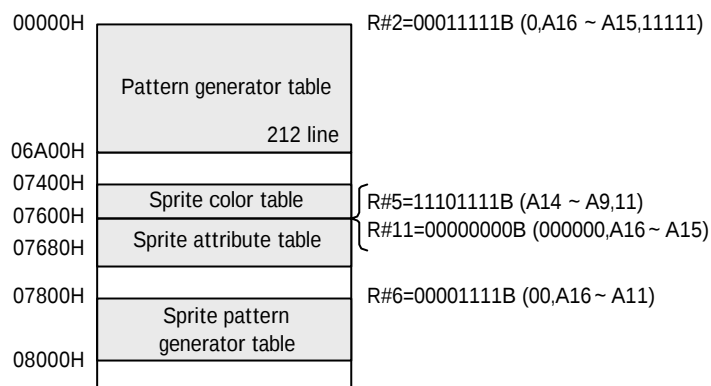
バックドロップについても同様に、偶数ドットと奇数ドットにそれぞれ異なる色を表示出来ます。

図 2.4.23 ハードウェアタイリング



2.4.8.4 VRAM マップ

図 2.4.24 GRAPHIC5 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です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが 512×212 (192) の時は図 2.4.24 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 512×256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、07680H に作成されます。

2.4.9 GRAPHIC6 (SCREEN 7)

512×212(192)ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で16色です。また、モード2のスプライトの表示が出来ます。

VRAM を128KB実装したV9938, V9958で使用出来ます。

2.4.9.1 特徴

解像度	横 512 ドット×縦 212 ドット (192 ドット)
表示色	16 色 (画面全体)
スプライト	モード 2
1 画面に必要なVRAM容量	64KB

GRAPHIC6 モードを使用するには、VRAM が128KB 必要です。

2.4.9.2 関係するレジスタとVRAM 領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAMスプライトアトリビュートテーブル VRAMスプライトカラーテーブル VRAMスプライトパターンジェネレーターテーブル

2.4.9.3 レジスタとVRAMの設定

2.4.9.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	1	0	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958のみ)

M5～M1を10100、YAE・YJKを0に設定します。GRAPHIC6モードでは、LN（R#9のbit7）によって縦方向の解像度が変わります。LN=1で212ドット、LN=0で192ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00001010、01100000、00001000、10000000に、R#25を00000000に初期化します。

2.4.9.3.2 パターンネームテーブル

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ドット毎にパレットにより選択された512色中の16色から選んで表示出来ます。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来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と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから64KB単位の位置になります。

GRAPHIC6モード以降の画面モードでは、R#2のA16の位置がGRAPHIC5以下と異なります。

R# 2	0	0	A16	1	1	1	1	1	Pattern name table base address register
------	---	---	-----	---	---	---	---	---	--

GRAPHIC6以降ではA16の位置が異なる

図 2.4.25 GRAPHIC6 モードのパターンネームテーブル

Base Address

+00000	(0, 0)	(1, 0)
+00001	(2, 0)	(3, 0)
:	:	:
+00255	(510, 0)	(511, 0)
+00256	(0, 1)	(1, 1)
:	:	:
+54270	(508,211)	(509,211)
+54271	(510,211)	(511,211)

	(0, 0)	(1, 0)	(510, 0)	(511, 0)
	(0, 1)			(511, 1)
LN=0	(0,191)			(511,191)
LN=1	(0,211)			(511,211)

2.4.9.3.3 カラーレジスタ

R#7でバックドロップの色を設定出来ます。

R# 7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
------	-----	-----	-----	-----	-----	-----	-----	-----	-------------------------------------

TC3～TC0 無効
BD3～BD0 バックドロップの色を指定

2.4.9.3.4 スプライト

スプライトモード2を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを 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 R#6 にセットします。詳細は、2.5.3 スプライトモード2を参照して下さい。

Sprite attribute table base address register

R# 5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15

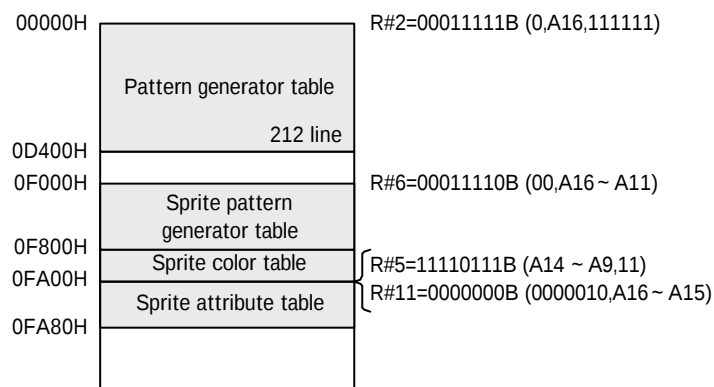
A9 必ず1をセット

Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

2.4.9.4 VRAM マップ

図 2.4.26 GRAPHIC6 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です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが 512×212 (192) の時は図 2.4.26 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 512×256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、0FA80H に作成されます。

2.4.10 GRAPHIC7 (SCREEN 8)

256×212(192) ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で 256 色です。また、モード2のスプライトの表示が出来ます。

VRAM を 128KB 実装した V9938, V9958 で使用出来ます。

2.4.10.1 特徴

解像度	横 256 ドット×縦 212 ドット (192 ドット)
表示色	256 色 (画面全体)
スプライト	モード2
1 画面に必要なVRAM容量	64KB

GRAPHIC7モードを使用するには、VRAM が128KB 必要です。

2.4.10.2 関係するレジスタとVRAM領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7
スプライト	VRAMスプライトアトリビュートテーブル VRAMスプライトカラーテーブル VRAMスプライトパターンジェネレータテーブル

2.4.10.3 レジスタとVRAMの設定

2.4.10.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	1	1	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	0	WTE	MSK	SP2	(V9958のみ)

M5～M1を11100、YAE・YJKを0に設定します。GRAPHIC7モードでは、LN (R#9のbit7)によって縦方向の解像度が変わります。LN=1で212ドット、LN=0で192ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00001110、01100000、00001000、10000000に、R#25を00000000に初期化します。

2.4.10.3.2 パターンネームテーブル

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ドット毎に256色から選んで表示出来ます。

パターンネームテーブル上の1バイトと表示色の関係は図2.4.27の通りです。

図 2.4.27 GRAPHIC7モードにおけるパターンネームテーブル上のデータと表示色

G2	G1	G0	R2	R1	R0	B1	B0
G2～G0			緑の輝度 (0～7)			赤の輝度 (0～7)	
R2～R0			赤の輝度 (0～7)			青の輝度 (0～3) (青の輝度 (0～3)はパレットで表現すると (0, 7/3, 14/3, 7)に相当する)	
B1～B0			青の輝度 (0～3)				

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来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と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから64KB単位の位置になります。

GRAPHIC6モード以降の画面モードでは、R#2のA16の位置がGRAPHIC5以下と異なります。

R# 2	0	0	A16	1	1	1	1	1	Pattern name table base address register
GRAPHIC6以降ではA16の位置が異なる									

図 2.4.28 GRAPHIC7モードのパターンネームテーブル

Base Address	+00000	(0, 0)	(0, 0)	(1, 0)	(254, 0)	(255, 0)
	+00001	(1, 0)	(0, 1)			(255, 1)
	+00002	(2, 0)				
	:	:				
	+00255	(255, 0)	(0,191)			(255,191)
	+00256	(0, 1)				
	:	:				
	+54270	(254,211)	(0,211)			(255,211)
	+54271	(255,211)				

2.4.10.3.3 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R# 7

BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
-----	-----	-----	-----	-----	-----	-----	-----

 Text color/Back drop color register
BD7~BD0 バックドロップの色を指定

2.4.10.3.4 スプライト

スプライトモード2を使用出来ますが、スプライトのカラーコードの扱いが他のモードと異なるので注意して下さい。

スプライトアトリビュートテーブルの先頭アドレスを 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 R#6 にセットします。詳細は、2.5.3 スプライトモード2を参照して下さい。

Sprite attribute table base address register

R# 5

A14	A13	A12	A11	A10	A9	1	1
-----	-----	-----	-----	-----	----	---	---

R#11

0	0	0	0	0	0	A16	A15
---	---	---	---	---	---	-----	-----

A9 必ず1をセット

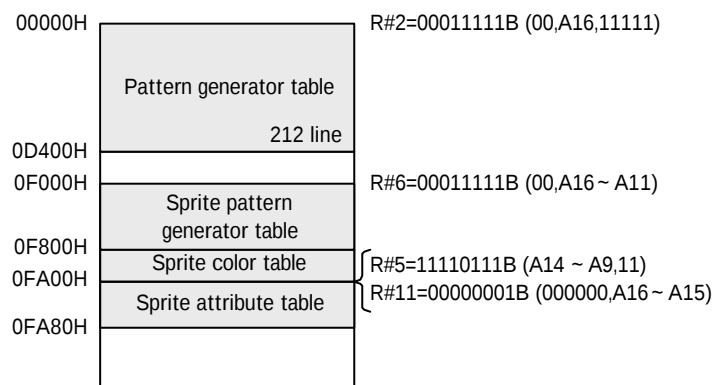
Sprite pattern generator table base address register

R# 6

0	0	A16	A15	A14	A13	A12	A11
---	---	-----	-----	-----	-----	-----	-----

2.4.10.4 VRAM マップ

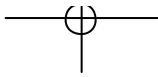
図 2.4.29 GRAPHIC7 モードのVRAMマップ



以下同様に、2 ページまで割り付け可能です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが 256 × 212 (192) の時は図 2.4.29 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 256 × 256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、0FA80H に作成されます。



2.4.11 YJK/RGB 混在 (SCREEN 10・11)

256×212 (192) ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で 12499 色です。
また、モード 2 のスプライトの表示が出来ます。

VRAM を 128KB 実装した V9958 でのみ使用出来ます。

2.4.11.1 YJK 方式

色の情報を扱う方法として、TMS9918A, V9938 では色を赤, 青, 緑に分解してその比率を表現する方式 (RGB 方式) が採用されています。これに対し、YJK 方式は色を輝度成分 Y と色相成分 J, K に分解して VRAM 上に置き、表示する時に RGB 方式に変換するものです。

V9958 では VRAM を増やさずに多色表示を実現する為、輝度情報は 1 ドット毎に持ちますが、色相情報を 4 ドットでまとめて管理しています。

YJK 方式は自然物の表現に向いており、人工的な物体の表現能力はそれほど高くはありません。

以下に、YJK と RGB の相互変換の式を示します。

1 . YJK から RGB への変換式

$$\begin{aligned} R &= Y + J \\ G &= Y + K \\ B &= Y \times 5 / 4 - J / 2 - K / 4 \end{aligned}$$

2 . RGB から YJK への変換式

$$\begin{aligned} Y &= B / 2 + R / 4 + G / 8 \\ J &= R - Y \\ K &= G - Y \end{aligned}$$

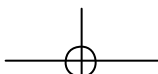
2.4.11.2 特徴

解像度	横 256 ドット×縦 212 ドット (192 ドット)
表示色	12499 色 (画面全体)
スプライト	モード 2
1 画面に必要な VRAM 容量	64KB

YJK/RGB 混在モードを使用するには、VRAM が 128KB 必要です。

2.4.11.3 関係するレジスタと VRAM 領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAM スプライトアトリビュートテーブル
	VRAM スプライトカラーテーブル
	VRAM スプライトパターンジェネレーターテーブル



2.4.11.4 レジスタとVRAMの設定

2.4.11.4.1 モードレジスタ

R# 0	0	DG	IE2	IE1	1	1	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	1	1	WTE	MSK	SP2	(V9958のみ)

M5～M1を11100、YAE・YJKを共に1に設定します。YJK/RGB混在モードでは、LN(R#9のbit7)によって縦方向の解像度が変わります。LN=1で212ドット、LN=0で192ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00001110、01100000、00001000、10000000に、R#25を00011000に初期化します。

2.4.11.4.2 パターンネームテーブル

パターンネームテーブルは4バイトを単位として、その4バイトが画面上の水平方向に連続した4ドットに対応しており、1ドット毎にパレットにより選択された512色中の16色、もしくは12499色から選んで表示出来ます。

パターンネームテーブル上の4バイトと表示色の関係は図2.4.30の通りです。

図 2.4.30 YJK/RGB 混在モードにおけるパターンネームテーブル上のデータと表示色

4N+0	Y1	A1	KL
4N+1	Y2	A2	KH
4N+2	Y3	A3	JL
4N+3	Y4	A4	JH

An=0の時 Yn, KL, KH, JL, JHでYJK方式の1ドットの色情報として扱われる

An=1の時 Ynはカラーコードとして扱われる

カラーコード0を指定した場合、TP(R#8のbit5)の値にかかわらず必ずカラーパレットの色になる

KLとKHはそれぞれ色相情報Kの下位3ビットと上位3ビット、JLとJHはそれぞれ色相情報Jの下位3ビットと上位3ビットです。色相情報KとJは4ドット共通で使用されます。

Ynは、アトリビュートAnの値によって扱いが変わります。An=0すなわちアトリビュートがリセットされている時、Ynは各ドットの輝度情報Y(4bit)として扱われ、データはYJK→RGB変換テーブルを通して出力されます。An=1の時は、Ynはカラーコードとして扱われカラーパレットを通してRGB出力されます。

YJKとRGBの変換については、2.4.11.1 YJK方式を参照して下さい。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来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と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから64KB単位の位置になります。

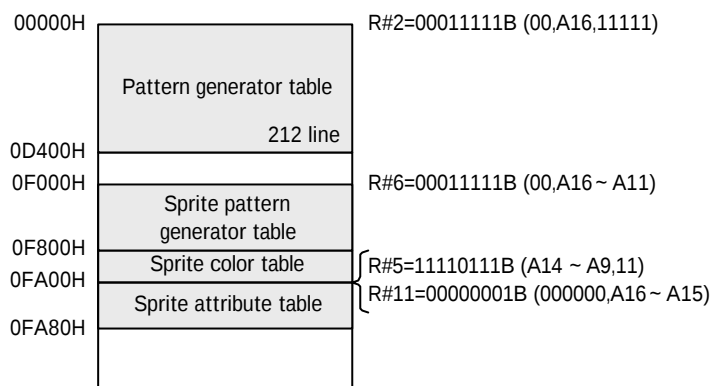
GRAPHIC6モード以降の画面モードでは、R#2のA16の位置がGRAPHIC5以下と異なります。

R# 2	0	0	A16	1	1	1	1	1	Pattern name table base address register
------	---	---	-----	---	---	---	---	---	--

GRAPHIC6以降ではA16の位置が異なる

2.4.11.6 VRAM マップ

図 2.4.32 YJK/RGB 混在モードのVRAM マップ



以下同様に、2 ページまで割り付け可能です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが $256 \times 212(192)$ の時は図 2.4.32 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 256×256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、0FA80H に作成されます。

2.4.12 YJK (SCREEN 12)

$256 \times 212 (192)$ ドットの解像度を持つ画面モードです。使用出来る色数は、画面全体で 19268 色です。また、モード 2 のスプライトの表示が出来ます。

VRAM を 128KB 実装した V9958 でのみ使用出来ます。

2.4.12.1 特徴

解像度	横 256 ドット×縦 212 ドット (192 ドット)
表示色	19268 色 (画面全体)
スプライト	モード 2
1 画面に必要な VRAM 容量	64KB

YJK モードを使用するには、VRAM が 128KB 必要です。

2.4.12.2 関係するレジスタと VRAM 領域

グラフィックス	パターンネームテーブル (VRAM)
バックドロップのカラーコード	R#7 下位 4bit
スプライト	VRAM スプライトアトリビュートテーブル
	VRAM スプライトカラーテーブル
	VRAM スプライトパターンジェネレータテーブル

2.4.12.3 レジスタとVRAMの設定

2.4.12.3.1 モードレジスタ

R# 0	0	DG	IE2	IE1	1	1	1	0	Mode register 0
R# 1	0	BL	IE0	0	0	0	SI	MAG	Mode register 1
R# 8	MS	LP	TP	CB	VR	0	SPD	BW	Mode register 2 (V9938,V9958)
R# 9	LN	0	S1	S0	IL	EO	NT	DC	Mode register 3 (V9938,V9958)
R#25	0	CMD	VDS	0	1	WTE	MSK	SP2	(V9958のみ)

M5～M1を11100、YAEを0、YJKを1に設定します。YJKモードでは、LN(R#9のbit7)によって縦方向の解像度が変わります。LN=1で212ドット、LN=0で192ドットとなります。その他のビットは任意です。

システムは、Mode register 0～3をそれぞれ00001110、01100000、00001000、10000000に、R#25を00001000に初期化します。

2.4.12.3.2 パターンネームテーブル

パターンネームテーブルは4バイトを単位として、その4バイトが画面上の水平方向に連続した4ドットに対応しており、1ドット毎に19268色から選んで表示出来ます。

パターンネームテーブル上の4バイトと表示色の関係は図2.4.33の通りです。

図 2.4.33 YJKモードにおけるパターンネームテーブル上のデータと表示色

4N+0	Y1	KL
4N+1	Y2	KH
4N+2	Y3	JL
4N+3	Y4	JH

Yn, KL, KH, JL, JH でYJK方式の1ドットの色情報

KLとKHはそれぞれ色相情報Kの下位3ビットと上位3ビット、JLとJHはそれぞれ色相情報Jの下位3ビットと上位3ビットです。色相情報KとJは4ドット共通で使用されます。

Ynは各ドットの輝度情報Y(5bit)として扱われ、データはYJK→RGB変換テーブルを通して出力されます。

YJKとRGBの変換については、2.4.11.1 YJK方式を参照して下さい。

パターンネームテーブルの先頭アドレスは、R#2にセットします。指定出来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と見なされます。よって、パターンネームテーブルの先頭アドレスとして指定出来るのは、0000Hから64KB単位の位置になります。

GRAPHIC6モード以降の画面モードでは、R#2のA16の位置がGRAPHIC5以下と異なります。

R#2	0	0	A16	1	1	1	1	1	Pattern name table base address register
-----	---	---	-----	---	---	---	---	---	--

GRAPHIC6以降ではA16の位置が異なる

図 2.4.34 YJK モードのパターンネームテーブル

Base Address	7	6	5	4	3	2	1	0
+00000	Y1	(0, 0)						KL
+00001	Y2	(1, 0)						KH
+00002	Y3	(2, 0)						JL
+00003	Y4	(3, 0)						JH
+00004	Y1	(4, 0)						KL
:	:	:						:
+00255	Y4	(255, 0)						JH
+00256	Y1	(0, 1)						KL
:	:	:						:
+54270	Y3	(254, 211)						JL
+54271	Y4	(255, 211)						JH

	(0, 0)	(1, 0)		(254, 0)	(255, 0)
	(0, 1)				(255, 1)
LN=0	(0, 191)				(255, 191)
LN=1	(0, 211)				(255, 211)

2.4.12.3.3 カラーレジスタ

R#7 でバックドロップの色を設定出来ます。

R#7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0	Text color/Back drop color register
	TC3~TC0 無効				BD3~BD0 バックドロップの色を指定 (パレットが有効)				
	カラーコード 0 を指定した場合、TP (R#8 の bit5) の値にかかわらず必ずカラーパレットの色になる								

2.4.12.3.4 スプライト

スプライトモード 2 を使用出来ます。

スプライトアトリビュートテーブルの先頭アドレスを R#5 と R#11 に、スプライトパターンジェネレーターテーブルの先頭アドレスを R#6 にセットします。詳細は、2.5.3 スプライトモード 2 を参照して下さい。

スプライトの色はカラーコードで指定します (パレットが有効です)。

Sprite attribute table base address register

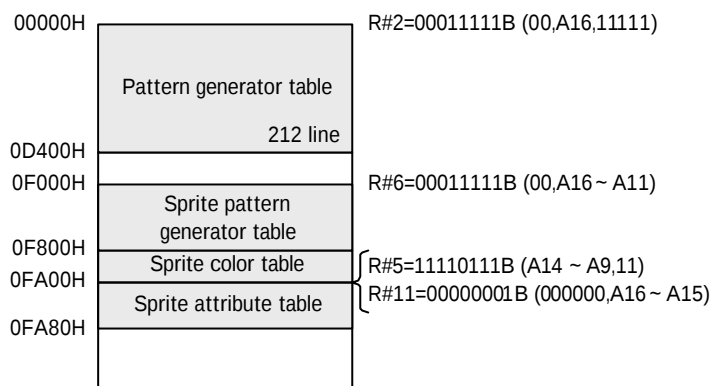
R# 5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15
	A9 必ず 1 をセット							

Sprite pattern generator table base address register

R# 6	0	0	A16	A15	A14	A13	A12	A11
------	---	---	-----	-----	-----	-----	-----	-----

2.4.12.4 VRAM マップ

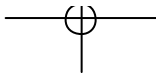
図 2.4.35 YJK モードのVRAMマップ



以下同様に、2 ページまで割り付け可能です。BASIC からは、SET PAGE 命令で切り替える事が出来ます。

表示エリアが 256×212 (192) の時は図 2.4.35 の VRAM マップの通りですが、ハードウェア縦スクロールによって表示エリアを 256×256 とする場合は、ページ全域がビットマップ表示に使用されます。この為、スプライト関連のテーブルは別のページに確保しなければなりません。

パレットテーブルは、0FA80H に作成されます。



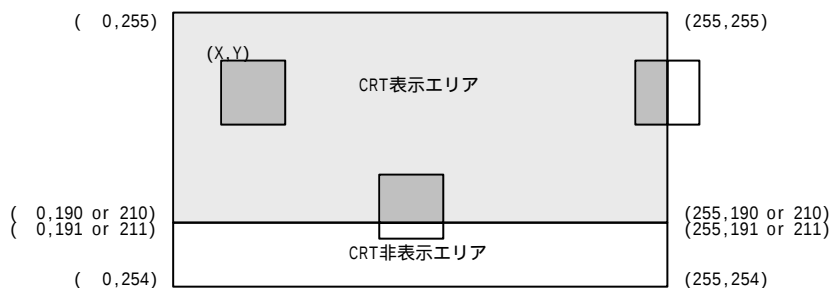
5章 スプライト

この章では、TMS9918A, V9938, V9958 のスプライトについて説明します。

2.5.1 スプライトに関する基礎知識

TMS9918A, V9938, V9958 では1画面に32個のスプライトを表示する事が出来ます。スプライトのサイズは8×8ドット又は16×16ドットです。横方向の1ドットの大きさは画面モードに関わらず一定で画面全体を256に分割した大きさです。縦方向に関しても1ドットの大きさは一定ですが、画面モードによって表示可能な範囲が変化します。このドットを単位として、画面上の任意の位置に表示する事が出来ます。

図 2.4.36 スプライト表示範囲



スプライトは独立した表示面を持っている為、原則的に他の表示面に影響を及ぼしません(詳しくは、2.2.7 画面構成を参照)。

V9938, V9958 には2種類のスプライトモードが存在し、画面モードによって自動的に選択されます。

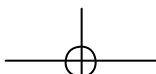
- ・スプライトモード1 GRAPHIC1～2, MULTI COLOR (SCREEN 1～3)
- ・スプライトモード2 GRAPHIC3～7, YJK/RGB 混在, YJK (SCREEN 4～8, 10～12)

2.5.2 スプライトモード1

TMS9918A 及び V9938, V9958 の TMS9918A 互換モードで使用されます。画面モードに GRAPHIC1～2, MULTI COLOR (SCREEN 1～3) を選択すると、自動的にスプライトモード1が使用されます。

2.5.2.1 特徴

1画面に表示出来るスプライトは32個で、それぞれに#0～#31までの番号が付けられています。番号の小さい方が優先度が高く、重なった場合は優先度の高いものが上に表示されます。また、1水平線の上に5個以上のスプライトが並んだ場合は優先度の高い順に4個までが表示され、5個目以降は表示されません。



The diagram illustrates six rectangular blocks arranged horizontally, labeled #0 through #5. Blocks #0, #1, #2, and #3 are solid gray. Block #4 is a dashed outline with a solid gray top section. Block #5 is a dashed outline with a solid gray bottom section.

また、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）時には、S#0 のビット 5 が 1 になり、これによって衝突の発生を知る事が出来ます。

スプライトを表示する時には、次の設定を行って下さい。

スプライトを表示する時には、次の設定を行って下さい。

R#1 で、スプライトのサイズ・拡大を、R#8 でスプライトの表示状況・カラーコード 0 の扱いを選択します。

R# 1

	-	0	SI	MAG
--	---	---	----	-----

Mode register 1

SI スプライトのサイズ 1=16×16ドット、0=8×8ドット
MAG スプライトを 1=横2倍×縦2倍に拡大する、0=拡大しない

R# 8	-	TP	-	0	SPD	-	Mode register 2 (V9938,V9958)
	TP	カラーコード0の色を 1=カラーパレットの色にする。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する。 0=透明色として扱う。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ない。					
	SPD	スプライトを 1=表示しない、0=表示する					

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)。

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・パターン番号・色等をセットします(#0~#31)。

スプライトパターンジェネレータテーブルは、スプライトのパターンを記憶する領域です。
パターン1個のフォントは8バイトで構成されており、スプライトパターンジェネレータテーブルは2KBの大きさを持ちますが、必ずしも全てのパターンを設定する必要はありません。

パターンには#0～#255の番号が付けられており、スプライトのサイズが8×8ドットの時はスプライト1個にパターン1個が対応します。16×16ドットの時はスプライト1個に4個のパターンが対応し、パターン番号4N, 4N+1, 4N+2, 4N+3が左上、左下、右上、右下の順に並べられます。

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は、R#6 にセットします。指定出来るのは先頭

アドレスの上位 6 ビット（A16～A11、TMS9918A では A13～A11）のみで、下位 11 ビット（A10～A0）は 0 と見なされます。よって、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として指定出来るのは、0000H から 2KB 単位の位置になります。

Sprite pattern generator table base address register

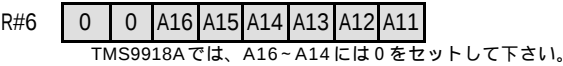
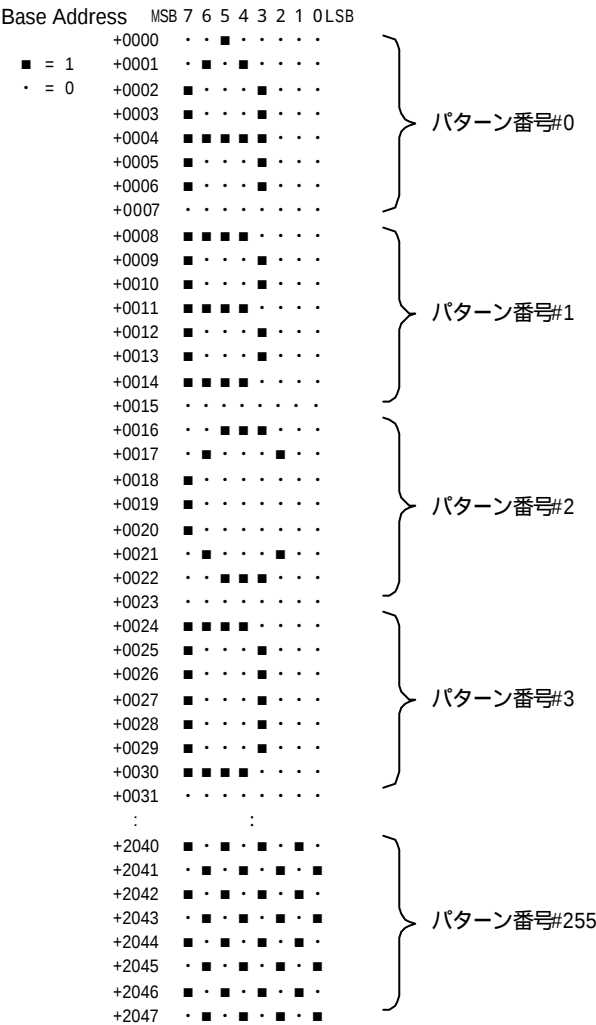
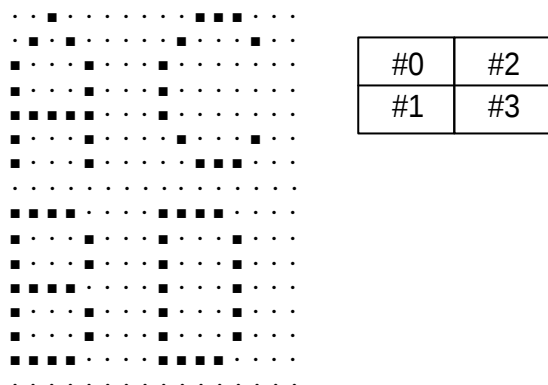


図 2.5.1 スプライトパターンジェネレータテーブル設定例



16×16 ドットの場合、図 2.5.1 のスプライトパターンジェネレータテーブルを例に取れば、次のようになります。

図 2.5.2 スプライトパターンジェネレータテーブルデータ設定例



2.5.2.4 スプライトアトリビュートテーブル

スプライトアトリビュートテーブルは、32 個のスプライトの各々の表示位置(X,Y)、1 の部分の色 (0 の部分は透明)、表示するパターン番号等を指定する為に VRAM 上に確保される領域です。

1 個のスプライトに 4 バイトのデータ領域が確保されており、テーブル全体で 128 バイトの大きさを持ちます。

スプライトアトリビュートテーブルの先頭アドレスは、R#5 と R#11 にセットします。指定出来るのは先頭アドレスの上位 10 ビット (A16~A7、TMS9918A では A13~A7) のみで、下位 7 ビット (A6~A0) は 0 と見なされます。よって、スプライトアトリビュートテーブルの先頭アドレスとして指定出来るのは、0000H から 128 バイト単位の位置になります。

Sprite attribute table base address register

R# 5	A14	A13	A12	A11	A10	A9	A8	A7
R#11	0	0	0	0	0	0	A16	A15

TMS9918A では、A16~A14 には 0 をセットして下さい。

図 2.5.3 スプライトモード 1 のスプライトアトリビュートテーブル

+000	Y 座標 (0~255)			
+001	X 座標 (0~255)			
+002	パターン番号 (0~255)			
+003	EC	0	0	0
:	:	:	:	:
+124	Y 座標 (0~255)			
+125	X 座標 (0~255)			
+126	パターン番号 (0~255)			
+127	EC	0	0	0

スプライト番号#0
アトリビュートエリア

スプライト番号#31
アトリビュートエリア

Y 座標 (0~255) スプライトの Y 座標を設定します。画面最上端の Y 座標が 255 である事に注意して下さい。

スプライトの Y 座標の値を 208 に設定すると、そのスプライトとそのスプライトより優先度の低い (パターン番号の大きい) スプライトは表示されません。例えば、スプライト番号#16 のスプライトの Y 座標を 208 にすると、#16 ~ #31 までのスプライトは表示されません。特定のスプライトのみを画面から消去するには、表示座標を画面外に設定します (図 2.4.36 を参照)。

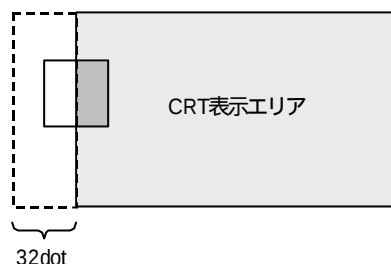
X 座標 (0~255) スプライトの X 座標を設定します。

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 の時は、4 個のパターンを 1 個のスプライトに使いますが、この場合、4 個のパターン番号のうち、どれを指定しても構いません。スプライトのサイズを 8×8 にした場合は 256 種のパターンを、16×16 の場合は 64 種のパターンを指定出来ます。

カラーコード (0~15) スプライトパターンの bit が 1 の部分の色を指定します。

EC (Early Clock) 1 の時、スプライトの表示は 32 ドット左にシフトされます。これにより、スプライトを画面左端から 1 ドットずつ出現させる事が出来ます。

図 2.5.4 EC を 1 にセットした例



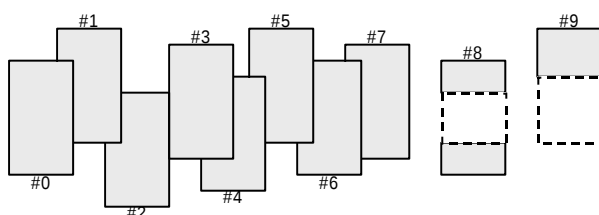
2.5.3 スプライトモード 2

V9938, V9958 の GRAPHIC3 以降 (SCREEN 4 以降) で使用されます。画面モードに GRAPHIC3 ~ 7, YJK/RGB 混在, YJK (SCREEN 4 ~ 8, 10 ~ 12) を選択すると、自動的にスプライトモード 2 が使用されます。

2.5.3.1 特徴

1 画面に表示出来るスプライトは 32 個で、それぞれに #0 ~ #31 までの番号が付けられています。番号の小さい方が優先度が高く、重なった場合は優先度の高いものが上に表示されます。また、1 水平線上に 9 個以上のスプライトが並んだ場合は優先度の高い順に 8 個までが表示され、9 個目以降は表示されません。

図 2.5.5 スプライトモード 2 の水平方向のスプライト表示数



1 水平線上に 9 個以上のスプライトが並んだ時には、S#0 のビット 6 が 1 になり、S#0 のビット 4 ~ 0 に 9 個目のスプライト番号 (0 ~ 31) がセットされます。

また、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) 時には、S#0 のビット 5 が 1 になり、これによって衝突の発生を知る事が出来ます。この時、衝突座標が S#3 ~ S#6 にセットされます。

スプライトの色を 1 ライン毎に指定出来ます。また、スプライトを組み合わせることで、横方向の色を更に多く指定出来ます。

2.5.3.2 表示

スプライトを表示する時には、次の設定を行って下さい。

2.5.3.2.1 スプライトのサイズ・拡大・消去・カラーコード 0 の扱い

スプライトモード 1 と同様ですが、GRAPHIC7 (SCREEN 8) で R#8 のビット 5 に 1 を指定した (カラーコード 0 を非透明色にした) 場合、カラーコード 0 が R=0, B=0, G=0 の固定色になる事に注意して下さい。

2.5.3.2.2 スプライトパターンジェネレータテーブル

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）。

2.5.3.2.3 スプライトアトリビュートテーブル

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・パターン番号をセットします（#0～#31）。

2.5.3.2.4 スプライトカラーテーブル

VRAM 上のスプライトカラーテーブルに、スプライトの各ラインの色・衝突検出の有無、優先順位の有無等をセットします（#0～#31）。

2.5.3.3 スプライトパターンジェネレータテーブル

スプライトモード1と同様です。2.5.2.3 スプライトパターンジェネレータテーブルを参照して下さい。

2.5.3.4 スプライトアトリビュートテーブル

スプライトアトリビュートテーブルは、32 個のスプライトの各々の表示位置(X,Y)、表示するパターン番号を指定する為に VRAM 上に確保される領域です。

1 個のスプライトに 4 バイトのデータ領域が確保されており、テーブル全体で 128 バイトの大きさを持ちます。

スプライトアトリビュートテーブルの先頭アドレスは、R#5 と R#11 にセットします。指定出来るのは先頭アドレスの上位 8 ビット（A16～A9）のみで、下位 9 ビット（A8～A0）は 0 と見なされます。ただし、A9 には必ず 1 をセットします。よって、スプライトアトリビュートテーブルの先頭アドレスとして指定出来るのは、0200H から 1KB 単位の位置になります。

Sprite attribute table base address register

R# 5	A14	A13	A12	A11	A10	A9	1	1
R#11	0	0	0	0	0	0	A16	A15

A9 必ず 1 をセット

図 2.5.6 スプライトモード2のスプライトアトリビュートテーブル

+000	Y 座標 (0～255)	}	スプライト番号#0 アトリビュートエリア
+001	X 座標 (0～255)		
+002	パターン番号 (0～255)		
+003	未使用		
:	:	}	スプライト番号#31 アトリビュートエリア
+124	Y 座標 (0～255)		
+125	X 座標 (0～255)		
+126	パターン番号 (0～255)		
+127	未使用		

Y 座標 (0～255) スプライトの Y 座標を設定します。画面最上端の Y 座標が 255 である事に注意して下さい。

スプライトの Y 座標の値を 216 に設定すると、そのスプライトとそのスプライトより優先度の低い（パターン番号の大きい）スプライトは表示されません。例えば、スプライト番号#16のスプライトの Y 座標を 216 にすると、#16～#31 までのスプライトは表示されません。特定のスプライトのみを画面から消去するには、表示座標を画面外に設定します（図 2.4.36 を参照）。

X 座標 (0～255) スプライトの X 座標を設定します。

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の時は、4個のパターンを1個のスプライトに使いますが、この場合、4個のパターン番号のうち、どれを指定しても構いません。スプライトのサイズを8×8にした場合は256種のパターンを、16×16の場合は64種のパターンを指定出来ます。

2.5.3.5 スプライトカラーテーブル

スプライトカラーテーブルは、スプライトの各ラインの1の部分の表示色（パターンの0の部分は透明）、優先順位の有無、衝突検出の有無、EC（Early Clock）の指定を行うテーブルです。各スプライトに16ライン分の領域が確保されていますが、スプライトサイズが8×8の時は、先頭の8ライン分を使用して下さい。

図 2.5.7 スプライトカラーテーブル

+000	EC	CC	IC	0	カラーコード	1st line
+001	EC	CC	IC	0	カラーコード	2nd line
:	:	:	:	:	:	:
+015	EC	CC	IC	0	カラーコード	16 th line
:	:	:	:	:	:	:
+496	EC	CC	IC	0	カラーコード	1st line
:	:	:	:	:	:	:
+511	EC	CC	IC	0	カラーコード	16 th line

スプライト番号#0 カラーテーブル

スプライト番号#31 カラーテーブル

EC（Early Clock） 1の時、スプライトの指定されたラインの表示は32ドット左にシフトされます。これにより、スプライトを画面左端から1ドットずつ出現させる事が出来ます。

CC 1の時、スプライトの指定されたラインの優先順位が無くなり、衝突検出が行われなくなります。スプライトのCCを1にしたラインは、そのスプライトより番号が小さく、かつCCが0のスプライトが表示されている水平線上のみに表示されます（図2.5.8を参照）。

CCを1にした部分が、そのスプライトより番号が小さく、かつ番号が最も近いスプライトのCCが0の部分と重なった場合、衝突検出は行われません。例えば、スプライト番号#2（CC=ALL 0）、#3（CC=ALL 0）、#4（CC=ALL 1）を表示した場合、#2と#3、#2と#4が重なった場合は衝突が発生しますが、#3と#4が重なっても衝突検出は行われません。また、CCが1の部分同士が重なっても、衝突検出は行われません。衝突検出が行われない場合、重なった部分のカラーコードは各スプライトのカラーコードの論理和（OR）になります（上の例では、#3と#4が重なった場合）。

この機能を利用して2枚以上のスプライトを重ねて使うことにより、横方向の色数を増やせます。

IC 1の時、スプライトの指定されたラインの衝突検出が行われなくなります。

Color code スプライトの各ラインのカラーコード（0～15）を指定します。GRAPHIC7（SCREEN 8）の時はパレットは使用されず、固定色が使われます（表2.5.1を参照）。

図 2.5.8 スプライトカラーテーブルの CC の効果

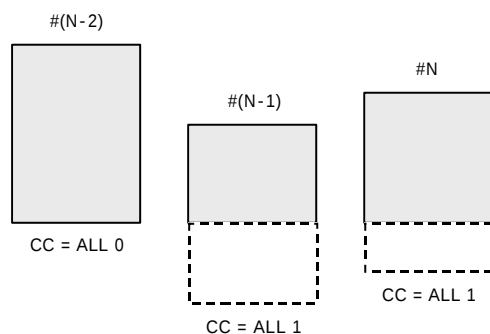
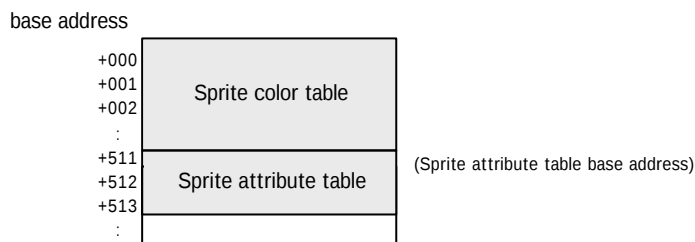


表 2.5.1 GRAPHIC7（SCREEN 8）モード時のスプライトの色

カラーコード	G	R	B
0	0	0	0
1	0	0	2
2	0	3	0
3	0	3	2
4	3	0	0
5	3	0	2
6	3	3	0
7	3	3	2
8	4	7	2
9	0	0	7
10	0	7	0
11	0	7	7
12	7	0	0
13	7	0	7
14	7	7	0
15	7	7	7

スプライトカラーテーブルは、スプライトアトリビュートテーブルの先頭アドレスから512バイトを引いた場所に作成されます。1個のスプライトに16バイトの領域が確保され、スプライトカラーテーブル全体は512バイトの大きさを持ちます。

図 2.5.9 スプライトカラーテーブルとスプライトアトリビュートテーブルの位置



2.5.3.6 スプライトの衝突

スプライトの、CC, IC が 0 でかつ透明でない部分同士が重なると衝突が発生します(正確な条件は、2.5.3.5 スプライトカラーテーブルの CC, IC に関する説明を参照)。衝突が発生すると、S#0 のビット 5 が 1 になります。このビットは S#0 を読み出す事によってリセットされます。

この時、R#8 のビット 7, 6 がセットされていないければ (MSX ではセットされる事はない)、S#3 ~ S#6 にスプライトの衝突座標がセットされます。

S# 3	X7	X6	X5	X4	X3	X2	X1	X0	Column register(low)
S# 4	1	1	1	1	1	1	1	X8	Column register(high)
X8 ~ X0 衝突座標 X (12 ~ 267)									
S# 5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Row register(low)
S# 6	1	1	1	1	1	1	E0	Y8	Row register(high)
Y8 ~ Y0 衝突座標 Y (9 ~ 264)									
E0 表示フィールドフラグ (インターレースモード & 2画面交互表示の時に意味を持つ)									
現在の表示フィールドが 0=第1フィールド、1=第2フィールド									

これらのレジスタは、S#5 を読み出した時にリセットされます。

また、このレジスタ群にセットされる値には、オフセットがつけられており、実際の衝突座標は次のようになります。

実際の衝突座標 (XC, YC)
 $XC = X - 12$ ($X = X8 \sim X0$)
 $YC = Y - 9$ ($Y = Y8 \sim Y0$)

2.5.4 スプライトに対するハードウェアスクロールの影響

R#23 (Display offset register) によるハードウェア縦スクロールは、スプライトにも影響を与えます。

R#23 を設定する事により、スプライト面はパターン面と同様に縦方向にスクロールします。この為、パターン面を縦スクロールさせながらスプライトを一定の位置に置く場合 (シューティングゲーム等)、スプライトの Y 座標に補正を加える等の操作が必要となります。Y 座標を補正する場合、Y 座標を 208 (スプライトモード 1 の場合、モード 2 では 216) に設定すると、そのスプライトより優先順位の低いスプライトが表示されなくなる事に注意して下さい。

なお、R#26, R#27 によるハードウェア横スクロールは、パターン面に対してのみ有効であり、スプライトがその影響を受けることはありません。

6章 VDP コマンド

この章では、V9938, V9958 の VDP コマンドについて説明します。

2.6.1 V9938, V9958 のコマンド

VRAM にアクセスする方法の一つに、V9938, V9958 に備わっているコマンドを用いる、というものがあ
ります。このコマンドを VDP コマンドと呼び、描画 (LINE, PSET 等) や VRAM の部分転送が手軽に行えま
す。

VDP コマンドは、V9938 では GRAPHIC4 ~ 7, YJK/RGB 混在, YJK (SCREEN 5 ~ 8, 10 ~ 12) でのみ使用可
能です。V9958 では、R#25 のビット 6 をセットすると、全ての画面モードで VDP コマンドの使用が可能に
なります。この時のパラメータは、GRAPHIC7 (SCREEN 8) の座標系で指定します。

表 2.6.1 に VDP コマンドの一覧を示します。

表 2.6.1 V9938, V9958 のコマンド一覧

コマンド名	コマンドの内容	転送単位	論理演算	R#46 の bit 7 ~ 4
HMMC	CPU → VRAM 高速転送	byte	無効	1 1 1 1
YMMM	Y 方向の VRAM 間高速転送	byte	無効	1 1 1 0
HMMM	VRAM 間高速転送	byte	無効	1 1 0 1
HMMV	矩形領域の高速塗り潰し	byte	無効	1 1 0 0
LMMC	CPU → VRAM 論理転送	dot	有効	1 0 1 1
LMCM	VRAM → CPU 論理転送	dot	無効	1 0 1 0
LMMM	VRAM 間論理転送	dot	有効	1 0 0 1
LMMV	矩形領域の論理塗り潰し	dot	有効	1 0 0 0
LINE	直線の描画	dot	有効	0 1 1 1
SRCH	カラーコードのサーチ	dot	無効	0 1 1 0
PSET	点の描画	dot	有効	0 1 0 1
POINT	カラーコードの読み出し	dot	無効	0 1 0 0
STOP	VDP コマンドの中断	-	-	0 0 0 0

2.6.2 VDP コマンドにおける座標系

VDP コマンドでの位置パラメータは、(X,Y) 座標で指定します。この時の座標系は、VRAM の全領域をカ
バーし、ページに関わらず連続的にアクセス出来ます。表 2.6.2 に、VDP コマンドの座標系と各画面モード
との関係を示します。

表 2.6.2 VDP コマンドの座標系と画面モード

画面モード	X 座標の範囲	Y 座標の範囲
GRAPHIC4	0 ~ 255	0 ~ 1023
GRAPHIC5	0 ~ 511	0 ~ 1023
GRAPHIC6	0 ~ 511	0 ~ 511
GRAPHIC7	0 ~ 255	0 ~ 511
YJK/RGB 混在	0 ~ 255	0 ~ 511
YJK	0 ~ 255	0 ~ 511

2.6.3 ロジカルオペレーション

一部のコマンドでは、転送元のデータと転送先のデータとで論理演算を行う事が出来ます。論理演算（ロジカルオペレーション）は、コマンドを実行する時に R#46 のビット 3 ~ 0 に書き込みます。表 2.6.3 に論理演算の一覧を示します。

表 2.6.3 ロジカルオペレーションコード一覧

演算名	BASIC での演算名	演算動作(書き込まれる値)	R#46 の bit 3 ~ 0
IMP	PSET	SC	0 0 0 0
AND	AND	SC AND DC	0 0 0 1
OR	OR	SC OR DC	0 0 1 0
EOR	XOR	SC XOR DC	0 0 1 1
NOT	PRESET	NOT SC	0 1 0 0
TIMP	TPSET	if SC=0 then DC else SC	1 0 0 0
TAND	TAND	if SC=0 then DC else SC AND DC	1 0 0 1
TOR	TOR	if SC=0 then DC else SC OR DC	1 0 1 0
TEOR	TXOR	if SC=0 then DC else SC XOR DC	1 0 1 1
TNOT	TPRESET	if SC=0 then DC else NOT SC	1 1 0 0

SC = Source Color code (転送元のカラーコード)

DC = Destination Color code (転送先のカラーコード)

2.6.4 各コマンドの説明

各コマンドの内容・実行方法を説明します。

2.6.4.1 HMMC (CPU→VRAM 高速転送)

CPU から VRAM 上の矩形領域へ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為、X 座標として指定出来る値は、画面モードによって制限を受けます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register (low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register (high)

DX8 ~ DX0 転送先基準点の X 座標 (0 ~ 511)

GRAPHIC4, 6 では DX0 が、GRAPHIC5 では DX1 ~ DX0 が無視される

R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register (low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register (high)

DY9 ~ DY0 転送先基準点の Y 座標 (0 ~ 1023)

R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of X dot register (low)
R#41	0	0	0	0	0	0	0	NX8	Number of X dot register (high)

NX8 ~ NX0 X 方向転送ドット数 (1 ~ 512)

512 を指定する時は、NX8 ~ NX0 に 0 をセットする

GRAPHIC4, 6 では NX0 が、GRAPHIC5 では NX1 ~ NX0 が無視される

R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register (low)
R#43	0	0	0	0	0	0	NY9	NY8	Number of Y dot register (high)

NY8 ~ NY0 Y 方向転送ドット数 (1 ~ 1024)

1024 を指定する時は、NY9 ~ NY0 に 0 をセットする

R#44

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

 Color register
CR7～CR0 転送データの1バイト目

R#45

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

 Argument register
MXD 転送先のメモリ選択 0=VRAM、1=拡張RAM (MSXでは常に0を指定)
DIX 転送先基準点からの転送方向(X軸方向) 0=右、1=左
DIY 転送先基準点からの転送方向(Y軸方向) 0=下、1=上

コマンドの実行

R#46 に、11110000B を書き込むとコマンドが実行されます。

R#46

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

 Command register

コマンド終了の確認

S#2 を読んで、CE (S#2 の bit0) が0であれば、コマンドの終了です。

2バイト目以降の転送

S#2 を読んで、TR (S#2 の bit7) が1であれば、データ1バイトを R#44 に書き込みます。TR が0の時は、1になるまで待ちます。

以下、手順 ～ を繰り返します。

2.6.4.2 YMMM (Y方向のVRAM間高速転送)

VRAM 上の転送元基準点、Y方向転送ドット数、転送方向と画面の右端(左端)で指定される矩形領域をY方向に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為、X座標として指定出来る値は、画面モードによって制限を受けます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#34

SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
-----	-----	-----	-----	-----	-----	-----	-----

 Source Y register(low)

R#35

0	0	0	0	0	0	SY9	SY8
---	---	---	---	---	---	-----	-----

 Source Y register(high)
SY9～SY0 転送元基準点のY座標(0～1023)

R#36

DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination X register(low)

R#37

0	0	0	0	0	0	0	DX8
---	---	---	---	---	---	---	-----

 Destination X register(high)
DX8～DX0 転送元及び転送先基準点のX座標(0～511)
GRAPHIC4, 6ではDX0が、GRAPHIC5ではDX1～DX0が無視される

R#38

DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination Y register(low)

R#39

0	0	0	0	0	0	DY9	DY8
---	---	---	---	---	---	-----	-----

 Destination Y register(high)
DY9～DY0 転送先基準点のY座標(0～1023)

R#42

NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
-----	-----	-----	-----	-----	-----	-----	-----

 Number of Y dot register(low)

R#43

0	0	0	0	0	0	NY9	NY8
---	---	---	---	---	---	-----	-----

 Number of Y dot register(high)

NY9~NY0 Y 方向転送ドット数 (1~1024)

1024 を指定する時は、NY9~NY0 に 0 をセットする

R#45

0	0	MXD	MSX	DIY	DIX	0	0
---	---	-----	-----	-----	-----	---	---

 Argument register

MXD 転送先のメモリ選択 0=VRAM、1=拡張 RAM (MSX では常に 0 を指定)

MSX 転送元のメモリ選択 0=VRAM、1=拡張 RAM (MSX では常に 0 を指定)

DIX 転送元基準点からどちらの画面端までを転送するのかを指定 0=右端、1=左端

DIY 転送元基準点からの転送方向 (Y 軸方向) 0=下、1=上

コマンドの実行

R#46 に、11100000B を書き込むとコマンドが実行されます。

R#46

1	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

 Command register

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

2.6.4.3 HMMM (VRAM 間高速転送)

VRAM から VRAM へ矩形領域の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為、X 座標として指定出来る値は、画面モードによって制限を受けます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#32

SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0
-----	-----	-----	-----	-----	-----	-----	-----

 Source X register(low)

R#33

0	0	0	0	0	0	0	SX8
---	---	---	---	---	---	---	-----

 Source X register(high)

SX8~SX0 転送元基準点の X 座標 (0~511)

GRAPHIC4, 6 では SX0 が、GRAPHIC5 では SX1~SX0 が無視される

R#34

SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
-----	-----	-----	-----	-----	-----	-----	-----

 Source Y register(low)

R#35

0	0	0	0	0	0	SY9	SY8
---	---	---	---	---	---	-----	-----

 Source Y register(high)

SY9~SY0 転送元基準点の Y 座標 (0~1023)

R#36

DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination X register(low)

R#37

0	0	0	0	0	0	0	DX8
---	---	---	---	---	---	---	-----

 Destination X register(high)

DX8~DX0 転送先基準点の X 座標 (0~511)

GRAPHIC4, 6 では DX0 が、GRAPHIC5 では DX1~DX0 が無視される

R#38

DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination Y register(low)

R#39

0	0	0	0	0	0	DY9	DY8
---	---	---	---	---	---	-----	-----

 Destination Y register(high)

DY9~DY0 転送先基準点の Y 座標 (0~1023)

R#40	<table><tr><td>NX7</td><td>NX6</td><td>NX5</td><td>NX4</td><td>NX3</td><td>NX2</td><td>NX1</td><td>NX0</td></tr></table>	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X register(low)
NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0			
R#41	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>NX8</td></tr></table>	0	0	0	0	0	0	0	NX8	Number of dot X register(high)
0	0	0	0	0	0	0	NX8			

NX8 ~ NX0 X方向転送ドット数 (1~512)

512を指定する時は、NX8 ~ NX0 に0をセットする

GRAPHIC4, 6 ではNX0 が、GRAPHIC5ではNX1 ~ NX0 が無視される

R#42	<table><tr><td>NY7</td><td>NY6</td><td>NY5</td><td>NY4</td><td>NY3</td><td>NY2</td><td>NY1</td><td>NY0</td></tr></table>	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register(low)
NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0			
R#43	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>NY9</td><td>NY8</td></tr></table>	0	0	0	0	0	0	NY9	NY8	Number of Y dot register(high)
0	0	0	0	0	0	NY9	NY8			

NY9 ~ NY0 Y方向転送ドット数 (1~1024)

1024を指定する時は、NY9 ~ NY0 に0をセットする

R#45	0	0	MXD	MXS	DIY	DIX	0	0	Argument register
	MXD	転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)							
	MXS	転送元のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)							
	DIX	転送基準点からの転送方向 (X 軸方向) 0=右、1=左							
	DIY	転送基準点からの転送方向 (Y 軸方向) 0=下、1=上							

コマンドの実行

R#46 に、11010000B を書き込むとコマンドが実行されます。

R#46	<table><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	0	1	0	0	0	0	Command register
1	1	0	1	0	0	0	0			

コマンド終了の確認

コマンド実行中はCE (S#2 の bit0) が1になります。CE が0になれば、コマンドの終了です。

2.6.4.4 HMMV (矩形領域の高速塗り潰し)

VRAM 上の矩形領域を塗り潰すコマンドです。データの転送は1バイト単位で行われる為、X座標として指定出来る値は、画面モードによって制限を受けます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register(low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register(high)

DX8 ~ DX0 転送先基準点のX座標 (0~511)

GRAPHIC4, 6 ではDX0 が、GRAPHIC5ではDX1 ~ DX0 が無視される

R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register(low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register(high)

DY9 ~ DY0 転送先基準点のY座標 (0~1023)

R#40	<table><tr><td>NX7</td><td>NX6</td><td>NX5</td><td>NX4</td><td>NX3</td><td>NX2</td><td>NX1</td><td>NX0</td></tr></table>	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X register(low)
NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0			
R#41	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>NX8</td></tr></table>	0	0	0	0	0	0	0	NX8	Number of dot X register(high)
0	0	0	0	0	0	0	NX8			

NX8 ~ NX0 X方向転送ドット数 (1~512)

512を指定する時は、NX8 ~ NX0 に0をセットする

GRAPHIC4, 6 ではNX0 が、GRAPHIC5ではNX1 ~ NX0 が無視される

R#42

NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
-----	-----	-----	-----	-----	-----	-----	-----

 Number of Y dot register(low)

R#43

0	0	0	0	0	0	NY9	NY8
---	---	---	---	---	---	-----	-----

 Number of Y dot register(high)

NY9~NY0 Y 方向転送ドット数 (1~1024)

1024 を指定する時は、NY9~NY0 に 0 をセットする

R#44

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

 Color register

CR7~CR0 塗り潰しデータ (1 バイト)

GRAPHIC4 CR7~CR4: X = 2N (N=0~127)

CR3~CR0: 2N+1

GRAPHIC5 CR7~CR6: X = 4N (N=0~127)

CR5~CR4: 4N+1

CR3~CR2: 4N+2

CR1~CR0: 4N+3

GRAPHIC4 CR7~CR4: X = 2N (N=0~255)

CR3~CR0: 2N+1

GRAPHIC7 CR7~CR0: X = N (N=0~255)

R#45

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

 Argument register

MXD 転送先のメモリ選択 0=VRAM、1=拡張 RAM (MSX では常に 0 を指定)

DIX 転送先基準点からの転送方向 (X 軸方向) 0=右、1=左

DIY 転送先基準点からの転送方向 (Y 軸方向) 0=下、1=上

コマンドの実行

R#46 に、11000000B を書き込むとコマンドが実行されます。

R#46

1	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 Command register

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

2.6.4.5 LMMC (CPU→VRAM 論理転送)

CPU から VRAM 上の矩形領域へデータを転送するコマンドです。データの転送はドット単位で行われ、転送先のデータと論理演算出来ます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36

DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination X register (low)

R#37

0	0	0	0	0	0	0	DX8
---	---	---	---	---	---	---	-----

 Destination X register (high)

DX8~DX0 転送先基準点の X 座標 (0~511)

R#38

DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
-----	-----	-----	-----	-----	-----	-----	-----

 Destination Y register (low)

R#39

0	0	0	0	0	0	DY9	DY8
---	---	---	---	---	---	-----	-----

 Destination Y register (high)

DY9~DY0 転送先基準点の Y 座標 (0~1023)

R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of X dot register(low)
R#41	0	0	0	0	0	0	0	NX8	Number of X dot register(high)
NX8 ~ NX0 X 方向転送ドット数 (1~512)									
512 を指定する時は、NX8 ~ NX0 に 0 をセットする									

R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register(low)
R#43	0	0	0	0	0	0	0	NY9	Number of Y dot register(high)
NY8 ~ NY0 Y 方向転送ドット数 (1~1024)									
1024 を指定する時は、NY9 ~ NY0 に 0 をセットする									

Color register

R#44	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
R#44	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7,YJK/RGB 混在,YJK
CR7 ~ CR0 転送データの1バイト目									

R#45	0	0	MXD	0	DIY	DIX	0	0	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張RAM (MSXでは常に0を指定)									
DIX 転送先基準点からの転送方向(X軸方向) 0=右、1=左									
DIY 転送先基準点からの転送方向(Y軸方向) 0=下、1=上									

コマンドの実行

R#46 の上位4ビットに1011B、下位4ビットにロジカルオペレーションコード(表2.6.3を参照)を書き込むとコマンドが実行されます。

R#46	1	0	1	1	L03	L02	L01	L00	Command register
L03 ~ L00 ロジカルオペレーションコード									

コマンド終了の確認

S#2 を読んで、CE (S#2 の bit0) が0であれば、コマンドの終了です。

2バイト目以降の転送

S#2 を読んで、TR (S#2 の bit7) が1であれば、データ1バイトを R#44 に書き込みます。TR が0の時は、1になるまで待ちます。

以下、手順 ~ を繰り返します。

2.6.4.6 LMCM (VRAM → CPU 論理転送)

VRAM 上の矩形領域のデータを CPU へ転送するコマンドです。データの転送はドット単位で行われます。以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットし、更に S#7 を読んで TR (S#2 の bit7) を 0 にします。

R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X register(low)
R#33	0	0	0	0	0	0	0	SX8	Source X register(high)
SX8 ~ SX0 転送元基準点の X 座標 (0 ~ 511)									
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y register(low)
R#35	0	0	0	0	0	0	SY9	SY8	Source Y register(high)
SY9 ~ SY0 転送元基準点の Y 座標 (0 ~ 1023)									
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X register(low)
R#41	0	0	0	0	0	0	0	NX8	Number of dot X register(high)
NX8 ~ NX0 X 方向転送ドット数 (1 ~ 512) 512 を指定する時は、NX8 ~ NX0 に 0 をセットする									
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register(low)
R#43	0	0	0	0	0	0	NY9	NY8	Number of Y dot register(high)
NY9 ~ NY0 Y 方向転送ドット数 (1 ~ 1024) 1024 を指定する時は、NY9 ~ NY0 に 0 をセットする									
R#45	0	0	0	MXS	DIY	DIX	0	0	Argument register
MXS 転送元のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定) DIX 転送基準点からの転送方向 (X 軸方向) 0=右, 1=左 DIY 転送基準点からの転送方向 (Y 軸方向) 0=下, 1=上									

コマンドの実行

R#46 に、10100000B を書き込むとコマンドが実行されます。

R#46	1	0	1	0	0	0	0	0	Command register
------	---	---	---	---	---	---	---	---	------------------

データの読み込み

S#2 を読んで、TR (S#2 の bit7) が 1 であれば、S#7 を読み込みます。TR が 0 の時は、1 になるまで待ちます。

Color register

S#7	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
S#7	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
S#7	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 データ 1 ドット分がセットされる									

コマンド終了の確認

S#2 を読んで、CE (S#2 の bit0) が 0 であれば、コマンドの終了です。

以下、手順 ~ を繰り返します。

2.6.4.7 LMMM (VRAM 間論理転送)

VRAM から VRAM へ矩形領域のデータを転送するコマンドです。データの転送はドット単位で行われ、転送先のデータと論理演算出来ます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X register(low)
R#33	0	0	0	0	0	0	0	SX8	Source X register(high)
SX8 ~ SX0 転送元基準点の X 座標 (0 ~ 511)									
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y register(low)
R#35	0	0	0	0	0	0	SY9	SY8	Source Y register(high)
SY9 ~ SY0 転送元基準点の Y 座標 (0 ~ 1023)									
R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register(low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register(high)
DX8 ~ DX0 転送先基準点の X 座標 (0 ~ 511)									
R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register(low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register(high)
DY9 ~ DY0 転送先基準点の Y 座標 (0 ~ 1023)									
R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X register(low)
R#41	0	0	0	0	0	0	0	NX8	Number of dot X register(high)
NX8 ~ NX0 X 方向転送ドット数 (1 ~ 512)									
512 を指定する時は、NX8 ~ NX0 に 0 をセットする									
R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register(low)
R#43	0	0	0	0	0	0	NY9	NY8	Number of Y dot register(high)
NY9 ~ NY0 Y 方向転送ドット数 (1 ~ 1024)									
1024 を指定する時は、NY9 ~ NY0 に 0 をセットする									
R#45	0	0	MXD	MXS	DIY	DIX	0	0	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									
MXS 転送元のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									
DIX 転送基準点からの転送方向 (X 軸方向) 0=右, 1=左									
DIY 転送基準点からの転送方向 (Y 軸方向) 0=下, 1=上									

コマンドの実行

R#46 の上位 4 ビットに 1001B、下位 4 ビットにロジカルオペレーションコード (表 2.6.3 を参照) を書き込むとコマンドが実行されます。

R#46	1	0	0	1	L03	L02	L01	L00	Command register
L03 ~ L00 ロジカルオペレーションコード									

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

2.6.4.8 LMMV (矩形領域の論理塗り潰し)

VRAM 上の矩形領域を塗り潰すコマンドです。データの転送はドット単位で行われ、転送先のデータと論理演算出来ます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register(low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register(high)
DX8 ~ DX0 転送先基準点の X 座標 (0 ~ 511)									

R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register(low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register(high)
DY9 ~ DY0 転送先基準点の Y 座標 (0 ~ 1023)									

R#40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	Number of dot X register(low)
R#41	0	0	0	0	0	0	0	NX8	Number of dot X register(high)
NX8 ~ NX0 X 方向転送ドット数 (1 ~ 512) 512 を指定する時は、NX8 ~ NX0 に 0 をセットする									

R#42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	Number of Y dot register(low)
R#43	0	0	0	0	0	0	NY9	NY8	Number of Y dot register(high)
NY9 ~ NY0 Y 方向転送ドット数 (1 ~ 1024) 1024 を指定する時は、NY9 ~ NY0 に 0 をセットする									

Color register

R#44	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
R#44	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 塗り潰しデータ (1 ドット)									

R#45	0	0	MXD	0	DIY	DIX	0	0	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									
DIX 転送先基準点からの転送方向 (X 軸方向) 0=右, 1=左									
DIY 転送先基準点からの転送方向 (Y 軸方向) 0=下, 1=上									

コマンドの実行

R#46 の上位 4 ビットに 1000B、下位 4 ビットにロジカルオペレーションコード (表 2.6.3 を参照) を書き込むとコマンドが実行されます。

R#46	1	0	0	0	L03	L02	L01	L00	Command register
L03 ~ L00 ロジカルオペレーションコード									

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

2.6.4.9 LINE (直線の描画)

基準点と長辺・短辺で指定される長方形の対角線を描画するコマンドです。データの転送はドット単位で行われ、転送先のデータと論理演算出来ます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register(low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register(high)
DX8 ~ DX0 転送先基準点の X 座標 (0 ~ 511)									

R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register(low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register(high)
DY9 ~ DY0 転送先基準点の Y 座標 (0 ~ 1023)									

R#40	MJ7	MJ6	MJ5	MJ4	MJ3	MJ2	MJ1	MJ0	Number of dot X register(low)
R#41	0	0	0	0	0	0	MJ9	MJ8	Number of dot X register(high)
MJ9 ~ MJ0 長辺ドット数 (0 ~ 1023)									

R#42	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0	Number of Y dot register(low)
R#43	0	0	0	0	0	0	0	MI8	Number of Y dot register(high)
MI8 ~ MI0 短辺ドット数 (0 ~ 511)									

Color register

R#44	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
R#44	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 描画データ (1 ドット)									

R#45	0	0	MXD	0	DIY	DIX	0	MAJ	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									
DIX 転送先基準点からの転送方向 (X 軸方向) 0=右, 1=左									
DIY 転送先基準点からの転送方向 (Y 軸方向) 0=下, 1=上									
MAJ 長辺は 0=X 軸と平行, 1=Y 軸と平行 / 又は長辺=短辺									

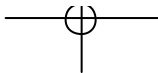
コマンドの実行

R#46 の上位 4 ビットに 0111B、下位 4 ビットにロジカルオペレーションコード (表 2.3.6 を参照) を書き込むとコマンドが実行されます。

R#46	0	1	1	1	L03	L02	L01	L00	Command register
L03 ~ L00 ロジカルオペレーションコード									

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。



2.6.4.10 SRCH (カラーコードのサーチ)

基準点から X 軸方向に境界色 (又は非境界色) をサーチするコマンドです。
以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X register(low)
R#33	0	0	0	0	0	0	0	SX8	Source X register(high)
SX8 ~ SX0 サーチ基準点の X 座標 (0 ~ 511)									

R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y register(low)
R#35	0	0	0	0	0	0	SY9	SY8	Source Y register(high)
SY9 ~ SY0 サーチ基準点の Y 座標 (0 ~ 1023)									

Color register

R#44	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
R#44	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 境界色									

R#45	0	0	MXD	0	0	DIX	EQ	0	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									
DIX サーチ基準点からのサーチ方向 (X 軸方向) 0=右, 1=左									
EQ 実行終了条件 0=境界色を発見した時, 1=境界色以外の色を発見した時									

コマンドの実行

R#46 に、01100000B を書き込むとコマンドが実行されます。

R#46	0	1	1	0	0	0	0	0	Command register
------	---	---	---	---	---	---	---	---	------------------

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

実行結果の読み出し

コマンド終了時、BD (S#2 の bit4) が 1 であれば EQ (R#45 の bit1) の終了条件を満たした事を示し、S#8 と S#9 に発見した X 座標がセットされます。BD が 0 の場合は、画面端まで検索して終了条件を満たさなかった事を示します。

S#2

	-	BD	1	1	-	CE
--	---	----	---	---	---	----

Status register 2

BD

EQ (R#45 の bit1) の終了条件を満たした時に 1 となる

CE

0 であればコマンド終了を示す

S#8

BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0
-----	-----	-----	-----	-----	-----	-----	-----

Border X register (low)

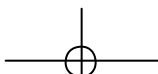
S#9

1	1	1	1	1	1	1	BX8
---	---	---	---	---	---	---	-----

Border X register (high)

BX8 ~ BX0

境界色を発見した X 座標 (又は、境界色以外の色を発見した X 座標)



2.6.4.11 PSET (点の描画)

VRAM 上に点を描画するコマンドです。データの転送はドット単位で行われ、転送先のデータと論理演算出来ます。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	Destination X register(low)
R#37	0	0	0	0	0	0	0	DX8	Destination X register(high)
DX8 ~ DX0 転送先基準点の X 座標 (0 ~ 511)									

R#38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	Destination Y register(low)
R#39	0	0	0	0	0	0	DY9	DY8	Destination Y register(high)
DY9 ~ DY0 転送先基準点の Y 座標 (0 ~ 1023)									

Color register

R#44	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
R#44	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
R#44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 描画色 (1 ドット)									

R#45	0	0	MXD	0	0	0	0	0	Argument register
MXD 転送先のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									

コマンドの実行

R#46 の上位 4 ビットに 0101B、下位 4 ビットにロジカルオペレーションコード (表 2.6.3 を参照) を書き込むとコマンドが実行されます。

R#46	0	1	0	1	L03	L02	L01	L00	Command register
L03 ~ L00 ロジカルオペレーションコード									

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

2.6.4.12 POINT (カラーコードの読み出し)

VRAM 上の基準点のカラーコードを読み出すコマンドです。

以下に、実行手順を説明します。

レジスタのセット

以下のレジスタをセットします。

R#32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	Source X register(low)
R#33	0	0	0	0	0	0	0	SX8	Source X register(high)
SX8 ~ SX0 転送元基準点の X 座標 (0 ~ 511)									
R#34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	Source Y register(low)
R#35	0	0	0	0	0	0	SY9	SY8	Source Y register(high)
SY9 ~ SY0 転送元基準点の Y 座標 (0 ~ 1023)									
R#45	0	0	0	MXS	0	0	0	0	Argument register
MXS 転送元のメモリ選択 0=VRAM, 1=拡張 RAM (MSX では常に 0 を指定)									

コマンドの実行

R#46 に、01000000B を書き込むとコマンドが実行されます。

R#46	0	1	0	0	0	0	0	0	Command register
------	---	---	---	---	---	---	---	---	------------------

コマンド終了の確認

コマンド実行中は CE (S#2 の bit0) が 1 になります。CE が 0 になれば、コマンドの終了です。

実行結果の読み出し

読み出されたカラーコードが S#7 にセットされます。

Color register

S#7	0	0	0	0	CR3	CR2	CR1	CR0	GRAPHIC4,6
S#7	0	0	0	0	0	0	CR1	CR0	GRAPHIC5
S#7	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	GRAPHIC7, YJK/RGB 混在, YJK
CR7 ~ CR0 読み出されたカラーコード									

2.6.4.13 STOP (VDP コマンドの中断)

VDP コマンドを中断します。R#46 に、00000000B を書き込むと、実行中のコマンドが中断されます。中断したコマンドは、R#46 に同じ値を書き込むことで再開できます。

R#46	0	0	0	0	0	0	0	0	Command register
------	---	---	---	---	---	---	---	---	------------------

2.6.5 コマンド終了時のレジスタの状態

VDP コマンド終了時のレジスタの状態は表 2.6.4 のようになります。

表 2.6.4 コマンド終了時のレジスタの状態

	R#32, 33	R#34, 35	R#36, 37	R#38, 39	R#40, 41	R#42, 43	R#44	R#45	R#46(H)	R#46(L)
HMMC	-	-	-	(2)	-	(3)	-	-	0	-
YMMM	-	(1)	-	(2)	-	(3)	-	-	0	-
HMMM	-	(1)	-	(2)	-	(3)	-	-	0	-
HMMV	-	-	-	(2)	-	(3)	-	-	0	-
LMMC	-	-	-	(2)	-	(3)	-	-	0	-
LMCM	-	(1)	-	-	-	(3)	(4)	-	0	-
LMVM	-	(1)	-	(2)	-	(3)	-	-	0	-
LMMV	-	-	-	(2)	-	(3)	-	-	0	-
LINE	-	-	-	(2)	-	-	-	-	0	-
SRCH	-	-	-	-	-	-	-	-	0	-
PSET	-	-	-	-	-	-	-	-	0	-
POINT	-	-	-	-	-	-	(4)	-	0	-

- ... 変化しない

(1) コマンド実行前の値を SY、終了時の値を SY*とし、Y 方向実行ドット数を N とすれば、

$$SY^* = SY + N \quad (DIY = 0)$$

$$SY^* = SY - N \quad (DIY = 1)$$

(2) コマンド実行前の値を DY、終了時の値を DY*とし、Y 方向実行ドット数を N とすれば、

$$DY^* = DY + N \quad (DIY = 0)$$

$$DY^* = DY - N \quad (DIY = 1)$$

ただし、LINE で MAJ (R#45 の bit0) が 0 の時は $N = N - 1$ となる。

(3) コマンド実行前の値を NY、終了時の値を NYB とし、Y 方向実行ドット数を N とすれば、

$$NYB = NY - N$$

(4) コマンド終了時のカラーコード

2.6.6 コマンド処理の高速化

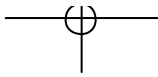
次の操作を行う事で、VDP コマンドの処理速度が上がります。

2.6.6.1 スプライトの表示を禁止する

SPD (R#8 の bit1) を 1 にすると、スプライトの処理に使われていた時間がコマンドの実行に使用される様になり、コマンドの処理速度が上がります

2.6.6.2 画面の表示を禁止する

BL (R#1 の bit6) を 0 にすると、画面表示の処理に使われていた時間がコマンドの実行に使用される様になり、コマンドの処理速度が上がります



2.6.7 VDP コマンドの実行時間

各 VDP コマンドの実行時間は表 2.6.5 の様になっています。表 2.6.5 の値は全て実測値であり、数%の誤差が見込まれます。

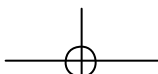
表.2.6.5 VDP コマンドの実行時間

コマンド名	画面表示	画面非表示	スプライト非表示	単位
HMMC	4.20	4.20	4.20	μ秒 / byte
YMMM	5.79	3.04	3.18	μ秒 / byte
HMMM	6.37	4.25	4.55	μ秒 / byte
HMMV	3.03	2.27	2.89	μ秒 / byte
LMMC	6.44	5.04	5.60	μ秒 / dot
LMCM	5.88	4.20	5.04	μ秒 / dot
LMMM	9.09	6.07	6.17	μ秒 / dot
LMMV	6.37	4.56	5.81	μ秒 / dot

注意：表の値は全て実測値です。YMMM, HMMM, HMMV, LMMM, LMMV に関しては、コマンド実行から終了までの時間を MSXturboR のシステムタイマーで測定し、転送量で平均した値を示しました。HMMC, LMMC, LMCM の値は、等間隔でデータの入出力を行った時の、正常動作に最低限必要なアクセス間隔です。

2.6.8 VDP コマンド実行中の処理

VDP の通常の処理と VDP コマンドの処理は独立しており、両者の処理を同時に実行する事が可能です。つまり、VDP コマンドの実行中に、2.2.4 VRAM (VIDEO RAM) のアクセスで説明した方法を用いて VRAM を読み書きしたり、レジスタを書き換えたりする事が出来ます。ただし、実行中の VDP コマンドに関連するレジスタ、VRAM にアクセスした場合、正常な動作は期待出来ません。また、VDP コマンドと並行して VRAM を読み書きした場合、VDP コマンドの実行速度が若干低下します。



7章 その他

この章では、1章から6章までに説明されなかった事項について説明します。

2.7.1 V9958 で削除された機能

V9938 に存在する機能で、V9958 では削除された物を説明します。

- ・コンボジットビデオ出力
- ・マウス、ライトペンインターフェイス

以上の機能が削除されています。いずれも、MSX2 では使用されていなかった機能です。
これに伴い、レジスタの機能にも変更があります。削除されたビットには常に0を設定して下さい。

R#0

0	DG	IE2	IE1	M5	M4	M3	0
---	----	-----	-----	----	----	----	---

 Mode register 0
IE2 ライトペンによる割り込み
 常に0を指定する

R#8

MS	LP	TP	CB	VR	0	SPD	BW
----	----	----	----	----	---	-----	----

 Mode register 2
MS マウス 常に0を指定する
LP ライトペン 常に0を指定する

S#1

FL	LPS	ID#		FH
----	-----	-----	--	----

 Status register 1
FL ライトペンスイッチ・マウススイッチ2 意味を持たない
LPS ライトペンスイッチ・マウススイッチ1 意味を持たない

2.7.2 コントロールレジスタ R#9 の IL, EO

コントロールレジスタ R#9 の IL, EO (bit3, bit2) の組合せによって2画面交互表示 & インターレース表示を行います。EO を1にしてこれらの機能を利用する時は、パターンネームテーブルを奇数ページに設定して下さい(画面モードが GRAPHIC4~7, YJK/RGB 混在, YJK の時のみ有効)。

表 2.7.1 IL・EO の機能

IL	EO	画面モード	動作
0	0	テキスト	ノンインターレース・同一画面
		グラフィック	ノンインターレース・同一画面
0	1	テキスト	ノンインターレース・同一画面
		グラフィック	ノンインターレース・1/60秒周期で2画面交互表示
1	0	テキスト	インターレース・第1フィールドと第2フィールドに同一画面
		グラフィック	インターレース・第1フィールドと第2フィールドに同一画面
1	1	テキスト	インターレース・第1フィールドと第2フィールドに同一画面
		グラフィック	インターレース・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

テキスト : TEXT1・2, GRAPHIC1~3, MULTI COLOR

グラフィック : GRAPHIC4~7, YJK/RGB 混在, YJK

TEXT1・2, GRAPHIC1~3, MULTI COLOR で 1/60 秒周期の2画面交互表示や、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するインターレース表示を行うには、垂直帰線割り込み内で EO(S#2 の bit1) を参照しながらページ切り替えを行って下さい。

2.7.3 水平帰線割り込み

V9938・V9958 は、IE1 (R#0 の bit4) を 1 にセットすると水平帰線割り込みを発生します。この時、R#19 に割り込みを発生させる走査線番号を設定出来ます。走査線番号とは、表示される VRAM のラインに対応する番号で、0～255 の値を取ります。R#23=0 の時、画面最上部の走査線番号が 0 となり、以下 1, 2, 3...と続いています。R#23 によるハードウェア縦スクロールを行うと、割り込みを発生させる走査線番号もずれるので、割り込み位置を変化させない場合には R#19 にスクロール量を加算する等の補正を行って下さい。

R#0

0	-	IE1	-	0
---	---	-----	---	---

 Mode register 0
IE1 水平帰線による割り込みを 1=使用する、0=使用しない

R#19

IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0
-----	-----	-----	-----	-----	-----	-----	-----

 Interrupt line register
IL7～IL0 割り込みを発生させる走査線番号
実際に水平帰線割り込みが発生するのは、192 ラインモードでは走査線番号 0～234 (212 ラインモードでは 0～244) のみであり、それ以外の値を指定しても水平帰線割り込みは発生しない。

IE1 に 1 がセットされている場合、V9938, V9958 は R#19 に設定された走査線を描き終わると、割り込みを発生し、FH (S#1 の bit0) を 1 にします。

割り込みが発生すると、CPU が割り込みを禁止していなければ、割り込みベクタ 0038H が割り込みを禁止されてコールされます。通常は、0038H からシステムの割り込みルーチンにジャンプしますが、0038H を直接書き換えて直接アプリケーションの割り込みルーチンにジャンプする事も可能です。システムの割り込みルーチンを利用する場合は、【FD9AH(H.KEYI: 全ての割り込みが通る)】からアプリケーションの割り込みルーチンにジャンプして下さい。

割り込みルーチン内では、まず FH (S#1 の bit0) の値を調べ、0 であればそのままリターンします。FH が 1 であれば、水平帰線割り込みであると判断し割り込み処理を行います。

なお FH を調べた後、システムの割り込みルーチンに実行を渡す場合は R#15 に 0 を設定してからリターンして下さい。

S#1

FL	LPS	ID#	FH
----	-----	-----	----

 Status register 1
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 (R#0 の bit4 が 1 の時) が発生すると 1 になる。
S#1 を読み出すと 0 になる。

R#19 を割り込みルーチンの中で再設定する事で、1/60 秒間に複数回の水平帰線割り込みをかけることが可能です。

2.7.4 PAL 方式

海外製 MSX で使われている PAL 方式について説明します。

PAL 方式とは、テレビ放送の方式の一つで、オランダ等の国々で使われています。日本では NTSC と呼ばれる方式が使われており、画面表示の周波数が異なります。

画面表示の周波数は、NTSC では 60Hz、PAL では 50Hz となっており、垂直帰線割り込みの周波数に同等です。1 画面あたりの水平走査は NTSC で 262 回、PAL で 313 回です。

MSX1 では、NTSC に TMS9918A (コンポジットビデオ信号出力) / TMS9928A (色差信号出力)、PAL に TMS9929A (色差信号出力) と、両者で異なる VDP (出力信号以外は同一の機能を持つ) を使用していますが、V9938, V9958 では NT (R#9 の bit1) を指定する事で PAL と NTSC を切り替える事が出来ます。

2.7.5 画面モードによる VRAM 構成の変化

V9938, V9958 における画面モードによる VRAM 構成の変化について説明します。

GRAPHIC5 (SCREEN 6) 以下の画面モードと GRAPHIC6 (SCREEN 7) 以上の画面モードでは VRAM 構成が異なります。GRAPHIC6 以上の 00000H ~ 0FFFFH (ページ 0) は、GRAPHIC5 以下での 00000H ~ 07FFFH, 10000H ~ 17FFFH (ページ 0, 2) に、GRAPHIC6 以上の 10000H ~ 1FFFFH (ページ 1) は GRAPHIC5 以下での 08000H ~ 0FFFFH, 18000H ~ 1FFFFH (ページ 1, 3) に対応しています (表 2.7.2 を参照)。

表 2.7.2 画面モードと VRAM アドレスの対応

GRAPHIC6以降	GRAPHIC5以下
00000H	00000H
00001H	10000H
00002H	00001H
00003H	10001H
00004H	00002H
00005H	10002H
00006H	00003H
⋮	⋮
0FFFC H	07FFE H
0FFFD H	17FFE H
0FFFE H	07FFF H
0FFFF H	17FFF H
10001H	08000H
10002H	18000H
10003H	08001H
⋮	⋮
1FFFE H	0FFFF H
1FFFF H	1FFFF H

2.7.6 表示タイミング

VDP の表示タイミングを説明します。

画面の表示は、まず画面上端の 1 ラインを左端から右端に向かって表示する事から始まります (水平走査)。1 ライン表示し終わると、その下のラインを同じ様に左端から表示します。これを画面下まで繰り返し、下端まで到達したら再び上端に戻ります (垂直走査)。それぞれの周期を水平同期期間、垂直同期期間と言います。垂直同期期間は、日本等で使われている NTSC 方式で 1/60 秒、海外で使われている PAL 方式で 1/50 秒です (詳細は 2.7.4 PAL 方式を参照)。

各同期期間は、帰線期間(1)、表示期間、帰線期間(2)、同期信号期間に分かれます (図 2.7.1)。帰線期間には画面周辺部を表示している期間を含みます。また、2 つあるのは表示期間の前後にあるからです。

図 2.7.1 水平・垂直同期期間

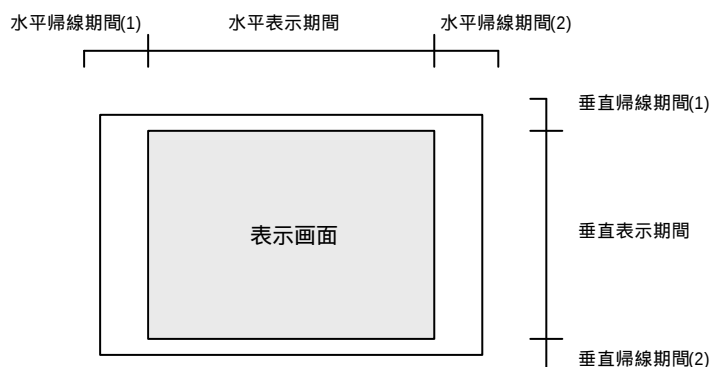


表 2.7.3 に NTSC 方式の場合の各期間の長さを示します (PAL 方式でも水平帰線に関するデータは同じです)。ただし、表示位置補正やインターレース走査によってタイミングが変わりますので注意して下さい。

表 2.7.3 TMS9918A ディスプレイパラメータ

画面モード	水平同期期間		垂直同期期間
	GRAOHIC1~2 / MULTI COLOR	TEXT1	
同期信号期間	26	26	3
帰線期間(1)	37	43	40
表示期間	256	240	192
帰線期間(2)	23	33	27
同期期間	342	342	262

単位 水平 ピクセルクロックサイクル = $1 / (10.74\text{MHz} / 2) = 0.1862 \mu\text{秒}$
 垂直 ピクセルクロックサイクル $\times 342 = 63.69 \mu\text{秒}$

表 2.7.4 V9938, V9958 水平ディスプレイパラメータ (NTSC/PAL 共通)

画面モード	GRAOHIC1~7 / MULTI COLOR / YJK・RGB 混在 / YJK		TEXT1~2	
	01B / 10B	00B	01B / 10B	00B
同期信号期間	100	100	100	100
帰線期間(1)	158	158	194	194
表示期間	1024	1024	960	960
帰線期間(2)	83	86	111	114
同期期間	1365	1368	1365	1368

単位 ピクセルクロックサイクル = $1 / (21.477\text{MHz} / 2) = 0.04656 \mu\text{秒}$

表 2.7.5 V9938, V9958 垂直ディスプレイパラメータ (NTSC)

ライン数	192Line (LN=0)			212Line (LN=1)		
	Non-Interlace	Interlace		Non-Interlace	Interlace	
Field		1st	2nd		1st	2nd
同期信号期間	3	3	3	3	3	3
帰線期間(1)	39	39	39.5	29	29	29.5
表示期間	192	192	192	212	212	212
帰線期間(2)	28	28.5	28	18	18.5	18
同期期間	262	262.5	262.5	262	262.5	262.5

単位 ピクセルクロックサイクル $\times 1368 = 63.69 \mu\text{秒}$

表 2.7.6 V9938, V9958 垂直ディスプレイパラメータ (PAL)

ライン数	192Line (LN=0)			212Line (LN=1)		
	Non-Interlace	Interlace		Non-Interlace	Interlace	
Field		1st	2nd		1st	2nd
同期信号期間	3	3	3	3	3	3
帰線期間(1)	66	66	66.5	56	56	56.5
表示期間	192	192	192	212	212	212
帰線期間(2)	52	51.5	51	42	41.5	41
同期期間	313	312.5	312.5	313	312.5	312.5

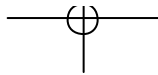
単位 ピクセルクロックサイクル $\times 1368 = 63.69 \mu\text{秒}$

2.7.7 レジスタ更新の反映タイミング

VDP の処理は、即時的に行われるものと、水平走査毎に行われるもの、垂直走査毎に行われるものの 3 種類が存在し、レジスタ、VRAM の変更が画面に反映されるタイミングが異なります。

表 2.7.7 レジスタ更新の反映タイミング

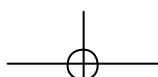
即時的	パターンジェネレータテーブル、パターンネームテーブル、カラーテーブルの各ベースアドレスの切り替え、ハードウェアスクロール (縦・横)、カラーレジスタ、パレットレジスタの書き換え、画面モードの切り替え
水平走査毎	スプライトパターンジェネレータテーブル、スプライトアトリビュートテーブルの各ベースアドレスの切り替え、スプライトパターンジェネレータテーブル、スプライトアトリビュートテーブルの各テーブルの書き換え、アジャストレジスタ (横)、画面表示の禁止・許可
垂直走査毎	アジャストレジスタ (縦)

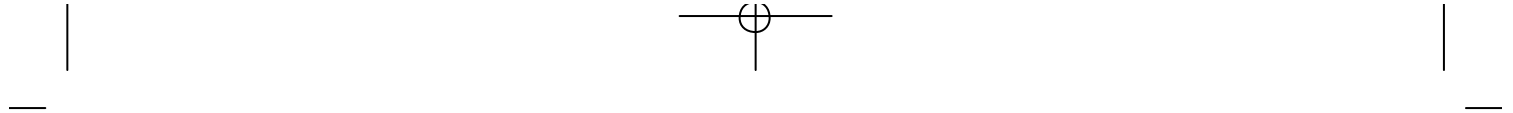


即時的に行われる処理では、レジスタを変更した直後に、その変更が画面に反映されます。

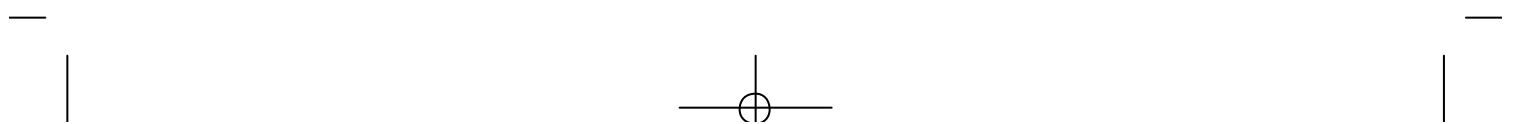
水平走査毎に行われる処理では、水平走査中にレジスタ、VRAM の書き換えが発生した場合でもデータの変更が無いものとして走査線を書き終え、次の走査線から変更が画面に反映されます。ただし、スプライトの表示において変更が表示に反映されるのは、テーブルを変更した時の走査線の次の走査線ではなく、更に 1 ライン下の走査線からとなります。

垂直走査毎に行われる処理では、垂直走査中にレジスタの書き換えが行われてもデータの変更が無いものとしてフレームを書き終え、次のフレームから表示に反映されます。





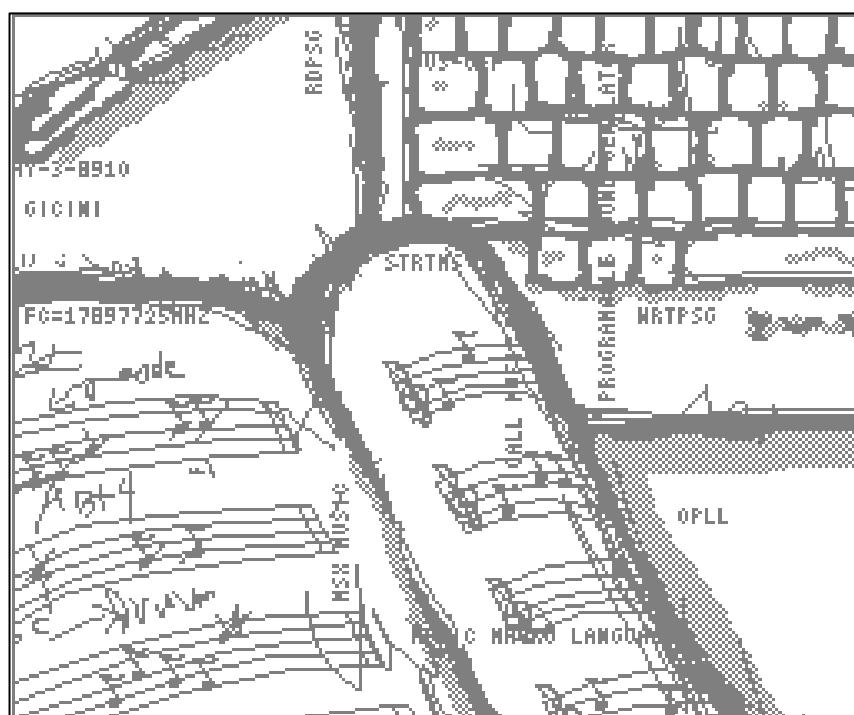
第
2
部

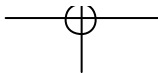


第 3 部

音源

第 3 部





1章 PSG



3.1.1 PSG の概要

PSG (Programmable Sound Generator) は MSX に一番最初から搭載されていた音源で、MSX であればどの機種にも搭載されている、最も基本的な音源です。

基本的な仕様として、

- ・矩形波を 3 チャンネル出力可。それぞれのチャンネルに別の周波数 (音程)、音量を与えることができる。
- ・ノイズ発生器を内蔵し、任意のチャンネルから任意の周波数でノイズを発生させることができる。
但し、発生器自体は 3 チャンネルで共通。
- ・エンベロープ発生器を内蔵。任意のチャンネルに対し、簡単な時間的音量変化を与えることができる。
但し、エンベロープ発生器は 3 チャンネルで共通。
- ・独立した 2 つの I/O ポート (8 ビット) を持つ。

ということが挙げられます。

音源自体は、GI 社の AY-3-8910 相当品が使用されています。後期の MSX では、カスタムチップの中に他の IC 等と一緒にまとめられている場合もあります。但し、互換チップごとに若干出てくる音が違うことがあるようです (エンベロープの周波数など)。

3.1.2 PSG の操作方法

ここでは、PSG のプログラムからの操作方法を紹介します。

まず、PSG を操作するための BIOS について以下に示します。

● GICINI (0090H/MAIN) MSX1

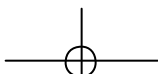
機能	PSG の初期化
入力	なし
出力	なし
変更	すべて
説明	PSG を初期化し、PLAY 文のためのワークエリア等を設定します。

● WRTPSG (0093H/MAIN) MSX1

機能	PSG のレジスタの書き込み
入力	A レジスタ番号 E 書き込むデータ
機能	なし
変更	なし
説明	PSG のレジスタにデータを書き込みます。

● RDPSPG (0096H/MAIN) MSX1

機能	PSG のレジスタの読み込み
入力	A レジスタ番号
出力	A 読み込んだ値
変更	なし
説明	PSG のレジスタからデータを読み込みます。



しかし、これらの BIOS を使用すると、コールから帰ってくる際に割り込みが許可された状態となっています。

BIOS を利用したくない場合には、直接 I/O をアクセスして、PSG を操作することになります。なお、PSG の I/O アドレスは turboR まで変更された形跡はありませんので、安心してアクセスすることができます。

PSG は、MSX の I/O アドレス上で、

0A0H	アドレスラッチ
0A1H	データライト
0A2H	データリード

に存在しています。基本的には、0A0H に読み書きするアドレスを書き込んだ後、0A1H に書き込む / 0A2H から読み込むことになります。

3.1.3 PSG の内部レジスタ

ここでは、PSG をコントロールする際にソフト側から書き込むことになる、PSG の内部レジスタについて記載します。

PSG には 16 本のレジスタがあり、それぞれ 3.1.2 PSG の操作方法で示した方法によって読み込み・書き込みを行うことができます。

各レジスタの詳細を以下に示します。

- が書き込まれているビットは未使用です。0 で埋めておいてください。

レジスタ#0～#5 チャンネルごとの周波数設定

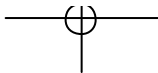
#0	F-num A		チャンネル A	周波数 L
#1	-	F-num A	"	周波数 H
#2	F-num B		チャンネル B	周波数 L
#3	-	F-num B	"	周波数 H
#4	F-num C		チャンネル C	周波数 L
#5	-	F-num C	"	周波数 H

F-num 発声する周波数から求められる音程データを書き込み、各チャンネルごとの音程を設定します。
音程データは 12 ビットで表され、下位 8 ビットをレジスタ 0, 2, 4 に、上位 4 ビットをレジスタ 1, 3, 5 に、それぞれ書き込みます。

このとき、音程データ F-num と発声する音程周波数 Freq (Hz)の間には、

$$F\text{-num} = f_c / (16 \times \text{Freq}) = 111860.78125 / \text{Freq}$$

という関係が成り立っています。ここで f_c は PSG に入力される基準周波数で、 $f_c = 1.7897725$ (MHz)です。



各音程と F-num の関係を以下に示します。

Octave	C	C+	D	D+	E	F	F+	G	G+	A	A+	B
1	D5D	C9C	BE7	B3C	A9B	A02	973	8EB	86B	7F1	780	714
2	6AF	64E	5F4	59E	54E	501	4BA	476	436	3F8	3C0	38A
3	357	327	2FA	2CF	2A7	281	25D	23B	21B	1FD	1E0	1C5
4	1AC	194	17D	168	153	140	12E	11D	10D	FE	F0	E3
5	D6	CA	BE	B4	AA	A0	97	8F	87	7F	78	71
6	6B	65	5F	5A	55	50	4C	47	43	3F	3C	39
7	35	32	30	2D	2A	28	26	24	22	1F	1E	1C
8	1B	19	18	16	15	14	13	12	11	10	F	E

レジスタ#6 ノイズ周波数設定

#6

-	N F-num
---	---------

 ノイズ周波数

N F-num ノイズを発声する場合のノイズ周波数を設定します。値は0～31 で bit0～bit4 で指定します。
基本的に音程というものの存在しないノイズですが、このノイズ周波数というのは、ノイズの「平均」周波数のことです。

ノイズ周波数 Noise_freq(Hz)と周波数データ N_F-num との間には、

$$N_F\text{-num} = 111860.78125 / \text{Noise_freq}$$

との関係が成り立っており、1 N_F-num 31 より、3.6kHz Noise_freq 111.9KHz となります。

具体的には、N_F-num の値が小さくなると「サー」といった感じのノイズになり、値が大きくなるにつれ、「ザー」から「ゴー」といった感じの音になります。

なお、ノイズ発声器は A・B・C 各チャンネルで共通ですので、チャンネルごとにノイズの周波数を変えることは出来ません。

レジスタ#7 ミキシング・I/O ポート設定

#7

IOB	IOA	NC	NB	NA	TC	TB	TA
-----	-----	----	----	----	----	----	----

 ミキシング設定

各チャンネルが通常の矩形波の音を出すか、それともノイズを出すか、あるいは両方とも出すか、といったミキシングの設定と、内蔵 I/O ポートのそれぞれの入出力の方向の設定を行います。

NA, NB, NC それぞれのチャンネルのノイズ出力の設定 0=出力、1=発声しない

TA, TB, TC それぞれのチャンネルの通常の矩形波トーン出力の設定 0=出力、1=発声しない
Nn, Tn 両方とも 0 にすることで、ノイズとトーンを同時に発声することも出来ます。
また、両方とも 1 にすると、発声しくなくなります。

IOA, IOB それぞれ I/O ポートの入出力の設定 0=入力、1=出力

MSXでは通常、ポートA=入力 ポートB=出力 として使用しているので、このレジスタに値を書き込む際には常に、bit6を0、bit7を1とすることになります。

レジスタ#8～#10 音量設定

#8

-	M	Volume A
---	---	----------

 チャンネル A 音量

#9

-	M	Volume B
---	---	----------

 チャンネル B 音量

#10

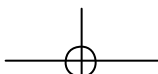
-	M	Volume C
---	---	----------

 チャンネル C 音量

各チャンネルが発声する際の音量を設定します。

Volume 値は0～15で、0のとき無音、15のとき最大となります。

M 1にすると、そのチャンネルではエンベロープを指定したことになり、レジスタ#11以降で指定するエンベロープ形状に従って、音量が時間的に変化します。但し、このとき Volume (bit0～bit3) は無視されます。
すなわち、エンベロープに対しては音量を設定出来ず、エンベロープ形状は常に最大音量(15)に対する変位になっています。エンベロープに対しても音量変化を付けたい場合は、エンベロープを使用せず、ソフトウェア的に音量の時間変化を行う(ソフトエンベロープ)といったことが必要になります。



レジスタ#11～#12 エンベロープ周期の設定

#11	Envelope_freq L	エンベロープ周期 L
-----	-----------------	------------

#12	Envelope_freq H	エンベロープ周期 H
-----	-----------------	------------

Envelope_freq

エンベロープの変化する周期を設定します。値は 16 ビットで指定され、それぞれ下位 8 ビットをレジスタ 11 で、上位 8 ビットをレジスタ 12 に書き込みます。

変化周期 T (μs) と設定する値 Envelope_freq には、

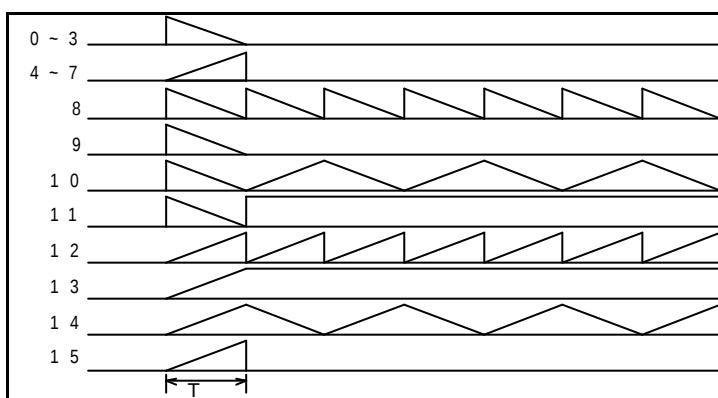
$$\text{Envelope_freq} = T / 143.03493$$

という関係がありますが、実際には使用されている PSG 互換チップの種類にもよるようです。

レジスタ#13 エンベロープ形状の設定

#13	-	Envelope_form	エンベロープ形状
-----	---	---------------	----------

エンベロープの形状を指定します。形状と値との関係は以下の通りです。



このレジスタに書き込んだ時点でエンベロープはリセットされ、エンベロープを使用している全チャンネルとも、波形の頭からもう一度変位を始めます。

なお、エンベロープ周期・形状とも、全チャンネルで共通です。

レジスタ#14～#15 I/O ポート

#14	I/O _ port A	I/O ポート A
-----	--------------	-----------

#15	I/O _ port B	I/O ポート B
-----	--------------	-----------

それぞれ、8 ビット幅の I/O ポートです。入出力の方向はレジスタ#7 のビット 6, ビット 7 で指定されますが、MSX では常に A を入力・B を出力として使用しています。

参考までに、具体的な内容を以下に示します。

#14	Cin	KEY	J7	J6	J4	J3	J2	J1
#15	KANA	PSEL	J8-2	J8-1	J7-2	J6-2	J7-1	J6-1

Cin カセットテープインタフェースからのデータ入力
 KEY キーボード配列 (0=50音 1=JIS)
 J1~J7 ジョイスティックインタフェースの1~7番ピンに接続
 KANA かなモード表示ランプ (0=ON 1=OFF)
 PSEL J1~J7を 0=ジョイスティックインタフェース1につなぐ 1=ジョイスティック2につなぐ
 J8-2 ジョイスティックインタフェース2の8番ピンに接続
 J8-1 ジョイスティックインタフェース1の8番ピンに接続
 J7-2 ジョイスティックインタフェース2の7番ピンに接続
 J6-2 ジョイスティックインタフェース2の6番ピンに接続
 J7-1 ジョイスティックインタフェース1の7番ピンに接続
 J6-1 ジョイスティックインタフェース1の6番ピンに接続

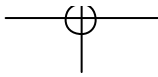
3.1.4 PSG を使用する上での諸注意

レジスタ#13の項でも書いた通り、エンベロープの発生器は3チャンネルで共通です。また、エンベロープ全体の音量も設定できないため、通常の楽曲を演奏する上でエンベロープ機能自体、非常に使い辛いものとなっています。

このため、時間的に音量を変化させソフトウェア的にエンベロープの波形を作るという、ソフトエンベロープという手法が一般的には用いられています。

PSGに限りませんが、2チャンネルを使用し全く同じ周波数の音を出させると、かなり潰れたような、あまり気持ち良くない音となって出てくることがあります。こういった場合は、片方のチャンネルから出る周波数を若干（レジスタの値で2、3程度）上下にずらしてやると、艶としたかなり気持ち良い音になります。これは一般的にデチューンと呼ばれる技で、意図的に使用される場合も多々あります。

PSGを読み書きする際、ウェイトは特に必要ないようです。これはturboRでも同様です（当然I/Oアクセスに対するウェイトが掛かっていますが）。無論、本体をクロックアップしている場合は保証の限りではありません。また、使用に当たって特に初期化のデータ等を書き込むということも必要ないようです。



2章 OPLL

3.2.1 FM 音源 (OPLL) の概要

FM 音源・OPLL (YM-2413) は、高機能すぎて価格の上昇を招いた「MSX-AUDIO」を反省し (と思われる)、そこから幾つかの機能を省略して価格を抑えた規格「MSX-MUSIC」で使用されたヤマハ社の音源チップです。最初は Panasonic 社の「FM パック」という音源カートリッジにのみ搭載されていましたが、価格の安さと対応ソフトの豊富さにより一気に普及し、その後 MSX2+ で規格上にオプションとして設定され、turboR 以降では必須の機能となりました。

基本的な仕様として、

- ・ 2 オペレータの FM 方式の音源発音ポートを 9 つ搭載。最大 9 音 (ドラム音使用時 6 音) を発音可能
- ・ ドラム音専用発音ポートを搭載。但しドラム使用時は FM の発音数は 6 音
- ・ 15 種類の音色を ROM で搭載。面倒な設定無しに音色を使用することが可能。
オリジナルの音色も 1 つだけ設定出来る

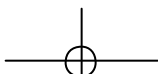
ということが挙げられます。

3.2.2 FM 音源 (OPLL) の操作方法

MSX の正式な規格ということで、OPLL のコントロールにはちゃんと BIOS が用意されています。但し、使用するためには MSX-MUSIC の BIOS が接続されているスロットを探し、そのスロット上の特定のアドレスを呼び出すという作業が必要となります。

MSX-MUSIC が接続されているスロットには、アドレス 401CH からの 4 バイトに認識のために「OPLL」という文字列が書き込まれています。MSX-MUSIC が存在するかどうかのチェックは、この文字列が任意のスロットのアドレスに存在するかどうかのチェックになります。ここで発見されたスロットのページ 1 には、MSX-MUSIC の BIOS, BASIC の拡張コマンド、音色データなどが書き込まれています。

ちなみに、turboR では各スロットの状態も規格化され、MSX-MUSIC は必ずスロット 0-2 に存在することになっています。よって、turboR で運用されることが決まっているアプリケーションでは MSX-MUSIC の存在チェックが省略できます。



MSX-MUSIC に装備されている BIOS (FM-BIOS) を以下に示します。

● WRTOPL (4110H/FM)

機能 OPLL のレジスタの書き込み
 入力 A レジスタ番号
 E 書き込むデータ
 出力 なし
 変更 なし
 説明 OPLL の指定レジスタにデータを書き込みます。

● INIOPL (4113H/FM)

機能 MSX-MUSIC と OPLL の初期化
 入力 HL MSX-MUSIC 用のワークエリアの先頭アドレス (偶数)
 出力 なし
 変更 AF BC DE HL IX IY
 説明 MSX-MUSIC、および OPLL のレジスタを初期化します。
 EI 状態で返ってきます。
 ページ 1 のスロットは MSX-MUSIC が存在するスロットに変更されます。
 HL で指定されるアドレス以降 0A0H バイトを MSX-MUSIC 用のワークエリアとして確保し、そのワークエリアおよび OPLL のレジスタを初期化します。HL の内容は【SLTWRK (FD09h)】の自分が存在するスロット用のワークに保存されます。但し、HL が指すアドレスをページ 1 に置くことはできません。また、ページ 0 であった場合も機種によっては不都合が起こる場合があります。
 全ての MSX-MUSIC の BIOS は事前にこのエントリを呼び出す必要があります。

● MSTART (4116H/FM)

機能 演奏の開始
 入力 HL ミュージックデータの先頭アドレス
 A エンドレスフラグ
 0 無限ループ
 1-254 指定回数だけループする
 255 動作保証なし
 (HL)~ ミュージックデータ (詳細は後述)
 出力 なし
 変更 AF BC DE HL IX IY
 説明 HL で指定するアドレス以降に格納されたミュージックデータを認識し、そのデータに含まれる情報 (ヘッダ部分に存在) を元に FM-BIOS のワークエリアを初期化し、音楽の演奏が可能な状態にします。実際の演奏は、コール OPLDRV が一定間隔 (【H.TIMI】など) で呼び出されることで行われます。このコールでは演奏自体は行われません。
 EI 状態でリターンします。

● MSTOP (4119H/FM)

機能 音楽演奏の中止
 入力 なし
 出力 なし
 変更 AF BC DE HL IX IY
 説明 現在出力している OPLL の全チャンネルの音を止め、ワークエリアを初期化し演奏を停止します。
 EI 状態でリターンします。

● RDDATA (411CH/FM)

機能	ROM 内の音色データの読みだし
入力	HL データ転送用ワークの先頭アドレス A 音色番号 (0~63)
出力	なし
変更	F
説明	ROM に格納されている音色データを読みだし、HL で指定されるユーザーのワークエリアに転送します。データは1音色につき8バイトで、先頭からそのまま OPLL のレジスタ#0~#7 に転送されるデータが書かれています。 なお、このコールで得られるデータと、BASIC で使用されている音色データは若干異なっています。

● OPLDRV (411FH/FM)

機能	OPLL ドライバのインタラプトエントリ
入力	なし
出力	なし
変更	なし
説明	【H_TIMI(0FD9FH)】等から呼ばれる、ミュージックデータを実際に演奏するためのコールです。必ず、INIOPL で OPLL およびワークを初期化し MSTART で演奏データを設定した後呼ぶようにしてください。

● TSTBGM (4122H/FM)

機能	演奏の終了チェック
入力	なし
出力	A 0 演奏終了 0 以外 演奏中
変更	AF
説明	MSTART で開始したミュージックデータの演奏が終了したかどうかをチェックします。

FM-BIOS で演奏が可能なミュージックデータの形式を以下に示します。

・ヘッダ部分

データの先頭 12H バイト (9 音モード、以下 9 音)、0EH バイト (リズム音使用モード、以下リズム) はヘッダ部分です。そのミュージックデータに関する情報 (各チャンネルデータのアドレス) が格納されています。

各データのアドレスはミュージックデータそのものの先頭からのオフセット (下位・上位で格納されている) です。データは9チャンネル (9 音) / リズムチャンネル + 6 チャンネル = 7 チャンネル (リズム) 分、すなわち 12H バイト (9 音) / 0EH バイト (リズム) 存在します。このとき、データの先頭 1 チャンネル目はヘッダの直後に存在しますから、それぞれの先頭アドレスは 0012H (9 音) / 000EH (リズム) バイト目から存在し、ヘッダ部分にもそのように書かれます。すなわち、ヘッダ部分の先頭には、9 音モードの場合 12H が、リズムモードの場合 0EH が書き込まれていることになります。

・データ部分

ヘッダで指定される各チャンネル（リズムを除く）のデータ領域には、以下の形式で指定されるデータが並びます。

00H 休符
休符です。後ろに音長データが続きます。

01H ~ 5FH 音程指定
指定した番号を音階とし、発声します。01H をオクターブ1のド、02H をド#、以下 5FH まで半音ずつ上がっていきます。

後ろに休符と同じく音長が続きます。単位は 1/60 秒（H_TIMI で OPLDRV を呼び出した場合）です。但し、0FFH だった場合は更にその次のバイトも音長データとし、0FFH 以外になるまで音長として認識します。つまり、25H 0FFH 32H となる場合は、オクターブ4のドの音を（0FFH + 32H）= 131H カウント分発音を続けます。

60H ~ 6FH 音量指定
以下このチャンネルで発音する音の音量をそれぞれ 0 ~ 15 にします。

70H ~ 7FH 音色指定
以下このチャンネルで発音する音の音色を設定します。音色番号をそれぞれ 0 ~ 15 としますが、対応する音色は後述のレジスタマップの項で指定するものと同じです。0 で指定されるユーザー拡張音色は以下の 82H・83H で設定することが出来ます。

80H サスティン解除
81H サスティン設定

82H 拡張音色指定
ROM 内に格納されている拡張音色を OPLL のレジスタ 0 ~ 7 に設定し、以後音色指定の 0 番（70H）として使用出来るようにします。後続の 1 バイトで音色番号（0 ~ 63）を指定します。

83H ユーザー音色設定
ユーザー側から ROM の音色データ群に存在しないような拡張音色を設定します。続く 2 バイトをアドレスの下位・上位とし、そのアドレスから 8 バイトを音色データとして設定します。

84H レガートオフ
音を切らずに続けます。1 音ごとにキーオンをしなくなります。

85H レガートオン
1 音ごとにキーオンを行います。

86H Q 指定
音長に対し、実際に発音する長さを設定します。1 ~ 8 で、8 のとき音長と発音する時間が一致します。レガートオンの場合は効果はありません。

87H ~ FEH 現在未使用

FFH 終了フラグ
そのチャンネルのデータが終了したことを示します。

なお、リズムパートでは以上とは異なるミュージックデータが使用されます。

V	0	1	B	S	T	C	H
---	---	---	---	---	---	---	---

この1バイトのデータ後ろに音長、あるいはデータが続きます。

B, S, T, C, Hはそれぞれバスドラム、スネアドラム、タム、シンバル、ハイハットの発音のON/OFFを表し、対応するビットを1にすることでその楽器が選択されます。

最上位ビット(V)はボリュームフラグです。このビットが1の場合、下位7ビットは無視され、後続の1バイトが音量(全楽器)を表します。音量の値は0~15です。

同ビットが0の場合、ビット0~4で指定される楽器の発音を行い、後続で指定される音長だけ待ちます。なお、音長の形式は上記のメロディ部のものと同じです。

終了フラグは0FFHです。

以上のルーチンはMSX-MUSICが接続されているスロット上に存在するので、コールするにはインタースロットコール、あるいはスロットを直接表に出してコールするということになります。時間当たりに割と大量に使用するこの手のルーチンに対してこれでは時間がかかりすぎるというわけで、次に直接OPLLを操作する方法を示します。

I/Oから直接OPLLを使用する場合、MSX本体にOPLLが内蔵されている機種とそうでない機種で若干準備が異なります。まず内蔵とそうでない場合の区別ですが、上記のMSX-MUSICが接続されているスロットの4018Hから4バイトの内容が「APRL」であれば(要するに、4018Hから8バイトが「APRLOPLL」であれば)OPLLは内蔵されていることになり、そのままI/Oを操作してOPLLを使用することができます。外付けの場合には、MSX-MUSICが接続されたスロットの7FF6H(読み書き可)のビット0を1にすることで、外付けのOPLLが使用できるようになります。この操作が面倒臭い場合は、INIOPLをコールしてやれば以上の操作をちゃんとやってくれます。

OPLLは、MSXのI/Oアドレス上の、

7CH	アドレス指定
7DH	データ

に存在しています。書き込みのみで読み込むことは出来ません。実際には、7CHで書き込むアドレスを指定し、その後7DHに実際のデータを書き込みます。7DHにデータを書き込んだ後もアドレスは変化しないので、同じアドレスに連続してデータを書き込む場合は7CHへの書き込みを省略出来ます。但し、それぞれのI/Oアドレスに書き込む際には一定のウェイトが必要で、7CHに対しては3.36μ秒、7DHに対しては23.52μ秒の間隔を必ずおいてから書き込む必要があります(ウェイトが足りないとOPLLが誤動作します)。なお、外付けのOPLLでは7CH・7DH以外にメモリマップドI/Oが存在します。位置はMSX-MUSICが接続されたスロットの7FF4H(アドレス指定)・7FF5H(データ書き込み)で、これにより複数の外付けOPLLを独立に制御できます(内蔵OPLLは不可)

BIOS経由で書き込んだ場合にはきちんとこのウェイトを守ってくれますが、時間の掛かる命令などで本当に律儀に待ってくれますので、実際に使用する場合には必要以上のウェイトとなる場合があります。

3.2.3 FM音源(OPLL)の内部レジスタ

ここでは、OPLLをコントロールする際にソフト側から書き込むことになる、OPLLの内部レジスタにつ

いて記載します。

OPLL には 47 (25H) 本のレジスタがあり、それぞれ 3.2.2 FM 音源 (OPLL) の操作方法で示した方法によって書き込みを行うことができます。

その一覧を以下に示します。レジスタの番号は 16 進数です。

00	AM	VIB	EGT	KSR	Multiple			音色設定 (M)
01	AM	VIB	EGT	KSR	Multiple			音色設定 (C)
02	KSR(M)		Total_level					音色設定
03	KSR(C)			DC	DM	Feedback		音色設定
04	Attack			Decay				音色設定 (M)
05	Attack			Decay				音色設定 (C)
06	Sustain			Release				音色設定 (M)
07	Sustain			Release				音色設定 (C)
08 ~ 0D	未使用							未使用
0E	-	R	BD	SD	TOM	TCY	HH	ドラムス指定
0F	?	?	?	?	?	?	?	検査用レジスタ
10 ~ 18	F_number							発音周波数設定 L
19 ~ 1F	-							未使用
20 ~ 28	-	SUS	KEY	BLOCK		F_n		各種データ指定
29 ~ 2F	-							未使用
30 ~ 38	Instruments			Volume				音色・音量指定

表中、(C)(M) はそれぞれキャリア、モジュレータについての値であることを示します。
- が書き込まれているビットは未使用です。0 で埋めておいてください。

レジスタ 00 ~ 07H は音色設定の部分です。これらのレジスタについては 3.2.4 音色設定で詳しく説明します。

それ以降のレジスタの詳細を以下に示します。

08H~0DH	-	未使用
19H~1FH	-	未使用
29H~2FH	-	未使用

未使用です。アドレス計算をやりやすくするために、意図的にアドレスの飛びを作っているものと思われます。

0EH	-	R	BD	SD	TOM	TCY	HH	ドラムス指定
-----	---	---	----	----	-----	-----	----	--------

リズムモードの設定と各ドラム音の発音の設定です。

ビット 5 を 1 にすると、OPLL はリズムモード (FM 6 音・ドラム 5 音) になります。その上で、ビット 0 ~ ビット 4 の任意のビットを 1 にすることで、各ドラム音が発音されます (0 から 1 に変化した時点でキーオンされます)。1 から 0 への変化でキーオフです。このとき、レジスタ 26 ~ 28 の KEY は常に 0 にしておいてください。

各ビットとドラム音の対応は以下の通りです。

bit0 (HH)	ハイハットシンバル (Hi_Hat cymbal)
bit1 (TCY)	トップシンバル (Top Cymbal)
bit2 (TOM)	トムトム (Tom Tom)
bit3 (SD)	スネアドラム (Snare Drum)
bit4 (BD)	バスドラム (Bass Drum)

0FH

?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---

 検査用レジスタ

検査時に使用される？レジスタです。0 以外では正常に動作しません。

10H ~ 18H

F_number			
----------	--	--	--

 発音周波数設定 L

20H ~ 28H

-	x	x	BLOCK	F_n
---	---	---	-------	-----

 発音周波数設定 H

発声する周波数から求められる音程データを書き込み、各チャンネルごとの音程を設定します。

音程データは、オクターブとオクターブ内の音程データ（12 個）で指定されます。すなわち、オクターブが違っていても同じ音名（ドレミ）であれば、F_number は等しくなります。

レジスタ 10H ~ 18H および 20H ~ 28H のビット 0（最上位ビットになる）の計 9 ビットで F_number を指定し、同 20H ~ 28H のビット 1 ~ 3 の 3 ビットで BLOCK（オクターブ）が指定されます。

実際に発声する音程周波数 F_mus（Hz）と音程データの関係は、

$$F_number = (F_mus \times 2^{18} / 50000) / 2^{BLOCK}$$

$$BLOCK = \text{オクターブ} - 1 \quad (1 \text{ オクターブ } 8)$$

で表され、たとえばオクターブ 4 の「ラ」の音（440Hz）であれば以下のようになります。

$$F_number = (440 \times 2^{18} / 50000) / 2^{(4-1)} = 288$$

$$BLOCK = 4 - 1$$

音程データと各音名の関係は以下の通りです。

音程	10H+ch.	20H+ch.
C	172	$32 \times (SUS) + 16 \times (KEY) + 2 \times (BLOCK)$
C+	182	"
D	194	"
D+	205	"
E	217	"
F	230	"
F+	244	"
G	2	$32 \times (SUS) + 16 \times (KEY) + 2 \times (BLOCK) + 1$
G+	17	"
A	32	"
A+	51	"
B	69	"

20H ~ 28H

-	-	SUS	KEY	x	x	x	x
---	---	-----	-----	---	---	---	---

 サスティン・キーオン指定

各チャンネルのサスティン・キーのオン・オフを設定します。

ビット 4（KEY）は、チャンネルのキーオン・オフ情報です。0 から 1 に変化することでそのチャンネルの発音が開始され、1 から 0 への変化で消音（エンベロープがリリースに突入）します。但し、リズムモード時（レジスタ 0EH のビット 5 が 1）のレジスタ 26H ~ 28H では、必ず 0 にしてください。

ビット 5（SUS）を 1 にすると、以降そのチャンネルで発音される音色のリリースレイトが内蔵・設定音色に関わらず強引に 5 に設定されます。これを指定することでキーオフをした後も若干余韻が残るような効

果を得ることができます。

30H ~ 38H Instruments Volume 音色・音量設定

各チャンネルの音色および音量を設定します。

ビット0 ~ 3で音量を設定します。最大音量からの変位で表され、0のとき最大、15で最小の音量となります。

ビット4 ~ 7で音色名を設定します。数値と発音する音色の関係は以下の通りです。

番号	音色名	番号	音色名
0	レジスタ0~7で設定する音色	8	オルガン
1	バイオリン	9	ホルン
2	ギター	10	シンセ
3	ピアノ	11	ハーブシコード
4	フルート	12	ビブラフォン
5	クラリネット	13	シンセベース
6	オーボエ	14	ウッドベース
7	トランペット	15	エレキベース

リズムモード時には、レジスタ 36H ~ 38H の機能が変化して、各ドラム音の音量設定用に使用されます。

36	-	Bass Drum	ドラムの音量設定
37	Hi Hat	Snare Drum	"
38	Tom Tom	Top Cymbal	"

レジスタ 36H のビット0 ~ 3でバスドラムの音量、37H のビット0 ~ 3でスネアドラム、ビット4 ~ 7でハイハットの音量...のように設定します。数値は通常音と同じく0で最大15で最小となります。

3.2.4 音色設定

FM 音源の詳しい原理等については本書では触れません。ものすごく大雑把な例えをするなら、各チャンネルには2つのオペレータ（周波数発生器）「キャリア」と「モジュレータ」がありまして、サイン波で唄おうとしている「キャリア」を「モジュレータ」が小突いて邪魔している、というようなイメージでしょうか。結果、キャリアはサイン波で唄うことはできず、モジュレータの小突きかたに従った音色で唄うことになるわけですね。もちろん唄うのはキャリアですから、音程・音量はキャリアの唄い方（=アプリケーションから指定する）によります。

OPLL は最初から内蔵で上掲の 15 個の音色を持っていて面倒な設定無しにいきなり使用できるのですが、それだけでは表現力が不足する場合にはアプリケーションが1つだけ音色を自作して追加することができます。そのためのレジスタがレジスタ 00 ~ 07H です。但し、音色設定のためのレジスタは1セットしかないため、アプリケーションが作れる音色は同時には1つだけしか存在できません。

OPLL の音色は、大雑把に「エンベロープ」「マルチプル」「変調量」で設定されます。それぞれ、音量の時間変化、倍音の構成、音色の明るさ・強さ（？）に関わっていますが、レジスタに書き込む際にはそれらを幾つかのパラメータに分けて設定します。具体的には、エンベロープをアタック・ディケイ・サスティン・リリースとエンベロープ形状等、マルチプルをマルチプル、変調量をトータルレベル・フィードバック等に分けて設定します。また、これ以外にもいくつかのパラメータが用意され、音色の表現力を増すことに役買っています。

以下に音色設定に関する各レジスタの詳細を示します。

(C), (M)は、それぞれキャリア、モジュレータを表します。

00H

AM

VIB

EGT

KSR

Multiple

音色設定 (M)

01H

AM

VIB

EGT

KSR

Multiple

音色設定 (C)

Multiple

各オペレータのマルチプル (Multiple, 倍音) 設定

各オペレータが、音程データで指定された音程の何倍の周波数を出力するかを指定します

各オペレータは、音程データで指定される周波数にマルチプルにより決定される倍率を掛けたものを発音周波数として使用します。マルチプルと倍率は次のような関係を持ちます

MUL	倍率
0	1/2
1 ~ 9	1 ~ 9
10・11	10
12・13	12
14・15	15

KSR

キースケールレイト (Key Scale Rate)

音程が上がるほどエンベロープの変化が速くなるという補正を加えます。0のときはそれほど加えず、1のときは大きく補正します

EGT

エンベロープ形状 (Envelope gene later type)

音色のエンベロープの形状を指定します。レジスタ 04 ~ 07H で指定される各パラメータは、この値によってその機能が変わります。0のとき減衰音タイプ、1のとき持続音タイプです

VIB, AM

音程に対する LFO (Vibration)、音量に対する LFO (Amplitude Modulation)

それぞれ、音量・音程に対し LFO (Low Frequency Oscillation) を掛けるかどうかの設定です。1にすることで、そのオペレータに LFO が掛かります。キャリアに対して掛けると音程や音量に対するビブラートとなり、モジュレータに対して掛けると周期的に音色が変化するような効果を得ることができます

02H	KSR(M) Total_level	音色設定
	Total_level	トータルレベル
		モジュレータの出力の強さを設定します。すなわち、キャリアに対する変調量の強さを設定することになります。値は最大値に対する変化量で、0のとき最大出力、63のとき最小になります。値が大きいほど丸い音色になり、値が小さくなるにつれて大きく変調が掛かり、華やかな音色になっていきます。但し、あまり小さな値だと音色自体が飽和してしまいます
	KSL	キースケールレベル/モジュレータ (Key Scale Level)
		音程が上がるほどオペレータの出力が小さくなるという補正を加えます。設定する値が大きいほど大きな補正を加えます

03H

KSR(C)		DC	DM	Feedback
--------	--	----	----	----------

音色設定

Feedback

フィードバック

モジュレータに対するフィードバックを指定します。モジュレータが出力した波形を自分自身に戻ってきて更に自分自身に変調を掛けるというものです。値が大きいほど強く効果が出ます

DM, DC

半波整流 (Distortion Modulator) , 半波整流 (Distortion Carrier)

各オペレータから出力される波形を半波整流 (ディストーション) します。1にすると機能し、掛けない場合に比べてかなりキャラクターの異なった (歪んだ) 音色になります

KSL

キースケールレベル / キャリア (Key Scale Level)

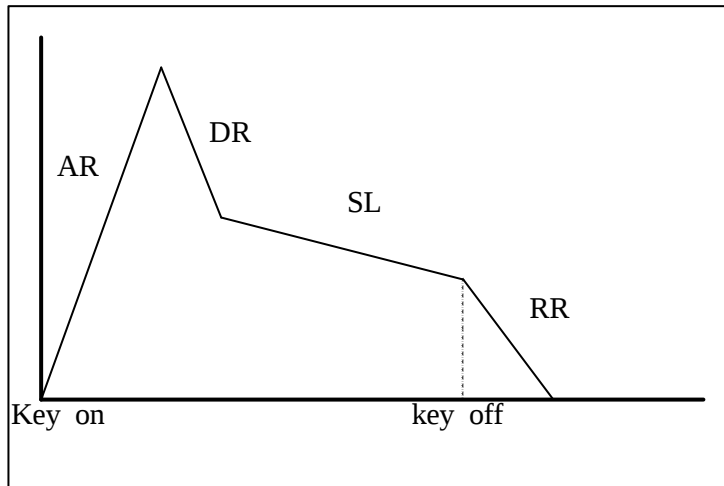
キャリアに対するKSLです

04H	Attack	Decay	音色設定 (M)
05H	Attack	Decay	音色設定 (C)
06H	Sustain	Release	音色設定 (M)
07H	Sustain	Release	音色設定 (C)
Decay			ディケイレイト (DR)
Attack			アタックレイト (AR)
Release			リリースレイト (RR)
Sustain			サスティンレベル (SL)

レジスタ04H~07Hでは、各オペレータのエンベロープ (音量の時間変化) を設定します。それぞれの役割は EGT (レジスタ00~01Hのbit5) の値によって異なります。

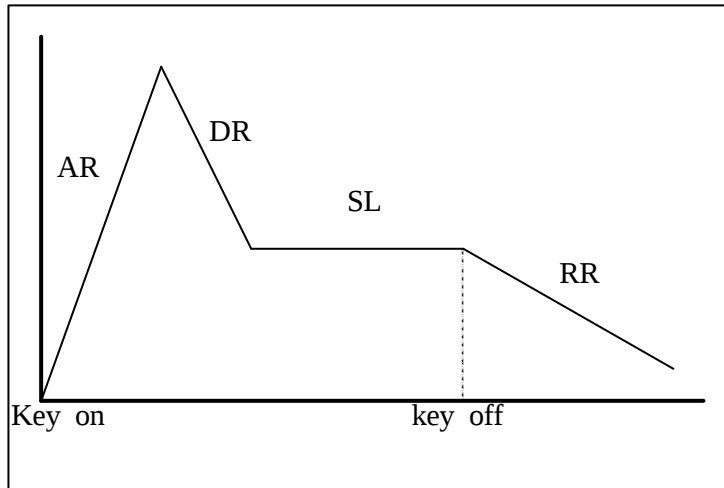
EGT=0（減衰音タイプの時）

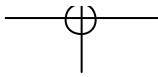
- Attack :発音してから最大音量になるまでの速度。15で最速（一瞬で音が立ち上がる）。
 Decay :最大音量に達してからサステインレベルになるまでの音量の減衰率。15で最大。
 Sustain :ディケイレイトからリリースレイトに変化する音量（音量そのものを指定。但し最大値に対する変位であるから、0のとき最大を表す）。
 Release :サステインレベルからの音量の変化率。15で最速。



EGT=1（持続音タイプの時）

- Attack :発音してから最大音量になるまでの速度。15で最速。
 Decay :最大音量に達してからサステインレベルになるまでの音量の減衰率
 Sustain :サステインレベルの音量（発音中持続する音量。最大値からの変位で表す）。
 Release :キーオフしてから音量0になるまでの時間。15のとき最速、0のとき音が止まらなくなる。





3.2.5 FM 音源 (OPLL) を使用する上での諸注意

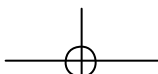
たとえば、ある周波数と別の周波数の間を滑らかに繋ぎたいような場合(ポルタメント)を考えます。2つの音が同じオクターブであれば、単純に音程データを増やす・減らす、してやれば問題はありません。しかし、オクターブをまたぐ場合にはどうでしょうか。データの構造上、「シ」の音とその上のオクターブの「ド」の音は値が繋がっていません(迂闊に BLOCK を上位3ビットとして12ビットデータだ、なんて計算をすると、間違いなくハマりますよ)。さて、どうしたものでしょうか。

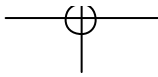
ここで、考え方を音源内部にまで立ち入らせてみましょう。オクターブが一つ上がると、それぞれ対応する音は周波数が2倍になります(例:o4A=440Hz、o5A=880Hz)。2オクターブ上なら4倍、3オクターブ上なら8倍...nオクターブ上なら2のn乗倍です。ところで、OPLLは音程を「基準クロックに対して何倍であるか」という形態で持っていますので、内部的な音程データは周波数に単純に比例した値を持つと考えられます。同じF-numberを持つデータでも、内部ではそのデータを($2^{\text{オクターブ}}$)倍していると考えerわけです。そこで、OPLLでは与えられた音程データ(F-numberとBLOCK)を、内部で $F\text{-internal} = F\text{-number} \times (2^{\text{BLOCK}})$ としている、と考えることができます。実際の発音周波数は、 $\text{Frequency} = F\text{-internal} \times \text{定数}$ となるわけですね。

これに従ってポルタメントを考え直すと、2つの音程データF-number 1・2、BLOCK 1・2の間を結ぶ場合には、これらのデータから $F\text{-internal} \cdot 2$ を計算し、その間を順次結んでいってやればよいことになります。当然、レジスタに書き込む際には元のF-numberとBLOCKの組みに戻してやらなければなりません。その際の計算は、F-internalが2の何乗で割れるか、という計算になりますが、その答えがBLOCK、余りがF-numberになります。割り算といっても、ビットシフトだけで処理できます。

MSX-MUSICのBIOSのINIOPLですが、初期化時に確保するA0HバイトのワークはMSX-MUSICのBIOSを使用しない限りその後は使用されないようです。フリーエリアの設定(【HIMEM】等)もアプリケーション任せのようで、ワークを使用し続ける必要がない(BIOSを使わない)場合ではしなくても構わないようです。更に、BIOSといってもWRTOPLに関しては別にワークを必要とはしていないようです。というわけで、レジスタなどの初期設定のためだけにINIOPLを使う場合は、何処か適当なアドレス(偶数であること)をダミーの先頭アドレスとして設定してやるだけで問題はないかと思われます。但し初期化中に思いつき値が書き込まれますので、壊れて困るようなデータがあるところには設定しないのが賢明です。

OPLLでは、ボリュームを15(音量最小)に設定していても若干音が出ます。完全に音を消音する際にはキーオフ(KEYを0)する必要があります。リズムパートでも同様です。





3章 SCC

3.3.1 SCC の概要

SCC (Sound Creative Chip) はコナミが開発し自社の MSX 用ゲームに搭載していた音源です。同社の「スナッチャー」「SD スナッチャー」に、外付けの音源として独立に添付されたことから音源としての認知が高まり、「MuSICA」「MGSDRV」などで実際にユーザー側からの操作が可能になりました。現在では MSX の音楽シーンに欠かせない音源の一つとなっています。

基本的な仕様として、

- ・独立した 5 つの発音ポートを持ち、それぞれに独自の波形を割り当てることが出来る
- ・128 バイトの波形メモリを持ち、1 ポート当たり 32 バイトの波形を設定することが出来る (ポート D, E の波形は共通)
- ・音程データ・ボリュームデータなどは、PSG と共通
- ・最大 512 キロバイトのローカルメモリをコントロール出来るメモリコントローラ搭載

ということが挙げられます。

ここでは音源のみに注目して紹介していきます。

なお、以下の記事で「SCC」と記載した場合は、特に断らない限り「SCC の音源部分」を指すものとします。

3.3.2 SCC の操作方法

ここでは、SCC のプログラムからの操作方法を紹介します。

SCC はコナミの独自規格ということで BIOS といった便利な機能は用意されていませんので、操作するためには直接 SCC 自身を叩いてやる必要があります。SCC はメモリマップド I/O として存在しています。よって、SCC をコントロールするためには、まず最初に SCC の存在するスロットを探し出す作業が必要となります。

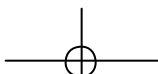
探す方法ですが、

- ページ 2 に調べるスロットを出す
- 9000h ~ 97FFh のいずれかのアドレスに 3Fh を書き込む (これで SCC の音源レジスタが表に出る)
- 波形メモリである 9800h ~ 987Fh に対し、読み書きの動作が行えるかをチェックする
- のチェックを通りかつそのスロットが RAM のスロットでなければ、SCC はそのスロットに存在する
- 違っていた場合は、書き込んだ際のアドレスを元に戻しておく

という順序で可能のようです。

SCC スロット (以下このように呼称) の 9000h ~ 97FFh は、SCC の 8000h ~ 9FFFh におけるバンク切り換えレジスタで、SCC に接続された ROM や RAM のバンクを切り替え、8000h ~ 9FFFh に出せるようになっているものです。SCC の音源レジスタは、そのバンクの 3Fh 番に存在する、というわけです。

また、波形メモリである 9800h ~ 987Fh は RAM として動作するので、読み書きができます。ここではそ



の機能を利用して SCC のチェックを行っています。

SCC スロットを発見し、SCC の音源レジスタを表に出した後は、そのスロットの特定の番地に対し読み書きの動作 (LD 命令等) を行うことで SCC を操作することができます。

3.3.3 SCC の内部レジスタ

ここでは、SCC をコントロールする際にソフト側から書き込むことになる、SCC の内部レジスタについて記載します。

以下、SCC に搭載されている 5 つの発音ポートを、それぞれチャンネル A ~ チャンネル E と呼称します。

アドレス	レジスタ	内容
9800 ~ 981Fh	Wave-data _ A	波形メモリ (Ch.A 用)
9820 ~ 983Fh	Wave-data _ B	波形メモリ (Ch.B 用)
9840 ~ 985Fh	Wave-data _ C	波形メモリ (Ch.C 用)
9860 ~ 987Fh	Wave-data _ D・E	波形メモリ (Ch.D, E 共用)
9880h	F-num _ A	チャンネル A 周波数 L
9881h	- F-num _ A	チャンネル A 周波数 H
	⋮	
9888h	F-num _ E	チャンネル E 周波数 L
9889h	- F-num _ E	チャンネル E 周波数 H
988Ah	- Volume _ A	チャンネル A 音量
	⋮	
988Eh	- Volume _ E	チャンネル E 音量
988Fh	- EE ED EC EB EA	出力スイッチ
98E0 ~ 98FFh	TEST	テスト用スイッチ(?)

- が書き込まれているビットは未使用です。0 で埋めておいてください。

各レジスタの詳細を以下に示します。

9800 ~ 987Fh	Wave-data _ A ~ E	波形メモリ (Ch.A ~ E 用)
--------------	-------------------	--------------------

各チャンネルが発音する音色の波形を設定します。
データは符号付き 8 ビット (-128(最小) ~ 0(中点) ~ 127(最大)) で、各チャンネルごとに 32 バイト構成です。
但し、チャンネル D とチャンネル E では、波形メモリは共通です。
音色データの詳細については 3.4.4 SCC 波形メモリで触れます。

9880 ~ 9889h チャンネルごとの周波数設定

9880h ~ 9888h	F-num _ A	チャンネル A ~ E 周波数 L
9881h ~ 9889h	- F-num _ A	チャンネル A ~ E 周波数 H

発声する周波数から求められる音程データを書き込み、各チャンネルごとの音程を設定します。
音程データは 12 ビットで表され、下位 8 ビットを偶数アドレス、上位 4 ビットを奇数アドレスに、それぞれ書き込みます。
なお、この際書き込む周波数データ F-num は PSG と共通になっています。具体的な数値については、1 章 PSG の該当項を参照してください。

988Ah ~ 988Eh チャンネルごとの音量設定

988Ah ~ 988Eh

-	Volume	A ~ E
---	--------	-------

 チャンネル A ~ E 音量

各チャンネルが発声する際の音量を設定します。
 値は 0 ~ 15 で、0 のとき無音、15 のとき最大となります。

988Fh

-	EE	ED	EC	EB	EA
---	----	----	----	----	----

 出力スイッチ

各チャンネルごとの発音の許可・不許可を設定します。
 EA (Enable ch.A) ~ EE (Enable ch.E) の各ビットに対し、それぞれ 0 (出力不許可)、1 (出力可) を設定します。

98E0 ~ 98FFh

TEST

 テスト用スイッチ(?)

0 を書き込むと SCC が初期化され、1 を書き込むと無音になります。
 ((SD) スナッチャーのカートリッジでは、アドレスが 9890 ~ 98DFh に変更されています)

なお、それぞれのレジスタに対する読み書きは、

9800 ~ 987Fh	読み込み・書き込み共に可
9880 ~ 988Fh	書き込み専用
9890 ~ 98DFh	読み込み・書き込み共に不可
98E0 ~ 98FFh	書き込み専用

(SD) スナッチャー附属カートリッジでは、

9800 ~ 987Fh	読み込み・書き込み共に可
9880 ~ 988Fh	書き込み専用
9890 ~ 98DFh	書き込み専用
98E0 ~ 98FFh	読み込み・書き込み共に不可

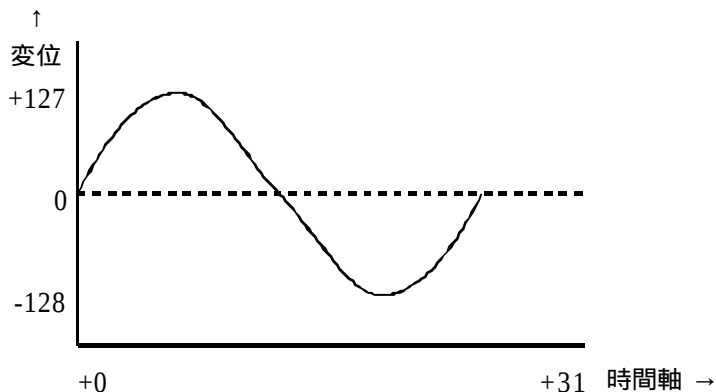
となっています。

3.3.4 SCC の波形メモリ

SCC が発音の際に出力する音色 (波形) は、PSG のように固定されたものではなく、ユーザーが自由に設定できるものです。

音色を設定するためには、まず使用したい音色の波形 1 周期分の音量変化を用意し、それを 32 個に区切り (標本化 / サンプリング)、-128 ~ 127 の変位に割り当て (量子化)、波形メモリに設定することで行われます。波形を時間軸に沿って変化させるフィルタ等は装備されていないので、出力される波形は基本的には設定された波形の繰り返しとなります。

例を挙げますと、たとえばサイン波を設定したい場合、



このように1周期分の波形を用意し、それを32等分して波形メモリ上に書き込むことになります。波形メモリ上での数値は「符号付き8ビット(-128(最小)~127(最大))」で表されているため、上記のグラフの縦軸のようなスケールで波形を数値化します。

なお、32バイトの波形メモリ中に2周期以上の波形を書くこともできます。当然発声される音程は2倍、3倍、...となります。また、波形の振幅も-128~127までフルに使う必要はなく、適当な幅で設定することができます。但し、あまりこの振幅を小さくすると、出てくる音色にノイズが乗りやすくなります。

波形を設定する場合、なるべく波形の最初と最後を繋いでおいた方がよいようです。具体的には、波形メモリ上で0バイト目と31バイト目の値を等しくしておいた方が、発音時のノイズが少なくなります。

3.3.5 SCCを使用する上での諸注意

SCCに対して書き込みを行う際、短時間に続けて大量に書き込み動作を行うと、非常にノイズっぽい音が出力されます。ウェイトを入れてもあまり効果がないようなので、SCCに対し書き込みを行う際には必要最小限のデータのみを書き込むだけにするのがよさそうです。

SCCの発音周波数ですが、周波数データであまり小さな値(10以下くらい?)を書き込んだ場合、SCCが誤動作する(というか発音出来ない)ようです。実際に音楽で使用する周波数をかなり越えているので実用には問題がないとは思われますが、特殊な用途(PCMに使うとか?)の場合には注意してください。

「スナッチャー」「SDスナッチャー」に附属していたSCCカートリッジは、他のカートリッジに内蔵されたSCCとは若干仕様が異なります。具体的には、チャンネルD・Eそれぞれに独立した波形メモリが設定され、音源レジスタのアドレスも移動しています。但し、他のSCCと同じ動作をするモードも存在し、通常はそのモードになっています(上記のモードにするためには特別の操作が必要)。

4章 MIDI

3.4.1 MSX における MIDI の位置付け

MIDI (Musical Instrument Digital Interface) とは、楽器同士、あるいは楽器と各種機器（コンピュータ含む）を接続してデータをやりとりするための統一規格です。これらに対応した機器・ソフトウェアなどを使用することで、コンピュータによる MIDI 対応楽器のコントロール・演奏等が可能になります。

MSX にも古くからインターフェイスが存在し、一部の本体では最初から実装されている場合もあったようです。最近では、turboR で制定された「MSX-MIDI」という規格によって、アプリケーションや BASIC からでも簡単に MIDI が操作できるようになりました。また、一部の音源ドライバでは独自のインターフェイスを利用することで MIDI を操作できるようにしているものもあります。

MIDI は単に楽器間の通信規格というだけでなく、民間主導の数少ない統一規格でもあります。そのため MIDI には様々な機能があり、またその用途も非常に多岐に渡ります。その全てを紹介することはとてもできませんので、詳細は市販の参考書に譲ることにしてここでは MSX で使用される主な MIDI インターフェイスの操作方法のみを取り上げることにします。

紹介するインターフェイスは MSX の正式な規格の MSX-MIDI です。

MSX-MIDI	MSX の正式な規格
MIDI-SAURUS	(株)Bit2 の製品
只 MIDI / DUAL-MIDI	戯音匠者氏作
へるへる 5 号	相田剛志氏作

ちなみに、MIDI という単語自体に「インターフェイス」という言葉が含まれているにも関わらず、「MIDI インターフェイス」などという言葉を使用したりしていますが、ここではとりあえず俗称ということでこの用語を使用させて頂きました。

3.4.2 MSX-MIDI

MSX-MIDI とは turboR 発表後に制定された規格です。基本的には従来の「MSX-MUSIC」に MIDI 機能を追加したという形態を採っていますが、ハードウェア的には独立した全く新しいものとなっています。パナソニックの FS-A1GT では本体に標準で実装され、同 FS-A1ST にも外付のカートリッジが用意されました。また、比較的入手しやすい部品で回路が構成できることから、ユーザー側からも独自に turboR 以前の機種でも使用できる互換インターフェイスが発表されました（MIDI インターフェイス 3：藤本昌利氏作）。拡張 BASIC が無いなど若干の使用の違いはあるものの、純正のものと同様にアプリケーションから利用できるようになっています。なお、MIDI インターフェイス 3 は内蔵 MSX-MIDI の互換品です。外付け MSX-MIDI 固有の機能（8251 のアドレス変更）は使えません。

仕様としては、以下のことが挙げられます。

- ・ MIDI IN・OUT を搭載。
- ・ UART を搭載(8251)。通信を全てハードウェアで行える。
- ・ カウンタを搭載(8253 または 8254。以下では 8253 とする)。

3.4.3 MSX-MIDIの認識

ここでは、MSX-MIDI が存在するかどうかをアプリケーション側から判別する方法を紹介します。

A1GTのように内蔵インターフェイスが存在する場合、メイン ROM の 002EH のビット 0 が 1 になっていますので、このビットをチェックすることで存在の確認ができます。純正の外付けインターフェイスの場合は、MSX-MIDI が接続されたスロットの 401CH ~ 401FH に「MIDI」という文字列が書き込まれていますので、そこをチェックすれば確認ができます。

が、「MIDI インターフェイス 3」のようなユーザー側から提唱されている互換インターフェイスの場合は ROM が搭載されていない(できない)ので、こういった認識方法が存在しません。そのため、全ての MSX-MIDI に対し存在のチェックをするためには、たとえば MSX-MIDI が接続されている I/O にカウンタなど必要なハードが接続されているかどうかを調べるなどの方法が採られているようです。あるいは、ユーザーにオプション等で指定させるという方法もあります。

3.4.4 MSX-MIDIの操作

従来の内蔵音源の場合、その操作は BIOS が用意されていたので比較的簡単に行うことができました。しかし MSX-MIDI に関しては、処理速度的に素早い操作が必要とされるため、BIOS 経由でなく直接 I/O を操作するということになっています。

MSX-MIDI は、送信・受信を専用の IC を使用して行うため、アプリケーションプログラム側の負担は比較的小さくなっています。また、専用のカウンタも内蔵しているため、データ転送時のタイミングも従来の【H_TIMI】を使用した 1/60 秒単位のものより精度を高くすることができます (BASIC では入力クロックである 4MHz を 2 万分周して 200Hz (5 ミリ秒毎) の割り込みを得ています)。

MSX-MIDI で使用される I/O ポートのアドレスを以下に示します。

8251

ラベル	アドレス	内容
UARTsend	0E8H	8251 データ送信
UARTrecv	0E8H	8251 データ受信
UARTcmd	0E9H	8251 コマンド・モードレジスタ
UARTstat	0E9H	8251 ステータス

外付け MIDI の場合、8251 が接続されるアドレスを変更することができます。これにより、内蔵 MIDI を持っている機種に外付け MIDI を接続することで、2 系統の MIDI I/O を持つことができるようになります。

0E2H	EN						E8	書き込み時
	EN	0=MIDI インターフェイスの使用許可、1=不可 (初期値)						
	E8	8251 アドレス設定 0=0E8H, 0E9H、1= 0E0H, 0E1H (初期値)						

E8 を 1 にした場合、8251 のアドレスはそれぞれ 0E0H, 0E1H になります。このとき、0E0H ~ 0E1H (8253) へのアクセスおよびカウンタからの割り込みは禁止されます。

なお、内蔵 MIDI の場合にはこのポートはありません。

8253

ラベル	アドレス	内容
tm_int	0EAH	8253 OUT2 端子の信号のラッチ
timer0	0ECH	8253 カウンタ#0
timer1	0EDH	8253 カウンタ#1
timer2	0EEH	8253 カウンタ#2
tm_cmd	0EFH	8253 コマンド

8253 のアドレスは、内蔵・外付けで共通です。

各レジスタの内容です。

8251 送信データ

0E8H/0E0H  書き込み時(UARTsend)
このレジスタに書き込まれたデータが MIDI-OUT を通して送信されます。

8251 受信データ

0E8H/0E0H  読み込み時(UARTrecv)
MIDI-IN に受信したデータはこのレジスタを通して読み取ることができます。
なお、このレジスタからデータを読み込んだ時点で MIDI-IN からの割り込みは解除されます。

以上 2 つのポートを操作する場合は、それぞれのレジスタが操作可能な状況であるかどうかを調べる必要があります。

8251 コマンドレジスタ (8251 の動作モードを設定します)

0E9H/0E1H  書き込み時(UARTcmd)

RIE	1=MIDI-IN 割り込みの可、0=不可
ER	1=なにもしない、0=エラーフラグをリセット
RE	1=MIDI-IN からの受信の可、0=不可
TIE	1=8253 カウンタ#2 割り込みの可、0=不可
TEN	1=MIDI-OUT への送信の可、0=不可

8251 ステータスレジスタ (8251 の状態を返します)

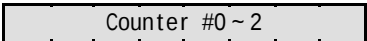
0E9H/0E1H  読み込み時(UARTstat)

DSR	8253 割り込みフラグ 1=割り込みあり、0=なし
BRK	1=8251 ブレークコード検出
FE	8251 フレームエラーフラグ (1=エラー発生)
OE	8251 オーバーランエラーフラグ (1=エラー発生)
PE	8251 パリティエラーフラグ (1=エラー発生)
EMP	8251 送信バッファステータス (1=送信バッファ空)
RRD	8251 受信バッファステータス (1=データあり)
TRD	8251 送信ステータス (1=送信可能)

8253 OUT2 端子の信号のラッチ (0EBH にもイメージ)

0EAH/0EBH  書き込み時(tm_int)
読み込みは無効。
このレジスタに値を書き込んだ時点で 8253 のカウンタ#2 からの割り込みが解除されます。

8253 カウンタ#0~#2

0ECH~0EEH  読み書き可(timer0~2)
それぞれ 16 ビットカウンタです。MSX-MIDI では、それぞれ 8251 のボーレートジェネレータ(500kHz:#0)、汎用(#1)、CPU への割り込み(#2)として使用されています。#0 と #2 には入力信号として 4MHz が入っていますが、#1 には #2 からの出力がそのまま入ってきます。

8253 コマンドレジスタ

0EFH	SC1	SC0	RW1	RW0	M2	M1	M0	BCD	書き込み時(tm_cmd)
	SC1,SC0	カウンタ選択 どのカウンタへの操作なのかを指定します。 00=カウンタ 0、01=カウンタ 1、10=カウンタ 2、11=無効							
	RW1,RW0	カウンタリードライトモード そのカウンタへのカウント値の設定方法を設定します。 00=カウンタ・ラッチ動作 01=LSB のロード 10=MSB のロード 11=LSB,MSB の順にロード							
	M2,M1,M0	カウンタモード そのカウンタのカウント方法を設定します。 000=モード 0 (カウント終了時に割り込み) 001=モード 1 (プログラマブルワンショット) x10=モード 2 (レートジェネレータ) x11=モード 3 (方形波レートジェネレータ) 100=モード 4 (ソフトウェアトリガストローブ) 101=モード 5 (ハードウェアトリガストローブ)							
	BCD	カウント形式 カウント形式 (バイナリ・BCD) を設定します 0=バイナリ (16 桁)、1=BCD カウント選択 (4 桁)							

読み込みは無効です。

それぞれのビットの詳細については、8251・8253 のマニュアルを参照してください。

MSX-MIDI を使用するためには、まずインターフェイスを初期化する必要があります。

まず、実際のシリアルデータ送信時などに使用されるカウンタ IC(8253)の初期化です。これは、以下のように行われます。

(tm_cmd) に 16H を出力
(timer0) に 08H を出力 ; カウンタ#0 をモード 3・8 分周(500kHz)と設定
(tm_cmd) に B4H を出力
(timer2) に 20H を出力
(timer2) に 4EH を出力 ; カウンタ#2 をモード 2・2 万分周(200Hz)と設定 (BASIC で使用)

この順にデータを書き込んでください。これでカウンタ IC が初期化され、初期値が設定されます。

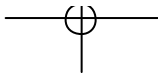
続いて、データ送受信用の IC(8251)を初期化します。これも以下の順にデータを I/O に書き込んでください。

(UARTcmd) に 00H 00H 00H 40H を出力。但し、1 つの書き込みの後 4 μ 秒以上待つこと
(UARTcmd) に 4EH を出力
4 μ 秒以上待つ
(UARTcmd) に 05H を出力
(UARTrecv) から値を読み込む
(tm_int) に値を書き込む

次に、インターフェイスの機能を停止させる手順を示します。アプリケーションが MIDI を使用しなくなったら、インターフェイスを停止させておきましょう (カウンタ割り込みのオーバーヘッドを減らすため)。

(UARTcmd) に 01H を書き込む
(UARTrecv) から値を読み込む
(tm_int) に値を書き込む

以上のインターフェイスの初期化を行った後、アプリケーションからインターフェイスを使用できるようになります。



まず、インターフェイスを通して MIDI データを送信する方法を示します。

(UARTstat) を読む
読んだ値のビット 0 が 0 なら 戻る
(UARTsend) に送信したいデータを書き込む

次に、データを MIDI-IN を通して受信する方法です。なお、(UARTcmd) のビット 5 を立てることで、MIDI-IN にデータが入ってきた時点で MSX に割り込みを掛けることもできます。

(UARTstat) を読む
読んだ値のビット 1 が 0 なら受信データは存在しない
(UARTrecv) を読む。ここで得られた値が受信データ

第 3 部

MIDI-IN がデータを受信すると、インターフェイスは MSX に対して割り込みをかけます((UARTcmd) のビット 5 が立っていた場合)。その際、内蔵 MIDI では【H_MDIN(0FF75H)】が呼ばれますが、外付けのインターフェイスの場合、通常の汎用割り込みフック【H_KEYI(0FD9AH)】が呼ばれます。よって、外付け MIDI の場合には掛かった割り込みが MSX-MIDI からのものかどうかを判断する必要があります。

(UARTstat) を読む
読んだ値のビット 1 が 0 なら MSX-MIDI の MIDI-IN による割り込みではない
(UARTrecv) を読む。ここで得られた値が受信データ。また、MIDI-IN での割り込みもこれで解除される

MSX-MIDI に搭載されているカウンタ (#2) は、指定した時間ごとに MSX に対し割り込みを掛けることができるようになっています。その際、内蔵 MIDI ではカウンタからの割り込み専用のフックである【H_MDTM(0FF93H)】がコールされますが、外付け MIDI では通常の汎用割り込みフック【H_KEYI(0FD9AH)】が呼ばれます。そのため、外付け MIDI の場合は掛かった割り込みが MIDI カウンタからのものかどうかを調べる必要があります。その判別方法ですが、

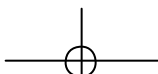
(UARTstat) を読む
読んだ値のビット 7 が 0 なら MSX-MIDI のカウンタによる割り込みではない
(tm_int) に対し書き込み動作を行う (out (tm_int),a)。これでカウンタがリセットされ、カウンタからの割り込みも解除される

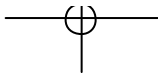
となっています。

カウンタへの値の設定方法ですが、

(tm_cmd) にカウント方法を設定
各カウントレジスタに値を設定

です。アプリケーションで自由に使用できるカウンタは #1 です。またカウンタ #2 はカウント終了時に CPU に対して割り込みを掛けますので、テンポ信号の生成などに使用できます (BASIC では 5 ミリ秒ごとの割り込みに固定)。カウンタ #0 は 8251 のボーレートジェネレータとして使用されていますので、「モード 3・8 分周」として初期化してください。





5章 1 ビットサウンドポート

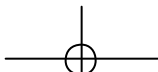
MSX に一番最初から搭載されていた音源には PSG の他に実はもう一つありまして、「1 ビットサウンドポート」と呼ばれるものがあります。これは、言ってみれば「1 ビットの PCM」で、出力の ON・OFF だけができるという非常に単純なものです。

使い方は、まずは BIOS の

- CHGSND (0135H/MAIN) MSX1
- | | |
|----|----------------------|
| 機能 | 1 ビットサウンドポートの操作 |
| 入力 | A 0:OFF 1:ON |
| 出力 | なし |
| 変更 | なし |
| 説明 | 1 ビットサウンドポートの状態を変えます |

を利用するか、あるいは直接 I/O ポートの AAh のビット 7（他のビットは別のことに使われているので、きちんと保存すること）を操作してください。

0AAh	<table><tr><td>SND</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	SND							
SND									



6章 PCM

3.6.1 PCM とは

PCM とは、Pulse Code Modulation の略で、音声を微小時間ごとに測定し、デジタル化して扱う録音・再生方式のことです。turboR では標準装備されています。

3.6.2 BIOS でのアクセス

マシン語での PCM アクセスには、次の BIOS が用意されています。

PCMPPLY	(0186H / MAIN)	PCM の再生
PCMREC	(0189H / MAIN)	PCM の録音

3.6.3 I/O ポートでのアクセス

3.6.3.1 ハードウェア

I/O ポートの直接操作による PCM のアクセスも可能です。

A4H	0	0	0	0	0	0	CT1	CT0	(Read)
A4H	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	(Write)
A5H	COMP	0	0	SMPL	SEL	FILT	MUTE	BUFF	(Read)
A5H	0	0	0	SMPL	SEL	FILT	MUTE	ADDA	(Write)

ADDA(BUFF)バッファモード 0=シングルバッファ 1=ダブルバッファ

D/A コンバータの出力信号を指定します。D/A時はダブルバッファ、A/D時はシングルバッファにしてください。リセット時はシングルバッファになっています。

MUTE ミューティング制御 0=音声出力 OFF 1=音声出力 ON

システム全体の音声出力を制御します。リセット時は OFF になっています。

FILT サンプル・ホールド回路入力信号の選択 0=基準信号 1=フィルタ出力信号

A/D 時にサンプル・ホールド回路の入力信号を選択します。リセット時は基準信号 (GND レベル) になっています。

SEL フィルタ入力信号の選択 0=D/A コンバータ出力信号 1=マイクアンプ出力信号

ローパスフィルタに入力する信号を選択します。リセット時は D/A コンバータ出力信号になっています。

SMPL サンプルホールド信号 0=サンプル 1=ホールド

入力信号のサンプル・ホールドを選択します。A/D 時は信号をホールドする前に、最低7μ秒はサンプルしなければいけません。リセット時はサンプルになっています。

COMP コンバータの出力信号 0=D/A 出力-サンプル・ホールド出力 1=D/A 出力+サンプル・ホールド出力

サンプル・ホールドの出力信号と、D/A コンバータの出力信号とを比較した結果を読み取ることができます。この機能は録音時に使用します。

DA7 ~ DA0 D/A 出力データ

CT1, CT0 カウンタデータ

63.5μ秒ごとにカウントアップされます (15.75kHz)。

D/A 時は、カウントアップに同期して A4H 番地に書かれたデータが繰り返し出力されます。A4H 番地に新たなデータを書き込むと、カウンタはクリアされます。

A/D 時は、A4H 番地に書かれたデータはただちに出力されます。A4H 番地にデータを書き込んでも、カウンタはクリアされません。

3.6.3.2 再生手順

PCM 再生は以下の手順で行います。

- A5H 番地に 00000011b を出力します。
- A4H 番地を読んで、サンプリング周期を検出します。
 - 1...15.75 kHz
 - 2... 7.875 kHz
 - 3... 5.25 kHz
 - 0... 3.9375kHz (一度 3 になった後に 0 まで待ちます)
- A4H 番地に PCM データを出力します。カウンタは自動的にクリアされます。
- 、 を繰り返します。

3.6.3.3 録音手順

PCM 録音は少々複雑な手順をとります。

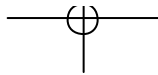
- A5H 番地に 00001100b を出力します。
- A4H 番地を読んで、サンプリング周期を検出します。カウンタは初期化されないで前回との差分を利用します。
 - 1...15.75 kHz
 - 2... 7.875 kHz
 - 3... 5.25 kHz
 - 0... 3.9375kHz (カウンタが 1 周するまで待ちます)
- 最低 7 μ 秒待ちます。 で既に 7 μ 秒以上経過していれば待つ必要はありません。
- A5H 番地に 00011100b を出力して、入力アナログ信号をホールドします。
- 逐次変換のシーケンスによって、D/A コンバータのデータを上位ビットから変化させながら、入力信号との比較結果を COMP から読み込んで各ビットを決定します。具体的には以下のようにします。
 - a データの候補として 80H をとりあげます。
 - b7 候補データを A4H 番地に出力します。
 - c7 A5H 番地を読んで、MSB が 1 ならば候補データの bit7 を 0 にします。
(R800 命令の「IN F(C)」を使うと便利です)
 - d7 候補データの bit6 を 1 にします。
 - b6 候補データを 0A4H 番地に出力します。
 - c6 A5H 番地を読んで、MSB が 1 ならば候補データの bit6 を 0 にします。
 - d6 候補データの bit5 を 1 にします。
 - ⋮
 - ⋮
 - ⋮ (ループにしない方が無難です)
 - b1 候補データを A4H 番地に出力します。
 - c1 A5H 番地を読んで、MSB が 1 ならば候補データの bit1 を 0 にします。
 - d1 候補データの bit0 を 1 にします。
 - b0 候補データを A4H 番地に出力します。
 - c0 A5H 番地を読んで、MSB が 1 ならば候補データの bit0 を 0 にします。
 - e データをメモリに格納します。
- A5H 番地に 00001100b を出力し、入力信号のホールドを解除します。
- ～ を繰り返します。
- A5H 番地に 00000011b を出力します。

3.6.4 その他

PCM データのフォーマットは 8 ビットですが、BASIC および BIOS の場合には 0 は圧縮用コードで、再生時はその後に続く 1 バイトの分だけ 0 レベル(127)を出力することになっています。

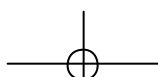
ここで疑問のある方もいらっしゃると思います。通常データは 1～255 で、Bias がかかっているの、0 レベル、すなわち中間値は 128 とするのが自然です。ところが BASIC や BIOS では 127 となっているのです。さて、どちらが 0 レベルでしょうか？

BASIC や BIOS の方を信用するのが良いと思いますが、結局のところ、どちらを 0 レベルとみても大差はありません。例えば BASIC で録音し、I/O ポートを直接操作して再生するときでも、ハードウェアやデータ



そのものが同じなので（0レベルのみなし方はソフトウェア上の違いなので）、全く気にする必要はありません。仮に、録音せずに PCM データを作ってしまう、それを再生したときにもし0レベルのみなし方が作成時と異なっていたとしても、出力信号全体の電位がわずかに変わるだけです（平たく言えばスピーカーの振動の中心がわずかにずれるだけです）。

また I/O ポートの直接操作による PCM アクセスの場合、 63.5μ 秒ごとにカウントアップする PCM 用のカウンタの代わりに 3.911μ 秒ごとにカウントアップするシステムタイマを使用すれば、サンプリングレートの幅が広がります。あまり速いサンプリングレートでは、特に録音時に CPU の速度が間に合わなくなることもあるのでご注意ください。



7章 PAC (Pana Amusement Cartridge)

3.7.1 PACについて

PAC (Pana Amusement Cartridge)とは 1987 年に松下電器産業株式会社パナソフトセンターから発売された MSX ゲーム用 S-RAM カートリッジです (品番 SW-M001 / 価格 3800 円)。8KB の S-RAM を搭載しており、これを 8 エリアに分ける事で複数のゲームデータを 1 つの PAC に共存させる事が可能です (領域が重ならない場合)。1988 年に発売された FM-PAC (品番 SW-M004 / 価格 7800 円) に搭載されている S-RAM も PAC と同様に扱えます。

PAC の S-RAM はページ 1 の 4000H ~ 5FFDH に配置されており、4000H ~ 43FFH がエリア 1、4400H ~ 47FFH がエリア 2、.....、5800H ~ 5BFFH がエリア 7、5C00H ~ 5FFDH がエリア 8 に対応しています。5FFEh, 5FFFh はメモリマップド I/O になっている為、S-RAM としてアクセスする事は出来ません。

3.7.2 PACの検索

PAC の検索は次のアルゴリズムを用いて行います。

ページ 1 を切り替える
 5FFEh からの 2 バイトを保存
 5FFEh に FFh, 5FFFh に FFh を書き込む
 4010h が書き込み可能かどうかを調べる
 書き込み可能であれば通常の RAM である (判定終了 を実行)
 書き込み不可能であれば ~ を実行
 5FFEh に 4DH, 5FFFh に 69h を書き込む
 4010h が書き込み可能かどうかを調べる
 書き込み不可能であれば ROM, 可能であれば PAC である (判定終了)
 で保存した値を、5FFEh からの 2 バイトに書き込む
 ~ を全てのスロットに対して行う

サンプルプログラム

● JDGPAC

機能 指定されたスロットに PAC が有るか判定する

入力 A 判定を行うスロットのスロットアドレス

出力 Z フラグ セット ; PAC が存在する
 リセット ; PAC が存在しない

変化 F BC DE HL

割込 禁止

特記 このルーチンから実行が返った後も、ページ 1 のスロットは切り替わったままです。

ENASLT	EQU	00024H
PAC_IO	EQU	05FFEh
PAC_TS	EQU	04010H

JDGPAC:

LD	(JDGPAC@1+1),A	; スロットアドレスを保存
LD	H,40H	
CALL	ENASLT	
LD	HL,(PAC_IO)	; 5FFEh の内容を保存
PUSH	HL	

```

LD      HL,0FFFFH      ; 5FFEh に FFFFh を書き込んだ状態で
CALL    JDGPAC3         ; 4010h に書き込めるか？
JR      Z,JDGPAC1       ; 書き込めたら通常の RAM と見なす

LD      HL,694DH        ; 5FFEh に 4DH,5FFFh に 69h を書き込んだ状態で
CALL    JDGPAC3         ; 4010h に書き込めるか？
JR      Z,JDGPAC2       ; 書き込めたら S-RAM(PAC)と見なす

JDGPAC1:
OR      H               ; ROM・RAM の場合は Zflag をリセット

JDGPAC2:
POP     HL
LD      (PAC_IO),HL     ; 5FFEh の内容を元に戻す
JDGPAC@1:
LD      A,0             ; A=与えられたスロットアドレス
RET     ; Zero flag が、Z;S-RAM(PAC) / NZ;RAM or ROM

JDGPAC3:                ; RAM/ROM の判定
LD      (PAC_IO),HL     ; 5FFEh に HL の値を書き込む
LD      HL,PAC_TS       ; 4010h に書き込めるか調べる
LD      A,(HL)           ; 1byte 読み
CPL     ; 反転させて
LD      (HL),A           ; 書き込む
CP      (HL)             ; 書き込んだ値と実際の 4010h の値を比較
CPL     ; 再反転させて元の値に戻し
LD      (HL),A           ; 書き込む
RET     ; Z=1;RAM / Z=0;ROM

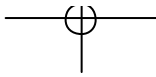
```

3.7.3 P A C へのアクセス

ページ 1 を PAC の存在するスロットに切り替え、5FFEh に 4DH、5FFFh に 69h を書き込むと PAC への読み書きが可能になります。

3.7.4 注意

FM-PAC を PAC として使用する場合、FM-BIOS の存在する領域が S-RAM に切り替わる為にタイマ割り込み等から FM-BIOS が呼ばれていると暴走します。これを防ぐには、PAC アクセス中の割り込みを禁止します。FM-PAC の FM-BIOS が使われていない事が明らかな場合は、割り込みを許可したままアクセスしても問題ありません。



8章 その他

ここでは、MSX の音源周りに関する、その他ちょっとしたこと・どうでもいいことを紹介します。

PSG の音声出力は、実は IC 自体からは3チャンネル別々に出力されています。互換チップ（カスタムチップにまとめられている場合も）でも同様な場合があります。通常の MSX では、これらの出力を1つにまとめて最終的な出力としているわけです。ということは、この IC から直接音を引きずり出して適当にミキシングしてしまえば、PSG がステレオで聴けるということになります（何をもってステレオと見なすかはこの際議論しない）。もちろん、特別なハードを付けない限りパンポットの設定はソフト側からは出来ないんですけど。

ちなみに、MSX の機種ごとに使用している PSG チップがまちまちなので、一概に「このピンが PSG だ」とは書けません（ちなみに本家 AY-3-8910 では、4, 3, 38 番ピンがそれぞれチャンネル A, B, C だったりします）。必要でしたら、本体の音声回路を出力側から逆方向に辿ってみてください。

また、OPLL もドラムと通常の音色を別々に引き出せるようになっていまして、これまたステレオ化することができます。ドラムが片方だけから聴こえてきても、あんまり嬉しくないかも知れませんが。14・15 番ピンからそれぞれの音が出ていますが、実際に使用する際には適当なフィルタ・バッファアンプ回路等を通してやる必要があるようです。

turboR から搭載された音源として PCM 音源があります。詳しい使い方はテクニカルガイドブックに譲りますが、ここでは単純に鳴らす方法を紹介しておきます。

1. I/O の 0A5H 番地に 03H を出力する
2. I/O の 0A4H 番地に出力したいデータを、任意のタイミングで好きなだけ出力する

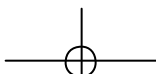
たとえば、SCC の出力周波数を存分高くして、更に適当なタイミングで波形メモリを書き換えてやりますと、PCM のようなこともできます。実際にコナミもゲーム中で使用したことがありました。が、「大量書き込みによるノイズ」「あまり高い周波数は発音不可」などの制限により、品質の高い PCM を実現するのは現実にはかなり難しいものがあるようです。

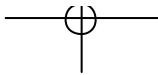
SCC、外付けの OPLL など外部スロット（SNDIN:各スロットの 49 番ピン）を通して接続される音源は、接続される本体ごとに内部音源に対する相対的な音量が異なっています。これは、スロットから入力される音声の音量が MSX の機種ごとに異なってミキシングされるためで、極端な例ではスロットからの音声入力が出来ないものもあります。

このため、各音源を同時に発音させた場合の音量のバランスは一意的には決められず、きちんと音源ごとにバランスを取るためには機種ごとに相対的な音量を設定する必要があります。SCC 他外付け音源を使用するアプリケーションでは、ユーザーが相対音量を設定出来るようなオプション・コンフィグを付けるべきです。

ちなみに、パナソニックの turboR（ST, GT）では、PSG と内蔵 OPLL のバランスは取れているものの、外付けの音源カートリッジの音量はそれらに比べ非常に大きくなっています。

内蔵・外付けに関わらず（但し MIDI は除く）、MSX に接続される音源は本体からのクロックを受けて音源内部の周波数を設定します。このため、本体のクロックを上げると音源から出てくる音の周波数も上がってしまうという現象が起きます。クロックアップにともなうリスクの一つですね。

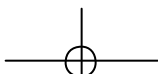




但し、turboR ではきちんと MSX 互換の周波数を音源に送るようになっており、7.16MHz な turboR でもちゃんと MSX2+以前と同じ周波数で音が鳴るようになっています。でもやっぱりクロックを上げると周波数は変わっちゃうんですけど。

MSX に限らず各種パソコン・ゲーム機などでも、音声出力を注意深く聴いていると僅かに「サー」「ザー」といったノイズが入っていると思います。また、画面の動き・キーボードの入力などに併せてノイズが大きくなったりする場合があります。これは別に故障ではなく（もちろん故障の場合もあるけど）、ノイズ発生源ともなるデジタル信号が走り回っている回路の中を、音声信号のような「直接五感に触れる」アナログ信号が通っているためです。

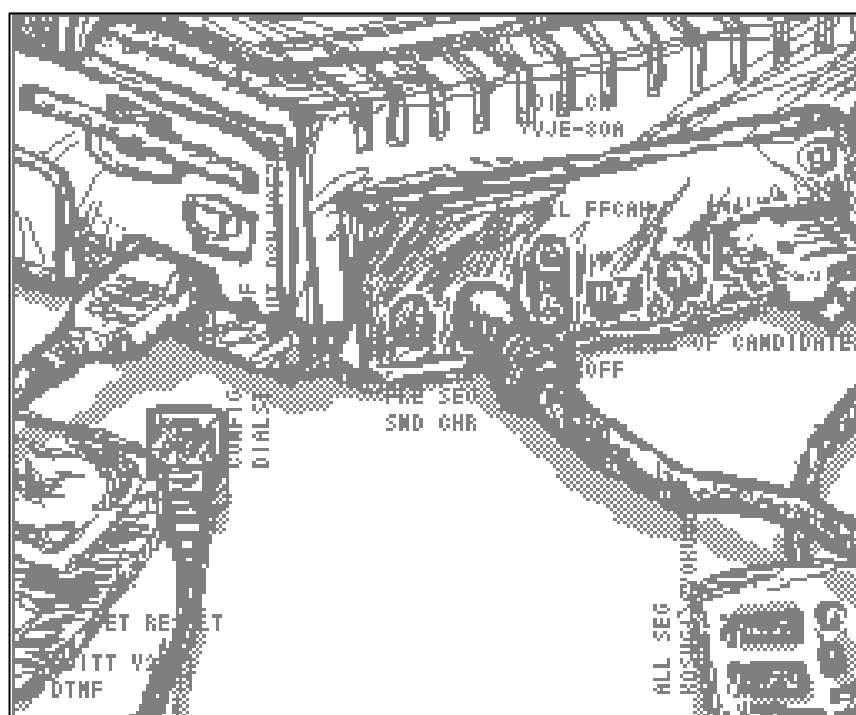
対策は立てようと思えば立てられるのですが、非常に大規模かつ高価な回路などが必要になりますので、低価格なパソコン・ゲーム機・音源ボードなどではその辺りで若干手を抜いている場合が多いようです。もちろん、高価な MIDI 音源・デジタルオーディオ機器などでは存分に対策が講じられてあります。

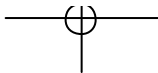


第 4 部

周边装置

第 4 部





1章 汎用入出力インターフェイス

4.1.1 インターフェイスの構造

いわゆるジョイスティックポートです。MSX には1つ又は2つ（MSX2以降では必ず2つ）の汎用入出力インターフェイスがあり、ジョイスティックやマウスなどを接続して使用しています。

汎用入出力インターフェイスはPSGが内蔵している入出力ポートA及びBに接続されています。PSGの入出力ポートはPSGのレジスタ#14（ポートA）及び#15（ポートB）を通してアクセス出来ます。ポートAは入力モード、ポートBは出力モードで固定して使われます。ポートAのビット0～5は汎用入出力インターフェイス1及び2で共用になっており、ポートBのビット6によって接続を切り換えます。

第4部

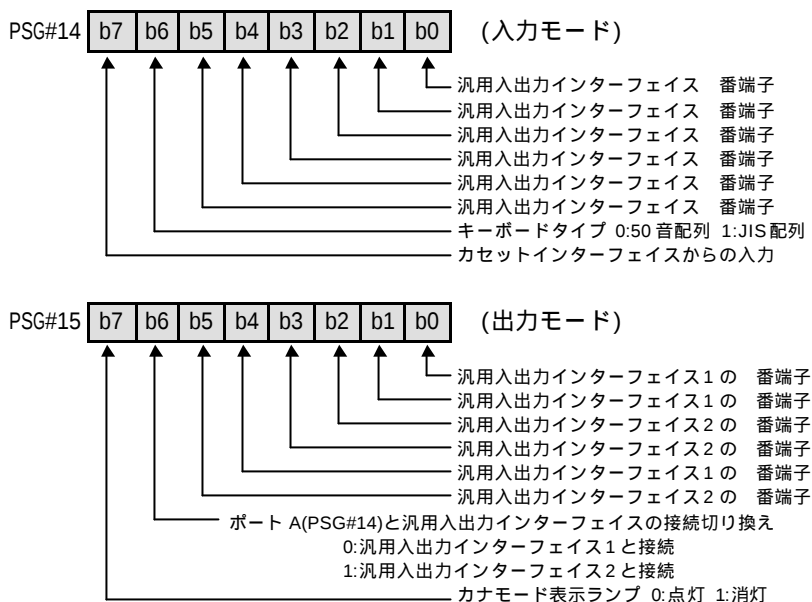
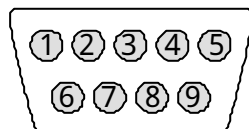
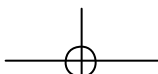
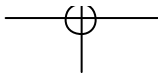


図 4.1.1 汎用入出力ポートのピン配置



ピン番号	信号名	方向
	FWD	入力
	BACK	入力
	LEFT	入力
	RIGHT	入力
	+5V	
	TRG1	入出力
	TRG2	入出力
	COM	出力
	GND	
レベル	TTL	
コネクタ	D-SUB9	ピン(本体側オス)



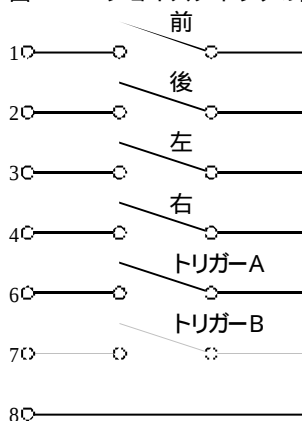


4.1.2 ジョイスティック

ジョイスティックのアクセスには専用の BIOS があります。通常は BIOS を使って行って下さい。BIOS の GTSTCK(00D5H/MAIN) 及び GTTRIG(00D8H/MAIN) を使用します。

直接ジョイスティックを読む場合は、 番端子に 0 を出力している状態で ~ 、 ~ の各端子からジョイスティックの状態を読む事ができます。ボタンが押されていれば 0、押されていなければ 1 が読み出されます。

図 4.1.2 ジョイスティックの回路図



4.1.3 マウス

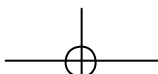
マウスにはカウンタモードとジョイスティックモードがあります。パワーON 時に左ボタンを押しているとジョイスティックモードになります。ジョイスティックモードではマウスをジョイスティックとして使用することができます。

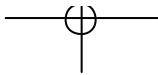
カウンタモードはマウスをマウスとして使うモードです。マウスは内部で X 及び Y 方向の移動量を 8 ビットでカウントし、MSX 本体からの要求があるとその値を返します。

マウスの値を読むには BIOS の GTPAD(00DBH/MAIN) を使います。マウスのボタンの状態の読み取りには GTTRIG を使います。

4.1.4 インターフェイス使用上の注意

システムのタイマー割り込み処理では ON STRIG の処理を行う為にジョイスティックをスキャンします。この時汎用入出力インターフェイス 1 及び 2 は、 番端子が HIGH に、 番端子が LOW になります。これらの端子を出力として使う機器を設計する場合は、この状態を定常状態として回路を設計し、アクセスする時は割り込みを禁止するようにして下さい。





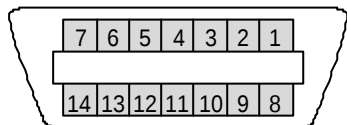
2章 プリンターフェイス

4.2.1 プリンターフェイス

プリンタの出力は 8 ビットの平行ポートで、BUSY と STROBE 信号によるハンドシェーク方式でプリンタを制御します。

ピン番号	信号名	ピン番号	信号名
1	-PSTB	8	PDB6
2	PDB0	9	PDB7
3	PDB1	10	NC
4	PDB2	11	BUSY
5	PDB3	12	NC
6	PDB4	13	NC
7	PDB5	14	GND

図 4.2.1 プリンターフェイス



アンフェノール 14 ピン(本体側メス)

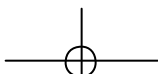
4.2.2 MSX 仕様のプリンタ

MSX 仕様のプリンタでは、MSX の使うグラフィックキャラクタやひらがなも印刷することができますが、MSX 仕様でないプリンタではこれらのキャラクタは印刷できません。

MSX 仕様でないプリンタに印刷する場合の為に、ひらがなをカタカナに変換して印刷するモードがあります。これは SCREEN 文によって設定できる他、【NTMSXP(F417H,1)】を書き換えて設定することもできます。【NTMSXP】が 0 だと MSX 仕様でないプリンタの為にひらがなをカタカナに変換します。0 以外ならひらがなをそのまま出力します。

4.2.3 プリンタへの出力

プリンタにデータを出力するには BIOS、LPOUT(00A5H/MAIN)、LPTSTT(00A8H/MAIN)、OUTDLP(014DH/MAIN) を使って下さい。



3章 キーボード

4.3.1 キー配列

MSX のキーボードの配列は、英数字は JIS 標準配列を、カナは JIS 標準配列および 50 音配列をジャンパ線により切り換えています。しかし、ジャンパ線による設定はシステム起動時に設定されるだけで、ワークエリア【KANAMD】の値を書き換えることで任意に変更可能です。

【KANAMD(0FCADH,1)】

0=50 音配列、0 以外=JIS 配列

4.3.2 キーマトリクス

キーボードは次のようなマトリクスになっています。キーの読み込みは読み込むキーの行を指定する事によって、1 行分のキーの状態が読み出されます。

図 4.3.1 キーマトリクス

	b7	b6	b5	b4	b3	b2	b1	b0
0	7	6	5	4	3	2	1	0
1	;	[	@	¥	^	-	9	8
2	B	A	_	/	.	,	]	:
3	J	I	H	G	F	E	D	C
4	R	Q	P	O	N	M	L	K
5	Z	Y	X	W	V	U	T	S
6	F8/F3	F7/F2	F6/F1	かな	CAPS	GRAPH	CTRL	SHIFT
7	RETURN	SELECT	BS	STOP	TAB	ESC	F10/F5	F9/F4
8	→	↓	↑	←	DEL	INS	HOME	SPACE

[TEN KEY]

9	4	3	2	1	0	option	option	option
10	.	,	-	9	8	7	6	5

キーマトリクスでは、押されているキーに対応するビットが 0、押されていないキーに対応するビットが 1 になります。

4.3.3 キースキャン

各キーをリアルタイムにスキャンする方法は、BIOS を用いる方法、I/O を用いる方法、およびワークエリアを読む方法の3種類があります。

4.3.3.1 BIOS を用いる方法

SNSMAT(0141H/MAIN) を使います。

4.3.3.2 I/O ポートを用いる方法

I/O ポート 0AAH に、以下のデータを書き込みます。

AAH		-		K	K	K	K
	K			読みだすキーマトリクスの行 (0~10)			
	-			変更してはならない			

ポート 0A9H からキーマトリクスの状態を読み出します。押されているキーに対応するビットが0、押されていないキーに対応するビットが1になります。

4.3.3.3 ワークエリアを読む方法

システムはタイマー割り込みルーチン内でキーマトリクスをスキャンしており、その状態はワークエリアに保存されています。キーが押されたままかを判断するために、ワークエリアは2種類用意されています。なお、タイマー割り込みを禁止するとこれらのワークエリアの状態は更新されませんので注意してください。

【OLDKEY(FBDAH,11)】 旧キーの状態
【NEWKEY(FBE5H,11)】 新キーの状態

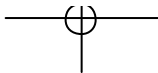
4.3.4 文字の入力

タイマー割り込みのキースキャンでは、キーの状態をワークエリアに保存すると同時に、押されたキーの文字コードをキーバッファに格納します。

4.3.4.1 キーバッファ

キーバッファは 40 文字分のリング状の構造をしています。文字が入力されるとバッファの最後に入れられ、CHGET によりバッファの先頭から文字が読み出されます。

【KEYBUF(FBF0H,40)】 キーバッファ
【PUTPNT(F3F8H,2)】 次にキーが押されたときに文字の入る位置
【GETPNT(F3FAH,2)】 次に CHGET ルーチンで得られる文字の位置



4.3.4.2 BIOS

キーボードバッファを使用したキー入力関係の BIOS を以下に示します。タイマー割り込みを禁止するとこれらのルーチンは使用できません。

CHSNS	(009CH/MAIN)	キーバッファの残りのチェック
CHGET	(009FH/MAIN)	キーバッファから 1 文字入力
KILBUF	(0156H/MAIN)	キーバッファクリア
CNVCHR	(00ABH/MAIN)	グラフィック文字の変換
PINLIN	(00AEH/MAIN)	文字列 1 行入力
INLIN	(00B1H/MAIN)	文字列 1 行入力
QINLIN	(00B4H/MAIN)	文字列 1 行入力

また、内部で以下のワークエリアを使用しています。

【CLIKSW(F3DBH,1)】	キークリック音 (0:OFF)
【SCNCNT(F3F6H,1)】	キースキャン間隔
【REPCNT (F3F7H,1)】	オートリピート開始までの時間
【MODE(FAFCH,1)】	ローマ字かな変換モード
【CSTYLE(FCAAH,1)】	カーソルの形 (0:■)
【CAPSTC(FCABH,1)】	CAPS キーの状態 (0:OFF)
【KANSTC(FCACH,1)】	カナキーの状態 (0:OFF)
【BUF(F55EH,258)】	ラインバッファ
【LINTTB(FBB2H,24)】	物理的 1 行が上の行に継続されるなら 0

4.3.5 ファンクションキー

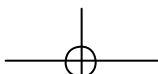
ファンクションキーは 10 個あり、ユーザーが自由に定義できます。ワークエリア【FNKSTR(F87FH,160)】の先頭から 16 バイトずつ、各キーに割り当てられています。1 つのキーにつき 15 文字まで定義可能（文末は 00H）です。また、次の BIOS ルーチンを使用するとファンクションキーを初期設定の状態に戻すことができます。

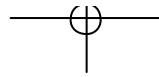
INIFNK	(003EH/MAIN)	ファンクションキー初期化
--------	--------------	--------------

4.3.6 CTRL+STOP の判定

CHGET などのルーチンはタイマー割り込みを禁止すると使用できませんが、割り込みが禁止されていても以下の BIOS で CTRL+STOP の判定を行うことが出来ます。このルーチンは、割り込みの状態を変化させません。また、必ず割り込み禁止の状態で呼び出してください。

BREAKX	(00B7H/MAIN)	CTRL+STOP 判定
--------	--------------	--------------





4章 リアルタイムクロック

4.4.1 リアルタイムクロック

MSX2 以降の機種にはリアルタイムクロック IC が内蔵されています。この IC は内蔵電池でバックアップされており、本体の電源を切っても動作し続けます。これによって現在の日時を知る事ができます。また、リアルタイムクロックの動作とは無関係に使えるレジスタがあり、MSX ではスクリーンモードなどのバックアップに使用しています。

4.4.2 クロック IC

4.4.2.1 レジスタの構成

クロック IC のレジスタは 4 ビットの構成になっています。

レジスタ#0～#12 の 13 個のレジスタが 4 ブロックと、#13～#15 の 3 個のレジスタがあります。レジスタ#0～#12 については 4 つのブロックのうちどれか一つを指定してからアクセスすることにより、どのレジスタもアクセスできます。

各ブロックのレジスタ#0～#12、レジスタ#13 は読み書きが可能です。レジスタ#14、#15 は書き込み専用で読み出しはできません。

どのブロックのレジスタをアクセスするかはレジスタ#13 で設定します。

4.4.2.2 クロックとアラーム

クロックの設定にはブロック 0 のレジスタを使います。アラームの設定はブロック 1 を使います。またブロック 1 には閏年と 12 時間計/24 時間計の選択があります。内容は表 4.1.1 を参照して下さい。

年は 2 桁で表されますが、MSX ではこの値に 80 のオフセットを加えて西暦の下 2 桁としています。

12 時間計/24 時間計の選択で 12 時間計が選択されている時は、10 時間カウンタの b1 ビットで午前(0)/午後(1)を表します。

閏年カウンタは年のカウントと同時に更新される 4 進カウンタで、このカウンタが 0 なら閏年として 2 月は 29 日までカウントします。日付を設定する時は西暦を 4 で割った値をこのレジスタに設定します。

4.4.2.3 バッテリバックアップメモリ

レジスタのブロック 2 と 3 は CLOCK の機能とは関係なく、電源を切っても内容を保持できるメモリとして使用することができます。

MSX ではこの領域をスクリーンモードなどの保存に使用しています。

ブロック 3 はレジスタ#0 の内容により 3 通りのうちどれかの機能を持ちます。

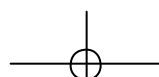


表 4.4.1 クロック / アラームレジスタ

		ブロック 0 (CLOCK)				ブロック 1 (ALARM)			
		b3	b2	b1	b0	b3	b2	b1	b0
0	秒 (1位)	x	x	x	x	-	.	.	.
1	秒 (10位)	.	x	x	x	-	.	.	.
2	分 (1位)	x	x	x	x	分 (1位)	x	x	x
3	分 (10位)	.	x	x	x	分 (10位)	.	x	x
4	時 (1位)	x	x	x	x	時 (1位)	x	x	x
5	時 (10位)	.	.	x	x	時 (10位)	.	.	x
6	曜日	.	x	x	x	曜日	.	x	x
7	日 (1位)	x	x	x	x	日 (1位)	x	x	x
8	日 (10位)	.	.	x	x	日 (10位)	.	.	x
9	月 (1位)	x	x	x	x	-	.	.	.
10	月 (10位)	.	.	.	x	12時 or 24時	.	.	x
11	年 (1位)	x	x	x	x	閏年カウンタ	.	.	x
12	年 (10位)	x	x	x	x	-	.	.	.

「・」で示したビットは常に「0」で変更は出来ない。

表 4.4.2 バッテリバックアップメモリ

	ブロック 2			
	b3	b2	b1	b0
0	ID			
1	Adjust X (-8 ~ +7)			
2	Adjust Y (-8 ~ +7)			
3	-	-	Interlace Mode	Screen Mode
4	WIDTH の値 (Lo)			
5	WIDTH の値 (Hi)			
6	前景色			
7	背景色			
8	周辺色			
9	Cassette Speed	Printer Mode	Key Click	Key ON/OFF
10	BEEP 音色		BEEP 音量	
11	-	-	タイトルカラー	
12	国別コード			

起動時タイトル (6 文字以内) 表示

	ブロック 3 (ID=0)	
0	0	
1	Lo 1	タイトル1 文字目
2	Hi 1	
:	:	
11	Lo 6	タイトル6 文字目
12	Hi 6	

パスワード設定

	ブロック 3 (ID=1)	
0	1	
1	Usage ID=1	
2	Usage ID=2	
3	Usage ID=3	
4	パスワード	パスワードを4ビット×4 に圧縮して格納
5	パスワード	
6	パスワード	
7	パスワード	
8	Key カートリッジ存在フラグ	
9	Key カートリッジの値	
10	Key カートリッジの値	
11	Key カートリッジの値	
12	Key カートリッジの値	

BASIC プロンプト設定

ブロック 3 (ID=2)			
0	2		
1	Lo 1	プロンプト 1 文字目	
2	Hi 1		
⋮	⋮	プロンプト 6 文字目	
11	Lo 6		
12	Hi 6		

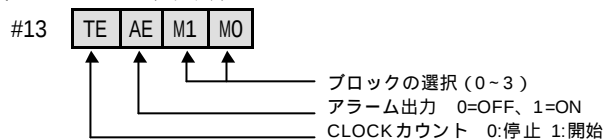
4.4.2.4 MODE レジスタ(#13)

MODE レジスタはブロックの選択、アラーム出力の ON/OFF、CLOCK カウント停止の機能があります。レジスタ#0～#12 をアクセスする時は、まずこのレジスタでアクセスするブロックを選択します。レジスタ#13～#15 はこのブロックの選択には関係なくアクセスできます。

アラーム出力は設定した時刻になるとアラーム信号が出力される機能ですが、MSX ではアラーム出力はオプションになっており、ほとんどの機種では出力端子が実装されていません。通常はアラーム OFF にしておきます。

CLOCK カウント停止の状態にすると、秒以降のカウントを停止します。ただし秒以前の分周段は停止しません。

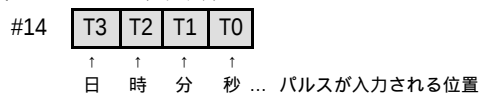
表 4.4.3 MODE レジスタ



4.4.2.5 TEST レジスタ(#14)

このレジスタの各ビットを 1 にすると対応するカウンタに 16384Hz のパルスが入力されます。カウンタのカウントアップ動作を確認するために使います。

表 4.4.4 TEST レジスタ

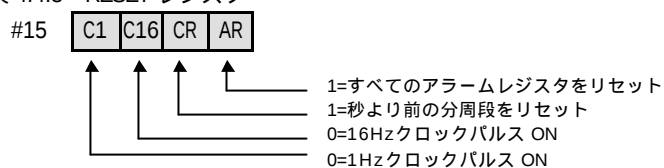


4.4.2.6 RESET レジスタ(#15)

このレジスタにはアラームリセット、秒合わせ、クロックパルス ON/OFF 機能があります。アラームリセットビットを 1 にすると、すべてのアラームレジスタをリセットします。秒合わせのビットを 1 にすると、秒以前の分周段をリセットします。1 秒の始まりを正確に合わせるために使用します。

クロックパルスの出力は MSX ではサポートされていません。

表 4.4.5 RESET レジスタ



4.4.3 アクセス

レジスタのアクセスには BIOS を使用します。REDCLK(01F5H/SUB)でデータを読みだし、WRTCLK(01F9H/SUB)でデータを書き込みます。

5章 漢字 ROM

4.5.1 漢字 ROM とは

MSX の漢字 ROM は、縦横 16×16 ドットのフォントを持っています。
現在、第 1 水準・第 2 水準の 2 種類の ROM が存在し、それぞれ 4096 文字分のフォントを持っています。

4.5.2 漢字 ROM の I/O ポート

D8H (Write)	漢字番号下位 6bit	第 1 水準漢字 ROM
D9H (Write)	漢字番号上位 6bit	
D9H (Read)	フォントデータ	
DAH (Write)	漢字番号下位 6bit	第 2 水準漢字 ROM
DBH (Write)	漢字番号上位 6bit	
DBH (Read)	フォントデータ	

漢字 ROM は I/O ポートに接続されています。

漢字番号（後述）を I/O ポートに書き込んだ後、I/O ポートを 32 回読むとフォントデータが得られます。データは、8 バイトずつ左上、右上、左下、右下の順になっています。スプライトパターンの場合とは順番が異なるのでご注意ください。

4.5.3 漢字 ROM のチェック

MSX の漢字 ROM 仕様ではフォントパターンは決まっておらず、各メーカーによって異なります。しかし、ある特定のフォントパターンは規定されており、そのパターンの有無をチェックすることにより、漢字 ROM が接続されているかどうかをチェックします。

4.5.3.1 第 1 水準漢字 ROM のチェック

JIS コード 2140H（1 区 32 点）の先頭 8 バイトが、
00H、40H、20H、10H、08H、04H、02H、01H となっています。

4.5.3.2 第 2 水準漢字 ROM のチェック

JIS コード 737EH（83 区 94 点）の先頭 8 バイトのチェックサム（8bit）が 95H となっています。

4.5.4 漢字コード変換法

漢字コードは、シフト JIS、JIS、区点、そして I/O 用の漢字番号、と 4 種類が存在します。
これらのコードの変換法を以下に示します(便宜上、パラメータの範囲ごとに計算式を分けています)。

変換元	変換先	変換法
シフト JIS(SH SL)	JIS	$SL(\sim 7EH) \quad JH = (SH \text{ and } 3FH) \times 2 + 31 \quad JL = SL - 31$ $(80H \sim 9EH) \quad JH = (SH \text{ and } 3FH) \times 2 + 31 \quad JL = SL - 32$ $(9FH \sim) \quad JH = (SH \text{ and } 3FH) \times 2 + 32 \quad JL = SL - 126$
	区点	$SL(\sim 7EH) \quad \text{区} = (SH \text{ and } 3FH) \times 2 - 1 \quad \text{点} = SL - 63$ $(80H \sim 9EH) \quad \text{区} = (SH \text{ and } 3FH) \times 2 - 1 \quad \text{点} = SL - 64$ $(9FH \sim) \quad \text{区} = (SH \text{ and } 3FH) \times 2 \quad \text{点} = SL - 158$
	漢字番号	JIS あるいは区点に変換した後、漢字番号に変換
JIS(JH JL)	S-JIS	$JH(\sim 5DH: \text{奇数}) \quad JL(\sim 5FH) \quad SH = (JH - 31) / 2 + 128 \quad SL = JL + 31$ $JH(\sim 5DH: \text{奇数}) \quad JL(60H \sim) \quad SH = (JH - 31) / 2 + 128 \quad SL = JL + 32$ $JH(\sim 5EH: \text{偶数}) \quad SH = (JH - 32) / 2 + 128 \quad SL = JL + 126$ $JH(5FH \sim : \text{奇数}) \quad JL(\sim 5FH) \quad SH = (JH - 31) / 2 + 192 \quad SL = JL + 31$ $JH(5FH \sim : \text{奇数}) \quad JL(60H \sim) \quad SH = (JH - 31) / 2 + 192 \quad SL = JL + 32$ $JH(60H \sim : \text{偶数}) \quad SH = (JH - 32) / 2 + 192 \quad SL = JL + 126$
	区点	$\text{区} = JH - 32 \quad \text{点} = JL - 32$
	漢字番号	(第1) $JH(\sim 29H) \quad N = (JH - 32) \times 96 + JL - 32$ $JH(30H \sim) \quad N = (JH - 32) \times 96 + JL - 544$ (第2) $N = (JH - 80) \times 96 + JL - 32$
区点	S-JIS	$(\sim 61 \text{ 区: 奇数}) \quad (\sim 63 \text{ 点}) \quad SH = (\text{区} + 1) / 2 + 128 \quad SL = \text{点} + 63$ $(\sim 61 \text{ 区: 奇数}) \quad (64 \text{ 点} \sim) \quad SH = (\text{区} + 1) / 2 + 128 \quad SL = \text{点} + 64$ $(\sim 62 \text{ 区: 偶数}) \quad SH = (\text{区} + 0) / 2 + 128 \quad SL = \text{点} + 158$ $(63 \text{ 区} \sim : \text{奇数}) \quad (\sim 63 \text{ 点}) \quad SH = (\text{区} + 1) / 2 + 192 \quad SL = \text{点} + 63$ $(63 \text{ 区} \sim : \text{奇数}) \quad (64 \text{ 点} \sim) \quad SH = (\text{区} + 1) / 2 + 192 \quad SL = \text{点} + 64$ $(64 \text{ 区} \sim : \text{偶数}) \quad SH = (\text{区} + 0) / 2 + 192 \quad SL = \text{点} + 158$
	JIS	$JH = \text{区} + 32 \quad JL = \text{点} + 32$
	漢字番号	(第1) $(\sim 9 \text{ 区}) \quad N = (\text{区} - 0) \times 96 + \text{点}$ $(16 \text{ 区} \sim) \quad N = (\text{区} - 0) \times 96 + \text{点} - 512$ (第2) $N = (\text{区} - 48) \times 96 + \text{点}$
漢字番号(N)	S-JIS	JIS あるいは区点に変換した後、シフト JIS に変換
	JIS	(第1) $(\sim 927) \quad JH = (N + 0) \text{ ¥ } 96 + 32 \quad JL = (N + 0) \bmod 96 + 32$ $(1024 \sim) \quad JH = (N + 512) \text{ ¥ } 96 + 32 \quad JL = (N + 512) \bmod 96 + 32$ (第2) $JH = (N + 0) \text{ ¥ } 96 + 80 \quad JL = (N + 0) \bmod 96 + 32$
	区点	(第1) $(\sim 927) \quad \text{区} = (N + 0) \text{ ¥ } 96 \quad \text{点} = (N + 0) \bmod 96$ $(1024 \sim) \quad \text{区} = (N + 512) \text{ ¥ } 96 \quad \text{点} = (N + 512) \bmod 96$ (第2) $\text{区} = (N + 0) \text{ ¥ } 96 + 48 \quad \text{点} = (N + 0) \bmod 96$

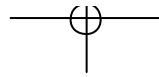
第1水準 シフト JIS $\sim 989EH$ 、JIS $\sim 4F7EH$ 、区点 ~ 47 区

第2水準 シフト JIS 989FH $\sim$ 、JIS 5021H $\sim$ 、区点 48 区 $\sim$

4.5.5 半角文字のフォント

半角文字 (ASCII コード 21H $\sim$ 7EH、0A0H $\sim$ 0DFH) の漢字番号は、以下のように計算します。
全角 1 文字分のフォントの左半分が 8 $\times$ 16 ドットの、右半分が 8 $\times$ 12 ドットの半角文字フォントです。

21H $\sim$ 7EH	漢字番号=ASCII コード - 32
A0H $\sim$ DFH	漢字番号=ASCII コード + 704



6章 システムタイマー

turboR から、短い時間のタイミングを計測するためにシステムタイマーが新設されました。turboR では CPU が R800 に変わったため命令実行時間によって正確なタイミングを取る事ができません。これにより、正確なタイミングが必要な I/O アクセスなどに支障が出る可能性があります。

正確なウェイトやタイミングを要するプログラムは、ソフトウェアループを使用せずにシステムタイマーを使って時間を計測します。

システムタイマーは 3.911 μ 秒毎にカウントアップする 16 ビットのタイマーです。システムタイマーは 2 つの I/O ポートに下位 8 ビット、上位 8 ビットがそれぞれ接続されています。上位 8 ビットだけで 1.001 ミリ秒毎にカウントアップするタイマーとして使うこともできます。

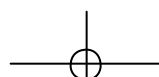
カウンタをクリアしてからカウントが 1 になるまでの時間は 0 ~ 3.911 μ 秒の範囲です。その範囲内で誤差が発生しますので、ウェイトに使う場合は注意してください。

16 ビットのカウンタ値を得る為には I/O ポートを 2 回アクセスする必要がありますが、システムタイマーにラッチの機能はないので、上位と下位のデータを読み出す間にカウントアップが発生すると正確な値を読み出すことができません。システムタイマーの上位または下位 8 ビットのみを使用するか、複数回の読み出しを行って上位への繰り上がりの発生を検出できるようにするなどの注意が必要です。

割り込み処理ではシステムタイマーをリセットしないで下さい。フォアグラウンドで動作しているプログラムが誤動作する可能性があります。システムタイマーを使用する場合はシステムタイマーをリセットせず、最初に一度読み込んだ値と現在の値の差からタイミングを取るようにします。

システムタイマーのカウントアップクロックは 10.738635MHz のシステムクロックを 42 分周したものです。

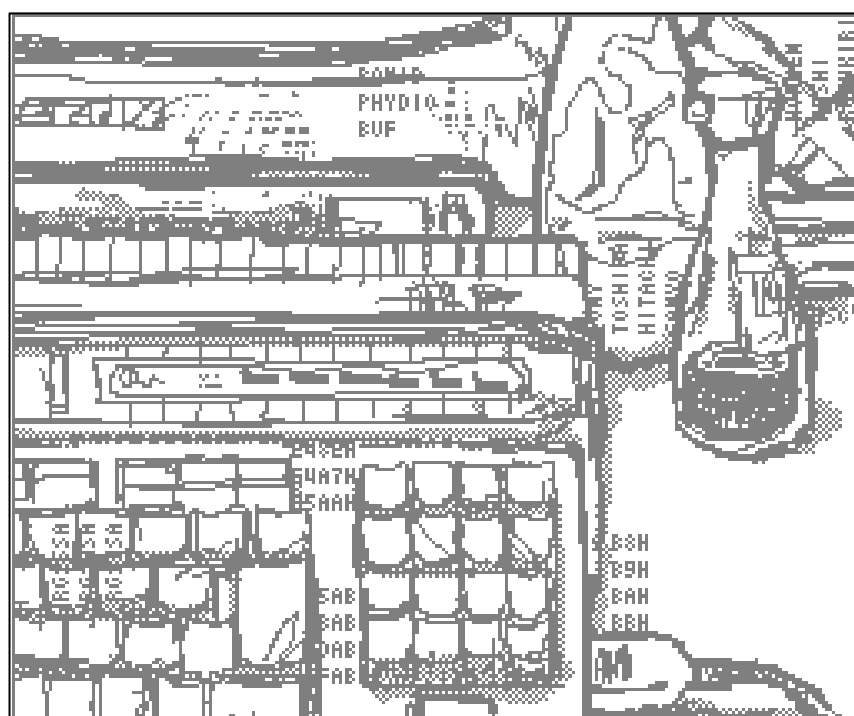
I/O ポート	R / W	機能
0E6H	W	任意のデータを書き込むとカウンタがクリアされる
0E6H	R	カウンタの下位 8 ビットの読み出し
0E7H	R	カウンタの上位 8 ビットの読み出し



第 5 部

資料

第
5
部



1章 BIOS

MAIN ROM

CHKRAM (0000H / MAIN) MSX1

機能 RAMのチェックをし、システム用のRAMに使うスロットを選択します。このルーチンの実行後は、さらに初期化のルーチンへ分岐します。

入力 なし
出力 なし
変更 なし

SYNCHR (0008H / MAIN) MSX1

機能 HLレジスタが指す文字が指定した文字かどうか調べます。違っていたら「Syntax error」を発生し、同じであればCHRGTR(0010H/MAIN)にジャンプします。

入力 HLにチェックする文字。このルーチンを呼び出すRST命令の後に比較する文字をセットする。(インラインパラメータ)

例) LD HL,MOJI
RST 08H
DEFB 'A'
:
MOJI: DEFB 'B'

出力 HL インクリメント(+1される)
A インクリメントされたHLの指す文字
Cy チェックした文字が数字であればセット
Z ステートメントの終わり(00Hまたは3AH)ならばセット

変更 AF HL

RDSLT (000CH / MAIN) MSX1

機能 他のスロットを1バイト読み込む。このルーチンを呼び出すと割り込みを禁止し実行を終えても割り込みは解除しません。

入力 A スロット番号
HL 読み出すアドレス
出力 A 読み出したメモリの値
変更 AF BC DE

CHRGTR (0010H / MAIN) MSX1

機能 BASICテキストから文字(またはトークン)を取り出します。

入力 HL 読み込む文字が入っているアドレス
出力 HL インクリメント
A インクリメントされたHLの指す文字
Cy チェックした文字が数字であればセット
Z ステートメントの終わり(00H又は3AH)ならばセット

変更 AF HL

WRSLT (0014H / MAIN) MSX1

機能 他のスロットに1バイト書き込む。このルーチンを呼び出すと割り込みを禁止し実行を終えても割り込みは解除しない。

入力 A スロット番号
HL 書き込むアドレス
E 書き込む値

出力 なし
変更 AF BC D

OUTDO (0018H / MAIN) MSX1

機能 現在使っているデバイスに値を出力します。

入力 A 出力する値
【PRTFLG(F416H)】0以外であれば、プリンタに出力

【PTRFIL(F864H)】0以外であれば、PTRFILで示されるファイルへ出力

出力 なし
変更 なし

CALSLT (001CH / MAIN) MSX1

機能 他のスロットのルーチンを呼び出す。(インタースロットコール)

入力 IY 上位8ビットにスロット番号
IX コールするアドレス

出力 呼び出すルーチンによる。
変更 IX IY 裏レジスタ
その他は呼び出すルーチンによる。

DCOMPR (0020H / MAIN) MSX1

機能 HLレジスタとDEレジスタの内容を比較します。

入力 HL 比較する値
DE 比較する値
出力 Z HL=DEならセット
Cy HL<DEならセット
変更 AF

ENASLT (0024H / MAIN) MSX1

機能 スロットを切り替えます。このルーチンを呼び出すと割り込みを禁止し実行を終えても割り込みは解除しません。

入力 A スロット番号
HL 呼び出すアドレス

出力 なし
変更 すべて

GETYPR (0028H / MAIN) MSX1

機能 DAC(デシマルアキュムレータ)の型を調べます。
出力 なし

(【VALTYP(F663H)】にはDACの型が入っている)
出力 DACの型によってS,Z,P/V,Cyのフラグが以下のように変化する。

整数型	単精度実数型	文字型	倍精度実数型
C=1	C=1	C=1	*C=0
*S=1	S=0	S=0	S=0
Z=0	Z=0	*Z=1	Z=0
P/V=1	*P/V=0	P/V=1	P/V=1

各型は*のついたフラグを調べればチェックできる。

変更 AF

CALLF (0030H / MAIN) MSX1

機能 他のスロットのルーチンを呼び出す。

入力 RST 0030H
DEFB nn ; nnは呼び出すスロット番号
DEFW nnnn ; nnnnは呼び出すアドレス

出力 呼び出すルーチンによる
変更 IX IY 裏レジスタ
その他は呼び出すルーチンによる

KEYINT (0038H / MAIN) MSX1

機能 割り込みルーチンを実行します。

入力 なし
出力 なし
変更 なし

INITIO (003BH / MAIN) MSX1

機能 デバイスを初期化します。

入力 なし

出力 なし
変更 すべて

INIFNK (003EH / MAIN) MSX1

機能 ファンクションキーの内容を初期化します。このルーチンを実行後、画面をクリアするとファンクションキーの表示も変わります。

入力 なし
出力 なし
変更 すべて

DISSCR (0041H / MAIN) MSX1

機能 画面表示を禁止します。

入力 なし
出力 なし
変更 AF BC

ENASCR (0044H / MAIN) MSX1

機能 画面を表示します。

入力 なし
出力 なし
変更 AF BC

WRTVDP (0047H / MAIN) MSX1

機能 VDP のレジスタに値を書き込みます。

入力 C VDP のレジスタ番号
B 書き込む値

出力 なし
変更 AF BC

RDVRM (004AH / MAIN) MSX1

機能 VRAM の指定したアドレスの内容を読み出します。ただし、このルーチンは TMS9918(MSX1 の VDP) に対するもので、VRAM のアドレスは下位 14 ビットのみ有効です。全ビットを使うときは、NRDVRM(0174H / MAIN) を使います。

入力 HL VRAM のアドレス
出力 A 読み出した値
変更 AF

WRTVRM (004DH / MAIN) MSX1

機能 VRAM にデータを書き込みます。ただし、このルーチンは TMS9918 に対するもので、VRAM のアドレスは下位 14 ビットのみ有効です。全ビットを使うときは、NWRVRM(0177H / MAIN) を使います。

入力 HL VRAM のアドレス
A 書き込む値
出力 なし
変更 AF

SETRD (0050H / MAIN) MSX1

機能 VDP に VRAM アドレスをセットして、読み出せる状態にします。このルーチンは VDP のアドレスオートインクリメントの機能を使って、連続した VRAM 領域からデータを読み出すときに使います。このルーチンの実行後は、ポートから直接 VRAM を読み出します。したがって、RDVRM をループ中で使うよりも高速な読み出しが出来ます。ただし、このルーチンは TMS9918 に対するもので、VRAM のアドレスは下位 14 ビットのみが有効です。全ビットを使うときは、NSETRD(016EH / MAIN) を使います。

入力 HL VRAM アドレス
出力 なし
変更 AF

SETWRT (0053H / MAIN) MSX1

機能 VDP に VRAM アドレスをセットして、書き込める状態にします。使用目的は SETRD と同じです。ただし、このルーチンは TMS9918 に対するもので、VRAM のアドレスは下位 14 ビットのみが有効です。全ビットを使うときは、NSTWRT(0171H / MAIN) を使います。

入力 HL VRAM アドレス

出力 なし
変更 AF

FILVRM (0056H / MAIN) MSX1

機能 VRAM の領域を同一のデータで埋めます。ただし、このルーチンは TMS9918 に対するもので、VRAM のアドレスは下位 14 ビットのみが有効です。全ビットを使うときは、BIGFIL(016BH / MAIN) を使います。

入力 HL 書き込みを開始する VRAM アドレス
BC 書き込む領域の長さ (バイト数)
A 書き込む値

出力 なし
変更 AF BC

LDIRMV (0059H / MAIN) MSX1

機能 VRAM からメモリへデータをブロック転送します。

入力 HL 転送元の VRAM アドレス (指定する VRAM のアドレスは全ビットが有効)
DE 転送先の RAM アドレス
BC 転送する長さ (バイト数)

出力 なし
変更 すべて
注意 転送先が 0 ~ 3FFFH は実行時 MAIN ROM が表にでているため転送できません。

LDIRVM (005CH / MAIN) MSX1

機能 メモリから VRAM へデータをブロック転送します。

入力 HL 転送元の RAM アドレス
DE 転送先の VRAM アドレス (指定するの VRAM アドレスは全ビットが有効)
BC 転送する長さ (バイト数)

出力 なし
変更 すべて
注意 転送元が 0 ~ 3FFF は実行時 MAIN ROM が表にでているため転送できません。

GHGMOD (005FH / MAIN) MSX1

機能 スクリーンモードを変えます。パレットは初期化しません。パレットの初期化が必要なときは、CHGMDP(00D1H / SUB) を使います。

入力 A スクリーンモード
出力 なし
変更 すべて

CHGCLR (0062H / MAIN) MSX1

機能 画面の色を変えます。

入力 【FORCLR(F3E9H)】前景色
【BAKCLR(F3EAH)】背景色
【BDRCLR(F3EBH)】周辺色

出力 なし
変更 すべて

NM (0066H / MAIN) MSX1

機能 NMI (Non Maskable Interrupt) 処理ルーチンを実行します。

入力 なし
出力 なし
変更 なし

CLRSRPR (0069H / MAIN) MSX1

機能 すべてのスプライトを次のように初期化します。

スプライトパターン	ヌル
スプライト番号	スプライト面番号
スプライトカラー	前景色
スプライトの垂直位置	209 (SCREEN0 ~ 3) 217 (SCREEN4 ~ 12)

入力 なし
出力 なし
変更 すべて

INITXT	(006CH / MAIN)	MSX1
機能	画面を TEXT1 モード (SCREEN0、40×24) に初期化します。このルーチンはパレットを初期化しません。パレットの初期化が必要であれば、このルーチンを実行後、INIPLT(0141H / SUB) を実行します。	
入力	【TXTNAM(F3B3H)】パターンネームテーブルのアドレス 【TXTCGP(F3B7H)】パターンジェネレータテーブルのアドレス 【LINL40(F3AEH)】1行の幅 (WIDTH 文によって設定する値)	
出力	なし	
変更	すべて	
INIT32	(006FH / MAIN)	MSX1
機能	画面を TEXT2 モード (SCREEN1、32×24) に初期化します。このルーチンはパレットを初期化しません。パレットの初期化が必要であれば、このルーチンを実行後、INIPLT(0141H / SUB) を実行します。	
入力	【T32NAM(F3BDH)】パターンネームテーブルのアドレス 【T32COL(F3BFH)】カラーテーブルのアドレス 【T32CGP(F3C1H)】パターンジェネレータテーブルのアドレス 【T32ATR(F3C3H)】スプライトアトリビュートテーブルのアドレス 【T32PAT(F3C5H)】スプライトジェネレータテーブルのアドレス 【LINL32(F3AFH)】1行の幅 (WIDTH 文によって設定する値)	
出力	なし	
変更	すべて	
INIGRP	(0072H / MAIN)	MSX1
機能	画面を GRAPHIC1 モード (SCREEN2) に初期化します。このルーチンはパレットを初期化しません。パレットの初期化が必要であれば、このルーチンを実行後、INIPLT(0141H / SUB) を実行します。	
入力	【GRPNAM(F3C7H)】パターンネームテーブルのアドレス 【GRPCOL(F3C9H)】カラーテーブルのアドレス 【GRPCGP(F3CBH)】パターンジェネレータテーブルのアドレス 【GRPATR(F3CDH)】スプライトアトリビュートテーブルのアドレス 【GRPPAT(F3CFH)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
INIMLT	(0075H / MAIN)	MSX1
機能	画面をマルチカラーモード (SCREEN3) に初期化します。このルーチンはパレットを初期化しません。パレットの初期化が必要であれば、このルーチンを実行後、INIPLT(0141H / SUB) を実行します。	
入力	【MLTNAM(F3D1H)】パターンテーブルのアドレス 【MLTCOL(F3D3H)】カラーテーブルのアドレス 【MLTCGP(F3D5H)】パターンジェネレータテーブルのアドレス 【MLTATR(F3D7H)】スプライトアトリビュートテーブルのアドレス 【MLTPAT(F3D9H)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
SETTXT	(0078H / MAIN)	MSX1
機能	VDP のみを TEXT1 モード (SCREEN0、40×24) にします	
入力	【TXTNAM(F3B3H)】パターンネームテーブルのアドレス 【TXTCGP(F3B7H)】パターンジェネレータテーブルのアドレス 【LINL40(F3AEH)】1行の幅 (WIDTH 文によって設定する値)	

出力 なし
変更 すべて

SETT32	(007BH / MAIN)	MSX1
機能	VDP のみを TEXT2 モード (SCREEN1、32×24) にします	
入力	【T32NAM(F3BDH)】パターンネームテーブルのアドレス 【T32COL(F3BFH)】カラーテーブルのアドレス 【T32CGP(F3C1H)】パターンジェネレータテーブルのアドレス 【T32ATR(F3C3H)】スプライトアトリビュートテーブルのアドレス 【T32PAT(F3C5H)】スプライトジェネレータテーブルのアドレス 【LINL32(F3AFH)】1行の幅 (WIDTH 文によって設定する値)	
出力	なし	
変更	すべて	
SETGRP	(007EH / MAIN)	MSX1
機能	VDP のみを GRAPHIC1 モード (SCREEN2) にします。	
入力	【GRPNAM(F3C7H)】パターンネームテーブルのアドレス 【GRPCOL(F3C9H)】カラーテーブルのアドレス 【GRPCGP(F3CBH)】パターンジェネレータテーブルのアドレス 【GRPATR(F3CDH)】スプライトアトリビュートテーブルのアドレス 【GRPPAT(F3CFH)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
SETMLT	(0081H / MAIN)	MSX1
機能	VDP のみをマルチカラーモード (SCREEN3) にします。	
入力	【MLTNAM(F3D1H)】パターンネームテーブルのアドレス 【MLTCOL(F3D3H)】カラーテーブルのアドレス 【MLTCGP(F3D5H)】パターンジェネレータテーブルのアドレス 【MLTATR(F3D7H)】スプライトアトリビュートテーブルのアドレス 【MLTPAT(F3D9H)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
CALPAT	(0084H / MAIN)	MSX1
機能	スプライトジェネレータテーブルの開始アドレスを獲得します。	
入力	A スプライト番号	
出力	HL アドレス	
変更	AF DE HL	
CALATR	(0087H / MAIN)	MSX1
機能	スプライトアトリビュートテーブルの開始アドレスを獲得します。	
入力	A スプライト番号	
出力	HL アドレス	
変更	AF DE HL	
GSPSIZ	(008AH / MAIN)	MSX1
機能	在のスプライトサイズを獲得します。	
入力	なし	
出力	A スプライトサイズ (バイト数) Cy 16×16 の場合のみセットし、それ以外のときはリセット	
変更	AF	
GRPPRT	(008DH / MAIN)	MSX1
機能	グラフィック画面に文字を表示します。	
入力	A キャラクターコード 【LOGOPR(FB02H)】スクリーンモードが 5～12 ならば口	

ジカルオペレーションコード

出力 なし
変更 なし

GICNI (0090H / MAIN) MSX1

機能 PSG を初期化し、PLAY 文のための初期値を設定します。
入力 なし
出力 なし
変更 すべて

WRTPSG (0093H / MAIN) MSX1

機能 PSG のレジスタにデータを書き込みます。
入力 A PSG のレジスタ番号
E 書き込む値
出力 なし
変更 なし

RDPSG (0096H / MAIN) MSX1

機能 PSG レジスタの値を読み出します。
入力 A PSG のレジスタ番号
出力 A 読み出した値
変更 なし

STRTMS (0099H / MAIN) MSX1

機能 バックグラウンドタスクとして、PLAY 文が実行中であるかどうかをチェックして、実行中でなければ音楽の演奏を始めます。
入力 【QUEUE(F3F3H)】の示すアドレスに、中間語に変換された MML データをセット
出力 なし
変更 すべて

CHSNS (009CH / MAIN) MSX1

機能 キーボードバッファの状態を調べます。
入力 なし
出力 Z バッファが空いていればセット、そうでなければセット
変更 AF

CHGET (009FH / MAIN) MSX1

機能 文字を 1 文字入力（入力待ちあり）します。
入力 なし
出力 A 入力された文字コード
変更 AF
補足 バッファに文字がない場合は、カーソルを表示して入力待ちになります。

CHPUT (00A2H / MAIN) MSX1

機能 文字を 1 文字表示します。
入力 A 出力する文字コード
出力 なし
変更 なし

LPTOUT (00A5H / MAIN) MSX1

機能 プリンタに 1 文字出力します。
入力 A 出力する文字コード
出力 Cy 失敗したときセット
変更 F

LPTSTT (00A8H / MAIN) MSX1

機能 プリンタの状態をチェックします。
入力 なし
出力 A 255 Z がリセットされていればプリンタは READY
0 Z がセットされていればプリンタは NOT READY
変更 AF

CNVCHR (00ABH / MAIN) MSX1

機能 グラフィックヘッダバイトかどうかをチェックします
入力 A チェックする文字コード

Cy=0	グラフィックコードではない
Cy=0 Z=0	グラフィックキャラクタコードである。 (A レジスタには変換後のコードが入る)
Cy=1 Z=0	グラフィックキャラクタではない。(A レジスタには渡したコードがそのまま残る)

変更 AF

補足 CHGET の後にこのルーチンを実行すると、2 バイトのグラフィック文字は 1 バイトに変換し（グラフィックヘッダが切り捨てられます）、グラフィック文字以外はそのまま返します。

PINLIN (00AEH / MAIN) MSX1

機能 リターンキーや STOP キーがタイプされるまで、入力された文字コードをバッファに格納します。
入力 なし
出力 HL バッファの終了アドレス - 1
Cy STOP キーで終了したときのみセット
変更 すべて

INLIN (00B1H / MAIN) MSX1

機能 【AUTFLG(F6AAH)】がセットされる以外は PINLIN と同じ。
入力 なし
出力 HL バッファの先頭アドレス - 1
Cy STOP キーで終了したときのみセット
変更 すべて

QINLIN (00B4H / MAIN) MSX1

機能 「?」とスペース 1 個を表示して、INLIN を実行します。
入力 なし
出力 HL バッファの先頭アドレス - 1
Cy STOP キーで終了したときのみセット
変更 すべて

BREAKX (00B7H / MAIN) MSX1

機能 CTRL+STOP キーが押されているかをチェックします。このルーチンは割り込みが禁止された状態でコールして下さい。
入力 なし
出力 Cy 押されていればセット
変更 AF

BEEP (00C0H / MAIN) MSX1

機能 ブザーを鳴らします。
入力 なし
出力 なし
変更 すべて

CLS (00C3H / MAIN) MSX1

機能 画面をクリアします。
入力 なし
出力 なし
変更 AF BC DE

POSIT (00C6H / MAIN) MSX1

機能 カーソルを移動します。
入力 H カーソルの X 座標
L カーソルの Y 座標
出力 なし
変更 AF

FNKSB (00C9H / MAIN) MSX1

機能 ファンクションキーの表示がアクティブかどうかをチェックし、アクティブなら表示し、そうでなければ消します。
入力 【FNKFLG(FBCEH)】
出力 なし
変更 すべて

ERAFNK (00CCH / MAIN) MSX1

機能 ファンクションキーの表示を消します。

入力 なし

出力 なし

変更 すべて

DSPFNK (00CFH / MAIN) MSX1

機能 ファンクションキーを表示します。

入力 なし

出力 なし

変更 すべて

TOTEXT (00D2H / MAIN) MSX1

機能 画面を強制的にテキストモードにします。

入力 なし

出力 なし

変更 すべて

GTSTCK (00D5H / MAIN) MSX1

機能 ジョイスティックまたはカーソルの状態調べます。

入力 A 調べるジョイスティックの番号

0=カーソルキー

1=ジョイスティック 1

2=ジョイスティック 2

出力 A カーソルまたはジョイスティックの押された方向

0=押されていない

1=上

2=右上

3=右

5=下

6=左下

7=左

8=左上

変更 すべて

GTTRIG (00D8H / MAIN) MSX1

機能 トリガーボタンの状態調べます。

入力 A 調べるトリガーボタンの番号

0=スペースキー

1=ジョイスティック 1 のトリガ A

2=ジョイスティック 1 のトリガ B

3=ジョイスティック 2 のトリガ A

4=ジョイスティック 2 のトリガ B

出力 A 0 トリガボタンは押されていない。

FF トリガボタンは押されている。

変更 AF, BC

GTPAD (00DBH / MAIN) MSX1

機能 各種入出力の状態調べます。

入力 A 調べる入出力装置の番号

0~3 タッチパネル 1

4~7 タッチパネル 2

8~11 ライトペン

12~15 マウス 1 またはトラックボール 1

16~19 マウス 2 またはトラックボール 2

出力 A 値

変更 すべて

装置 ID	指定装置	帰ってくる値
0	タッチパネル 1	パネル面に触っていれば 0FFH、 触っていなければ 00H
1		X 座標の増分 (-128 ~ 127)
2		Y 座標の増分 (-128 ~ 127)
3		ボタンが押されていれば 0FFH、 触っていなければ 00H
4~7	タッチパネル 2	同上

8	ライトペン	0FFH であればデータは有効、 00H であれば無効
9		X 座標の増分 (-128 ~ 127)
10		Y 座標の増分 (-128 ~ 127)
11		スイッチが押されていれば 0FFH、 いなければ 00H
12	マウスまたはトラック ボール 1	つねに 0FFH を返す (入力の要求に使用)
13		X 座標の増分 (-128 ~ 127)
14		Y 座標の増分 (-128 ~ 127)
15		常に 00H を返す (無意味)
16~19	マウスまたはトラック ボール 2	同上

MSX1 では、マウス・トラックボールの状態を調べることはできません。

マウス・トラックボールは自動的に判別します。

マウスやトラックボールの座標値を求める場合、まず入力要求コール (A:12 または A:16) を行いその後実際に座標値を求めるコールを実行するという手順をふみます。このとき 2 つのコールの時間間隔はできる限り小さくしてはなりません。

マウス・トラックボールのボタンの状態は GTTRIG(00D8H / MAIN) を使います。

ライトペンは turboR では削除されています。

GTPDL (00DEH / MAIN) MSX1

機能 バドルの状態調べます。

入力 A バドルの番号 (1 ~ 12)

出力 A バドルの回転角 (0 ~ 255)

変更 すべて

補足 この BIOS は turboR では削除されています。turboR では常に 0 が返されます。

TAPION (00E1H / MAIN) MSX1

機能 カセットレコーダーのモーターを動かし、テープのヘッダブロックを読み出します。

入力 なし

出力 Cy 失敗したらセット

変更 すべて

補足 この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。

TAPIN (00E4H / MAIN) MSX1

機能 テープからデータを読み出します。

入力 なし

出力 A 読みこんだ値

Cy 失敗したらセット

変更 なし

補足 この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。

TAPIOF (00E7H / MAIN) MSX1

機能 テープからの読み込みを停止します。

入力 なし

出力 A 読み込んだ値

変更 なし

補足 この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。

TAPOON (00EAH / MAIN) MSX1

機能 カセットレコーダーのモーターを動かし、テープのヘッダブロックを書き込みます。

入力 A 0 ショートヘッダ

0 以外 ロングヘッダ

出力 Cy 失敗したらセット

変更 すべて

補足 この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。

TAPOUT (00EDH / MAIN)	MSX1
機能	テープにデータを書き込みます。
入力	A 書き込む値
出力	Cy 失敗したらセット
変更	すべて
補足	この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。
TAPOOF (00F0H / MAIN)	MSX1
機能	テープへの書き込みを停止します。
入力	なし
出力	なし
変更	なし
補足	この BIOS は turboR では削除されています。turboR では常に Cy フラグをセットしてエラーを返します。
STMOTR (00F3H / MAIN)	MSX1
機能	カセットレコーダーのモーターの動作を設定します。
入力	A 0 ストップ 1 スタート FFH 現在と逆の動作状態にする
出力	なし
変更	AF この BIOS は turboR では削除されています。turboR では何もせずに戻ります。
CHGCAP (0132H / MAIN)	MSX1
機能	CAPS ランプの状態を変えます。
入力	A 0 ランプをつける 0 以外 ランプを消す
出力	なし
変更	AF
CHGSND (0135H / MAIN)	MSX1
機能	1 ビットサウンドポートの状態を変えます。
入力	A 0 サウンドポートのビットを OFF 0 以外 サウンドポートのビットを ON
出力	なし
変更	AF
RSLREG (0138H / MAIN)	MSX1
機能	基本スロット選択レジスタに出力している内容を読み出します。
入力	なし
出力	A 読み込んだ値
変更	A
WSLREG (013BH / MAIN)	MSX1
機能	基本スロット選択レジスタにデータを書き出します。
入力	A 書き込む値
出力	なし
変更	なし
RDVDP (013EH / MAIN)	MSX1
機能	VDP のステータスレジスタを読み出します。このルーチンは TMS9918 に対するものです。
入力	なし
出力	A 読み込んだ値
変更	A
SNSMAT (0141H / MAIN)	MSX1
機能	キーボードマトリクスから指定した行の値を読み出します。
入力	A 指定する行
出力	A 読み出した値（押しているキーに対するビットが 0 になる）
変更	AF C

ISFLIO (014AH / MAIN)	MSX1
機能	デバイスが動作中かどうかをチェックします。
入力	なし
出力	A 0 動作中 0 以外 動作中ではない
変更	AF
OUTDLP (014DH / MAIN)	MSX1
機能	文字を 1 文字プリンタに出力します。LPTOUT(00A5H / MAIN)とは以下の点で異なります。 1 . TAB はスペースに展開される。 2 . MSX 仕様でないプリンタに対しては、ひらがなはカタカナに、グラフィック文字は 1 バイト文字に変換する。 3 . 失敗したときは、「device I/O error」になる。
入力	A データ
出力	なし
変更	F
KILBUF (0156H / MAIN)	MSX1
機能	キーボードバッファをクリア
入力	なし
出力	なし
変更	HL
CALBAS (0159H / MAIN)	MSX1
機能	BASIC インタープリタ内のルーチンをインタースロットコール
入力	IX に呼び出すアドレス
出力	呼び出すルーチンによる
変更	呼び出すルーチンによる
SUBROM (015CH / MAIN)	MSX2
機能	SUB ROM をインタースロットコールします。
入力	IX に呼び出すアドレス、同時に IX をスタックに積む
出力	呼び出すルーチンによる
変更	裏レジスタと IX, IY は保存される その他は呼び出したルーチンによる
EXTROM (015FH / MAIN)	MSX2
機能	SUB ROM をインタースロットコールします。IX レジスタはスタックに積みません。
入力	IX に呼び出すアドレス
出力	呼び出すルーチンによる
変更	裏レジスタと IY は保存される その他は呼び出したルーチンによる
EOL (0168H / MAIN)	MSX2
機能	行の終わりまで削除します。
入力	H カーソルの X 座標 L カーソルの Y 座標
出力	なし
変更	すべて
BIGFIL (016BH / MAIN)	MSX2
機能	VRAM の指定領域を同一のデータで埋めます。機能的には FILVRM(0056H / MAIN)と同じです。ただし、次の点で異なります。FILVRM では、スクリーンモードが 0 ~ 3 であるかをチェックし、もしそうなら VDP は 16K バイトの VRAM しか持っていないものとして扱います。（MSX1 との互換性のため）。しかし、BIGFIL はモードのチェックは行わず、与えられたパラメータどおりに動作します。
入力	HL 書き込みを開始する VRAM アドレス BC 書き込む領域の長さ（バイト数） A 書き込む値
出力	なし
変更	AF BC

NSETRD (016EH / MAIN) MSX2

機能	VDP にアドレスをセットして、VRAM の内容が読み込める状態にします。
入力	HL VRAM アドレス 【ACPAGE(FAF6H)】アクティブページ
出力	なし
変更	AF

NSTWRT (0171H / MAIN) MSX2

機能	VDP にアドレスをセットして、VRAM に書き込める状態にします。
入力	HL VRM アドレス 【ACPAGE(FAF6H)】アクティブページ
出力	なし
変更	AF

NRDVRM (0174H / MAIN) MSX2

機能	VRAM の内容を読み出します。
入力	HL 読み出す VRM アドレス 【ACPAGE(FAF6H)】アクティブページ
出力	A 読み出した値
変更	F

NWRVRM (0177H / MAIN) MSX2

機能	VRAM にデータを書き込みます。
入力	HL 書き込む VRAM アドレス A 書き込む値 【ACPAGE(FAF6H)】アクティブページ
出力	なし
変更	AF

RDRES (017AH / MAIN) MSX2+

機能	RESET ポートの内容を読み出します。
入力	なし
出力	A MSB (ビット 7) が 0 ならばハードウェアリセット
変更	なし

RWRES (017D / MAIN) MSX2+

機能	RESET ポートに値を書き込みます。ハードウェアリセットをシミュレートするときは、A レジスタの MSB (ビット 7) を 0 にして、この BIOS をコールした後、MAIN-ROM の 0 番地にジャンプします。
入力	A 書き込む値
出力	なし
変更	A

CHGCPU (0180H / MAIN) MSXtR

機能	CPU を切り換えます。
入力	A b7 6 5 4 3 2 1 0 L 0 0 0 0 0 M M +-+----モード +-+---+-----常に 0 +-----LED
出力	なし
変更	AF
補足	A レジスタの内容によって CPU を切り換えます。もしビット 7 が 1 ならどちらの CPU が動作中かを示します。つまりビット 7 が 0 なら LED の状態は変化しません。

CPU の切り換えはシステムコントロール LSI S1990 内の設定を変えるだけです。そのため Z80 や R800 ROM モードに切り換えた後、空いた DRAM エリアの内容を破壊して、R800 DRAM モードに戻してもシステムは元に戻りません。もし、破壊した場合はシステムの転送はユーザー側で行わなければなりません。

DOS1 モードでディスクに R800CPU でアクセスした場合 DOS1 が R800 に対応していないためディスクを破壊する場合があります。

DOS1 でディスクにアクセスする場合は Z80 モードに切り換えて下さい。

GETCPU (0183H / MAIN) MSXtR

機能	現在、どの CPU で動作しているかを調べます。
入力	なし
出力	A 現在の CPU のモード 0 Z80 1 R800 ROM 2 R800 DRAM
変更	F

PCMPPLY (0186H / MAIN) MSXtR

機能	PCM のデータを再生します。
入力	A b7 6 5 4 3 2 1 0 R 0 0 0 0 0 S S +-+サンプリングレート +-+---+-----常に 0 +-----VRAM / メイン RAM

サンプリングレート	00	15.75KHz
	01	7.875KHz
	10	5.25KHz
	11	3.9375KHz
メモリ	0	メイン RAM
	1	VRAM

HL PCM データのアドレス
VRAM 指定時は、E レジスタと HL レジスタを合わせて、3 バイトで指定します。E レジスタが最上位バイトです。

BC PCM データの長さ
VRAM 指定時は、D レジスタと BC レジスタを合わせて、3 バイトで指定します。D レジスタが最上位バイトです。

出力	Cy	0 正常終了 1 異常終了
	A	異常終了原因 1 サンプリング周波数指定誤り 2 STOP キーによる中断

HL 中断時のアドレス
VRAM 指定時は、E レジスタと HL レジスタとを合わせた 3 バイトで返します。E レジスタが最上位バイトです。

変更 すべて
補足 Z80 モードの時は、自動的に R800 ROM モードに切り換えて実行し、終わると Z80 モードに戻ります。
PCM を再生している間は、割り込みは禁止されます。なお、STOP キーが押されると実行を中断します。この BIOS は実行がページ 1 で行われますので、データをメイン RAM に置くときは、8000H 以降でなければなりません。

PCMREC (0189H / MAIN) MSXtR

機能	PCM のデータを録音します。
入力	A b7 6 5 4 3 2 1 0 R T T T T C S S +-+サンプリングレート +----圧縮 +-+---+-----常に 0 +-----VRAM / メイン RAM

サンプリングレート	00	15.75KHz
	01	7.875KHz
	10	5.25KHz
	11	3.9375KHz
圧縮	0	圧縮しない
	1	圧縮する
メモリ	0	メイン RAM
	1	VRAM

トリガレベル

		録音のきっかけとなる音の大きさを指定します。	
HL	PCM データのアドレス	VRAM 指定時は、E レジスタと HL レジスタを合わせて、3 バイトで指定します。E レジスタが最上位バイトです。	
BC	PCM データの長さ	VRAM 指定時は、D レジスタと BC レジスタを合わせて、3 バイトで指定します。D レジスタが最上位バイトです。	
出力 Cy	0 正常終了 1 異常終了		
A	異常終了原因	1 サンプリング周波数指定誤り 2 STOP キーによる中断	
HL	中断時のアドレス	VRAM 指定時は、E レジスタと HL レジスタとを合わせた 3 バイトで返します。E レジスタが最上位バイトです。	
変更	すべて	Z80 モードの時は、自動的に R800 ROM モードに切り換えて実行し、終わると Z80 モードに戻ります。Z80 モードや R800 ROM モードのときに、サンプリングレートを 15.75kHz に指定すると、「サンプリング周波数指定誤り」エラーになります。 録音中は割り込みは禁止されます。なお、STOP キーが押されると中断します。 メイン RAM に録音するときは PCMPLY と同じように 8000H 以上でなければなりません。	

SUB ROM

BASIC 上から SUB ROM 内のルーチンと呼ぶときは以下のようにします。

```
LD IX,SUB ROM のコールアドレス
CALL EXTROM
```

DOS 上から SUB ROM を呼ぶときは MSX2 ではインタースロットコールは使用できません（暴走することがあります）。そのためかなり複雑な方法を取らなくてはなりません。その方法はデータバックやテクニカルガイドブック(ASCAT)などに載っていますので（どちらも方法が違いますが）そちらを参考にしてください。

なお、2+, turboR ではスロットが統一されているのでインタースロットコールしても問題ありません。

GRPPRT (0089H / SUB)		MSX2
機能	グラフィック画面に 1 文字出力します（スクリーン 5 ~ 8 のみで動作）。	
入力 A	文字コード	
出力	なし	
変更	なし	
NVBXLN (00C9H / SUB)		MSX2
機能	ボックスを描きます。	
入力 BC	始点の X 座標	
	始点の Y 座標	
DE	【GXPOS(FCB3H)】終点の X 座標	
	【GYPOS(FCB5H)】終点の Y 座標	
出力	【ATRBYT(F3F2H)】アトリビュート（色）	
	【LOGOPR(FB02H)】ロジカルオペレーションコード	
変更	なし	
NVBXFL (00CDH / SUB)		MSX2
機能	塗りつぶされたボックスを描きます。	
入力 BC	始点の X 座標	
	始点の Y 座標	
DE	【GXPOS(FCB3H)】終点の X 座標	
	【GYPOS(FCB5H)】終点の Y 座標	
出力	【ATRBYT(F3F2H)】アトリビュート（色）	

【LOGOPR(FB02H)】ロジカルオペレーションコード
出力 なし
変更 なし

CHGMOD (00D1H / SUB)		MSX2
機能	スクリーンモードを変更します。	
入力 A	スクリーンモード	
出力	なし	
変更	すべて	
INITXT (00D5H / SUB)		MSX2
機能	画面をテキストモード (20 × 40) にして初期化します。	
入力	【TXTNAM(F3B3H)】パターンネームテーブルのアドレス	
	【TXTCGP(F3B7H)】パターンジェネレーターテーブルのアドレス	
出力	なし	
変更	すべて	

INIT32 (00D9H / SUB)		MSX2
機能	画面をテキストモード (32 × 24) にして初期化します。	
入力	【T32NAM(F3BDH)】パターンネームテーブルのアドレス	
	【T32COL(F3BFH)】カラーテーブルのアドレス	
出力	【T32CGP(F3C1H)】パターンジェネレーターテーブルのアドレス	
	【T32ATR(F3C3H)】スプライトアトリビュートテーブルのアドレス	
変更	【T32PAT(F3C5H)】スプライトジェネレーターテーブルのアドレス	
出力	なし	
変更	すべて	

INIGRP (00DDH / SUB)		MSX2
機能	画面を高解像グラフィックモードにして初期化します。	
入力	【GRPNAM(F3C7H)】パターンネームテーブルのアドレス	
	【GRPCOL(F3C9H)】カラーテーブルのアドレス	
出力	【GRPCGP(F3CBH)】パターンジェネレーターテーブルのアドレス	
	【GRPATR(F3CDH)】スプライトアトリビュートテーブルのアドレス	
変更	【GRPPAT(F3CFH)】スプライトジェネレーターテーブルのアドレス	
出力	なし	
変更	すべて	

INIMLT (00E1H / SUB)		MSX2
機能	画面をマルチカラーモードにして初期化します。	
入力	【NLTNAM(F3D1H)】パターンネームテーブルのアドレス	
	【MLTCOL(F3D3H)】カラーテーブルのアドレス	
出力	【MLTCGP(F3D5H)】パターンジェネレーターテーブルのアドレス	
	【MLTATR(F3D7H)】スプライトアトリビュートテーブルのアドレス	
変更	【MLTPAT(F3D9H)】スプライトジェネレーターテーブルのアドレス	
出力	なし	
変更	すべて	

SETTXT (00E5H / SUB)		MSX2
機能	VDP をテキストモード (40 × 24) にします。	
入力	【TXTNAM(F3B3H)】パターンネームテーブルのアドレス	
	【TXTCGP(F3B7H)】パターンジェネレーターテーブルのアドレス	
出力	【LINL40(F3AEH)】1 行の幅 (WIDTH 文によって設定する値)	
出力	なし	
変更	すべて	

SETT32	(00E9H / SUB)	MSX2
機能	VDP をテキストモード (32×24) にします。	
入力	【T32NAM(F3BDH)】パターンネームテーブルのアドレス 【T32COL(F3BFH)】カラーテーブルのアドレス 【T32CGP(F3C1H)】パターンジェネレータテーブルのアドレス 【T32ATR(F3C3H)】スプライトアトリビュートテーブルのアドレス 【T32PAT(F3C5H)】スプライトジェネレータテーブルのアドレス 【LINL32(F3AFH)】1行の幅 (WIDTH 文によって設定する値)	
出力	なし	
変更	すべて	
SETGRP	(00EDH / SUB)	MSX2
機能	VDP を高解像度モードにします。	
入力	GRPNAM(F3C7H)】パターンネームテーブルのアドレス 【GRPCOL(F3C9H)】カラーテーブルのアドレス 【GRPCGP(F3CBH)】パターンジェネレータテーブルのアドレス 【GRPATR(F3CDH)】スプライトアトリビュートテーブルのアドレス 【GRPPAT(F3CFH)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
SETMLT	(00F1H / SUB)	MSX2
機能	VDP をマルチカラーモードにします。	
入力	【MLTNAM(F3D1H)】パターンネームテーブルのアドレス 【MLTCOL(F3D3H)】カラーテーブルのアドレス 【MLTCGP(F3D5H)】パターンジェネレータテーブルのアドレス 【MLTATR(F3D7H)】スプライトアトリビュートテーブルのアドレス 【MLTPAT(F3D9H)】スプライトジェネレータテーブルのアドレス	
出力	なし	
変更	すべて	
CLRSPR	(00F5H / SUB)	MSX2
機能	すべてのスプライトを初期化します。スプライトパターンをヌルに、スプライト番号をスプライト面番号に、スプライトの色を前景色にします。 スプライトの垂直位置は217にセットします。	
入力	【SCRMOD(FCAFH)】スクリーンモード	
出力	なし	
変更	すべて	
CALPAT	(00F9H / SUB)	MSX2
機能	スプライトジェネレータテーブルのアドレスを返します。 このルーチンはMAIN ROMの同名のBIOSと同じです。	
入力	A スプライト番号	
出力	HL アドレス	
変更	AF DE HL	
CALATR	(00FDH / SUB)	MSX2
機能	スプライトアトリビュートテーブルの開始アドレスを返します。このルーチンはMAIN ROMの同名のBIOSと同じです。	
入力	A スプライト番号	
出力	HL アドレス	
変更	AF DE HL	
GSPSIZ	(0101H / SUB)	MSX2
機能	現在のスプライトサイズを返します。このルーチンはMAIN ROMの同名のBIOSと同じです。	
入力	なし	
出力	A スプライトサイズ	

Cy 16×16のサイズの場合のみセット
変更 すべて

GETPAT	(0105H / SUB)	MSX2
機能	キャラクタパターンを返します。	
入力	A 文字コード	
出力	【PATWRK(FC40H)】キャラクタパターン	
変更	すべて	
WRTVRM	(0109H / SUB)	MSX2
機能	VRAM にデータを書き込みます。	
入力	HL 書き込むVRAMアドレス A 書き込む値	
出力	なし	
変更	AF	
RDVRM	(010DH / SUB)	MSX2
機能	VRAM の内容を読み出します。	
入力	HL 読み出すVRAMアドレス	
出力	A 読みだした値	
変更	F	
CHGCLR	(0111H / SUB)	MSX2
機能	画面の色を変えます。	
入力	A スクリーンモード 【FORCLR(F3E9H)】前景色 【BAKCLR(F3EAH)】背景色 【BDRCLK(F3EBH)】周辺色	
出力	なし	
変更	すべて	
CLSSUB	(0115H / SUB)	MSX2
機能	画面をクリアします。	
入力	なし	
出力	なし	
変更	すべて	
DSPFNK	(011DH / SUB)	MSX2
機能	ファンクションキーを表示します。	
入力	なし	
出力	なし	
変更	すべて	
WRTVDP	(012DH / SUB)	MSX2
機能	VDP レジスタにデータを書き込みます。	
入力	C VDP のレジスタ番号 B 書き込む値	
出力	なし	
変更	AF BC	
VDPSTA	(0131H / SUB)	MSX2
機能	VDP レジスタを獲得します。	
入力	A VDP のレジスタ番号 (0～9)	
出力	A 取得した値	
変更	F	
SETPAG	(013DH / SUB)	MSX2
機能	VRAM のページの切り換えます。	
入力	【DPPAGE(FAF5H)】ディスプレイページ番号 【ACPAGE(FAF6H)】アクティブページ番号	
出力	なし	
変更	AF	
INIPLT	(0141H / SUB)	MSX2
機能	パレットの初期化します。現在のパレットはVRAM にセーブされています。	
入力	なし	
出力	なし	
変更	AF BC DE	

RSTPLT (0145H / SUB) MSX2

機能 パレットを VRAM からリストアします。
入力 なし
出力 なし
変更 AF BC DE

GETPLT (0149H / SUB) MSX2

機能 パレットからカラーコードを獲得します。
入力 A パレット番号 (0 ~ 15)
出力 B 上位 4 ビットに赤のコード、
下位 4 ビットに青のコード
C 下位 4 ビットに緑のコード
変更 AF DE

SETPLT (014DH / SUB) MSX2

機能 カラーコードをパレットにセットします。
入力 D パレット番号
A 上位 4 ビットに赤のコード、
下位 4 ビットに青のコード
E 下位 4 ビットに緑のコード
出力 なし
変更 AF

BEEP (017DH / SUB) MSX2

機能 ブザーを鳴らします。
入力 なし
出力 なし
変更 すべて

PROMPT (0181H / SUB) MSX2

機能 プロンプトを表示します。
入力 なし
出力 なし
変更 すべて

NEWPAD (01ADH / SUB) MSX2

機能 マウス・ライトペンの状態を調べます。
入力 A 以下のデータをいれてコールします。

8	ライトペンの接続状態を返す (FFH で有効)
9	X 座標を返す
10	Y 座標を返す
11	ライトペンスイッチの状態を返す (押されたとき FFH)
12	必ず FFH (接続されている)
13	X 方向のオフセットを返す
14	Y 方向へのオフセットを返す
15	常に 0
16	必ず FFH (接続されている)
17	X 方向のオフセットを返す
18	Y 方向のオフセットを返す
19	常に 0

出力 A
変更 すべて
補足 MSXturboR ではライトペンは値を返しません。

CHGMDP (01B5H / SUB) MSX2

機能 VDP のモードを変えます。
パレットは初期化します。
入力 A スクリーンモード
出力 なし
変更 すべて

KNJPRT (01BDH / SUB) MSX2

機能 グラフィック画面(スクリーン 5 ~ 8)に漢字を表示します。
入力 BC JIS 漢字コード (MSX2 は第一水準のみ、2+・turboR は第二水準もサポート)

A 表示モード

0 16 × 16 ドットで表示
1 偶数番目のドットを表示
2 奇数番目のドットを表示

【GRPACX(FCB7H)】X 座標

【GRPACY(FCB9H)】Y 座標

【ATRBYT(F3F2H)】色

【LOGOPR(FB02H)】ロジカルオペレーション

出力 なし

変更 AF

REDCLK (01F5H / SUB) MSX2

機能 CLOCK-IC を読み出します。
入力 C 読み込む CLOCK-IC のレジスタアドレス
00MMAAAA
||++++----アドレス (0 ~ 15)
++-----モード (0 ~ 3)

出力 A 読みだした値 (下位 4 ビットのみ有効)

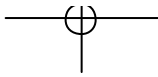
変更 F

WRTCLK (01F9H / SUB) MSX2

機能 クロックデータを書き込みます。
入力 C クロック RAM アドレス
A 書き込む値

出力 なし

変更 F



2章 ワークエリア



アドレス	ラベル	長さ	初期値	内容
------	-----	----	-----	----

ディスクドライブ用 (ディスクがある時のみ有効)

F323H	DISKVE	2		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
F325H	BREAKV	2		CTRL+C 処理ルーチンへのポインタへのポインタ
F341H	RAMAD0	1		ページ0の RAM のスロット
F342H	RAMAD1	1		ページ1の RAM のスロット
F343H	RAMAD2	1		ページ2の RAM のスロット
F344H	RAMAD3	1		ページ3の RAM のスロット
F348H	MASTERS	1		マスターカートリッジのスロット
F34BH	DOSIM	2		DOS の HIMEM

インタースロットコール用サブルーチン

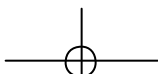
F380H	RDPRIM	5		基本スロットから読み込み
F385H	WRPRIM	7		基本スロットへ書き込み
F38CH	CLPRIM	14		基本スロットコール

USR 関数のマシン語プログラム開始番地、テキスト画面

F39AH	USRTAB	20	475AH	USR 関数(0~9)のマシン語プログラムの開始番地。定義前はエラールーチン FCERR(475AH)を指す
F3AEH	LINL40	1	39	SCREEN0 のときの 1 行の幅。WIDTH 文により設定
F3AFH	LINL32	1	29	SCREEN1 のときの 1 行の幅。WIDTH 文により設定
F3B0H	LINLEN	1	29	現在の画面の 1 行の幅
F3B1H	CRTCNT	1	24	現在の画面の行数
F3B2H	CLMLST	1	14	PRINT 文の制御に使用。LINENE-(LINLEN MOD 14)-14

VRAM の各テーブルの番地、その他の画面設定

F3B3H	TXTNAM	2	0000H	SCREEN0 のパターンネームテーブル
F3B5H	TXTCOL	2	0800H	SCREEN0 のカラーテーブル
F3B7H	TXTCGP	2	0800H	SCREEN0 のパターンジェネレータテーブル
F3B9H	TXTART	2		未使用
F3BBH	TXTPAT	2		未使用
F3BDH	T32NAM	2	1800H	SCREEN1 のパターンネームテーブル
F3BFH	T32COL	2	2000H	SCREEN1 のカラーテーブル
F3C1H	T32CGP	2	0000H	SCREEN1 のパターンジェネレータテーブル
F3C3H	T32ATR	2	1B00H	SCREEN1 のスプライトアトリビュートテーブル
F3C5H	T32PAT	2	3800H	SCREEN1 のスプライトジェネレータテーブル
F3C7H	GRPNAM	2	1800H	SCREEN2 のパターンネームテーブル
F3C9H	GRPCOL	2	2000H	SCREEN2 のカラーテーブル
F3CBH	GRPCGP	2	0000H	SCREEN2 のパターンジェネレータテーブル
F3CDH	GRPATR	2	1B00H	SCREEN2 のスプライトアトリビュートテーブル
F3CFH	GRPPAT	2	3800H	SCREEN2 のスプライトジェネレータテーブル
F3D1H	MLTNAM	2	0800H	SCREEN3 のパターンネームテーブル
F3D3H	MLTCOL	2		未使用
F3D5H	MLTCGP	2	0000H	SCREEN3 のパターンジェネレータテーブル
F3D7H	MLTATR	2	1B00H	SCREEN3 のスプライトアトリビュートテーブル
F3D9H	MLTPAT	2	3800H	SCREEN3 のスプライトジェネレータテーブル
F3DBH	CLIKSW	1	FFH	キークリックスイッチ (0:OFF、0 以外:ON)
F3DCH	CRRY	1		カーソルの Y 座標
F3DDH	CSRX	1		カーソルの X 座標
F3DEH	CNSDFG	1	FFH	ファンクションキーの表示スイッチ (0:非表示、0 以外:表示)



VDP レジスタのセーブエリアなど

F3DFH	RG0SAV	1		VDP レジスタ0 の値
F3E0H	RG1SAV	1		VDP レジスタ1 の値
F3E1H	RG2SAV	1		VDP レジスタ2 の値
F3E2H	RG3SAV	1		VDP レジスタ3 の値
F3E3H	RG4SAV	1		VDP レジスタ4 の値
F3E4H	RG5SAV	1		VDP レジスタ5 の値
F3E5H	RG6SAV	1		VDP レジスタ6 の値
F3E6H	RG7SAV	1		VDP レジスタ7 の値
F3E7H	STATFL	1		ステータスレジスタ0 の内容 (MSX1 では VDP のステータス)
F3E8H	TRGFLG	1	FFH	ジョイスティックのトリガボタンの状態 (押された時、対応する bit が 0 になる) bit7 Trig.B (Port2) bit6 Trig.A (Port2) bit5 Trig.B (Port1) bit4 Trig.A (Port1) bit3 予約 bit2 予約 bit1 予約 bit0 Space key
F3E9H	FORCLR	1	15	前景色 (フォアグラウンドカラー) 省略値
F3EAH	BAKCLR	1	4	背景色 (バックグラウンドカラー) 省略値
F3EBH	BDRCLR	1	7	周辺色 (ボーダーカラー) 省略値
F3ECH	MAXUPD	3	JP 0000H	LINE 文が内部で使用 (C3H、00H、00H)
F3EFH	MINUPD	3	JP 0000H	LINE 文が内部で使用 (C3H、00H、00H)
F3F2H	ATRBYT	1	15	グラフィック使用時の色。アトリビュートバイト
F3F3H	QUEUES	2	F959H	PLAY 文実行時のキューテーブルを指す
F3F5H	FRCNEW	1	FFH	BASIC インタープリタが内部で使用
F3F6H	SCNCNT	1	1	キースキャンの時間間隔
F3F7H	REPCNT	1	50	キーのオートリピート開始までの時間間隔
F3F8H	PUTPNT	2	FBF0H	キーバッファへの書き込みを行う番地を指す
F3FAH	GETPNT	2	FBF0H	キーバッファへからの読み込みを行う番地を指す (KEYBUF)

カセット用パラメータなど

F3FCH	CS120	10	初期値 83	1200 ボーのビット 0 を表す LOW の幅
			初期値 92	1200 ボーのビット 0 を表す HIGH の幅
			初期値 38	1200 ボーのビット 1 を表す LOW の幅
			初期値 45	1200 ボーのビット 1 を表す HIGH の幅
			初期値 15	ショートヘッダビットの長さ、HEADLEN:(1200) × 2 / 256
			初期値 37	2400 ボーのビット 0 を表す LOW の幅
			初期値 45	2400 ボーのビット 0 を表す HIGH の幅
			初期値 14	2400 ボーのビット 1 を表す LOW の幅
			初期値 22	2400 ボーのビット 1 を表す HIGH の幅
			初期値 31	ショートヘッダビットの長さ、HEADLEN:(1200) × 2 / 256
F406H	LOW	2	83	現在のボーレートのビット 0 を表す LOW と HIGH の幅。 SCREEN 文で設定
F408H	HIGH	2	33	現在のボーレートのビット 1 を表す LOW と HIGH の幅。 SCREEN 文で設定
F40AH	HEADER	1	15	現在のボーレートのショートヘッダ用のヘッダビット (HEDLEN:2000)、SCREEN 文で設定
F40BH	ASPCT1	2	1	256 / アスペクト比、SCREEN 文で設定し CIRCLE 文で使用
F40DH	ASPCT2	2	1	256 / アスペクト比、SCREEN 文で設定し CIRCLE 文で使用
F40FH	ENDPRG	5	":."	RESUME NEXT 文のための仮のプログラムの終わり

BASIC が内部で使うワーク

F414H	ERRFLG	1	0	エラーコードの保存
F415H	LPTPOS	1	0	プリンタのヘッド位置
F416H	PRTFLG	1	0	プリンタへ出力するかどうか
F417H	NTMSXP	1	0	プリンタの種別 0=MSX 用プリンタ、0 以外=MSX 用プリンタではない)
F418H	RAWPRT	1	0	0 以外なら RAW MODE (文字コード変換なし) でプリンタ出力 (TAB やシフト JIS → JIS コード変換機能の禁止)
F419H	VLZADR	2	0	VAL 関数で置き換えられる文字のアドレス
F41BH	VLZDAT	1	0	VAL 関数で 0 に置き換わる文字
F41CH	CURLIN	2	FFFFH	BASIC が現在実行中の行番号
F41FH	KBUF	318		クランチバッファ、BUF(F55EH) から中間言語に直されて入る
F55DH	BUFMIN	1	":."	INPUT 文用のコマ
F55EH	BUF	258		タイプした文字が入るバッファ。ダイレクトステートメントがアスキーコードで入る

F660H	ENDBUF	1	0	BUF(F55EH)の内容がオーバーフローするを防ぐ
F661H	TTYPOS	1	0	BASIC が内部で持つ仮想的なカーソル位置
F662H	DIMFLG	1		BASIC が配列変数を単純変数と区別するためのフラグ
F663H	VALTYP	1	0	変数の型の識別に使用
F664H	OPRTYP	1	0	演算子の保存、またはクランチできる語かどうかのフラグ
F665H	DONUM	1	0	クランチ用のフラグ
F666H	CONXT	2	0	CHGET で使うテキストポインタの保存
F668H	CONSAV	1	0	CHGET が呼ばれた後で定数の内部形 (トークン) を保存
F669H	CONTYP	1	0	保存した定数のタイプ
F66AH	CONLO	8	0	保存した定数の値
F672H	MEMSIZ	2		BASIC が使用するメモリの最上位番地
F674H	STKTOP	2		スタックとして使える最上位番地。CLEAR 文で変更される
F676H	TXTTAB	2		BASIC テキストエリアの先頭番地
F678H	TEMPPT	2	F67AH	テンポラリディスクリプタの空きエリアの先頭番地
F67AH	TEMPST	30		3 × NUMTMP 分使用される
F698H	DSCTMP	3		文字列関数の答えのストリングディスクリプタ
F69BH	FRETOP	2		文字列領域の空きエリアの先頭番地
F69DH	TEMP3	2	0	ガベージコレクション用
F69FH	TEMP8	2	0	ガベージコレクション用
F6A1H	ENDFOR	2	0	FOR 文の次の番地 (ループ時に FOR 文の次から実行するため)
F6A3H	DATLIN	2	0	READ 文で読まれた DATA 文の次の行番号
F6A5H	SUBFLG	1	0	USR 関数などで配列を使うときのフラグ
F6A6H	FLGINP	1	0	INPUT 文や READ 文で使われるフラグ
F6A7H	TEMP	2		ステートメントコード用の一時保存場所
F6A9H	PTRFLG	1	0	変換する行番号がなければ 0、あれば 0 以外
F6AAH	AUTFLG	1	0	AUTO コマンド有効・無効フラグ (0 以外=有効中、0=無効中)
F6ABH	AUTLIN	2	0	AUTO で入力された最新の行番号
F6ADH	AUTINC	2	0	AUTO コマンドの行番号の増分
F6AFH	SAVTXT	2		RESUME 文で復帰するためのテキストのアドレス
F6B1H	SAVSTK	2		エラーが起きたとき回復ルーチン用のスタックの保存
F6B3H	ERRLIN	2	0	エラーが起きたときの行番号
F6B5H	DOT	2	0	現在の行番号、LIST.などで使う
F6B7H	ERRTXT	2	0	エラーが起きた行番号。RESUME 文用
F6B9H	ONELIN	2	0	エラーが起きたときの飛び先の行
F6BBH	ONEFLG	1	0	エラーによる割り込みルーチン実行中は 1、他の場合は 0
F6BCH	TEMP2	2	0	一時保存用
F6BEH	OLDLIN	2	0	CTRL+STOP、STOP 命令、END 命令で中断されたか、あるいは最後に実行された行番号
F6C0H	OLDTXT	2	0	次に実行する文のテキストアドレス
F6C2H	VARTAB	2		単純変数の開始番地。NEW 文で TXTTAB(F676H)+2 に設定される
F6C4H	ARYTAB	2		配列テーブルの開始番地
F6C6H	STREND	2		BASIC がテキストや変数に使用中のメモリの最後の番地
F6C8H	DATPTR	2	0	READ 文実行で読まれたデータのテキストアドレス
F6CAH	DEFTBL	26	8	各英文字で始まる変数のデフォルトの型を保持する

ユーザー関数のパラメータ用ワーク

F6E4H	PRMSTK	2	0	ガベージコレクション用スタック上の以前の定義のブロック
F6E6H	PRMLN	2	0	処理対象のテーブルのバイト数
F6E8H	PARM1	100	0	PRMSIZ で設定される処理パラメータ定義テーブル
F74CH	PRMPRV	2	PRMSTK	以前のパラメータブロックのポインタガベージコレクション用
F74EH	PRMLN2	2	0	パラメータブロックの大きさ
F750H	PARM2	100	0	PRMSIZ で設定されるパラメータ保存場所
F7B4H	PRMFLG	1	0	PARM1 がサーチ済みかどうかを示すフラグ
F7B5H	ARYTA2	2	0	サーチの終点
F7B7H	NOFUNS	1	0	処理対象関数がない場合は 0
F7B8H	TEMP9	2	0	ガベージコレクション用
F7BAH	FUNACT	2	0	処理対象関数の数
F7BCH	SWPTMP	8	0	SWAP 文の最初の変数の値の一時保存場所
F7C4H	TRCFLG	1	0	トレースフラグ。0=TRACE OFF、0 以外=トレース ON

Math-pack 用ワークエリア

F7C5H	FBUFR	43	Math-pack ルーチン一時保存場所として使用する
F7F0H	DECTMP	2	10 進数を浮動小数点にするときに使用する
F7F2H	DECTM2	2	除算ルーチンで使用する
F7F4H	DECCNT	1	除算ルーチンで使用する
F7F6H	DAC	16	演算の対象となる値を設定するエリア
F806H	HOLD8	48	10 進数の乗算のためのレジスタ保存エリア
F836H	HOLD2	8	Math-pack ルーチンが内部で使用する
F83EH	HOLD	8	Math-pack ルーチンが内部で使用する
F847H	ARG	16	DAC(0F7F6H)と演算対象となる値を設定するエリア
F857H	RNDX	8	最新の乱数を倍精度実数で設定するエリア

BASIC インタープリタが使うデータエリア

F85FH	MAXFIL	1	1	MAXFILES 文により設定されるファイル番号の最大値
F860H	FILTAB	2		ファイルデータの先頭アドレス
F862H	NULBUF	2		SAVE、LOAD、で BASIC インタープリタが使用するバッファ
F864H	PTRFIL	2	0	指定ファイルのファイルデータのある位置
F866H	RUNFLG	0	0	ロード後実行するなら 0 でない値
F866H	FILNAM	11	""	ファイル名の保存エリア
F871H	FILNM2	11	""	ファイル名の保存エリア
F87CH	NLONLY	1	0	プログラムロード中は 0 でない値
F87DH	SAVEND	2	0	セーブするマシン後プログラムの最終アドレス
F87FH	FNKSTR	160		ファンクションキーの文字列保存エリア (16 文字 × 10)
F91FH	CGPNT	3		文字フォント格納スロットとアドレス
F922H	NAMBAS	2		現在のパターンネームテーブルのベースアドレス
F924H	CGBAS	2		現在のパターンネームテーブルのベースアドレス
F926H	PATBAS	2		現在のスプライトジェネレーターテーブルのベースアドレス
F928H	ATRBAS	2		現在のスプライトアトリビュートテーブルのベースアドレス
F92AH	CLOC	2		グラフィックルーチンが内部で使用する
F92CH	CMASK	1		グラフィックルーチンが内部で使用する
F92DH	MINDEL	2		グラフィックルーチンが内部で使用する
F92FH	MAXDEL	2		グラフィックルーチンが内部で使用する

CIRCLE、PAINT 分で使うデータエリア

F931H	ASPECT	2		円の縦横の比率。CIRCLE 文の>比率 により設定される
F933H	CENCNT	2		CIRCLE 文が内部で使用する
F935H	CLINEF	1		中心へ線を引くかどうかのフラグ
F936H	CNPNTS	2		プロットする点
F938H	CPLOTF	1		CIRCLE 文が内部で使用する
F939H	CPCNT	2		円の 1 / 8 分割の数
F93BH	CPCNT8	2		CIRCLE 文が内部で使用する
F93DH	CRCSUM	2		CIRCLE 文が内部で使用する
F93FH	CSTCON	2		CIRCLE 文が内部で使用する
F941H	CSCLXY	1		X と Y のスケール
F942H	CSAVEA	2		ADVGRP の保存エリア
F944H	CSAVEM	1		ADVGRP の保存エリア
F945H	CXOFF	2		中心から X のオフセット
F947H	CYOFF	2		中心から Y のオフセット
F949H	LOHMSK	1		PAINT 文が内部で使用する
F94AH	LOHDIR	1		PAINT 文が内部で使用する
F94BH	LHOADR	2		PAINT 文が内部で使用する
F94DH	LOHCNT	2		PAINT 文が内部で使用する
F94FH	SKPCNT	2		スキップカウント
F951H	MIVCNT	2		移動カウント
F953H	DIREC	1		ペイントの方向
F954H	LFPROG	1		PAINT 文が内部で使用する
F955H	RTPROG	1		PAINT 文が内部で使用する

PLAY 文で使用するデータエリア

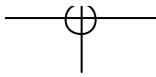
F956H	MCLTAB	2		PLAY 文マクロ、DROW 文マクロのテーブルの先頭を指す
F958H	MCLFLG	1		PLAY-DRAW の指示
F959H	QUETAB	24		キューテーブル、4 キュー × 6 バイト
F971H	QUEBAK	4		BCKQ で使用
F975H	VOICAQ	n		n は MUSQLN、音声 1 のキュー
F9F5H	VOICBQ	n		n は MUSQLN、音声 2 のキュー
FA75H	VOICCQ	n		n は MUSQLN、音声 3 のキュー

MSX2 で追加されたワークエリア

FAF5H	DPPAGE	1	ディスプレイページ番号
FAF6H	ACPAGE	1	アクティブページ番号
FAF7H	AVCSAV	1	AV コントロールポートの保存
FAF8H	EXBRSA	1	SUB-ROM のスロットアドレス
FAF9H	CHRCNT	1	ローマ字カナ変換で使用、キャラクタカウンタで 0~2
FAFAH	ROMA	2	ローマ字ローマ字カナ変換で使用。バッファ中のキャラクタをいれて置くエリア
FAFCH	MODE	1	ローマ字カナ変換モード、VRAM のサイズなど
			b7 6 5 4 3 2 1 0 K 0 0 0 W V V C +---- 0=ローマ字変換しない、1=する +----- 00=VRAM16K、01=VRAM64K 10=VRAM128K +----- 0=マスクする、1=マスクしない (SCREEN0 ~ 3 の VRAM アドレス) +----- 0=クリッピングする、1=しない +----- 0=SCREEN10、1=SCREEN11 (MSX2+で RGB 処理のフラグ) +----- 0=第 2 水準漢字 ROM なし、1=あり +----- 0=ひらがな (ローマ字変換) 1=カタカナ
			注意 ビット 3 は MSX2 以降で SCREEN0 ~ 3 の VRAM アドレスを指定する際に、3FFFF と AND をとって指定するかどうかのフラグ。SCREEN4 以降では常にマスクしない
FAFDH	NORUSE	1	漢字ドライバが使用
FAFEH	XSAVE	2	I0000000,XXXXXXX I=1 ライトペンのインタラプト要求あり XXXXXXXX:X座標 × 0000000,YYYYYYYY 0000000=符号無しオフセット YYYYYYYY=Y 座標
FB00H	YSAVE	2	
FB02H	LOGOPR	1	V9938 以降のロジカルオペレーションコード

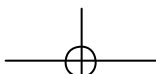
RS-232C で使うデータエリア

FB03H	RSTMP	50	RS-232C またはディスクが使用する
FB03H	TOCNT	1	RS-232C が内部で使用する
FB04H	RSFCB	2	RS-232C の LOW アドレス
			RS-S3SC の HIGH アドレス
FB06H	RSIQLN	1	RS-232C が内部で使用する
FB07H	MEXBIH	5	FB07H+0 RST 30H(0F7H) FB07H+1 バイトデータ FB07H+2 (LOW) FB07H+3 (HIGH) FB07H+4 RET(0C9H)
FB0CH	OLDSTT	5	FB0CH+0 RST 30H(0F7H) FB0CH+1 バイトデータ FB0CH+2 (LOW) FB0CH+3 (HIGH) FB0CH+4 RET(0C9H)
FB11H	OLDINT	5	FB11H+0 RST 30H(0F7H) FB11H+1 バイトデータ FB11H+2 (LOW) FB11H+3 (HIGH) FB11H+4 RET(0C9H)
FB16H	DEVNUM	1	RS-232C が内部で使用する
FB17H	DEV CNT	3	FB17H+0 バイトデータ FB17H+1 バイトポインタ FB17H+2 バイトポインタ
FB1AH	ERRORS	1	RS-232C が内部で使用する
FB1BH	FLAGS	1	RS-232C が内部で使用する
FB1CH	ESTBLS	1	RS-232C が内部で使用する
FB1DH	COMMSK	1	RS-232C が内部で使用する
FB1EH	LSTCOM	1	RS-232C が内部で使用する
FB1FH	LSTMOD	1	RS-232C が内部で使用する
FB20H	HOKVLD	1	拡張 BIOS の有無
FB21H	DRV TBL	8	DISK-ROM のスロットアドレスなど



PLAY 文で使用するデータエリア (以下は MSX1 と共通)

FB35H	PRSCNT	1	
FB36H	SAVSP	2	PLAY 文実行中にスタックポインタを保存
FB38H	VOICEN	1	解釈中の現在の音声
FB39H	SAVVOL	2	休止のために音量を保存
FB3BH	MCLLEN	1	PLAY 文が内部で使用する
FB3CH	MCPTR	2	PLAY 文が内部で使用する
FB3EH	QUEUEN	1	PLAY 文が内部で使用する
FC3FH	MUSICF	1	音楽演奏用の割り込みフラグ
FB40H	PLYCNT	1	キューに格納されているPLAY文の数
FB41H	VCBA	n	n は VCBSIZ、音声1 のスタティックデータ
FB66H	VCBB	n	n は VCBSIZ、音声2 のスタティックデータ
FB8BH	VCBC	n	n は VCBSIZ、音声3 のスタティックデータ
FB80H	ENSTOP	1	0 以外は「SHIFT+CTRL+GRAPH+カナ」でウォームスタート可能
FB81H	BASROM	1	BASIC テキストの存在場所を示す(0:RAM 上、0 以外:ROM 上)
FB82H	LINTTB	24	ラインターミナルテーブル。テキスト画面の各行の情報を保持するエリア
FBCAH	FSTPOS	2	LININ(00B1H / MAIN)で入力した行の最初の文字の位置
FBCCH	CODSAV	1	カーソルが重なった部分のキャラクタを保存するエリア
FBCDH	FNKSWI	1	KEY ON 時にどのファンクションキーが表示されているか Q:F1 ~ F5 0:F6 ~ F10 が表示)
FBCFH	FNKFLG	10	ON KEY GOSUB 文により定義された行の実行を許可、禁止、停止するかファンクションキーごとに保存するためのエリア
FBD8H	ONGSBF	1	TRPTBL(0FC4CH)で待機中のイベントが発生したかどうかのフラグ
FBD9H	CLIKFL	1	キークリックフラグ
FBD9H	OLDKEY	11	キーマトリクスの状態(旧)
FBE5H	NEWKEY	11	キーマトリクスの状態(新)
FBF0H	KEYBUF	40	キーコードバッファ
FC18H	BUFEND	0	KEYBUFの終わり
FC18H	LINWRK	40	スクリーンハントラ用一時保存場所
FC40H	PATWRK	8	パターンコンバータ用一時保存場所
FC48H	BOTTOM	2	実装した RAM の先頭(低位)アドレス
FC4AH	HIMEM	2	利用可能なメモリの上位番地
FC4CH	TRPTBL	n	n は 3 × NUMTPR。割り込み処理で使うトラップテーブル
FC4CH	Function key [1]		(ON KEY GOSUB)
:	:		
FC67H	Function key [10]		(ON KEY GOSUB)
FC6AH	[CTRL]+[STOP]		(ON STOP GOSUB)
FC6DH	Sprite 衝突		(ON SPRITE GOSUB)
FC70H	Space key		(ON STRIG GOSUB)
FC73H	Trig.A [1]		(ON STRIG GOSUB)
FC76H	Trig.A [2]		(ON STRIG GOSUB)
FC79H	Trig.B [1]		(ON STRIG GOSUB)
FC7CH	Trig.B [2]		(ON STRIG GOSUB)
FC7FH	1/60sec		(ON INTERVAL GOSUB)
FC82H	予約 [1]		
:	:		
FC91H	予約 [6]		
FC94H	システム予約 [1]		(ユーザーの使用は禁止)
FC97H	システム予約 [2]		(ユーザーの使用は禁止)
FC4C+3n +0	フラグ		
	bit7 ~ 3	予約	
	bit2	トラップの実行要求中は「1」	
		実際に実行されると「0」に変化	
	bit1	device STOP で「1」	
	bit0	device ON で「1」 / device OFF で「0」	
+1	トラップ先の行アドレス(下位)		
+2	トラップ先の行アドレス(上位)		
FC9AH	RTYCNT	1	BAISC が内部で使用する
FC9BH	INTFLG	1	CTRL+STOP が押された場合など、ここに 03H をいれることによりストップする
FC9CH	PADY	1	パドルの Y 座標
FC9DH	PADX	1	パドルの X 座標
FC9EH	JIFFY	1	PLAY 文が内部で使用する
FCA0H	INTVAL	2	インターバルの期間。ON INTERVAL GOSUB 文により設定される
FCA2H	INTCNT	2	インターバルのためのカウンタ
FCA4H	LOWLIM	2	カセットテープからの読み込み中に使う
FCA5H	WINWID	1	カセットテープからの読み込み中に使う
FCA6H	GRPHED	1	グラフィック文字を出すときのフラグ
FCA7H	ESCCNT	1	エスケープシーケンスのカウンタ
FCA8H	INSFLG	1	挿入モードフラグ (0=通常モード、0 以外=挿入モード)



FCA9H	CSRW	1	カーソル表示の有無 0=表示無し、0 以外=表示あり)
FCAAH	CSTYLE	1	カーソルの型 0=■、0 以外=□)
FCABH	CAPST	1	CAPS キーの状態 0=CAP OFF、0 以外=CAP ON)
FCACH	KANAST	1	かなキーの状態 0=かな OFF、0 以外=かな ON)
FCADH	KANAMD	1	かなキー配列の状態 0=50 音配列、0 以外=JIS 配列)
FCAEH	FLBMEM	1	BASIC プログラムをロード中は0
FCAFH	SCMROD	1	現在のスクリーンモードの番号
FCB0H	OLDSCR	1	スクリーンモード保存場所
FCB1H	CASPRV	1	CAS: が使う文字保存場所
FCB2H	BRDATR	1	PAINT で使用する境界色のカラーコード
FCB3H	GXPOS	2	X 座標
FCB5H	GYPOS	2	Y 座標
FCB7H	GRPACX	2	グラフィックアキュムレータ (X 座標)
FCB9H	GRPACY	2	グラフィックアキュムレータ (Y 座標)
FCBBH	DRWFLG	1	DRAW 文で使用するフラグ
FCBCH	DRWSCL	1	DRAW スケーリングファクタ 0=スケーリングしない、0 以外=する)
FCBDH	DRWANG	1	DRAW の角度。0 ~ 3
FCBEH	RUNBUF	1	BLOAD 中、BSAVE 中、どちらでもない、のいずれかのフラグ
FCBFH	SAVENT	2	BSAVE の開始番地
FCC1H	EXPTBL	4	各スロットの拡張の有無を示すフラグテーブル
FCC5H	SLTTBL	4	各拡張スロットレジスタ用の現在のスロット選択状況
FCC9H	SLTATR	64	各スロット用に属性を保存
FD09H	SLTWRK	128	各スロット用に特定のワークエリアを確保
FD89H	PROCNM	16	CALL 文による拡張文の名前。0 は終わり
FD99H	DEVICE	1	カートリッジ用の装置識別に使う
FD9AH	H.KEYI	5	割り込み処理の始め
FD9FH	H.TIMI	5	タイマー割り込み処理の追加
FDA4H	H.CHPH	5	CHPUT (L 文字出力) の始め
FDA9H	H.DSPC	5	DSPCSR (カーソル表示) の始め
FDAEH	H.ERAC	5	ERACSR (カーソル消去) の始め
FDB3H	H.DSPF	5	DSPFNK (ファンクションキー表示) の始め
FDB8H	H.ERAF	5	ERAFNK (ファンクションキー消去) の始め
FDBDH	H.TOTE	5	TXTENT (画面をテキストモードにする) の始め
FDC2H	H.CHGE	5	CHGET (L 文字取り出し) の始め
FDC7H	H.INIP	5	INIPAT (文字バターンの初期化) の始め
FDCCH	H.KEYC	5	KEYCOD (キーコード変換) の始め
FDD1H	H.KYEA	5	KYEASY (KEY EASY) の始め
FDD6H	H.NMI	5	NMI (ノンマスカブルインタラプト) の始め
FDDBH	H.PINL	5	PINLIN (プログラム行入力) の始め
FDE0H	H.OINL	5	QINLIN (「?」を表示して行入力) の始め
FDE5H	H.INLI	5	INLIN (行入力) の始め
FDEAH	H.ONGO	5	ONGOTP (ON GOTO) の始め

ディスク装置接続

FDEFH	H.DSKO	5	DSKO\$ (ディスク出力) の始め
FDF4H	H.SETS	5	SETS\$ (セットアトリビュート) の始め
FDF9H	H.NAME	5	NAME (リネーム) の始め
FDFEH	H.KILL	5	KILL (ファイル削除) の始め
FE03H	H.IPL	5	IPL (初期プログラムロード) の始め
FE08H	H.COPY	5	COPY (ファイルのコピー) の始め
FE0DH	H.CMD	5	CMD (コマンド) の始め
FE12H	H.DSKF	5	DSKF (ディスクの空き) の始め
FE17H	H.DSKI	5	DSKI (ディスク入力) の始め
FE1CH	H.ATTR	5	ATTR\$ (アトリビュート) の始め
FE21H	H.LSET	5	LSET (左詰め代入) の始め
FE26H	H.RSET	5	RSET (右詰め代入) の始め
FE2BH	H.FIEL	5	FIELD (フィールド) の始め
FE30H	H.MKIS	5	MKIS (整数作成) の始め
FE35H	H.MKS\$	5	MKS\$ (単数度作成) の始め
FE3AH	H.MKD\$	5	MKD\$ (倍精度作成) の始め
FE3FH	H.CVI	5	CVI (整数変換) の始め
FE44H	H.CVS	5	CVS (単数度変換) の始め
FE49H	H.CVD	5	CDV (倍精度変換) の始め
FE4EH	H.GETP	5	GETPTR (ファイルポインター取り出し)
FE53H	H.SETF	5	SETFIL (ファイルポインター設定)
FE58H	H.NOFO	5	NOFOR (OPEN 文に FOR が無い)
FE5DH	H.NULO	5	NULOPN (空ファイルオープン)
FE62H	H.NTFI	5	NTFILO (ファイル番号が 0 でない)
FE67H	H.MERG	5	MERGE (プログラムファイルのマージ)
FE6CH	H.SAVE	5	SAVE (セーブ)

FE71H	H.BINS	5	BINSAV (機械語セーブ)
FE76H	H.BINL	5	BINLOD (機械語ロード)
FE7BH	H.FILE	5	FILES (ファイル一覧表示)
FE80H	H.DGET	5	DGET (ディスクGET)
FE85H	H.FILO	5	FILOU1 (ファイル出力)
FE8AH	H.INDS	5	INDSKC (ディスクの属性を入力)
FE8FH	H.RSLF	5	(前のドライブを再び選択する)
FE94H	H.SAVD	5	(現在選択しているドライブを保存する)
FE99H	H.LOC	5	LOC 関数、場所を示す)
FE9EH	H.LOF	5	LOF 関数、ファイルの長さ)
FEA3H	H.EOF	5	EOF 関数、ファイルの終わり)
FEA8H	H.FPOS	5	FPOS 関数、ファイルの場所)
FEADH	H.BAKU	5	BAKUPT (バックアップ)

論理装置名の拡張

FEB2H	H.BAKU	5	PARDEV (装置名の取り出し)
FEB7H	H.PARD	5	NODEVN (装置名なし)
FEBCH	H.POSD	5	POSDSK (ディスク装置)
FEC1H	H.DEVN	5	DEVNAM (装置名の処理)
FEC6H	H.GEND	5	GENDSP (装置割り当て)

BASIC 内部で使用

FECBH	H.RUNC	5	RUNC (RUN のためのクリア)
FED0H	H.CLEA	5	CLEARC (CLEAR のためのクリア)
FED5H	H.LOPD	5	LOPDFT (繰り返しと省略値設定)
FEDAH	H.STKE	5	STKERR (スタックエラー)
FEDFH	H.ISFL	5	ISFLIO (ファイルの入出力かどうか)
FEE4H	H.OUTD	5	OUTDO (OUT を実行)
FEE9H	H.CRDO	5	CRDO (CRLF を実行)
FEEDH	H.DSKC	5	DSKCHI (ディスクの属性を入力)
FEF3H	H.DOGR	5	DOGRPH (グラフィックを実行)
FEF8H	H.PRGE	5	PRGEND (プログラム終了)
FEFDH	H.HRRP	5	ERRPRT (エラー表示)
FF02H	H.ERRF	5	
FF07H	H.READ	5	READY
FF0CH	H.MAIN	5	MAIN
FF11H	H.DIRD	5	DIRDO (ダイレクトステートメント実行)
FF16H	H.FINI	5	
FF1BH	H.FINE	5	
FF20H	H.CRUN	5	
FF25H	H.CRUS	5	
FF2AH	H.ISRE	5	
FF2FH	H.NTFN	5	
FF34H	H.NOTR	5	
FF39H	H.SNGF	5	
FF3EH	H.NEWS	5	
FF43H	H.GONE	5	
FF48H	H.CHRG	5	
FF4DH	H.RETU	5	
FF52H	H.PRTF	5	
FF57H	H.COMP	5	
FF5CH	H.FINP	5	
FF61H	H.TRMN	5	
FF66H	H.FRME	5	
FF6BH	H.NTPL	5	
FF70H	H.EVAL	5	
FF75H	H.OKNO	5	
FF7AH	H.FING	5	
FF7FH	H.ISMI	5	ISMID\$ (MID\$かどうか)
FF84H	H.WIDT	5	WIDTHS (WIDTH)
FF89H	H.LIST	5	LIST
FF8EH	H.BUFL	5	BUFLIN (バッファライン)
FF93H	H.FRQI	5	FRQINT
FF98H	H.SCNE	5	
FF9DH	H.FRET	5	FRETMP
FFA2H	H.PTRG	5	PTRGE (省略値以外の変数使用のポインタ)
FFA7H	H.PHYD	5	PHYDIO (物理ディスク入出力)
FFACH	H.FORM	5	FORMAT (ディスクのフォーマット)
FFB1H	H.ERRO	5	ERROR(アプリケーションのエラーを処理)

FFB6H	H.LPTO	5	LPTOUT (省略値以外のプリンタで出力)
FFB8H	H.LPTS	5	LPTSTT (省略値以外のプリンタで状態を知る)
FFC0H	H.SCRE	5	SCREEN
FFC5H	H.PLAY	5	PLAY

拡張 BIOS が使用

FFCAH	FCALL	5	拡張 BIOS が使用
FFCFH	DISINT	5	DOS が使用
FFD4H	ENAIN	5	DOS が使用

データエリア

FFE7H	RG8SAV	1	VDP レジスタ#8 のセーブエリア
FFE8H	RG9SAV	1	VDP レジスタ#9 のセーブエリア
FFF9H	RG10SA	1	VDP レジスタ#10 のセーブエリア
FFEAH	RG11SA	1	VDP レジスタ#11 のセーブエリア
FFEBH	RG12SA	1	VDP レジスタ#12 のセーブエリア
FFECH	RG13SA	1	VDP レジスタ#13 のセーブエリア
FFEDH	RG14SA	1	VDP レジスタ#14 のセーブエリア
FFEEH	RG15SA	1	VDP レジスタ#15 のセーブエリア
FFEFH	RG16SA	1	VDP レジスタ#16 のセーブエリア
FFF0H	RG17SA	1	VDP レジスタ#17 のセーブエリア
FFF1H	RG18SA	1	VDP レジスタ#18 のセーブエリア
FFF2H	RG19SA	1	VDP レジスタ#19 のセーブエリア
FFF3H	RG20SA	1	VDP レジスタ#20 のセーブエリア
FFF4H	RG21SA	1	VDP レジスタ#21 のセーブエリア
FFF5H	RG22SA	1	VDP レジスタ#22 のセーブエリア
FFF6H	RG23SA	1	VDP レジスタ#23 のセーブエリア
FFF7H			システム予約
FFFAH	RG25SA	1	VDP レジスタ#25 のセーブエリア
FFFBH	RG26SA	1	VDP レジスタ#26 のセーブエリア
FFFBH	RG27SA	1	VDP レジスタ#27 のセーブエリア
FFFDH			システム予約

3章 I/O マップ

00H ~ 3FH	ユーザー定義
40H ~ 4FH 40H 41H ~ 4FH	拡張 I/O ポート メーカーID メーカー定義
50H ~ 6FH	リザーブ
70H ~ 73H	MIDI サウルス (BIT2)
74H ~ 7BH	リザーブ
7CH ~ 7DH 7CH 7DH	MSX-MUSIC アドレス データ
7EH ~ 7FH	リザーブ
80H ~ 87H 80H 81H 82H 83H 84H 85H 86H 87H	RS-232C 8251 データ 8251 ステータス / コマンド ステータスリード / インタラプタマスク 未使用 8253 カウンタ 0 8253 カウンタ 1 8253 カウンタ 2 8253 モードレジスタ
88H ~ 8BH 88H 89H 8AH 8BH	外付け VDP turboR では削除 VRAM アクセス レジスタアクセス (write) パレットアクセス (write) レジスタ間接指定
8CH ~ 8DH	MSX-MODEM
8EH ~ 8FH	リザーブ
90H ~ 91H 90H 91H	プリンタ (Read) b1 1=BUSY (Write) b0 0=STROBE データ
92H ~ 97H	リザーブ
98H ~ 9BH 98H 99H 9AH 9BH	VDP VRAM アクセス レジスタアクセス (write) パレットアクセス (write) レジスタ間接指定
9CH ~ 9FH	リザーブ
A0H ~ A3H A0H A1H A2H A3H	PSG アドレス (Write) データ (Read) データ
A4H ~ A5H A4H A5H	PCM turboR 専用 (Read) カウンタ (Write) データ (Read) b7 コンパレータ (Write) モードセット
A6H	リザーブ
A7H	ポーズキー制御 turboR 専用 b0 PAUSE ランプ (1=ON) b7 高速モードランプ
A8H ~ ABH A8H A9H AAH ABH	パラレルポート (8255) 基本スロット指定信号 (MSB) CS3,CS2,CS1,CS0 (LSB) (2bit ずつ) キーボードリターン信号 (0=ON) (MSB) SOUND,CAPS,CASW *,CASON *,KB3,KB2,KB1,KB0 (LSB) *turboR で削除された モードセット
ACH ~ AFH	MSX-ENGINE
B0H ~ B3H B0H B1H B2H B3H	拡張メモリ(ソニー仕様) (8255) turboR では削除 アドレス a7-a0 アドレス a15-a13,a10-a8, コントロール,Read/Write アドレス a12-a11, データ モードセット

B4H ~ B5H B4H B5H	CLOCK-IC (RP-5C01) アドレス データ
B6H ~ B7H	リザーブ
B8H ~ BBH B8H B9H BAH BBH	ライトペンコントロール (サンヨー仕様) turboR では削除 Read/Write Read/Write Read/Write Write
BCH ~ BFH BCH BDH BEH BFH	VHD コントロール(JVC) (8255) turboR では削除 ポート A ポート B ポート C モードセット
C0H ~ C1H C0H C1H	MSX-AUDIO(マスター) turboR では削除 アドレス データ
C2H ~ C3H C2H C3H	MSX-AUDIO(スレーブ) turboR では削除 アドレス データ
C4H ~ C7H	リザーブ
C8H ~ CFH	MSX-INTERFACE turboR では削除
D0H ~ D7H	フロッピーディスクコントローラ(FDC) turboR では削除
D8H ~ D9H D8H D9H	漢字 ROM 第 1 水準 (Write) b5-b0 アドレス下位 6bit (Write) b5-b0 アドレス上位 6bit (Read) データ
DAH ~ DBH DAH DBH	漢字 ROM 第 2 水準 (Write) b5-b0 アドレス下位 6bit (Write) b5-b0 アドレス上位 6bit (Read) データ
DCH ~ DDH	漢字 ROM 拡張
DEH ~ DFH	リザーブ
E0H ~ E2H E0H E1H E2H	外付け MSX-MIDI (Read) 8251 受信データ (Write) 8251 送信データ (Read) 8251 ステータス (Write) モード/コマンド b7 0: イネーブル b0 0: 8251 を内蔵 MSX-MIDI のアドレスにする
E3H	リザーブ
E4H ~ E5H E4H E5H	turboR CPU 切り換え制御 06H イネーブル b6 0: DRAM 1: ROM b5 0: R800 1: Z80
E6H ~ E7H E6H E7H	システムタイマ turboR 専用 (Read) カウンタ下位 8bit (Write) カウンタクリア (Read) カウンタ上位 8bit
E8H ~ EFH E8H E9H EAH EBH ECH EDH EEH EFH	内蔵 MSX-MIDI (Read) 8251 受信データ (Write) 251 送信データ (Read) 8251 ステータス (Write) 8251 モード/コマンド (Write) 8253 OUT2 端子の信号ラッチ (Read) EAH のイメージ カウンタ #0 カウンタ #1 カウンタ #2 (MSB) SC1, SC0, RW1, RW0, M2, M1, M0, BCD (LSB)
F0H ~ F2H	リザーブ
F3H	VDP 表示モード 2+以降 (MSB) YAE, YUV, TP, M1, M2, M5, M4, M3 (LSB)
F4H	リセットステータス 2+以降
F5H	デバイスイネーブル (write) 1 で使用可能 b0 漢字 ROM 第 1 水準 b1 漢字 ROM 第 2 水準 b2 MSX-AUDIO b3 スーパーインポーズ b4 MSX-INTERFACE b5 RS-232C b6 ライトペン b7 CLOCK-IC
F6H	カラーバス I/O

F7H	AV コントロール b0 (Write) オーディオ R ミキシング (0:ON) b1 (Write) オーディオ L ミキシング (1:ON) b2 (Write) ビデオ入力選択 (0:21pinRGB) b3 (Read) ビデオ入力検出 b4 (Write) AV コントロール (0:TV) b5 (Write) Ym コントロール (0:TV) b6 (Write) R#9 b4 の反転 b7 (Write) R#9 b5 の反転
F8H ~ FBH	リザーブ
FCH ~ FFH	メモリマップ (Write) FCH ページ 0 FDH ページ 1 FEH ページ 2 FFH ページ 3

4章 キャラクターコード

下の表は16進数にしたキャラクターコードの上位1桁で縦の行が決まり、下位1桁で横の行が決まります。グラフィックキャラクターはコントロールコード&H01に続けてキャラクタのコードを出力します。

表 5.4.1 キャラクターコード

上位 下位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	コントロールコード		空	0	@	P	`	p	♠			-	タ	ミ	た	み
1			!	1	A	Q	a	q	♥	あ	。	ア	チ	ム	ち	む
2			"	2	B	R	b	r	♣	い	「	イ	ツ	メ	つ	め
3			#	3	C	S	c	s		う	」	ウ	テ	モ	て	も
4			\$	4	D	T	d	t	○	え	、	エ	ト	ヤ	と	や
5			%	5	E	U	e	u	●	お	・	オ	ナ	ユ	な	ゆ
6			&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ
7			'	7	G	W	g	w	あ	き	ア	キ	ヌ	ラ	ぬ	ら
8			(	8	H	X	h	x	い	く	イ	ク	ネ	リ	ね	り
9			)	9	I	Y	i	y	う	け	ウ	ケ	ノ	ル	の	る
A			*	:	J	Z	j	z	え	こ	エ	コ	ハ	レ	は	れ
B			+	;	K	[	k	{	お	さ	オ	サ	ヒ	ロ	ひ	ろ
C			,	<	L	¥	l		や	し	ヤ	シ	フ	ワ	ふ	わ
D			-	=	M	]	m	}	ゆ	す	ユ	ス	ヘ	ソ	へ	ん
E			.	>	N	^	n	~	よ	せ	ヨ	セ	ホ	。	ほ	
F			/	?	O	_	o	騰	つ	そ	ッ	ソ	マ	°	ま	訓

表 5.4.2 グラフィックキャラクター

	4	5
0		π
1	月	⊥
2	火	⊥
3	水	⊥
4	木	⊥
5	金	⊥
6	土	⊥
7	日	—
8	年	「
9	円	」
A	時	「
B	分	」
C	秒	×
D	百	大
E	千	中
F	万	小

5章 コントロールコード

表 5.5.1 コントロールコード表

コード	機能	対応キー
00H		CTRL+@
01H	グラフィックキャラクター出力時のヘッダ	CTRL+A
02H	カーソルを直前の語の先頭へ移動	CTRL+B
03H	入力待ち状態を終了	CTRL+C
04H		CTRL+D
05H	カーソル以降を削除	CTRL+E
06H	カーソルを次の語の先頭へ移動	CTRL+F
07H	スピーカーを鳴らす (BEEP と同じ)	CTRL+G
08H	カーソルの 1 つ前の文字を削除	CTRL+H BS
09H	次の水平タブ位置へ移動	CTRL+I TAB
0AH	行送り (ラインフィード)	CTRL+J
0BH	カーソルをホームポジション (左上) に戻す	CTRL+K HOME
0CH	画面をクリアし、カーソルをホームポジションに戻す	CTRL+L CLS
0DH	カーソルを左端に戻す (キャリッジリターン)	CTRL+M RETURN
0EH	カーソルを行末へ移動	CTRL+N
0FH		CTRL+O
10H		CTRL+P
11H		CTRL+Q
12H	挿入モードの ON/OFF スイッチ	CTRL+R INS
13H		CTRL+S
14H		CTRL+T
15H	1 行を画面から削除	CTRL+U
16H		CTRL+V
17H		CTRL+W
18H		CTRL+X SELECT
19H		CTRL+Y
1AH		CTRL+Z
1BH		CTRL+[ESC
1CH	カーソルを右へ移動	CTRL+¥ →
1DH	カーソルを左へ移動	CTRL+] ←
1EH	カーソルを上へ移動	CTRL+^ ↑
1FH	カーソルを下へ移動	CTRL+_ ↓
7FH	カーソルの指す文字を削除	DEL

6章 エスケープシーケンス

BIOS が次のエスケープシーケンスをサポートしています。これらのコードを出力することで、対応した動作を行います。

表 5.6.1 エスケープシーケンス表

<ESC>A	カーソルを上に移動
<ESC>B	カーソルを下に移動
<ESC>C	カーソルを右に移動
<ESC>D	カーソルを左に移動
<ESC>H	カーソルをホームポジションに移動
<ESC>Y<Y座標+20H><X座標+20H>	カーソルを指定位置に移動
<ESC>J	画面をクリア
<ESC>E	画面をクリア
<ESC>K	カーソル位置から行の終わりまで削除
<ESC>I	カーソルのある行の終わりまで削除（小文字の L）
<ESC>J	画面の終わりまで削除
<ESC>L	1行挿入
<ESC>M	1行削除
<ESC>x4	カーソルの形を ■ にする
<ESC>x5	カーソルを消す
<ESC>y4	カーソルの形を _ にする
<ESC>y5	カーソルを表示する

<ESC>はエスケープコードを意味します。

7章 メーカーコード

表 5.7.1 メーカーコード

コード	メーカー名
0	アスキー
1	マイクロソフト
2	キャノン
3	カシオ計算機
4	富士通
5	富士通ゼネラル
6	日立製作所
7	京セラ
8	松下電気産業
9	三菱電気
10	日本電気
11	ヤマハ
12	日本ビクター
13	フィリップス
14	パイオニア
15	三洋電気
16	シャープ
17	ソニー
18	スペクトラビデオ
19	東芝
20	ミツミ電機
21	テレマティカ
22	グラディエンテ
23	シャープドブラジル
24	GOLD STAR
25	DEAWOO
26	SumSong

表 5.7.1 に示すメーカーコードは拡張 BIOS 用で用いられるもので、拡張 I/O ポート (40H ~ 4FH) のメーカー ID とは次の点で異なります。

- ・マイクロソフトのメーカー ID が存在せず、アスキーがメーカー ID に登録されている
- ・メーカー ID 0 が存在しない
- ・メーカー ID 27 ~ 127 が将来のために予約されている

これ以外のメーカー ID 2 ~ 26 に関しては、メーカーコードと同一です。

索引

- 数字 -		MIDI	150	VDP	45
1 ビットサウンドポート	155	MIDI-SAURUS	150	VDP(n)関数	53
12 時間計/24 時間計	170	MIDI インターフェイス3	150	VDP コマンド	107
16 ビットアドレスI/O	34	MODE レジスタ	172	VDP に関連するワークエリア	54
50 音配列	167	MSX1	14	VRAM	47, 50
8251	150	MSX2	14	VRAM マップ	65, 68, 72, 76, 80, 82, 84, 87, 89, 91, 95, 98
8251 コマンドレジスタ	152	MSX2+	14		
8251 受信データ	152	MSX-AUDIO	135	- X -	
8251 ステータスレジスタ	152	MSX-MIDI	150	XOR	108
8251 送信データ	152	MSX-MUSIC	135		
8253	150	MSXturboR	14	- Y -	
8253 OUT 端子の信号のラッチ	152	MSX 仕様のプリンタ	166	YJK	56, 95
8253 カウンタ#0 ~ #2	152	MULTI COLOR	77	YJK 方式	92
8253 コマンドレジスタ	153			YJK → RGB 変換テーブル	93
8254	150	- N -		YJK/RGB 混在	56, 92
		NOT	108	YMMM (Y 方向の VRAM間高速転送)	109
		NTSC 方式	124	- Z -	
- A -		- O -		Z80	15
AND	108	OPLL	135		
AY-3-8910	161	OR	108	- あ -	
				アクセスレジスタ	60
- B -		- P -		アタック	142
BIOS	39, 178	PAC	159	アタックレイト	143
BLOCK	140	PAL 方式	124	アラーム	170
		PCM	156		
- D -		POINT (カラーコードの読み出し)	120	- い -	
D/A コンバータ出力信号	156	PRESET	108	インタースロットコール	26
DMA	16	PSET	108	インターセグメントコール	31
DUAL-MIDI	150	PSET (点の描画)	119	インターレース	56
		PSG	130		
- E -				- う -	
EOR	108	- R -		ウェイト機能	56
		R800	16	閏年カウンタ	170
- F -		RESET レジスタ	172		
F_number	141	RGB 方式	92	- え -	
FM-BIOS	136			エスケープシーケンス	202
FM 音源	135	- S -		エンベロープ	142
FM パック (FM-PAC)	135, 159	SCC	146	エンベロープ形状	143
		SCREEN 0 40 字モード	63	エンベロープ形状の設定	133
- G -		SCREEN 0 80 字モード	66	エンベロープ周期の設定	133
GRAPHIC1	69	SCREEN 1	69	エンベロープ発生器	130
GRAPHIC2	72	SCREEN 2	72		
GRAPHIC3	80	SCREEN 3	77	- お -	
GRAPHIC4	82	SCREEN 4	80	オートインクリメントモード	49
GRAPHIC5	84	SCREEN 5	82	音色設定	142
GRAPHIC6	87	SCREEN 6	84	音量設定	132, 142, 148
GRAPHIC7	89	SCREEN 7	87		
		SCREEN 8	89	- か -	
- H -		SCREEN 10	94	カートリッジスロット	21
HMMC (CPU → VRAM 高速転送)	108	SCREEN 10 11	92	外部メモリ	17
HMMM (VRAM間高速転送)	110	SCREEN 11	94	カウンタモード	165
HMMV (矩形領域の高速塗り潰し)	111	SCREEN 12	95	拡張 BIOS	39
		SRCH (カラーコードのサーチ)	118	拡張 I/O ポート	34
- I -		STOP (VDP コマンドの中断)	120	拡張 RAM	50
IFF	35	SUB ROM	39, 185	拡張スロット	20
IMP	108			画面モード	55
I/O ポート	17, 34	- T -		カラーバス	55
I/O マップ	197	TAND	108	カラーテーブル	67, 71, 75
		TEOR	108	カラーレジスタ	57, 65, 68, 71, 75, 79, 83, 85, 88, 91, 94, 97
- J -		TEST レジスタ	172	漢字 ROM	174
JIS	175	TEXT1	63	漢字コード	175
JIS 標準配列	167	TEXT2	66	漢字番号	174
		TIMP	108	間接指定	48
- L -		TMS9918A	46		
LINE (直線の描画)	117	TMS9918A 互換モード	52		
LMMC (VRAM → CPU 論理転送)	114	TNOT	108		
LMMC (CPU → VRAM 論理転送)	112	TOR	108		
LMMM (VRAM間論理転送)	115	TPRESET	108		
LMMV (矩形領域の論理塗り潰し)	116	TPSET	108		
		TXOR	108		
- M -					
M1 サイクル	15	- V -			
MAIN ROM	39, 178	V9938	46		
		V9958	46		

- き -

キーオン	141
キースキャン	168
キースケールレイト	143
キースケールレベル	143
キーバッファ	168
キーボード	167
キーマトリクス	167
基準信号	156
輝度成分 Y	92
基本スロット	20
キャラクタコード	200
キャラア	142
境界色発見フラグ	62

- く -

矩形波	130
区点	175
クロック IC	170

- け -

減衰音タイプ	144
--------	-----

- こ -

高速 I/O アクセス	16
コマンドレジスタ	61
コマンド実行フラグ	62
コントロールコード	201
コントロールレジスタ	46, 48, 55
コンパレータの出力信号	156
コンボジットビデオ出力	123

- さ -

最後部面	52
サイン波	148
サスティン	141, 142
サスティンレベル	143
サンプリング	148
サンプリングレート	158
サンプル・ホールド回路入力信号	156
サンプルホールド信号	156

- し -

色相成分 J,K	92
システムエクスクルーシブ	42
システムセグメント	30
システムタイマー	18, 176
持続音タイプ	144
シフト JIS	175
周波数設定	131, 147
出力スイッチ	148
ジョイスティック	165
ジョイスティックポート	164
ジョイスティックモード	165
衝突フラグ	61
シングルバッファ	156

- す -

スーパーインポーズ面	52
垂直帰線割り込み	35
垂直帰線割り込みフラグ	61
垂直帰線期間	62, 125
垂直表示期間	125
水平帰線割り込み	59, 124
水平帰線割り込みフラグ	61
水平帰線期間	62, 125
水平表示期間	125
ステータスレジスタ	46, 49, 61
スプライト	71, 76, 79, 81, 84, 86, 89, 91, 94, 97, 99

スプライトアトリビュートテーブル

	52, 100, 102, 104
スプライトカラーテーブル	52, 104, 105
スプライトのサイズ	55
スプライトの衝突	106
スプライト表示範囲	99
スプライトパターンジェネレーターテーブル	52, 100, 104
スプライト面	52
スプライトモード 1	99
スプライトモード 2	99, 103
スロット	20
スロット番号	25

- セ -

セカンダリスロット	20
セグメント	27

- そ -

ソフトウェアエンベロープ	134
ソフトウェアトリガストロープ	153

- た -

第 1 水準	174
第 2 水準	174
第 5 スプライトフラグ	61
タイマ割り込み	35
タイミング A	17
タイミング B	17
只 MIDI	150
縦 192 ドット表示	56
縦 212 ドット表示	56
縦方向の表示位置	59
ダブルバッファ	156

- ち -

直接指定	48
------	----

- て -

ディケイ	142
ディケイレイト	143
ディスプレイレジスタ	59
テーブルベースアドレスレジスタ	57
デジタイズ機能	55
デバイス番号	39
転送レディフラグ	62

- と -

同期モード	56
トータルレベル	142, 143
ドラムス指定	140
ドラムの音量設定	142

- な -

内蔵マップ	16
内蔵 ROM	16

- の -

ノイズ周波数設定	132
ノイズ発生器	130
ノーウェイト	17
ノン・インターレース	56

- は -

ハードウェアタイリング	86
ハードウェアトリガストロープ	153
ハードウェア横スクロール	56
波形メモリ	146, 147
パターンジェネレーターテーブル	52, 64, 70, 73, 77

パターンネームテーブル

	52, 65, 66, 70, 74, 79, 83, 85, 88, 90, 93, 96
パターン面	52
発音周波数設定	141
バックアップ	170
バックドロップ面	52
バックドロップの色	57
バッテリーバックアップメモリ	170
バッファモード	156
パラレルポート	166
パレットテーブル	53
パレットのセーブエリア	53
パレットレジスタ	47, 50
ハンドシェイク方式	166
半波整流	143
汎用入出力インターフェイス	164

- ひ -

非オー トインクリメントモード	48
表示開始ライン	59
表示タイミング	125
表示範囲	59
表示フィールドフラグ	62
標本化	148

- ふ -

ファンクションキー	169
フィードバック	142, 143
フィルタ出力信号	156
フィルタ入力信号	156
フック	37
プライマリスロット	20
プリנק	57, 68
プリנק属性	68
プリנקレジスタ	68
プリンタインターフェース	166
ブロードキャストコマンド	40
プログラマブルワンショット	153

- へ -

ページ	21
ページブレーク	16
ページモード	16
へろへろ 5 号	150
変調量	142

- ほ -

方形波レートジェネレータ	153
ポーレートジェネレータ	152
ボルタメント	145

- ま -

マイクアンプ出力信号	156
マウス	165
マウススイッチ	61
マップサポートルーチン	29
マルチプル	142, 143

- み -

ミキシング設定	132
ミュートینگ制御	156

- め -

命令実行クロック	18
メーカーコード	42, 203
メモリマップ	27
メモリマップ H/O	146

- も -

モード 1 割り込み	35
------------	----

モード3 割り込み	16
モードレジスタ	55, 64, 66, 69, 73, 77, 81, 83, 85, 88, 90, 93, 96
モジュレータ	142
- ゆ -	
ユーザーセグメント	30
- よ -	
横スクロール	60
横方向の表示位置	59
- ら -	
ライトペン	55
ライトペンスイッチ	61
- り -	
リアルタイムクロック	170
リフレッシュ	17
量子化	148
リリース	142
リリースレイト	143
- れ -	
レートジェネレータ	153
- ろ -	
ローパスフィルタ	156
ロジカルオペレーション	108
論理演算	108
- わ -	
ワークエリア	188
割り込み	35
割り込み許可フリップフロップ	35

本書を作成するに当たっては数々の資料を参考にさせて頂き、また引用させて頂きました。下にそれら資料を挙げます。もし本書より詳しい情報を知りたい場合は、下記の文献を当たるとより詳しい情報があるかもしれません。とは言うものの、入手はどれも難しくなっています。

MSX2 テクニカルハンドブック (アスキー)
MSX turboR テクニカル・ハンドブック (アスキー)
MSX Datapack (アスキー)
MSX Datapack turboR 版 (アスキー)
グラフィックス秘伝 (アスキー)
ゲーム作りのテクニック -スプライト編- (アスキー)
V9958 E-VDP-II テクニカルデータブック (ヤマハ)
ビデオディスプレイプロセッサ TMS9918A/9928A/9929A
ユーザーズマニュアル (日本テキサスインスツルメンツ)
MSX テクニカルガイドブック (ASCAT)
MSX・FAN (徳間書店)
バックアップ活用テクニック (三オブックス)

G-NET クリガイプロジェクト スタッフ

代表：にゃんたっく
執筆：にゃんたっく
A t o C
本社前
MIJINKO
挿絵：Hack EST

レイアウト主任：X i .
レイアウト協力：ねこにゃんにゃん
澤田修宏
赤樹将
落書き
表紙カバー：Takun
版下製作・発行：夢神楽遊睡

奥付

MSX クリエイターズガイドブック

1998 年 8 月 16 日 初版発行
定価 2,000 円 (送料別)

発行者 夢神楽遊睡
発行所 GENUINE NETWORK
〒812-0018 福岡県福岡市博多区住吉 2-16-1 メゾン住吉 606
FAX : 092-263-7835 E-mail : support@g-net.org