

FM-7シリーズ テクニカルノウハウ

FM-Techknow

FM-7 / NEW 7 / 77 & FM77 *AV*

Writing

阪井末幸

Sakai Sueyuki



BNN
Bug News Network

FM-7シリーズ テクニカルノウハウ

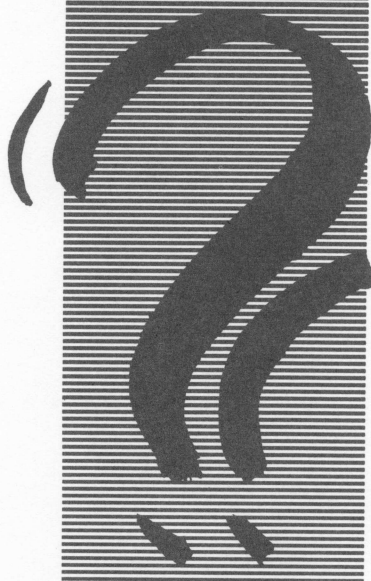
FM-Techknow

FM-7/NEW 7/77 & FM77AV

Writing

阪井末幸

Sakai Sueyuki



BNN
Bug News Network

発刊にあたって

本書は、富士通の8ビットパーソナルコンピュータ FM-7 シリーズを対象として、本体ならびに周辺機器の内部構造から活用の方法までを解説しています。本書の執筆にあたっては、次の点が考慮されています。

◆ FM-7 シリーズ (FM-7・FM-NEW7・FM-77・FM77AV) の4種類に対応するように心掛けていますが、特に最新の FM77AV を中心にして記述しています。FM-7・FM-NEW7・FM-77 に対しての記述にあたっては、“F-BASIC V3.0 では”、“サブモニタ ROM タイプ C では”、あるいは“FM-7 では”という記述によって区別されています。

FM-77L4 では、漢字機能を強化した F-BASIC V3.5 も動作しますが、この V3.5 については考慮しておりませんので予め御了承ください。

◆ マシン語プログラムは富士通のアブソリュートアセンブラで作成されています。本文中のサンプルプログラムの入力にあたっての注意事項を、付録の「プログラムの入力」にまとめましたので是非ご参照ください。

なお、マシン語プログラムのソースリストは、メディアサービスにおいて提供しています。興味のある方は、是非メディアサービスを御利用ください。

はじめに

パソコンも、一時期の熱狂的なブームが去ると共に社会に定着して、ワープロ等の OA 機器あるいはゲームなどに利用されて私達の身近な存在となっています。ビジネスユースでは 16 ビット機が主流になりましたが、パーソナルなコンピュータとしてはまだ 8 ビット機が大きな部分を占めています。特にファミリーコンピュータの普及には目を見はられます。そしてこのファミコンブームにより、ゲーム中心の従来の 8 ビット機は大きな転換を迫られているといっていでしょう。

このような状況のなかで FM77AV は、FM 音源標準サポート、4096 色同時発色という AV (オーディオ・ビジュアル) 機能を前面に押しだして発売されました。テレビ画面のデータをパソコンに取り込むことのできるビデオデジタイズ機能を含めて、従来のゲームパソコンとは一味異なった方向づけがなされているようです。

FM-7 に対して FM-NEW7、FM-77 は小幅な改良にとどめられていましたが、この FM77AV の仕様は大幅に変更されています。そこで FM77AV のユーザーがその機能を十二分に引き出し、より有効に使うためには、本体の内部や周辺機器について詳しく知ることがポイントになってきます。

そこで本書では、FM77AV の新機能を中心にとりあげ、FM-7 シリーズの本体はもちろん、プリンタ、ディスクユニットに至るまで、内部解析情報や活用のノウハウを実践に役立つようにまとめています。またすぐに使える便利なプログラムも豊富に紹介しています。

限られたページの中で、充分意を尽すことはできませんが、読者の方々自らの内容補完により、FM-7 シリーズに対する理解をより深めていただきたいと存じます。FM-7 シリーズを使いこなすための座右の書として利用いただければ幸いです。

なお本書の執筆は、阪井末幸、高橋暢生、梅野香織が共同で行ないました。

本書を執筆するにあたり、富士通株式会社殿、富士通 OA 株式会社殿よりハードウェア及びソフトウェアを快く提供していただきましたことを深謝いたします。特に富士通ブラザ梅田の方々の協力には、心からお礼申しあげます。

また編集を担当していただいた株式会社ビー・エヌ・エヌと株式会社システムソフトのスタッフの方々には大変なお世話になりました。この場を借りてお礼申しあげます。

1986 年 8 月 著者

 発刊にあたって

はじめに

第1章 メモリマップ 11

1-1	メインシステムのメモリマップ	11
1-1-1	裏RAM	13
1-1-2	RAM空間	16
1-1-3	サブCPU空間	19
1-1-4	拡張RAM空間	20
1-1-5	標準空間	21
1-2	イニシエータROM	23
1-3	サブシステムのメモリマップ	25
1-3-1	サブモニタROMの選択	27
1-3-2	キャラクタROMの選択	28
1-3-3	VRAMのページ切り替え	29
1-4	共有RAM	33
1-5	メモリモード (DOS/BASIC)	33
1-6	裏RAM活用	36

第2章 F-BASICの内部構造 39

2-1	メモリマップ	39
2-2	メモリマップの変化	40
2-3	プログラムの格納状態	42
2-4	中間言語	44
2-5	変数の格納状態	46
2-5-1	単純変数	46
2-5-2	配列変数	48
2-6	文字列エリア	50
2-7	F-BASIC V3.3の変更点	52
2-8	マシン語領域の確保	55
2-9	BASICの未定義命令	56
2-10	EXEC文にパラメータ指定	57
2-11	裏RAMからBASIC ROMルーチンをコール	58
2-12	BASICプログラム復活法	59

2-13	テキストユーティリティ	60
2-13-1	テキストサーチ & リプレイス	60
2-13-2	拡張AUTO & EDIT	62
2-13-3	行の複写, 移動	65
2-13-4	クロスリファレンス	66
2-13-5	ラベルコンダクタ	68
第3章	F-BIOS	71
3-1	F-BIOSの位置づけ	71
3-2	F-BIOS, OPNBIOSの種類	72
3-3	BIOSインターフェース	74
3-4	BIOSの呼び出し手順	78
3-5	BIOSのエラー処理	80
3-6	ブザーに対するBIOS	81
3-7	CRT1 文字表示	82
3-8	音色データ読み込み	83
第4章	サブシステム	85
4-1	サブシステムのメモリマップ	85
4-2	メイン・サブインターフェース	89
4-3	サブシステムに対するBIOS	91
4-4	サブシステムコマンド	94
4-5	サブシステムの直接制御	97
4-5-1	共有RAMアクセス	97
4-5-2	TESTコマンド	98
4-5-3	プログラムの転送	102
4-6	コンソールバッファ	104
4-7	ハードウェアスクロール	107
4-8	SPECIALコマンド	107
4-9	サブシステムへのPEEK, POKE	109

第5章 キー入力,タイマー	113
5-1 キーボード入力	113
5-1-1 FM-7のキー入力	113
5-1-2 キー入力バッファ	114
5-1-3 BASICでのキー入力の比較	116
5-1-4 キーボードに対するBIOS	117
5-1-5 キーデータの読み取り	118
5-1-6 エンコーダ機能	119
5-1-7 PFキーの入力	127
5-1-8 PFキー文字列の定義	127
5-1-9 PFキー割り込み定義	128
5-1-10 PFキー文字列の初期設定	129
5-2 ジョイスティック	130
5-2-1 ジョイスティック・インターフェース	130
5-2-2 ジョイスティックのアクセス	131
5-3 マウス	134
5-3-1 マウス・インターフェース	134
5-3-2 マウスのアクセス	135
5-4 タイマー	140
5-4-1 タイマーの読み取り	140
5-4-2 タイマーへの書き込み	143
第6章 割り込み	147
6-1 BASICにおける割り込み処理	147
6-2 割り込みの種類	149
6-3 割り込み要因の検出	150
6-4 割り込みマスク	151
6-5 割り込みベクトル	154
6-6 BREAKキーをキャンセル	155
6-7 SWI命令でマシン語のデバッグ	156
第7章 カセットファイル	157
7-1 CMTインターフェース	157
7-2 データフォーマット	157
7-3 カセットファイルに対するBIOS	160
7-4 カセットファイルの拡張ディレクトリ表示	163

7-5	2倍速LOAD	165
第8章 フロッピーディスク		167
8-1	フロッピーディスクの構造	167
8-1-1	物理構造	167
8-1-2	セクタアドレスとクラスタ	168
8-1-3	ディスクマップ	169
8-1-4	IDセクタ	170
8-1-5	ディレクトリ	171
8-1-6	FAT	172
8-1-7	ファイルの形式	174
8-1-8	セクタシーケンス	175
8-2	フロッピーに対するBIOS	176
8-2-1	BIOS	176
8-2-2	BOOT ROM	178
8-3	FDCについて	178
8-3-1	FDCレジスタ	179
8-3-2	FDCインターフェース	180
8-3-3	FDCコマンド	180
8-3-4	FDCステータス	185
8-4	ディスクユーティリティ	188
8-4-1	セクタダンプ	188
8-4-2	アスキーダンプ	190
8-4-3	ファイルインフォメーション	192
8-4-4	ファイルダンプ	195
8-4-5	高速ファイルコピー	196
8-4-6	横表示FILES	197
8-4-7	フォーマット & トラックコピー	198
8-4-8	オートスタートユーティリティ	206
8-4-9	ファイルメニュー	209
第9章 プリンタ		215
9-1	プリンタに対するBIOS	215
9-2	画面コピー機能	218
9-3	拡張画面コピープログラム	220
9-4	PC用プリンタをFMIに接続	226

第10章 音源	229
10-1 PSG	229
10-1-1 PSGのハードウェア	229
10-1-2 PSGの基礎	230
10-1-3 PSGの応用	233
10-1-4 PSGのマシン語アクセス	237
10-1-5 PSG効果音ルーチン	239
10-1-6 PSG音楽ルーチン	243
10-2 FM音源	244
10-2-1 FM音源の基礎	244
10-2-2 FM音源のマシン語アクセス	252
10-2-3 FM音源音楽ルーチン	255
10-3 OPNBIOS	262
第11章 漢字ROM	265
11-1 漢字ROMへのアクセス	265
11-2 漢字ROMに対するBIOS	268
11-3 漢字JISコード対応表示	269
11-4 JISコード表にない漢字ROMデータ	271
11-5 ファンクションキーエリアに漢字表示	272
11-6 拡張漢字表示プログラム	273
11-7 高速漢字表示プログラム	275
11-8 漢字ビットイメージプリント	279
11-9 外字フォントの作成,登録	283
11-10 漢字SYMBOL文	287
第12章 RS-232C	291
12-1 RS-232Cインターフェース用ケーブル	291
12-2 通信パラメータ	294
12-2-1 データのビット長	294
12-2-2 データ伝送速度	294
12-2-3 パリティ・チェック	294
12-2-4 通信モード	295
12-2-5 ストップ・ビット	295
12-2-6 Xパラメータ	296

12-3	通信パラメータの設定	296
12-4	データの転送	299
12-5	プログラムの転送	301
12-6	ターミナルモード	302
12-7	音響カブラ／モデム	305
第13章 FM77AVの特色		309
13-1	多色グラフィック	309
13-2	アナログパレット	313
13-3	VRAMダイレクトアクセス	324
13-4	ハードウェア描画機能	326
13-5	ハードウェアスクロール	329
13-6	メモリ管理	334
13-7	AVTV制御	339
	13-7-1 スーパーインポーズ機能	340
	13-7-2 ビデオデジタイズ機能	341
	13-7-3 VRAMのセーブ／ロード	341
	13-7-4 アナログパレット設定テクニック	343
13-8	F-BASIC V3.0とV3.3のベンチマークテスト	345
第14章 ユーティリティ&ランダムテクニック		351
14-1	漢字ビットイメージプリント2	351
14-2	FLEX→F-BASICコンバータ	353
14-3	ディレクトリアレンジャー	357
14-4	FATの整序	631
14-5	WIDTHプリセット	363
14-6	FATセーブプレートの変更	364
14-7	万年カレンダー	365
14-8	マシン語データ文作成	368
14-9	圧縮漢字表示	370
14-10	データサーチャ	371

第15章 ハードウェア回路図(FM77AV回路図) 373

付 録 475

A. メインシステムI/Oアドレスマップ	475
B. サブシステムI/Oアドレスマップ	483
C. F-BASIC V3.0メモリマップ	486
D. F-BASIC V3.3メモリマップ	498
E. OPNBIOSメモリマップ	508
F. BIOS(V3.3)メモリマップ	509
G. サブモニタROM1メモリマップ	510
H. サブモニタROM2メモリマップ	510
I. サブシステム変数一覧	511
J. エラーメッセージ一覧	514
K. キャラクタコード表	517
L. キー配列とキーコード	518
M. CPU6809命令表	524
N. 音色データ一覧	527
O. プログラムの入力にあたって	534

索 引 538

メモリマップ

第 1 章

1-1 メインシステムのメモリマップ

FM-7 および FM77AV のメインシステムのメモリマップを、図 1-1、図 1-2 に示します。

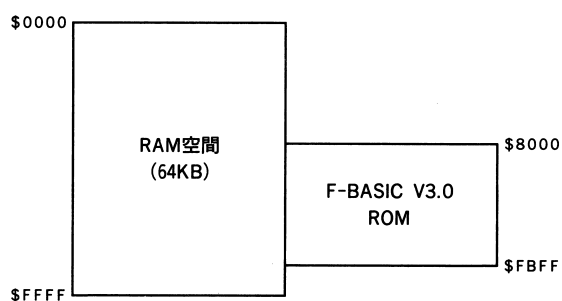


図1-1 FM-7のメモリマップ

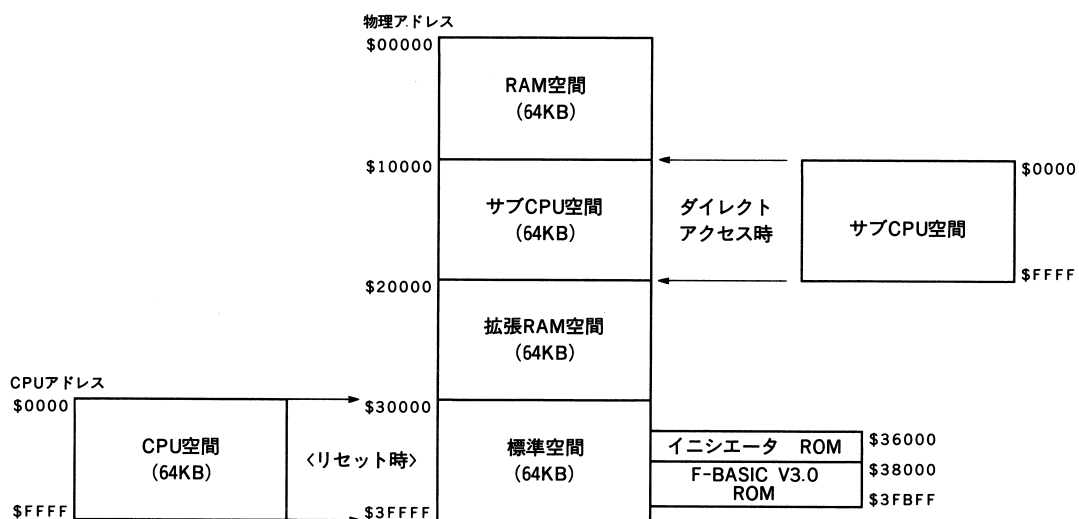


図1-2 FM 77AVのメモリマップ

FM77AV では、4096 色同時発色、FM 音源サポート等により増大した CPU 空間を、8 ビット CPU にて管理するため、ふたつの新しいメモリ管理手法を導入しています。

① MMR (Memory Manegement Register) によるメモリ・マッピング機能

② TWR (Text Window Register) によるテキスト・ウィンドウ機能

メモリ・マッピング機能とは、64KB の CPU 空間を 4KB 単位に分割し、そのメモリ単位ごとにそれぞれを、最大 256KB にもおよぶ巨大な実メモリ空間にマッピングさせることです。そしてこのマッピングをつかさどる MMR の値を変えることにより、8 ビット CPU6809 が 256KB のすべての実メモリ空間を動的に利用できるになっています(図 1-3)。

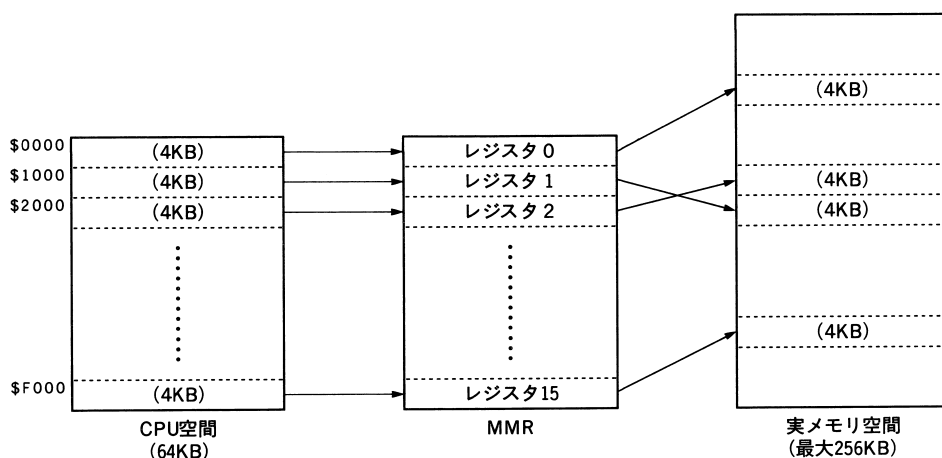


図1-3 メモリマッピング機能

一方、テキスト・ウィンドウ機能は、主として F-BASIC V3.3 が、BASIC テキストを管理するために使用します。BASIC テキストは、中間言語に変換されテキスト領域(\$07C00~\$0EFFF)に格納されます。そして必要な BASIC テキストだけが、ウィンドウ領域(CPU 空間の \$7C00 ~ \$7FFF)にうつしだされて、F-BASIC に利用されるわけです。この機能により F-BASIC V3.3 では、64KB の CPU 空間内にテキスト領域を置く必要がなくなり、16 ビット機なみの大きな BASIC のフリーエリアを実現しています(図 1-4)。

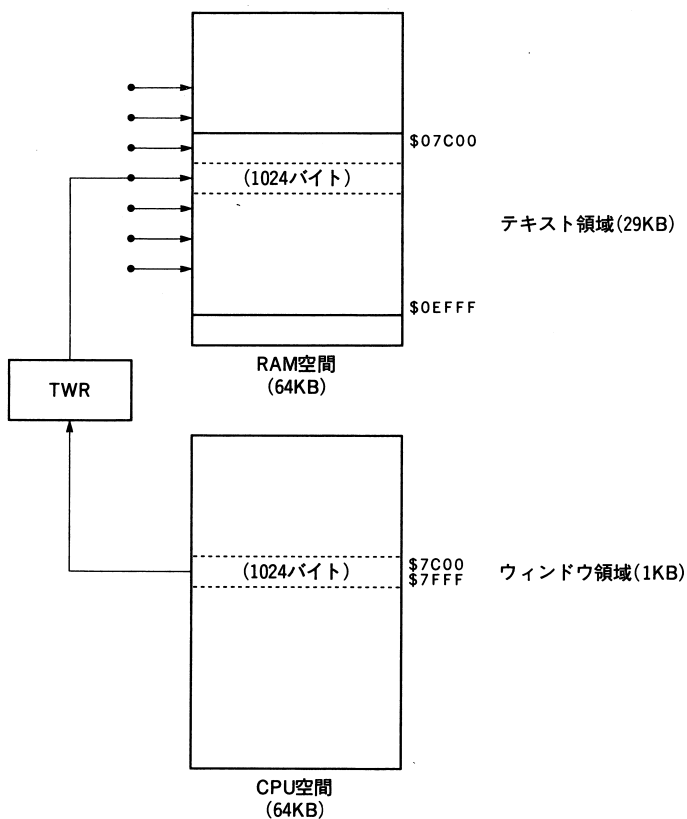


図1-4 テキストウィンドウ機能

1-1-1 裏 RAM

FM-7 および、FM77AV の F-BASIC V3.0 の動作時には、RAM 空間(64KB)の半分近くが、BASIC ROM と重なっているため使用されていません。この BASIC ROM と重なって使用されていない RAM 空間(\$8000~\$FBFF)を、裏 RAM と称しています(図1-5)。

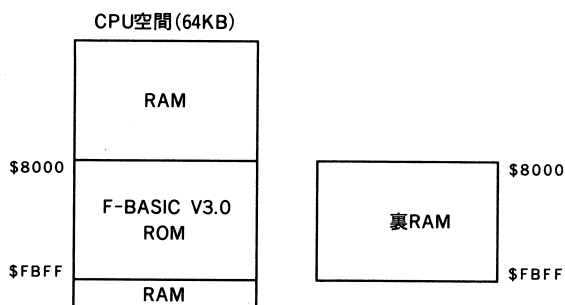


図1-5 F-BASIC V3.0動作時のCPU空間

F-BASIC V3.3は、システム起動時にシステムディスクより読み込まれ、RAM 空間、標準空間に展開されます。そのため、F-BASIC V3.3のROMは存在していません。そして、BASICが標準空間に展開されるので、裏RAMも存在していません。

この裏RAMは、F-BASIC動作中には、CPU空間から完全に切り離されています。それで、BASICテキストをNEWしても、あるいは、リセットスイッチを押したときでさえも、裏RAMの内容は消去されません。もちろん、電源を切った場合には、消去されてしまいます。

BASIC ROMと裏RAMの切り替えは、メインシステムI/Oレジスタ(\$FD0F)をリード／ライトすることでできます。\$FD0FをライトすればRAMモードが選択され、CPU空間の\$8000～\$FBFFにRAMが設定され、F-BASIC V3.0は切り離されます。また\$FD0Fをリードすれば、ROMモードが選択され、F-BASIC V3.0 ROMがCPU空間に組み込まれ、F-BASICが動作します(図1-6)。

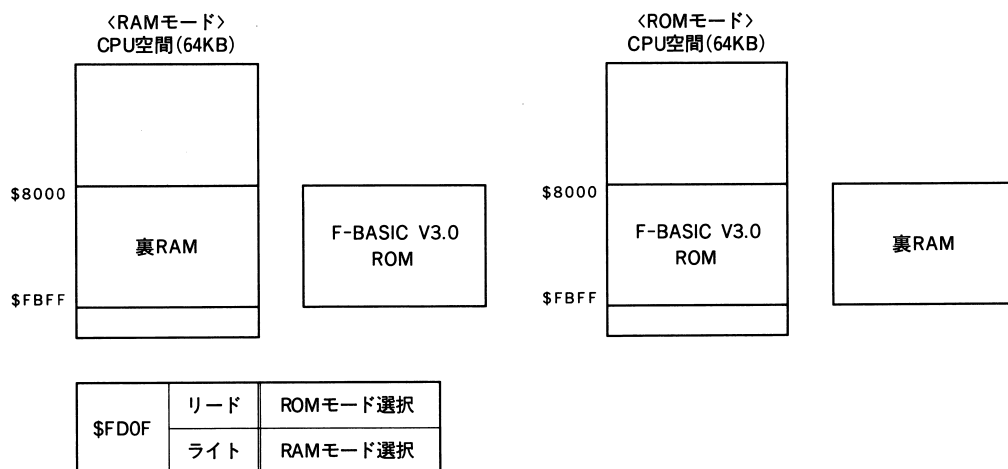


図1-6 ROMモードとRAMモード

それでは、裏RAMを実際にアクセスしてみましょう。BASICのPOKE文にて\$FD0Fをライトして、RAMモードにしてみます。書き込む値は、何であってもかまいません。

POKE &HFD0F, 1

いかがですか。暴走してしまいましたね。これは、CPU空間からBASIC ROMが切り離されてしまって、POKE文実行後の処理がおかしくなったためです。

これは、機械語モニタのMコマンドで\$FD0Fの内容を変更しても同じです(機械語モニタは、BASIC ROMの\$ABF4～\$AD53にあります)。

これでは、裏RAMへのアクセスを確認できません。そこで、マシン語の裏RAMリード／ライトルーチンを作成してみました。簡単なプログラムですから、機械語モニタで直接入力すれば

よいでしょう(リスト1-1)。

\$5000番地を実行すると、\$5100番地の内容を裏RAMの\$8000番地に書き込みます。\$5002番地を実行すると、裏RAMの\$8000番地の内容を\$5101番地に書き込みます(リスト1-2)。

リセットスイッチを押してみても、裏RAMの内容が消されないことも、確認してみてください。リセットスイッチを押すと、裏RAMリード/ライトルーチンは消されてしまいますので、もう一度裏RAMリード/ライトルーチンを入力し直す必要があります。

リスト1-1 裏RAMリード/ライトルーチン

```

00100
00110
00120
00130
00140
00150 5000
00170 5000 20 02 5004 ENTRY
00180 5002 20 00 5011
00200 5004 B7 F00F WRTUR
00210 5007 B6 5100
00220 500A B7 8000
00230 500D B6 F00F
00240 5010 39
00260 5011 B7 F00F READU
00270 5014 B6 8000
00280 5017 B7 5101
00290 501A B6 F00F
00300 501D 39
00320 5000
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=501D
PROGRAM ENTRY ADDR=5000

```

 * ウラRAM READ/WRITE *
 * (LIST 1-1) V3.0 *

OPT NOGEN
 ORG \$5000
 BRA WRTUR
 BRA READU
 STA \$F00F ウラRAM SELECT
 LDA \$5100
 STA \$8000
 LDA \$F00F BASIC-ROM SELECT
 RTS
 STA \$F00F ウラRAM SELECT
 LDA \$8000
 STA \$5101
 LDA \$F00F BASIC-ROM SELECT
 RTS
 END ENTRY

リスト1-2 裏RAMリード/ライトルーチン実行

```

POKE &H5100,123

Ready
EXEC &H5000

Ready
PRINT PEEK(&H5101)
0

Ready
EXEC &H5002

Ready
PRINT PEEK(&H5101)
123

Ready

```

1-1-2 RAM空間

この領域には、システム起動時、システムディスクより読み込まれたF-BASIC V3.3の非常駐モジュールが展開されています(図1-7)。

この非常駐モジュールは、MMR経由で、CPU空間の非常駐マッピング領域(\$8000～\$8FFF)にマッピングされ、アクセスされます。ただし、FM音源用非常駐モジュールだけは、CPU空間の\$D000～\$DFFFにマッピングされて、アクセスされます。

またテキスト領域(\$07C00～\$0EFFF)には、BASICのテキストが、中間コードに変換されて格納されます。このテキスト領域へのアクセスは、TWR経由にて行なわれています。

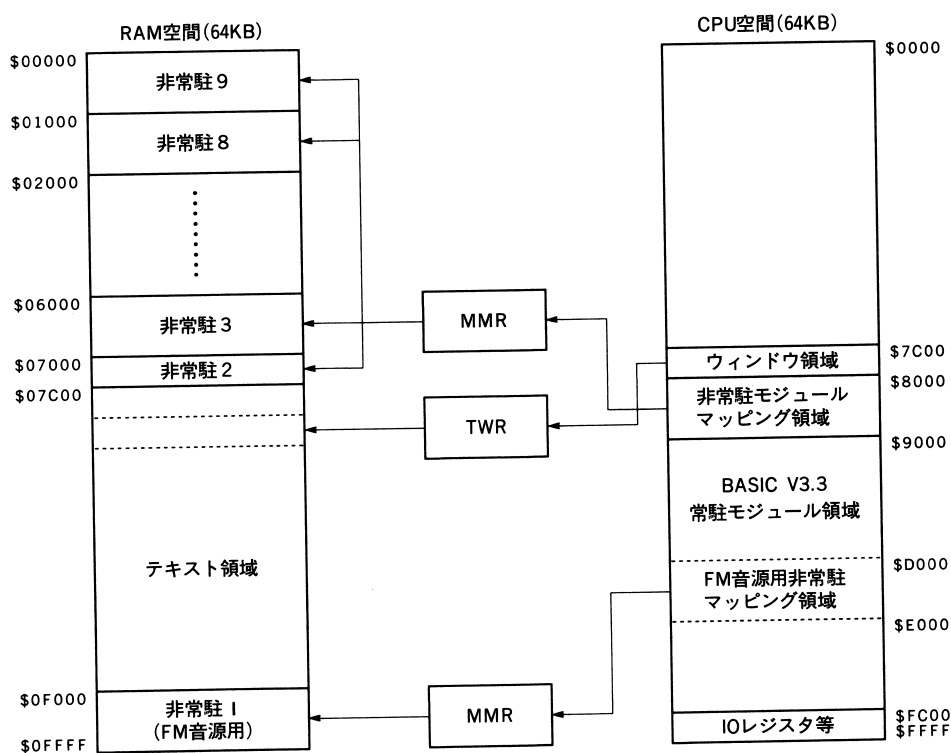


図1-7 RAM空間のメモリマップ

それではテキスト領域の内容をのぞいてみましょうまずリセットを押して、F-BASIC V3.3を起動してください(リスト1-3-A)。

リスト 1-3-A テキスト領域のダンプ

```

mon
*Q7C00
7C00 00 00 00 00 00 00 00 00
7C08 00 00 00 00 00 00 00 00
7C10 00 00 00 00 00 00 00 00
7C18 00 00 00 00 00 00 00 00
7C20 00 00 00 00 00 00 00 00
7C28 00 00 00 00 00 00 00 00
7C30 00 00 00 00 00 00 00 00
7C38 00 00 00 00 00 00 00 00
*
```

ウィンドウ領域(\$7C00~\$7FFF)の値は、すべて 00 になっています。それでは、何か BASIC のテキストを入力してみましょう。そしてもう一度、ウィンドウ領域の値を確認してください(リスト 1-3-B)。

リスト 1-3-B テキスト領域のダンプ

```

10 A=1
MON
*D7C00
7C00 00 00 00 0C 00 0A 41 E6
7C08 FE 01 01 00 00 00 00 00
7C10 00 00 00 00 00 00 00 00
7C18 00 00 00 00 00 00 00 00
7C20 00 00 00 00 00 00 00 00
7C28 00 00 00 00 00 00 00 00
7C30 00 00 00 00 00 00 00 00
7C38 00 00 00 00 00 00 00 00
.*
```

ウィンドウ領域の値が、変化していますね。これが、いま入力したテキストの中間コードです。中間コードの意味は、第 2 章を参照してください。

それでは、ちょっと、この中間コードを変えてみましょう。先程入力した BASIC のテキストが変化してしまいます(リスト 1-4)。

リスト 1-4 テキスト領域の変更

```

PUKE &H7C06.&H42
Ready
LIST
10 B=1
Ready
```

ここで注意すべき点は、ウィンドウ領域に見えているメモリが、RAM 空間のテキスト領域そのものかどうかです。テキスト領域の内容をウィンドウ領域にコピーしているのではないのです。だから、ウィンドウ領域の内容を機械語モニタにて変更したとき、BASIC のテキストが変化したのです。

それでは、テキスト・ウィンドウ機能を使わなくしたら、ウィンドウ領域はどうなるのでしょうか。テキスト・ウィンドウ機能を無効にするには、メインシステム I/O レジスタ(\$FD93)のビット6を、オフにします(リスト1-5)。

リスト1-5 テキスト・ウィンドウ機能の無効

```
POKE &HFD93,&HBF
```

```
Ready
MON
```

```
*D7C00
```

```
7C00 00 4E 45 58 54 20 77 69
7C08 74 68 6F 75 74 20 46 4F
7C10 52 00 53 79 6E 74 61 78
7C18 20 65 72 72 6F 72 00 52
7C20 45 54 55 52 4E 20 77 69
7C28 74 68 6F 75 74 20 47 4F
7C30 53 55 42 00 4F 75 74 20
7C38 6F 66 20 64 61 74 61 00
```

```
*
```

リスト1-6 メッセージテーブル

7C00	00 4E 45 58 54 20 77 69 74 68 6F 75 74 20 46 4F	NEXT without FO
7C10	52 00 53 79 6E 74 61 78 20 65 72 72 6F 72 00 52	R Syntax error R
7C20	45 54 55 52 4E 20 77 69 74 68 6F 75 74 20 47 4F	ETURN without GO
7C30	53 55 42 00 4F 75 74 20 6F 66 20 64 61 74 61 00	SUB Out of data
7C40	49 6C 6C 65 67 61 6C 20 66 75 6E 63 74 69 6F 6E	Illegal function
7C50	20 63 61 6C 6C 00 4F 76 65 72 66 6C 6F 77 00 4F	call Overflow O
7C60	75 74 20 6F 66 20 6D 65 6D 6F 72 79 00 55 6E 64	ut of memory Und
7C70	65 66 69 6E 65 64 20 6C 69 6E 65 20 6E 75 6D 62	efined line numb
7C80	65 72 00 53 75 62 73 63 72 69 70 74 20 6F 75 74	er Subscript out
7C90	20 6F 66 20 72 61 6E 67 65 00 44 75 70 6C 69 63	of range Duplic
7CA0	61 74 65 20 64 65 66 69 6E 69 74 69 6F 6E 00 44	ate definition D
7CB0	69 76 69 73 69 6F 6E 20 62 79 20 7A 65 72 6F 00	ivision by zero
7CC0	49 6C 6C 65 67 61 6C 20 64 69 72 65 63 74 00 54	Illegal direct T
7CD0	79 70 65 20 6D 69 73 6D 61 74 63 68 00 4F 75 74	ype mismatch Out
7CE0	20 6F 66 20 73 74 72 69 6E 67 20 73 70 61 63 65	of string space
7CF0	00 53 74 72 69 6E 67 20 74 6F 6F 20 6C 6F 6E 67	String too long
7D00	00 53 74 72 69 6E 67 20 66 6F 72 6D 75 6C 61 20	String formula
7D10	74 6F 6F 20 63 6F 6D 70 6C 65 78 00 43 61 6E 27	too complex Can'
7D20	74 20 63 6F 6E 74 69 6E 75 65 00 55 6E 64 65 66	t continue Undef
7D30	69 6E 65 64 20 75 73 65 72 20 66 75 6E 63 74 69	ined user functi
7D40	6F 6E 00 4E 6F 20 52 45 53 55 4D 45 00 52 45 53	on No RESUME RES
7D50	55 4D 45 20 77 69 74 68 6F 75 74 20 65 72 72 6F	UME without erro
7D60	72 00 55 6E 70 72 69 6E 74 61 62 6C 65 20 65 72	r Unprintable er
7D70	72 6F 72 00 4D 69 73 73 69 6E 67 20 6F 70 65 72	ror Missing oper
7D80	61 6E 64 00 46 4F 52 20 77 69 74 68 6F 75 74 20	and FOR without
7D90	4E 45 58 54 00 57 48 49 4C 45 20 77 69 74 68 6F	NEXT WHILE witho
7DA0	75 74 20 57 45 4E 44 00 57 45 4E 44 20 77 69 74	ut WEND WEND wit

おかしな値がなっていますが、アスキーコードに変換するとはっきりします。ここは、F-BASIC のメッセージテーブルになっているのです(リスト 1-6)。

F-BASIC では、メッセージが必要になると、テキスト・ウィンドウ機能を無効にして、このメッセージテーブルを参照しています。

それでは、このメッセージテーブルの内容を書きかえてみましょう(リスト 1-7)。

リスト 1-7 メッセージテーブルの変更

```
A$="マチカ"ツテルヨ"+CHR$(0):POKE &HFD93,&HBF:FOR I=0 TO 8:POKE &H7C12+I,ASC(MID$(A$,I+1,1)):NEXT:POKE &HFD93,&HFF
```

```
Ready
10 BBBB
R.
```

```
マチカ"ツテルヨ in 10
Ready
```

1-1-3 サブ CPU 空間

メイン CPU が、サブシステムを直接にアクセスするダイレクトアクセスモードのとき、サブ CPU 空間が割り当てられる領域です。ダイレクトアクセスモードでないときには、この空間は存在しません。

ダイレクトアクセスモードにするには、サブ CPU を HALT します。

ダイレクトアクセスモード時には、サブシステムの 64KB の全メモリ領域を、メインシステムより直接アクセスできます。アクセスには、MMR を通して、メイン CPU 空間にマッピングして行ないます。

メイン CPU は、ダイレクトアクセスにより、次のことができるようになります。

- ① VRAM のリード／ライト
- ② サブシステム I/O レジスタのリード／ライト
- ③ コンソール RAM のリード／ライト
- ④ キャラクタ ROM のリード

ダイレクトアクセスのサンプルプログラムを、リスト 1-8 に示します。これは、INS LED ランプを点灯、消灯するものです。EXEC &H5000 でランプ点灯、EXEC &H5002 でランプ消灯します。ただし、これはランプの点灯、消灯だけで、INSERT モードは変化しません。

リスト1-8 ダイレクトアクセス

```

00001 *****
00011 *      INS LED ON/OFF      *
00021 *      ( LIST 1-8 )  V3.3  *
00031 *****
00041          OPT      NOGEN
00051 5000          ORG      $5000
00061
00071          *
00081 5000 20      02      5004 ENTRY  BRA      INSON
00091 5002 20      17      501B        BRA      INSOFF
00101          *
00111 5004 8D      5032        INSON  JSR      SUBHLT  SUB HALT
00121          *                                DIRECT ACCESS OK
00131 5007 86      1D                                LDA      #$1D  MMR SET
00141 5009 87      FD8D          STA      $FD8D
00151 500C 7F      D410          CLR      $D410  オンリエンサツ DISSABLE
00161 500F 86      D40D          LDA      $D40D  INS LED ON
00171 5012 86      3D                                LDA      #$3D  MMR RECOVER
00181 5014 87      FD8D          STA      $FD8D
00191 5017 8D      5044          JSR      SUBMOV  SUB MOVE
00201 501A 39                                RTS
00211          *
00221 5018 8D      5032        INSOFF JSR      SUBHLT
00231 501E 86      1D                                LDA      #$1D
00241 5020 87      FD8D          STA      $FD8D
00251 5023 7F      D410          CLR      $D410
00261 5026 87      D40D          STA      $D40D  INS LED OFF
00271 5029 86      3D                                LDA      #$3D
00281 502B 87      FD8D          STA      $FD8D
00291 502E 8D      5044          JSR      SUBMOV
00301 5031 39                                RTS
00311          *
00321 5032 86      FD05        SUBHLT LDA      $FD05
00331 5035 2B      FB      5032        BMI      SUBHLT
00341 5037 1A      50                                ORCC      #$50
00351 5039 86      80                                LDA      #$80
00361 503B 87      FD05          STA      $FD05
00371 503E 86      FD05          LDA      $FD05
00381 5041 2A      FB      503E        BPL      *-3
00391 5043 39                                RTS
00401 5044 F6      FC80        SUBMOV LDB      $FC80
00411 5047 CA      80                                ORB      #$80
00421 5049 F7      FC80          STB      $FC80
00431 504C 4F                                CLRA
00441 504D B7      FD05          STA      $FD05
00451 5050 1C      AF                                ANDCC   #$AF
00461 5052 39                                RTS
00471          5000          END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5052
PROGRAM ENTRY ADDR=5000

```

1-1-4 拡張 RAM 空間

オプションの拡張 RAM カード用の領域です。それで、拡張 RAM カードが装着されていないと意味がありません。この領域をアクセスするときには、MMR 経由で行ないます(図1-8)。

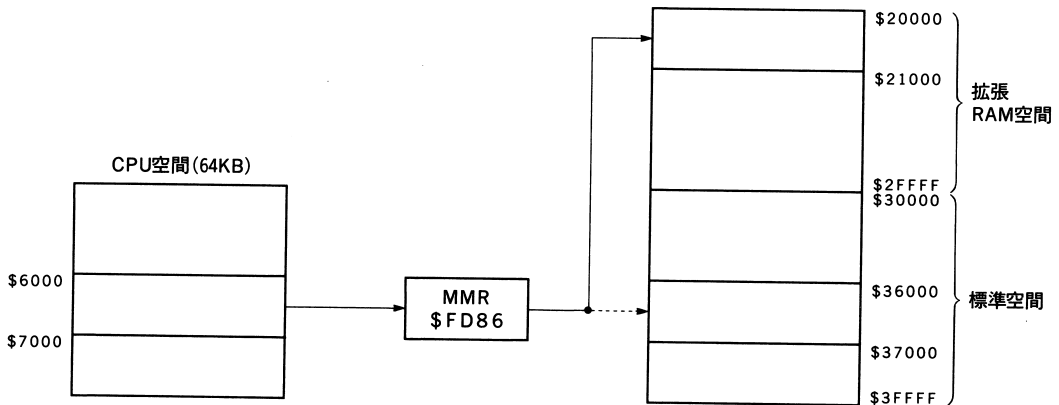


図1-8 拡張RAM空間のアクセス

それでは、MMRを設定してこの拡張RAMをアクセスしてみます。MMRを設定するには、メインシステム I/O レジスタ (\$FD80～\$FD8F) に値をセットします。

POKE & HFD86, & H20

これによって、拡張RAM空間の\$20000～\$20FFFの領域が、CPU空間の\$6000～\$6FFFの領域にマッピングされます。たとえばメインCPUが、\$6000番地に書き込むと、それは、拡張RAM空間の\$20000番地に書き込まれます。

F-BASIC V3.3には、裏RAMがありません。しかしこの拡張RAM空間を使えば、裏RAM以上に便利な使い方ができます。なぜならば、F-BASIC V3.0での裏RAMはBASICのPEEK、POKE文で直接アクセスできません。しかしこの拡張RAM空間は、MMRを正しく設定すれば、BASICのPEEK、POKE文で直接アクセスできるからです。また、この拡張RAMの内容は、裏RAMと同様にRESETスイッチによって内容が消されることはありません。

1-1-5 標準空間

システム起動時および、リセット時のCPU空間です。F-BASIC V3.0では、この空間のみを使用します(図1-9)。

F-BASIC V3.3起動時には、イニシエータROMが切り離され、F-BASIC V3.3の常駐モジュールが展開されます(図1-10)。

MMRモード時、この標準空間についても、他の空間と同様に再配置することができます。ただし\$3FC00～\$3FFFFの領域は、MMRの値にかかわらず、常時CPU空間の\$FC00～\$FFFFに対応する常駐領域です。

\$7C00～\$7FFFのウィンドウ領域には、テキスト・ウィンドウ機能のため、RAM空間のテキスト領域の内容がうつしだされています。それで、メッセージテーブルをアクセスするには、テ

第1章 メモリマップ

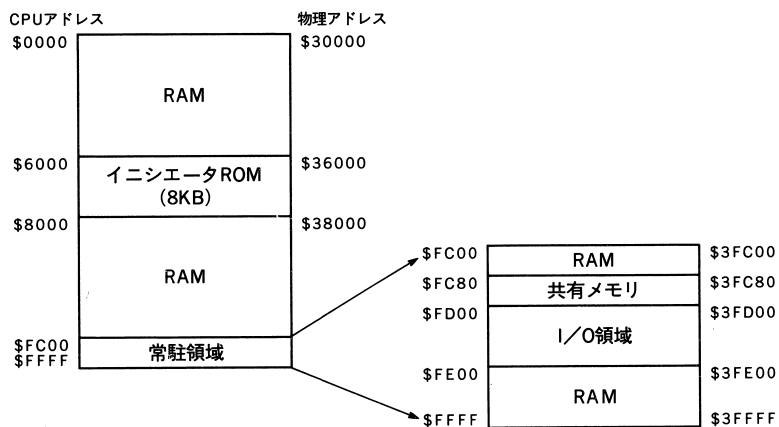


図1-9 システム起動時およびリセット時のメモリマップ

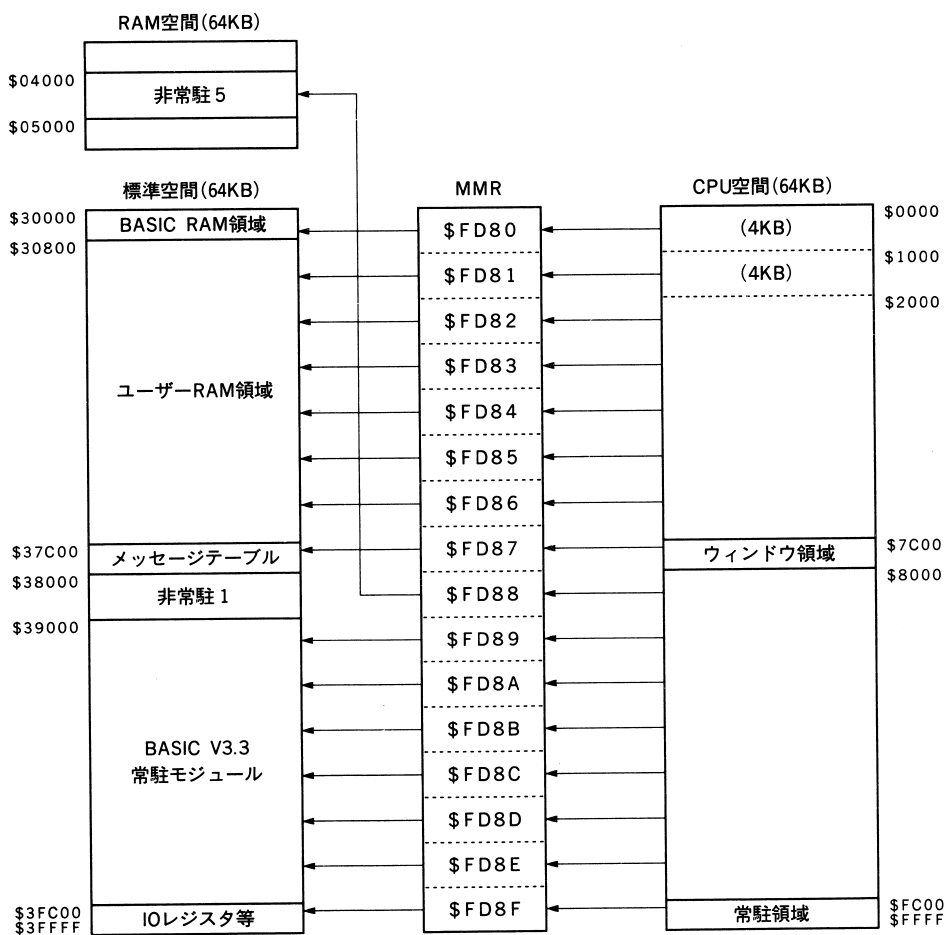


図1-10 F-BASIC V3.3起動時のメモリマップ

キスト・ウィンドウ機能を無効にします(RAM 空間の項, 参照)。

F-BASIC V3.3 が非常駐モジュール 1~9 をアクセスするときには, MMR(\$FD88)の値を設定して CPU 空間の \$8000~\$8FFF にうつしだして使用しています。

1-2 イニシエータ ROM

FM77AV のシステム起動時および, リセット時には, 図 1-11 に示すようにイニシエータ ROM が, CPU 空間の \$6000~\$7FFF に割り当てられます。

\$6000	イニシエータ域
約\$6200	あ き
\$6C00	音色データ 77エントリ (1エントリは34バイト)
\$763A	音色データリザーブ
\$7800	BASIC BOOT
\$7A00	320KB DOS BOOT
\$7C00	あ き
\$7FFF	

図1-11 イニシエータROMのメモリマップ

システム起動時および, リセット時には, イニシエータ ROM によってシステム初期化処理がなされます。

- ① FM 音源, アナログパレットなどのハードウェアリソースの初期設定
- ② サブシステムのモード設定
- ③ メモリ配置の決定
- ④ ブートプログラムの展開
- ⑤ システムプログラムへの制御の移行

そして F-BASIC が起動したときには, イニシエータ ROM はディセーブルされ, \$0000~\$7FFF の領域は, RAM になります。

イニシエータ ROM には, FM 音源の音色データも入っており, F-BASIC V3.3 が音色データを取り出すときには, イニシエータ ROM がアクセスされます。

それでは, イニシエータ ROM の音色データの内容を見てみることにしましょう(図 1-12)。

WRTPRMによる書き込みデータ										
スロット										
ビット	7	6	5	4	3	2	1	0		
0	—	DT DeTune			MULTI MULTiple				1	○
1	—								3	○
2	—								2	○
3	—								4	○
4	—	TL Total Level							1	—
5	—								3	—
6	—								2	—
7	—								4	—
8	KS Key Scale		—	AR Attack Rate				1	○	
9			—					3	○	
10			—					2	○	
11			—					4	○	
12	—		DR Decay Rate				1	○		
13							3	○		
14							2	○		
15							4	○		
16	—		SR Sustain Rate				1	○		
17							3	○		
18							2	○		
19							4	○		
20	SL Sustain Level			RR Release Rate				1	○	
21								3	○	
22								2	○	
23								4	○	
24	—	Feed Back		Connect				—		
25	Flags								—	
26	Key Scale Depth								—	
27	PMS			AMS					—	
28	AMD								—	
29	PMD								—	
30	LFO frequency								—	
31	Delay Time								—	
32	Wave Form								—	
33	Sync Flag								—	

図1-12 音色データの形式

イニシエータ ROM を有効にするには、\$FD10 のビット 1 を "0" にします。しかし、イニシエータ ROM のアドレスが F-BASIC のスタックエリアと重なっていますので、不用意にイニシエータ ROM を有効にすると暴走してしまいます。そこで、CLEAR 文でスタックエリアを \$5000 に設定してから、\$FD10 の値を変えてみます(リスト 1-9)。

リスト 1-9 FM 音源音色データ

```
CLEAR .&H5000:Poke &HFD10.0:MON
```

```
*D6C00
```

```
6C00 02 02 02 02 19 0F 0F 0F
6C08 8F 92 92 92 04 03 03 03
6C10 00 00 00 00 FA FA FA FA
6C18 3D 04 00 2F 00 0A 20 04
6C20 02 00 42 02 12 34 20 09
6C28 09 22 0D 0E 4E 4B 1E 1E
6C30 1E 1E 0C 0B 01 0A 09 09
6C38 0A 09 3D 00 00 1F 00 00
```

```
*
```

1-3 サブシステムのメモリマップ

FM77AV のサブシステムのメモリマップを、図 1-13、図 1-14、図 1-15 に示します。図 1-13 が、FM-7 互換のサブシステムのメモリマップです。

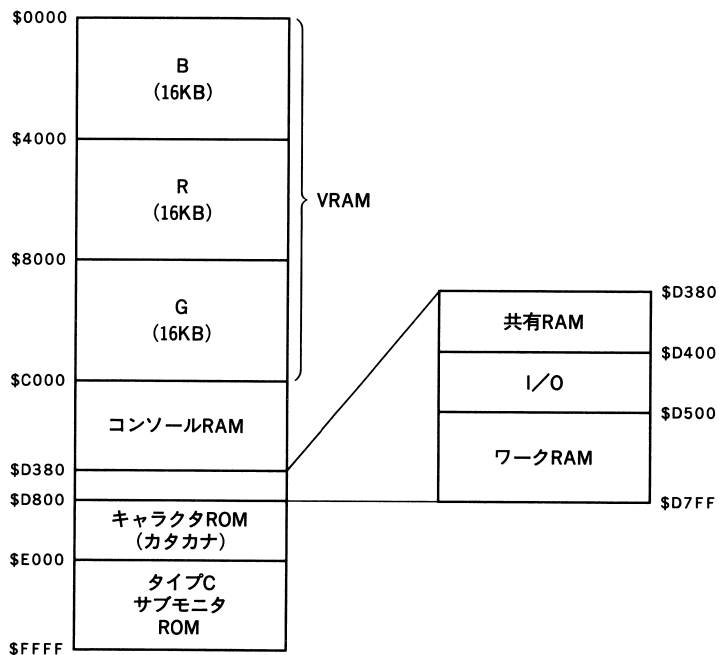


図1-13 F-BASIC V3.0のメモリマップ

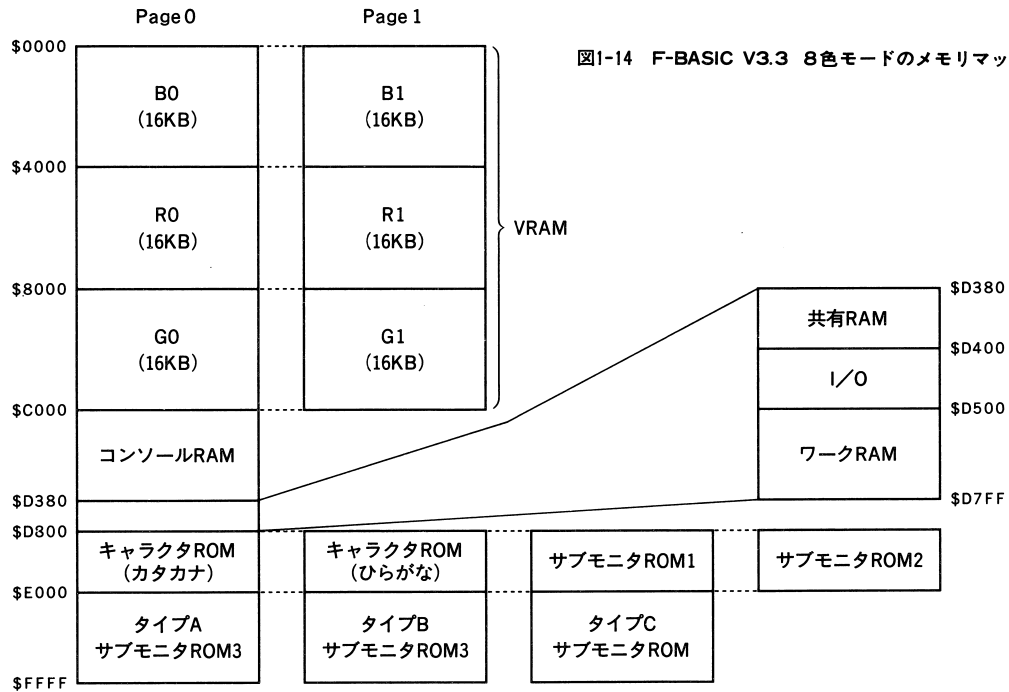


図1-14 F-BASIC V3.3 8色モードのメモリマップ

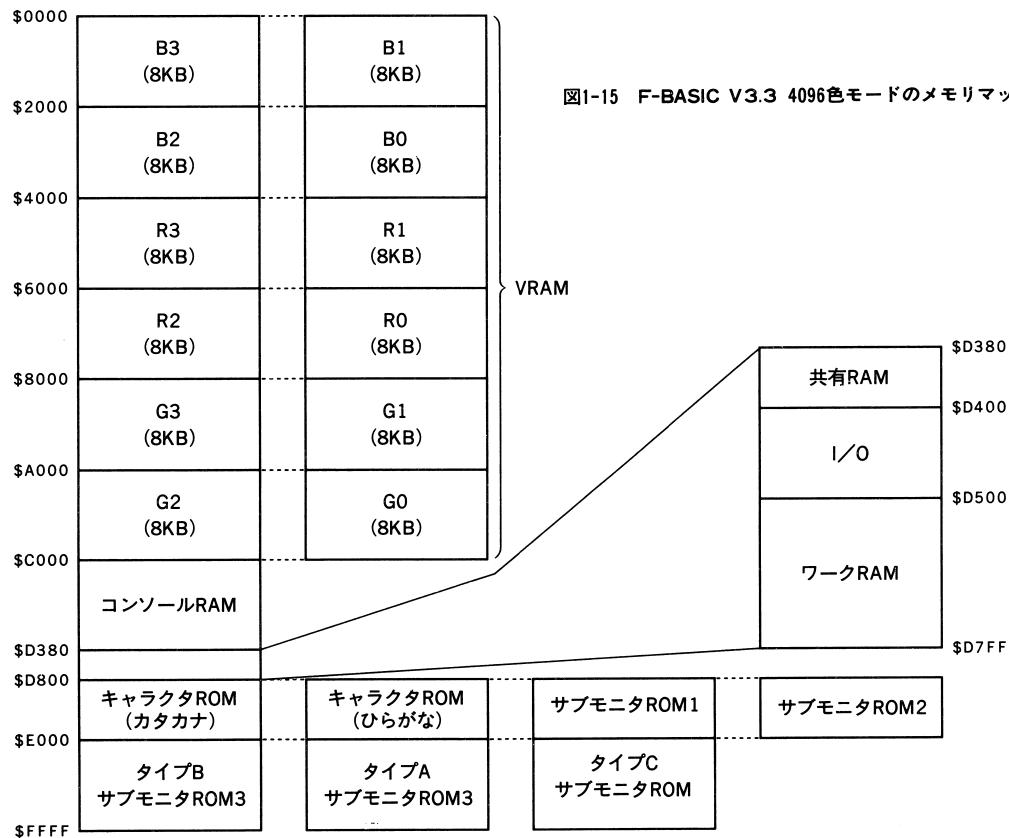


図1-15 F-BASIC V3.3 4096色モードのメモリマップ

1-3-1 サブモニタ ROM の選択

FM77AV サブシステムのサブモニタ ROM には、タイプ A、タイプ B、タイプ C の 3 種類があり、グラフィックモードに応じて切り替えて使われます。

F-BASIC V3.0 で起動すると、タイプ C (FM-7 互換) のサブモニタ ROM が選択されます。そして、F-BASIC V3.3 で起動すると、タイプ A が選択され、SCREEN@ 命令にて、タイプ B に切り替えることができます。

SCREEN@ 0  タイプ A (8 色 2 画面モード)

SCREEN@ 1  タイプ B (4096 色モード)

このサブモニタ ROM の選択は、メインシステム I/O レジスタ (\$FD12 と \$FD13) の値を、図 1-16 のように設定することによって行なわれます。

サブモニタ ROM	解 像 度	カラー	画 面	\$FD13		\$FD12 ビット 6
				ビット 1	ビット 0	
タイプ C (FM-7 互換モード)	640×200 ドット	8 色	1 画面	0	0	0
タイプ A	640×200 ドット	8 色	2 画面	0	1	0
タイプ B	320×200 ドット	4096 色	1 画面	1	0	1

図 1-16 サブモニタ ROM の選択

I/O レジスタの設定は、F-BASIC のバージョンとは無関係に行なうことができます。ですから、F-BASIC V3.3 で、タイプ C (FM-7 互換) のサブモニタ ROM を動作させることができます。

では、確認してみましょう。まず F-BASIC V3.3 を起動して、リスト 1-10 のように入力して、その結果を記憶してください。その後で、リスト 1-11 を実行してみてください。

結果が違いますね。後者では、タイプ C のサブモニタ ROM が動作しているため、8 色 2 画面の切り替えができないからです。他にも V3.3 のグラフィック拡張機能がどうなるか、ためしてみてください。

リスト 1-10 V3.3 でタイプ A を動作

```
SCREEN@ 0
```

```
Ready
SCREEN ..1.1
```

```
Ready
LINE (0.0)-(20.20),PSET,1,BF
```

```
Ready
SCREEN ..0.0
```

```
Ready
```

リスト 1-11 V3.3 でタイプ C を動作

```
SCREEN 0

Ready
POKE &HFD13.0

Ready
SCREEN ..1.1

Ready
LINE (0.0)-(20.20).PSET.1.BF

Ready
SCREEN ..0.0

Ready
```

しかし、これはあまり意味のあることではないでしょう。なぜならば、V3.3 は V3.0 の機能拡張バージョンで、V3.0 でできて V3.3 でできなくなった機能はないからです。しかし、FM-7 上で開発したサブシステムのプログラムを、F-BASIC V3.3 で動作させたいときには、有効なのかもしれません。

1-3-2 キャラクタ ROM の選択

FM77AV では、キャラクタ ROM として、カタカナとひらがなの 2 種類を持っています。そして、F-BASIC V3.3 では、カタカナとひらがなを、CONSOLE 文にて切り替えて使うことができます。

CONSOLE ,,,,0 標準(カタカナ, グラフィック)モード
CONSOLE ,,,,1 ひらがな(ひらがな, グラフィック)モード

この切り替えは、サブシステム I/O レジスタ(\$D430)のビット 0, ビット 1 によって行なわれます。

\$D430		ROM
ビット 1	ビット 0	
0	0	カタカナ
0	1	ひらがな
1	0	サブモニタ ROM 1
1	1	サブモニタ ROM 2

図1-17 キャラクタROMの選択

マシン語での設定には、ダイレクトアクセスで行うか、サブシステムに SELECT DISPLAY MODE コマンドを送るかします。

1-3-3 VRAM のページ切り替え

(1) ページ切り替え

FM77AV では、48KB の VRAM を 2 枚持っていて、ページ切り替えによってそれぞれを独立にアクセスします。

CRT 画面に表示されるページをディスプレイページ、CPU からアクセスされるページをアクティブページといいます。FM77AV8 色 2 画面モード(サブモニタ ROM タイプ A 選択)のときには、ディスプレイページ、アクティブページを独立して選択できます。そして、アクティブページに指定されていないページの内容は、すべてのグラフィックコマンド処理において影響されません。

しかし、4096 色モード(サブモニタ ROM タイプ B 選択)のときには、1 ページしかないので、ディスプレイページ、アクティブページの指定は意味を持ちません。

これは見方を変えると、タイプ A のサブモニタ ROM は、VRAM のバンク切り替えをせずに処理するのに対して、タイプ B のサブモニタ ROM は、VRAM のバンク切り替えをしながら処理しているということです。

BASIC では、SCREEN 文にてアクティブページ、ディスプレイページを指定します。

```
SCREEN ,,0 ..... アクティブページにページ 0 を選択
SCREEN ,,1 ..... アクティブページにページ 1 を選択
SCREEN ,,,0 ..... ディスプレイページにページ 0 を選択
SCREEN ,,,1 ..... ディスプレイページにページ 1 を選択
```

アクティブページにページ 0、ディスプレイページにページ 1 を選択したとします。すると、以後画面への表示が行なわれなくなります。しかし、ページ 0 には確かに書き込まれています。それが、ディスプレイ画面に表示されなくなっただけなのです。ですから、ディスプレイページにページ 0 を選択すると、書き込んだ内容が表示されます(リスト 1-12)。

リスト 1-12 ページ切り替え

```
SCREEN@ 0:SCREEN ,,0.1:LINE (0.0)-(50.50),PSET,7,BF:SCREEN ,,,0
Ready
SCREEN ...1

Ready
SCREEN ...0

Ready
```

ディスプレイページとアクティブページの指定は、通常は同じにします。しかし、異なる指定をすることにより、おもしろい描画を行なうことができます。

たとえば、ページ0の図形を見せている間に、ページ1に描画を行ないます。そして、描画が終わったらページ1をディスプレイページにします。これにより、描画の途中を見せないで、つぎつぎに図形を表示することができます。

しかし、このページ切り替えでは、2ページの画面の重ね合わせ表示はできません。

このページ切り替えをマシン語で行なうには、サブシステム I/O レジスタ(\$D430)を設定します。この設定には、ダイレクトアクセスで行なうか、サブシステムの SELECT DISPLAY MODE コマンドを利用します(図1-18)。

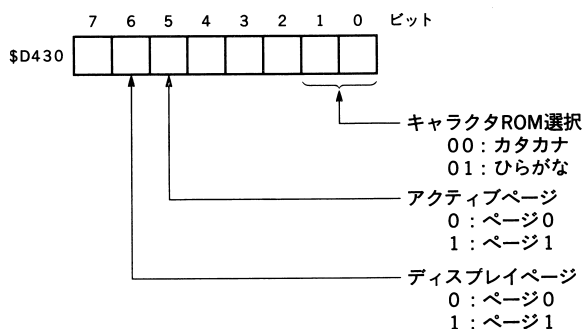


図1-18 ページ切替え

(2) マルチページ

FM-7シリーズでは、VRAMのR(レッド), G(グリーン), B(ブルー)3要素へのアクセス方法を、それぞれ独立に指定できます。

SCREEN[アクティブVRAMコード][, ディスプレイVRAMコード]

アクティブVRAMコードは、CPUからの画面上への動作に対して、RGBのどのVRAMがアクセス可能かを指定するものです(図1-19)。

アクティブ VRAMコード	2 1 0 ビット	意 味
	G R B	
0	0 0 0	すべてのVRAMがアクセス禁止
1	0 0 1	Bのみがアクセス可
2	0 1 0	Rのみがアクセス可
3	0 1 1	BとRのみがアクセス可
4	1 0 0	Gのみがアクセス可
5	1 0 1	BとGのみがアクセス可
6	1 1 0	RとGのみがアクセス可
7	1 1 1	すべてのVRAMがアクセス可

図1-19 アクティブVRAMコード

アクティブ VRAM コードを指定すると、以後のステートメントにおけるパレットコードに、注意しなければなりません。たとえば、

SCREEN 2 

LINE(0,0)-(50,50), PSET, 7, BF 

を実行すると、白のパレットを指定したにもかかわらず、R(レッド)の VRAM のみがアクセスされ、赤いボックスが描かれてしまいます。そして、アクティブ VRAM コードを元に戻しても、赤いボックスのままです。

SCREEN 7 

一方、ディスプレイ VRAM コードは、RGB のどの VRAM を画面に表示するかを、指定します(図 1-20)。

ディスプレイ VRAMコード	2 1 0 ビット	意 味
	  	
0	0 0 0	どのVRAMも表示しない
1	0 0 1	Bのみを表示
2	0 1 0	Rのみを表示
3	0 1 1	BとRのみを合成して表示
4	1 0 0	Gのみを表示
5	1 0 1	BとGのみを合成して表示
6	1 1 0	RとGのみを合成して表示
7	1 1 1	すべてのVRAMを表示

図1-20 ディスプレイVRAMコード

アクティブ VRAM コードとディスプレイ VRAM コードの違いについて、よく理解しておく必要があります。

SCREEN 7,2 

LINE(0,0)-(50,50), PSET, 7, BF 

を実行すると、赤いボックスが描かれます。これは、アクティブ VRAM コードの例と同じですが、ここでディスプレイ VRAM コードを元に戻すと、白いボックスに変化してしまいます。

SCREEN 7,7 

ディスプレイ VRAM コード、アクティブ VRAM コードを、マシン語で設定するには、メインシステム I/O レジスタ(\$FD37)に値を設定します(図 1-21)。

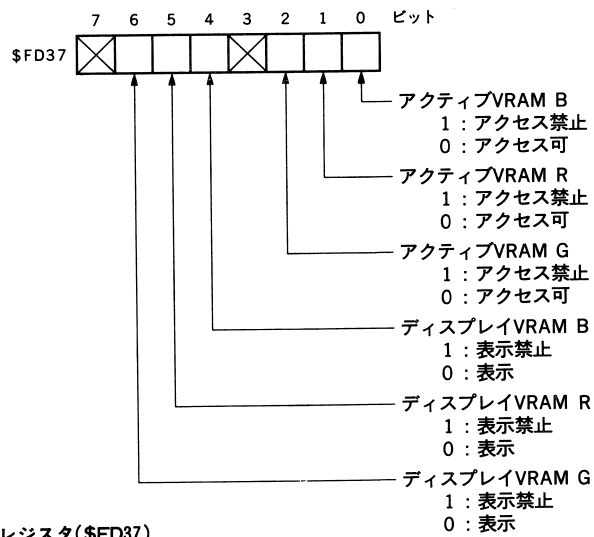


図1-21 マルチページレジスタ(\$FD37)

FM77AV8色2画面モードでは、マルチページとページ切り替えの指定を独立にできます。ですから、たとえば、

SCREEN 2,,1

を指定すると、VRAM の R1 のみがアクセス可となります。そして、

SCREEN ,4,,0

を指定すると、VRAM の G0 の内容のみが表示されます。つまり、6 画面を独立に扱うことが可能です(図 1-22)。

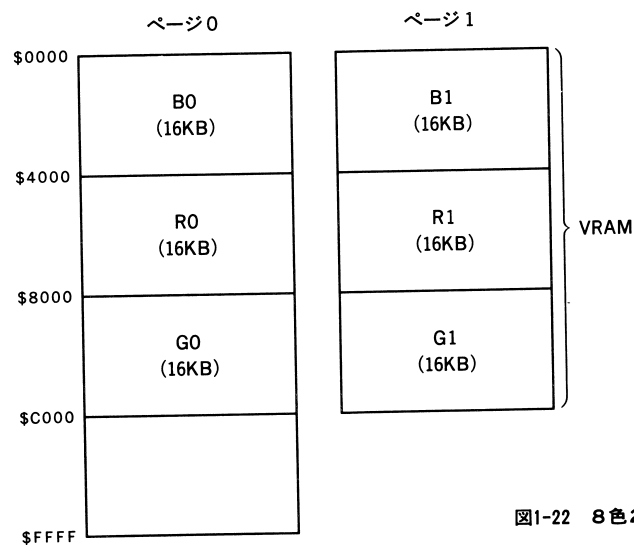


図1-22 8色2画面モードのVRAM

FM77AV4096 色モードでマルチページを指定すると、(B0, B1, B2, B3), (R0, R1, R2, R3), (G0, G1, G2, G3)の3つの VRAM 領域に対して、作用します。そしてその場合、16 色 3 画面としての扱いが可能となります(図 1-23)。

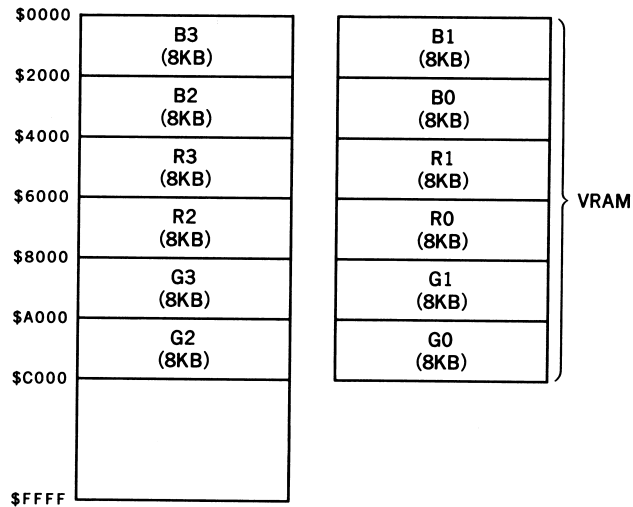


図1-23 4096色モードのVRAM

1-4 共有 RAM

メインシステムの \$FC80～\$FCFF の 128 バイトと、サブシステムの \$D380～\$D3FF の 128 バイトは、メイン CPU とサブ CPU で共有するメモリ領域で、共有 RAM と呼ばれます。

たとえば、メイン CPU にて \$FC80 番地に \$AA というデータを書き込めば、サブ CPU が、\$D380 番地をアクセスすることによって、同じ \$AA というデータを読み込むことが可能です。

ただし、メイン CPU とサブ CPU が、同時に共有 RAM にアクセスすることはできず、必ずどちらか一方のみしかアクセスできません。

共有 RAM へのアクセスは、第 4 章で説明します。

1-5 メモリモード(DOS/BASIC)

FM-7 シリーズでは、システム起動時のモードとして、BASIC モードと DOS モードの 2 種類を選択して使用するようになっています。動作させたいソフトウェアの種類によってどちらかを選ぶわけですが、この項では、それぞれのモードでシステムを起動した場合の内部処理について説明します。

6809CPU は、RESET がかけると RESET ベクトル(\$FFFE, \$FFFF)の示すアドレスへジャンプします。FM-7 シリーズでは、このアドレスが \$FE00 となっており、BOOT ROM の先頭から実行が始まるようになっていました。そして、この BOOT ROM が、モードスイッチの選択によってそれぞれ専用のものに切り替わるわけです。

FM77AV では、\$FE00 が実行される前にもう 1 ステップ処理が入ります。まず、RESET されると、イニシエータ ROM にジャンプして、FM 音源やアナログパレット等のイニシャライズが行なわれます。そして、FM77AV には BOOT ROM が存在しないので、イニシエータ ROM の中の BOOT ROM と同じ内容のものを \$FE00 番地以降に転送して、\$FE00 番地にジャンプするのです。

BOOT ROM が BASIC 用の場合の主な動作は、次のようになります。

- ① BREAK キーが押されているかチェックして、押されていればホットスタートの処理をする。
- ② ディスクが接続されているかチェックし、接続されていれば、ドライブ 0、トラック 0、セクタ 1 の内容(IPL)を \$100 番地以降に読み込みます。
- ③ RS-232C 等のインターフェースをイニシャライズします。

ここで、IPL が正常に読めた場合には、その先頭である \$100 番地にジャンプします。ディスクが接続されていなかったり、IPL の読み込みに失敗した場合は、ROM BASIC が起動されます。

DOS モードの場合、ドライブ 0、トラック 0、セクタ 1 および 2 の内容(IPL)が、\$300 番地以降に読み込まれます。そして、IPL が正常に読み込まれると、\$300 番地の IPL にジャンプします。IPL の読み込みに失敗したときは「ピー」というブザー音が鳴ってとまってしまいます。

IPL では、動作させたいプログラムを読み込んで、そのプログラムに制御をわたします。たとえば、F-BASIC V3.0 の IPL の動作は次のようになります。

- ① トラック 0、セクタ 3 の ID セクタを読んで、F-BASIC で使えるものかチェックします。
- ② トラック 0、セクタ 15、16 の DISK BASIC イニシャライズルーチンを、\$6E00 番地以降に読み込みます。
- ③ トラック 0、セクタ 17~32 の DISK コードを、\$7000 番地以降に読み込みます。
- ④ すべてうまく読めた場合は、X レジスタにイニシャライズルーチンの先頭アドレス(\$6E00)、A レジスタに 0 以外の値をセットして、BASIC のコールドスタート(\$848B)にジャンプします。読み込みに失敗したときは A レジスタに 0 がセットされ、コールドスタートにジャンプします(ROM BASIC が起動されます)。

それでは最後に、DOS モードで F-BASIC V3.0 を起動する IPL のサンプルプログラムを載せておきます。この場合、IPL の中で強制的に ROM モードにしてしまうことも可能です(F-BASIC V3.0 L2.0 では実際にそうなっている)。しかしこのプログラムでは、ROM BASIC を含めてすべての BASIC 本体を RAM へ読み込むようにしました。

BASIC を拡張したり、改造したりしたいときには、すべてが RAM に存在していますので便利ではないでしょうか。

この IPL プログラムでは、BOOT ROM の DISK アクセスルーチンを使用しています。このルーチンの使用方法は BIOS とほぼ同じですが、エントリアド레스が次のようになっています。

* \$FE02 リストア
 * \$FE05 1 セクタライト
 * \$FE08 1 セクタリード

このルーチンの注意点としては、DP に \$FD をセットすること、レジスタの保存をユーザー側で行なうこと、エラーステータスが A レジスタに返されることです。

IPL プログラムの作成、実行は次の手順で行なってください。

① SYSDSK のユーティリティを使って、F-BASIC V3.0 のシステムディスクを作成してください。

② リセットして V3.0 が起動することを確認します。

③ DSKINI 0

Are you sure (Y or N)? Y ☐

を入力してディスクを初期化する。

④ SAVEM"任意のファイル名",&H8000,&HFBFF,&H8000 ☐ と入力して ROM BASIC をセーブする。

⑤ リスト 1-13 を入力して、EXEC &H5000 ☐ で実行する。

リスト 1-13 RAM BASIC(V3.0)の起動

ADR :	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
5000 :	8E	50	07	6E	9F	FB	FA	09	00	50	0F	00	01	00	00	20	:	70
5010 :	24	0A	00	6E	00	00	0F	00	00	02	0A	00	70	00	00	01	:	28
5020 :	01	00	10	0A	00	7F	FB	02	01	00	00	70	08	00	00	00	:	1D
5030 :	00	00	00	00	08	32	8C	D7	B7	FD	0F	C6	FD	1F	9B	30	:	0D
5040 :	8C	E1	8D	1F	30	8C	D3	8D	1A	30	8C	C5	8D	15	4F	1F	:	E0
5050 :	8B	43	8E	6E	00	6E	9F	FB	FE	34	10	30	8C	CE	BD	FE	:	59
5060 :	02	35	10	34	10	8D	FE	08	35	10	24	0C	6A	8C	C5	26	:	A4
5070 :	E8	86	81	B7	FD	03	20	FE	6C	02	E6	05	5C	C1	11	25	:	70
5080 :	0C	C6	01	A6	06	88	01	A7	06	26	02	6C	04	E7	05	6A	:	A3
5090 :	08	26	00	39	00	00	00	00	00	00	00	00	00	00	00	00	:	37
50A0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50B0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50C0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50D0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50E0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50F0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00

[cs] : C8 25 94 3D EA EE 21 17 77 EB D0 B5 59 36 82 23 : E9

SAVEM "L1-13M",&H5000,&H5093,&H5000

以上で完了です。リセットして起動してみましょう。MODE スイッチが DOS でも、あるいは、BASIC でも、RAM に展開された F-BASIC V3.0 が起動します。なお、この IPL プログラムでは ID セクタのチェックは行なっていません。

1-6 裏 RAM 活用

F-BASIC V3.0 の動作時には、RAM (64KB) の半分近くが、BASIC ROM と重なっているため使用されていません。そこで、通常は未利用なこの裏 RAM をマシン語ルーチンやデータ等の一時格納場所として有効活用するための 4 つのユーティリティを作成しました。

- ① 裏 RAM へ POKE (リスト 1-14-1)
- ② 裏 RAM を PEEK (リスト 1-14-2)
- ③ 裏 RAM を SAVEM (リスト 1-14-3)
- ④ 裏 RAM へ LOADM (リスト 1-14-4)

これらのプログラムはすべてポジションインディペンデントに作られていますから、適当なアドレスにロードして使用してください。使用方法是次のとおりです。

① POKE

EXEC ロードアドレス 書き込みアドレス, 書き込みデータ

② PEEK

DEFUSR = ロードアドレス

変数 = USR (読み出しアドレス)

③ SAVEM

EXEC ロードアドレス ファイル名, セーブ開始アドレス, 終了アドレス, 実行開始アドレス

④ LOADM

EXEC ロードアドレス ファイル名 [, [オフセット] [, R]]

リスト 1-14-1 裏 RAM へ POKE

ADR	:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
7100	:	BD	99	F9	1A	50	F7	FD	0F	E7	84	F5	FD	0F	1C	AF	39	:	2C
7110	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7120	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7130	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7140	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7150	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7160	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7170	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7180	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
7190	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
71A0	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
71B0	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00

```

71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : BD 99 F9 1A 50 F7 FD 0F E7 84 F5 FD 0F 1C AF 39 : 2C

```

```
SAVEM "L1-14-1M",&H7100,&H710F,&H7100
```

リスト1-14-2 裏RAMをPEEK

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : BD 9A 05 1A 50 B7 FD 0F E6 84 B5 FD 0F 1C AF 7E : FD
7010 : 97 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : E3
7020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 54 E6 05 1A 50 B7 FD 0F E6 84 B5 FD 0F 1C AF 7E : E0

```

```
SAVEM "L1-14-2M",&H7000,&H7011,&H7000
```

リスト1-14-3 裏RAMをSAVE

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : BD CC 37 BD CD F0 BD CD F0 AC 62 25 56 BD CD F0 : B7
7010 : 9D D8 BD 9F 5C C6 11 D7 BF 7F 02 DC 7F 02 D4 86 : 02
7020 : 02 B7 02 D8 BD CE DF 4F BD D0 8E EC 62 A3 64 C3 : 82
7030 : 00 01 1F 02 BD CD FC EC 64 BD CD FC AE 64 1A 50 : FA
7040 : B7 FD 0F A6 80 B5 FD 0F BD D0 8E 31 3F 26 F1 1C : 68
7050 : AF 86 FF BD D0 8E 4F 5F BD CD FC 35 36 BD CD FC : 74
7060 : 7E CE 48 7E 96 63 00 00 00 00 00 00 00 00 00 00 : 0E
7070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 40 AD 6E 1A 89 F7 F5 4D 4A 55 49 4F 5A A9 DD A1 : EF

```

```
SAVEM "L1-14-3M",&H7000,&H7065,&H7000
```

リスト 1-14-4 表 RAM ~ LOADM

ADR	:	+0	:	+1	:	+2	:	+3	:	+4	:	+5	:	+6	:	+7	:	+8	:	+9	:	+A	:	+B	:	+C	:	+D	:	+E	:	+F	:	[cs]
7000	:	4F	:	5F	:	FD	:	02	:	F1	:	FD	:	02	:	EE	:	BD	:	CC	:	37	:	9D	:	D8	:	27	:	0F	:	8D	:	C4
7010	:	92	:	92	:	81	:	2C	:	27	:	06	:	BD	:	9A	:	02	:	BF	:	02	:	F1	:	9D	:	D8	:	27	:	0F	:	84
7020	:	8D	:	92	:	92	:	C6	:	52	:	BD	:	92	:	94	:	26	:	62	:	86	:	03	:	87	:	02	:	EE	:	86	:	4A
7030	:	02	:	E6	:	2B	:	07	:	81	:	02	:	27	:	03	:	7E	:	CC	:	FC	:	C6	:	11	:	D7	:	BF	:	8D	:	37
7040	:	CE	:	DC	:	BE	:	02	:	D8	:	8C	:	02	:	00	:	26	:	45	:	BD	:	D0	:	27	:	8D	:	D0	:	68	:	EA
7050	:	1F	:	02	:	BD	:	D0	:	68	:	F3	:	02	:	F1	:	1F	:	01	:	1A	:	50	:	BD	:	D0	:	27	:	87	:	F4
7060	:	FD	:	0F	:	A7	:	80	:	86	:	FD	:	0F	:	31	:	3F	:	26	:	F1	:	1C	:	AF	:	BD	:	D0	:	27	:	3F
7070	:	8D	:	D0	:	68	:	BD	:	D0	:	68	:	F3	:	02	:	F1	:	FD	:	02	:	17	:	BD	:	CE	:	48	:	7D	:	3F
7080	:	02	:	EE	:	26	:	01	:	39	:	87	:	FD	:	0F	:	6E	:	9F	:	02	:	17	:	7E	:	92	:	A0	:	7E	:	67
7090	:	CF	:	A8	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	7A
70A0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00
70B0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00
70C0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00
70D0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00
70E0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00
70F0	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00	:	00

[cs]																	:	F2																

SAVEM "LI-14-4M", &H7000, &H7091, &H7000

F-BASICの内部構造

第 2 章

FM77AV には F-BASIC V3.0 と V3.3 の 2 つの BASIC がサポートされています。この章では、メモリ構成が単純で理解しやすい V3.0 を中心に、その内部構造を説明していきたいとします。

2-1 メモリマップ

F-BASIC V3.0(ディスクモード)のメモリマップは、図 2-1 のようになっています。ユーザー領域は、どこからどこまでが何とはっきり決められているわけではなく、場合によって変化します。そのために各々の現在の領域を示すポインタが、ワークエリアの中にあって、そのポインタの値を参照することにより各領域の範囲を知ることができるようになっています。それぞれのポインタのアドレスも図 2-1 に示します。

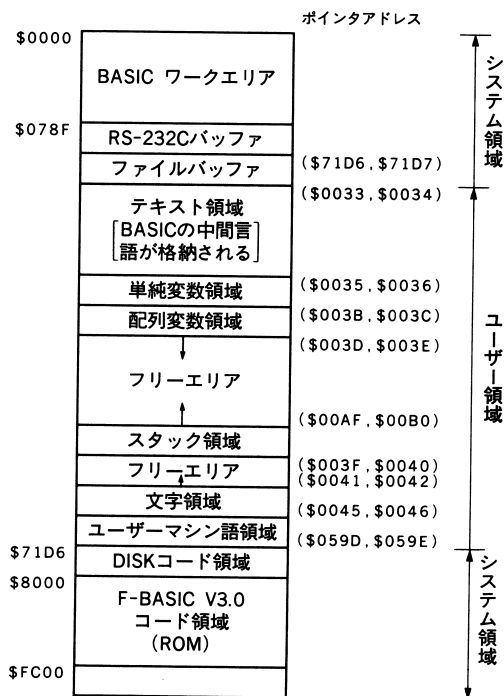


図 2-1 F-BASIC V3.0メモリマップ

これらのポインタのうち、ファイルバッファの先頭、テキスト領域の先頭、ユーザー領域の最後はリセット時に決められます。また文字領域の最後、スタックエリアの最後は、CLEAR 文によって変更できます。単純変数領域の先頭、配列変数領域の先頭、変数用フリーエリアの先頭、文字用フリーエリアの最後は、プログラムの実行時にダイナミックに変化します。

たとえば、CLEAR 1000,&H54FF [] とすると文字領域の最後は\$54FE、スタック領域の最後は\$54FE-1001=\$5115 となります。実際にモニタで見てみましょう(リスト 2-1)。

リスト 2-1 メモリマップ

```
CLEAR 1000,&H54FF
```

```
Ready
MON
```

```
*D0045
```

```
0045 [54 FE] FF FF 00 00 00 00
004D 00 00 03 49 00 00 0B F8
0055 00 00 00 05 47 00 00 00
005D 00 00 00 4D 00 00 00 00
0065 00 00 00 00 00 00 00 00
006D 00 00 00 00 00 00 00 87
0075 00 00 45 00 00 00 00 00
007D 00 07 8D 59 00 00 64 00
*D00AF
```

```
00AF [51 15] 00 00 00 00 00 03
00B7 49 00 00 00 00 00 00 00
00BF 00 00 00 00 28 00 00 00
00C7 00 00 00 00 00 00 0E 0E
00CF 00 28 00 0C DA 26 02 0C
00D7 09 B6 04 42 7E C7 60 7E
00DF F1 7D 00 03 E8 00 00 00
00E7 00 00 00 00 00 00 00 00
```

2-2 メモリマップの変化

(1) テキスト領域

テキスト領域はファイルバッファの直後におかれ、BASIC プログラムの大きさによって変化します。

(2) 変数領域

単純変数領域は、テキスト領域の直後におかれます。それで単純変数領域の先頭を示すポインタ(\$0035,\$0036)は、BASIC テキスト領域の最後+1 のアドレスを示しています。単純変数領域は、新たな単純変数やユーザー関数(DEF FN にて宣言される)が使用されると大きくなります。

配列変数領域は、単純変数領域の直後におかれ、DIM 文にて新たな配列変数が宣言されると大きくなります。そして ERASE 文にて配列変数が消去されると小さくなります。配列変数が単純

変数の直後におかれるため、大きな配列変数を宣言しているプログラムを実行すると思いがけない大きなロスが生じる可能性があります。それは新しい単純変数を使用するごとに、配列変数領域の移しかえが起きるからです。これは、文字領域のガベージコレクションより大きなロスとなることがあります。それを防ぐには、大きな配列変数を宣言する前にすべての単純変数を使用しておくようにすればいいでしょう。

プログラム実行中の変数領域の変化を、図 2-2 に示します。

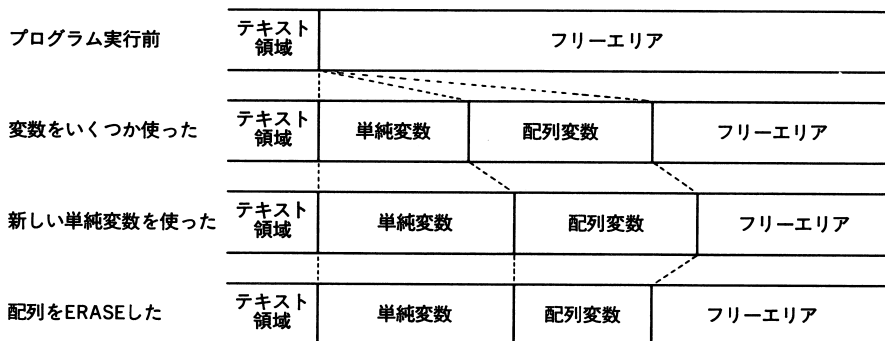


図2-2 変数領域の変化

(3) 文字領域

プログラム実行中の文字領域の変化を、図 2-3 に示します。文字列は文字領域の後のアドレスから順に格納されていきます。このとき同じ文字変数に新しい文字列を代入すると、古い文字列はそのままで新たな文字領域を増やしてそこに新しい文字列を書き込みます。そのため文字領域内には、もう使われていないゴミとなった文字列がたまっていきます。

そして文字領域が増えてフリーエリアがなくなったとき、これらのゴミを処分し新しい文字列のためのエリアをつくります。ゴミの部分を消して、使用中の文字列だけを後から順番につめていくわけです。これをガベージコレクションと呼びます。多くの文字列を使ったプログラムを実行していると、しばらく実行が止まってしまうことがありますが、それはガベージコレクション

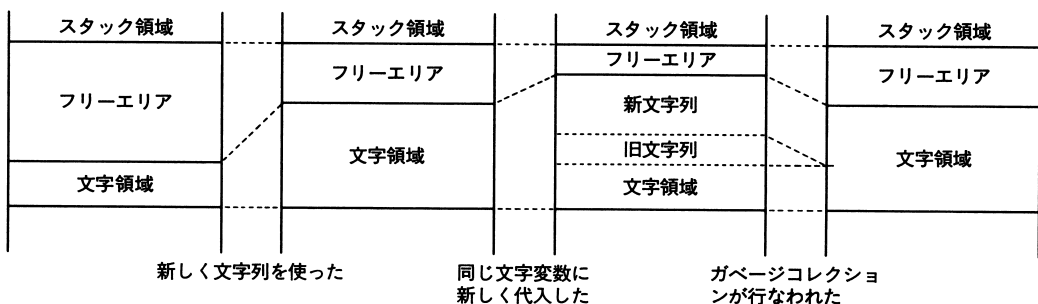


図2-3 文字領域の変化

が行なわれているためです。

なお、FRE 関数を実行してもガベージコレクションが行なわれます。それで文字領域のフリーエリアの大きさを見て、適時 FRE 関数を実行すれば、起こっては困るときにガベージコレクションが起こるのを防ぐことができます。文字領域のフリーエリアの大きさは、(\$00AF,\$00B0)と(\$0041,\$0042)の2つのポインタの値の差を見ればわかります。

2-3 プログラムの格納状態

BASIC プログラムは、テキスト領域の先頭を示すポインタ(\$0033,\$0034)に記されたアドレスを先頭として、行番号順に格納されています。それぞれの行は、図 2-4 のような形式になっています。

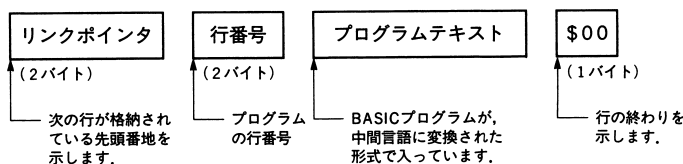


図2-4 行の格納形式

プログラムの最後の行番号のデータでは、リンクポインタの値が\$0000 となっていて後に続く行がないことを示します。

それでは BASIC のテキストが、実際にどのように格納されているかを見てみることにしましょう。例としてリスト 2-2 のプログラムを使います。

リスト 2-2 サンプル

```
100 FOR I=0 TO 20 STEP 5
110 PRINT I,SQR(I)
120 NEXT
```

テキスト領域の先頭アドレスは、ポインタ(\$0033,\$0034)の値を見ればわかります。リスト 2-3 に実際に格納されている BASIC プログラムの様子を示します。何が何だかわからない数字が並んでいますが、その意味は図 2-5 のとおりです。図 2-4 とあわせて理解してください。

リスト 2-3 プログラムの格納状態

MON

*D0033

```

0033 0B F9 0C 28 00 00 00 00
003B 0C 28 0C 28 70 A8 71 D4
0043 71 D3 71 D4 FF FF 00 00
004B 00 00 00 00 00 00 00 00
0053 0B F8 00 00 00 05 45 00
005B 00 00 00 00 00 40 00 00
0063 03 FD 00 00 00 00 00 F9
006B 00 00 03 3F 00 00 00 00

```

*D0BF9

```

0BF9 0C 11 00 64 81 20 49 E6
0C01 FE 01 00 20 CD 20 FE 01
0C09 14 20 D8 20 FE 01 05 00
0C11 0C 20 00 6E 20 B9 20 49
0C19 2C FF 85 28 49 29 00 0C
0C21 26 00 78 82 00 00 00 00
0C29 00 00 00 00 00 00 00 00
0C31 00 00 00 00 00 00 00 00

```

*

0BF9 0C	リンクポインタ	0C11 0C	リンクポインタ	0C20 0C	リンクポインタ
0BFA 11		0C12 20		0C21 26	
0BFB 00	行番号(=100)	0C13 00	行番号(=110)	0C22 00	行番号(=120)
0BFC 64		0C14 6E		0C23 78	
0BFD 81	FORの中間コード	0C15 20	スペース	0C24 82	NEXTの中間コード
0BFE 20	スペース	0C16 B9	PRINTの中間コード	0C25 00	行の終わり
0BFF 49	I	0C17 20	スペース		
0C00 E6	=の中間コード	0C18 49	I		
0C01 FE	整数型定数を示す	0C19 2C	,		
0C02 01	中間コード	0C1A FF	SQRの中間コード		
0C03 00	0	0C1B 85		0C26 00	プログラムの終わり
0C04 20	スペース	0C1C 28	(	0C27 00	
0C05 CD	T0の中間コード	0C1D 49	I		
0C06 20	スペース	0C1E 29	)		
0C07 FE	整数型定数を示す	0C1F 00	行の終わり		
0C08 01	中間コード				
0C09 14	20				
0C0A 20	スペース				
0C0B D8	STEPの中間コード				
0C0C 20	スペース				
0C0D FE	整数型定数を示す				
0C0E 01	中間コード				
0C0F 05	5				
0C10 00	行の終わり				

図2-5 サンプルプログラムの格納形式

2-4 中間言語

前節では、BASICプログラムのテキストが短縮された形式で格納されていることをみました。これは、BASICのキーワード(FOR, PRINT など)が1バイトか2バイトの中間言語で表されているからです。それでは、F-BASICでどのような中間言語が使われているかを見ていくことにします。

中間コード	キーワード	中間コード	キーワード	中間コード	キーワード	中間コード	キーワード
\$80	END	\$A0	WIDTH	\$C0	NEW	\$E0	XOR
\$81	FOR	\$A1	UNLIST	\$C1	GET	\$E1	EQV
\$82	NEXT	\$A2	MON	\$C2	PUT	\$E2	IMP
\$83	DATA	\$A3	LOCATE	\$C3	CIRCLE	\$E3	MOD
\$84	DIM	\$A4	CLS	\$C4	CONNECT	\$E4	¥
\$85	READ	\$A5	CONSOLE	\$C5	SYMBOL	\$E5	>
\$86	LET	\$A6	PSET	\$C6	GCURSOR	\$E6	=
\$87	GO	\$A7	PRESET	\$C7	BUBINI	\$E7	<
\$88	RUN	\$A8	MOTOR	\$C8	BUBW	\$E8	DSKINI
\$89	IF	\$A9	SKIPF	\$C9	BUBR	\$E9	DSKO\$
\$8A	RESTORE	\$AA	SAVE	\$CA	KILL	\$EA	NAME
\$8B	RETURN	\$AB	LOAD	\$CB	INTERVAL	\$EB	FIELD
\$8C	REM	\$AC	MERGE	\$CC	TAB(	\$EC	LSET
\$8D	'(注釈文)	\$AD	EXEC	\$CD	TO	\$ED	RSET
\$8E	STOP	\$AE	OPEN	\$CE	SUB	\$EE	CHAIN
\$8F	ELSE	\$AF	CLOSE	\$CF	FN	\$EF	ERASE
\$90	TRON	\$B0	FILES	\$D0	SPC(	\$F0	LLIST
\$91	TROFF	\$B1	COM	\$D1	USING	\$F1	LPRINT
\$92	SWAP	\$B2	KEY	\$D2	USR	\$F2	SOUND
\$93	DEFSTR	\$B3	PAINT	\$D3	ERL	\$F3	PLAY
\$94	DEFINT	\$B4	BEEP	\$D4	ERR		
\$95	DEFSNG	\$B5	COLOR	\$D5	OFF		
\$96	DEFDBL	\$B6	LINE	\$D6	THEN		
\$97	ON	\$B7	DEF	\$D7	NOT		
\$98	HARDC	\$B8	POKE	\$D8	STEP		
\$99	RENUM	\$B9	PRINT	\$D9	+		
\$9A	EDIT	\$BA	CONT	\$DA	-		
\$9B	ERROR	\$BB	LIST	\$DB	*		
\$9C	RESUME	\$BC	CLEAR	\$DC	/		
\$9D	AUTO	\$BD	RANDOMIZE	\$DD	^		
\$9E	DELETE	\$BE	WHILE	\$DE	AND		
\$9F	TERM	\$BF	WEND	\$DF	OR		

表2-6A 中間言語一覧表(1バイトコード)

中間コード	キーワード	中間コード	キーワード	中間コード	キーワード
\$264F	&O(8進定数)	\$FF8F	VAL	\$FFA5	INKEY\$
\$2648	&H(16進定数)	\$FF90	ASC	\$FFA6	INPUT
\$FE01	1バイト整定数	\$FF91	CHR\$	\$FFA7	CSRLIN
\$FE02	2バイト整定数	\$FF92	CINT	\$FFA8	POINT
\$FE04	単精度型定数	\$FF93	CSNG	\$FFA9	TIME
\$FE08	倍精度型定数	\$FF94	CDBL	\$FFAA	DATE
\$FEF2	行番号定数	\$FF95	FIX	\$FFAB	DSKF
\$FF80	SGN	\$FF96	SPACE\$	\$FFAC	CVI
\$FF81	INT	\$FF97	HEX\$	\$FFAD	CVS
\$FF82	ABS	\$FF98	OCT\$	\$FFAE	CVD
\$FF83	FRE	\$FF99	LOF	\$FFAF	MKI\$
\$FF84	POS	\$FF9A	EOF	\$FFB0	MKS\$
\$FF85	SQR	\$FF9B	PEN	\$FFB1	MKD\$
\$FF86	LOG	\$FF9C	LEFT\$	\$FFB2	LOC
\$FF87	EXP	\$FF9D	RIGHT\$	\$FFB3	DSKI\$
\$FF88	COS	\$FF9E	MID\$	\$FFB4	LPOS
\$FF89	SIN	\$FF9F	INSTR		
\$FF8A	TAN	\$FFA0	SCREEN		
\$FF8B	ATN	\$FFA1	ANPORT		
\$FF8C	PEEK	\$FFA2	VARPTR		
\$FF8D	LEN	\$FFA3	STRING\$		
\$FF8E	STR\$	\$FFA4	RND		

表2-6B 中間言語一覧表(2バイトコード)

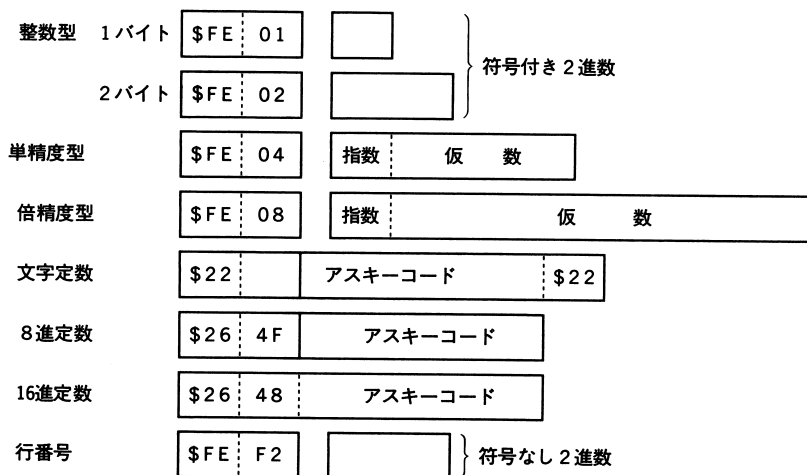


図2-7 定数の中間言語

数値定数の中間言語は、数値の前に「\$FEXX」をつけた形となります。「XX」は数値型を示す識別子で、数値データのメモリの大きさ(バイト数)を示しています。文字定数と8進定数、16進定数は、アスキーコードでそのままの形で格納されます。また BASIC テキストの行番号(GOTO 文などで使用される)は、行番号の前に「\$FEF2」をつけた符号なし2バイト整数(0~65535)で示されます。定数の中間言語の形式を図2-7に示します。そして F-BASIC の中間言語の一覧表を、図2-6A、図2-6Bに示します。

2-5 変数の格納状態

2-5-1 単純変数

単純変数領域は、テキスト領域の後に作られます。この領域は、単純変数領域先頭ポインタ(\$0035,\$0036)で示されるアドレスから配列変数領域先頭ポインタ(\$003B,\$003C)で示されるアドレスのひとつ前までです。

プログラム中で変数が使われると、その変数の型に応じた形式で使われた順番に登録されていきます。変数の値の参照はこの領域の先頭から順に行なわれていきますので、頻繁に使う変数を早めに定義しておくことと実行速度をあげることができます。各変数は、その型によって図2-8のような形式で登録されます。

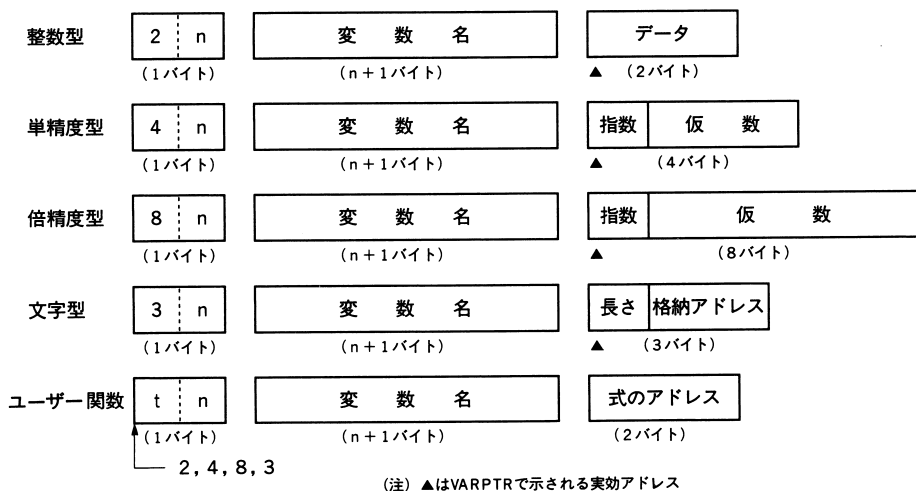


図2-8 単純変数の格納形式

1バイト目の上位4ビットは各データの型を示すと同時に数値データのバイト数をも示しています。そして下位4ビットには、変数名(関数名)の長さ-1がセットされます。ですからこのふたつの値に演算を施すことにより、次の単純変数のアドレスを求めることができるようになります。

変数名の長さが4ビットで表わされているため、変数名として格納されるのは最大16文字となります。BASICの変数名の識別される最大長が16であるという制約は、このためなのです。

またユーザー関数の関数名では、最初の文字の最上位ビットがオンとなっていて他の単純変数の変数名と区別されています。そして式のアドレスには、ユーザー関数が定義されている BASIC テキストのアドレスが格納されます。

それでは実際の例で確かめてみましょう。次のプログラムを実行した後、各ポインタおよび単純変数領域の内容を見てみます(リスト2-4)。ちょっとわかりにくいですが、それでは各変数ごとに説明します(図2-9)。

リスト2-4 単純変数領域

```

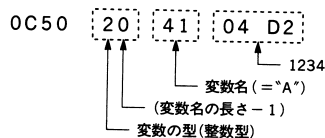
100 A%=1234
110 BCD%=-1234
120 EFG!=1.2345
130 HIJH=1.234567890123456H
140 KLM$="1234567890"
RUN

Ready
MON

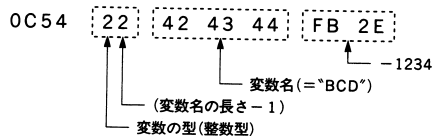
*D0035      .....単純変数領域
              開始アドレス
0035 0C 50 00 00 00 00 0C 75 .....配列変数領域
003D 0C 75 70 A8 71 D4 71 D3 .....開始アドレス
0045 71 04 FF FF 00 8C 00 00
004D 0C 4D 03 3D 00 00 0B F8
0055 00 00 03 0C 72 0C 72 00
005D 00 00 0C 4D 00 00 0C 75
0065 00 00 00 00 0C 6E 00 00
006D 05 78 00 00 00 00 86
*D0C50

0C50 20 41 04 D2 22 42 43 44
0C58 FB 2E 42 45 46 47 81 1E
0C60 04 18 82 48 49 4A 81 1E
0C68 06 52 14 62 CF C1 32 4B
0C70 4C 4D 0A 0C 42 20 00 00
0C78 00 00 00 00 00 00 00 00
0C80 00 00 00 00 00 00 00 00
0C88 00 00 00 00 00 00 00 00
*
```

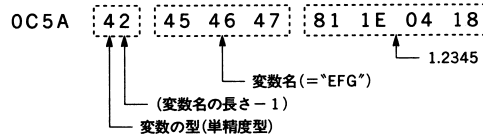
・A%=1234



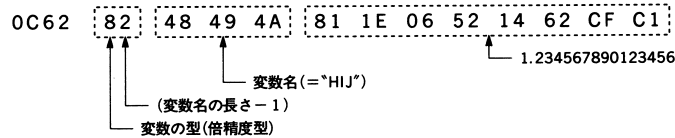
・BCD%=-1234



• EFG!=1.2345



• HIJ#=1.234567890123456#



• KLM\$="1234567890"

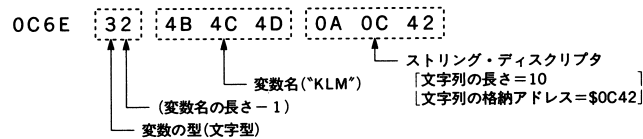


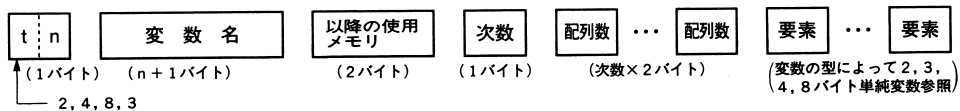
図2-9 変数ごとの例

2-5-2 配列変数

配列変数領域は、単純変数領域の後に作られます。この領域は、配列変数領域先頭ポインタ(\$003B,\$003C)で示されるアドレスからフリーエリア先頭ポインタ(\$003D,\$003E)で示されるアドレスの1つ前までです。

ここも単純変数領域と同じように、DIM 文で配列変数を宣言するとその順番で登録されます。ERASE 文を実行すると、その配列変数が消されてうしろにある領域が前に移動してきます。

配列変数の格納形式は、各配列について図 2-10 のようになります。それでは整数型配列変数と文字型配列変数について、実際に見てみることにします(リスト 2-5, リスト 2-6)。



要素の順番はDIM A(2,3,4)の場合、次のようになる。

A(0,0,0)
 A(1,0,0)
 A(2,0,0)
 A(0,1,0)
 A(1,1,0)
 ⋮
 A(1,3,4)
 A(2,3,4)

(注) 配列の宣言と逆順になる

図2-10 配列変数の格納形式

リスト 2-5 配列変数領域(1)

```
100 DIM A%(3,2)
110 FOR I=0 TO 3
120   FOR J=0 TO 2
130     A%(I,J)=I+J
140   NEXT J
150 NEXT I
RUN
```

Ready
MON

*D003B	配列変数領域 開始アドレス															
003B	0C	65	0C	86	70	A8	71	D4								
0043	71	D3	71	D4	FF	FF	00	96								
0048	00	00	0C	56	03	3D	00	00								
0053	0B	F8	00	00	01	0C	5B	0C								
0058	5B	79	04	00	0C	4D	00	00								
0063	0C	65	00	00	00	00	0C	5F								
0068	00	00	05	78	00	00	00	00								
0073	00	86	00	00	3B	14	62	CF								
*D0C65																
0C65	20	41	00	1F	02	00	03	00								
0C6D	04	00	00	00	01	00	02	00								
0C75	03	00	01	00	02	00	03	00								
0C7D	04	00	02	00	03	00	04	00								
0C85	05	00	00	00	00	00	00	00								
0C8D	00	00	00	00	00	00	00	00								
0C95	00	00	00	00	00	00	00	00								
0C9D	00	00	00	00	00	00	00	00								
*																
要素 A % (1, 0)																
要素 A % (0, 0)				要素 A % (3, 2)												

変数の型(整数型)
変数名の長さ-1
変数名("A")
以降の使用メモリ
次数
配列数
配列数

リスト 2-6 配列変数領域(2)

```
100 DIM ABC$(3,2)
110 FOR I=0 TO 3
120   FOR J=0 TO 2
130     ABC$(I,J)="ABC"
140   NEXT J
150 NEXT I
RUN
```

Ready
MON

*D003B	配列変数領域 開始アドレス															
003B	0C	68	0C	9A	70	A8	71	D4								
0043	71	D3	71	D4	FF	FF	00	96								
0048	00	00	0C	5C	03	3D	00	00								
0053	0B	F8	00	00	01	0C	61	0C								
0058	61	79	04	00	0C	4D	00	00								
0063	0C	68	00	00	00	00	0C	65								
0068	00	00	05	78	00	00	00	00								
0073	00	86	00	00	3B	14	62	CF								
*D0C68																
0C68	32	41	42	43	00	2B	02	00								
0C73	03	00	04	03	0C	46	03	0C								
0C78	46	03	0C	46	03	0C	46	03								
0C83	0C	46	03	0C	46	03	0C	46								
0C88	03	0C	46	03	0C	46	03	0C								
0C93	46	03	0C	46	03	0C	46	00								
0C98	00	00	00	00	00	00	00	00								
0CA3	00	00	00	00	00	00	00	00								
*																
配列数																

変数の型(文字型)
変数名の長さ-1
変数名("ABC")
以降の使用メモリ
次数
配列数
要素ABC\$(0,0)のSTRING
ディスクリプタ
要素ABC\$(3,2)のSTRING
ディスクリプタ

2-6 文字列エリア

文字列エリアには、文字変数の実際の文字列が格納されています。この領域は、文字領域先頭ポインタ(\$0045,\$0046)で示されるアドレスからフリーエリア先頭ポインタ(\$0041,\$0042)で示されるアドレスの1つ後までで、逆向きに登録されていきます。

文字列エリアの格納形式は、図 2-11 のようになります。文字列エリアには、文字列のほかにその文字列を指している変数テーブルのアドレス(文字変数の VARPTR で示される実行アドレス)も格納されています。そして使用済みのゴミとなった文字列には、このアドレスにゴミの文字列の長さが入り、ガベージコレクションを容易にできるように工夫されています。また BASIC テキスト中で A\$= "AB" の様に文字定数が使われた場合には、"AB" は文字列エリアに格納されません。そして単純変数テーブル中の格納アドレスには、BASIC テキスト中の "AB" アドレスが直接格納されます。

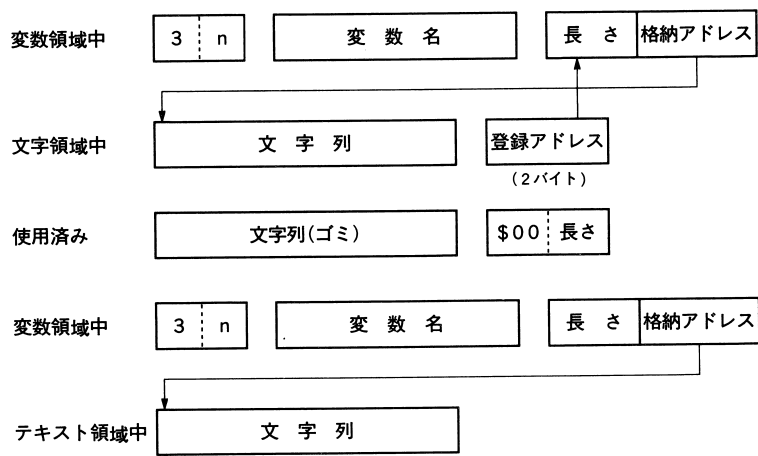


図2-11 文字列エリアの格納形式

それでは例として次のプログラムを実行します。

```
100 A$= "ABC"  
110 B$=A$+ "D"
```

このプログラム実行後の文字列エリアは、図 2-12A のようになります。A\$の文字列は BASIC テキスト中の文字列を直接使うため、文字列エリアにはとられません。

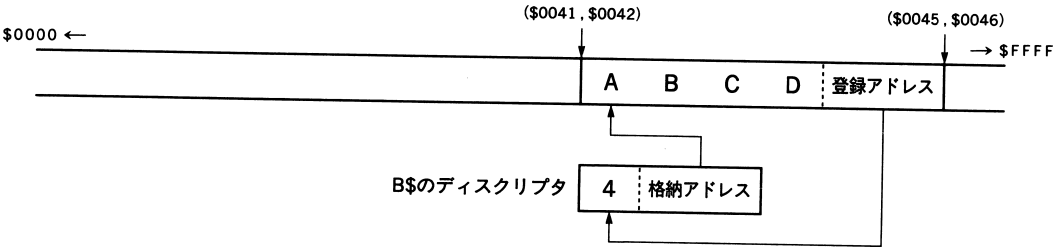


図2-12A 文字列エリアの様子(1)

ここでダイレクトモードで B\$ = "12345" を実行すると、図 2-12B のようになります。最初に B\$ 代入された "ABCD" は、メモリ上から消されずにそのまま残っているのです。

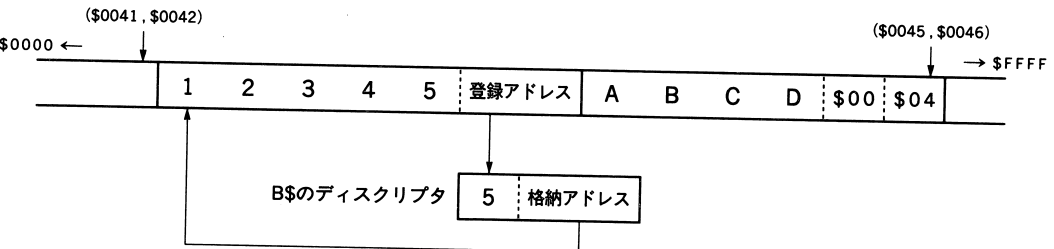


図2-12B 文字列エリアの様子(2)

ここでページコレクションを行なってみましょう。PRINT FRE("A") を実行すると、図 2-12C のようになります。現在使用中の "12345" が文字列エリアの後ろにつめられ、ポインタ (\$0041, \$0042) が示しているアドレスも移動しています。

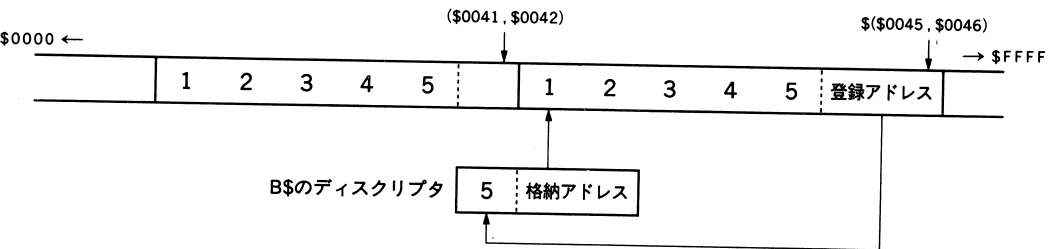


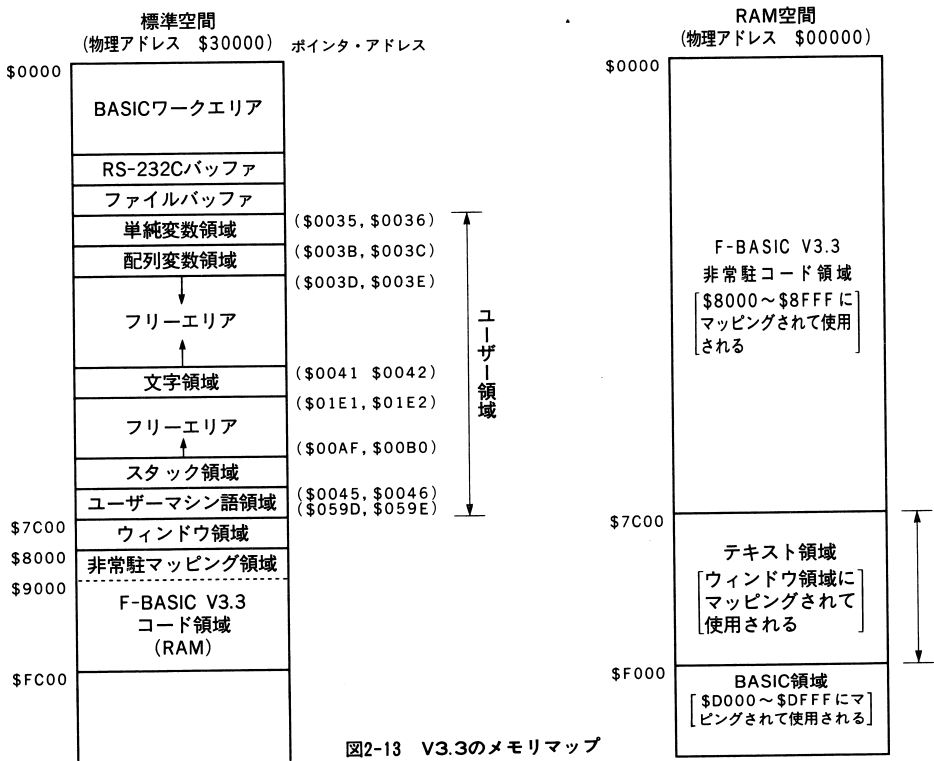
図2-12C 文字列エリアの様子(3)

2-7 F-BASIC V3.3の変更点

前節まで F-BASIC V3.0 の内部構造について説明してきました。そこでここでは、F-BASIC V3.3 と V3.0 の内部構造の相違点について考えてみたいと思います。

(1) メモリマップ

F-BASIC V3.3 のメモリマップを、図 2-13 に示します。



V3.0 のメモリマップと比較したとき、大きな相違点が3つあります。

- ① F-BASIC コード領域が、常駐領域と非常駐領域とに分離した。
- ② テキスト領域が RAM 空間に移され、テキストウィンドウ機能によって利用されるようになった。
- ③ スタック領域と文字領域の位置関係が逆転した。

V3.3 の非常駐コード領域は 4KB ごとにモジュール化されていて、MMR 機能によって \$8000 ~ \$8FFF または、\$D000 ~ \$DFFF にメモリマッピングされて使用されます。MMR 機能については、「13-6 メモリ管理」を参照ください。

非常駐コード領域の各モジュールの機能は、図 2-14 のようになっています。

領域	アドレス (物理アドレス)	機 能
非常駐1	\$38000 } \$38FFF	システムイニシャライズ処理, カセットファイル処理
非常駐9	\$00000 } \$00FFF	音楽演奏処理
非常駐8	\$01000 } \$01FFF	テキスト処理, 音声合声処理, デジタイズ処理, クロック処理
非常駐7	\$02000 } \$02FFF	プリンタ処理, データ入力処理
非常駐6	\$03000 } \$03FFF	フロッピーディスク処理, ジョイスティック処理
非常駐5	\$04000 } \$04FFF	数値関数処理, モニタコマンド処理
非常駐4	\$05000 } \$05FFF	ハードコピー処理
非常駐3	\$06000 } \$06FFF	グラフィック処理
非常駐2	\$07000 } \$07BFF	VRAM制御処理
非常駐1	\$0F600 } \$0FFFF	FM音源処理

図2-14 非常駐モジュールの機能

V3.3のテキスト領域は、FM77AVで新しく設けられたRAM空間に移されました。BASICインタプリタがこのテキスト領域をアクセスするには、テキストウィンドウ機能によってウィンドウ領域(\$7C00~\$7FFF)にメモリマッピングして使用します。テキストウィンドウ機能については、「13-6 メモリ管理」を参照ください。

V3.3ではスタック領域と文字領域の位置関係が逆転してしまいました。このためV3.0ではCLEAR文で文字領域の大きさを指定できたのに対して、V3.3ではスタック領域の大きさを指定するようになりました。

(2) プログラムの格納状態

テキストのリンクポインタの値が、次の行が格納されている先頭アドレスではなくて、次の行が格納されている相対アドレス(テキスト領域の先頭を0とする)に変更されています。これはテキストウィンドウ機能の採用により、絶対アドレスより相対アドレスの方が処理しやすくなったためと思われます。

またテキストの格納状態を示す各種ポイントの値も、相対アドレスが使用されるようになって
います(図2-15)。

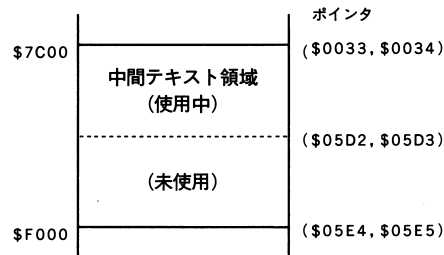


図2-15 テキスト領域のポイント

中間コード	キーワード	中間コード	キーワード
\$F4	WRITE	\$FFB5	——
\$F5	CLOCK	\$FFB6	JIS
\$F6	ROLL	\$FFB7	KNJ\$
\$F7	KANJI	\$FFB8	AKCNV\$
\$F8	——	\$FFB9	KACNV\$
\$F9	——	\$FFBA	KTYPE
\$FA	OUT	\$FFBB	KLEFT\$
\$FB	CALL	\$FFBC	KRIGHT\$
		\$FFBD	KMD\$
		\$FFBE	KEXT\$
		\$FFBF	KLEN
		\$FFC0	KINSTR
		\$FFC1	SEARCH
		\$FFC2	TALK
		\$FFC3	——
		\$FFC4	——
		\$FFC5	——
		\$FFC6	——
		\$FFC7	——
		\$FFC8	STICK
		\$FFC9	STR!G
		\$FFCA	SIMPOSE
		\$FFCB	PALETTE
		\$FFCC	BGM
		\$FFCD	VOICE

図2-16 V3.3追加の中間コード一覧

(3) 中間言語

V3.3では多くの命令が追加されています。また V3.5 において拡張されていた命令も使用することはできませんが、V3.3 の中間言語には反映されています。V3.3 において拡張された中間言語の一覧表を、図 2-16 に示します。

(4) 変数

識別可能な変数名の長さが 40 文字に拡張されたため、変数名の長さのセットされる領域が 1 バイトとられるようになりました。図 2-17 に V3.3 における変数の格納形式を示します。

また V3.0 では文字変数に文字定数を代入する場合、文字変数のストリングディスクリプタにテキスト中の文字定数のアドレスが格納されました。しかし V3.3 ではテキストウィンドウ機能が採用されたため、必ず文字領域に文字列がとられるようになっています。

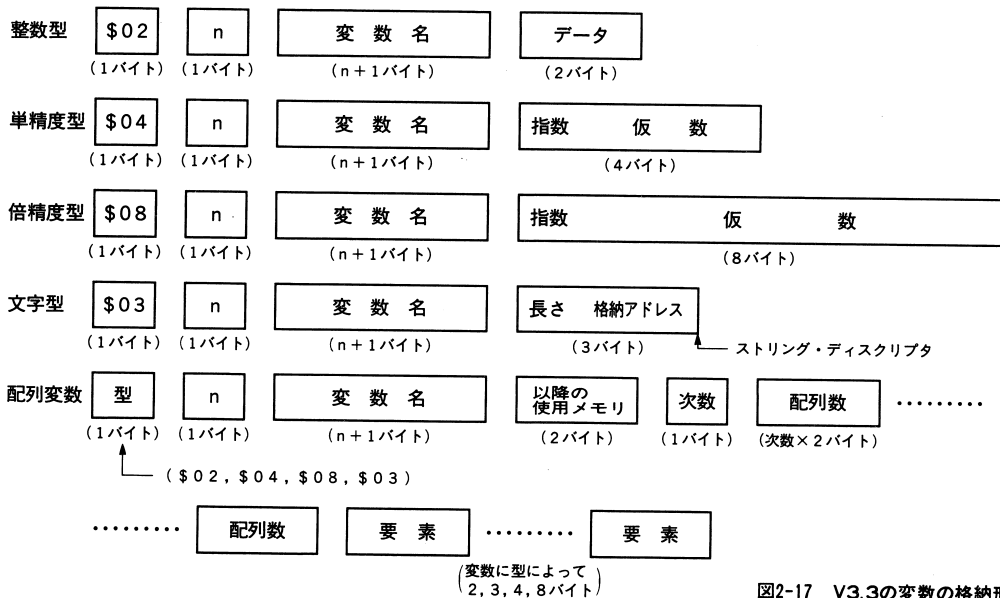


図2-17 V3.3の変数の格納形式

2-8 マシン語領域の確保

マシン語ルーチンは、通常 CLEAR 文にて確保されるユーザーマシン語領域に置きます。しかしこの領域は CLEAR 文の再実行によって変化してしまうため、ユーティリティ的なプログラムの領域には不向きです。そこでたとえば BASIC の未定義命令を活用したシステムのマシン語ルーチンのための領域について考えてみました。

- ① ファイルバッファとテキスト領域 (V3.3 では単純変数領域) との間 (図 2-18)
- ② ユーザーマシン語領域とシステム領域との間 (図 2-19)

テキスト領域の開始アドレスは、(\$0033,\$0034)のポインタに書き込まれています。通常この値は、ファイルバッファに続いた値となっています。この値を大きな値に書き換えてやれば、ファイルバッファとテキスト領域の間に BASIC インタープリタが使用しない領域ができるわけです。V3.3 の場合にも (\$0035,\$0036) の値を書き換えれば、単純変数領域とファイルバッファの間にマシン語領域を設定することができます。この領域の確保で注意することは、テキスト領域の開始1つ前のアドレスには、\$00 を書き込んで NEW  を入力しておく必要があることです。

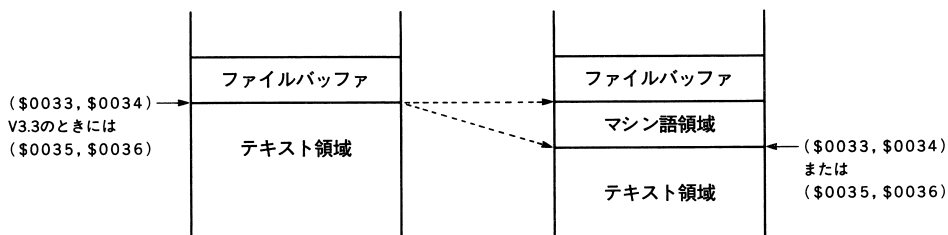


図2-18 マシン語領域の確保(1)

ユーザーマシン語領域とシステム領域の間にマシン語領域を確保するには、(\$059D,\$059E)のポインタを書き換え CLEAR 文を実行しておきます。そうしておくで、(\$059D,\$059E)が示すアドレスからシステム領域の間につくられた領域は、それ以後の CLEAR 文によっても影響されない領域となります。

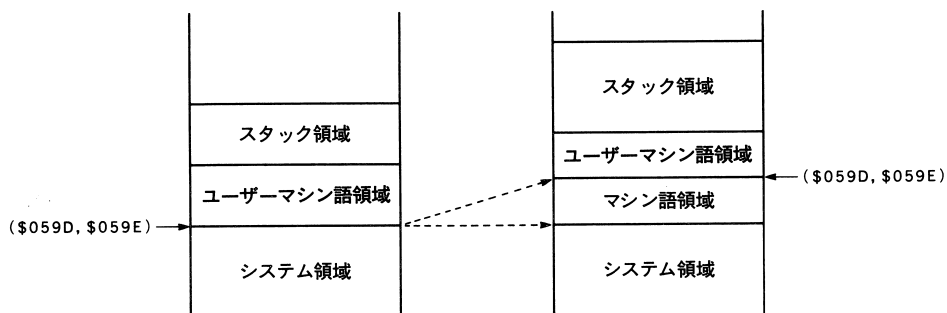


図2-19 マシン語領域の確保(2)

2-9 BASIC の未定義命令

F-BASIC V3.0 には、いくつかの未定義命令があります。これらの命令は、以前サポートされていて使用されなくなったバブルメモリおよびライトペンに関する命令です。これらの命令は RAM 上のフックを通じてエラールーチンへジャンプするようになっていますから、フックの飛

び先を変えれば別の命令として使用できるわけです。未定義命令とそのフックのアドレスは、次のとおりです。

BUBINI\$05D3~\$05D5
 BUBW\$05D6~\$05D8
 BUBR\$05D9~\$05DB
 PEN\$0251~\$0453

V3.3 の場合、フックは使われておらず直接エラールーチンにエントリしています。

ただし V3.3 の BASIC は RAM 上にありますから、書き換えることは可能です。それぞれの未定義命令のジャンプテーブルのアドレスは次のとおりです。

BUBINI\$9B97, \$9B98
 BUBW\$9B99, \$9B9A
 BUBR\$9B9B, \$9B9C
 PEN\$987B, \$987C

2-10 EXEC 文にパラメータ指定

BASIC インタープリタがコマンドを実行するときは、中間コードに従ってその処理ルーチンのエントリアドレスを求めて、そのアドレスをサブルーチンコールしています。各処理ルーチンでは中間コードに続くテキストをパラメータとして読み取っていき、それに応じた処理を行なった後、リターンするわけです。

EXEC 文は、この処理ルーチンのエントリアドレスをパラメータとして与えるものです。ですから他のコマンドやステートメントは、EXEC 文で代用することができるわけです。たとえば次のような文を、F-BASIC V3.0 で実行してみてください。

EXEC &HDDA8 (250,80), "FM",6,5,4 

画面中央に緑色で "FM" と表示されましたね。&HDDA8 というのは、SYMBOL 文のエントリアドレスです。ですから先ほどの命令は、次の命令と同じ働きをするわけです。

SYMBOL(250,80), "FM",6,5,4 

つまりこれでわかるとおり、EXEC 文でパラメータの受渡しをすることができるのです。しかしそのためには、BASIC のコマンド処理ルーチンと同様の処理を行なう必要があります。パラメータの読み取りは複雑な処理に思えますが、BASIC ROM 内ルーチンを使えば割と簡単にできます。この使用法は、テキストユーティリティの中で多用していますので関心のある方は、プログラムを解析してみてください。

2-11 裏 RAM から BASIC ROM ルーチンをコール

F-BASIC V3.0 には、裏 RAM と称される RAM 領域があります。これはバンク切り替えによって BASIC ROM (V3.0) と切り替えて使用されます。ですから裏 RAM においたマシン語プログラムから、BASIC の ROM 内ルーチンを利用するときには工夫が必要です。

普通は ROM 内ルーチンコール用のジャンプテーブルを作っておく方法がとられますが、ジャンプ先が多くなるとなかなか大変になります。そこでスタックを利用して ROM 内ルーチンの実行を行なうシステムコールルーチンを考えてみました。

使い方はまずリスト 2-7 のプログラムを、裏 RAM 以外の RAM 領域に置きます。そして裏 RAM のマシン語ルーチンから ROM 内ルーチンへジャンプする命令、

```
JSR $XXXX
```

または、

```
JMP $XXXX
```

を次のように書き換えます。

```
JSR SYSTEM
```

```
JSR $XXXX
```

または

```
JSR SYSTEM
```

```
JMP $XXXX
```

これだけで正常に ROM 内ルーチンが実行され、実行終了後、裏 RAM のマシン語ルーチンに制御が戻ってきます。

リスト 2-7 システムコール

01000				*****
01010				* SYSTEM CALL *
01020				* (LIST 2-7) V3.0 *
01030				*****
01032				OPT NOGEN
01034	7000			ORG \$7000
01040	7000	32	7C	SYSTEM LEAS -4.S
01050	7002	34	13	PSHS CC,A,X
01060	7004	AE	68	LDX 8.S
01070	7006	A6	80	LDA .X+
01080	7008	81	7E	CMPA #7E
01090	700A	27	19	BEQ JUMP
01100	700C	30	02	LEAX 2.X
01110	700E	AF	68	STX 8.S
01120	7010	AE	1E	LDX -2.X
01130	7012	AF	64	STX 4.S
01140	7014	30	8C 07	LEAX <RET,PCR
01150	7017	AF	66	STX 6.S

```

01160 7019 85 FD0F      BITA  $FD0F
01170 701C 35 93      PULS  CC,A,X,PC
01180 701E 34 01      RET   PSHS  CC
01190 7020 87 FD0F      STA   $FD0F
01200 7023 35 81      PULS  CC,PC
01210
01220 7025 AE 84      JUMP  LDX   ,X
01230 7027 AF 68      STX   8,S
01240 7029 85 FD0F      BITA  $FD0F
01250 702C 35 13      PULS  CC,A,X
01260 702E 32 64      LEAS  4,S
01270 7030 39      RTS
01280
01290      7000      END   SYSTEM
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=7000
PROGRAM END   ADDR=7030
PROGRAM ENTRY ADDR=7000

```

2-12 BASICプログラム復活法

NEW コマンドを実行すると、BASIC プログラムは消えてしまいます。しかし実際にメモリから抹消されてしまうわけではなく、プログラムの最初のリンクポインタとプログラムの終わりを示すポインタ(\$35,\$36)がクリアされるだけです。したがってこれらの値を戻してやれば、プログラムが復活します。

さいわい、F-BASIC V3.0 の内部にリンクポインタ再設定の ROM 内ルーチンがあります。これを利用することにします。このルーチンをコールすると、リンクポインタの値が修復され X レジスタにプログラム・エンドのアドレスが格納されてリターンしてきます。BASIC プログラム復活のプログラムをリスト 2-8 に示します。

リスト 2-8 プログラム復活(V3.0)

```

00100      *****
00110      *   BASIC フックツ   *
00120      *   ( LIST 2-8 )   V3.0   *
00130      *****
00140      OPT      NOGEN
00150 5000      ORG      $5000
00160      *
00170 5000 6C 9F 0033  ENTRY  INC   [$33]
00180 5004 8D C730      JSR   $C730
00190 5007 30 02      LEAX  2,X
00200 5009 9F 35      STX   $35
00210 500B 39      RTS
00220      5000      END   ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=500B
PROGRAM ENTRY ADDR=5000

```

V3.3の場合には、次の命令をダイレクトモードで実行すれば、BASICプログラムを復活させることができます。

POKE &HFD92,0: POKE &H7C03,1: EXEC &H9521

2-13 テキストユーティリティ

2-13-1 テキストサーチ&リプレイス

これは、BASICテキストの文字列をサーチし更にそれを別の文字列に置き換えて再格納するプログラムです(リスト 2-9)。動作する F-BASIC は V3.0 のみです。まず CLEAR, &H6F00 でマシン語領域を確保します。それから \$6F00 からプログラムをロードして、EXEC &H6F00 で実行してください。テキストサーチ&リプレイスのプログラムが、未定義命令の BUBR に割り当てられます。実行を終了するときには、もう一度 EXEC &H6F00 を入力してください。テキストサーチ&リプレイスの実行コマンド形式は、次のとおりです。

BUBR [,開始行番号][-終了行番号] 文字列1 [TO 文字列2][, [N]][L]]

N の指定があると行番号のみを表示します。また L の指定があると、サーチ結果をプリンタに出力します。文字列 2 が指定されるとサーチされた文字列 1 が文字列 2 にリプレイスされます。このリプレイスを使うとちょっと面白いことができます。

10 '@4 COLOR REM

BUBR "@ " TO CHR\$(17)

注釈文が緑色になりましたね。これは CHR\$(17) という制御コードを使ったものです。これでつくった行に対しては、スクリーンエディットできなくなります。

リスト 2-9 テキストサーチ&リプレイス

ADR	:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
6F00	:	8E	CF	0A	BC	05	0A	27	15	BF	05	0A	86	39	B7	02	9F	:	F3
6F10	:	A7	8C	ED	AE	8C	5D	34	10	30	8C	4C	20	29	30	8C	5F	:	67
6F20	:	BF	05	DA	86	7E	B7	02	9F	30	8C	4A	BF	02	A0	86	FF	:	E6
6F30	:	97	00	97	01	0F	02	0F	03	BE	05	9D	AF	8C	35	30	8C	:	DE
6F40	:	BE	34	10	30	8C	17	BD	9B	DB	35	10	BF	05	9D	30	1F	:	FD
6F50	:	9F	45	30	89	FE	D4	9F	3F	BD	8F	51	7E	8F	E1	42	55	:	6F
6F60	:	41	52	20	4F	4E	0D	0A	00	42	55	42	52	20	4F	46	46	:	8E
6F70	:	0C	0A	00	00	00	81	C9	26	05	CC	C5	08	ED	E4	39	4F	:	7E
6F80	:	5F	DD	37	8E	F9	FF	9D	DB	0F	27	81	2E	27	04	81	FE	:	FD
6F90	:	26	09	BD	9F	62	0C	27	9E	4B	9F	37	81	DA	26	14	8E	:	02
6FA0	:	F9	FF	9D	D2	81	2E	27	04	81	FE	26	07	BD	9F	62	0C	:	87
6FB0	:	27	9E	4B	9F	39	0D	27	27	03	BD	92	92	0F	20	0F	27	:	8C
6FC0	:	8D	9B	F1	D7	C1	27	52	9F	67	9D	DB	81	CD	26	21	03	:	6A
6FD0	:	20	9E	3F	D6	C1	DE	67	9F	67	A6	C0	A7	8D	9C	45	27	:	74
6FE0	:	40	5A	26	F5	9D	D2	BD	9B	F1	D7	C4	9F	8F	17	01	1E	:	69

```

6FF0 : 9D DB 27 1E BD 92 92 81 4E 26 04 03 27 9D D2 81 : AE
-----
[cs] : 96 20 21 57 E7 18 B5 BF A7 C8 45 BD 61 CC 74 1A : CD
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : 4C 26 0F 86 11 97 BF CC 03 00 FD 02 E6 BD CE DF : 8C
7010 : 9D D2 9D DB 9F 5C 20 10 7E 96 63 C6 0D 7E 8D : 21
7020 : 01 C6 0E 7E 8D 01 7E 02 CD DC 37 BD 8F 1E 9F 4F : 09
7030 : BD D6 78 0F 28 EC 84 27 07 EE 02 11 93 39 23 06 : 06
7040 : BD CE 48 7E 8F E1 DF 47 DF 4B BD C1 69 8E 04 3D : CA
7050 : 20 02 9E 23 33 01 DF 23 D6 C1 DE 67 A6 80 27 38 : 7A
7060 : A1 C0 26 EE 5A 26 F5 97 28 0D 20 27 31 D0 C1 30 : EF
7070 : 85 9F 25 50 A6 85 A7 80 26 FA D6 C4 33 85 11 83 : F1
7080 : 05 3C 22 A2 A6 82 A7 85 9C 25 26 F8 DE 8F 5D 27 : 29
7090 : C3 A6 C0 A7 80 5A 20 F7 0D 28 27 19 8D 1E DC 4B : 08
70A0 : DD A6 BD B6 15 BD 9C 22 0D 27 26 09 8E 04 3D BD : 75
70B0 : D9 0F BD 9B 50 9E 4F AE 84 16 FF 72 DC 4B DD E4 : 1E
70C0 : FD 03 3A 7F 03 3C 8E 04 3D 9F D9 30 1F BD C2 88 : 95
70D0 : DD 13 DC E4 DD 4B BD 8F 1C 9F 49 25 09 9F A2 AE : 45
70E0 : 84 9F A4 BD 9F D3 B6 03 3C 27 1E 9E 4B 9F A6 DC : 3A
70F0 : 35 DD 61 03 13 DD 5F BD 8D 93 CE 03 38 A6 C0 A7 : 88
-----
[cs] : 8B EC DD 57 62 EE 89 05 46 DD DD C8 C1 21 28 B5 : 10
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : 80 9C 63 26 F8 9E 5F 9F 35 9E 49 BD C7 32 9E 45 : EE
7110 : 9F 41 9E 35 9F 38 9F 3D 0F 4D 0F 4E BD 8F 95 39 : 3C
7120 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7130 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 1F DD 01 5B 97 D9 FE DC 44 EB 5B 0B 84 C1 33 7E : 2A

```

SAVEM "L2-9M",&H6F00,&H711F,&H6F00

2-13-2 拡張 AUTO&EDIT

この拡張 AUTO は、BASIC の AUTO コマンドに対して次の機能拡張を施したものです。

- ① フルスクリーンエディットで、入力中の行以外の修正も可能。
- ② 文字列オプションにより、DATA 文、PLAY 文等の入力がしやすくなった。
- ③ 既に存在している行に対しては、その行全体が表示され修正も可能。

このコマンド形式は次のとおりです。

PEN AUTO [行番号][, [増分値][, 文字列]]

拡張 EDIT は、ロールアップダウン可能なフルスクリーンエディタです。これにはテキストを上下にスクロール表示していくスクロールモードと、表示されている画面を修正するエディットモードがあります。2つのモードは、リターンキーで相互に変換されます。有効なコマンドは、以下のとおりです。

[スクロールモード]

↓	: 1行ロールダウン
↑	: 1行ロールアップ
SHIFT+↑	: ページバック
SHIFT+↓	: ページフォワード
HOME	: 連続ロールダウン
DUP	: 連続ロールアップ
↵	: エディットモードに切り換え
BREAK	: 終了

[エディットモード]

CLS	: 1画面やり直し
CTRL+C	: カーソル位置に1行インサート
↵	: 入力後、スクロールモードに移る
BREAK	: 終了

このプログラムはまず CLEAR, &H6B00 ↵ でマシン語領域を確保してから \$6B00 からロードして、EXEC &H6B00 ↵ で実行してください(動作する F-BASIC は V3.0 のみです)。テキストサーチ&リプレイスと同様に未定義命令の PEN に割り当てられます。終了するときには、EXEC &H6B00 ↵ にて PEN への割り当てを解除してください。

なお、拡張 EDIT のコマンド形式は次のとおりです。

PEN EDIT [行番号]

リスト 2-10 拡張 AUTO & EDIT

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6B00 : 8E CF 0A 8C 02 58 27 12 BF 02 58 86 39 A7 8C F0 : 81
6B10 : AE 8C 50 34 10 30 8C 40 20 1E 30 8C 48 BF 02 58 : 25
6B20 : 86 FF 97 00 97 01 0F 02 0F 03 8E 05 9D AF 8C 33 : A5
6B30 : 30 8C CC 34 10 30 8C 17 BD 9B DB 35 10 BF 05 9D : 78
6B40 : 30 1F 9F 45 30 89 FE 04 9F 3F BD 8F 51 7E 8F E1 : 27
6B50 : 50 45 4E 20 4F 4E 0D 0A 00 50 45 4E 20 4F 46 46 : 95
6B60 : 0D 0A 00 00 00 9D 02 27 0A 81 9A 10 27 01 1B 81 : A6
6B70 : 9D 27 03 7E 92 A0 8E 03 E8 9F 9E 8E 00 0A 9F A0 : 04
6B80 : 0F C1 0F C4 0F AC 0F 27 9D 02 27 35 81 2C 27 0B : 3E
6B90 : BD 9F 62 9E 4B 9F 9E 9D DB 27 26 BD 92 92 81 2C : 34
6BA0 : 27 10 BD 9A 3A 9E 4B 9F A0 26 03 7E 96 63 9D 0B : 05
6BB0 : 27 0F BD 92 92 BD 98 F1 D7 C1 9F 67 9D 0B BD 9F : CC
6BC0 : 5C 86 7E B7 02 60 30 8C 0B 8F 02 61 32 62 7E 8E : FF
6BD0 : 88 32 66 0D AC 10 26 00 82 7D 05 A9 26 34 DC 9E : 90
6BE0 : DD 4B BD B6 15 BD 9C 22 BD 8F 1C 24 1C D6 C1 27 : 91
6BF0 : 0C 9E 67 A6 80 BD 0D 8E 5A 26 F8 20 15 86 20 D6 : 7B

```

```

[cs] : 03 9B A0 B5 33 5D 0B 03 C9 3E 65 EC 95 97 EB 37 : 37

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6C00 : C4 27 0F BD D0 8E 5A 20 F8 BD C1 69 8E 04 3D BD : FA
6C10 : 09 0F 30 8D 04 26 C6 06 F7 05 B7 17 03 E4 BD D7 : E0
6C20 : FA 24 0B 86 39 B7 02 60 7E 8E 72 9F D9 9D 02 23 : 86
6C30 : 09 CC 39 38 B7 02 60 7E 8D 01 4D 26 26 7D 05 A9 : FF
6C40 : 26 1E 0D 27 27 09 0F 27 BE 03 3A 9C 9E 26 11 DC : 26
6C50 : 9E D3 A0 D0 9E 25 04 81 FA 25 05 86 39 B7 02 60 : 32
6C60 : 7E 8E 88 0C 27 BD 91 62 9E 4B 9F E4 BF 03 3A 5F : 3E
6C70 : F7 03 3C 9E D9 21 5C A6 82 81 20 27 F9 81 09 27 : C4
6C80 : F5 5D 27 01 5A D7 C4 7E 8E CF 4F 5F D0 67 9D 02 : AB
6C90 : 27 0C BD 9F 62 9E 4B 9F 67 9D DB 9F 5C DC 67 : 50
6CA0 : BD 8F 1E 6D 84 26 03 7E 96 63 EC 02 FD 71 45 0F : AB
6CB0 : AC 0F 30 30 8D 03 85 C6 06 17 03 46 86 7E B7 02 : 19
6CC0 : 60 30 8C 0B BF 02 61 32 62 7E 8E 88 32 66 0D AC : BF
6CD0 : 27 0B 86 39 B7 02 60 16 00 F7 86 05 A9 26 78 96 : AC
6CE0 : 30 26 55 17 02 73 17 00 F3 BD DB 6D 26 1D 7D 03 : 09
6CF0 : 13 27 34 AE BD 04 4A 30 1C AF BD 04 46 17 02 24 : 06

```

```

[cs] : 28 34 BE F9 5B 92 3B BD D4 DC F7 D4 65 D5 A0 D5 : F2

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6D00 : 86 39 B7 02 60 7F 05 A9 7E DB 62 81 0D 27 3B 81 : 31
6D10 : 19 10 27 01 17 81 1A 10 27 00 89 81 1E 10 27 01 : CA
6D20 : B5 81 1F 10 27 01 46 B6 FD 01 81 0B 10 27 01 A6 : F1
6D30 : 81 11 10 27 01 37 20 B1 30 8D 03 56 C6 5C 17 02 : 23
6D40 : C1 30 8D 03 A9 C6 53 17 02 88 73 05 A9 0F 30 BD : 31
6D50 : DA F6 CE DA 01 20 03 CE DA 09 BD D9 F4 7F 03 13 : 6C
6D60 : 9D DE BD DA EF B6 03 13 26 89 B6 04 3D 81 02 26 : 1C
6D70 : 10 30 8D 02 C7 C6 06 17 02 88 17 01 DC BD 5D 20 : 01
6D80 : CC 97 30 27 0D 34 10 30 8D 02 FD C6 0A 17 02 72 : 22
6D90 : 35 10 EC 04 83 00 04 22 0E 26 07 B6 05 A9 27 07 : AB
6DA0 : 20 B5 4F 5F F7 05 A9 BD DB 5B 9F D9 9D 02 23 09 : 2B
6DB0 : CC 39 3B B7 02 60 7E BD D1 4D 27 03 7E 8E AF B6 : 1A
6DC0 : 05 A9 26 0D 96 30 26 09 30 8D 02 70 C6 06 17 02 : EA
6DD0 : 31 7E 8E 88 17 01 9D BD 03 16 FF 0D BD 8F 1E 9F : 35
6DE0 : 67 BD C8 B1 30 8D 03 5D AF 8D 03 57 F6 03 0D CE : 24
6DF0 : FF FF 4F EF 81 A7 8D A7 8D 5A 26 F7 AF BD 03 41 : 02

```

```

[cs] : A6 87 20 69 E6 9B 65 65 7C 95 90 69 09 9B 4C 28 : 20

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6E00 : 9E 67 EE 84 27 22 DF 67 EC 02 17 01 B2 AE BD 03 : FC
6E10 : 32 5B 5B 3A AC BD 03 29 22 0E 27 09 17 01 12 AF : BA
6E20 : BD 03 20 2D DB 17 01 09 17 00 F9 39 BD C8 B1 17 : 62
6E30 : 01 27 30 8D 03 0F F6 03 0D CE FF FF 4F EF 81 A7 : 2F
6E40 : 8D A7 8D 5A 26 F7 AF BD 02 F9 17 01 69 EE BD 02 : 53

```

```

6E50 : F2 50 58 58 30 C5 8C 71 45 25 0C AF 8D 02 E4 17 : 93
6E60 : 00 CF 17 01 2A 24 E3 17 00 8A 16 FE 7C EE 8D 02 : F6
6E70 : D0 E6 5F 27 0D 8D 32 5A 26 F8 17 00 F7 17 01 36 : DF
6E80 : 20 1F 17 00 EF EE 8D 02 88 EF 8D 02 86 C3 00 01 : 72
6E90 : BD 8F 1E EE 84 27 0D EC 02 17 01 23 8D 08 5A 26 : 51
6EA0 : FB 17 00 8D 8D 7E 16 FE 40 34 04 30 8D 01 93 C6 : 4D
6EB0 : 03 17 01 4E 3D 8D 02 91 EE 81 EF 1A AC 8D 02 81 : ED
6EC0 : 26 F6 4F 5F ED 1E 43 53 ED 1C EE 8D 02 75 33 5C : F5
6ED0 : EF 8D 02 6F 35 84 17 00 80 17 00 83 25 15 17 00 : 58
6EE0 : DE BE 71 45 30 01 26 04 F0 71 48 5A 8D 0A 5A 26 : C7
6EF0 : FB 8D 3E 8D 2F 16 FD F1 34 04 30 8D 01 47 C6 43 : CC

```

```

[cs] : 69 3F 1A AE EF 18 58 D0 18 14 6D 86 6F 92 29 F4 : 0F

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6F00 : 17 00 FF AE 8D 02 3A 30 1C EE 83 EF 04 8C 71 45 : 7F
6F10 : 26 F7 4F 5F ED 02 43 53 ED 84 33 8D 02 27 EF 8D : 26
6F20 : 02 21 35 84 EC 8D 02 18 83 71 45 54 54 BD DA 6D : 27
6F30 : 39 8D F1 DC 48 BD 86 15 BD 9C 22 8E 04 3D BD D9 : 46
6F40 : 0F AE 8D 01 FE DE 48 EF 81 4F D6 C1 CE FF FF 54 : EE
6F50 : ED 81 27 04 EF 81 20 F7 39 33 8D 01 E8 F6 03 0D : 08
6F60 : A6 C4 4C 26 06 33 44 5A 26 F6 39 EE 8D 01 D4 EC : 45
6F70 : C4 D0 48 39 EE 8D 01 C9 F6 03 0D 33 5C A6 C4 4C : 85
6F80 : 26 04 5A 26 F6 39 EF 8D 01 B9 EC C4 D0 48 39 DC : FC
6F90 : 48 9E 33 EE 84 27 05 10 A3 42 23 06 AE 84 EE 84 20 F0 EC : F8
6FA0 : AE C4 27 08 10 A3 42 23 06 AE 84 EE 84 20 F0 EC : 62
6FB0 : 02 D0 48 1C FE 39 DC 48 BD 8F 1E 24 04 5F 39 DD : 4B
6FC0 : 48 C6 06 34 10 AE 02 8C 27 10 24 13 5A 8C 03 E8 : D6
6FD0 : 24 0D 5A 8C 00 64 24 07 5A 8C 00 04 24 01 5A 35 : 4A
6FE0 : 10 34 04 BD C1 69 33 A9 FB C4 35 04 33 C5 4F D6 : 20
6FF0 : C3 34 06 34 02 1F 30 6C E4 A3 61 22 FA 35 44 D7 : 42

```

```

[cs] : 41 F3 28 BD ED 43 80 6F E6 F5 30 5E 27 98 1C 36 : 85

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : C1 39 1A 50 B6 FD 05 2B FB 86 8D B7 FD 05 34 14 : 49
7010 : CE FC 82 3D 8D 00 18 C6 0D A6 8D A7 C0 5A 26 F9 : FA
7020 : 35 14 A6 8D A7 C0 5A 26 F9 7F FD 05 1C AF 39 3F : 13
7030 : 31 32 33 34 35 36 37 38 93 03 8F 90 0F 33 87 D4 : F6
7040 : 0D 39 7E E8 2E 8E 3C E0 CE 40 0D BD E2 F4 DC 1F : 20
7050 : 93 38 B5 D4 09 D0 1F FD D4 0E B7 D4 09 8E C0 00 : 10
7060 : 8D 11 8E C7 D0 8D 0C 4F 97 7D 4C 97 7E BD E3 31 : F1
7070 : 7E E2 C5 34 10 DC 36 3D 33 8B 93 35 30 8B EC 83 : 68
7080 : 36 06 AC E4 26 F8 35 9D DC 58 DD 79 5F D0 45 7E : 38
7090 : E1 FC DC 79 D0 58 BD E3 C3 96 58 E6 A4 28 01 4C : BA
70A0 : 91 4E 25 01 39 BD E2 21 34 06 4F 5F 93 3B 85 D4 : 3D
70B0 : 09 0D 23 26 1F CE 3E 8D 30 C8 AC E4 27 28 EC 83 : 56
70C0 : 36 06 EC 89 40 0D ED C9 40 0C EC 89 8D 00 ED C9 : 92
70D0 : 8D 00 20 E6 CE 8E 8D 30 C8 A6 E4 8A 8D A7 E4 AC : 58
70E0 : E4 27 06 EC 83 36 06 20 F6 B7 D4 09 35 90 96 58 : 19
70F0 : 9E 5C E6 84 28 01 4C 91 4E 24 26 97 7D 34 04 BD : 0E

```

```

[cs] : 89 C8 C3 4E 4D 97 1C 76 52 14 1C A5 F0 E4 07 9E : 78

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : E2 08 34 20 CE C0 00 8D 21 35 10 CE C7 D0 8D 1A : C8
7110 : 96 7D 4C 97 7E BD E3 28 E6 E0 2A 05 96 7D BD E2 : E3
7120 : C7 0F 0D DC 58 DD 45 7E E1 FC 34 10 DC 36 3D 31 : 4B
7130 : C8 93 35 30 C8 AC E4 27 06 EC 83 ED A3 20 F6 35 : 95
7140 : 9D 00 00 00 00 00 00 00 00 00 00 00 00 00 : 90
7150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 9A 27 B5 C3 6F 06 0C 5A EE FD F1 D0 DC A3 7D 62 : 1E

```

```
SAVEM "L2-10M",&H6B00,&H71A8,&H6B00
```

2-13-3 行の複写, 移動

プログラムを入力するときに同じような処理がいくつか出てくることがよくあります。また入力した処理の順番やサブルーチンの位置を変えたいときがあります。この作業は、F-BASICのスクリーンエディタでは非常に面倒で間違えやすいものです。そこでこれをコマンド入力で自動的にやってしまおうというのが、このユーティリティです。

このプログラムでは、コマンドの拡張定義という方法でCOPYとMOVEの2つのコマンドを拡張して使用しています。コマンドの形式は次のとおりです(動作するF-BASICはV3.0のみです)。

COPY 行番号 1[-行番号 2], 行番号 3[, 増分値]

MOVE 行番号 1[-行番号 2], 行番号 3[, 増分値]

どちらも行番号 1, 行番号 2 で記される範囲が複写元(移動元)で、行番号 3 が複写先(移動先)を示します。

プログラムの起動、終了方法はテキストサーチ&リプレイスと同じです。\$6F00 がプログラムのロード開始アドレス、実行開始アドレスです(リスト 2-11)。

リスト 2-11 行の複写, 移動

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6F00 : 86 06 B1 02 03 27 22 B7 02 03 8E 80 30 BF 02 04 : 4A
6F10 : 8E 92 A0 BF 02 10 86 39 B7 02 9F A7 8C E2 AE 8C : F7
6F20 : 57 34 10 30 8D 00 5C 20 37 86 08 B7 02 03 30 8C : 11
6F30 : 5E 8F 02 04 30 8D 00 8B 8F 02 10 30 8D 00 76 BF : 2E
6F40 : 02 A0 86 7E B7 02 9F 86 FF 97 00 97 01 0F 02 0F : D2
6F50 : 03 BE 05 9D AF 8C 21 30 8C A5 34 10 30 8D 00 19 : 3A
6F60 : 8D 9B 08 35 10 BF 05 9D 30 1F 9F 45 30 89 FE 04 : 97
6F70 : 9F 3F BD 8F 51 7E 8F E1 00 00 43 4F 50 59 20 4F : 13
6F80 : 4E 0D 0A 00 43 4F 50 59 20 4F 46 46 0D 0A 00 43 : F5
6F90 : 48 41 49 CE 45 52 41 53 C5 4C 4C 49 53 D4 4C 50 : 34
6FA0 : 52 49 4E D4 53 4F 55 4E C4 50 4C 41 09 43 4F 50 : 5E
6FB0 : D9 4D 4F 56 C5 81 F5 22 09 81 F4 25 05 CC C5 08 : 69
6FC0 : ED E4 39 1F 89 9D 02 C0 F4 27 06 5A 27 04 7E 92 : 97
6FD0 : A0 C6 5F D7 20 BD 9F 62 BD 8F 1C 9F 84 9D 08 81 : FB
6FE0 : DA 26 05 9D D2 BD 9F 62 DC 4B C3 00 01 BD 8F 1E : 87
6FF0 : 9F 86 DC 69 93 84 22 03 7E 96 63 D3 35 C3 00 80 : 68
-----
[cs] : F1 FD EF C8 37 9B 65 72 27 EB 75 0A 1B 30 BB C2 : A7

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : 10 93 3F 25 03 7E 8D C6 BD 92 92 BD 9F 62 9E 4B : 63

```

```

7010 : 9F 82 8E 00 0A 9D 0B 27 0B 8D 92 92 8D 9A 3A 8D : 8F
7020 : 9F 5C 9E 4B 9F 7A 8D 8F 51 9E 84 DE 3F 33 44 11 : 61
7030 : 93 41 25 05 C6 0E 7E 8D 01 9C 86 27 1B EC 02 ED : ED
7040 : 5C DC 82 81 FA 24 0E ED 5E D3 7A 24 02 86 FA DD : 82
7050 : 82 AE 84 20 D8 7E 96 63 86 FF A7 5C 10 9E 3F EC : 84
7060 : 22 8D 52 24 F0 31 24 86 FF A1 A4 26 F2 10 9E 3F : 39
7070 : EC A4 81 FF 27 42 8D 3D EC 84 DD 5F 93 69 DD 13 : DB
7080 : CE 03 3C A6 80 A7 C0 9C 5F 26 F8 EC 22 FD 03 3E : FF
7090 : 8D 23 DC 35 DD 61 D3 13 DD 5F 8D 8D 93 CE 03 3C : 0B
70A0 : A6 C0 A7 80 9C 63 26 F8 9E 5F 9F 35 9E 69 BD C7 : 06
70B0 : 32 31 24 20 BB 7E 8F 1E 0D 20 27 0F 10 9F 78 10 : 27
70C0 : 9E 3F 31 22 10 9F 88 EC A4 20 1E 10 9E 3F EC A4 : B2
70D0 : 81 FF 27 12 8D DF EC 84 DD A4 BD 9F 05 9E 69 BD : 0B
70E0 : C7 32 31 24 20 E8 CC 0D 00 8D CA 86 FF 33 04 AE : E3
70F0 : 84 9F 76 AE C0 8C FE F2 26 1B 33 43 AE 5E 9F 7A : 5F
-----
[cs] : 6A 93 4B BA 8C 93 7B 43 47 F0 23 8E DD F9 05 FB : 90

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : 10 9E 3F AE A4 9C 7A 27 0B A1 A4 27 0B 31 24 20 : 6D
7110 : F2 AE 22 AF 5E 11 93 76 26 D9 9E 76 10 AE 84 27 : 65
7120 : 1A 0D 20 26 05 10 AE 02 20 C3 10 9E 88 31 24 10 : 8D
7130 : 9F 88 10 9C 78 22 04 EC A4 20 AE 7E 8F E1 00 00 : 8D
7140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : BB E1 91 1F 7F DF BF 8B F2 5D 00 B9 2F F1 CC 57 : 3F

```

SAVEM "L2-11M",&H6F00,&H713D,&H6F00

2-13-4 クロスリファレンス

GOTO 文の飛び先、サブルーチンの使われ方、RESTORE の行など、行番号の使用状態を調べるユーティリティです。

使用される命令ごとに分類されて、

指定された行 ← 指定している行

という形式で出力されます(図 2-20)

プログラムの実行は、CLEAR,&H6F00  としてから \$6F00 からロードして EXEC &H6F00  で実行します (動作する F-BASIC は V3.0 のみです)。

```

GOTO..
290 <- 150
360 <- 860
440 <- 540
530 <- 470 480
570 <- 920
610 <- 650 660 670 680 690 700
620 <- 730 760 860
630 <- 630
650 <- 630
660 <- 630
670 <- 630
680 <- 630
690 <- 630
700 <- 630

GOSUB..
180 <- 610 650 660 670 680 690 700 900

RESTORE
210 <- 330
220 <- 340
230 <- 950
260 <- 970

```

図 2-20 クロスリファレンス出力例

リスト 2-12 クロスリファレンス

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6F00 : 8D 8F 51 17 02 36 CE 00 00 8E 80 00 9F A0 EF 81 : 77
6F10 : 8C FC 00 26 F9 17 02 2A 9E 33 20 04 9E 9E AE 84 : 4D
6F20 : 9F 9E EC 81 10 27 01 57 EC 81 DD 47 9F D9 20 02 : 64
6F30 : 9D D2 9D D8 81 3A 27 F8 4D 27 E1 81 8C 27 DD 81 : A5
6F40 : 8D 27 D9 81 FF 26 04 9D D2 20 E5 81 83 26 07 BD : 99
6F50 : 9D A0 9D D8 20 DC 81 22 26 08 9D D2 81 22 27 D0 : 7E
6F60 : 4D 27 CF 20 F5 81 FE 26 0E 9D D2 9E D9 81 F2 27 : 8B
6F70 : 33 30 86 9F D9 20 B9 81 87 26 0B 5F 9D D2 81 CD : 8F
6F80 : 27 01 5C 16 00 C2 81 D3 26 4B 9D D2 81 E5 25 A2 : 8D
6F90 : 81 E7 23 F6 AE 9F 00 D9 8C FE F2 26 95 C6 03 D7 : 7E
6FA0 : C1 16 00 A8 30 1F A6 82 81 20 27 FA 9F D9 5F 81 : 10
6FB0 : 88 10 27 00 93 81 D6 10 27 00 8D 81 8F 10 27 00 : B4
6FC0 : 87 81 8B 10 27 00 81 C6 02 81 8A 27 7B 5C 81 9C : 39
6FD0 : 27 76 7E 92 A0 81 97 26 30 9D D2 C6 03 81 9B 27 : 36
6FE0 : 20 81 CB 27 1A 81 81 27 13 81 B2 27 0F AE 9F 00 : CF
6FF0 : D9 8C FF A9 10 26 FF 3A 9D D2 20 03 BD 92 8B C6 : AB
-----
[cs] : BA 2B 1E D4 D8 7A F9 6A A0 31 2E A6 70 8A 2C 8C : E6
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : 04 D7 C1 9D D2 9D D2 20 41 81 99 27 39 81 9A 27 : 97
7010 : 35 81 9D 27 31 81 9E 27 2D 81 A1 27 29 81 B8 27 : F3
7020 : 06 81 F0 10 26 FF 09 9D D2 10 27 FF 05 81 2C 27 : 33
7030 : 15 81 DA 27 11 81 2E 27 0D 81 FE 26 06 C6 05 D7 : D8
7040 : C1 20 09 BD 9B F1 C6 05 D7 C1 9D D2 9D D8 10 27 : AE
7050 : FE E0 81 2C 27 F4 81 DA 27 F0 81 2E 27 EC 81 FE : 59
7060 : 10 26 FE CE BD 9A 3A 17 00 D2 9E A0 D6 C1 E7 80 : 8B
7070 : DC 4B E0 81 DC 47 ED 81 9F A0 17 00 C5 20 CD 17 : 45
7080 : 00 BA 10 8E 80 00 30 A4 30 05 9C A0 24 32 A6 84 : 9D
7090 : A1 A4 22 F4 25 10 EC 01 10 A3 21 22 EB 25 07 EC : 76
70A0 : 03 10 A3 23 24 E2 A6 84 E6 A4 A7 A4 E7 84 EC 01 : 36
70B0 : EE 21 ED 21 EF 01 EC 03 EE 23 ED 23 EF 03 20 C8 : F7
70C0 : 31 25 10 9C A0 25 BF 8D 79 30 8D 00 CA 9D DE 25 : B3

```

```

70D0 : 03 73 05 AC C6 FF D7 30 CE 80 00 BD D6 78 11 93 : F0
70E0 : A0 10 27 00 A7 8D 55 E6 C0 8D 57 C1 05 27 76 D1 : 1E
70F0 : 30 27 10 D7 30 58 30 8D 00 9F AE 85 BD 4A 8E FF : 89
-----
[cs] : 95 29 A8 18 87 60 DE DE 05 01 15 9F E3 52 77 C9 : 53
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : FF 9F 67 8D 37 AE C1 8D 39 9C 67 27 1A 9F 67 8E : D6
7110 : FF FF 9F 8F 8D 48 DC 67 8D 3D BD 9C 22 86 3C BD : 08
7120 : 00 8E 86 2D BD D0 8E 8D 13 AE C1 8D 15 9C 8F 27 : 2F
7130 : AA 9F 8F BD 9C 22 DC 8F BD 1D 20 9F 1A 50 B7 FD : 45
7140 : 0F 39 1C AF 85 FD 0F 39 34 50 BD 98 50 BD 98 50 : E1
7150 : 35 10 BD 98 D8 35 C0 34 40 BD B6 15 35 C0 34 40 : D2
7160 : BD 98 50 35 C0 30 8C 62 8D DE 33 42 8E FF FF 9F : C6
7170 : 8F 8D C9 EC C1 8D C8 10 93 8F 27 07 DD 8F BD D7 : 1A
7180 : BD 9C 22 11 93 A0 24 04 33 43 20 E5 8D D0 7F 05 : 43
7190 : AC BD 8F 48 7E 8E 72 17 00 71 A2 71 A9 71 B1 71 : 98
71A0 : 89 71 C1 47 4F 54 4F 52 2E 00 47 4F 53 55 42 2E : 2E
71B0 : 2E 00 52 45 53 54 4F 52 45 00 2E 2E 45 52 52 4F : E6
71C0 : 52 00 4F 4E 20 69 6E 74 2E 2E 00 45 44 49 54 2E : 04
71D0 : 2E 00 0A 00 00 00 00 00 00 00 00 00 00 00 00 : 45
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : D8 13 2A A7 01 16 CF FE CE 00 09 00 6D 4D 5C 96 : 23

```

SAVEN "L2-12M", &H6F00, &H71D3, &H6F00

2-13-5 ラベルコンダクタ

行番号というのは無味乾燥でわかりにくく、変数名のような文字列が使えないものかと思えます。このような文字列による行番号(ラベル)がサポートされたBASICはすでにいくつもあります。F-BASIC V3.0にもラベルをサポートしようというのがこのエータイライです(リスト2-13)。

プログラム起動は、CLEAR, &H7000 』としてから\$7000番地以降にプログラムをロードして、EXEC &H7000 』で実行します。プログラムを終了させるときには、もう一度EXEC &H7000 』を入力する必要があります(動作するF-BASICはV3.0のみです)。

このプログラム動作中には、次のような形式で定義したラベルをBASICのGOTO文などの行番号の代わりに使用できるようになります。

ラベルの定義……行番号 *ラベル名

ラベル名の規則は、変数名の規則に準じます。ラベルは、行番号を使用するほとんどの命令に対応しています。ラベルを使用するうえでの注意点は、次のものがあります。

- ① ON~GOTO 文等での行番号とラベルの混在はできない
- ② RENUM コマンドは第2パラメータのみ有効
- ③ AUTO コマンドは使用できない

〔使用例〕

```
100 * LOOP: IF INKEY$="" THEN * LOOP
```

リスト2-13 ラベルコンダクタ

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : 30 8C 74 BC 02 70 27 24 B6 02 6F FE 02 70 A7 8D : 74
7010 : 01 2F EF 8D 01 2C 86 7E 87 02 6F BF 02 70 30 8C : F2
7020 : 38 FE 05 9D EF 8C 31 33 8C 05 20 19 A6 8D 01 11 : 96
7030 : EE 8D 01 0E 87 02 6F FF 02 70 86 39 A7 8C C1 30 : 06
7040 : 8C 25 EE 8C 13 DC 3F 93 45 FF 05 9D 0F 45 33 CB : F4
7050 : DF 3F BD 98 DB 7E 8F 51 00 00 0D 0A 4C 61 62 65 : 3A
7060 : 6C 20 6F 6E 2E 0D 0A 00 0D 0A 4C 61 62 65 6C 20 : C5
7070 : 6F 66 66 2E 0D 0A 00 DE D9 34 56 DC 47 BD 8F 1E : 4E
7080 : 30 04 9F D9 9D DB 9E D9 9F 29 35 70 DF D9 11 93 : 61
7090 : 29 26 0C 81 DB 26 08 17 01 29 32 62 7E 90 A0 1E : 86
70A0 : 12 10 8C FE 02 26 1A AE 62 8C A9 C0 27 07 8C 9F : 4C
70B0 : 71 10 26 00 84 32 64 17 01 09 BD 9A 02 9F 48 0E : 33
70C0 : 08 10 8C DB 02 26 72 30 1E 9C D9 27 77 AE 62 8C : E6
70D0 : A7 68 26 05 8E A7 5A 20 12 8C A9 FF 26 05 8E A9 : 91
70E0 : F5 20 08 8C 9F 38 26 0C 8E 9F 21 32 62 17 00 D3 : 81
70F0 : AF E4 0E DB 9E D9 A6 B2 81 20 27 FA 81 D6 27 04 : 5C
-----
[cs] : 9C F6 0E 53 9D D2 E1 29 68 54 CF 71 2B 70 C8 32 : FD
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : 81 8F 26 08 32 62 17 00 BA 7E 90 55 C6 FF 81 E5 : 31
7110 : 25 0D 81 E7 22 23 5F A6 82 81 20 27 FA 20 EF 10 : 47
7120 : 83 D3 00 26 14 30 8D 00 06 AF 62 DE D9 20 48 9E : 21
7130 : 48 9F 76 17 00 8D 7E BD 19 0D 81 26 04 06 AC D7 : 99
7140 : 30 00 00 00 1F 13 AE 62 8C 8F C1 27 2A 8C 90 5A : 15
7150 : 27 25 8C 91 5A 27 20 8C 9F 71 27 18 8C A7 1E 27 : 60
7160 : 16 8C A7 5D 27 11 8C A9 C0 27 0C 8C C9 15 27 07 : 9E
7170 : 8C C9 D0 27 02 20 C2 33 41 DF 29 DE 33 30 44 9F : D0
7180 : D9 9D DB 81 DB 26 20 9E 29 9D D2 27 06 A1 80 26 : 9A
7190 : 16 20 F6 A6 84 27 20 81 3A 27 1C 81 2C 27 18 81 : 08
71A0 : DA 27 14 81 20 27 10 EE C4 26 D2 C6 08 D7 AC 9E : 86
71B0 : 29 9F D9 9D DB 20 8A EC 42 DD 4B 9F D9 32 62 8D : AF
71C0 : 02 0E DB 0F AC 0D B1 27 04 D6 30 D7 AC 39 00 00 : 4E
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 61 19 C0 95 0D 4E 2B 4D F4 5E 1B 10 0E 97 23 63 : 47

```

```
SAVEM "L2-13M",&H7000,&H71CD,&H7000
```

3-1 F-BIOS の位置づけ

F-BASIC には、BIOS(Basic Input Output System)と呼ばれるプログラム・モジュールがあります。そして、I/O 機器への入出力は、BIOS を介してまとめて行なうようになっています。I/O 機器への入出力は、各 I/O 機器のハードに左右され複雑なものです。しかし FM シリーズでは、BIOS を利用することにより比較的容易に入出力が行なえるようになっています。

この BIOS がサポートしている I/O 機器には、以下のものがあります。

- ① 内蔵ブザー
- ② サブシステム(ディスプレイ、キーボード)
- ③ オーディオカセット
- ④ 320KB・フロッピーディスク
- ⑤ プリンタ
- ⑥ 漢字 ROM

これらの BIOS には、パラメータの違いによって、RCB(Request Control Block)インターフェースとレジスタインターフェース(F-BASIC V3.3 のみ)のふたつのインターフェースがあります。

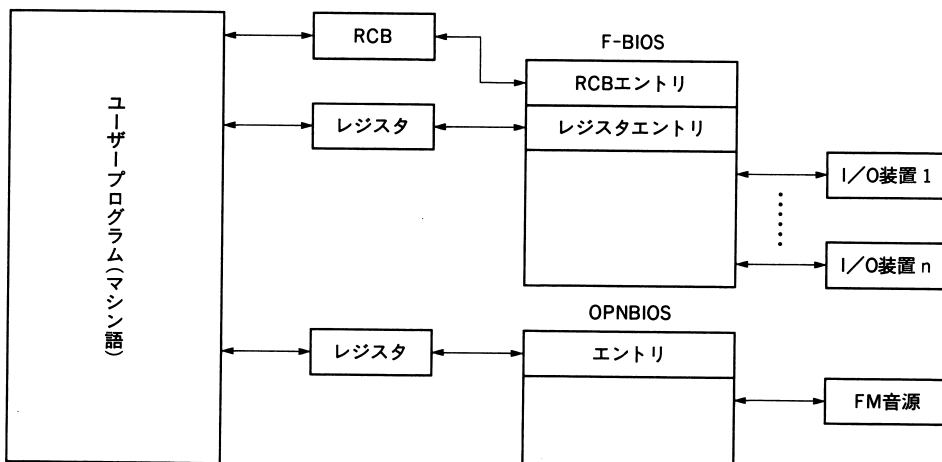


図3-1 F-BIOSの位置づけ

またF-BASIC V3.3には、FM音源制御用のOPNBIOSがあります。

図3-1に、BIOSとユーザープログラム(マシン語)およびI/O機器との関係を示します。

3-2 F-BIOS, OPNBIOSの種類

I/O機器に対するプログラムからの処理要求には、種々雑多なものがあります。そのためBIOSには、リクエスト番号によって識別される多くの種類が用意されています。そして、BIOSエントリあるいはOPNBIOSエントリにおいてリクエスト番号が識別され、各I/Oドライバールーチンへと分岐します。

リクエスト番号	名 前	内 容	F-BASIC*	
			V3.3	V3.0
1	MOTOR	テープレコーダのモーターのON, OFFを行ないます。	○	○
2	CTBWRT	カセットテープに1バイトのデータを書き込みます。	○**	○
3	CTBRED	カセットテープより1バイトのデータを読み込みます。	○**	○
4	リザーブ	—	—	—
5	SCREEN	ハードコピーを行ないます(HARDC2に相当)。	×	○
6~7	リザーブ	—	—	—
8	RESTOR	ドライブのヘッドをトラック0に移動します。	○	○
9	DWRITE	ディスクに1セクタ分のデータを書き込みます。	○	○
10	DREAD	ディスクから1セクタ分のデータを読み込みます。	○	○
11	リザーブ	—	—	—
12	BEEPON	ベルをONにして音を出します。	○	○
13	BEEPOF	ベルをOFFにして音を止めます。	○	○
14	LPOUT	プリンタにプリントデータを出力します。	○	○
15	HDCOPY	ハードコピーを行ないます(HARDC1に相当)。	×	○
16	SUBOUT	サブシステムにコマンドやデータを送ります。	○	○
17	SUBIN	サブシステムにコマンドやデータを送り、結果を受け取ります。	○	○
18	INPUT	サブシステムよりキー入力された1行を入力します。	○	○
19	INPUTC	サブシステムからの1行入力を継続して行ないます。	○	○
20	OUTPUT	画面に文字列を出力します。	○	○
21	KEYIN	キーボードより1文字入力します。	○	○
22	KANJIR	漢字のパターン(文字フォント)を取り出します。	○	○
23	LPCHK	プリンタのレディチェックを行ないます。	○	○
24	BIINIT	BIOSを初期化(イニシャライズ)します。	○	○
25~35	リザーブ	—	—	—
36	SEEKT5	フロッピーディスクドライブのヘッドを移動します。	○	×

*F-BASICの欄は、“○”が使用可を、“×”が使用不可を示します。

**V3.3では1ブロックのデータの読み込み/書き込み処理となります。

図3-2-A F-BIOS(RCBインタフェース)一覧表

F-BIOS には、21 種類の RCB インターフェースと 3 種類のレジスタインターフェースがあります。また OPNBIOS には、15 種類のインターフェースがあります。

これら F-BIOS および OPNBIOS の一覧表を、図 3-2-A、図 3-2-B、図 3-2-C に示します。F-BASIC V3.0 と V3.3 において使用できる BIOS に、若干の違いがあります。

リクエスト番号	名 前	内 容	F-BASIC	
			V3.3	V3.0
0	ACHROT	画面に1文字出力します。	○	×
1	リザーブ	—	—	—
2	APRTOT	プリンタヘプリントデータを出力します。	○	×
3～6	リザーブ	—	—	—
7	PRTCHK	プリンタのレディチェックを行ないます。	○	×
8	リザーブ	—	—	—

図3-2-B F-BIOS(レジスタインタフェース)一覧表

リクエスト番号	名 前	内 容	F-BASIC	
			V3.3	V3.0
0	WRTSSG	SSG 音源レジスタに1バイトデータを書き込みます。	○	×
1	DWTSSG	SSG 音源レジスタに2バイトデータを書き込みます。	○	×
2	REDSSG	SSG 音源レジスタより1バイトデータを読み込みます。	○	×
3	SSGCLR	全てのSSG 音源レジスタの内容をクリアします。	○	×
4	FRQSET	SSG 周波数データを書き込みます。	○	×
5	VOLSET	SSG 音量データを書き込みます。	○	×
6～7	リザーブ	—	—	—
8	WRTFM	FM 音源レジスタに1バイトデータを書き込みます。	○	×
9	KEYON	1チャンネルのキーオン/キーオフを設定します。	○	×
10	KOFFAL	すべてのFM 音源チャンネルのスロットをキーオフします。	○	×
11	WRTPRM	音色データを書き込みます。	○	×
12	FNOSSET	音階データを書き込みます。	○	×
13	TTLSET	トータルレベルを書き込みます。	○	×
14	REDSTR	ステータスレジスタの内容を読み込みます。	○	×
15	TRSPRM	音色データを転送します。	○	×
16	SELCAR	キャリアを判定します。	○	×

図3-2-C OPNBIOS 一覧表

3-3 BIOS インターフェース

ユーザープログラムと BIOS とのデータのやり取りにおけるきまりを、BIOS インターフェースと呼びます。F-BIOS には、RCB インターフェースとレジスタインターフェースおよび OPN-BIOS インターフェースの 3 種類があります。

(1) RCB インターフェース

RCB インターフェースは、メモリ上に用意した 8 バイトの RCB と呼ばれるパラメータの受け渡し領域を用いて、BIOS を使用するインターフェースです。

RCB は、メモリ上にユーザープログラムにて確保された BIOS のインターフェース領域であり、BIOS の種類を示すリクエスト番号、エラーステータス、各種パラメータからなります(図 3-3)。

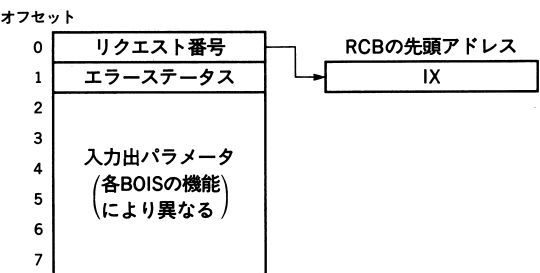


図3-3 RCBの形式

RCB の領域の確保は、6809 アセンブラの RMB(Reserve Memory Byte)命令を用いて行なうことができます。たとえば、次のような RMB 命令をアセンブラプログラム中に書くことによって RCB 領域を確保します。

RCB RMB 8 Reserve RCB area

また RCB の各パラメータごとにラベルをつけて、RCB 領域を確保するとわかりやすくなるでしょう(リスト 3-1-A)。

リスト 3-1-A RCB の確保

1000	'RGND	RMB	1	Request No.
1010	'RCBSTA	RMB	1	Error Status
1020	'RCBOBA	RMB	2	Data Buffer Top Address
1030	'RCBTRK	RMB	1	Track No.
1040	'RCBSCT	RMB	1	Sector No.
1050	'RCBSID	RMB	1	Side No.
1060	'RCBUNT	RMB	1	Unit No.

RCBのパラメータの値は、BIOSが処理を行なった結果(復帰情報)を返すためのパラメータを除いて、BIOSによって変更されないもので、FCB(FDB)命令を用いて、あらかじめ設定しておくのも便利です(リスト 3-1-B)。

リスト 3-1-B RCBの確保

1000	'RQND	FCB	20	Request No.
1010	'RCBSTA	RMB	1	Error Status
1020	'RCBDBA	FDB	PRTBUF	Data Buffer Top Address
1030	'RCBLNH	FDB	6	Number of Character
1040	'	RMB	2	RESERVE
1050	'			
1060	'			
1070	'PRTBUF	FCC	/FM77AV/	

プログラムによって確保されたRCBの受け渡しには、XレジスタにRCBの先頭アドレスを設定することにより行ないます。

F-BIOSを呼び出すには、BIOSをサブルーチンコールします。RCBインターフェースのエントリアドレスは、\$FBFA, \$FBFB番地に格納されていますので、\$FBFA番地の間接アドレッシングを用いて、サブルーチンコールします。

JSR [\$FBFA]

あるいは、

BIOS EQU \$FBFA

JSR [BIOS]

F-BIOSは、エラーの有無をキャリーフラグに、エラー番号をRCBのエラーステータスバイトに設定します。キャリーフラグの値は、エラーのあったときにON("1")に、正常終了のときにOFF("0")になります。エラーステータスバイトは、正常終了のとき0、エラーのあったときには、そのエラー番号が設定されます。

F-BIOSでは単にエラーを検出するだけで、エラー処理は一切行ないません。それで、F-BIOSからユーザープログラムに戻ってきたとき、ユーザープログラムにてキャリーフラグ、エラーステータスをチェックして、適切なエラー処理を行なう必要があります。

F-BIOSには、リクエストされた機能によりRCBに出力パラメータ(復帰情報)を設定するものがあります。詳細は、各BIOSの説明または、FM77AV BIOS解説書を参照してください。

RCBインターフェースでは、以下のレジスタがBIOSの呼び出し前と呼び出し後にて変化しない(保存される)ことが、保証されています。

- ① AccA, AccB (アキュムレータ)
- ② IX, IY (インデックスレジスタ)

- ③ US (ユーザースタックポインタ)
- ④ DP (ダイレクトページレジスタ)
- ⑤ CC レジスタの E, F, I フラグ

(2) レジスタインターフェース

F-BASIC V3.3 において新たに設定された BIOS です。V3.0 では使用できません。このレジスタインターフェースは、CPU のレジスタを直接用いて、リクエスト番号やパラメータの受け渡しを行なう高速インターフェースです(図 3-4)。

BIOS を呼び出すときには、AccB にリクエスト番号、AccA または IX にパラメータをセットします。そしてレジスタインターフェースのエントリアドレスは、\$FBA7, \$FBA8 番地に格納されていますので、\$FBA7 番地の間接アドレッシングを用いて、サブルーチンコールします。

JSR [\$FBA7]

レジスタインターフェースでは、エラーの有無をキャリーフラグに、エラー番号を\$FF98 番地のエラーステータスバイトに設定します。キャリーフラグの値は、エラーのあったときに ON ("1")に、正常終了のときに OFF("0")になります。エラーステータスバイトは、正常終了のときに 0、エラーのあったときにはそのエラー番号が設定されます。

レジスタインターフェースでは、復帰情報(BIOS が設定する出力パラメータ)が AccA, AccB に設定されることになっています。しかし、現在サポートされている 3 種類のレジスタインターフェースでは、復帰情報はありません。

レジスタインターフェースでは、以下のレジスタが BIOS の呼び出し前と呼び出し後に変化しない(保存される)ことが保証されています。

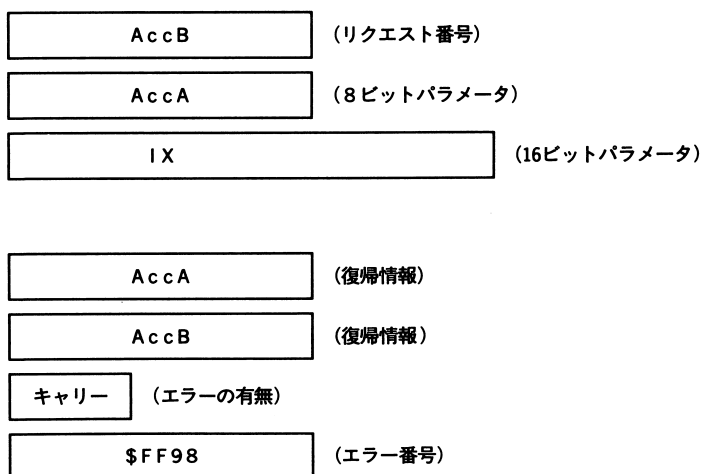


図3-4 レジスタインタフェースの形式

- ① IX, IY (インデックスレジスタ)
- ② US (ユーザースタックポインタ)
- ③ DP (ダイレクトページレジスタ)
- ④ CC レジスタの E, F, I フラグ

(3) OPNBIOS インターフェース

OPNBIOS は、F-BASIC V3.3 において新たにサポートされた FM 音源制御用 BIOS で、V3.0 では使用できません。

FM-7 では、FM 音源カードを拡張しても、その難解な FM 音源制御がサポートされていなかったため、FM 音源の利用には大きな困難が伴いました。しかし FM77AV では、この OPNBIOS がサポートされ BASIC でも FM 音源を扱えるようになったため、FM 音源が私達の身近なものとなりました。

OPNBIOS のインターフェースは、エントリアドレスが異なる点を除いて、レジスタインターフェースと同様です(図 3-5)。OPNBIOS のエントリアドレスは、\$FB9B、\$FB9C 番地に格納されていますので、\$FB9B 番地の間接アドレッシングを用いて、サブルーチンコールします。

JSR [\$FB9B]

OPNBIOS の処理モジュールは、RAM 空間の\$0F000～\$0FFFF の領域におかれる非常駐モジュールです。OPNBIOS がコールされると MMR の設定がなされ、OPNBIOS の非常駐モジュール領域が、CPU 空間の\$D000～\$DFFF にマッピングされ処理されます。処理が終了すると MMR が元に戻され、ユーザープログラムに復帰します。ですからユーザープログラムでは、MMR に関しては何も考慮する必要はありません。

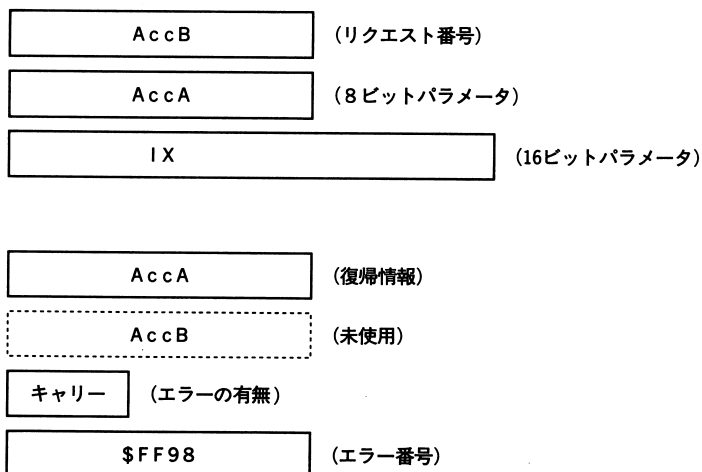


図3-5 OPNBIOSインタフェースの形式

3-4 BIOS の呼び出し手順

(1) RCB インターフェース

ユーザプログラムが RCB インターフェースの F-BIOS を使用するときの手順は、次のようになります(図 3-6)。

- ① RCB 領域として 8 バイトをメモリ上に確保します。
- ② RCB に、BIOS のリクエスト番号を始めとするパラメータをすべて設定する。
- ③ IX レジスタに RCB の先頭番地をセットします。
- ④ \$FBFA 番地の間接アドレッシングを用いてサブルーチンコールします。(JSR[\$FBFA])
- ⑤ キャリーフラグによってエラーの有無を調べて、エラーがあったときには、ステータスバイトを調べ必要に応じてエラー処理を行ないます。

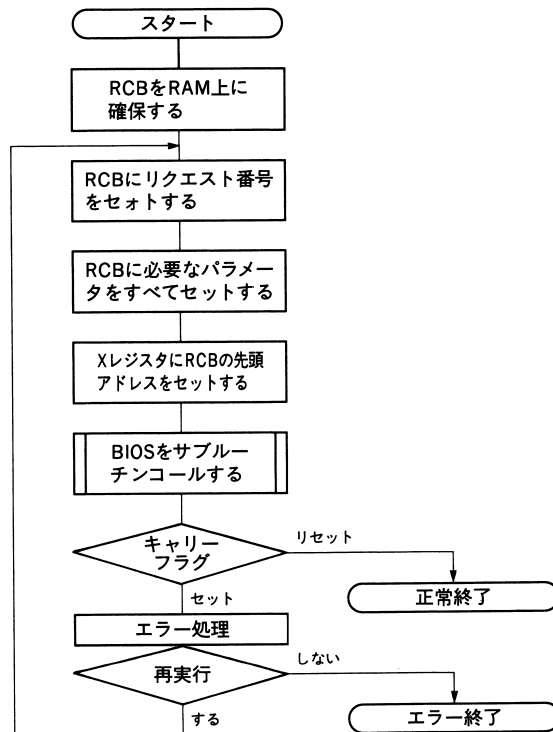


図3-6 RCBインタフェースの呼び出し手順

(2) レジスタインターフェース

ユーザプログラムがレジスタインターフェースの F-BIOS を使用するときの手順は、次のようになります(図 3-7)。

- ① アキュムレータ B(AccB)にリクエスト番号を, アキュムレータ A(AccA)またはインデックスレジスタ X(IX)にパラメータを設定します。
- ② \$FBA7 番地の間接アドレッシングを用いてサブルーチンコールします(JSR[\$FBA7])。
- ③ キャリーフラグによってエラーの有無を調べて, エラーがあったときには, ステータスバイト(\$FF98 番地)を調べ必要に応じてエラー処理を行ないます。

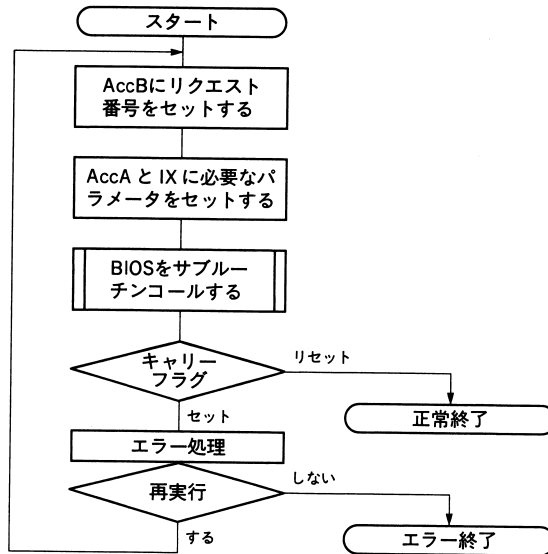


図3-7 レジスタインタフェースの呼び出し手順

(3) OPNBIOS

ユーザープログラムがOPNBIOSを使用するときの手順は, 次のようになります。

- ① アキュムレータ B(AccB)にリクエスト番号を, アキュムレータ A(AccA)およびインデックスレジスタ X(IX)にパラメータを設定します。
- ② \$FB9B 番地の間接アドレッシングを用いてサブルーチンコールします(JSR[\$FB9B])。
- ③ キャリーフラグによってエラーの有無を調べて, エラーがあったときには, ステータスバイト(\$FF98 番地)を調べ必要に応じてエラー処理を行ないます。

OPNBIOS の呼び出し手順は, レジスタインターフェースの場合と同様です。

3-5 BIOS のエラー処理

BIOS は、各種 I/O エラーについてエラー番号を定めており、エラーが起きた場合にそのエラー番号を、エラーステータスバイトに設定します。

(1) BIOS システムエラー

エラー番号	エ ラ ー 内 容
1	RCB エラー
2	Device Unavailable エラー
3	ブレーク

(2) フロッピーディスク関係エラー

エラー番号	エ ラ ー 内 容
10	ドライブノットレディ
11	ディスクライトプロテクトド
12	ハードエラー(シークエラー, ロストデータ, レコードノットファウンド)
13	CRC エラー
14	DD マーク検出 (DD マーク = Deleted Data Mark)
15	タイムオーバーエラー

(3) プリンタ、オーディオカセット関係エラー

エラー番号	エ ラ ー 内 容
50	ペーパーエンプティ
51	プリンタノットレディ
52	オーディオカセットリードエラー

(4) サブシステムエラー

エラー番号	エ ラ ー 内 容
60	INIT コマンドパラメータエラー
61	コンソール座標エラー
62	複数バイトのオーダシーケンスにおいて必要なデータがない
63	グラフィック座標エラー
64	使用できないファンクションコードまたは、未定義ファンクションコードを使用した
65	座標数が規定数より多い、または少ない
66	文字数より多い、または少ない
67	色指定数が規定数より多い、または少ない
68	ファンクションキー番号エラー
69	パラメータエラー
70	コマンドエラー

図3-8 BIOS エラー番号一覧

入出力ルーチンの実行中にエラーが発生した場合、BIOS はその入出力ルーチンの実行を中止して、エラーステータスバイトを設定して呼び出し元に戻ります。そのときには、BIOS が出力パラメータに設定することになっている値が正しく設定されているとは限りません。ただし、エラーステータスバイトだけは、正しく設定されています。

BIOS はエラー処理を一切行なわないので、エラーが発生したときには、このエラー番号をもとに必要なエラー処理を行なわなければなりません。

図 3-8 に BIOS のエラー番号を示します。なおエラーのないときのエラーステータスバイトの値は、0 になっています。

3-6 ブザーに対する BIOS

F-BIOS の利用例として、ブザーに対する BIOS コールを説明します。その他の BIOS については、各 I/O 機器の章で説明します。

BEEPON(リクエスト番号\$0C)と、BEEPOF(リクエスト番号\$0D)は、F-BASIC の BEEP1, BEEP0 に相当する動作をします。RCB には、リクエスト番号のみをセットすればいいのですが、BIOS のエラー番号セット用に 2 バイトを確保しなければなりません(図 3-9)。

リスト 3-2 にブザー音発生、停止のサンプルプログラムを示します。\$5000 番地から入力して実行してください。

EXEC &H5000 □ …… ブザー音発生

EXEC &H5002 □ …… ブザー音停止

BEEPON (ブザー音発生)

オフセット	内 容	ラベル名	ユーザ	BIOS
0	リクエスト番号	RQNO	12	
1	エラーステータス	RCBSTA		○
2~7	リザーブ	—	—	—

BEEPOF (ブザー音停止)

オフセット	内 容	ラベル名	ユーザ	BIOS
0	リクエスト番号	RQNO	13	
1	エラーステータス	RCBSTA		○
2~7	リザーブ	—	—	—

図3-9 ブザーに対する BIOS の RCB

リスト 3-2 ブザー音発生, 停止

```

00100                                     *****
00110                                     *   BEEP ON/OFF   *
00120                                     * ( LIST 3-2 )   V3.0/V3.3 *
00130                                     *****
00140                                     OPT      NOGEN
00150 5000                                     ORG      $5000
00170 5000 20 04 5006 ENTRY BRA      BEEPON
00180 5002 20 0E 5012 BRA      BEEPOF
00190                                     *
00200 5004 0002 RCB      RMB      2
00210 5006 30 8C FB BEEPON LEAX     RCB,PCR
00220 5009 86 0C LDA      #$0C
00230 5008 A7 84 STA      ,X
00240 500D AD 9F FBFA JSR      [$FBFA]
00250 5011 39 RTS
00260                                     *
00270 5012 30 8C EF BEEPOF LEAX     RCB,PCR
00280 5015 86 0D LDA      #$0D
00290 5017 A7 84 STA      ,X
00300 5019 AD 9F FBFA JSR      [$FBFA]
00310 5010 39 RTS
00320                                     *
00330 5000 END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=501D
PROGRAM ENTRY ADDR=5000

```

3-7 CRT1 文字表示

レジスタインターフェースの BIOS (ACHROT) を用いて, CRT 画面に 1 文字表示のサンプルプログラムを考えてみます。

リスト 3-3 がそれです。リクエスト番号は 0 で, AccA に CRT 画面に表示する文字コードを設

入力パラメータ

レジスタ	内 容	値
AccB	リクエスト番号	0
AccA	出力データ	文字コード
IX	_____	_____

出力パラメータ

レジスタ A	内 容	値
AccA	_____	_____
AccB	_____	_____
Carry	エラーフラグ	Carry = 1 の時エラー有り

図3-10 ACHROTの入出力インタフェース

定します(図 3-10)。RCB インターフェースを用いた BIOS コールより、ずっとわかりやすくなっています。

\$5000 番地よりプログラムを入力して、EXEC &H5000  で実行してみてください。

リスト 3-3 CRT1 文字表示

```

00100
00110
00120
00130
00140
00150 5000
00160
00170 5000 5F
00180 5001 86 46
00190 5003 AD 9F FBA7
00200 5007 25 01 500A
00210 5009 39
00220 500A 39
00230
00240 5000
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=500A
PROGRAM ENTRY ADDR=5000

*****
* 1 MOJI DISPLAY *
* ( LIST 3-3 ) V3.3 *
*****
OPT NOGEN
ORG $5000
*
ENTRY CLRB
LDA #$46
JSR [$FBA7]
BCS ERROR
RTS
ERROR RTS
*
END ENTRY

```

3-8 音色データ読み込み

OPNBIOS の TRSPRM を用いて、イニシエータ ROM 内の音色データの読み込みを行なうサンプルプログラムを考えてみます。

リスト 3-4 がそれです。リクエスト番号は 15 で、AccA に音色データの番号(1~74)、IX に音色データの転送先アドレスを設定します(図 3-11)。

入力パラメータ

レジスタ	内 容	値
AccB	リクエスト番号	15
AccA	音色番号	1~74
IX	データの転送先アドレス	16ビットアドレス

出力パラメータ

レジスタ	内 容	値
AccA	_____	_____
Carry	エラーフラグ	Carry=1の時エラー有り

図3-11 TRSPRMのインタフェース

音色データはイニシエータ ROM 内にあり、そのデータを読み込むときには、イニシエータ ROM が\$6000~\$7FFF のアドレスにマッピングされます。それで、この範囲内に OPNBIOS コールルーチンおよび音色データの転送先を置くことはできません。

なお、音色データは 34 バイトで構成され、音色および LFO データを含みます。

\$5000 番地よりリスト 3-4 を入力して、EXEC &H5000 [] で実行してください。\$5100 番地から音色データ(音色番号=1)が読み込まれます。

リスト 3-4 音色データ読み込み

```

00100                                *****
00110                                *   オンショク データ READ   *
00120                                *   ( LIST 3-4 )  V3.3   *
00130                                *****
00140                                OPT      NOGEN
00150  5000                          ORG      $5000
00160                                *
00170  5000 C6  0F      SYSTEM LDB      #15
00180  5002 86  01      LDA      #1
00190  5004 8E  5100     LDX      #$5100
00200  5007 AD  9F FB9B   JSR      [FB9B]
00210  5008 25  01  500E   BCS      ERROR
00220  500D 39          RTS
00230  500E 39      ERROR RTS
00240                                *
00250                          5000      END      SYSTEM
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END  ADDR=500E
PROGRAM ENTRY ADDR=5000

```

FM-7 シリーズでは、システム全体の制御を行なうメイン CPU と、画面処理およびキー入力処理を行なうサブ CPU のデュアル CPU 方式が採用されています。以下本章では、サブ CPU が制御するサブシステムについて説明します。

4-1 サブシステムのメモリマップ

FM77AV のサブシステムには、次のような 3 つの動作タイプがあります。

- ① タイプ C …… 8 色 1 画面 (640×200 ドット)、FM-7 完全互換
- ② タイプ A …… 8 色 2 画面 (640×200 ドット)
- ③ タイプ B …… 4096 色 1 画面 (320×200 ドット)

それぞれのタイプにおけるメモリマップを、図 4-1A、図 4-1B、図 4-1C に示します。

VRAM はページ 0 とページ 1 の 2 バンクからなり、バンクレジスタ (\$D430) のビット 5、ビット 6 によってアクティブページ、ディスプレイページが切り換えられます。しかしタイプ C では常にページ 0 が使用され、ページ切り換えは行なえません。タイプ B では B、R、G それぞれが 4 つの色バンクからなり、各色につき 16 段階の輝度が設定可能で、 $16 \times 16 \times 16 = 4096$ 色の同時表示が可能です。このタイプ B では 1 画面しかないので、ディスプレイページの指定は無意味です。しかし VRAM をアクセスするときには、アクティブページによってアクセスするページを選択しなければなりません。

4000 バイト (タイプ B では 2000 バイト) のコンソールバッファは、表示文字のキャラクタコードが格納されているキャラクタコードバッファと、表示文字の属性、表示色等のアトリビュートが格納されるアトリビュートバッファの 2 つの部分からなっています。画面に表示される最大文字数は、 $80 \times 25 = 2000$ (タイプ B では $40 \times 25 = 1000$) 文字ですので、キャラクタコードバッファおよびアトリビュートバッファはそれぞれ 2000 (タイプ B では 1000) バイト必要です。

\$CFA0~\$CFFF 番地の 96 バイトはスタックとして使用される領域で、サブ CPU のスタックポインタは \$D000 に初期設定されます。

システム領域は、サブシステムの各処理ルーチンから常時アクセスされているフラグ・レジスタ類が割り当てられている RAM 領域です。ですからこのシステム変数の内容を書き換えることによって、いろいろな処理を実現できます。たとえばインサートモードフラグ (タイプ A・B では \$D01B、タイプ C では \$D033) を \$FF にするとキー入力が入サートモードとなります。これは **INS**

LEDの点灯・消灯とは無関係に行なわれます。

このシステム変数領域に対して補助ワークエリアは、各処理ルーチン内において局所的に使われる変数のために使用される領域です。この領域の値の意味は各処理ルーチンによって異なり、各処理ルーチンの実行中にのみ意味を持ちます。

拡張コマンドテーブルは、サブシステム拡張コマンド(コマンドコード\$80～\$8F)の実行番地が登録されるテーブルです。この拡張コマンドについては後述します。

割り込みフックテーブルは、各割り込み処理の分岐先アドレスが登録されるテーブルです。このフックテーブルを書き換えることにより、割り込み処理を定義することができます。なお、拡張コマンドテーブルと割り込みフックテーブルは、V3.3のときのみ存在します。

PFキー定義テーブルは、各PFキーに対して16バイトずつ、合計160バイトの領域を占め、PFキー設定文字列およびPFキー割り込み定義フラグが格納されます。このPFキー定義テーブルの構成は、「5-1-7 PFキーの入力」を参照ください。

キー入力バッファは、キー先行入力を可能にするためキー入力データを次々に格納するための領域です。このキー入力バッファのしくみについては、「5-1-2 キー入力バッファ」を参照ください。

	ページ0	ページ1		
\$0000	VRAM(青)	VRAM(青)		
\$4000	VRAM(赤)	VRAM(赤)		
\$8000	VRAM(緑)	VRAM(緑)		
\$C000	コンソール バッファ	}	キャラクタコードバッファ	
\$C7D0			アトリビュートバッファ	
\$CFA0	ハードウェアスタック			
\$D000	システム変数領域			
\$D0C7	補助ワークエリア			
\$D278	拡張コマンドテーブル			
\$D2A8	割り込みフックテーブル			
\$D2C0	PFキー定義テーブル			
\$D360	キー入力バッファ			
\$D380	共有RAM			
\$D400	I/Oレジスタ領域			
\$D500	ワークエリア			
\$D800	キャラクタフォント (カタカナ)	キャラクタフォント (ひらがな)	サブモニタROM1	サブモニタROM2
\$E000	サブモニタROM3 (タイプA)			
\$FFF0	割り込みベクトルテーブル			
\$FFFF				

図4-1A タイプAのメモリマップ

4-1 サブシステムのメモリアップ

	ページ 0	ページ 1		
\$0000	VRAM(青)バンク3	VRAM(青)バンク1		
\$2000	VRAM(青)バンク2	VRAM(青)バンク0		
\$4000	VRAM(赤)バンク3	VRAM(赤)バンク1		
\$6000	VRAM(赤)バンク2	VRAM(赤)バンク0		
\$8000	VRAM(緑)バンク3	VRAM(緑)バンク1		
\$A000	VRAM(緑)バンク2	VRAM(緑)バンク0		
\$C000	コンソール バッファ	} キャラクタコードバッファ } アトリビュートバッファ		
\$C3E8				
\$C7D0	ワークエリア			
\$CFA0	ハードウェアスタック			
\$D000		} タイプ A と同様		
\$D500	ワークエリア			
\$D800	キャラクタフォント (カタカナ)	キャラクタフォント (ひらがな)	サブモニタROM1	サブモニタROM2
\$E000	サブモニタROM3 (タイプB)			
\$FFF0	割り込みベクトルテーブル			
\$FFFF				

図4-1B タイプBのメモリマップ

\$0000	VRAM(青)	
\$4000	VRAM(赤)	
\$8000	VRAM(緑)	
\$C000	コンソール バッファ	} キャラクタコードバッファ } アトリビュートバッファ
\$C7D0		
\$CFA0	ハードウェアスタック	
\$D000	システム変数	
\$D0A0	補助ワークエリア	
\$D2C0	PFキー定義テーブル	
\$D360	キー入力バッファ	
\$D380	共有RAM	
\$D400	I/Oレジスタ領域	
\$D410	未使用	
\$D800	キャラクタフォント (カタカナ)	
\$E000	サブモニタROM (タイプC)	
\$FFF0	割り込みベクトルテーブル	

図4-1C タイプCのメモリマップ

\$D380～\$D3FF 番地の共有 RAM(128 バイト)は、メイン CPU とサブ CPU がコマンドおよびパラメータなどの情報交換をするための共通 RAM 領域です。この共有 RAM は、メイン CPU からみると\$FC80～\$FCFF 番地に対応します。

I/O レジスタ領域は、サブシステムの I/O ポートが割り付けられている領域です。I/O レジスタについては、付録の I/O レジスタ一覧表を参照ください。

\$D800～\$DFFF 番地には、キャラクタフォント ROM(カタカナおよびひらがな)、サブモニタ ROM1、サブモニタ ROM2 が必要に応じて切り換えられて使用されます(タイプ C ではカタカナのキャラクタフォント ROM のみ)。この切り換えは、バンクレジスタ(\$D430)のビット 0、ビット 1 によって行なわれます。サブモニタ ROM1、サブモニタ ROM2 には、タイプ A、タイプ B のサブモニタにおいて共通に使用される処理ルーチンが割り当てられています。その具体的内容については、付録のサブモニタ・アドレスマップを参照ください。

\$E000～\$FFEF 番地は、サブモニタ ROM(タイプ A、タイプ B、タイプ C)の本体が割り当てられています。この切り換えは、サブバンクレジスタ(\$FD13)のビット 0、ビット 1 によって行なわれます。

\$FFF0～\$FFFF 番地には、サブ CPU 割り込みベクトルテーブルが割り当てられています。この値は、図 4-2 のとおりです。

ア ド レ ス	割り込み	設 定 値		
		タイプA	タイプB	タイプC
\$FFF0, \$FFF1	リザーブ	—	—	—
\$FFF2, \$FFF3	SWI3割り込み	\$D2A8	\$D2A8	\$E000
\$FFF4, \$FFF5	SWI2割り込み	\$D2AB	\$D2AB	\$E000
\$FFF6, \$FFF7	FIRQ割り込み	\$D2AE	\$D2AE	\$FDAC
\$FFF8, \$FFF9	IRQ割り込み	\$D2B1	\$D2B1	\$E06E
\$FFFA, \$FFFB	SWI割り込み	\$D2B4	\$D2B4	\$E000
\$FFFC, \$FFFD	NMI割り込み	\$D2B7	\$D2B7	\$FEBF
\$FFFE, \$FFFF	RESET	\$E112	\$E0D8	\$E000

図4-2 サブCPU割り込みベクトルテーブル

PAINT 処理ルーチンでは、通常補助ワークエリアをワークに使用しています。しかし処理によっては、\$D500～\$D7FF 番地(タイプ B では\$C7D0～\$CF9F 番地のエリアも)のワークエリアが使用されます。

4-2 メイン・サブインターフェース

FM-7 シリーズはデュアル CPU システムで、メイン CPU とサブ CPU が共有 RAM を介して情報交換をしながら処理していきます。この共有 RAM のアクセスは、サブ CPU、メイン CPU の同時アクセスによる不具合を防止するため同時アクセスが禁止されています。そのためメイン CPU、サブ CPU はインターフェース信号により、排他制御をしながら共有 RAM をアクセスすることになります(図 4-3)。

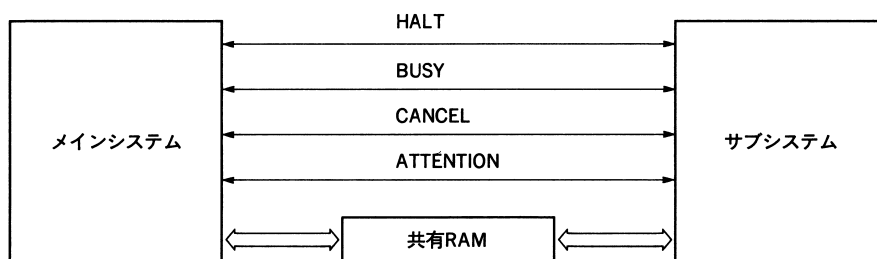


図4-3 メイン・サブインターフェース

(1) HALT(メイン→サブ)

サブ CPU の動作を停止(HALT)させて、メイン CPU が共有 RAM をアクセスできるようにするための信号です。メインシステム I/O レジスタ(\$FD05)のビット 7 に割り当てられています。

(2) BUSY(メイン←サブ)

サブ CPU がコマンドを実行中(ビジー状態)であることを示します。メインシステムの\$FD05のビット 7 に割り当てられています。メイン CPU がサブ CPU に対して処理要求するときには、この BUSY 信号のチェックを行なう必要があります。いっぽうサブ CPU 側では、\$D40A のリード(READY)/ライト(BUSY)で BUSY 信号の制御を行ないます。

メイン CPU からの HALT 要求をサブ CPU が受け取ると、この BUSY 信号はハード的に BUSY 状態となります。つまり、HALT アクノリッジ信号の役目も果たしています。

(3) CANCEL(メイン→サブ)

サブ CPU が実行中のコマンドを中断させて、ビジー状態を解除させるための信号です。メインシステムの\$FD05のビット 6 に割り当てられています。

(4) ATTENTION(メイン←サブ)

メイン CPU から依頼されていた割り込み事象(PF キー割り込み、タイマー割り込み)が発生したことを、メイン CPU に通知するための割り込み信号です。メインシステムの\$FD04のビット 0 に

割り当てられています。なお割り込み事象の原因については、共有 RAM のステータスバイトにセットされます。

(5) 共有 RAM(メイン→サブ)

メイン、サブ間でコマンドおよびデータなどの交換をするための RAM 領域です。この領域の内容は、図 4-4 のとおりです。この共有 RAM のアクセス権はサブ CPU が優先となっていて、メイン CPU がアクセスするためにはサブ CPU を HALT する必要があります。

① READY REQ ……… サブ CPU を HALT して、新たにコマンドを設定せず HALT 解除するときには、このビットを ON にします。

② ERROR CODE ……… サブシステムのコマンド実行においてエラーが発生したときに、エラーコードが設定されます。

③ データ継続フラグ ……サブ CPU とメイン CPU とのデータ転送が 1 回で終了しない場合、データが継続していることを示すために用います。

④ STATUS …………… サブ CPU がメイン CPU より依頼されている割り込みが発生するとき、ここにその割り込み原因が設定されます。

ビット 0～3：PF キー割り込み(PF キー番号)

ビット 4 ：タイマー割り込み

ビット 5 ：インターバルタイマー割り込み

ビット 6 ：0 時割り込み

⑤ COMMAND …………… サブシステムへのコマンドコードが設定されます。

⑥ DATA …………… サブ CPU とメイン CPU とでやりとりされるデータが書き込まれます。

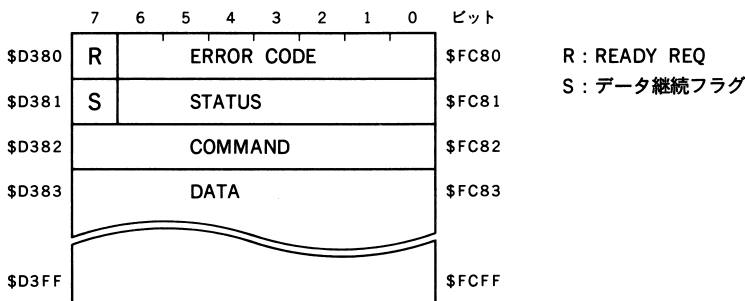


図4-4 共有RAMの内容

4-3 サブシステムに対する BIOS

サブシステムに対する BIOS には、図 4-5 に示す 7 種類があります。また、その RCB パラメータの内容を図 4-6 に示します。

リクエスト番号	名 前	内 容	V3.0	V3.3
18	INPUT	1行入力ルーチン	○	○
19	INPUTC	1行継続入力ルーチン	○	○
20	OUTPUT	画面表示ルーチン	○	○
21	KEYIN	キー入力ルーチン	○	○
16	SUBOUT	サブシステムアウトプットルーチン	○	○
17	SUBIN	サブシステムインプットルーチン	○	○
0	ACHROT	画面1文字表示ルーチン	×	○

(注) ACHROT はレジスタインタフェースの BIOS です。

図4-5 サブシステムに対する BIOS 一覧表

(4) INPUT

CRT 画面に文字列を表示した後、オペレータによって変更されたフィールドのなかで画面最上部のフィールドの内容を読み込みます。フィールド内容の読み込みは、変更の終了キー(**⏏**, **CTRL** + **C**, **CTRL** + **X**, **CLS**)が押されたときに一括してデータバッファに転送されます。

(2) INPUTC

INPUT ルーチンにおいて転送されなかった残りの変更フィールドの入力を行いません。INPUT ルーチンによって変更フィールドの内容を入力するときには、INPUTC のデータバイト数が 0 になるまで継続して INPUTC コマンドを実行しなければなりません。

(3) OUTPUT

文字列を CRT 画面に表示します。

(4) KEYIN

キーボードより 1 文字分のキーコードを入力します。「第 5 章 キー入力」を参照ください。

(5) SUBOUT

サブシステムにサブシステムコマンドおよびデータを転送します。一度に転送できるデータは 128 バイトまでであり、それ以上のデータを転送するときには、共有 RAM の継続フラグの処理をユーザプログラムで行なう必要があります。SUBOUT にてサブシステムコマンドを転送するとサブシステムで実行されますが、この SUBOUT はデータの転送をするのみで、サブシステムの実行完了を待つことなく呼び出し元へ戻ります。

SUBOUT を使用するサブシステムコマンドには、以下のものがあります。

・INIT, ERASE, PUT, PUT CHARACTER BLOCK1, PUT CHARACTER BLOCK2, TAB SET, CONSOLE CONTROL, ERASE2, CHARACTER LINE, LINE, LINE2, CHAIN, POINT, PAINT, SYMBOL, CHANGE COLOR, PUT BLOCK1, PUT BLOCK2, PUT BLOCK, SELECT DISPLAY MODE, SET VIEWPORT COORDINATE, TILE BOX, DEFINE STRING OF PF, INTERRUPT CONTROL, SET TIMER, SET RTC, KEYBOARD CONTROL

(6) SUBIN

サブシステムにコマンドおよびデータを転送してから、サブシステムの実行終了を待ってその処理結果を入力します。処理結果がデータバッファに転送されるので、サブシステムに転送したコマンドとデータは消されてしまいます。ですから連続して SUBIN コマンドを実行するときには、そのたびにコマンドとデータをセットする必要があります。

SUBIN を使用しなくてはならないサブシステムコマンドには、以下のものがあります。

・GET, GETC, GET CHARACTER BLOCK1, GET CHARACTER BLOCK2, GET BUFFER ADDRESS, READ CONSOLE PARAMETER, GET BLOCK1, GET BLOCK2, GET BLOCK, GRAPHIC CURSOR, READ DISPLAY STATUS, INKEY, GET STRING OF PF, READ TIMER, READ RTC, KEYBOARD CONTROL

(7) ACHROT

CRT 画面に 1 文字表示するレジスタインターフェースの BIOS です。F-BASIC V3.0 では使用できません。

SUBOUT を利用して LINE を描画するサンプルプログラムをリスト 4-1 に示します。

リスト 4-1 SUBOUT のサンプルプログラム

00100			*****		
00110			* SUBOUT SAMPLE *		
00120			* (LIST 4-1) V3.0/V3.3 *		
00130			*****		
00140			OPT NOGEN		
00150	5000		ORG \$5000		
00160			* ENTRY		
00170	5000 8E	5008	LDX #SUBOUT		
00180	5003 AD	9F FBFA	JSR [FBFA]		
00190	5007 39		RTS		
00200			* SUBOUT		
00210	5008	10	FCB 16	REQ NO	
00220	5009	0001	RMB 1	ERROR STATUS	
00230	500A	5010	FDB LINE	DATA ADR	
00240	500C	0010	FDB 16	DATA LEN	
00250	500E	0002	RMB 2		
00260	5010	0002	LINE RMB 2		
00270	5012	15	FCB \$15	COMMAND CODE	

```

00280 5013      07          FCB      7      COLOR
00290 5014      00          FCB      0      FUNCTION
00300 5015      0000        FDB      0      X1
00310 5017      0000        FDB      0      Y1
00320 5019      027F        FDB     639      X2
00330 5018      00C7        FDB     199      Y2
00340 5010      00          FCB      0      BOX FLG
00350 501E      FFFF        FDB     $FFFF    LINE STYLE
00360
00370          5000          *      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=501F
PROGRAM ENTRY ADDR=5000

```

(1) INPUT

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	18	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH	○	
6, 7	データバッファ長(16ビット)	RCBBMH	○	

(2) INPUTC

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	19	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH		○
6, 7	データバッファ長(16ビット)	RCBBMH	○	

(3) OUTPUT

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	20	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH	○	
6, 7	リザーブ			

(4) KEYIN

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	21	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH		○
6, 7	リザーブ			

(5) SUBOUT

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	16	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH	○	
6, 7	リザーブ			

(6) SUBIN

相 対 値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	17	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH	○	○
6, 7	入力バイト数(16ビット)	RCBBMH	○	

(7) ACHROT

[入力情報]

レジスタ	内 容	値
AccB	リクエスト番号	0
AccA	出力データ	文字コード
IX		

[出力情報]

レジスタ	内 容	値
AccA		
AccB		
Carry	エラーフラグ	Carry = 1 のときにエラーあり

図4-6 サブシステムに対するBIOSのRCB

4-4 サブシステムコマンド

サブシステムに用意されているコマンドを一覧表(図 4-7A, 図 4-7B, 図 4-7C, 図 4-7D)で示します。なおコマンドの詳細については本書ではとりあげませんので、サブシステムコマンドの使用に際しては、ディスプレイサブシステム解説書を参照ください。

FM77AV タイプ A, タイプ B のサブモニタには、直線描画用のサブシステムコマンドとして LINE と LINE2 の 2 種類があります。これは、FM77AV では直線補間回路によるハードウェア描画機能が採用されたのですが、この直線補間回路による直線補間ロジックが FM-7 と異なっているからです。そのため FM-7 と同様の直線を描くように工夫したのが LINE2 コマンドです。

LINE2 コマンドでの斜めの線の描画に関しては、直線補間回路を使用しないので LINE コマ

コ マ ン ド 名	コード	内 容	タ イ プ		
			A	B	C
INIT	\$01	コンソール機能の初期化	○	○	○
ERASE	\$02	画面およびコンソールの初期化	○	○	○
PUT	\$03	文字列を画面に表示	○	○	○
GET	\$04	文字列の表示後、変更フィールドの入力	○	○	○
GETC	\$05	GETによる未転送フィールドの入力	○	○	○
GET CHARACTER BLOCK 1	\$06	枠内の文字コードの読み取り	○	○	○
PUT CHARACTER BLOCK 1	\$07	枠内への文字の表示	○	○	○
GET CHARACTER BLOCK 2	\$08	枠内の文字コードとアトリビュートの読み取り	○	○	○
PUT CHARACTER BLOCK 2	\$09	枠内への文字(アトリビュート)の表示	○	○	○
GET BUFFER ADDRESS	\$0A	カーソル位置の読み込み	—	—	○
READ CONSOLE PARAMETER	\$0A	カーソル位置の読み込み	○	○	—
TAB SET	\$0B	TAB 位置の設定	○	○	○
CONSOLE CONTROL	\$0C	コンソール機能の選択	○	○	○
ERASE 2	\$0D	画面およびコンソールの初期化	○	○	○
CHARACTER LINE	\$20	文字によって直線または四角を描く	○	—	○

図4-7A コンソールコマンド一覧

コ マ ン ド 名	コード	内 容	タ イ プ		
			A	B	C
LINE	\$15	直線または四角形を描く	○	○	○
CHAIN	\$16	指定座標間を直線で結ぶ	○	○	○
POINT	\$17	点を表示する	○	○	○
PAINT	\$18	枠内に色をぬる	○	○	○
SYMBOL	\$19	文字列を大きさ、角度を変えて表示する	○	○	○
CHANGE COLOR	\$1A	枠内の色を変える	○	—	○
GET BLOCK 1	\$1B	枠内の指定した色のドットパターンを読み取る	○	—	○
PUT BLOCK 1	\$1C	枠内に指定した色のドットパターンを表示	○	—	○
GET BLOCK 2	\$1D	枠内のR.G.B.のドットパターンを読み取る	○	—	○
GET BLOCK	\$1D	枠内のR.G.B.のドットパターンを読み取る	—	○	—
PUT BLOCK 2	\$1E	枠内にR.G.B.のドットパターンを表示	○	—	○
PUT BLOCK	\$1E	枠内にR.G.B.のドットパターンを表示	—	○	—
GRAPHIC CURSOR	\$1F	グラフィック座標値の読み取り	○	○	○
SELECT DISPLAY MODE	\$22	カタカナ/ひらがな表示の切り替え	○	○	—
READ DISPLAY STATUS	\$24	グラフィックVRAMの動作状態の読み取り	○	○	—
LINE 2	\$36	従来のソフトウェアによる直線・四角形を描く	○	○	—
TILE BOX	\$38	タイルストリングパターンで四角形を描く	○	—	—
SET VIEWPORT COORDINATE	\$39	ビューポート座標の設定	○	○	—

図4-7B グラフィックコマンド一覧

コ マ ン ド 名	コード	内 容	タ イ プ		
			A	B	C
INKEY	\$29	キーコードを読み取る	○	○	○
DEFINE STRING OF PF	\$2A	PFキーに文字列を定義する	○	○	○
GET STRING OF PF	\$2B	PFキー定義文字列を読み取る	○	○	○
INTERRUPT CONTROL	\$2C	PFキー割り込みの選択を行なう	○	○	○
SET TIMER	\$3D	タイマレジスタの内容を設定する	○	○	○
READ TIMER	\$3E	タイマレジスタの内容を読み取る	○	○	○
SET RTC	\$41	RTCに値を設定する	○	○	—
READ RTC	\$42	RTCから値を読み取る	○	○	—
KEYBOARD CONTROL	\$45	キーエンコードを制御する	○	○	—

図4-7C キーボード、タイマコマンド一覧

コ マ ン ド 名	コード	内 容	タ イ プ		
			A	B	C
CONTINUE	\$64	データの継続転送を行なう	○	○	○
TEST	\$3F	サブシステム内を絶対アドレスにて参照	○	○	○
CALL MACHINE	\$7F	共有RAM内のユーザプログラムを実行	○	○	—
INIT CONTROL	\$14	サブシステムの切り替え時のパラメータを引き継ぐかの選択	○	○	—
DIZITIZE	\$43	デジタイズを行なう	—	○	—
TELEVISION CONTROL	\$44	AVTVを制御する	○	○	—

図4-7D その他コマンド一覧

ンドと比べるとかなり低速です。ですから特に不具合がないならば、LINE コマンドを使用する方が賢明です。なお F-BASIC V3.3 の LINE 文は、LINE コマンドの方を採用しています。

なお参考までに、2 点 (0, 0), (639, 199) を結ぶ直線のロジックの違いを図 4-8 に示します。

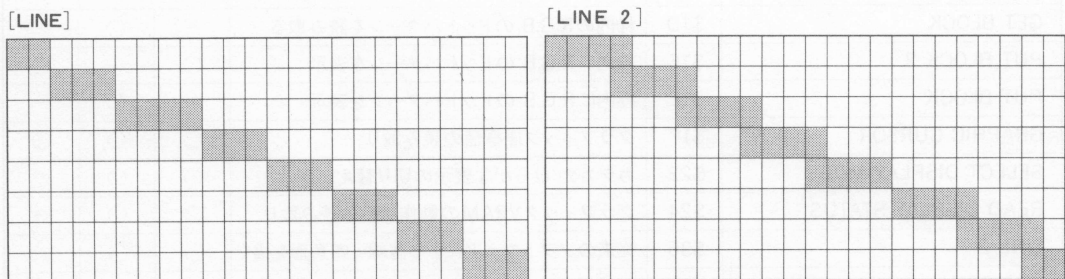


図4-8 LINEとLINE 2 コマンドの直線補間ロジック

4-5 サブシステムの直接制御

4-5-1 共有 RAM アクセス

BASIC あるいは、BIOS によってサブシステムを利用するときには、BIOS がメイン・サブインターフェース処理を行なってくれるため、ユーザーはメイン・サブインターフェースを意識する必要がありません。しかし BASIC や BIOS を通してのサブシステムの利用では実行速度の点で物足りず、直接サブシステムを制御したくなることがあります。その場合には所定のメイン・サブインターフェース処理をユーザープログラムで実行し、共有 RAM をアクセスする必要があります。そこで本項では、メイン CPU による共有 RAM アクセスの手順を説明します。

共有 RAM をメイン CPU がアクセスするためには、まずサブ CPU を HALT(停止)させる必要があります。サブ CPU の HALT には、\$FD05 のビット 7 に 1 を書き込みます。しかしサブ CPU の動作中(BUSY 状態)に急に HALT すると不具合が生じる可能性があるため、BUSY フラグをチェックして、READY 状態を確認してから HALT 信号をセットします。HALT 要求がサブ CPU に受け付けられると、BUSY フラグが BUSY となりますので、その信号によりサブ CPU の HALT を確認します。

サブ CPU が HALT したら、メイン CPU から共有 RAM へコマンドおよびデータを転送します。そしてサブ CPU の HALT を解除すると共有 RAM に設定されたコマンドがサブ CPU により解析されて実行されます。サブ CPU の HALT 解除には、\$FD05 のビット 7 に 0 を書き込みます。

サブ CPU の HALT 解除をすると共有 RAM 上のコマンドが実行されるのですが、実行すべきコマンドがない場合があります。そのときには、共有 RAM の READY フラグを ON にしてから HALT 解除します。

それでは共有 RAM アクセスのサンプルとして、LINE コマンドを共有 RAM に書き込んで実行させるプログラムをリスト 4-2 に示します。

リスト 4-2 共有 RAM アクセス

00100				*****			
00110				* キョウリウRAM ACCESS *			
00120				* (LIST 4-2) V3.0/V3.3 *			
00130				*****			
00140					OPT	NOGEN	
00150	5000				ORG	\$5000	
00160				*			
00170	5000	8D	500A	ENTRY	JSR	SUBHLT	SUB HALT
00180	5003	8D	5023		JSR	DATMOV	DATA MOVE
00190	5006	8D	501C		JSR	SUBMOV	SUB MOVE
00200	5009	39			RTS		
00210				*			
00220	500A	86	FD05	SUBHLT	LDA	\$FD05	SUB BUSY CHECK
00230	500D	28	FB		BMI	SUBHLT	
00240	500F	1A	50		ORCC	#\$50	IRQ.FIRQ キンシ
00250	5011	86	80		LDA	#\$80	SUB HALT REQUEST

```

00260 5013 B7 FD05 STA $FD05
00270 5016 B6 FD05 LDA $FD05 SUB HALT CHECK
00280 5019 2A FB 5016 BPL *-3
00290 501B 39 RTS
00300 501C 4F SUBMOV CLRA SUB HALT カイジ"ヨ
00310 501D B7 FD05 STA $FD05
00320 5020 1C AF ANDCC #$AF IRQ,FIRQ キンシ カイジ"ヨ
00330 5022 39 RTS
00340
00350 5023 8E 5034 DATMOV LDX #LINE
00360 5026 108E FC82 LDY #$FC82
00370 502A A6 80 LOOP LDA ,X+
00380 502C A7 A0 STA ,Y+
00390 502E 8C 5042 CMPX #LINEND
00400 5031 26 F7 502A BNE LOOP
00410 5033 39 RTS
00420
00430 5034 15 LINE FCB $15 COMMAND CODE
00440 5035 07 FCB 7 COLOR
00450 5036 00 FCB 0 FUNCTION
00460 5037 0000 FDB 0 X1
00470 5039 0000 FDB 0 Y1
00480 503B 027F FDB 639 X2
00490 503D 00C7 FDB 199 Y2
00500 503F 00 FCB 0 BOX FLG
00510 5040 FFFF FDB $FFFF LINE STYLE
00520 5042 LINEND EQU *
00530 *
00540 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5041
PROGRAM ENTRY ADDR=5000

```

4-5-2 TEST コマンド

前項では、LINE コマンドを共有 RAM に書き込んで実行させてみました。しかしこれはあまり面白い例ではありませんでした。なぜなら BIOS の SUBOUT のかわりをさせたただだからです。

そこで今度はユーザープログラムを送り込んで、そのプログラムをサブ CPU に実行させることを考えてみたいと思います。

それを行なうためには、TEST コマンドをしっかりと理解する必要があります。この TEST コマンドはサブシステムの開発およびメンテナンスのために作られたコマンドで、FM-8 では YAMAUCHI というパスワードを必要としたため、YAMAUCHI コマンドとして有名なものです。ただし FM-7 以後ではこのパスワードは不要となっています。

この TEST コマンドには、MOVE、JMP、JMSR、END の4つのサブコマンドがあります。MOVE サブコマンドは、指定されたサブメモリ間のデータ転送命令です。JMP サブコマンドは、サブ CPU によって読み込まれるサブコマンド列のアドレスを変更する命令です。サブ CPU の実行番地が分岐されるものではありません。JMSR サブコマンドは、指定されたアドレスのマシン語

サブルーチンをサブルーチンコールします。END コマンドはサブコマンド列の終了を示し、TEST コマンドの実行が終了します。これら TEST コマンドの形式を図 4-9 に示します。

〔TESTコマンド〕

相対値	名 称	値
0, 1	———	———
2	コマンドコード	\$3F
3～10	パスワード	
11～	サブコマンド・列	

〔ENDサブコマンド〕

相対値	名 称	値
0	サブコマンド	\$90

〔MOVEサブコマンド〕

相対値	名 称	値
0	サブコマンド	\$91
1, 2	転送元アドレス	
3, 4	転送先アドレス	
5, 6	転送バイト数	

〔JMPサブコマンド〕

相対値	名 称	値
0	サブコマンド	\$92
1, 2	アドレス	

〔JMSRコマンド〕

相対値	名 称	値
0	サブコマンド	\$93
1, 2	アドレス	

図4-9 TESTコマンドの形式

それではこの TEST コマンドを使った例として、サブシステムの割り込みベクトルの読み込み (リスト 4-3) と画面のハードウェアスクロールのプログラム (リスト 4-4) を考えてみます。

画面のハードウェアスクロールをするには、サブシステムのオフセットアドレスレジスタ (\$D40E, \$D40F) を書き換えるだけで簡単にできます。そしてオフセットアドレスレジスタに書き込んだ値は、\$D008, \$D009 番地 (タイプ C では \$D01F, \$D020 番地) のシステム変数に保存されています。ですからその値を参照して 160 づつ減算した値をオフセットレジスタに書き込むことにします。これによって 2 ラインごとの下スクロールが実現できます。

リスト 4-3 割り込みベクトルの読み込み

```

00100
00110
00120
00130
00140
00150 5000
00160
00170 5000 BD 5016
00180 5003 BD 5038
00190 5006 BD 5031
00200 5009 BD 5016

*****
* SUB INTERRUPT VECTOR READ *
* ( LIST 4-3 ) V3.0/V3.3 *
*****
OPT NOGEN
ORG $5000
*
ENTRY JSR SUBHLT ヲフ" HALT
JSR MOVCMO TEST COMMAND SET
JSR SUBMOV ヲフ" HALT カイシ"ヨ
JSR SUBHLT ヲフ" HALT

```

第4章 サブシステム

00210	500C	8D	5049		JSR	MOVDAT	テ-タ READ
00220	500F	8D	5028		JSR	RDYREQ	サ" READY REQUEST
00230	5012	8D	5031		JSR	SUBMOV	サ" HALT カイシ"ヨ
00240	5015	39			RTS		
00250				*			
00260	5016	86	FD05		SUBHLT	LDA	\$FD05 サ" BUSY CHECK
00270	5019	2B	FB	5016		BMI	*-3
00280	501B	1A	50			ORCC	#\$50 FIRQ.IRQ キンシ
00290	501D	86	80			LDA	#\$80 サ" HALT
00300	501F	87	FD05			STA	\$FD05
00310	5022	86	FD05			LDA	\$FD05 サ" HALT CHECK
00320	5025	2A	FB	5022		BPL	*-3
00330	5027	39				RTS	
00340	5028	86	FC80		RDYREQ	LDA	\$FC80 サ" READY REQUEST
00350	5028	8A	80			DRA	#\$80
00360	502D	87	FC80			STA	\$FC80
00370	5030	39				RTS	
00380	5031	4F			SUBMOV	CLRA	サ" HALT カイシ"ヨ
00390	5032	87	FD05			STA	\$FD05
00400	5035	1C	AF			ANDCC	#\$AF FIRQ.IRQ キョカ
00410	5037	39				RTS	
00420				*			
00430	5038	8E	505A		MOVCMO	LDX	#TESTCD TEST COMMAND SET
00440	503B	108E	FC82			LDY	#\$FC82
00450	503F	A6	80		LOOP1	LDA	.X+
00460	5041	A7	A0			STA	.Y+
00470	5043	8C	506B			CMPL	#TESTED
00480	5046	26	F7	503F		BNE	LOOP1
00490	5048	39				RTS	
00500	5049	8E	FCA0		MOVDAT	LDX	#\$FCA0 テ-タ READ
00510	504C	108E	6000			LDY	#\$6000
00520	5050	C6	10			LDB	#16
00530	5052	A6	80		LOOP2	LDA	.X+
00540	5054	A7	A0			STA	.Y+
00550	5056	5A				DECB	
00560	5057	26	F9	5052		BNE	LOOP2
00570	5059	39				RTS	
00580				*			
00590	505A	3F			TESTCD	FCB	\$3F TEST COMMAND
00600	505B	000B				RMB	8 PASS WORD
00610	5063	91				FCB	\$91 MOVE SUB-COMMAND
00620	5064	FFF0				FDB	\$\$\$F0 テンソクモト アト"レス
00630	5066	D3A0				FDB	\$D3A0 テンソクサキ アト"レス
00640	5068	0016				FDB	\$0016 テンソク ハイストウ
00650	506A	90				FCB	\$90 END SUB-COMMAND
00660		506B			TESTED	EQU	*
00670		5000			END	END	ENTRY

PAGE 002 (860623,115350)

TOTAL ERRORS 00000--00000

TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000

PROGRAM END ADDR=506A

PROGRAM ENTRY ADDR=5000

リスト 4-4 ハードウェアスクロール

```

00100
00110 *****
00120 *   SCROLL   *
00130 * ( LIST 4-4 )   V3.0 *
00140 *****
00150          OPT      NOGEN
00160          ORG      $5000
00170 5000 8D      500A  ENTRY JSR      SUBHLT
00180 5003 8D      502C      JSR      MOVCMO
00190 5006 8D      5025      JSR      SUBMOV
00200 5009 39      RTS
00210 *
00220 500A 86      FD05  SUBHLT LDA      $FD05
00230 500D 2B      FB    500A      BMI      *-3
00240 500F 1A      50      ORCC     #$50
00250 5011 86      80      LDA      #$80
00260 5013 87      FD05      STA      $FD05
00270 5016 86      FD05      LDA      $FD05
00280 5019 2A      FB    5016      BPL      *-3
00290 501B 39      RTS
00300 501C 86      FC80  ROYREQ LDA      $FC80
00310 501F 8A      80      ORA      #$80
00320 5021 87      FC80      STA      $FC80
00330 5024 39      RTS
00340 5025 4F      SUBMOV CLRA
00350 5026 87      FD05      STA      $FD05
00360 5029 1C      AF      ANDCC    #$AF
00370 502B 39      RTS
00380 *
00390 502C 8E      503D  MOVCMO LDX      #TESTCD
00400 502F 108E    FC82      LDY      #$FC82
00410 5033 A6      80      LOOP1  LDA      ,X+
00420 5035 A7      A0      STA      ,Y+
00430 5037 8C      5057      CMPX    #ENDCD
00440 503A 26      F7    5033      BNE     LOOP1
00450 503C 39      RTS
00460 *
00470 503D      3F      TESTCD FCB      $3F
00480 503E      0008    RMB      8
00490 5046      93      FCB      $93
00500 5047      D38F    FDB      $D38F
00510 5049      90      FCB      $90
00520 *
00530 504A FC      D01F      LDD      $D01F  OFFSET ADR REG WORK
00540 504D 83      00A0      SUBD     #160
00550 5050 FD      D01F      STD      $D01F
00560 5053 FD      D40E      STD      $D40E  OFFSET ADR REG
00570 5056 39      RTS
00580      5057  ENDCD  EQU      *
00590 *
00600      5000  END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5056
PROGRAM ENTRY ADDR=5000

```


4-5-3 プログラムの転送

共有 RAM へプログラムを書き込んでサブ CPU に実行させる方法は、わかっていただけたと思います。しかし共有 RAM は 128 バイトしかありません。これではちょっとしたプログラムでもメモリオーバーとなってしまいます。そこでサブ CPU 空間の共有 RAM 以外の領域を使用することを考えてみます。しかしサブシステムのメモリマップを見ていただければおわかりだと思うのですが、サブ CPU 空間はほとんどすべてが使用されています。ですからその CPU 空間を使用するときには、制約に注意して使用する必要があります。使用できそうなサブ CPU 空間には、つぎの領域が考えられます。

- ① 共有 RAM (\$D380～\$D3FF 番地, 128 バイト)
- ② ワークエリア (\$D500～\$D7FF 番地, 768 バイト)
- ③ 補助ワークエリア (\$D0C7～\$D277, 569 バイト)
- ④ コンソールバッファ (\$C000～\$CF9F 番地, 4000 バイト)
- ⑤ VRAM

ただし、アドレスと大きさはタイプ A の場合の値です。

共有 RAM は無条件で使用できます。しかもサブモニタの実行によってその内容が変更されることがないため、定常的なプログラムの実行にも利用できます。

ワークエリアは、PAINT ルーチンのワークとして使用される領域です。サブモニタ ROM 内の PAINT ルーチンを使用しないなら、このエリアはすべて使用できます。また内容が変更されることもありません。しかも後述する SPECIAL コマンドを使えば、この領域をユーザー領域として占有することも可能です。しかし FM-7 では、この領域に RAM が実装されていないので使用できません。

補助ワークエリアは、サブモニタ ROM 内の各処理のローカル変数に利用される領域です。ですからサブモニタ ROM 内ルーチンを使用しないなら、この領域も使用できます。ただしサブモニタ ROM 内ルーチンが動作したときにはこの領域の値は保証されないため、定常的なプログラムの実行には不適当なエリアです。

コンソールバッファは、キャラクタ表示におけるワークエリアです。ですからこのエリアを使用するときには、コンソールバッファをアクセスするサブモニタ ROM 内ルーチンはすべて使用できません。またプログラムの終了時点でコンソールバッファの初期設定をしないと、画面が乱れてしまいます。ただしタイプ A、タイプ B 使用時、40 字モードにすればキャラクタコードバッファ、アトリビュートバッファの後半 1000 バイトは使用されないため、自由に使うことができます。

VRAM は大きな領域ですが、CRTC と共通に使用されること、スクロールによって論理アドレスが変わってしまうこと等、使用に際して問題の多い領域です。ですから VRAM をユーザーマシン語領域として使用するのには、やめた方がいいと思われます。

使用する領域が決まれば、プログラムの転送、実行は TEST コマンドによって容易にできます。
 それではコンソールバッファを使用したサンプルプログラムをリスト 4-5 に示します。

リスト 4-5 プログラムの転送

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 50 71 10 00 50 09 00 53 00 00 3F 00 00 00 00 : 3A
5010 : 00 00 00 00 91 D3 93 00 00 00 40 90 00 00 00 00 : C7
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 10 00 50 62 : C2
5060 : 00 0F 00 00 3F 00 00 00 00 00 00 00 93 C0 00 : A1
5070 : 90 8E 50 A3 CE C0 00 FF 50 17 CE 50 1C C6 40 A6 : EB
5080 : 80 A7 C0 SA 26 F9 34 10 8E 50 03 AD 9F FB FA FE : C4
5090 : 50 17 33 C8 40 35 10 8C 52 18 25 DB 8E 50 5C 6E : 88
50A0 : 9F FB FA 4F 5F B7 C1 74 B7 C1 72 FD C1 76 86 01 : D3
50B0 : B7 C1 75 86 02 B7 C1 73 8D 46 B6 C1 74 8B C1 75 : 0F
50C0 : 81 4C 25 0A B6 C1 75 40 B7 C1 75 B8 C1 74 B7 C1 : 7D
50D0 : 74 B6 C1 72 B8 C1 73 81 B8 25 0A B6 C1 73 40 B7 : 95
50E0 : C1 73 B8 C1 72 B7 C1 72 8D 37 8E 0B B8 30 1F 26 : 96
50F0 : FC FC C1 76 C3 00 01 FD C1 76 83 03 E8 25 B9 39 : AC

[cs] : E6 D8 85 5D 0B B8 0C B2 84 1C EE E4 B0 11 BC C1 : D1

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 8D 45 B6 D4 09 CE 00 00 C6 03 86 10 EF 81 EF 81 : 72
5110 : 30 88 4C 4A 26 FE 30 89 38 00 SA 26 ED B7 D4 09 : 5F
5120 : 39 8D 24 B6 D4 09 10 8E C0 B2 C6 03 86 10 EE A1 : 7B
5130 : EF 81 EE A1 EF 81 30 88 4C 4A 26 F2 30 89 38 00 : C9
5140 : SA 26 E9 B7 D4 09 39 B6 C1 72 C6 50 3D FB C1 74 : A2
5150 : 89 00 1F 01 39 00 00 00 00 07 FF FF E0 0F FF FF : D4
5160 : F0 1F FF FF FB 3F FF FF FC 3F FF FF FC 3F FF FF : B4
5170 : FC 3F FF FF FC 3F FF FF FC 3F FF FF FC 3F FF FF : E4
5180 : FC 3F FF FF FC 1F FF FF FB 0F FF FF F0 07 FF FF : 4C
5190 : E0 00 00 00 00 0F FF FF F0 18 00 00 18 30 00 00 : 3D
51A0 : 0C 67 FE C0 36 C3 02 E0 73 C3 00 F0 F3 C3 10 09 : D1
51B0 : B3 C3 F0 CF 33 C3 10 C6 33 C3 00 C0 33 C3 00 C0 : 6D
51C0 : 33 C3 00 C0 33 67 80 C0 36 30 00 00 0C 18 00 00 : 1A
51D0 : 18 0F FF FF F0 00 00 00 00 00 00 00 00 00 00 : 15
51E0 : 00 07 FE C0 30 03 02 E0 70 03 00 F0 F0 03 10 09 : 19
51F0 : B0 03 F0 CF 30 03 10 C6 30 03 00 C0 30 03 00 C0 : 61

[cs] : 4A A4 F4 07 DB F6 49 5D 2A D9 8E D7 01 34 C9 CD : 93

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5200 : 30 03 00 C0 30 07 80 C0 30 00 00 00 00 00 00 00 : 9A
5210 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5220 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5230 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5240 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5250 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5260 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5270 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5280 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5290 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

[cs] : 30 03 00 C0 30 07 80 C0 30 00 00 00 00 00 00 00 : 9A

```

SAVEM "L4-SM", &H5000, &H521A, &H5000

FM77AV の F-BASIC V3.3 を使用するときには、ダイレクトアクセスによってサブ CPU 空間をアクセスする方が簡単ようです。もちろんダイレクトアクセスは MMR を使用するため、CPU クロックが 1.6MHz に落ちてしまいます。しかしメイン・サブインターフェースで大量のデータ転送をしなければならない場合には、かえってダイレクトアクセスの方が効率的なようです。ダイレクトアクセスに関しては、「13-3 ダイレクトアクセス」を参照ください。

4-6 コンソールバッファ

画面に表示される文字は、その文字情報がまずコンソールバッファに格納されます。そしてそのコンソールバッファの内容をもとにキャラクタフォントが読み出され、VRAM に展開され表示されます。このコンソールバッファと VRAM との対応を、80 文字 25 行 (WIDTH 80, 25) 表示の場合について図 4-10 に示します。

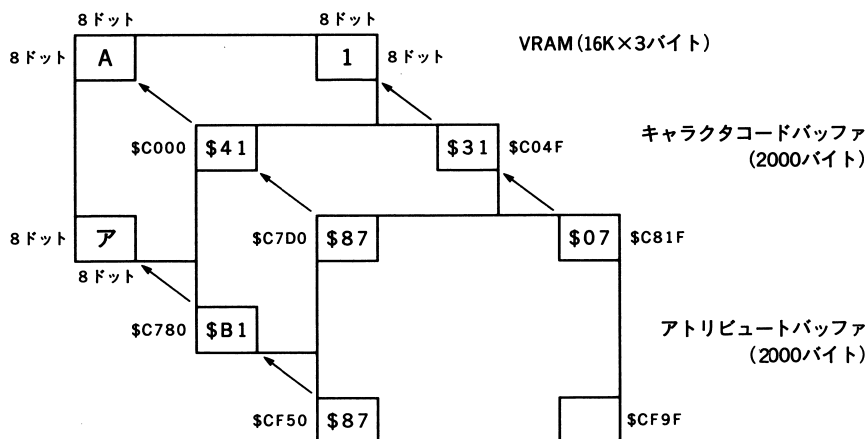


図4-10 VRAMとコンソールバッファの対応

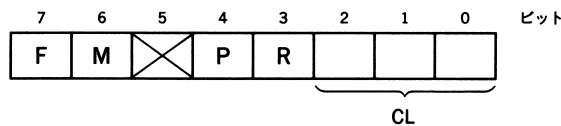
40 文字表示の場合は、VRAM の文字フォントの横が 16 ドットに拡大表示されます。そしてキャラクタコードバッファ、アトリビュートバッファともに前半の 1000 バイトのみが使用されます。20 行表示の場合には、文字と文字との間に 2 ドットのすき間がおかれ、VRAM の文字フォントの縦が 10 ドットとなります。

キャラクタコードバッファには、表示文字のアスキーコードがそのまま格納されます。それに対してアトリビュートバッファには、表示文字のアトリビュート属性が格納されます。このアトリビュート属性の意味は、図 4-11 に示すとおりです。

このコンソールバッファの内容は、BASIC の SCREEN 関数にて知ることができます。

PRINT SCREEN(0,0,0) ……キャラクタコードバッファの内容

PRINT SCREEN(0,0,1) ……アトリビュートバッファの内容



F : フィールドスタートフラグ
M : モディファイフラグ
P : フィールド保護フラグ
R : リバース表示フラグ
CL : 表示色(カラーコード)

図4-11 アトリビュート属性

ただしアトリビュートバッファの内容は、CL(表示色)とR(リバース表示フラグ)のみが返されます。

P(フィールド保護フラグ)がONの文字は、スクリーンエディットによるキーボードの文字入力に対して保護され、変更できなくなります。しかしPRINT文での変更は可能です。これはBASICではサポートされていませんが、\$61D番地(V3.0では\$1E5番地)の内容がアトリビュートバッファに書き込まれるようになっていきますので、この内容を書き換えてやればBASICでも活用できます。次のプログラムを実行してみて、“DON'T MODIFY”という文字列を変更してみてください。変更しようとするときブザーが鳴って変更できません。

```
10 POKE &H61D,&H12: PRINT "DON'T MODIFY";
20 POKE &H61D,&H06: PRINT "MODIFY OK"
```

M(モディファイフラグ)は、スクリーンエディットにおいて変更された文字に対してONとなります。これは、サブシステムがスクリーンエディットによって変更された内容の処理のために使われる内部フラグです。スクリーンエディット状態になったときモディファイフラグがすべてOFFにされ、変更された文字に対してのみONにされます。ですからアトリビュートバッファを直接アクセスすれば、ユーザープログラムにおいて変更された文字を知ることができます。

F(フィールドスタートフラグ)はフィールドの開始を示すフラグで、フィールドの先頭だけがONにセットされます。FM-7シリーズでは、画面(コンソール画面)はフィールドと呼ばれる単位に分割され管理されています。このフィールドが文字列の表示、入力処理の単位となります。ですからスクリーンエディットにおいて文字を入力していった、次のフィールドにかかるとブザーが鳴って警告されます。インサートモードのときには、入力処理中のフィールド内がいっぱいになる(文字コードが\$00の領域がなくなる)とそれ以上の文字の挿入ができなくなってしまいます。BASICプログラムをスクリーンエディットで修正していると、ピープ音が鳴って修正できなくなることがありますが、それはこれが原因なのです。

このフィールドスタートフラグは、画面を消去したときには画面の左上端にだけセットされます。つまり画面全体がひとつのフィールドなわけです。そしてPRINT文などで文字列が表示されると、その表示文字列の先頭にフィールドスタートフラグがセットされ、新たなフィールドが

設定されます。

新たなフィールドが設定されると、そのフィールド内(次のフィールドスタートフラグがセットされているところまで)のアトリビュートバッファは同一のアトリビュートによって埋められます。BASIC では、そのアトリビュートが\$61D 番地(V3.0 では\$1E5 番地)の内容だったわけです。それでは次のプログラムを実行してみてください。

```
CLS: LOCATE 10,10: POKE &H61D,&H12: PRINT "A" 
```

表示された“A”という文字以降はすべて保護フィールドとなり、キーボードから文字入力を行うことができなくなってしまいます。ここで  などを入力しようものなら、画面全体が保護フィールドとなってまったくキーボードを受け付けなくなってしまいます。

もうひとつ、フィールドに関する面白いサンプルを示します。次のプログラムを実行してみてください。そしてカーソルを表示されている文字の上へ持って行ってください。

```
10 CLS
20 COLOR 2: LOCATE 10,10: PRINT "ABCDE"
30 COLOR 1: LOCATE 7,10: PRINT "1234"
```

どうですか。“BCDE”という文字の上にカーソルを持っていくと色が変わってしまいましたね。これは“1234”の表示が“ABCDE”のフィールドに重なってしまい、“ABCDE”のフィールドのフィールドスタートフラグが書き換えられてしまい、1つのフィールドになってしまったためなのです。そのため“BCDE”に対するアトリビュートが書き換わってしまい、カーソルを持ったときその新しいアトリビュートで表示し直されたからです。

それでは最後にフィールドの働きを示すプログラムを実行してみます。

```
10 CLS
20 LOCATE 0,5: PRINT"      A1      A2"
30 LOCATE 0,6: PRINT"      A3";: PRINT"      A4"
```

これはひとつのプリント文で表示するか、分割して表示するかの違いだけであまり違いはないように思えます。しかし A1, A2 は同じフィールド内の文字なのですが、A3, A4 は異なるフィールドの文字なのです。サブシステムの扱いはフィールド単位なので、この両者は異なる扱いを受けます。たとえば A1, A2 の前にカーソルを持っていき、SAVE”を入力して  キーを押してみてください。エラーとなりますね。ところが A3, A4 に同様のことを行なうと、SAVE コマンドが正常に実行されます。

コンソールバッファというのは、通常はあまり意識されていませんし、FM-7 シリーズを使用するうえであまり意識する必要のない部分です。しかしそのしくみがわかると、なにげない処理のなかにサブシステムの複雑な処理の一端をかいまみることができて、興味深いと思います。

4-7 ハードウェアスクロール

FM-7 シリーズでは、サブシステムのオフセットレジスタ(\$D40E,\$D40F)に値をセットすることにより、簡単にスクロールを実現することができます。FM-7 ではこのオフセットレジスタの下位 5 ビットの値は無効であり、256 ドット単位のスクロールとなります。これに対して FM77AV では、\$D430 のビット 2 を ON にするとオフセットレジスタの下位 5 ビットも有効となり、8 ドット単位のスクロールが可能となります。

ハードウェアスクロールのしくみは、この点だけを注意すれば FM-7 も FM77AV も同様ですから、13-5 ハードウェアスクロールの項を参照ください。

なおオフセットレジスタに書き込んだ値は、\$D008, \$D009 番地(FM-7 では\$D01F, \$D020 番地)に保存されていますので、その値を参照すればスクロールの開始からスムーズなスクロールが実現できます。

4-8 SPECIAL コマンド

FM77AV のサブモニタ(タイプ A, タイプ B)には、ディスプレイサブシステム解説書に記述されていない SPECIAL コマンドが追加されています。このコマンドの実行には "IKEMOTO" というパスワードを必要としています。FM-8 の TEST コマンド(パスワード "YAMAUCHI" を必要とした)をほうふつとさせますが、この SPECIAL コマンドの機能の有用性は、TEST コマンドには及ばないようです。

富士通の伝統ともいうべきでしょうか。そういえば FM77AV で F-BASIC V3.3 を起動して以下のキーを押すと、キーボードエンコーダからのメッセージを見ることができます。

- ・**[CAP]**, **[INS]**, **[かな]** LED を点灯させる。
- ・左右の **[SHIFT]**, **[CTRL]**, **[GRAPH]**, **[T]** キーを同時に押す。

それでは、この SPECIAL コマンドのコマンド形式と機能を説明します(図 4-13)。SPECIAL コマンドは、以下の 5 つのサブコマンドからなっています。

STORE KEY(サブコマンドコード=1)
 GET DOMAIN(サブコマンドコード=2)
 RELEASE DOMAIN(サブコマンドコード=3)
 DEFINE EXTENTION COMMAND(サブコマンドコード=4)
 DELETE EXTENTION COMMAND(サブコマンドコード=5)

STORE KEY コマンドは、指定されたコードをキー入力バッファに格納するものです。この機能を使えば、キーボードが押されたかのような処理をプログラムで実現できます。

GET DOMAIN コマンドは、サブシステムのワークエリア(\$D500~\$D7FF)にユーザー領域を確保するものです。ちょうど BASIC の CLEAR 文にてユーザーマシン語領域を確保するのと似ています。ただしこのサブコマンドで指定するパラメータは、新たに確保すべきユーザー領域の大きさ(バイト)です。

RELEASE DOMAIN コマンドは、GET DOMAIN コマンドにて確保したユーザー領域をサ

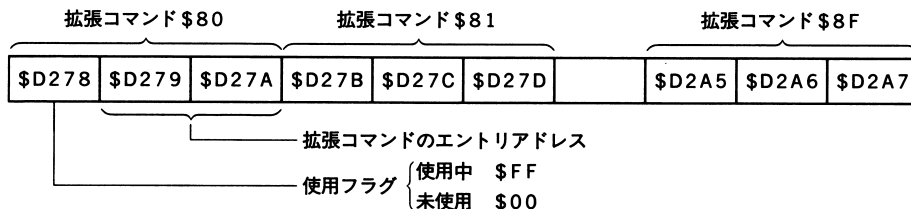


図4-12 拡張コマンドテーブル

〔SPECIALコマンド〕

相対値	名 称	内 容
0, 1	—	—
2	コマンドコード	\$4F
3~9	パスワード	IKEMOTO
10	サブコマンドコード	1~5
11~	パラメータ	

〔STORE KEY〕

相対値	名 称	内 容
10	サブコマンドコード	1
11	キーコード	アスキーコード

〔GET DOMAIN〕入力情報

相対値	名 称	内 容
10	サブコマンドコード	2
11, 12	バイト数	1~768

復帰情報

相対値	名 称	内 容
3, 4	TOPアドレス	ユーザー領域のTOPアドレス
5, 6	BOTTOMアドレス	ユーザー領域のBOTTOMアドレス

〔RELEASE DOMAIN〕入力情報

相対値	名 称	内 容
10	サブコマンドコード	3
11, 12	バイト数	1~768

復帰情報

相対値	名 称	内 容
3, 4	TOPアドレス(処理前)	ユーザー領域TOPアドレス
5, 6	TOPアドレス(処理後)	ユーザー領域TOPアドレス

〔DEFINE EXTENTION COMMAND〕

相対値	名 称	内 容
10	サブコマンドコード	4
11	拡張コマンドコード	\$80~\$8F
12, 13	エントリアドレス	

〔DELETE EXTENTION COMMAND〕

相対値	名 称	内 容
10	サブコマンドコード	5
11	拡張コマンドコード	\$80~\$8F

図4-13 SPECIALコマンドの形式

ブシステムに解放するものです。解放するユーザー領域の大きさをパラメータとして与えます。

DEFINE EXTENTION COMMAND コマンドは、サブシステムの拡張コマンドを拡張コマンドテーブル(\$D278~\$D2A7)に登録するものです。登録された拡張コマンドは、通常のサブシステムのコマンドと同様に BIOS の SUBIN, SUBOUT コマンドからでも、利用できるようになります。拡張コマンドテーブルの構造は図 4-12 のようで、テーブルの前の方から\$80~\$8F の拡張コマンドに対応します。このサブコマンドで与えるパラメータは、登録する拡張コマンドコード(\$80~\$8F)とその拡張コマンドのエントリアドレスです。なおこの拡張コマンドとして登録されるルーチンは RTS で終了する必要があります。

DELETE EXTENTION COMMAND コマンドは、登録済みの拡張コマンドを拡張コマンドテーブルより削除するものです。削除したい拡張コマンドコード(\$80~\$8F)をパラメータとして指定します。

4-9 サブシステムへの PEEK, POKE

サブシステムにアクセスする場合、いちいち TEST コマンドを使っていたのでは、わずらわしくて大変です。そこで BASIC から手軽にサブシステムを扱うために、次のユーティリティを作成しました。動作する F-BASIC は V3.0 のみです。

- ① サブシステムへ POKE (リスト 4-6-1)
- ② サブシステムを PEEK (リスト 4-6-2)
- ③ サブシステムを SAVEM (リスト 4-6-3)
- ④ サブシステムへ LOADM (リスト 4-6-4)

①と②はサブシステムの I/O レジスタを操作するのに便利でしょう。また、③と④は、VRAM 全体をフロッピーにセーブ/ロードするのに利用できます。使用方法はプログラムを適当なアドレスにロードして、EXEC によりサブルーチンコールします。

[POKE]

EXEC ロードアドレス 書き込みサブアドレス, 書き込みデータ

[PEEK]

DEFUSR=ロードアドレス

変数=USR(読み込みサブアドレス)

[SAVEM]

EXEC ロードアドレス ファイル名, セーブ開始サブアドレス, 終了アドレス, 実行開始アドレス

[LOADM]

EXEC ロードアドレス ファイル名 [, [オフセット] [,R]]

たとえば、青の VRAM 全体を "B1" というファイル名でセーブするときには、次のようになります。

EXEC &H7000 "B1", &H0, &H3E7F, &H0

リスト 4-6-1 サブシステムへ POKE

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : BD 99 F9 33 8C 13 AF 43 E7 45 30 8C 06 EF 02 6E : 60
7110 : 9F FB FA 10 00 00 00 00 1D 00 00 3F 00 00 00 00 : 00
7120 : 00 00 00 00 93 D3 8F 90 B6 D3 85 F6 04 09 A7 9F : AC
7130 : D3 83 F7 04 09 39 00 00 00 00 00 00 00 00 00 00 : 63
7140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 2F 17 EA 17 2B 1F 3E D3 BA 18 B5 C1 DA F8 A9 0D : 6F

```

SAVEM "L4-6-1M", &H7100, &H7135, &H7100

リスト 4-6-2 サブシステムを PEEK

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : BD 9A 05 8D 05 8D 15 7E 97 4C 34 10 33 8C 30 AF : D3
7010 : 43 30 8C 25 EF 02 AD 9F FB FA 35 90 B6 FD 05 2B : FE
7020 : FB 1A 50 86 80 87 FD 05 B6 FD 05 2A FB F6 FC 85 : 78
7030 : 73 FC 86 7F FD 05 1C AF 39 10 00 00 00 00 2A 00 : B4
7040 : 00 3F 00 00 00 00 00 00 00 00 93 D3 8F 90 BE D3 : 55
7050 : 83 F6 04 09 A6 84 F7 04 09 5F FD D3 85 F6 04 0A : DC
7060 : F6 D3 86 27 FB F7 04 0A 39 00 00 00 00 00 00 00 : 7F
7070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
7090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
70F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : E7 E8 C1 E7 12 C6 A6 AF C3 B2 FE 70 FB 05 ED 3C : AD

```

SAVEM "L4-6-2M", &H7000, &H7068, &H7000

第4章 サブシステム

71E0	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00	
71F0	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00	

[cs]	:	04	09	C3	F2	85	73	2E	04	09	AD	9F	D3	83	00	00	00	:	67

SAVEM "L4-6-4M",&H7000,&H711C,&H7000

キー入力, タイマー

第 5 章

5-1 キーボード入力

5-1-1 FM-7 のキー入力

FM-7 以外のユーザーが FM-7 のゲームをプレイしたとき、まずとまどうのが、FM-7 独特のキー入力のくせなのではないでしょうか。FM-7 では、動かしているキャラクタを止めようとしてテン・キーから手を離しても止まってくれないのです。動いているキャラクタの動きを止めるには、"5" のキーを押さなければならないのです。

FM ユーザーのあなたにとっては、もうっとくに慣れてしまったことだと思いますが、その原因について考えてみたいと思います。

PC-8801 等では、キーボードの各キーが押されているかどうかの情報が、I/O ポートを読むことによって常時知ることができます。ですから現在キーが押されているかどうか、さらには同時に押された複数のキーコードさえも知ることができます。

しかし FM-7 では、状況がまったく異なります。

キーが入力されると、キーボードインターフェース回路(4 ビット LSI MB88401)がキーボードからの情報をデコードして、FIRQ 割り込みを発生させます。そして FIRQ 割り込みを受け付けたサブ CPU が、デコードされたキーデータを受け取りキー入力バッファに格納します。

一方、メイン CPU はキー入力コマンドをサブ CPU に送り、キー入力バッファの先頭データを受け取ることになります。

キー入力バッファの大きさは 32 バイトですが、通常は 1 バイトだけが、使用されています。ですから実際には、キー入力コマンドによってメイン CPU は、常に、最後に押されたキーコードだけを得ることになります。

しかしこれは、あくまでも最後に押されたキーコードであって、現在押されているキーコードではないのです。ですから今、現在キーが押されていないということをメイン CPU (サブ CPU も)は、知ることができないのです。

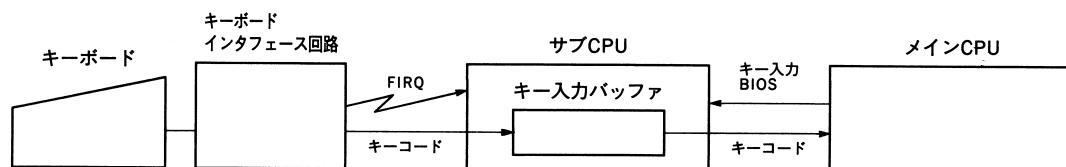


図5-1 FM-7のキー入力概念図

このことが、FM-7において動かしているキャラクタを止めるとき、止めるためのキー(たいていは“5”のキーが使われます)を押さなければならない理由なのです。

FM-7のキー入力の仕組みは、図5-1のとおりです。

FM77AVでは、キーボード情報のデコード方法が3種類に拡張されています。そしてその3種類のキーコード系においてスキャンコードモードを選択すると、PC-8801のようなリアルタイム・キースキャンが可能となっています。詳細は、キーコード系の項を参照ください。

5-1-2 キー入力バッファ

キー入力バッファには、サブCPU空間の\$D360～\$D37Fの32バイトが割り当てられています。そしてヘッドポインタ(\$D005,\$D006)、テイルポインタ(\$D007,\$D008)、バッファカウンタ(\$D004)の3つのワークにより図5-2の様に循環的に使用されています。

ヘッドポインタは、キー入力コマンドに読み取られるキーデータのアドレスを示しています。そして、キーデータが読み取られるとカウントアップされます。

テイルポインタは、キーデータがつぎに格納されるアドレスを示しています。そして、キーデータが格納されるとカウントアップされます。

バッファカウンタは、現在キー入力バッファに格納されているキーデータの数を示します。キーデータが格納されるとカウントアップされ、読み取られるとカウントダウンします。

以上の操作でテイルポインタまたはヘッドポインタがキー入力バッファの末尾にきたら、ポインタの値をキー入力バッファの先頭を指すようにします。これによってキー入力バッファは、ひとつのリングのように循環して使用されます。キー入力バッファがいっぱいときには、入力されたキーは無視されキー入力バッファに格納されません。

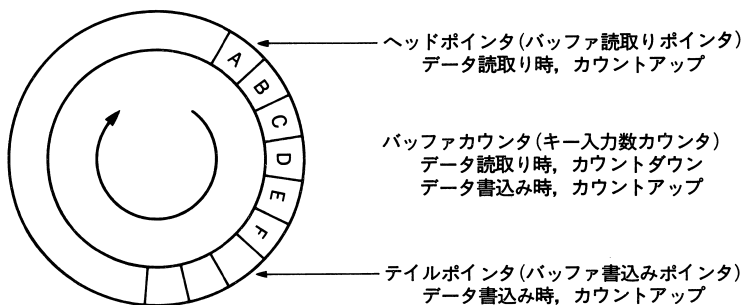


図5-2 キー入力バッファ概念図

キー入力バッファの働きを見てみましょう。まずF-BASIC V3.0を起動して、次のプログラムを実行してみてください(リスト5-1-A)。そしてキーをいくつか入力してから、BREAKキーを押してください。プログラムの実行がBreakされると、最後に押したキーだけが表示されます。

これは、F-BASIC V3.0 がキーの先行入力を禁止していて、キー入力バッファが1バイトだけしか使用されていないからです。

リスト 5-1-A キー入力バッファ

```
10 GOTO 10
RUN
```

```
Break In 10
Ready
5
```

それでは、キーの先行入力を許可して同じことをやってみましょう。キーの先行入力を許可するには、

```
PRINT CHR$(&H1B) + "g" 
```

を実行します。その後で先ほどのプログラムを実行させ、キーを 40 個程押してから BREAK してください(リスト 5-1-B)。

リスト 5-1-B キー入力バッファ

```
10 GOTO 10
PRINT CHR$(&H1B) + "g"
```

```
Ready
RUN
```

```
Break In 10
Ready
12345678901234567890123456789012
```

今度は、32 個のキーが表示されましたね。しかもこれは、最初に押した 32 個のキーで、33 番目以降のキー入力は捨てられています。なおキーの先行入力を元に戻すには、

```
PRINT CHR$(&H1B) + "h" 
```

とします。

それでは、F-BASIC V3.3 では、どうなのでしょう。V3.3 では V3.0 の場合と異なっていて、最初からキーの先行入力が許可されています。キーの先行入力の許可／禁止の方法は、V3.0 と同じですので、V3.3 でも試してみてください。

なお、キーの先行入力を許可しているときにキー入力バッファをクリアするには、

```
PRINT CHR$(&H1B)+`9" 
```

を実行します。

5-1-3 BASIC でのキー入力の比較

BASIC でのキー入力には、4つの命令があります。

INPUT

LINE INPUT

INPUT\$(n)

INKEY\$

そしてそれらには微妙な違いがあり、その使い分けに頭を悩ます場合があります。そこで、BASIC でキー入力のプログラムをするときの参考として、入力やキーセンスの方法を一覧表としてまとめてみました(図 5-3)。

表中のエコーバックとは、入力したキーデータの CRT 表示のことです。

		INPUT	LINE INPUT	INPUT\$()	INKEY\$
		文	文	関 数	関 数
入 力 表 示	プロンプト文	可 能	可 能	不可能	不可能
	プロンプトマーク ?の表示	可 能 表示しなくする事も可能	無	無	無
	カーソル表示	有	有	無	無
	エコーバック	有	有	無	無
	入 力 待 ち	待 っ	待 っ	待 っ	待たない
デ ー タ 入 力	カンマ(,)の入力	ダブルクォート で囲めば可能	可 能	可 能	可 能
	ダブルクォート (") の 入 力	不 可	可 能	可 能	可 能
	コントロールコード カーソルキーの入力	不 可	不 可	可 能	可 能
	複 数 の 変 数 へ の 入 力	可 能	不 可	不 可	不 可
	入 力 文 字 数	255文字以内	255文字以内	指定した文字数 (255文字以内)	1 文字
	入 力 終 了	 キー	 キー	指定した文字数を 入力した時	——
	入力文字なしで  キー	ヌルストリング	ヌルストリング	CHR\$(13)	CHR\$(13)
 BREAK キ	BREAK	可 能	可 能	不 可	——
	CONTによる再開	可 能	可 能	不 可	——

図 5-3 キー入力方法の比較

5-1-4 キーボードに対する BIOS

キーボード 1 文字入力の BIOS(KEYIN)のリクエスト番号は、21 です。パラメータは、キー入力データを格納するデータバッファ先頭アドレスだけです(図 5-4)。

KEYIN を実行すると、指定したデータバッファに入力キーコードとキー入力フラグが返されます。キー入力フラグは、0 のときにキー入力なし、1 のときにキー入力ありを示します。RCB のデータバイト数には、KEYIN によって常に\$0002 がセットされます。

KEYIN を用いて入力したキーコードを逆向きに表示するサンプルプログラムを、リスト 5-2 に示します。\$5000 番地よりプログラムを入力して、EXEC &H5000  で実行してください。

KEYIN(キーボード 1 文字入力)

オフセット	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	21	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数	RCBLNH		○
6, 7	リザーブ	—	—	—

図5-4 KEYINのRCB

リスト 5-2 キーボード 1 文字入力

```

01000                                *****
01010                                * KEYIN SAMPLE *
01020                                * ( LIST 5-2 ) V3.0/V3.3 *
01030                                *****
01040                                OPT NOGEN
01050 5000                        ORG $5000
01060 5000 20      20      5022 ENTRY BRA KEY
01070                                *
01080 5002      15      KEYIN FCB 21      REQ NO.
01090 5003      0001      RMB 1
01100 5004      500A      FDB KEYDT DATA ADR.
01110 5006      0002      RMB 2 DATA LEN.
01120 5008      0002      RMB 2
01130 500A      0002      KEYDT RMB 2 DATA BUFFER
01140 500C      10      SUBOUT FCB 16 REQ NO.
01150 500D      0001      RMB 1
01160 500E      5014      FDB SYMBOL DATA ADR.
01170 5010      000E      FDB 14 DATA LEN.
01180 5012      0002      RMB 2
01190 5014      0002      SYMBOL RMB 2
01200 5016      19      FCB $19 COMMAND CODE
01210 5017      07      FCB 7 COLOR
01220 5018      00      FCB 0 FUNCTION
01230 5019      02      FCB 2 ANGLE
01240 501A      02      FCB 2 CHAR.WIDTH
01250 501B      02      FCB 2 CHAR.HIGH
01260 501C      0270      FDB 624 X
01270 501E      00B8      FDB 184 Y
01280 5020      01      FCB 1 CHAR.NUMBER
01290 5021      0001      STRING RMB 1 STRING
01300                                *
01310 5022 8E      5002      KEY LDX #KEYIN KEY INPUT

```



```

01320 5025 AD 9F FBFA JSR [$FBFA]
01330 5029 25 4A 5075 BCS ERROR
01340 502B B6 500B LDA KEYDT+1
01350 502E 27 F2 5022 BEQ KEY
01360 5030 B6 500A LDA KEYDT
01370 5033 81 00 CMPA #$0D =CR
01380 5035 27 27 505E BEQ CRRTN
01390 5037 B7 5021 STA STRING
01400 503A 8E 500C LDX #SUBOUT CHAR.DISPLAY
01410 503D AD 9F FBFA JSR [$FBFA]
01420 5041 25 32 5075 BCS ERROR
01430 5043 EC 88 10 LDD 16.X
01440 5046 26 0E 5056 BNE KEY01
01450 5048 EC 88 12 LDD 18.X
01460 504B 83 0010 SUBD #16
01470 504E 2B 24 5074 BMI ENDRTN
01480 5050 ED 88 12 STD 18.X
01490 5053 CC 0280 LDD #640
01500 5056 83 0010 KEY01 SUBD #16
01510 5059 ED 88 10 STD 16.X
01520 505C 20 C4 5022 BRA KEY
01530
01540 505E 8E 500C * CRRTN LDX #SUBOUT
01550 5061 CC 0270 LDD #624
01560 5064 ED 88 10 STD 16.X
01570 5067 EC 88 12 LDD 18.X
01580 506A 83 0010 SUBD #16
01590 506D 2B 05 5074 BMI ENDRTN
01600 506F ED 88 12 STD 18.X
01610 5072 20 AE 5022 BRA KEY
01620
01630 5074 39 * ENDRTN RTS
01640 5075 39 * ERROR RTS
01650
01660 5000 * END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5075
PROGRAM ENTRY ADDR=5000

```

5-1-5 キーデータの読み取り

キーボードから押されたデータは、デコードされてサブCPUおよび、メインCPUのI/Oレジスタにも送られています。ですから最後に押されたキーのキーコードを知りたいときには、このレジスタを読み取ってもいいことになります(図5-5)。

このI/Oレジスタの値はキー入力バッファと異なり、最後に押されたキーコードが、別のキーが押されるまでそのまま残っています。英数、カタカナ、グラフィック、記号、カーソル等は、アスキーコードの形で\$FD01(メインCPU)および\$D401(サブCPU)にセットされています。またPFキーを押すと、\$FD00および\$D400のビット7が"1"にセットされ、\$FD01および\$D401に押されたPFキー番号がセットされます。

それではリスト5-3を実行して、いろいろなキーを押して\$FD00, \$FD01の内容を確認してみてください。

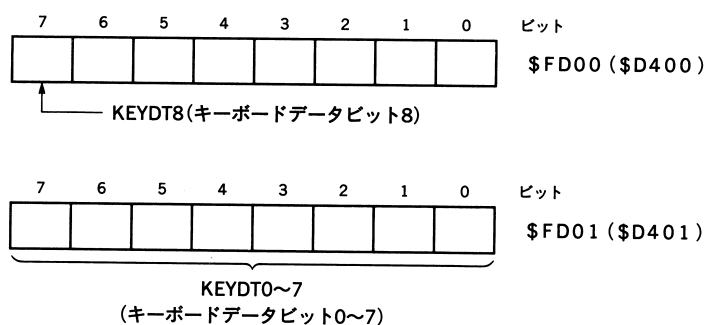


図5-5 キーデータレジスタ

リスト 5-3 キーデータの読み取り

```

10 PRINT "FD00=";HEX$(PEEK(&HFD00) AND &
H80). "FD01=";HEX$(PEEK(&HFD01))
20 GOTO 10

```

5-1-6 エンコーダ機能

FM77AV では、3 個の LSI(MB88551, MB88201, MBL80C49) からなるキーボードエンコーダにより、以下のようなキー入力の拡張機能を実現しています(図 5-6)。

- ① 3 種類のキーコード系によるキーエンコード
- ② スーパーインポーズ、ビデオデジタイズ等の AVTV 制御
- ③ バッテリーバックアップの RTC(Real Time Clock)制御
- ④ コマンドによるエンコーダ制御

AVTV 制御については「第 13 章」、RTC 制御については「5-4 タイマー」を参照ください。

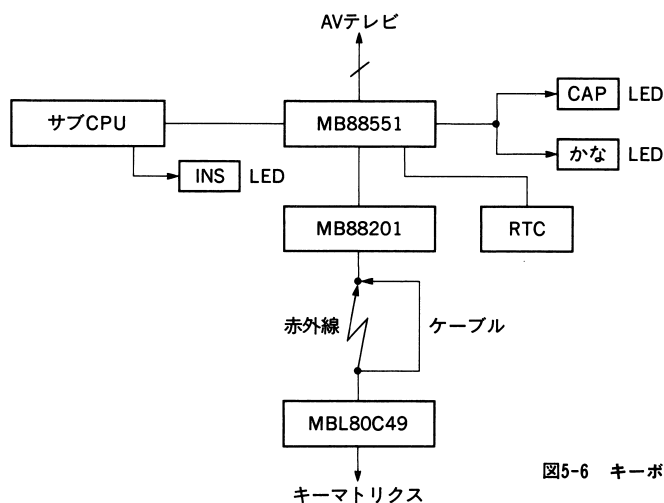


図5-6 キーボードエンコーダブロック図

(1) キーエンコード

FM-7のキーエンコードは、JIS9bitモードだけでした。FM77AVでは、JIS9bitモードの他にFM16 β 準拠モードとスキャンコードモードが選択できるようになっています。

特にこのスキャンコードモードにおいては、離されたキーのキーコード情報も送られるため、本章の冒頭で説明したFM-7独特のキーのくせを解消することができます。

- ① JIS9bit モード
- ② FM16 β 準拠モード
- ③ スキャンコードモード

JIS9bitモードは、FM-7互換のキーコードが発生するモードで、アスキーコードが使われています。ただし、PFキーが押されたときには、そのPFキーの番号とビット9を“1”にしたデータが使われます。

FM16 β 準拠モードは、FM16 β 互換のキーコードが発生するモードです。JIS9bitモードとは、変換キーと無変換キーのコードだけが異なります。変換キーと無変換キーが押されると、JIS9bitモードでは、\$20(スペース)が送られます。一方FM16 β 準拠モードでは、変換キーが\$0B、無変換キーが\$0Cで、PFキーと同じくビット9が“1”となります。このFM16 β 準拠モードは、ワープロ等に便利なモードです。

スキャンコードモードは、FM16 β スキャンコードと互換のあるキーコードが発生するモードです。このモードでは、キーマトリックスで定められたキーアドレスコードが使われます。このモードの特徴は、押されているキーが離されたときに、離されたキーのキーアドレスコードのビット7を“1”にしたコードが送られることです。そしてもう一点は、**CAP**、**SHIFT**等のコントロールキーを含むすべてのキースキャンが可能だという点です。ですから、リアルタイムゲーム等のキースキャンに便利なモードです。ただし、このスキャンモードで使われるキーアドレスコードは、アスキーコードとまったく異なるので、使用には注意が必要です(図5-7)。

$\frac{B6}{36}$	$\frac{B7}{37}$	$\frac{B8}{38}$	$\frac{B9}{39}$
$\frac{BA}{3A}$	$\frac{BB}{3B}$	$\frac{BC}{3C}$	$\frac{BD}{3D}$
$\frac{BE}{3E}$	$\frac{BF}{3F}$	$\frac{C0}{40}$	$\frac{C1}{41}$
$\frac{C2}{42}$	$\frac{C3}{43}$	$\frac{C4}{44}$	$\frac{C5}{45}$
$\frac{C6}{46}$	$\frac{C7}{47}$		

上段はキーを離した時のコード
下段はキーを押した時のコード

図5-7 テン・キーのキーアドレスコード

(2) LED 制御

FM77AV では、キーボードの入力モードを示す **[CAP]**、**[かな]**、**[INS]** LED を制御することができます。

[INS] LED は、サブシステム I/O レジスタ (\$D40D) のリード／ライトにて変更することができます。

\$D40D 番地を READ **[INS]** LED の点灯

\$D40D 番地を WRITE **[INS]** LED の消灯

これは、LED の表示状態の変更だけで、入力モード自体は変化しません。**[CAP]**、**[かな]** LED は、キーボードエンコーダの MB88551 がコントロールしています。ですから **[CAP]**、**[かな]** LED を変化させるためには、MB88551 にコマンドを送って直接制御する必要があります。MB88551 を制御するには、KEYBOARD CONTROL を使用します。なお、**[CAP]**、**[かな]** LED を変化させると、入力モードも変化します。

(3) オートリピート機能

オートリピート機能とは、一定時間以上あるキーを押し続けていると、そのキーが離されるまで連続して同じキーコードを発生させる働きのことです。オートリピート機能の設定は、KEYBOARD CONTROL コマンドで行ないます。なおキーエンコードが 9bit モードのときには、以下のキーオペレーションによってもオートリピート機能の ON/OFF を制御することが可能です。

[CTRL]+[SHIFT]+[0] オートリピート OFF

[CTRL]+[SHIFT]+[1] オートリピート ON

9bit モードのビット 9 は、オートリピート動作の ON/OFF の意味も含んでいます。つまりオートリピート機能 ON であっても、ビット 9 が "1" のときには、オートリピート動作は行なわれません。ですから PF キーを押し続けてもキーコードは一度しか発生しません。

キー押下後、キーが押されたままでオートリピート開始時間が経過すると、オートリピート機能が動作開始します。そしてその後は、オートリピート動作時間ごとにキーが離されるまで、キーコードが発生します。F-BASIC 起動時には、オートリピート開始時間に 0.7 秒、オートリピート動作時間に 0.07 秒が設定されています。これは、FM-7 互換の値です(図 5-8)。

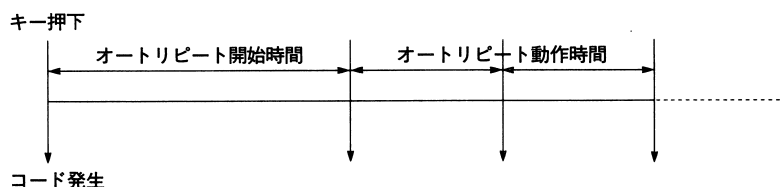


図5-8 オートリピート動作

(4) KEYBOARD CONTROL コマンド

キーエンコードに対して、以下の動作状態の設定または、その動作状態の読み取りを行いません。

- ① キーボードのキーコード系の設定
- ② 現在のキーボードのキーコード系の読み取り
- ③ LED(CAP, かな)の設定
- ④ LED(CAP, かな)の点灯状態の読み取り
- ⑤ オートリピート動作の選択
- ⑥ オートリピートの動作時間設定

コマンドの詳細について、図 5-9 に示します。

〔出力コマンド形式〕

オフセット	記号	名 称	内 容
0～1	—	—	—
2	C	コマンドコード	\$45
3	CMD	サブコマンドコード	\$00～\$05
4～5	PARAM	パラメータ	サブコマンドコードにより異なる

〔復帰情報〕

オフセット	記号	名 称	内 容
0	E	エラーコード	E = 0 の時にエラーあり
1	—	—	—
2	—	—	\$00
3	DATA	データ	サブコマンドコードにより異なる

〔サブコマンドコードの機能〕

サブコマンドコード	機 能
\$00	キーボードのキーコード系の設定
\$01	キーボードが現在扱っているキーコード系の読み取り
\$02	LEDの点灯, 消灯
\$03	LEDの点灯状態の読み取り
\$04	オートリピート動作を行なうか行なわないかの選択
\$05	オートリピートの動作時間を設定

図5-9 KEYBOARD CONTROLのコマンド形式

サブコマンドコード\$00 のときには、PARAM の上位バイトにキーコード系のモードを指定します。

PARAM 上位バイト \$00 : JIS9bit モード
 \$01 : FM16 β 準拠モード
 \$02 : スキャンコードモード

サブコマンドコード\$01 のときには、特に設定すべきパラメータはありません。コマンドを送ると、復帰情報の DATA に次のコードが返されます。

DATA \$00 : JIS9bit モード
 \$01 : FM16 β 準拠モード
 \$02 : スキャンコードモード

サブコマンドコード\$02 のときには、PARAM の上位バイトに LED の状態を指定します。

PARAM 上位バイト \$00 : CAP LED 点灯
 \$01 : CAP LED 消灯
 \$02 : かな LED 点灯
 \$03 : かな LED 消灯

サブコマンドコード\$03 のときには、特に設定すべきパラメータはありません。コマンドを送ると、復帰情報の下位 2 ビットに LED の状態が返されます。

DATA のビット 0 `0" : CAP LED 消灯
 `1" : CAP LED 点灯
 DATA のビット 1 `0" : かな LED 消灯
 `1" : かな LED 点灯

サブコマンドコード\$04 のときには、PARAM の上位バイトにオートリピート動作の設定を行います。

PARAM 上位バイト \$00 : オートリピート動作を行う
 \$01 : オートリピート動作を行わない

サブコマンドコード\$05 のときには、PARAM の上位バイトにオートリピート開始時間、下位バイトにオートリピート動作時間を設定します。両方とも設定値 $\times$ [10ms]の時間が、実際の時間となります。そして、開始時間と動作時間のどちらか一方でも\$00 が指定されると、FM-7 互換の標準設定値がとられます。

[標準設定値] オートリピート開始時間=0.7(S)
 オートリピート動作時間=0.07(S)

それでは, KEYBOARD CONTROL コマンドのサンプルとして, スキャンコードモード設定のプログラムを示します。リスト 5-4 を\$5000 番地より入力して, 続いてリスト 5-5 を入力後 RUN  で実行してみてください。実行を止めるときには, **BREAK** キーではなくて, **ESC** キーを押します。

リスト 5-4 スキャンコードモードの設定

```

01000                                *****
01010                                *   SCAN CODE MODE SET   *
01020                                *   ( LIST 5-4 )   V3.3   *
01030                                *****
01040                                OPT      NOGEN
01050      5000                      ORG      $5000
01060      5000 20      0E      5010 ENTRY  BRA      SCAN
01070                                *
01080      5002          10          SUBOUT FCB      16      REQ NO.
01090      5003          0001          RMB      1
01100      5004          500A          FDB      KEYCON DATA ADR.
01110      5006          0006          FDB      6      DATA LEN.
01120      5008          0002          RMB      2
01130      500A          0002          KEYCON RMB      2
01140      500C          45          FCB      $45      COMMAND CODE
01150      500D          00          FCB      $00      SUBCOMMAND CODE
01160      500E          02          FCB      $02      SCAN CODE
01170      500F          0001          RMB      1
01180                                *
01190      5010 8E      5002          SCAN  LDX      #SUBOUT
01200      5013 AD      9F FBFA          JSR      [$FBFA]
01210      5017 25      01      501A          BCS      ERROR
01220      5019 39          RTS
01230      501A 39          ERROR  RTS
01240                                *
01250          5000          END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=501A
PROGRAM ENTRY ADDR=5000

```

リスト 5-5 スキャンコードモード設定プログラムの実行

```

10 EXEC &H5000:WIDTH 40
20 PRINT "FD00=";HEX$(PEEK(&HFD00) AND &
H80). "FD01=";HEX$(PEEK(&HFD01))
30 IF PEEK(&HFD01)<>&H01 THEN 20
40 POKE &H500E,0:EXEC &H5000
50 POKE &H500E,2

```

つぎに, LED の点灯状態を変更するプログラムをリスト 5-6 に示します。 **CAP** キーを押すと **かな** LED が点滅し, **かな** キーを押すと **CAP** LED が点滅します。\$5000 番地より入力して, EXEC &H5000  で実行してみてください。

リスト 5-6 LED の点灯状態の変更

```

01000 *****
01010 * LED ON/OFF *
01020 * ( LIST 5-6 ) V3.3 *
01030 *****
01040 OPT NOGEN
01050 5000 ORG $5000
01060 5000 20 12 5014 ENTRY BRA LED
01070 *
01080 5002 00 CAPF FCB 0
01090 5003 00 KANAF FCB 0
01100 5004 00 CAPLED FCB 0
01110 5005 00 KNLED FCB 0
01120 *
01130 5006 11 SUBIN FCB 17 REQ NO.
01140 5007 0001 RMB 1
01150 5008 500E FDB KEYCON DATA ADR.
01160 500A 0006 FDB 6 DATA LEN.
01170 500C 0004 FDB 4 INPUT DATA LEN.
01180 500E 0002 KEYCON RMB 2
01190 5010 45 FCB $45 COMMAND CODE
01200 5011 03 FCB $03 SUBCOMMAND CODE
01210 5012 00 FCB $00
01220 5013 0001 RMB 1
01230 *
01240 5014 8E 5006 LED LDX #SUBIN LED 3ミトリ
01250 5017 A0 9F FBFA JSR [FBFA]
01260 5018 1025 00AC 50CB LBCL ERROR
01270 501F A6 0B LDA 8.X
01280 5021 1026 00A6 50CB LBNE ERROR
01290 5025 A6 0B LDA 11.X
01300 5027 85 01 BITA #01 CAP LED ON?
01310 5029 27 03 502E BEQ LED01
01320 502B 7C 5004 INC CAPLED
01330 502E 85 02 LED01 BITA #02 KANA LED ON?
01340 5030 27 03 5035 BEQ LED02
01350 5032 7C 5005 INC KNLED
01360 5035 86 10 LED02 LDA #16
01370 5037 A7 84 STA .X
01380 5039 CC 500E LDD #KEYCON
01390 503C ED 02 STD 2.X
01400 503E CC 0006 LDD #6
01410 5041 ED 04 STD 4.X
01420 5043 86 45 LDA #$45
01430 5045 A7 0A STA 10.X
01440 5047 CC 0002 LDD #$0002
01450 504A ED 0B STD 11.X
01460 504C A0 9F FBFA JSR [FBFA] SCAN CODE SET
01470 5050 25 79 50CB BCS ERROR
01480 5052 86 FD01 KEY01 LDA $FD01
01490 5055 81 55 CMPA #$55 CAP KEYIN?
01500 5057 27 40 5099 BEQ KANAON
01510 5059 81 05 CMPA #$05 CAP KEYOFF?
01520 505B 27 69 50C6 BEQ KNOFF
01530 505D 81 5A CMPA #$5A KANA KEYIN?
01540 505F 27 06 5067 BEQ CAPON
01550 5061 81 DA CMPA #$DA KANA KEYOFF?
01560 5063 27 2F 5094 BEQ CAPOFF
01570 5065 20 EB 5052 BRA KEY01
01580 5067 86 5002 CAPON LDA CAPF
01590 506A 26 E6 5052 BNE KEY01
01600 506C 7C 5002 INC CAPF
01610 506F 86 5004 LDA CAPLED
01620 5072 27 10 5084 BEQ CAPO1
01630 *
01640 5074 CC 0201 LDD #$0201 :CAP LED ON
01650 5077 ED 0B STD 11.X :LET CAP LED OFF

```



```

01660 5079 AD 9F FBFA JSR [$FBFA]
01670 507D 25 4C 50CB BCS ERROR
01680 507F 7F 5004 CLR CAPLED
01690 5082 20 CE 5052 BRA KEY01
01700 * ;CAP LED OFF
01710 5084 CC 0200 CAP01 LDD #$0200 ;LET CAP LED ON
01720 5087 ED 0B STD 11.X
01730 5089 AD 9F FBFA JSR [$FBFA]
01740 508D 25 3C 50CB BCS ERROR
01750 508F 7C 5004 INC CAPLED
01760 5092 20 BE 5052 BRA KEY01
01770 5094 7F 5002 CAPOFF CLR CAPF
01780 5097 20 B9 5052 BRA KEY01
01790 5099 B6 5003 KANAON LDA KANAF
01800 509C 26 B4 5052 BNE KEY01
01810 509E 7C 5003 INC KANAF
01820 50A1 B6 5005 LDA KNLED
01830 50A4 27 10 50B6 BEQ KANA01
01840 * ;KANA LED ON
01850 50A6 CC 0203 LDD #$0203 ;LET KANA LED OFF
01860 50A9 ED 0B STD 11.X
01870 50AB AD 9F FBFA JSR [$FBFA]
01880 50AF 25 1A 50CB BCS ERROR
01890 50B1 7F 5005 CLR KNLED
01900 50B4 20 9C 5052 BRA KEY01
01910 * ;KANA LED OFF
01920 50B6 CC 0202 KANA01 LDD #$0202 ;LET KANA LED ON
01930 50B9 ED 0B STD 11.X
01940 50BB AD 9F FBFA JSR [$FBFA]
01950 50BF 25 0A 50CB BCS ERROR
01960 50C1 7C 5005 INC KNLED
01970 50C4 20 8C 5052 BRA KEY01
01980 50C6 7F 5003 KNDOFF CLR KANAF
01990 50C9 20 87 5052 BRA KEY01
02000 50CB 39 ERROR RTS
02010 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=50CB
PROGRAM ENTRY ADDR=5000

```

このKEYBOARD CONTROL コマンドは、サブモニタ ROM のタイプ A, タイプ B で使用可能です。それで FM77AV の F-BASIC V3.0 で起動したときには、サブモニタ ROM のタイプ C が選択されるので、このコマンドは活用できません。しかし、このKEYBOARD CONTROL コマンドが関係するのは、キーエンコーダのはずですから、F-BASIC V3.0 においても動作するはずです。

それで、F-BASIC V3.0 で起動してサブモニタ ROM のタイプ A を選択した後で、このKEYBOARD CONTROL コマンドをサブ CPU に送ってみました。リスト 5-4, リスト 5-5 を入力した後、POKE &HFD13,1:RUN  で実行してみてください。

どうですか。確かにスキャンコードモードになっていますね。他のコマンドも確かめてみてください。

5-1-7 PF キーの入力

PF(プログラマブル・ファンクション)キーの入力は、PF キー割り込み制御の指定がある場合とない場合で異なります。

PF キー割り込み制御が指定されていない PF キーが押されると、PF キー定義テーブルに格納されている文字列が、キー入力バッファに格納されます。そしてキー入力バッファに格納後の処理は、通常のキーボードの入力と同じです。PF キー定義テーブルは、サブ CPU 空間の \$D2C0 ~ \$D35F に、各 PF キーに対して 16 バイトずつ割り当てられています。16 バイトのうち先頭の 1 バイトは、定義文字列の長さと割り込み制御フラグに使用されるので、定義できる文字列の最大は 15 文字となっています。PF キー定義テーブルの構造を図 5-10 に示します。

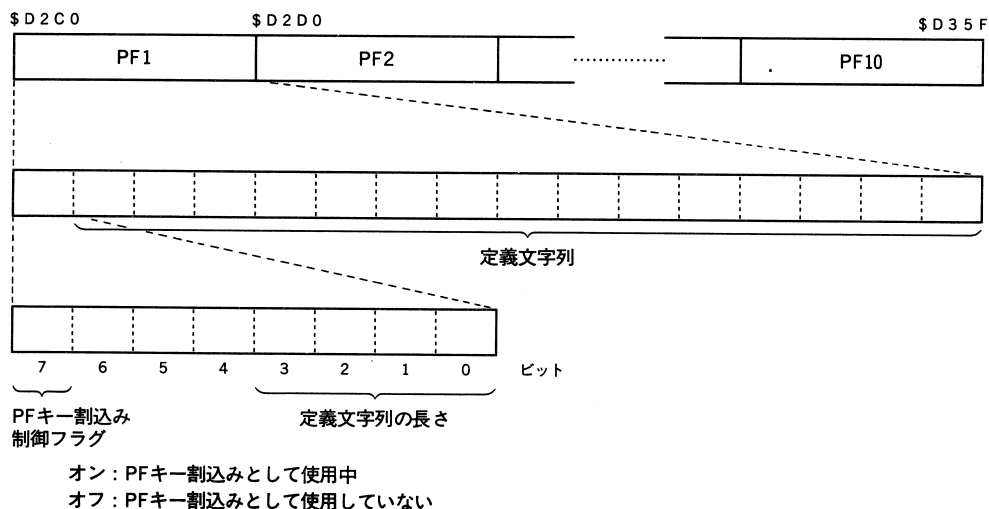


図5-10 PFキー定義テーブル

PF キー割り込み制御が指定されている PF キーが押されると、キー入力バッファにはデータは格納されません。そのかわりに \$D009 番地に、押された PF キーの番号が格納されます。そして、サブシステムのタイマー割り込み時に、\$D009 番地の内容がチェックされ、0 でなかったらメイン CPU に対して、FIRQ (PF キー割り込み) がかけられます。

5-1-8 PF キー文字列の定義

PF キーに文字列を定義するには、BASIC では KEY 文にて行ないます。

```
KEY1, "KEYLIST"+CHR$(13)
```

これをマシン語で行なうには、DEFINE STRING OF PF というサブシステムのコマンドを使います(図 5-11)。リスト 5-7 にサンプルプログラムを示します。

〔出力コマンド形式〕

オフセット	記号	名 称	内 容
0～1	—	—	—
2	C	コマンドコード	\$2A
3	NO	PFキー番号	定義するPFキー番号(1～10)
4	N	文字数	定義する文字列数(0～15)
5～	STRING	定義文字列	定義する文字列

〔復帰情報〕

オフセット	記号	名 称	内 容
0	E	エラーコード	Eキ0の時にエラーあり

図5-11 DEFINE STRING OF PFのコマンド形式

リスト 5-7 PF キー文字列定義

```

01000                *****
01010                *   DEFINE STRING OF PF   *
01020                *   ( LIST 5-7 )   V3.0/V3.3   *
01030                *****
01040                OPT      NOGEN
01050      5000            ORG      $5000
01060      5000 20      15      5017 ENTRY  BRA      PF
01070                *
01080      5002            SUBOUT FCB      16      REQ NO.
01090      5003            RMB      1
01100      5004            FDB      DEFPPF      DATA ADR.
01110      5006            FDB      13      DATA LEN.
01120      5008            RMB      2
01130      500A            RMB      2      DEFPPF
01140      500C            2A      FCB      $2A      COMMAND CODE
01150      500D            01      FCB      1      PF KEY NO.
01160      500E            08      FCB      8      STRING NUMBER
01170      500F            4B      FCC      'KEYLIST' STRING
01180      5016            00      FCB      $00      STRING
01190                *
01200      5017 8E      5002      PF      LDX      #SUBOUT
01210      501A AD      9F FBFA      JSR      [$FBFA]
01220      501E 39      RTS
01230                5000      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=501E
PROGRAM ENTRY ADDR=5000

```

5-1-9 PF キー割り込み定義

PF キー割り込みを定義するには、BASIC では KEY(n)ON/OFF/STOP 文にて行ないます。そして PF キー割り込み処理ルーチンは、ON KEY(n)GOSUB~で指定します(リスト 5-8)。マシン語で PF キー割り込み定義をするために、INTERRUPT CONTROL というサブシステムのコマンドが用意されています。PF キー文字列と同様に、サブシステムにコマンドを送って実行させます(図 5-12)。

リスト 5-8 PF キー割り込み定義

```
10 ON KEY(1) GOSUB 100
20 KEY(1) ON
30 GOTO 30
100 PRINT "PF1 INTERRUPT"
110 RETURN
```

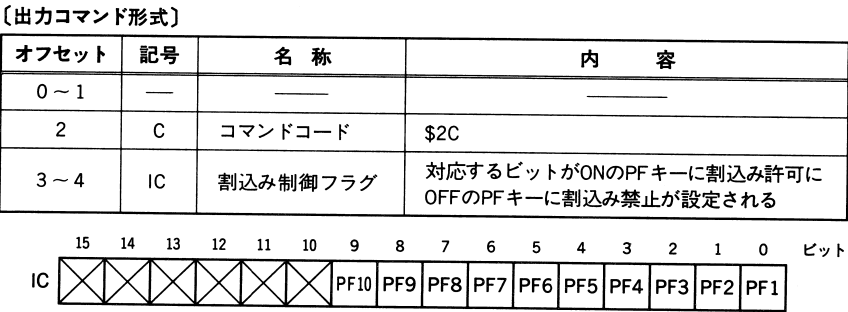


図5-12 INTERRUPT CONTROLのコマンド形式

5-1-10 PF キー文字列の初期設定

F-BASIC 起動時には、PF キー文字列に次の値が設定されています (図 5-13)。

〔V3.0 ROMモード〕	〔V3.0 DISKモード〕	〔V3.3〕
PF 1 : AUTO	PF 1 : AUTO	PF 1 : AUTO
PF 2 : LIST	PF 2 : LIST	PF 2 : LIST
PF 3 : RUN	PF 3 : RUN	PF 3 : RUN
PF 4 : CONT	PF 4 : CONT	PF 4 : CONT
PF 5 : LLIST	PF 5 : LLIST	PF 5 : LLIST
PF 6 : LOAD	PF 6 : LOAD	PF 6 : LOAD
PF 7 : SAVE	PF 7 : SAVE	PF 7 : SAVE
PF 8 : ?DATE\$,TIME\$	PF 8 : FILES	PF 8 : FILES
PF 9 : SCREEN 7,7	PF 9 : SCREEN 7,7	PF 9 : SCREEN 7,7,0,0
PF10 : HARDC	PF10 : HARDC	PF10 : HARDC

図5-13 PFキー文字列の設定値

PF キー文字列をこの設定値に戻すには、もう一度 PF1～PF10 の値を設定し直せばよいのですが、BASIC の内部ルーチンを使うと簡単にできます。V3.0 では、EXEC &H86CC で 10 個の PF キー文字列すべてが初期設定されます。

5-2 ジョイスティック

リアルタイムゲームをプレイするとき、私達とパソコンとの間に入って、マン・マシン・インターフェースの役割を果してくれるのが、ジョイスティックです。特にFM-7シリーズでは、キーボードが独特のクセを持っているため、ジョイスティックの有無は、ゲームの楽しさに大きな影響を与えます。

FM-7の発売当初はジョイスティック仕様が決まっていなかったため、周辺機器メーカーから様々の種類のジョイスティックが発売されました。たとえば、キーボードのスイッチに並列になぐものとか、プリンタインターフェースになぐもの等々……。

後になってFM音源カードの発売と同時に、FM音源カードのハードウェアを使用した仕様が決められ、それが標準仕様になっています。

この節では、そのFM音源カードのハードウェアを使用したジョイスティック入力についてのみ説明します。

5-2-1 ジョイスティック・インターフェース

FM-7シリーズのジョイスティック・インターフェースは、FM音源カードのハードウェアを利用して、図5-14の構成になっています。また、FM音源ICの図5-15に示すレジスタが、ジョイスティックの入出力をコントロールします。

ここで注意しないとイケないのは、R7です。下位6ビットが音源のコントロール、上位2ビットがジョイスティックのコントロールになっています。それでジョイスティックのコントロールだけを考えて書き換えてしまうと、誤動作の原因になってしまいます。特にFM77AVでは、FM音源でPSGを代用していますので、注意が必要です。

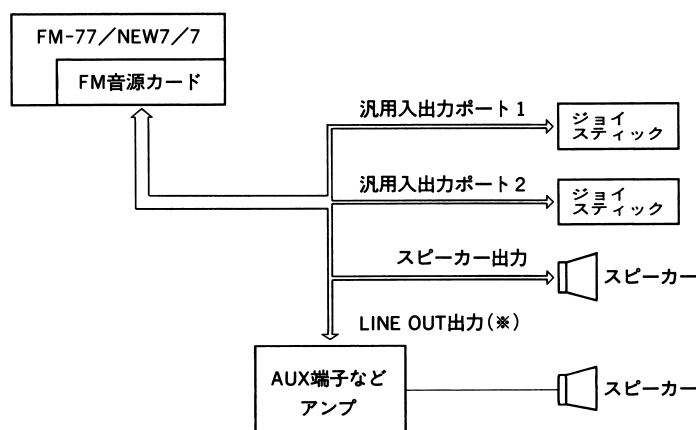


図5-14 ジョイスティックインターフェースの構成

ビット								
レジスタ	7	6	5	4	3	2	1	0
R7 (イネーブル)	Bポート 入出力方向	Aポート 入出力方向	NOISE			TONE		
R14 (ポートA入力)	1	1	ジョイスティック ステータス 0 : プレス 1 : Not					
			トリガ2	トリガ1	右	左	後	前
R15 (ポートB出力)	0	ジョイスティック切換 0 : J1 1 : J2	J2 COM出力	J1 COM出力	J2 トリガ2	J2 トリガ1	J1 トリガ2	J1 トリガ1

図5-15 ジョイスティック入力用レジスタ

5-2-2 ジョイスティックのアクセス

FM77AV では、F-BASIC V3.3 の STIC, STRIG 関数でジョイスティックの状態を読み取ることができます。しかし FM-7 では、マシン語によるジョイスティック読み取りプログラムを作成する必要があります。この項では、このマシン語ルーチンについて説明します。

まず、すべての処理に先だって、FM 音源 IC の初期設定が必要です。

- ① R15 に\$3F または、\$7F を書き込む。
- ② R7 に\$BF を書き込む。

注意として、これ以後 R7 の上位 2 ビットの状態を変更してはいけません。

ジョイスティックのデータの読み取りは、

- ① R15 に\$2F(ジョイスティック 1)または、\$5F(ジョイスティック 2)を書き込む。
- ② R14 からジョイスティックリードコマンドを使ってデータを読む。

となります。

FM 音源のレジスタ(R0～R15)を読み書きするときは、コマンドレジスタ(\$FD15)、データレジスタ(\$FD16)に、以下の手順でアクセスします。

書き込む場合は、

- ① データレジスタにアクセスしたいレジスタ番号を書く。
- ② コマンドレジスタにラッチアドレスコマンド(\$03)を書く。
- ③ コマンドレジスタにインアクティブコマンド(\$00)を書く。
- ④ データレジスタに書き込みたいデータを書く。
- ⑤ コマンドレジスタにライトデータコマンド(\$02)を書く。
- ⑥ コマンドレジスタにインアクティブコマンド(\$00)を書く。

となります。読み込む場合は、上記①～③を実行する。

④ コマンドレジスタにジョイスティックリードコマンド(\$09)を書く。

⑤ データレジスタからデータを読む。

⑥ コマンドレジスタにインアクティブコマンド(\$00)を書く。

となります。

以上の手順で作成したプログラムをリスト 5-9 に示します。初期設定は\$5000, 読み取り処理は\$5003 で, \$5006 にジョイスティックの方向, \$5007 にトリガー 1 の状態, \$5008 にトリガー 2 の状態がセットされます(図 5-16)。リスト 5-10 が, リスト 5-9 を利用したジョイスティック入力のサンプルプログラムです。

リスト 5-9 ジョイスティック読み取りルーチン

01000			*****		
01010			* JOYSTICK READ ROUTINE	*	
01020			* (LIST 5-9) V3.0/V3.3	*	
01030			*****		
01040			OPT NOGEN		
01050	5000		ORG \$5000		
01060	5000	7E	ENTRY JMP JOYINI	イニシャライズ	
01070	5003	7E	JMP JOYR	ヨミトリ	
01080			*		
01090			* ワークエリア		
01100			* ホウコウ : 8 1 2		
01110			* 7 + 3		
01120			* 6 5 4		
01130			*		
01140	5006	0001	JMOVE RMB 1	ホウコウ	
01150	5007	0001	JTRIG1 RMB 1	トリカ-1	
01160	5008	0001	JTRIG2 RMB 1	トリカ-2	
01170			*		
01180		FD15	OPNCOM EQU \$FD15	OPN コマンドレジスター	
01190		FD16	OPNDAT EQU \$FD16	OPN テーザーレジスター	
01200			*		
01210		5009	WRTSSG EQU *	<< レジスター カキコミ >>	
01220	5009	34	PSHS A	AccA ホソソ	
01230	5008	87	STA OPNDAT	レジスター ハソソウ カキコミ	
01240	500E	86	LDA #3	ラッチ アトレス コメント	
01250	5010	87	STA OPNCOM	(\$03) カキコミ	
01260	5013	7F	CLR OPNCOM	インアクティブコメント カキコミ	
01270	5016	F7	STB OPNDAT	テーザー カキコミ	
01280	5019	4A	DECA	ライトテーザーコメント	
01290	501A	87	STA OPNCOM	(\$02) カキコミ	
01300	5010	7F	CLR OPNCOM	インアクティブコメント カキコミ	
01310	5020	35	PULS A,PC	AccA フック / リターン	
01320			*		
01330		5022	RDSSG EQU *	<< レジスター ヨミタシ >>	
01340	5022	86	LDA #14	レジスターハソソウ	
01350	5024	87	STA OPNDAT	(\$0E) カキコミ	
01360	5027	86	LDA #3	ラッチ アトレス コメント	
01370	5029	87	STA OPNCOM	(\$03) カキコミ	
01380	502C	7F	CLR OPNCOM	インアクティブコメント カキコミ	
01390	502F	86	LDA #9	ジョイスティックリット コメント	
01400	5031	87	STA OPNCOM	(\$09) カキコミ	
01410	5034	F6	LOB OPNDAT	テーザー ヨミトリ	
01420	5037	7F	CLR OPNCOM	インアクティブコメント カキコミ	
01430	503A	39	RTS		

```

01440
01450          503B      *      JOYINI EQU      *      << イニシャリス >>
01460      503B CC      0F3F      LDD      #$0F3F      R15=$3F
01470      503E 8D      C9      5009      BSR      WRTSSG
01480      5040 CC      07BF      LDD      #$07BF      R7=$BF
01490      5043 20      C4      5009      BRA      WRTSSG
01500
01510          5045      *      JOYR      EQU      *      << ジョイスティック ヨミタシ >>
01520      5045 CC      0F2F      LDD      #$0F2F      R15=$2F
01530      5048 8D      BF      5009      BSR      WRTSSG
01540      504A 8D      D6      5022      BSR      RDSSG      B << JOY テーマター
01550      504C 86      01      LDA      #1
01560      504E C5      10      BITB      #$10      TRIG1=ON ナラ
01570      5050 27      01      5053      BEQ      JOYR_1      JTRIG1=1
01580      5052 4F      CLRA
01590      5053 B7      5007      JOYR_1 STA      JTRIG1
01600      5056 86      01      LDA      #1
01610      5058 C5      20      BITB      #$20      TRIG2=ON ナラ
01620      505A 27      01      505D      BEQ      JOYR_2      JTRIG2=1
01630      505C 4F      CLRA
01640      505D B7      5008      JOYR_2 STA      JTRIG2
01650      5060 8E      506B      LDX      #JOYTBL      X=ハンカンテーフル セントウ
01660      5063 C4      0F      ANDB      #15      TRIGヒット マスク
01670      5065 A6      85      LDA      B,X      A << ホウコク テーマター
01680      5067 B7      5006      STA      JMOVE
01690      506A 39      RTS
01700
01710          506B      *      JOYTBL EQU      *      << ハンカン テーフル >>
01720      506B 00      FCB      0      R L D U
01730      506C 00      FCB      0      R L D .
01740      506D 00      FCB      0      R L . U
01750      506E 00      FCB      0      R L . .
01760      506F 00      FCB      0      R . D U
01770      5070 04      FCB      4      R . D .
01780      5071 02      FCB      2      R . . U
01790      5072 03      FCB      3      R . . .
01800      5073 00      FCB      0      . L D U
01810      5074 06      FCB      6      . L D .
01820      5075 08      FCB      8      . L . U
01830      5076 07      FCB      7      . L . .
01840      5077 00      FCB      0      . . D U
01850      5078 05      FCB      5      . . D .
01860      5079 01      FCB      1      . . . U
01870      507A 00      FCB      0      . . . .
01880
01890          5000      *      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=507A
PROGRAM ENTRY ADDR=5000

```

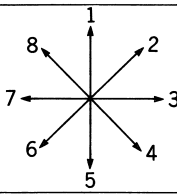
アドレス	内 容	
\$5006	ジョイスティックの 方向	0 : 中立位置 
\$5007	トリガー1の状態	0 : 押されていない 1 : 押されている
\$5008	トリガー2の状態	0 : 押されていない 1 : 押されている

図5-16 ジョイスティック入力ルーチンのインターフェース

リスト 5-10 ジョイスティックの利用プログラム

```

10 '*****
11 '*      JOYSTIC TEST      *
12 '*      ( LIST 5-10 )    V3.0/V3.3  *
13 '*      LIST 5-9   カ"   ヒツウ テ"ス  *
14 '*****
20 WIDTH 80,25:LOADM "LS-9M"
30 EXEC&H5000
40 A$="..... ウェ..... ミキ"ウェ.. ミキ"..... ミキ"シタ.. シタ..... ヒタ"リシタ. ヒタ"リ...
   ヒタ"リウェ. "
50 EXEC&H5003
60 I=PEEK(&H5006)
70 LOCATE 36,12
80 PRINT MID$(A$,I*8+1,8)
90 LOCATE 36,14
100 IF PEEK(&H5007) THEN PRINT"TRIG(1) ";
110 IF PEEK(&H5008) THEN PRINT"TRIG(2) ";
120 PRINT"      "
130 GOTO 50

```

5-3 マウス

5-3-1 マウス・インターフェース

パソコンの入力装置のなかでは、マウスは他の装置とはひと味違った趣をもっています。マウスが持つ特徴を最も生かせるのは、やはりグラフィックツール等の座標入力でしょう。しかし、それ以外の分野でも大活躍で、今やパソコンにはなくてはならない存在となっています。

さて FM シリーズのマウスですが、FM-7 シリーズには残念ながらマウスのためのインターフェースは内蔵されていません。そのため、インターフェースカードがセットになった「マウスセット」というかたちで売られています。このマウスセット以外にも、FM 音源カードのジョイスティック端子に MSX マウスを接続する方法もあります。しかしここでは、富士通純正のマウスセットについてのみ説明します。

このマウスインターフェースカードには、マウスからの信号をカウントするカウンタや、CPU にインターバル割り込みをかけるための PTM (プログラマブル・タイマ・モジュール) が載っていて、図 5-17 のような構成になっています。またマウス本体の内部は図 5-18 のようになっています。

ここで、このマウスを平板上で移動させると、中心のボールが自由に回転します。このボールには 2 個のエンコーダが直交配置されており、ボールの回転に応じてエンコーダが回転し、X-Y 方向の各成分に分かれた電気信号となり、インターフェースカード上のカウンタに加えられます。そして、このカウンタの値を読み出すことによって、X-Y 方向の移動量を知ることができるのです。

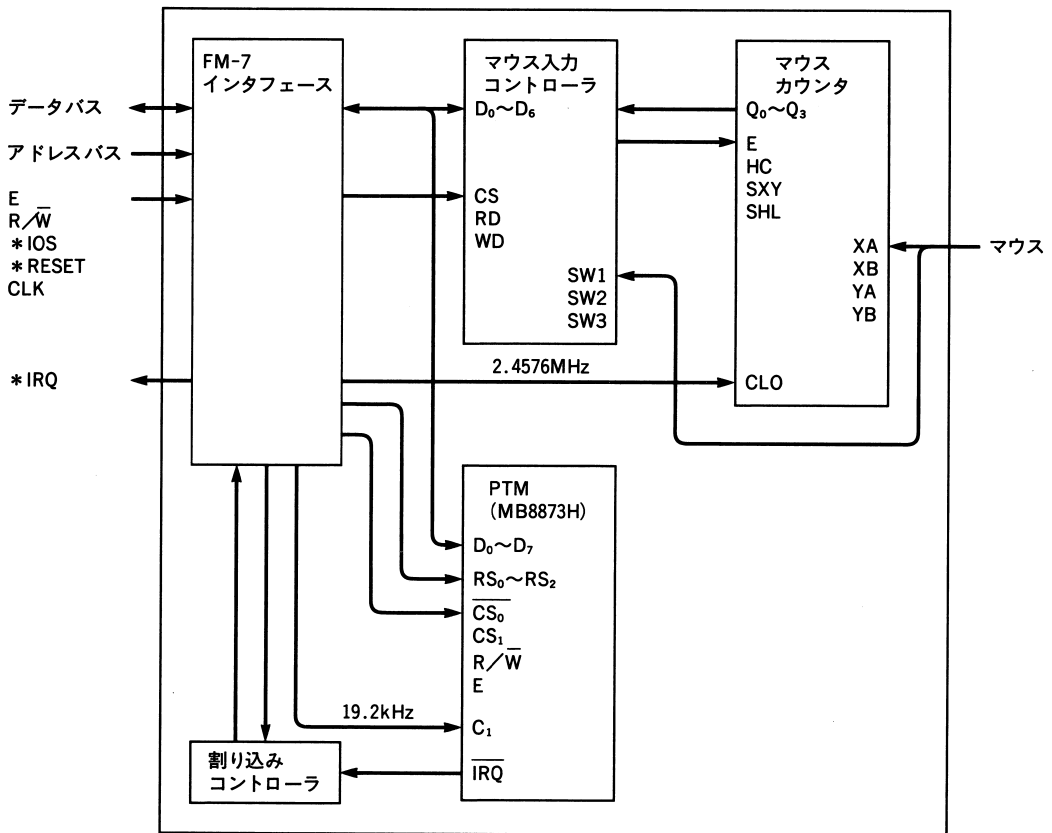


図5-17 マウスインターフェースの構成

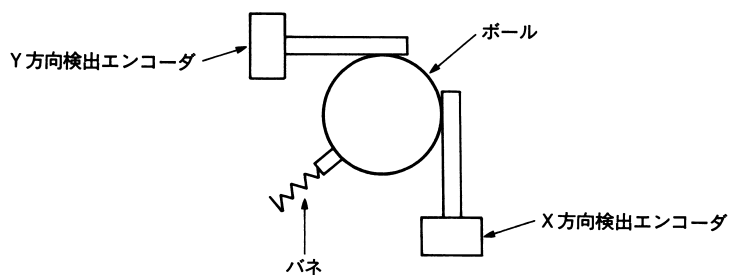


図5-18 マウスの内部構造

5-3-2 マウスのアクセス

FM-7 シリーズ用のマウスセットには、マウスドライバというマウスを読み取るためのマシン語プログラムがついています。通常の使用ではこのマウスドライバを通してマウスをアクセスするわけなのですが、このプログラムは汎用性を重視するあまり、かなり大がかりなプログラムになっています。

そこで、マウスドライバを使わずに直接インターフェースカード上のICをアクセスしてデータを読み取る方法について考えてみます。

それでは、図5-19を見てください。たくさんのレジスタがありますが、このうちマウスに直接関係のあるレジスタは、\$FDE8です。それ以外のレジスタは、CPUにインターバル割り込みをかけるためのタイマーやフラグなどです。

アドレス	内 容	R/W	ビット 構 成								
			7	6	5	4	3	2	1	0	
FD17	割り込み レジスタ	R	リ ザ ー ブ					割り込みフラグ 0 : 割り込み	リ ザ ー ブ		
		W	リ ザ ー ブ					割り込み 1 : 許可	リ ザ ー ブ		
FDE0		W	コントロールレジスタ2のビット0=0のときコントロールレジスタ3								
			出力許可 0 : 禁止 1 : 許可	割り込み許可 0 : 禁止 1 : 許可	動 作 モ ー ド			カウントモード 0 : 16ビット 1 : 8ビット	クロック源 0 : C 1 : E	クロック 0: 1/1クロック 1: 1/8クロック	
			コントロールレジスタ2のビット0=1のときコントロールレジスタ1								
			出力許可 0 : 禁止 1 : 許可	割り込み許可 0 : 禁止 1 : 許可	動 作 モ ー ド			カウントモード 0 : 16ビット 1 : 8ビット	クロック源 0 : C 1 : E	タイマ 0 : 動作 1 : プリセット	
FDE1	PTM (MB8873H)	R	ス テ ー タ ス レ ジ ス タ								
		複合割り込み フラグ	未 使 用 *0*			*0*	*0*	*0*	タイマ3 割り込み	タイマ2 割り込み	タイマ1 割り込み
		W	コ ン ト ロ ー ル レ ジ ス タ 2								
			出力許可 0 : 禁止 1 : 許可	割り込み許可 0 : 禁止 1 : 許可	動 作 モ ー ド			カウントモード 0 : 16ビット 1 : 8ビット	クロック源 0 : C 1 : E	レジスタ選択 0 : レジスタ3 1 : レジスタ1	
FDE2～E3		R/W	タイマ 1 カウンタ (16bit/8bit)								
FDE4～E5		R/W	タイマ 2 カウンタ (16bit/8bit)								
FDE6～E7		R/W	タイマ 3 カウンタ (16bit/8bit)								
FDE8	マウス	R	未使用 *1*	SW3* 0 : OFF 1 : ON	SW2 0 : OFF 1 : ON	SW1 0 : OFF 1 : ON	カ ウ ン ト デ ー タ Q ₃ Q ₂ Q ₁ Q ₀				
		W	コ ン ト ロ ー ル レ ジ ス タ							HC 0: カウント動作 1: 出力データ ホールド 内部カウンタ リセット	E 0: カウント動作 1: 出力データ ホールド 内部カウンタ 停止

図5-19 マウスインターフェースのレジスタ

マウスの高度な応用法としては、インターバル割り込みを働かせ、一定時間ごとにマウスの状態をチェックするという方法があります。しかし、我々が通常使用する場合は、インターバル割り込みを使用しなくても十分です。

では具体的にデータを読み取る方法を考えてみます。\$FDE8は読み出しと書き込みで機能が異なっていて、書き込み時はコントロールレジスタ、読み出し時はステータスレジスタをアクセスします。マウスのデータはステータスレジスタのビット0～3にあって、4回のアクセスにわけてX-Y各8ビットのデータを読み出します。またビット4、5には、スイッチの状態が常に現われています。

- ① コントロールレジスタに\$01(カウンタ停止)または,\$02(カウンタリセット)を書き込む。
- ② ステータスレジスタからデータを4回読み取る。
 - 1 回目……Xの下位4ビット
 - 2 回目……Xの上位4ビット
 - 3 回目……Yの下位4ビット
 - 4 回目……Yの上位4ビット
- ③ コントロールレジスタに\$00(カウンタスタート)を書き込む。

以上の手順を BASIC で記述したサンプルプログラムをリスト 5-11 に示します。この中で 1200～1220 行の部分で、マウスからのデータを読み取っています。

ここでもうお気付きの方もいるかと思いますが、X-Y のデータは各 8 ビットしかありません。リスト 5-11 を実行した場合も最高値は 255 となっています。グラフィック画面は 640×200 ドットの分解能があるのに、これでは画面の半分の領域さえもカバーできないことになります。そこで次のように処理することにします。

- ① マウスカウンタの値を直接利用するのではなく、X-Y の値は別に設けた各 16 ビットのワークエリアの値を使用する。
- ② マウスからデータを読み出すごとに、カウンタにリセットをかける。
- ③ 読み出したデータにワークエリアの値を加算してデータとする。

つまり前の例では、マウスカウンタから読み出したデータをそのまま絶対座標として使っていました。ところが毎回カウンタにリセットをかけることにより、前回のアクセスからどれだけ移動したかという相対的な移動量が得られます。そして前回までのデータと加算することにより、絶対座標を得ます。以上の変更点を元に、リスト 5-11 に手を加えたものがリスト 5-12 です。実際に動作させて、画面全体がカバーできるか確認してください。

最後にマシン語によるマウスのアクセスプログラムをリスト 5-13 に示します。使用法は、EXEC &H5000  です。

\$5003, \$5004 に X の座標位置(0～639), \$5005, \$5006 に Y の座標位置(0～199), \$5007 にスイッチ 1 の状態(0,1), \$5008 にスイッチ 2 の状態(0,1)がセットされます。

リスト 5-14 は、リスト 5-13 をテストするためのプログラムです。リスト 5-13 を "L5-13M" として SAVE したあとで RUN してください。

SAVEM "L5-13M", &H5000, &H505B, &H5000 

リスト 5-11 マウス読み取りプログラム 1

```

1000 '*****
1001 '*   MOUSE READ (1)   *
1002 '*   ( LIST 5-11 )   V3.0/V3.3 *
1003 '*****
1010 MOUSE=&HFDEB
1020 POKE MOUSE,2
1030 CLS
1040 POKE MOUSE,1
1050 GOSUB 1200
1060 X=DAT
1070 GOSUB 1200
1080 Y=DAT:IF Y>199 THEN Y=199
1090 POKE MOUSE,0
1100 LOCATE 32,0:PRINT"X=";X;"   "
1110 LOCATE 32,2:PRINT"Y=";Y;"   "
1120 LINE(0,0)-(XX,YY),PRESET,7
1130 LINE(0,0)-(X,Y),PSET,PUSH+4
1140 XX=X:YY=Y:GOTO 1040
1200 PUSH=PEEK(MOUSE)
1210 DAT=(PUSH AND 15)+(PEEK(MOUSE) AND 15)*16
1220 PUSH=(PUSH ¥ 16) AND 3
1230 RETURN

```

リスト 5-12 マウス読み取りプログラム 2

```

1000 '*****
1001 '*   MOUSE READ (2)   *
1002 '*   ( LIST 5-12 )   V3.0/V3.3 *
1003 '*****
1010 MOUSE=&HFDEB
1020 CLS
1030 POKE MOUSE,2
1040 GOSUB 1180
1050 X=X+DAT
1060 IF X<0 THEN X=0
1070 IF X>639 THEN X=639
1080 GOSUB 1180
1090 Y=Y+DAT
1100 IF Y<0 THEN Y=0
1110 IF Y>199 THEN Y=199
1120 POKE MOUSE,0
1130 LOCATE 32,0:PRINT"X=";X;"   "
1140 LOCATE 32,2:PRINT"Y=";Y;"   "
1150 LINE(0,0)-(XX,YY),PRESET,7
1160 LINE(0,0)-(X,Y),PSET,PUSH+4
1170 XX=X:YY=Y:GOTO 1030
1180 PUSH=PEEK(MOUSE)
1190 DAT=(PUSH AND 15)+(PEEK(MOUSE) AND 15)*16
1200 PUSH=(PUSH ¥ 16) AND 3
1210 IF DAT>=128 THEN DAT=DAT-256
1220 RETURN

```

リスト 5-13 マウス読み取りルーチン

```

01000 ****
01010 *      マウス ヨミトリ サブルーチン      *
01020 *      ( LIST 5-13 )      V3.0/V3.3      *
01030 ****
01040          OPT      NOGEN
01050 5000          ORG      $5000
01060 5000 7E      5009  ENTRY  JMP      M_READ
01070          *
01080          FDE8      MOUSE  EQU      $FDE8
01090          *
01100          *      ワークエリア
01110          *
01120 5003          0002  XPOS  RMB      2      X ホシ"ション
01130 5005          0002  YPOS  RMB      2      Y ホシ"ション
01140 5007          0001  PUSHF1 RMB      1      SW1 ノ シ"ョウタイ
01150 5008          0001  PUSHF2 RMB      1      SW2 ノ シ"ョウタイ
01160          *
01170          5009      M_READ EQU      *      << ヨミトリ エントリー >>
01180          86      02      LDA      #2      カウンター リセット
01190          87      FDE8      STA      MOUSE
01200 500E 8D      2E      503E  BSR      M_R_1  DX ヨミトリ
01210 5010 F3      5003      ADDD     XPOS      X=X+DX
01220 5013 2B      0A      501F  BMI      M_R_01 IF X<0 THEN X=0
01230 5015 1083   0280      CMPD     #640
01240 5019 25      06      5021  BCS      M_R_02 IF X>=640 THEN X=639
01250 501B CC      027F     LDD      #639
01260 501E          8C      FCB      $8C      CMPX # (SKIP 2 Bytes)
01270 501F 4F      M_R_01 CLRA
01280 5020 5F      CLRB
01290 5021 FD      5003      M_R_02 STD     XPOS
01300 5024 8D      18      503E  BSR      M_R_1  DY ヨミトリ
01310 5026 F3      5005      ADDD     YPOS      Y=Y+DY
01320 5029 2B      0A      5035  BMI      M_R_03 IF Y<0 THEN Y=0
01330 502B 1083   00C8      CMPD     #200
01340 502F 25      06      5037  BCS      M_R_04 IF Y>=200 THEN Y=199
01350 5031 CC      00C7     LDD      #199
01360 5034          8C      FCB      $8C      CMPX # (SKIP 2 Bytes)
01370 5035 4F      M_R_03 CLRA
01380 5036 5F      CLRB
01390 5037 FD      5005      M_R_04 STD     YPOS
01400 503A 7F      FDE8      CLR      MOUSE  カウンター スタート
01410 503D 39      RTS
01420          *
01430 503E 7F      5007      M_R_1  CLR      PUSHF1 SW1ノ シ"ョウタイ クリア
01440 5041 7F      5008      CLR      PUSHF2 SW2ノ シ"ョウタイ クリア
01450 5044 86      FDE8      LDA      MOUSE  カイ 4ヒット ヨミトリ
01460 5047 84      0F      ANDA     #15
01470 5049 34      02      PSHS     A
01480 504B F6      FDE8      LDB      MOUSE  シ"ョウタイ 4ヒット & SW ヨミトリ
01490 504E 58      LSLB
01500 504F 58      LSLB
01510 5050 58      LSLB
01520 5051 79      5008      ROL      PUSHF2 SW2ノ シ"ョウタイ セット
01530 5054 58      LSLB
01540 5055 79      5007      ROL      PUSHF1 SW1ノ シ"ョウタイ セット
01550 5058 EB      E0      ADDB     ,S+
01560 505A 1D      SEX      16ヒット ニ カクチョウ
01570 505B 39      RTS
01580          *
01590          5000      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=505B
PROGRAM ENTRY ADDR=5000

```

リスト 5-14 マウス読み取りルーチンのテスト

```

1000 '*****
1001 '*  MOUSE READ  (3)      *
1002 '*  ( LIST 5-14 )  V3.0/V3.3  *
1003 '*  LIST 5-13   カ"   ヒツヨウ テ"ス   *
1004 '*****
1010 CLEAR 300,&H5000
1020 LOADM"L5-13M"
1030 CLS
1040 EXEC &H5000
1050 X=PEEK(&H5003)*256+PEEK(&H5004)
1060 Y=PEEK(&H5005)*256+PEEK(&H5006)
1070 PUSHF1=PEEK(&H5007)
1080 PUSHF2=PEEK(&H5008)
1090 LOCATE 0,0
1100 PRINT USING"  X= ###  Y= ###  SW1= #  SW2= #";X,Y,PUSHF1,PUSHF2
1110 GOTO 1040

```

5-4 タイマー

FM-7シリーズでは、サブシステムの内部にソフトウェアにて制御されているタイマーを持っていて、システムにおける時間管理を行なっています。しかし、このタイマーは電源が切れるとその内容が消えてしまっていました。

FM77AV では、バッテリーバックアップされる内蔵時計 RTC(Real Time Clock)が新たに実装されています。そして電源が入ったとき、その RTC の値でサブシステムが制御しているタイマーを、初期設定するようになりました。このため FM77AV のタイマーは、電源を切っても正確な時を示すようになってます。

5-4-1 タイマーの読み取り

サブシステムの管理しているタイマーレジスタの値は、BASIC では DATE\$, TIME\$関数で知ることができます。

```
PRINT DATE$, TIME$
```

マシン語で読み取るには、サブシステムの READ TIMER コマンドを使います(図 5-20)。

FM77AV の RTC の値を読み取ることは、BASIC ではできません。しかし、サブシステムが管理しているタイマーと RTC の値は一致していますから、特に問題は起きません。しかし RTC は曜日も管理しているので、曜日を知りたいとか、タイマーの値をプログラムで正しい値に設定し直したい場合には、RTC の値が必要となります。

RTC の値は、サブシステムの READ RTC コマンドにて読み取ることができます(図 5-21)。リスト 5-15 に RTC の値を読み取って時刻を表示するプログラムを示します。リスト 5-15 を \$5000 番地より入力して次の BASIC プログラムで実行してください。(動作する F-BASIC は V 3.0 のみです)。

```
10 EXEC &H5000 : GOTO 10
```

〔出力コマンド形式〕

オフセット	記号	意 味	内 容
0～1	—	—	—
2	C	コマンドコード	\$3E

〔復帰情報〕

オフセット	記号	意 味	内 容
0～3	—	—	—
4	TC	制御レジスタ	タイマ割込み許可フラグ
5～8	T1	24時間時計レジスタ	現在の時刻(時, 分, 秒, 20ms)
9～12	T1-I	割込み予約時刻レジスタ	タイマ割込み発生時刻の設定値
13～16	T2	20msデクリメントカウンタレジスタ	20msごとにカウントダウンされるインターバルタイマ用レジスタ
17～20	T2-D	再設定値レジスタ	インターバルタイマの時間間隔の設定値

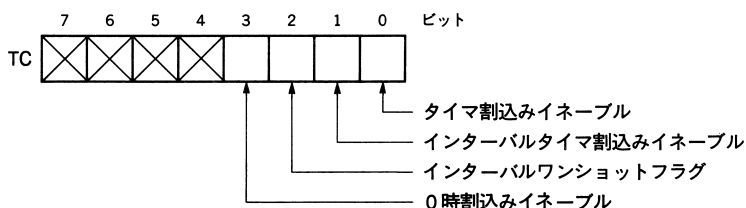


図5-20 READ TIMERのコマンド形式

〔出力コマンド形式〕

オフセット	記号	意 味	内 容
0～1	—	—	—
2	C	コマンドコード	\$42

〔復帰情報〕

オフセット	記号	意 味	内 容
0～2	—	—	—
3	Y10	年の上位桁	0～9
4	Y01	年の下位桁	0～9
5	M010	月の上位桁	0, 1
6	M001	月の下位桁	0～9
7	D10	日の上位桁	日の上位桁およびうるう年の選択 0～15
8	D01	日の下位桁	0～9
9	WK	曜日の選択	0～5
10	H10	時刻の上位桁	時刻の上位桁, AM/PMおよび24/12時計選択 0～15
11	H01	時刻の下位桁	0～9
12	MI10	分の上位桁	0～5
13	MI01	分の下位桁	0～9
14	S10	秒の上位桁	0～5
15	S01	秒の下位桁	0～9

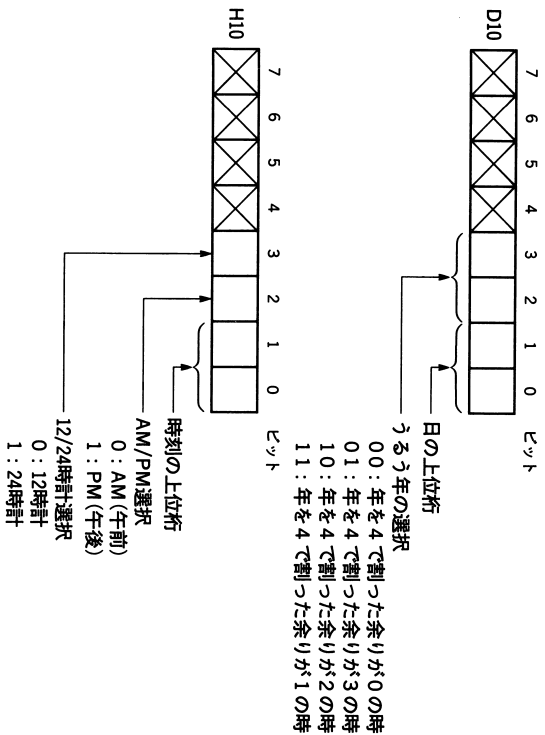


図5-21 READ RTCのコマンド形式

リスト 5-15 カレンダー時計の表示プログラム

```
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 16 01 1D 16 00 E2 16 00 B1 00 40 00 50 00 05 07 : 8F
5010 : 00 11 00 50 19 00 03 00 10 00 00 42 00 00 00 : CF
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 16 00 00 00 00 00 00 10 00 : 26
5040 : 50 46 00 2E 00 00 00 00 1C 00 40 00 50 00 4F 00 : BF
5050 : 5F 07 00 20 00 00 00 00 00 00 00 00 00 00 00 : 86
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 3E 3C 4F 42 23 40 23 40 47 2F 23 40 : AA
5080 : 23 40 37 6E 23 40 23 40 46 7C 21 4A 23 40 40 4B : F6
5090 : 46 7C 21 4B 23 40 38 61 23 40 23 40 23 40 3B 7E : 0C
50A0 : 23 40 23 40 4A 2C 23 40 23 40 49 43 36 62 45 5A : C5
50B0 : 46 7C 37 6E 32 50 3F 65 4C 5A 8E 50 11 86 11 A7 : 60
50C0 : 84 31 08 10 AF 02 CC 00 10 ED 04 CC 00 03 ED 06 : 0D
50D0 : 86 41 A7 0A 10 8E 50 29 33 08 C6 0D A6 A0 A7 C0 : 4D
50E0 : 5A 26 F9 AD 9F FB FA 39 8E 50 29 10 8E 50 1C C6 : C4
50F0 : 0D A6 80 A7 A0 5A 26 F9 CC 00 A8 FD 50 0B CC 01 : 8C

[cs] : 08 15 F7 89 17 FF 77 E3 75 DE 59 85 F8 95 E1 9E : 4A

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 05 FD 50 0D CC 02 00 FD 50 0F BD 51 3E CC 00 88 : 59
5110 : FD 50 08 CC 00 05 FD 50 0D CC 07 00 FD 50 0F 39 : EB
5120 : 8E 50 11 86 11 A7 84 31 08 10 AF 02 CC 00 03 ED : 67
5130 : 04 CC 00 10 ED 06 86 42 A7 0A AD 9F FB FA 8E 50 : 6B
5140 : 19 10 8E 50 74 A6 07 84 0C 27 0C CC 3E 3C ED A4 : C2
5150 : CC 4F 42 ED 22 20 0A CC 23 31 ED A4 CC 23 39 ED : 5C
5160 : 22 E6 03 4F C3 23 30 ED 24 E6 04 4F C3 23 30 ED : BD
5170 : 26 E6 05 4F C3 23 30 ED 2A E6 06 4F C3 23 30 ED : CB
5180 : 2C E6 07 C4 03 4F C3 23 30 ED 48 EC C6 ED A8 18 : EA
5190 : 23 30 ED A8 12 CE 50 AC A6 09 48 EC C6 ED A8 13 : 1A
51A0 : A6 0A 85 08 27 28 84 03 C6 0A 3D EB 08 C1 0C 25 : 0B
51B0 : 11 C9 0C 1F 98 BD 52 7F 34 06 CC 38 65 ED A8 24 : 7E
51C0 : 20 29 1F 98 BD 52 7F 34 06 CC 41 30 ED A8 24 20 : DE
51D0 : 1A 85 40 27 08 CC 38 65 ED A8 24 20 06 CC 41 30 : 93
51E0 : ED A8 24 E6 0A C4 03 A6 08 34 06 35 06 34 02 4F : 1B
```

```

S1F0 : C3 23 30 ED A8 26 35 04 4F C3 23 30 ED A8 28 E6 : 12
-----
[cs] : B1 ED 7C 6F 31 CD 50 7E A6 8A AA D4 94 AE 60 42 : E7
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S200 : 0C 4F C3 23 30 ED A8 2C E6 0D 4F C3 23 30 ED A8 : 1F
S210 : 2E E6 0E 4F C3 23 30 ED A8 32 E6 0F 4F C3 23 30 : A8
S220 : ED A8 34 8E 50 46 FC 50 09 ED 03 C3 00 0F ED 07 : F8
S230 : FC 50 0B ED 05 C3 00 0F ED 09 8E 50 36 10 8E 50 : 13
S240 : 54 CE 50 74 10 AF 02 C6 1C 34 04 8E 50 36 EC C1 : 82
S250 : ED 04 AD 9F FB FA 8E 50 3E FC 50 0D ED 88 13 AD : DC
S260 : 9F FB FA FC 50 0F ED 88 13 AD 9F FB FA EC 0B C3 : 72
S270 : 00 10 ED 0B C3 00 0F ED 0F 35 04 5A 26 CB 39 5F : F2
S280 : 80 0A 25 03 5C 20 F9 8B 0A 39 00 00 00 00 00 : F5
S290 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 83 14 19 0A C2 F1 59 8E 0A 80 BD 05 87 CE BF : 89

```

SAVEM "LS-15M".&H5000.&H5289.&H5000

5-4-2 タイマーへの書き込み

サブシステムが管理しているタイマーレジスタに値をセットするには、BASICでは、DATA\$, TIMES\$関数に値を代入します。

DATE\$="28/05/17"(年/月/日)

TIME\$="12/20/00"(時/分/秒)

マシン語では、サブシステムのSET TIMER コマンドを使います(図 5-22)。

RTC への設定は、BASIC からではできません。設定するには、サブシステムのSET RTC コマンドにて行ないます。RTC の設定値は通常、西暦で記録されています。それを和暦に変更したいときには、SET RTC コマンドを使用しなければなりません。

そこで RTC の値の設定プログラムを作ってみました。リスト 5-16 です。マシン語部分は、リスト 5-15 のプログラムを共通に使用します。西暦、和暦の選択から秒の設定値までを入力すると、設定値が表示されます。時計などでタイミングを見計らって、 Y キーを押してください。 Y キーを押した瞬間に、設定値が RTC に書き込まれます。

F-BASIC V3.3 のシステムディスクに入っている SETTIME によっても RTC にセットできます。しかしこの SETTIME は西暦しか対象としていないので、和暦をセットすると「うるう年」が狂ってきてしまいます。

〔出力コマンド形式〕

オフセット	記号	名 称	内 容
0～1	—	—	—
2	C	コマンドコード	\$3D
3	RC	設定レジスタ選択フラグ	設定するレジスタを選択
4	TC	制御レジスタ	タイマ割込み許可フラグ
5～8	T1	24時間時計レジスタ	現在の時刻(時, 分, 秒, 20ms)
9～12	T1-I	割込み予約時刻レジスタ	タイマ割込み発生時刻の設定値
13～16	T2	20msデクリメントカウンタレジスタ	20msごとにカウントダウンされるインターバルタイマ用レジスタ
17～20	T2-D	再設定値レジスタ	インターバルタイマの時間間隔の設定値

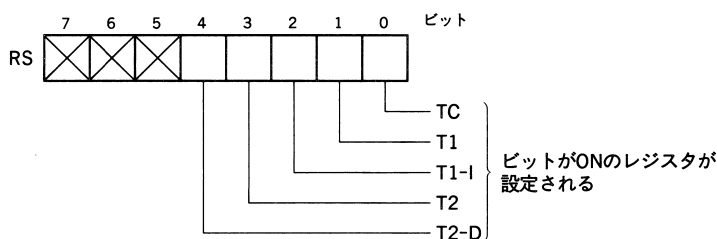


図5-22 SET TIMERのコマンド形式

リスト 5-16 RTC の設定

```

10 *****
20 *   RTC   SET   *
30 *   ( LIST 5-16 )   V3.3   *
31 *   LIST 5-15   カ"   ヒツウ テ"ス   *
32 *****
40 CLEAR .&H5000:LOADM"LS-15M":RTC=&H5000
50 DIM YB$(6),DMAX(11):FOR I=0 TO 6:READ YB$(I):NEXT:FOR I=0 TO 11:REA
D DMAX(I):NEXT
60 POKE RTC+12,184
70 INTERVAL 1:ON INTERVAL GOSUB 500:INTERVAL ON
80 CLS:RESTORE 370:FOR I=0 TO 176 STEP 16:READ A$:PRINT@ (I+112,8),VAL
("&H"+A$):NEXT
90 FOR I=0 TO 448 STEP 16:READ A$:PRINT@ (I+112,72),VAL("&H"+A$):NEXT
100 FOR I=0 TO 96 STEP 16:READ A$:PRINT@ (64,I+24),VAL("&H"+A$):NEXT
110 PRINT@ (48,72),&H4D4B
120 X=32:Y=8:GOSUB 450:IF A>1 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,0,
BF:GOTO 120
130 NEN=A
140 X=32:Y=24:GOSUB 450:YY=A*10:X=48:GOSUB 450:YY=YY+A:IF YY=0 THEN BE
EP:LINE (32,24)-(63,39),PSET,0,BF:GOTO 140
150 X=32:Y=40:GOSUB 450:IF A>1 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,0
,BF:GOTO 150
160 MM=A*10:X=48:Y=40:GOSUB 450:MM=MM+A:IF MM>12 OR MM=0 THEN BEEP:LIN
E (32,40)-(63,55),PSET,0,BF:GOTO 150
170 X=32:Y=56:GOSUB 450:IF A>3 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,0
,BF:GOTO 170
180 DD=A*10:X=48:Y=56:GOSUB 450:DD=DD+A:IF DD>DMAX(MM-1) OR DD=0 THEN
BEEP:LINE (32,56)-(64,71),PSET,0,BF:GOTO 170
190 X=32:Y=72:GOSUB 450:IF A>6 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,0
,BF:GOTO 190

```

```

200 YOBI=A
210 X=32:Y=88:GOSUB 450:IF A>2 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,0
,BF:GOTO 210
220 HH=A*10:X=48:Y=88:GOSUB 450:HH=HH+A:IF HH>23 THEN BEEP:LINE (32,88
)-(63,103),PSET,0,BF:GOTO 210
230 X=32:Y=104:GOSUB 450:IF A>5 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,
0,BF:GOTO 230
240 MI=A*10:X=48:Y=104:GOSUB 450:MI=MI+A:IF MI>59 THEN BEEP:LINE (32,1
04)-(63,119),PSET,0,BF:GOTO 230
250 X=32:Y=120:GOSUB 450:IF A>5 THEN BEEP:LINE (X,Y)-(X+15,Y+15),PSET,
0,BF:GOTO 250
260 SS=A*10:X=48:Y=120:GOSUB 450:SS=SS+A:IF SS>59 THEN BEEP:LINE (32,1
20)-(63,135),PSET,0,BF:GOTO 250
270 GOSUB 510
280 FOR I=0 TO 432 STEP 16:READ A$:PRINT@ (I+112,144),VAL("&H"+A$):NEX
T
290 A$=INKEY$:IF A$="" THEN 290
300 IF A$="N" OR A$="n" THEN 80
310 IF A$<>"Y" AND A$<>"y" THEN BEEP:GOTO 290
320 EXEC RTC+6
330 GOTO 330
340 END
350 DATA 467C,376E,3250,3F65,4C5A,3662,455A
360 DATA 31,29,31,30,31,30,31,30,31,30,31
370 DATA 214C,2330,2127,403E,4E71,2340,2340,2331,2127,3E3C,4F42,214D
380 DATA 214C,2330,2127,467C,2340,2331,2127,376E,2340,2332,2127,3250
390 DATA 2340,2333,2127,3F65,2340,2334,2127,4C5A,2340,2335,2127,3662
400 DATA 2340,2336,2127,455A,214D
410 DATA 472F,376E,467C,467C,3B7E,4A2C,4943
420 DATA 334E,4727,234F,234B,2129,2340,214A,2359,213F,234E,214B
430 DATA 2340,2340,2359,2472,3221,2439,244B,3B7E,3456,242C,405F
440 DATA 446A,2435,246C,245E,2439,2123
450 A$=INKEY$:IF A$="" THEN 450
460 IF A$<CHR$(&H30) OR A$>CHR$(&H39) THEN BEEP:GOTO 450
470 A=VAL(A$)
480 IF Y=72 AND A<7 THEN PRINT@ (X,Y),VAL("&H"+YB$(A)) ELSE PRINT@ (X,
Y),&H2330+A
490 RETURN
500 EXEC RTC:RETURN
510 POKE RTC+41,YY*10:POKE RTC+42,YY MOD 10
520 POKE RTC+43,MM*10:POKE RTC+44,MM MOD 10
530 POKE RTC+45,(DD*10) OR (NEN*4):POKE RTC+46,DD MOD 10
540 POKE RTC+47,(YOBI+2) MOD 7
550 POKE RTC+48,(HH*10) OR &H08:POKE RTC+49,HH MOD 10
560 POKE RTC+50,MI*10:POKE RTC+51,MI MOD 10
570 POKE RTC+52,SS*10:POKE RTC+53,SS MOD 10
580 EXEC RTC+3
590 RETURN

```

6-1 BASIC における割り込み処理

F-BASIC では、4 種類の割り込み処理を設定できるようになっています。そして設定した割り込み条件が発生すると、それぞれの割り込み処理ルーチンが実行されます。

- ① PF キー割り込み PF キーが押されたとき、割り込み発生
- ② インターバルタイマー割り込み..... 指定した時間が経過することに割り込み発生
- ③ 予約時刻割り込み..... 指定した時刻になったとき、割り込み発生
- ④ RS-232C 割り込み RS-232C インターフェースへの信号入力受け付け時に、割り込み発生

ここでは、PF キー割り込みを例にとりて、BASIC における割り込み処理を考えてみます。

(1) 割り込みステートメントの機能

プログラムの最初に ON KEY(n) GOSUB XXX で割り込み処理ルーチンを定義し、KEY(n) ON 文で割り込みを可能にします。指定した PF キーが押されると、定義した割り込み処理ルーチンが実行されます。この割り込み関係のステートメントとその機能は、次のとおりです。

- ON KEY(n) GOSUB 割り込み処理ルーチンの定義
- KEY(n) ON 割り込み許可
- KEY(n) OFF 割り込み禁止
- KEY(n) STOP 割り込み一時保留

この中で、KEY(n)OFF と KEY(n)STOP の違いは明確にしておいてください(図 6-1)。KEY(n)OFF は、割り込みそのものを無視するもので、KEY(n)OFF の状態のときにキーが押されても割り込みは起こりません。一方、KEY(n)STOP の方では、割り込みは発生しますが、割り込みを受け付け実際の処理ルーチンが実行されるのを保留しておくのです。ですから、KEY(n)STOP が解除された段階で受け付けられ、処理ルーチンが実行されます。割り込み処理が一時保留されているとき、KEY(n)OFF を実行すると、保留されていた割り込みはキャンセルされます。

割り込み禁止を解除し割り込み許可にするのは、KEY(n)ON です。ですから、割り込み禁止状態から割り込み保留状態にするには、いったん割り込み許可にし、その後割り込み保留にしなければなりません。図 6-2 にその関係を示します。

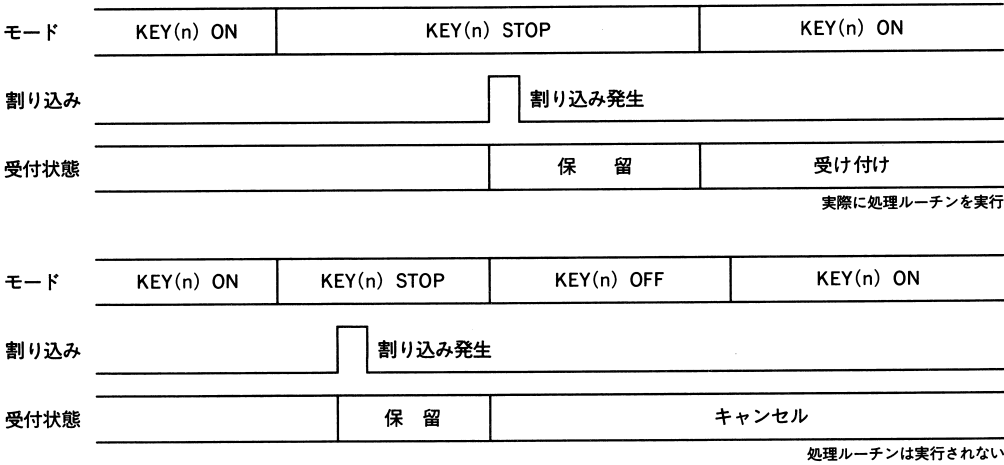


図6-1 KEY(n) ON/OFF/STOP

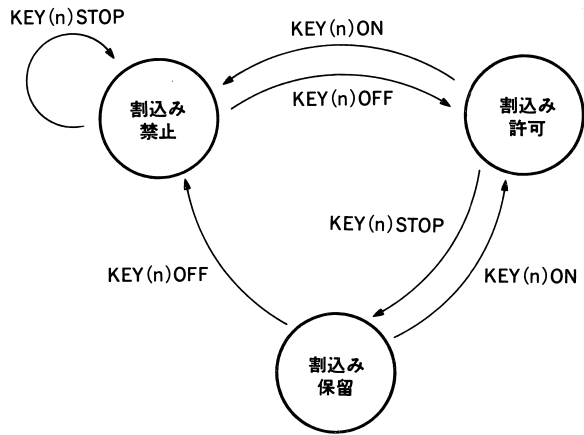


図6-2 割り込みモードの変化

(2) 割り込みはどこでかかるか

いままでは、単に割り込み許可状態のときに、PF キーを押すと割り込みがかかると思ってきました。しかし、はたして押した瞬間に割り込みがかかるのでしょうか。リスト 6-1 のプログラムを実行して、PAINT を行なっている間に PF キーを押してみてください。

PAINT が終わってから PF(1) INTERRUPT !! と表示されましたね。つまり、BASIC における割り込みは、文の実行中に割り込み要因が起こっても、文の実行が終わるまでは保留になっているわけです。別の表現をすれば、BASIC における割り込みは、BASIC の文と文の間でかかるわけです。

リスト 6-1 割り込みのタイミング

```

10 '*****
20 '*   INTERRUPT TIMING   *
30 '*   ( LIST 6-1 )   V3.0/V3.3   *
40 '*****
100 CLS
110 ON KEY(1) GOSUB 200:KEY(1) ON
120 LINE (0,0)-(639,199),PSET,5,B
130 A$=CHR$(&HDB)+CHR$(&HDB)+CHR$(&HDB)+CHR$(&HE7)+CHR$(&HE7)+CHR$(&HE
7)
140 PAINT (10,10),A$,5
150 FOR I=1 TO 1000:NEXT
160 KEY(1) OFF:END
200 '
210 PRINT "PF(1) INTERRUPT!!"
220 KEY(1) OFF:END

```

6-2 割り込みの種類

CPU6809 の割り込みを大きく分けると、ハードウェアによるものとソフトウェアによるものの2種類に分類されます。

ハードウェア割り込みには、IRQ 割り込み、FIRQ 割り込み、NMI 割り込みがあり、NMI 割り込み以外は、ソフトウェアで割り込みのマスクが可能です。

ソフトウェア割り込みには、SWI, SWI2, SWI3 の3種類があります。以下、CPU6809 の割り込みをまとめてみます(図 6-3)。

IRQ(Interrupt Request)割り込みには、キーボード、プリンタ、タイマー、EXT 割り込みの4つの原因があります。この IRQ 割り込みは、CPU6809 の IRQ ピンが“L”レベルで、かつ I フラグが“0”のとき、作動します。そして IRQ 割り込みが発生すると、I フラグが“1”にセットされ、他の IRQ 割り込みはマスクされます。

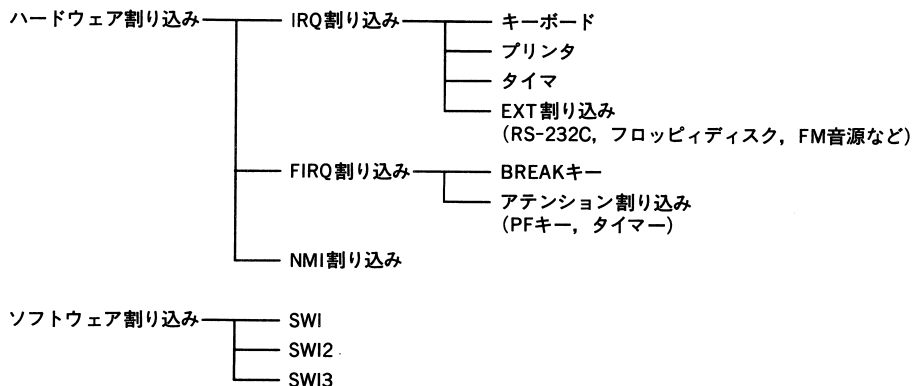


図6-3 割り込みの種類

FIRQ(Fast Interrupt Request)割り込みには、サブ CPU からのアテンション割り込みと BREAK キー割り込みの 2 つの原因があります。この FIRQ 割り込みは、IRQ 割り込みより優先度の高い割り込みで、CPU6809 の FIRQ ピンが“L”レベルで、かつ F フラグが“0”のときに作動します。そして FIRQ 割り込みが発生すると、F フラグ・I フラグが共に“1”にセットされ、他の FIRQ 割り込み、IRQ 割り込みはマスクされます。

NMI(Non Maskable Interrupt)割り込みは、20ms 毎にサブ CPU に対して発生する割り込みで、マスクすることはできません。この NMI 割り込みによって、タイマーがカウントアップ・ダウンされています。NMI 割り込みは、メイン CPU にはかかりません。

ソフトウェア割り込みは、IRQ などの外部信号による割り込みに対して、ソフト(命令)レベルで発生できる割り込みです。このソフトウェア割り込みには、SWI, SWI2, SWI3 の 3 種類があり、各命令が CPU6809 にて実行されたとき、作動します。そして SWI 命令が実行されると、F フラグ、I フラグが共に“1”にセットされ、FIRQ 割り込み、IRQ 割り込みはマスクされます。一方、SWI2, SWI3 命令が実行された場合には、FIRQ 割り込み、IRQ 割り込みはマスクされません。

6-3 割り込み要因の検出

前項で説明したように、割り込みには非常に多くの種類があります。ですから割り込み処理をプログラミングする場合、その割り込み要因を知ることとは、非常に重要です。たとえば IRQ 割り込みが発生したとします。しかし、その割り込みがキーボードが押されたための割り込みか、タイマーからの割り込みかがわからないのでは、処理しようがありません。

FM-7 シリーズでは、\$FD03 に IRQ の要求元、\$FD04 に FIRQ の要求元がセットされるので、そのアドレスの IO レジスタをリードすることにより、割り込み要因を知ることができます(図 6-4)。

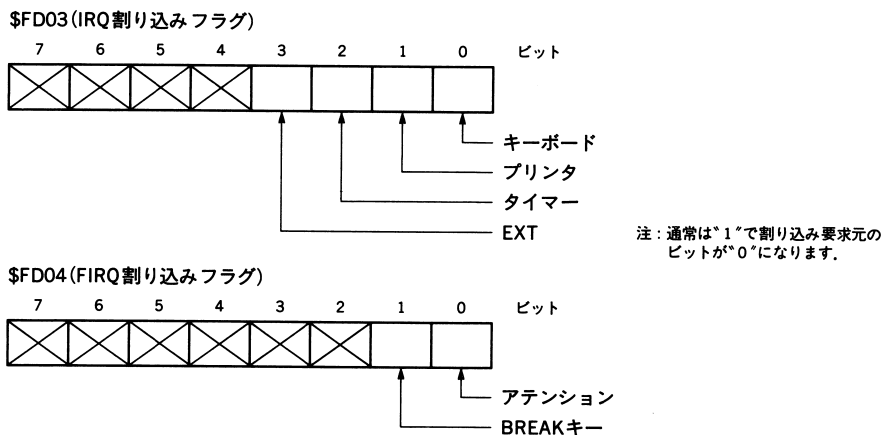


図6-4 割り込みフラグ

6-4 割り込みマスク

IRQ 割り込み全体をマスクするには、CCR(コンディション・コード・レジスタ)のIフラグを"1"にセットします。割り込みを許可するには、Iフラグを"0"にクリアします。これは、ANDCC 命令、ORCC 命令を用いればよいでしょう(図 6-5)。

FIRQ 割り込み全体をマスクするには、CCR の F フラグを"1"にセットします。許可するには"0"でクリアします。

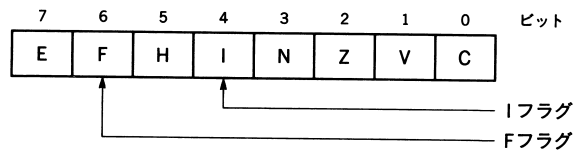


図6-5 コンディションコードレジスタ(CCR)

(1) IRQ 割り込みマスク

IRQ 割り込みは、各要因ごとに割り込みマスクをすることができます。つまり、図 6-6 に示す \$FD02 の対応のビットを"0"にすれば、その割り込みはマスクされてしまいます。

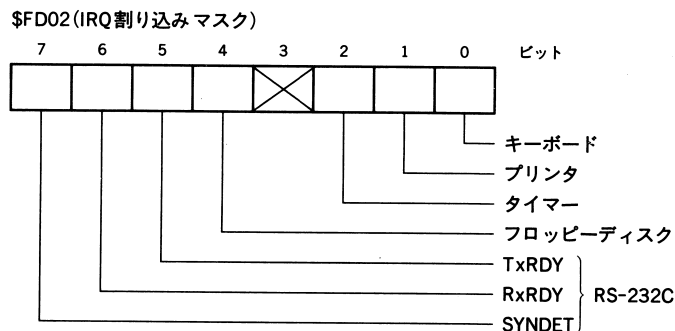


図6-6 IRQ 割り込み マスク

F-BASIC では、タイマー割り込みと RxRDY 割り込み(通信用受信バッファにデータを受信)だけを使用していて、他の IRQ 割り込みはマスクされています。

\$FD02 のビット 0 を"1"にすると、キーボード割り込みがメイン CPU の方に加わるようになります。ちょっと確かめてみましょう。

POKE &HFD02, &H45

キーボードからの入力を受け付けられなくなっていました。これは、キーボード割り込みがサブ CPU の方にかからなくなったからです。

(2) アテンション割り込みの条件設定

アテンション割り込みは、サブCPUに割り込み条件を設定することにより、はじめて割り込みがかかるようになります。このアテンション割り込みには、PFキー割り込みとタイマー割り込み(IRQのタイマー割り込みとは異なる)があります。

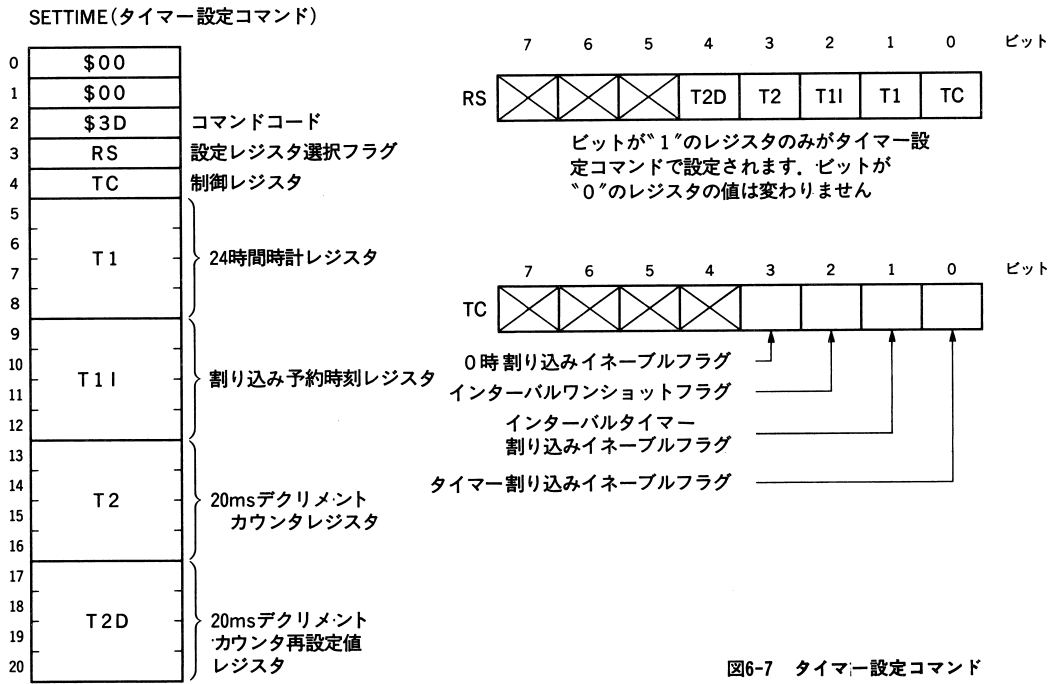


図6-7 タイマー設定コマンド

それでは、タイマー割り込みのサンプルプログラム(リスト6-2)を示します。これは、60秒ごとにブザーを鳴らす時報プログラムです。時間間隔を適当に変えて使ってみてください(図6-7)。時報の開始は、EXEC &H5000 、停止はEXEC &H5003 です。

リスト 6-2 時報プログラム

01000				*****			
01010				* シ"ホウ プログラム *			
01020				* (LIST 6-2) V3.0/V3.3 *			
01030				*****			
01040				OPT	NOGEN		
01050 5000				ORG	\$5000		
01060 5000 7E 5009				ENTRY JMP	TINIT	イニシャライズ	
01070 5003 7E 503E				JMP	TCLOSE	シュリョウ ショリ	
01090				* I / O & ワーク エリア			
01110				FIRQVC EQU	\$FFF6	FIRQへ"クトル	
01120				FIRQFL EQU	\$FD04	FIRQフラグ	
01130				BEEP EQU	\$FD03	BEEPホ"ート	
01150				HLTFLG EQU	\$FD05	BUSY/HALTホ"ート	
01160				KYORAM EQU	\$FC80	キョウラム	
01180 5006 0002				FIRQJP RMB	2	FIRQへ"クトル タイ	

01190	5008	0001	TCOUNT	RMB	1	フリコム カイスク
01200			*			
01210		5009	TINIT	EQU	*	<< INITIALIZE >>
01220	5009 1A	40		ORCC	#\$40	FIRQ キンシ
01230	5008 8E	FFF6		LDX	FIRQVC	ﾊﾞｸﾄﾙ カｷカエｽﾑ
01240	500E 8C	507D		CMPX	#FIRQET	ﾅﾗ SKIP
01250	5011 27	0C	501F	BEQ	TINIT1	
01260	5013 BF	5006		STX	FIRQJP	FIRQﾊﾞｸﾄﾙ ﾎﾂﾝ
01270	5016 8E	507D		LDX	#FIRQET	
01280	5019 BF	FFF6		STX	FIRQVC	FIRQﾊﾞｸﾄﾙ=FIRQET
01290	501C 7F	5008		CLR	TCOUNT	フリコムカイスク クリア
01300	501F 8E	5029	TINIT1	LDX	#RCB1	
01310	5022 C6	15		LDB	#21	
01320	5024 80	32	5058	BSR	SUBMOV	ｲﾝﾀｰﾊﾞﾙﾀｲﾑｰ ﾙｯﾄ
01330	5026 1C	BF		ANDCC	#\$BF	FIRQ ﾕｯｶ
01340	5028 39			RTS		
01360	5029	00	RCB1	FCB	0.0.\$3D	SET TIMER コﾑﾝﾄ"
01370	502C	19		FCB	\$19	RS
01380	502D	02		FCB	\$02	TC
01390	502E	00		FCB	0.0.0.0	T1
01400	5032	00		FCB	0.0.0.0	T1-I
01410	5036	00		FCB	0.0.0.50	T2
01420	503A	00		FCB	0.0.0.50	T2-D
01440		503E	TCLOSE	EQU	*	<< フリコム ショリ カイホウ >>
01450	503E 1A	40		ORCC	#\$40	FIRQ キンシ
01460	5040 8E	5006		LDX	FIRQJP	ﾊﾞｸﾄﾙ ﾎﾂﾓﾂﾑｽ
01470	5043 7F	FFF6		STX	FIRQVC	
01480	5046 7F	FD03		CLR	BEEP	ｵﾄﾖ ﾂﾑﾙ
01490	5049 8E	5053		LDX	#RCB2	
01500	504C C6	05		LDB	#5	
01510	504E 80	08	5058	BSR	SUBMOV	ｲﾝﾀｰﾊﾞﾙﾀｲﾑｰ ﾘｾｯﾄ
01520	5050 1C	BF		ANDCC	#\$BF	;FIRQ ﾕｯｶ
01530	5052 39			RTS		
01550	5053	00	RCB2	FCB	0.0.\$3D	SET TIMER コﾑﾝﾄ"
01560	5056	01		FCB	1	RS
01570	5057	00		FCB	0	TC
01590		5058	SUBMOV	EQU	*	<< SUB ｾﾞｰﾀｰ ｾﾝｿｳ >>
01600	5058 8D	0E	5068	BSR	SUBSTP	SUB CPU ｾｲｼ
01610	505A CE	FC80		LDU	#KYORAM	
01620	505D A6	80	S_MOV1	LDA	,X+	ｾﾞｰﾀｰ ｾﾝｿｳ
01630	505F A7	C0		STA	,U+	
01640	5061 5A			DECB		
01650	5062 26	F9	505D	BNE	S_MOV1	
01660	5064 7F	FD05		CLR	HLTFLG	SUB CPU ﾏｰﾄ
01670	5067 39			RTS		
01690		5068	SUBSTP	EQU	*	<< SUB CPU HALT >>
01700	5068 86	FD05		LDA	HLTFLG	READY ﾏﾁ ﾎｰﾌﾟ
01710	5068 2B	F8	5068	BMI	SUBSTP	
01720	506D 86	80		LDA	#\$80	HALT ﾙｯﾄ
01730	506F 87	FD05		STA	HLTFLG	
01740	5072 86	0A		LDA	#10	
01750	5074 7D	FD05	S_STP1	TST	HLTFLG	ｾｲｼ ﾕｸﾆﾝ
01760	5077 2B	03	507C	BMI	S_STP2	
01770	5079 4A			DECA		
01780	507A 26	F8	5074	BNE	S_STP1	
01790	507C 39		S_STP2	RTS		
01810		507D	FIRQET	EQU	*	<< FIRQ ﾍﾝﾄﾘｰ >>
01820	507D 34	16		PSHS	D,X	
01830	507F 86	FD04		LDA	FIRQFL	SUB CPU ﾍｶﾞｲﾝ
01840	5082 84	01		ANDA	#1	フリコム ﾅﾗ ﾘﾀｰﾝ
01850	5084 26	18	509E	BNE	FIRQRT	
01860	5086 86	5008		LDA	TCOUNT	カイスク ﾕｳﾝﾄｱｯﾌﾟ
01870	5089 4C			INCA		
01880	508A 87	5008		STA	TCOUNT	
01890	508D 81	02		CMPA	#2	カイスク=2 ﾅﾗ ｵﾄﾖ ﾂﾑﾙ
01900	508F 27	1F	5080	BEQ	BEEP_2	
01910	5091 81	39		CMPA	#57	カイスク<57 ﾅﾗ ﾘﾀｰﾝ
01920	5093 25	09	509E	BCS	FIRQRT	

```
01930 5095 81 3C          CMPA  #60      カイスク>60 ナラ クリック
01940 5097 26 08 50A4     BNE   BEEP_1
01950 5099 7F 5008       CLR   TCOUNT   フリコミカイスウ クリアー
01960 509C 20 17 50B5     BRA   BEEP_3     BEEP ON
01980 509E 35 16          FIRQRT PULS  D,X
01990 50A0 6E 9F 5006     JMP   [FIRQJP] ;ホソライノ フリコミシヨリ ヲ
02010          50A4       BEEP_1 EQU   *          << BEEP シュツリョク >>
02020 50A4 86 81          LDA   #81      BEEP ON
02030 50A6 87 FD03       STA   BEEP
02040 50A9 8E 0258       LDX   #600     シ"カンマチ
02050 50AC 30 1F          LEAX  -1,X
02060 50AE 26 FC 50AC     BNE   *-2
02070 50B0 7F FD03       BEEP_2 CLR   BEEP      BEEP OFF
02080 50B3 20 E9 509E     BRA   FIRQRT
02100 50B5 86 81          BEEP_3 LDA   #81      BEEP ON
02110 50B7 87 FD03       STA   BEEP
02120 50BA 20 E2 509E     BRA   FIRQRT
02140          5000       END   ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
PROGRAM BEGIN ADDR=5000
PROGRAM END  ADDR=50B8
PROGRAM ENTRY ADDR=5000
```

6-5 割り込みベクトル

CPU6809 では、割り込みが発生すると、マスクされている場合は別として、レジスタ類をスタックへ退避し、対応する割り込みベクトルで示されたアドレスへジャンプします。そして、RTI 命令の実行によって退避されていたレジスタ値を復元し、割り込みが発生した時点の PC(プログラムカウンタ)の示すアドレスに制御が戻されます。これによって、あたかも何もなかったように元の処理が続行できるわけです。

CPU6809 の割り込みベクトルは、図 6-8 に示すように、\$FFF2～\$FFFC に割り当てられています。

ソフトウェア割り込み(SW1, SWI2, SWI3)と NMI 割り込みは、RTI 命令を指しているので、結局何も行なわれません。

〔割り込みベクトル〕

割り込み	アドレス	起動時設定値	
		V3.0	V3.3
IRQ 割り込み	(\$FFF8, \$FFF9)	\$01DD	\$F06E
FIRQ 割り込み	(\$FFF6, \$FFF7)	\$01E0	\$F057
NMI 割り込み	(\$FFFC, \$FFFD)	\$01DA	\$F04B
SWI 割り込み	(\$FFFA, \$FFFB)	\$01D7	\$01D7
SWI2 割り込み	(\$FFF4, \$FFF5)	\$01D4	\$01D4
SWI3 割り込み	(\$FFF2, \$FFF3)	\$01D1	\$01D1
RESET	(\$FFFE, \$FFFF)	\$FE00	\$FE00

〔割り込みフック〕

割り込み	起 動 時 設 定 値			
	V 3.0		V 3.3	
IRQ 割り込み	(\$01DD~\$01DF)	JMP \$D2FC	(\$F06E~	
FIRQ 割り込み	(\$01E0~\$01E2)	JMP \$C953	(\$F057~	
NMI 割り込み	(\$01DA~\$01DC)	RTI	(\$F04B)	RTI
SWI 割り込み	(\$01D7~\$01D9)	RTI	(\$01D7~\$01D9)	RTI
SWI2割り込み	(\$01D4~\$01D6)	RTI	(\$01D4~\$01D6)	RTI
SWI3割り込み	(\$01D1~\$01D3)	RTI	(\$01D1~\$01D3)	RTI

図6-8 割り込みベクトル

ユーザーが、これらの割り込みを独自の処理に使用する場合には、\$FFF2~\$FFFCの割り込みベクトルか、\$01D1~\$01E2のアドレスの内容を、ユーザー割り込み処理のアドレスに書き換えることになります。

6-6 BREAK キーをキャンセル

BREAK キーは FIRQ 割り込みを使って処理されています。それでは、この FIRQ の処理をスキップさせたらどうなるでしょうか？

FIRQ がかかると CPU は、\$FFF6、\$FFF7 の内容を読み込んで、そのアドレスにジャンプするようになっています。F-BASIC V3.0 では、その値が \$01、\$E0 になっていますので、\$01E0 へとジャンプすることになります (V3.3 では \$F057)。それでは、ここの内容を書き換えてみましょう。割り込み処理をスキップさせるためには、何もせずにリターンさせればよいわけですから RTI 命令に置き換えてみます。

キーボードから POKE &H1E0,&H3B [] と入力してみましょう (V3.3 では POKE &HF057,&H3B [])。その後で、BREAK キーを押してみてください。確かに BREAK キーが、キャンセルされていますね。

FIRQ を元に戻すには、POKE &H1E0,&H7E [] とします (V3.3 では POKE &HF057,&H34 [])。

6-7 SWI 命令でマシン語のデバッグ

F-BASIC V3.0 には、簡単な機械語モニタが付いていますが、機能が貧弱なことからあまり使われていないようです。一方、ソフトウェア割り込みも割り込みベクトルには、ちゃんと値が書かれており、ジャンプ用のフックまで用意されています。しかし実際には、そのフック先に RTI 命令が書かれているため、SWI 命令を実行してもそのまま何もせずに帰ってしまいます。

そこで、SWI 命令のフック先に機械語モニタのエントリアドレスを書き込んで、SWI 命令で機械語モニタに入るようにしてみました(リスト 6-3)。ユーザーのマシン語プログラムで途中で SWI 命令を入れておくと、ソフトウェア割り込みが発生して機械語モニタにジャンプして止まります。ですから、機械語モニタの R コマンドでその時点でのレジスタの内容を確認することができます。これによって、ソフトウェア割り込みを一種のブレークポイントとして活用できるわけです。

リスト 6-3 マシン語易デバッガ

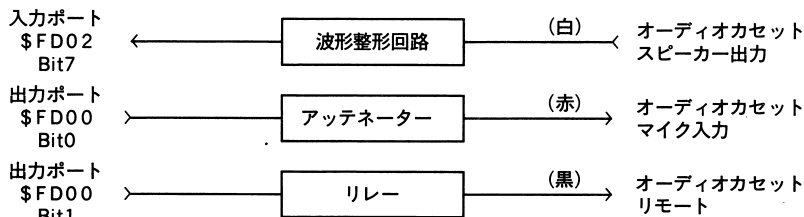
```
100 '*****
110 '*      マシンコ" カンイ テ"イハ"カ"      *
120 '*      ( LIST 6-3 )      V3.0      *
130 '*****
140 POKE &H01D7,&H7E
150 POKE &H01D8,&HAB
160 POKE &H01D9,&HF4
```

7-1 CMT インターフェース

FM-7 シリーズには、プログラム／データの入出力用として、一般のオーディオカセットレコーダー用のインターフェースが内蔵されています。そのインターフェースの構成は、1 ビットの入力／出力ポートとカセットのモーターを ON/OFF させるためのポートがあるだけのシンプルなものとなっています(図 7-1)。

制御はすべてソフトウェアで行なうことになるため、データフォーマットやボーレートは自由に決めることができるわけです。しかし、その反面ソフトウェアに大きな負担がかかります。また入力信号の位相が反転していた場合、データの読み込み状態が悪くなります。これは、録音状態のあまり良くないテープを入力したときに、顕著に起こるようです。

このように、ソフトウェアに大きく依存したインターフェースとなっていますが、そのボーレートは専用データレコーダを使用しないものとしては最高の部類に入ります。



アドレス/ビット	7	6	5	4	3	2	1	0
\$FD00 ライト	プリンタ SLCTIN	プリンタ STRB					リモート 0 : OFF 1 : ON	データ 出力
\$FD02 リード	データ 入力		プリンタ DET2	プリンタ DET1	プリンタ PE	プリンタ ACKNG	プリンタ ERROR	プリンタ BUSY

図7-1 CMTインタフェースの構成

7-2 データフォーマット

データフォーマットはソフトウェアで自由に設定することが可能なのですが、BIOS で扱う標準的なフォーマットが次のように決められています。

- "0"のビット 2400Hz 1 波形
- "1"のビット 1200Hz 1 波形
- スタートビット "0"のビット 1 波形
- ストップビット "1"のビット 2 波形

たとえば\$A5 のデータは、図 7-2 のようになります。

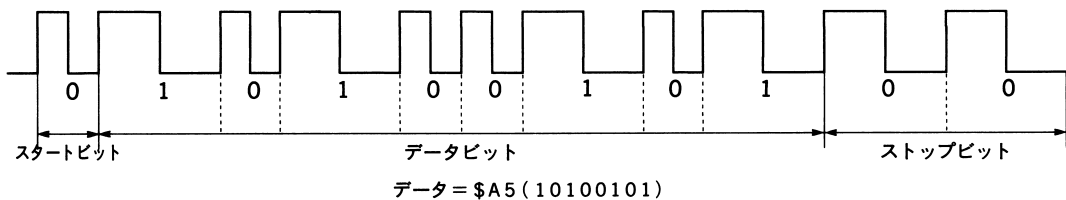


図7-2 カセットファイルのデータ波形

図を見ていただければお気づきかと思いますが、ボーレートは出力するデータによって変化することになります。それを平均すると約 1600 ボーになります。

F-BASIC のカセットファイルは、次の 3 種類の“ブロック”という単位より構成されています。

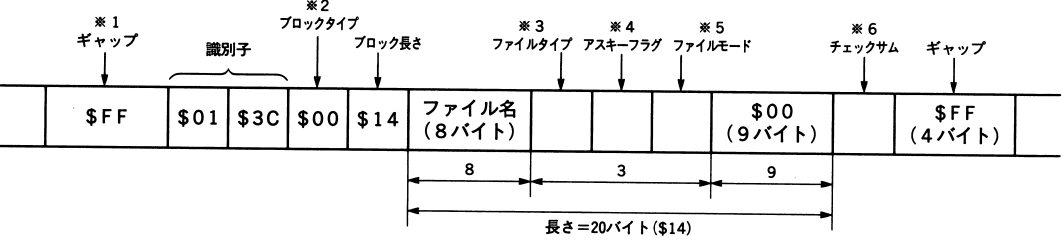
- ① ヘッダブロック
- ② データブロック
- ③ エンドブロック

ブロックの内容は、ギャップ(\$FF)が 10 または 255 バイト、そして識別子(\$01,\$3C)、ブロックタイプ、データ長と続いた後、データが 0~255 バイト出力されます。そして、ギャップと識別子以外のすべてのデータを 2 進加算して求めたチェックサムが出力され、最後にエンドギャップ(\$FF が 4 バイト)で終わります(図 7-3)。

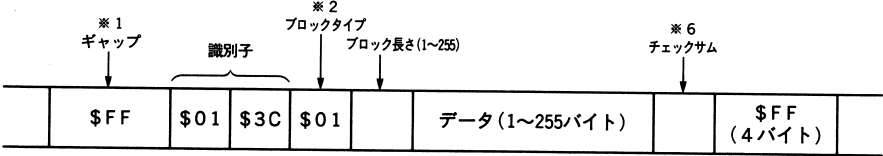
ヘッダブロックはデータ長が 20 バイト固定で、ファイル名やファイルのタイプなどのディスクのディレクトリに相当するものが設定されています。データブロックには、実際のプログラム／データが 255 バイト毎に区切って設定されます。データ長は 255 バイトが基本ですが、最終のデータブロックだけは 255 バイトとは限りません。また BASIC プログラムの場合には UNLIST の行番号、マシン語の場合にはプログラムの長さやロード開始番地、実行開始番地等の情報も設定されています(図 7-4)。エンドブロックは、データ長が 0 バイトでファイルの終わりを示しています。

それでは最後に、メモリ上のマシン語データを SAVEM "MDATA", &H5000, &H501F, &H5010 としてセーブしたときのデータフォーマットを、図 7-5 に示します。マシン語データはすべて、\$55 とします。

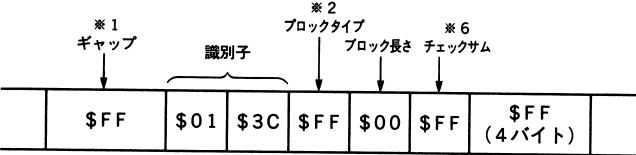
■ヘッダーブロック



■データブロック



■エンドブロック



- ※1 : ギャップ モーターがONのとき10バイト, OFFのとき255バイト
- ※2 : ブロックタイプ \$00→ヘッダーブロック, \$01→データブロック, \$FF→エンドブロック
- ※3 : ファイルタイプ \$00→BASICソース, \$01→BASICデータ, \$02→マシン語
- ※4 : アスキーフラグ \$00→バイナリ, \$FF→アスキー
- ※5 : ファイルモード \$00→バイナリプログラム, \$FF→アスキープログラム及びデータ
- ※6 : チェックサム ギャップと識別子を除いた1ブロックの総和

図7-3 ブロックの形式

\$00	プログラムの長さ	ロード開始番地	プログラム	///	\$FF	\$0000	実行開始番地
------	----------	---------	-------	-----	------	--------	--------

図7-4 データブロックの形式

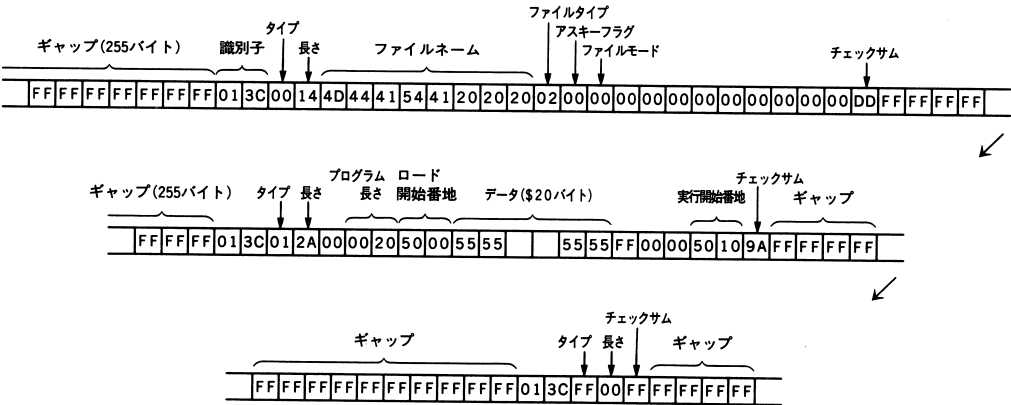


図7-5 マシン語データファイルのフォーマット

7-3 カセットファイルに対する BIOS

FM-7 シリーズでは多くの種類の入出力装置をサポートするため、BIOS と呼ばれるプログラムが存在しています。そしてカセットに関係する BIOS として、次に示す 3 種類が用意されています。

- ① カセットモーターコントロール(MOTOR)
- ② カセットテープ 1 バイトライト(CTBWRT)
- ③ カセットテープ 1 バイトリード(CTBRED)

これらの BIOS コマンドは、CMT インターフェースのポートを直接操作して、1 バイトのデータをシリアルにテープとやりとりするだけに過ぎません。つまり、前項で説明したブロックごとに区切られたデータフォーマットは、すべて BASIC インタープリタで作られている訳です (F-BASIC V3.3 では、1 ブロックのリード/ライトとなっています)。

また忘れてはならないこととして、FM-7 のクロック周波数切り替え機能との関係があります。読み出し、書き込みのタイミングはすべて、ソフトウェアでカウントして行なっています。ですからクロック周波数が切り替わると、タイミングが合わなくなって正常なアクセスができなくなります。そのため BIOS では、2 種類のカセット 1 バイトリード/ライトルーチンを用意して、クロック周波数に応じて切り替えて使用しています。それで BIOS 経由にてリード/ライトしたデータは、クロック周波数にかかわらず一定となっています。

それでは、簡単に BIOS のコマンドを説明します(図 7-4)。

(1) カセットモーターコントロール(MOTOR)

リクエスト番号は\$01 で、RCB+2 にモーターフラグがあります。これに\$FF をセットして BIOS をコールすればモーター ON、それ以外ならモーター OFF になります。

(2) カセットテープ 1 バイトライト(CTBWRT)

リクエスト番号は\$02 で、RCB+2 に書き込みたいデータをセットして BIOS をコールします。

(3) カセットテープ 1 バイトリード(CTBRED)

リクエスト番号が\$03 で、BIOS をコールすると RCB+2 にデータがセットされて戻ります。そしてこの BIOS では、1 バイトのデータを正常に読み取るまでリターンしてきません。ただし BREAK キーが押されると、処理を中断して BASIC の ABORT 処理を実行します。

それでは図 7-5 と同じ内容のカセットテープを、BIOS を使って作ってみたいと思います。リスト 7-1 がそれです。実行は、EXEC &H5000  とします。カセットテープ作成後、実際に LOADM で読み込むことが可能か確認してみてください。

●F-BASIC V3.0 BIOS

・オーディオカセット モーターコントロール

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	1	
1	エラーステータス		○
2	モーターフラグ	○	

※ モーターフラグ : \$FF → ON
\$FF以外 → OFF

・オーディオカセット 1バイトライト

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	2	
1	エラーステータス		○
2	ライトデータ	○	

・オーディオカセット 1バイトリード

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	3	
1	エラーステータス		○
2	リードデータ		○

●F-BASIC V3.3 BIOS

・オーディオカセット モーターコントロール

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	1	
1	エラーステータス		○
2	モーターフラグ	○	
3			
4			
5			
6			
7			

※ モーターフラグ : \$FF → ON
\$FF以外 → OFF

・オーディオカセット 1ブロックライト

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	2	
1	エラーステータス		○
2	ライトデータ	○	
3	アドレス		
4	データ長	○	
5			
6	ブロックタイプ	○	
7	ギャップフラグ	○	

※ ギャップフラグ : \$00 → ノーマルデータ
\$00以外 → ギャップ

・オーディオカセット 1ブロックリード

相対値	内 容	ユーザー セット	BIOS セット
0	リクエスト番号	3	
1	エラーステータス		○
2	リードデータ	○	
3	アドレス		
4	データ長	○	
5			
6	ブロックタイプ		○
7			

図7-6 BIOSのRCB

リスト 7-1 カセットテープの作成

```
01000
01010
01020
01030
01040
01050      5000
01060      5000 7E      5006
01080
01100
01120      5003      FBFA      0003
01140      5006 8D      39      5041
01150      5008 8D      48      5052
01160      500A 8E      506F
01170      500D C6      19
01180      500F 8D      51      5062
01190      5011 8D      45      5058
01200      5013 8D      55      506A
01210      5015 8D      38      5052
01220      5017 8E      5085
01230      501A C6      2F
01240      501C 8D      44      5062
01250      501E 8D      38      5058
01260      5020 8D      48      506A
01270      5022 8D      31      5055
01280      5024 8E      50B4
01290      5027 C6      05
01300      5029 8D      37      5062
01310      502B 8D      28      5058

*****
*      カセット テープ° サクセイ      *
*      ( LIST 7-1 )      V3.0      *
*****
OPT      NOGEN
ORG      $5000
ENTRY    JMP      START
*      ワーク Etc...
BIOS     EQU      $FBFA
RCBW     RMB      3
START    BSR      MOTON
          BSR      GAP255
          LDX      #HEADBL
          LDB      #25
          BSR      OUTPUT
          BSR      GAP4
          BSR      WAIT
          BSR      GAP255
          LDX      #DATABL
          LDB      #47
          BSR      OUTPUT
          BSR      GAP4
          BSR      WAIT
          BSR      GAP10
          LDX      #ENDBL
          LDB      #5
          BSR      OUTPUT
          BSR      GAP4
```

第7章 カセットファイル

01320	502D	8D	15	5044	BSR	MOTOFF	
01330	502F	39			RTS		
01350	5030	34	14		WRTBYT	PSHS	B.X 1 ハイト カ*コミ
01360	5032	8E	5003		LDX	#RCBW	
01370	5035	C6	02		LDB	#2	
01380	5037	E7	84		STB	.X	
01390	5039	A7	02		STA	2.X	
01400	503B	AD	9F	FBFA	JSR	[BIOS]	
01410	503F	35	94		PULS	B.X,PC	
01430	5041	86	FF		MOTON	LDA	#\$FF モータ ON/OFF
01440	5043		21		FCB	\$21	
01450	5044	4F			MOTOFF	CLRA	
01460	5045	8E	5003		LDX	#RCBW	
01470	5048	A7	02		STA	2.X	
01480	504A	86	01		LDA	#1	
01490	504C	A7	84		STA	.X	
01500	504E	6E	9F	FBFA	JMP	[BIOS]	
01520	5052	C6	FF		GAP255	LDB	#255 キ*アップ° シュツリョク
01530	5054		8C		FCB	\$8C	
01540	5055	C6	0A		GAP10	LDB	#10
01550	5057		8C		FCB	\$8C	
01560	5058	C6	04		GAP4	LDB	#4
01570	505A	86	FF		GAPZ	LDA	#\$FF
01580	505C	8D	D2	5030	BSR	WRTBYT	
01590	505E	5A			DECB		
01600	505F	26	F9	505A	BNE	GAPZ	
01610	5061	39			RTS		
01630	5062	A6	80		OUTPUT	LDA	.X+ シタイ ハイトスウ シュツリョク
01640	5064	8D	CA	5030	BSR	WRTBYT	
01650	5066	5A			DECB		
01660	5067	26	F9	5062	BNE	OUTPUT	
01670	5069	39			RTS		
01690	506A	4F			WAIT	CLRA	シ*カン マチ
01700	506B	4A			DECA		
01710	506C	26	FD	506B	BNE	*-1	
01720	506E	39			RTS		
01740	506F		01		HEADBL	FCB	\$01.\$3C テーマ
01750	5071		00		FCB	\$00.\$14	
01760	5073		4D		FCC	'MDATA'	
01770	5078		02		FCB	2.0.0	
01780	507B		00		FCB	0.0.0.0.0.0.0.0	
01790	5084		DD		FCB	\$DD	
01810	5085		01		DATABL	FCB	\$01.\$3C
01820	5087		01		FCB	\$01.\$2A	
01830	5089		00		FCB	\$00	
01840	508A		00		FCB	\$00.\$20	
01850	508C		50		FCB	\$50.\$00	
01860	508E		55		FCB	\$55.\$55.\$55.\$55	
01870	5092		55		FCB	\$55.\$55.\$55.\$55	
01880	5096		55		FCB	\$55.\$55.\$55.\$55	
01890	509A		55		FCB	\$55.\$55.\$55.\$55	
01900	509E		55		FCB	\$55.\$55.\$55.\$55	
01910	50A2		55		FCB	\$55.\$55.\$55.\$55	
01920	50A6		55		FCB	\$55.\$55.\$55.\$55	
01930	50AA		55		FCB	\$55.\$55.\$55.\$55	
01940	50AE		FF		FCB	\$FF	
01950	50AF		00		FCB	\$00.\$00	
01960	50B1		50		FCB	\$50.\$10	
01970	50B3		9A		FCB	\$9A	
01990	50B4		01		ENDBL	FCB	\$01.\$3C
02000	50B6		FF		FCB	\$FF.\$00	
02010	50B8		FF		FCB	\$FF	
02030			5000		END	ENTRY	
TOTAL ERRORS 00000--00000							
TOTAL WARNINGS 00000--00000							
PROGRAM BEGIN ADDR=5000							
PROGRAM END ADDR=50B8							
PROGRAM ENTRY ADDR=5000							

7-4 カセットファイルの拡張ディレクトリ表示

F-BASIC のファイル管理は大変しっかりしていて、カセットファイルに対しても一部を除いてディスクと同様に扱えるようになっていました。ですからカセットファイルに対しても、FILES コマンドを使うことができます。

FILES "CAS0 : " 

時間はかかりますが、ファイル名とファイルタイプ(BASIC かマシン語か……)が表示されます。ただし得られる情報のわりに時間がかかるので、あまり利用価値のあるコマンドとはいえません。

そこでこの FILES コマンドをパワーアップしたプログラムを考えてみました。ファイル名とファイルタイプ以外に、マシン語ファイルのアドレス情報も表示するようにしてみました。またプリンタに印字することも可能で、次の節で紹介する 2 倍速 LOAD プログラムと組み合わせれば、さらに便利なものとなります。

ファイル名、ファイルタイプ、アスキーフラグ、ファイルモードは、ヘッダブロックに記録されています。したがってマシン語のアドレス情報以外は、ヘッダブロックを読むだけですんでします。マシン語ファイルの場合は、さらに次のデータブロックを読み、必要なアドレス情報を求めます。

それでは実際のプログラムをごらんください。リスト 7-2 が F-BASIC V3.0 用、リスト 7-3 が V3.3 用です。BIOS 内部での処理内容が、V3.0 と V3.3 で大きく異なるため別々のプログラムになってしまいました。

実行方法はいずれの場合も EXEC &H5000  です。また、EXEC &H5003  とするとプリンタにも出力されます。カセットテープの読み取り中に **BREAK** キーを押すと、停止します。

リスト 7-2 拡張ディレクトリ (V3.0 用)

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 52 94 7E 52 91 00 00 00 00 00 00 00 00 00 00 : C5
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 7E 52 94 7E 52 91 00 00 00 00 00 00 00 00 00 00 : C5

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S100 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S110 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S120 : 00 00 00 00 00 00 00 00 00 00 00 00 10 8E 00 00 : 9E
S130 : 8D 47 31 21 81 FF 27 F8 10 8C 00 08 25 EE 81 01 : FE
S140 : 26 EA 8D 35 81 3C 26 E4 10 8E 50 06 8D 2B A7 A0 : 8C
S150 : 87 51 08 8D 24 A7 A0 1F 89 BB 51 08 B7 51 08 5D : 31
S160 : 27 0D 8D 15 A7 A0 BB 51 08 B7 51 08 5A 26 F3 8D : 41
S170 : 08 B1 51 08 10 26 01 31 39 8E 51 10 86 03 A7 84 : 63
S180 : AD 9F FB FA A6 02 39 86 FF 21 4F 8E 51 1D A7 02 : 8C
S190 : 86 01 A7 84 6E 9F FB FA A6 A0 A7 80 5A 26 F9 39 : D3
S1A0 : 8D 8A B6 50 06 26 F9 8D E1 10 8E 50 08 8E 51 09 : 8E
S1B0 : C6 22 E7 80 C6 08 8D E0 EC A4 10 8E 51 DF 81 02 : 6B
S1C0 : 27 08 10 8E 51 E7 5D 26 04 10 8E 51 EF C6 08 8D : C8
S1D0 : C7 8E 51 09 C6 11 8D 9C 16 8D 9B 50 8D A9 39 22 : 2E
S1E0 : 20 20 20 2E 4F 42 4A 22 20 20 20 2E 41 53 43 22 : 12
S1F0 : 20 20 20 2E 42 41 53 B6 50 10 81 02 27 01 39 7F : DD
-----
[cs] : 4D 65 84 41 65 F2 1A 04 E6 8C A1 F8 41 94 F9 A5 : 6A

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S200 : 50 07 8D 71 8D 6F 87 51 28 8D 6A B7 51 29 8D 65 : 9B
S210 : B7 51 22 8D 60 87 51 23 FC 51 28 1F 01 F3 51 22 : 3D
S220 : 83 00 01 FD 51 24 30 03 34 10 8D 49 35 10 30 1F : D7
S230 : 26 F6 8D 41 B7 51 26 8D 3C 87 51 27 17 FF 4B 8E : FF
S240 : 52 69 C6 0C 8D 9C 16 FC 51 22 8D AC 37 86 2C 8D : 7A
S250 : D0 8E FC 51 24 8D AC 37 86 2C 8D D0 8E FC 51 26 : AF
S260 : 8D AC 37 8D 9B 50 16 FF 1E 20 20 20 20 41 44 44 : C4
S270 : 52 45 53 3D 20 F6 50 07 27 0C 8E 51 2A A6 8D BF : E5
S280 : 51 2A 7A 50 07 39 17 FE A3 8E 50 08 BF 51 2A 20 : 7D
S290 : E4 86 01 21 4F 87 05 AC 0F BF 10 FF 51 20 17 FE : A6
S2A0 : E6 17 FE FC 17 FF 50 20 F8 8E 52 BC C6 12 8D 9C : 42
S2B0 : 16 8D 9B 50 17 FE D3 10 FE 51 20 39 20 44 45 56 : 5D
S2C0 : 49 43 45 20 49 2F 4F 20 45 52 52 4F 52 07 00 00 : 69
S2D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 5B FD E2 70 5E 56 14 37 9D 9D EC 7E F5 62 DD 2A : AB

```

SAVEM "L7-2M",&H5000,&H52CD,&H5000

リスト 7-3 拡張ディレクトリ (V3.3 用)

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 52 5E 7E 52 5B 00 00 00 00 00 00 00 00 00 : 59
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 7E 52 5E 7E 52 5B 00 00 00 00 00 00 00 00 00 : 59

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5110 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5120 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5130 : 8E 51 1C 86 03 A7 84 CC 50 08 ED 02 AD 9F FB FA : 03
5140 : A6 01 10 25 01 36 A6 05 B7 50 07 A6 06 B7 50 06 : 85
5150 : 39 86 FF 21 4F 8E 51 1C A7 02 86 01 A7 84 6E 9F : 91
5160 : FB FA A6 A0 A7 80 5A 26 F9 39 8D C4 B6 50 06 26 : 97
5170 : F9 8D E1 10 8E 50 08 8E 51 08 C6 22 E7 80 C6 08 : 61
5180 : 8D E0 EC A4 10 8E 51 A9 81 02 27 0B 10 8E 51 B1 : EA
5190 : 5D 26 04 10 8E 51 B9 C6 08 8D C7 8E 51 08 C6 11 : 0F
51A0 : BD AE A6 BD AD BF 8D A9 39 22 20 20 20 2E 4F 42 : EA
51B0 : 4A 22 20 20 20 2E 41 53 43 22 20 20 20 2E 42 41 : 04
51C0 : 53 86 50 10 81 02 27 01 39 7F 50 07 8D 71 8D 6F : 1D
51D0 : B7 51 2C 8D 6A B7 51 2D 8D 65 B7 51 26 8D 60 B7 : 24
51E0 : 51 27 FC 51 2C 1F 01 F3 51 26 83 00 01 FD 51 28 : 75
51F0 : 30 03 34 10 8D 49 35 10 30 1F 26 F6 8D 41 B7 51 : D3
-----
[cs] : DD 66 14 0B 97 28 63 3D 44 97 AB B6 09 0B 22 B1 : 81

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5200 : 2A 8D 3C B7 51 2B 17 FF 48 8E 52 33 C6 0C BD AE : D7
5210 : A6 FC 51 26 BD 93 D3 86 2C BD DF 23 FC 51 28 BD : DF
5220 : 93 D3 86 2C BD DF 23 FC 51 2A BD 93 D3 BD AD BF : 9A
5230 : 16 FF 1E 20 20 20 20 41 44 44 52 45 53 3D 20 F6 : B9
5240 : 50 07 27 0C BE 51 2E A6 8D BF 51 2E 7A 50 07 39 : 35
5250 : 17 FE DD 8E 50 08 BF 51 2E 20 E4 86 01 21 4F B7 : C8
5260 : FC 04 0F BF 0F 27 10 FF 51 24 86 01 97 C4 7F FF : E8
5270 : 9A 17 FE DD 17 FE F3 17 FF 47 20 F8 0F C4 7F FF : 5A
5280 : 9A 8E 52 9B 81 03 26 03 8E 52 AD C6 12 BD AE A6 : 38
5290 : BD AD BF 17 FE BE 10 FE 51 24 39 20 44 45 56 49 : 00
52A0 : 43 45 20 49 2F 4F 20 45 52 52 4F 52 07 3D 3D 3D : D7
52B0 : 3D 3D 20 41 42 4F 52 54 20 3D 3D 3D 3D 07 00 : 6A
52C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 4D 38 93 9B 0F 9A C5 69 5B 0B 8D 50 A3 CC 4E 3A : C1

```

SAVEM "L7-3M", &H5000, &H52BE, &H5000

7-5 2倍速 LOAD

最近ではMSX用で2倍速ロードが可能なデータレコーダを見かけます。これは1200ボートで普通にセーブしたテープを、テープスピードを2倍にして再生して疑似的に2400ボートに見せかけているのです。

FM-7シリーズがボーレートをソフトウェアで決めていることは、前にも述べました。そこで今回は、ボーレートを決めている部分を書き換えて、無理矢理2倍速でテープを読ませるようにしてみました。

F-BASIC V3.0の場合、BIOSのカセットテープ1バイトリードルーチンの中に3カ所ほど、タイミング調節用のループがあります。ボーレートを2倍にしたいわけですから、ここの値を半

分にすれば良いこととなります。実際には他のルーチンで消費される時間のことも考えて、少し小さめの値にしておきます。

しかし V3.0 は ROM ですので、そのままでは書き換えられません。そこで \$8000~\$FBFF の BASIC ROM をそっくりそのまま裏 RAM へ転送してしまい、そこで書き換えを行なうことにします。

F-BASIC V3.0 用のプログラムをリスト 7-4 に示します。RUN  にて実行してください。(F)AST, (N)ORMAL? と聞いてきますから、 または  のキーを押して選択してください。

次に F-BASIC V3.3 の場合です。V3.3 は、BASIC インタープリタが RAM 上にありますから、話は非常に簡単です。しかもタイミング調節用ループカウンタにセットする値は、V3.0 とまったく同じになっています。したがって V3.0 と同じデータを、POKE 文で書き込んでやればよいこととなります。リスト 7-5 が、F-BASIC V3.3 用 2 倍速 LOAD のプログラムです。

リスト 7-4 2 倍速 LOAD (V3.0 用)

```

100 '*****
110 '* FAST LOAD FOR CMT *
120 '* ( LIST 7-4 ) V3.0 *
130 '*****
1010 PRINT"(F)AST,(N)ORMAL?"
1020 A$=INPUT$(1)
1030 IF A$="F" OR A$="f" THEN 1080
1040 IF A$="N" OR A$="n" THEN 1060
1050 BEEP:GOTO 1010
1060 A=PEEK(&HFD0F)
1070 END
1080 FOR I=&H500 TO &H526
1090 READ A$:POKE I,VAL("&H"+A$)
1100 NEXT
1110 EXEC &H500
1120 END
1130 DATA 34.13.1A.50
1140 DATA 8E.80.00.B6
1150 DATA FD.0F.A6.84
1160 DATA B7.FD.0F.A7
1170 DATA 80.8C.FC.00
1180 DATA 25.F1.86.19
1190 DATA B7.F4.50.86
1200 DATA 16.B7.F4.5C
1210 DATA 86.13.B7.F4
1220 DATA 8F.35.93

```

リスト 7-5 2 倍速 LOAD (V3.3 用)

```

100 '*****
110 '* FAST LOAD FOR CMT *
120 '* ( LIST 7-5 ) V3.3 *
130 '*****
1010 PRINT"(F)AST,(N)ORMAL?"
1020 A$=INPUT$(1)
1030 IF A$="F" OR A$="f" THEN 1060
1040 IF A$="N" OR A$="n" THEN 1100
1050 BEEP:GOTO 1010
1060 POKE &HF34D,&H19
1070 POKE &HF34D,&H16
1080 POKE &HF37D,&H13
1090 END
1100 POKE &HF34D,&H36
1110 POKE &HF34D,&H30
1120 POKE &HF37D,&H2C
1130 END

```


フロッピーディスク

第 8 章

8-1 フロッピーディスクの構造

8-1-1 物理構造

FM-7 シリーズで使用されるマイクロ(ミニ)フロッピーディスクは、図 8-1 のような物理構造(フォーマット)を持っています。この図はディスクの片面を表わしたもので、ディスク 1 枚の仕様は次のようになります。

シリンダ数 40
サイド数 2(両面)
トラック数 80(40×2)
セクタ数 16(1 トラック)
セクタ長 256 バイト
総記憶容量 320K バイト($80 \times 16 \times 256 = 327680$)

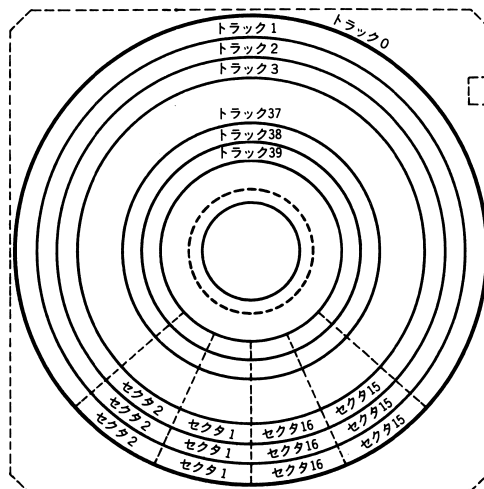


図8-1 フロッピーディスクの物理構造

ここでディスク関係の用語を説明しておきます。

- ① セクタ………ディスクの読み書きにおける最小単位です。その長さはフォーマット時に決められ、F-BASIC 等では 256 バイトに固定されています。

- ② サイド……ディスク面を識別するもので、表面を0、裏面を1で表わします。
- ③ トラック……ディスク上で輪状になったセクタのひと続きを指します。
- ④ シリンダ……読み書き用ヘッドが止まる位置のことです。読み書き用ヘッドはディスクの両面にあって同じ動きをするので、同じ半径のトラックならサイドが異なっても同一のシリンダ番号となります。

FM-7シリーズで使用されるマイクロフロッピー(3.5インチ)とミニフロッピー(5.25インチ)の論理構造(ソフトウェアからみた構造)は、まったく同一です。トラックの半径が小さくなっただけと見ることができます。したがってデータの互換性には、何ら問題がありません。

機械的な特徴を考えると、マイクロフロッピーはミニフロッピーの問題点をすべて解消したものといえます。

- ① 機械的強度を増した。
- ② 小型にした。
- ③ 書き込み保護をやすくした。
- ④ 誤挿入できないようにした。
- ⑤ 読み書きウィンドウを保護した。

操作上の違いで間違えやすいのは、書き込み保護のやり方です。マイクロフロッピーでは、成形片を外側にずらす(穴が見えるようにする)と書き込み禁止となります。一方ミニフロッピーでは、切り欠きの上にシールを貼って切り欠きをなくすと、書き込み禁止になります。

8-1-2 セクタアドレスとクラスタ

フロッピーディスクのある特定のセクタを指定するには、シリンダ番号、ヘッド番号、セクタ番号を組み合わせで用います。しかしF-BASICでは、図8-2のようにサイド0、サイド1の全セクタに通し番号を付け、その通し番号とトラック番号により、セクタを指定しています。

ですから、BIOSやFDCにおいては前者のセクタアドレスを、またF-BASIC内部では後者のセクタアドレスを考える必要があります。

フロッピーディスクの読み書きの最小単位は1セクタですが、F-BASICでは、8セクタを管理の最小単位として扱います。この単位をクラスタと呼びます。

	サイド0			サイド1		
トラック0	セクタ1	………	セクタ16	セクタ17	………	セクタ32
トラック39	セクタ1	………	セクタ16	セクタ17	………	セクタ32

図8-2 F-BASICでのセクタアドレス

トラック 2 のセクタ 1～8 がクラスタ 0, トラック 39 のセクタ 25～32 がクラスタ 151 となります。

8-1-3 ディスクマップ

F-BASIC で使用するフロッピーディスクの各トラックには、図 8-3 のような内容に割り当てられています。

〔V3.0〕

トラック番号	内 容
0	IPL, ID, DISKコード
1	FAT, ディレクトリ
2～39	ユーザー領域

〔V3.3〕

トラック番号	内 容
0	IPL, ID, DISKコードなど
1	FAT, ディレクトリ
2～9	BASICコード, エラーメッセージ
10～39	ユーザー領域

図8-3 ディスクマップ

- ① IPLDISK コードをメモリ上に展開するプログラムです。(Initial Program Loader)
- ② IDF-BASIC で使用できるディスクであることを識別するための領域です。(Identification)
- ③ DISK コードF-BASIC のディスクに関する命令が格納されています。
- ④ FAT各クラスタの使用状況を記録する領域です。(File Allocation Table)
- ⑤ ディレクトリファイル名, ファイル形式, ファイル位置を記録している領域です。
- ⑥ ユーザー領域ユーザーが実際にファイルとして使用可能な領域です。

各トラック内でのアロケーションマップは、図 8-4 (V3.0), 図 8-5 (V3.3) のようになっています。

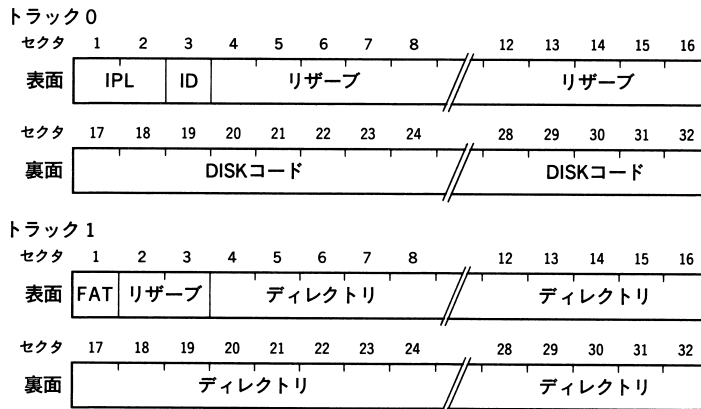


図8-4 アロケーションマップ(V3.0)

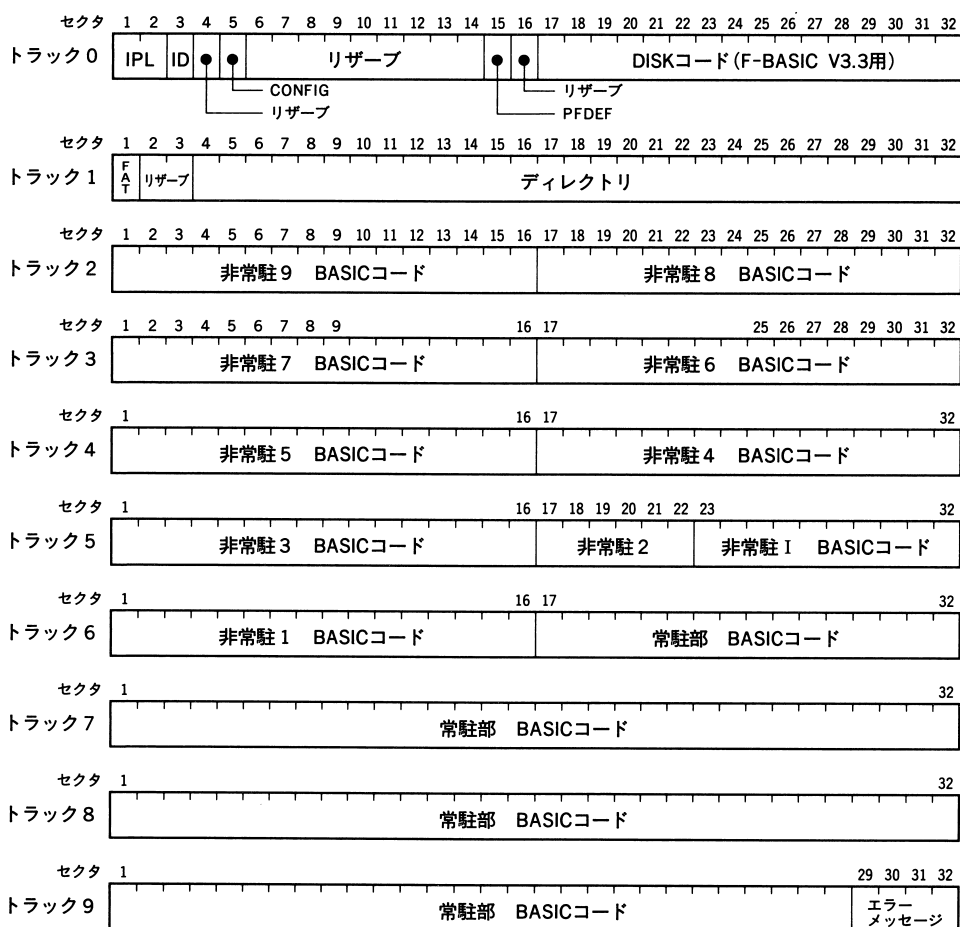


図8-5 アロケーションマップ(V3.3)

8-1-4 ID セクタ

ディスクがF-BASICで使えるものかどうかを確認するためのコードが記録されています。システムが参照しているのは、最初の1文字("S")だけで、次の2文字("YS")は、ユーティリティなどで使用されています。

4バイト目と5バイト目は、オートスタート時に設定されるドライブ数と一度にOPENできるファイル数です。これらは、AUTOUTYを使用すると書き込まれます(図8-6)。

[0] (0) 「 3 」																															
cs	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	cs	0123456789ABCDEF													
00	53	59	53	02	02	00	00	00	00	00	00	00	00	00	00	00	03	SYS													
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00														
cs	53	59	53	02	02	00	00	00	00	00	00	00	00	00	00	00	03														
ID情報		ドライブ数		1度にOPENできるファイルの数																											

図8-6 IDセクタの内容

8-1-5 ディレクトリ

ディレクトリには、ファイル名、ファイルタイプ、ファイル先頭クラスタ番号などが格納されています。これによりファイル名と実際にファイルが格納されている場所との対応がつけられます。

ディレクトリの位置は、トラック1のセクタ4～セクタ32となっています。ディレクトリ部のセクタは、32バイトずつの8ブロックに分割され、この32バイトをディレクトリスロットと呼びます。ディレクトリスロットは232個ありますが、クラスタ数は152なので、実際登録できるファイル数は152となります(図8-7、図8-8)。

位置(バイト)	内 容
0 ～ 7	ファイル名
8 ～ 10	——
11	ファイルタイプ 00 : BASICソース 01 : BASICデータ 02 : 機械語ファイル
12	アスキーフラグ FF : アスキー 00 : バイナリ
13	ランダムアクセスフラグ FF : ランダムファイル 00 : シーケンシャルファイル
14	ファイル先頭クラスタ番号
15～31	——

図8-7 ディレクトリスロットの構成

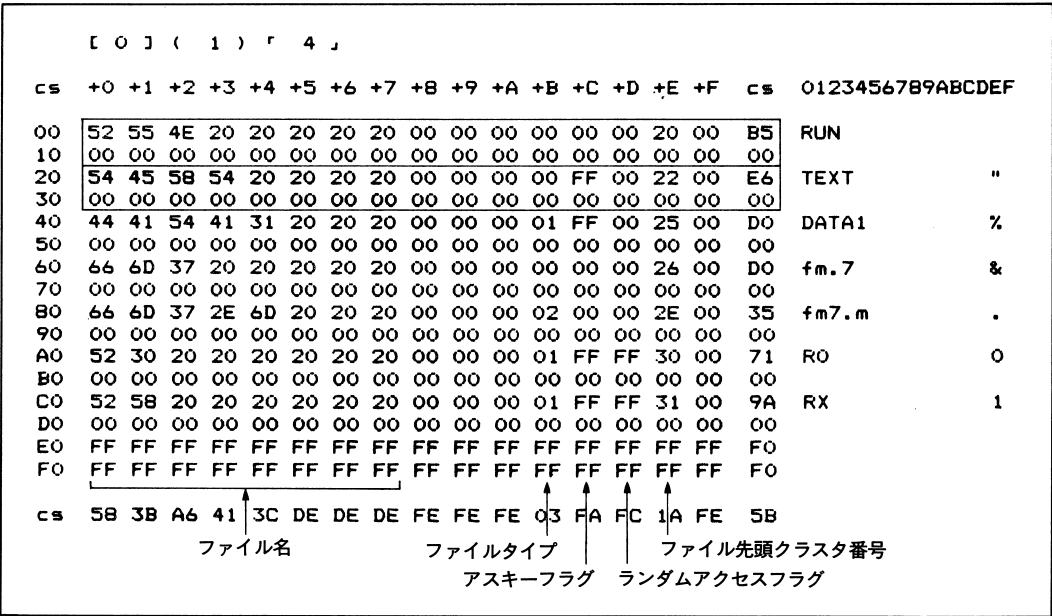


図8-8 ディレクトリの内容

8-1-6 FAT

FATはファイルの格納状態を示します。ファイルが1クラスタに納まらない場合、残りを別のクラスタに書き込まなければなりません。この別のクラスタを捜すときに、どのクラスタが空いているかが分からなければなりません。また、どここのクラスタに書き込んだかを記録しておかないと、後で困ることになります。これらの情報を記録したのが、FATです。

FATの位置は、トラック1の最初のセクタ内にあり、そのセクタ内の6～157バイトがクラスタの0～151のFAT情報に対応しています。このFAT情報の意味は、図8-9のようです。

図8-10のFATの内容を見てください。これは、図8-8のディレクトリに対するFATです。RUNというファイルのファイル先頭クラスタ番号は、\$20になっています。それでは、クラス

値(16進)	クラスタの使用状態
\$00～\$97	このクラスタは使用中であり、後続するクラスタを持つ。値が後続するクラスタの番号を示している。
\$C0～\$C7	このクラスタは使用中であり、ファイル最後のクラスタである。下位4ビットがそのクラスタで実際に使われているセクタの数－1を示している。
\$FD	クラスタ中の使用セクタはない。
\$FE	予約済みのクラスタで、ファイルとして使うことはできない。(BASICコードなどのシステム領域である。
\$FF	このクラスタは未使用である。

図8-9 FATの意味

[0] (1) ' 1 '															
cs	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E +F
00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE
20	FE	FE	FE	FE	21	C6	23	24	C0	C5	27	28	29	2A	2B
30	2C	2D	C7	2F	C5	FD	32	33	C3	FF	FF	FF	FF	FF	FF
40	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
cs	1C	1C	B6	1E	B4	0E	E8	46	D7	AF	B4	16	17	18	19 1A AE

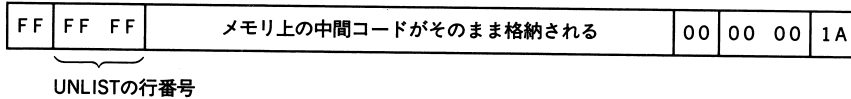
図8-10 FATの内容

タ 32 (\$20) の FAT を見て下さい。\$21 という値が入っています。これはデータがクラスタ 32 に入り切れず、クラスタ 33 (\$21) に続いていることを示します。クラスタ 33 の FAT には、\$C6 と いう値が入っています。つまり RUN というファイルは、クラスタ 32 の 16 セクタとクラスタ 33 の 7 セクタを使って終わっていることを示しています。

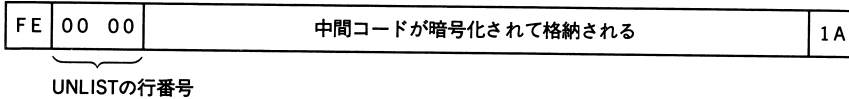
ここで TEXT というファイルを KILL してみます。そのときのディレクトリと FAT の内容を 図 8-11 に示します。ファイルを KILL すると、ディレクトリのファイル名の先頭 1 文字が、\$00 に変更されます。そしてそのファイルが占めていたクラスタに対する FAT が、\$FF に変更され

[0] (1) ' 1 '															
cs	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E +F
00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE	FE
20	FE	FE	FE	FE	21	C6	FF	FF	FF	C5	27	28	29	2A	2B
30	2C	2D	C7	2F	C5	FD	32	33	C3	FF	FF	FF	FF	FF	FF
40	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
cs	1C	1C	B6	1E	B4	0E	E8	22	B2	EE	B4	16	17	18	19 1A A4

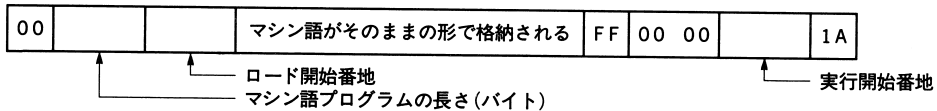
・BASICソース(バイナリ・ノーマル)



・BASICソース(バイナリ・Pオプション)



・マシン語



・アスキーファイル(ソース・データ)

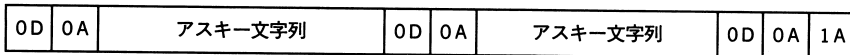


図8-13 ファイルの形式

8-1-8 セクタシーケンス

トラック上でのセクタ番号の物理的な並び方をセクタシーケンスと言いますが、これを、適当に設定することによりディスクアクセスの処理効率をあげることができます。

ディスクの回転速度は一定ですから、1セクタを読み書きする時間は決まっています。しかし読み書きのための回転待ち時間は、そのときによって大きく異なります。

あるトラックの第1セクタから第3セクタを続けて読む場合を例にとって考えます。ディスクのヘッドが第1セクタを読み取った後、第2セクタを読みに行くまでの間にはいろいろな処理が必要です。ですからセクタが順に並んでいる場合には、第2セクタを読み取ろうとしたときにはすでに第2セクタの先頭がヘッドを行き過ぎてしまい、第2セクタがぐるっとトラックを1周してくるまで待つ必要があります。

一般に番号の連続したセクタをまとめて読み書きすることが多くなりますが、適当な間隔をおいてセクタを並べてやると、この待ち時間を最小にすることができます。ディスクの読み書きのための処理にかかる時間は、システムや処理内容によりまちまちですから、それぞれに合わせたセクタシーケンスを選ぶ必要があります。

F-BASIC V3.3の"SYSDSK"を実行すると、あらかじめ変則的なセクタシーケンス(スキュー)のかかったフォーマットを行なってくれます。しかしこれは、必ずしも最適なスキューとはいえません。ところがBASICのIPLはこのスキューに合わせてBASIC本体をロードしていますので、システム領域のトラックはこのセクタシーケンスにしておかないと逆に遅くなってしまいます。またV3.0でも同じことがいえ、システム領域のトラックは、スキューがかかっていない状態で最適になります。

それでは最後に“SYSDSK”によって作られるセクタシーケンスと、ユーザー領域に最適なセクタシーケンスを示します。図 8-14 が V3.0 用で、図 8-15 が V3.3 用です。

なお、いろいろなセクタシーケンスを指定してフォーマットするプログラムを、ユーティリティの項でご紹介します。

“SYSDSK”によって作られるセクタシーケンス															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

ユーザー領域に最適なセクタシーケンス															
13	9	5	1	14	10	6	2	15	11	7	3	16	12	8	4

図8-14 セクタシーケンス(V3.0)

“SYSDSK”によって作られるセクタシーケンス															
1	12	7	2	13	8	3	14	9	4	15	10	5	16	11	6

ユーザー領域、ディレクトリに最適なセクタシーケンス															
9	1	10	2	11	3	12	4	13	5	14	6	15	7	16	8

図8-15 セクタシーケンス(V3.3)

8-2 フロッピーに対する BIOS

8-2-1 BIOS

(1) RESTORE

指定されたドライブの状態をチェックし、ヘッドをトラック 0 に戻します。初めてディスクをアクセスするときや、エラー時のリトライ処理に先立って実行します(図 8-16)。

(2) DREAD

指定された 1 セクタのデータを読み取り、データバッファに格納します。セクタの長さは、フロッピーディスクに記録されている長さに従いますので、256、512、1024 バイトのいずれかになります。通常は 256 バイトです。

(3) DWRITE

データバッファの値を、指定された 1 セクタに書き込みます。セクタの長さは、DREAD 同様にディスクのフォーマットに従います。

(4) SEEK5

フロッピーディスクドライブのヘッドを、指定したトラック上へ移動します。この動作は、シークトラックと呼ばれます。なおこの BIOS は、V3.3 でしかサポートされていません。

[RESTOR]

相対値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	8	
1	エラーステータス	RCBSTA		○
2～6	リザーブ	—		
7	ドライブ番号	RCBUNT	○	

[DWRITE]

相対値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	9	
1	エラーステータス	RCBSTA		○
2,3	データバッファ先頭アドレス	RCBDBA	○	
4	トラック番号 (0～39)	RCBTRK	○	
5	セクタ番号 (1～16)	RCBSCT	○	
6	サイド番号 (0, 1)	RCBSID	○	
7	ドライブ番号 (0～3)	RCBUNT	○	

[DREAD]

相対値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	10	
1	エラーステータス	RCBSTA		○
2,3	データバッファ先頭アドレス	RCBDBA	○	
4	トラック番号 (0～39)	RCBTRK	○	
5	セクタ番号 (1～16)	RCBSCT	○	
6	サイド番号 (0, 1)	RCBSID	○	
7	ドライブ番号 (0～3)	RCBUNT	○	

[SEEK5]

相対値	内 容	ラベル名	ユーザー	BIOS
0	リクエスト番号	RQNO	36	
1	エラーステータス	RCBSTA		○
2,3	リザーブ	—		
4	トラック番号 (0～39)	RCBTRK	○	
5,6	リザーブ	—		
7	ドライブ番号 (0～3)	RCBUNT	○	

図8-16 フロッピーに対するBIOSのRCB

8-2-2 BOOT ROM

FM-7 モードでは、BIOS の実際の処理を\$FE00～\$FFEF の BOOT ROM で行なっています。裏 RAM を選択したときには CPU 空間に BIOS が存在しなくなりますので、BOOT ROM を利用することになります。

F-BASIC V3.3 では、システム起動時にイニシエータ ROM から BOOT プログラムが、\$FE00～\$FFDF の RAM 領域に転送され利用されます。しかしこの内容は、BASIC 起動時に消されてしまいます。V3.3 では BIOS が常に使用可能ですので、BIOS を使うようにすればよいでしょう。

これらのブートプログラムのディスクアクセスルーチンと BIOS との相違点は、次のとおりです。

① エントリアドレス

RESTORE …\$FE02

DREAD ……\$FE08

DWRITE ……\$FE05

② レジスタはすべて保存されない。

③ DP レジスタに\$FD をセットしなければならない(ただし、F-BASIC に戻るときには\$00 に戻しておく必要がある)。

④ エラーステータスは RCB に書き込まれず、A レジスタにセットされる。

以上の点を除けば、X レジスタ、RCB パラメータ、そしてデータの読み書き動作はまったく同じです。

8-3 FDC について

FM-7 シリーズでは、フロッピーディスクコントロール用に MB8877A という LSI を採用しています。この MB8877A は、CPU6809 とフロッピーディスクとの間に立ち、CPU からの指定された動作をフロッピーディスクに対し制御し、その結果を CPU へ応答します。CPU からの指定は、“コマンド”として FDC に送られます。FDC は、このコマンドに従って動作を行ない、動作結果(ステータス)を CPU に返して終了します。このとき、割り込み要求(IRQ)も出力します(図 8-17)。

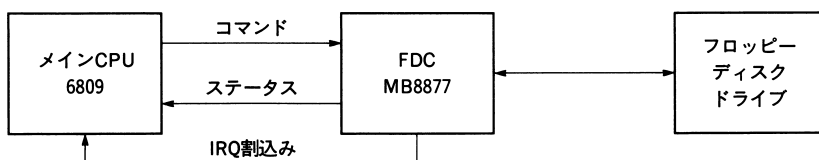


図8-17 FDCの位置

8-3-1 FDC レジスタ

FDC はコマンド実行およびステータス表示のために 5 個のレジスタをもっていて、図 8-18 に示すようにメインシステム I/O レジスタに割り当てられています。

- ① コマンドレジスタ……………FDC へのコマンドを書き込むためのレジスタです。コマンドの書き込みはフォースインタラプトコマンドを除いて、前のコマンドが終了してからでなければなりません。
- ② ステータスレジスタ……………フロッピーディスクドライブおよび FDC の状態(ステータス)を示します。実行のコマンドタイプによって内容が異なります。
- ③ トラックレジスタ……………データの書き込みおよび読み出しを行なうときに対象とするセクタの含まれるトラック番号および、ドライブのヘッドを移動するときの目的とするトラックの番号を指定します。また、このレジスタを読み出すことによってヘッドがどのトラックに位置するかを知ることができます。
- ④ セクタレジスタ……………データの書き込みおよび、読み出しを行なうときに、対象とするセクタ番号を指定します。
- ⑤ データレジスタ……………フロッピーディスクから読み出したデータおよび、フロッピーディスクへ書き込むデータを読み書きするレジスタです。このレジスタへの読み書きは、DRQ フラグの指示に従って行なわなければなりません。

FDC には、その外付け回路により以下の信号をサポートしています。メイン CPU は、それらを FDC の機能として利用できます。

- ① SIDE ……………フロッピーディスクの面(サイド)を指定します。
- ② IRQ……………このビットが ON になっていると、FDC からの IRQ(割り込み要求)が発生していることを示します。
- ③ DRQ ……………このビットが ON になっていると、FDC からの DRQ(データ転送要求)が発生していることを示します。
- ④ ドライブ番号……………フロッピーディスクドライブを選択するためのビットです。
- ⑤ ドライブディセーブル……………このビットを ON にすると、ドライブ番号の設定にかかわらずすべてのドライブが非選択状態になります。FM-7 には、この機能はサポートされていません。
- ⑥ モーターオンフラグ……………フロッピーディスクドライブのモーターの ON/OFF を制御します。このビットを ON にするとモーターが回転し、OFF にするとモーターが停止します。

アドレス	名 称	R/W	内 容
\$FD 18	ステータスレジスタ	R	ステータス
	コマンドレジスタ	W	コマンド
\$FD19	トラックレジスタ	RW	トラック番号
\$FD1A	セクタレジスタ	RW	セクタ番号
\$FD1B	データレジスタ	RW	リード/ライトのためのデータ
\$FD1C	ヘッドレジスタ	RW	ビット0 : サイド番号
\$FD1F		R	ビット6 : IRQ ビット7 : DRQ
\$FD1D	ドライブレジスタ	RW	ビット0, 1 : ドライブ番号 ビット6 : ドライブディセーブル ビット7 : モーターオンフラグ

図8-18 FDC I/Oアドレス

8-3-2 FDC インターフェース

(1) ハードウェアタイマー

FDC に対してリード/ライト動作コマンドを実行させるとき、以下の条件のときは一定時間、MB8877A のリード/ライト動作が開始できなくなっています。

- ① モーターを ON にしたとき 1 sec
- ② HEAD LOAD したとき 60 msec
- ③ ドライブレジスタに書き込みをしたとき 60 msec
- ④ STEP 動作(ヘッドの移動)をしたとき 60 msec

(2) CPU への割り込み

MB8877A はリセット時、コマンド実行終了時、タイプIVコマンド実行時に CPU への IRQ 割り込みをかけます。この割り込みは\$FD02 のビット 4 をオフにすることでマスク可能です。

(3) データ転送

MB8877A からの DRQ(データ転送要求)に従って、ソフトウェアでデータの転送を行ないます。リード/ライトアクセス中は、32 μ sec 間隔で DRQ がオンになるので、他の割り込みはマスクする必要があります。

8-3-3 FDC コマンド

FDC には 11 種類のコマンドがあり、図 8-19 のように 4 つのタイプに分類されています。

タイプ	コマンド	動作
I	リストア	トラック 0 にヘッドを移動する
	シーク	データレジスタに書き込んだトラックへヘッドを移動する
	ステップ	直前に動いた向きにヘッドを 1 トラック分移動する
	ステップイン	ヘッドを 1 トラック内側へ移動する
	ステップアウト	ヘッドを 1 トラック外側へ移動する
II	リードデータ	1 セクタ分のデータを読み出す
	ライトデータ	1 セクタ分のデータを書き込む
III	リードアドレス	ディスクの ID フィールドを読み出す
	リードトラック	ディスクの 1 トラック分のデータをすべて読み出す
	ライトトラック	ディスクへ 1 トラック分のデータをすべて書き込む
IV	フォースインラプト	実行中のコマンドを中断させ、IRQ を発生させる

図8-19 FDCコマンド一覧

(1) タイプ I コマンド

タイプ I コマンドは、ヘッドの移動とトラックの照合を行いません。コマンドレジスタに書き込むコマンドは、図 8-20 に示すとおりです。

コマンド	コマンドコード								ビット
	7	6	5	4	3	2	1	0	
リストア	0	0	0	0	h	v	r1	r0	
シーク	0	0	0	1	h	v	r1	r0	
ステップ	0	0	1	u	h	v	r1	r0	
ステップイン	0	1	0	u	h	v	r1	r0	
ステップアウト	0	1	1	u	h	v	r1	r0	

u: アップデートフラグ
h: ヘッドロードフラグ
v: ベリファイフラグ
r1, r0: ステッピングレート

図8-20 タイプIコマンド表

- ① アップデートフラグ……………ON のときは移動後のヘッド位置が、トラックレジスタに書き込まれます。
- ② ヘッドロードフラグ……………コマンド実行時にヘッドをフロッピーディスクに押し付ける (LOAD) か、離す (OFF) かを指定します。ON のときヘッドロードとなります。
- ③ ベリファイフラグ……………コマンド実行後に現在のヘッド位置のトラック番号とトラックレジスタの内容との照合を行なうかどうかを指定します。ON のとき照合が行なわれます。
- ④ ステッピングレート……………ヘッドを移動するときの移動時間間隔を指定します。通常は 6msec を選択します (図 8-21)。

r1	r0	ステッピングレート
0	0	6msec
0	1	12msec
1	0	20msec
1	1	30msec

図8-21 ステッピングレート

(2) タイプII コマンド

タイプII コマンドは、ディスクへのセクタ単位のデータの読み書きを行ないます。

コマンドレジスタに書き込むコマンドは、図 8-22 に示すとおりです。なおコマンドレジスタに書き込むに先立って、トラックレジスタとセクタレジスタに読み書きするセクタのトラック番号とセクタ番号を書き込んでおく必要があります。

コマンド	コマンドコード								ビット
	7	6	5	4	3	2	1	0	
リードデータ	1	0	0	m	S	E	C	0	
ライトデータ	1	0	1	m	S	E	C	a	

m : マルチセクタフラグ
S : サイドフラグ
E : ディレイフラグ
C : チェックフラグ
a : アドレスマークフラグ

図8-22 タイプII コマンドコード表

- ① マルチセクタフラグ……………このフラグが ON のときには、データの読み書きと共にセクタレジスタが順次 1 ずつ増やされ、その値のセクタが見つからなくなるまで連続して読み書きされます。つまり通常の 1 トラックを丸ごと読み書きできるわけです。
- ② サイドフラグ……………ディスクのサイド番号の比較を行なうとき、ディスクの面(サイド)を指定します。チェックフラグが OFF のときには無効です。
- ③ ディレイフラグ……………ディスクへの読み書きに先立ってヘッドの状態を調べるのに、約 30msec 待つかどうかを指定します。
- ④ チェックフラグ……………ディスクへの読み書きに先立って、ディスクのサイド番号の比較を行なうかどうかを指定します。
- ⑤ アドレスマークフラグ……………ディスクのフォーマット時、データフィールドの始まりを示すためデータマーク(\$FB)が書き込まれますが、セクタのデータを使用禁止にしたいときにはデリーテッドマーク(\$F8)を書いておきます。ライトデータコマンド時、このフラグを ON にするとデリーテッドマークが、OFF にするとデータマークが書き込まれます。

(3) タイプⅢコマンド

タイプⅢコマンドは、ディスクへのトラック単位のデータの読み書きおよび、ディスク ID フィールドの読み出しを行ないます(図 8-23, 図 8-24)。

コマンド	コマンドコード								ビット
	7	6	5	4	3	2	1	0	
リードアドレス	1	1	0	0	0	E	0	0	
リードトラック	1	1	1	0	0	E	0	0	
ライトトラック	1	1	1	1	0	E	0	0	

E : ディレイフラグ

図8-23 タイプⅢコマンドコード表

① ディレイフラグ……タイプⅡコマンドのディレイフラグと同様です。

リードアドレスコマンドでは、一番最初に発見した ID フィールドの内容(6 バイト)が、順次データレジスタに読み出されます。

- ① トラック番号
- ② ディスクサイド番号
- ③ セクタ番号
- ④ セクタ長
- ⑤ CRC1
- ⑥ CRC2

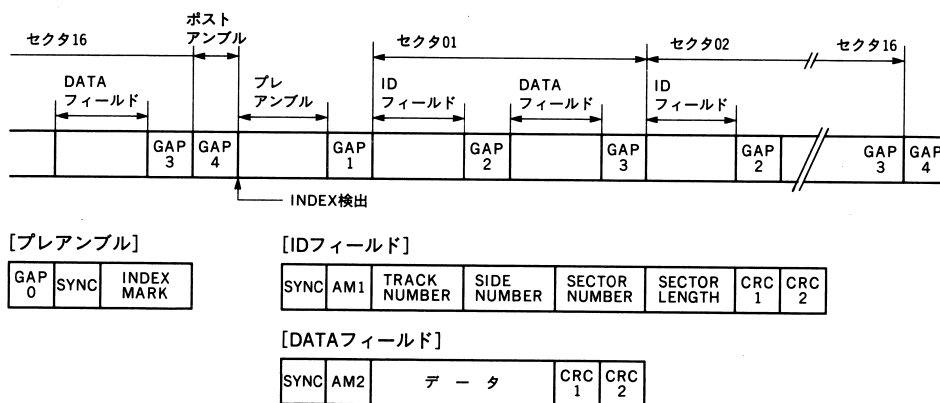


図8-24 フロッピーディスクの物理フォーマット

リードトラックコマンドは、ハードウェアの制約でその動作結果は保障できません。タイプⅡのマルチセクタリードで代用します。

ライトトラックコマンドは、1トラックの内容を ID フィールドやギャップデータを含めてすべてを書き込むためのコマンドです。フロッピーディスクを初期化(フォーマット)するために用います。

ライトトラックコマンドでは、データレジスタに書き込む内容に特別な意味のあるものがあります。図 8-25 に示します。

データレジスタの値	ディスクへ出力される値	データの意味
\$F5	\$A1	IDアドレスマーク, DATAアドレスマークの前提データ
\$F6	\$C2	INDEXマークの前提データ
\$F7	2バイトのCRC	内部で作成された2バイトのCRC
\$F8	\$F8	DELETED DATA MARK (AM2)
\$FB	\$FB	DATA MARK (AM2)
\$FC	\$FC	INDEXマーク
\$FE	\$FE	IDアドレスマーク (AM1)

コード	
GAP0	80バイトの\$4E
GAP1	50バイトの\$4E
GAP2	22バイトの\$4E
GAP3	(54, 84, 116)バイトの\$4E
GAP4	(598, 400, 654)バイトの\$4E
SYNC	12バイトの\$00
INDEXマーク	3バイトの\$C2 + 1バイトの\$FC
AM1	3バイトの\$A1 + 1バイトの\$FE
AM2	3バイトの\$A1 + 1バイトの\$FBまたは\$F8

(注) GAP3, GAP4はセクタ長(256, 512, 1024)に対応

図8-25 ライトトラックコマンドでのデータの意味

(4) タイプIVコマンド

現在実行中のコマンドを中断させるフォースインタラプトコマンドです(図 8-26)。

コマンド	コマンドコード								ビット
	7	6	5	4	3	2	1	0	
フォースインタラプト	1	1	0	1	I3	I2	I1	I0	

I0 ~ I3 : IRQ発生条件

図8-26 タイプIVコマンドコード表

① I0 ……このビットが ON だと、フロッピーディスクドライブからの READY 入力の立上りにて IRQ を発生します。

- ② I1 ……このビットがON だと、フロッピーディスクドライブからの READY 入力の立下りにて IRQ を発生します。
- ③ I2 ……このビットがON だと、インデックスパルス (INDEX 発見) にて IRQ を発生します。
- ④ I3 ……このビットがON だと、ただちに IRQ を発生します。

ただし、I0～I3 がすべて OFF だと IRQ は発生しません。

8-3-4 FDC ステータス

FDC には、コマンドの実行結果、実行中の状態を示すステータスレジスタがあります。ステータスレジスタの各ビットの内容は、実行されるコマンドによって異なります。コマンドが実行されると所要のビットがプリセットされ、コマンド実行中に更新され、コマンド実行終了時にすべてのビットの内容が確定します。

以下に各コマンドタイプ別のステータスレジスタの内容の一覧を示します(図 8-27)。

タイプ I コマンド

ビット \ コマンド	すべてのタイプ I コマンド
ビット 0	BUSY
ビット 1	INDEX
ビット 2	TRACK00
ビット 3	CRC ERROR
ビット 4	SEEK ERROR
ビット 5	HEAD ENGAGED
ビット 6	WRITE PROTECT
ビット 7	NOT READY

タイプ II コマンド

ビット \ コマンド	リードデータ	ライトデータ
ビット 0	BUSY	BUSY
ビット 1	DATA REQUEST	DATA REQUEST
ビット 2	LOST DATA	LOST DATA
ビット 3	CRC ERROR	CRC ERROR
ビット 4	RECORD NOT FOUND	RECORD NOT FOUND
ビット 5	RECORD TYPE	WRITE FAULT
ビット 6	0	WRITE PROTECT
ビット 7	NOT READY	NOT READY

タイプⅢコマンド

ビット \ コマンド	リードアドレス	ライトトラック
ビット0	BUSY	BUSY
ビット1	DATA REQUEST	DATA REQUEST
ビット2	LOST DATA	LOST DATA
ビット3	CRC ERROR	0
ビット4	RECORD NOT FOUND	0
ビット5	0	WRITE FAULT
ビット6	0	WRITE PROTECT
ビット7	NOT READY	NOT READY

※リードトラックコマンドは使用できないため省略してあります。

タイプⅣコマンド

ビット \ コマンド	コマンド実行中のとき	コマンド実行中でないとき
ビット0	実行中のコマンドのステータスレジスタと同様の内容	0
ビット1		INDEX
ビット2		TRACK 00
ビット3		0
ビット4		0
ビット5		HEAD ENGAGED
ビット6		WRITE PROTECT
ビット7		NOT READY

図8-27 ステータスレジスタの内容

それでは最後に FDC を直接制御して、フロッピーディスクへのセクタリード／ライトするサンプルプログラムを紹介します(リスト 8-1)。\$5000 番地を実行すると、0 トラック 4 セクタの内容を \$5200 番地以降に読み出します。そして、\$5003 番地を実行すると \$5300 番地から 256 バイトの内容を 0 トラック 4 セクタに書き込みます。

リスト 8-1 FDC サンプルプログラム

01000				*****
01010				* FDC TEST PROGRA *
01020				* (LIST 8-1) V3.3 *
01030				*****
01040				OPT NOGEN
01050	5000			ORG \$5000
01060		00FD		SETDP \$FD
01070	5000 7E	50A1	ENTRY JMP	DREAD READ ENTRY
01080	5003 7E	50B7	JMP	DWRITE WRITE ENTRY
01090		FD18	FDCCMD EQU	\$FD18 FDC コメント レジスタ
01100		FD18	FDCSTA EQU	\$FD18 FDC ステータス レジスタ
01110		FD19	FDCTRK EQU	\$FD19 FDC トラック レジスタ
01120		FD1A	FDCSCT EQU	\$FD1A FDC セクタ レジスタ
01130		FD1B	FDCDAT EQU	\$FD1B FDC データ レジスタ
01140		FD1C	FDCSID EQU	\$FD1C FDC ヘッド レジスタ

01150		FD1F	FDCDRQ	EQU	\$FD1F	FDC テーマ リクエスト	
01160		FD03	BUZZER	EQU	\$FD03	BUZZER ホート	
01170		FBFA	BIOS	EQU	\$FBFA	BIOS ヘクトル	
01180	5006	00	TRKNO	FCB	0	トラック ハンコウ	
01190	5007	04	SCTNO	FCB	4	セクタ ハンコウ	
01200	5008	00	SIDNO	FCB	0	サイト ハンコウ	
01210	5009	5200	DATA_R	FDB	\$5200	ヨミタシ アドレス	
01220	5008	5300	DATA_W	FDB	\$5300	カキコミ アドレス	
01230	5000	0000	STKSAV	FDB	0	スタック ポゾン	
01240	500F	08	RCB	FCB	8.0.0.0	RESTORE RCB	
01250	5013	00		FCB	0.0.0.0	DRV=0	
01260							
01270	5017 B6	5006	SEEK	LDA	TRKNO	モクテキ トラック NO	
01280	501A 97	18		STA	FDCDAT		
01290	501C 86	1A		LDA	#\$1A	SEEK S.RATE=20ms	
01300	501E 97	18		STA	FDCCMD		
01310	5020 D6	1F	SEEK_1	LDB	FDCDRQ		
01320	5022 C5	40		BITB	#\$40		
01330	5024 26	06	502C	BNE	SEEK_2		
01340	5026 4F			CLRA			
01350	5027 4A			DECA		シカン マチ	
01360	5028 26	FD	5027	BNE	*-1		
01370	502A 20	F4	5020	BRA	SEEK_1	クリカエシ	
01380	502C 96	18	SEEK_2	LDA	FDCSTA	エラー ステータス チェック	
01390	502E 84	91		ANDA	#\$91		
01400	5030 26	49	507B	BNE	ERROR		
01410	5032 39			RTS			
01420							
01430	5033 B6	5008	* << 1セクタ READ >>	READ	LDA	SIDNO	サイト NO SET
01440	5036 97	1C			STA	FDCSID	
01450	5038 B6	5007		LDA	SCTNO	セクタ NO SET	
01460	5038 97	1A		STA	FDCSCT		
01470	503D 86	80		LDA	#\$80	READ S.RATE=20ms	
01480	503F 97	18		STA	FDCCMD		
01490	5041 BE	5009		LDX	DATA_R	ヨミタシ アドレス SET	
01500	5044 D6	1F	READ_1	LDB	FDCDRQ	テーマ リクエスト ?	
01510	5046 2A	06	504E	BPL	READ_2	NO.	
01520	5048 96	18		LDA	FDCDAT		
01530	504A A7	80		STA	.X+		
01540	504C 20	F6	5044	BRA	READ_1		
01550	504E C5	40	READ_2	BITB	#\$40	コメント シュウリョウ ?	
01560	5050 27	F2	5044	BEQ	READ_1	NO	
01570	5052 96	18		LDA	FDCSTA	エラー ステータス チェック	
01580	5054 26	25	507B	BNE	ERROR		
01590	5056 39			RTS			
01600							
01610	5057 B6	5008	* << 1セクタ WRITE >>	WRITE	LDA	SIDNO	サイト NO SET
01620	505A 97	1C			STA	FDCSID	
01630	505C B6	5007		LDA	SCTNO	セクタ NO SET	
01640	505F 97	1A		STA	FDCSCT		
01650	5061 86	A0		LDA	#\$A0	WRITE S.RATE=20ms	
01660	5063 97	18		STA	FDCCMD		
01670	5065 BE	5008		LDX	DATA_W	テーマ アドレス SET	
01680	5068 A6	80	WRIT_1	LDA	.X+		
01690	506A D6	1F	WRIT_2	LDB	FDCDRQ	テーマ リクエスト	
01700	506C 2A	04	5072	BPL	WRIT_3	NO.	
01710	506E 97	18		STA	FDCDAT	1バイト カキコミ	
01720	5070 20	F6	5068	BRA	WRIT_1		
01730	5072 C5	40	WRIT_3	BITB	#\$40	コメント シュウリョウ	
01740	5074 27	F4	506A	BEQ	WRIT_2	NO.	
01750	5076 96	18		LDA	FDCSTA	エラー ステータス チェック	
01760	5078 26	01	507B	BNE	ERROR		
01770	507A 39			RTS			
01780	507B C6	05	ERROR	LDB	#\$5	スカイ フォン ヲ ナラス	
01790	507D 86	81	ERR_1	LDA	#\$81		
01800	507F 97	03		STA	BUZZER	BEEP1	
01810	5081 8E	0000		LDX	#0		
01820	5084 30	1F		LEAX	-1,X		

```

01830 5086 26 FC 5084 BNE *-2
01840 5088 0F 03 CLR BUZZER BEEP 0
01850 508A 30 1F LEAX -1,X
01860 508C 26 FC 508A BNE *-2
01870 508E 5A DECB
01880 508F 26 EC 507D BNE ERR_1
01890 5091 10FE 500D LDS STKSAV スタック フック
01900 5095 35 89 PULS DP,CC,PC
01910 * << RESTORE >>
01920 5097 8E 500F RESTOR LDX #RCB DRIVE 0 RESTORE
01930 509A AD 9F FBFA JSR [BIOS]
01940 509E 25 DB 507B BCS ERROR
01950 50A0 39 RTS
01960 * << READ ENTRY >>
01970 50A1 34 09 DREAD PSHS DP,CC
01980 50A3 10FF 500D STS STKSAV スタック ホソ"ン
01990 50A7 1A 50 DRCC #50
02000 50A9 86 FD LDA #5FD
02010 50AB 1F 8B TFR A,DP
02020 50AD 8D E8 5097 BSR RESTOR リストア
02030 50AF 17 FF65 5017 LBSR SEEK ヘッド" シーク
02040 50B2 17 FF7E 5033 LBSR READ 1セクタ ヨミトリ
02050 50B5 35 89 PULS DP,CC,PC
02060 * << WRITE ENTRY >>
02070 50B7 34 09 DWRITE PSHS DP,CC
02080 50B9 10FF 500D STS STKSAV スタック ホソ"ン
02090 50BD 1A 50 DRCC #50
02100 50BF 86 FD LDA #5FD
02110 50C1 1F 8B TFR A,DP
02120 50C3 8D D2 5097 BSR RESTOR リストア
02130 50C5 17 FF4F 5017 LBSR SEEK ヘッド" シーク
02140 50C8 8D 8D 5057 BSR WRITE 1セクタ カキコミ
02150 50CA 35 89 PULS DP,CC,PC
02160 *
02170 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=50CB
PROGRAM ENTRY ADDR=5000

```

8-4 ディスクユーティリティ

ディスクに関する応用として、いくつかのディスクユーティリティを紹介します。その使用方法、テクニックについて筆足らずの部分が多いかと思いますが、ソースリストから判断してください。

8-4-1 セクタダンプ

F-BASIC フォーマットのフロッピーディスクのセクタダンプです(図 8-28, リスト 8-2)。

プログラムを起動すると、ドライブ 0, トラック 0, セクタ 1 の内容を表示して、以下に示すコマンドの入力待ちとなります。

	 1セクタ進める
	 1セクタ戻す
+	 8セクタ進める
+	 8セクタ戻す
	 1トラック進める
	 1トラック戻す
+	 5トラック進める
+	 5トラック戻す
	 トラック 0, セクタ 1 へ戻す
, , ,	 ドライブの選択
	 画面のハードコピー
	 終了

[0] (0) 「 15 」																cs 0123456789ABCDEF																		
cs	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	cs	0	1	2	3	4	5	6	7	8	9	A	B					
00	7E	6E	A9	04	41	55	54	4F	05	4C	49	53	54	0D	04	52	76	~n AUTO LIST R								91	UN CONT LLIST							
10	55	4E	0D	05	43	4F	4E	54	0D	06	4C	4C	49	53	54	0D	7D	LOAD" SAVE" FIL								78	ES SCREEN 7,7 H							
20	05	4C	4F	41	44	22	05	53	41	56	45	22	05	46	49	4C	27	ARDC								7B								
30	45	53	0B	53	43	52	45	4E	20	37	2C	37	0D	06	48																			
40	41	52	44	43	0D	00	00	00	00	00	00	00	00	00	00	00																		
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																		
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																		
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																		
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																		
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																		
A0	00	00	00	38	34	30	35	30	35	34	40	C6	29	8E	6E	B9	4E	8405054@二)■h7																
B0	CE	FC	00	BD	85	97	7E	6F	15	C6	C7	E1	C4	27	02	6C	6C	ホ ス ■ r o ニ ヌ ト ' 1																
C0	C4	4D	7E	79	83	A6	24	AE	2D	34	12	86	07	A7	24	CC	9A	TM~y=7\$ヨ-4 ■ 7\$7																
D0	00	FF	ED	2D	BD	79	90	35	14	E7	24	AF	2D	A7	C4	7E	F8	O-スy+5 7\$y-7t~																
E0	79	89	B6	72	13	34	02	8E	72	13	BF	71	FC	CC	00	02	80	yl カ r 4 ■ r ヲ q7																
F0	F7	71	FF	B7	72	02	86	0D	B7	72	00	7F	71	FE	BD	74	6D	7q 7r ■ 7rqst																
cs	60	EF	74	A4	96	34	DB	58	55	62	0D	B9	67	80	BC	DB	5C																	

図8-28 セクタダンプ例

リスト 8-2 セクタダンプ

```

1000 *****
1020 *      Sector Dump      *
1030 *      ( LIST 8-2 )  V3.0  *
1040 *****
1080 CLEAR 3000,&H6000:DEFINT A-Z
1090 DEFINT A-Z
1100 DEFFNC(PAGE)=(PAGE+1280)MOD1280
1110 WIDTH 80,25:CONSOLE...1:COLOR=(4,7)
1120 GOSUB 1500
1130 CMD$=CHR$(27)+CHR$(17)+CHR$(11)+CHR$(28)+CHR$(29)+CHR$(30)+CHR$(3
1)+CHR$( 6)+CHR$( 2)+CHR$(25)+CHR$(26)+"0123"

```

```

1140 LOCATE 0,2:PRINT "os  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +
E +F  cs  0123456789ABCDEF
1150 DRV=0:PAGE=0:CMD=1
1170 WHILE CMD>0
1180   GOSUB 1430
1190   CMD=1
1200   WHILE CMD=1
1210     CMD=-1:WHILE CMD=-1:CMD=INSTR(CMD$,INPUT$(1))-1:WEND
1220     ON CMD GOSUB 1270,1280,1290,1300,1310,1320,1330,1340,1350,136
0,
1230     WEND
1240   WEND
1250 CONSOLE...,0:COLOR=(4,4):END
1270 HARDC:RETURN
1280 PAGE=0:RETURN
1290 PAGE=FNC(PAGE+32):RETURN
1300 PAGE=FNC(PAGE-32):RETURN
1310 PAGE=FNC(PAGE-1):RETURN
1320 PAGE=FNC(PAGE+1):RETURN
1330 PAGE=FNC(PAGE+5*32):RETURN
1340 PAGE=FNC(PAGE-5*32):RETURN
1350 PAGE=FNC(PAGE-8):RETURN
1360 PAGE=FNC(PAGE+8):RETURN
1370 DRV=0:RETURN
1380 DRV=1:RETURN
1390 DRV=2:RETURN
1400 DRV=3:RETURN
1430 TRK=PAGE *32:SCT=PAGE MOD32+1
1450 LOCATE 4,0:PRINT USING "[ # ] ( ## ) 「 ## 」";DRV,TRK,SCT
1460 EXEC &H6000:DRV,TRK,SCT
1470 RETURN
1500 LOCATE 4,0:PRINT "# Wait a moment.
1510 READ M$:A=&H6000
1520 WHILE M$>"/":POKE A,VAL("&H"+M$):A=A+1:READ M$:WEND
1530 RETURN
1550 DATA BD,99,BA,34,04,BD,99,FD,34,04,BD,99,FD,34,04,BD
1560 DATA 9F,5C,30,8D,00,FF,5F,35,02,81,11,25,03,80,10,5C
1570 DATA ED,05,35,06,A7,04,E7,07,31,8D,01,52,10,AF,02,86
1580 DATA 0A,A7,84,AD,9F,FB,FA,CC,00,04,BD,DA,6D,30,8D,00
1590 DATA 0D,CC,00,10,A7,80,5A,26,FB,34,02,30,8D,00,0F,17
1600 DATA 00,AC,86,20,A7,80,6F,8D,00,C3,33,8D,00,C0,C6,10
1610 DATA 34,04,C6,20,E7,80,A6,A4,AB,8D,00,B1,A7,8D,00,AD
1620 DATA A6,A4,AB,C4,A7,C0,A6,A0,17,00,83,6A,E4,26,E5,32
1630 DATA 61,E7,80,E7,80,A6,8D,00,94,8D,73,E7,80,E7,80,31
1640 DATA 30,C6,10,A6,A0,26,02,86,20,A7,80,5A,26,F5,E7,84
1650 DATA 34,20,86,15,8D,0B,E4,30,8D,00,83,8D,09,0F,86,16
1660 DATA BD,0B,E4,8D,9B,50,35,20,35,02,8B,10,24,8B,8D,9B
1670 DATA 50,30,8C,6A,CC,63,73,ED,81,CC,10,20,E7,80,6F,8C
1680 DATA 4C,34,02,33,8C,48,E7,80,A6,C4,AB,8C,40,A7,8C,3D
1690 DATA A6,C0,8D,1A,6A,E4,26,EE,32,61,E7,80,E7,80,A6,8C
1700 DATA 2C,8D,0B,6F,84,30,8C,36,8D,09,0F,7E,9B,50,34,02
1710 DATA 44,44,44,44,8D,04,35,02,84,0F,81,0A,25,02,8B,07
1720 DATA 8B,30,A7,80,39,./

```

8-4-2 アスキーダンプ

セクタダンプでは16進表示とキャラクタ表示の両方が行なわれていますが、キャラクタ部分だけを表示するのがこのアスキーダンプです。使い方はセクタダンプと同じです(図8-29, リスト8-3)。

このプログラムではオーダー動作のOFFを、BASICのコンソール制御ルーチン(\$DBD5)を使って行なっています。


```

[ 0 ] ( 2 ) 「 1 」
* * * * * L * * * * *
|| n : || Sector Dump || ± : || Ver 1.2 * 198
6,5,10 || イ : || from T'N'T utys. || ↑ : ||
|| * * * * * || * * * * * 8シ ■

[ 0 ] ( 2 ) 「 9 」
D,9B,50,35,20,35,02,8B,10,24,8B,BD,9B a 50,30,8C,6A,CC,63,73
,ED,81,CC,10,20,E7,80,6F,8C 4C,34,02,33,8C,48,E7,80,A6,C4,
AB,8C,40,A7,8C,3D ^ - A6,C0,8D,1A,6A,E4,26,EE,32,61,E7,80,E7,8
0,A6,8C 2C,8D,0B,6F,84,30,8C,36,BD,D9,0F,7E,9B,50,34,02 9

[ 0 ] ( 2 ) 「 17 」
* * * * * NAM SCTDMP ) * * * * * B : * * * * * ORG $6000 I
: * * * * * EXEC : DRV,TRK,SCT (based on F-BASIC) * * * * * I
$ : * * * * * ENTRY JSR $99BA DRV テ 8 : * * * * * PSHS B
B : * * * * * JSR $99FD TRK L : * * * * * PSHS B V : * * * * *

[ 0 ] ( 2 ) 「 25 」
X : * * * * * JSR $DBE4 order on G b : * * * * * JSR $9B50
CR.LF f 1 : * * * * * PULS Y m v : * * * * * PULS A
I : * * * * * ADDA #$10 イ ~ : * * * * * BCC LOOP ケ ' : * * * * *
JSR $9B50 CR.LF ■ イ : * * * * * シ : * * * * * LEAX <LIN,PCR >

```

図8-29 アスキーダンプ例

リスト8-3 アスキーダンプ

```

1000 '*****
1020 '* Sector ASCII Dump *
1030 '* ( LIST 8-3 ) V3.0 *
1040 '*****
1080 CLEAR 3000.&H7000:DEFINT A-Z
1090 DEFINT A-Z
1100 DEFFNC(PAGE)=(PAGE+1280)MOD1280
1110 DEFUSR=&H0BD5
1120 WIDTH 80.25:CONSOLE...1:COLOR=(4,7)
1130 CMD$=CHR$(27)+CHR$(17)+CHR$(11)+CHR$(28)+CHR$(29)+CHR$(30)+CHR$(3
1)+
CHR$( 6)+CHR$( 2)+CHR$(25)+CHR$(26)+"0123"
1180 FIELD 0.64 AS D0$.64 AS D1$.64 AS D2$.64 AS D3$
1190 DRV=0:PAGE=0:CMD=1
1200 '
1210 WHILE CMD>0
1220 GOSUB 1490
1230 CMD=1
1240 WHILE CMD=1
1250 CMD=-1:WHILE CMD=-1:CMD=INSTR(CMD$.INPUT$(1))-1:WEND
1260 ON CMD GOSUB 1310,1320,1330,1340,1350,1360,1370,1380,1390,140
0.
1410,1420,1430,1440
1270 WEND
1280 WEND
1290 CONSOLE...0:COLOR=(4,4):END
1300 '
1310 HARDC:RETURN
1320 PAGE=0:CLS:RETURN
1330 PAGE=FNC(PAGE+32) :RETURN
1340 PAGE=FNC(PAGE-32) :RETURN
1350 PAGE=FNC(PAGE-1) :RETURN
1360 PAGE=FNC(PAGE+1) :RETURN
1370 PAGE=FNC(PAGE+5*32):RETURN
1380 PAGE=FNC(PAGE-5*32):RETURN
1390 PAGE=FNC(PAGE-8) :RETURN

```

```

1400 PAGE=FNC(PAGE+8) :RETURN
1410 DRV=0:RETURN
1420 DRV=1:RETURN
1430 DRV=2:RETURN
1440 DRV=3:RETURN
1470 :
1480 :
1490 TRK=PAGE *32:SCT=PAGE MOD32+1
1500 D$=DSKI$(DRV,TRK,SCT)
1510 PRINT "["DRV"]("TRK")("SCT")"
1520 OPEN"O",#1,"SCRN:"
1530 PRINT #1,USR(" ") D0$ USR(" ")
1540 PRINT #1,USR(" ") D1$ USR(" ")
1550 PRINT #1,USR(" ") D2$ USR(" ")
1560 PRINT #1,USR(" ") D3$ USR(" ")
1570 CLOSE #1:PRINT
1580 RETURN

```

8-4-3 ファイルインフォメーション

ディスクファイルに関する様々な情報を表示します(図8-30, リスト8-4)。

- ① ファイル名
- ② ファイルタイプ
- ③ ファイルアロケーション
- ④ 占有クラスタ数
- ⑤ 最終クラスタの使用状況
- ⑥ 総バイト数

ファイル名の指定で  のみを入力した場合は、そのドライブ上のすべてのファイルについて表示します。ファイルアロケーション情報のセクタは、A=1~8セクタ、B=9~16、C=17~24、D=25~32を示しています。

```

>> File Informations for F-BASIC

Drive #0
File Name ? FINFO

FINFO      : Binary Source
( 4 - C )
( 4 - D )
2 Clusters used.
Last Cluster : 6 Sectors & 100 Bytes
3684 Bytes all.

```

図8-30 ファイルインフォメーション表示例

リスト 8-4 ファイルインフォメーション

```

1000 *****
1020 *   File Informations   *
1040 * ( LIST 8-4 )   V3.0   *
1060 *****
1080 CLEAR 3000,&H7000:DEFINT A-Y:DIM DIR$(?)
1090 DEFFNZA(AD$)=ASC(AD$)*256+ASC(MID$(AD$,2))
1100 FIELD 0.32 AS DIR$(0),32 AS DIR$(1),32 AS DIR$(2),32 AS DIR$(3),
      32 AS DIR$(4),32 AS DIR$(5),32 AS DIR$(6),32 AS DIR$(7)
1110 FIELD 0.255 AS SCT0$,1 AS SCT1$
1120 '
1130 PRINT:PRINT ">> File Informations for F-BASIC"
1140 PRINT:PRINT"Drive #":DRV$=INKEY$
1150 WHILE DRV$="":DRV$=INKEY$:WEND
1160 IF DRV$=CHR$(27) THEN END
1170 DRV=VAL(DRV$):PRINT DRV
1180 FAT$=MID$(DSKI$(DRV,1,1),6)
1190 LINEINPUT"File Name ? ":F$
1200 IF F$="" THEN FILES OCT$(DRV)+"":GOTO 1190
1210 FD$=LEFT$(F$+"",8)
1220 IF F$="" THEN 1380
1230 '
1240 '■ One file ■
1250 OPEN "O",1,"SCRN:"
1260 DSCT=4:FLG=0
1270 WHILE FLG<255
1280   DMY$=DSKI$(DRV,1,DSCT)
1290   FOR SLT=0 TO 7
1300     DIR$=DIR$(SLT):FLG=ASC(DIR$)
1310     IF FD$=LEFT$(DIR$,8) THEN GOSUB 1540:SLT=8:FLG=255
1320   NEXT
1330   DSCT=DSCT+1
1340 WEND
1350 CLOSE 1
1360 GOTO 1140
1370 '
1380 '■ ALL files ■
1390 OPEN "O",1,"SCRN:"
1400 DSCT=4:FLG=0
1410 WHILE FLG<255
1420   FOR SLT=0 TO 7
1430     DMY$=DSKI$(DRV,1,DSCT)
1440     DIR$=DIR$(SLT):FLG=ASC(DIR$)
1450     IF FLG>0 AND FLG<255 THEN GOSUB 1540
1460     IF INKEY$="" THEN SLT=7:FLG=255
1470   NEXT
1480   DSCT=DSCT+1
1490 WEND
1500 CLOSE 1
1510 GOTO 1140
1520 '
1530 '
1540 '■ File informations ■
1550 PRINT #1:PRINT #1, LEFT$(DIR$,8) " " : " :
1560 FT=3
1570 ON ASC(MID$(DIR$,12))+1 GOTO 1590,1610,1630
1580 PRINT #1,"???":GOTO 1650
1590 IF ASC(MID$(DIR$,13)) THEN PRINT #1,"ASCII Source"
      ELSE PRINT #1,"Binary Source":FT=1
1600   GOTO 1650
1610 IF ASC(MID$(DIR$,14)) THEN PRINT #1,"Random Data"
      ELSE PRINT #1,"Sequential Data"
1620   GOTO 1650
1630 PRINT #1,"Machine Language":FT=2
1640 '
1650 TC=ASC(MID$(DIR$,15))
1660 ON FT GOTO 1670,1720,1950

```

```

1670 CLN=TC:CSCT=1:GOSUB 2230
1680 IF ASC(SCT0$)=&HFE THEN PRINT #1," [P]";
1690 ZUL=FNZA(MID$(SCT0$,2,2))
1700 IF ZUL<&HFFFF THEN PRINT #1," UNLIST" ZUL ELSE PRINT #1
1710 GOTO 1950
1720 CLN=TC:CSCT=1:GOSUB 2230
1730 ZVL=FNZA(MID$(SCT0$,2,2))
1740 ZSA=FNZA(MID$(SCT0$,4,2))
1750 PRINT #1,"Start Address = $" RIGHT$("000"+HEX$(ZSA),4)
1760 PRINT #1,"End Address = $" RIGHT$("000"+HEX$(ZSA+ZVL-1),4)
1770 CS=(ZVL+9)MOD256:IF CS=0 THEN 1860
1780 RC=TC
1790 WHILE RC<&HCO
1800 NC=RC
1810 RC=ASC(MID$(FAT$,RC+1))
1820 WEND
1830 CLN=NC:CSCT=RC-&HBF:GOSUB 2230
1840 IF CS=255 THEN EA$=MID$(SCT0$,255)+SCT1$ ELSE EA$=MID$(SCT0$,CS,2)
1850 GOTO 1940
1860 RC=TC:NC=RC:MC=NC
1870 WHILE RC<&HCO
1880 MC=NC
1890 NC=RC
1900 RC=ASC(MID$(FAT$,RC+1))
1910 WEND
1920 CLN=MC:CSCT=8:GOSUB 2230:EA$=SCT1$
1930 CLN=NC:CSCT=1:GOSUB 2230:EA$=EA$+LEFT$(SCT0$,1)
1940 PRINT #1,"Entry Address = $" RIGHT$("000"+HEX$(FNZA(EA$)),4)
1950 '
1960 CC=0:LC=TC
1970 WHILE LC<&HCO
1980 CC=CC+1
1990 NC=LC
2000 PRINT #1,USING "< ## - ! >";NC*4+2,CHR$(65+NC MOD 4)
2010 LC=ASC(MID$(FAT$,LC+1))
2020 WEND
2030 PRINT #1,CC "Clusters used."
2040 '
2050 PRINT #1,"Last Cluster : ";
2060 IF LC=&HFD THEN SC=0 ELSE IF FT=3 THEN SC=LC-&HBF ELSE SC=LC-&HCO
2070 PRINT #1,USING "## Sectors";SC;
2080 BC=0:EF$=CHR$(&H1A)
2090 IF FT=3 THEN 2170
2100 CLN=NC:CSCT=SC+1:GOSUB 2230
2110 IF SCT1$=EF$ THEN BC=255:GOTO 2170
2120 IF RIGHT$(SCT0$,1)=EF$ THEN BC=254:GOTO 2170
2130 B$=SCT0$:BC=1
2140 WHILE INSTR(BC+1,B$,EF$)>0
2150 BC=INSTR(BC+1,B$,EF$)
2160 WEND
2170 PRINT #1,USING "_& ### Bytes ";BC
2180 PRINT #1,((CC-1)*8+SC)*256+BC "Bytes all."
2190 RETURN
2200 '
2210 '
2220 '
2230 '■■■ Cluster - Sector Read ■■■
2240 TRK=CLN*4+2:SCT=(CLN MOD 4)*8+CSCT
2250 D$=DSKI$(DRV,TRK,SCT)
2260 RETURN

```

8-4-4 ファイルダンプ

BASIC プログラムやシーケンシャルファイルのデータを、順に画面に表示していきます。BREAK キーで中断しますが、中断したら必ず CLOSE を実行してください。

このプログラムでは BASIC プログラムを表示するために、BASIC のリストルーチンを利用しています(リスト 8-5)。

リスト 8-5 ファイルダンプ

```

1000 '*****
1020 '*   File Dump   *
1030 '*   ( LIST 8-5 )   V3.0   *
1040 '*****
1080 CLEAR 300:&H7000:DEFINT A-T:GOSUB 1380
1090 ON ERROR GOTO 1310
1100 LINE INPUT "File descriptor = ? ";FD$
1110 OPEN "I",1,FD$
1120 FLG=ASC(INPUT$(1,1))
1130 IF FLG=0 THEN PRINT "It's a machine language file.":GOTO 1340
1140 IF FLG=&HFE THEN PRINT "It's a protected program file.":GOTO 1340

1150 IF FLG=&HFF THEN 1250
1160 CLOSE 1
1170 '
1180 OPEN "I",1,FD$
1190 WHILE EOF(1)=0
1200   LINE INPUT #1,LIN$
1210   PRINT LIN$
1220 WEND
1230 GOTO 1340
1240 '
1250 PRINT
1260 UL=ASC(INPUT$(1,1))*256+ASC(INPUT$(1,1))
1270 IF UL<65535! THEN PRINT "UNLIST" UL:PRINT
1280 EXEC &H6000
1290 GOTO 1340
1300 '
1310 PRINT "ERROR" ERR "in" ERL
1320 RESUME 1340
1330 '
1340 CLOSE 1
1350 END
1360 '
1370 '
1380 RESTORE 1450:READ M$:A=&H6000
1390 WHILE M$>"/"
1400   POKE A,VAL("&H"+M$)
1410   A=A+1:READ M$
1420 WEND
1430 RETURN
1440 '
1450 DATA 20,15,7E,D0,72,BD,B6,15,7E,9C,22,7E,C1,69,BD,D9
1460 DATA 0F,BD,9B,50,7E,D6,78,C6,01,D7,BF,80,E5,34,02,8D
1470 DATA E1,AA,E0,27,40,8D,DB,1F,89,8D,07,1E,89,34,06,8E
1480 DATA 03,3C,8D,CE,81,FE,26,13,A7,80,8D,C6,A7,80,1F,89
1490 DATA C4,0F,8D,8E,A7,80,5A,26,F9,20,E7,A7,80,26,E3,0F
1500 DATA BF,35,06,8D,80,8E,03,38,8D,B1,8E,04,3D,8D,AF,C6
1510 DATA 01,D7,BF,20,B2,0F,BF,0F,C0,39,/

```

8-4-6 横表示 FILES

F-BASIC V3.0 の見づらい FILES 表示を、図 8-31 のように見やすい表示に変更します(リスト 8-7)。このプログラムは、システムディスクの FILES に関する BASIC コードを直接変更してしまいます。このプログラムは、\$7FE0~\$7FFF(V3.00)あるいは\$FC29~\$FC48(V3.02)の領域を使用しますので、他の目的にこの領域を使わないようにしてください。

この横表示 FILES では、ファイル名の前の空文字に RUN, LOAD 等のコマンドを入力すると、そのコマンドが正常に実行されます。またファイルタイプは、A(アスキーソース)、B(バイナリソース)、C(シーケンシャルデータ)、D(ランダムデータ)、F(マシン語)です。

FILES											
"O:SCTDMP	"B	2	"O:SCTDMP.T"	B	2	"O:ASCDMP	"B	1	"O:ASU	"B	1
"O:RUN.O	"B	2	"O:RUN.3	"B	2	"O:FINFO	"B	2	"O:FDISP	"B	1
"O:FDISP.T	"B	1	"O:ADIR	"B	2	"O:LFAT	"B	2	"O:FCOPYF	"B	2
"O:FCOPYF.T"	B	3	"O:WSET	"B	1	"O:HFILES	"B	2	"O:HFILES.T"	B	1
"O:HF/SUB.T"	B	1	"O:FIPL	"B	1	"O:FIPL.T	"B	1	"O:FATSRC	"B	1
"O:FORMAT	"B	6	"O:DSKRW.M	"F	1	"O:DSKRW.T	"B	5			
109 C. Free											

図8-31 横表示FILES表示例

リスト 8-7 横表示 FILES

```

1000 '*****
1020 '* Horizontal FILES converter *
1030 '* ( LIST 8-7 ) V3.0 *
1040 '*****
1080 CLEAR 1000.&H6A00:DEFINT A-Z
1090 COLOR 7,0:WIDTH 40,20
1100 FIELD 0,163 AS DMY$, 6 AS ID$
1110 D$=DSKI$(0,0,15):LVL=-1
1120 IF ID$="821001" THEN LVL=0 ELSE IF ID$="840505" THEN LVL=2
1130 IF LVL<0 THEN PRINT"Unidentified System Disk !!":GOTO 1220
1140 '
1150 PRINT "## Horizontal FILES Conversion"
1160 PRINT:PRINT USING ">> for F-BASIC V 3.0# ":LVL
1170 PRINT:PRINT">> Sure ? ";
1180 IF INSTR(">>"+CHR$(13),INPUT$(1))=0 THEN 1220
1190 PRINT "Yes"
1200 IF LVL=0 THEN GOSUB 1260 ELSE GOSUB 1370
1210 PRINT:PRINT ">> OK !";
1220 PRINT:END
1230 '
1240 '
1250 '■ for Level 0
1260 FIELD 0,87 AS DMY$,167 AS CNV$
1270 F$="":RESTORE 1520:READ M$
1280 WHILE M$>"/":F$=F$+CHR$(VAL("&H"+M$)):READ M$:WEND
1290 D$=DSKI$(0,0,28):LSET CNV$=F$:DSKD$ 0,0,28
1300 FIELD 0,224 AS DMY$,32 AS CNV$
1310 F$="":RESTORE 1660:READ M$
1320 WHILE M$>"/":F$=F$+CHR$(VAL("&H"+M$)):READ M$:WEND
1330 D$=DSKI$(0,0,32):LSET CNV$=F$:DSKD$ 0,0,32
1340 RETURN
1350 '

```

```

1360 '■ for Level 2
1370 FIELD 0,87 AS DMY$,167 AS CNV$
1380 F$="":RESTORE 1520:READ M$
1390 WHILE M$>"/":F$=F$+CHR$(VAL("&H"+M$)):READ M$:WEND
1400 MID$(F$,55,2)=CHR$(&HFC)+CHR$(&H29)
1410 MID$(F$,61,2)=CHR$(&HFC)+CHR$(&H39)
1420 D$=DSKI$(0,0,28):LSET CNV$=F$:DSKO$ 0,0,28
1430 FIELD 0, 21 AS DMY$,43 AS CNV$
1440 F$="":RESTORE 1640:READ M$
1450 WHILE M$>"/":F$=F$+CHR$(VAL("&H"+M$)):READ M$:WEND
1460 D$=DSKI$(0,0,16):LSET CNV$=F$:DSKO$ 0,0,16
1470 FIELD 0,183 AS DMY$, 2 AS CNV$
1480 D$=DSKI$(0,0,15):LSET CNV$=CHR$(&H6F)+CHR$(&H15):DSKO$ 0,0,15
1490 RETURN
1500 '
1510 '
1520 DATA BD,76,BF,BD,9B,50,86,01,B7,72,02,4C,B7,71,FF,4C
1530 DATA B7,72,00,8E,72,13,BF,71,FC,BD,74,CE,BD,06,78,A6
1540 DATA 84,27,33,43,27,3F,34,10,8E,7B,ED,B6,71,FE,8A,30
1550 DATA A7,06,BD,74,A6,BD,7F,E0,BD,09,2F,BD,7F,F0,E6,06
1560 DATA 8D,6B,1F,89,C1,64,24,08,8D,60,C1,0A,24,02,8D,5A
1570 DATA 4F,BD,B6,15,35,10,30,88,20,8C,73,13,25,BE,B6,72
1580 DATA 00,81,1F,25,AA,BD,9B,50,BD,74,BD,30,06,6F,E2,CC
1590 DATA FF,98,A1,80,26,02,6C,E4,5A,26,F7,4F,35,04,BD,B6
1600 DATA 15,8E,7B,E2,BD,74,A6,7F,05,AC,39,0A,20,43,2E,20
1610 DATA 46,72,65,65,0D,0A,07,1C,1C,1C,1C,22,30,3A,00,00
1620 DATA 00,00,00,00,00,00,00,00,/
1630 '
1640 DATA C6,20,8E,6F,20,BD,85,97,7E,6F,6F
1650 '
1660 DATA 86,1D,BD,09,AE,62,C6,08,BD,9B,FD,86,16,7E,DB,E4
1670 DATA 86,22,BD,09,A6,03,48,8B,42,AB,04,A0,05,7E,D0,8E,/

```

8-4-7 フォーマット&トラックコピー

F-BASICのためのディスクフォーマット、トラックコピーユーティリティです。スキューフォーマットも行なえます(リスト8-8, リスト8-9)。

図8-32 が処理メニューで、左側のキーを押すと各処理が選択されます。

```

♦♦ Disk formatter for F-BASIC ♦♦

Object:D1 / Source:D0 (Step rate: 6ms)

1 : SYSDSK for Ver 3.0
2 : SYSDSK for Ver 3.3
3 : Volume copy
I : 'DSKINI'
R : Read addresses
F : Format track
C : Copy track (selectors data only)
V : Copy track (selectors data & skew)
T : Compare data
S : Convert format skew
D : Change drives position
A : Change step rate
Q : Quit

>> Select menu ?

```

図8-32 フォーマット&トラックコピー処理メニュー

① SYSDSK for Ver3.0

V3.0 に最適なスキューフォーマットされたシステムディスクを作ります。ソースドライブに V3.0 のシステムディスクをセットしておきます。

② SYSDSK for Ver3.3

V3.3 に最適なスキューフォーマットされたシステムディスクを作ります。ソースドライブに V3.3 のシステムディスクをセットしておきます。

③ Volume copy

全トラックにわたって V コマンドを実行してコピーします。

④ DSKINI

BASIC の DSKINI 命令と同じです。

⑤ Read addresses

指定されたトラックのセクタシーケンス(スキュー)を表示します。

⑥ Format track

指定されたスキュータイプでフォーマットを行ないます。スキュータイプについては後述します。

⑦ Copy track(sectors data only)

セクタ内のデータをトラック単位でコピーします。

⑧ Copy track(sectors data & skew)

セクタのデータをスキューを含めてコピーします。

⑨ Compare data

2 つのディスクの内容を比較チェックします。

⑩ Convert format skew

セクタデータはそのまま、スキュータイプだけを指定されたタイプに変換します。

⑪ Change drives position

このフォーマット&トラックコピープログラムでのドライブポジション(オブジェクトドライブ, ソースドライブ)を設定します。

⑫ Change step rate

ステッピングレートを 0~3 で設定します。各ディスクドライブに対する最適値は、次のとおりです。

富士通純正厚型ドライブ……20 msec(“2”を指定します)

TF-20 ……15 msec(“2”を指定します)

TF-10 ……10 msec(“1”を指定します)

⑬ Quit

プログラムの終了

このプログラムではスキューの自由度をあげるために、2種類のアルゴリズムを持っています。タイプ1はセクタ番号を1から始め、次のセクタ番号は前のセクタ番号にパラメータの値を加算して、その値に16の剰余をとります。パラメータが11ならセクタの並びは、[1, 12, 7, 2, 13, 8, 3, 14, 9, 4, 15, 10, 5, 16, 11, 6]となります。当然ながら指定できるパラメータは、1から15までの奇数となります。

一方、タイプ2はパラメータの値をnとすると、インデックスホールからセクタをn個ごとに置いていくというものです。ポストアンプルを考慮してセクタが17個あるものとして数えるため、パラメータは1~15が偶数も含めて許されます。たとえばパラメータが4のときセクタシーケンスは、[13, 9, 5, 1, 14, 10, 6, 2, 15, 11, 7, 3, 16, 12, 8, 4]となります。

リスト 8-8 フォーマット&トラックコピー

```

1000 '*****
1020 '* Ordinary Disk Formatter for F-BASIC (Ver.3.0) *
1030 '* ( LIST 8-8 ) V3.0 *
1032 '* LIST 8-9 か" ヒツヨウ デ"ス *
1040 '*****
1070 CLEAR 300.&H4000:DEFINT A-Z:DIM CMD(6),SR(3)
1080 SDRV=0:ODRV=1
1090 SR(0)=6:SR(1)=12:SR(2)=20:SR(3)=30
1100 COLOR,0:WIDTH 40,20:CONSOLE...0
1110 LOADM OCT$(PEEK(&HFD1D)AND3)+" :L8-9M"
1120 DEFUSR=&H4000
1130 ON ERROR GOTO 4430
1140 '
1150 LOCATE 0,0:COLOR 7
1160 PRINT" ◆◆ Disk formatter for F-BASIC ◆◆
1170 PRINT:COLOR 4
1180 PRINTUSING"Object:DH / Source:DH (Step rate:##ms)";ODRV,SDRV,SR(C
MD(6))
1190 LOCATE 0,4:COLOR 7
1200 PRINT" 1 : SYSDSK for Ver 3.0
1210 PRINT" 2 : SYSDSK for Ver 3.3
1220 PRINT" 3 : Volume copy
1230 COLOR 5
1240 PRINT" I : 'DSKINI'
1250 PRINT" R : Read addresses
1260 PRINT" F : Format track
1270 PRINT" C : Copy track (sectors data only)
1280 PRINT" V : Copy track (sectors data & skew)
1290 PRINT" T : Compare data
1300 PRINT" S : Convert format skew
1310 COLOR 4
1320 PRINT" D : Change drives position
1330 PRINT" A : Change step rate
1340 COLOR 7
1350 PRINT" Q : Quit
1360 BEEP 1:LOCATE 0,18:COLOR 6:PRINT" >> Select menu ?":BEEP 0
1370 MENU$="112233IIRrFfCcVvTtSsDdAaQq"
1380 TSK=(INSTR(MENU$,INPUT$(1))+1)*2
1390 IF TSK=0 THEN 1360
1400 IF TSK=13 THEN FOR I=1 TO 20:PRINT:NEXT:COLOR 7:WIDTH 80,25:END
1410 '
1420 FOR C=18 TO 4 STEP -1:LOCATE 0,C:PRINT CHR$(5):NEXT
1430 ON TSK GOSUB 2160,2330,2520,2650,2780,3190,3360,3540,3730,3950,41
40,4300
1440 FOR C=CSRLIN TO 4 STEP -1:LOCATE 0,C:PRINT SPC(39):NEXT
1450 GOTO 1150

```

```

1460 '
1470 '
1480 '■ Drive check / DRV.WP ■
1490 CMD(2)=DRV:CMD(1)=0:D=USR(CMD(1)):IK$=INKEY$:ABT=0
1500 IF (CMD(0) AND 128)=0 THEN 1530
1510 BEEP 1:LOCATE 10,18:COLOR 3:PRINT"Insert the disk in #"DRV::BEEP
0
1520 IF IK$=CHR$(27) THEN ABT=1:GOTO 1560 ELSE 1490
1530 IF (CMD(0) AND WP)=0 THEN 1560
1540 BEEP 1:LOCATE 10,18:COLOR 3:PRINT"Write protected in #"DRV::BEEP
0
1550 IF IK$=CHR$(27) THEN ABT=1 ELSE 1490
1560 LOCATE 10,18:PRINT SPC(29)
1570 RETURN
1580 '
1590 '■ Track / STTTRK.LSTTRK ■
1600 COLOR 6
1610 LOCATE 0,6:INPUT" Start Track = ":ST$:IF ST$="" THEN 1610
1620 ST=VAL(ST$):IF ST<0 OR 39<ST THEN 1610 ELSE LOCATE 16,6:PRINT ST
1630 LOCATE 19,6:INPUT" / Side = ":SS$:IF SS$="" THEN SS$="0"
1640 SS=VAL(SS$):IF SS<0 OR 1<SS THEN 1630 ELSE LOCATE 30,6:PRINT SS
1650 LOCATE 0,7:INPUT" Last Track = ":LT$:IF LT$="" THEN LT$=ST$
1660 LT=VAL(LT$):IF LT<ST OR 39<LT THEN 1650 ELSE LOCATE 16,7:PRINT LT

1670 LOCATE 19,7:INPUT" / Side = ":LS$:IF LS$="" THEN LS=1 ELSE LS=VAL
(LS$)
1680 STTTRK=ST*2+SS:LSTTRK=LT*2+LS
1690 IF LS<0 OR 1<LS OR STTTRK>LSTTRK THEN 1670 ELSE LOCATE 30,7:PRINT
LS
1700 LOCATE 0,8:COLOR 5:PRINT" again -> [BS]":OK$=INPUT$(1)
1710 FOR C=8 TO 6 STEP -1:LOCATE 0,C:PRINT SPC(39):NEXT
1720 IF OK$=CHR$(8) THEN 1600
1730 LOCATE 0,6:COLOR 4:PRINTUSING" ++ Track : ##/# - ##/#":ST,SS,LT,L
S
1740 RETURN
1750 '
1760 '■ Factor / CMD(5) ■
1770 LOCATE 0,8:COLOR 6:PRINT" >> Skew type (1/2) ? "":T$=INPUT$(1)
)
1780 IF T$="2" THEN 1960 ELSE IF T$<>"1" THEN 1770
1790 '
1800 LOCATE 0,8:INPUT" >> Type 1 skew (1-15) ? ".FCT$:FCT=VAL(FCT$)
1810 IF FCT<1 OR FCT>15 OR (FCT MOD 2)=0 THEN 1800
1820 LOCATE 23,8:PRINT":"+STR$(FCT)
1830 LOCATE 0,9:PRINT SPC(34)
1840 SN=1
1850 FOR SA=1 TO 16
1860 SYMBOL(SA*24+60,90).STR$(SN).1,1
1870 SN=(SN+FCT)MOD 16:IF SN=0 THEN SN=16
1880 NEXT
1890 LOCATE 0,10:COLOR 5:PRINT" again -> [BS]"
1900 BS$=INPUT$(1):LOCATE 0,10:PRINT SPC(20)
1910 IF BS$=CHR$(8) THEN 1770
1920 LOCATE 0,9:PRINT SPC(34)
1930 CMD(5)=FCT
1940 GOTO 2110
1950 '
1960 LOCATE 0,8:INPUT" >> Type 2 skew (1-16) ? ".FCT$:FCT=VAL(FCT$)
1970 IF FCT<1 OR FCT>16 THEN 1960
1980 LOCATE 23,8:PRINT":"+STR$(FCT)
1990 LOCATE 0,9:PRINT SPC(34)
2000 SA=FCT
2010 FOR SN=1 TO 16
2020 SYMBOL(SA*24+60,90).STR$(SN).1,1
2030 SA=(SA+FCT)MOD 17
2040 NEXT
2050 LOCATE 0,10:COLOR 5:PRINT" again -> [BS]"
2060 BS$=INPUT$(1):LOCATE 0,10:PRINT SPC(20)

```

```
2070 IF BS$=CHR$(8) THEN 1770
2080 LOCATE 0,9:PRINT SPC(34)
2090 CMD(5)=-FCT
2100 '
2110 LOCATE 0,8:COLOR 4:PRINT" ++ Skew factor ("T$") ="FCT:SPC(10)
2120 RETURN
2130 '
2140 '
2150 '
2160 '■■■ 1 ■■■
2170 LOCATE 0,4:COLOR 7:PRINT" ## SYSOSK for Ver 3.0
2180 LOCATE 0,6:COLOR 4:PRINT" * format 0-1:<1> . 2-39:<2-4>
2190 PRINT:PRINT" * track 0 copy & DSKINI
2200 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 2300
2210 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 2300
2220 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
2230 IF INSTR("Yy",INPUT$(1))=0 THEN 2300
2240 '
2250 CMD(5)= 1:STTRK=0:LSTTRK= 3:GOSUB 3270
2260 CMD(5)=-4:STTRK=4:LSTTRK=79:GOSUB 3270
2270 STTRK=0:LSTTRK= 1:GOSUB 3440
2280 CMD(1)=0:D=USR(CMD(1))
2290 LOCATE 0,10:COLOR 5:PRINT" ++ DSKINI"SPC(11):DSKINI ODRV
2300 RETURN
2310 '
2320 '
2330 '■■■ 2 ■■■
2340 LOCATE 0,4:COLOR 7:PRINT" ## SYSOSK for Ver 3.3
2350 LOCATE 0,6:COLOR 4:PRINT" * format 0,2-9:<1-11> 1,10-39:<2-2>
2360 PRINT:PRINT" * track 0-9 copy & DSKINI
2370 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 2490
2380 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 2490
2390 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
2400 IF INSTR("Yy",INPUT$(1))=0 THEN 2490
2410 '
2420 CMD(5)=11:STTRK= 0:LSTTRK= 1:GOSUB 3270
2430 CMD(5)=-2:STTRK= 2:LSTTRK= 3:GOSUB 3270
2440 CMD(5)=11:STTRK= 4:LSTTRK=19:GOSUB 3270
2450 CMD(5)=-2:STTRK=20:LSTTRK=79:GOSUB 3270
2460 STTRK= 0:LSTTRK=19:GOSUB 3440
2470 CMD(1)=0:D=USR(CMD(1))
2480 LOCATE 0,10:COLOR 5:PRINT" ++ DSKINI"SPC(11):DSKINI ODRV
2490 RETURN
2500 '
2510 '
2520 '■■■ 3 ■■■
2530 LOCATE 0,4:COLOR 7:PRINT" ## Volume copy
2540 LOCATE 0,6:COLOR 4:PRINT" * Copy format skew & data
2550 PRINT:PRINT" * Track 0/0 - 39/1
2560 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 2620
2570 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 2620
2580 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
2590 IF INSTR("Yy",INPUT$(1))=0 THEN 2620
2600 '
2610 STTRK=0:LSTTRK=79:GOSUB 3620
2620 RETURN
2630 '
2640 '
2650 '■■■ I ■■■
2660 LOCATE 0,4:COLOR 5:PRINT" ## DSKINI
2670 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 2750
2680 BEEP 1:LOCATE 0,6:COLOR 3:PRINT" >> Sure ?":BEEP 0
2690 IF INSTR("Yy",INPUT$(1))=0 THEN 2750
2700 '
2710 BEEP 1:LOCATE 0,6:COLOR 4:PRINT" ++ writing..":BEEP 0
2720 FIELD 0,1 AS ID$:DMY=LEN(DSKI$(ODRV,0,3))
2730 LSET ID$="S":DSKID$ ODRV,0,3
2740 DSKINI ODRV
```

```

2750 RETURN
2760 '
2770 '
2780 '■■■ R ■■■
2790 DRV=ODRV:WP=0:GOSUB 1480:IF ABT THEN 3160
2800 LOCATE 0,4:COLOR 5:PRINT" ## Read address (sector sequence)
2810 LOCATE 0,6:COLOR 6:INPUT" Track = ";TRK$:IF TRK$="" THEN 2810
2820 TRK=VAL(TRK$):IF TRK<0 OR 39<TRK THEN 2810
2830 SYMBOL(40,110),["+CHR$(30)+"] = - 1 track",1,1,5
2840 SYMBOL(40,120),["+CHR$(31)+"] = + 1 track",1,1,5
2850 SYMBOL(40,130),["SHIFT]+["+CHR$(30)+"] = -10 track",1,1,5
2860 SYMBOL(40,140),["SHIFT]+["+CHR$(31)+"] = +10 track",1,1,5
2870 SYMBOL(40,150),["+CHR$(29)+"] = drive decrement",1,1,5
2880 SYMBOL(40,160),["+CHR$(28)+"] = drive increment",1,1,5
2890 SYMBOL(40,170),["ESC] = quit",1,1,5
2900 IST$=CHR$(30)+CHR$(31)+CHR$(25)+CHR$(26)+CHR$(29)+CHR$(28)+CHR$(2
7)
2910 LOCATE 0,6:PRINTUSING" ++ Drive # / Track : ## ";DRV,TRK
2920 CMD(1)=1:CMD(2)=DRV
2930 CMD(3)=TRK
2940 CMD(4)=0:LOCATE 0,8:PRINT" 0:"SPC(30)
2950 D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
2960 FOR A=0 TO 15
2970 SYMBOL(24*A+60,80),STR$(PEEK(&H4602+A*6)),1,1
2980 NEXT
2990 CMD(4)=1:LOCATE 0,9:PRINT" 1:"SPC(30)
3000 D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3010 FOR A=0 TO 15
3020 SYMBOL(24*A+60,90),STR$(PEEK(&H4602+A*6)),1,1
3030 NEXT
3040 BEEP 1:FOR W=0 TO 50:NEXT:BEEP 0
3050 DMY$=INKEY$:IKY=INSTR(IST$,INPUT$(1))
3060 IF IKY=0 THEN 3040
3070 IF IKY=1 THEN TRK=(TRK+39)MOD40
3080 IF IKY=2 THEN TRK=(TRK+1)MOD40
3090 IF IKY=3 THEN TRK=(TRK+30)MOD40
3100 IF IKY=4 THEN TRK=(TRK+10)MOD40
3110 IF IKY<5 THEN 2910
3120 IF IKY=5 THEN DRV=(DRV+1)MOD(PEEK(&H71FA)+1)
3130 IF IKY=6 THEN DRV=(DRV+PEEK(&H71FA))MOD(PEEK(&H71FA)+1)
3140 IF IKY<7 THEN GOSUB 1480:IF ABT=0 THEN 2910
3150 LINE(40,110)-(400,180),PRESET,.BF
3160 RETURN
3170 '
3180 '
3190 '■■■ F ■■■
3200 LOCATE 0,4:COLOR 5:PRINT" ## Format disk
3210 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 3330
3220 GOSUB 1590
3230 GOSUB 1760
3240 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
3250 IF INSTR("yY",INPUT$(1))=0 THEN 3330
3260 '
3270 CMD(1)=6:CMD(2)=ODRV:COLOR 5
3280 FOR T=STTRK TO LSTTRK
3290 CMD(3)=T*2:CMD(4)=T MOD2
3300 LOCATE 0,10:PRINTUSING" ++ Track ## / Side #";CMD(3),CMD(4)
3310 D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3320 NEXT
3330 RETURN
3340 '
3350 '
3360 '■■■ C ■■■
3370 LOCATE 0,4:COLOR 5:PRINT" ## Copy sectors data
3380 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 3510
3390 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 3510
3400 GOSUB 1590
3410 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0

```

```
3420 IF INSTR("Yy".INPUT$(1))=0 THEN 3510
3430 '
3440 COLOR 5
3450 FOR T=STTRK TO LSTTRK
3460   CMD(3)=T*2:CMD(4)=T MOD2
3470   LOCATE 0,10:PRINTUSING" ++ Track ## / Side #":CMD(3),CMD(4)
3480   CMD(2)=SDRV:CMD(1)=2:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3490   CMD(2)=ODRV:CMD(1)=4:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3500 NEXT
3510 RETURN
3520 '
3530 '
3540 '■■■■ V ■■■■
3550 LOCATE 0,4:COLOR 5:PRINT" ## Copy skew factor & sectors
3560 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 3700
3570 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 3700
3580 GOSUB 1590
3590 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
3600 IF INSTR("Yy".INPUT$(1))=0 THEN 3700
3610 '
3620 COLOR 5
3630 FOR T=STTRK TO LSTTRK
3640   CMD(3)=T*2:CMD(4)=T MOD2
3650   LOCATE 0,10:PRINTUSING" ++ Track ## / Side #":CMD(3),CMD(4)
3660   CMD(2)=SDRV:CMD(1)=2:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3670   CMD(2)=ODRV:CMD(1)=5:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3680   CMD(1)=4:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3690 NEXT
3700 RETURN
3710 '
3720 '
3730 '■■■■ T ■■■■
3740 LOCATE 0,4:COLOR 5:PRINT" ## Compare data on sectors
3750 DRV=ODRV:WP= 0:GOSUB 1480:IF ABT THEN 3920
3760 DRV=SDRV:WP= 0:GOSUB 1480:IF ABT THEN 3920
3770 GOSUB 1590
3780 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
3790 IF INSTR("Yy".INPUT$(1))=0 THEN 3920
3800 '
3810 COLOR 5
3820 FOR T=STTRK TO LSTTRK
3830   CMD(3)=T*2:CMD(4)=T MOD2
3840   LOCATE 0,10:PRINTUSING" ++ Track ## / Side #":CMD(3),CMD(4):
3850   CMD(2)=SDRV:CMD(1)=2:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3860   CMD(2)=ODRV:CMD(1)=3:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
3870   IF CMD(1)=0 THEN 3910
3880   LOCATE 25,10:COLOR 6:PRINT"different !"
3890   IF INPUT$(1)=CHR$(27) THEN T=LSTTRK
3900   LOCATE 25,10:COLOR 5:PRINT SPC(12)
3910 NEXT
3920 RETURN
3930 '
3940 '
3950 '■■■■ S ■■■■
3960 LOCATE 0,4:COLOR 5:PRINT" ## Convert format skew
3970 DRV=ODRV:WP=64:GOSUB 1480:IF ABT THEN 4110
3980 GOSUB 1590
3990 GOSUB 1760
4000 BEEP 1:LOCATE 0,10:COLOR 3:PRINT" >> Sure ?":BEEP 0
4010 IF INSTR("Yy".INPUT$(1))=0 THEN 4110
4020 '
4030 COLOR 5:CMD(2)=ODRV
4040 FOR T=STTRK TO LSTTRK
4050   CMD(3)=T*2:CMD(4)=T MOD2
4060   LOCATE 0,10:PRINTUSING" ++ Track ## / Side #":CMD(3),CMD(4)
4070   CMD(1)=2:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
4080   CMD(1)=6:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
4090   CMD(1)=4:D=USR(CMD(1)):IF CMD(0) THEN ERROR 30
```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
4000 : 17 02 26 22 92 58 1A 50 86 FD 1F 88 A6 01 26 07 17 : 28
4010 : 18 01 0F 20 02 8D 15 A7 1F 26 08 6D 01 26 07 17 : 94
4020 : 00 02 96 18 A7 1F 4F 1F 88 1C AF 39 CE 42 08 A6 : 04
4030 : 03 84 03 6F C6 D6 1D 8A 80 97 1D 8D 49 96 1D 2A : 23
4040 : 4A 5D 28 08 86 02 17 00 AB 4A 26 FA 86 08 20 19 : 55
4050 : 8D 68 25 37 CE 42 D8 E6 47 A6 C5 97 19 A6 05 A1 : 00
4060 : C5 27 22 A7 C5 97 18 86 18 A4 08 97 18 86 06 4A : 04
4070 : 27 1D 10 10 8E C8 00 31 3F 27 F5 D6 1F C5 40 27 F6 : 4D
4080 : D6 18 17 00 92 4F 8D 00 8D 00 39 86 0A 20 02 86 : 71
4090 : 0F 1A 01 39 D6 18 54 24 04 C6 D0 07 18 96 18 43 : 46
40A0 : 97 18 8D 80 E2 91 18 26 D0 CE F0 00 33 5F 11 83 00 : 84
40B0 : 00 27 06 D6 18 28 F4 C6 01 39 CE 42 D8 D6 1D E7 : FF
40C0 : 44 C4 03 E7 48 A6 03 84 03 A7 47 6D 44 28 16 86 : 00
40D0 : 80 34 42 17 FF 58 35 42 4C 81 84 26 F4 A6 47 8A : C0
40E0 : 80 97 1D 20 08 A1 48 27 04 8A 80 97 1D 8D 45 27 : 87
40F0 : A0 1C FE 39 CE 03 D9 8D 09 33 5F 11 83 00 26 : 7F
[cs] : BE 81 58 85 1C 07 2D 8C 9D 39 43 12 6E 6E 37 F3 : 2C

```

1/2 8-9 フキ-アット&トラックコピ- アシシ

```

4100 NEXT
4110 RETURN
4120
4130
4140
4150 LOCATE 0,4:COLOR 4:PRINT" ** Set drive position"
4160 COLOR 6
4170 LOCATE 0,6:PRINT" >> Object: "ODRV"=>"
4180 INPUT"ODRV$:"IF DRV$="" THEN DRV=ODRV ELSE DRV=VAL(DRV$)
4190 LOCATE 17,6:PRINT DRV
4200 IF DRV<0 OR PEEK(&H71FA)<DRV THEN 4170
4210 ODRV=DRV
4220 LOCATE 0,7:PRINT" >> Source: "SDRV"=>"
4230 INPUT"SDRV$:"IF DRV$="" THEN DRV=SDRV ELSE DRV=VAL(DRV$)
4240 LOCATE 17,7:PRINT DRV
4250 IF DRV<0 OR PEEK(&H71FA)<DRV OR DRV=ODRV THEN 4220
4260 SDRV=DRV
4270 RETURN
4280
4290
4300
4310 LOCATE 0,4:COLOR 4:PRINT" ** Change step rate
4320 LOCATE 0,6:COLOR 6
4330 PRINTUSING" >> Current: # (##ms) => " :CMD(6),SR(CMD(6)):
4340 INPUT"SR$:"IF SR$="" THEN SR=2 ELSE SR=VAL(SR$)
4350 IF SR<0 OR 3<SR THEN 4320
4360 LOCATE 26,6:PRINTUSING"## (##ms)":SR,SR(SR)
4370 CMD(6)=SR
4380 RETURN
4390
4400
4410
4420
4430
4440
4450
4460 ON ERROR GOTO 0
4470 LOCATE 7:LOCATE 0,19
4480 IF ERR=50 THEN 4470
4490
4500
4510
4520
4530
4540
4550
4560
4570
4580
4590
4600
4610
4620
4630
4640
4650
4660
4670
4680
4690
4700
4710
4720
4730
4740
4750
4760
4770
4780
4790
4800
4810
4820
4830
4840
4850
4860
4870
4880
4890
4900
4910
4920
4930
4940
4950
4960
4970
4980
4990
5000
5010
5020
5030
5040
5050
5060
5070
5080
5090
5100
5110
5120
5130
5140
5150
5160
5170
5180
5190
5200
5210
5220
5230
5240
5250
5260
5270
5280
5290
5300
5310
5320
5330
5340
5350
5360
5370
5380
5390
5400
5410
5420
5430
5440
5450
5460
5470
5480
5490
5500
5510
5520
5530
5540
5550
5560
5570
5580
5590
5600
5610
5620
5630
5640
5650
5660
5670
5680
5690
5700
5710
5720
5730
5740
5750
5760
5770
5780
5790
5800
5810
5820
5830
5840
5850
5860
5870
5880
5890
5900
5910
5920
5930
5940
5950
5960
5970
5980
5990
6000
6010
6020
6030
6040
6050
6060
6070
6080
6090
6100
6110
6120
6130
6140
6150
6160
6170
6180
6190
6200
6210
6220
6230
6240
6250
6260
6270
6280
6290
6300
6310
6320
6330
6340
6350
6360
6370
6380
6390
6400
6410
6420
6430
6440
6450
6460
6470
6480
6490
6500
6510
6520
6530
6540
6550
6560
6570
6580
6590
6600
6610
6620
6630
6640
6650
6660
6670
6680
6690
6700
6710
6720
6730
6740
6750
6760
6770
6780
6790
6800
6810
6820
6830
6840
6850
6860
6870
6880
6890
6900
6910
6920
6930
6940
6950
6960
6970
6980
6990
7000
7010
7020
7030
7040
7050
7060
7070
7080
7090
7100
7110
7120
7130
7140
7150
7160
7170
7180
7190
7200
7210
7220
7230
7240
7250
7260
7270
7280
7290
7300
7310
7320
7330
7340
7350
7360
7370
7380
7390
7400
7410
7420
7430
7440
7450
7460
7470
7480
7490
7500
7510
7520
7530
7540
7550
7560
7570
7580
7590
7600
7610
7620
7630
7640
7650
7660
7670
7680
7690
7700
7710
7720
7730
7740
7750
7760
7770
7780
7790
7800
7810
7820
7830
7840
7850
7860
7870
7880
7890
7900
7910
7920
7930
7940
7950
7960
7970
7980
7990
8000
8010
8020
8030
8040
8050
8060
8070
8080
8090
8100
8110
8120
8130
8140
8150
8160
8170
8180
8190
8200
8210
8220
8230
8240
8250
8260
8270
8280
8290
8300
8310
8320
8330
8340
8350
8360
8370
8380
8390
8400
8410
8420
8430
8440
8450
8460
8470
8480
8490
8500
8510
8520
8530
8540
8550
8560
8570
8580
8590
8600
8610
8620
8630
8640
8650
8660
8670
8680
8690
8700
8710
8720
8730
8740
8750
8760
8770
8780
8790
8800
8810
8820
8830
8840
8850
8860
8870
8880
8890
8900
8910
8920
8930
8940
8950
8960
8970
8980
8990
9000
9010
9020
9030
9040
9050
9060
9070
9080
9090
9100
9110
9120
9130
9140
9150
9160
9170
9180
9190
9200
9210
9220
9230
9240
9250
9260
9270
9280
9290
9300
9310
9320
9330
9340
9350
9360
9370
9380
9390
9400
9410
9420
9430
9440
9450
9460
9470
9480
9490
9500
9510
9520
9530
9540
9550
9560
9570
9580
9590
9600
9610
9620
9630
9640
9650
9660
9670
9680
9690
9700
9710
9720
9730
9740
9750
9760
9770
9780
9790
9800
9810
9820
9830
9840
9850
9860
9870
9880
9890
9900
9910
9920
9930
9940
9950
9960
9970
9980
9990
10000
10010
10020
10030
10040
10050
10060
10070
10080
10090
10100
10110
10120
10130
10140
10150
10160
10170
10180
10190
10200
10210
10220
10230
10240
10250
10260
10270
10280
10290
10300
10310
10320
10330
10340
10350
10360
10370
10380
10390
10400
10410
10420
10430
10440
10450
10460
10470
10480
10490
10500
10510
10520
10530
10540
10550
10560
10570
10580
10590
10600
10610
10620
10630
10640
10650
10660
10670
10680
10690
10700
10710
10720
10730
10740
10750
10760
10770
10780
10790
10800
10810
10820
10830
10840
10850
10860
10870
10880
10890
10900
10910
10920
10930
10940
10950
10960
10970
10980
10990
11000
11010
11020
11030
11040
11050
11060
11070
11080
11090
11100
11110
11120
11130
11140
11150
11160
11170
11180
11190
11200
11210
11220
11230
11240
11250
11260
11270
11280
11290
11300
11310
11320
11330
11340
11350
11360
11370
11380
11390
11400
11410
11420
11430
11440
11450
11460
11470
11480
11490
11500
11510
11520
11530
11540
11550
11560
11570
11580
11590
11600
11610
11620
11630
11640
11650
11660
11670
11680
11690
11700
11710
11720
11730
11740
11750
11760
11770
11780
11790
11800
11810
11820
11830
11840
11850
11860
11870
11880
11890
11900
11910
11920
11930
11940
11950
11960
11970
11980
11990
12000
12010
12020
12030
12040
12050
12060
12070
12080
12090
12100
12110
12120
12130
12140
12150
12160
12170
12180
12190
12200
12210
12220
12230
12240
12250
12260
12270
12280
12290
12300
12310
12320
12330
12340
12350
12360
12370
12380
12390
12400
12410
12420
12430
12440
12450
12460
12470
12480
12490
12500
12510
12520
12530
12540
12550
12560
12570
12580
12590
12600
12610
12620
12630
12640
12650
12660
12670
12680
12690
12700
12710
12720
12730
12740
12750
12760
12770
12780
12790
12800
12810
12820
12830
12840
12850
12860
12870
12880
12890
12900
12910
12920
12930
12940
12950
12960
12970
12980
12990
13000
13010
13020
13030
13040
13050
13060
13070
13080
13090
13100
13110
13120
13130
13140
13150
13160
13170
13180
13190
13200
13210
13220
13230
13240
13250
13260
13270
13280
13290
13300
13310
13320
13330
13340
13350
13360
13370
13380
13390
13400
13410
13420
13430
13440
13450
13460
13470
13480
13490
13500
13510
13520
13530
13540
13550
13560
13570
13580
13590
13600
13610
13620
13630
13640
13650
13660
13670
13680
13690
13700
13710
13720
13730
13740
13750
13760
13770
13780
13790
13800
13810
13820
13830
13840
13850
13860
13870
13880
13890
13900
13910
13920
13930
13940
13950
13960
13970
13980
13990
14000
14010
14020
14030
14040
14050
14060
14070
14080
14090
14100
14110
14120
14130
14140
14150
14160
14170
14180
14190
14200
14210
14220
14230
14240
14250
14260
14270
14280
14290
14300
14310
14320
14330
14340
14350
14360
14370
14380
14390
14400
14410
14420
14430
14440
14450
14460
14470
14480
14490
14500
14510
14520
14530
14540
14550
14560
14570
14580
14590
14600
14610
14620
14630
14640
14650
14660
14670
14680
14690
14700
14710
14720
14730
14740
14750
14760
14770
14780
14790
14800
14810
14820
14830
14840
14850
14860
14870
14880
14890
14900
14910
14920
14930
14940
14950
14960
14970
14980
14990
15000
15010
15020
15030
15040
15050
15060
15070
15080
15090
15100
15110
15120
15130
15140
15150
15160
15170
15180
15190
15200
15210
15220
15230
15240
15250
15260
15270
15280
15290
15300
15310
15320
15330
15340
15350
15360
15370
15380
15390
15400
15410
15420
15430
15440
15450
15460
15470
15480
15490
15500
15510
15520
15530
15540
15550
15560
15570
15580
15590
15600
15610
15620
15630
15640
15650
15660
15670
15680
15690
15700
15710
15720
15730
15740
15750
15760
15770
15780
15790
15800
15810
15820
15830
15840
15850
15860
15870
15880
15890
15900
15910
15920
15930
15940
15950
15960
15970
15980
15990
16000
16010
16020
16030
16040
16050
16060
16070
16080
16090
16100
16110
16120
16130
16140
16150
16160
16170
16180
16190
16200
16210
16220
16230
16240
16250
16260
16270
16280
16290
16300
16310
16320
16330
16340
16350
16360
16370
16380
16390
16400
16410
16420
16430
16440
16450
16460
16470
16480
16490
16500
16510
16520
16530
16540
16550
16560
16570
16580
16590
16600
16610
16620
16630
16640
16650
16660
16670
16680
16690
16700
16710
16720
16730
16740
16750
16760
16770
16780
16790
16800
16810
16820
16830
16840
16850
16860
16870
16880
16890
16900
16910
16920
16930
16940
16950
16960
16970
16980
16990
17000
17010
17020
17030
17040
17050
17060
17070
17080
17090
17100
17110
17120
17130
17140
17150
17160
17170
17180
17190
17200
17210
17220
17230
17240
17250
17260
17270
17280
17290
17300
17310
17320
17330
17340
17350
17360
17370
17380
17390
17400
17410
17420
17430
17440
17450
17460
17470
17480
17490
17500
17510
17520
17530
17540
17550
17560
17570
17580
17590
17600
17610
17620
17630
17640
17650
17660
17670
17680
17690
17700
17710
17720
17730
17740
17750
17760
17770
17780
17790
17800
17810
17820
17830
17840
17850
17860
17870
17880
17890
17900
17910
17920
17930
17940
17950
17960
17970
17980
17990
18000
18010
18020
18030
18040
18050
18060
18070
18080
18090
18100
18110
18120
18130
18140
18150
18160
18170
18180
18190
18200
18210
18220
18230
18240
18250
18260
18270
18280
18290
18300
18310
18320
18330
18340
18350
18360
18370
18380
18390
18400
18410
18420
18430
18440
18450
18460
18470
18480
18490
18500
18510
18520
18530
18540
18550
18560
18570
18580
18590
18600
18610
18620
18630
18640
18650
18660
18670
18680
18690
18700
18710
18720
18730
18740
18750
18760
18770
18780
18790
18800
18810
18820
18830
18840
18850
18860
18870
18880
18890
18900
18910
18920
18930
18940
18950
18960
18970
18980
18990
19000
19010
19020
19030
19040
19050
19060
19070
19080
19090
19100
19110
19120
19130
19140
19150
19160
19170
19180
19190
19200
19210
19220
19230
19240
19250
19260
19270
19280
19290
19300
19310
19320
19330
19340
19350
19360
19370
19380
19390
19400
19410
19420
19430
19440
19450
19460
19470
19480
19490
19500
19510
19520
19530
19540
19550
19560
19570
19580
19590
19600
19610
19620
19630
19640
19650
19660
19670
19680
19690
19700
19710
19720
19730
19740
19750
19760
19770
19780
19790
19800
19810
19820
19830
19840
19850
19860
19870
19880
19890
19900
19910
19920
19930
19940
19950
19960
19970
19980
19990
20000
20010
20020
20030
20040
20050
20060
20070
20080
20090
20100
20110
20120
20130
20140
20150
20160
20170
20180
20190
20200
20210
20220
20230
20240
20250
20260
20270
20280
20290
20300
20310
20320
20330
20340
20350
20360
20370
20380
20390
20400
20410
20420
20430
20440
20450
20460
20470
20480
20490
20500
20510
20520
20530
20540
20550
20560
20570
20580
20590
20600
20610
20620
20630
20640
20650
20660
20670
20680
20690
20700
20710
20720
20730
20740
20750
20760
20770
20780
20790
20800
20810
20820
20830
20840
20850
20860
20870
20880
20890
20900
20910
20920
20930
20940
20950
20960
20970
20980
20990
21000
21010
21020
21030
21040
21050
21060
21070
21080
21090
21100
21110
21120
21130
21140
21150
21160
21170
21180
21190
21200
21210
21220
21230
21240
21250
21260
21270
21280
21290
21300
21310
21320
21330
21340
21350
21360
21370
21380
21390
21400
21410
21420
21430
21440
21450
21460
21470
21480
21490
21500
21510
21520
21530
21540
21550
21560
21570
21580
21590
21600
21610
21620
21630
21640
21650
21660
21670
21680
21690
21700
21710
21720
21730
21740
21750
21760
21770
21780
21790
21800
21810
21820
21830
21840
21850
21860
21870
21880
21890
21900
21910
21920
21930
21940
21950
21960
21970
21980
21990
22000
22010
22020
22030
22040
22050
22060
22070
22080
22090
22100
22110
22120
22130
22140
22150
22160
22170
22180
22190
22200
22210
22220
22230
22240
22250
22260
22270
22280
22290
22300
22310
22320
22330
22340
22350
22360
22370
22380
22390
22400
22410
22420
22430
22440
22450
22460
22470
22480
22490
22500
22510
22520
22530
22540
22550
22560
22570
22580
22590
22600
22610
22620
22630
22640
22650
22660
22670
22680
22690
22700
22710
22720
22730
22740
22750
22760
22770
22780
22790
22800
22810
22820
22830
22840
22850
22860
22870
22880
22890
22900
22910
22920
22930
22940
22950
22960
22970
22980
22990
23000
23010
23020
23030
23040
23050
23060
23070
23080
23090
23100
23110
23120
23130
23140
23150
23160
23170
23180
23190
23200
23210
23220
23230
```

- | | |
|---------------------|---------------------|
| ① トライル数, トライル名..... | トライル 0, セクダ 3 |
| ② トライル名..... | トライル 0, セクダ 16(3.0) |
| | トライル 6, セクダ 7(V3.3) |


```

## Auto Start Utility ##

Drive # ? 0

F-BASIC Ver 3.0 system disk in Drive 0

How many disk drives (0 or 1-4) ?      2 => 2

How many disk files (0-15) ?           1 => 2

What is file name (STARTUP) ? "STARTUP " => RUN

Sure [Y] ?

```

図8-33 オートスタートユーティリティのパラメータ

```

[ 0 ] ( 0 ) 「 3 」

cs +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F cs 0123456789ABCDEF
   ドライブ数.....ファイル数
00 53 20 20 02 01 00 00 00 00 00 00 00 00 00 00 96 S
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

cs 53 20 20 02 01 00 00 00 00 00 00 00 00 00 00 96

```

```

[ 0 ] ( 0 ) 「 16 」

os +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F cs 0123456789ABCDEF
00 E6 E6 84 33 88 56 30 01 A6 80 A7 D1 5A 26 F9 35 DE 31 V0 ラ_74Z&5
10 02 4D 7E 71 6F 00 00 00 00 00 00 00 00 00 00 AD M^qo
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 34 40 BD 31 48ス
70 CB BC BE 72 13 BF 71 FC CC 00 02 F7 71 FF B7 72 21 31 70 q7 *q *r
80 02 86 02 B7 72 00 7F 71 FE BD 74 E6 B6 72 01 26 07 31 70 q7 *q *r
90 63 A6 03 4A B1 03 22 5C E6 04 C1 0F 22 56 FD 71 FB c7 J_ " * * " Vq
A0 FA CC 01 03 B7 72 02 F7 72 00 B6 72 00 81 20 27 4E 71 70 q7 *r 71 70 q7 *r
B0 43 BD 74 E6 7D 72 01 26 3B 8E 72 13 CE 6F EA C6 AB Cst 71 r & ; 71 r 71 r
C0 07 A6 84 27 13 81 FF 27 2B A6 85 A1 C5 26 09 5A 57 ラ_ ' _ ' +ラ_ 71 r 71 r
D0 2A F7 F7 72 13 16 00 AC 30 8B 20 8C 73 13 26 E1 50 * * 71 r 71 r 71 r 71 r
E0 7C 72 00 20 C5 52 55 4E 20 22 53 54 41 52 54 55 ED 71 r 71 r "STARTU
F0 50 20 22 00 7F 72 13 BE 71 9C 8D 5A 8D 5B BE 71 FF P "r 71 r 71 r 71 r

cs 4F D3 A7 B9 9B 57 AC 96 EF BB 8B 1D 77 F7 09 E9 68

```

ファイル名

[1] (6) 「 7 」																		
os	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	cs	0123456789ABCDEF
00	C5	26	0B	5A	2A	F7	F7	90	45	16	01	1E	20	6E	30	88	B8	ナ& Z*ネー-E nOI
10	20	8C	91	45	26	DD	7C	90	29	20	BB	34	14	8D	22	5C	E8	■→E&シ!+) 94 ■"キ
20	20	04	34	14	8D	1B	5A	58	8E	FC	34	AE	85	A7	88	13	F9	4 ■ ZX■4ヨ■71
30	35	94	34	14	8D	0B	5A	58	8E	FC	34	AE	85	A6	86	35	AD	5-4 ■ ZX■4ヨ■7■5
40	94	F6	FF	EE	C4	60	57	57	57	57	39	34	12	8D	12	6C	7-9 /ト'WWW94 ■	
50	4C	20	04	34	12	8D	0B	4A	48	8E	FC	34	AE	86	E6	85	3D	L 4 ■ JH■4ヨ■
60	35	92	B6	FF	EE	84	60	47	47	47	47	39	52	55	4E	DF	5+カ /■'GGGGG9RUN	
70	20	22	53	54	41	52	54	55	50	20	22	00	7F	90	45	8E	99	"STARTUP "→E■
80	88	37	8D	3E	8D	39	8E	88	46	8D	37	8E	88	50	8D	32	95	17■>■9■ F■7■ P■2
90	8E	88	56	8D	2D	8E	88	5D	8D	28	8D	23	8D	95	CF	8D	DC	■ V■-■ J■(■#ス-マス
A0	E5	A1	25	E0	9F	D9	C6	02	6D	01	27	05	8D	86	FE	25	CB	■ %ニルニ m ' ス■%
B0	D3	5A	28	D0	C1	03	22	CC	1F	98	8D	86	22	20	08	7E	9C	モZ+ミチ "フ ィス■" ~
C0	E6	8E	E6	80	7E	AE	84	8D	F6	8E	88	46	8D	F4	8E	88	30	セ~'~ヨ~'~分■ F■E■
D0	50	8D	EF	8E	88	65	8D	EA	8D	E5	8D	95	CF	BD	E5	A1	94	F■~■ e■~■ス-マス■
E0	25	E5	9F	D9	C6	01	6D	01	27	08	8D	12	25	D9	9D	D8	F8	%~'ルニ m ' ■ %ルヘリ
F0	26	D5	5D	2B	D2	C1	0F	22	CE	F7	90	24	20	2C	5F	34	9F	&11ヨ+メチ "ホネ+ネ ,_4
cs	BE	D3	14	C9	27	35	C8	5A	97	3A	EA	AF	9D	03	3E	66	9A	

ファイル名

図8-34 オートスタート条件

リスト 8-10 オートスタートユーティリティ

```

1000 '*****
1020 '* Auto Start Utility *
1030 '* ( LIST 8-10 ) V3.0/V3.3 *
1040 '*****
1070 CLEAR 1000,&H7000:DEFINT A-Z:COLOR 4,0:WIDTH 80,25:CONSOLE...0
1080 FIELD 0.3 AS ID$,1 AS OD$,1 AS DF$
1090 FIELD 0.234 AS DMY0$,8 AS SFN0$
1100 FIELD 0.114 AS DMY3$,8 AS SFN3$
1110 '
1120 PRINT"## Auto Start Utility ##"
1130 BEEP 1:LOCATE 0,2:COLOR 6:PRINT"Drive # ?":BEEP 0
1140 D=(INSTR(CHR$(13)+"0112233",INPUT$(1))-1)*2:IF D>3 THEN 1130
1150 PRINT D
1160 D$=DSKI$(D,0,15):ID3$=ID$:D$=DSKI$(D,0,32)
1170 IF ID3<>"1" THEN LOCATE 0,4:COLOR 2:PRINT"Not Ver 3.0/3 DISK.":
GOTO 1520
1180 V=-3*(ASC(ID3$)=0)
1190 '
1200 LOCATE 0,4:COLOR 5:PRINTUSING"F-BASIC Ver 3_.# system disk in Drive #":V,D
1210 D$=DSKI$(D,0, 3):ND=ASC(DD$):NF=ASC(DF$)
1220 IF V=0 THEN D$=DSKI$(D,0,16):SF$=SFN0$
1230 IF V=3 THEN D$=DSKI$(D,6, 7):SF$=SFN3$
1240 '
1250 LOCATE 0,6:COLOR 7:PRINT"How many disk drives (0 or 1-4) ? "
1260 COLOR 6:PRINT ND::INPUT"=> ".HD$
1270 IF HD$="" THEN DD=ND ELSE DD=VAL(HD$)
1280 IF DD<0 OR 4<DD THEN BEEP:GOTO 1250
1290 LOCATE 43,6:PRINT DD
1300 IF DD=0 THEN LOCATE 47,6:COLOR 3:PRINT "-> Auto Start Cancel ! ":
GOTO 1440
1310 '
1320 LOCATE 0,8:COLOR 7:PRINT"How many disk files (0-15) ? "
1330 COLOR 6:PRINT NF::INPUT"=> ".HF$

```

```

1340 IF HF$="" THEN DF=NF ELSE DF=VAL(HF$)
1350 IF DF<0 OR 15<DF THEN BEEP:GOTO 1320
1360 LOCATE 43,8:PRINT DF
1370 '
1380 LOCATE 0,10:COLOR 7:PRINT"What is file name (STARTUP) ? ";
1390 COLOR 6:PRINT CHR$(34)SF$CHR$(34)::INPUT" => ",WN$
1400 IF WN$="" THEN WN$=SF$
1410 IF LEN(WN$)>8 THEN BEEP:GOTO 1380 ELSE SFD$=LEFT$(WN$+"
8)
1420 LOCATE 44,10:PRINT SFD$
1430 '
1440 BEEP 1:LOCATE 0,12:COLOR 3:PRINT"Sure [Y] ?":BEEP 0
1450 IF INSTR("Yy",INPUT$(1))=0 THEN 1520
1460 '
1470 D$=DSKI$(D,0,3):LSET DD$=CHR$(DD):LSET DF$=CHR$(DF):DSKO$ D,0,3
1480 IF V=0 THEN D$=DSKI$(D,0,16):LSET SFNO$=SFD$:DSKO$ D,0,16
1490 IF V=3 THEN D$=DSKI$(D,6, 7):LSET SFN3$=SFD$:DSKO$ D,6, 7
1500 LOCATE 0,12:COLOR 5:PRINT"OK !"
1510 '
1520 COLOR 7:PRINT:CLEAR 300:END

```

8-4-9 ファイルメニュー

プログラムの LOAD, RUN を行なうのに最適なファイルメニュープログラムを紹介します(リスト 8-11, リスト 8-12).

プログラムを起動すると図 8-35 のようなファイルメニュー(ドライブ 0)が表示されます。青いカーソルを移動させてファイルを選択し、 キーを押すとそのプログラムの実行、 キーを押すとプログラムのロードが行なわれます。

 ,  ,  , 	……カーソル移動
 + 	……カーソル移動(上へ 6 個)
 + 	……カーソル移動(下へ 6 個)
	……ロードおよび実行
	……ロードのみ
 ,  ,  , 	……ドライブの選択
	……コールドスタート
	……システムリセット
	……終了

なおファイル名の表示は、次のように区別されています。

[表示色]

白色	…… BASIC プログラム
黄色	…… BAISC データ
水色	…… マシン語

[先頭の文字]

- ・ バイナリファイル
- | アスキーファイル
- > ランダムファイル

```

109c/DO (d4:f1) Run=[CR],Load=[BS],Drv=[0]-[3],Cold=[ESC],Reset=[TAB],Quit=[CLS]
.SCTDMP 2 .FIPL.T 1
.SCTDMP.T 2 .FATSR 1
.ASCDMP 1 .FORMAT 6
.ASU 1 .DSKRW.M 1
.RUN.0 2 .DSKRW.T 5
.RUN.3 2
.FINFO 2
.FDISP 1
.FDISP.T 1
.EDIR 3
.OFAT 2
.FCOPYF 2
.FCOPY.T 2
.WSET 1
.HFILES 2
.HFILES.T 1
.HF/SUB.T 1
.FIPL 1

```

File Loader for F-BASIC Ver 3.0

図8-35 ファイルメニュープログラムの表示

リスト 8-11 ファイルメニュー(V3.0 用)

```

100 '*****
110 '* File Menu *
120 '* ( LIST 8-11 ) V3.0 *
130 '*****
140 '
150 GOTO 290
160 '
170 '◆◆ Cursor ◆◆
180 X=NF*18:Y=NF MOD 18:LINE(X*104,(Y+1)*10)-(X*104-1,(Y+2)*10-4),X
OR,1,BF
190 RETURN
200 '
210 DATA 255,255,0,0,189,143,54,189,207,119,126,171,244
220 DATA 126,124,4,52,16,230,3,141,247,53,16,167,3,57
230 DATA 142,96,31,157,222,108,2,108,5,157,222,108,2,204,1,1,237,5
240 DATA 157,222,108,2,108,5,166,5,129,17,38,244,57
250 DATA 10,0,110,0,0,15,0,0,-1
260 DATA 142,110,0,79,67,126,132,139
270 '
280 '■ Initialize ■
290 SCREEN 7,7:COLOR 7,0:WIDTH 80,20:CONSOLE,,0:FOR A=0 TO 7:COLOR=(A
.A):NEXT
300 FOR A=0 TO 3:ID$=ID$+CHR$(PEEK(A-1030)):NEXT
310 IF ID$<>"月" THEN BEEP:PRINT"Not Ver 3.0 System !!!":NEW
320 A=65057!+PEEK(&HFE04):POKE &HFD93,1:POKE A,8:POKE A+28,&H1C:POKE &
HFD93,0
330 POKE 14,0:RESTORE 210:FOR A=0 TO 12:READ AM:POKE A,AM:NEXT
340 AD=&H6000:RESTORE 220:FOR A=AD TO AD+13:READ AM:POKE A,AM:NEXT
350 '■ Main ■
360 CLS:CLEAR 1000,&H6000:DEFINT A-Y:DIM D$(7):DEFUSR8=&HBD5:DEFUSR9=
&H6003

```

```

370 DRV=PEEK(14):LOCATE 0,0:COLOR 6:PRINT USING"###c/DH":DSKF(DRV),DRV
;
380 DV=PEEK(&H71FA):FV=PEEK(&H71FB)
390 COLOR 4:PRINT USING" (d#:f!)":DV+1:HEX$(FV):
400 LOCATE 25,19,0:COLOR 1:PRINT"File Loader for F-BASIC Ver 3.0":
410 FIELD #0,32 AS D$(0),32 AS D$(1),32 AS D$(2),32 AS D$(3),
      32 AS D$(4),32 AS D$(5),32 AS D$(6),32 AS D$(7)

420 FAT$=DSKI$(DRV,1,1):DS=4:AF=0:NF=0:ND=AF:NL$=CHR$(0)
430 OPEN "O",#1,"SCRN:"
440 D$=DSKI$(DRV,1,DS)
450 FOR I=0 TO 7
460   D$=D$(I):DT=ASC(D$)
470   IF DT=255 OR INKEY$="" THEN I=9:GOTO 530
480   IF DT=0 THEN 530
490   LOCATE 13*(ND*18),ND MOD 18+1:COLOR 7-ASC(MID$(D$,12,1)):T=165:P
PRINT " ";
500   IF ASC(MID$(D$,13,1))=255 THEN T=124:IF ASC(MID$(D$,14,1))=255 T
HEN T=62
510   PRINT #1,CHR$(T) USRB(" ") LEFT$(D$,8) USRB(USR9(ASC(MID$(D$,15,1
)))));
520   AF=AF+1:ND=AF MOD 108
530 NEXT
540 IF I=8 THEN DS=DS+1:GOTO 440
550 CLOSE #1
560 IF AF>ND THEN AF=108
570 LOCATE 16,0:COLOR 3
580 PRINT"Run=[CR],Load=[BS],Drv=[0]-[3],Cold=[ESC],Reset=[TAB],Quit=[
CLS]";
590 CMD$=CHR$( 0)+CHR$(28)+CHR$(29)+CHR$(30)+CHR$(31)+CHR$( 6)+CHR$(
2)
      +CHR$(25)+CHR$(26)+CHR$(13)+CHR$( 8)+CHR$(12)+CHR$(
27)+CHR$( 9)+"0123"
600 LOCATE 0,18,1:COLOR 7
610 BEEP 1:GOSUB 180:BEEP 0
620 CMD=0:WHILE CMD<2:CMD=INSTR(CMD$,INKEY$):WEND
630 ON CMD GOTO 630,650,660,670,690,650,660,680,700,720,730,880,900,99
0,
      860,860,860,860
640 '■ File select ■
650 GOSUB 180:NF=Nf+18:IF NF<AF THEN 610 ELSE NF=Nf-18:GOTO 610 ' righ
t
660 GOSUB 180:NF=Nf-18:IF NF=>0 THEN 610 ELSE NF=Nf+18:GOTO 610 ' left
670 GOSUB 180:NF=Nf-1: IF NF=>0 THEN 610 ELSE NF=AF-1: GOTO 610 ' up
680 GOSUB 180:NF=Nf-6: IF NF=>0 THEN 610 ELSE NF=AF-1: GOTO 610 ' + up
690 GOSUB 180:NF=Nf+1: IF NF<AF THEN 610 ELSE NF=0: GOTO 610 ' down
700 GOSUB 180:NF=Nf+6: IF NF<AF THEN 610 ELSE NF=0: GOTO 610 ' + do
wn
710 '■ Run & Load ■
720 R=3 '(RUN)
730 CX=13*(NF*18):CY=Nf MOD 18+1:IF SCREEN(CX,CY)=62 THEN BEEP:GOTO 62
0
740 FD$=HEX$(DRV)+" ":FOR X=CX+1 TO CX+8:FD$=FD$+CHR$(SCREEN(X,CY)):NE
XT
750 FT=7-SCREEN(CX,CY,1):IF FT<2 THEN Z=PEEK(&H59D)*256+PEEK(&H59E):GO
TO 780
760 IF FV=0 THEN BEEP:GOTO 620
770 OPEN"I",1,FD$:D$=INPUT$(3,1):Z=ASC(INPUT$(1,1))*256+ASC(INPUT$(1,1
)):CLOSE
780 POKE &H2DD,8:POKE &H2E6,&H80+DRV
790 POKE &H2EE,R:POKE &H2EF,0:POKE &H2F0,2-FT:POKE &H2F1,0:POKE &H2F2,
0
800 FOR I=1 TO 8:POKE &H2DD+I,ASC(MID$(FD$,I+2,1)):NEXT
810 Z=Z- 1:H1=FIX(Z/256):L1=Z-H1*256
820 Z=Z-300:H0=FIX(Z/256):L0=Z-H0*256
830 POKE &H3F,H0:POKE &H40,L0:POKE &H45,H1:POKE &H46,L1
840 WIDTH,25:CLS:EXEC 4

```

```

850 '■ Drive select ■
860 D=CMD-15:IF D<=DV THEN POKE 14,D:GOTO 360 ELSE BEEP:GOTO 620
870 '■ Quit ■
880 COLOR 7:WIDTH,25:CLEAR 300,PEEK(&H59D)*256+PEEK(&H59E):NEW:END
890 '■ Cold start ■
900 GOSUB 180:LOCATE 16,0:COLOR 2:PRINT CHR$(5) " ";
910 IF LEFT$(DSKI$(0,0,29),3)="■" THEN 930
920 BEEP:PRINT">> Different System Disk !! <<":FOR I=0 TO 2000:NEXT:GO
TO 570
930 PRINT"<< Disk BASIC cold start >> ";
940 COLOR 5:PRINT"## Just a moment please! ##"
950 AD=&H6000:RESTORE 230:READ MD
960 WHILE MD=>0:POKE AD,MD:AD=AD+1:READ MD:WEND:EXEC &H6000
970 RESTORE 260:FOR AD=4 TO 11:READ MD:POKE AD,MD:NEXT:POKE &H6F8F,&H2
0:EXEC 4
980 '■ Reset ■
990 COLOR 7:WIDTH,25:PRINT"Reset.":TIME$="00:00:00":EXEC-512

```

リスト 8-12 ファイルメニュー(V3.3 用)

```

100 '*****
110 '* File Menu *
120 '* ( LIST 8-12 ) V3.3 *
130 '*****
140 GOTO 230
150 '
160 DATA 189,159,229,189,221,84,126,147,199
170 DATA 134,3,183,253,136,126,138,153,52,16,230,3,141,242,53,16,167,3
,57
180 '
190 '◆◆ Cursor ◆◆
200 X=NF*18:Y=NF MOD 18:LINE(X*104,(Y+1)*10)-(X+1)*104-1,(Y+2)*10-4),X
OR,1,BF
210 RETURN
220 '
230 '■ Initialize ■
240 FOR A=0 TO 5:ID$=ID$+CHR$(PEEK(A-1030)):NEXT
250 IF ID$(<"X">)+CHR$(15)+"4," THEN BEEP:PRINT"Not Ver 3.3 System.":N
EW
260 CLEAR ,&H5FFF,512:DEFINT A:POKE &H33C,0
270 SCREEN 0:SCREEN 7,7,1,1:COLOR 7,0:WIDTH 80,20:CONSOLE,...0,0
280 FOR A=0 TO 7:COLOR=(A,A):NEXT:PALETTE 0:PRINT CHR$(27)+"h
290 RESTORE 160:FOR A=0 TO 12:READ AM:POKE A+&H340,AM:NEXT
300 AD=&H6000:RESTORE 170:FOR A=AD TO AD+18:READ AM:POKE A,AM:NEXT
310 '■ Main ■
320 CLS:CLEAR:DEFINT A-Y:DIM D$(7):DEFUSR7=&HE944:DEFUSR8=&HE94B:DEFUS
R9=&H6008
330 DRV=PEEK(&H33C):LOCATE 0,0:COLOR 6:PRINT USING"###c/DH":DSKF(DRV),
DRV;
340 DV=4:FOR A=0 TO 3:DV=DV+(PEEK(A+&HFC30)=255):NEXT:FV=PEEK(&H9024)
350 COLOR 4:PRINT USING" (d#:f!)" :DV:HEX$(FV);
360 LOCATE 20,19,0:COLOR 1:PRINT"File Loader for F-BASIC Ver 3.3";
370 FIELD 0,32 AS D$(0),32 AS D$(1),32 AS D$(2),32 AS D$(3),
32 AS D$(4),32 AS D$(5),32 AS D$(6),32 AS D$(7)
380 FAT$=DSKI$(DRV,1,1):DS=4:AF=0:NF=0:ND=AF
390 OPEN "D",#1,"SCRN:"
400 D$=DSKI$(DRV,1,DS)
410 FOR I=0 TO 7
420 D$=D$(I):DT=ASC(D$)
430 IF DT=255 OR INKEY$="" THEN I=9:GOTO 490
440 IF DT=0 THEN 490
450 LOCATE 13*(ND*18),ND MOD 18+1:COLOR 7-ASC(MID$(D$,12,1)):T=165:P
RINT " ";

```

```

460 IF ASC(MID$(D$,13,1))=255 THEN T=124:IF ASC(MID$(D$,14,1))=255 T
HEN T=62
470 PRINT #1,CHR$(USR8(T)) LEFT$(D$,8) USR7(USR9(ASC(MID$(D$,15,1)))
):
480 AF=AF+1:ND=AF MOD 108
490 NEXT
500 IF I=8 THEN DS=DS+1:GOTO 400
510 CLOSE #1
520 IF AF>ND THEN AF=108
530 LOCATE 16,0:COLOR 3
540 PRINT"Run=[CR],Load=[BS],Drv=[0]-[3],Cold=[ESC],Reset=[TAB],Quit=[
CLS]";
550 CMD$=CHR$( 0)+CHR$(28)+CHR$(29)+CHR$(30)+CHR$(31)+CHR$( 6)+CHR$(
2) +CHR$(25)+CHR$(26)+CHR$(13)+CHR$( 8)+CHR$(12)+CHR$(
27)+CHR$( 9)+"0123"
560 LOCATE 0,18:COLOR 7
570 BEEP 1:GOSUB 200:BEEP 0
580 CMD=0:WHILE CMD<2:CMD=INSTR(CMD$,INKEY$):WEND
590 ON CMD GOTO 590,610,620,630,650,610,620,640,660,680,690,820,850,93
0,
800,800,800,800
600 '■ File select ■
610 GOSUB 200:NF=NF+18:IF NF<AF THEN 570 ELSE NF=NF-18:GOTO 570 ' righ
t
620 GOSUB 200:NF=NF-18:IF NF=>0 THEN 570 ELSE NF=NF+18:GOTO 570 ' Left
630 GOSUB 200:NF=NF-1: IF NF=>0 THEN 570 ELSE NF=AF-1: GOTO 570 ' up
640 GOSUB 200:NF=NF-6: IF NF=>0 THEN 570 ELSE NF=AF-1: GOTO 570 ' + up
650 GOSUB 200:NF=NF+1: IF NF<AF THEN 570 ELSE NF=0: GOTO 570 ' down
660 GOSUB 200:NF=NF+6: IF NF<AF THEN 570 ELSE NF=0: GOTO 570 ' + do
wn
670 '■ Run & Load ■
680 R=3 '(RUN)
690 CX=13*(NF*18):CY=NF MOD 18+1:IF SCREEN(CX,CY)=62 THEN BEEP:GOTO 58
0
700 FD$=HEX$(DRV)+"":FOR X=CX+1 TO CX+8:FD$=FD$+CHR$(SCREEN(X,CY)):NE
XT
710 FT=7-SCREEN(CX,CY,1):IF FT<2 THEN Z=PEEK(&H59D)*256+PEEK(&H59E):GO
TO 740
720 IF FV=0 THEN BEEP:GOTO 580
730 OPEN"I",1,FD$:D$=INPUT$(3,1):Z=ASC(INPUT$(1,1))*256+ASC(INPUT$(1,1
)):CLOSE
740 POKE &H2DD,8:POKE &H1D,&H80+DRV
750 POKE &H2EE,R:POKE &H2EF,0:POKE &H2F0,2-FT:POKE &H2F1,0:POKE &H2F2,
0
760 FOR I=1 TO 8:POKE &H2DD+I,ASC(MID$(FD$,I+2,1)):NEXT
770 CLEAR,2-1,512:WIDTH,25:CLS:LOCATE 0,0,1
780 EXEC &H340
790 '■ Drive select ■
800 D=CMD-15:IF D<=DV THEN POKE &H33C,D:GOTO 320 ELSE BEEP:GOTO 580
810 '■ Quit ■
820 COLOR 7:WIDTH,25:LOCATE 0,0,1
830 CLEAR,PEEK(&H59D)*256+PEEK(&H59E)-1,512:NEW:END
840 '■ Cold start ■
850 GOSUB 200:LOCATE 16,0:COLOR 2:PRINT CHR$(5) " ";
860 IF ASC(DSKI$(0,0,15))=0 THEN 880
870 BEEP:PRINT">> Different System Disk ! <<":FOR I=0 TO 2000:NEXT:GOT
O 530
880 PRINT"<< Disk BASIC cold start >> ";
890 COLOR 5:PRINT"## Just a moment please! ##"
900 POKE &HFD86,&H38:POKE &H65AA,&H20:POKE &HFD86,&H36
910 EXEC &H922C
920 '■ Reset ■
930 COLOR 7:POKE &HFD13,0:PRINT"Reset."
940 POKE &HFD02,0:POKE &HFD10,128:POKE &HFD86,&H36:POKE &HFD10,0:EXEC
&H6000

```

このプログラムでは、BASIC に関する様々なテクニックを駆使しています。興味のある方は、ソースリストを解析してみてください。

ドライブ数、ファイル数は、DISK-BASIC のワークエリアを参照しています。クラスタ数のカウントは、FILES で使うシステムルーチンを利用します。ロードおよび実行では、LOAD (LOADM) に必要なパラメータを設定して、その ROM 内ルーチン呼び出しています。コールドスタートは、BASIC のイニシャライズルーチンをオートスタートしないように変更した上で実行しています。ですからオートスタートの指定がしてあっても、オートスタートしません。ソフトウェアリセットでは、ソフト的にリセットできるものはすべてリセットし、ブート ROM (イニシエータ ROM) を呼び出しています。このときディスクドライブのモーターを止めないので、リセットキーを押すのよりも速く立ち上がります。なお、FM77AV の V3.0 でリセットしたときには、FDC ステッピングレートを 20ms から 6ms に切り換えています。

9-1 プリンタに対する BIOS

F-BASIC では、ほとんどの周辺機器に対する入出力処理を BIOS を経由して行なっています。そしてプリンタに関係する BIOS としては、次のものが用意されています(図 9-1)。

- ① LPOUT : プリンタにデータを出力します。
- ② LPCHK : プリンタが Ready か、チェックします。
- ③ HDCOPY : 画面のハードコピーをとります。(HARDC1 相当)
- ④ SCREEN : 画面のハードコピーをとります。(HARDC2 相当)

①の LPOUT は、データバッファの内容をそのままプリンタに出力します。出力したいデータの先頭アドレスとデータの長さをパラメータとしてセットします。

②の LPCHK は、プリンタの状態をチェックします。プリンタが Ready 状態であればステータスは正常終了を示し、ペーパー・エンプティ(用紙切れ)状態のときは 50 を、NOT Ready のときには 51 を示します。

③の HDCOPY、④の SCREEN は画面のハードコピーをとるもので、それぞれ F-BASIC の HARDC1、HARDC2 命令に対応しています。そして F-BASIC V3.3 では動作しません。パラメータとしては作業領域(209 バイト以上)の先頭アドレスと、印字カラーの指定(HDCOPY では黒、灰の 2 種類)をセットします。

一般に画面のハードコピーをとる場合、プリンタの持つグラフィック処理の制御コマンドが問題となってきます。BIOS のハードコピールーチンでは、初期の FM シリーズ純正プリンタであるエプソン系の制御コマンドを採用しています。しかし、エプソン系の制御コマンドを持たない富士通の漢字プリンタ等でも、この BIOS のハードコピールーチンで使用しているコマンドだけは、受け付けるように工夫されています。

F-BASIC V3.3 では、ハードコピー関係の BIOS(HDCOPY、SCREEN)が削除されました。そして、新しくレジスタインターフェースの BIOS がサポートされました。

- ① APRTOT : 1 文字出力
- ② PRTCHK : Ready チェック

このレジスタインターフェースの BIOS について少し説明します。

たとえば、LPOUT を使用して文字列をプリントする場合を考えてみます。この BIOS を使う

には、データバッファの先頭アドレスとデータの長さ、そしてリクエスト番号の計5バイトのパラメータが必要となります。プリントする文字列が長い場合は問題にはならないのですが、短い場合はどうでしょうか？. 1バイトのデータをプリントするのにも5バイトのパラメータが必要

●LPOUT

相対値	内 容	ラベル名	ユーザーセット パラメータ	BIOSセット パラメータ
0	リクエスト番号	RQNO	14	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4, 5	データバイト数(16ビット)	RCBLNH	○	

●LPCHK

相対値	内 容	ラベル名	ユーザーセット パラメータ	BIOSセット パラメータ
0	リクエスト番号	RQNO	23	
1	エラーステータス	RCBSTA		○

RCBSTAの値	意 味
50	ペーパーエンプティ
51	ノットレディ

●SCREEN

相対値	内 容	ラベル名	ユーザーセット パラメータ	BIOSセット パラメータ
0	リクエスト番号	RQNO	5	
1	エラーステータス	RCBSTA		○
2, 3	データバッファ先頭アドレス	RCBDBA	○	
4	印字カラー指定バイト	RCBCDT	○	

RCBCDT, RCBCTB, RCBCTGの値

7	6	5	4	3	2	1	0
☐	☐	☐	☐	☐	☐	☐	☐

白 黄 水色 緑 紫 赤 青 黒

“1”のビットが黒(灰)で印字される

●HDCOPY

相対値	内 容	ラベル名	ユーザーセット パラメータ	BIOSセット パラメータ
0	リクエスト番号	RQNO	15	
1	エラーステータス	RCBSTA		○
2, 3	バッファ先頭アドレス	RCBDBA	○	
4	黒印字カラー指定バイト	RCBCTB	○	
5	灰印字カラー指定バイト	RCBCTG	○	

●APRTOT(レジスタインターフェース)

AccB, リクエスト番号

AccA, 印字データ

エラーステータス, \$FF98

●PRTCHK(レジスタインターフェース)

AccB, リクエスト番号

AccA, プリンタポート番号

エラーステータス, \$FF98

図9-1 プリンタ関係のBIOSパラメータ

なのです。

レジスタインターフェースではこの点に配慮がなされています。たとえばプリンタ1文字出力 (APRTOT) では、AccA に出力したいデータ、AccB にリクエスト番号をセットするだけです。このレジスタインターフェースについては、第3章で説明していますので、併せて参照してください。

BIOS の LPOUT を使用したサンプルプログラムをリスト 9-1 に示します。またレジスタインターフェースの APRTOT を使用したサンプルプログラムをリスト 9-2 に示します。どちらも \$5000 番地からプログラムを入力して、EXEC &H5000  にて実行してください。ただし APRTOT の方は、V3.3 でしか動作しません。

リスト 9-1 LPOUT のサンプルプログラム

```

01000 *****
01020 *      LPOUT サンプルプログラム      *
01030 *      ( LIST 9-1 )      V3.0/V3.3      *
01040 *****
01050          OPT      NOGEN
01060 5000          ORG      $5000
01080          FBFA    BIOS  EQU      $FBFA
01090          *
01100 5000 8E      5007  ENTRY  LDH      #RCB
01110 5003 6E      9F FBFA    JMP      [BIOS]
01120          *
01130 5007          0E      RCB      FCB      14.0
01140 5009          500D      FDB      DATA_S
01150 500B          0012      FDB      DATA_E-DATA_S
01170 500D          46      DATA_S FCC      'FM-Techknow 77AV'
01180 5010          00      FCB      $00,$0A
01190          501F      DATA_E EQU      *
01200          *
01210          5000          END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=501E
PROGRAM ENTRY ADDR=5000

```

リスト 9-2 APRTOT のサンプルプログラム

```

01000 *****
01020 *      APRTOT サンプルプログラム      *
01030 *      ( LIST 9-2 )      V3.3      *
01040 *****
01050          OPT      NOGEN
01060 5000          ORG      $5000
01070          *
01080          FBA7    BIOS  EQU      $FBA7
01090          *
01100 5000 CE      5015  ENTRY  LDH      #DATA_S
01110 5003 86      12      LDA      #DATA_E-DATA_S
01120 5005 34      02      PSHS      A
01130 5007 A6      C0      LOOP   LDA      .U+
01140 5009 C6      02      LDB      #2

```

```

01150 500B AD 9F FBA7 JSR [BIOS]
01160 500F 6A E4 DEC ,S
01170 5011 26 F4 5007 BNE LOOP
01180 5013 35 82 PULS A,PC
01190
01200 5015 46 DATA_S FCC 'FM-Techknow 77AV'
01210 5025 0D FCB $0D,$0A
01220 5027 DATA_E EQU *
01230 *
01240 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5026
PROGRAM ENTRY ADDR=5000

```

9-2 画面コピー機能

F-BASIC V3.0/V3.3 は, HARDC 命令によって画面上に表示されている文字やグラフィックを, プリンタに出力する機能を持っています。画面コピーには, 機能の異なる3種類のものが用意されています。

HARDC ……テキスト(コンソールバッファ内の文字)のみをプリンタにコピーします。

HARDC1 ……画面の内容を 90°回転して, さらに画面の1ドットをプリンタの横方向4ドットに拡大コピーします。そして, 黒, 灰, 白の濃淡レベルでコピーされます。

黒・赤・緑・白 …… 黒

紫・水・黄 …… 灰

黒 …… 白

HARDC2 ……画面の内容をそのままコピーします。画面上の1ドットは, そのままプリンタの1ドットに対応します。濃淡はつきません。

F-BASIC V3.3 では, 画面コピーを行なう範囲を指定できるようになっています。そして 4096 色モード時には, HARDC1, HARDC2 は使用できません。

それでは, HARDC, HARDC1, HARDC2 によってどのようにコピーされるか調べてみることにしましょう。リスト 9-3 を入力して, RUN  してください。実行は, V3.3 でも 8 色 2 画面モードなら可能です。

実行すると 2 つの円等が描かれ, BEEP 音がして止まりましたね。ここで 0~2 の Key を押してみましょう。それぞれの画面コピーモードでのハードコピーがとれます(図 9-2, 図 9-3, 図 9-4),

リスト 9-3 画面コピーのサンプルプログラム

```

1000 *****
1001 '*   HARDC TEST PROGRAM           *
1002 '*   ( LIST 9-3 )   V3.0/V3.3     *
1003 '*   "3" --> LIST 9-4   カ"   ヒツヨウ テ"ス   *
1004 '*   "4" --> LIST 9-5   カ"   ヒツヨウ テ"ス   *
1005 *****
1010 CLEAR 300,&H5000
1020 WIDTH40,25
1030 PRINT"CIRCLE(320,100),150,7,.5"
1040 CIRCLE(320,100),150,7,.5
1050 LOCATE 0.23:PRINT"CIRCLE(100,140),70,5,.5,,,F"
1060 CIRCLE(100,140),70,5,.5,,,F
1070 FOR I=1 TO 7
1080   LOCATE 30,I*3
1090   COLOR I
1100   PRINT"COLOR ":I
1110   LOCATE 30,I*3+1
1120   PRINT"      "
1130 NEXT
1140 BEEP
1150 A$=INKEY$:A$=INPUT$(1)
1160 IF A$="0" THEN HARDC
1170 IF A$="1" THEN HARDC1
1180 IF A$="2" THEN HARDC2
1190 IF A$="3" THEN LOADM"L9-4M",.R
1192 IF A$="4" THEN LOADM"L9-5M",.R
1200 GOTO 1140

```

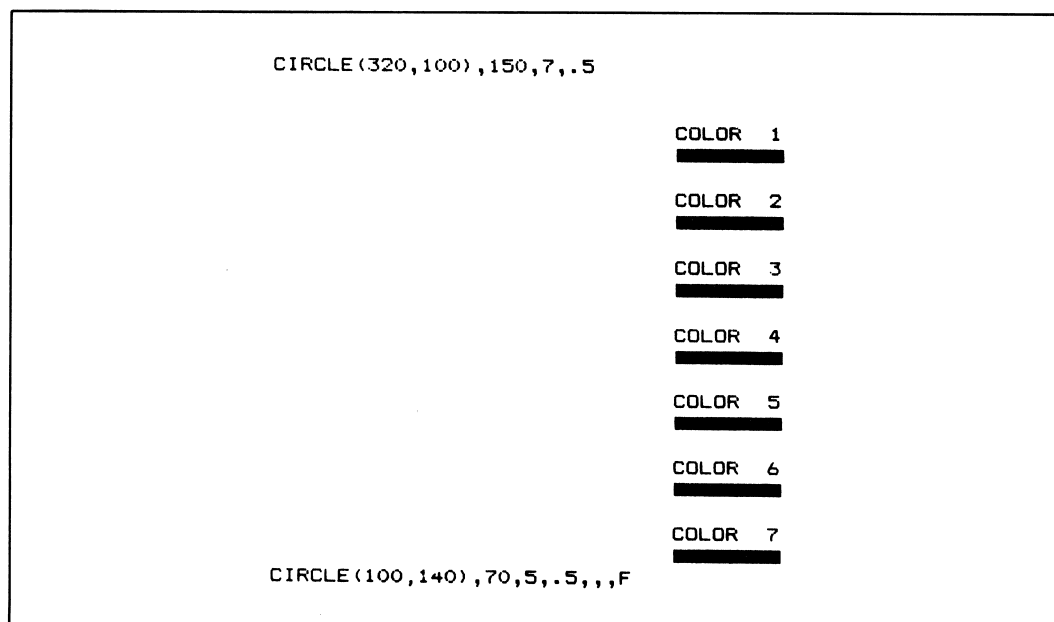


図9-2 HARDCのサンプル

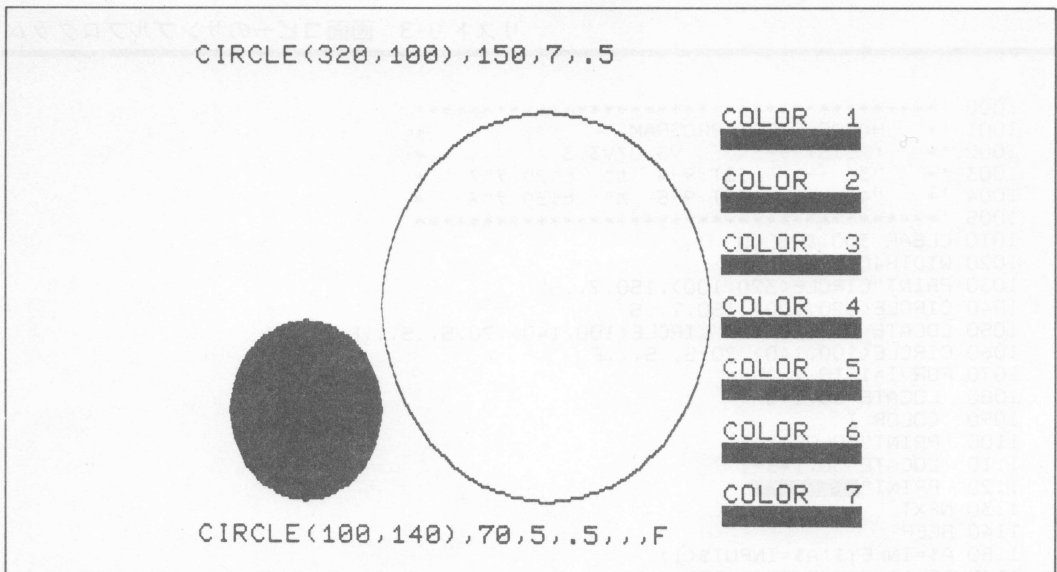


図9-3 HARDC1のサンプル

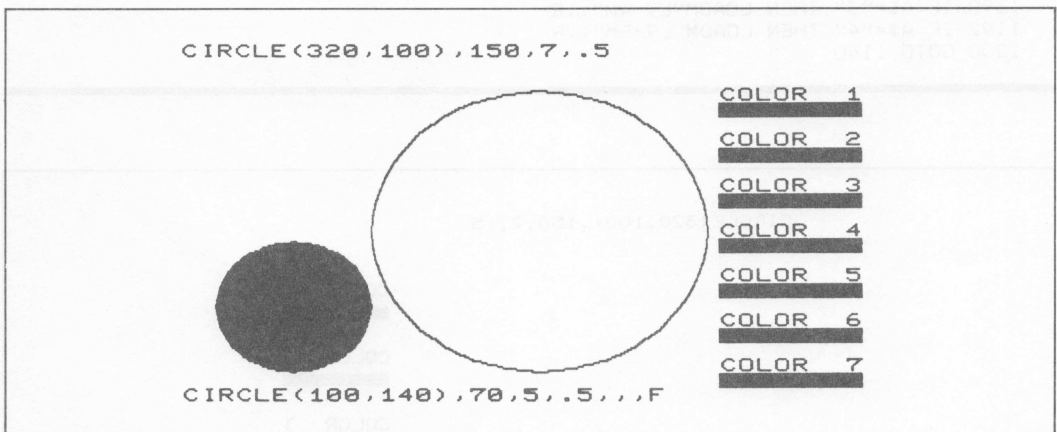


図9-4HARDC2のサンプル

9-3 拡張画面コピープログラム

先ほどの出力例を見ればおわかりのように、BASICでサポートしている画面コピー機能だけでは、少し物足りないですね。たとえば、

- ① 色の濃淡がわかりにくい。
- ② HARDC1では縦長、HARDC2では横長にコピーされる。
- ③ プリンタの印字ヘッドが、しゃっくりをする。

等々があります。さらに F-BASIC V3.3 では、4096 色モードでグラフィックのコピーがとれないという欠点があります。

そこではっきりと濃淡印字のできる拡張画面コピープログラムを、8 色モード用、4096 色モード用の各々作成しました。対象のプリンタは、エプソン系の制御コマンドを持つ 9 ピンのプリンタです。たとえば、MP80Type II, Type III, RP80, FP80, MB27401 などです。

このプログラムでは、8 色モード時 8 レベルの濃淡、4096 色モード時 26 レベルの濃淡で画面コピーを行ないます。十分、実用に耐えるレベルかと自負しています。

また、縦長、横長の問題についても配慮しました。しかし FM77AV では、真円そのものの定義があいまいであるため(CIRCLE(320,100),150 が画面上では縦長となってしまう)、厳密な意味での真円でコピーされていないかもしれません。HARDC1 でコピーしたものより、少し横長になります。

そしてこのプログラムでは、1 行文のデータをバッファにためて一気に処理するようにしました。これによりプリンタの印字ヘッドがしゃっくりする(動作にむらができる)のをなくしました。

実際のプログラムをリスト 9-4(8 色モード用)、リスト 9-5(4096 色モード用)に示します。実行はいずれも EXEC &H5000  とします。

先ほどのリスト 9-3 にリンクするときには、リスト 9-4(またはリスト 9-5)を“L9-4M”(または“L9-5M”)というファイル名で SAVEM しておいて、円の表示後、“3”または“4”を入力します。

SAVEM“L9-4M”, &H5000, &H541E, &H5000

そのときの出力例を、図 9-5 に示します。HARDC1, HARDC2 でコピーしたものより、円らしくコピーされていると思います。

さらに 8 色モードでの出力例として、FM 音源カード付属のデモプログラムの画面を図 9-6 に示します。4096 色モードでの出力例としては、F-BASIC V3.3 の README の画面を図 9-7 に示します。

リスト 9-4 拡張画面コピープログラム(8 色モード用)

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 53 99 00 00 00 00 00 00 00 00 00 00 00 00 00 : 6A
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

50F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 7E 53 99 00 00 00 00 00 00 00 00 00 00 00 00 00 : 6A

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S100 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S110 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S120 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S130 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S1F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S200 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S210 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S220 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S230 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S240 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S250 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S260 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S270 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S280 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S290 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S2A0 : 00 00 00 00 00 00 00 00 00 00 00 00 8E 00 00 BF 50 : 9D
S2B0 : 10 34 10 30 88 13 BF 50 14 FC 52 A9 FD 50 0E C3 : 57
S2C0 : 00 07 FD 50 12 86 1D B7 50 0D 8E 50 03 86 11 A7 : 3C
S2D0 : 84 CC 50 0B ED 02 CC 00 0B ED 04 C6 40 ED 06 AD : 08
S2E0 : 9F FB FA 35 10 CE 50 0F 31 89 50 51 C6 14 A6 C8 : A9
S2F0 : 28 A7 A9 01 90 A6 C8 14 A7 A9 00 C8 A6 C0 A7 A0 : 50
-----
[cs] : 5B A9 00 C1 27 0F C0 2A 47 28 34 66 AC 97 31 CF : 31

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S300 : 5A 26 EB 30 88 14 8C 00 C8 25 A3 39 7F 50 0B 7F : E5
S310 : 50 0C 7F 50 0D 86 01 5F A5 A4 27 02 CB 03 A5 A9 : AC
S320 : 00 C8 27 02 CB 06 A5 A9 01 90 27 02 CB 0C 34 42 : 17
S330 : 33 C5 EC C4 A4 E4 E4 BA 50 0B FA 50 0C FD 50 : 80
S340 : 0B A6 42 A4 E4 BA 50 0D B7 50 00 35 42 48 24 C7 : 50
S350 : 6E 9F FB FA 8E 50 03 86 0E A7 84 CC 53 85 ED 02 : 35
S360 : CC 00 14 ED 04 AD 9F FB FA CC 50 0B ED 02 CC 00 : F4
S370 : 03 ED 04 10 8E 51 18 86 C8 34 02 8D 8F 31 3F 6A : 75
S380 : E4 26 FB 35 82 20 20 20 20 20 20 20 20 20 20 : 19
S390 : 20 20 20 20 20 1B 4C 58 02 CC 00 00 FD 52 A9 17 : 3C
S3A0 : FF 09 CE 53 EF 8D AD CC 53 E4 ED 02 CC 00 04 ED : 01
S3B0 : 04 AD 9F FB FA CE 54 07 8D 9A CC 53 E8 ED 02 CC : 57
S3C0 : 00 04 ED 04 AD 9F FB FA FC 52 A9 C3 00 0B FD 52 : 47
S3D0 : A9 83 02 80 25 C9 CC 53 EC ED 02 CC 00 03 ED 04 : 56
S3E0 : 6E 9F FB FA 1B 4A 01 0D 1B 4A 17 0D 1B 32 0C 00 : 57
S3F0 : 00 00 55 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 : 51
-----
[cs] : 43 13 96 02 80 D3 55 A5 B3 93 7A E0 62 07 C1 33 : 38

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S400 : FF FF 00 FF FF FF FF 00 00 00 00 00 00 00 00 : FA
S410 : 00 55 00 00 FF 00 00 FF 00 FF FF 00 FF FF FF 00 : 4E
S420 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

5430 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5440 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5450 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5460 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5470 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5480 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5490 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
54F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : FF 54 00 FF FE FF FF FF 00 FF FF 00 FF FF FF 00 : 48

```

SAVEM "L9-4M",&H5000.&H541E.&H5000

リスト 9-5 拡張画面コピープログラム(4096色モード用)

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 54 6E 00 00 00 00 00 00 00 00 00 00 00 00 : 40
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 7E 54 6E 00 00 00 00 00 00 00 00 00 00 00 00 : 40

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5110 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5120 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5130 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5200 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

5210 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5220 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5230 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5240 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5250 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5260 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5270 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5280 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5290 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5300 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5310 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5320 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5330 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5340 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5350 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5360 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5370 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5380 : 00 8E 00 00 BF 50 10 34 10 30 09 BF 50 14 FC 53 : 9C
5390 : 7F FD 50 0E C3 00 03 FD 50 12 86 1D 87 50 0D 8E : 44
53A0 : 50 03 86 11 A7 84 CC 50 0B ED 02 CC 00 0B ED 04 : F3
53B0 : C6 54 ED 06 AD 9F FB FA EC E4 58 49 58 49 10 8E : FE
53C0 : 50 5F 31 AB CE 50 0F C6 28 34 04 A6 C0 44 44 34 : 00
53D0 : 02 5F A6 C4 48 59 48 59 48 59 EB E4 E7 E4 E6 C0 : EE
53E0 : C4 0F EB E0 E7 A0 6A E4 26 E1 35 14 30 0A 8C 00 : 89
53F0 : C8 25 91 39 7F 50 0B 7F 50 0C 7F 50 0D 86 03 E6 : 87
-----
[cs] : 73 D4 16 AD 52 0C A6 FD 3D 8D 8C DF 43 70 BF 4D : FF
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5400 : A2 58 EB A4 34 42 33 C5 EC C4 A4 E4 E4 E4 BA 50 : 01
5410 : 0B FA 50 0C FD 50 0B A6 42 A4 E4 BA 50 0D B7 50 : 47
5420 : 0D 35 42 48 48 24 D8 6E 9F FB FA 8E 50 03 86 0E : 87
5430 : A7 84 CC 54 5A ED 02 CC 00 14 ED 04 AD 9F FB FA : A6
5440 : CC 50 0B ED 02 CC 00 03 ED 04 10 8E 53 7F 86 C8 : 94
5450 : 34 02 8D A0 6A E4 26 FA 35 82 20 20 20 20 20 20 : 48
5460 : 20 20 20 20 20 20 20 20 20 20 1B 4C 58 02 CC 00 : CD
5470 : 00 FD 53 7F 17 FF 0A CE 54 C4 8D AF CC 54 B9 ED : D7
5480 : 02 CC 00 04 ED 04 AD 9F FB FA CE 55 12 8D 9C CC : 2E
5490 : 54 BD ED 02 CC 00 04 ED 04 AD 9F FB FA FC 53 7F : D0
54A0 : C3 00 04 FD 53 7F 83 01 40 25 C9 CC 54 C1 ED 02 : 18
54B0 : CC 00 03 ED 04 6E 9F FB FA 1B 4A 01 0D 1B 4A 17 : B1
54C0 : 0D 1B 32 0C 00 00 00 44 00 00 44 11 00 44 11 22 : 76
54D0 : 44 11 AA 44 11 AA 44 11 AA 44 11 AA 44 11 AA 44 : 3F
54E0 : 11 AA 44 11 EE 44 11 FF 00 11 FF 00 00 FF 00 22 : 83
54F0 : FF 00 AA FF 00 EE FF 00 FF FF 22 FF FF 22 FF FF : D3
-----
[cs] : C7 D9 12 C8 85 3F 8F 6C 45 1C 3D B0 78 63 FD 68 : C7
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5500 : 22 FF FF AA FF FF AA FF FF BB FF FF BB FF FF : E1
5510 : FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 : FE
5520 : 00 22 00 00 22 88 00 22 CC 00 33 CC 00 33 CC 00 : 88
5530 : 33 CC 00 33 CC 00 FF CC 00 FF FF 00 FF FF 00 FF : C4
5540 : FF 00 FF FF 00 FF FF 00 FF FF 00 FF FF 88 FF FF : 7D
5550 : 99 FF FF 99 FF FF DD FF FF DD FF FF FF FF FF FF : E0
5560 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5570 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5580 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

5590 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
55F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

[cs] : EC EB FD 75 EC 85 85 EC C9 96 30 C9 88 88 C9 FC : B8

```

```

SAVEM "L9-5M", &H5000, &H555F, &H5000

```

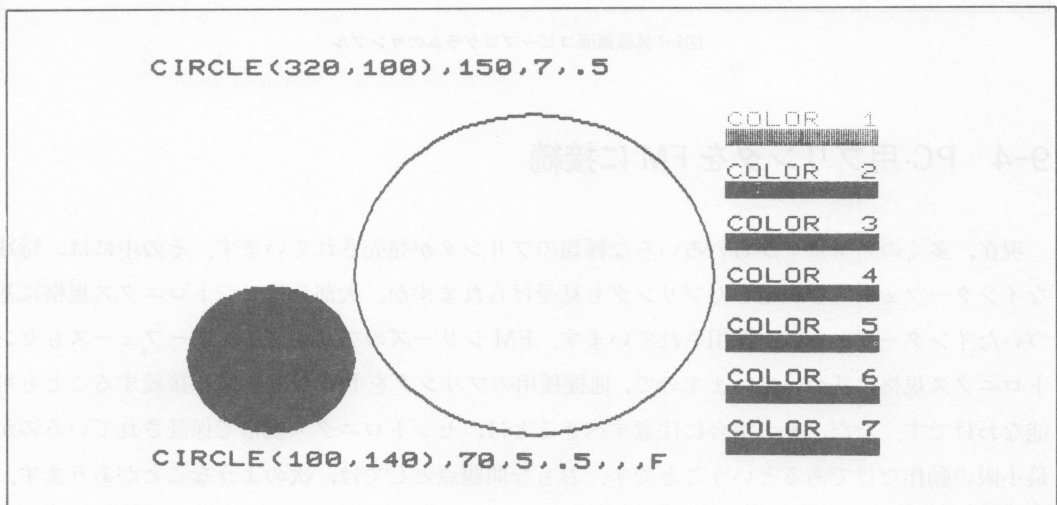


図9-5 拡張画面コピープログラムのサンプル

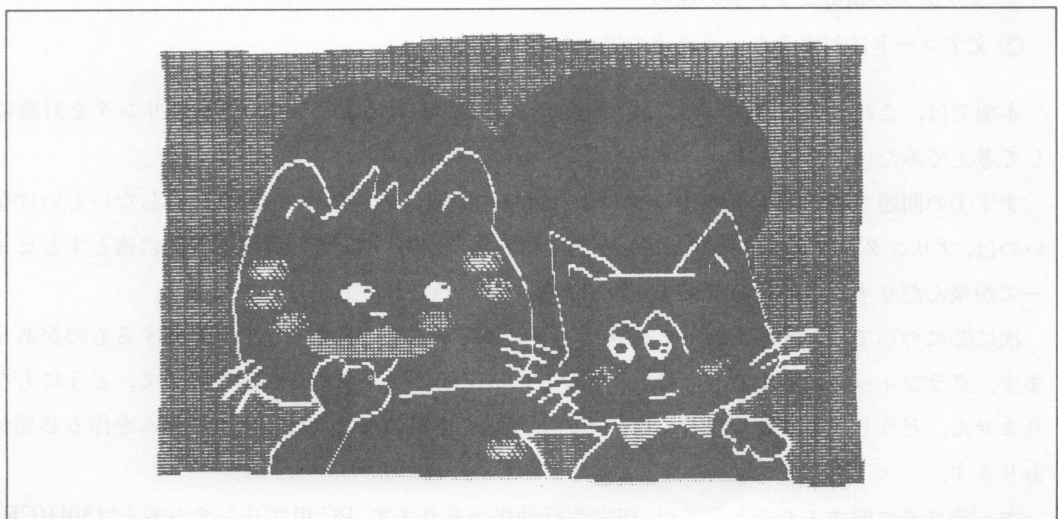


図9-6 拡張画面コピープログラムのサンプル

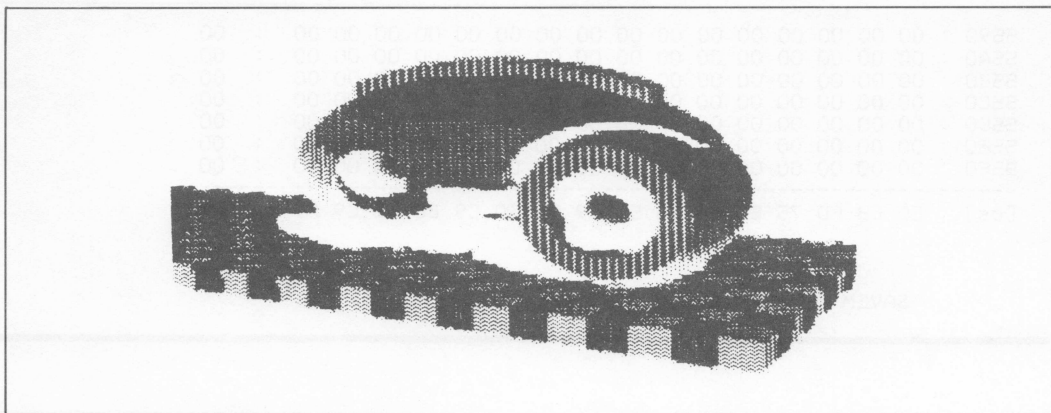


図9-7 拡張画面コピープログラムのサンプル

9-4 PC 用プリンタを FM に接続

現在、多くのメーカーからいろいろな種類のプリンタが発売されています。その中には、特殊なインターフェースを採用したプリンタも見受けられますが、大部分はセントロニクス規格に基づいたインターフェースが採用されています。FM シリーズのプリンタインターフェースもセントロニクス規格に基づいていますので、他機種用のプリンタを FM シリーズに接続することも可能なわけです。ただ、その場合に注意すべきことは、セントロニクス規格で保証されているのが最小限の動作だけであるということです。おもな問題点としては、次のようなことがあります。

- ① 信号の出力されるタイミングの違い
- ② プリンタの制御コマンドの違い
- ③ 文字コードに対するキャラクタの違い

本項では、これらの問題について最も種類が多いと思われる PC シリーズ用プリンタを対象にして考えてみたいと思います。

まず①の問題ですが、PC 用プリンタに限れば特に問題はありません。ただ注意しないといけないのは、プリンタのコネクタに電源ピンが出ている場合があって、うっかり GND に落とすとヒューズが飛んだりすることがあります。説明書等で確かめておく方が良いでしょう。

次に②については、大きくわけてグラフィックに関するものとキャラクタに関するものがあります。グラフィックに関するものは、エプソンの MP80 シリーズの PC 用を除いて、どうにもなりません。どうしても必要な場合には、そのプリンタ専用の画面コピープログラムを作る必要があります。

キャラクタに関するものとしては、印字改行動作があります。PC 用プリンタの多くは \$0D (CR) を印字改行指令コマンドとします。しかし FM シリーズ用プリンタでは、通常 \$0D (CR) を送出せ

ず、\$0A(LF)を印字改行指令コマンドとして送出しています。ですからPC用プリンタをFMシリーズに接続してそのまま印字すると、たいていのプリンタでは、改行動作が無効となってしまいます。そのためプログラムリストなどをとろうとすると、切れ目のないグラグラとつながったリストができあがってしまいます。

③については、\$F8以降のキャラクタ(〒市区町村人■)が印字されません。他は全く同一で問題はありません。

印字改行動作について、もう少し説明しましょう。PCシリーズ用では、データは図9-8のように送出されます。一方FMシリーズ用では、図9-9のようになります。両者の違いを比較してください。

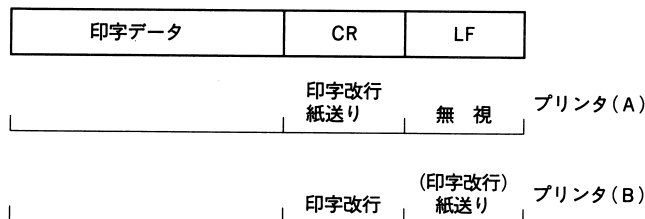


図9-8 PC用プリンタのデータ

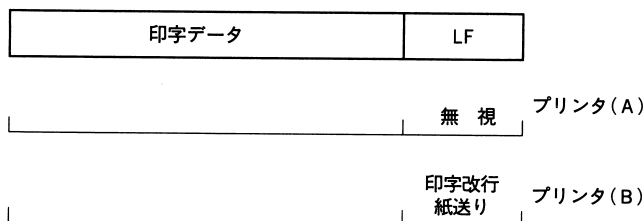


図9-9 FM用プリンタのデータ

結論としては、\$0D(CR)が送出されるように何らかの手を打ってやれば、よいことになります。FMシリーズでは、\$0D(CR)が出力禁止コード扱いになっています。ですから、CRコードの出力禁止を解除してやればよいことになります。

F-BASIC V3.0では、\$01E3番地に出力禁止コードが格納されています。PRINT HEX\$(PEEK(&H1E3)) ☐ として、\$0Dが出力禁止コードとなっていることを確認してみてください。そして次にPOKE &H1E3,0 ☐ を入力してください。これでCRコードの出力禁止が解除され、PC用プリンタにも正常なリストが出力できるようになります。

F-BASIC V3.0 L2.0またはF-BASIC V3.3のシステムディスク付属のユーティリティプログラム"SYSUTY"でも、CRコードの出力禁止解除を行なうことができます。"SYSUTY"を起動して、CR CODE FOR PRINTERの項目に、00を指定します。

FM-7 シリーズでは、音楽演奏および効果音の発生用として PSG が標準実装されています。またオプションの FM 音源カードを実装することにより様々な楽器音を合成することが可能となり、より自然な音楽演奏ができるようになっています。

一方、FM77AV では、従来の FM-7 シリーズでオプションだった FM 音源が標準実装となり、FM 音源が PSG の機能をすべて含んでいることから、FM-7 シリーズとの互換性を保ちつつ PSG は取り去られています。

10-1 PSG

10-1-1 PSG のハードウェア

PSG(Programmable Sound Generator)には、G・I 社の AY-3-8910 という LSI が使われています。この PSG のピン配置図を図 10-1 に、そしてハードウェア・ブロック図を図 10-2 に示

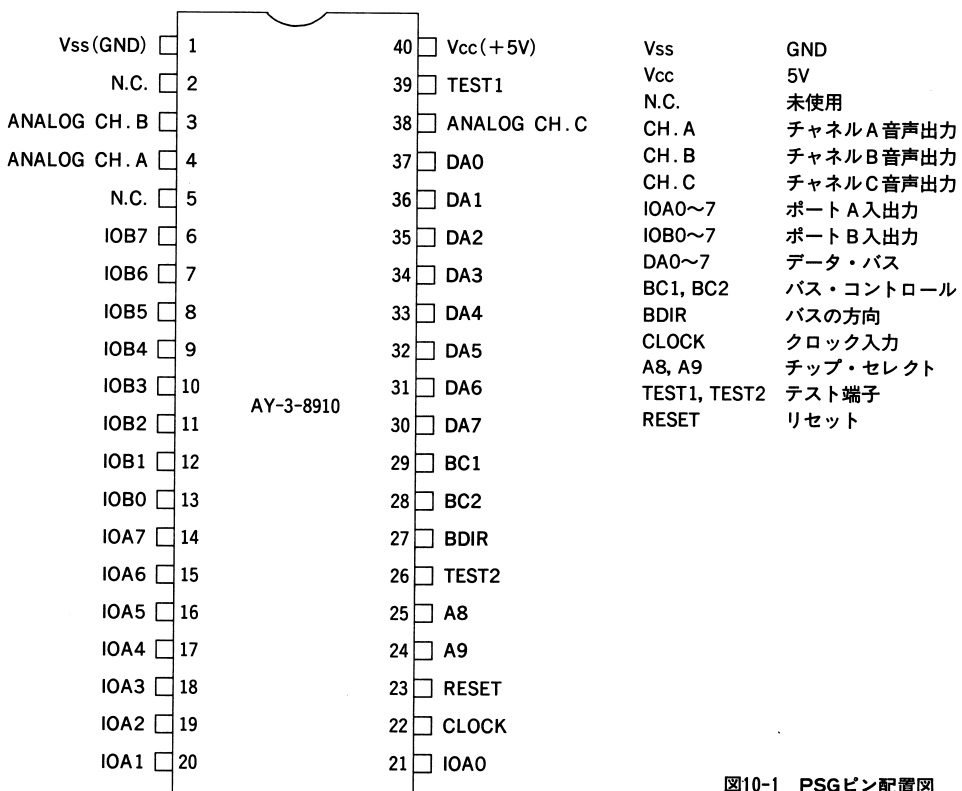


図10-1 PSGピン配置図

します。

PSG の内部は、3 個のオシレータとノイズ・ジェネレータ、ミキサー、エンベロープ・ジェネレータ(アンプ)から構成されていて、それぞれに 16 個のレジスタ(以下 R0~R15 と記す)が割り当てられています。R14, R15 には通常 I/O ポートが割り当てられますが、FM-7 シリーズでは使用されていません。

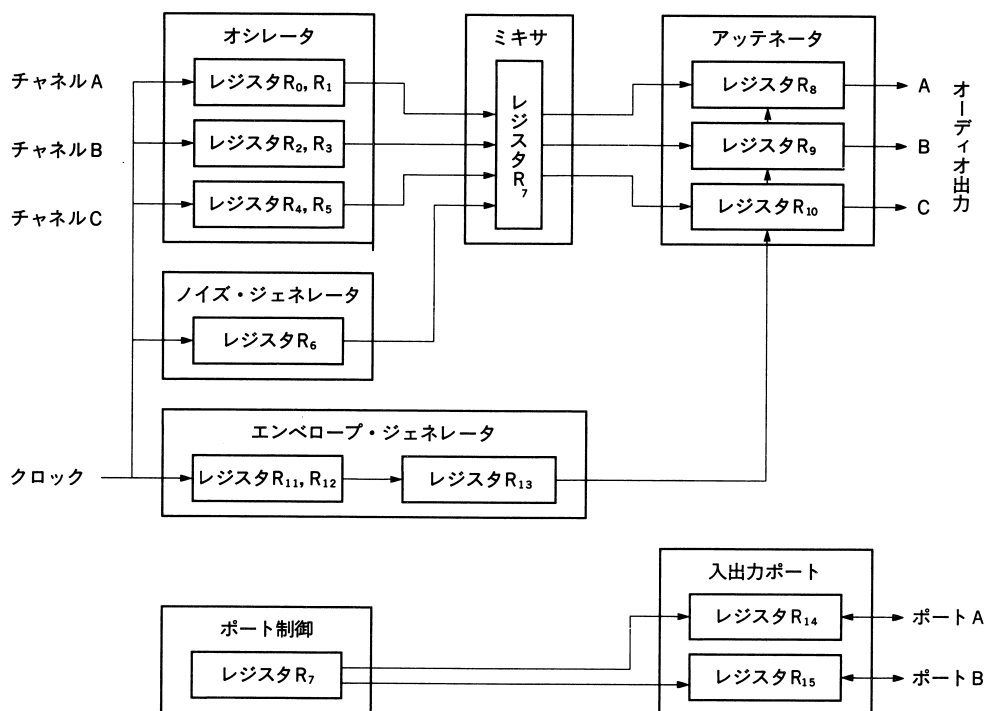


図10-2 PSGハードウェア・ブロック図

10-1-2 PSG の基礎

PSG の機能を利用していろいろな音を合成するわけですが、それにはまず PSG のレジスタの役割を知る必要があります。図 10-3 に PSG レジスタ・マップを示します。

R0 と R1, R2 と R3, R4 と R5 が、それぞれペアになって 12 ビットのオシレータを形成しています。そしてこれらのペア・レジスタに適当な値を書き込むことにより、いろいろな周波数の矩形波を得ることができます。得られる周波数(f)は、書き込むデータを(D)とすると

$$f = \frac{1.2288 \times 10^6}{16 \times D}$$

となります。たとえば楽器のチューニングで使う周波数 440Hz の音を出したいときには、ペア・

レジスタ	機 能	7	6	5	4	3	2	1	0	備 考
R ₀	チャンネルAの周波数	下位8ビット・データ								
R ₁						上位4ビット・データ				
R ₂	チャンネルBの周波数	下位8ビット・データ								
R ₃						上位4ビット・データ				
R ₄	チャンネルCの周波数	下位8ビット・データ								
R ₅						上位4ビット・データ				
R ₆	ノイズの平均周波数				5ビット・データ					
R ₇	ミキシング, ポート制御	入出力選択		ノ イ ズ			ト ー ン			対応するビットが1の時 オフ, 0の時オン
		ポート A	ポート B	C	B	A	C	B	A	
R ₈	チャンネルAの音量				M	4ビット・データ				M=0の時 下位4ビット・データによる 音量調節 M=1の時 エンベロープ作動
R ₉	チャンネルBの音量				M	4ビット・データ				
R ₁₀	チャンネルCの音量				M	4ビット・データ				
R ₁₁	エンベロープ周期	下位8ビット・データ								
R ₁₂		上位8ビット・データ								
R ₁₃	エンベロープ波形					CONT	ATT	ALT	HOLD	
R ₁₄	ポートA									R ₇ の入出力選択が1のとき 出力, 0のとき入力
R ₁₅	ポートB									

図10-3 PSGレジスタ・マップ

レジスタに175を書き込めばよいわけです。ペア・レジスタに書き込むデータと音階の関係を図10-4に示します。

レジスタR₆は、ノイズ・ジェネレータが発生するノイズの平均周波数を決定するレジスタで、下位5ビットで指定します。書き込む値が小さいほど平均周波数が高い(乾燥した感じの)ノイズ

オクターブ	1		2		3		4	
音 階	周波数	データ	周波数	データ	周波数	データ	周波数	データ
C	32.7	2349	65.4	1174	130.8	587	261.6	294
C#	34.6	2220	69.3	1108	138.6	554	277.2	277
D	36.7	2093	73.4	1046	146.8	523	293.7	261
D#	38.9	1974	77.8	987	155.6	494	311.1	247
E	41.2	1864	82.4	932	164.8	466	329.6	233
F	43.7	1757	87.3	880	174.6	440	349.2	220
F#	46.2	1662	92.5	830	185.0	415	370.0	208
G	49.0	1567	98.0	784	196.0	392	392.0	196
G#	51.9	1480	103.8	740	207.7	370	415.3	185
A	55.0	1396	110.0	698	220.0	349	440.0	175
A#	58.3	1317	116.5	659	233.1	329	466.2	165
B	61.7	1245	123.5	622	246.9	311	493.9	155

オクターブ	5		6		7		8	
音 階	周波数	データ	周波数	データ	周波数	データ	周波数	データ
C	523.2	147	1046.5	73	2093.0	37	4186.0	18
C #	554.4	139	1108.7	69	2217.4	35	4434.9	17
D	587.3	131	1174.7	65	2349.3	33	4698.6	16
D #	622.3	123	1244.5	62	2489.0	31	4978.0	15
E	659.3	116	1318.5	58	2637.0	29	5274.0	15
F	698.5	110	1396.9	55	2793.8	27	5587.6	14
F #	740.0	104	1480.0	52	2959.9	26	5919.9	13
G	784.0	98	1568.0	49	3135.9	24	6271.9	12
G #	830.6	92	1661.2	46	3322.4	23	6644.8	12
A	880.0	87	1760.0	44	3520.0	22	7040.0	11
A #	932.3	82	1864.6	41	3729.3	21	7458.6	10
B	987.8	78	1975.5	39	3951.0	19	7902.1	10

(注) 周波数はHz

図10-4 音階と周波数データ

が得られます。このノイズ・ジェネレータは、効果音を作る際に重要な音源です。

次にレジスタ R7 は、各チャンネル(A, B, C)に割り当てる音源の種類を決定するレジスタです。対応するビットに0を書き込んだときに出力がON となります。たとえば、チャンネル A にトーン、チャンネル B にノイズ、そしてチャンネル C を未使用とするときには、レジスタ R7 に&HAE と書き込みます。また、レジスタ R7 の上位2 ビットは特別な意味を持っており、FM-7 シリーズでは何でもいいのですが、FM77AV では FM 音源で PSG を代用している関係上、必ず "10" にする必要があります。

R8, R9, R10 は各チャンネルの音量を決定するレジスタで、ビット 0~3 の4 ビットで指定します。0 が音量最小で 15 が最大となります。またビット 4 を 1 にするとエンベロープ・ジェネレータを選択したことになり、そのときには音量の指定は無視されます。R11~R13 はエンベロープ・ジェネレータに関するレジスタで、R11, R12 でその周波数を、R13 でエンベロープ・パターンを指定します。エンベロープとは、比較的長い時間でみた音量変化のことで、PSG では(図 10-5)で示す 10 個のパターンを選択することができます。エンベロープの周波数(f)は、書き込むデータを(D)とすると、

$$f = \frac{1.2288 \times 10^6}{256 \times D}$$

となります。エンベロープ制御にて音が鳴り出す(エンベロープのトリガーがかかる)のは、レジスタ R13 に値を書き込んだ瞬間で、再トリガーをかける場合には再度レジスタ R13 に値を書き込まなければなりません。エンベロープを用いるとピアノのようにアタックの鋭い音やトレモロの

ような効果を出すこともできますが、エンベロープ・ジェネレータは1個しかありませんから、音楽などよりも爆発音などの効果音に応用するのがよいでしょう。

R ₁₃ の下位4ビット	16進値	エンベロープ・パターン
0 0 - -	0 ~ 3	
0 1 - -	4 ~ 7	
1 0 0 0	8	
1 0 0 1	9	
1 0 1 0	A	
1 0 1 1	B	
1 1 0 0	C	
1 1 0 1	D	
1 1 1 0	E	
1 1 1 1	F	

→ t ←

図10-5 エンベロープ・パターン

10-1-3 PSG の応用

電源スイッチを投入した状態では、R0~R15 のレジスタにはすべて 0 が書かれていますから、そのままでは何も音は出ません。各レジスタに必要なデータを書き込むことによって初めて音が出せるわけです。その手順は、まずどんな音を出すかによって R7 の割り当てを決定します。つづいてその音源に対応している各種レジスタ (R0~R5, R6) の設定をします。その後で音量の設定をすれば、とりあえず音は出ることになります。もしエンベロープ・ジェネレータを使用するときには、さらに R11, R12 の設定をしたあと、R13 にエンベロープ・パターンを書き込むことになります。

前置きが長くなって退屈された方もおられるかと思いますが、いよいよ音を出してみましょう。手始めに、楽器のチューニングに使用する 440Hz の音を出してみましょう。この例では 1 音しか

使いませんから、 $R7=\&HBE$ とします。 $R0, R1$ の値は、 $f=440\text{Hz}$ ですから $D=175$ となり、 $R0$ に 175, $R1$ に 0 を書き込めばよいことになります。音量は $R8$ に書き込むのですが、とりあえず最大(15)にしておきましょう。では実際に、これらの値を PSG に書き込んでみましょう。マシン語を使って PSG のレジスタに書き込む方法は後述するとして、とりあえず手軽な F-BASIC の SOUND 命令を使うことにします。

SOUND 7,&HBE □

SOUND 0,175 □

SOUND 1,0 □

SOUND 8,15 □

さあ、どうでしょうか？ ピーッという音が鳴りましたね。ここで $R8$ に他の値(0~15)を書き込めば音量が変化しますし、 $R0$ と $R1$ に他の値を書き込めば、周波数(音程)が変化します。いろいろな値を書き込んで試してみてください。音を止めたいときには、

SOUND 8,0 □

を実行します。

次にノイズ・ジェネレータを使って波の音を作ってみましょう。まず、チャンネル A にノイズを設定するため、 $R7=\&HB7$ とします。エンベロープ・ジェネレータも使用するので、 $R8=16$ としてレジスタ $R8$ のビット 4 を 1 にしておきます。エンベロープ周波数を 0.5Hz として $R11, R12$ の値を求めると $D=9600$ となります。そして、エンベロープ・パターンは $\&H0E$ を選択します。

SOUND 7,&HB7 □

SOUND 6,5 □

SOUND 8,16 □

SOUND 11,9600 MOD 256 □

SOUND 12,9600 ¥ 256 □

SOUND 13,&H0E □

いかがでしょうか？ また前の例のように $R6$ や $R11, R12$ の値をいろいろと変化させてみてください。

この2つの例では、1つの音を出すだけですが、2つ以上の音を合成したりレジスタに書き込む値を変数にしてループさせながらその変数の値を変化させる……等の処理をすれば、さらにいろいろな音が作れます。リスト 10-1 にそのサンプル・プログラムを示します。

いままでの説明で、ある程度“PSG に音を出させる方法”はわかっていたかと思います。サンプル・プログラムをいろいろと改良して、My Sound を創って楽しむのもいいでしょう。しかし、ゲームなどへの応用を考えるいろいろと欠点が目だってきます。たとえばひとつの音ごと

リスト 10-1 PSG サンプルプログラム

```

10 *****
20 '*   PSG SAMPLE PROGRAM   *
30 '*   ( LIST 10-1 )   V3.0/V3.3   *
40 *****
1000 '■ ハ"クハツオン
1010 PRINT"ハ"クハツオン"
1020 SOUND 7.&HB6:SOUND 8.16:SOUND 11.0:SOUND 12.30:SOUND 13.0:SOUND 0
    .0
1080 FOR I=0 TO 300
1090     SOUND 1,RND(1)*10+1:SOUND 6,RND(1)*31
1110 NEXT
1120 GOSUB 9000
1200 '■ キラキラ
1210 PRINT"キラキラ"
1230 SOUND 7.&HBC:SOUND 8.12:SOUND 9.12
1260 FOR I=0 TO 200
1270     SOUND 0,RND(1)*20+30:SOUND 1.0:SOUND 2,RND(1)*20+30:SOUND 3.0
1300     FOR J=0 TO 30:NEXT
1310 NEXT
1320 SOUND 8.0:SOUND 9.0
1340 GOSUB 9000
1400 '■ ラッカオン
1410 PRINT"ラッカオン"
1420 SOUND 7.&HBE:SOUND 1.0:SOUND 8.10
1450 FOR I=30 TO 200
1460     SOUND 0,I
1470     FOR J=0 TO 20:NEXT
1480 NEXT
1490 SOUND 7.&HB6:SOUND 8.16:SOUND 11.0:SOUND 12.10:SOUND 6.31:SOUND 0
    .15:SOUND 13.0
1560 GOSUB 9000
1600 '■ ヲリコフター
1610 PRINT"ヲリコフター"
1620 SOUND 7.&HA4:SOUND 0.0:SOUND 1.3:SOUND 6.10:SOUND 8.16:SOUND 11.0
    :SOUND 12.1:SOUND 2.13:SOUND 3.0:SOUND 9.16:SOUND 13.14
1730 FOR I=0 TO 10000:NEXT
1740 SOUND 8.0:SOUND 9.0
1760 GOSUB 9000
1800 '■ レーザーカ"ン
1810 PRINT"レーザーカ"ン"
1820 SOUND 7.&HBC:SOUND 8.12:SOUND 9.10
1850 FOR I=14 TO 3 STEP -1
1860     FOR J=20 TO 50 STEP 4
1870         SOUND 0,J:SOUND 1.0:SOUND 2,J+2:SOUND 3.0
1910     NEXT
1920     SOUND 8.I:SOUND 9,I-2
1940 NEXT
1950 SOUND 8.0:SOUND 9.0
1970 GOSUB 9000
2000 '■ UFO
2010 PRINT"UFO"
2020 SOUND 7.&HBC:SOUND 8.12:SOUND 9.4:SOUND 1.0:SOUND 3.0
2050 FOR I=0 TO 30
2060     FOR J=40 TO 120 STEP 4
2070         SOUND 0,J:SOUND 2,J/2
2080     NEXT
2090 NEXT
2100 SOUND 8.0:SOUND 9.0
2110 GOSUB 9000
2120 GOTO 1000
9000 '■ PUSH ANY KEY
9010 PRINT"PUSH ANY KEY"
9020 A$=INKEY$:A$=INPUT$(1)
9030 RETURN

```

にプログラムを書かなければいけないとか、複雑な音を作ろうとすると処理に時間がかかってしまう等、等。

次に紹介するプログラムはある程度ゲームなどに向けた方法で、あらかじめ計算しておいた PSG 用のデータを配列にためておき、必要に応じてそのデータを順次読みだして PSG に送り、音を出しています。これは、後で紹介する本格的なマシン語効果音サブルーチンと同じ考え方で作られているので、この考え方は是非理解してください。このプログラムでは、画面にサインカーブを書くルーチンとキー入力ルーチン、サウンドルーチンが、メインループから呼ばれています。そしてスペースキーが押されるとキー入力ルーチンがフラグを立て、そのフラグの変化をサウンドルーチンが検出して PSG 用データを PSG に送るという処理になっています。音楽が鳴っている間もサインカーブが書かれつづけていることに注目してください。なお、PSG 用データは、周波数と音量とで 1 組となっており、1 回のループで 1 組のデータを PSG に送り、データエンド(周波数と音量が共に 0 のデータ)を検出すると、フラグをクリアしてサウンドルーチンをバイパスするようになっています(リスト 10-2)。

リスト 10-2 PSG 効果音プログラム

```

10 '*****
20 '*      PSG EFFECT SOUND PROGRAM *
30 '*      ( LIST 10-2 ) V3.0/V3.3 *
32 '*      SPACE KEY ニテ SOUND ON *
40 '*****
1000 X=0:FLAG=0:C=7:CLS
1010 GOSUB 5000'■■ SOUND INIT
1020 GOSUB 2000'■■ SIN カーブ
1030 GOSUB 3000'■■ KEY INPUT
1040 GOSUB 4000'■■ SOUND
1050 GOTO 1020
2000 '■■ SIN カーブ
2010 Y=SIN(X/180*3.14159)*99+100
2020 PSET(X,Y,C)
2030 X=X+1
2040 IF X>=640 THEN C=C-1:X=0
2050 C=C AND 7
2060 RETURN
3000 '■■ KEY INPUT
3010 A$=INKEY$
3020 IF A$=" " THEN FLAG=1
3030 RETURN
4000 '■■ SOUND
4010 IF FLAG=0 THEN RETURN
4020 REG0=SND%(FLAG-1)
4030 REG8=SND%(FLAG)
4040 SOUND 0,REG0 MOD 256
4050 SOUND 1,REG0 ¥ 256
4060 SOUND 8,REG8
4070 FLAG=FLAG+2
4080 IF REG0=0 AND REG8=0 THEN FLAG=0
4090 RETURN
5000 '■■ SOUND INIT
5010 DIM SND%(31)
5020 FOR I=0 TO 31
5030 READ SND%(I)
5040 NEXT
5050 SOUND 7,&HBE

```

```

5060 SOUND 8.0
5070 RETURN
5080 DATA 20.14,30.13,40.12,55.12
5090 DATA 70.12,90.10,50.8,40.7
5100 DATA 20.12,30.11,40.10,55.9
5110 DATA 70.8,90.7,50.6,0.0

```

10-1-4 PSG のマシン語アクセス

F-BASIC には SOUND 命令という PSG をアクセスするための命令がありますが、マシン語で PSG に何かをさせようとする、SOUND 命令に相当するものを自分で作らないといけません。FM-7 シリーズの PSG は、\$FD0D、\$FD0E に割り付けられていますが、そのアクセスは少々複雑です。その手順は、

- ① データレジスタ(\$FD0E)にアクセスしたいレジスタ番号を書く。
- ② コマンドレジスタ(\$FD0D)にラッチ・アドレスコマンド(\$03)を書く。
- ③ コマンドレジスタにインアクティブコマンド(\$00)を書く。
- ④ データレジスタに書き込みたいデータを書く。
- ⑤ コマンドレジスタにライト・データコマンド(\$02)を書く。
- ⑥ コマンドレジスタにインアクティブコマンド(\$00)を書く。

これを実際にプログラムしたのが、リスト 10-3 です。PSG のレジスタ番号をアキュムレータ A に、データをアキュムレータ B にセットしてこのルーチンと呼ばば、PSG のレジスタに書き込みが行われます。プログラムの先頭は\$5000 になっていますが、このままでこのアドレスに移しても使えるようになっています。

リスト 10-3 PSG レジスタ書き込みルーチン

```

01000 *****
01020 *           PSG WRITE ROUTINE           *
01040 *           ( LIST 10-3 )   V3.0/V3.3   *
01060 *****
01070             OPT      NOGEN
01080 5000         ORG      $5000
01100             F00D    PSGCOM EQU  $FD0D    PSG コメント"レジスター
01110             F00E    PSGDAT EQU  $FD0E    PSG テーマ"レジスター
01120             *
01130 5000 B7     F00E    WRTPSG STA  PSGDAT レジスター バンコウ カキコミ
01140 5003 34     02      PSHS  A          AccA ホソソ
01150 5005 86     03      LDA   #3        ラッチ アドレス コメント"
01160 5007 B7     F00D    STA  PSGCOM  (<$03> カキコミ
01170 500A 7F     F00D    CLR  PSGCOM  インアクティブ"コメント" カキコミ
01180 500D F7     F00E    STB  PSGDAT  テーマ" カキコミ
01190 5010 4A     DECA   DECA   ライト"テーマ"コメント"
01200 5011 B7     F00D    STA  PSGCOM  (<$02> カキコミ
01210 5014 7F     F00D    CLR  PSGCOM  インアクティブ"コメント" カキコミ
01220 5017 35     B2      PULS  A,PC     AccA フック / リターン
01240             5000    END    WRTPSG

```

```
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5018
PROGRAM ENTRY ADDR=5000
```

では試しに一番最初の例でも出て来ました 440Hz の音を出すプログラムを作ってみましょう。リスト 10-4 がその例です。ここでは、\$5000 から PSG 書き込みルーチンがあるものとして作っていますから、リスト 10-3 も \$5000 から入力しておく必要があります。プログラムの実行は、BASIC から EXEC &H5100 [] です。音を止めたいときには、EXEC &H5103 [] と入力してください。

リスト 10-4 PSG マシン語サンプルプログラム

```
01000
01020
01030
01040
01050
01060 5100
01080
01100 5100 7E 5000
01110 5103 7E 5106
01120
01130
01140
01150 5106 86 07
01160 5108 C6 BE
01170 510A B0 5000
01180 510D 86 00
01190 510F C6 AF
01200 5111 B0 5000
01210 5114 86 01
01220 5116 C6 00
01230 5118 B0 5000
01240 511B 86 08
01250 511D C6 0F
01260 511F B0 5000
01270 5122 39
01280
01290
01300
01310 5123 86 08
01320 5125 C6 00
01330 5127 B0 5000
01340 512A 39
01360
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5100
PROGRAM END   ADDR=512A
PROGRAM ENTRY ADDR=5100
```

```
*****
*      440Hz チューニング" サウンド"      *
*      ( LIST 10-4 )      V3.0/V3.3      *
*****
          OPT      NOGEN
          ORG      $5100
WRTPSG   EQU      $5000
ENTRY    JMP      S_ON
          JMP      S_OFF
*
*      オトラ アス
*
S_ON     LDA      #7          SOUND 7.&HBE
          LDB      #$BE
          JSR      WRTPSG
          LDA      #0          SOUND 0.175
          LDB      #175
          JSR      WRTPSG
          LDA      #1          SOUND 1.0
          LDB      #0
          JSR      WRTPSG
          LDA      #8          SOUND 8.15
          LDB      #15
          JSR      WRTPSG
          RTS
*
*      オトラ トメル
*
S_OFF    LDA      #8          SOUND 8.0
          LDB      #0
          JSR      WRTPSG
          RTS
          END      ENTRY
```

PSG のレジスタは、値を書き込むだけではなく読み出しも可能です。その手順は、

- ①～③ 書き込みの手順と同じ。
- ④ コマンドレジスタにリード・データコマンド(\$01)を書く。
- ⑤ データレジスタからデータを読み取る。
- ⑥ コマンドレジスタにインアクティブコマンド(\$00)を書く。

通常の処理では、PSG のレジスタの値を調べることはないと思いますが、覚えておいて損はないと思います。リスト 10-5 に PSG 読み込みルーチンを示します。PSG のレジスタ番号をアキュムレータ A にセットして呼べば、アキュムレータ B に PSG レジスタの値がセットされます。

リスト 10-5 PSG レジスタ読み込みルーチン

```

01000                                *****
01020                                *           PSG READ ROUTINE           *
01040                                *      ( LIST 10-5 )   V3.0/V3.3      *
01060                                *****
01070                                OPT      NOGEN
01080      5000                      ORG      $5000
01100                                PSGCOM EQU    $FD00      PSG コメント"レジスタ"
01110                                PSGDAT EQU    $FD0E      PSG テーマ"レジスタ"
01120                                *
01130      5000 B7      FD0E      RDPSG STA    PSGDAT レジスタ"ハット"カキコミ
01140      5003 C6      03              LOB      #3      ラッチ アドレス コメント"
01150      5005 F7      FD0D              STB      PSGCOM ($03) カキコミ
01160      5008 7F      FD0D              CLR      PSGCOM インアクティブ"コメント" カキコミ
01170      500B C6      01              LOB      #1      リート"テーマ"コメント"
01180      500D F7      FD0D              STB      PSGCOM ($01) カキコミ
01190      5010 F6      FD0E              LOB      PSGDAT テーマ"ヨミトリ"
01200      5013 7F      FD0D              CLR      PSGCOM インアクティブ"コメント" カキコミ
01210      5016 39                      RTS      リターン
01230                                END      RDPSG
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5016
PROGRAM ENTRY ADDR=5000

```

10-1-5 PSG 効果音ルーチン

最後にゲームなどの効果音出力に最適な PSG 効果音ルーチンを紹介します。リスト 10-6 とリスト 10-7 がそれぞれですが、BASIC で書かれているリスト 10-6 の方は、リスト 10-7 のプログラムをテストするためのもので、あまり意味がありません。

さて、この PSG 効果音ルーチンの使い方ですが、まず、リスト 10-7 を入力して“L10-7M”というファイル名で SAVEM してください。

SAVEM “L10-7M”, &H5000, &H5438, &H5000

そしてリスト 10-6 を入力して RUN  と入力します。“HIT A-R OR @”と表示されますか

ら、A から R までのキーを押してみてください。いろいろな効果音がでてきましたね、もうこれだけでゲームをしているような気分になってきませんか？ なお、@ キーを押すと PSG 効果音ルーチンの処理が終了します。

リスト 10-6 PSG 効果音ルーチンの実行

```

10 '*****
11 '* PSG EFFECT SOUND RTN
12 '* ( LIST 10-6 ) V3.0/V3.3
13 '* LIST 10-7 か" ビツウ チ"ス
14 '*****
20 CLEAR 300.&H4FFF
30 LOADM "L10-7M"
40 PRINT" HIT A-R or @ "
50 EXEC &H5000
60 A$=INKEY$:A$=INPUT$(1)
70 I=ASC(A$)-65
80 IF I=-1 THEN 200
90 IF I<0 OR I>17 THEN BEEP:GOTO 60
100 POKE &H5010+I,0
110 GOTO 60
200 EXEC &H5003
210 END

```

リスト 10-7 PSG 効果音ルーチン

ADR	:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
5000	:	7E	51	2C	7E	51	75	00	00	00	00	00	00	00	00	00	00	:	3F
5010	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5020	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5030	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	08	FE	:	01
5040	:	F7	10	FD	02	EF	20	FB	04	DF	B6	FD	03	84	04	26	02	:	59
5050	:	80	01	3B	10	8E	50	25	CE	50	10	C6	06	80	13	31	28	:	CF
5060	:	CE	50	16	C6	06	80	0A	31	28	CE	50	1C	C6	06	80	01	:	84
5070	:	39	8E	00	00	A6	C4	81	FF	26	08	30	02	33	41	5A	26	:	05
5080	:	F3	39	34	40	33	41	5A	27	06	86	FF	A7	C4	20	F5	35	:	D5
5090	:	40	A6	C4	27	S2	A6	22	26	3F	A6	23	A7	22	AE	A4	EC	:	20
50A0	:	84	27	38	A6	24	26	21	A6	25	48	4C	E6	80	C1	0F	25	:	AE
50B0	:	03	5F	30	1F	8D	50	4A	E6	80	80	58	A6	25	8B	08	E6	:	74
50C0	:	80	80	50	AF	A4	6A	22	39	86	06	E6	80	80	45	A6	25	:	04
50D0	:	88	08	E6	80	8D	3D	AF	A4	6A	22	39	A6	25	8B	08	5F	:	98
50E0	:	8D	31	86	FF	A7	C4	39	6C	C4	EC	26	30	8B	AE	84	EC	:	02
50F0	:	81	A7	22	A7	23	E7	24	AF	A4	8E	50	30	E6	25	58	EB	:	08

[cs]	:	DC	12	8B	S7	AB	F2	C0	D3	BF	3F	9E	94	8B	23	76	D9	:	87

ADR	:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
5100	:	24	58	3A	F6	50	24	EA	84	E4	01	F7	50	24	86	07	80	:	F8
5110	:	02	20	8A	B7	FD	0E	34	02	86	03	B7	FD	00	7F	FD	00	:	77
5120	:	F7	FD	0E	4A	B7	FD	00	7F	FD	00	35	82	1A	10	CE	50	:	95
5130	:	10	C6	12	86	FF	A7	C0	5A	26	F8	C0	07	BF	F7	50	24	:	4C
5140	:	8D	D1	8E	50	25	4F	CE	51	86	A7	05	EF	06	30	08	4C	:	7A
5150	:	CE	51	92	A7	05	EF	06	30	08	4C	CE	51	9E	A7	05	EF	:	2E
5160	:	06	FC	FF	F8	FD	50	22	CC	50	49	FD	FF	F8	86	04	B7	:	02
5170	:	FD	02	1C	EF	39	34	01	1A	10	CC	07	BF	8D	95	FC	50	:	A2
5180	:	22	FD	FF	F8	35	81	51	AA	51	E2	52	22	51	AA	51	E2	:	9C
5190	:	52	22	52	4A	52	70	52	B2	52	EE	52	4A	52	70	53	30	:	F7
51A0	:	53	64	53	44	53	30	53	64	53	A4	06	00	19	0D	23	0C	:	3A
51B0	:	32	0B	46	0A	50	0A	50	0A	64	0A	69	0A	6E	0A	19	0A	:	C7
51C0	:	23	0A	32	0A	46	0A	50	0A	5A	0A	64	0A	69	0A	6E	0A	:	D0

```

S1D0 : 19 0A 23 0A 32 0A 46 0A 50 0A 5A 0A 64 0A 69 0B : 79
S1E0 : 00 00 04 00 14 0E 1E 0D 28 0C 37 0C 46 0C 5A 0A : 7E
S1F0 : 32 08 28 07 14 0D 1E 0C 28 0B 37 0B 46 0B 5A 0A : DE
-----
[cs] : F2 05 8A 66 2D F2 04 BD CF BD C5 75 B6 5A 9A 9E : 05

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S200 : 32 08 28 07 14 0C 1E 0B 28 0A 37 0A 46 0A 5A 09 : 0B
S210 : 32 08 28 07 14 0B 1E 0A 28 09 37 0B 46 06 5A 04 : CA
S220 : 00 00 04 00 50 0D 46 0C 3C 0A 37 0A 46 0A 5A 0B : EC
S230 : 64 06 64 00 64 00 64 00 64 00 64 00 32 0D 3C 0C : E5
S240 : 28 0A 37 0B 46 06 5A 04 00 00 04 00 19 0A 19 0C : 67
S250 : 19 0C 19 0B 19 0B 19 0A 19 0A 19 0A 19 09 19 0B : 19
S260 : 19 07 19 06 19 05 19 04 19 03 19 02 19 01 00 00 : CB
S270 : 10 01 00 05 03 06 05 07 07 08 09 08 0B 08 0D 0B : 73
S280 : 0F 0B 11 09 13 09 14 09 15 09 16 08 17 0B 18 0B : E5
S290 : 19 08 1A 07 1B 07 1C 07 1D 07 1E 06 1F 06 1E 06 : 18
S2A0 : 19 05 14 05 19 05 14 04 19 04 14 03 19 02 14 01 : D1
S2B0 : 00 00 0F 00 FA 0A FA 00 E6 0A E6 00 D2 0A D2 00 : 91
S2C0 : BE 0A BE 00 AA 0A AA 00 96 0A 96 00 82 0A 82 00 : 2B
S2D0 : 6E 0A 6E 00 5A 0A 5A 00 46 0A 46 00 37 0A 37 00 : B2
S2E0 : 2A 0A 2A 00 22 09 22 00 1A 07 1A 00 00 00 0A 00 : F0
S2F0 : 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A : CC
-----
[cs] : F1 71 FC 4B E6 86 12 5B 78 75 A3 4B 5C 7B 9F 56 : 26

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S300 : 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A : CC
S310 : 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A 28 0A 37 0A : CC
S320 : 28 0A 3C 0A 28 0A 3C 0A 28 0A 3C 0A 28 0A 00 00 : 9A
S330 : 04 00 14 0D 14 0C 19 0B 1E 0A 1E 0A 19 0A 14 0C : FC
S340 : 1E 0C 1E 0A 1E 0A 19 09 14 0B 19 0B 14 0B 19 0B : 16
S350 : 19 0B 1E 0B 1E 0B 19 0B 14 06 19 06 14 04 19 04 : FC
S360 : 19 02 00 00 10 01 00 0F 05 0E 0A 0E 0F 0C 1F 0A : AA
S370 : 05 0E 0A 0D 0F 0C 1F 0A 00 0E 05 0D 0A 0C 14 0A : C2
S380 : 0A 0D 0F 0C 14 0B 1F 0A 05 0B 0A 0A 0F 09 14 07 : D1
S390 : 0F 09 0A 0A 05 09 00 0B 05 07 0A 06 14 05 1F 04 : 9A
S3A0 : 00 02 00 00 0B 00 05 00 0F 02 00 0C 03 00 0D 05 : 41
S3B0 : 00 0A 01 00 0E 04 00 0C 02 00 0D 06 00 09 03 00 : 4A
S3C0 : 0C FA 0A 02 00 0D 03 00 06 05 00 0A 03 00 0B 04 : 46
S3D0 : 00 07 06 00 0A 02 00 05 04 00 0B 0B 00 0C 04 00 : 42
S3E0 : 07 02 00 0A 03 00 09 FA 06 05 00 0A 02 00 04 03 : 37
S3F0 : 00 0B 01 00 09 05 00 07 03 00 05 07 00 09 06 00 : 3C
-----
[cs] : FD 6F 2F 6C 2C 75 44 77 F1 70 37 96 FD 78 40 57 : 9D

  ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
S400 : 04 02 00 06 04 00 05 03 00 09 05 00 05 02 00 04 : 31
S410 : 01 00 05 03 00 02 07 00 05 06 00 02 02 00 05 04 : 2A
S420 : 00 02 05 00 04 03 00 03 01 00 02 02 00 03 05 00 : 1E
S430 : 04 02 00 02 01 00 01 00 00 00 00 00 00 00 00 00 : 0A
S440 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S450 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S460 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S470 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S480 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S490 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
S4F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 09 06 0A 0B 09 05 0D 06 06 0F 07 04 07 05 0A 0B : 83

```

SAVEM "L10-7M",&H5000,&H5438,&H5000

さて、このプログラムを使用する上での注意点ですが、このプログラムはタイマー割り込みを独自にコントロールして使っているので、他の割り込み処理とは共用できません。たとえばPLAY文などを実行しても、このPSG効果音ルーチンが動いているときには無視されます。また、BREAKキーを押したりエラーを起こしたりすると、割り込み処理がストップしてしまいPSG効果音ルーチンは動作しなくなります。そのときには、POKE &HFD02,4 []と入力してください。

このプログラムには、10種類の音データが入れてあります。自分の音データを入れたい方のために、フラグと音データのフォーマットを説明して終わりたいと思います。

まず、処理がチャンネルごとになっていますので、3チャンネル分別々にフラグと音データがあります。\$5010番地からフラグで、各チャンネル6個ずつ、計18個設けてあります。このフラグを0にすると、そのフラグに対応する音が出力されます。そしてこのフラグは、それぞれのチャンネルで上にあるものから優先順位が高くなっています。ですから、たとえば\$5010番地に0をセットして音が出ているとき、\$5011~\$5015に0をセットしても無視されます。

\$5186番地からは音データテーブルで、これも各チャンネル6個ずつ、計18個あって、それぞれの音データの先頭番地を書き込みます。

音データは最初の1バイトが分周比で、2mSのタイマー割り込みの何回に1回、音データを処理するかを決定します。つまり、テンポが設定されるわけです。次の1バイトはノイズかトーンかの選択フラグで、0でトーン、1でノイズになります。その次からが、2バイト1組でPSGにセットされるデータです。ノイズのときには、初めが平均周波数(R6)、次が音量(チャンネルAのときR8)です。トーンのときには、初めが周波数(R0)、次が音量(R8)になります。ただし、トー

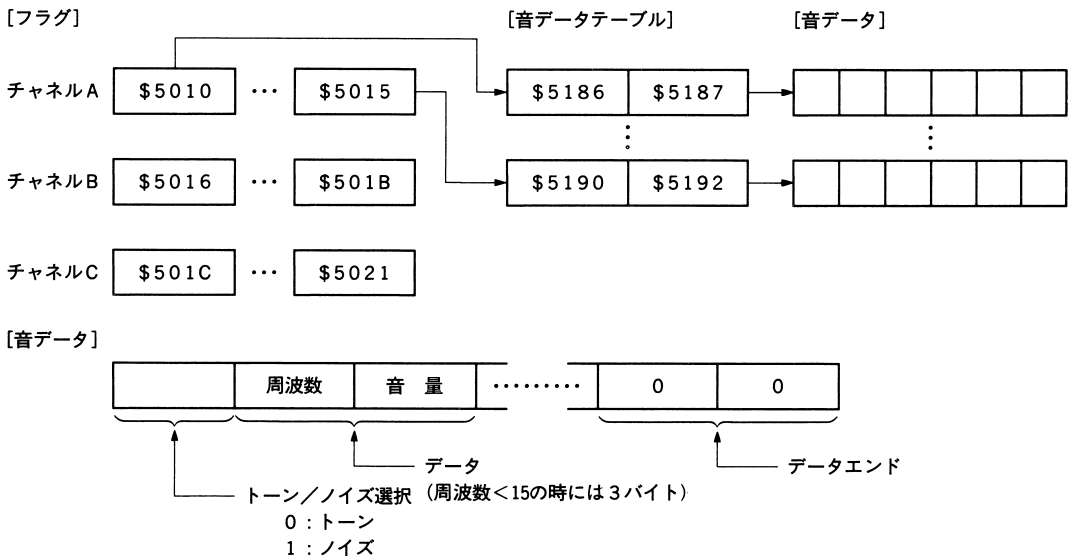


図10-6 効果音ルーチンのパラメータ

ンのときには例外処理があります。初めのデータが15未満のときには、それがR1(チャンネルAのとき)に書かれ、その次のデータがR0,R8に書かれ、3バイトで1組になります。またいずれの場合でも、0,0がデータエンドとなります(図10-6)。

10-1-6 PSG 音楽ルーチン

マシン語でゲームなどを作成していると、ちょっとした音楽を鳴らしたくなることがあると思います。しかし、少しばかりの音楽のために、わざわざ専用の音楽ルーチンを作るのも面倒なものです。そこで、BASIC ROMの音楽ルーチンを利用する方法を考えてみました。

F-BASIC V3.0では、PLAY命令の中間言語(\$F3)を見つけると、データの先頭を汎用読み込みポインタ(\$00D9,DA)にセットして、PLAY文の処理エントリ(\$EC72)にジャンプします。データは中間言語で書かれていますが、ダブルクォーテーション(")内はすべてASCIIコードとなっています。

したがって通常のPLAY文と同じMMLを用意して、その先頭アドレスを(\$00D9,DA)にセットして、\$EC72番地をコールすれば音楽が演奏できることになります。リスト10-8が、サンプルプログラムです。実行は、EXEC &H5000としてください。

最後に注意点ですが、MMLは必ずダブルクォーテーション(")で始まって、\$00で終わるようにしてください。また、長いMMLを書くと、「Out of String Space」のエラーが出るることがあ

リスト10-8 PSG 音楽ルーチン

```

01000                                *****
01020                                *   PSG MUSIC RTN (ROM ROUTINE)   *
01040                                * ( LIST 10-8 ) V3.0             *
01060                                *****
01070                                OPT   NOGEN
01080      5000                      ORG   $5000
01100                                PLAY EQU   $EC72   PLAY ショリ エントリ
01110                                POINTR EQU $00D9   ハンコク ヨミコミ ホイインター
01150      5000 8E 5008              ENTRY LDX #PLAYDT ヨミコミホイインターニ テーターノ
01160      5003 9F 09                STX   <POINTR セントウラ セット
01170      5005 7E EC72              JMP   PLAY   PLAY ショリハ
01210      5008 22                  PLAYDT FCC  '""T20005L8'
01220      5011 45                  FCC  'E4.DC4D4E.R16E.R16E4.R8'
01230      5028 44                  FCC  'D.R16D.R16D4.R8E4G.R16G4.R8'
01240      5043 45                  FCC  'E4.DC4D4E.R16E.R16E4.R8'
01250      505A 44                  FCC  'D.R16D4E4.DC1',
01260      5069 22                  FCC  '""T20004L4'
01270      5072 47                  FCC  'GEGEGEGE'
01280      507A 46                  FCC  'FDFDECEC'
01290      5082 47                  FCC  'GEGEGEGE'
01300      508A 46                  FCC  'FDFDECEC'
01310      5093 00                  FCB   0,0,0
01330                                END   ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5095
PROGRAM ENTRY ADDR=5000

```

ります。そのときには、MML を分割するか、BASIC の文字領域を広げてください。他に BASIC ROM が有効になっていなかったり、IRQ が禁止されていると動作しませんから、オールマシン語で利用する場合などは要注意です。

10-2 FM 音源

FM-7 シリーズでオプションとして発売され、大好評だった FM 音源カードは、FM77AV で標準実装となり、いっそう私達の身近なものとなりました。FM 音源の音色は PSG のそれとはあまりに違いすぎていて、一度 FM 音源で鳴らした音楽を聞くと、PSG の音楽は聞く気がなくなるほどです。

そんなにいいことずくめの FM 音源ですが、唯一の欠点があります。それは、PSG に比べて使いこなすのが桁違いに難しいことです。なにせ FM 音源の内部には、100 個以上ものレジスタがあって、それらのひとつひとつが互いに影響を及ぼしあって、あの素晴らしい音を作り出しているのですから。

本章では、できるだけ平易に説明をしたつもりですが、筆足らずでわかりにくい点も多いかと思います。しかしパソコンの世界では、覚えるより慣れろです。是非とも本章を足がかりにして、この素晴らしい FM 音源の世界に一步ずつ足を踏み出してみてください。

10-2-1 FM 音源の基礎

(1) PSG との違い

まず、PSG には音色という考え方そのものが存在しませんでした。出力できるのは、単純な矩形波とノイズのみです。一方、FM 音源では波形を合成することにより、いろいろな音色を持たせることが可能になっています。合成といっても単なる加算だけではなく、ひとつの音源の出力で他の音源に周波数変調(FM)をかけることもできるのです。FM 音源と呼ばれるのも、その点にゆえんがあるわけです。

(2) FM 音源の原理

まず、テープレコーダーを考えてみましょう。“ピーツ”という正弦波を録音したテープがあるとします。これを一定のスピードで普通に再生すれば、元通りの正弦波が出力されます。再生スピードを速くすれば出力される正弦波の周波数も高くなりますし、遅くすれば低くなります。これが音程に相当するわけです。では、再生するスピードが一定でない場合はどうでしょうか？ 図 10-7 を見てください。実際にはこんなに速く再生スピードを変化させることは不可能ですが、まあこんなもんだと思って見ておいてください。このように、正弦波を読み出すタイミングを変化させて他の波形に加工してしまうことが、FM 音源の基本というわけです。

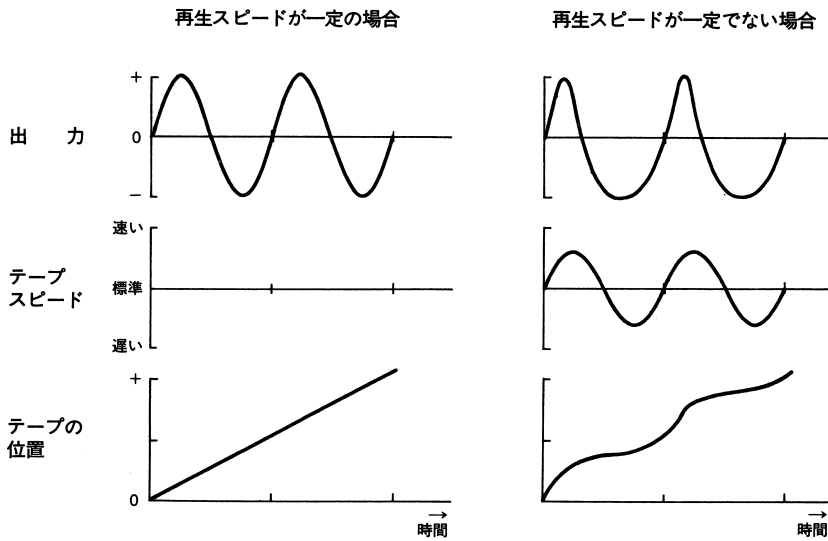


図10-7 FM音源の原理

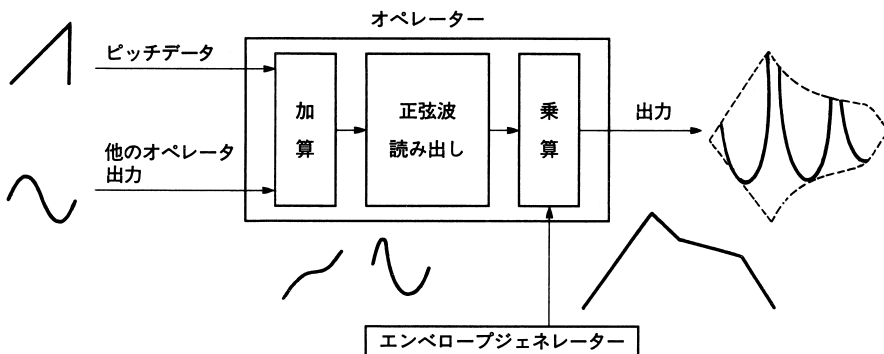


図10-8 オペレータのブロック図

図10-8を見てください。これは、先ほどの説明のテープレコーダー1台分に相当するもので、オペレータと呼ばれています。この中で「正弦波読み出し」と書いてあるのが、テープに相当する部分で、正弦波をA/D交換したデータが入っていると考えてください。また、「ピッチデータ」は、テープの位置に相当するもので、正弦波一周周期毎に対応したノコギリ波が入力されています。そしてこのノコギリ波に他のオペレータからの出力を加算して正弦波を読み出せば、複雑な波形が合成できるというわけです。

さらにオペレータ1個ごとにエンベロープ・ジェネレータが付きますから、PSGのような不自由さはありません。音量、音色の変化も自由自在です。FM-7シリーズに採用されているFM音源は、OPN(オペレータN型)といってこのようなオペレータが1音につき4個、全部で12個あ

て8通りの接続方法を選択できます(図10-9)。これをアルゴリズムといいます。また、特にオペレータのことをスロット、アルゴリズムのことをコネクションといういい方をします。

図10-9の点線の右側、つまり出力につながっているオペレータをキャリアといいます。左側のオペレータは、他のオペレータに変調をかける目的で使われることからモジュレータと呼ばれます。

一般的に、キャリアになっているオペレータの出力レベルを変えると音量の変化となり、モジュレータになっているオペレータの出力レベルを変えると音色の変化となります。

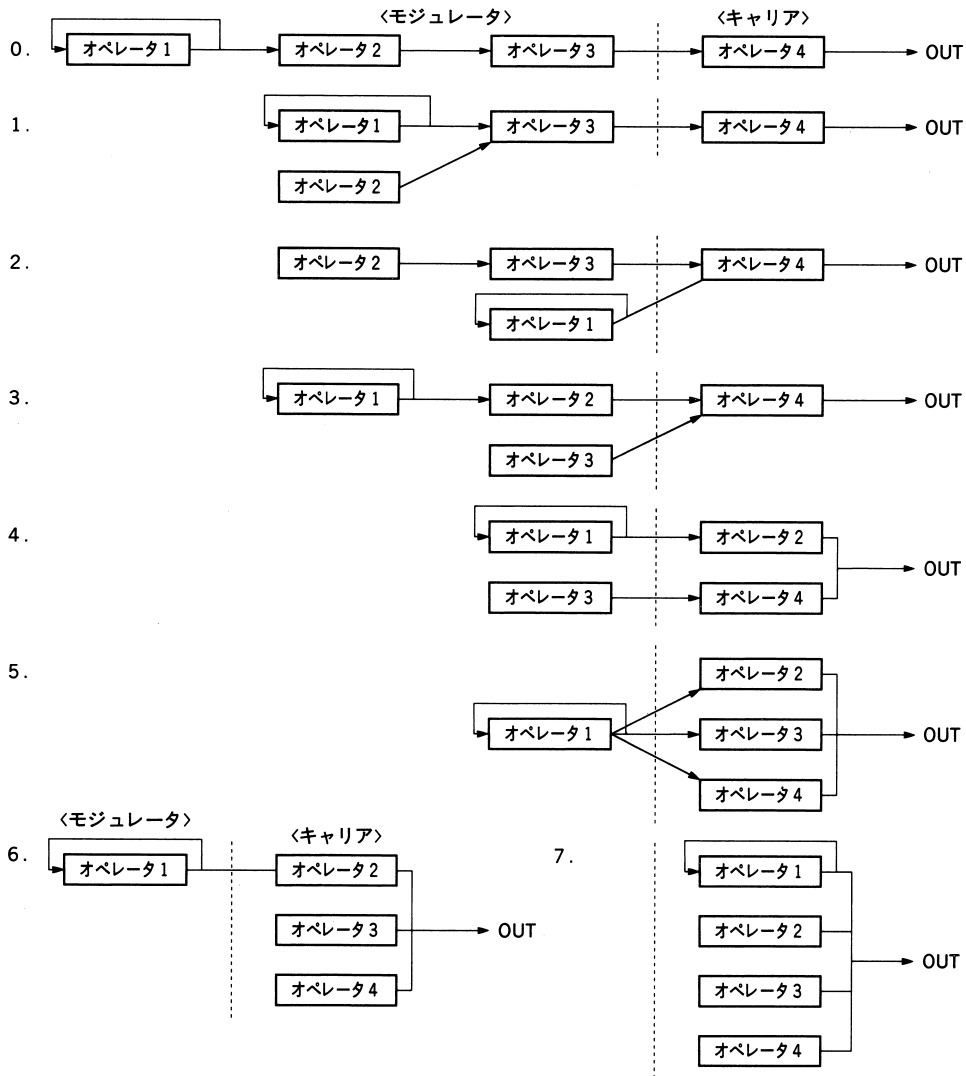


図10-9 コネクション

(3) FM音源のレジスタ

FM音源の内部には100個以上のレジスタがあり、それらに値を書き込むことによって音楽演奏などを行なうわけです。それらのレジスタひとつひとつの意味を理解するのは、大変なことです。そこで、重要な役割をもつものから順に説明していきます。

アドレス

00	Fine Tune A				チャンネルAの 周波数		
01			Coase A				
02	Fine Tune B				チャンネルBの 周波数		
03			Coase B				
04	Fine Tune C				チャンネルCの 周波数		
05			Coase C				
06			Period Control		ノイズ周波数		
07	IN/OUT	NOISE		TONE	ミキサー・ I/Oコントロール		
08			M	LEVEL A	チャンネルAの 音量		
09			M	LEVEL B	チャンネルBの 音量		
0A			M	LEVEL C	チャンネルCの 音量		
0B	Fine Tune				エンベロープ 周波数		
0C	Coase Tune						
0D			C	A T	A T	H D	エンベロープの形 状をコントロール
0E	I/OポートA						
0F	I/OポートB						

スロ ット	アドレス			ビット												
	7	6	5	4	3	2	1	0								
1	30	31	32	DT	MULTI				ディチューン/ マルチプル							
3	34	35	36													
2	38	39	3A													
4	3C	3D	3E													
1	40	41	42	TL					トータルレベル							
3	44	45	46													
2	48	49	4A													
4	4C	4D	4E													
1	50	51	52	KS	AR				キースケール/ アタックレイト							
3	54	55	56													
2	58	59	5A													
4	5C	5D	5E													
1	60	61	62	DR				ディケイレイト								
3	64	65	66													
2	68	69	6A													
4	6C	6D	6E													
1	70	71	72	SR				サスティーンエイト								
3	74	75	76													
2	78	79	7A													
4	7C	7D	7E													
1	80	81	82	SL				RR				サスティーンレベル/ リリースレイト				
3	84	85	86													
2	88	89	8A													
4	8C	8D	8E													
1	90	91	92	SSG-EG				SSGタイプ エンベロープ コントロール								
3	94	95	96													
2	98	99	9A													
4	9C	9D	9E													
	A0	A1	A2	F-Num1				Fナンバ/ブロック								
	A4	A5	A6	BLOCK		F-Num2										
	A8	A9	AA	3CH-F-Num1								3CH Fナンバ/ブ ロック(効果音モード)				
	AC	AD	AE	3CH BLOCK		3CH F-Num2										
	B0	B1	B2	FB		CONNECT				フィードバック/ コネクション						
	チャ ン ネ ル 1	チャ ン ネ ル 2	チャ ン ネ ル 3													

アドレス

21	TEST					
22						
23					ビット 9~2 タイマA 10ビット カウンタ	
24	TIMER-A					
25				TIMER A	ビット 0~1 タイマB 8ビットカウンタ	
26	TIMER-B					
27	MODE	RESET BA	ENABLE BA	LOAD BA	キーのオン/オフ	
28	SLOT			CH		
29						プリスケアラ 機能
2A						
2B						
2C						
2D						
2E						
2F						

図10-10 FM音源のレジスタマップ

まず図 10-10 を見てください。このうちで\$00～\$0F までは、PSG と同機能のものです。\$0E、\$0F は、ジョイスティックの読み取りに使用されます。

\$21 は、IC のテスト用のレジスタで、必ず 0 にしておかないといけません。とにかくこのレジスタにはさわらないことです。\$24、\$25 は、10 ビットのタイマー(A)を構成していて、カウント中にオーバーフローを起こしたときに CPU に IRQ 割り込みをかけることができます。また\$26 は、8 ビットのタイマー(B)で、タイマー(A)と同様に IRQ 割り込みをかけることができます。\$27 は、タイマー(A)、(B)およびチャンネル 3 の動作モードの設定に使用します。このレジスタも通常の使用方法で、かつ内部のタイマーを使わない場合には、さわらなくてもかまいません。

\$28 は、Key の ON/OFF(エンベロープ・ジェネレータヘスタート/ストップの合図を送る)に使われます。\$2D～\$2F は、プリスケラ機能といって、FM 音源に入力されるクロックをあらかじめ分周するためのものです。FM-7 シリーズでは、\$2D、\$2E をアクセスして FM 音源の分周比を 1/3 にすることが標準となっています。\$2D～\$2F には、データビットはありません。レジスタ番号を指定してアクセスするのみでセットされます。なお、\$21～\$2F までのレジスタの説明を、図 10-11 にまとめました。

\$30～\$8E までのレジスタは、オペレータの操作に関するものです。これらのレジスタの意味をすべて説明すると、長大なものとなってしまいます。それで\$30～\$8E とあとで説明する\$B0～\$B2 のレジスタを“音色データ”というブラック・ボックスとしてまとめて扱うことにします。

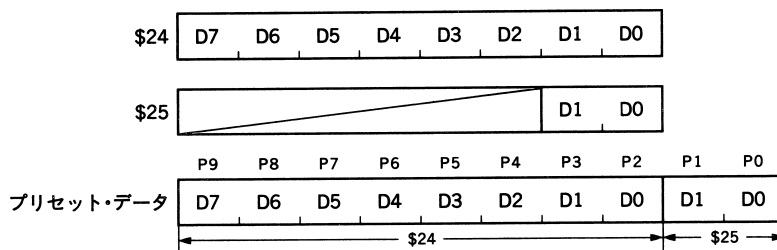
\$90～\$9E は、PSG タイプのエンベロープを使用する場合のレジスタです。\$A0～\$A6 は、F ナンバー(音程のデータ)とブロック(オクターブのデータ)を、チャンネル毎にセットするレジスタで、図 10-12 のようになっています。実際のオペレーションでは、1 オクターブ 12 音階分の F ナンバーをテーブルで持っておきます。そして出したい音の F ナンバーの上位 3 ビットとブロックの OR を取ったものを書き込みます。その後で、F ナンバーの下位 8 ビットを書き込みます。この順番で書き込まないと、正しいデータはセットされません。

\$A8～\$AE は、チャンネル 3 の F ナンバーとブロックを各スロット毎に単独にセットするためのレジスタです。\$27 のレジスタに、効果音モードまたは音声合成モードを選択したときのみ有効です。

最後に\$B0～\$B2 のレジスタは、チャンネル毎にスロット 1 のフィードバック量(自分自身の出力で自分自身を FM 変調する量)と、コネクションを各 3 ビットで指定します。図 10-9 を参照してください。

(4) FM 音源の操作手順

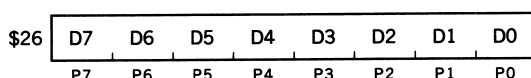
ここまで一気に説明してきましたが、わかっていただけましたでしょうか？ PSG に比べてかなりとっつきにくいモノであるというのは、一目でわかりますね。でも逃げださないでください。では、まずとにかく何でもいから音を出すことを考えてみたいと思います。手順としては、

\$24, \$25 タイマー(A)

$$\text{TOVA (ms)} = 12 * (1024 - \text{NA}) / f_{\text{FM (kHz)}}$$

$$f_{\text{FM}} = 1228.8 (\text{kHz}) * 1/3$$

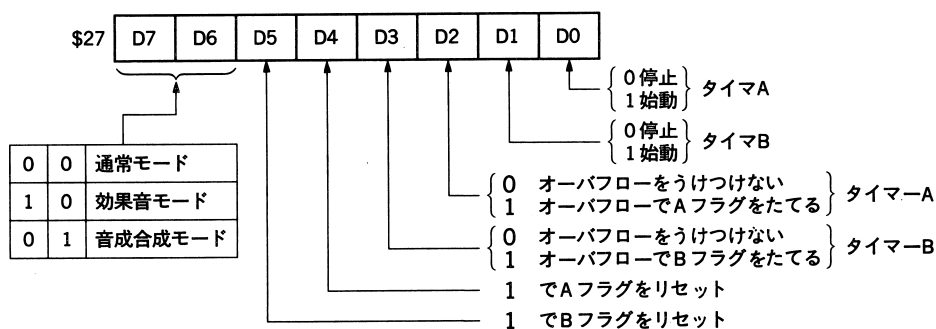
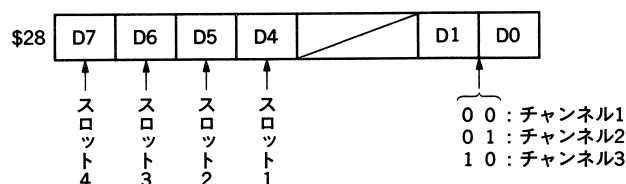
$$\text{NA} : P9 * 2^9 + P8 * 2^8 + P7 * 2^7 + \dots + P1 * 2^1 + P0$$

\$26 タイマー(B)

$$\text{TOVB (ms)} = 192 * (256 - \text{NB}) / f_{\text{FM (kHz)}}$$

$$f_{\text{FM}} = 1228.8 (\text{kHz}) * 1/3$$

$$\text{NB} : P7 * 2^7 + P6 * 2^6 + \dots + P1 * 2^1 + P0$$

\$27 タイマー, チャンネル3のモード**\$28 キーのON/OFF****\$2D, \$2E, \$2F プリスケaler**

\$2D	\$2E	\$2F	FM音源の分周数	SSG音源の分周数	OPNに入力できる最大周波数
—	—	A	1/2	1/1	1.4MHz
A	—	—	1/6	1/4	4.2MHz
A	A	—	1/3	1/2	2.1MHz

(Aはそのアドレスを入力し, —は入力しないことを意味する.)

図10-11 レジスタ(\$24~\$27)の詳細

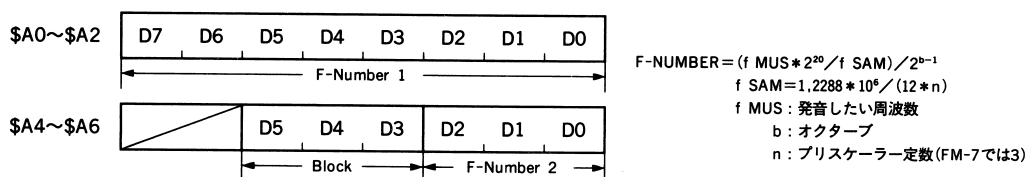


図10-12 ブロックとFナンバー

- ① IC を初期設定する。
- ② 音色データを書き込む。
- ③ F ナンバーとブロックを書く。
- ④ Key を ON にする。

となります。音を止める場合には、

- ⑤ KEY を OFF にする。

とつづきます。実際の音楽演奏プログラムでは、必要に応じて②～⑤または、③～⑤の処理を繰り返すことになります。

さて、ここでひとつ問題が出てきました。FM 音源のレジスタにデータを書き込む手段がないのです。もちろん、FM77AV をお使いの方は、F-BASIC V3.3 の SOUND 命令で簡単にできます。しかし他の FM-7 シリーズの場合は、そういうわけにはいきません。まず、レジスタにデータを書くルーチンから作らないといけません。この手順は複雑で、

- ① コマンドレジスタ(\$FD0D)にステータスリードコマンド(\$04)を書き込む。
- ② データレジスタ(\$FD0E)からステータスを読む。
- ③ コマンドレジスタにインアクティブコマンド(\$00)を書き込む。
- ④ ステータスのビット 7 がオンなら、①へ戻る。
- ⑤ データレジスタにアクセスしたいレジスタ番号をセットする。
- ⑥ コマンドレジスタにラッチアドレスコマンド(\$03)を書き込む。
- ⑦ コマンドレジスタにインアクティブコマンド(\$00)を書き込む。
- ⑧ データレジスタに書き込みたいデータをセットする。
- ⑨ コマンドレジスタにライトデータコマンド(\$02)を書き込む。
- ⑩ コマンドレジスタにインアクティブコマンド(\$00)を書き込む。

となります。PSG にくらべて、かなり長いですね。両者の違いを簡単に説明します。第一に、FM 音源はデータの取り込みが入力クロックに同期して行なわれます。それで新しいデータを書く前に、古いデータがすでに取り込まれたかどうかのチェックが必要です。(手順の①～④) 第二は、

\$00～\$0Fを除いてレジスタの読み出しができないことです。

では、まず、リスト 10-9 をごらんください。FM 音源を利用した簡易オルガンです。キーボードの Z, X, C, ……がドレミに対応していて、半音を含む1オクターブ分の音階を演奏することができます。また数字の1～8までで、オクターブを指定することも可能です。なお、実行はCAPS LOCK(大文字入力)の状態で行なってください。

リスト 10-9 簡易オルガンプログラム

```

1000 '*****
1001 '*      Tiny Organ ( FMオンケ"ン )      *
1002 '*      ( LIST 10-9 )   V3.3             *
1003 '*****
1010 OPNCOM=&HFD15
1020 OPNDAT=&HFD16
1030 '■ フ"リスケーラー センテイ
1040 A=&H2D:B=0:GOSUB 1400
1050 A=&H2E:B=0:GOSUB 1400
1060 '■ オンショク テ"ーター カキコミ
1070 FOR A=&H30 TO &HBC STEP 4
1080 READ B$:B=VAL("&H"+B$)
1090 GOSUB 1400
1100 NEXT
1110 READ B$:B=VAL("&H"+B$)
1120 A=&HB0:GOSUB 1400
1130 '■ ファンハ"ー シ"ュンヒ"
1140 DIM FDAT(13)
1150 FOR I=1 TO 13
1160 READ FDAT(I)
1170 NEXT
1180 '■ ORGAN メイン
1190 BLOCK=(4-1)*8
1200 A$=INKEY$:IF A$="" THEN 1200
1210 '■ KEY OFF
1220 A=&H28:B=0:GOSUB 1400
1230 I=INSTR("&ZSXDCVGBHJNM",A$)
1240 IF I=0 THEN 1350
1250 '■ ファンハ"ー / フ"ロック カキコミ
1260 FZ=FDAT(I)
1270 A=&HA4:B=(FZ * 256)OR BLOCK
1280 GOSUB 1400
1290 A=&HA0:B=FZ MOD 256
1300 GOSUB 1400
1310 '■ KEY ON
1320 A=&H28:B=&HF0
1330 GOSUB 1400
1340 GOTO 1200
1350 I=INSTR("&12345678",A$)
1360 IF I=0 THEN 1200
1370 BLOCK=(I-1)*8
1380 GOTO 1200
1390 '■ OPN アクセス ルーチン
1400 POKE OPNCOM,4
1410 STATUS=PEEK(OPNDAT)
1420 POKE OPNCOM,0
1430 IF STATUS AND 128 THEN 1400
1440 POKE OPNDAT,A
1450 POKE OPNCOM,3
1460 POKE OPNCOM,0
1470 POKE OPNDAT,B
1480 POKE OPNCOM,2
1490 POKE OPNCOM,0
1500 RETURN

```

```

1510 '■ オンショク テーター
1520 DATA 31.23.0C.01
1530 DATA 23.26.32.03
1540 DATA 9F.9F.0F.0F
1550 DATA 04.04.04.04
1560 DATA 1F.1F.1F.1F
1570 DATA F3.F3.F3.F3
1580 DATA 3A
1590 '■ ファンハマー テーター
1600 DATA 1004.1064.1127.1194
1610 DATA 1265.1340.1420.1505
1620 DATA 1594.1689.1790.1896
1630 DATA 2008

```

10-2-2 FM 音源のマシン語アクセス

(1) FM 音源サブルーチン

FM 音源のレジスタは、\$FD15、\$FD16 に配置されています。先ほど説明した手順をマシン語で記述したのが、リスト 10-10 です。PSG の場合と同様に、アキュムレータ A にレジスタ番号、アキュムレータ B に書き込むデータをセットしてこのルーチンをコールします。

例として、リスト 10-9 の改造リストをリスト 10-11 として載せておきます。あらかじめ、リスト 10-10 を \$5000 から入力した上で実行してみてください。

リスト 10-10 FM 音源レジスタ書き込みルーチン

```

01000 *****
01020 *      OPN WRITE ROUTINE      *
01040 *      ( LIST 10-10 )      V3.3      *
01060 *****
01070          OPT      NOGEN
01080 5000          ORG      $5000
01100          FD15    OPNCOM EQU    $FD15    OPN コメント"レシ"スター
01110          FD16    OPNDAT EQU    $FD16    OPN テーターレシ"スター
01130 5000 34      02      WRTOPN PSHS    A      AccA ホソ"ン
01140 5002 86      04      BUSY   LDA     #4      ステータスリット"コメント"
01150 5004 B7      FD15     STA     OPNCOM    <$04> カキコミ
01160 5007 B6      FD16     LDA     OPNDAT    ステータス >> AccA
01170 500A 7F      FD15     CLR     OPNCOM    インアクティブ"コメント" カキコミ
01180 500D 40          TSTA     ステータスノ Bit7=1
01190 500E 28      F2      5002    BMI     BUSY     ナラ ループスル
01200 5010 A6      E4      LDA     .S      AccA << レシ"スター"ハンコウ
01210 5012 B7      FD16     STA     OPNDAT    レシ"スター"ハンコウ カキコミ
01220 5015 86      03      LDA     #3      ラッチ アト"レス コメント"
01230 5017 B7      FD15     STA     OPNCOM    <$03> カキコミ
01240 501A 7F      FD15     CLR     OPNCOM    インアクティブ"コメント" カキコミ
01250 501D F7      FD16     STB     OPNDAT    テーター カキコミ
01260 5020 4A          DECA     ライトテーター"コメント"
01270 5021 B7      FD15     STA     OPNCOM    <$02> カキコミ
01280 5024 7F      FD15     CLR     OPNCOM    インアクティブ"コメント" カキコミ
01290 5027 35      82      PULS    A,PC     AccA フック / リターン
01310          5000      END      WRTOPN
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5028
PROGRAM ENTRY ADDR=5000

```

リスト 10-11 簡易オルガンプログラム(2)

```

1000 *****
1001 '*      Tiny Organ 2 ( FMオンケ"ン )      *
1002 '*      ( LIST 10-11 )      V3.3      *
1003 '*      LIST 10-10   カ"   ヒツヨウ テ"ス   *
1004 *****
1010 CLEAR 300,&H4FFD:LOADM"L10-10M"
1020 POKE &H4FFD,&HCC
1030 '■■ フ"リスケーラー センティ
1040 A=&H2D:B=0:GOSUB 1400
1050 A=&H2E:B=0:GOSUB 1400
1060 '■■ オンショク テ"ター カキコミ
1070 FOR A=&H30 TO &H8C STEP 4
1080 READ B$:B=VAL("&H"+B$)
1090 GOSUB 1400
1100 NEXT
1110 READ B$:B=VAL("&H"+B$)
1120 A=&H80:GOSUB 1400
1130 '■■ ファンハ"ー シ"ュンヒ"
1140 DIM FDAT(13)
1150 FOR I=1 TO 13
1160 READ FDAT(I)
1170 NEXT
1180 '■■ ORGAN メイン
1190 BLOCK=(4-1)*8
1200 A$=INKEY$:IF A$="" THEN 1200
1210 '■■ KEY OFF
1220 A=&H28:B=0:GOSUB 1400
1230 I=INSTR("ZSXDCVGBHJNM",A$)
1240 IF I=0 THEN 1350
1250 '■■ ファンハ"ー / フ"ロック カキコミ
1260 FZ=FDAT(I)
1270 A=&HA4:B=(FZ ¥ 256)OR BLOCK
1280 GOSUB 1400
1290 A=&HA0:B=FZ MOD 256
1300 GOSUB 1400
1310 '■■ KEY ON
1320 A=&H28:B=&HF0
1330 GOSUB 1400
1340 GOTO 1200
1350 I=INSTR("12345678",A$)
1360 IF I=0 THEN 1200
1370 BLOCK=(I-1)*8
1380 GOTO 1200
1390 '■■ OPN アクセス ルーチン
1400 POKE &H4FFE,A
1410 POKE &H4FFF,B
1420 EXEC &H4FFD
1500 RETURN
1510 '■■ オンショク テ"ター
1520 DATA 31,23,0C,01
1530 DATA 23,26,32,03
1540 DATA 9F,9F,0F,0F
1550 DATA 04,04,04,04
1560 DATA 1F,1F,1F,1F
1570 DATA F3,F3,F3,F3
1580 DATA 3A
1590 '■■ ファンハ"ー テ"ター
1600 DATA 1004,1064,1127,1194
1610 DATA 1265,1340,1420,1505
1620 DATA 1594,1689,1790,1896
1630 DATA 2008

```

(2) 音色データ

FM-7シリーズ用のFM音源カードには、40種類の音色データがテープに収められて付属しています。またFM77AVには、イニシエータROMの内部に77種類(FM77AVだから77種類なのではないか?)の音色データが収められています。

FM-7シリーズの場合は、付属の"TEXTED"を立ちあげた状態で音色データは、\$6000~\$654Fにロードされます。FM77AVの場合は、まずイニシエータROMを有効にしないといけません。F-BASIC V3.0を立ち上げた状態で、

```
CLEAR , &H6000
POKE &HFD10, 0
```

とすれば、\$6000~\$7FFFまでがイニシエータROMに切り換わります。音色データは、このうち

ビット	7	6	5	4	3	2	1	0	スロット								
0	—		DT DeTune		MULTI MULTiple				1								
1	—								3								
2	—								2								
3	—								4								
4	—		TL Total Level						1								
5	—								3								
6	—								2								
7	—								4								
8	KS Key Scale		—		AR Attack Rate				1								
9			—						3								
10			—						2								
11			—						4								
12	—		DR Decay Rate						1								
13	—								3								
14	—								2								
15	—								4								
16	—		SR Sustain Rate						1								
17	—								3								
18	—								2								
19	—								4								
20	SL Sustain Level				RR Release Rate				1								
21													3				
22																	2
23																	
24	—		Feed Back		Connect												
25	Flags																
26	Key Scale Depth																
27	PMS				AMS												
28	AMD																
29	PMD																
30	LFO frequency																
31	Delay Time																
32	Wave Form																
33	Sync Flag																

図10-13 音色データの形式

\$6C00～\$7639 までに収められています。データのフォーマットはいずれの場合も同じで、図 10-13 のようになっています。34 バイトのデータのうち、前の 25 バイトが実際に FM 音源に送られるデータです。残りは、LFO 関係(ビブラートやトレモロなど)のデータです。収められている音色データを図 10-14 に示します。リスト 10-9 の音色データは、これらと同じフォーマットなので、1520 行～1580 行のデータ文をこれらの音色データで置き換えることも可能です(前半 25 バイトのみ使用すること)。

FM音源カード付属データ				FM-77AVイニシエーターROM							
No.	音色名	No.	音色名	No.	音色名	No.	音色名	No.	音色名	No.	音色名
1	プラス1	21	ビブラフォン	1	トランペット	21	E, オルガン2	41	シンバル	61	バード
2	プラス2	22	シロフォン	2	ホルン	22	ハーブシコード	42	ティンパニー	62	ドッグ
3	トランペット	23	コト	3	チューバ	23	クラビネット	43	カウベル	63	テレフォン
4	ストリングス1	24	シタール	4	プラス1	24	ギター	44	ベル1	64	アラーム
5	ストリングス2	25	クラビネット	5	プラス2	25	E, ベース1	45	ベル2	65	ウィングラス
6	E, ピアノ1	26	ハーブシコード	6	ベル+プラス	26	E, ベース2	46	スチールドラム1	66	クラッシュ
7	E, ピアノ2	27	ベル	7	ピッコロ	27	シンセベース	47	スチールドラム2	67	ハートビート
8	E, ピアノ3	28	ハーブ	8	フルート	28	シロフォン	48	パーカッション1	68	フットステップ
9	ギター	29	ベル+プラス	9	クラリネット	29	グロッケン	49	パーカッション2	69	ユーフォー(UFO)
10	E, ベース1	30	ハーモニカ	10	オーボエ	30	ビブラフォン	50	トレイン1	70	レーザーガン
11	E, ベース2	31	スチールドラム	11	ファゴット	31	ハーブ	51	トレイン2	71	エクспロージョン1
12	E, オルガン1	32	ティンパニー	12	ストリングス1	32	リコーダ	52	カー	72	エクспロージョン2
13	E, オルガン2	33	トレイン	13	ストリングス2	33	ハーモニカ	53	モーターサイクル	73	サウンドエフェクト1
14	パイプオルガン1	34	アンビュランス	14	ピアノ	34	チター	54	グランプリ	74	サウンドエフェクト2
15	パイプオルガン2	35	トゥイート	15	E, ピアノ1	35	コト	55	パトロールカー	75	サウンドエフェクト3
16	フルート	36	レインドロップ	16	E, ピアノ2	36	スネアドラム1	56	アンビュランス	76	サウンドエフェクト4
17	ピッコロ	37	ホルン	17	E, ピアノ3	37	スネアドラム2	57	ヘリコプター	77	サインウェイブ
18	オーボエ	38	スネアドラム	18	パイプオルガン1	38	バスドラム	58	シップ		
19	クラリネット	39	カウベル	19	パイプオルガン2	39	オープンハイハット	59	ウェイブ		
20	グロッケン	40	パーカッション	20	E, オルガン1	40	クローズハイハット	60	レカンドロップ		

図10-14 音色データ一覧

10-2-3 FM 音源音楽ルーチン

今までの説明で、音の出し方はだいたいわかっていただけたと思います。しかし、ただ音を出すだけではあまりにもつまらないですね。そこで音楽を……ということになるわけなのですが、今までの説明以外にやらなければならない処理が、かなりあります。たとえば、旋律をきざむための音長やテンポの処理が必要ですし、音色や音量を変えるための処理も必要になってきます。ここでは、具体的にプログラムの仕様を決めながら音楽演奏プログラムを作成することにします。

まず最初に音長とテンポの処理を行なう方法には、大まかにわけて次の3通りの方法があります。

- ① プログラムでループを組む。
- ② FM 音源内蔵のタイマーで、IRQ 割り込みをかける。
- ③ FM-7 の 2mS のタイマー IRQ 割り込みを使う。

①の方法は、他のプログラムとの並行動作ができないという大きな欠点があります。そこで②か③かということになりますが、ここではコントロールの楽な③の方法を採用することにします。

つづいて音色データですが、自分で作るのはきわめて困難なので、とりあえずどこからか借用することにします。具体的には、FM 音源カードの付属データまたは FM77AV のイニシエータ ROM のどちらかとなります。データの吸い上げ方を最後にまとめておきますので、都合のいい方でお試してください。また、よく使う音色データだけを取り出してまとめておいてもよいでしょう。

さて、つづいて音量の設定を考えます。実は FM 音源を使う上で、音色データを作る次ぐらいにやっかいな処理がこれなのです。処理としては、オペレータの出力レベル(トータル・レベル)を下げればいいのですが、ただ下げるといっわけにはいきません。FM 音源の原理のところでも少し触れましたが、キャリアになっているオペレータのみを抽出して処理してやる必要があるのです。

前にもどって図 10-9 を見てください。アルゴリズム 0~3 まではオペレータ 4 のみがキャリアですが、アルゴリズム 4 ではさらにオペレータ 2、アルゴリズム 5、6 ではさらにオペレータ 3 もキャリアになります。また、アルゴリズム 7 はすべてのオペレータがキャリアです。以上のことより、音量の設定手順は次のようになります。

- ① オペレータ 4 に対してトータル・レベルの処理をする。
- ② アルゴリズム ≥ 4 なら、オペレータ 2 も同様にする。
- ③ アルゴリズム ≥ 5 なら、オペレータ 3 も同様にする。
- ④ アルゴリズム = 7 なら、オペレータ 1 も同様にする。

なお、トータル・レベルは減衰量で表しますので、実際の処理では音量の逆数を足し込むことになります。

さて、以上のことを元にして MML (Music Macro Language) の仕様を決めたのが、図 10-15 です。プログラムをわかりやすくするために、LFO 関係の処理はサポートしていません。したがって、音色データも前半 25 バイトのみ使用します。

MML は、F-BASIC V3.0 の文法になるべく近づけるよう努力しました。FM-7 シリーズのユーザーの方には、説明は特に必要はないと思います。また注意すべき点として、エラーに関する処理を特にしていないので、間違った MML を指定したときの動作は、保証されません。音長は計算精度の点から、なるべく 2 の n 乗(1, 2, 4, 8……)にしてください。また同じ理由で、符点音符

MMLコマンド	機 能	初期値
A ~ G [n]	音程	×
R [n]	休符 ※	×
L n	音長 ($1 \leq n \leq 64$)	4
O n	オクターブ ($1 \leq n \leq 8$)	4
V n	音量 ($0 \leq n \leq 15$)	8
# (シャープ)	半音上げる	×
. (ビリオド)	音長を1.5倍にする.	×
@ n	音色 ($1 \leq n \leq 40$ または77)	1

※ Rnでnを指定しない場合、音長は4とはならず、Lnの指定が採用されます。

図10-15 MMLの仕様

は音長 32 以上でお使いください。テンポは、Tn で指定するのではなく、プログラム実行時にワ
ークエリアに書き込みます。

かなり前置きが長くなってしまいました。それでは、以上の仕様を元にして作成した FM 音源
音楽演奏プログラムを、リスト 10-12 に示します。また、サンプルデータもリスト 10-13 に示
します。

リスト 10-12 FM 音源音楽プログラム

```
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5500 : 7E 56 B7 7E 57 02 7E 56 92 00 00 00 00 00 00 00 : C8
5510 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5520 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5530 : 00 00 00 00 00 B6 FD 03 84 04 26 02 8D 01 38 : 2F
5540 : B6 55 1A 26 10 10 8E 55 1B 8D 0E 8D 0C 8D 0A B6 : EA
5550 : 55 19 B7 55 1A 7A 55 1A 39 EC A4 27 0B A6 22 26 : 66
5560 : 05 17 01 27 8D 05 6A 22 31 29 39 EE A4 A6 C0 27 : 14
5570 : 1C 81 40 25 F8 27 1E 81 48 25 23 81 4F 27 30 81 : F8
5580 : 52 27 81 56 27 34 81 4C 27 3E 20 E0 CE 00 00 : D5
5590 : EF A4 16 00 F6 8D 38 4A A7 23 8D 78 20 CF 80 41 : 2D
55A0 : 48 E6 C4 C1 23 26 03 33 41 4C 17 00 8D 20 40 8D : 80
55B0 : 1E 4A 84 07 48 48 48 A7 24 20 B2 8D 12 48 84 1E : F1
55C0 : 40 8B 1E A7 26 8D 6D 20 A4 8D 24 A7 27 20 9E A6 : 57
55D0 : C4 80 30 81 0A 24 16 33 41 E6 C4 C0 30 C1 0A 24 : 36
55E0 : 0B 34 04 C6 0A 3D 1F 98 33 41 AB E0 39 4F 39 8D : 54
55F0 : DE A7 E2 26 04 A6 27 20 08 CC 00 40 4C E0 E4 22 : C4
-----
[cs] : 3E 3D 85 A2 FB 6E 1F 15 DA 81 39 F5 B7 A2 26 24 : 68

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5600 : FB A7 E4 E6 C4 C1 2E 26 05 33 41 44 AB E4 A7 22 : 5A
5610 : EF A4 35 84 C6 22 3D 8E 58 00 30 88 86 30 AB 28 : 9B
5620 : E6 80 17 00 F9 88 04 81 8F 25 F5 88 20 E6 84 17 : 5B
5630 : 00 EC E7 25 A6 23 C6 22 3D 8E 58 00 30 88 30 08 : BF
5640 : E6 25 C4 07 86 4C AB 28 8D 13 C1 04 25 0E 8D 00 : AD
5650 : C1 05 25 08 8D 07 C1 07 25 02 8D 01 39 34 04 E6 : 5B
5660 : 26 EB 82 17 00 88 80 04 35 84 48 8E 57 47 30 86 : C9
5670 : 86 A4 AB 28 E6 84 EA 24 17 00 A3 80 04 E6 01 17 : B1
5680 : 00 9C 86 28 C6 F0 EA 28 16 00 93 86 28 E6 28 16 : 8D
5690 : 00 8C 1A 10 10 8E 55 1B 8E 55 10 8D 0F 8D 0D 8D : 7A
56A0 : 0B A6 84 B7 55 19 7F 55 1A 1C EF 39 EC 81 ED A4 : 8A
56B0 : 6F 22 8D 07 31 29 39 1A 10 CC 2D 00 8D 60 CC 2E : 92
```


01370	S0AD	42	FCC	'BAG#ABAG#A'
01380	S0B7	4F	FCC	'06C405AR06C32.05G32.A32'
01390	S0CE	42	FCC	'BRARGRA32.G32.A32'
01400	S0DF	42	FCC	'BRARGRA32.G32.A32'
01410	S0F0	42	FCC	'BRARGRF#R'
01420	S0F9	45	FCC	'E8.ROSERFR'
01430	S103	47	FCC	'GRGRAGFE'
01440	S108	44	FCC	'04ERFR'
01450	S111	47	FCC	'GRGRAGFE'
01460	S119	44	FCC	'08.RCRDR'
01470	S121	45	FCC	'ERERFEDC'
01480	S129	4F	FCC	'04B405CRDR'
01490	S133	45	FCC	'ERERFEDC'
01500	S138	4F	FCC	'04B8.RBAG#A'
01510	S146	4F	FCC	'05C8R80C04B0SC'
01520	S154	45	FCC	'E8R8FED#E'
01530	S150	42	FCC	'BAG#ABAG#A'
01540	S167	4F	FCC	'06C405ARBR'
01550	S171	4F	FCC	'06CRO5BRARG#R'
01560	S17E	41	FCC	'ARERFRDR'
01570	S186	43	FCC	'C404B32.05C32.'
01572	S194	4F	FCC	'04B32B32.A32.B32'
01580	S1A4	41	FCC	'AB.R'
01590	S1AB	00	FCB	0
01610				
* << チャンネル2 テーマ >>				
01630	S1A9	40	P2DATA FCC	'020V7L1604'
01640	S1B3	52	FCC	'R4.ERERER'
01650	S1BC	52	FCC	'RBERERER'
01660	S1C4	52	FCC	'RBER8.ER'
01670	S1CC	52	FCC	'RBERERER'
01680	S1D4	52	FCC	'RBERERER'
01690	S1DC	52	FCC	'RBERERER'
01700	S1E4	52	FCC	'RBER8.05D#R'
01710	S1EF	4F	FCC	'04R4'
01720	S1F3	52	FCC	'R4.ERERER'
01730	S1FC	52	FCC	'RBERERER'
01740	S204	52	FCC	'RBER8.ER'
01750	S20C	52	FCC	'RBERERER'
01760	S214	52	FCC	'RBERERER'
01770	S21C	52	FCC	'RBERERER'
01780	S224	52	FCC	'RBER8.05D#R'
01790	S22F	52	FCC	'R404CRDR'
01800	S237	45	FCC	'ERERR4'
01810	S230	4F	FCC	'04B8GRO5CRDR'
01820	S249	45	FCC	'ERERR4'
01830	S24F	4F	FCC	'04B8.RARBR'
01840	S259	4F	FCC	'05CRCRR4'
01850	S261	4F	FCC	'04G#8ERARBR'
01860	S26C	4F	FCC	'05CRCRR4'
01870	S274	4F	FCC	'04G#8.RR4'
01880	S270	52	FCC	'RBERERER'
01890	S285	52	FCC	'RBERERER'
01900	S280	52	FCC	'RBER8.ER'
01910	S295	52	FCC	'R8D#R0#RD#R'
01920	S2A0	52	FCC	'RBER8.03BR'
01930	S2AA	52	FCC	'R8AR8.BR'
01940	S2B2	41	FCC	'ARARG#RG#R'
01950	S2BC	41	FCC	'AB.R'
01960	S2C0	00	FCB	0
01980				
* << チャンネル3 テーマ >>				
02000	S2C1	40	P3DATA FCC	'020V7L1603'
02010	S2C8	52	FCC	'R4A804CRCRCR'
02020	S2D7	4F	FCC	'03A804CRCRCR'
02030	S2E3	4F	FCC	'03A804CRO3A804CR'
02040	S2F3	4F	FCC	'03A804CRCRCR'
02050	S2FF	4F	FCC	'03E8BRBRBR'
02060	S309	45	FCC	'E8BRBRBR'
02070	S311	45	FCC	'E8BR02B803BR'

```

02080 531D      45      FCC      'E8.R'
02090 5321      52      FCC      'R4A804CRCRCR'
02100 5320      4F      FCC      '03A804CRCRCR'
02110 5339      4F      FCC      '03A804CR03A804CR'
02120 5349      4F      FCC      '03A804CRCRCR'
02130 5355      4F      FCC      '03E8BRBRBR'
02140 535F      45      FCC      'E8BRBRBR'
02150 5367      45      FCC      'E8BR02B803BR'
02160 5373      45      FCC      'E8.RR4'
02170 5379      4F      FCC      '03C804CR03E804ER'
02180 5389      4F      FCC      '03G8.RR4'
02190 5391      43      FCC      'C804CR03E804ER'
02200 539F      4F      FCC      '03G8.RR4'
02210 53A7      4F      FCC      '02A803ARC804CR'
02220 53B5      4F      FCC      '03E8.RR4'
02230 53B0      4F      FCC      '02A803ARC804CR'
02240 53CB      4F      FCC      '03E8.RR4'
02250 53D3      41      FCC      'A804CRCRCR'
02260 53D0      4F      FCC      '03A8CRCRCR'
02270 53E7      4F      FCC      '03A804CR03A804CR'
02280 53F7      4F      FCC      '03F8ARARAR'
02290 5401      45      FCC      'E8ARD8FR'
02300 5409      43      FCC      'C8ERD8FR'
02310 5411      45      FCC      'ERERERER'
02320 5419      4F      FCC      '02A8.R'
02330 541F      00      FCB      0
02350          5000      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=541F
PROGRAM ENTRY ADDR=5000

```

さてプログラムの実行方法ですが、サンプルデータ(リスト10-13)を例にして説明します。まず、リスト10-12とリスト10-13、それに後で述べる音色データを入力した後、EXEC &H5500 [] としてください。つづいて、EXEC &H5000 [] としてみましょう。いかがでしょうか？ 音が出ましたね。なお、実行を止めるときには必ず EXEC &H5503 [] を実行してください。

それでは最後にオリジナルな音楽データの入力方法を説明して、FM音源の説明を終わりにします。

まず、EXEC &H5500 [] としてください。その後に、チャンネル1のMMLデータの先頭アドレスを(\$5510, 11)に、チャンネル2を(\$5512, 13)に、チャンネル3を(\$5514, 15)に、そして、テンポデータを\$5516に書き込んでください。そして EXEC &H5506 [] とすれば、あなたのオリジナル・メロディの演奏が始まります。

テンポデータは、2mSのタイマー割り込み何回に1回処理をするかという分周比で表され、1が最も速く、255が最も遅くなります。テンポデータが16のとき、 $\downarrow \approx 120$ 程度になります。

注意点としては、BREAKキーを押したり、エラーを発生させたりすると、タイマー割り込みが止まってしまい、音が出なくなります。そのときには、POKE &HFD02, 4 [] と入力してください。

最後に音色データのリンク方法を説明します。この音楽プログラムでは、\$5800から音色データ

が入っていることになっています。そのため、音色データを FM 音源カードの付属データまたは、FM77AV のイニシエータ ROM から転送する必要があります。この音色データ転送プログラムを、リスト 10-14 (FM 音源カード付属データ用) およびリスト 10-15 (FM-77AV イニシエータ ROM 用) を示します。

プログラムの実行方法は、FM 音源カード付属データを使用する場合には、まず "TEXTED" を立ち上げてください。続いてリスト 10-14 を入力してから RUN  でプログラムを実行してください。ファイル名を入力すると、入力したファイル名で音色データがセーブされます。

FM77AV をお使いの方は、まず、F-BASIC V3.0 を立ち上げてください。その後でリスト 10-15 を入力して RUN  でプログラムを実行してください。ファイル名を入力すると、入力したファイル名で音色データがセーブされます。

リスト 10-14 音色データセーブ (FM 音源カード用)

```

1  '*****
2  '*   オンショク データ SAVE   *
3  '*   ( LIST 10-14 ) V3.0   *
4  '*****
10 CLEAR.&H5800
20 FOR I=0 TO &H54F
30 POKE &H5800+I,PEEK(&H6000+I)
40 NEXT
50 INPUT"FILE NAME",F$
60 SAVEM F$,&H5800,&H5D4F,&H5800

```

リスト 10-15 音色データセーブ (FM77AV 用)

```

1  '*****
2  '*   オンショク データ SAVE   *
3  '*   ( LIST 10-15 ) V3.3   *
4  '*****
10 CLEAR.&H4000
20 POKE&HFD10,0
30 FOR I=0 TO &HA39
40 POKE &H4000+I,PEEK(&H6C00+I)
50 NEXT
60 POKE&HFD10,2
70 FOR I=0 TO &HA39
80 POKE &H5800+I,PEEK(&H4000+I)
90 NEXT
100 INPUT"FILE NAME",F$
110 SAVEM F$,&H5800,&H6239,&H5800

```

10-3 OPNBIOS

OPNBIOS には、図 10-16 で示すように 15 種類の FM 音源アクセスに関する BIOS がサポートされています。そこでこの節では、OPNBIOS を利用して FM 音源を使用する例を示します。

リスト 10-16 が、そのサンプルプログラムです。このプログラムは、整数型変数に音色番号 (PLAY 文と同じもの) をセットしてコールすると、オクターブ 4 で “ドレミファソラシド” と音階を演奏するものです。

プログラムの実行は、まずリスト 10-16 を “L10-16M” というファイル名でセーブした後、リスト 10-17 を RUN してください。

SAVEM “L10-16M”, &H5000, &H509A, &H5000 

“オンショクバンゴウ=” と音色番号の入力を要求してきますので、1~77 で入力してください。このプログラムは OPNBIOS を使用したサンプルですから、F-BASIC V3.0 では動作しません。

リクエスト	名 称	機 能
0	WRTSSG	SSG 音源レジスタへ 1 バイトデータを書き込む
1	DWTSSG	SSG 音源レジスタへ 2 バイトデータを書き込む
2	REDSSG	SSG 音源レジスタから 1 バイトデータを読み込む
3	SSGCLR	全 SSG 音源レジスタのクリア
4	FRQSET	SSG 周波数データの書き込み
5	VOLSET	SSG 音量データの書き込み
6,7	-----	システムリザーブ
8	WRTFM	FM 音源レジスタへ 1 バイトデータを書き込む
9	KEYON	1 チャンネルのキーオン/キーオフ
10	KOFFAL	全 FM 音源チャンネルのスロットのキーオフ
11	WRTPRM	音色データの書き込み
12	FNOSSET	音階データの書き込み
13	-----	システムリザーブ
14	TTLSET	トータルレベルの書き込み
15	REDSTR	ステータスレジスタの読み込み
16	TRSPRM	音色データの転送
17	SELCAR	キャリアの判定

図10-16 OPNBIOS一覧

リスト 10-16 OPNBIOS サンプルプログラム

```

01000
01020
01040
01060
01090
01100      5000
01110      FB9B      FMBIOS
01120      5000 118C 6000      START      CMPS      #6000      SP>=$6000      ナラ      リターン
01130      5004 24      62      5068      BCC      ERR
01140      5006 81      02      CMA      #2      セイスウカ"タヘンスウ      イカ"イナラ      リターン
01150      5008 26      5E      5068      BNE      ERR
01160      500A A6      03      LDA      3,X      A <<      オンショクハ"ンコ"ウ
01170      500C 8E      5069      LDX      #WORK
01180      500F C6      0F      LDB      #15      TRSPRM      コマント"
01190      5011 AD      9F FB9B      JSR      [FMBIOS]      オンショクテ"ーター      ヨミコミ
01200      5015 25      51      5068      BCS      ERR
01210      5017 C6      0A      LDB      #10      KOFFAL      コマント"
01220      5019 AD      9F FB9B      JSR      [FMBIOS]      ス"チノチャンネルノ      KEY      OFF
01230      501D CC      000B      LDD      #11      WRTPRM      コマント"
01240      5020 8E      5069      LDX      #WORK
01250      5023 AD      9F FB9B      JSR      [FMBIOS]      オンショクテ"ーター      カキコミ
01260      5027 CC      F00D      LDD      #F00D      TTLSET      コマント"
01270      502A 8E      506D      LDX      #WORK+4
01280      502D AD      9F FB9B      JSR      [FMBIOS]      オンリョウ      センテイ
01290      5031 108E      508B      LDY      #FNUM
01300      5035 86      08      LDA      #8      LOOPカイスウ=8
01310      5037 34      02      PSHS      A
01320      5039 8E      5069      LOOP1      LDX      #WORK      FNUMBER      セット
01330      503C EC      A1      LDD      .Y++
01340      503E 8A      18      ORA      #18      オクターフ"=4
01350      5040 ED      84      STD      .X
01360      5042 CC      000C      LDD      #12      FNOSET      コマント"
01370      5045 AD      9F FB9B      JSR      [FMBIOS]      FNUMBER      カキコミ
01380      5049 CC      F009      LDD      #F009      KEYON      コマント"
01390      504C AD      9F FB9B      JSR      [FMBIOS]      KEY      ヲ      ON      ニスル
01400      5050 8D      0D      505F      BSR      WAIT      シ"カンマチ
01410      5052 CC      0009      LDD      #9      KEYON      コマント"
01420      5055 AD      9F FB9B      JSR      [FMBIOS]      KEY      ヲ      OFF      ニスル
01430      5059 6A      E4      DEC      .S      カウンター      テ"クリメント
01440      505B 26      DC      5039      BNE      LOOP1      LOOP      カイスウ      クリカエシ
01450      505D 35      82      PULS      A,PC
01470      505F 8D      00      5061      WAIT      BSR      *+2      シ"カンマチ      ルーチン
01480      5061 8E      0000      LDX      #0
01490      5064 30      1F      LEAX      -1,X
01500      5066 26      FC      5064      BNE      *-2
01510      5068 39
01550      5069      0022      ERR      RTS
01570      WORK      RMB      34
01590      5088      03EC      * <<      FNUMBER      テ"タ >>
01600      508D      0467      FNUM      FDB      1004      C<ト>"
01610      508F      04F1      FDB      1127      D<レ>"
01620      5091      053C      FDB      1265      E<ミ>"
01630      5093      05E1      FDB      1340      F<フ>"
01640      5095      0699      FDB      1505      G<ソ>"
01650      5097      0768      FDB      1689      A<ラ>"
01660      5099      07D8      FDB      1896      B<シ>"
01670      *
01680      5000      END      START

```

TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=509A
PROGRAM ENTRY ADDR=5000

リスト 10-17 OPNBIOS サンプル起動プログラム

```
1000 '*****
1002 '*  OPNBIOS TEST PROGRAM      *
1004 '*  ( LIST 10-17 )    V3.3    *
1006 '*  LIST 10-16   カ"   ヒツヨウ テ"ス  *
1008 '*****
1010 CLEAR 300,&H5000
1020 LOADM"L10-16M"
1030 DEF USR=&H5000
1040 INPUT "オンショク ハ"ンコ"ウ=",A%
1050 DUMMY=USR(A%)
1060 GOTO 1040
```

11-1 漢字 ROM へのアクセス

(1) 漢字 ROM データの読み出し

漢字キャラクタジェネレータ ROM として、FM-7 では MB83256×4、FM77AV では MB831124 が使用されています。そして、FM-7 ではオプションでしたが、FM77AV では標準装備されています。どちらの漢字 ROM にも、JIS 非漢字 453 字と JIS 第一水準漢字 2965 字が記録されています。この漢字 ROM をアクセスするには、

- ① BASIC の PRINT@文
- ② BIOS の KANJIR
- ③ I/O レジスタ (\$FD20～\$FD23)

の 3 とおりの方法があります。

I/O レジスタを直接アクセスして漢字 ROM データを読み出すには、メインシステム I/O レジスタ (\$FD20, \$FD21) に漢字 ROM アドレスを書き込みます。すると、メインシステム I/O レジスタ (\$FD22, \$FD23) に漢字 ROM のデータが読み出されます。そのとき、\$FD22 番地は漢字パターンの左側の 8 ドット分のデータに、\$FD23 番地は右側の 8 ドット分のデータになります(図 11-1)。

I/Oレジスタ	名 称	内 容
\$FD20	漢字アドレス	漢字アドレス (HIGH)
\$FD21		漢字アドレス (LOW)
\$FD22	漢字データ	漢字データ (LEFT)
\$FD23		漢字データ (RIGHT)

図11-1 漢字ROMアクセス用I/Oレジスタ

(2) 漢字パターンと漢字データとの対応

漢字アドレスは、ビット 4～ビット 15 の 12 ビットで漢字の種類を指定し、ビット 0～ビット 3 の 4 ビットで漢字パターンの列 No を指定します。ですから、たとえば「亜」の漢字アドレスは \$4010～\$401F で、その各漢字アドレスに対して 16 ドットの漢字パターンが図 11-2 のように対応します。

漢字ROM アドレス		列	漢字データ(LEFT)								漢字データ(RIGHT)							
\$FD20		\$FD21	\$FD22								\$FD23							
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
4	0	1	0															
		1																
		2																
		3																
		4																
		5																
		6																
		7																
		8																
		9																
		A																
		B																
		C																
		D																
		E																
		F																

図11-2 漢字パターンと漢字データとの対応

(3) JIS コードから漢字 ROM アドレスへの変換

漢字 ROM アドレス(12 ビット)は JIS コードとは別のものであり、漢字 ROM をアクセスするには、JIS コードから漢字 ROM アドレスへのコード変換が必要になります。ただし、BIOS(F-BASIC も含む)を利用して漢字 ROM をアクセスするときには、BIOS 自体が変換をしてくれるので JIS コードを指定すれば、アクセスできます。

ここでは、直接に漢字 ROM をアクセスするとき必要となるコード変換方法と変換プログラムを紹介します。

16 ビットの JIS コード(JC0~JC15 で示す)から、12 ビットの漢字 ROM アドレス(RA0~RA11 で示す)への変換方法は、図 11-3 のとおりです。

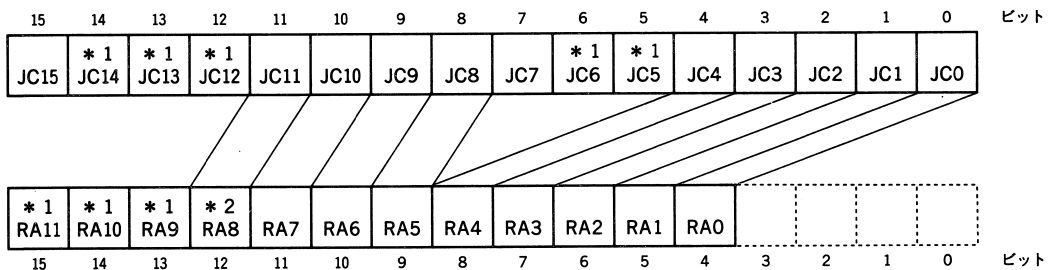
- ① JIS コードの JC0~JC4 の 5 ビットは、そのまま漢字 ROM アドレスの RA0~RA4 とします。
- ② JIS コードの JC8~JC11 の 4 ビットは、そのまま漢字 ROM アドレスの RA5~RA8 とします。ただし、(JC14, JC13, JC12, JC6, JC5)の内容が、(0, 1, 0, 1, 1)のときには、RA8 は"1"とします。
- ③ JIS コードの JC5, JC6, JC12, JC13, JC14 の 5 ビットは、下記の変換表に従ってコード変換を行ない、漢字 ROM アドレスの RA9~RA11 を決定します。

変換表

JC14	JC13	JC12	JC6	JC5	RA11	RA10	RA9
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	1	0
0	1	1	1	0	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	0	1	1	0
1	0	0	1	1	1	1	1

*2

〈JISコード〉



〈漢字ROMアドレス〉

*1 変換表による

*2 (JC14, JC13, JC12, JC6, JC5)が(01011)の時 RA8="1"

図11-3 JISコードから漢字ROMアドレスへの変換

以上の変換方法を用いて漢字ROMデータの読み出しを行なうプログラムを、リスト11-1に示します。漢字のJISコードを16進4桁で入力すると、その漢字ROMアドレスを求め表示します。そして指定された漢字パターンを読み出して拡大表示します。

リスト11-1 漢字パターン表示

```

10 *****
20 '* KANJI PATTERN DISPLAY *
30 '* ( LIST 11-1 ) V3.0/V3.3 *
32 *****
40 WIDTH 40,25:DEFINT A-Z:DIM JC(15),RA(11)
50 CLS:LOCATE 0,0:INPUT "JIS CODE=";A$
60 FOR I=0 TO 3
70   A=VAL("&H"+MID$(A$,4-I,1))
80   FOR J=0 TO 3
90     JC(I*4+J)=A AND &H01
100    A=INT(A/2)
110   NEXT
120 NEXT
130 FOR I=0 TO 3:RA(I)=JC(I):RA(I+5)=JC(I+8):NEXT:RA(4)=JC(4)
140 A=JC(5)*16+JC(6)*8+JC(12)*4+JC(13)*2+JC(14)
150 IF A=18 THEN RA(11)=0:RA(10)=0:RA(9)=0

```

```

160 IF A=10 THEN RA(11)=0:RA(10)=0:RA(9)=1
170 IF A=26 THEN RA(11)=0:RA(10)=0:RA(9)=0:RA(8)=1
180 IF A=22 THEN RA(11)=0:RA(10)=1:RA(9)=0
190 IF A=14 THEN RA(11)=0:RA(10)=1:RA(9)=1
200 IF A=30 THEN RA(11)=1:RA(10)=0:RA(9)=0
210 IF A=17 THEN RA(11)=1:RA(10)=0:RA(9)=1
220 IF A=9 THEN RA(11)=1:RA(10)=1:RA(9)=0
230 IF A=25 THEN RA(11)=1:RA(10)=1:RA(9)=1
240 AD1$=HEX$(8*RA(11)+4*RA(10)+2*RA(9)+RA(8))
250 AD2$=HEX$(8*RA(7)+4*RA(6)+2*RA(5)+RA(4))
260 AD3$=HEX$(8*RA(3)+4*RA(2)+2*RA(1)+RA(0))
270 AD4$=""
280 PRINT"ROM ADR = ";AD1$;AD2$;AD3$;AD4$
290 X=10:Y=32
300 FOR I=0 TO 15
310     POKE &HFD20,VAL("&H"+AD1$+AD2$)
320     POKE &HFD21,VAL("&H"+AD3$+AD4$)+I
330     GOSUB 390
340 NEXT
350 LOCATE 0,21:PRINT"Hit any key!!"
360 IF INKEY$<>"" THEN 50 ELSE 360
370 LOCATE 0,20
380 END
390 W=0:POKE VARPTR(W),PEEK(&HFD22):POKE VARPTR(W)+1,PEEK(&HFD23)
400 LOCATE 20,Y/8:PRINT RIGHT$("0000"+HEX$(W),4)
410 FOR J=15 TO 0 STEP -1
420     IF W AND &H01 THEN CC=2 ELSE CC=7
430     LINE (X+J*16,Y)-(X+J*16+12,Y+6),PSET,CC,BF
440     W=INT(W/2)
450 NEXT
460 Y=Y+8
470 RETURN

```

11-2 漢字 ROM に対する BIOS

漢字 ROM データを読み取るには、BIOS の KANJIIR を利用してもできます(図 11-4)。この KANJIIR では、JIS コードを指定すると漢字 ROM データ 32 バイトが、一度に読み取られます。しかも漢字 ROM アドレスへのコード変換をする必要もなく、たんへん便利です。KANJIIR を用いて漢字 ROM データを読み取るサンプルプログラムを、リスト 11-2 に示します。\$5000 番地からプログラムを入力して、EXEC &H5000 にて実行してください。\$5020 番地～\$503F 番地の 32 バイトに、“亜”の漢字 ROM データが読み出されます。リスト 11-1 の実行結果と比べてみてください。

KANJIIR (漢字パターン読み込み)

オフセット	内 容	ラベル名	ユーザ	BIOS
0	リクエスト番号	RQNO	22	
1	エラーステータス	RCBSTA		○
2,3	データバッファ先頭アドレス	RCBDBA	○	
4,5	JIS コード	RCBJCD	○	
6,7	リザーブ	——	——	——

図11-4 KANJIIRのRCB

リスト 11-2 BIOS による漢字 ROM 読み出し

```

00010 *****
00020 *      KANJIR SAMPLE PROGRAM      *
00030 *      ( LIST 11-2 )  V3.0/V3.3 *
00040 *****
01000          OPT      NOGEN
01002 5000          ORG      $5000
01010 5000 20      3E      5040 ENTRY  BRA      KANJI
01020 *
01030 5002          16      KANJIR FCB      22      REQ NO.
01040 5003          0001      RMB      1
01050 5004          5020      FDB      PAT      DATA ADR.
01060 5006          3021      FDB      $3021  JIS CODE
01070 5008          0002      RMB      2
01090 500A          10      SUBOUT FCB      16      REQ NO.
01100 500B          0001      RMB      1
01110 500C          5012      FDB      SUBBUF  DATA ADR.
01120 500E          002E      FDB      46      DATA LEN.
01130 5010          0002      RMB      2
01150 5012          0001      SUBBUF RMB      1
01160 5013          00      FCB      0
01170 5014          1C      FCB      $1C      COMMAND CODE
01180 5015          0064      FDB      100      X1
01190 5017          0064      FDB      100      Y1
01200 5019          0073      FDB      115      X2
01210 501B          0073      FDB      115      Y2
01220 501D          02      FCB      2      COLOR
01230 501E          00      FCB      0      FUNCTION
01240 501F          20      FCB      32      DATA MOVE LEN.
01250 5020          0020      PAT      RMB      32      PATTERN DATA
01260 5040 8E          5002      KANJI LDX      #KANJI
01270 5043 AD          9F FBFA      JSR      [$FBFA] KANJI PATTERN READ
01280 5047 8E          500A      LDX      #SUBOUT
01290 504A AD          9F FBFA      JSR      [$FBFA] KANJI PATTERN DISPLAY
01300 504E 25          01      5051      BCS      ERROR
01310 5050 39          RTS
01320 5051 39          ERROR  RTS
01330          5000      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5051
PROGRAM ENTRY ADDR=5000

```

11-3 漢字 JIS コード対応表示

FM-7 シリーズで漢字を表示する場合、通常、PRINT@文を使います。しかし漢字コードに JIS コードを指定しないとイケないので、調べるのがなかなか大変です。そこで、画面に漢字とその JIS コードを対応させて表示するプログラムを考えてみました(リスト 11-3)。

このプログラムは、漢字の読みを 1 文字入力すると、その読みの漢字を JIS コードとともに表示するものです。プリンタにコピーを取ることもできますから、よく使うものは保存しておくと便利です(図 11-5)。

プログラムは、RUN  で起動します。読みを入力するように求めてきますから、ア～ワの1文字を入力します。漢字の表示が1画面におさまらないときは、(N)EXT or (E)ND? と表示しますから、“N” か “E” のキーを押してください。漢字の表示が終了すると、HIT ANY KEY と表示します。いずれの場合でも、“P” を押すと画面のハードコピーがとれます。

リスト 11-3 漢字 JIS コード対応表示

```

10 *****
20 '*      KANJI <--> JIS CODE      *
30 '*      ( LIST 11-3 )      V3.0/V3.3 *
40 *****
1000 '■■■■ KANJI >> JIS CODE
1010 DIM CODE$(44)
1020 FOR I=0 TO 44
1030 READ CODE$:CODE$(I)=VAL("&H"+CODE$)
1040 NEXT
1050 WIDTH 40,25
1060 INPUT"ヨミカタ";Y$
1070 IF Y$="" THEN BEEP:GOTO 1050
1080 YOMI=ASC(Y$)-&H81
1090 IF YOMI<0 OR YOMI>43 THEN BEEP:GOTO 1050
1100 KCODE=CODE$(YOMI):KEND=CODE$(YOMI+1)
1110 CLS:PRINT"*** ヨミカタ('";Y$;"') ***"
1120 Y=8
1130 X=0
1140 PRINT@ (X+8,Y),KCODE
1150 SYMBOL(X,Y+16),HEX$(KCODE),1,1,7,0
1160 KCODE=KCODE+1
1170 IF (KCODE AND &H7F)=&H7F THEN KCODE=KCODE+&H8A
1180 IF KCODE=KEND THEN 1280
1190 X=X+40:IF X<640 THEN 1140
1200 Y=Y+25:IF Y<180 THEN 1130
1210 LOCATE 0,24
1220 PRINT"(N)EXT or (E)ND?";
1230 A$=INKEY$:A$=INPUT$(1)
1240 IF A$="E" OR A$="e" OR A$="I" THEN 1050
1250 IF A$="N" OR A$="n" OR A$="三" THEN 1110
1260 IF A$="P" OR A$="p" OR A$="t" THEN HARDC2
1270 BEEP:GOTO 1230
1280 LOCATE 0,24
1290 PRINT"HIT ANY KEY";
1300 A$=INKEY$:A$=INPUT$(1)
1310 IF A$="P" OR A$="p" OR A$="t" THEN HARDC2
1320 GOTO 1050
1330 DATA 3021,304A,3126,3141
1340 DATA 3177,323C,346B,3665
1350 DATA 3735,3843,3A33,3B45
1360 DATA 3F5A,4024,4139,423E
1370 DATA 434D,4445,4462,4546
1380 DATA 4660,4673,4728,4729
1390 DATA 4735,4743,485B,4954
1400 DATA 4A3A,4A5D,4B60,4C23
1410 DATA 4C33,4C3D,4C4E,4C69
1420 DATA 4C7B,4D3D,4D65,4D78
1430 DATA 4ESC,4E61,4F24,4F41
1440 DATA 4F54

```

*** ヨミカタ(カ) ***															
下	化	飯	何	伽	伽	住	加	可	嘉	夏	嫁	家	寡	科	瞞
323C	323D	323E	323F	3240	3241	3242	3243	3244	3245	3246	3247	3248	3249	324A	324B
果	架	歌	河	火	河	禍	禾	稼	箇	花	奇	茄	荷	華	菓
324C	324D	324E	324F	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	325A	325B
嶋	課	唾	貨	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蟻
325C	325D	325E	325F	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	326A	326B
賀	課	饑	駕	介	公	解	回	塊	塊	廻	快	怪	悔	恢	懷
326C	326D	326E	326F	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279	327A	327B
戒	拐	改	魁	晦	械	海	灰	界	皆	給	芥	蟹	開	階	目
327C	327D	327E	3321	3322	3323	3324	3325	3326	3327	3328	3329	332A	332B	332C	332D
凱	効	外	咳	害	崖	懼	懼	汪	碍	惹	芥	該	鑑	骸	淫
332E	332F	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	333A	333B	333C	333D
馨	蛙	垣	柿	蟻	鈎	劃	嚇	名	廟	拈	櫻	格	核	毅	獲
333E	333F	3340	3341	3342	3343	3344	3345	3346	3347	3348	3349	334A	334B	334C	334D
<N>EXT or <E>ND?															

図11-5 漢字JISコード対応表示実行例

11-4 JISコード表にない漢字ROMデータ

漢字ROMをアクセスするには、通常、BASICのPRINT@文か、BIOSのKANJIにてJISコードを指定して行ないます。これによって、JIS非漢字453字とJIS第一水準漢字2965字はすべて表示できます。

漢字ROMアドレス	パターン	漢字ROMアドレス	パターン	漢字ROMアドレス	パターン	漢字ROMアドレス	パターン
\$3010	(a)	\$3100	(p)	\$3440	㊦	\$3890	⦿
\$3020	(b)	\$3110	(q)	\$3450	㊧	\$38A0	----
\$3030	(c)	\$3120	(r)	\$3460	㊨	\$38B0	⋮
\$3040	(d)	\$3130	(s)	\$3470	㊩	\$3C10	<
\$3050	(e)	\$3140	(t)	\$3480	㊪	\$3C20	□
\$3060	(f)	\$3150	(u)	\$3490	㊫	\$3C30	▷
\$3070	(g)	\$3160	(v)	\$34A0	㊬	\$3C50	➡
\$3080	(h)	\$3170	(w)	\$3810	┌	\$3C60	←
\$3090	(i)	\$3180	(x)	\$3820	└	\$3C70	◆
\$30A0	(j)	\$3190	(y)	\$3830	┌	\$3C80	▶
\$30B0	(k)	\$31A0	(z)	\$3840	└	\$3C90	■
\$30C0	(l)	\$3400	㊭	\$3850	┌	\$3CA0	◀▶
\$30D0	(m)	\$3410	㊮	\$3860	┌	\$3CC0	====
\$30E0	(n)	\$3420	㊯	\$3870	┌	\$3CD0	△
\$30F0	(o)	\$3430	㊰	\$3880	┌	\$3CE0	→

図11-6 JISコード表にない漢字ROMデータ

しかし、漢字 ROM (MB831124) には、JIS コード表にない特殊記号用データが 60 種類も記録されています。図 11-6 にその一覧表を示します。これらは、JIS コードと対応づけられていないので、BASIC の PRINT@ では表示できません。それで、メインシステム I/O レジスタ (\$FD20 ~ \$FD23) を用いて、漢字 ROM アドレスによって直接漢字 ROM をアクセスするプログラムをリスト 11-4 に示します。漢字 ROM アドレスを入力すると、その漢字パターンを拡大表示します。図 11-6 の表をもとに、実際に確かめてください。

リスト 11-4 JIS コード表にない漢字 ROM データ表示

```

10 '*****
20 '* KANJI ROM DATA DISPLAY *
30 '* ( LIST 11-4 ) V3.0/V3.3 *
40 '*****
100 WIDTH 40,25:DEFINT A-Z
110 CLS:LOCATE 0,0:INPUT "ROM ADDRESS = ";A$
120 X=10:Y=32
130 FOR I=0 TO 15
140     POKE &HFD20,VAL("&H"+LEFT$(A$,2))
150     POKE &HFD21,VAL("&H"+RIGHT$(A$,2))+I
160     GOSUB 220
170 NEXT
180 LOCATE 0,21:PRINT"Hit any key!!"
190 IF INKEY$(">") THEN 110 ELSE 190
200 LOCATE 0,20
210 END
220 W=0:POKE VARPTR(W),PEEK(&HFD22):POKE VARPTR(W)+1,PEEK(&HFD23)
230 LOCATE 20,Y/8:PRINT RIGHT$("0000"+HEX$(W),4)
240 FOR J=15 TO 0 STEP -1
250     IF W AND &H01 THEN CC=2 ELSE CC=7
260     LINE (X+J*16,Y)-(X+J*16+12,Y+6),PSET,CC,BF
270     W=INT(W/2)
280 NEXT
290 Y=Y+8
300 RETURN

```

11-5 ファンクションキーエリアに漢字表示

漢字をファンクションキーに定義できないことは、キー入力の章の説明で明らかです。しかし市販のパッケージソフトには、ちゃんとファンクションキー表示エリアに漢字を表示して、それに対応するキーが押されると、その処理を行なうようになっているものがあります。いったい、どうしているのでしょうか。

これは実は、ファンクションキーエリアの位置に PRINT@ 文で漢字を書いて、あたかもファンクションキーの表示であるかのように見せかけているだけなのです。そしてファンクションキー割り込みを使って、それぞれに対応する処理を行なうようにしているのです。

そこで、以上のことをプログラムした簡単なメニュー(処理選択)プログラムを紹介します。PF1 ~ PF5 のファンクションキーを押すと、それに対応したメッセージが表示されます(リスト 11-5)。

リスト 11-5 漢字メニュープログラム

```

10 '*****
20 '* KANJI MENU PROGRAM *
30 '* ( LIST 11-5 ) V3.0/V3.3 *
40 '*****
50 WIDTH 40,25:CONSOLE 0,23,0,0
60 ON KEY(1) GOSUB 200:KEY(1) ON
70 ON KEY(2) GOSUB 210:KEY(2) ON
80 ON KEY(3) GOSUB 220:KEY(3) ON
90 ON KEY(4) GOSUB 230:KEY(4) ON
100 ON KEY(5) GOSUB 240:KEY(5) ON
110 FOR I=0 TO 4:LINE (I*128,184)-(I*128+112,199),PSET,5,BF:NEXT
120 RESTORE 270:COLOR 0
130 FOR I=0 TO 4
140     READ A$:PRINT@ (I*128+24,184),VAL("&H"+A$)
150     READ A$:PRINT@ (I*128+64,184),VAL("&H"+A$)
160 NEXT
170 COLOR 7:PRINT@ (0,0),&H3D68,&H4D7D,&H3960,&H4C5C,&H2472,&H412A,&H2
473,&H2447,&H242F,&H2440,&H2435,&H2424,&H2123
180 GOTO 180
190 END
200 PRINT@ &H3A6E,&H2340,&H402E:RETURN
210 PRINT@ &H4A51,&H2340,&H3939:RETURN
220 PRINT@ &H3A6F,&H2340,&H3D7C:RETURN
230 PRINT@ &H3821,&H2340,&H3A77:RETURN
240 PRINT@ &H3D2A,&H2340,&H4E3B
250 KEY(1) OFF:KEY(2) OFF:KEY(3) OFF:KEY(4) OFF:KEY(5) OFF
260 CONSOLE 0,25,0,0:END
270 DATA 3A6E,402E,4A51,3939,3A6F,3D7C,3821,3A77,3D2A,4E3B

```

11-6 拡張漢字表示プログラム

BASICで漢字表示するには、PRINT@文にて行ないます。しかしこのPRINT@文による漢字表示は、機能的にみて便利とはいえません。それで、マシン語による拡張漢字表示プログラムを考えてみました。ただし、4096色モードにおける漢字表示方法は、8色モード時とまったく異なりますので、次節でとりあげます。

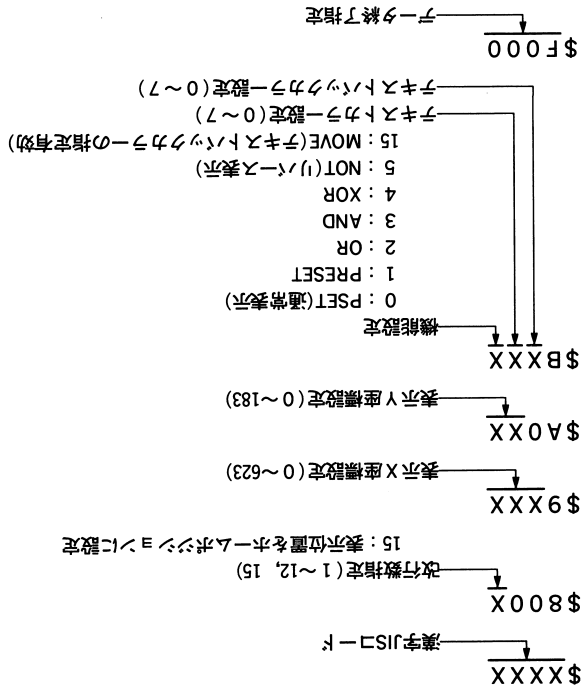
リスト 11-6、リスト 11-7 がそのアセンブルリストです。このプログラムは、漢字表示用データをセットして、(\$5002,\$5003)番地にそのデータアドレスを書き込み、EXEC &H5000 ㊦ で実行します。

漢字表示用データの形式は、2バイト単位のデータで図 11-7 のとおりです。機能に“5”を指定すると、漢字のリバース表示となります。また機能に“15”を指定すると、テキストバックカラーの指定が有効となり、テキストバックカラーで画面消去してから、テキストカラーで漢字が表示されます。改行指定において改行数=15を指定すると、表示位置がホームポジション(X=0,Y=0)に設定されます。

ADR :	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	[cs]
5000 :	ED	88	15	C3	00	0F	ED	88	19	16	FF	43	7F	50	04	C5	DA
5010 :	84	0F	ED	88	13	C3	00	0F	ED	88	17	16	FF	51	84	0F	72
5100 :	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	[cs]
5110 :	ED	88	15	C3	00	0F	ED	88	19	16	FF	43	7F	50	04	C5	DA
5120 :	08	27	0A	7C	50	04	C4	F0	84	0F	87	50	4E	1F	98	44	A0
5130 :	44	44	44	C4	0F	ED	88	18	16	FF	24	39	00	00	00	00	A1
5140 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5150 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5160 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5170 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

リスト 11-6 拡張漢字表示プログラム

図11-7 拡張漢字表示用データの形式



```

5180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : BD 02 50 8B 72 C3 39 A2 A0 AC F1 E2 CC C0 20 18 : BD

```

```
SAVEM "L11-6M",&H5000,&H513B,&H5000
```

リスト 11-7 拡張漢字表示の実行例

```

10 '*****
20 '* SUPER KANJI DISPLAY SAMPLE *
30 '* < LIST 11-7 > V3.0 *
32 '* LIST 11-6 カ" ヒツウ テ"ス *
34 '*****
40 CLEAR .&H5000:LOADM"L11-6M":CLS
50 POKE &H5002,&H52:POKE &H5003,&H00
60 FOR I=0 TO 1000 STEP 2
70 READ A$
80 IF A$="FFFF" THEN 120
90 POKE &H5200+I,VAL("&H"+LEFT$(A$,2))
100 POKE &H5201+I,VAL("&H"+RIGHT$(A$,2))
110 NEXT
120 EXEC &H5000
130 END
140 DATA 9040,A030,B12F
150 DATA 2340,2346,234D,2337,2337,2341,2356,244E,4643,4427,2340
160 DATA 8002,B035,2340,2331,2125
170 DATA 8070,2340,3643,305B,244E,2334,2330,2339,2336,3F27,4631,3B7E,4
930,3C28,2123
180 DATA 8002,B035,2340,2332,2125
190 DATA 8070,2340,2346,234D,323B,383B,493B,3060,4175,4877,2123
200 DATA 8002,B035,2340,2333,2125
210 DATA 8070,2340,392D,4267,244A,2340,2566,213C,2536,2126,2561,2562,2
56A
220 DATA 214A,234D,2341,2358,2331,2339,2332,234B,2550,2524,2548,214B,2
123
230 DATA F000,FFFF

```

11-7 高速漢字表示プログラム

4096色モードにおける漢字表示は、BASICでは、8色モードと同様にPRINT@文にて行なえます。しかし、このPRINT@文による漢字表示は、驚くほど低速です。これは、PUT BLOCK (コマンドコード=\$1E)という4096色モード用のサブシステムのコマンドを用いて表示しているからなのです。このPUT BLOCKを用いてひとつの漢字を表示するには、何と512バイトものパターンデータをサブシステムに送らないといけません。このパターンデータの送出のハンドシェイクに、多くの時間が費やされているようです。

そこでPUT BLOCKを使わずに、ダイレクトアクセスによってメインシステムから直接VRAMへ漢字パターンデータを送り、高速に漢字表示するプログラムを考えてみました。

リスト11-8, リスト11-9 がその高速漢字表示プログラムです。利用方法は、前節の拡張漢字表示プログラムと同様にしました。ただしこのプログラムでは、指定されたテキストカラーの最低輝度のVRAM(R0, G0, B0)のみに漢字パターンを書き込みます。それで、PALETTE文にて色の設定をしておく必要があります。また機能設定には、PSET(通常表示), NOT(リバース表示), MOVE(テキストバックカラーの指定)のみが可能です。さらに表示X座標は、8の倍数である必要があります。設定データの形式を図11-8に示します。

画面全体に漢字表示するのに、この高速表示プログラムを使った場合と、PRINT@を使った場合で比較してみました。リスト11-10とリスト11-11です。ほぼ15倍ほど高速に表示されることがわかります。

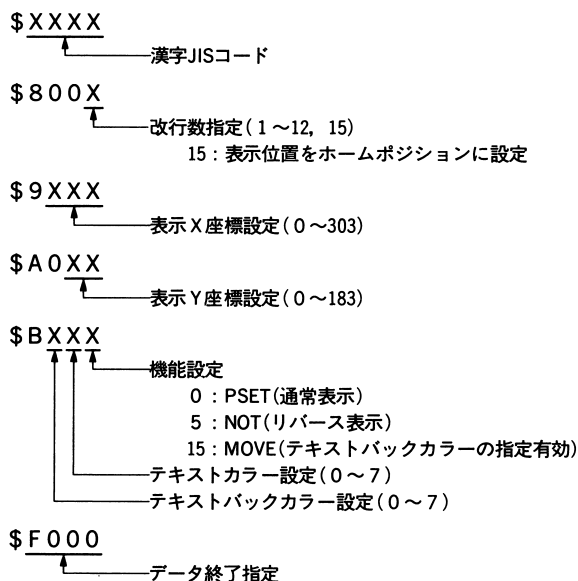


図11-8 高速漢字表示用データの形式

リスト11-8 高速漢字表示プログラム

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 20 4C 00 00 00 00 00 00 07 00 00 00 16 00 50 14 : ED
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 10 00 50 3C 00 12 00 00 00 00 15 00 : C3
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 02 FF FF 10 BE : CE
5050 : 50 02 17 01 41 8E 50 0C EC A1 85 80 27 1C 85 40 : 2F
5060 : 10 26 01 89 85 20 27 09 85 10 10 27 01 52 16 01 : FB
5070 : 57 85 10 10 26 01 41 16 01 85 ED 04 AD 9F FB FA : 32
5080 : 86 50 0A 27 32 17 01 27 17 01 1D 8E 50 34 FC 50 : 3B
5090 : 04 ED 0F C3 00 0F ED 88 13 FC 50 06 ED 88 11 C3 : F5

```

```

50A0 : 00 0F ED 88 15 AD 9F FB FA 17 00 EA 86 1D B7 FD : 32
50B0 : 89 4F B7 94 10 20 14 B6 50 09 81 05 26 0D CE 50 : 4D
50C0 : 14 C6 20 A6 C4 43 A7 C0 5A 26 F8 8E 90 00 FC 50 : FD
50D0 : 04 44 56 44 56 44 56 30 88 FC 50 06 86 28 3D 30 : FA
50E0 : 88 86 1D B7 FD 89 4F B7 94 30 8D 7C 86 1D B7 FD : 95
50F0 : 89 86 6D B7 94 30 8D 70 B6 50 08 85 01 27 08 86 : 33

[cs] : 46 AA D8 28 FE E2 82 DE 1C 07 4D C5 70 5E 98 70 : 38

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 12 B7 FD 89 4C B7 FD 8A 8D 42 B6 50 08 85 02 27 : 64
5110 : 08 86 16 B7 FD 89 4C B7 FD 8A 8D 30 B6 50 08 85 : BE
5120 : 04 27 08 86 1A B7 FD 89 4C B7 FD 8A 8D 1E FC 50 : 94
5130 : 04 C3 00 10 10 83 01 40 25 0C FC 50 06 C3 00 10 : 01
5140 : FD 50 06 CC 00 00 FD 50 04 16 FF 09 34 10 CE 50 : FD
5150 : 14 C6 10 34 04 EC C1 AA 84 EA 01 ED 84 30 88 28 : 39
5160 : 35 04 5A 26 EE 35 10 39 86 10 C6 06 34 16 B7 FD : 85
5170 : 89 4C B7 FD 8A CE 50 14 C6 10 34 04 EC C1 43 53 : 96
5180 : 44 84 E4 01 ED 84 30 88 28 35 04 5A 26 EC 35 16 : 4E
5190 : 4C 4C 5A 26 D7 39 B6 FD 05 28 FB 1A 50 86 80 87 : 2D
51A0 : FD 05 B6 FD 05 2A F8 39 4F B7 FD 05 1C AF 39 F6 : 1A
51B0 : FC 80 CA 80 F7 FC 80 39 84 0F FD 50 04 16 FE 95 : FF
51C0 : 84 0F FD 50 06 16 FE 8D 7F 50 04 C5 08 27 22 7C : F2
51D0 : 50 0A C4 F0 84 0F B7 50 08 84 01 B7 50 3F B6 50 : 84
51E0 : 08 84 02 44 B7 50 40 86 50 08 84 04 44 44 B7 50 : 44
51F0 : 41 1F 98 44 44 44 44 44 C4 0F FD 50 08 16 FE 56 C4 : 5E

[cs] : FD 9E 5E 65 34 05 FF 9F 88 81 0E AB 71 AC 27 0C : A7

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5200 : 0F C1 0F 27 13 58 58 58 F3 50 06 FD 50 06 CC : E1
5210 : 00 00 FD 50 04 16 FE 3D CC 00 00 20 EF 86 39 B7 : F3
5220 : FD 89 4C B7 FD 8A 8D 87 17 FF 7D 39 00 00 00 00 : F0
5230 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5240 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5250 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5260 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5270 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5280 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5290 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
52F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

[cs] : 0C 4A 58 2E 14 F8 E3 1C 3B F2 CD 5F EC D6 3F 83 : C4

```

SAVEM "L11-8M",&H5000,&H5228,&H5000

リスト11-9 高速漢字表示の実行例

```

10 '*****
20 '* FAST KANJI DISPLAY *
30 '* (LIST 11-9) V3.3 *
32 '* LIST 11-8 カ E"ヨ"テ"入 *
34 '*****
40 CLEAR,&H5000:LOADM"L11-8M":SCREEN0 1:CLS
50 PALETTE0
60 PALETTE 1,[0,0,240]
70 PALETTE 16,[0,240,0]
80 PALETTE 17,[0,240,240]

```

```

90 PALETTE 256,[240,0,0]
100 PALETTE 257,[240,0,240]
110 PALETTE 272,[240,240,0]
120 PALETTE 273,[240,240,240]
130 POKE &H5002,&H53:POKE &H5003,&H00
140 FOR I=0 TO 1000 STEP 2
150     READ A$
160     IF A$="FFFF" THEN 200
170     POKE &H5300+I,VAL("&H"+LEFT$(A$,2))
180     POKE &H5301+I,VAL("&H"+RIGHT$(A$,2))
190 NEXT
200 EXEC &H5000
210 END
220 DATA 9040,A020,B12F
230 DATA 2340,2346,2340,2337,2337,2341,2356,244E,4643,4427,2340
240 DATA 8002,B035,2340,2331,2125
250 DATA B070,2340,3643,305B,244E,2334,2330,2339,2336,3F27,4631,3B7E,4
93D,3C28,2123
260 DATA 8002,B035,2340,2332,2125
270 DATA B070,2340,2346,234D,323B,383B,4938,3060,4175,4877,2123
280 DATA 8002,B035,2340,2333,2125
290 DATA B070,2340,392D,4267,244A,2340,2566,213C,2536,2126,2561,2562,2
56A
300 DATA 214A,234D,2341,2358,2331,2339,2332,234B,2550,2524,2548,214B,2
123
310 DATA F000,FFFF

```

リスト 11-10 高速漢字表示プログラムによる表示

```

10 '*****
20 '* FAST KANJI DISPLAY TEST *
30 '* ( LIST L11-10 ) V3.3 *
32 '* LIST 11-8 カ ヒツヨウ ティス *
34 '*****
40 CLEAR ,&H5000:LOADM"L11-8M":SCREEN0 1:CLS:LOCATE 0,10
50 TIME$="00:00:00"
60 PALETTE0
70 PALETTE 1,[0,0,240]
80 PALETTE 16,[0,240,0]
90 PALETTE 17,[0,240,240]
100 PALETTE 256,[240,0,0]
110 PALETTE 257,[240,0,240]
120 PALETTE 272,[240,240,0]
130 PALETTE 273,[240,240,240]
140 POKE &H5002,&H53:POKE &H5003,&H00
150 POKE &H5300,&H80
160 POKE &H5301,&H70
170 POKE &H5302,&H23
180 POKE &H5303,&H46
190 POKE &H5304,&HFO
200 POKE &H5305,&H00
210 FOR I=0 TO 20*12-1
220     EXEC &H5000
230 NEXT
240 PRINT TIME$
250 END

```

リスト 11-11 PRINT@文による表示

```

10 *****
20 '* PRINT@ KANJI DISPLAY TEST *
30 '* ( LIST 11-11 ) V3.3 *
32 *****
40 SCREEN@ 1:CLS:LOCATE 0,10
50 TIME$="00:00:00"
60 COLOR 7
70 FOR Y=0 TO 11*16 STEP 16
80     FOR X=0 TO 19*16 STEP 16
90         PRINT@ (X,Y), &H2346
100     NEXT
110 NEXT
120 PRINT TIME$
130 END

```

11-8 漢字ビットイメージプリント

通常、漢字をプリントアウトするには、プリンタに漢字 ROM が付いている漢字プリンタが必要です。しかし、ビットイメージ機能(ドット対応グラフィック)が可能なプリンタであれば、何らかの方法で漢字のビットパターンを出力してやることで、漢字のプリントアウトができます。ここでは、エプソン系9ピンプリンタ(MP80, RP80, MB27401 など)でパソコン本体の漢字 ROM を用いて、漢字をプリントアウトする方法を紹介します。

ではとりあえず、漢字の「字」をプリントアウトすることを考えてみます。その手順は、次のようになります。

- ① BIOS を利用して、漢字フォントパターンをバッファに読み込む。
- ② その漢字 ROM のフォントパターンを、プリンタ用ビットイメージデータに変換する。
- ③ 変換したデータをプリンタに出力する。

①と③は簡単ですが、②は相当やっかいです。一般に、9ピンのプリンタで16ドットの漢字をプリントさせるには、8ドットずつ2回に分けてプリントします。

- ① 偶数番目のドットを、縦8ドット×横16ドット分プリントする。
- ② 1ドットの半分、紙送りする。
- ③ 奇数番目のドットを、縦8ドット×横16ドット分プリントする。

エプソン系9ピンプリンタで縦8ドット×横16ドット分のデータを出力するには、次の制御コマンドを送った後、ドットデータを16バイト送ります。

```
LPRINT CHR$(27); "L"; CHR$(16); CHR$(0);
```


また1ドットの半分の紙送りは次の制御コマンドを出力します。

```
LPRINT CHR$(27); "J"; CHR$(1);
```

漢字フォントパターンビットデータは、横につながっていますが、ビットイメージプリントするには、縦につながったデータを順に出力する必要があります。つまり、漢字フォントパターンを左に90°回転させたデータの偶数、奇数番目のドットのデータを作成しなければならないわけです。

それでは、「字」というフォントパターンをもとに、その変換方法を説明します。図11-9に示すように、「字」の偶数番目のビットデータは、(\$00,\$64,\$44,……)のようになります。そして奇数番目のビットデータは、(\$00,\$40,\$00,……)のようになります。したがって、BASICで「字」をビットイメージプリントするには、リスト11-12のようにします。

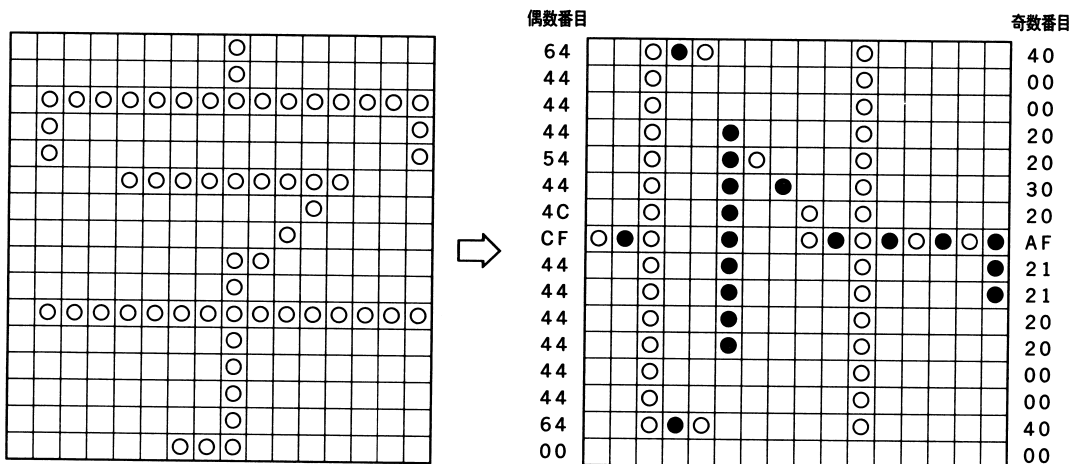


図11-9 ビットイメージプリント用データの変換方法

リスト 11-12 ビットイメージプリント

```
10 *****
20 *   BIT IMAGE PRINT (KANJI)   *
30 *   ( LIST 11-12 ) V3.0/V3.3 *
40 *****
1010 LPRINT CHR$(27);"L";CHR$(16);CHR$(0);
1020 FOR I=0 TO 15
1030   READ D$:D=VAL("&H"+D$)
1040   LPRINT CHR$(D);
1050 NEXT
1060 LPRINT CHR$(27);"J";CHR$(1);
1070 LPRINT CHR$(13);
1080 LPRINT CHR$(27);"L";CHR$(16);CHR$(0);
1090 FOR I=0 TO 15
1100   READ D$:D=VAL("&H"+D$)
1110   LPRINT CHR$(D);
```

```

1120 NEXT
1130 LPRINT
1140 DATA 00.64.44.44.44.44.44.44
1150 DATA CF.4C.44.54.44.44.44.64
1160 DATA 00.40.00.00.20.20.21.21
1170 DATA AF.20.30.20.20.00.00.40

```

フォントパターンから、プリンタ用ビットイメージデータに図を書いて変換していたのでは、非常に能率が悪いですね。そこで、データ変換を含めて漢字プリントのすべての手順をマシン語で高速に行なう漢字プリントプログラムを作ってみました。リスト 11-13 です。

使用法は、まず漢字プリントプログラムを入力して、“L11-13M”というファイル名で SAVEM します。

SAVEM “L11-13M”, &H5000, &H517B, &H5000 

それから、BASIC で出力したい漢字コードを整数型変数に入れて、このルーチンをコールします。56 文字出力するか、CR コード(\$000D)が送られると改行します。リスト 11-14 にそのサンプルプログラムを示しますので、参考にしてください。また、この漢字プリントプログラムの印字例を図 11-10 に示します。

リスト 11-13 漢字プリントプログラム

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 50 81 00 00 00 00 00 00 00 00 00 00 00 00 00 : 7F
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50A0 : 00 00 00 00 00 0E 00 50 74 00 04 0E 00 50 74 00 : A8
50B0 : 11 81 02 26 1B AE 02 8C 00 0D 27 15 F6 50 03 CE : 71
50C0 : 50 04 58 AF C5 F6 50 03 5C C1 38 F7 50 03 24 01 : 2D
50D0 : 39 F6 50 03 27 05 CE 50 85 8D 25 CC 01 0D 8D 10 : 7A
50E0 : F6 50 03 27 05 CE 50 87 8D 16 7F 50 03 CC 23 0D : 8B
50F0 : FD 50 76 CC 1B 4A FD 50 74 8E 50 A5 6E 9F FB FA : 3A

[cs] : 0B 6B D4 CB 27 CF 6D 06 56 FF 57 DB B8 1B 46 E6 : 04

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 10 8E 50 04 8E 50 A5 CC 1B 4C FD 50 74 F6 50 03 : B2
5110 : 86 11 3D F7 50 76 B7 50 77 AD 9F FB FA B6 50 03 : 59
5120 : 34 02 8E 50 74 CC 50 85 ED 02 EC A1 ED 04 86 16 : 32
5130 : A7 84 AD 9F FB FA C6 10 68 41 69 C4 49 68 45 69 : 77
5140 : 44 49 68 49 69 48 49 68 4D 69 4C 49 68 C8 11 69 : 95
5150 : C8 10 49 68 C8 15 69 C8 14 49 68 C8 19 69 C8 18 : 86
5160 : 49 68 C8 1D 69 C8 1C 49 A7 80 5A 26 C8 E7 84 8E : 97
5170 : 50 AB AD 9F FB FA 6A E4 26 A8 35 82 00 00 00 00 : 0F
5180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

SAVEM "L11-13M",&H5000,&H517B,&H5000

282

11-9 外字フォントの作成, 登録

JIS 第一水準漢字以外の漢字(外字)や, 簡単なグラフィックパターンを使用したい場合があります。そういうときのためにパターンデータの作成, 変更, 保存を手軽にできるようにしたプログラムを作ってみました(リスト 11-5)。

このプログラムでは, 16×16 ドットのパターンデータを最大 500 個まで作成できます。

使い方は, 次のとおりです。

PF1……編集(Edit)

- ① 変更したいパターンデータ No(0~499)を指定します。
- ② パターンデータが表示されたら, カーソルを変更したいドット位置へ移動させます。カーソルの移動は, カーソルキー(←, →, ↑, ↓)を使います。
- ③ スペースキーを押すと, カーソルがあるドットデータが反転します。
- ④ 作成, 変更が終わったら, リターンキー[Enter]を押します。そして作成, 変更したパターンデータを登録するパターンデータ No(0~499)を指定します。

PF2……表示(Display)

- ① 表示したいパターンデータ No(0~499)を指定します。
- ② 指定したパターンデータ以降の 50 個のパターンが表示されます。

PF3……ロード(Load)

- ① ロードしたいパターンデータが記録されているファイル名を指定します。

PF4……セーブ(Save)

- ① パターンデータをセーブするファイル名を指定します。
- ② セーブしたいパターンデータの開始 No(0~499), 終了 No を指定します。

PF5……終了(End)

リスト 11-15 外字フォント作成プログラム

```

10 '*****
20 '*      GAIJI PHONT GENERATE      *
30 '*      ( LIST 11-15 )    V3.0/V3.3  *
32 '*****
40 CLEAR 100,&H2C00:GOSUB 1600:WIDTH 40,25:CONSOLE 0,25,0,0:DEFINT A-Z
:DIM DT(15,15)
50 ON ERROR GOTO 1190
60 ON KEY(1) GOSUB 160:ON KEY(2) GOSUB 270:ON KEY(3) GOSUB 380:ON KEY(
4) GOSUB 470:ON KEY(5) GOSUB 590:ON KEY(10) GOSUB 590
70 GOSUB 640
80 COLOR 2:RESTORE 670
90 FOR I=0 TO 4
100     LINE (I*128,184)-(I*128+112,199),PSET,1,BF

```

```

110 READ A1$,A2$,A3$:PRINT@ (I*128+24,184).VAL("&H"+A1$),VAL("&H"+
A2$),VAL("&H"+A3$)
120 NEXT
130 COLOR 7
140 GOTO 140
150 END
160 '
170 ' PF1 << EDIT >>
180 '
190 GOSUB 660:LINE (0,32)-(639,167),PSET,0,BF
200 LOCATE 0,0:INPUT"EDIT CHR.NO = ";CHRNO
210 IF CHRNO<0 OR CHRNO>499 THEN BEEP:LOCATE 0,0:PRINT SPC(30):GOTO 20
0
220 GOSUB 680:GOSUB 860
230 LOCATE 0,1:INPUT"MOVE CHR.NO = ";CHRNO
240 IF CHRNO<0 OR CHRNO>499 THEN BEEP:LOCATE 0,1:PRINT SPC(30):GOTO 23
0
250 POKE &H2C07,INT(CHRNO/256):POKE &H2C08,CHRNO MOD 256:EXEC &H2C02
260 LOCATE 0,0:PRINT SPC(30):LOCATE 0,1:PRINT SPC(30):GOSUB 630:GOSUB
640:RETURN
270 '
280 ' PF2 << DISPLAY >>
290 '
300 GOSUB 660:LINE (0,32)-(639,167),PSET,0,BF
310 LOCATE 0,0:INPUT"DISPLAY CHR.NO = ";CHRNO
320 IF CHRNO<0 OR CHRNO>499 THEN BEEP:LOCATE 0,0:PRINT SPC(30):GOTO 31
0
330 CHRNO=INT(CHRNO/10)*10:IF CHRNO>450 THEN CHRNO=450
340 POKE &H2C07,INT(CHRNO/256):POKE &H2C08,CHRNO MOD 256:EXEC &H2C04
350 COLOR 5:FOR I=0 TO 9:LOCATE I*2+3,4:PRINT I:NEXT
360 FOR I=0 TO 4:LOCATE 0,I*3+5:PRINT USING "###":CHRNO+I*10:NEXT
370 LOCATE 0,0:PRINT SPC(30):GOSUB 630:GOSUB 640:RETURN
380 '
390 ' PF3 << LOAD >>
400 '
410 GOSUB 660:LINE (0,32)-(639,167),PSET,0,BF
420 LOCATE 0,0:INPUT"LOAD FILE NAME = ";FLN$
430 'LOCATE 0,1:INPUT"START CHR.NO = ";CHRNO
440 'IF CHRNO<0 OR CHRNO>499 THEN BEEP:LOCATE 0,1:PRINT SPC(30):GOTO 4
20
450 LOADM FLN$
460 LOCATE 0,0:PRINT SPC(30):LOCATE 0,1:PRINT SPC(30):GOSUB 630:GOSUB
640:RETURN
470 '
480 ' PF4 << SAVE >>
490 '
500 GOSUB 660
510 LOCATE 0,0:INPUT"SAVE FILE NAME = ";FLN$
520 LOCATE 0,1:INPUT"START CHR.NO = ";CHRNO
530 IF CHRNO<0 OR CHRNO>499 THEN BEEP:LOCATE 0,1:PRINT SPC(30):GOTO 52
0
540 LOCATE 0,2:INPUT"END CHR.NO = ";CHRNOE
550 IF CHRNOE<0 OR CHRNOE>499 THEN BEEP:LOCATE 0,2:PRINT SPC(30):GOTO
540
560 IF CHRNO>=CHRNOE THEN BEEP:LOCATE 0,1:PRINT SPC(30):LOCATE 0,2:PRI
NT SPC(30):GOTO 520
570 SAVEM FLN$,&H3000+CHRNO*32,&H3000+CHRNOE*32+31,&H3000
580 LOCATE 0,0:PRINT SPC(30):LOCATE 0,1:PRINT SPC(30):LOCATE 0,2:PRINT
SPC(30):GOSUB 630:GOSUB 640:RETURN
590 '
600 ' PFS << END >>
610 '
620 GOSUB 630:COLOR 7:WIDTH 80:END
630 KEY(1) OFF:KEY(2) OFF:KEY(3) OFF:KEY(4) OFF:KEY(5) OFF:KEY(10) OFF
:RETURN
640 COLOR 7:LOCATE 0,0:PRINT "SELECT PF1 <---> PF2":SPC(20)
650 KEY(1) ON:KEY(2) ON:KEY(3) ON:KEY(4) ON:KEY(5) ON:KEY(10) ON:RETUR
N

```

```

660 KEY(1) STOP:KEY(2) STOP:KEY(3) STOP:KEY(4) STOP:RETURN
670 DATA 4A54,2340,3D38,493D,2340,3C28,256D,213C,2549,253B,213C,2556,3
D2A,2340,4E3B
680 '
690 '   DISPLAY PATTERN
700 '
710 POKE &H2C07,INT(CHRNO/256):POKE &H2C08,CHRNO MOD 256:EXEC &H2C00:A
DR=&H2C1F
720 FOR Y=0 TO 15
730   YY=Y*8+32:W=0:POKE VARPTR(W),PEEK(ADR):POKE VARPTR(W)+1,PEEK(A
DR+1)
740   LOCATE 20,Y+4:PRINT RIGHT$("0000"+HEX$(W),4)
750   FOR X=15 TO 0 STEP -1
760     XX=X*16+32
770     IF W AND &H01 THEN CC=2:DT(X,Y)=1 ELSE CC=7:DT(X,Y)=0
780     LINE (XX,YY)-(XX+12,YY+6),PSET,CC,BF
790     W=INT(W/2)
800   NEXT
810   ADR=ADR+2
820 NEXT
830 LINE (434,32)-(446,38),PSET,2,BF:LINE (434,40)-(446,46),PSET,7,BF:
COLOR 1:LOCATE 27,6:PRINT CHR$(&HFE)
840 COLOR 7:LOCATE 30,4:PRINT "O N":LOCATE 30,5:PRINT "OFF":LOCATE 30,
6:PRINT "CURSOR"
850 RETURN
860 '
870 '   EDIT PATTERN
880 '
890 X=0:Y=0:COLOR 1
900 LOCATE X+2,Y+4:PRINT CHR$(&HFE)
910 A$=INKEY$:IF A$="" THEN 910
920 IF A$=CHR$(&H0D) THEN 1090
930 IF A$=" " THEN GOSUB 1040:GOSUB 1050:FOR T=0 TO 300:NEXT
940 IF A$=CHR$(&H1C) THEN GOSUB 1050:X=X+1
950 IF A$=CHR$(&H1D) THEN GOSUB 1050:X=X-1
960 IF A$=CHR$(&H1E) THEN GOSUB 1050:Y=Y-1
970 IF A$=CHR$(&H1F) THEN GOSUB 1050:Y=Y+1
980 GOSUB 990:GOTO 900
990 IF X<0 THEN X=15:Y=Y-1
1000 IF X>15 THEN X=0:Y=Y+1
1010 IF Y<0 THEN Y=15
1020 IF Y>15 THEN Y=0
1030 RETURN
1040 DT(X,Y)=DT(X,Y) XOR &H01:RETURN
1050 COLOR 0:LOCATE X+2,Y+4:PRINT CHR$(&HFE):COLOR 1
1060 IF DT(X,Y)=1 THEN CC=2 ELSE CC=7
1070 LINE (X*16+32,Y*8+32)-(X*16+44,Y*8+38),PSET,CC,BF
1080 RETURN
1090 GOSUB 1050:COLOR 7:ADR=&H2C1F
1100 FOR Y=0 TO 15
1110   W1=0:W2=0
1120   FOR X=0 TO 7
1130     W1=W1*2+DT(X,Y):W2=W2*2+DT(X+8,Y)
1140   NEXT
1150   POKE ADR,W1:POKE ADR+1,W2:ADR=ADR+2
1160   LOCATE 20,Y+4:PRINT RIGHT$("00"+HEX$(W1),2):RIGHT$("00"+HEX$(
W2),2)
1170 NEXT
1180 RETURN
1190 IF ERL=570 AND ERR=64 THEN KILL FLN$:RESUME
1200 COLOR 2:PRINT"ERROR !!   ERROR !!"
1210 PRINT "ERL=";ERL;" ERR=";ERR:STOP
1600 RESTORE 2000:ADR=&H2C00
1610 FOR I=0 TO &HE4:READ A$:POKE ADR+I,VAL("&H"+A$):NEXT
1620 RETURN
2000 DATA 20,3D,20,4C,16,00,5A,00,00,10,00,2C,11,00,2E,00
2010 DATA 00,00,00,1C,00,00,00,00,00,0F,00,0F,02,00,20,00
2020 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

```

```
2030 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,17
2040 DATA 00,90,10,8E,2C,1F,C6,20,A6,80,A7,A0,5A,26,F9,39
2050 DATA 17,00,7F,10,8E,2C,1F,C6,20,A6,A0,A7,80,5A,26,F9
2060 DATA 39,CC,00,40,FD,2C,14,C3,00,0F,FD,2C,18,CC,00,28
2070 DATA FD,2C,16,C3,00,0F,FD,2C,1A,C6,05,34,04,C6,0A,34
2080 DATA 04,17,FF,7C,8E,2C,09,CC,07,05,ED,88,13,AD,9F,FB
2090 DATA FA,CC,02,00,ED,88,13,AD,9F,FB,FA,EC,08,C3,00,20
2100 DATA ED,08,C3,00,0F,ED,0F,FC,2C,07,C3,00,01,FD,2C,07
2110 DATA 35,04,5A,26,CA,CC,00,40,ED,08,C3,00,0F,ED,0F,EC
2120 DATA 0D,C3,00,18,ED,0D,C3,00,0F,ED,88,11,35,04,5A,26
2130 DATA AA,39,8E,30,00,FC,2C,07,58,49,58,49,58,49,58,49
2140 DATA 58,49,30,8B,39
```

このプログラムで作成されるパターンデータの形式は、漢字フォントパターンと同様です。ですから作成した外字フォントやグラフィックパターンを表示するには、漢字表示プログラムを一部手直しすればできます。

リスト 11-16(8色モード用)、リスト 11-17(4096色モード用)にそのサンプルプログラムを示します。図 11-11 に示すようなパラメータをセットして、コールします。またこの表示用マシン語サブルーチンは、リロケータブルに作られていますので、ADR の値を変えれば任意のアドレスに配置することができます。

なお外字フォント作成プログラムが作成するパターンデータは、\$3000 番地より 32 バイトづつ格納されています。ですから、表示したいパターンデータの格納されているアドレスを計算して、適当なアドレスにセットして利用してください。

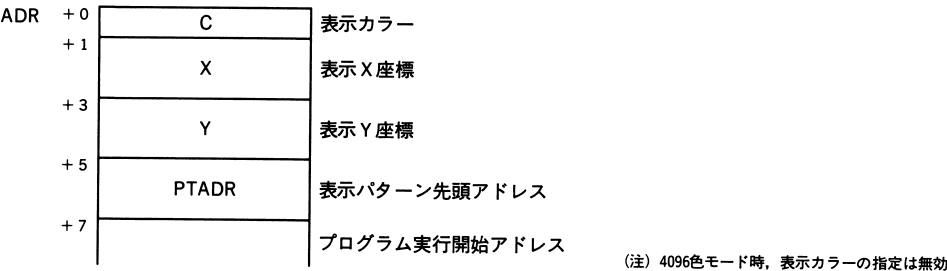


図11-11 外字フォント表示プログラムのパラメータ

リスト 11-16 外字フォント表示(8色モード用)

```
10 *****
20 '*      GAIJI PHONT DISPLAY (8 COLOR MODE) *
30 '*      ( LIST 11-16 )    V3.0/V3.3 8 COLOR *
32 *****
40 CLEAR ,&H5000:ADR=&H5000:CLS
50 FOR I=0 TO &H52:READ A$:POKE ADR+I,VAL("&H"+A$):NEXT
52 FOR I=0 TO 63:READ A$:POKE &H6000+I,VAL("&H"+A$):NEXT
60 C=4:X=50:Y=50:PTADR=&H6000
62 FOR I=0 TO 1:GOSUB 140:X=X+24:PTADR=PTADR+32:NEXT
70 END
```

```

80 DATA 07.00.00.00.00.00.00.30.8D.00.32.31.8C.F2.33.08
90 DATA EF.02.EC.21.ED.08.C3.00.0F.ED.0F.EC.23.ED.00.C3
100 DATA 00.0F.ED.88.11.A6.A4.A7.88.13.EE.25.31.88.16.C6
110 DATA 20.A6.C0.A7.A0.5A.26.F9.AD.9F.FB.FA.39.10.00.00
120 DATA 00.00.2E.00.00.00.00.1C.00.00.00.00.0F.00.0F
130 DATA 07.00.20
140 POKE ADR,C:POKE ADR+1,X*256:POKE ADR+2,X MOD 256:POKE ADR+3,Y*256:
POKE ADR+4,Y MOD 256:POKE ADR+5,PTADR*256:POKE ADR+6,PTADR MOD 256
150 EXEC ADR+7
160 RETURN
200 DATA 00.7F.01.F7.03.C7.07.87.0F.07.1E.07.1E.07.3C.07
210 DATA 3C.07.7F.E7.7F.E7.78.07.F0.07.F0.07.F0.07.F0.07
220 DATA E0.0F.E0.0F.E0.0F.E0.0F.E0.1E.E0.1E.E0.1E.E0.3C
230 DATA E0.3C.E0.78.E0.78.E0.F0.E1.E0.E3.C0.EF.80.FC.00

```

リスト 11-17 外字フォント表示(4096 色モード用)

```

10 '*****
20 '* GAIJI PHONT DISPLAY (4096 COLOR) *
30 '* ( LIST 11-17 ) V3.3 4096 COLOR *
32 '*****
40 CLEAR ,&H5000:ADR=&H5000:SCREEN@ 1:PALETTE 1,[240.0.0]:CLS
50 FOR I=0 TO &HAA:READ A$:POKE ADR+I,VAL("&H"+A$):NEXT
52 FOR I=0 TO 63:READ A$:POKE &H6000+I,VAL("&H"+A$):NEXT
60 X=48:Y=50:PTADR=&H6000
62 FOR I=0 TO 1:GOSUB 190:X=X+24:PTADR=PTADR+32:NEXT
70 END
80 DATA 01.00.00.00.00.00.00.86.FD.05.2B.FB.1A.50.86.80
90 DATA 87.FD.05.86.FD.05.2A.FB.8E.90.00.31.8C.E2.EC.21
100 DATA 44.56.44.56.44.56.44.56.30.8B.EC.23.86.28.3D.30.8B.86
110 DATA 1D.87.FD.8B.7F.B4.10.7F.B4.30.8D.42.86.60.B7.B4
120 DATA 30.8D.3B.86.12.87.FD.89.4C.B7.FD.8A.EE.25.C6.10
130 DATA 34.04.EC.C1.AA.84.EA.01.ED.84.30.88.28.35.04.5A
140 DATA 26.EE.86.39.B7.FD.89.4C.B7.FD.8A.4C.B7.FD.8B.F6
150 DATA FC.80.CA.80.F7.FC.80.4F.B7.FD.05.1C.AF.39.86.10
160 DATA C6.06.34.16.B7.FD.89.4C.B7.FD.8A.EE.25.C6.10.34
170 DATA 04.EC.C1.43.53.A4.84.E4.01.ED.84.30.88.28.35.04
180 DATA 5A.26.EC.35.16.4C.4C.5A.26.08.39
190 POKE ADR+1,X*256:POKE ADR+2,X MOD 256:POKE ADR+3,Y*256:POKE ADR+4,
Y MOD 256:POKE ADR+5,PTADR*256:POKE ADR+6,PTADR MOD 256
200 EXEC ADR+7
210 RETURN
300 DATA 00.7F.01.F7.03.C7.07.87.0F.07.1E.07.1E.07.3C.07
310 DATA 3C.07.7F.E7.7F.E7.78.07.F0.07.F0.07.F0.07.F0.07
320 DATA E0.0F.E0.0F.E0.0F.E0.0F.E0.1E.E0.1E.E0.1E.E0.3C
330 DATA E0.3C.E0.78.E0.78.E0.F0.E1.E0.E3.C0.EF.80.FC.00

```

11-10 漢字 SYMBOL 文

F-BASIC では画面に漢字を表示するのに、通常 PRINT@文を使用します。しかしこの命令は、出力される文字の大きさが一種類だけなので、見出し等を表示しようとすると困ってしまいます。英数カナ文字には SYMBOL 文という大変便利な命令があるのですが、漢字にはありません。

そこで、英数カナ文字用 SYMBOL 命令とはほぼ同じ機能を持った漢字版 SYMBOL プログラムを考えてみました。リスト 11-18、リスト 11-9 がそのプログラムです。

それでは、簡単に原理を説明しましょう。F-BASIC の PRINT@文では、BIOS を使って漢字フォントをワークエリア上に取り出した後、それをグラフィックパターンとして画面に PUT しています。本プログラムでは、取り出した漢字フォントのドットの有無を調べ、対応する点には、BF オプション付きの LINE 命令でボックスを書いています。したがって、LINE の持つオプション/ファンクションはすべて利用可能となります。

プログラムの使用法は、まず図 11-12 に従って、倍率や、色等のパラメータをワークエリアにセットします。そして整数型変数に JIS コードを代入して、それを引数にして &H5000 (640 ドットのとき)、または &H5003 (320 ドットのとき) を USR 関数でコールしてください。このプログラムでは 1 文字出力すると座標が更新されるので、つづけて漢字を出力する場合には、2 文字目からは漢字コードのみを送ってやれば OK です。

このあたりの詳しいことは、リスト 11-19 のサンプルプログラムを参照してください。リスト 11-19 を実行するには、まずリスト 11-18 を “L11-18M” というファイル名でセーブします。

SAVEM “L11-18M”, &H5000, &H51B3, &H5000 

それから、リスト 11-19 を RUN してください。図 11-13 がその結果です。なおこのプログラムは、F-BASIC V3.0/3.3 のどちらでも使用可能です。

アドレス	内 容	パラメータ
\$5010, 11	x座標(16Bit)	$0 \leq x < 640$ または 320
\$5012, 13	y座標(16Bit)	$0 \leq y < 200$
\$5014	横倍率	1~127
\$5015	縦倍率	1~127
\$5016	640ドットモード COLOR	0~7
\$5017	アングル	0:  1:  2:  3: 
\$5018	ファンクション	0: PSET 1: PRESET 2: OR 3: AND 4: XOR
\$5019	ボックスフラグ	0: ライン  1: ボックス  2: ボックスフル  (通常)
\$501A	320ドットモード COLOR(青)	0~15
\$501B	320ドットモード COLOR(赤)	0~15
\$501C	320ドットモード COLOR(緑)	0~15

図11-12 漢字SYMBOL文パラメータ

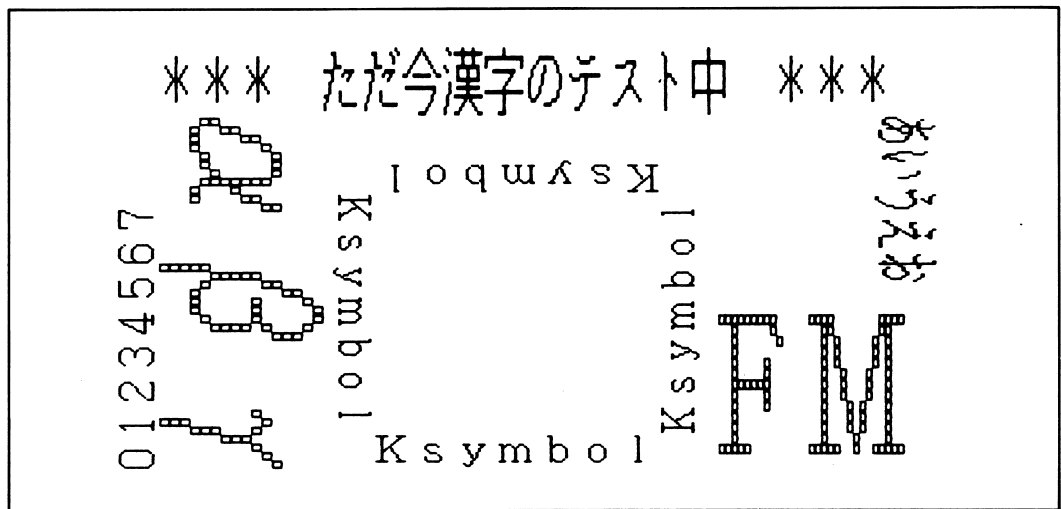


図11-13 漢字SYMBOL文実行例

リスト 11-18 漢字 SYMBOL 文

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 7E 50 80 7E 50 60 00 00 00 00 00 00 00 00 00 00 : 89
5010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5040 : 00 00 00 10 00 50 5F 00 0E 00 00 00 00 15 10 00 : F2
5050 : 50 50 00 10 00 00 15 00 00 00 00 00 00 00 00 00 : D2
5060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 FE 50 1A : 68
5070 : FF 50 60 F6 50 1C F7 50 62 CE 50 4E C6 09 20 0B : 20
5080 : F6 50 16 F7 50 62 CE 50 43 C6 0B 81 02 26 48 10 : 38
5090 : 8E 50 57 A6 C0 A7 A0 5A 26 F9 EC 02 8E 50 1D ED : 31
50A0 : 04 CC 16 00 ED 84 CC 50 23 ED 02 AD 9F FB FA B6 : 7C
50B0 : 50 18 B7 50 63 B6 50 19 B7 50 6C CE 50 23 BE 50 : B3
50C0 : 10 10 BE 50 12 C6 10 B6 50 17 27 0C 4A 27 36 4A : 57
50D0 : 27 62 4A 10 27 00 8D 39 34 14 C6 10 BF 50 64 B6 : 17
50E0 : 50 14 4A 30 86 10 BF 50 66 B6 50 15 4A 17 00 A3 : 08
50F0 : 30 86 5A 26 E7 BF 50 10 B6 50 15 31 A6 35 14 33 : AA

```

```

[cs] : 5C 8D C6 37 A6 B1 A1 B2 53 FB 07 AE 3E 73 4B FE : 8D

```

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 42 5A 26 04 39 34 24 C6 10 BF 50 64 B6 50 15 4A : D5
5110 : 40 30 86 10 BF 50 66 B6 50 14 4A 8D 76 40 31 A6 : F9
5120 : 5A 26 E6 10 BF 50 12 B6 50 15 30 86 35 24 33 42 : 36
5130 : 5A 26 D2 39 34 14 C6 10 BF 50 64 B6 50 14 4A 40 : C0
5140 : 30 86 10 BF 50 66 B6 50 15 4A 40 8D 46 40 30 86 : A9
5150 : 5A 26 E5 BF 50 10 B6 50 15 40 31 A6 35 14 33 42 : 74
5160 : 5A 26 D1 39 34 24 C6 10 BF 50 64 B6 50 15 4A 40 : D0
5170 : 30 86 10 BF 50 66 B6 50 14 4A 8D 17 31 A6 5A 26 : 9A
5180 : E7 10 BF 50 12 B6 50 15 40 30 86 35 24 33 42 5A : 51
5190 : 26 D2 39 31 A6 10 BF 50 6A BF 50 68 68 41 69 C4 : DE
51A0 : 24 07 8E 50 57 AD 9F FB FA BE 50 64 10 BE 50 66 : 97
51B0 : B6 50 14 39 00 00 00 00 00 00 00 00 00 00 00 : 53
51C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00

```

```

[cs] : 31 67 D4 AD 1E 5B FB A2 10 09 B6 2E 49 09 C5 24 : 64

```

SAVEM "L11-18M",&H5000,&H51B3,&H5000

```

10 *****
20 '* KANJI SYMBOL SAMPLE *
30 '* ( LIST 11-19 ) V3.0/V3.3 *
40 '* LIST 11-18 カ ヒツヨウ テ"ス *
50 *****
1010 CLEAR 300,&H5000
1020 LOADM"L11-18M"
1030 CLS
1040 DEF USR=&H5000
1050 READ K$:K%=VAL("&H"+K$)
1060 IF K%>&H2120 THEN DUMMY=USR(K%):GOTO 1050
1070 ON K%'GOTO 1120,1180
1080 BEEP
1090 A$=INKEY$:A$=INPUT$(1)
1100 IF A$="P" OR A$="p" THEN HARDC2
1110 END
1120 READ X,Y
1130 POKE &H5010,X*256
1140 POKE &H5011,X MOD 256
1150 POKE &H5012,0
1160 POKE &H5013,Y
1170 GOTO 1050
1180 READ MULTX,MULTY,COL,ANGLE,FUNC,BOXF
1190 POKE &H5014,MULTX
1200 POKE &H5015,MULTY
1210 POKE &H5016,COL
1220 POKE &H5017,ANGLE
1230 POKE &H5018,FUNC
1240 POKE &H5019,BOXF
1250 GOTO 1050
1260 DATA 1,32,0
1270 DATA 2,2,2,5,0,0,2
1280 DATA 2176,2176,2176,2121,243F,2440,3A23,3441
1290 DATA 387A,244E,2546,2539,2548,4366,2121,2176
1300 DATA 2176,2176
1310 DATA 1,0,199
1320 DATA 2,1,2,3,1,0,2
1330 DATA 2330,2331,2332,2333,2334,2335,2336,2337
1340 DATA 1,200,180
1350 DATA 2,2,1,2,0,0,2
1360 DATA 234B,2373,2379,236D,2362,236F,236C
1370 DATA 2,1,2,5,1,0,2
1380 DATA 234B,2373,2379,236D,2362,236F,236C
1390 DATA 2,2,1,6,2,0,2
1400 DATA 234B,2373,2379,236D,2362,236F,236C
1410 DATA 2,1,2,3,3,0,2
1420 DATA 234B,2373,2379,236D,2362,236F,236C
1430 DATA 1,639,32
1440 DATA 2,1,3,6,3,0,2
1450 DATA 2422,2424,2426,2428,242A
1460 DATA 1,460,120
1470 DATA 2,5,5,2,0,0,1
1480 DATA 2346,234D
1490 DATA 1,160,25
1500 DATA 2,4,8,7,3,0,1
1510 DATA 2641,2642,2643
1520 DATA 0

```

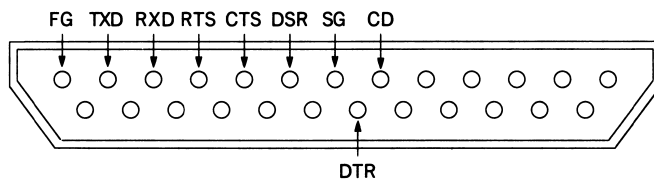
RS-232C とは、通信回線でデータを送受信するモデムとデータ端末装置とを接続するためのシリアルデータの伝送規格のことです。これは、CCITT(国際電信電話諮問委員会)の勧告を受け、アメリカのEIA(Electronic Industries Association)が決定したものです。現在では、モデムに限らずシリアルインターフェースとして広く用いられています。

FM-7 シリーズでは、オプションの RS-232C インターフェースカードを接続すれば、RS-232C インターフェース付きの機器とのデータ交換を行なうことができます。データ交換には、ターミナルモードと入出力モードの2通りの方法があります。ターミナルモードでは、大型計算センターの TSS 端末やオンラインシステムの端末として利用できます。一方、入出力モードでは、制御機器、計測機器、他のパーソナル・コンピュータなどとデータ交換を行なうことができます。

この章では、入出力モードによるデータ交換を主にとりあげ、パーソナル・コンピュータ同士でプログラムやデータの送信、受信を行なってみたいと思います。

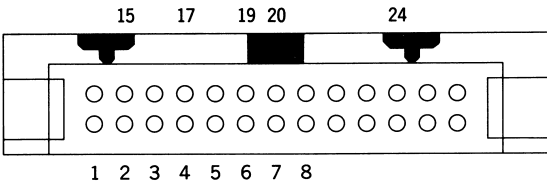
12-1 RS-232C インターフェース用ケーブル

RS-232C インターフェースには、25 ピンのコネクタが規定されており、各ピンには図 12-1 に示すような信号が割り当てられています。FM-7 シリーズの RS-232C インターフェースカード



ピン番号	記号	記号方向	信号の名称
1	FG		保安用接地(Frame Ground)
2	TXD	→ モデム	送信データ(Transmitted Data)
3	RXD	← モデム	受信データ(Recieved Data)
4	RTS	→ モデム	送信要求(Request to Send)
5	CTS	← モデム	送信許可(Cear to Send)
6	DSR	← モデム	データセットレディ(Data Set Ready)
7	SG		信号用接地(Signal Ground)
8	CD	← モデム	受信キャリア検出(Carrier Detector)
20	DTR	→ モデム	端末レディ(Data Terminal Ready)

図12-1 RS-232Cインターフェイスのコネクタ



(注) このRS-232C用フラットケーブルのピン番号の取り方は、通常のフラットケーブルと異なります。

ピン番号	記号	信号の名称
1	FG	保安用接地 (Frame Ground)
2	TXD	送信データ (Transmitted Data)
3	RXD	受信データ (Recieved Data)
4	RTS	送信要求 (Request to Send)
5	CTS	送信許可 (Clear to Send)
6	DSR	データセットレディ (Data Set Ready)
7	SG	信号用接地 (Signal Ground)
8	CD	受信キャリア検出 (Carrier Detector)
15	ST2	Send Data Timing 2
17	RT	Receive Timing
19	GND	Ground
20	DTR	端末レディ (Data Terminal Ready)
24	ST1	Send Data Timing 1

図12-2 RS-232Cインターフェースカードのコネクタ

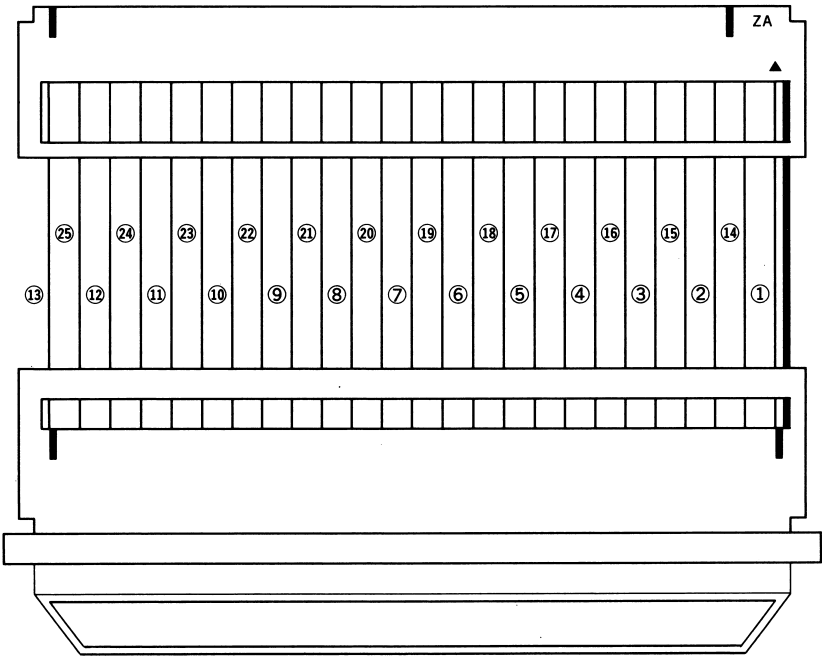


図12-3 RS-232Cケーブルの信号線

には 26 ピンのコネクタが使われており、そのピンの意味は、図 12-2 のようになっています。ですから両者を接続する FM-7 シリーズ用 RS-232C ケーブルは、図 12-3 のような構成となっています。

RS-232C は、もともとコンピュータ(データ端末装置)とモデムを接続するためのものでした(図 12-4)。それでコンピュータ同士を直接接続するには、RS-232C ケーブルの信号線を一部変更する必要があります。これは自分で作成してもよいのですが、FM 同士を接続するには、チェッカーという信号線をクロスする接続器がでていますので、それを利用するのが便利でしょう。

自分で作る場合には、RS-232C 用ケーブルの信号線を、図 12-5 の様に対になる信号線を入れ換えて接続します。ただし FM の場合、8 番ピンの DCD(キャリア検出信号)には対になるピンがなく、SG(7 番ピン)に接続しておきます。これで DCD は常にアクティブとなり、FM 同士が接続されます。対になる信号を入れ換えたことで、各 FM からは、他方が等価的にモデムのように見えるためです。

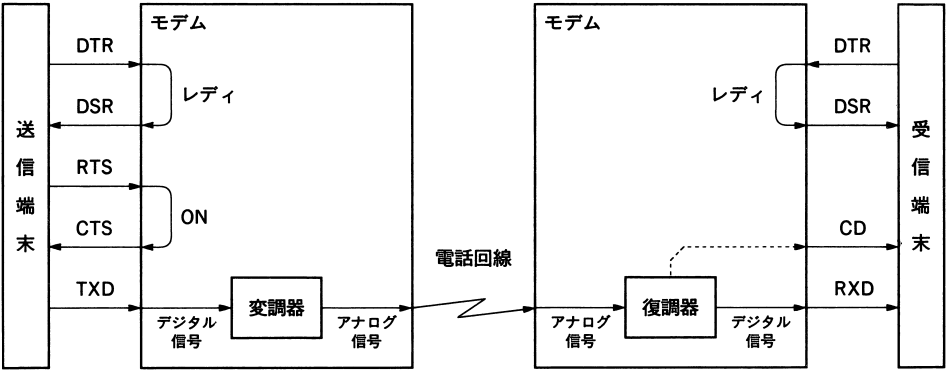


図12-4 モデム接続時の信号の流れ

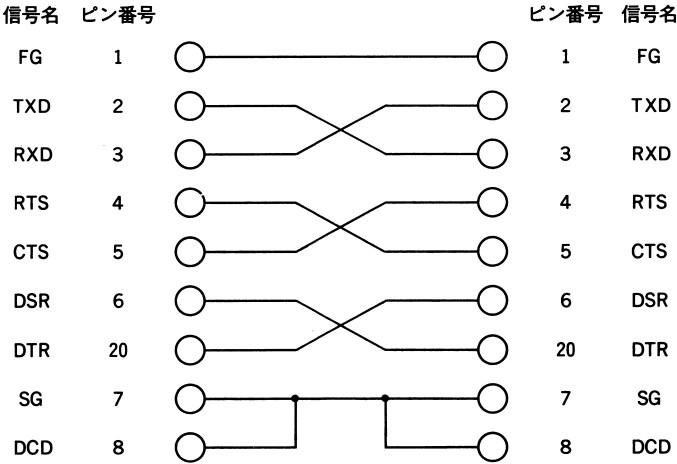


図12-5 専用ケーブル接続図

もし、FM 以外のコンピュータと接続するケーブルを作るときには、FM にないピンが使用されていたり、逆に FM にあるピンが使用されていないかったり、あるいは、DCD に+5V～+15V の電圧をかける必要があるものもあります。ですからコネクタの規格を十分検討してから接続してください。

12-2 通信パラメータ

12-2-1 データのビット長

英数字やカナ文字などを表すのに、情報処理の世界では、7 ビットか 8 ビットの長さの符号(コード)が使われます。

ここでは、広く使われているコードをまとめてみます。

(1) ASCII(アスキー)コード……7 ビット

A から Z までのアルファベットの大文字と小文字、0 から 9 までの数字、+ (プラス)や- (マイナス)などの記号を含む 128 種類の文字を、7 ビットの長さの符号で表わします。

(2) JIS(ANK)コード……8 ビット

ASCII コードにカナ文字を追加したもので、8 ビットの長さの符号でひとつの文字を表わします。パソコンの内部処理でも使用されているコードです。

(3) JIS6220 コード……7 ビット

シフトイン(SI)、シフトアウト(SO)という制御コードを用いることにより、7 ビットの符号でカナ文字も表現できるようにしたコードです。通常の使い方では、7 ビットの ASCII コードと全く同じですが、SI と SO の間にはさまれた文字は、カナ文字とみなす仕組みになっています。

12-2-2 データ伝送速度

データの伝送速度を表わすには、1 秒間に何ビットのデータを送れるかを示すボー(baud)という単位を使います。

たとえば 300 ボーとは、1 秒間に 300 ビットのデータを伝送することを意味します。JIS コードを使用すると 1 文字は 8 ビットで表されるので、37.5 文字となります。実際には制御信号も加わりますから、1 文字につき 10 ビット位となり、1 秒間に送れる文字数は約 30 文字となります。

12-2-3 パリティ・チェック

通信回線を使って送ったデータが、常に正しく伝えられるとは限りません。伝送の途中でデータの一部が誤ったデータに変化した場合、変化したデータを発見するひとつの手段としてパリティ・チェックがあります。

この原理を簡単に説明します。伝送しようとする文字のコードの中に含まれている1(オン)のビットの個数を数え、それが偶数個であるか奇数個であるかを調べます。そして1の個数が偶数のときは、パリティ・ビットの値を0とします。1の個数が奇数のときにはパリティ・ビットの値を1とします。このようにすることにより、8ビットのJISコードと1ビットのパリティ・ビットを合わせた9ビットのデータ中に含まれる1のビットの個数を常に偶数個とすることができます。受信側では、9ビットのデータ中の1のビットの個数が偶数であれば正しく伝送されたとみなし、奇数であれば誤って伝送されたと判断できます。もちろん、ふたつのビットが同時に変化した場合には、そうとは言えないのですが、その確率は非常に小さいと考えられます。

このように、データ中の1のビットの個数を常に偶数個とする方法を偶数パリティ (Even Parity) といい、常に奇数個とする方法を奇数パリティ (Odd Parity) といいます(図12-6)。

F	0 1 0 0	0 1 1 0	3	1
M	0 1 0 0	1 1 0 1	4	0
	JISコード		1のビット の個数	パリティ ビット

図12-6 偶数パリティ

12-2-4 通信モード

電話機では、こちらの話が相手に伝わるのと同時に相手の声も聞くことができます。これは、 $A \rightarrow B$ という通話の道と $B \rightarrow A$ という通話の道が1本ずつあり、二重になっているからです。このように送信用と受信用の回線が二重になっていて、同時に送受信できるシステムを全二重通信と呼んでいます。

これに対して、1本の通信回線を使って、送信と受信を交互に行なう方法もあります。これは、 $A \rightarrow B$ と $B \rightarrow A$ の相互の通信の道があるという意味では二重なのですが、同時には通信できないため半二重通信と呼ばれています。

12-2-5 ストップ・ビット

1本の通信線を使って文字の符号を次から次へと連続的に送る場合、文字の区切りを明確にしないと、情報が正しく伝達されません。そこで、文字の始まりと終わりを識別するための信号(スタート・ビットとストップ・ビットという)を付けて、1文字ずつ、データを送ることにします。

この文字符号の先頭に付けるビットをスタート・ビットと呼び、文字符号の末尾に付けるビットをストップ・ビットと呼びます。スタート・ビットは1ビットと統一されていますが、ストップ・ビットは1, 1.5, 2ビットとまちまちに使われています(図12-7)。

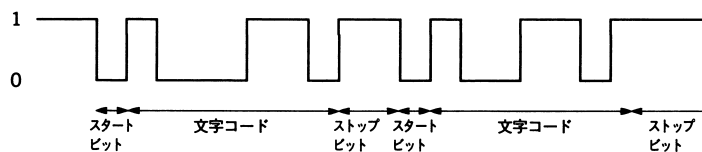


図12-7 スタート・ビットとストップ・ビット

12-2-6 Xパラメータ

XON/XOFF による通信制御バッファのビジー制御を行なうかどうかの設定をするパラメータです。

FM-7 シリーズでは通信回線からデータを受信すると、割り込みが発生してデータが通信制御バッファに貯えられます。そして、INPUT #, LINE INPUT, INPUT\$により通信制御バッファからデータが読み取られます。しかし通信制御バッファの大きさは、127 バイトしかないのに、まだ読み込まれていないデータが 127 バイトあるときにデータを受信すると、“Buffer overflow”のエラーが発生してしまいます。

そこで、通信制御バッファが送られてきたデータでいっぱいになりそうになると、自動的に XOFF を送出して送信側にデータ送出の一時停止を要求します。そして未処理のデータが読み取られて、通信制御バッファの中のデータが少なくなると、自動的に XON を送出して、送信側にデータ送出の再開を要求します。送信側では、受信側から XOFF が送られてきたらデータの送出を中止し、その後、XON が送られてきたらデータの送出を再開します。

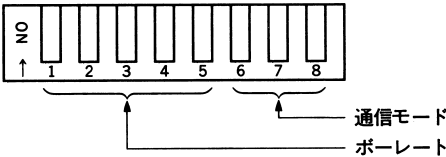
このような制御を、通信制御バッファのビジー制御と呼びます。

12-3 通信パラメータの設定

FM-7 シリーズにおける通信パラメータの設定は、OPEN 命令(入出力モード時)か、TERM 命令(ターミナルモード時)にて行ないます。以下、そのパラメータの指定方法を説明します。

(1) クロック指定……(c)

FM-7 シリーズでのデータ伝送速度は、RS-232C インタフェースカード上のディップスイッチによるボーレートの設定とクロック指定により決定されます。ボーレートは、図 12-8 に示すように 300, 600, 1200, 2400, 4800 の選択をすることができます。そして、クロック指定において S (slow クロック)を選択した場合には、データ伝送速度はディップスイッチで設定したボーレートになります。一方、F (fast クロック)を選択した場合には、データ伝送速度はディップスイッチで設定したボーレートの 4 倍となります。



[ボーレート]

S W 1	S W 2	S W 3	S W 4	S W 5	Slowモード	Fastモード	同期式
ON	OFF	OFF	OFF	OFF	300ボー	1200ボー	300ボー
OFF	ON	OFF	OFF	OFF	600ボー	2400ボー	600ボー
OFF	OFF	ON	OFF	OFF	1200ボー	4800ボー	1200ボー
OFF	OFF	OFF	ON	OFF	2400ボー	9600ボー	2400ボー
OFF	OFF	OFF	OFF	ON	4800ボー	19200ボー	4800ボー

[通信モード]

S W 6	S W 7	S W 8	使用するクロック	使用方法
ON	OFF	OFF	カード上のクロック	送受信ともに、同一のクロックを使用する場合
ON	OFF	ON	ST2端子より入力	
ON	ON	OFF	RT端子より入力	
OFF	ON	OFF	カード上のクロックを送信とし、RT端子よりクロックを入力する	送信、受信クロックを分離して使用する場合
OFF	ON	ON	ST2端子より送信用のクロックを入力して受信用のクロックをRT端子より入力する	

図12-8 ディップスイッチの設定

(2) データのビット長……(b)

8(8ビット/文字)または、7(7ビット/文字)を指定します。7ビット長を指定したときには、カナの前後にSI(シフトイン), SO(シフトアウト)コードが自動的に付けられ、英数字とカナの区別がされます。なお、FM-7シリーズでは、SIコードとして\$0F, SOコードとして\$0Eが使用されています。

(3) パリティ・チェック……(p)

N(パリティなし), O(奇数パリティ)または、E(偶数パリティ)のいずれかを指定します。

(4) ストップ・ビット……(s)

ストップ・ビットのビット数を、2(2ビット)または、1(1ビット)で指定します。FM-7シリーズでは、ストップ・ビットの1.5ビットは指定できません。ですから、他のパソコンと接続するときには注意する必要があります。

(5) 通信モード……(m)

F(全二重通信)または、H(半二重通信)を指定します。

(6) オート・LF……(d)

CRコードを受信したとき、自動的にLF(ライン・フィード)コードを出力して、改行を行なうかどうかの指定です。A(オート・LFを行なう)または、N(オート・LFをしない)を指定します。

(7) コントロールコード……(t)

データ受信時のコントロールコード(\$00~\$1F)を、F-BASIC オーダー(図 11-7)にするか、ADM-3A オーダー(図 11-8)にするかの設定を行ないます。F(F-BASIC オーダー)または、A(ADM-3A オーダー)を指定します。

(8) Xパラメータ……(x)

XON/XOFFによる通信制御バッファのビジー制御を行なうかどうかの設定を行ないます。Xを指定すると、ビジー制御が行なわれます。指定を省略すると行なわれません。なおFM-7シリーズでは、XONとしてDC1コード(\$11)、XOFFとしてDC3コード(\$13)を使用しています。

通信パラメータは、入出力モード時には(1)~(4)のパラメータを、(cbps)の形式でOPEN命令のオプションに指定します。

[例] OPEN "O",#1,"COM0:(S8N2)"

ターミナルモード時には、(1)~(8)のパラメータを"cbpsmatx"の形式で、TERM命令のオペランドに指定します。ただし、F-BASIC V3.0では、"tx"は指定できません。

[例] TERM "S8N2FA" ……(V3.0)

TERM "S8N2FNFX" ……(V3.3)

FM-7シリーズ以外のパソコンによっては、これ以外の通信パラメータを持つものがあります。それらについては、両方のパソコンの取り扱い説明書を注意深く読んでください。

たくさんのパラメータがあり、その設定にとまどわれる方もあるかと思います。そういうときには、通信相手のパラメータを聞き、それと同じにすればよいのです。しっかりした規格が決まっていて、電源を入れるだけでパソコン通信ができるようになればよいとも思います。しかしパソコン通信は、まだまだこれからの技術なのです。多くの人のいろいろな夢がぶつかりあって、ひとつにまとめることができないのでしょう。是非あなたも、このパソコン通信であなたの夢を咲かせてみてください。

12-4 データの転送

では、2 台の FM77AV でデータを転送してみます。このとき、双方のボーレートを合わせるのを忘れないでください。とりあえず本章では、ディップスイッチを次のように設定することになります。

SW1 (ON)	SW5 (OFF)
SW2 (OFF)	SW6 (ON)
SW3 (OFF)	SW7 (OFF)
SW4 (OFF)	SW8 (OFF)

RS-232C にデータを出力するには、他のシーケンシャルファイルと同様、リスト 12-1、リスト 12-2 のように行ないます。ここで注意すべき点は、受信側を、INPUT モードと OUTPUT モードの両方のモードで OPEN しておく必要があることです。INPUT モードだけの OPEN では、データの受信を行なうことができません。

リスト 12-1 データの送信

```

100 '*****
110 '*      DATA SEND      *
120 '*      ( LIST 12-1-1 )   V3.0/V3.3      *
122 '*****
130 A$="0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ"
140 OPEN "Q",#1,"COMO:(F8N2)"
160 FOR I=0 TO 50
170   PRINT #1,A$
190 NEXT
200 PRINT #1,"END"
210 CLOSE #1,#2
220 END

```

リスト 12-2 データの受信

```

100 '*****
110 '*      DATA RECEUVE    *
120 '*      ( LIST 12-1-2 )   V3.0/V3.3      *
130 '*****
140 OPEN "I",#1,"COMO:(F8N2)"
150 OPEN "Q",#2,"COMO:(F8N2)"
160   LINE INPUT #1,A$
170   IF A$="END" THEN 200
180   PRINT A$
190   GOTO 160
200 CLOSE #1,#2
210 END

```

さて、このように基本的にはディスクのシーケンシャルファイルと同様なのですが、ひとつだけ異なっていることがあります。それは、バッファの限界です。

RS-232Cの入力は、内部的には割り込みで処理されており、データを1文字受けるたびにメインRAM上のファイルバッファに蓄え、INPUT #文でこのバッファからデータを取り出すわけです。したがって、INPUT #によるデータの取り出し速度より、受信したデータを蓄積していく速度の方が速ければ、バッファ内のデータはどんどん増え、ついにはバッファからあふれてしまいます。特に、ボーレートが速く、データが多いときは注意しなければなりません。

この現象を防ぐには、受信側よりアクノリッジを返す方法があります。そして送信側は、データを送ったら、相手が受け取ったことを示すアクノリッジを出すまで、次のデータの送出を待っているのです。ただし、アクノリッジの送出を待っているため、その分出力側の処理速度が落ちてしまいます。アクノリッジを返してデータ転送を行なう例を、リスト12-3、リスト12-4に示します。

リスト12-3 アクノリッジを返してデータ送信

```

100 '*****
110 '* DATA SEND WITH ACK *
120 '* ( LIST 12-2-1 ) V3.0/V3.3 *
130 '*****
140 A$="0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ"
150 OPEN "O",#1,"COMO:(F8N2)"
160 OPEN "I",#2,"COMO:(F8N2)"
170 FOR I=0 TO 50
180 PRINT #1,A$
190 INPUT #2,AC$
200 NEXT
210 PRINT #1,"END"
220 CLOSE #1,#2
230 END

```

リスト12-4 アクノリッジを返してデータ受信

```

100 '*****
110 '* DATA RECEIVE WITH ACK *
120 '* ( LIST 12-2-2 ) V3.0/V3.3 *
122 '*****
130 OPEN "I",#1,"COMO:(F8N2)"
140 OPEN "O",#2,"COMO:(F8N2)"
150 ACK$=CHR$(6)
160 LINE INPUT #1,A$
170 IF A$="END" THEN 210
180 PRINT #2,ACK$
190 PRINT A$
200 GOTO 160
210 CLOSE #1,#2
220 END

```

12-5 プログラムの転送

次は、BASIC プログラムの転送を考えてみます。プログラムの転送をするには、まず BASIC プログラムをアスキーセーブします。

SAVE "ファイル名",A

そして、そのアスキーセーブされたプログラムファイルをシーケンシャルファイルと見なして読み込み、送出します。受信側では、送られてきたデータをシーケンシャルファイルとして書き込みます。これでプログラムの転送が終了します。書き込まれたシーケンシャルファイルをロードすると、確かに BASIC プログラムが現れます。もちろん実行もできます。

プログラム転送を行なうためのプログラムをリスト 12-5、リスト 12-6 に示します。このプログラムでは、受信したデータを送信側に送り返し、送信側にも表示させています。この処理をエ

リスト 12-5 プログラムの送信

```

100 '*****
110 '*      PROGRAM SEND      *
120 '*      ( LIST 12-3-1 )   V3.0/V3.3 *
130 '*      TESTPRO カ" ヒツヨウ テ"ス *
140 '*****
150 CLS
160 OPEN "O",#1,"COMO:(S8N2)"
170 OPEN "I",#2,"COMO:(S8N2)"
180 OPEN "I",#3,"TESTPRO"
190   LINE INPUT #3,A$
200   IF EOF(3) THEN 250
210   PRINT #1,A$
220   INPUT #2,B$
230   PRINT B$
240   GOTO 190
250 PRINT #1,"PROGRAM END"
260 CLOSE #1,#2,#3
270 END

```

リスト 12-6 プログラムの受信

```

100 '*****
110 '*      PROGRAM RECEIVE   *
120 '*      ( LIST 12-3-2 )   V3.0/V3.3 *
130 '*      TESTPRO カ" ツクラレマス *
140 '*****
150 OPEN "I",#1,"COMO:(S8N2)"
160 OPEN "O",#2,"COMO:(S8N2)"
170 OPEN "O",#3,"TESTPRO"
180   LINE INPUT #1,A$
190   IF A$="PROGRAM END" THEN 240
200   PRINT #2,A$
210   PRINT A$
220   PRINT #3,A$
230   GOTO 180
240 CLOSE #1,#2,#3
250 END

```

コーバックといいます。エコーバックによって、転送データの確認をすることができます。なお、このプログラムは一般のシーケンシャルファイルの転送にも利用できます。

12-6 ターミナルモード

F-BASIC では、TERM 文にて動作モードをターミナルモードにすることができます。このターミナルモードというのは、BASIC の動作モードから離れて通信回線におけるターミナル(端末)となるものです。

このターミナルに対してホストと呼ばれるものがあるのですが、その違いは今は深く考えないで、まず2台のFM77AVを両方ともターミナルモードにして接続してみます。両方のボーレートに合わせて、両側で TERM  と入力します。そして、キーを何か入力してみてください。入力したキーコードが、相手側の CRT 上に表示されますね。

ターミナルモードのとき、キーボードから文字を入力すると相手側に伝達されます。またデータを受信すると、そのデータを CRT に表示します。しかしターミナルモードでは、それ以上のことは行なえません。ですから、ターミナルモードになっている2台のパソコンで交信をしても、あまり意味のあることはできません。

それでは、リスト 12-7 をどちらかのパソコンで RUN させ、もう一方はターミナルモードにしてみてください。そしてターミナルモードの方のパソコンに、メッセージに従って名前と生年月日(西暦)を入力してみてください。すると、あなたの誕生日からの日数と 10000 日目の年月日が計算されて表示されます。ターミナルモードでない方のパソコンでは、誕生日からの日数と 10000 日目の年月日を計算して、ターミナルモードのパソコンへ結果を送ります。

この例のように、単に文字や数字を送信、受信できるだけのコンピュータをターミナルと呼び、受信したデータに対して何らかの処理をする側をホストと呼びます。一般的に、ホスト側のコンピュータのプログラムは、データチェック、エラー処理、回線制御処理等を含み、かなり複雑なものとなります。そして通常ホスト側は汎用コンピュータが、ターミナル側はデータ端末、インテリジェント端末、パソコン端末等が使われます。しかし小規模のシステムならば、パソコンをホストとして使用してもかまいません。

ホスト側よりオーダーコード(F-BASIC オーダーまたは、ADM-3A オーダー)を送出することにより、ターミナル側の表示をコントロールできます(図 12-9、図 12-10)。オーダー動作の詳細については、F-BASIC 文法書を参考にしてもらうことにして、よく使用されるオーダー動作の使用例(F-BASIC オーダー)を次に示します。

```
PRINT #1,CHR$(&H07)      ..... ブザーを鳴らす
PRINT #1,CHR$(&H0A)      ..... 改行する
PRINT #1,CHR$(&H0C)      ..... 画面消去
```

リスト 12-7 ターミナル制御プログラム例

```

10 '*****
20 '*      TERMINAL CONTROL PROGRAM      *
30 '*      ( LIST 12-4 )      V3.0/V3.3  *
40 '*****
100 DIM DAY(12):FOR I=1 TO 12:READ DAY(I):NEXT
110 OPEN "I",#1,"COM0:(S8N2)"
120 OPEN "O",#2,"COM0:(S8N2)"
130 D$=DATE$:D2$=DATE$
140 IF VAL(LEFT$(D2$,2))>65 THEN 160
150 YY$=STR$(VAL(LEFT$(D2$,2))+25):D2$=RIGHT$(YY$,2)+RIGHT$(D2$,6)
160 DATE$=D2$:DT2=DATE:DATE$=D$
170 PRINT #2,CHR$(&H0C) '-----> CLS
180 PRINT #2,CHR$(&H0B); '-----> CURSOR HOME POSITION
190 PRINT #2,"アタ ノ オマエ ハ ?";
200 GOSUB 1000 '-----> NAME INPUT
210 NM$=INDT$
220 PRINT #2,CHR$(&H0D):PRINT #2,NM$;" サン ノ セイネンカ"ッヒ° ハ ? 19";
230 GOSUB 1100 '-----> BIRTHDAY INPUT
240 D1$=INDT$
250 IF VAL(MID$(D1$,4,2))>12 THEN PRINT #2,CHR$(&H07):GOTO 220
260 IF VAL(MID$(D1$,7,2))>31 THEN PRINT #2,CHR$(&H07):GOTO 220
270 IF VAL(LEFT$(D1$,2))>VAL(LEFT$(D2$,2)) THEN PRINT #2,CHR$(&H07):G
OTO 220
280 GOSUB 1200 '-----> DAY CALCULATION
300 PRINT #2,CHR$(&H0A):PRINT #2,"アタ ノ イマデ"ノ シンセイハ 19";D1$;" ---> 1
9";D2$:DAY:CHR$(&HF4)+" テ"入。"
330 PRINT #2,"アタ ノ 10000 ";CHR$(&HF4);" ノ キネヒ° ハ";
340 PRINT #2,STR$(Y+1900):CHR$(&HF2):STR$(M):CHR$(&HF3):STR$(WK):CHR$(&HF4);
350 IF DAY>10000 THEN PRINT #2," テ"シタ。" ELSE PRINT #2," テ"入。"
360 PRINT #2,CHR$(&H0D)
370 GOTO 190
400 DATA 31,28,31,30,31,30,31,31,30,31,30,31
999 '*****
1000 INDT$="" '-----> NAME INPUT
1010 A$=INPUT$(1,#1) '-----> 1 CHAR. INPUT
1020 IF A$=CHR$(&H0D) THEN 1070 '----> RETURN KEY
1022 IF A$=CHR$(&H0B) OR A$=CHR$(&H7F) THEN GOSUB 1500:GOTO 1010
1030 IF A$<CHR$(&H20) THEN PRINT #2,CHR$(&H07):GOTO 1010
1040 INDT$=INDT$+A$
1050 PRINT #2,A$; '-----> INPUT CHAR. DISPLAY
1060 IF LEN(INDT$)<11 THEN 1010
1070 RETURN
1099 '*****
1100 INDT$="" '-----> BIRTH-DAY INPUT
1110 A$=INPUT$(1,#1) '-----> 1 CHAR. INPUT
1112 IF A$=CHR$(&H0B) OR A$=CHR$(&H7F) THEN GOSUB 1500:GOTO 1110
1120 IF A$<CHR$(&H30) OR A$>CHR$(&H39) THEN PRINT #2,CHR$(&H07):GOTO
1110
1130 INDT$=INDT$+A$
1132 PRINT #2,A$; '-----> INPUT CHAR. DISPLAY
1140 IF LEN(INDT$)=2 OR LEN(INDT$)=5 THEN A$="/":GOTO 1130
1150 IF LEN(INDT$)<8 THEN 1110
1160 RETURN
1199 '*****
1200 Y1=VAL(LEFT$(D1$,2)):Y2=VAL(LEFT$(D2$,2))
1210 DATE$=D1$:DT1=DATE:DATE$=D$
1220 IF (Y1 MOD 4)=0 THEN DAY1=367-DT1 ELSE DAY1=366-DT1
1230 DAY3=0
1240 IF Y2-Y1<2 THEN 1290
1250 FOR I=Y1+1 TO Y2-1
1260 YY$=STR$(I):IF I<10 THEN YY$="0"+RIGHT$(YY$,1)
1270 DATE$=RIGHT$(YY$,2)+"/12/31":DAY3=DAY3+DATE:DATE$=D$
1280 NEXT
1290 IF Y1=Y2 THEN DAY=DAY1+DT2-365 ELSE DAY=DAY1+DT2+DAY3

```



```

1300 IF (Y1 MOD 4)=0 THEN WK=9632+DT1 ELSE WK=9633+DT1
1310 FOR Y=Y1+1 TO Y1+30
1320   IF (Y MOD 4)=0 THEN NEN=366 ELSE NEN=365
1330   IF WK<=NEN THEN 1360
1340   WK=WK-NEN
1350 NEXT
1360 IF (I MOD 4)=0 THEN DAY(2)=29 ELSE DAY(2)=28
1370 FOR M=1 TO 12
1380   IF WK<=DAY(M) THEN 1410
1390   WK=WK-DAY(M)
1400 NEXT
1410 RETURN
1499 '*****
1500 IF LEN(INDT$)=0 THEN PRINT #2,CHR$(&H07);:RETURN
1510 PRINT #2,CHR$(&H1D);" ";CHR$(&H1D);
1520 INDT$=LEFT$(INDT$,LEN(INDT$)-1)
1530 RETURN

```

内部コード (16進)	アスキー ニーモニック	動 作 ・ 意 味
05	ENQ	現在のカーソル位置からそのフィールドの終りまでを消す。
07	BEL	ベルを鳴らす。
08	BS	カーソルを1つ左に移動する。
09	HT	次のタブ停止位置までスペースを発生する。
0A	LF	改行する。
0B	VT	カーソルをホームポジションへ移動する。
0C	FF	画面を消去する。
0D	CR	復帰する。
1C	ES	カーソルを右に移動する。
1D	GS	カーソルを左に移動する。
1E	RS	カーソルを上に移動する。
1F	VS	カーソルを下に移動する。

図12-9 F-BASIC オーダ

内部コード (16進)	アスキー ニーモニック	動 作 ・ 意 味
07	BEL	ベルを鳴らす。
08	BS	カーソルを1つ左に移動する。
09	HT	次のタブ停止位置までスペースを発生する。
0A	LF	改行する。
0B	VT	カーソルを上に移動する。
0C	FF	カーソルを右に移動する。
0D	CR	復帰する。
0E	SO	キーボードのロックを解除する。
0F	SI	キーボードをロックする。
1A	SUB	画面を消去する。
1E	RS	カーソルをホームポジションへ移動する。

図12-10 ADM-3A オーダ

12-7 音響カプラ／モデム

コンピュータが扱っている信号は、デジタル信号です。ところが電話回線は、音声信号(300Hzから4000Hz ぐらいまでのアナログ信号)しか送れません。そこで電話回線を使ってデータ通信を行なうためには、デジタル信号とアナログ信号を相互に変換するモデム(MODEM)という装置が必要となります。この MODEM というのは、MODulator(変調器)と DEMOduator(復調器)の頭文字を組み合わせた造語です。コンピュータからの送信データは、変調器によりアナログ信号に変換され電話回線を通して受信側に伝えられます。受信側では、復調器によりデジタル信号に戻されコンピュータに入力されます(図 12-11)。

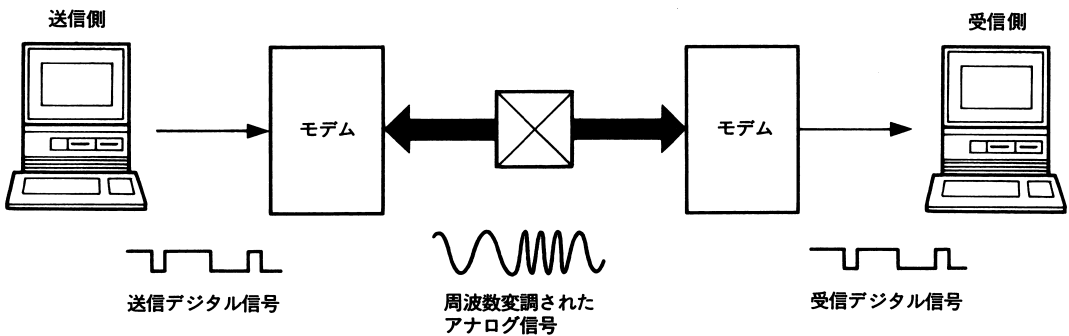


図12-11 モデムによるデータ通信

モデムを使って電話回線に接続するためには、電話線とモデムを直接接続しなければなりません。ところが音響カプラでは、変調器で変換されたアナログ信号をスピーカーによって音声信号に変換し、電話の送話器で直接送れるようになっていました。反対に受信側では、受話器から入力された音声信号をマイクロフォンにてアナログ信号に変換し、復調器によってデジタル信号にしてコンピュータに入力します。ですから音響カプラを使用するときには、電話機をそのまま使用することができます(図 12-12)。パソコンで簡単なデータ通信を行なおうとするときには、音響カプラは手軽で使いやすいのではないかと思います。しかし音響カプラでは、音声信号を介してデータ通信が行なわれるので、雑音、騒音等に注意する必要があります。

ところでモデムによってデジタル信号とアナログ信号の相互変換が行なわれるわけですが、この変換における規則を定めておかないと正しく変換できません。日本で使われているのは、主として CCITT V.21(300 ボー全二重)と CCITT V.23(1200 ボー半二重)の規格です。全二重通信では2本の通信経路(回線)必要ですが、電話線には一対のケーブルしかありません。1本の電話回線では、双方の送信信号は途中で衝突してしまいそうだという疑問がわいてきます。しかし電話線では、周波数の異なる4つの信号を使うことで全二重通信を実現しています。つまり電話回線ではアナログ信号という電気の波を使用しているので、水面の2つの波が衝突してもその先へ伝わっていくように、何事もなく双方の信号がすれ違って通るのです。

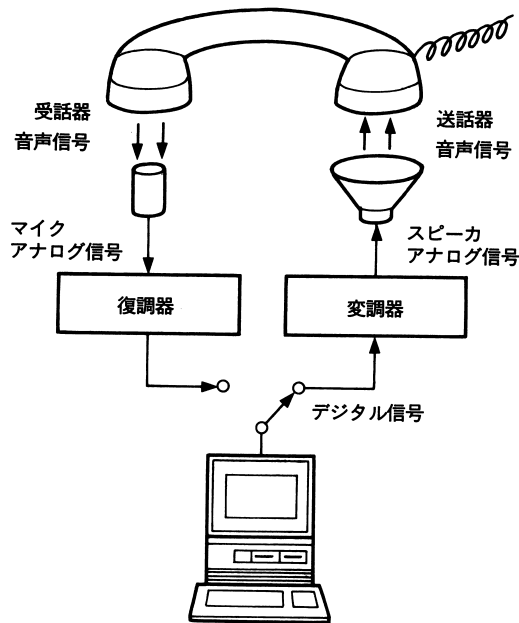


図12-12 音響カプラによるデータ通信

CCITT V.21 規格では、CALL モード(発信側)が 980Hz, 1180Hz を送信信号に使い、1650Hz, 1850Hz を受信信号に使います。ANSWER モード(受信側)では、その逆の組み合わせを使います。MODEM では、この CALL モードと ANSWER モードを切り換えスイッチで選択できるようになっているのですが、片一方が CALL モードを選択したら、もう一方は ANSWER モードを選択する必要があります。CCITT V.23 規格では、双方のモデムに HALT(半二重)を選択します。このとき、1300Hz が "1" のデータ、2100Hz が "0" のデータに変換されます(図 12-13)。

[300 ボー全二重]

周波数(Hz)	デジタル信号	CALLモード	ANSWERモード
980	"1"	送信信号	受信信号
1180	"0"		
1650	"1"	受信信号	送信信号
1850	"0"		

[1200 ボー半二重]

周波数(Hz)	デジタル信号
1300	"1"
2100	"0"

図12-13 電話回線で使用される信号

モデムあるいは音響カプラを使用してデータ通信を行なう際のホストあるいはターミナルのプログラムは、パソコン同士を直接つないだときと何らかわりません。電話回線網を正しく接続できれば、前節までのプログラムを使ってデータやプログラムの転送を行なうことができます。

(1) 音響カプラの使用

音響カプラを使用するには、以下の手順で行ないます。

- ① 音響カプラとコンピュータ(パソコン)を、RS-232C ケーブルで接続します。
- ② 音響カプラの各種スイッチ (CALL/ANS, FULL/HALF, TEST/DTE 等)を設定します。
スイッチの種類、表示等はメーカーによって異なりますので、説明書で確認します。
- ③ 音響カプラの電源を ON にします。
- ④ 相手側へダイヤルして、つながったら受話器を音響カプラに押し込みます。
- ⑤ コンピュータ(パソコン)からデータを送ります。

音響カプラを使用するうえでの注意点について、いくつか述べます。音響カプラは“音”を信号として使っていますから、周辺からの雑音により誤動作する危険性があります。ですから受話器はしっかりとセットする必要があります。また机と音響カプラの間にクッションを用いるのもよい方法です。受話器に消毒カバーがついているときは、はずした方がいいでしょう。それから最近流行のファッション電話機だと、受話器の形が音響カプラと適合しない場合がありますから注意が必要です。

(2) モデムの使用

モデムを使用すれば電話回線とモデムが直接接続されるので、音響カプラのような雑音の心配は必要ありません。しかしモデムには、電話の発信や着信をしたりモデムと電話回線との接続をコントロールする働きはありません。それで、モデムに NCU(Network Control Unit)という回線制御装置を接続する必要があります。NCU は電話交換機を直接起動することができる装置で、電話のフックやダイヤルと同じ動きをします(図 12-14)。

NCU には様々な種類があり、モデムと電話機を手動で切り換えるものから、自動発信、自動着信の機能をもつ NCU もあります。自動発着信のできる NCU を使えば自動的に電話回線の接続がなされるので、夜間のデータ収集や無人データベースなどのシステムを構築できます。

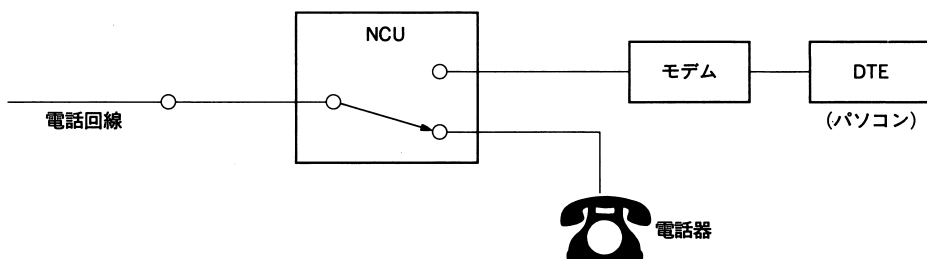


図12-14 NCUの接続

モデムを使用するとき注意することは、NCUをつけたモデムはNTTの認定を受けたものしか使用できない点です。取り付け工事も資格をもった工事担当者でなければできないことになっています。ただし認定を受けたモデム機器をモジュージャックに差し込んでつなぐのであれば、電話局で届け出るだけでできます。パソコン通信の盛んなアメリカにはすぐれた機能をもつモデムがありますが、アメリカではATTのBell規格によるものが多く、そのままでは使えないものが多いようです。

13-1 多色グラフィック

FM77AV は、8 色 2 画面 (640×200 ドット) と 4096 色 1 画面 (320×200 ドット) のふたつのグラフィックモードを持っています。

8 色 2 画面のモードでは、ページ切り換えによってアクティブページ、ディスプレイページが、それぞれ個別に設定可能となっています。これは、従来の FM-7 シリーズでの画面が 2 画面にふえて、それらが自由に切り換え可能になったということを意味しています。このモードでの VRAM アドレスと画面の対応は、図 13-1 のようになっています。セレクトされたページの B, R, G の 3 つの VRAM の内容が重ね合わされて画面に表示されているわけです。たとえば、X=632, Y=199 の座標の点は、\$3E7F (BLUE), \$7E7F (RED), \$BE7F (GREEN) のビット 7 の状態が重ね合わされて表示されます。ビットの状態と表示される色との関係は、図 13-2 のようになります。

一方、4096 色 1 画面モードでの VRAM アドレスと画面の対応は、図 13-3 に示すとおりです。12 個の VRAM の内容が重ね合わされて、画面に表示されます。VRAM の状態と表示色 (パレットコード) の関係を図 13-4 に示します。

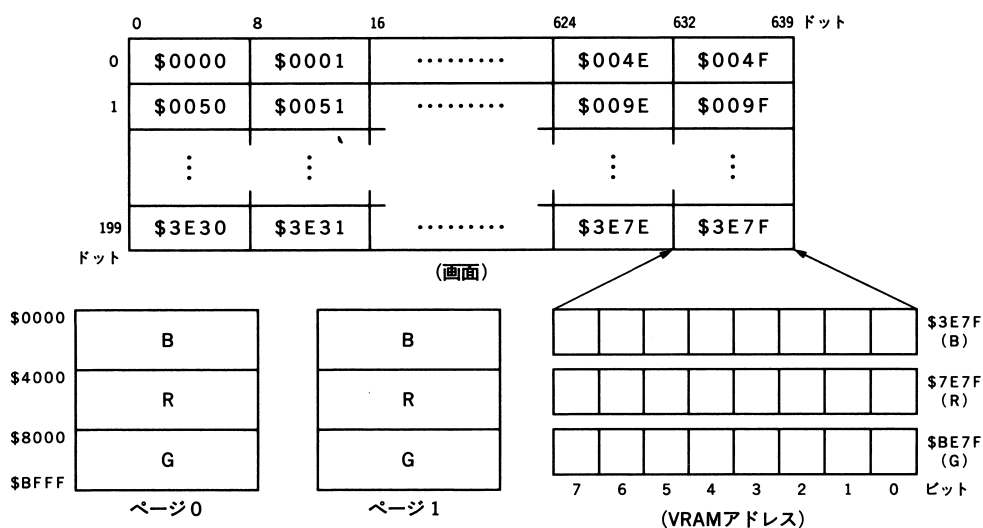


図13-1 VRAMアドレスと画面の対応

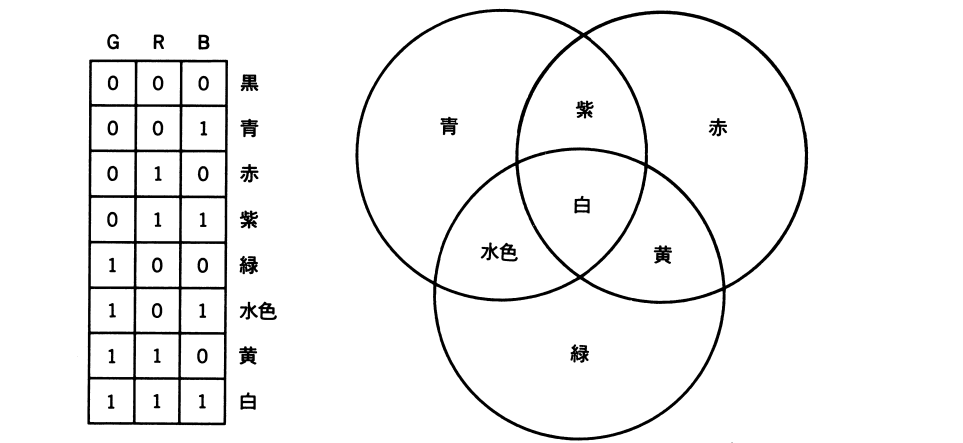


図13-2 VRAMと表示色の関係

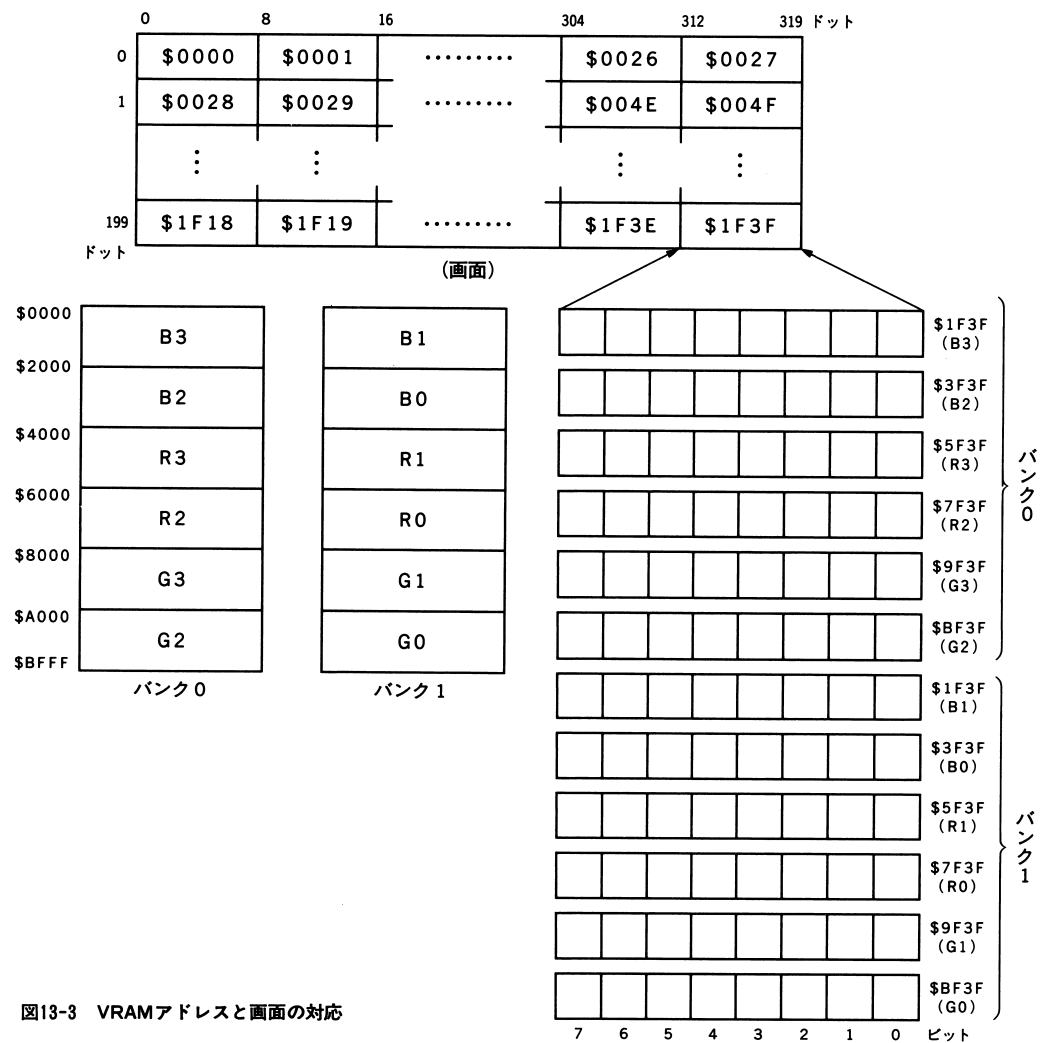


図13-3 VRAMアドレスと画面の対応

パレット番号													パレットコード		
	G3	G2	G1	G0	R3	R2	R1	R0	B3	B2	B1	B0	G輝度	R輝度	B輝度
0	0	0	0	0	0	0	0	0	0	0	0	0	[0 , 0 , 0]		黒
1	0	0	0	0	0	0	0	0	0	0	0	1	[0 , 0 , 16*1]		青[最低輝度]
2	0	0	0	0	0	0	0	0	0	0	1	0	[0 , 0 , 16*2]		
3	0	0	0	0	0	0	0	0	0	0	1	1	[0 , 0 , 16*3]		
⋮															
15	0	0	0	0	0	0	0	0	1	1	1	1	[0 , 0 , 16*15]		青[最高輝度]
16	0	0	0	0	0	0	0	1	0	0	0	0	[0 , 16*1 , 0]		赤[最低輝度]
17	0	0	0	0	0	0	0	1	0	0	0	1	[0 , 16*1 , 16*1]		紫[最低輝度]
⋮															
4095	1	1	1	1	1	1	1	1	1	1	1	1	[16*15, 16*15, 16*15]		白[最高輝度]

図13-4 VRAMと表示色の関係

VRAM をアクセスするときには、バンク 0 とバンク 1 をセレクトしてアクセスしなければなりません。たとえば B 輝度 = $16 * 5$ の青を表示するには、次の手順を実行します。

- ① バンク 0 をセレクトする。
- ② B2 の VRAM に書き込む。
- ③ バンク 1 をセレクトする。
- ④ B0 の VRAM に書き込む。

このバンクのセレクトは、8 色 2 画面モード時の切り替えと同じで、サブシステム I/O レジスタ (\$D430) のビット 5 を操作します(図 13-5)。4096 色モード用サブモニタ(タイプ B)では、バンクの切り換えを行ないながら画面表示を行なっています。ですから BIOS を利用して画面表示を行なうときには、バンク切り替えを意識する必要はありません。しかし、ダイレクトアクセスで VRAM を直接にメイン CPU よりアクセスするときには、バンク切り替えを行なってアクセスする必要があります。

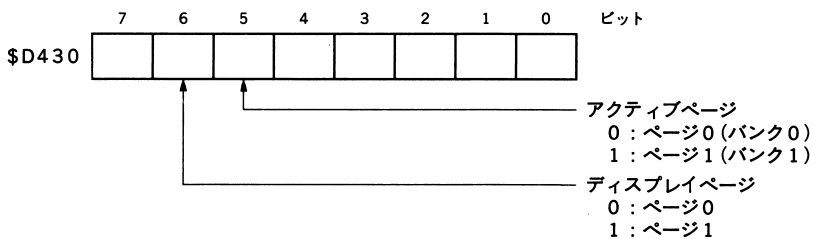


図13-5 バンク切り替えレジスタ

4096 色モードでは、RGB がそれぞれ 4 画面(色バンク)から構成されています。ですから、RGB 各色について 16 段階の濃淡を指定することができます。リスト 13-1 に RGB 各 16 段階の色調を表示するプログラムを示します。この RGB 各 16 段階の色を自在の組み合わせることにより、 $16 \times 16 \times 16 = 4096$ 色が表示できるわけです。ここで RGB の各輝度が同じ値の色を選択すると、白黒 16 段階の明度表示(SILVER)となります。リスト 13-2 に 4096 色を 256 色ずつ順次表示するプログラムを示します。いままでの 8 色表示と比べると何と色彩やかなことでしょう。まさしく天然ショックパソコンですね。

リスト 13-1 RGB 輝度表示

```

100 '*****
110 '*   RGB ノウタン ヒョウシ"           *
120 '*   ( LIST 13-1 )   V3.3           *
130 '*****
140 CLS:SCREEN@ 1:PALETTE@
150 FOR I=0 TO 15
160   LOCATE I*2+6,4:PRINT RIGHT$("00"+STR$(I),2);
170   LOCATE 0,6:PRINT "BLUE"
180   LOCATE 0,10:PRINT "RED "
190   LOCATE 0,14:PRINT "GREEN"
200   LOCATE 0,18:PRINT "SILVER"
210   LINE (I*16+52,44)-(I*16+63,59),PSET,[0,0,16*I],BF
220   LINE (I*16+52,76)-(I*16+63,91),PSET,[0,16*I,0],BF
230   LINE (I*16+52,108)-(I*16+63,123),PSET,[16*I,0,0],BF
240   LINE (I*16+52,140)-(I*16+63,155),PSET,[16*I,16*I,16*I],BF
250 NEXT
260 A$=INKEY$:IF A$="" THEN 260
270 END

```

リスト 13-2 4096 色表示

```

100 '*****
110 '*   4096 ショク ヒョウシ"           *
120 '*   ( LIST 13-2 )   V3.3           *
122 '*****
130 SCREEN@ 1:CLS:PALETTE@
140 LOCATE 15,2:PRINT "BLUE"
150 LOCATE 2,9:PRINT "R":LOCATE 2,10:PRINT "E":LOCATE 2,11:PRINT "D"
160 FOR J=0 TO 15
170   FOR I=0 TO 15
180     LOCATE I*2+6,4:PRINT RIGHT$(STR$(I),2)
190     LOCATE 4,J+5:PRINT RIGHT$(STR$(J),2)
200     LINE (I*16+48,J*8+40)-(I*16+63,J*8+47),PSET,[16,J*16,I*16],BF
210   NEXT
220 NEXT
230 FOR G=0 TO 15
240   LOCATE 0,1:PRINT "GREEN=":RIGHT$(STR$(G),2)
250   FOR R=0 TO 15
260     FOR B=0 TO 15
270       PALETTE 256+R*16+B,[G*16,R*16,B*16]
280     NEXT
290 NEXT
300 LOCATE 12,22:PRINT "Hit Any Key!!"
310 A$=INKEY$:IF A$="" THEN 310
320 LOCATE 12,22:PRINT SPC(15)
330 NEXT

```

13-2 アナログパレット

(1) パレットレジスタ

従来のFM-7シリーズでは、8色のTTLパレットが使われていました。これは、FM77AVでも8色モード用のパレットとして使用されています。このTTLパレットを設定するには、メインシステムI/Oレジスタ(\$FD38~\$FD3F)のビット2~ビット0を操作します(図13-6)。BASICでは、COLOR文または、PALETTE文にて設定できます。なおFM-7では、パレット番号0に対して黒のカラーコードしか設定できませんでしたが、FM77AVでは、8色が同等の扱いをうけるようになっています。

パレット番号	アドレス	ビット2	ビット1	ビット0	初期値	色
0	\$FD38	G	R	B	000	黒
1	\$FD39	G	R	B	001	青
2	\$FD3A	G	R	B	010	赤
3	\$FD3B	G	R	B	011	紫
4	\$FD3C	G	R	B	100	緑
5	\$FD3D	G	R	B	101	水色
6	\$FD3E	G	R	B	110	黄
7	\$FD3F	G	R	B	111	白

図13-6 TTLパレットレジスタ

4096色モードのときには、4096色から4096色へのマッピングができるアナログパレットが使われるようになりました。このアナログパレットは、色の数がTTLパレットの8色から4096色に増加しただけで、基本的な考え方は同じです。アナログパレットを操作するには、メインシステムI/Oレジスタ(\$FD30~\$FD34)をアクセスします(図13-7)。\$FD30、\$FD31の12ビットで0~4095のアナログパレット番号を指定します。LC0~LC3が青、LC4~LC7が赤、LC8~LC11が緑の色の輝度に対応しています。そして、\$FD32~\$FD34の下位4ビットでパレットコードの

アドレス	ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0
\$FD30	0	0	0	0	LC11	LC10	LC9	LC8
\$FD31	LC7	LC6	LC5	LC4	LC3	LC2	LC1	LC0

	アドレス	ビット3	ビット2	ビット1	ビット0
B輝度レジスタ	\$FD32	BL3	BL2	BL1	BL0
R輝度レジスタ	\$FD33	RL3	RL2	RL1	RL0
G輝度レジスタ	\$FD34	GL3	GL2	GL1	GL0

図13-7 アナログパレットレジスタ

B輝度, R輝度, G輝度を指定します。ここで注意することは、アナログパレットが変化するのが\$FD32, \$FD33, \$FD34のいずれかに値が書き込まれたときだということです。ですからアナログパレットの設定には、\$FD30, \$FD31をまず設定して、それから\$FD32, \$FD33, \$FD34に値をセットする必要があるのです。それを逆にすると、思わぬパレットが変化してしまいます。BASICでは、PALETTE文にて容易にパレットコードを設定できます。

(2) パレットによる重ね合わせ表示

それでは、リスト13-3をまず実行してみてください。これは8色モードにおいて、パレットを用いて画面の重ね合わせを実現したものです。これは、青と赤が重なったときのパレットコード(紫)を、青にするか、赤にするかによって、青と赤の重ね合わせ表示の優先度を変えたものです。紫のパレットコードに青のカラーコードを設定すると、青が手前で赤が背景の重ね合わせ表示となります。同様のことを緑についても行なえば、8色モードでは3色3画面または、4色2画面の重ね合わせ表示が可能となります。リスト13-4がそのサンプルプログラムです。これは、アルフォスやゼビウス等のスクロールゲームで活用されていた手法なのですが、大きな欠点があります。それは、ただでさえ少ない色がさらに減ってしまうことです。

しかし4096色モードでは、使用できるパレットが4096個もあるので、この手法を無理なく活用できます。リスト13-5を実行してみてください。表示優先度つき64色2画面の重ね合わせ表示を行ないます。そしてリスト13-6の方は、表示優先度つき16色3画面の重ね合わせ表示の例です。

リスト13-3 重ね合わせ表示(8色モード)

```

100 '*****
110 '*      8 ショク カサネアツセ ヒョウシ  テスト (1)      *
120 '*      ( LIST 13-3 )      V3.3      *
130 '*****
140 SCREEN 0:WIDTH 40,20
150 COLOR=(1,1):COLOR=(2,2):COLOR=(3,3)
160 CLS:LOCATE 7,2:PRINT "パレット ニヨル カサネアツセ テスト":LOCATE 10,4:PRINT "[
NORMAL  ]":GOSUB 250
170 GOSUB 310
180 COLOR=(1,1):COLOR=(2,2):COLOR=(3,2)
190 CLS:LOCATE 7,2:PRINT "パレット ニヨル カサネアツセ テスト":LOCATE 10,4:PRINT "[ R
ED  17セシ  ]":GOSUB 250
200 GOSUB 310
210 COLOR=(1,1):COLOR=(2,2):COLOR=(3,1)
220 CLS:LOCATE 7,2:PRINT "パレット ニヨル カサネアツセ テスト":LOCATE 10,4:PRINT "[ B
LUE 17セシ  ]":GOSUB 250
230 GOSUB 310
240 GOTO 150
250 LINE (200,80)-(350,130),PSET,2,BF
260 FOR X=426 TO 450-14*24 STEP -24
270     LINE (X,60)-(X+15,150),OR,1,BF
280     FOR I=0 TO 300:NEXT
290 NEXT
300 RETURN
310 COLOR 2:LOCATE 10,17:PRINT "Hit Any Key !!!":COLOR 7
320 A$=INKEY$:IF A$="" THEN 320
330 RETURN

```

リスト 13-4 重ね合わせ表示(8色モード)

```

100 '*****
110 '* 8ジョク カサネアワセ ヒョウシ ( 3ジョク 3カ"メン ) B < R < G *
120 '* ( LIST 13-4 ) V3.3 *
130 '*****
140 SCREEN0 0:WIDTH 40,20
150 COLOR=(1,1):COLOR=(2,2):COLOR=(3,3):COLOR=(4,4):COLOR=(5,5):COLOR=
(6,6):COLOR=(7,7)
160 CLS
170 LOCATE 8,2:PRINT "ハ°レット ニヨル カサネアワセ テスト"
180 LOCATE 11,3:PRINT "8 ジョク 1 カ"メン"
190 LOCATE 8,4:PRINT "[ NORMAL MODE ]"
200 GOSUB 440
210 GOSUB 510
220 COLOR=(1,1):COLOR=(2,2):COLOR=(3,2):COLOR=(4,4):COLOR=(5,4):COLOR=
(6,4):COLOR=(7,4)
230 CLS
240 LOCATE 8,2:PRINT "ハ°レット ニヨル カサネアワセ テスト"
250 LOCATE 11,3:PRINT "4 ジョク 3 カ"メン"
260 LOCATE 8,4:PRINT "[ 17セント" ] B < R < G"
270 GOSUB 440
280 GOSUB 510
290 COLOR=(1,1):COLOR=(2,2):COLOR=(3,2):COLOR=(4,4):COLOR=(5,5):COLOR=
(6,2):COLOR=(7,2)
300 CLS
310 LOCATE 8,2:PRINT "ハ°レット ニヨル カサネアワセ テスト"
320 LOCATE 11,3:PRINT "5 ジョク 2 カ"メン"
330 LOCATE 8,4:PRINT "[ 17セント" ] B+G < R"
340 GOSUB 440
350 GOSUB 510
360 COLOR=(1,1):COLOR=(2,2):COLOR=(3,1):COLOR=(4,4):COLOR=(5,5):COLOR=
(6,4):COLOR=(7,5)
370 CLS
380 LOCATE 8,2:PRINT "ハ°レット ニヨル カサネアワセ テスト"
390 LOCATE 11,3:PRINT "5 ジョク 2 カ"メン"
400 LOCATE 8,4:PRINT "[ 17セント" ] B+G > R"
410 GOSUB 440
420 GOSUB 510
430 GOTO 150
440 LINE (200,80)-(350,130),OR,1,BF
450 LINE (250,100)-(400,150),OR,4,BF
460 FOR X=474 TO 474-15*24 STEP -24
470 LINE (X,60)-(X+15,170),OR,2,BF
480 FOR I=0 TO 400:NEXT
490 NEXT
500 RETURN
510 LOCATE 11,18:PRINT "Hit Any Key !!!";
520 A$=INKEY$:IF A$="" THEN 520
530 RETURN

```

リスト 13-5 64色2画面重ね合わせ表示

```

100 '*****
110 '* 4096ジョク カサネアワセ ヒョウシ テスト ( 64 ジョク 2 カ"メン ) *
120 '* ( L13-5 ) V3.3 *
122 '*****
130 SCREEN0 1:WIDTH 40,20
140 PALETTE0
150 CLS:LOCATE 7,2:PRINT "ハ°レット ニヨル カサネアワセ テスト":LOCATE 10,4:PRINT "[
NORMAL ]":GOSUB 220
160 GOSUB 380
170 CLS:PALETTE0:LOCATE 7,2:PRINT "ハ°レット ニヨル カサネアワセ テスト":LOCATE 10,4:P
RINT "[ BANK0 17セン ]":GOSUB 410:BEEP:GOSUB 380:GOSUB 220

```

```

180 GOSUB 380
190 CLS:PALETTE@:LOCATE 7,2:PRINT "パレット ニョル カサネアツセ テスト":LOCATE 10,4:P
RINT "[ BANK1 17セシ ]":GOSUB 510:BEEP:GOSUB 380:GOSUB 220
200 GOSUB 380
210 GOTO 140
220 LINE (100+0*8,80+0*8)-(150+0*8,100+0*8),PSET,[8*16,0,0],BF
230 LINE (100+1*8,80+1*8)-(150+1*8,100+1*8),PSET,[4*16,0,0],BF
240 LINE (100+2*8,80+2*8)-(150+2*8,100+2*8),PSET,[0,8*16,0],BF
250 LINE (100+3*8,80+3*8)-(150+3*8,100+3*8),PSET,[0,4*16,0],BF
260 LINE (100+4*8,80+4*8)-(150+4*8,100+4*8),PSET,[0,0,8*16],BF
270 LINE (100+5*8,80+5*8)-(150+5*8,100+5*8),PSET,[0,0,4*16],BF
280 FOR X=210 TO 300-12*20 STEP -12
290   LINE (X,60)-(X+8,130),OR,[2*16,0,0],BF
300   LINE (X+4,68)-(X+12,138),OR,[1*16,0,0],BF
310   LINE (X+8,76)-(X+16,146),OR,[0,2*16,0],BF
320   LINE (X+12,84)-(X+20,154),OR,[0,1*16,0],BF
330   LINE (X+16,92)-(X+24,162),OR,[0,0,2*16],BF
340   LINE (X+20,100)-(X+28,170),OR,[0,0,1*16],BF
350   FOR I=0 TO 500:NEXT
360 NEXT
370 RETURN
380 COLOR 2:LOCATE 10,17:PRINT "Hit Any Key !!!":COLOR 7
390 A$=INKEY$:IF A$="" THEN 390
400 RETURN
410 LOCATE 10,17:PRINT "PALETTE SET チュウ !!!"
420 FOR G=0 TO 15
430   FOR R=0 TO 15
440     FOR B=0 TO 15
450       IF (G AND &HOC) OR (R AND &HOC) OR (B AND &HOC) THEN PALETTE
256*G+16*R+B,[(G AND &HOC)*16,(R AND &HOC)*16,(B AND &HOC)*16]
460     NEXT
470   NEXT
480 NEXT
490 LOCATE 10,17:PRINT SPC(20)
500 RETURN
510 LOCATE 10,17:PRINT "PALETTE SET チュウ !!!"
520 FOR G=0 TO 15
530   FOR R=0 TO 15
540     FOR B=0 TO 15
550       IF (G AND &H03) OR (R AND &H03) OR (B AND &H03) THEN PALETTE
256*G+16*R+B,[(G AND &H03)*16,(R AND &H03)*16,(B AND &H03)*16]
560     NEXT
570   NEXT
580 NEXT
590 LOCATE 10,17:PRINT SPC(20)
600 RETURN

```

リスト 13-6 16色3画面重ね合わせ表示

```

100 *****
110 * 4096ショク カサネアツセ ヒョウシ( 16 ショク 3 カメン ) *
120 * ( LIST 13-6 ) V3.3 *
130 *****
140 SCREEN@ 1:WIDTH 40,20
150 PALETTE@
160 CLS:LOCATE 7,2:PRINT "パレット ニョル カサネアツセ テスト":LOCATE 10,4:PRINT "[
NORMAL ]":GOSUB 210
170 GOSUB 370
180 CLS:PALETTE@:LOCATE 7,2:PRINT "パレット ニョル カサネアツセ テスト":LOCATE 10,4:P
RINT "[ 17セシ ] G<R<B":GOSUB 400:BEEP:GOSUB 370:GOSUB 210
190 GOSUB 370
200 GOTO 150
210 LINE (100+0*8,80+0*8)-(150+0*8,100+0*8),PSET,[8*16,0,0],BF
220 LINE (100+1*8,80+1*8)-(150+1*8,100+1*8),PSET,[4*16,0,0],BF

```

```

230 LINE (100+2*8,80+2*8)-(150+2*8,100+2*8),PSET,[2*16,0,0],BF
240 LINE (100+3*8,80+3*8)-(150+3*8,100+3*8),PSET,[1*16,0,0],BF
250 LINE (100+4*8,80+4*8)-(150+4*8,100+4*8),PSET,[0,0,8*16],BF
260 LINE (100+5*8,80+5*8)-(150+5*8,100+5*8),PSET,[0,0,4*16],BF
270 LINE (100+6*8,80+6*8)-(150+6*8,100+6*8),PSET,[0,0,2*16],BF
280 LINE (100+7*8,80+7*8)-(150+7*8,100+7*8),PSET,[0,0,1*16],BF
290 FOR X=210 TO 300-12*20 STEP -12
300     LINE (X,60)-(X+8,130),OR,[0,8*16,0],BF
310     LINE (X+4,68)-(X+12,138),OR,[0,4*16,0],BF
320     LINE (X+8,76)-(X+16,146),OR,[0,2*16,0],BF
330     LINE (X+12,84)-(X+20,154),OR,[0,1*16,0],BF
340     FOR I=0 TO 500:NEXT
350 NEXT
360 RETURN
370 COLOR 2:LOCATE 10,17:PRINT "Hit Any Key !!":COLOR 7
380 A$=INKEY$:IF A$="" THEN 380
390 RETURN
400 LOCATE 10,17:PRINT "PALETTE SET  ͡͡͡!!"
410 FOR G=0 TO 15
420     FOR R=0 TO 15
430         FOR B=0 TO 15
440             IF B<>0 THEN PALETTE 256*G+16*R+B,[0,0,B*16]:GOTO 460
450             IF R<>0 THEN PALETTE 256*G+16*R+B,[0,R*16,0]
460         NEXT
470     NEXT
480 NEXT
490 LOCATE 10,17:PRINT SPC(20)
500 RETURN

```

いかがでしょうか。表示優先度付きの重ね合わせ表示のためのパレットの設定方法を理解していただけたでしょうか。要するに重なったときの色を、優先度の高い色に設定すればいいわけです。しかし、それにしても BASIC でのパレットの設定では、時間がかかりすぎます。これについては、マシン語によるパレットの設定方法を後ほどいろいろ考えてみようと思います。

リスト 13-7 を実行してみてください。背景が緑の画面に、上下に動く長方形が青の画面に、そして左右に動く長方形が赤の画面に描かれています。これは BASIC だけでつくられているのですが、画面のちらつきもなく、BASIC だけにしてはスムーズな動きだと思います。FM77AV の大きな可能性を示す例のひとつではないでしょうか。とにかく 4096 色という豊富なアナログパレットを利用して、グラフィックテクニックをいろいろと考えてみてください。4096 色は、利用のしがいがありますよ。

リスト 13-7 パレットによる重ね合わせ表示

```

100 '*****
110 '*   ハ°レット ニヨル カサネアワセ ヒョウシ   SAMPLE   *
120 '*   ( LIST 13-7 )   V3.3   *
130 '*****
140 SCREEN@ 1:SCREEN 7:CLS:GOSUB 360
150 LINE (20,20)-(250,170),PSET,[16,0,0],BF:LINE (110,60)-(165,114),PS
ET,[32,0,0],BF
160 RESTORE 540
170 FOR I=1 TO 3
180     READ CL1,CL2,CL3,X1,Y1,X2,Y2:LINE (X1,Y1)-(X2,Y2),PSET,[CL1,CL
2,CL3]

```

```

190   FOR J=1 TO 3:READ X2,Y2:LINE -(X2,Y2),PSET,[CL1,CL2,CL3]:NEXT
200   READ X1,Y1:PAINT (X1,Y1),[CL1,CL2,CL3]
210   NEXT
220  X1=54:Y1=100:F1=1:X2=100:Y2=30:F2=1:X3=150:Y3=50:F3=1:X4=34:Y4=70:
    F4=1
230  SCREEN 2:IF F1>0 THEN LINE (X1,Y1)-(X1+3,Y1+23),PSET,[0,0,0],BF:LI
    NE (X1+4,Y1)-(X1+27,Y1+23),PSET,[0,240,0],BF:X1=X1+4 ELSE LINE (X1+20,
    Y1)-(X1+23,Y1+23),PSET,[0,0,0],BF:LINE (X1-4,Y1)-(X1+19,Y1+23),PSET,[0
    ,240,0],BF:X1=X1-4
240  SCREEN 1:IF F2>0 THEN LINE (X2,Y2)-(X2+23,Y2+3),PSET,[0,0,0],BF:LI
    NE (X2,Y2+4)-(X2+23,Y2+27),PSET,[0,0,208],BF:Y2=Y2+4 ELSE LINE (X2,Y2+
    20)-(X2+23,Y2+23),PSET,[0,0,0],BF:LINE (X2,Y2-4)-(X2+23,Y2+15),PSET,[0
    ,0,208],BF:Y2=Y2-4
250  SCREEN 1:IF F3>0 THEN LINE (X3,Y3)-(X3+23,Y3+3),PSET,[0,0,0],BF:LI
    NE (X3,Y3+4)-(X3+23,Y3+27),PSET,[0,0,240],BF:Y3=Y3+4 ELSE LINE (X3,Y3+
    20)-(X3+23,Y3+23),PSET,[0,0,0],BF:LINE (X3,Y3-4)-(X3+23,Y3+15),PSET,[0
    ,0,240],BF:Y3=Y3-4
260  SCREEN 2:IF F4>0 THEN LINE (X4,Y4)-(X4+3,Y4+23),PSET,[0,0,0],BF:LI
    NE (X4+4,Y4)-(X4+27,Y4+23),PSET,[0,208,0],BF:X4=X4+4 ELSE LINE (X4+20,
    Y4)-(X4+23,Y4+23),PSET,[0,0,0],BF:LINE (X4-4,Y4)-(X4+19,Y4+23),PSET,[0
    ,208,0],BF:X4=X4-4
270  IF X1=206 THEN F1=-1
280  IF X1=50 THEN F1=1
290  IF Y2=142 THEN F2=-1
300  IF Y2=26 THEN F2=1
310  IF Y3=122 THEN F3=-1
320  IF Y3=46 THEN F3=1
330  IF X4=226 THEN F4=-1
340  IF X4=30 THEN F4=1
350  GOTO 230
360  PALETTE@:RESTORE 400
370  FOR I=1 TO 39:READ A1,A2,A3,A4:PALETTE A1,[A2,A3,A4]:NEXT
380  RETURN
390  '*   ハレット   テータ
400  DATA 496,0,240,112,752,0,240,112,1008,0,240,112
410  DATA 1264,0,240,112,1520,0,240,112,509,0,240,112
420  DATA 765,0,240,112,1021,0,240,112,1277,0,240,112
430  DATA 1533,0,240,112,511,0,240,112,767,0,240,112
440  DATA 1023,0,240,112,1279,0,240,112,1535,0,240,112
450  DATA 271,80,192,240,527,80,192,240,1039,80,192,240
460  DATA 1295,80,192,240,479,80,192,240,735,80,192,240
470  DATA 1247,80,192,240,464,80,208,112,720,80,208,112
480  DATA 976,80,208,112,1232,80,208,112,477,80,208,112
490  DATA 733,80,208,112,989,80,208,112,269,224,144,144
500  DATA 525,224,144,144,781,224,144,144,1037,224,144,144
510  DATA 1293,224,144,144,256,48,48,48,512,96,96,96
520  DATA 768,32,32,32,1024,16,16,16,1280,64,64,64
530  '*   チョウ   テータ
540  DATA 48,0,0,90,60,109,60,109,114,90,134,90,60,100,70
550  DATA 64,0,0,165,60,185,60,185,134,165,114,165,60,175,70
560  DATA 80,0,0,110,114,165,114,185,134,90,134,110,114,130,120

```

(3) パレットの設定方法

重ね合わせ表示の例では、パレットの設定にかなりの時間がかかっています。これでは実用的とはいえません。しかし、PALETTE@によるパレットの設定は、4096色を瞬時に設定しています。ですから高速パレット設定の何らかの方法があるはずで、それをあれこれ考えてみたいと思います。

アナログパレットの設定には、メインシステム I/O レジスタ(\$FD30~\$FD34)に値を設定すればよいわけです。しかしアナログパレットは、ブランキング期間以外に書き込むと、アクセスノ

イズが発生してしまいます。したがってアクセスするタイミングをはかる必要があります。アナログパレットへの書き込みタイミングには、次のような方法が考えられます。

- ① アクセスノイズの発生を無視して、無条件でパレットを変更する(リスト 13-8)。
- ② VSYNC 期間中にパレットを変更する(リスト 13-9)。
- ③ ディスプレイページをオフにして、640 ドット 8 色モードに切り換えてパレットを変更する(リスト 13-10)。
- ④ ディスプレイタイミングをチェックして、ブランキング期間中にパレットを変更する(リスト 13-11)。

①の方法では、瞬時に 4096 色のパレット設定が行なえます。しかし画面全体にアクセスノイズが発生してしまいます。

②の方法だときれいにパレット設定が行なえますが、4096 色のパレット設定に約 8 秒ほどかかってしまいます。それでも BASIC で 4096 色のパレット設定には 1 分以上かかるのと比べればずっと高速なのですが、まだまだ不満が残ります。

③の方法が BASIC の PALETTE@文と同じ方法で、パレットの設定が瞬時に完了します。しかしパレットの設定中、瞬時的に画面が消えるのが不満です。

④の方法は、VSYNC 期間以外の水平ブランキング期間中にもパレット設定を行なう方法です。図 13-8 のディスプレイタイミングをみてください。

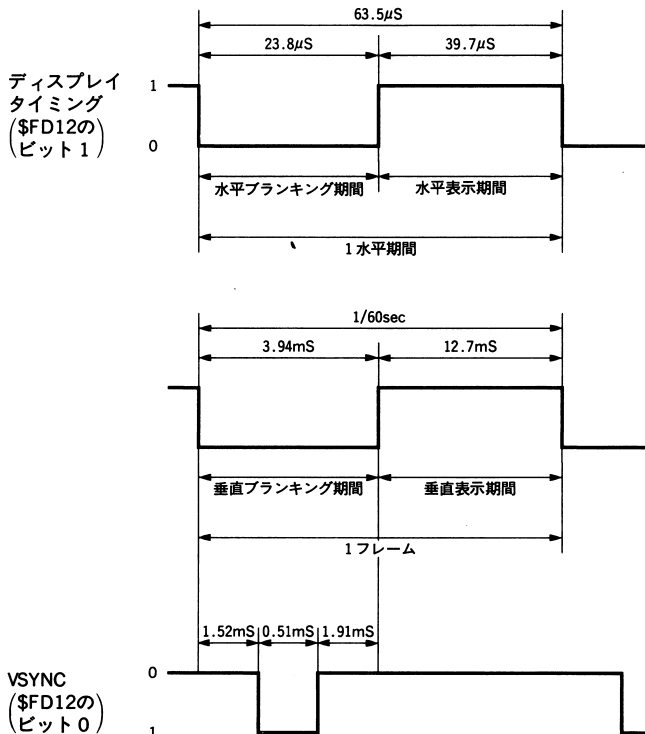


図13-8 ディスプレイタイミング

②の方法では、1フレーム(1/60秒)中の垂直同期期間(0.51m秒)だけにパレット設定を行いません。しかしこれでは時間がかかってしまいます。そこで水平ブランキング期間中にもパレット設定を行なおうとするのが④の方法です。ところが、水平ブランキング期間は 23.8μ 秒しかありません。ですからタイミングをうまくとらないと水平表示期間にかかってしまい、アクセスノイズが発生してしまいます。そこで、垂直同期期間があるいは、ディスプレイタイミング(\$FD12のビット1)がオンからオフに変化した瞬間をみはからってパレット設定を行なうことにします。しかも処理を高速にするためMMRを無効にして、サブシステムもHALTしておきます。こうすることにより、高速でかつアクセスノイズを発生させずにパレット設定が行なえます。この方法は、FM77AV 付属のデモプログラムの終わりの方で、4096色のパレット変更をしている部分と同じ

リスト 13-8 パレット変更(1)

```

00100 *****
00110 *      ハ°レット へンコウ (1)      *
00120 *      ( LIST 13-8 )      V3.3      *
00130 *****
01000      OPT      NOGEN
01002      5000      ORG      $5000
01010      5000 20      03      5005 ENTRY      BRA      LP00
01020      5002      0F      BB      FCB      15
01030      5003      0F      RR      FCB      15
01040      5004      0F      GG      FCB      15
01042      5005 108E 0FFF      LP00      LDY      #$0FFF
01050      5009 86      0F      LDA      #15
01060      500B 87      5004      STA      GG
01070      500E 86      0F      LP01      LDA      #15
01080      5010 87      5003      STA      RR
01090      5013 86      0F      LP02      LDA      #15
01100      5015 87      5002      STA      BB
01110      5018 86      5002      LP03      LDA      BB
01120      5018 27      12      502F      BEQ      LP04
01130      501D 108F FD30      STY      $FD30      ハ°レット NO.
01210      5021 86      5002      LDA      BB
01220      5024 87      FD32      STA      $FD32      BLUE
01230      5027 7F      FD33      CLR      $FD33      RED
01240      502A 7F      FD34      CLR      $FD34      GREEN
01250      502D 20      15      5044      BRA      LP05
01260      502F 86      5003      LP04      LDA      RR
01270      5032 27      10      5044      BEQ      LP05
01280      5034 108F FD30      STY      $FD30      ハ°レット NO.
01350      5038 7F      FD32      CLR      $FD32      BLUE
01360      503B 86      5003      LDA      RR
01370      503E 87      FD33      STA      $FD33      RED
01380      5041 7F      FD34      CLR      $FD34      GREEN
01382      5044 31      3F      LP05      LEAY      -1,Y
01390      5046 7A      5002      DEC      BB
01400      5049 2A      C0      5018      BPL      LP03
01410      504B 7A      5003      DEC      RR
01420      504E 2A      C3      5013      BPL      LP02
01430      5050 7A      5004      DEC      GG
01440      5053 2A      B9      500E      BPL      LP01
01450      5055 39      RTS
01460      5000      END      ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=5055
PROGRAM ENTRY ADDR=5000

```

方法です。4096 色ものパレットがあると、こんなテクニカルな方法も必要とされるのでしょうね。

なお、パレットの変更方法については AVTV 制御においてサンプルを挙げてあります。参照してください。

リスト 13-9 パレット変更(2)

```

00100
00110
00120
00130
01000
01002 5000
01010 5000 20 03 5005 ENTRY
01020 5002 0F BB FCB 15
01030 5003 0F RR FCB 15
01040 5004 0F GG FCB 15
01042 5005 108E OFFF LP00 LDY #OFFF
01050 5009 86 0F LDA #15
01060 5008 87 5004 STA GG
01070 500E 86 0F LP01 LDA #15
01080 5010 87 5003 STA RR
01090 5013 86 0F LP02 LDA #15
01100 5015 87 5002 STA BB
01110 5018 86 5002 LP03 LDA BB
01120 5018 27 15 5032 BEQ LP04
01130 5010 108F FD30 STY $FD30
01210 5021 8D 505C JSR TIMING
01240 5024 86 5002 LDA BB
01250 5027 87 FD32 STA $FD32
01260 502A 7F FD33 CLR $FD33
01270 502D 7F FD34 CLR $FD34
01280 5030 20 18 504A BRA LP05
01290 5032 86 5003 LP04 LDA RR
01300 5035 27 13 504A BEQ LP05
01310 5037 108F FD30 STY $FD30
01380 503B 8D 505C JSR TIMING
01410 503E 7F FD32 CLR $FD32
01420 5041 86 5003 LDA RR
01430 5044 87 FD33 STA $FD33
01440 5047 7F FD34 CLR $FD34
01442 504A 31 3F LP05 LEAY -1,Y
01450 504C 7A 5002 DEC BB
01460 504F 2A C7 5018 BPL LP03
01470 5051 7A 5003 DEC RR
01480 5054 2A 8D 5013 BPL LP02
01490 5056 7A 5004 DEC GG
01500 5059 2A B3 500E BPL LP01
01510 505B 39 RTS
01520 505C 86 FD12 TIMING LDA $FD12
01530 505F 85 01 BITA #$01
01540 5061 27 F9 505C BEQ TIMING
01550 5063 39 RTS
01560 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5063
PROGRAM ENTRY ADDR=5000

```

リスト 13-10 パレット変更(3)

```

00100
00110
00120
00130
01000
01002 5000
01010 5000 20 03 5005 ENTRY
01020 5002 0F BB
01030 5003 0F RR
01040 5004 0F GG
01042 5005 86 70 LP00
01044 5007 87 FD37 STA
01050 500A 7F FD12 CLR
01052 500D 108E OFFF LDY
01060 5011 86 0F LDA
01070 5013 87 5004 STA
01080 5016 86 0F LP01 LDA
01090 5018 87 5003 STA
01100 5018 86 0F LP02 LDA
01110 501D 87 5002 STA
01120 5020 86 5002 LP03 LDA
01130 5023 27 12 5037 BEQ
01140 5025 108F FD30 STY
01220 5029 86 5002 LDA
01230 502C 87 FD32 STA
01240 502F 7F FD33 CLR
01250 5032 7F FD34 CLR
01260 5035 20 15 504C BRA
01270 5037 86 5003 LP04 LDA
01280 503A 27 10 504C BEQ
01290 503C 108F FD30 STY
01370 5040 7F FD32 CLR
01380 5043 86 5003 LDA
01390 5046 87 FD33 STA
01400 5049 7F FD34 CLR
01402 504C 31 3F LP05 LEAY
01410 504E 7A 5002 DEC
01420 5051 2A CD 5020 BPL
01430 5053 7A 5003 DEC
01440 5056 2A C3 501B BPL
01450 5058 7A 5004 DEC
01460 505B 2A B9 5016 BPL
01470 505D 86 40 LDA
01480 505F 87 FD12 STA
01482 5062 7F FD37 CLR
01490 5065 39 RTS
01500 5000 END
TOTAL ERRORS 00000--00000 ENTRY
TOTAL WARNINGS 00000--00000

*****
* ハ°レット ^ンゴウ (3) *
* ( LIST 13-10 ) V3.3 *
*****

OPT NOGEN
ORG $5000
BRA LP00
FCB 15
FCB 15
FCB 15
LDA #$70
STA $FD37 DISPLAY PAGE DISABLE
CLR $FD12 640 DOT MODE
LDY #$OFFF
LDA #15
STA GG
LDA #15
STA RR
LDA #15
STA BB
LDA BB
BEQ LP04
STY $FD30 ハ°レット NO.
LDA BB
STA $FD32 BLUE
CLR $FD33 RED
CLR $FD34 GREEN
BRA LP05
LDA RR
BEQ LP05
STY $FD30 ハ°レット NO.
CLR $FD32 BLUE
LDA RR
STA $FD33 RED
CLR $FD34 GREEN
LEAY -1,Y
DEC BB
BPL LP03
DEC RR
BPL LP02
DEC GG
BPL LP01
LDA #$40
STA $FD12 320 DOT MODE
CLR $FD37 DISPLAY PAGE ENABLE
RTS
END ENTRY
PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5065
PROGRAM ENTRY ADDR=5000

```

リスト 13-11 パレット変更(4)

```

00100 *****
00110 *   ハレット \ンコウ (4)   *
00120 *   ( LIST 13-11 )   V3.3   *
00130 *****
01000 OPT      NOGEN
01002 5000      ORG      $5000
01010 5000 20 04 5006 ENTRY BRA LP00
01012 5002      OD      TIMER FCB 13
01020 5003      OF      BB FCB 15
01030 5004      OF      RR FCB 15
01040 5005      OF      GG FCB 15
01050 *
01060 5006 8D 508A LP00 JSR SUBHLT SUB HALT
01070 5009 7F FD93      CLR $FD93 MMR ムコウ
01080 500C 108E OFFF      LDY #$0FFF
01090 5010 86 0F      LDA #15
01100 5012 87 5005      STA GG
01110 5015 86 0F LP01 LDA #15
01120 5017 87 5004      STA RR
01130 501A 86 0F LP02 LDA #15
01140 501C 87 5003      STA BB
01150 501F 86 5003 LP03 LDA BB
01160 5022 27 15 5039 BEQ LP04
01170 5024 10BF FD30      STY $FD30 ハレット NO.
01172 5028 86 5003      LDA BB
01174 502B 5F      CLRBB
01176 502C 1F 01      TFR D,X
01180 502E 8D 506E      JSR TIMING タイミング WAIT
01190 5031 8F FD32      STX $FD32 BLUE,RED
01220 5034 7F FD34      CLR $FD34 GREEN
01240 5037 20 18 5051 BRA LP05
01250 5039 86 5004 LP04 LDA RR
01260 503C 27 13 5051 BEQ LP05
01270 503E 10BF FD30      STY $FD30 ハレット NO.
01272 5042 4F      CLRA
01274 5043 F6 5004      LDB RR
01276 5046 1F 01      TFR D,X
01280 5048 8D 506E      JSR TIMING タイミング WAIT
01290 504B 8F FD32      STX $FD32 BLUE,RED
01320 504E 7F FD34      CLR $FD34 GREEN
01330 5051 31 3F LP05 LEAY -1,Y
01340 5053 7A 5003      DEC BB
01350 5056 2A C7 501F BPL LP03
01360 5058 7A 5004      DEC RR
01370 505B 2A 8D 501A BPL LP02
01380 505D 7A 5005      DEC GG
01390 5060 2A 83 5015 BPL LP01
01400 5062 86 C0      LDA #$C0
01410 5064 87 FD93      STA $FD93 MMR ムコウ
01420 5067 8D 509C      JSR SUBRDY SUB READY REQ.
01430 506A 8D 50A5      JSR SUBMOV SUB HALT カイシヨ
01440 506D 39      RTS
01450 *
01460 506E 86 FD12 TIMING LDA $FD12
01470 5071 85 01      BITA #$01 VSYNC ?
01480 5073 26 14 5089 BNE TM04 VSYNC
01490 5075 86 FD12 TM01 LDA $FD12
01500 5078 85 02      BITA #$02 DISPLAY TIMING ?
01510 507A 26 F9 5075 BNE TM01 DISPLAY
01520 507C 86 FD12 TM02 LDA $FD12
01530 507F 85 02      BITA #$02 DISPLAY TIMING ?
01540 5081 27 F9 507C BEQ TM02 BLANKING
01550 5083 86 5002      LDA TIMER WAIT TIMER SET
01560 5086 4A TM03 DECA
01570 5087 26 FD 5086 BNE TM03 WAIT

```

```

01580  5089 39          TM04  RTS
01590                      *
01600  508A 86  F005    SUBHLT LDA  $F005  << SUB HALT >>
01610  508D 2B  FB  508A BMI  SUBHLT
01620  508F 1A  50      ORCC  #$50
01630  5091 86  80      LDA  #$80
01640  5093 87  F005    STA  $F005
01650  5096 86  F005    LDA  $F005
01660  5099 2A  FB  5096 BPL  *-3
01670  509B 39          RTS
01680  509C F6  FC80    SUBRDY LDB  $FC80  << SUB READY REQ. >>
01690  509F CA  80      ORB  #$80
01700  50A1 F7  FC80    STB  $FC80
01710  50A4 39          RTS
01720  50A5 7F  F005    SUBMOV CLR  $F005  << SUB HALT カイジ"ヨ >>
01730  50A8 1C  AF      ANDCC #$AF
01740  50AA 39          RTS
01750          5000      END  ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=50AA
PROGRAM ENTRY ADDR=5000

```

13-3 VRAM ダイレクトアクセス

FM-7 において VRAM をアクセスするには、

- ① サブシステムコマンドを BIOS によってサブシステムに送り、サブモニタに実行させる。
- ② サブシステムのワークエリアにプログラムを送り、TEST コマンドでそのプログラムを実行する。

のふたつの方法がありました。①の方法は利用しやすいのですが、やや低速です。②の方法は、サブシステムのワークエリアが限られているので多くを望めませんし、サブシステムの深い知識が必要とされます。この VRAM の扱いにくさが、Z80 あるいは i8086 に習熟している人たちに、FM-7 でプログラミングするのをためらわせてきた点だと思われます。

FM77AV では、ダイレクトアクセスにより VRAM をメイン CPU より直接アクセスできるようになっています。これは、Z80 等になれている人には大きな福音であり、ソフト開発者の層を厚くするものと思います。

ダイレクトアクセス時、サブ CPU 空間(64K バイト)は、メイン CPU 空間の物理アドレス \$10000～\$1FFFF の領域を占有します。ですから、MMR をこの物理アドレスに設定して、サブ CPU 空間をアクセスすることができます(図 13-9)。

ダイレクトアクセスモードに設定するには、サブシステムを HALT するだけです。そして念のため MMR を有効にして、MMR にアクセスしたいサブ CPU 空間を設定します。また論理演算回

路が動作していると VRAM をアクセスできませんので、論理演算回路もディセーブルしておきます。それでは、以上の手順のサンプルコーディングをリスト 13-12 に示します。

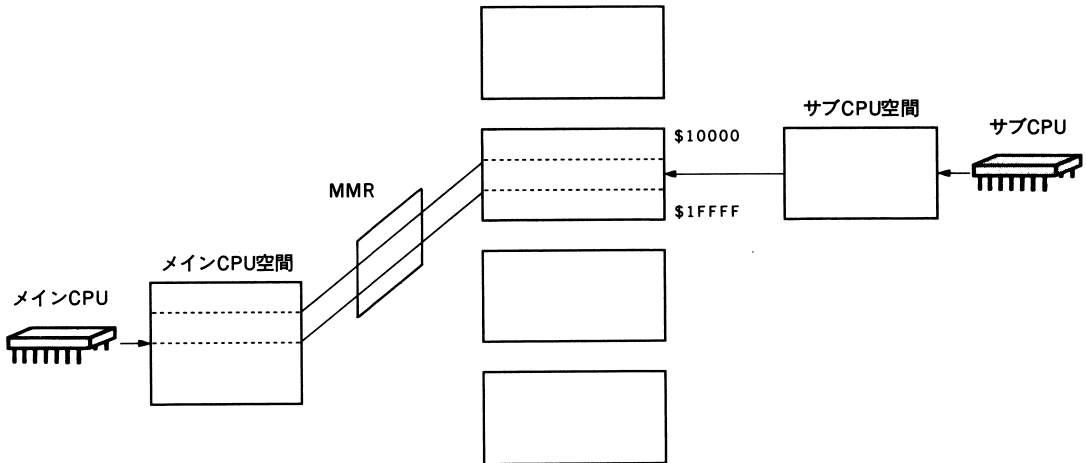


図13-9 ダイレクトアクセス

リスト 13-12 ダイレクトアクセスサンプル

```

01000 *****
01010 *   ダイレクト アクセス サンプル   *
01020 *   ( LIST 13-12 )   V3.3   *
01030 *****
01040
01050      5000      OPT      NOGEN
01060      5000 B0      5047  ENTRY JSR      $5000
01070      5003 B6      FD93      LDA      $FD93
01080      5006 B7      5068      STA      MSRSV      モート セレクト REG. SAVE
01090      5009 8A      80      ORA      #$80      MMR 2700
01100      5008 B7      FD93      STA      $FD93
01110      500E B6      FD80      LDA      $FD80
01120      5011 B7      5069      STA      MMRSV      メモリ マネジメント REG. SAVE
01130      5014 86      10      LDA      #$10
01140      5016 B7      FD80      STA      $FD80      MMR SET
01150      5019 7F      0410      CLR      $0410      ロンリエンサツン DISABLE
01160      501C 86      10      LDA      #$10
01170      501E B7      FD80  LP01  STA      $FD80      MMR SET
01180      5021 8E      0000      LDX      #$0000
01200      5024 E6      84  LP02  LDB      ,X
01201      5026 CA      F0      ORB      #$F0
01202      5028 E7      80      STB      ,X+
01210      502A 8C      E000      CMPX     #$E000
01220      502D 26      FS      5024  BNE      LP02
01230      502F 4C      INCA
01240      5030 81      1C      CMPA     #$1C
01250      5032 26      EA      501E  BNE      LP01
01260      5034 B6      5069      LDA      MMRSV      MMR LOAD
01270      5037 B7      FD80      STA      $FD80
01280      503A B6      5068      LDA      MSRSV      MSR LOAD
01290      503D B7      FD93      STA      $FD93
01300      5040 B0      5059      JSR      RDYREQ      SUB READY REQ.
01310      5043 B0      5062      JSR      SUBMOV      SUB HALT カイシヨ
01320      5046 39      RTS
01330

```

```

01340 5047 B6 FD05 SUBHLT LDA $FD05 << SUB HALT >>
01350 504A 2B FB 5047 BMI SUBHLT
01360 504C 1A 50 ORCC #$50
01370 504E 86 80 LDA #$80
01380 5050 B7 FD05 STA $FD05
01390 5053 B6 FD05 LDA $FD05
01400 5056 2A FB 5053 BPL *-3
01410 5058 39 RTS
01420 5059 F6 FC80 RDYREQ LDB $FC80 << SUB READY REQ. >>
01430 505C CA 80 ORB #$80
01440 505E F7 FC80 STB $FC80
01450 5061 39 RTS
01460 5062 7F FD05 SUBMOV CLR $FD05 << SUB HALT カイジツヨ >>
01470 5065 1C AF ANDCC #$AF
01480 5067 39 RTS
01490 *
01500 5068 0001 MSRSV RMB 1 モード セレクト REG. SAVE
01510 5069 0001 MMRSV RMB 1 メモリ マネジメント REG. SAVE
01520 5000 END ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=5069
PROGRAM ENTRY ADDR=5000

```

13-4 ハードウェア描画機能

FM77AV では、論理演算と直線補間を行なうハードウェア機構を実装しています。それでグラフィックの描画、特に直線をきわめて高速に描くことができます。

論理演算回路の使用方法には、CPU からのアクセスと直線補間からのアクセスの2通りがあります。CPU からアクセスする場合には、論理演算用レジスタ(\$D410～\$D41E)にコマンドを書き込んだ後、ターゲットとする VRAM のアドレスをダミーリードします。するとダミーリードされた VRAM に対してコマンドで指示された演算がなされ、演算後の値が書き込まれます。

直線補間からのアクセスの場合は、コマンドを書き込んだ後、直線補間をスタートさせます。

(1) 論理演算

論理演算コマンド用レジスタを図 13-10 に示します。論理演算回路では、PSET, OR, AND, XOR 等の論理演算だけではなく、タイリングペイントあるいは、色の比較演算も行なうことができます。特にタイリングペイントでは、RGB それぞれに対してパターンを指定できるため、複雑なパターンを描画できます。逆に、点を打ちたいときにもこのタイリングペイントを利用して可能です。

(2) 直線補間

直線補間を使用するには、まず論理演算にコマンドを設定します。それから直線補間用レジスタ(\$D420～\$D42B)に必要なデータをセットします。\$D42B に値が書き込まれた瞬間、直線補間

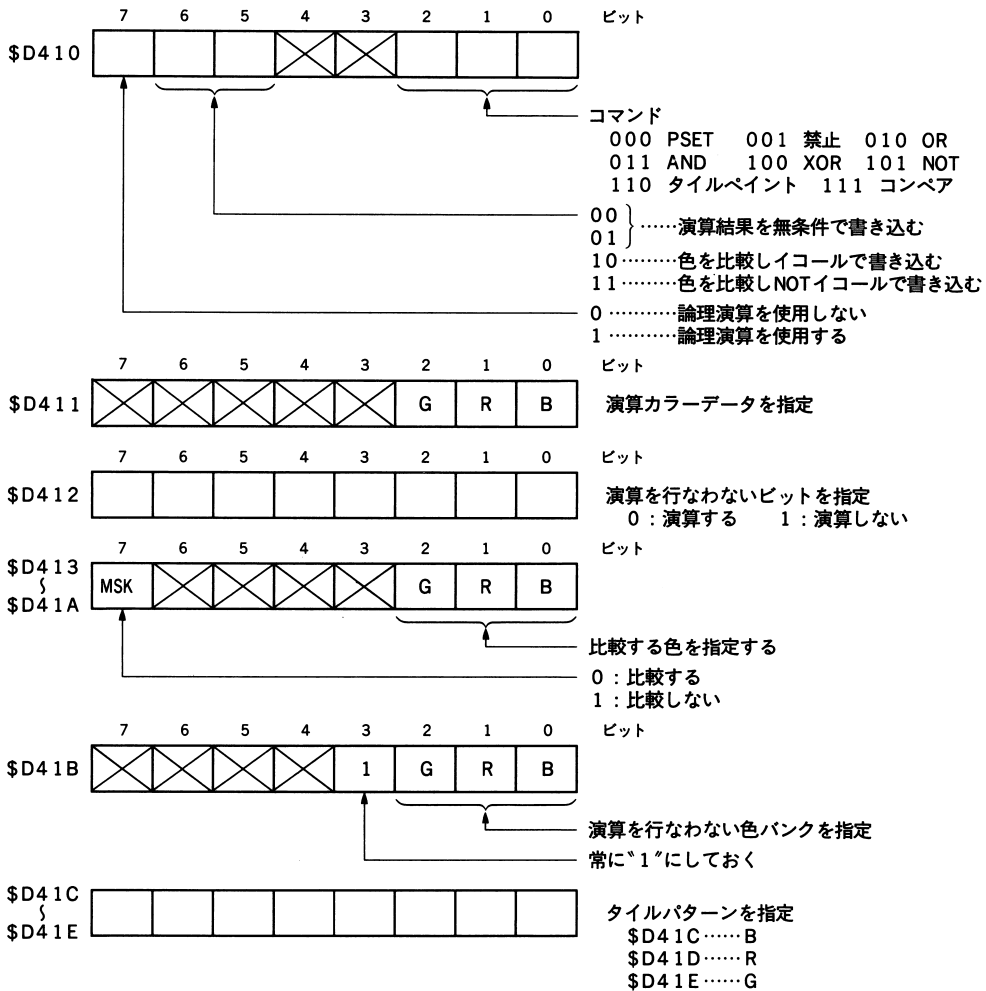


図13-10 論理演算レジスタ

がスタートします。直線補間が動作している間には、CPU は別の仕事をすることが可能です。しかし直線補間が BUSY のときに、直線補間のレジスタや VRAM 系のレジスタを変更してはいけません。

直線補間コマンド用レジスタを図 13-11 に示します。アドレスオフセットレジスタ(\$D420, \$D421)には、VRAM の色バンク内のオフセットを指定するのですが、オフセットを右に 1 ビットシフトした値(2 バイト単位となる)をセットします。通常アドレスオフセットレジスタには 0 を設定しておきます。ただし 4096 色モードのときには色バンクを選択するため、\$0000 または、\$1000 を設定します。直線の色は、論理演算レジスタの演算カラーデータで指定します。ラインスタイルは、ラインスタイルパターンレジスタ(\$D422, \$D423)にて指定しますが、論理演算レジスタのタイルパターンを指定しても可能です。

とにかくこの直線補間を用いると、驚くほど高速に直線を引くことができます。リスト 13-13, リスト 13-14 に直線補間を使用したサンプルプログラムを示しますから、是非実行して確かめてみてください。

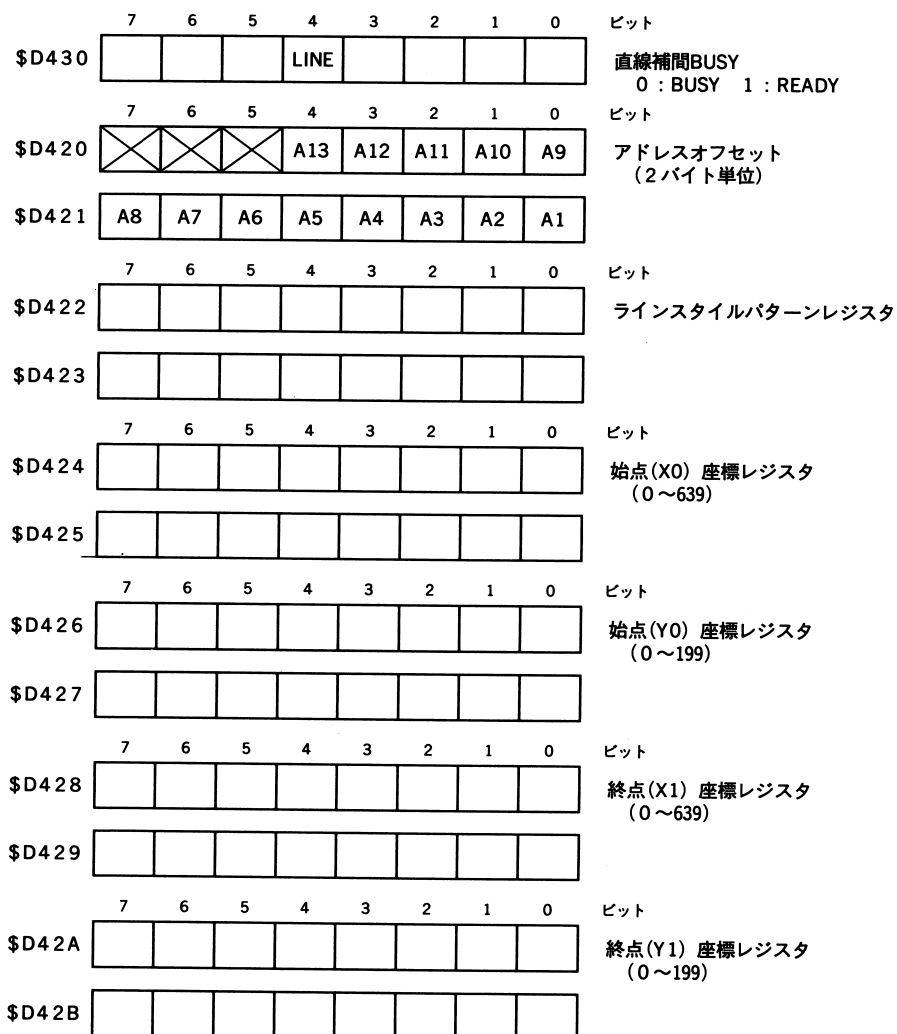


図13-11 直線補間レジスタ

リスト 13-13 直線補間サンプルプログラム

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 20 06 00 01 01 3F 00 63 8D 50 B3 B6 FD 93 8A 80 : 0A
5010 : B7 FD 93 B6 FD 8D B7 50 02 86 1D B7 FD 8D 8E 04 : 06
5020 : 10 86 84 A7 84 B6 50 03 A7 01 6F 02 86 08 A7 0B : A7
5030 : 8E 04 20 6F 84 6F 01 C6 FF E7 02 E7 03 10 8E 50 : 9B
5040 : 04 10 AF 04 10 8E 50 06 10 AF 06 6F 0A 10 8E 00 : C7
5050 : 00 10 AF 08 6F 08 8D 50 AB 31 21 10 8C 02 80 26 : 8F

```

```

5060 : F0 10 8E 00 00 10 AF 0A BD 50 AB 31 21 10 8C 00 : FD
5070 : C8 26 F2 10 8E 02 7F 86 C7 10 AF 08 A7 08 BD 50 : D2
5080 : AB 31 3F 10 8C FF FF 26 EE 10 8E 00 C7 10 AF 0A : F7
5090 : BD 50 AB 31 3F 10 8C FF FF 26 F2 7F 04 10 B6 50 : 43
50A0 : 02 B7 FD 8D BD 50 C5 BD 50 CE 39 B6 04 30 85 10 : 78
50B0 : 27 F9 39 B6 FD 05 28 FB 1A 50 86 80 B7 FD 05 B6 : 16
50C0 : FD 05 2A F8 39 F6 FC 80 CA 80 F7 FC 80 39 7F FD : 44
50D0 : 05 1C AF 39 00 00 00 00 00 00 00 00 00 00 00 : 09
50E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
50F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : C4 05 0E A1 D1 26 BA BF C5 D2 F8 BF 87 EB 42 42 : 2C

```

SAVEM "L13-13M",&H5000,&H5003,&H5000

リスト 13-14 直線補間サンプル実行

```

100 '*****
110 '*   チョクセン ホカン シ"ッコウ   *
120 '*   ( LIST 13-14 )   V3.3   *
122 '*   LIST 13-13   か"   ヒツヨウ テ"ス   *
124 '*****
130 CLEAR ,&H5000:CLS:LOADM "L13-13M"
140 FOR I=1 TO 7
150   POKE &H5003,I
160   EXEC &H5000:EXEC &H5000
170 NEXT
180 FOR I=0 TO 1
190   POKE &H5003,1:POKE &H5004,1:POKE &H5005,63:POKE &H5007,0
200   EXEC &H5000
210   POKE &H5003,2:POKE &H5004,0:POKE &H5005,0:POKE &H5007,99
220   EXEC &H5000
230   POKE &H5003,4:POKE &H5004,1:POKE &H5005,63:POKE &H5007,199
240   EXEC &H5000
250   POKE &H5003,1:POKE &H5004,2:POKE &H5005,127:POKE &H5007,99
260   EXEC &H5000
270   POKE &H5003,2:POKE &H5004,1:POKE &H5005,63:POKE &H5007,0
280   EXEC &H5000
290   POKE &H5003,4:POKE &H5004,1:POKE &H5005,63:POKE &H5007,99
300   EXEC &H5000
310 NEXT

```

13-5 ハードウェアスクロール

VRAM のアドレスは、VRAM アドレスジェネレータによって VRAM に与えられます。VRAM アドレスジェネレータでは、CPU アドレス、CRTC アドレスに対して VRAM オフセットレジスタの値を加算して VRAM アドレスとします。CPU アドレスと CRTC アドレスの両方に加算されるため、常に画面の左上端のアドレスが\$0000 となります。

この VRAM オフセットレジスタを使うと、画面をハードウェアにより高速にスクロールさせることができます。この VRAM オフセットレジスタに 1 を書き込むと、VRAM の\$0001 番地の内容が画面の左上端に表示されます。つまり画面全体が左に 8 ドットスクロールしたことになります。

になります。ですから、0 から 1 ずつふやした値を書き込んでいくと、左に 8 ドットづつスクロールしていきます。これを 80 ずつふやしていくと、上へ 1 ラインづつスクロールすることになります。そして反対に減らしていけば、右スクロール、下スクロールします。

また VRAM オフセットレジスタは、ふたつあり、BANK0, BANK1 それぞれに独立に設定できます。ですから 320 ドットモードでは、BANK0 のデータを上スクロール、BANK1 のデータを右スクロールという方法も実現可能です。

VRAM オフセットレジスタの設定は、まずサブシステム I/O レジスタ \$D430 のビット 5 によって、VRAM のバンクを選択します。その後で、\$D40E, \$D40F に VRAM のオフセットアドレスを設定します。そうすると、選択されたバンクの VRAM オフセットレジスタに値が設定されます。ただし、\$D430 のビット 2 がオフのときには、\$D40F の下位 5 ビットが無効となります。また、VSYNC 期間に VRAM オフセットレジスタの設定を行うと、画面が乱れずにきれいにスクロールします。

リスト 13-15 スクロールサンプル

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6000 : BD 60 5E B6 FD 01 81 1C 27 32 81 1D 27 3F 81 1E : C8
6010 : 27 08 81 1F 27 15 BD 60 77 39 BD 60 89 FC D0 08 : 52
6020 : C3 00 A0 FD D0 08 FD D4 0E 20 EB BD 60 89 FC D0 : 94
6030 : 08 83 00 A0 FD D0 08 FD D4 0E 20 DA BD 60 89 FC : 7B
6040 : D0 08 83 00 01 FD D0 08 FD D4 0E 20 C9 BD 60 89 : 9F
6050 : FC D0 08 C3 00 01 FD D0 08 FD D4 0E 20 B8 BD 60 : 41
6060 : 91 B6 FD 93 8A 80 B7 FD 93 86 1D B7 FD 80 7F D4 : 5F
6070 : 10 86 04 B7 D4 30 39 7F D4 10 7F D4 30 86 3D B7 : EE
6080 : FD 8D BD 60 A3 BD 60 AC 39 B6 D4 30 85 04 27 F9 : AF
6090 : 39 B6 FD 05 2B FB 1A 50 86 80 B7 FD 05 B6 FD 05 : FB
60A0 : 2A FB 39 F6 FC 80 CA 80 F7 FC 80 39 7F FD 05 1C : 63
60B0 : AF 39 00 00 00 00 00 00 00 00 00 00 00 00 00 : EB
60C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
60D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
60E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
60F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 2B 76 FE DA 1A D4 44 1D A2 32 D2 33 EC 63 D8 80 : 4B

```

SAVEM "L13-15M",&H6000,&H60B1,&H6000

リスト 13-16 スクロール実行

```

10 *****
20 *      スクロール シツコウ      *
30 *      ( LIST 13-16 )      V3.3      *
40 *      LIST 13-13   か"   ヒツヨウ テ"ス      *
50 *      LIST 13-15   か"   ヒツヨウ テ"ス      *
60 *****
70 CLEAR ,&H5000:SCREEN @0:CLS
80 LOADM "L13-13M":EXEC &H5000:'   チョクセン   ホカン
90 LOADM "L13-15M"
100 EXEC &H6000
120 GOTO 100

```

それでは、実際に画面をスクロールさせてみましょう。リスト 13-16、リスト 13-15 を実行させてみてください。カーソルキーに従って左右、上下方向にスクロールします。

しかし、ここで注意する点がふたつあります。ひとつは、上または下スクロールする際現れる黒い横線です。これは、未表示 VRAM が現れてきているのです。VRAM は、640 ドットモードのとき、RGB それぞれ 16K ($4096 \times 4 = 16384$ バイト) あります。そして表示されているのは、 $80 \times 200 = 16000$ バイトです。ですから残りの 384 バイトが、未表示 VRAM となっているのです(図 13-12)。一方 320 ドットモード時には、各色バンクに対して 192 バイトの未表示 VRAM が存在します。ところが VRAM オフセットレジスタをセットしたために、この未表示 VRAM が現れてきてしまったのです。本格的なスクロールプログラムをつくるには、この未表示 VRAM を処理してやる必要があります。

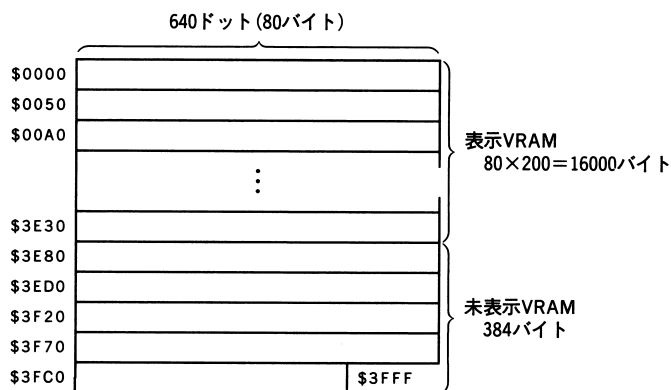


図13-12 未表示VRAM

また、たとえば左スクロールをつづけていると、だんだん画面が上にあがってきてしまいます。つまり 640 ドット (80 バイト) 左スクロールするということは、1 ライン上スクロールすることと同じなのです。これも困ったことです(図 13-13)。

それで、以上の点を考慮して上下左右スクロールプログラムをつくってみました。リスト 13-17、リスト 13-18 です。カーソルキーに従って、上下左右にスクロールします。今度はリスト 13-15 のような不具合はなく、きれいに画面が連続してスクロールします。

上スクロールさせるときには、VRAM オフセットレジスタに値をセットする前に、画面上部の 2 ラインを未表示 VRAM の先頭に送ってやります。それからオフセットレジスタに 160 加算してやると、画面上部より消えた 2 ラインが、そっくりそのまま画面下方から出てきます。下スクロールさせるときには、その逆をします。

左スクロールさせるときには、画面左端の 8 ドットを 1 ライン (80 バイト分) 下方にずらしておいてから、スクロールさせます。そうすると、ずらしたドットが画面右側に 1 ラインあがって出てきます。つまり、最初と同じラインの画面右側に表示されるわけです。これによって、画面が

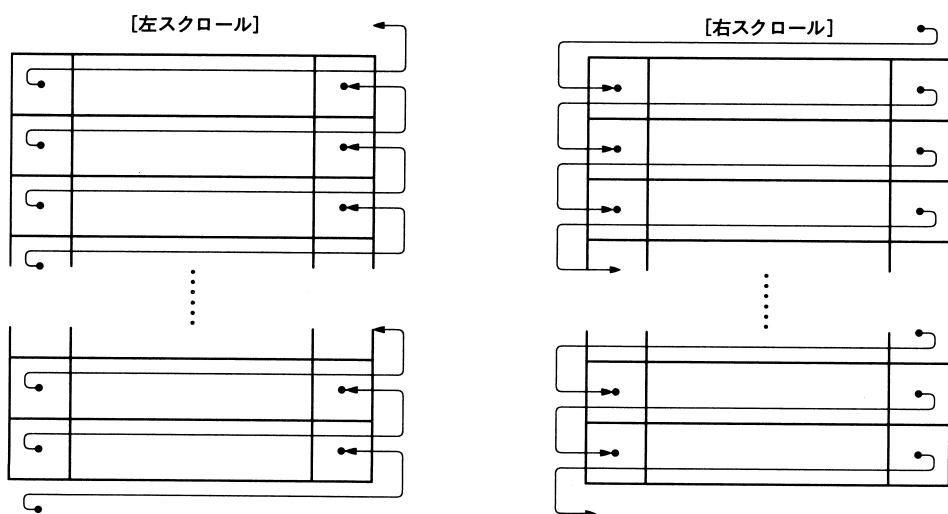


図13-13 左右スクロール

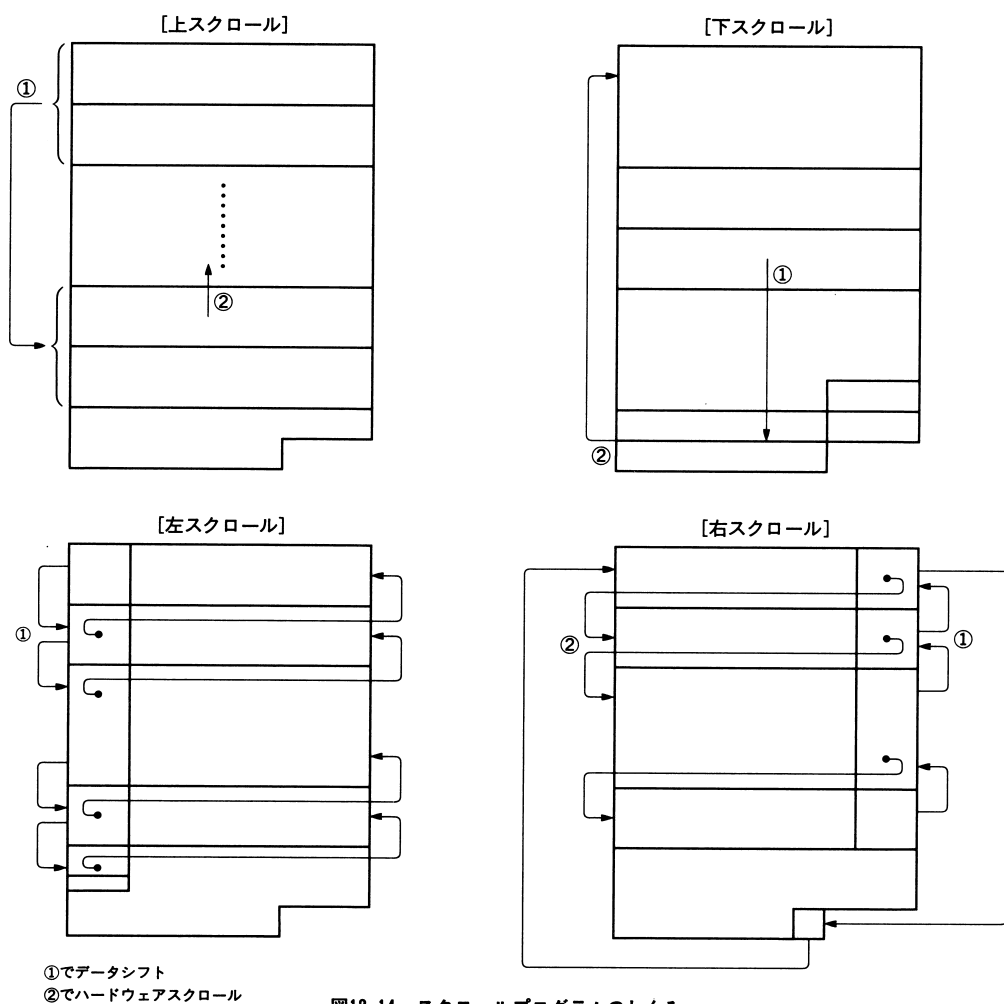


図13-14 スクロールプログラムのしくみ

だんだん上にスクロールしてしまふことがなくなります。右スクロールの場合には、やはり逆の処理をしておきます(図 13-14)。

リスト 13-17 スクロールプログラム

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6000 : 20 05 00 00 00 00 00 8D 60 89 B6 FD 01 81 1C 27 : 43
6010 : 4A 81 1D 27 5D 81 1E 27 0C 81 1F 27 23 81 20 26 : EF
6020 : E9 BD 60 B4 39 8E 90 00 10 8E CE 80 BD 60 D3 BD : AA
6030 : 61 4F FC D0 08 C3 00 A0 FD D0 08 FD D4 0E 20 CA : 85
6040 : 8E CD E0 10 8E CF 60 BD 60 D3 BD 61 4F FC D0 08 : 39
6050 : 83 00 A0 FD D0 08 FD D4 0E 20 AF 8E 90 4F BD 61 : 31
6060 : 1F BD 61 4F FC D0 08 83 00 01 FD D0 08 FD D4 0E : 98
6070 : 20 98 8E CE 30 BD 60 F7 BD 61 4F FC D0 08 C3 00 : 5C
6080 : 01 FD 00 08 FD D4 0E 20 81 BD 61 57 B6 FD 93 BA : 9B
6090 : 80 B7 FD 93 FC FD 89 FD 60 02 FC FD 88 FD 60 04 : 8D
60A0 : B6 FD 8D 87 60 06 86 1D B7 FD 8D 7F D4 10 86 04 : 2E
60B0 : B7 D4 30 39 7F D4 10 7F D4 30 FC 60 02 FD FD 89 : BB
60C0 : FC 60 04 FD FD 88 B6 60 06 B7 FD 8D BD 61 69 BD : 86
60D0 : 61 72 39 CE 10 11 C6 03 34 74 FF FD 89 33 C9 02 : EF
60E0 : 02 FF FD 88 C6 A0 A6 80 A7 A0 5A 26 F9 35 74 33 : B1
60F0 : C9 04 04 5A 26 E2 39 CE 10 11 C6 03 34 54 FF FD : AB
-----
[cs] : 1A 0E 80 10 F9 FF FB F9 01 85 65 42 F6 E4 6E 55 : 9E

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
6100 : 89 33 C9 02 02 FF FD 88 C6 C8 A6 84 A7 88 50 30 : 77
6110 : 88 80 5A 26 F5 35 54 33 C9 04 04 5A 26 DE 39 CE : 9F
6120 : 10 11 C6 03 34 54 FF FD 89 33 C9 02 02 FF FD 88 : 7E
6130 : A6 84 B7 CF FF 30 88 50 C6 C7 A6 84 A7 88 80 30 : 7D
6140 : 88 50 5A 26 F5 35 54 33 C9 04 04 5A 26 D6 39 B6 : 1F
6150 : D4 30 85 04 27 F9 39 B6 FD 05 2B F8 1A 50 86 80 : 34
6160 : B7 FD 05 B6 FD 05 2A FB 39 F6 FC 80 CA 80 F7 FC : 7E
6170 : 80 39 7F FD 05 1C AF 39 00 00 00 00 00 00 00 00 : 3E
6180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
6190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
61F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : 5A 2E 03 D7 48 07 3E 28 DD C5 44 39 80 93 EC EB : 20

```

SAVEM "L13-17M",&H6000.&H6177.&H6000

リスト 13-18 スクロール実行

```

10 *****
20 * スロール シ"ゴウ (2) *
30 * ( LIST 13-18 ) V3.3 *
40 * LIST 13-13 カ" ヒツヨ チ"ス *
50 * LIST 13-17 カ" ヒツヨ チ"ス *
60 *****
70 CLEAR &H5000:SCREEN 80:CLS
80 LOADM "L13-13M":EXEC &H5000:' チョクセシ ホカシ
90 LOADM "L13-17M"
100 EXEC &H6000

```

このスクロールプログラムでは、画面から消えたデータがそのまま反対側の画面に出てくるのですが、これを MAP に従った別のデータをセットするようにすれば、ゼビウス型のスクロールゲームができます。是非とも一度工夫してみてください。

13-6 メモリ管理

FM77AV では、MMR(Memory Management Register)と TWR(Text Window Register)を採用していて、最大 256KB のメモリ空間を管理することができます。

(1) MMR

FM77AV のメモリマッピング機能では、MMR を介して 64KB の CPU 空間を、4KB 単位に物理アドレスに対応させることによって、最大 256KB のメモリ空間(\$00000~\$3FFFF 番地)を管理することができます。

メモリマッピング機能は、64(16×4)個の MMR と MSR(MMR Segment Register)の 2 つのレジスタによって制御されます。MMR は、16 バイトのレジスタ群の 4 セグメントによって構成され、使用されるセグメントの選択は、MSR の内容によります(図 13-15)。

MMR の第 0 バイト目は、CPU 空間の\$0000~\$0FFF 番地、第 1 バイト目が\$1000~\$1FFF 番地、……のように、4KB 単位で CPU 空間に対応します。ただし、CPU 空間の\$FC00~\$FFFF の 1KB の空間は、常駐空間であり、MMR の値に関わりなく常に物理アドレス\$3FC00~\$3FFFF 番地に対応します。つまり、第 15 バイト目の MMR は、CPU 空間の\$F000~\$FBFF 番地の 3KB の空間にのみ対応することになります(図 13-16)。

CPU から出力される 16 本のアドレス線のうち、下位 12 ビット(A0~A11)は直接メモリへ、上位 4 ビット(A12~A15)は、MMR の下位 4 ビットに接続されます。また MSR の 2 ビット(S0, S1)は、MMR の上位 2 ビットに接続され、セグメントのひとつを決定します。そして MSR によ

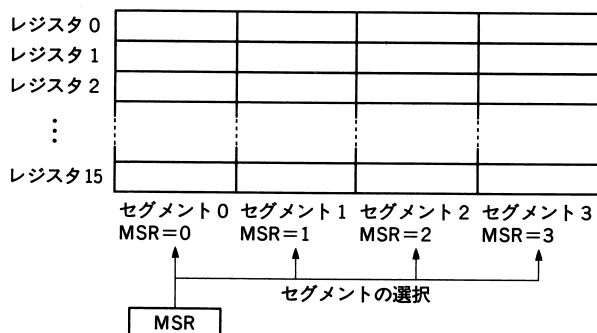


図13-15 MMRとMSR

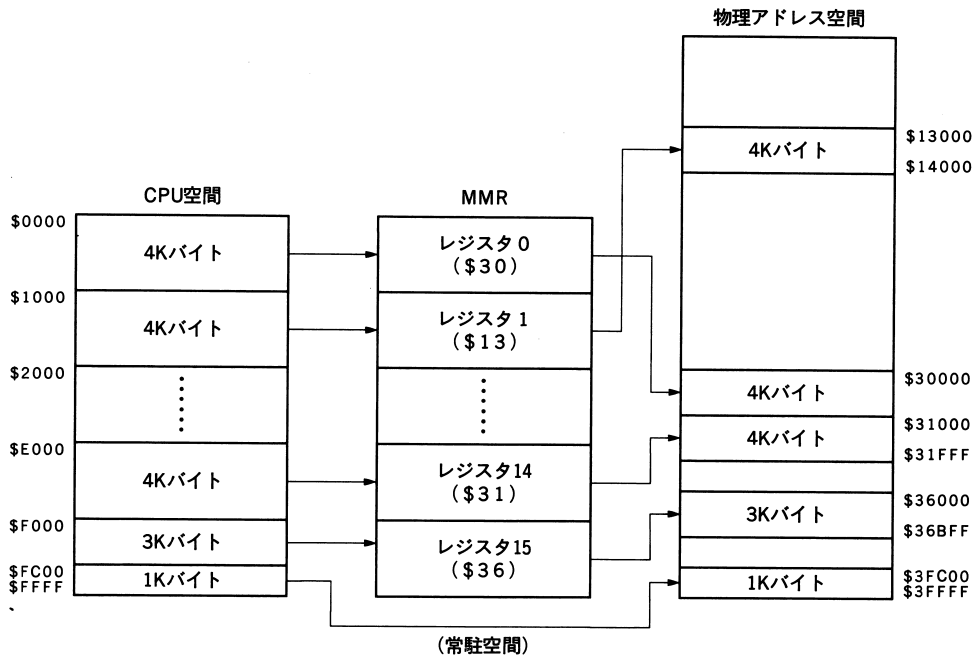


図13-16 メモリマッピング機能

って決定されるセグメント内で，CPU からの A12～A15 によって選ばれるレジスタの内容が，MMR の出力 6 ビットとなり，メモリの A12～A17 のアドレスとなります(図 13-17)。

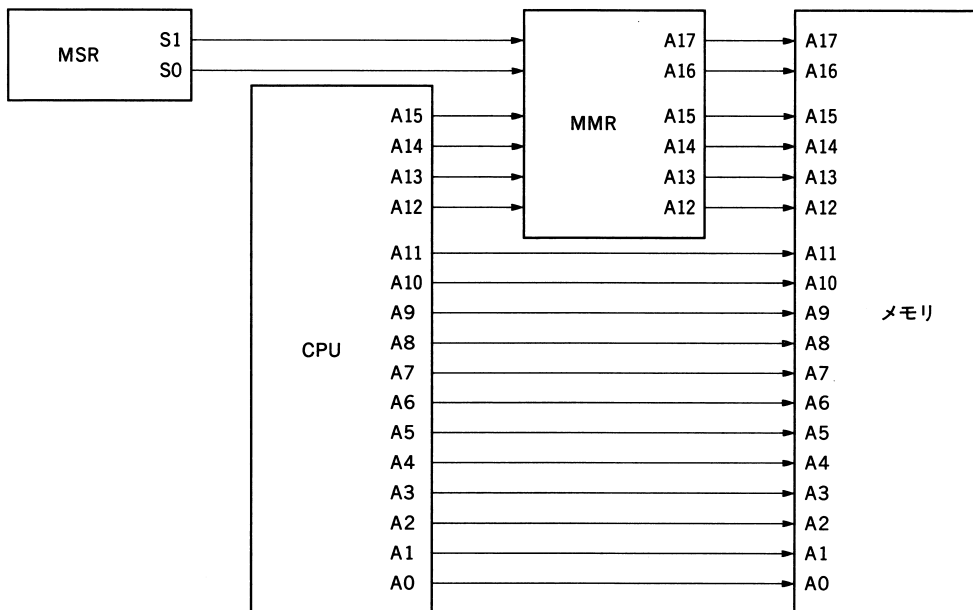


図13-17 MMR, MSRとメモリとの関係

F-BASIC V3.3 が起動したときの MMR の設定値は、図 13-18 の様です。MSR は 0 ですの
で、セグメント 0 の MMR が選択されています。

MMR の設定値を変更するには、ふたつの方法があります。ひとつは、現在選択されている
MMR セグメント内の一部のレジスタだけを変更することです。そしてもうひとつは、MSR を変
更して MMR セグメントを変更してしまうことです。前者は、メインシステム I/O レジスタ
(\$FD80～\$FD8F)に値を書き込むだけで可能です。一方、MSR を変更するには、\$FD90 に値を
設定します。MSR を変更すると、16 個の MMR がすべて一度に切り換わってしまいます。です
から MSR の変更は、慎重に行なう必要があります。少なくとも現在動作しているプログラムおよ
びスタック等が、MSR の変更によっても同一のアドレスになるように MMR を初期設定してお
く必要があります。なお、MMR の初期設定は MMR の初期値が不定のため、\$FD93 のビット 7
をオフにして MMR を無効にしてから行なうべきです。MMR を無効にすると、F-BASIC V3.
0 と同一の CPU 空間(物理アドレス\$30000～\$3FFFF)となります。

それでは、MMR 設定のサンプルとして、図 13-19 のように MMR を初期設定するプログラム
をリスト 13-19 に示します。セグメント 0 には、F-BASIC V3.3 起動時と同一の MMR、セグ
メント 1 がダイレクトアクセス用の MMR です。そしてセグメント 2 が拡張メモリアクセス用、
セグメント 3 が RAM 空間アクセス用の MMR となっています。

最後に MMR 使用における注意を述べます。それは、MMR 有効時の CPU クロックが 2MHz
ではなく 1.6MHz に落ちるということです。逆に言えば、MMR を使用する必要のないプログラ
ムの実行にあたって、MMR を無効にしておけば、処理速度が確実に 20 % UP します。ですから、
本当に MMR のメモリマッピングが必要なときだけに MMR を有効とするようなプログラミン
グをおすすめします。

	セグメント 0	セグメント 1	セグメント 2	セグメント 3
レジスタ 0	\$30	\$30	\$30	\$30
1	\$31	\$31	\$31	\$31
2	\$32	\$32	\$32	\$32
3	\$33	\$33	\$33	\$33
4	\$34	\$34	\$34	\$34
5	\$35	\$35	\$35	\$35
6	\$36	\$36	\$36	\$36
7	\$37	\$37	\$37	\$37
8	\$04	\$38	\$38	\$38
9	\$39	\$39	\$39	\$39
10	\$3A	\$3A	\$3A	\$3A
11	\$3B	\$3B	\$3B	\$3B
12	\$3C	\$3C	\$3C	\$3C
13	\$3D	\$3D	\$3D	\$3D
14	\$3E	\$3E	\$3E	\$3E
15	\$3F	\$3F	\$3F	\$3F

図13-18 F-BASIC V3.3起動時のMMR

レジスタ 0	\$30	\$10	\$20	\$00
1	\$31	\$11	\$21	\$01
2	\$32	\$12	\$22	\$02
3	\$33	\$13	\$23	\$03
4	\$34	\$34	\$34	\$34
5	\$35	\$35	\$35	\$35
6	\$36	\$36	\$36	\$36
7	\$37	\$37	\$37	\$37
8	\$04	\$38	\$38	\$38
9	\$39	\$39	\$39	\$39
10	\$3A	\$3A	\$3A	\$3A
11	\$3B	\$3B	\$3B	\$3B
12	\$3C	\$3C	\$3C	\$3C
13	\$3D	\$1D	\$3D	\$3D
14	\$3E	\$3E	\$3E	\$3E
15	\$3F	\$3F	\$3F	\$3F

図13-19 MMR初期設定

リスト 13-19 MMR 初期設定

```

01000 *****
01010 *      MMR INITIALIZE      *
01020 *      ( LIST 13-19 )      V3.3      *
01030 *****
01040                      OPT      NOGEN
01050      5000                      ORG      $5000
01060      5000 B6      FD93      ENTRY      LDA      $FD93
01070      5003 B4      3F                      ANDA      #3F      MMR 1707
01080      5005 B7      FD93                      STA      $FD93
01090      5008 4F                      CLRA
01100      5009 8E      502E                      LDX      #MMRDT
01110      500C B7      FD90      LP01      STA      $FD90      MMR セクメント REG.
01120      500F 108E      FD80                      LDY      #FD80
01130      5013 E6      80      LP02      LDB      ,X+
01140      5015 E7      A0                      STB      ,Y+      メモリ マネジメント REG.
01150      5017 108C      FD90                      CMPY      #FD90
01160      5018 26      F6      5013                      BNE      LP02
01170      501D 4C                      INCA
01180      501E 81      04                      CMPA      #4
01190      5020 26      EA      500C                      BNE      LP01
01200      5022 7F      FD90                      CLR      $FD90
01210      5025 B6      FD93                      LDA      $FD93
01220      5028 8A      C0                      ORA      #C0      MMR 1707
01230      502A B7      FD93                      STA      $FD93
01240      502D 39                      RTS
01250      502E      30      MMRDT      FCB      $30,$31,$32,$33      MSR=0
01251      5032      34                      FCB      $34,$35,$36,$37
01260      5036      04                      FCB      $04,$39,$3A,$3B
01261      503A      3C                      FCB      $3C,$3D,$3E,$3F
01270      503E      10                      FCB      $10,$11,$12,$13      MSR=1
01271      5042      34                      FCB      $34,$35,$36,$37
01280      5046      38                      FCB      $38,$39,$3A,$3B
01281      504A      3C                      FCB      $3C,$10,$3E,$3F
01290      504E      20                      FCB      $20,$21,$22,$23      MSR=2
01291      5052      34                      FCB      $34,$35,$36,$37
01300      5056      38                      FCB      $38,$39,$3A,$3B
01301      505A      3C                      FCB      $3C,$3D,$3E,$3F
01310      505E      00                      FCB      $00,$01,$02,$03      MSR=3
01311      5062      34                      FCB      $34,$35,$36,$37

```

```

01320 5066      38          FCB   $38,$39,$3A,$3B
01321 506A      3C          FCB   $3C,$3D,$3E,$3F
01330          5000          END   ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=506D
PROGRAM ENTRY ADDR=5000

```

(2) TWR

ウィンドウ機能は、主として F-BASIC がユーザープログラムテキストを管理するのに使われます。ウィンドウ機能では、TWR によって CPU 空間の \$7C00～\$7FFF のウィンドウ領域を \$00000～\$0FFFF 番地の領域に 256 バイト単位にマッピングさせます。

このマッピングは、ウィンドウ領域に割り当てる物理アドレスを、アドレス \$7C00 に対する 256 バイト単位のオフセット値で書き込むことにより行ないます。たとえば、ウィンドウ領域に物理アドレス \$08800～\$08BFF 番地を割り当てるには、TWR に \$0C を書き込みます (\$7C00+\$0C×256=\$8800)。またウィンドウ領域内のメモリアドレスは、\$00000～\$0FFFF 番地内でループしています。ですから TWR に \$82 を書き込んだ場合のウィンドウ領域と物理アドレスとの対応は、図 13-20 のようになります。この場合、\$7C00 番地をアクセスすると、それは \$0FE00 番地がアクセスされます。そして \$7E00 番地をアクセスすると、\$00000 番地がアクセスされます。

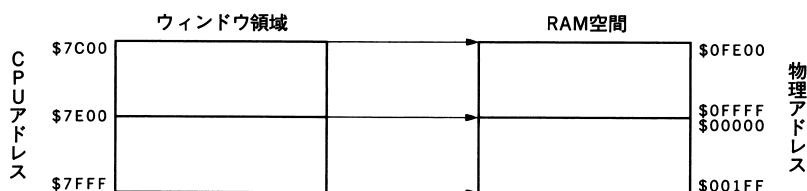


図13-20 TWRと物理アドレスとの対応

TWR に値をセットするには、メインシステムレジスタ \$FD92 に値を書き込みます (図 13-21)。F-BASIC では、必要に応じてこの TWR を書き換えて処理に必要な BASIC テキストをウィンドウ領域にうつしだして、プログラムの実行を行なっていきます。この TWR を利用者が意識する必要は、通常ないのですが、何らかの理由で RAM 空間をアクセスする必要があるときには、この TWR を利用して容易にアクセスできます。またウィンドウ機能が有効になっているときには、ウィンドウ領域 (\$7C00～\$7FFF) のメモリマッピングは、MMR に対して優先します。

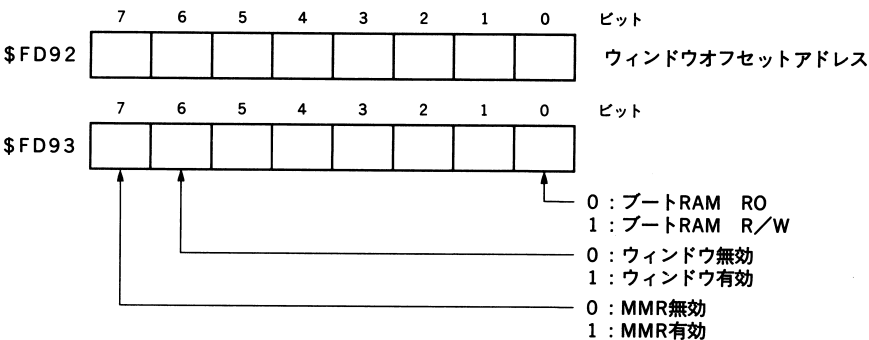


図13-21 TWR

13-7 AVTV 制御

FM77AV は、スーパーインポーズ機能とビデオデジタイズ機能という 2 つの AVTV 制御機能を持っています。スーパーインポーズ機能では、テレビ/ビデオ画面とパソコン画面を合成して表示させることができます。ビデオデジタイズ機能では、ビデオデジタイズカードから入力したビデオ入力信号を、デジタル化し、リアルタイムに表示しながら VRAM に画像データを取り込むことができます。

この AVTV を制御するための 4 つの方法が、用意されています。

- ① キーボード (PF7～PF10) から入力する。
- ② F-BASIC V3.3 の SIMPOSE 文, SINPUT 文を利用する。
- ③ サブシステムコマンドの TELEVISION CONTROL, DIGITIZE を利用する。
- ④ キーボードエンコーダに直接コマンドを送る。

AVTV 制御のブロック図を図 13-22 に示します。AVTV を制御しているのは、最終的にはキーボードエンコーダです。しかし、このキーボードエンコーダを直接制御するのは、メリットがなくやめた方がよいでしょう。サブシステムに AVTV 制御のすべてのコマンドが揃っているからです。

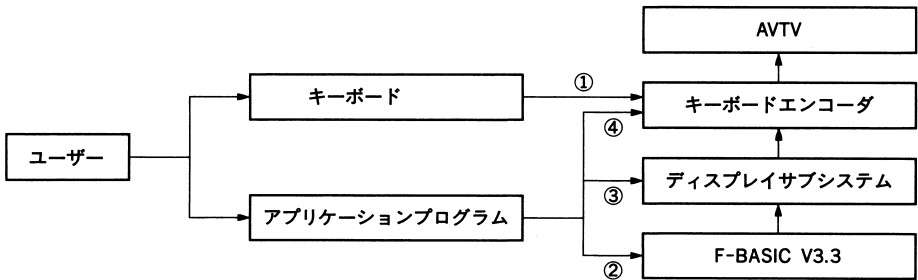


図13-22 AVTV制御ブロック図

13-7-1 スーパーインポーズ機能

スーパーインポーズ時には、次の4つの画面の状態があり、ソフトウェアによって切り換えることができます。

- ① パソコンモード……………パソコン画面のみが表示されます。音声は、パソコンのサウンド出力となります。
- ② テレビモード……………テレビの放映画面または、VTR, VD などの画面のみが表示されます。このときテレビ画面の輝度を、高輝度、低輝度の2段階に切り換えることができます。音声は、テレビ映像の音声となります。
- ③ スーパーインポーズ……上記のパソコンモード、テレビモードの画面および音声が合成されて出力されます。パソコン画面の表示の方が優先度が高く、透明色 (RGB の各輝度が 0) として指定された色の部分にテレビ画面が表示されます。テレビ画面の輝度を低輝度にするとパソコン画面が見やすくなります。
- ④ デジタイズモード……………表示はテレビモードの場合と同様です。しかしデジタイズモードのときには、TV 画面を表示すると同時に、VRAM にも TV 画面をデジタル化したデータが書き込み続けられます。ですから、このモードから他のモードに切り換えると、その直前の TV 画面が VRAMに残った状態となります。

デジタイズモードは、表示画面の1つの状態です。後述のデジタイズ機能は、TV 画面を VRAM に取り込むコマンド(命令)です。

スーパーインポーズ機能の選択方法について、図 13-23 にまとめます。

		キーボード	F-BASIC V3.3	サブシステムコマンド
パソコンモード		[SHIFT] + [PF 7]	SIMPOSE 0	TELEVISION CONTROL コマンド
スーパーインポーズモード	高輝度	[SHIFT] + [PF 8]	SIMPOSE 1,0	TELEVISION CONTROL コマンド
	低輝度	[SHIFT] + [PF 9]	SIMPOSE 1,1	TELEVISION CONTROL コマンド
TVモード	高輝度	[SHIFT] + [PF10]	SIMPOSE 2,0	TELEVISION CONTROL コマンド
	低輝度	————	SIMPOSE 2,1	TELEVISION CONTROL コマンド
デジタイズモード		————	SIMPOSE 3	TELEVISION CONTROL コマンド

図13-23 スーパーインポーズ機能制御

13-7-2 ビデオデジタイズ機能

ビデオデジタイズ機能は、ビデオデジタイズカードと専用カラー CRT テレビ-15 を使用することにより実現できます。この機能により AV テレビのテレビ画像あるいは、VTR、VD などのビデオ画像をデジタイズして VRAM に取り込むことができます。

このビデオデジタイズ機能は、キーボードからは指定できません。F-BASIC V3.3 では、SINPUT 文にて指定します。そしてサブシステムには、DIGITIZE コマンドが用意されています。

ビデオデジタイズ機能が指定されると、デジタイズモードと同様に VRAM に TV 画面が取り込み続けられます。そして任意のキーが押されると元の画面モードに復帰し、VRAM には、キーが押された瞬間の TV 画面が残ります。なお、このとき押されたキーは、キー入力バッファに登録されないでキー入力されません。

つまりこのビデオデジタイズ機能の動作は、デジタイズモードを利用した次のプログラムと、ほぼ等価です。

```
SIMPOSE 3:A$=INPUT$(1):SIMPOSE 0
```

13-7-3 VRAM のセーブ/ロード

ビデオデジタイズ機能により VRAM に取り込まれた画面データを、フロッピーディスクにセーブ/ロードするプログラムを紹介します。要するに 96KB の VRAM すべてをフロッピーディスクにセーブ/ロードすればよいわけですので、BASIC の SAVEM 文、LOADM 文を使うことにします。ただしダイレクトアクセスモードにして MMR の設定が必要です。リスト 13-20 がセーブ用、リスト 13-21 がロード用です。

リスト 13-20 を RUN して、ファイル名(ドライブ番号指定可)を入力します。すると SINPUT 文にてビデオデジタイズ機能が設定されますから、任意の画面でキーを押してください。キーを押した瞬間の画面が、フロッピーディスクにセーブされます。データファイルは、各色バンク毎の 12 個のファイルに分割してセーブされます。1 枚のフロッピーディスクには、V3.0 システム付きでは 3 画面分、V3.3 システム付きでは 2 画面分がセーブ可能です。

リスト 13-20 VRAM のセーブ

```
100 *****
110 '*   VRAM SAVE ( DIGITIZE )           *
120 '*   ( LIST 13-20 )       V3.3       *
122 *****
130 CLEAR .&H4000:SCREEN0 1:CLS
132 ON ERROR GOTO 540
140 LOCATE 5,10:LINE INPUT "FILE NAME =":F$
150 CLS:SINPUT
160 GOSUB 460
170 POKE &HFD85,&H1D:POKE &H5410,0
190 FOR I1=0 TO 1
200   POKE &HFD85,&H1D:POKE &H5430,I1*&H60
210   FOR I2=0 TO 5
```

```

220     POKE &HFD85,&H10+I2*2:POKE &HFD86,&H10+I2*2+1
230     SAVEM F$+RIGHT$(STR$(I1*6+I2),2),&H5000,&H6F3F,&H5000
240     NEXT
250 NEXT
260 POKE &HFD85,&H35:POKE &HFD86,&H36
270 ON ERROR GOTO 0
280 GOSUB 500
290 CLS:LOCATE 5,10:PRINT "Hit Any Key!":A$=INPUT$(1)
300 GOSUB 460
310 POKE &HFD85,&H10:POKE &H5410,0
320 ON ERROR GOTO 540
330 FOR I1=0 TO 1
340     POKE &HFD85,&H10:POKE &H5430,I1*&H60
350     FOR I2=0 TO 5
360         POKE &HFD85,&H10+I2*2:POKE &HFD86,&H10+I2*2+1
370         LOADM F$+RIGHT$(STR$(I1*6+I2),2)
380     NEXT
390 NEXT
400 POKE &HFD85,&H35:POKE &HFD86,&H36
410 ON ERROR GOTO 0
420 GOSUB 500
430 A$=INPUT$(1)
450 END
452 '***** SUB HALT *****
460 D=PEEK(&HFD05):IF D>127 THEN 460
470 POKE &HFD05,CINT(D) OR &H80
480 D=PEEK(&HFD05):IF D<128 THEN 470
490 RETURN
492 '***** SUB HALT カイシ"ヨ *****
500 POKE &HFC82,&HFF
510 D=PEEK(&HFD05):POKE &HFD05,CINT(D) AND &H7F
520 D=PEEK(&HFD05):IF D>127 THEN 520
530 RETURN
540 GOSUB 500
550 ON ERROR GOTO 0
560 END

```

リスト 13-21 VRAM へのロード

```

100 '*****
110 '*   VRAM LOAD ( DIGITIZE )           *
120 '*   ( LIST 13-21 )   V3.3           *
122 '*****
130 CLEAR ,&H4000:SCREEN0 1:CLS
132 ON ERROR GOTO 540
140 LOCATE 5,10:LINE INPUT "FILE NAME =":F$
300 GOSUB 460
310 POKE &HFD85,&H10:POKE &H5410,0
320 ON ERROR GOTO 540
330 FOR I1=0 TO 1
340     POKE &HFD85,&H10:POKE &H5430,I1*&H60
350     FOR I2=0 TO 5
360         POKE &HFD85,&H10+I2*2:POKE &HFD86,&H10+I2*2+1
370         LOADM F$+RIGHT$(STR$(I1*6+I2),2)
380     NEXT
390 NEXT
400 POKE &HFD85,&H35:POKE &HFD86,&H36
410 ON ERROR GOTO 0
420 GOSUB 500
430 A$=INPUT$(1)
450 END
452 '***** SUB HALT *****
460 D=PEEK(&HFD05):IF D>127 THEN 460
470 POKE &HFD05,CINT(D) OR &H80
480 D=PEEK(&HFD05):IF D<128 THEN 470

```

```

490 RETURN
492 '***** SUB HALT カイシヨ *****
500 POKE &HFC82,&HFF
510 D=PEEK(&HFD05):POKE &HFD05,CINT(D) AND &H7F
520 D=PEEK(&HFD05):IF D>127 THEN 520
530 RETURN
540 GOSUB 500
550 ON ERROR GOTO 0
560 END

```

13-7-4 アナログパレット設定テクニック

デジタイズモードのとき、デジタイズカードによってデジタル化された画面データは、アナログパレットを通してから表示されるので、アナログパレットを一種のデジタルフィルターとして使用できます。

たとえばアナログパレットの色の設定を、通常の逆の重みづけで設定しておけば、写真のネガのような映像をリアルタイムで楽しむことができます。さらに VTR などで輝度が不足しているような画面でも、パレットの輝度をシフトしてやることにより、見やすい画面を再生できます。テレビのクイズ番組のようなテクニカルな表示も楽しめます。このようにアナログパレットの設定に工夫することで、一風変わった画面を楽しむことができます。いろいろ工夫してみてください。

それでは、次にパレット設定のサンプルを示します。自分でパレット設定を工夫する際の参考にしてください。リスト 13-22 を入力してリスト 12-23 を実行し表示番号(1~6)を入力してください。

- ① 4096 色表示
- ② 256 色表示
- ③ 64 色表示
- ④ 8 色表示
- ⑤ 単色(白黒)表示
- ⑥ ネガ表示

リスト 13-22 アナログパレット設定あれこれ

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5000 : 20 08 00 09 00 00 00 00 00 00 7F FD 93 BD 51 1E : 6C
5010 : 86 50 02 4A 27 21 4A 27 2B 4A 27 37 4A 27 41 4A : DA
5020 : 27 76 4A 27 5C 4A 27 03 4A 27 00 BD 51 30 BD 51 : 9B
5030 : 39 86 C0 B7 FD 93 39 86 0F B7 50 04 B7 50 05 B7 : 62
5040 : 50 06 20 29 86 0C B7 50 04 86 0E B7 50 05 B7 50 : E3
5050 : 06 20 1A 86 0C B7 50 04 B7 50 05 B7 50 06 20 0D : 23
5060 : 86 08 B7 50 04 B7 50 05 B7 50 06 20 00 8E 00 00 : 60
5070 : BF FD 30 1F 13 BD 50 AD 30 01 8C 10 00 26 F1 20 : DC
5080 : AA 8E 00 00 CE 0F FF BF FD 30 BD 50 AD 33 5F 30 : 7C
5090 : 01 8C 10 00 26 F1 20 93 8E 00 00 BF FD 30 1F 13 : 13
50A0 : BD 50 B2 30 01 8C 10 00 26 F1 16 FF 7E BD 50 E7 : 2A

```



```

50B0 : 20 1F BD 50 E7 B6 50 07 B1 50 08 2C 03 B6 50 08 : B6
50C0 : B1 50 09 2C 03 B6 50 09 B7 50 07 B7 50 08 B7 50 : 6C
50D0 : 09 BD 51 02 B6 50 07 F6 50 08 1F 02 10 BF FD 32 : 93
50E0 : B6 50 09 B7 FD 34 39 1F 30 F4 50 04 F7 50 07 1F : 34
50F0 : 30 57 57 57 57 F4 50 05 F7 50 08 B4 50 06 B7 50 : 35
-----
[cs] : F9 BC 66 0B 12 A5 B0 32 B6 5C F4 3E 57 16 AC 10 : 2C
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
5100 : 09 39 B6 FD 12 84 01 26 14 B6 FD 12 84 02 26 F9 : 30
5110 : B6 FD 12 84 02 27 F9 F6 50 03 5A 26 FD 39 B6 FD : 1D
5120 : 05 28 F8 1A 50 86 80 B7 FD 05 B6 FD 05 2A F8 39 : 6A
5130 : F6 FC 80 CA 80 F7 FC 80 39 7F FD 05 1C AF 39 00 : ED
5140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5180 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
5190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
51F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : BA 5D 43 65 E4 28 76 53 9A 3D 0A 3A A2 14 10 2F : A4

```

SAVEN "L13-22M",&H5000,&H513E,&H5000

リスト13-23 アナログパレット設定実行

```

10 *****
20 *      プログラムパレット SET ユニコード          *
30 *      ( LIST 13-23 )      V3.3                      *
32 *      LIST 13-22                      カ"      E"ヨヲ テ"λ  *
40 *      E"ヲ"ヲ テ"テ"ヲ"ヲ"カ"      カ"      E"ヨヲ テ"λ  *
50 *****
100 CLEAR ,&H5000:LOADM"L13-22M":SCREEN a1
130 SIMPOSE 3
140 A$=INPUT$(1):IF A$=CHR$(&HOD) OR A$=" " THEN 200
142 A=VAL(A$):IF A<1 OR A>6 THEN 140
150 POKE &H5002,A:EXEC &H5000
170 GOTO 140
200 SIMPOSE 0:PALETTEa
210 END

```

13-8 F-BASIC V3.0 と V3.3 のベンチマークテスト

FM77AV では、F-BASIC V3.0 と V3.3 が動作します。V3.3 には、4096 色表示や FM 音源等の多くの機能拡張がなされていますが、BASIC の実行速度は、FM-7 より遅くなっているような気がします。そこで、V3.0 と V3.3 (8 色モードと 4096 色モード) で、10 種類のベンチマークテストを行ってみました。

- テスト 1 PRINT 文 (リスト 13-24-1)
- テスト 2 PRINT 文 (画面スクロール) (リスト 13-24-2)
- テスト 3 演算 (リスト 13-24-3)
- テスト 4 文字変数操作 (リスト 13-24-4)
- テスト 5 LINE 文 (リスト 13-24-5)
- テスト 6 PAINT 文 (リスト 13-24-6)
- テスト 7 GOTO 文 (リスト 13-24-7)
- テスト 8 整数変数操作 (リスト 13-24-8)
- テスト 9 ランダムファイルアクセス (リスト 13-24-9)
- テスト 10 シーケンシャルファイルアクセス (リスト 13-24-10)

結果は、図 13-24 に示すとおりです。まず目につくのが、V3.3 のグラフィックの驚異的な高速性です。ハードウェア描画機能の効果なのでしょう。FM77AV は 8 ビット機なのですが、フライトシミュレータのようなワイヤーフレームの高速 3D 処理が実現できるのではないのでしょうか。

しかし F-BASIC の内部処理は、概して V3.0 の方が高速です。これには、V3.3 が MMR を使用しているため、CPU クロックが 1.6MHz に落ちていることが最も大きいと思われます。また、大きなプログラムでは、TWR 使用のオーバーヘッドも大きいと思われます。MMR、TWR の使用で BASIC のフリーエリアが非常に増大しているのですが、処理速度の低下というデメリットももたらしめているのです。

	テスト1	テスト2	テスト3	テスト4	テスト5	テスト6	テスト7	テスト8	テスト9	テスト10
V3.0+タイプC	4'42"	7'59"	2'06"	1'24"	0'34"	0'12"	0'49"	2'53"	2'09"	1'00"
V3.3+タイプA (8色)	5'18"	8'28"	2'20"	1'38"	0'04"	0'02"	3'08"	3'06"	1'36"	1'39"
V3.3+タイプB (4096色)	6'50"	11'35"	2'11"	1'34"	0'04"	0'11"	2'57"	2'55"	1'37"	1'39"

図13-24 V3.0とV3.3のベンチマークシート

ところで第1章でも試したことなのですが、サブモニタ(タイプA, B, C)は、F-BASICのバージョンに無関係に選択できます。ですから、V3.0上でタイプAのサブモニタを動作させることが可能です。ひょっとすると、V3.3の高速グラフィックがV3.0でも実現するかもしれません。

まずV3.0で起動します。そしてPOKE &HFD13,1 を入力してV3.3 8色モード用サブモニタ(タイプA)を選択します。そして10種類のテストを行なってみました。結果は、図13-25に示すとおりです。V3.0の機動性とV3.3の高速グラフィックをあわせもったBASICとなっています。バイオテクノロジーの遺伝子組み替えによる品種改良を連想させるような結果です。ですから、V3.3の拡張機能を必要としない場合には、F-BASIC V3.0+サブモニタ タイプAで動作させるのが最善なのかもしれません。

ただし、これはV3.3を不要だといっているのではありません。V3.3は、多くのすぐれた拡張機能とシステムアーキテクチャーをとり入れた最新のBASICインタープリタです。FM77AVを真に活かすのは、やはりV3.3 4096色モードだと思います。

	テスト1	テスト2	テスト3	テスト4	テスト5	テスト6	テスト7	テスト8	テスト9	テスト10
V3.0+タイプA	4'57"	8'20"	2'06"	1'24"	0'03"	0'02"	0'49"	2'53"	2'09"	1'00"

図13-25 V3.0+タイプAのベンチマークテスト

リスト 13-24-1 テスト 1

```

100 *****
110 '* TEST PROGRAM 1      *
120 '* ( LIST 13-24-1 )   *
122 *****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 FOR I=1 TO 20000
180   PRINT I:
190 NEXT
200 PRINT TIME$
210 END

```

リスト 13-24-2 テスト 2

```

100 *****
110 '* TEST PROGRAM 2      *
120 '* ( LIST 13-24-2 )   *
122 *****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 FOR I=1 TO 20000
180   PRINT I
190 NEXT
200 PRINT TIME$
210 END

```

リスト 13-24-3 テスト 3

```

100 *****
110 '* TEST PROGRAM 3      *
120 '* ( LIST 13-24-3 )    *
122 *****
130 SCREEN=0
140 WIDTH 40,25:CONSOLE 0,25
150 DEFNNG A-Z
160 DEFFNY(X)=SIN(X)
170 X0=0:X1=3.14159:P=3000
180 TIME$="00:00:00"
190 H=(X1-X0)/2/P
200 D=0:X=X0:Y=FNY(X)
210 D=Y+D:X=X+H:Y=FNY(X)
220 D=Y*4+D:X=X+H:Y=FNY(X)
230 D=Y+D:P=P-1
240 IF P>0 THEN 210
250 ANSWER=D*H/3
260 PRINT TIME$
270 PRINT "ANSWER : ";ANSWER
280 END

```

リスト 13-24-4 テスト 4

```

100 *****
110 '* TEST PROGRAM 4      *
120 '* ( LIST 13-24-4 )    *
122 *****
130 SCREEN=0
140 WIDTH 40,25:CONSOLE 0,25
150 CLEAR 1000:DEFINT A-Z
160 TIME$="00:00:00"
170 FOR LOOP=1 TO 20
180   X$=""
190   FOR I=32 TO 255
200     X$=X$+CHR$(I)
210   NEXT
220   QI$=X$:QO$=""
230   IF QI$="" THEN 260
240     QO$=LEFT$(QI$,1)+QO$:QI$=MID$(QI$,2,LEN(QI$)-1)
250   GOTO 230
260   X$=QO$
270 NEXT
280 PRINT TIME$
290 END

```

リスト 13-24-5 テスト 5

```

100 *****
110 '* TEST PROGRAM 5      *
120 '* ( LIST 13-24-5 )    *
122 *****
130 SCREEN=0
140 WIDTH 40,25:CONSOLE 0,25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 FOR I=0 TO 319
180   LINE (I,0)-(319-I,199),PSET,I MOD 8
190 NEXT
200 FOR I=0 TO 199
210   LINE (0,I)-(319,199-I),PSET,I MOD 8
220 NEXT
230 PRINT TIME$
240 END

```

リスト 13-24-6 テスト 6

```
100 '*****
110 '* TEST PROGRAM 6      *
120 '* ( LIST 13-24-6 )    *
122 '*****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 H=200
170 MAX=H/2:YAXIS=H-1:YCENT=H/2
180 TIME$="00:00:00"
190 FOR I=0 TO MAX STEP 5
200     COL=(I/5) MOD 8
210     LINE (I,I)-(319-I,YAXIS-I),PSET,COL,B
220     PAINT (160,YCENT),COL,COL
230 NEXT
240 PRINT TIME$
250 END
```

リスト 13-24-7 テスト 7

```
100 '*****
110 '* TEST PROGRAM 7      *
120 '* ( LIST 13-24-7 )    *
122 '*****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 FOR I=1 TO 10
1000 GOTO 9980
1010 GOTO 9970
1020 GOTO 9960
1030 GOTO 9950
1040 GOTO 9940
1050 GOTO 9930

    )

5460 GOTO 5520
5470 GOTO 5510
5480 GOTO 5500
5490 NEXT:PRINT TIME$:END
5500 GOTO 5490
5510 GOTO 5480
5520 GOTO 5470

    )

9940 GOTO 1050
9950 GOTO 1040
9960 GOTO 1030
9970 GOTO 1020
9980 GOTO 1010
```

リスト 13-24-8 テスト 8

```

100 '*****
110 '* TEST PROGRAM 8 *
120 '* ( LIST 13-24-8 ) *
122 '*****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 FOR I=1 TO 2
1000 A1000=A1000+A8980
1010 A1010=A1010+A8970
1020 A1020=A1020+A8960
1030 A1030=A1030+A8950
1040 A1040=A1040+A8940
1050 A1050=A1050+A8930

    {

8950 A8950=A8950+A1030
8960 A8960=A8960+A1020
8970 A8970=A8970+A1010
8980 A8980=A8980+A1000
8990 A8990=A8990+A0990
10000 NEXT:PRINT TIME$:END

```

リスト 13-24-9 テスト 9

```

100 '*****
110 '* TEST PROGRAM 9 *
120 '* ( LIST 13-24-9 ) *
122 '* RANDOM カ" ツクラマス *
123 '*****
130 SCREEN=0
140 WIDTH 40.25:CONSOLE 0.25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 MX=200
180 F$="1:RANDOM"
190 OPEN "R",#1,F$
200 FOR I=1 TO MX
210     PUT #1,I
220 NEXT
230 CLOSE
240 OPEN "R",#1,F$
250 FOR I=1 TO MX
260     GET #1,I
270     PUT #1,MX-I+1
280 NEXT
290 CLOSE
300 PRINT TIME$
310 END

```

```
100 '*****
110 '* TEST PROGRAM 10      *
120 '* ( LIST 13-24-A )    *
121 '* SERIAL カ" ツクラレマス *
122 '*****
130 SCREEN@0
140 WIDTH 40,25:CONSOLE 0,25
150 DEFINT A-Z
160 TIME$="00:00:00"
170 MX=4000
180 F$="1:SERIAL"
190 OPEN "O",#1,F$
200 FOR I=1 TO MX
210     PRINT #1,USING "#####";I
220 NEXT
230 CLOSE
240 OPEN "I",#1,F$
250 WHILE NOT EOF(1)
260     INPUT #1,I
270 WEND
280 CLOSE
290 PRINT TIME$
300 END
```

14-1 漢字ビットイメージプリント 2

「第 11 章 漢字 ROM」の章で紹介した漢字プリントプログラムに機能追加して強力にした漢字ビットイメージプリント 2 を紹介します。追加した機能は、漢字の回転印字、倍角印字で両方の共存も可能です。回転印字の指定をすれば漢字の縦書き印字ができます。特に何も指定しなければ、通常の横書き印字となります。

使用法は整数型変数に漢字コードまたは機能コードを設定して、その整数型変数を引数として USR 関数にてサブルーチンコールしてください。

[機能コード]

- \$0000(0)アングル 0(通常印字)
- \$0001(1)アングル 1(90°左回転)
- \$0002(2)アングル 2(180°左回転)
- \$0003(3)アングル 3(270°左回転)
- \$000D(13)改行
- \$000E(14)倍角印字モード設定
- \$000F(15)倍角印字モード解除

リスト 14-2 にサンプルプログラムを示します。リスト 14-1 の漢字ビットイメージプリント 2 を「L14-1M」というファイル名で SAVEM してから実行してください。実行結果を図 14-1 に示します。

SAVEM "L14-1M",&H5000,&H531E,&H5000

リスト 14-1 漢字ビットイメージプリント 2

ADR	:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	[cs]
5000	:	7E	51	SF	00	00	00	00	00	00	00	00	00	00	00	00	00	:	2E
5010	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5020	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5030	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5040	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5050	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5060	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5070	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5080	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
5090	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00
50A0	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	00

[illegible]

ファイルに変換する“ZLODER”を作成しました。いずれもメモリの制約から必要最小限の機能しか付いていませんが、コンパクトにまとめてみました。

(1) XLODER

BASIC 部分をリスト 14-3, マシン語部分をリスト 14-4 に示します。リスト 14-4 を入力したら“L14-4M”というファイル名で SAVEM してください。

SAVEM “L14-4M”, &H1500, &H1682, &H1500 

実行すると“FLEX バイナリファイル:”と入力要求がありますから、FLEX と同じ書式でファイル名を入力します(例……0.MAIN.BIN, FM77AV.SYS 等)。このときドライブ番号の指定をしないとドライブ 1の方が選択されます。指定されたファイルが存在すると“OFFSET (HEX):”と聞いてきますから、ロード時のメモリオフセットを 16 進数で入力してください。  キーだけを押したときには、0 が指定されたと見なされます。

リスト 14-3 XLODER BASIC 部分

```

1000 '*****
1002 '* FLEX --> BASIC CONVERTER (XLODER) *
1004 '* ( LIST 14-3 ) V3.0/V3.3 *
1006 '* LIST 14-4 カ ヒツウ テス *
1008 '*****
1010 CLEAR 100,&H1500
1020 IF PEEK(&H1500)<>&H7E THEN LOADM"L14-4M"
1030 DRIVE=&H502:ECODE=&H50D:OFFSET=&H511:JUMP=&H513:LOW=&H515:HIG=&H5
17:SNAME=&H520
1040 INPUT "FLEX バイナリファイル : ",FX$
1050 K1=INSTR(FX$,"."):IF K1<2 THEN BEEP:GOTO 1040
1060 IF K1=2 THEN UNIT=VAL(FX$) ELSE UNIT=1
1070 IF K1>2 THEN K2=K1:K1=0:GOTO 1090
1080 K2=INSTR(K1+1,FX$,"."):IF K2=0 THEN BEEP:GOTO 1040
1090 FM$=MID$(FX$,K1+1,K2-K1-1)
1100 KK$=MID$(FX$,K2+1,3)
1110 IF INSTR("BIN SYS COM ",KK$) THEN 1130
1120 PRINT "BAD FILE MODE":CHR$(7):GOTO 1040
1130 FX$=STRING$(11,CHR$(0))
1140 MID$(FX$,1)=FM$:MID$(FX$,9)=KK$
1150 POKE DRIVE,UNIT
1160 FOR I=0 TO 10:POKE SNAME+I,ASC(MID$(FX$,I+1,1)):NEXT
1170 EXEC&H1503
1180 ERC=PEEK(ECODE):IF ERC THEN 1310
1190 INPUT "OFFSET (HEX) : ",S$:S$=RIGHT$("0000"+S$,4):OFS1=VAL("&H"
+LEFT$(S$,2)):OFS2=VAL("&H"+RIGHT$(S$,2))
1200 POKE OFFSET,OFS1:POKE OFFSET+1,OFS2
1210 EXEC&H1500
1220 ERC=PEEK(ECODE):IF ERC THEN 1310
1230 AD1=PEEK(LOW)*256+PEEK(LOW+1)
1240 AD2=PEEK(HIG)*256+PEEK(HIG+1)
1250 AD3=PEEK(JUMP)*256+PEEK(JUMP+1)
1260 PRINT
1270 PRINT"LOAD アドレス : ";HEX$(AD1);" >> ";HEX$(AD2)
1280 PRINT"ジャンプアドレス : ";IF AD3=0 THEN PRINT "****" ELSE PRINT HEX$(
AD3)
1290 PRINT"*** END ***":CHR$(7)
1300 END
1310 ERROR ERC
1320 END

```


も出力することができます。続いて“FLEX テキストファイル:”と聞いてきますから、XLODERと同様に FLEX のファイル名を入力します。ただし拡張子は、TXT, BAK, OUT, C に限りません。またこのときに FILES  と入力するとファイルリストを見ることができます。

この他にラインナンバーとスペース圧縮の機能も選択します。これはコンバートしたファイルをアブソリュートアセンブラでアセンブルする場合に使います。

 リスト 14-5 ZLODER BASIC 部分

```

1000 '*****
1002 '* FLEX -> BASIC TEXT CONVERTER (ZLODER) *
1004 '* ( LIST 14-5 ) V3.0 *
1006 '* LIST 14-6 カ" ヒツヨウ テ"ス *
1008 '*****
1010 CLEAR 300,&H3000
1020 IF PEEK(&H3000)<>&H7E THEN LOADM"L14-6M"
1030 DRIVE=&H502:ECODE=&H50D:FEND=&H50E:SPFLAG=&H50F:SNAME=&H520
1040 INPUT "BASIC シュツリョクファイル :",F$
1050 INPUT "FLEX テキストファイル :",FX$
1060 IF INSTR("FILES DIR DSP CAT",FX$)=0 THEN 1110
1070 INPUT "トライフ"ナンバ" :",UNIT
1080 IF UNIT<0 OR UNIT>3 THEN BEEP:GOTO 1070
1090 POKE DRIVE,UNIT:EXEC&H3000
1100 PRINT:GOTO 1050
1110 K1=INSTR(FX$,"."):IF K1<2 THEN BEEP:GOTO 1050
1120 IF K1=2 THEN UNIT=VAL(FX$) ELSE UNIT=1
1130 IF K1>2 THEN K2=K1:K1=0:GOTO 1150
1140 K2=INSTR(K1+1,FX$,"."):IF K2=0 THEN BEEP:GOTO 1050
1150 FM$=MID$(FX$,K1+1,K2-K1-1)
1160 KK$=MID$(FX$,K2+1,3)
1170 IF INSTR("TXT BAK OUT C",KK$) THEN 1190
1180 PRINT "BAD FILE MODE":CHR$(7):GOTO 1050
1190 FX$=STRING$(11,CHR$(0))
1200 MID$(FX$,1)=FM$:MID$(FX$,9)=KK$
1210 INPUT"ラインナン" (0:OFF) :",LNUM
1220 INPUT"ス"-"ス" (Y/N) :",SP$
1230 IF INSTR("Y",SP$) THEN POKE SPFLAG,1 ELSE POKE SPFLAG,0
1240 POKE DRIVE,UNIT
1250 FOR I=0 TO 10:POKE SNAME+I,ASC(MID$(FX$,I+1,1)):NEXT
1260 EXEC&H3009
1270 ERC=PEEK(ECODE):IF ERC THEN 1420
1280 EXEC&H3003
1290 ERC=PEEK(ECODE):IF ERC THEN 1420
1300 OPEN"0",#1,F$
1310 PRINT#1,""
1320 IF LNUM=0 THEN 1350
1330 LN$=STR$(LNUM):PRINT#1,RIGHT$(LN$,LEN(LN$)-1);" ";
1340 LNUM=LNUM+10
1350 EXEC&H3006
1360 ERC=PEEK(ECODE):IF ERC THEN 1420
1370 IF PEEK(FEND)=0 THEN 1320
1380 PRINT#1,""
1390 CLOSE #1
1400 PRINT"*** END ***";CHR$(7)
1410 END
1420 CLOSE #1
1430 ERROR ERC
1440 END

```

リスト 14-6 ZLODER マシン語部分

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
3000 : 7E 31 2A 7E 30 04 7E 30 0F 7F 05 00 86 05 87 05 : B3
3010 : 01 8D 7F 26 37 86 0A CE 31 8C 34 42 8E 05 20 C6 : 74
3020 : 0B A6 80 A1 C0 26 10 5A 26 F7 EC 42 87 05 00 F7 : 20
3030 : 05 01 7F 05 0D 35 C2 35 42 33 C8 18 4A 26 DB 86 : 19
3040 : 32 54 27 05 B6 05 01 26 C8 86 3F 8C 86 35 87 05 : 24
3050 : 0D 39 86 05 B7 05 0D 8D 13 24 10 8E 05 03 86 08 : 92
3060 : A7 84 AD 9F FB FA 7A 05 0D 26 EC 39 8E 05 03 CC : A5
3070 : 0A 00 ED 84 CC 31 7C ED 02 B6 05 00 F6 05 01 ED : 87
3080 : 04 C1 11 25 03 86 01 21 4F F6 05 02 ED 06 6E 9F : F2
3090 : FB FA 8D BE 25 1C B6 31 7C 87 05 00 81 28 24 12 : 7F
30A0 : B6 31 7D B7 05 01 81 21 24 08 8E 00 04 BF 05 08 : 50
30B0 : 5F 39 C6 FF 39 BE 05 0B 8C 01 00 25 0C 86 05 01 : DE
30C0 : 27 10 8D CE 26 EC BE 05 08 A6 89 31 7C 30 01 BF : 3E
30D0 : 05 08 5F 39 20 00 02 30 30 33 39 00 04 33 39 00 : 06
30E0 : 02 30 30 00 02 30 30 35 46 00 04 35 46 00 02 30 : F0
30F0 : 30 00 02 30 30 42 00 03 42 00 01 30 30 00 02 30 : AC
-----
[cs] : F1 E6 EE 47 46 A9 8B 1D A0 4A 8C AC 98 7D CD 1A : C1
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
3100 : 30 35 00 03 35 00 01 30 30 00 02 30 30 30 30 33 : F3
3110 : 30 44 30 00 08 33 30 44 30 00 04 30 30 30 30 00 : 47
3120 : 04 30 30 33 45 00 04 33 45 00 02 30 30 00 02 30 : EC
3130 : 30 42 46 00 04 42 46 00 02 30 30 00 02 30 30 31 : 39
3140 : 00 03 31 00 01 30 30 00 02 30 30 33 30 00 04 33 : 91
3150 : 30 00 02 30 30 00 02 30 30 37 43 00 04 37 43 00 : EC
3160 : 02 30 30 00 02 30 30 33 31 00 04 33 31 00 02 30 : C2
3170 : 30 00 02 30 30 38 39 00 04 38 39 00 02 30 30 00 : DA
3180 : 02 30 30 41 36 00 04 41 36 00 02 30 30 00 02 30 : E8
3190 : 30 42 00 03 42 00 01 30 30 00 02 30 30 35 00 03 : B2
31A0 : 35 00 01 30 30 00 02 30 30 42 45 00 04 42 45 00 : 0A
31B0 : 02 30 30 00 02 30 30 45 43 00 04 45 43 00 02 30 : 0A
31C0 : 30 00 02 30 30 32 36 00 04 32 36 00 02 30 30 00 : C8
31D0 : 02 30 30 43 45 00 04 43 45 00 02 30 30 00 02 30 : 0A
31E0 : 30 38 44 00 04 38 44 00 02 30 30 00 02 30 30 31 : 21
31F0 : 30 00 04 31 30 00 02 30 30 00 02 30 30 32 37 00 : C2
-----
[cs] : F1 2B E6 AE 3C A7 CD 63 62 73 9F FB 04 00 ED 8B : DB

```

```
SAVEM "L14-6M",&H3000,&H317B,&H3000
```

14-3 ディレクトリアレンジャー

ディレクトリの並びは、気にしだすと随分気になるものです。そこでこの並びを好きなように変更しようというのが、リスト 14-7 に示すプログラムです。ディレクトリにとってそのファイルの並びというのは、あまり意味がありません。BASIC は先頭から検索を行ないますので、よく使われるファイルが前の方にあれば効率がよくなります。またアスキー順にファイルが並んでいれば、ディレクトリが見やすくなります。

プログラムを実行しドライブ番号を入力すると、ファイル一覧が表示されます(図 14-2)。以下のコマンドを入力します。

- MOVE** **[J]** キーまたは **[M]** キーを押します。表示されるカーソルによって移動させたいファイルの範囲、移動させたい位置を選択します。選択は **[J]** キーによります。
- PACK** **[P]** キーを押します。KILL されたファイルのあとに残る空きディレクトリをすべて削除します。
- SORT** **[S]** キーを押します。ディレクトリをファイル名がアスキーコード順になるように並べ替えます。
- WRITE** **[W]** キーを押します。メモリ上で編集されたディレクトリを、実際にフロッピーディスクに書き込みます。
- READ** **[R]** キーを押します。最初からディレクトリの編集をやり直すために、フロッピーディスクからディレクトリを読み直します。
- 終了** **[ESC]** キーを押します。コマンド処理の中断または、プログラムの終了を行います。

リスト 14-7 ディレクトリアレンジャー

```

1000 '*****
1020 '*   Directory Arranger   *
1030 '*   ( LIST 14-7 )   V3.0   *
1040 '*****
1080 CLEAR 8000.&H7000:DEFINT A-Z:DIM D$(7),DIR$(152)
1090 DIR$(152)=STRING$(32,255)
1100 FIELD 0,32 AS D$(0),32 AS D$(1),32 AS D$(2),32 AS D$(3),
      32 AS D$(4),32 AS D$(5),32 AS D$(6),32 AS D$(7)

1110 '
1120 '■■■■ Load ■■■■
1130 COLOR 4,0:WIDTH 80,20:CONSOLE...0
1140 PRINT "Drive # ?";
1150 D$=INPUT$(1):IF D$<"0" OR D$>"3" THEN 1150
1160 PRINT CHR$(29) D$:DRV=VAL(D$)
1170 LOCATE 12,0:COLOR 6
1180 PRINT "[CR]=Move,[P]=Pack,[S]=Sort,[W]=Write,[R]=Retake";
1190 N=0
1200 LOCATE 63,0:COLOR 5:PRINT "Loading...";
1210 FOR SCT=4 TO 22
1220   DMY$=DSKI$(DRV,1,SCT)
1230   FOR I=0 TO 7
1240     DIR$(N)=D$(I):N=N+1
1250   NEXT
1260 NEXT
1270 GOSUB 1410:ND=N
1280 '
1290 '■■■■ Command ■■■■
1300 LOCATE 63,0:COLOR 7:PRINT CHR$(5)+"Sir?";
1310 CMD=INSTR(" PpMmSsWwRr "+CHR$(13)+CHR$(27),INPUT$(1))*2
1320 ON CMD GOTO 1560,1720,2150,2310,1120,1720,2440
1330 GOTO 1310
1340 '
1350 '
1360 '◆◆◆ Sub : Cursor ◆◆◆
1370 X=N*19:Y=N MOD 19
1380 LINE(X*80,Y*10+10)-(X*80+72,Y*10+10+W),XOR,CL,BF
1390 RETURN
1400 '
1410 '◆◆◆ Sub : Display ◆◆◆

```

```

1420 LINE(0,1)-(79,19)," ".BF
1430 N=0:FLG=ASC(DIR$(N))
1440 WHILE FLG<255
1450     LOCATE 10*(N*19),N MOD 19+1
1460     IF FLG=0 THEN COLOR 1:PRINT"XXXXXXXXX":GOTO 1510
1470     COLOR 7-ASC(MID$(DIR$(N),12,1)):T$=""
1480     IF ASC(MID$(DIR$(N),13,1))=255 THEN T$="|" ELSE 1500
1490     IF ASC(MID$(DIR$(N),14,1))=255 THEN T$=">"
1500     PRINT T$ LEFT$(DIR$(N),8);
1510     N=N+1:FLG=ASC(DIR$(N))
1520 WEND
1530 RETURN
1540 '
1550 '
1560 '■■■ Pack ■■■
1570 LOCATE 63,0:COLOR 5:PRINT "Pack";
1580 N=0:FLG=ASC(DIR$(N))
1590 WHILE FLG<255
1600     IF FLG>0 THEN 1670
1610     NO=N:N1=N
1620     WHILE ASC(DIR$(N1))=0:N1=N1+1:WEND
1630     * WHILE ASC(DIR$(N1))<255
1640         DIR$(NO)=DIR$(N1):NO=NO+1:N1=N1+1
1650     WEND
1660     FOR NO=NO TO N1:DIR$(NO)=STRING$(32,255):NEXT
1670     N=N+1:FLG=ASC(DIR$(N))
1680 WEND
1690 GOSUB 1410:ND=N
1700 GOTO 1290
1710 '
1720 '■■■ Move ■■■
1730 LOCATE 63,0:COLOR 5:PRINT "Move";
1740 N=0:CL=4:W=7:GOSUB 1360
1750 I$=INPUT$(1):IF I$=CHR$(27) THEN 2120
1760 IF I$=CHR$(28) AND N+19<ND THEN GOSUB 1360:N=N+19:GOSUB 1360
1770 IF I$=CHR$(29) AND N>18 THEN GOSUB 1360:N=N-19:GOSUB 1360
1780 IF I$=CHR$(30) AND N>0 THEN GOSUB 1360:N=N-1:GOSUB 1360
1790 IF I$=CHR$(31) AND N+1<ND THEN GOSUB 1360:N=N+1:GOSUB 1360
1800 IF I$<>CHR$(13) THEN 1750
1810 GOSUB 1360:CL=1:GOSUB 1360
1820 NO=N
1830 I$=INPUT$(1):IF I$=CHR$(27) THEN 2120
1840 IF I$=CHR$(30) AND N>NO THEN GOSUB 1360:N=N-1
1850 IF I$=CHR$(31) AND N+1<ND THEN N=N+1:GOSUB 1360
1860 IF I$<>CHR$(13) THEN 1830
1870 N1=N
1880 N=NO:CL=2:W=0:GOSUB 1360
1890 I$=INPUT$(1):IF I$=CHR$(27) THEN 2120
1900 IF I$=CHR$(28) AND N+19<ND THEN GOSUB 1360:N=N+19:GOSUB 1360
1910 IF I$=CHR$(29) AND N>18 THEN GOSUB 1360:N=N-19:GOSUB 1360
1920 IF I$=CHR$(30) AND N>0 THEN GOSUB 1360:N=N-1:GOSUB 1360
1930 IF I$=CHR$(31) AND N+1<ND THEN GOSUB 1360:N=N+1:GOSUB 1360
1940 IF I$<>CHR$(13) THEN 1890
1950 N2=N
1960 IF N2<NO THEN 1990
1970 IF N2>N1 THEN 2060
1980 GOTO 2120
1990 WHILE N2<NO
2000     BUF$=DIR$(NO-1)
2010     FOR N=NO TO N1:DIR$(N-1)=DIR$(N):NEXT
2020     DIR$(N1)=BUF$
2030     NO=NO-1:N1=N1-1
2040 WEND
2050 GOTO 2120
2060 WHILE N2>N1+1
2070     BUF$=DIR$(N1+1)
2080     FOR N=N1 TO NO STEP -1:DIR$(N+1)=DIR$(N):NEXT
2090     DIR$(NO)=BUF$

```



```

2100  NO=NO+1:N1=N1+1
2110  WEND
2120  GOSUB 1410
2130  GOTO 1290
2140  '
2150  '■■■ Sort ■■■
2160  LOCATE 63,0:COLOR 5:PRINT "Sort";
2170  N=0:FLG=ASC(DIR$(N))
2180  WHILE FLG<255
2190    BUF$=DIR$(N):NO=N:N1=N+1:FLG0=ASC(DIR$(N1))
2200    WHILE FLG0<255
2210      IF FLG0=0 THEN 2230
2220      IF ASC(BUF$)=0 OR BUF$>DIR$(N1) THEN BUF$=DIR$(N1):NO=N1
2230      N1=N1+1:FLG0=ASC(DIR$(N1))
2240    WEND
2250    IF NO>N THEN SWAP DIR$(N),DIR$(NO)
2260    N=N+1:FLG=ASC(DIR$(N))
2270  WEND
2280  GOSUB 1410
2290  GOTO 1290
2300  '
2310  '■■■ Save ■■■
2320  LOCATE 63,0:COLOR 2:PRINT "Write : Sure ?";
2330  IF INSTR("Yy",INPUT$(1))=0 THEN 2420
2340  LOCATE 63,0:COLOR 5:PRINT "Write      ";
2350  N=0
2360  FOR SCT=4 TO 22
2370    FOR I=0 TO 7
2380      LSET D$(I)=DIR$(N):N=N+1
2390    NEXT
2400    DSKD$ DRV.1,SCT
2410  NEXT
2420  GOTO 1290
2430  '
2440  '■■■ Exit ■■■
2450  COLOR 7:WIDTH 80,25:CLEAR 300
2460  END

```

```

Drive # 0  [CR]=Move,[P]=Pack,[S]=Sort,[W]=Write,[R]=Retake  Sir?
.SCTDMP    .FATSRC
.SCTDMP.T  .FORMAT
.ASCDMP    .DSKRW.M
.ASU       .DSKRW.T
.RUN.0
.RUN.3
.FINFO
.FDISP
.FDISP.T
.ADIR
.LFAT
.FCOPYF
.FCOPYF.T
.WSET
.HFILES
.HFILES.T
.HF/SUB.T
.FIPL
.FIPL.T

```

図14-2 ディレクトリアレンジャーファイル一覧

14-4 FAT の整序

ディレクトリを並べ替えると、FAT もきれいにしたくなるのが人情です。このプログラムでは、ディレクトリの順にファイルの格納場所(FAT)を並べ直します(図 14-8)。

プログラムを起動してドライブ番号を入力すると、処理中のファイル名を表示しながらファイルの格納場所を並べ直していきます。すべてのファイルを並べ直すと「Completed.」と表示され、プログラムの実行が終了します。場合によっては、かなり時間がかかることもありますから注意してください。

リスト 14-8 FAT の整序

```

1000 '*****
1020 '*   FAT in Line with directory   *
1030 '*   ( LIST 14-8 )   V3.0   *
1040 '*****
1080 WIDTH 80,25:PRINT"+ FAT in Line":PRINT
1090 CLEAR 10000,&H7000:DIM DR$(?),FBF$(?),LBF$(?),FBM$(?),LBM$(?),NMT
$(152)
1100 FIELD 0.5 AS DMY$.152 AS FATID$
1110 FIELD 0.32 AS DR$(0),32 AS DR$(1),32 AS DR$(2),32 AS DR$(3),
      32 AS DR$(4),32 AS DR$(5),32 AS DR$(6),32 AS DR
$(7)
1120   INPUT"Drive #":DRV$:IF DRV$="" THEN DRV$="1"
1130   IF DRV$<>"0" AND DRV$<>"1" THEN 1120
1140   DRV=VAL(DRV$)
1150 D$=DSKI$(DRV,1,1):FAT$=FATID$
1160 PST= 0:GOSUB 1530: TD$=TBL$
1170 PST=14:GOSUB 1530: CND$=TBL$
1180 GOSUB 1630
1190 DCN=0:CPM=0
1200 WHILE MID$(TD$,DCN+1,1)<CHR$(255)
1210   IF MID$(TD$,DCN+1,1)=CHR$(0) THEN 1400
1220   CPF=ASC(MID$(CND$,DCN+1,1)):PRINT"  "NMT$(DCN)" ";
1230   WHILE CPF<&HCO OR &HFD<CPF
1240     WHILE MID$(FAT$,CPM+1,1)=CHR$(254)
1250       CPM=CPM+1
1260     WEND
1270     IF CPF=CPM THEN 1380
1280     GOSUB 1970
1290     CNF$=MID$(FAT$,CPF+1,1)
1300     CNM$=MID$(FAT$,CPM+1,1)
1310     MID$(FAT$,CPF+1,1)=CNM$
1320     MID$(FAT$,CPM+1,1)=CNF$
1330     D$=DSKI$(DRV,1,1):LSET FATID$=FAT$:DSKD$ DRV,1,1
1340     SCN=CPF:GOSUB 1730:XCN=SCN
1350     SCN=CPM:GOSUB 1730
1360     WCN=SCN:WCP=CPF:GOSUB 1840
1370     WCN=XCN:WCP=CPM:GOSUB 1840
1380     CPF=ASC(MID$(FAT$,CPM+1,1)):CPM=CPM+1
1390   WEND
1400   DCN=DCN+1
1410 WEND
1420 BEEP 1:PRINT:PRINT"Completed.":BEEP 0:END
1430 '
1440 '
1450 '
1460 '
1470 '■ Read Direrctory ■
1480 DSN=DIRN*8+4:DRN=DIRN MOD 8:D$=DSKI$(DRV,1,DSN)
1490 DIR$=DR$(DRN):TD=ASC(DIR$)
1500 RETURN

```

```

1510 '
1520 '■ Make table ■
1530 TBL$=STRING$(153,255):DIRN=0:GOSUB 1480
1540 WHILE TD<255
1550   MID$(TBL$,DIRN+1,1)=MID$(DIR$,PST+1,1)
1560   DIRN=DIRN+1:DRN=DIRN MOD 8
1570   IF DRN=0 THEN D$=DSKI$(DRV,1,DIRN*8+4)
1580   DIR$=DR$(DRN):TD=ASC(DIR$)
1590 WEND
1600 RETURN
1610 '
1620 '■ Make name table ■
1630 DIRN=0:GOSUB 1480
1640 WHILE TD<255
1650   NMT$(DIRN)=LEFT$(DIR$,8)
1660   DIRN=DIRN+1:DRN=DIRN MOD 8
1670   IF DRN=0 THEN D$=DSKI$(DRV,1,DIRN*8+4)
1680   DIR$=DR$(DRN):TD=ASC(DIR$)
1690 WEND
1700 RETURN
1710 '
1720 '■ Searech cluster number ■
1730 PCN=INSTR(FAT$,CHR$(SCN))
1740 IF PCN>0 THEN SCN=PCN:GOTO 1810
1750   PCN=INSTR(CND$,CHR$(SCN))
1760   IF PCN=0 THEN 1800
1770   WHILE PCN>0 AND MID$(TD$,PCN,1)=CHR$(0)
1780     PCN=INSTR(PCN+1,CND$,CHR$(SCN))
1790   WEND
1800   SCN=-PCN
1810 RETURN
1820 '
1830 '■ Write cluster number ■
1840 IF WCN=0 THEN 1930
1850   WCP$=CHR$(WCP)
1860   IF WCN<0 THEN 1900
1870   MID$(FAT$,WCN,1)=WCP$
1880   D$=DSKI$(DRV,1,1):LSET FATID$=FAT$:DSKID$ DRV,1,1
1890   GOTO 1930
1900   MID$(CND$,-WCN,1)=WCP$
1910   DIRN=-WCN-1:GOSUB 1480
1920   MID$(DIR$,15,1)=WCP$:LSET DR$(DRN)=DIR$:DSKID$ DRV,1,DSN
1930 RETURN
1940 '
1950 '
1960 '■ Exchange cluster ■
1970 FIELD 0,128 AS FH$,128 AS LH$
1980 TNF=CPF*4+2:SNF=(CPF MOD 4)*8+1
1990 TNM=CPM*4+2:SNM=(CPM MOD 4)*8+1
2000 FOR C=0 TO 7
2010   D$=DSKI$(DRV,TNF,SNF+C)
2020   FBF$(C)=FH$:LBF$(C)=LH$
2030 NEXT
2040 FOR C=0 TO 7
2050   D$=DSKI$(DRV,TNM,SNM+C)
2060   FBM$(C)=FH$:LBM$(C)=LH$
2070 NEXT
2080 FOR C=0 TO 7
2090   LSET FH$=FBF$(C):LSET LH$=LBF$(C)
2100   DSKID$ DRV,TNF,SNM+C
2110 NEXT
2120 FOR C=0 TO 7
2130   LSET FH$=FBM$(C):LSET LH$=LBM$(C)
2140   DSKID$ DRV,TNF,SNF+C
2150 NEXT
2160 RETURN

```


00000	6F5E	CC 5019	LDD	#\$5019
00001	6F61	97 C3	STA	<\$00C3
00002	6F63	F7 030D	STB	\$030D
00003	6F66	5A	DECB	
00004	6F67	F7 030F	STB	\$030F
00005	6F6A	7E C8BC	JMP	\$C8BC
00006	6F6D	34 40	FSHS	U
00007	6F6F	BD 6F5E	JSR	\$6F5E

図14-4 改造後のシステムディスクの内容

ドライブ0に書き換えたいディスクをセットしてプログラムを起動します。そして BASIC の WIDTH 文の要領で、文字数と行数を入力してください。

リスト 14-9 WIDTH プリセット

```

1000 '*****
1020 '*   WIDTH preset           *
1030 '*   ( LIST 14-9 )   V3.0   *
1040 '*****
1080 CLEAR 1000,&H7000:DEFINT A-U
1090 COLOR 7,0:WIDTH 40,20
1100 FIELD 0.3 AS ID0$,160 AS OMY$,6 AS ID1$
1110 D$=DSKI$(0.0,15):DT$=ID1$:D$=DSKI$(0.0,32)
1120 IF ID0$="I" AND (DT$="821001" OR DT$="840505") THEN VER=3 ELSE
VER=0
1130 IF VER=0 THEN PRINT"Unidentified System Disk !!!":END
1140 '
1150 PRINT"WIDTH preset for F-BASIC Ver 3.0":PRINT
1160 INPUT"--> WIDTH ".W.C
1170 IF W<>80 AND W<>40 THEN 1160
1180 IF C<>25 AND C<>20 THEN 1160
1190 FIELD 0.94 AS D1$,15 AS WSET$,3 AS D2$,2 AS HOOK$
1200 D$=DSKI$(0.0,16)
1210 LSET WSET$=CHR$(&HCC)+CHR$(W)+CHR$(C)+CHR$(&H97)+CHR$(&HC3)+CHR
$(&HF7)
+CHR$(3)+CHR$(&HOD)+CHR$(&H5A)+CHR$(&HF7)+
CHR$(3)+CHR$(&HOF)
+CHR$(&H7E)+CHR$(&HC8)+CHR$(&HBC
)
1220 LSET HOOK$=CHR$(&H6F)+CHR$(&H5E)
1230 DSK0$ 0.0,16
1240 PRINT:PRINT"OK!":END

```

14-6 FAT セーブプレートの変更

FAT はディスクファイルの格納状況を示すものですから、頻繁に書き換えられています。この書き換えの目安となるのが、セクタの書き換え回数です。

DISK-BASIC では、FAT はワークエリアのドライブテーブルに読み込まれていて、プログラムのセーブなどが行なわれるとデータをセクタに書き込みながらメモリ上の FAT にそのことを記録していきます。そしてセクタへの書き込み回数が FAT セーブプレートになると、ドライブテーブルの FAT をフロッピーディスクに書き戻します。

このFATセーブレートは、V3.00では何と1回になっていました。V3.02, V3.3では、17回となっています。このFATセーブレートは、RAM上の次のエリアに設定されていますから容易に変更可能です。

\$05E5(V3.0)

\$03686(V3.3)

システム起動後に上記の値を書き換えてもいいのですが、手間を省くためシステムディスクのBASICコードを書き換えるプログラムを作ってみました(リスト14-10)。プログラムを起動後、"HOW many SECTORS EACH ?" のメッセージに対して数値を入力すると、FATセーブレートが入力した値に設定されてDISK-BASICが起動するようになります。

リスト14-10 FATセーブレートの変更

```

1000 '*****
1020 '*   FAT save rate change   *
1030 '*   ( LIST 14-10 )   V3.0/V3.3   *
1040 '*****
1080 CLEAR 1000.&H7000:DEFINT A-U
1090 COLOR 7.0:WIDTH 40.25
1100 FIELD 0.3 AS ID0$.160 AS DMY$.6 AS ID1$
1110 D$=DSKI$(0.0,15):DT$=ID1$:D$=DSKI$(0.0,32):VER=0
1120 IF ID0$="\i'" AND (DT$="821001" OR DT$="840505") THEN VER=3
1130 IF ID0$="\i'" AND DT$="      " THEN VER=3.3
1140 IF VER=0 THEN PRINT"Unidentified System Disk !!!":GOTO 1270
1150 '
1160 PRINT"## FAT save counts change
1170 PRINT:PRINTUSING "F-BASIC Ver #.# System Disk":VER
1180 IF VER=3! THEN TRK=0:SCT=18:FIELD 0.104 AS D1$.1 AS SC$
1190 IF VER=3.3 THEN TRK=3:SCT=23:FIELD 0.134 AS D1$.1 AS SC$
1200 D$=DSKI$(0.TRK,SCT):SC=ASC(SC$)
1210 PRINT:PRINT"How many sectors each ( CR ="SC")";
1220 INPUT" ";SE$:IF SE$="" THEN SE=SC ELSE SE=VAL(SE$)
1230 IF SE<1 OR 255<SE THEN 1210
1240 LSET SC$=CHR$(SE)
1250 DSK0$ 0,TRK,SCT
1260 PRINT:PRINT "OK !!!"
1270 END

```

14-7 万年カレンダー

カレンダー作成のプログラムを作ってみました(リスト14-11)。曜日、月の印字が漢字となっていますが、ビットイメージプリントを使用しているため通常のプリンタに出力できます(図14-5)。

リスト 14-11 万年カレンダー

```

1000 *****
1002 '*      マンネン カレンタ'      *
1004 '*      ( LIST 14-11 )      V3.0/V3.3      *
1006 *****
1010 CLEAR 500.&H5000
1020 WIDTH80,25
1030 PRINT "マンネンノ カレンタ"-ヲ コ"キホ"ウチ"スカ?"
1040 INPUT "      セイレキ テ" ニュウリョク シテクタ"サイ",YEAR
1050 INPUT "フ"リンター ニ ウチタ"シマスカ (Y/N)? ",A$
1060 IF INSTR("Yン",A$) THEN LP=1 ELSE LP=0
1070 DIM YOUBIM(13),DAY(13),MOON$(13),WORK1(1)
1080 FOR MOON=1 TO 13
1090   GOSUB 1590
1100   YOUBIM(MOON)=YOUBI
1110   JJ=YOUBI-(YOUBIM(MOON-1)):IF JJ<0 THEN JJ=JJ+7
1120   DAY(MOON-1)=JJ+28
1130 NEXT
1140 GOSUB 1950
1150 GOSUB 1650
1160 IF LP=0 THEN OUT$="SCRN:":CLS ELSE OUT$="LPT0:"
1170 OPEN"O",#1,OUT$
1180 PRINT#1,STRING$(58,"*");
1190 PRINT#1,USING" CALENDAR #### ":YEAR;
1200 PRINT#1,"*****"
1210 FOR MOON=1 TO 11 STEP 2
1220   FOR LC=0 TO 9
1230     IF LP=1 AND LC=3 THEN EXEC&H5000:GOTO 1290
1240     PRINT#1,"*      ";
1250     JJ=0:GOSUB 1340
1260     PRINT#1,"      ";
1270     JJ=1:GOSUB 1340
1280     PRINT#1,"      *"
1290   NEXT
1300   GOSUB 1470
1310 NEXT
1320 END
1330 '■■■
1340 IF LC=0 OR LC=2 THEN PRINT#1,SPACE$(28)::RETURN
1350 IF LC=1 AND LP=0 THEN PRINT#1," ":MOON$(MOON+JJ):SPACE$(19)::RETU
RN
1360 IF LC=1 AND LP=1 THEN PRINT#1,CHR$(14)+" "+MOON$(MOON+JJ):"      "
:CHR$(20)::RETURN
1370 IF LC=3 THEN PRINT#1,"SUN MON TUE WED THU FRI SAT ":RETURN
1380 IF LC=4 THEN WORK1(JJ)=1-YOUBIM(MOON+JJ)
1390 FOR II=1 TO 7
1400   IF WORK1(JJ)<=0 OR WORK1(JJ)>DAY(MOON+JJ) THEN 1450
1410   PRINT#1,USING" ## ":WORK1(JJ);
1420   WORK1(JJ)=WORK1(JJ)+1
1430 NEXT
1440 RETURN
1450 PRINT#1,"      ">::GOTO 1420
1460 '■■■
1470 IF MOON=3 OR MOON=7 THEN 1500
1480 IF MOON=11 THEN 1510
1490 RETURN
1500 IF LP<>0 THEN RETURN
1510 PRINT#1,"*":SPACE$(76);"*"
1520 PRINT#1,STRING$(78,"*")
1530 PRINT "HIT ANY KEY!"
1540 A$=INKEY$:A$=INPUT$(1)
1550 IF LP<>0 THEN RETURN
1560 IF MOON<>11 THEN CLS:PRINT#1,STRING$(78,"*")
1570 RETURN
1580 '■■■
1590 YY=YEAR:MZ=MOON-3:IF MZ<0 THEN MZ=MZ+12:YY=YY-1

```

```

1600 MX=MZ MOD 5:MY=MX MOD 2:YZ=YY MOD 400
1610 MW=(MX*2)*5+MY*3+(MZ*5)*13
1620 YQUBI=(YZ+YZ*4-YZ*100+MW+3) MOD 7
1630 RETURN
1640 '■
1650 IF LP=0 THEN RETURN
1660 RESTORE 1800
1670 FOR ADD=&H5000 TO &H51FF:POKE ADD,0:NEXT
1680 READ KK$:IF KK$="*" THEN 1720
1690 DA=VAL("&H"+KK$)
1700 IF DA>255 THEN ADD=DA:GOTO 1680
1710 POKE ADD,DA:ADD=ADD+1:GOTO 1680
1720 FOR JJ=0 TO 6
1730   FOR II=0 TO 11
1740     READ KK$:DA=VAL("&H"+KK$)
1750     POKE &H5010+JJ*24+II,DA
1760     POKE &H50C5+JJ*24+II,DA
1770   NEXT
1780 NEXT
1790 RETURN
1800 DATA 5000.8E.50.08.6E.9F.FB.FA.00
1810 DATA      0E.00.50.10.01.5C.00.00
1820 DATA      2A.20.20.20.20.20.20.20
1830 DATA 20.18.4B.9C.00.50B9.20.20.20
1840 DATA      20.20.20.20.20.1B.4B.9C
1850 DATA 00.5161.20.20.20.20.20.20.20
1860 DATA      20.2A.00.0A.*
1870 DATA 00.FF.FF.91.91.91.91.91.FF.FF.00
1880 DATA 01.FF.FE.94.94.94.94.94.FF.FF.00
1890 DATA C1.E1.23.02.06.FC.FC.06.02.23.E1.C1
1900 DATA 21.23.26.2C.38.FF.FF.3C.66.C2.81.01
1910 DATA 21.23.26.2C.38.FF.FF.38.2C.26.23.21
1920 DATA 21.23.23.55.55.9F.9F.55.55.23.23.21
1930 DATA 01.01.11.11.11.FF.FF.11.11.11.01.01
1940 '■
1950 RESTORE 2000
1960 FOR JJ=1 TO 12
1970   READ MOON$(JJ)
1980 NEXT
1990 RETURN
2000 DATA < 1 月 >.< 2 月 >.< 3 月 >.< 4 月 >.< 5 月 >.< 6 月 >
2010 DATA < 7 月 >.< 8 月 >.< 9 月 >.< 10 月 >.< 11 月 >.< 12 月 >

```

***** CALENDAR 1986 *****																															
< 1 月 >														< 2 月 >																	
日	月	火	水	木	金	土								日	月	火	水	木	金	土											
				1	2	3	4															1									
5	6	7	8	9	10	11								2	3	4	5	6	7	8											
12	13	14	15	16	17	18								9	10	11	12	13	14	15											
19	20	21	22	23	24	25								16	17	18	19	20	21	22											
26	27	28	29	30	31								23	24	25	26	27	28													
< 3 月 >														< 4 月 >																	
日	月	火	水	木	金	土								日	月	火	水	木	金	土											
						1											1	2	3	4	5										
2	3	4	5	6	7	8								6	7	8	9	10	11	12											
9	10	11	12	13	14	15								13	14	15	16	17	18	19											
16	17	18	19	20	21	22								20	21	22	23	24	25	26											
23	24	25	26	27	28	29								27	28	29	30														
30	31																														

図14-5 カレンダー出力例

14-8 マシン語データ文作成

マシン語サブルーチンは BASIC プログラムと組み合わせて使うことが多いものですが、短いものであれば BASIC の DATA 文で持っておいて、プログラムでメモリに書き込むというをよく行ないます。こうするとファイルをふたつに分けなくてすむので便利なのですが、DATA 文に変換するのは結構面倒で間違いやすい作業です。そこでマシン語を BASIC の DATA 文に自動的に変換するプログラムを作成しました(リスト 14-12)。

プログラムを起動して、以下のパラメータを入力すると、DATA 文を指定のフロッピーディスク上にアスキー形式で作成します。マシン語サブルーチンを利用する BASIC プログラムとマージして利用してください。

- ・ マシン語ファイルのファイル名
- ・ マシン語の開始アドレス
- ・ マシン語の終了アドレス
- ・ DATA 文の開始行番号
- ・ DATA 文の書式
 - 0:1 行 16 バイト
 - 1:1 行 16 バイトで行の最後に合計値をつける
 - 2:1 行 255 文字以内で詰め込む
- ・ DATA 文での数の形式
 - 0:16 進数
 - 1:10 進数
- ・ DATA の最後を示す文字(3 文字以内)
- ・ 出力ファイルのファイル名

リスト 14-12 マシン語データ文作成

```

1000 '*****
1010 '* Memory -> DATA statements. Converter *
1020 '* ( LIST 14-12 ) V3.0/V3.3 *
1030 '*****
1040 '
1050 CLEAR 1000,&H2000
1060 COLOR 5.0:WIDTH 80.25:ON ERROR GOTO 1680
1070 PRINT"## memory --> DATA conversion ##"
1080 COLOR 7:PRINT
1090 LINEINPUT "> File descriptor , if necessary ? ":FD$
1100 IF FD$="" THEN 1210
1110 OPEN "I",1,FD$
1120 DM$=INPUT$(1,1)
1130 VOL=CVI(INPUT$(2,1)):TA=CVI(INPUT$(2,1)):EA=TA+VOL-1
1140 CLOSE 1
1150 RT=PEEK(&H45)*256+PEEK(&H46)
1160 RE=PEEK(&H59D)*256+PEEK(&H59E)

```

```

1170 IF TA>RT OR EA<=RE THEN 1190
1180 BEEP:COL .6:PRINT "Out of free area":GOTO 1080
1190 LOADM FD$
1200 '
1210 PRINT "> Top address (&H"HEX$(TA))" ? "":CX=POS(0)+2:INPUT "&H".T
AS$
1220 IF TA$>"" THEN TA=VAL("&H"+TA$)
1230 IF TA<0 OR 65535!<TA THEN BEEP:LOCATE 0,CSRLIN-1:GOTO 1210
1240 LOCATE CX,CSRLIN-1:PRINT RIGHT$("000"+HEX$(TA),4)
1250 '
1260 PRINT "> End address (&H"HEX$(EA))" ? "":CX=POS(0)+2:INPUT "&H".E
AS$
1270 IF EA$>"" THEN EA=VAL("&H"+EA$)
1280 IF EA<0 OR 65535!<EA THEN BEEP:LOCATE 0,CSRLIN-1:GOTO 1260
1290 LOCATE CX,CSRLIN-1:PRINT RIGHT$("000"+HEX$(EA),4)
1300 '
1310 PRINT "> initial Line number ? "":CX=POS(0)-2:INPUT "",LN$
1320 IF LN$="" THEN LN=10000 ELSE LN=VAL(LN$)
1330 IF LN<0 OR 63999!<LN THEN BEEP:LOCATE 0,CSRLIN-1:GOTO 1310
1340 LOCATE CX,CSRLIN-1:PRINT LN
1350 '
1360 PRINT "> Data format ? ( 0:order 1:check sum 2:pack )":
1370 DF=VAL(INPUT$(1)):IF DF<0 OR 2<DF THEN BEEP:GOTO 1410
1380 PRINT DF
1390 '
1400 PRINT "> Data type ? ( 0:hex 1:dec )":
1410 DT=VAL(INPUT$(1)):IF DT<0 OR 1<DT THEN BEEP:GOTO 1370
1420 PRINT DT
1430 '
1440 LINEINPUT "> Anchor data ,if necessary ? ":AD$
1450 IF LEN(AD$)>3 THEN LOCATE 0,CSRLIN-1:GOTO 1440
1460 '
1470 LINEINPUT "> Output file descriptor ? ":FD$
1480 IF FD$="" THEN LOCATE 0,CSRLIN-1:GOTO 1470
1490 PCM=INSTR(FD$,":")
1500 IF LEN(FD$)-PCM>8 OR PCM=1 THEN LOCATE 0,CSRLIN-1:GOTO 1470
1510 '
1520 COLOR 4:PRINT:PRINT"++ Are you sure ?"
1530 IF INSTR("yY",CHR$(13),INPUT$(1))=0 THEN 1050
1540 '
1550 COLOR 7:PRINT
1560 OPEN"D",1,FD$:PRINT #1
1570 WHILE TAKEA
1580 LN$=STR$(LN):OP$=RIGHT$(LN$,LEN(LN$)-1)+" DATA":DLMT$=" "
1590 ON DF+1 GOSUB 1780,1890,2040
1600 PRINT #1,OP$:PRINT OP$
1610 LN=LN+10:IF LN>64009! THEN ERROR 31
1620 WEND
1630 CLOSE
1640 COLOR 5:PRINT:PRINT"OK !"
1650 COLOR 7:CLEAR 300.RE
1660 END
1670 '
1680 CLOSE
1690 IF NOT(ERR=64 AND ERL=1560) THEN 1740
1700 BEEP 1:COLOR 6:PRINT "KILL " CHR$(34) FD$ CHR$(34) " : sure ?":BE
EP 0
1710 IF INSTR("yY",INPUT$(1))=0 THEN RESUME 1050
1720 KILL FD$:RESUME 1550
1730 '
1740 IF ERL>1550 AND ERL<1640 THEN KILL OF$
1750 COLOR 7:ON ERROR GOTO 0
1760 '
1770 '
1780 IF TA>EA THEN OP$=OP$+DLMT$+AD$:GOTO 1870
1790 FOR I=1 TO 16
1800 MD=PEEK(TA)
1810 IF DT THEN 1830

```

```

1820     OP$=OP$+DLMT$+RIGHT$("00"+HEX$(MD),2):GOTO 1840
1830     OP$=OP$+DLMT$+RIGHT$(" "+STR$(MD),3)
1840     TA=TA+1:DLMT$=","
1850     IF TA>EA THEN OP$=OP$+DLMT$+AD$:I=16
1860 NEXT
1870 RETURN
1880 '
1890 IF TA>EA THEN OP$=OP$+DLMT$+AD$:GOTO 2020
1900 SUM=0
1910 FOR I=1 TO 16
1920     MD=PEEK(TA):SUM=SUM+MD
1930     IF DT THEN 1950
1940     OP$=OP$+DLMT$+RIGHT$("00"+HEX$(MD),2):GOTO 1960
1950     OP$=OP$+DLMT$+RIGHT$(" "+STR$(MD),3)
1960     TA=TA+1:DLMT$=","
1970     IF TA>EA THEN OP$=OP$+DLMT$+AD$:I=16
1980 NEXT
1990 IF DT THEN 2010
2000     OP$=OP$+DLMT$+RIGHT$("000"+HEX$(SUM),3):GOTO 2020
2010     OP$=OP$+DLMT$+RIGHT$(" "+STR$(SUM),4):
2020 RETURN
2030 '
2040 IF TA>EA THEN OP$=OP$+DLMT$+AD$:GOTO 2130
2050 WHILE LEN(OP$)<247 AND TA<=EA
2060     MD=PEEK(TA)
2070     IF DT THEN 2090
2080     OP$=OP$+DLMT$+HEX$(MD):GOTO 2100
2090     OP$=OP$+DLMT$+RIGHT$(STR$(MD),LEN(STR$(MD))-1)
2100     TA=TA+1:DLMT$=","
2110     IF TA>EA THEN OP$=OP$+DLMT$+AD$
2120 WEND
2130 RETURN

```

14-9 圧縮漢字表示

PRINT@による漢字は縦16ドットと大きくて、普通のキャラクタ表示と混用できません。そこで縦方向を半分に圧縮してPRINT文と似た使い方で漢字表示ができるプログラムを考えてみました(リスト14-13)。COLOR文やLOCATE文も正常に動作します。

プログラムはポジションインディペンデントに作ってありますので、適当なアドレスにロードして実行してください。実行形式は次のとおりです。

```

EXEC   ロードアドレス   { 漢字コード }   [ ; [ { 漢字コード }   ].....]
                        { 文字列   }           { 文字列   }

```

漢字コードを指定すると、その漢字が圧縮表示されます。文字列に英数字が指定されると英数字がそのままに、カタカナが指定されると、ひらがなが、他の文字は空白が表示されます。

[例]

```
EXEC &H7000 &H3441 ; &H3B7A ; "ヲテガルニ" ; &H493D ; &H3C28 ; PRINT
```

"!!" 

漢字をてがるに表示!!

リスト 14-13 圧縮漢字表示

```

ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7000 : BD 92 C3 D6 17 C1 03 27 0D C1 02 27 03 BD BC 74 : D1
7010 : DC 76 8D 3F 20 2E 9E 76 E6 84 27 28 AE 01 34 04 : 20
7020 : E6 80 C1 E0 24 14 C1 A0 25 10 34 10 C0 A0 58 30 : 01
7030 : 8D 00 08 3A EC 84 35 10 20 02 86 23 8D 15 6A E4 : 0F
7040 : 26 DE 32 61 9D 08 27 08 C6 38 BD 92 94 26 B1 39 : 2F
7050 : 7E 98 50 34 16 8E 05 A0 86 16 A7 84 CE 01 50 EF : 8B
7060 : 02 35 06 ED 04 9D DE 8E 01 70 C6 08 34 04 EC C1 : 5B
7070 : AA C0 EA C0 ED 81 6A E4 26 F4 32 61 BD D9 DE 8E : 7F
7080 : 01 02 86 1E A7 80 B6 03 08 81 4F 25 03 7E 96 63 : 01
7090 : D6 C3 C1 50 27 01 48 C6 08 3D ED 81 C3 00 0F ED : 52
70A0 : 02 F6 03 0D 54 C4 0A B6 03 0C 3D ED 81 C3 00 07 : 64
70B0 : ED 02 30 05 CC 00 30 ED 81 B6 01 E5 C6 08 44 25 : 61
70C0 : 0A CE 00 00 EF 81 5A 26 FB 20 08 10 8E 01 70 EE : EB
70D0 : A1 EF 81 5A 26 F9 8C 01 3E 26 E1 C6 3E BD DF 13 : 0F
70E0 : B6 03 08 D6 C3 C1 50 26 01 4C 4C B7 03 0B 90 C3 : 45
70F0 : 26 14 B7 03 08 7C 03 0C F6 03 0C F1 03 0D 25 06 : BB
-----
[cs] : A9 87 18 24 BC 07 7C 2C 72 21 FD F7 30 96 6A 49 : D7
-----
ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : [cs]
7100 : 7A 03 0C BD 9B 50 BD DA 70 35 90 24 20 21 23 21 : A6
7110 : 56 21 57 21 22 21 26 24 72 24 21 24 23 24 25 24 : E7
7120 : 27 24 29 24 63 24 65 24 67 24 43 21 3C 24 22 24 : 3D
7130 : 24 24 26 24 28 24 2A 24 2B 24 2D 24 2F 24 31 24 : 74
7140 : 33 24 35 24 37 24 39 24 3B 24 3D 24 3F 24 41 24 : F0
7150 : 44 24 46 24 48 24 4A 24 4B 24 4C 24 4D 24 4E 24 : 6E
7160 : 4F 24 52 24 55 24 58 24 5B 24 5E 24 5F 24 60 24 : E6
7170 : 61 24 62 24 64 24 66 24 68 24 69 24 6A 24 6B 24 : 53
7180 : 6C 24 6D 24 6F 24 73 21 2B 21 2C 00 00 00 00 00 : C0
7190 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
71F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : 00
-----
[cs] : AE 20 4E DA EF 6D 26 F7 E8 52 9D 1D 03 1D F5 1D : 95

```

SAVEM "L14-13M",&H7000,&H718A,&H7000

14-10 データサーチャ

このプログラムはメインメモリ上のデータを検索し、指定されたデータが見つかったとその先頭アドレスを表示するものです(リスト 14-14)。

プログラムを起動して、サーチしたいデータ、サーチ開始アドレス、サーチ終了アドレスを順次入力します。サーチデータに文字列を指定するときには、ダブルクォーテーション(")でくくって指定します。16進数データを指定するときには、&Hなしでそのまま指定します。

リスト 14-14 データサーチャ

```

100 *****
102 '*   Data searcher   *
110 '*   ( LIST 14-14 ) V3.0 *
112 *****
120 CLEAR 300,&H6000
130 DEFFNH$(N)=RIGHT$("000"+HEX$(N),4)
140 GOSUB 380
150 LINEINPUT "Object data ? ";D$
160 IF D$="" THEN END
170 LINEINPUT "Start address ? ";SA$
180 IF SA$="" THEN SA=0 ELSE SA=VAL("&H"+SA$)
190 LINEINPUT "Last address ? ";LA$
200 IF LA$="" THEN LA=&HFC80 ELSE LA=VAL("&H"+LA$)
210 PRINT FNH$(SA) " -> " FNH$(LA)
220 IF ASC(D$)=34 THEN GOSUB 270 ELSE GOSUB 310
230 EXEC &H6000,D$,SA,LA
240 PRINT
250 GOTO 150
260 '
270 D$=MID$(D$,2):D=INSTR(D$,CHR$(34))
280 IF D>0 THEN D$=LEFT$(D$,LEN(D$)-1)
290 RETURN
300 '
310 B$=D$:D$=""
320   D$=D$+CHR$(VAL("&H"+LEFT$(B$,2)))
330   B$=MID$(B$,3)
340   IF B$="" THEN 320
350 RETURN
360 '
370 '
380 A=&H6000:RESTORE 420:READ D$
390 WHILE D$>"/":POKE A,VAL("&H"+D$):A=A+1:READ D$:WEND
400 RETURN
410 '
420 DATA BD.92.92.BD.98.F1.AF.8C.67.E7.8C.66.27.5F.BD.92
430 DATA 92.BD.9A.02.AF.8C.5C.BD.92.92.BD.9A.02.AC.8C.53
440 DATA 25.48.AF.8C.50.30.8C.4F.AD.9F.FB.FA.6D.01.26.03
450 DATA 73.05.AC.AE.8C.3D.EE.8C.37.E6.8C.36.AC.8C.36.22
460 DATA 25.A6.80.AF.8C.2D.A1.C4.26.F2.33.41.SA.27.08.A6
470 DATA 80.A1.C0.27.F7.20.DC.AE.8C.19.30.1F.1F.10.BD.AC
480 DATA 37.BD.9C.22.20.CD.BD.9B.47.7F.05.AC.39.7E.96.63
490 DATA 00.00.00.00.00.00.00.17.00./

```

ハードウェア回路図

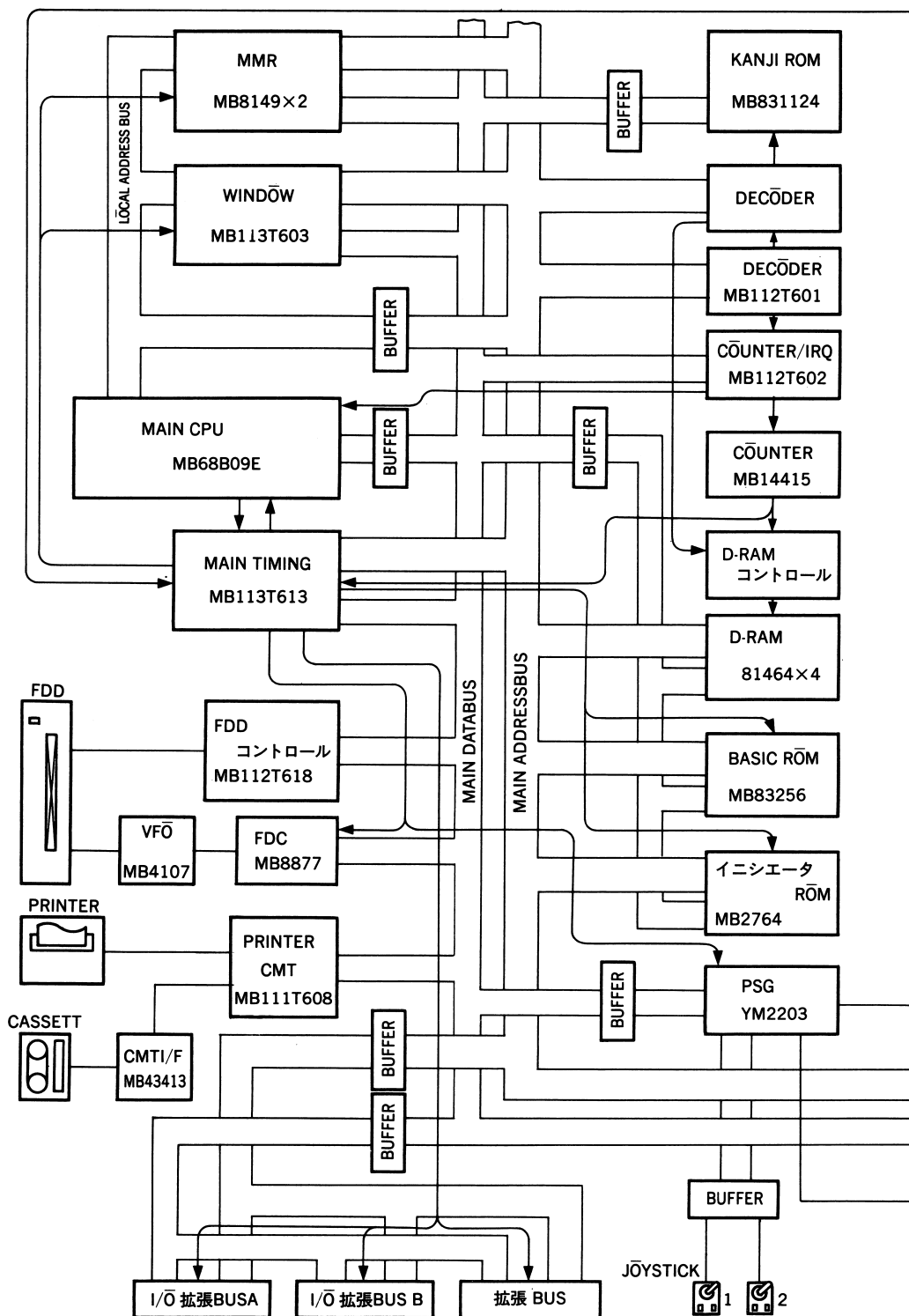
(FM77AV回路図)

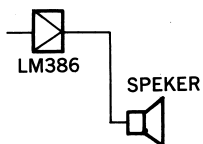
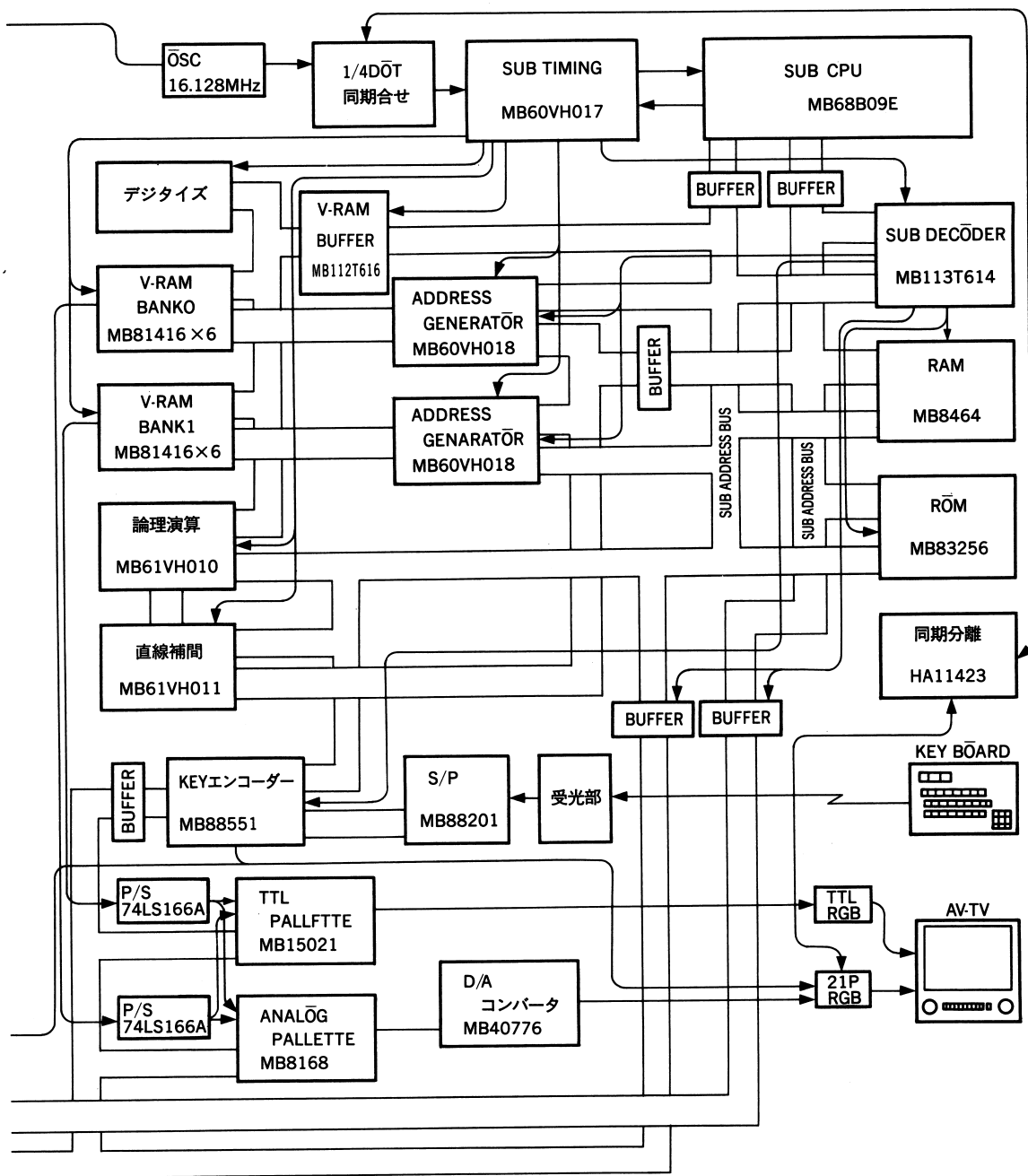
第
15
章

本章ではハードウェア回路図として、FM77AVの回路図を掲載しています。

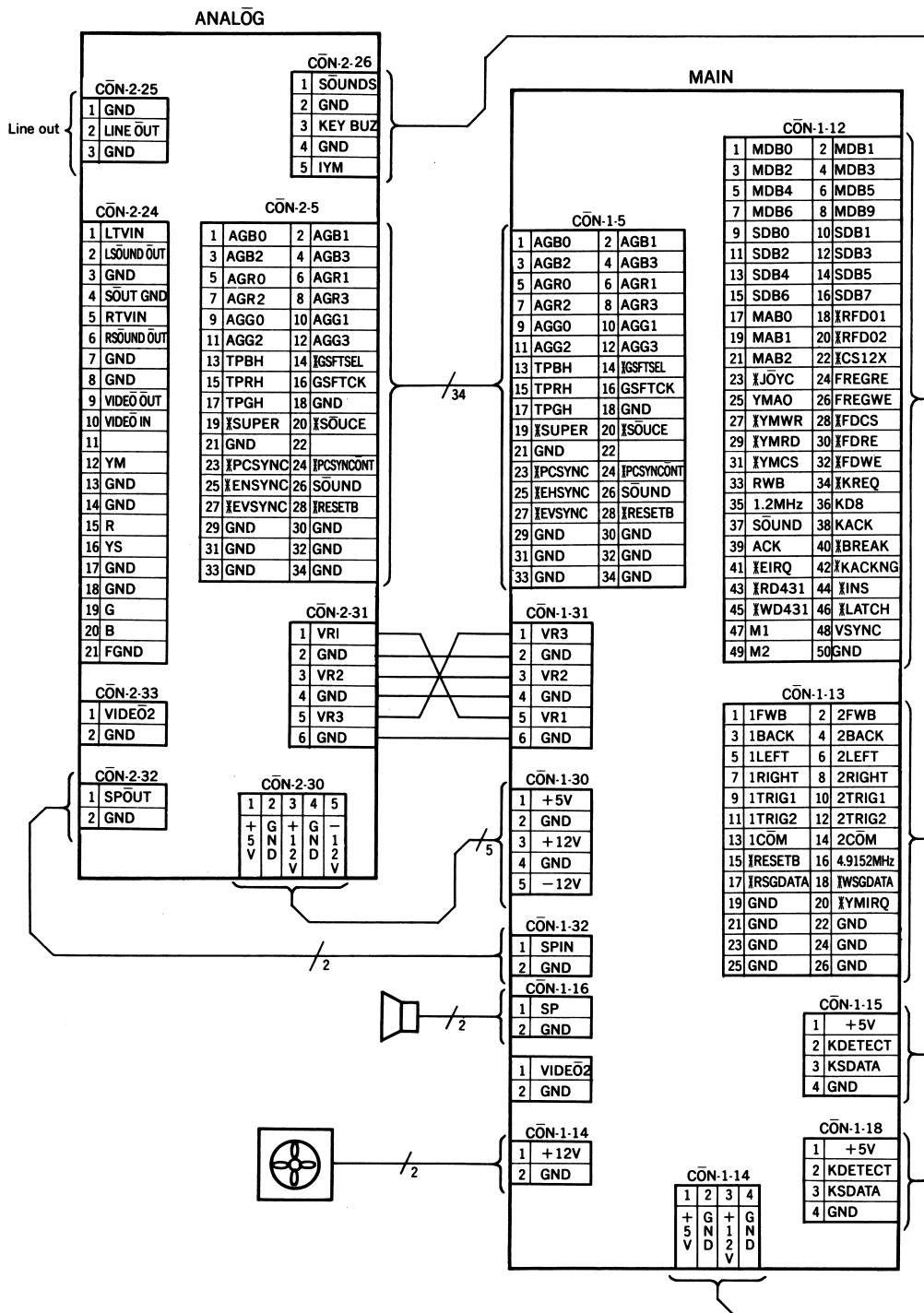
ソフトウェア開発やハードウェア製作の参考となれば幸いです。

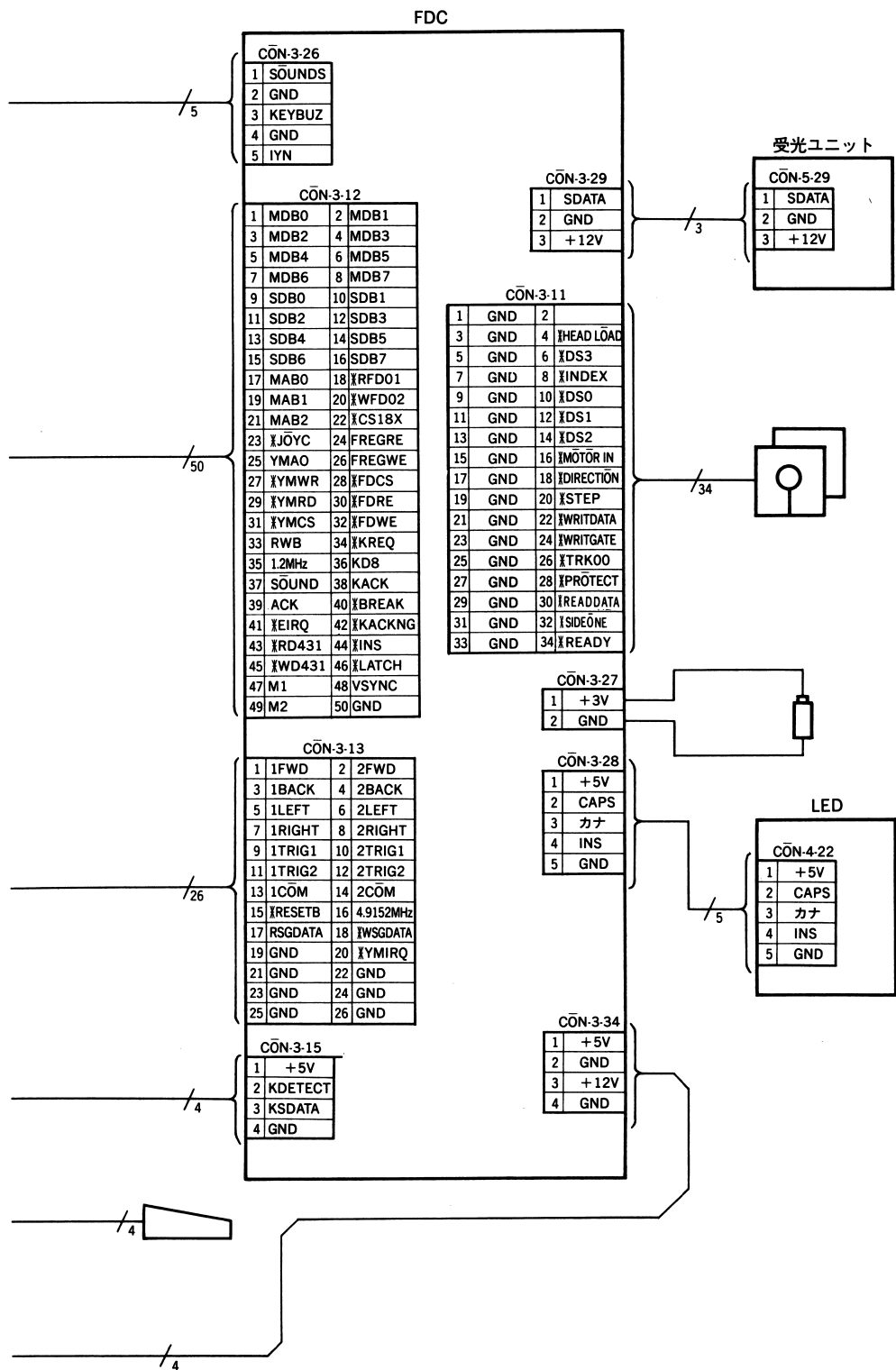
1. FM77AV



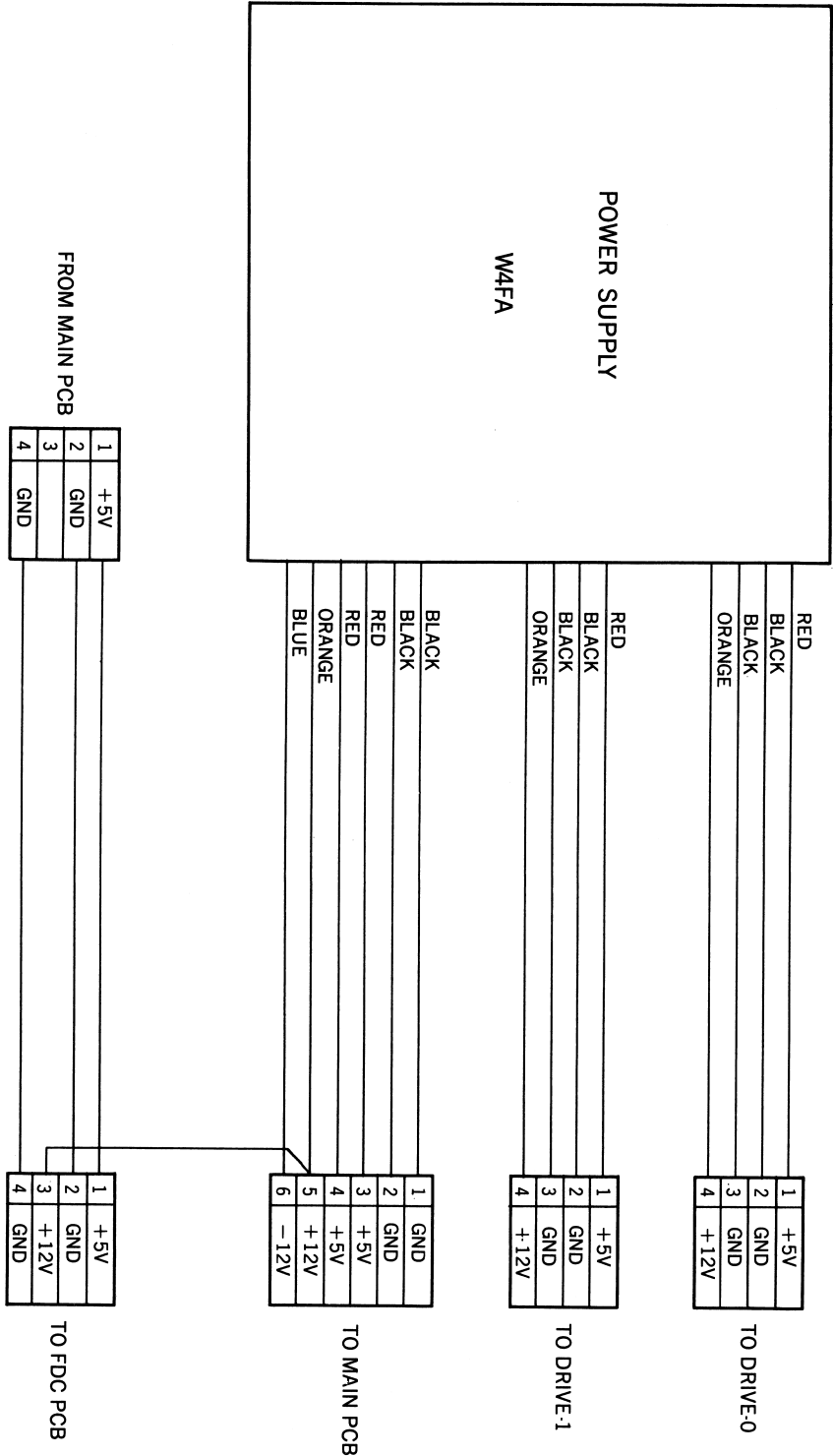


2. インターフェース



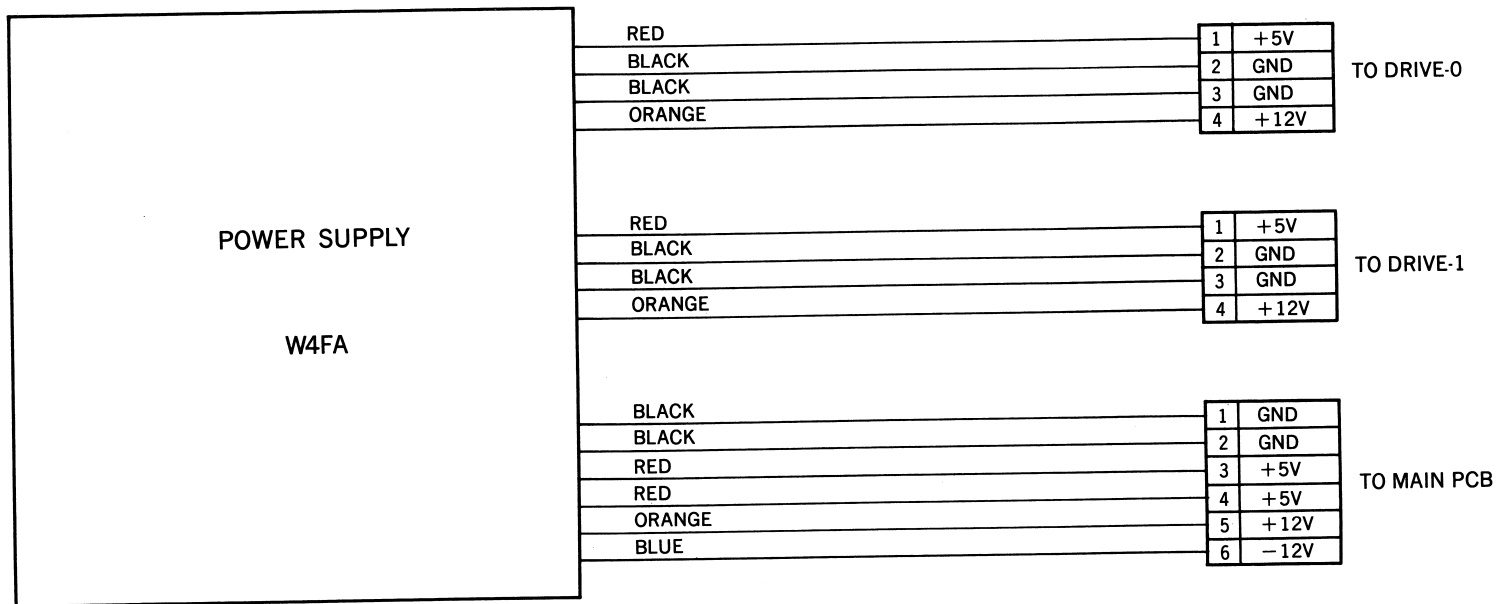


3. パワーサプライ(1)



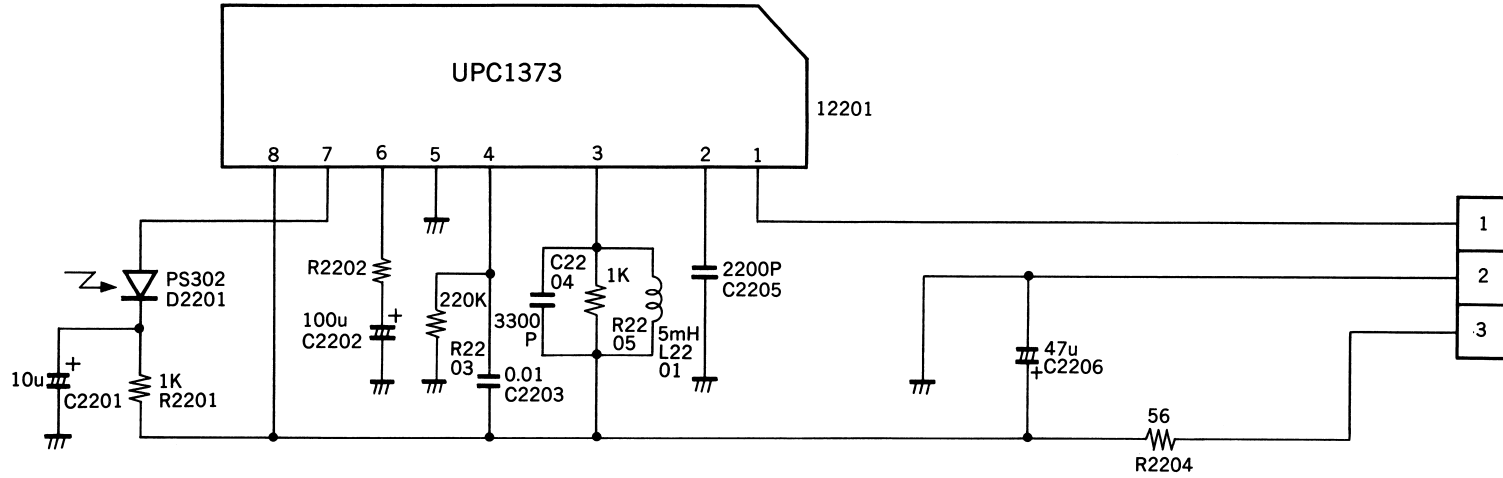
〈注〉シリアルナンバー C85100001～C85100220に適用

4. パワーサプライ(2)



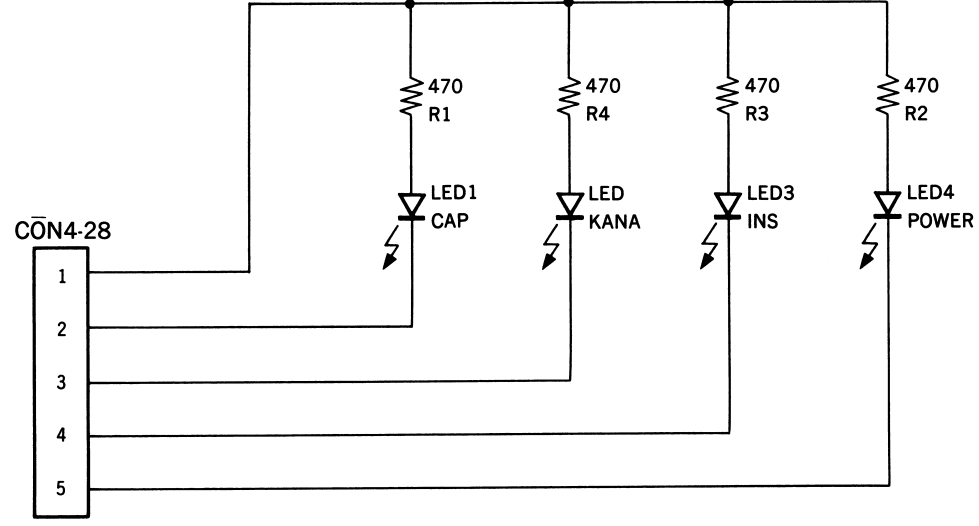
〈注〉シリアルナンバー C85100001～C85100220は除く

5. 受光ユニット

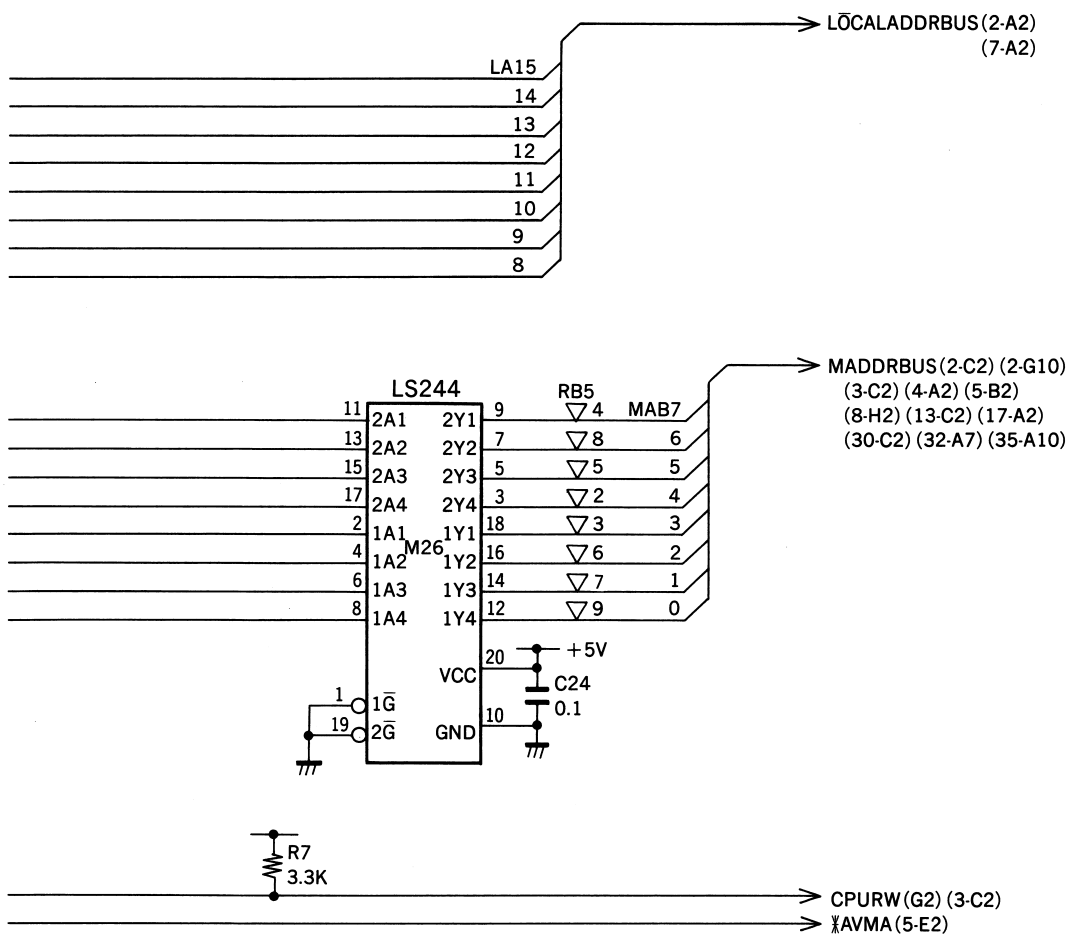


〈注〉R2205はパターン面にハンダ付けされている

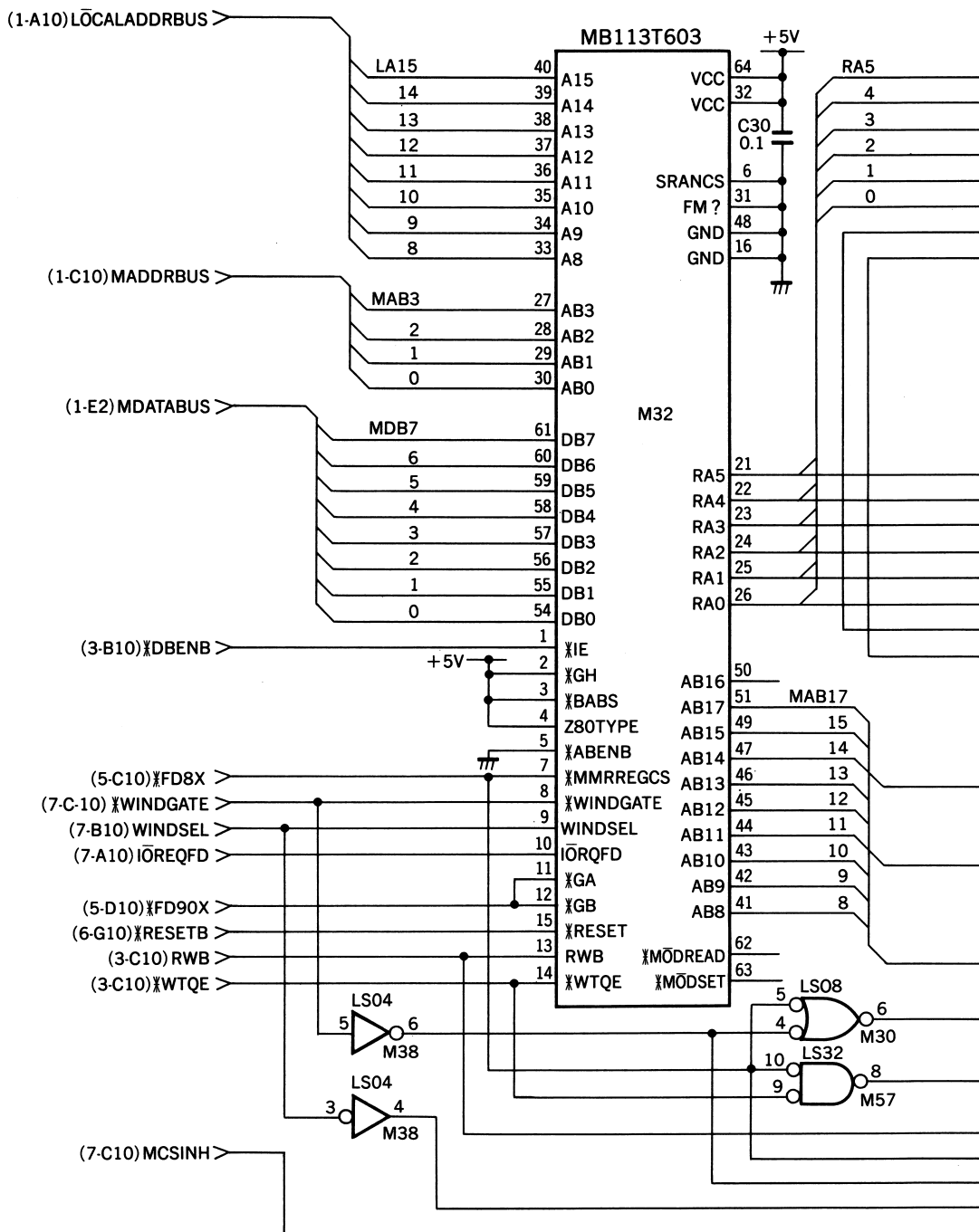
6. LED PCB

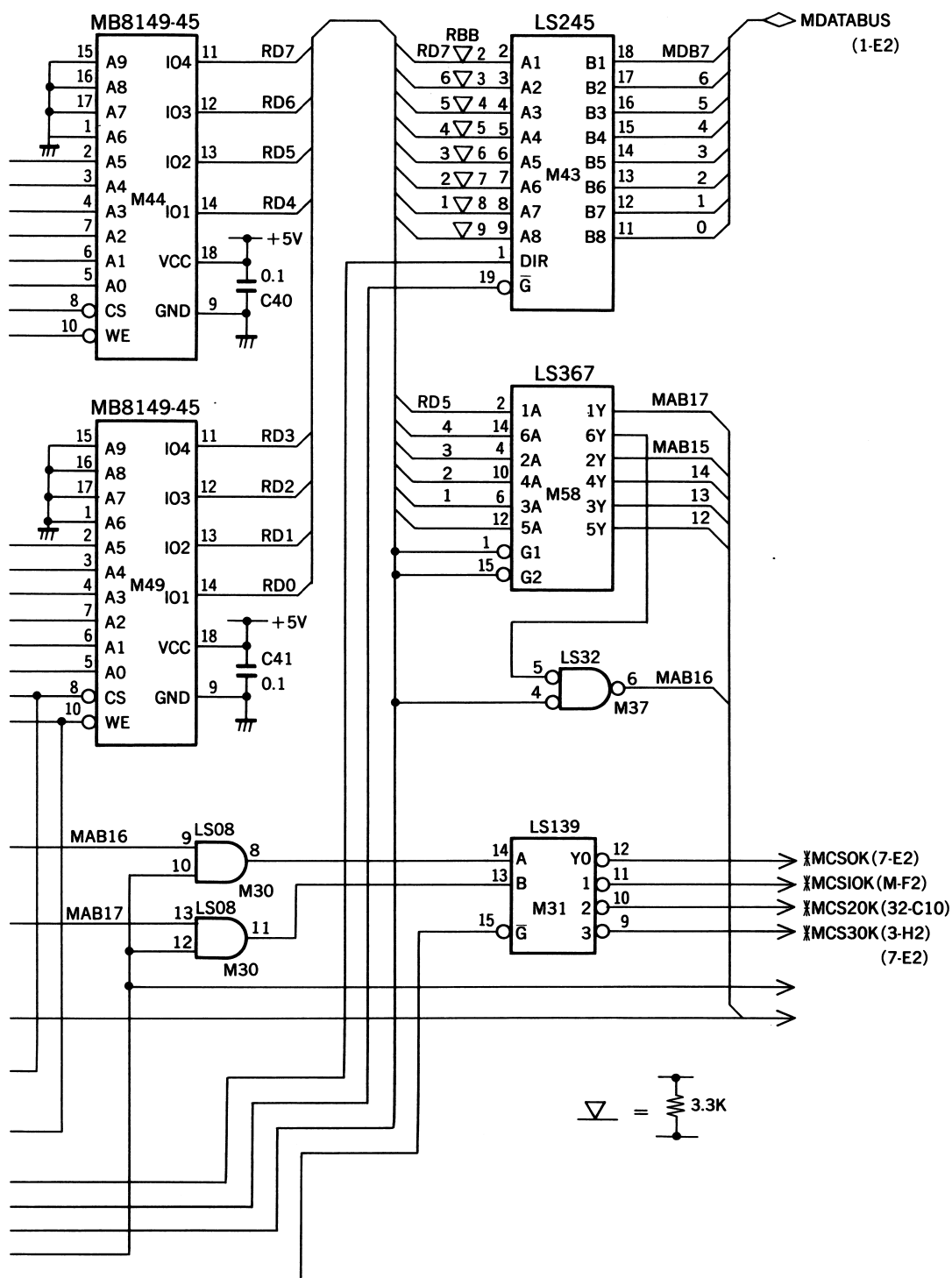




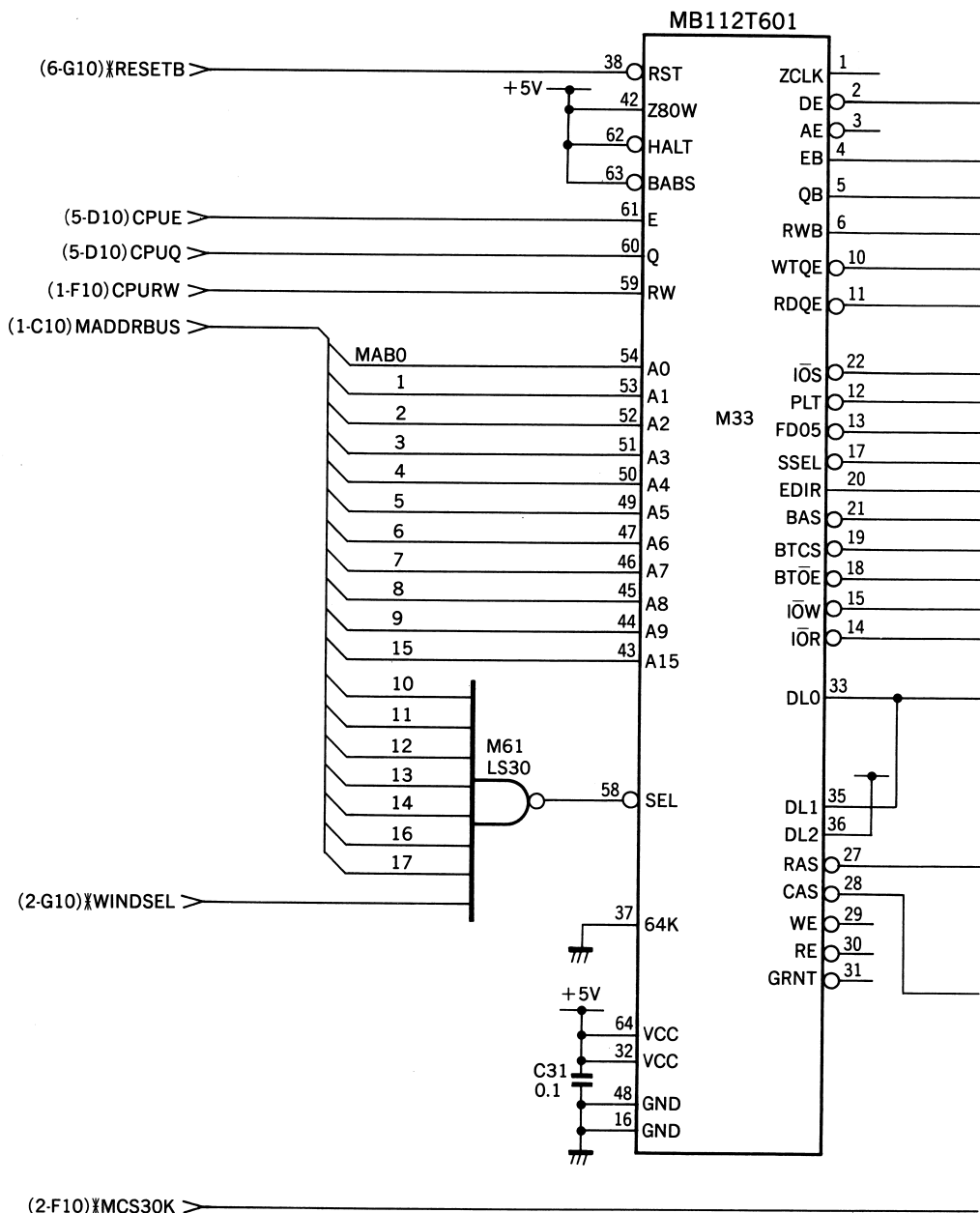


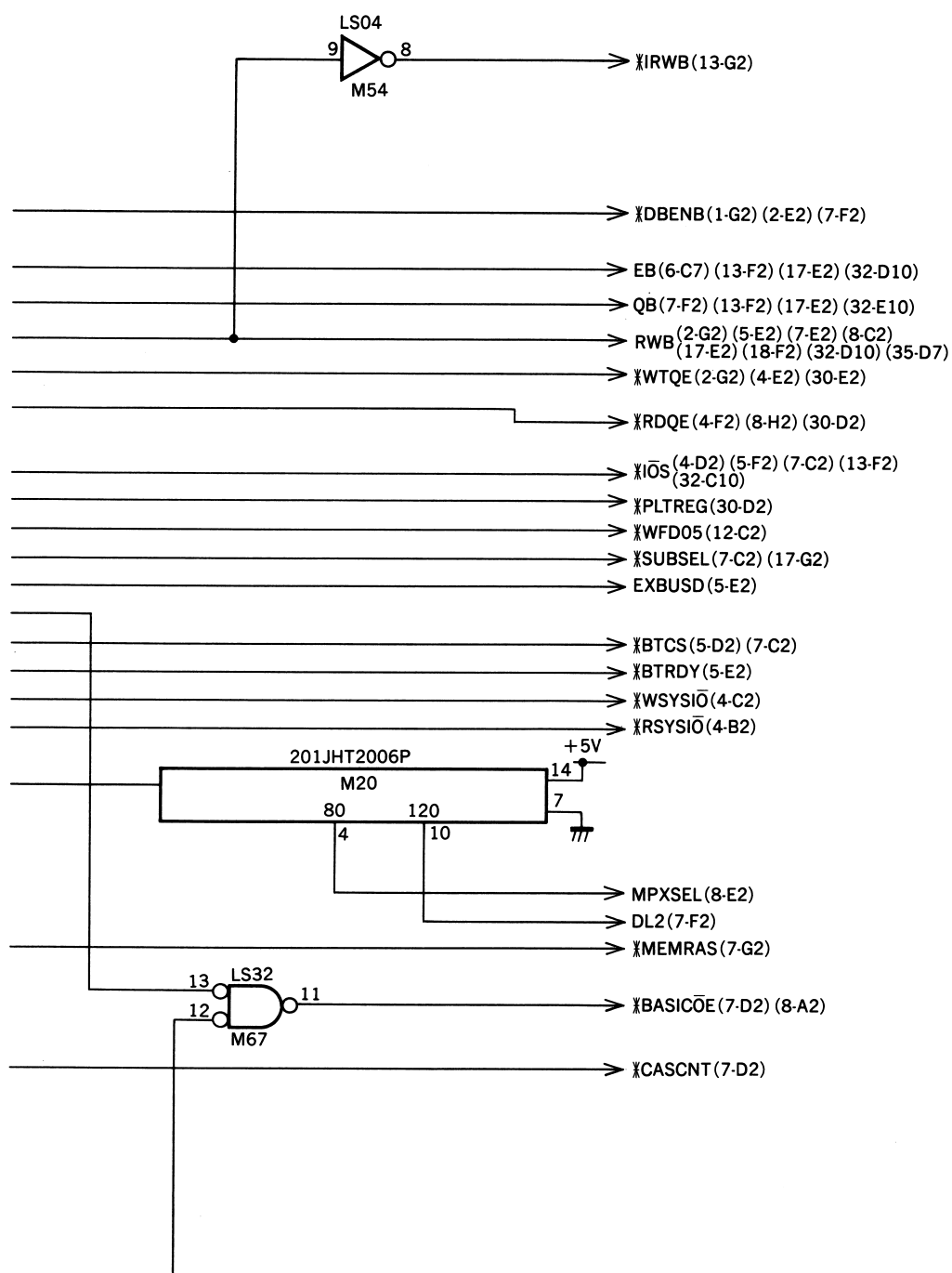
8. MMR/WINDOW etc.



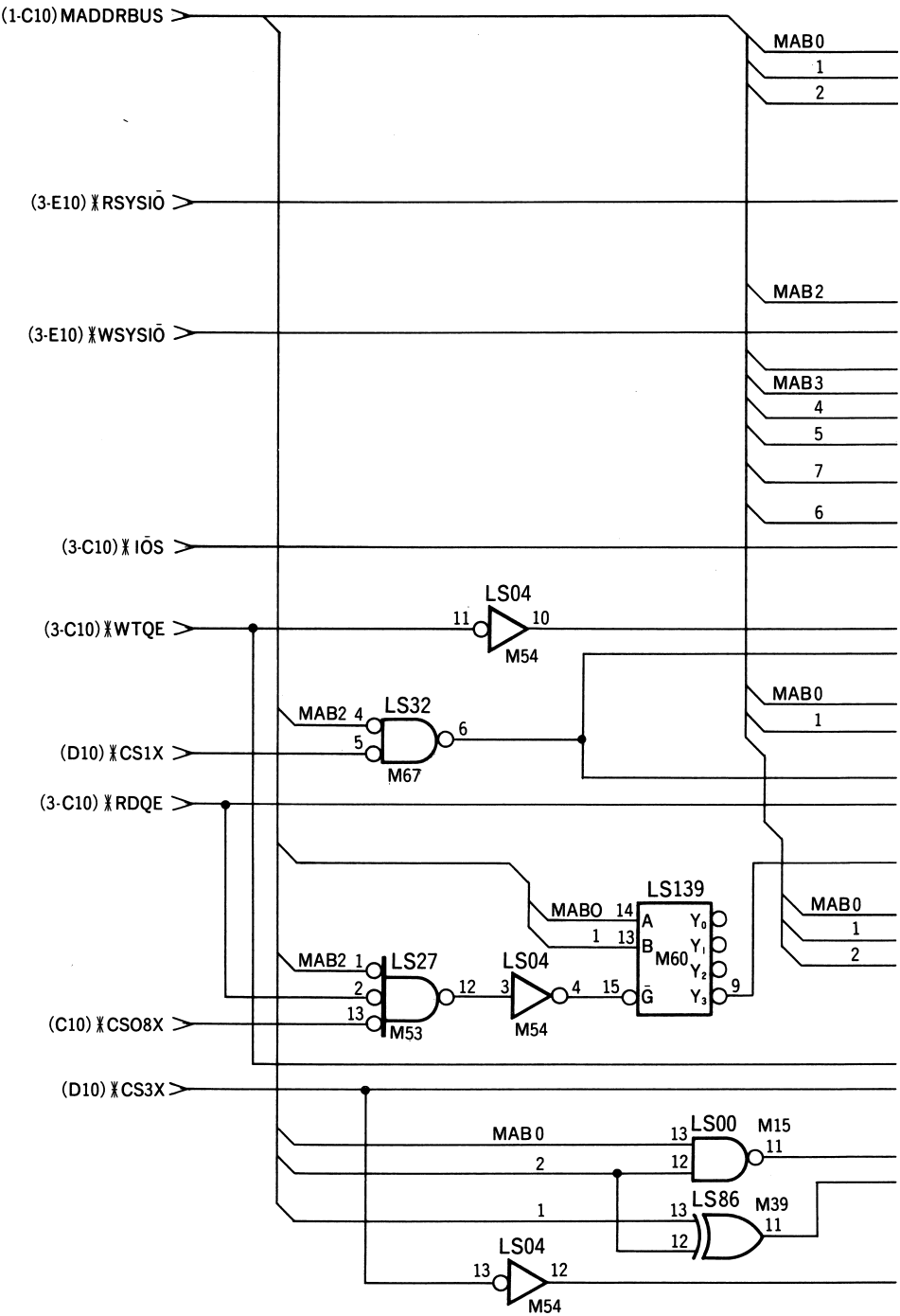


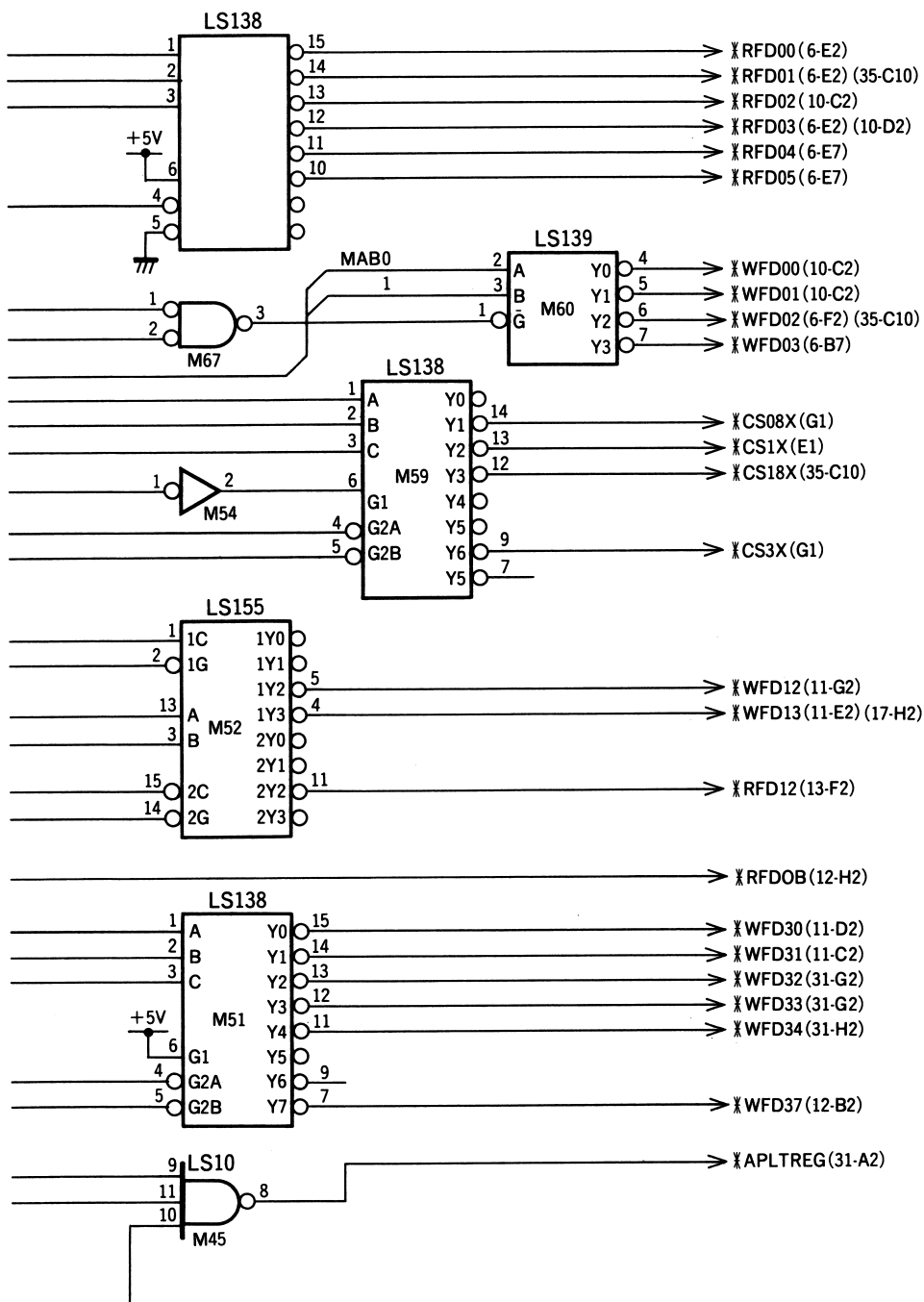
9. メインデコーダー／DRAMコントロール etc.



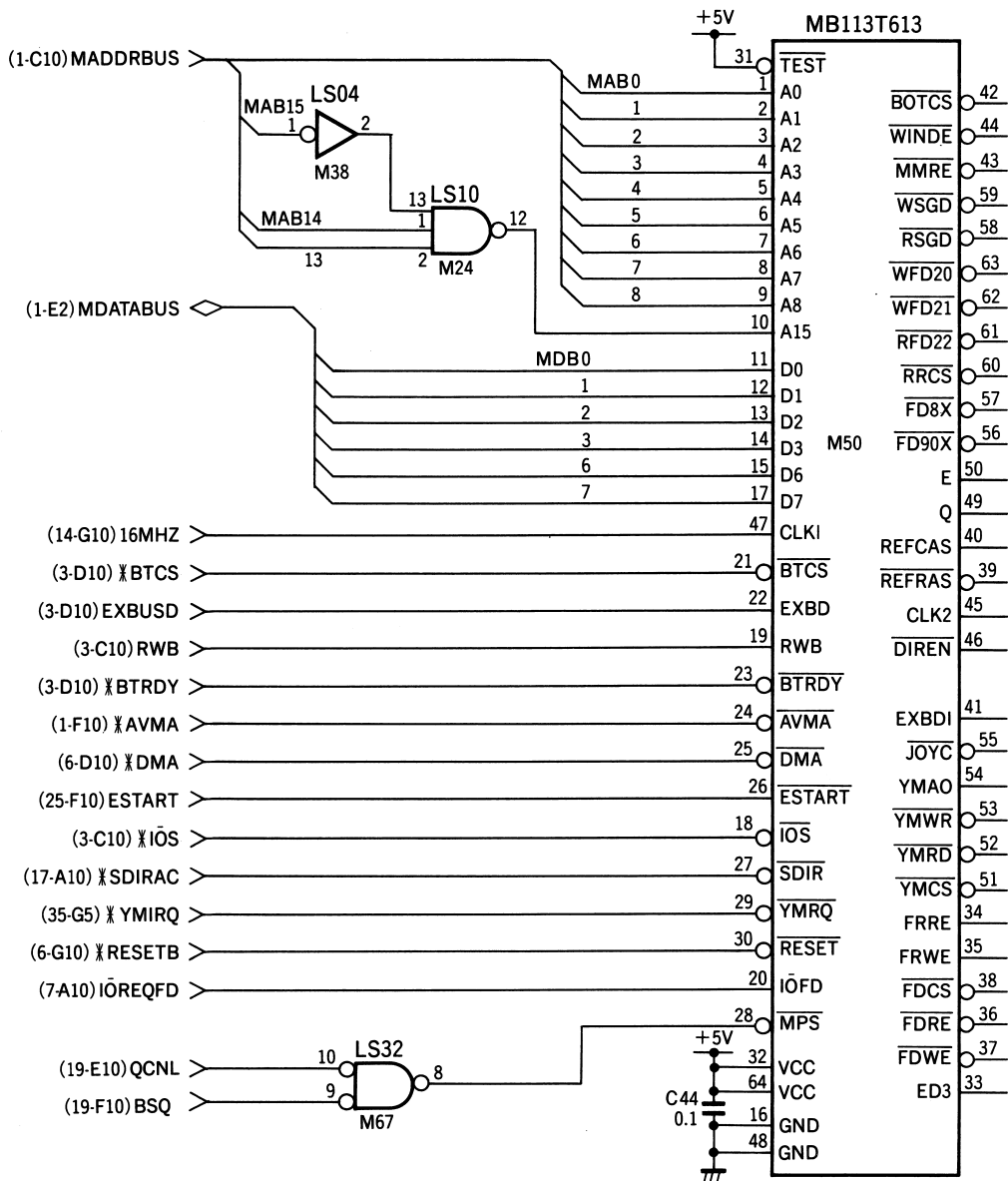


10. メインI/Oデコーダー



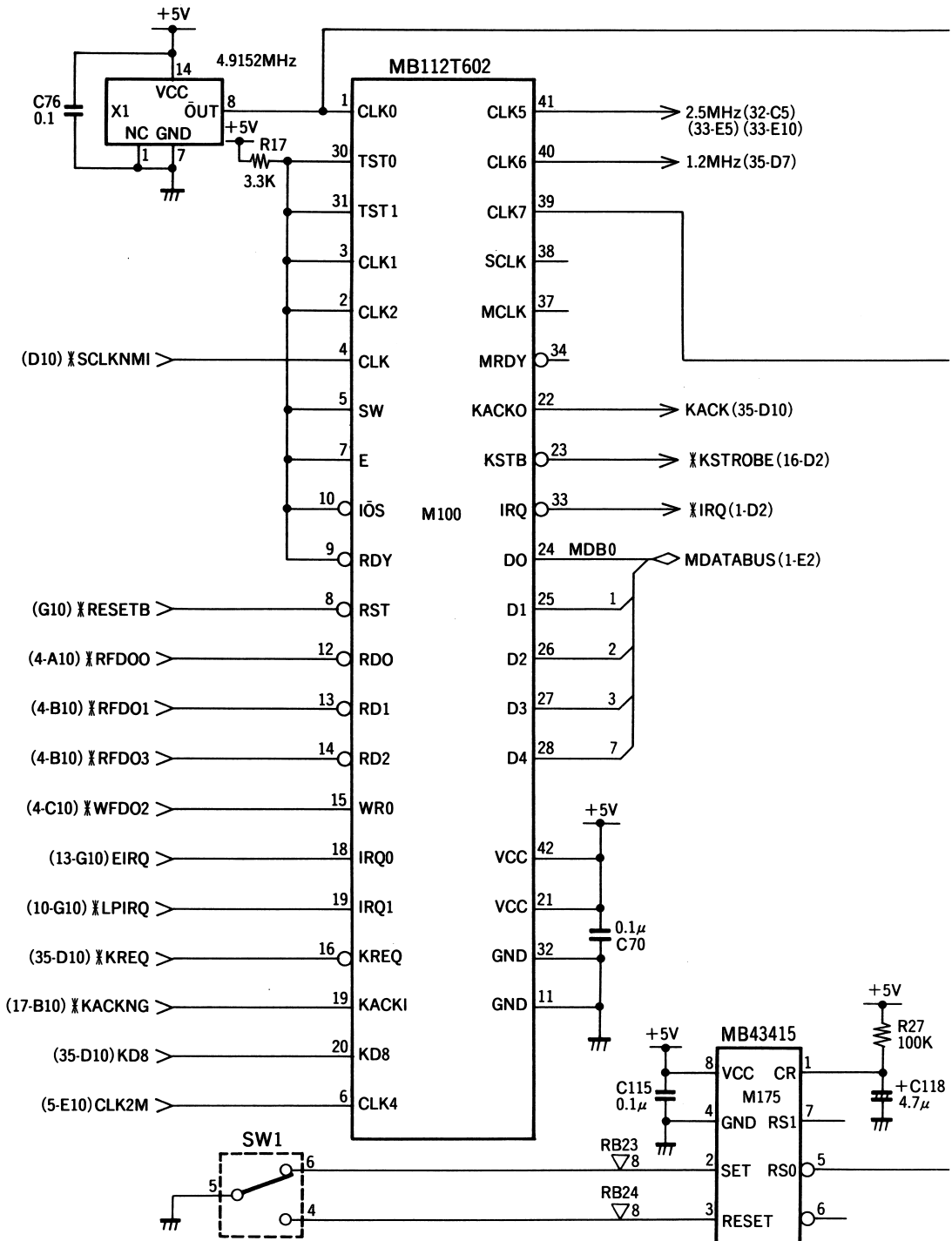


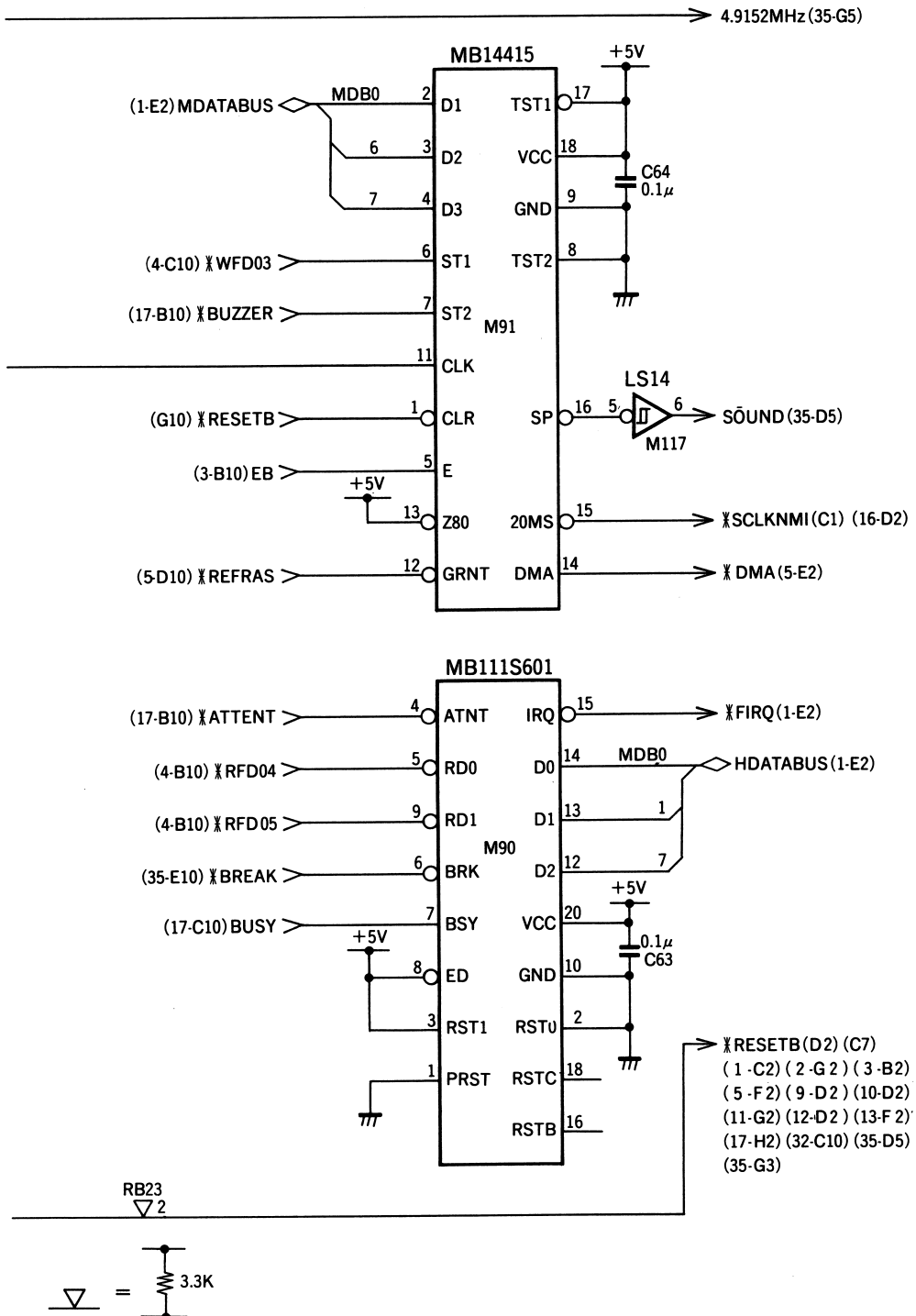
11. メインタイミングジェネレータ/ I/Oデコーダー etc.



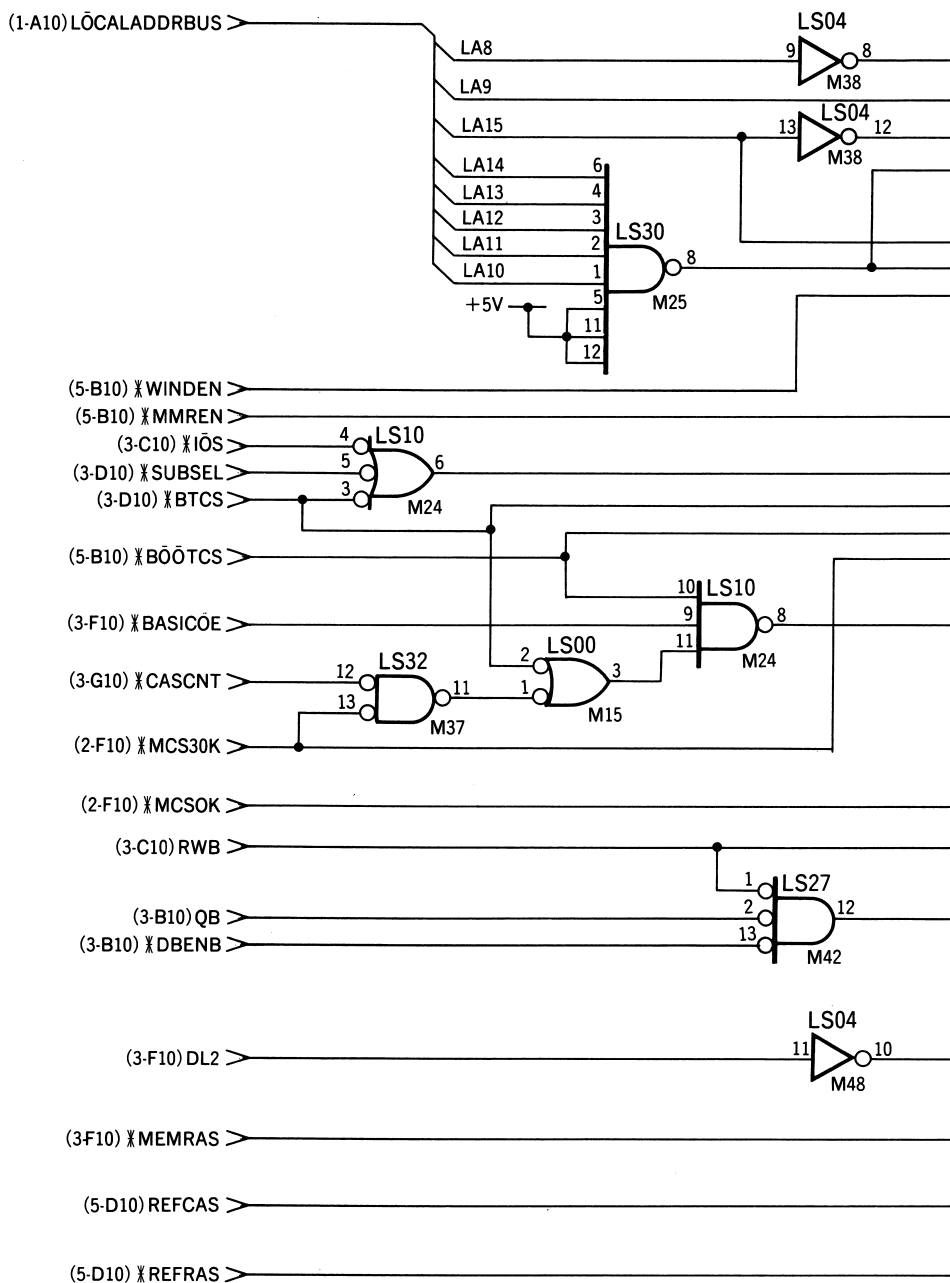
	> X BÖÖTCS (7-D2) (8-A2)
	> X WINDEN (7-C2)
	> X MMREN (7-C2)
	> X WSGDATA (35-G5)
	> X RSGDATA (35-G3)
	> X WFD20 (9-E2)
	> X WFD21 (9-C2)
	> X RFD22 (9-F2)
	> X KRÖMCS (9-F2)
	> X FD8X (2-F2)
	> X FD90X (2-F2)
	> CPUE (1-D2) (3-C2)
	> CPUQ (1-C2) (3-C2)
	> REFCAS (7-G2) (32-D7)
	> X REFRAS (6-D7) (7-H2)
	> CLK2M (6-G2)
	> X DIREN (19-C2)
	> EXBUSDIR (13-D2) (32-D5)
	> X JÖYC (35-C7)
	> YMAO (35-C7)
	> X YMWR (35-D7)
	> X YMRD (35-D7)
	> X YMCS (35-D7)
	> FREGRE (35-C10)
	> FREGWE (35-C10)
	> X FDCS (35-D10)
	> X FDRE (35-D10)
	> X FDWE (35-D10)
EDB3	> EDATABUS (13-C10)

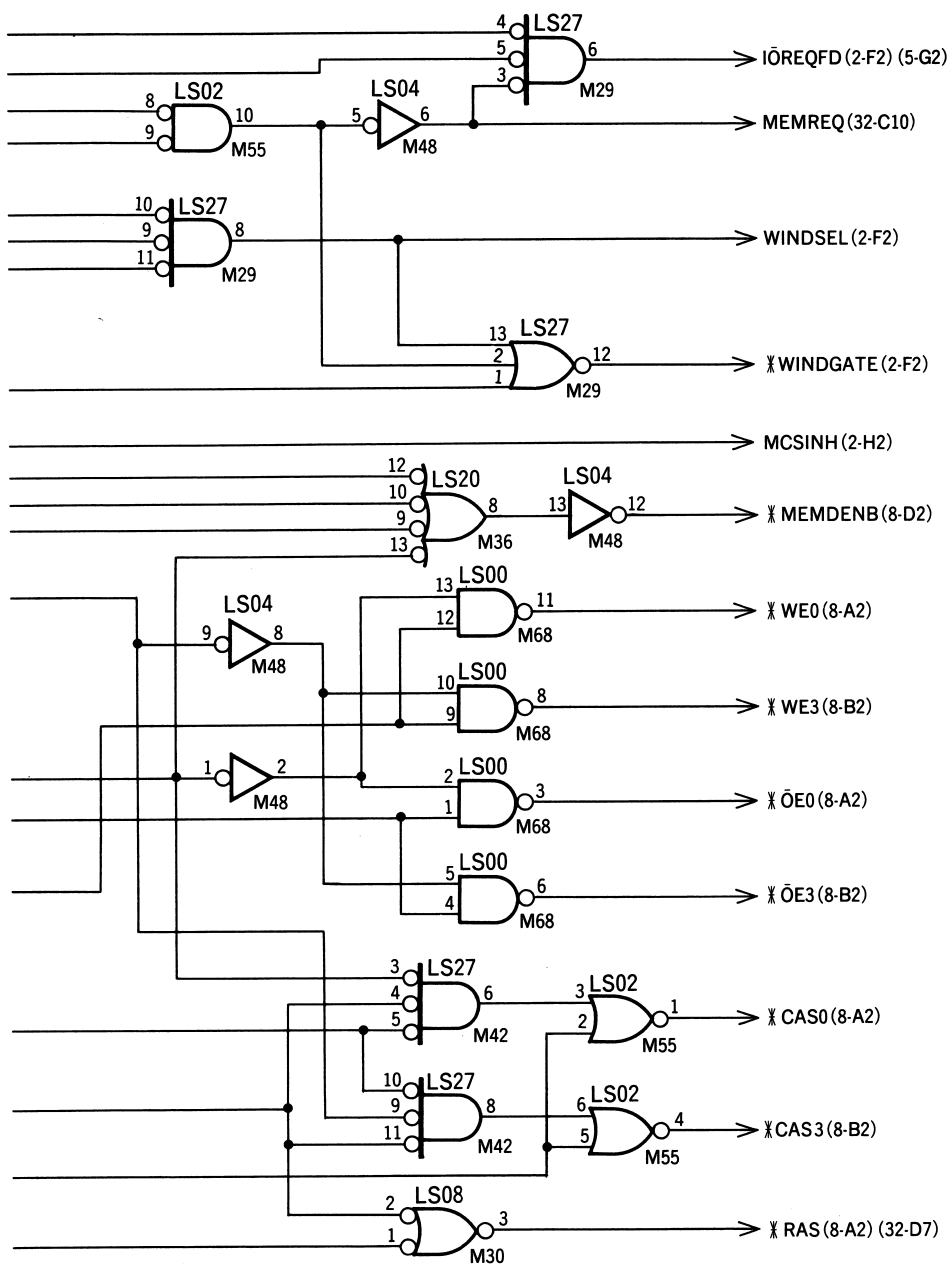
12. リセットサーキット/IRQ/FIRQサーキット etc.



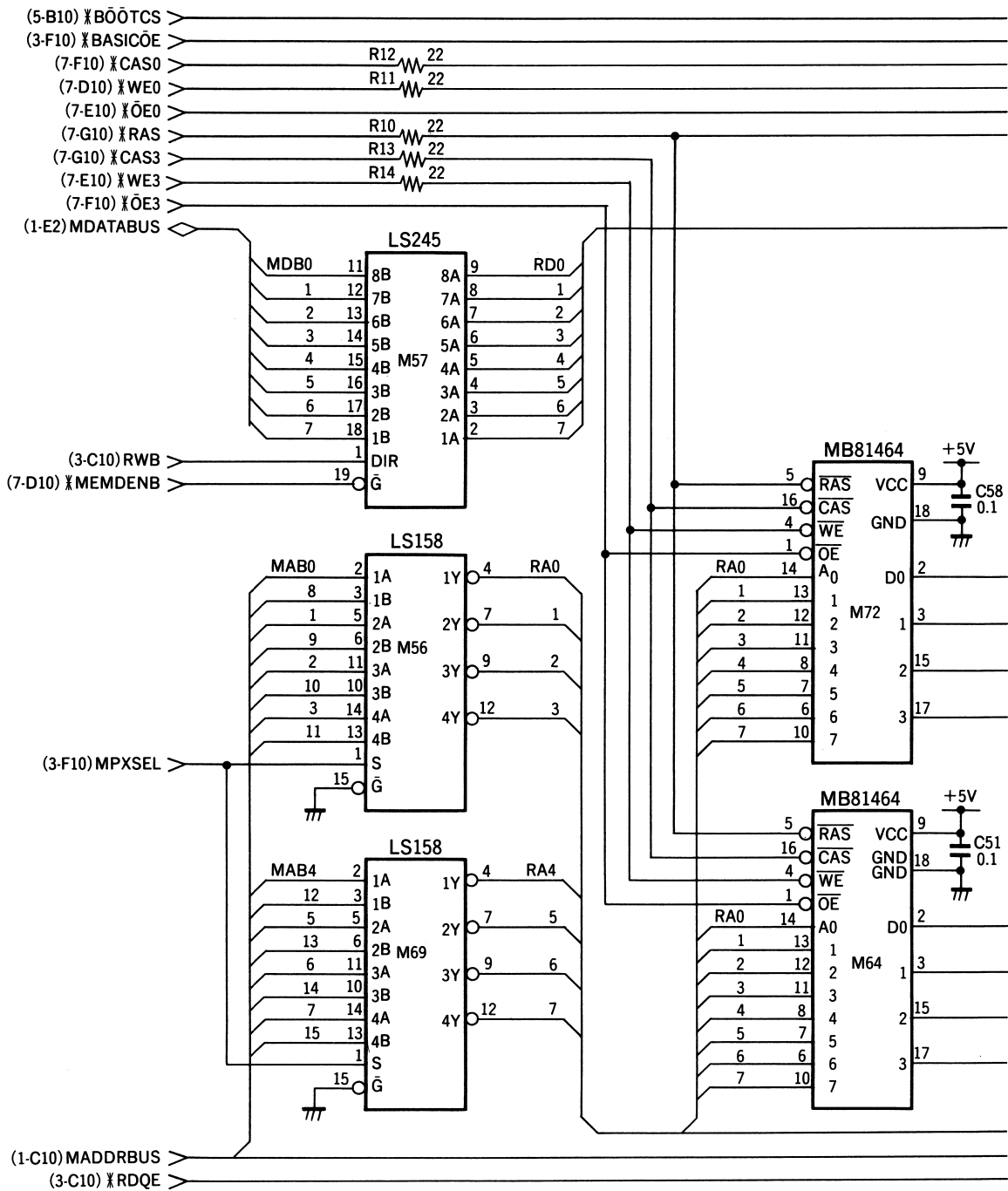


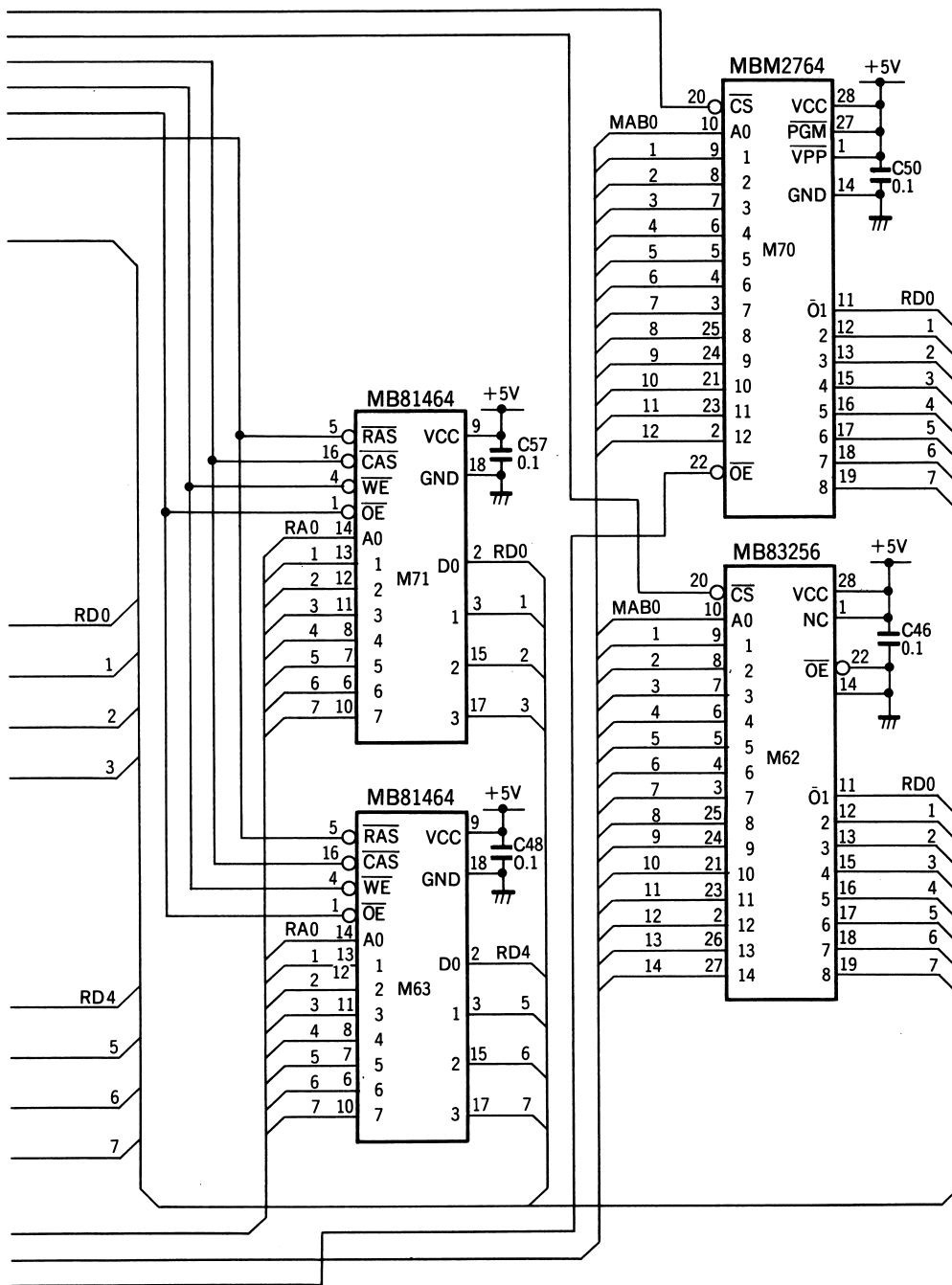
13. デコーダー, DRAMコントロール



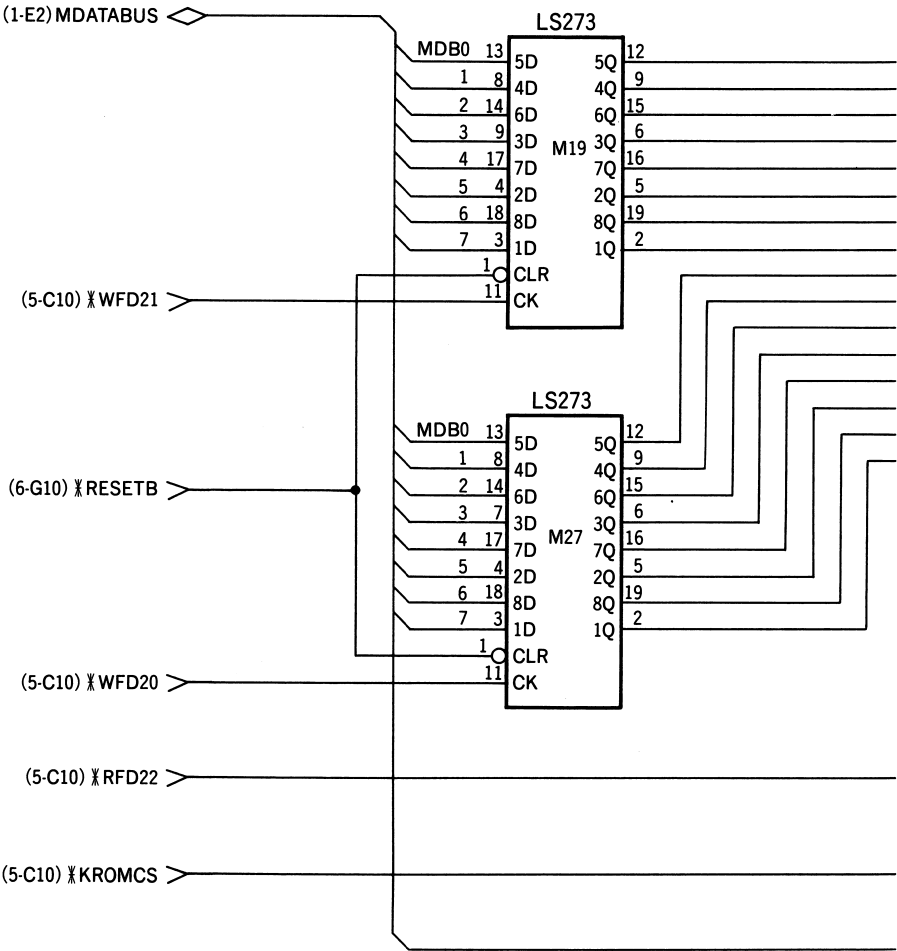


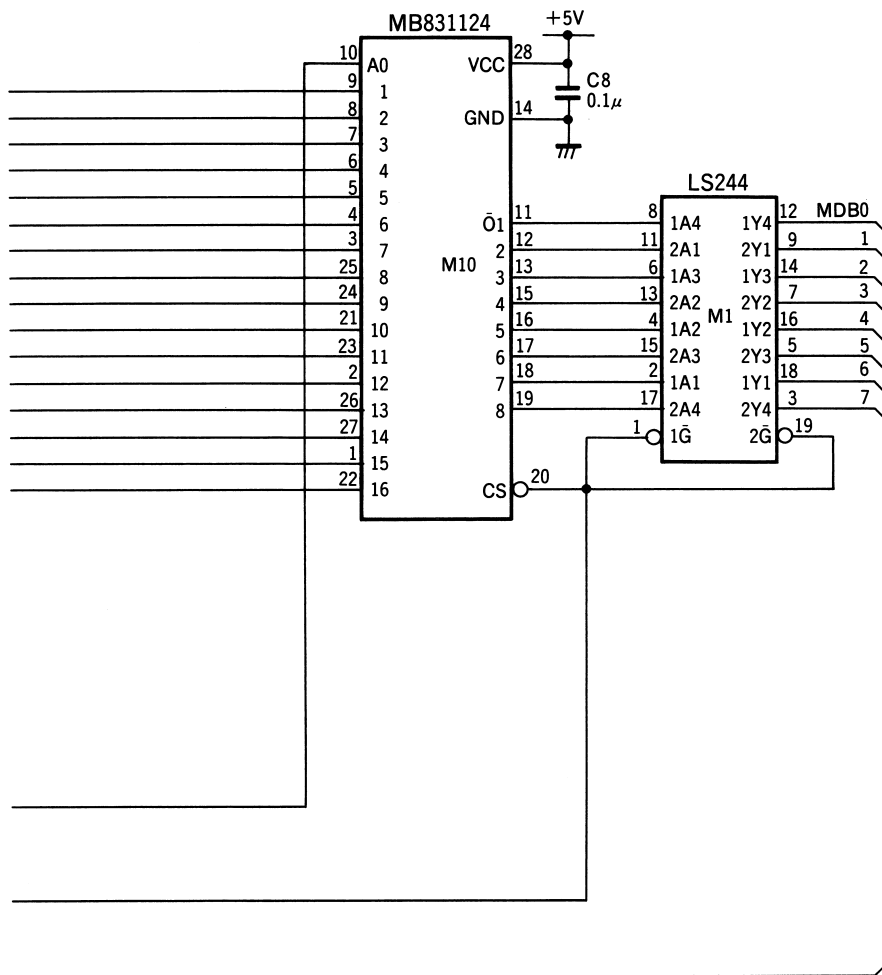
14. メインROM/RAM



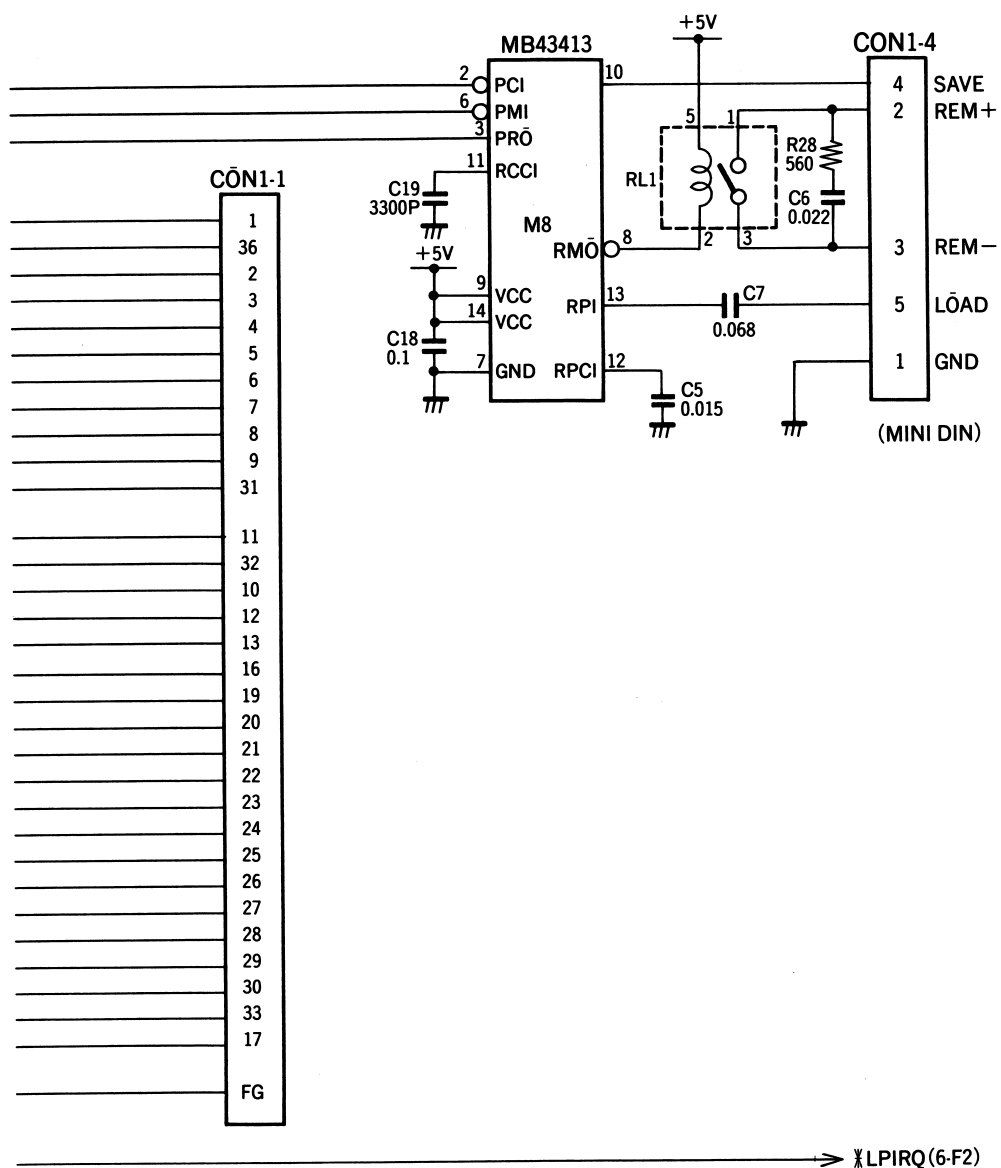


15. 漢字ROM

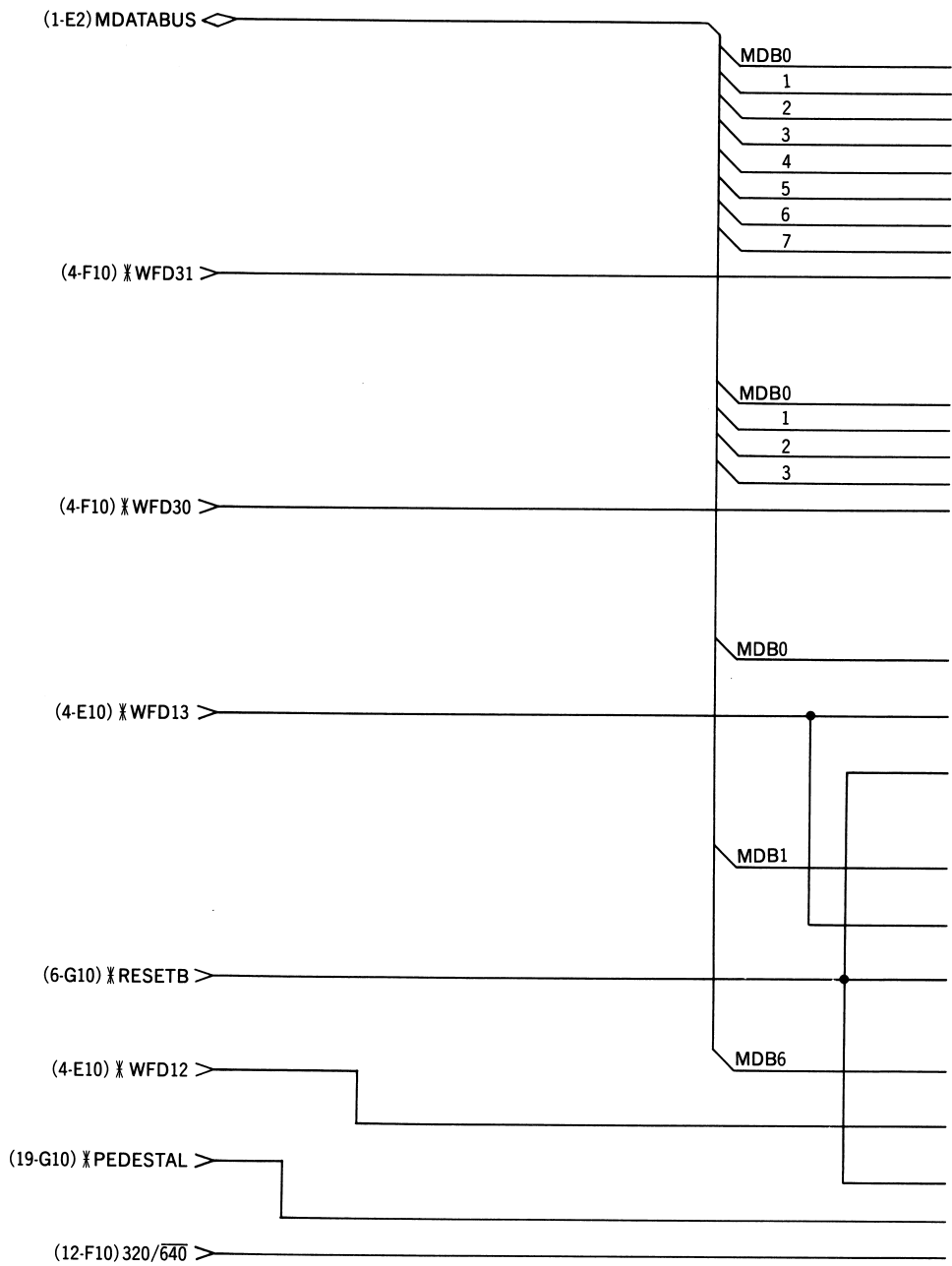


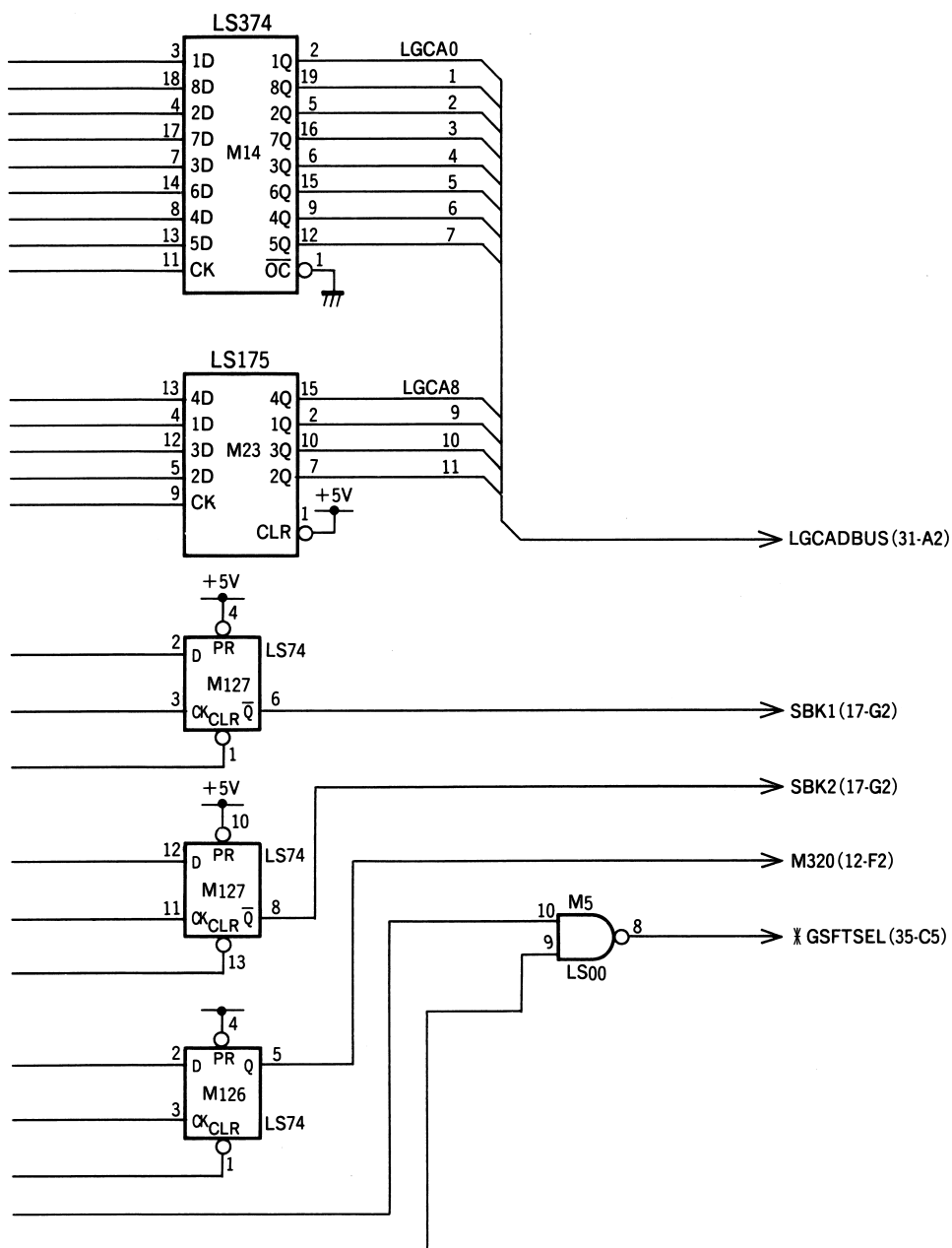




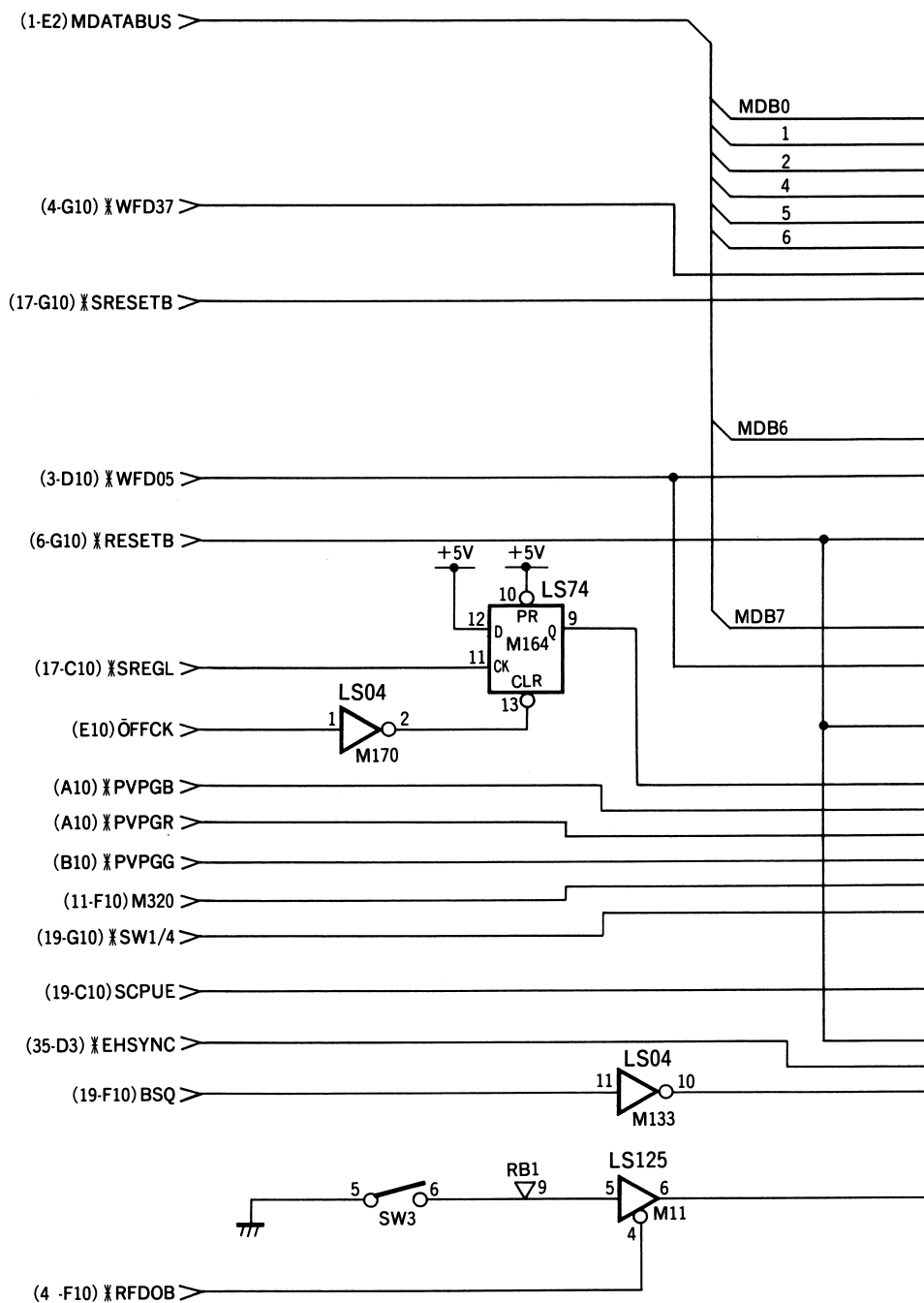


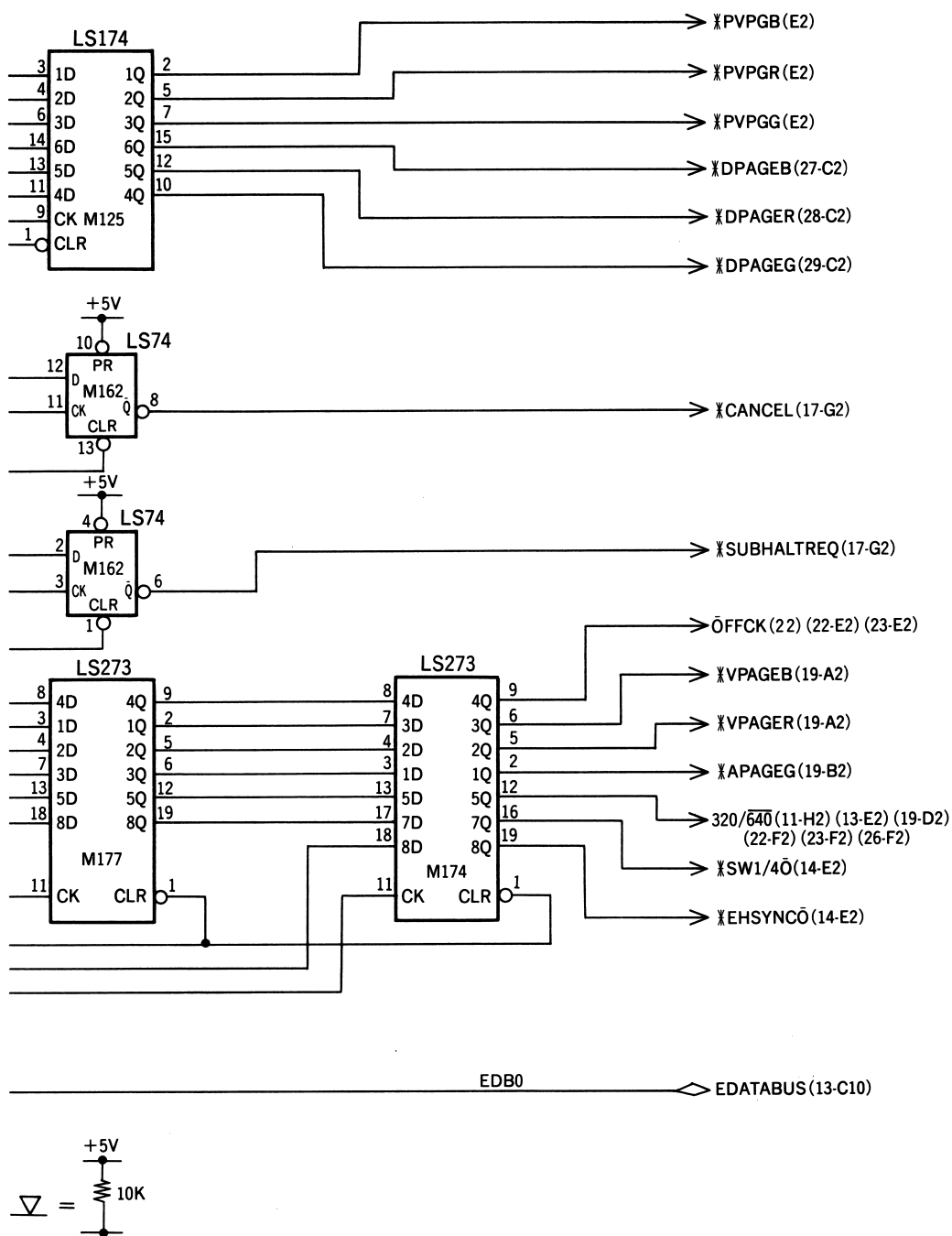
17. アナログバレットI/F, サブシステムコントロール



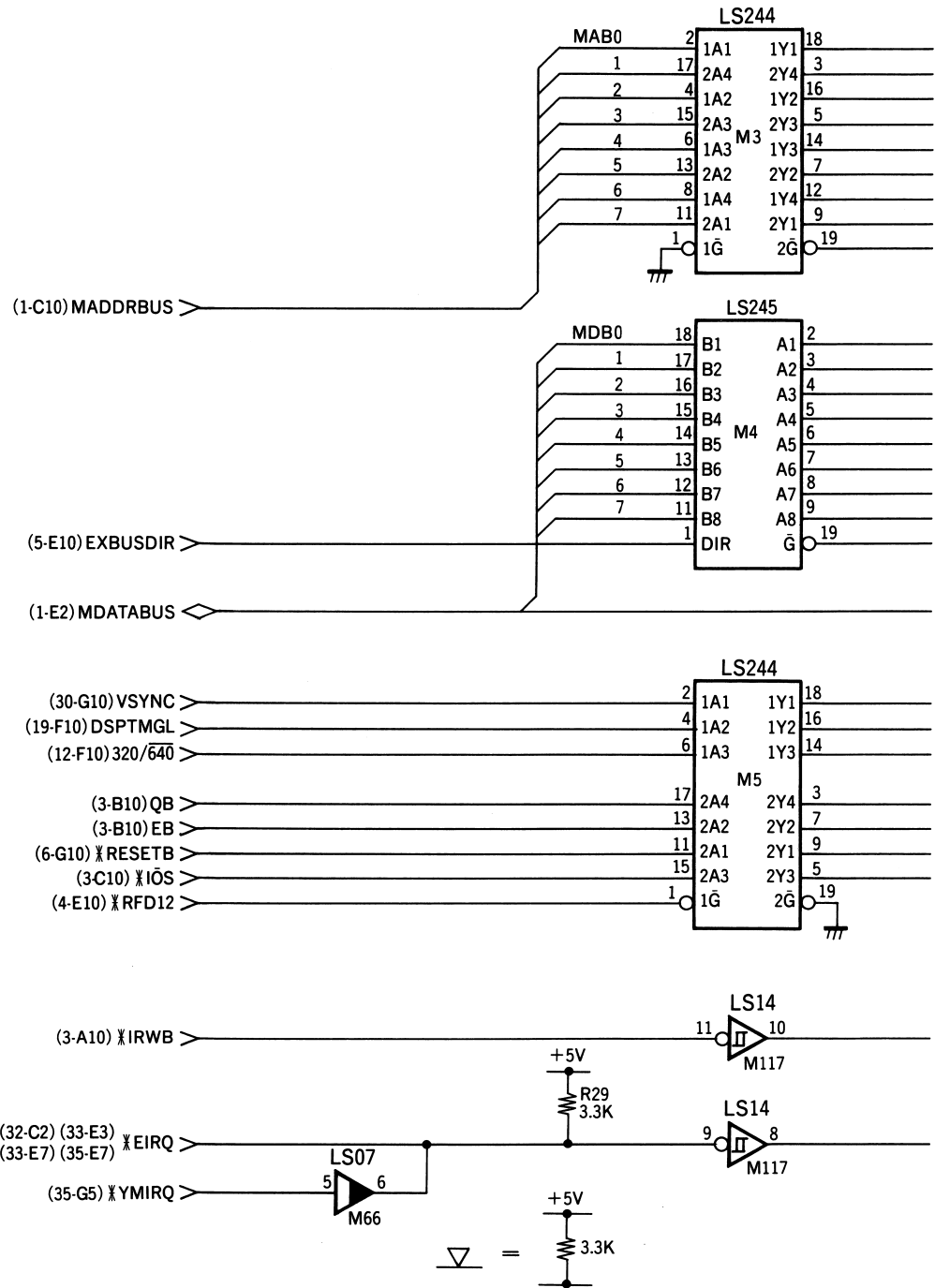


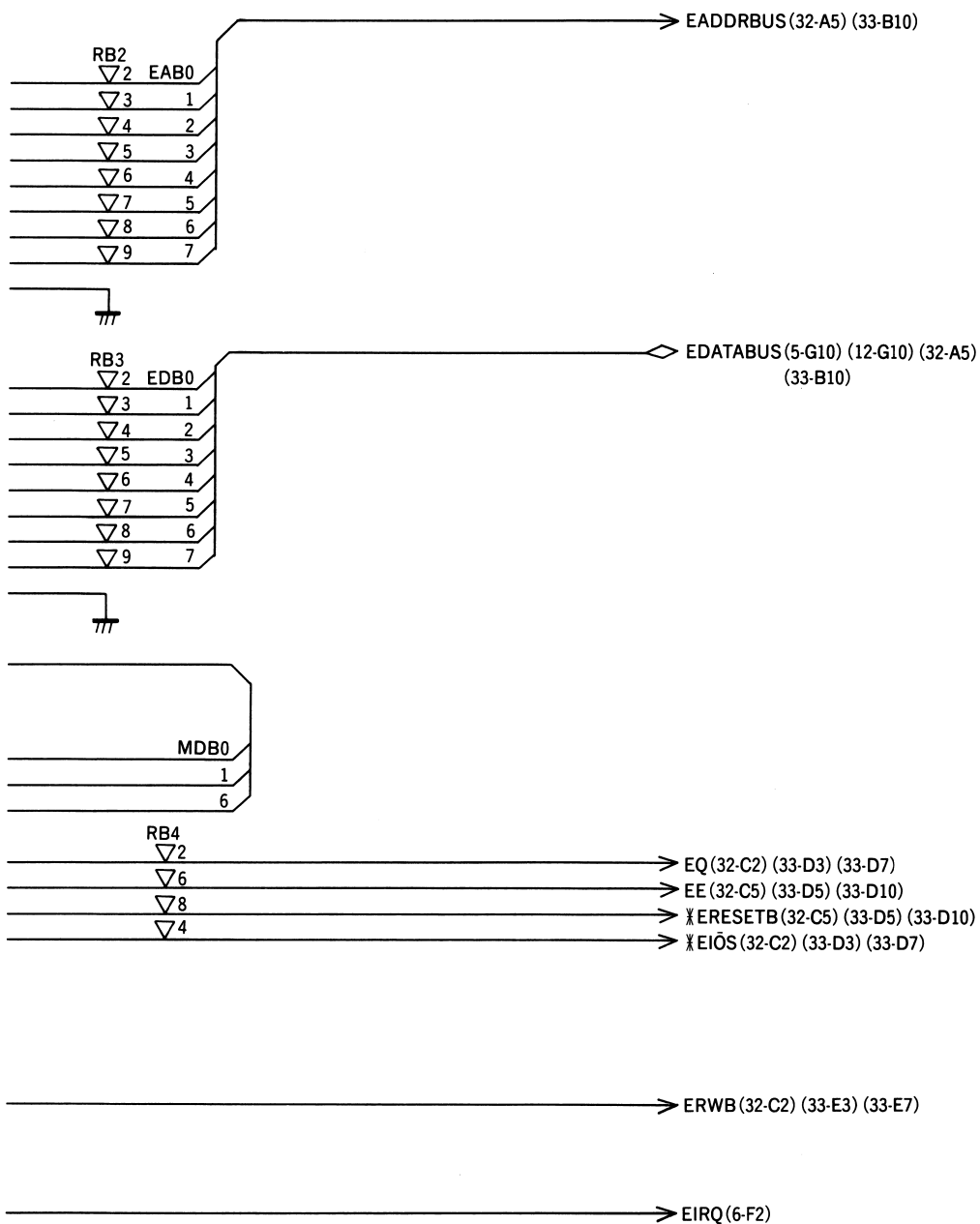
18. サブシステムコントロール, メイン-サブI/F etc.



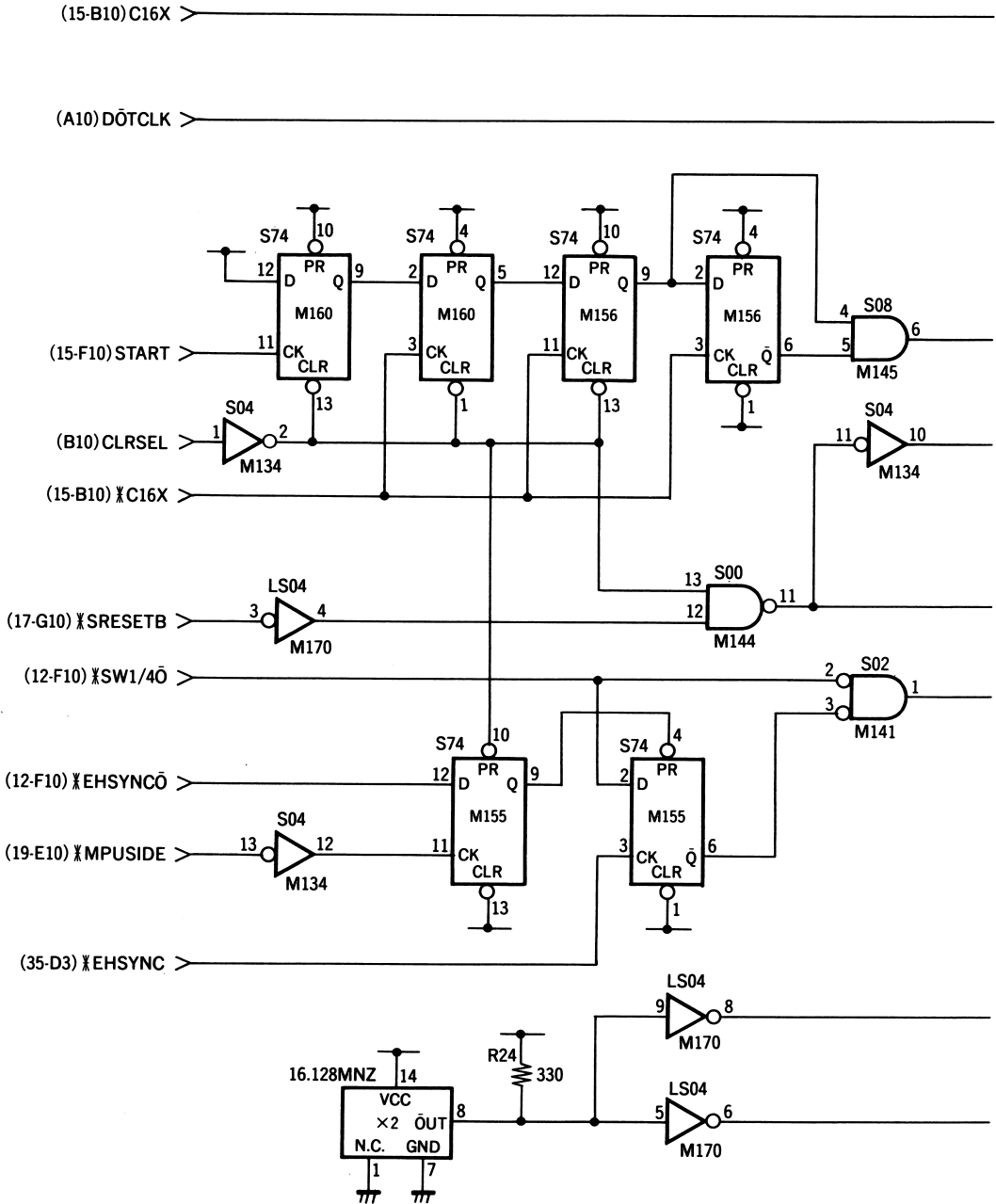


19. EXTERNAL I/Oバッファ

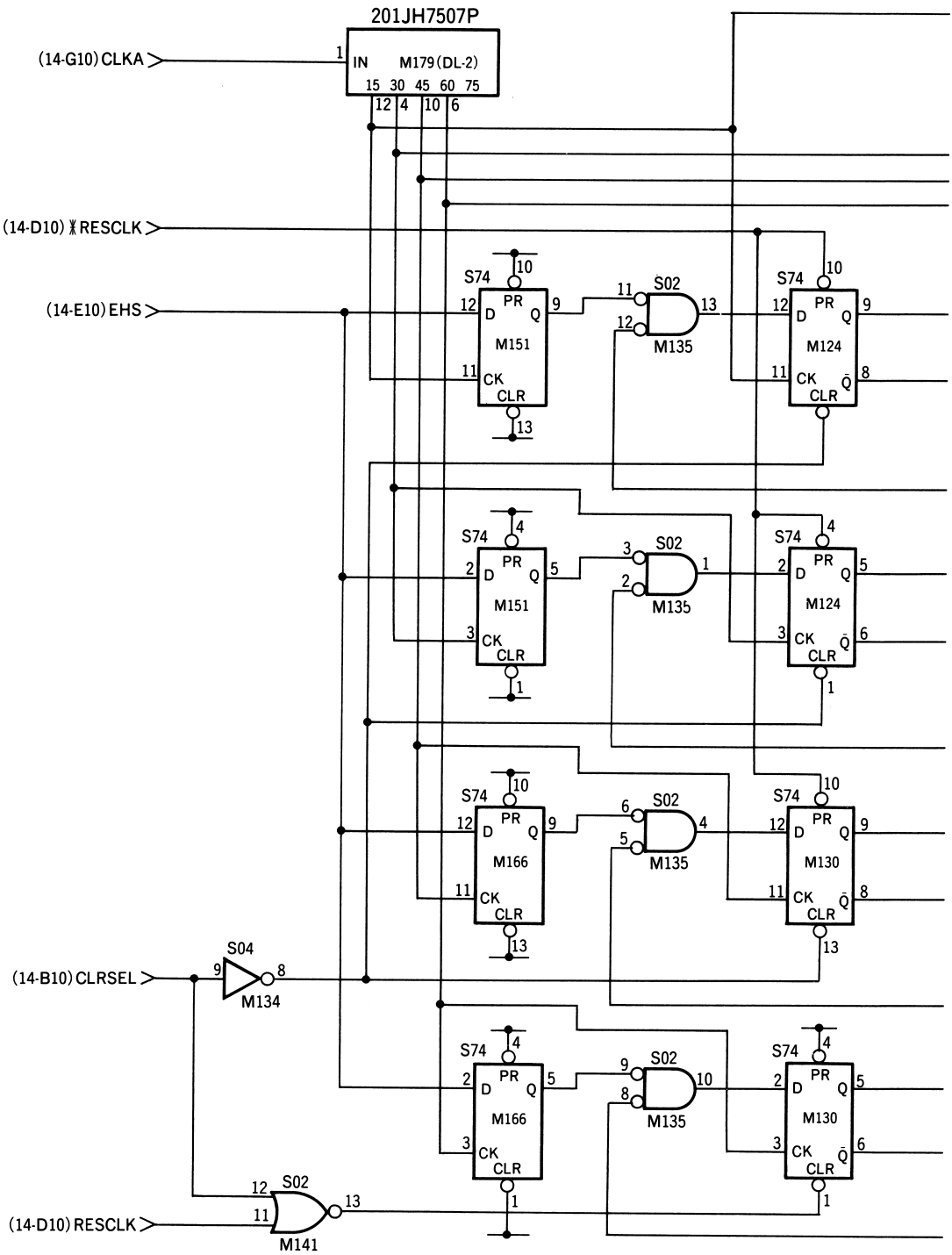


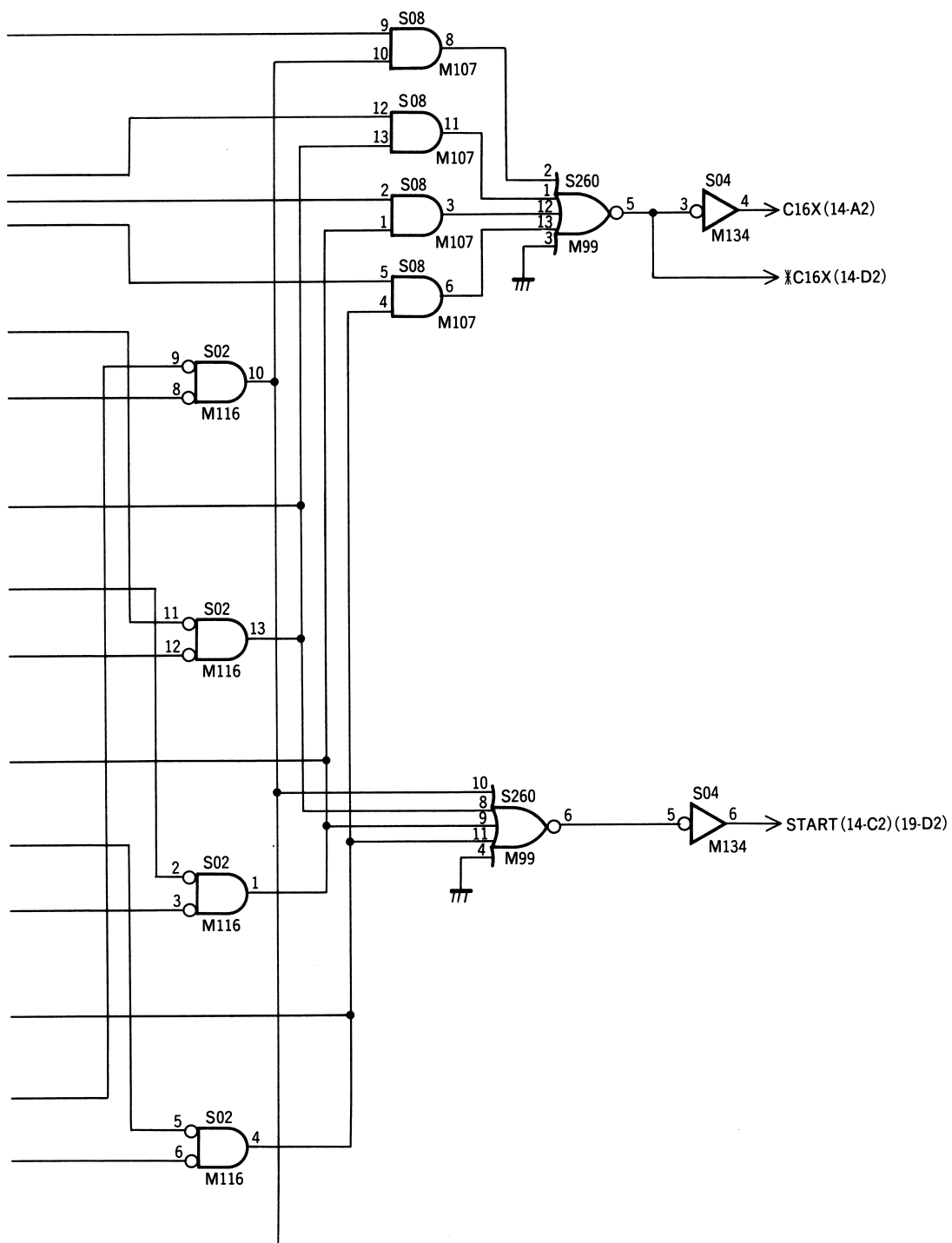


20. クロックスタート/ストップ

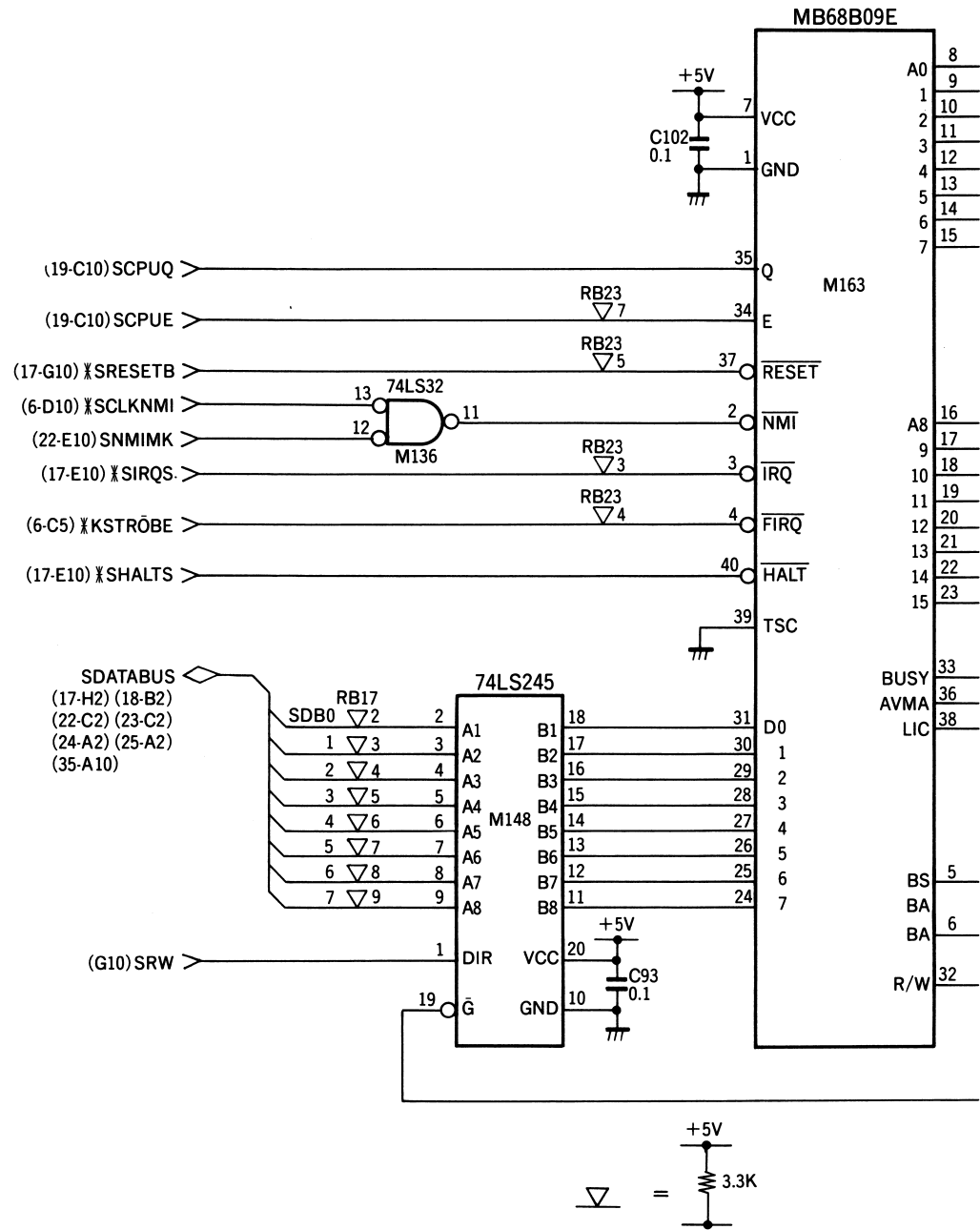


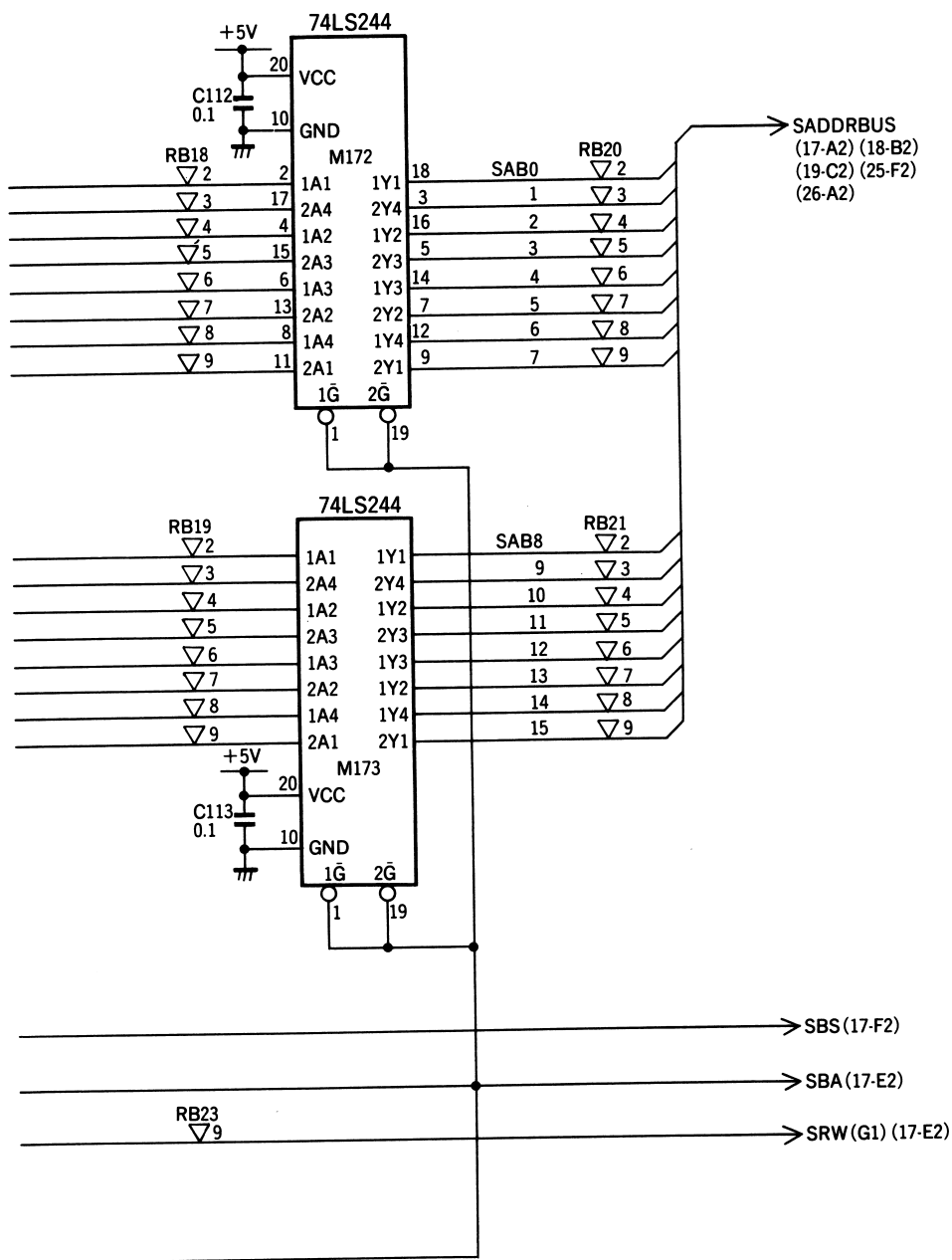
21. クロックセレクト(1/4ドット)



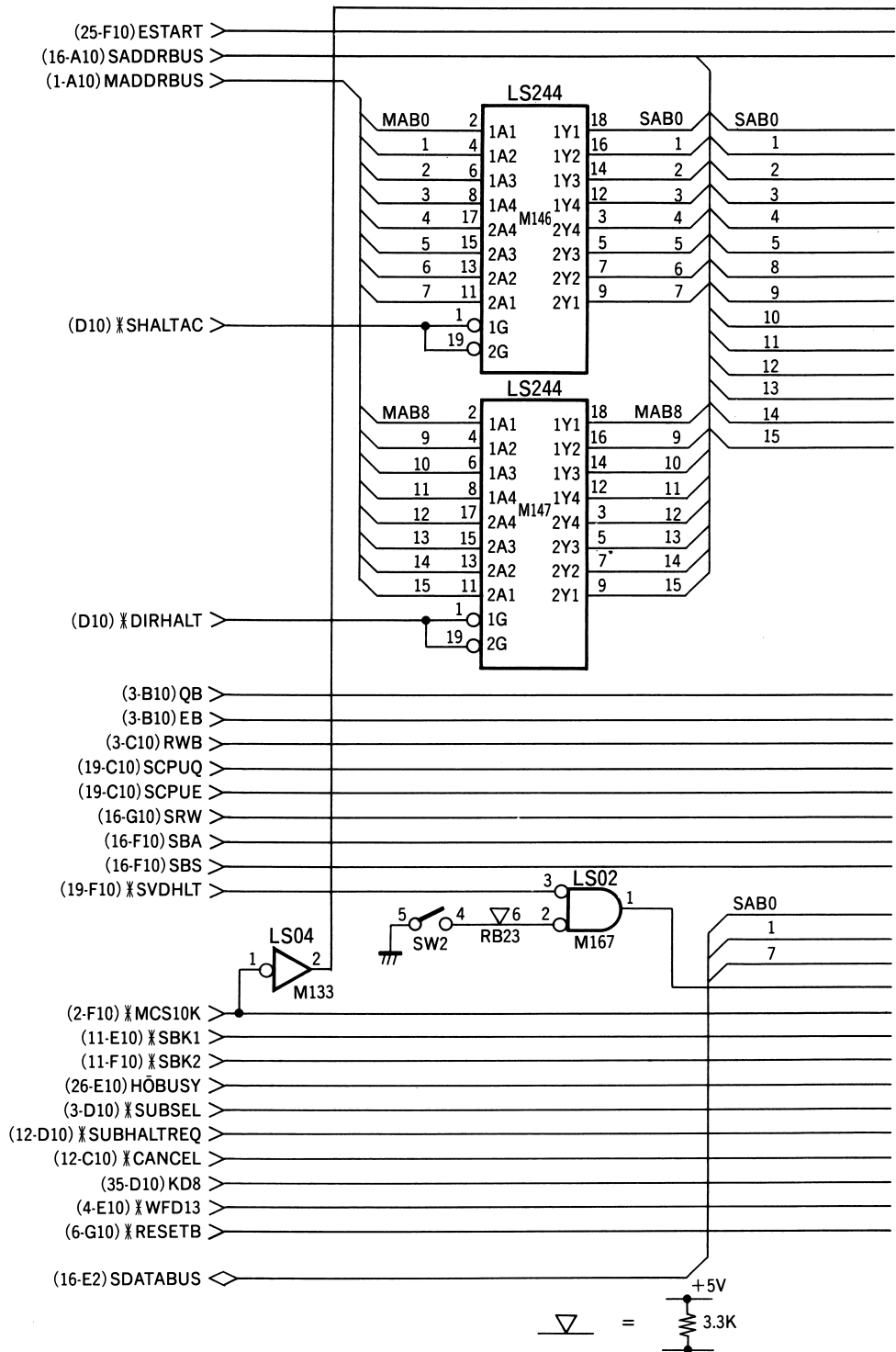


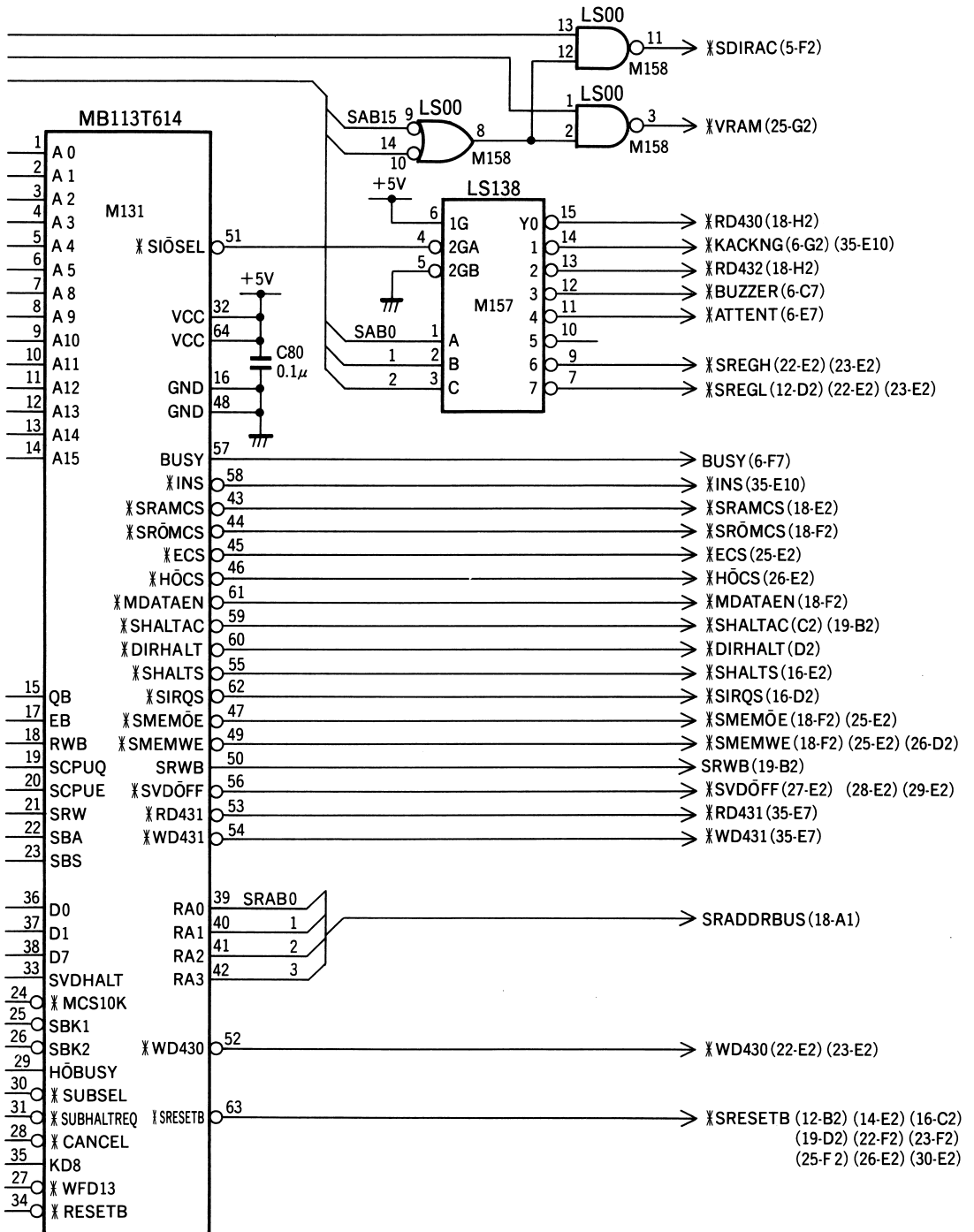
22. サブCPU



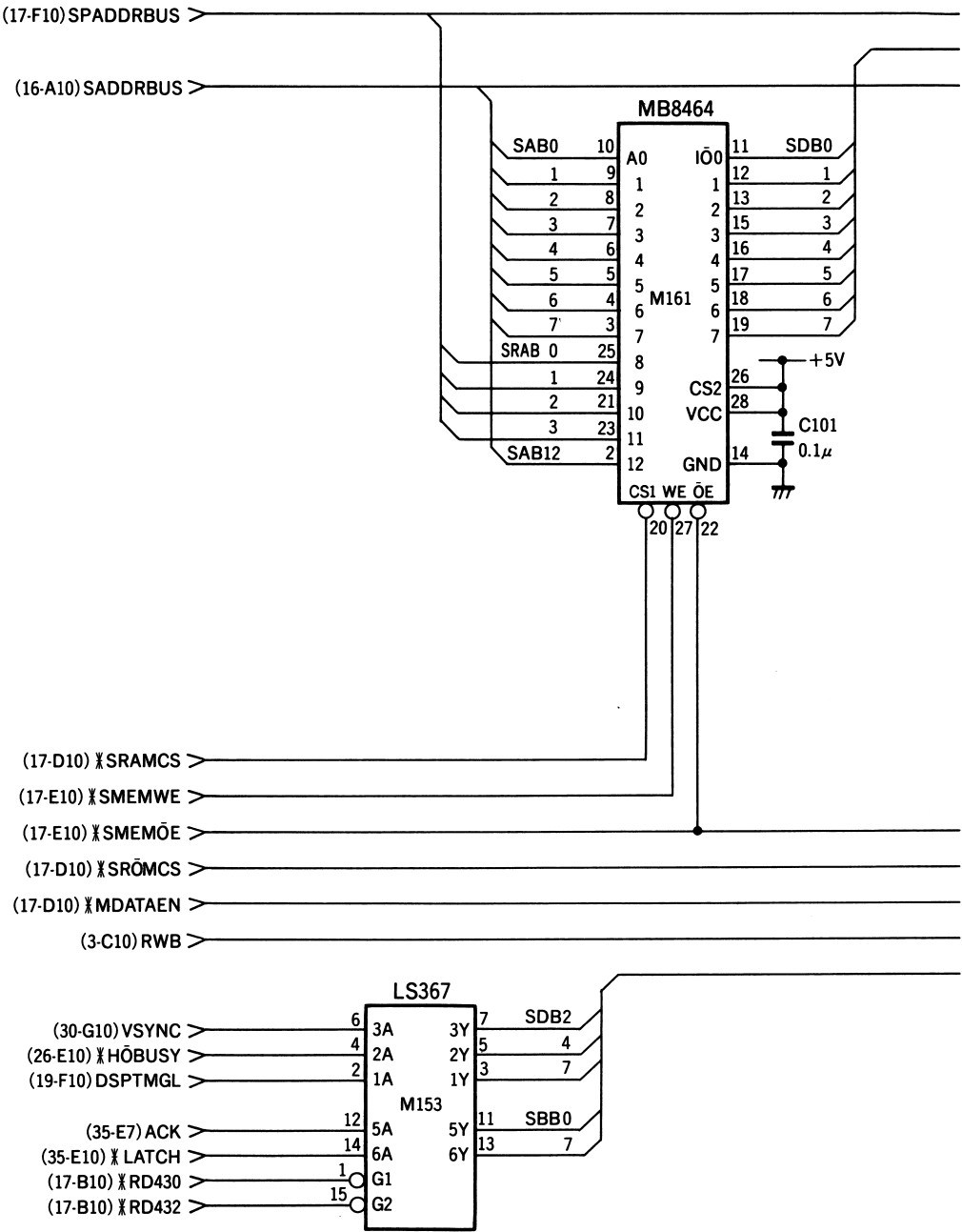


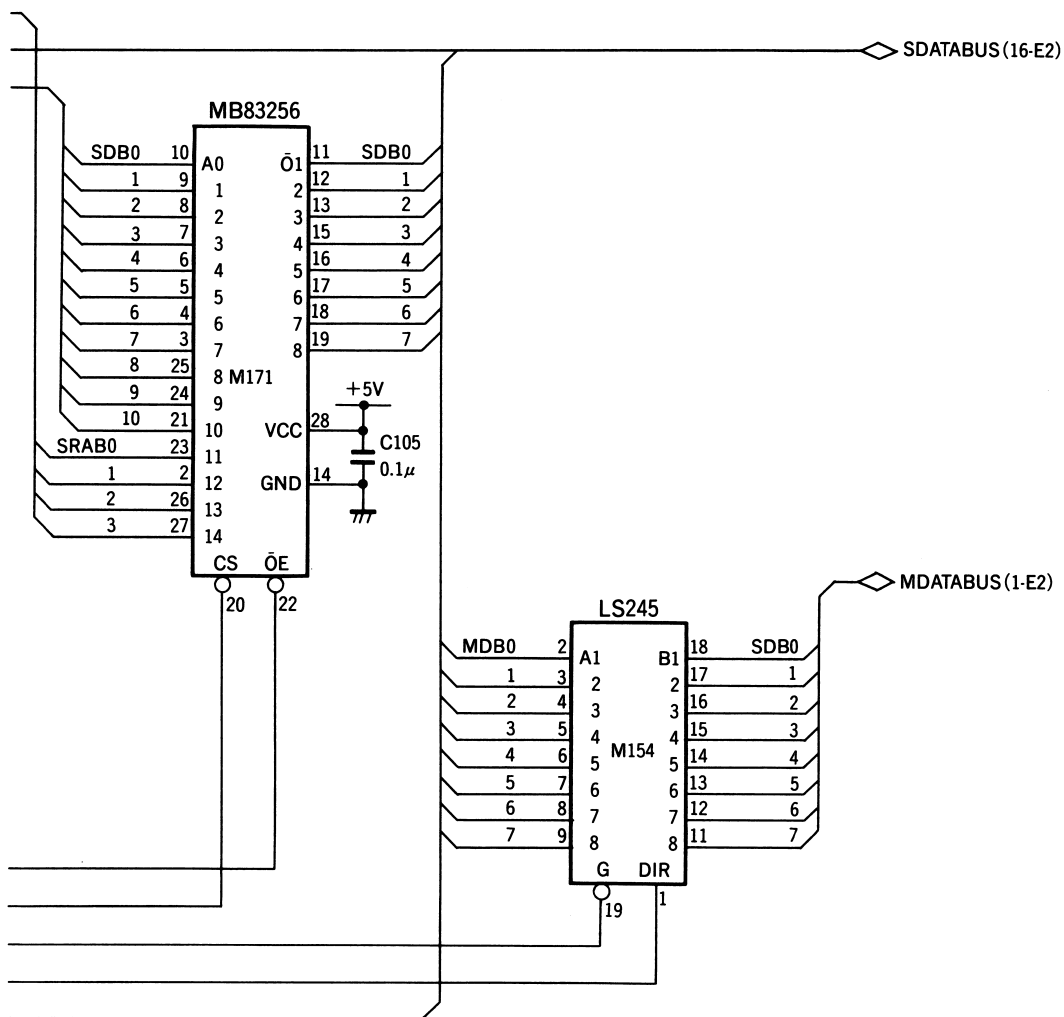
23. サブデコーダー



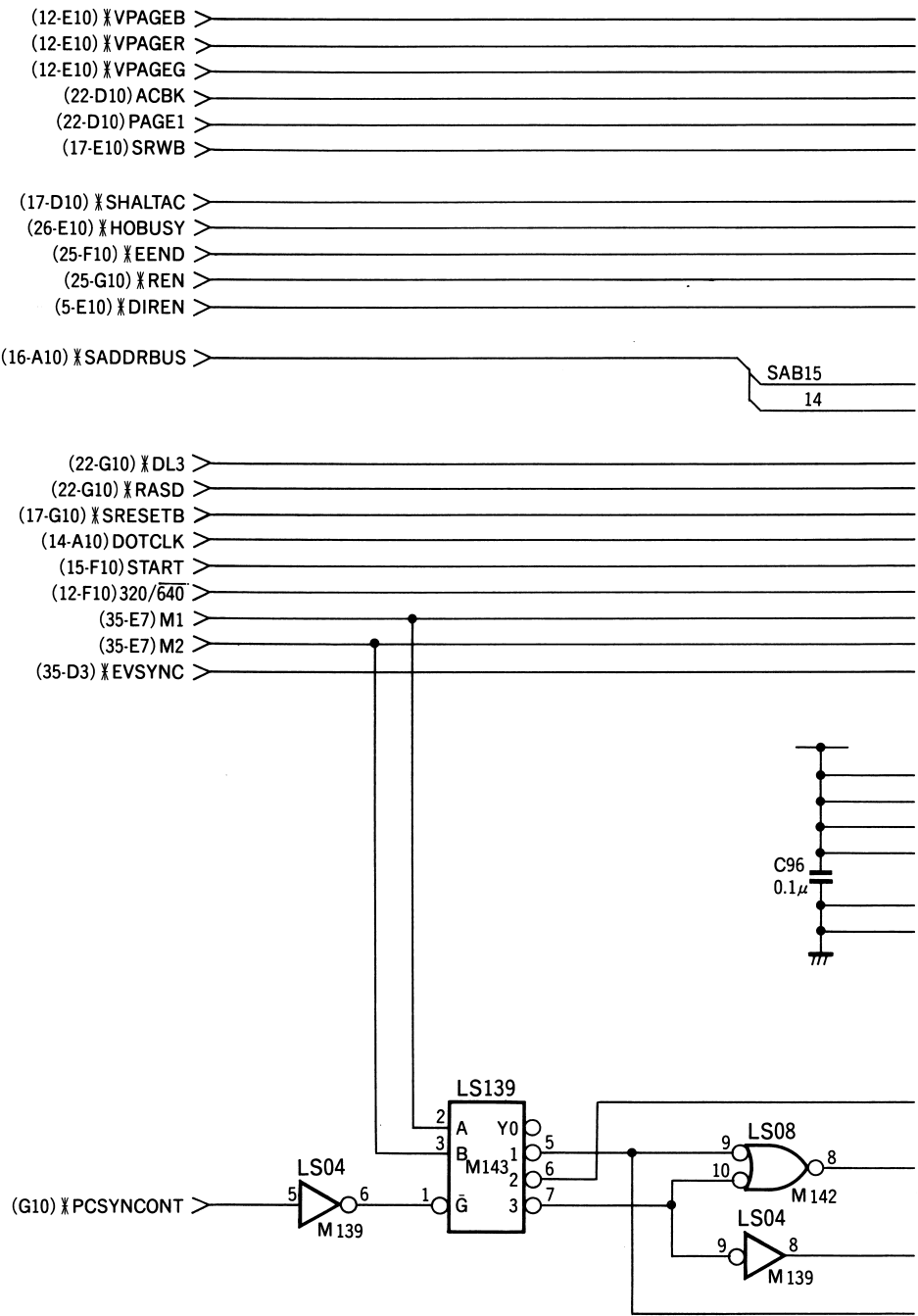


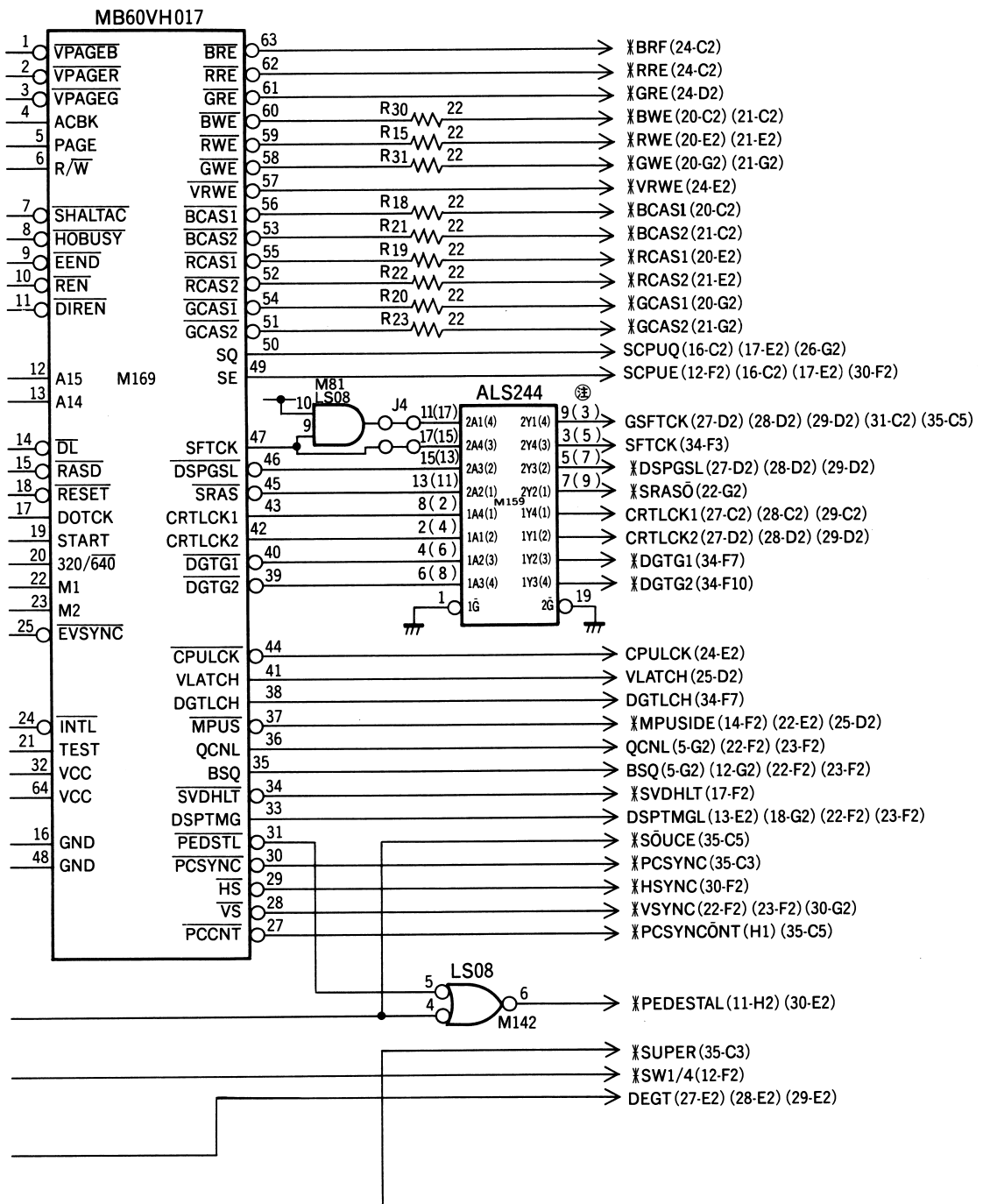
24. サブROM/RAM



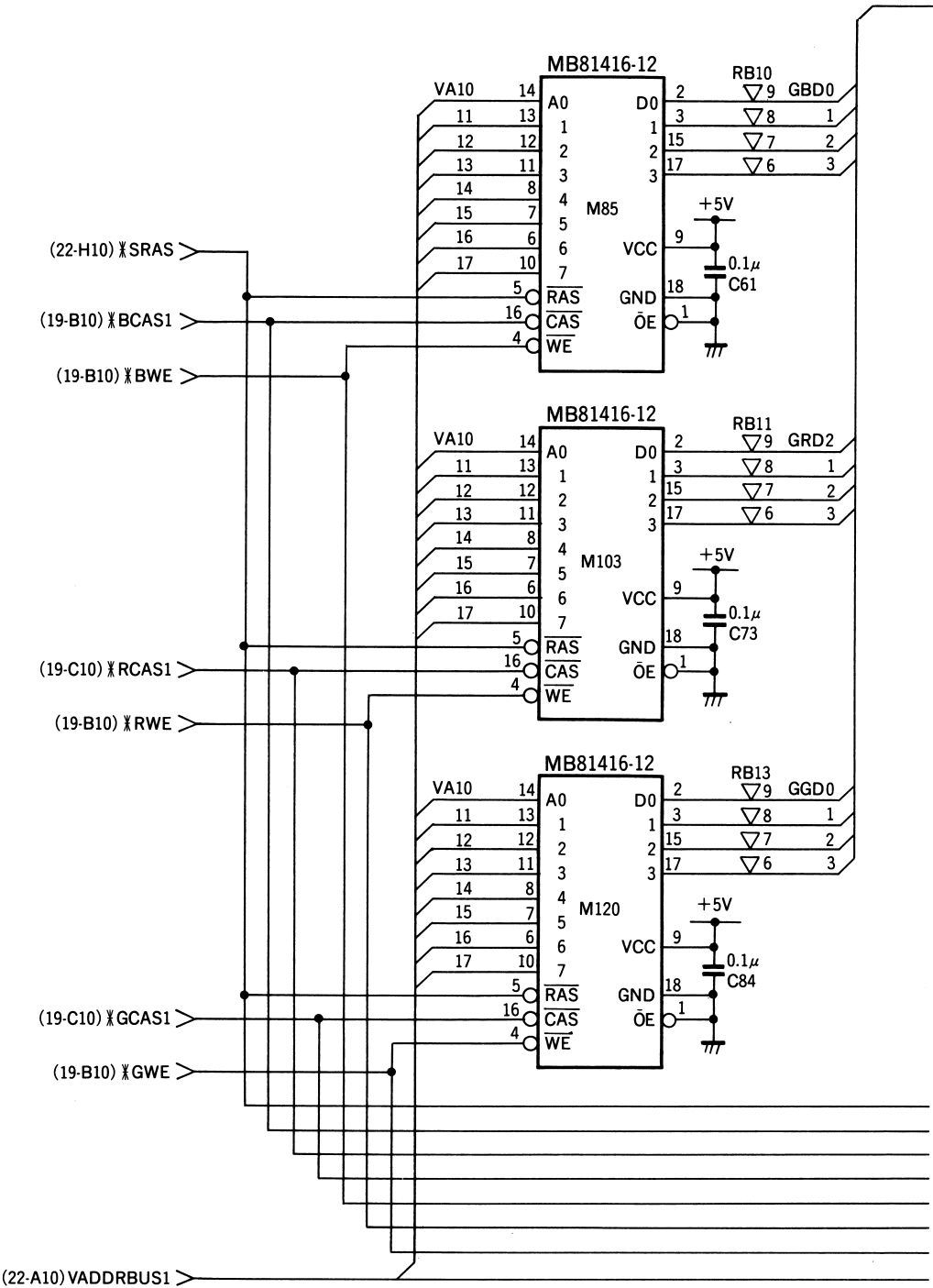


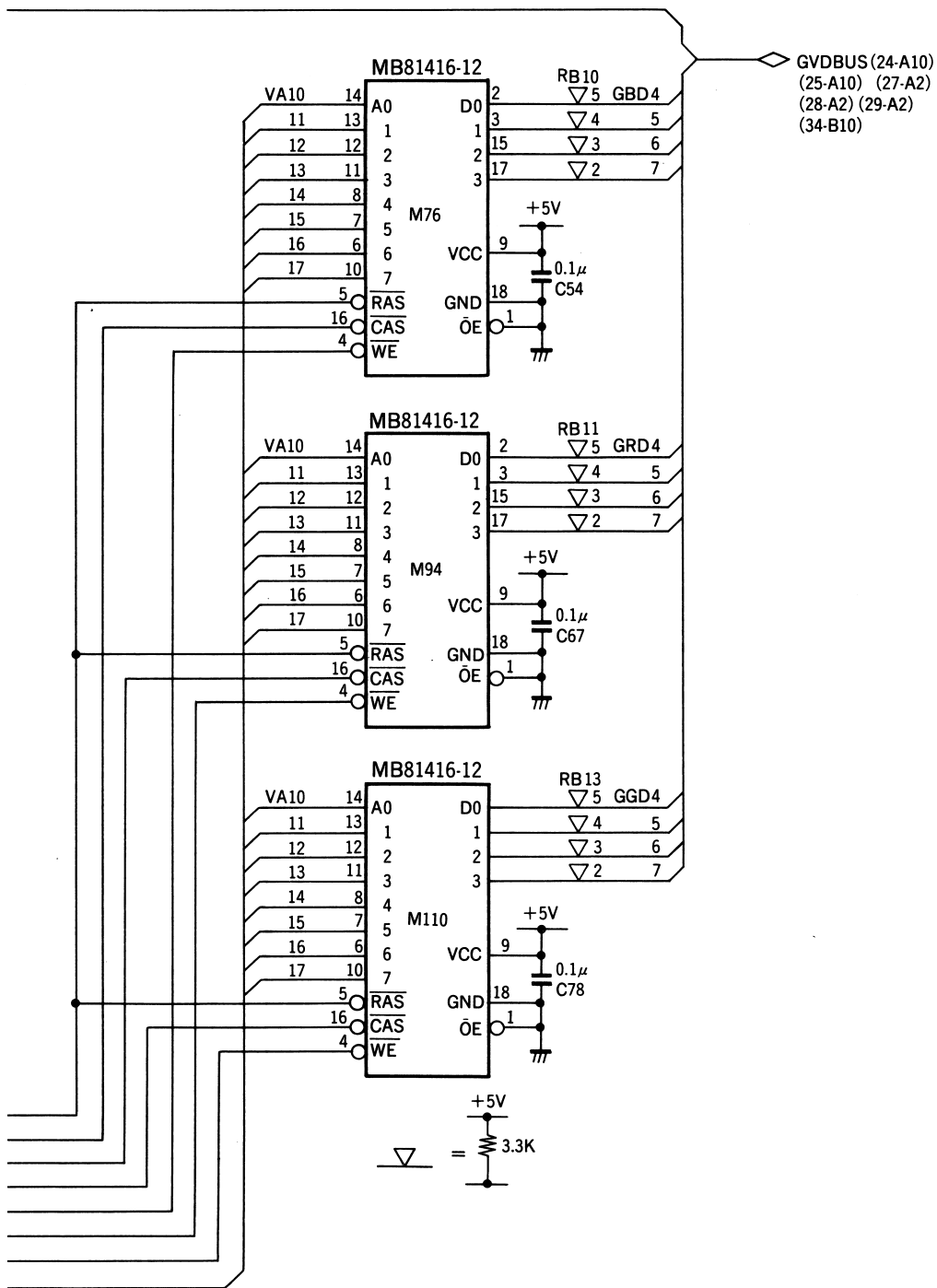
25. サブタイミングジェネレータ



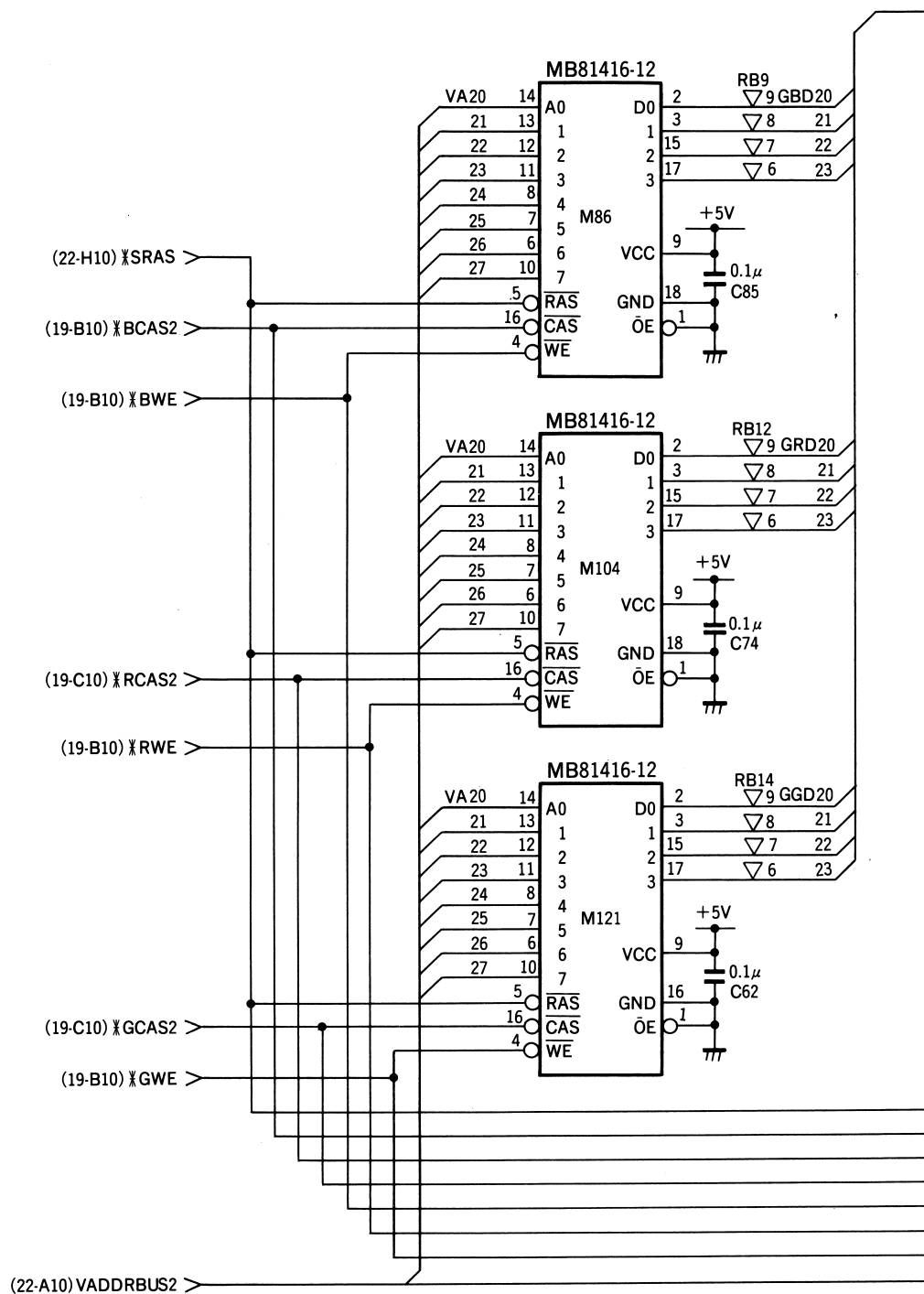


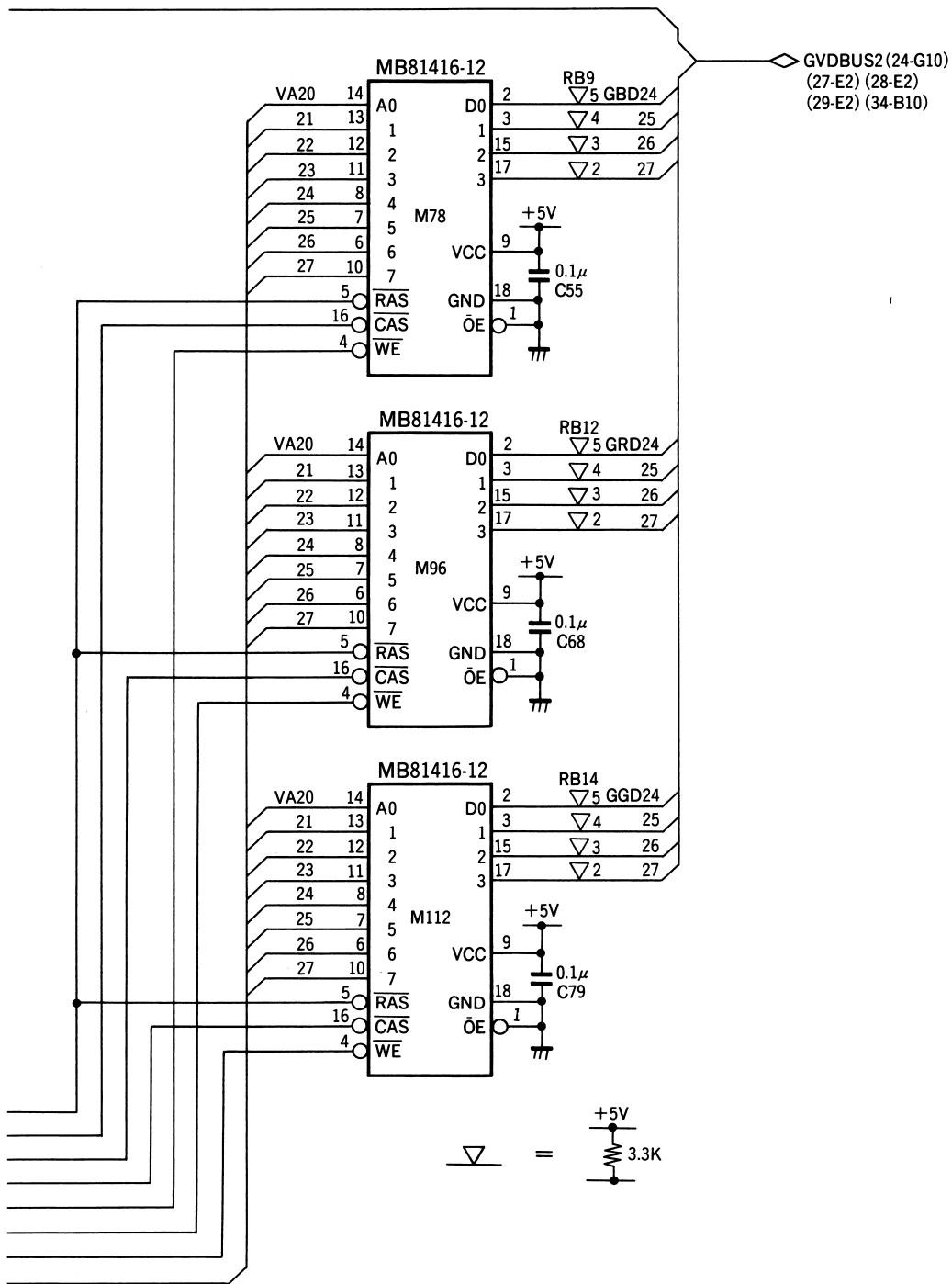
26. VRAM(BANK0)



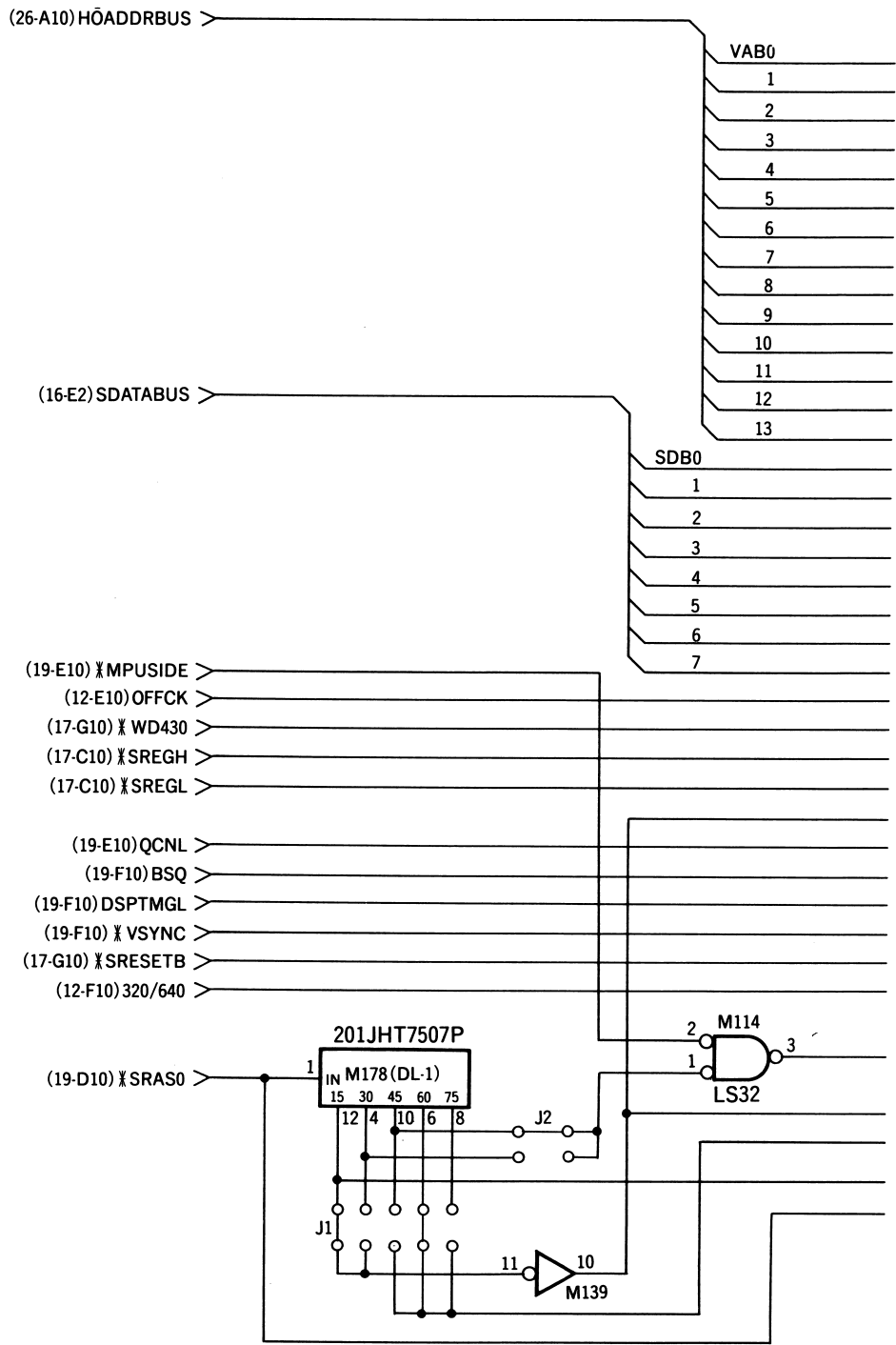


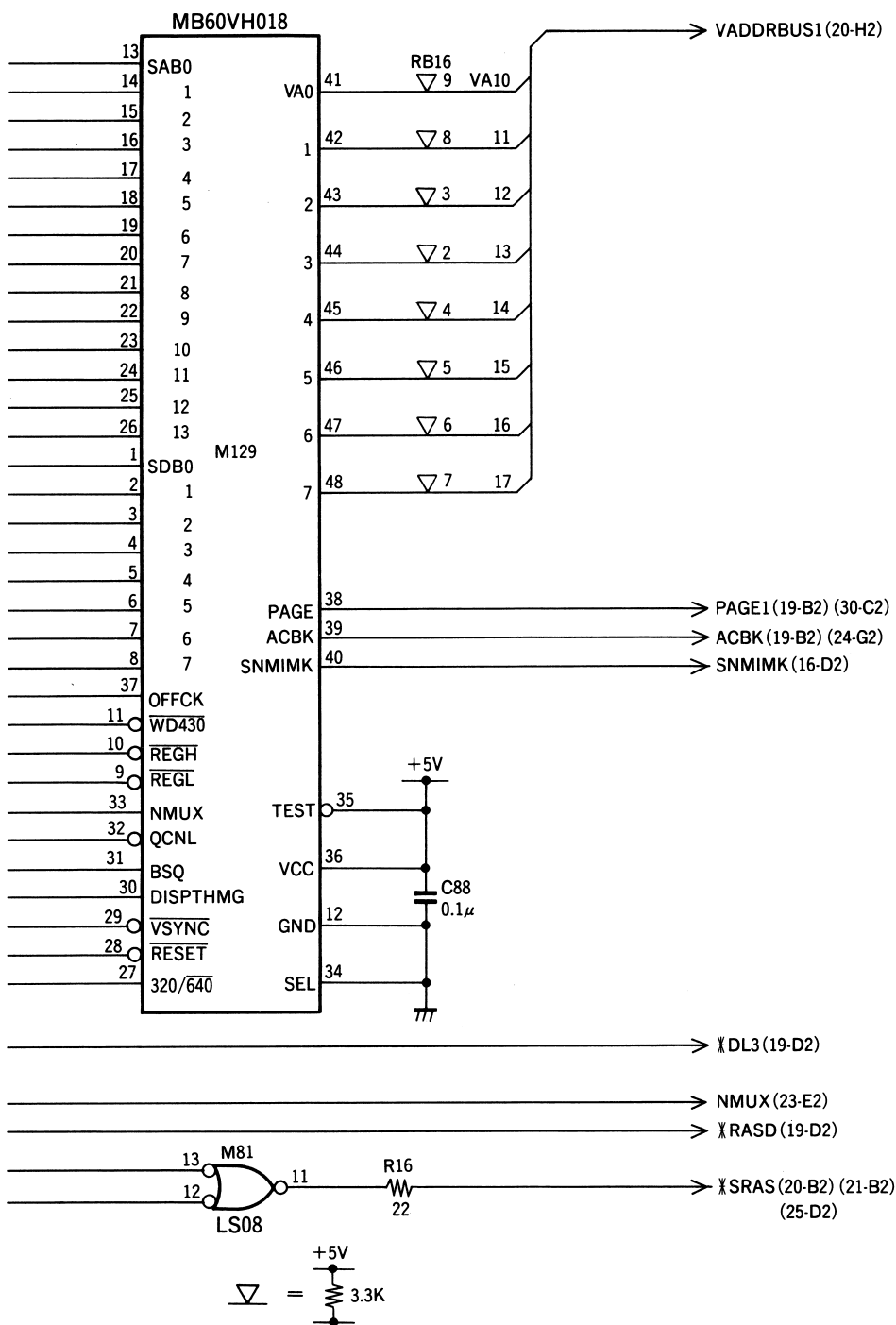
27. VRAM(BANK1)



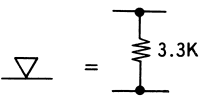
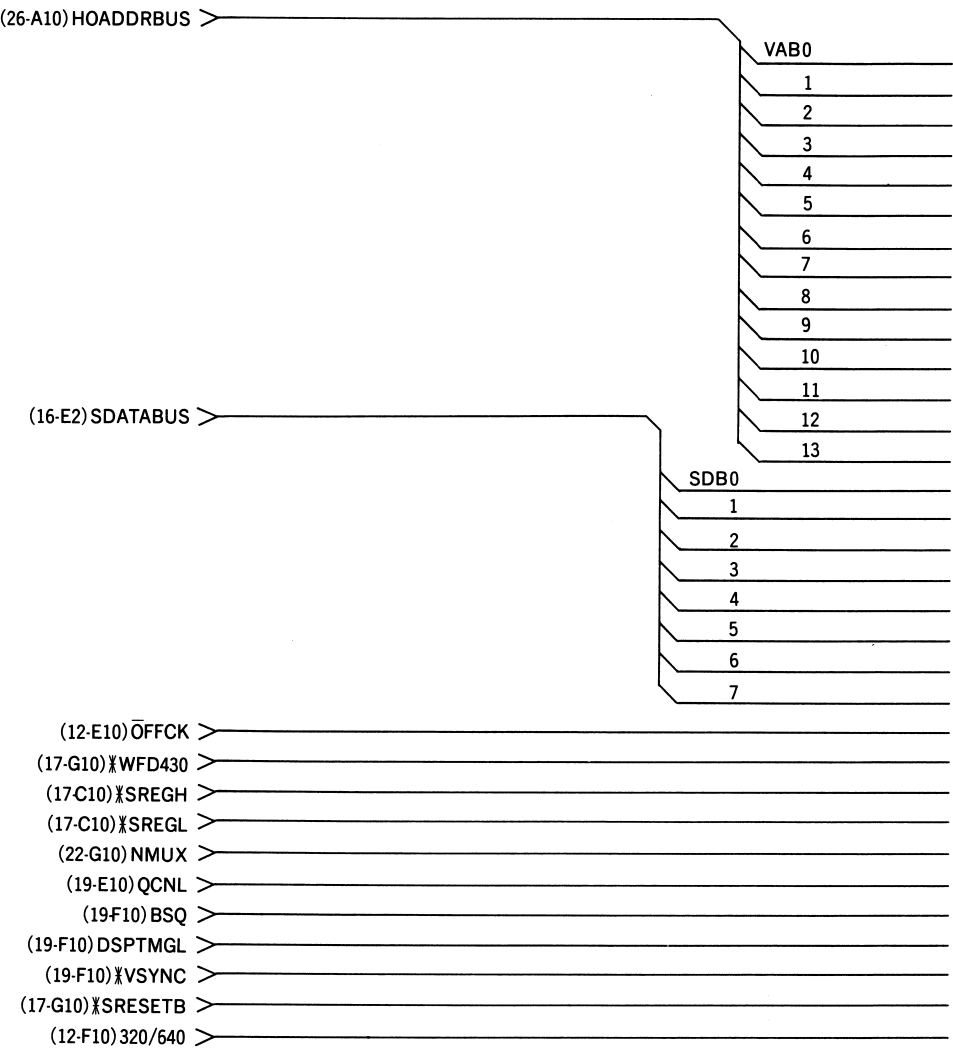


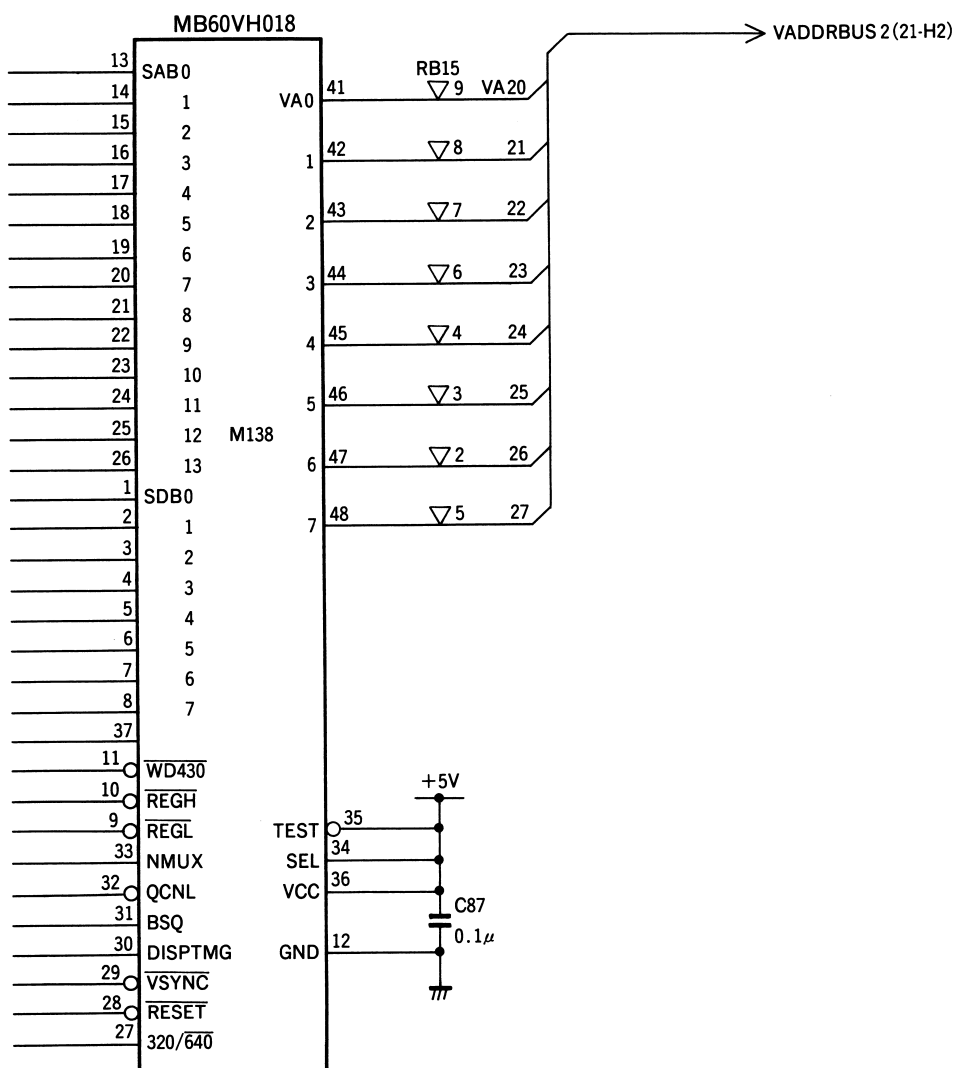
28. アドレスジェネレータ(BANK0)



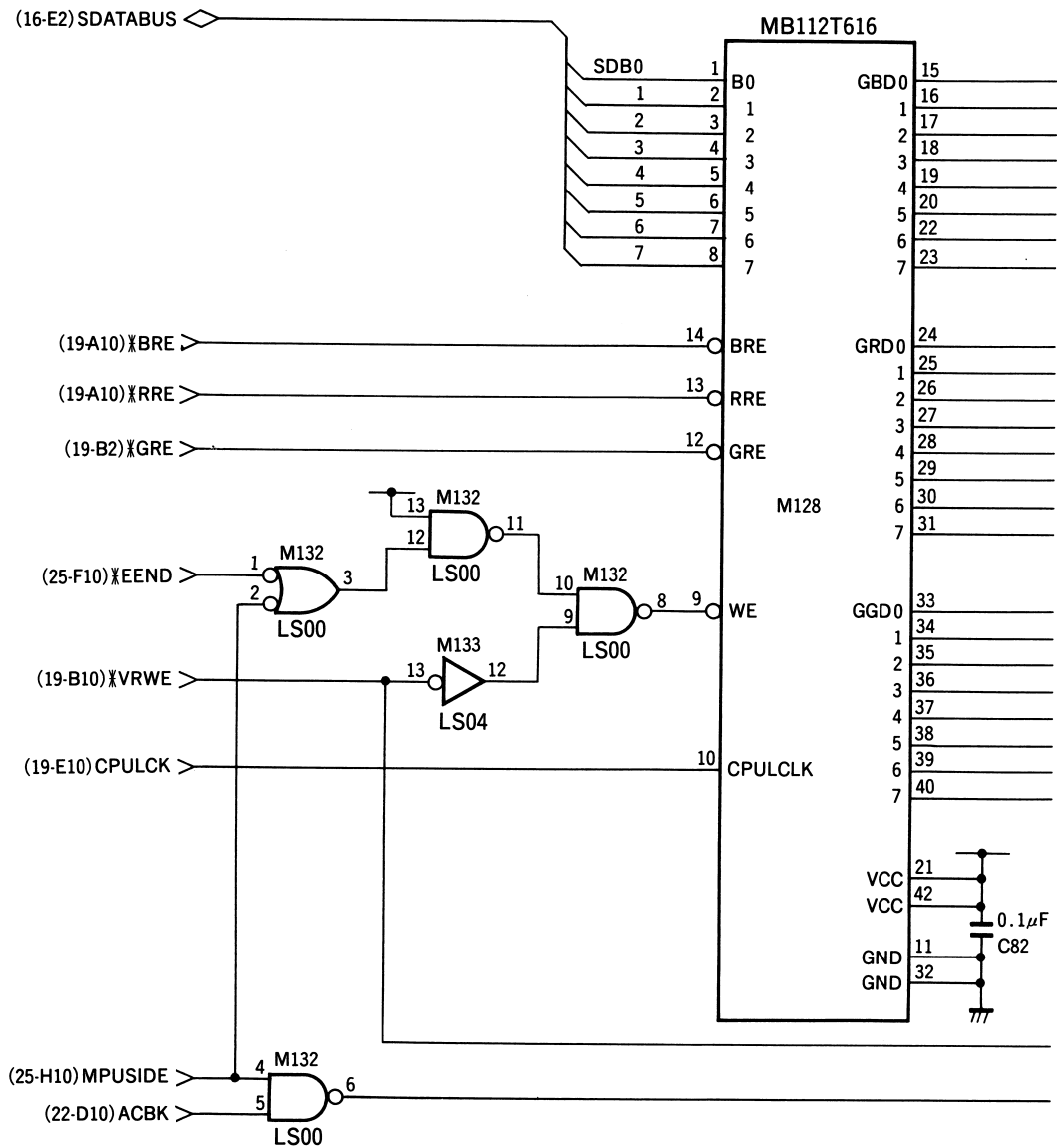


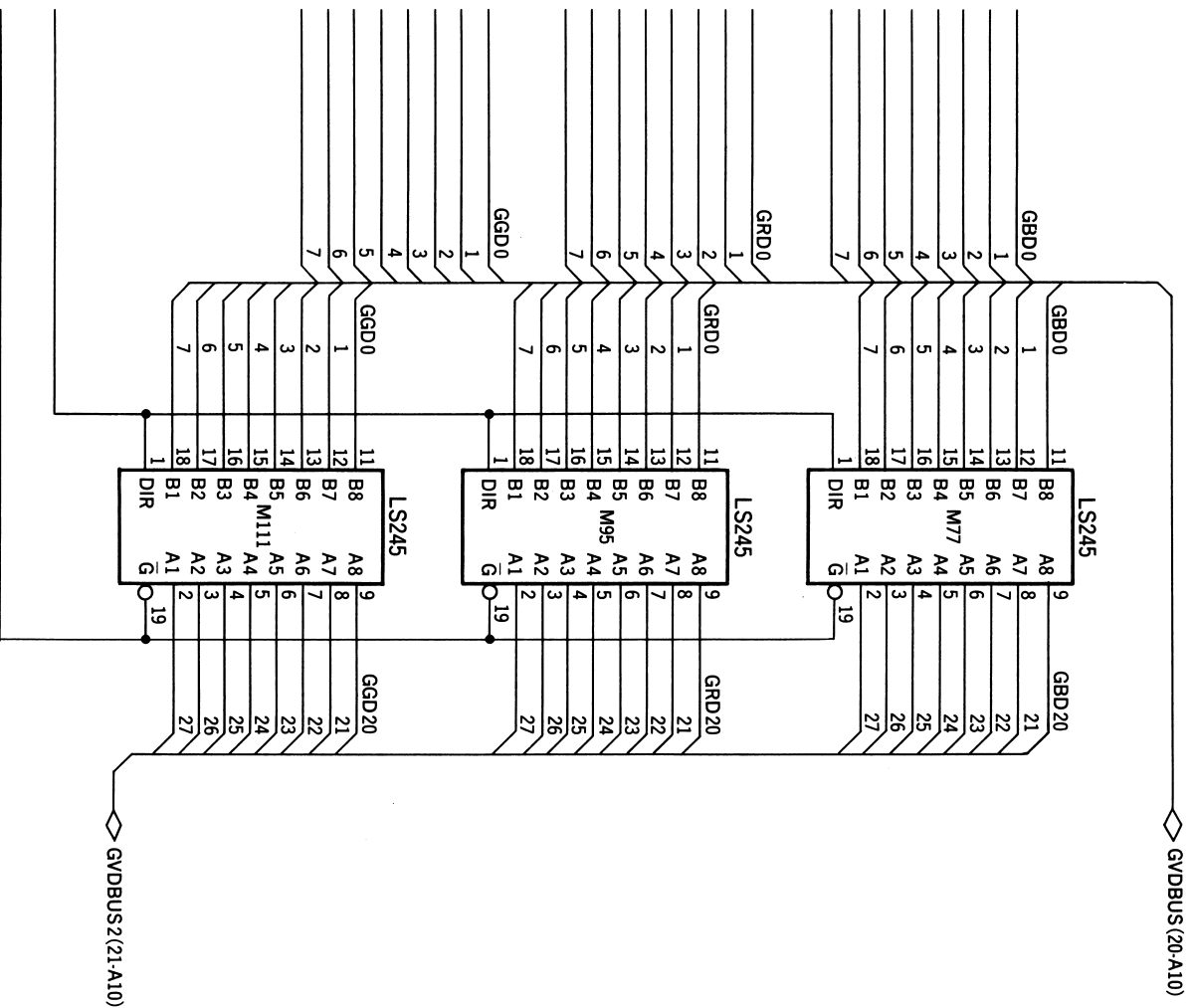
29. アドレスジェネレータ(BANK1)



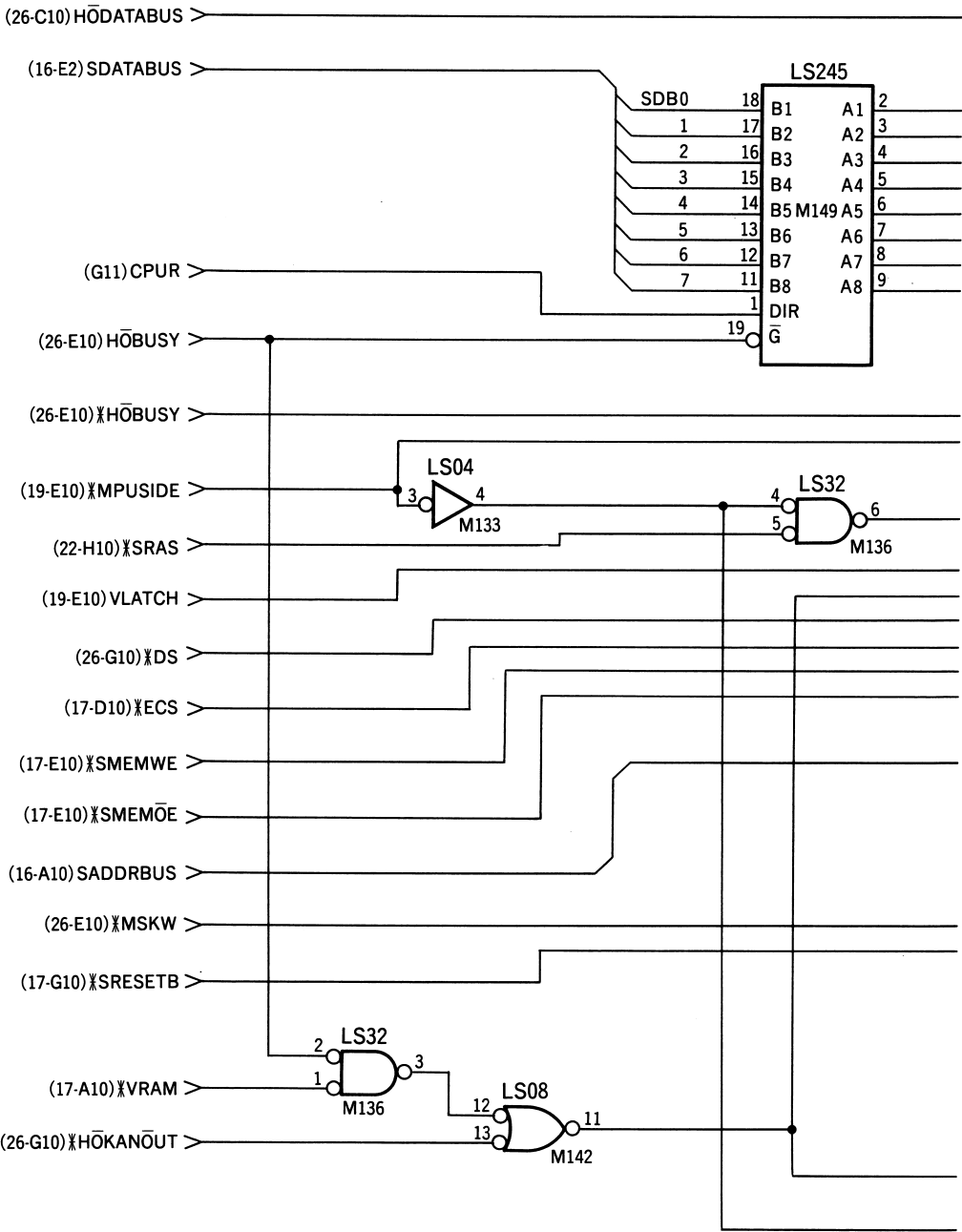


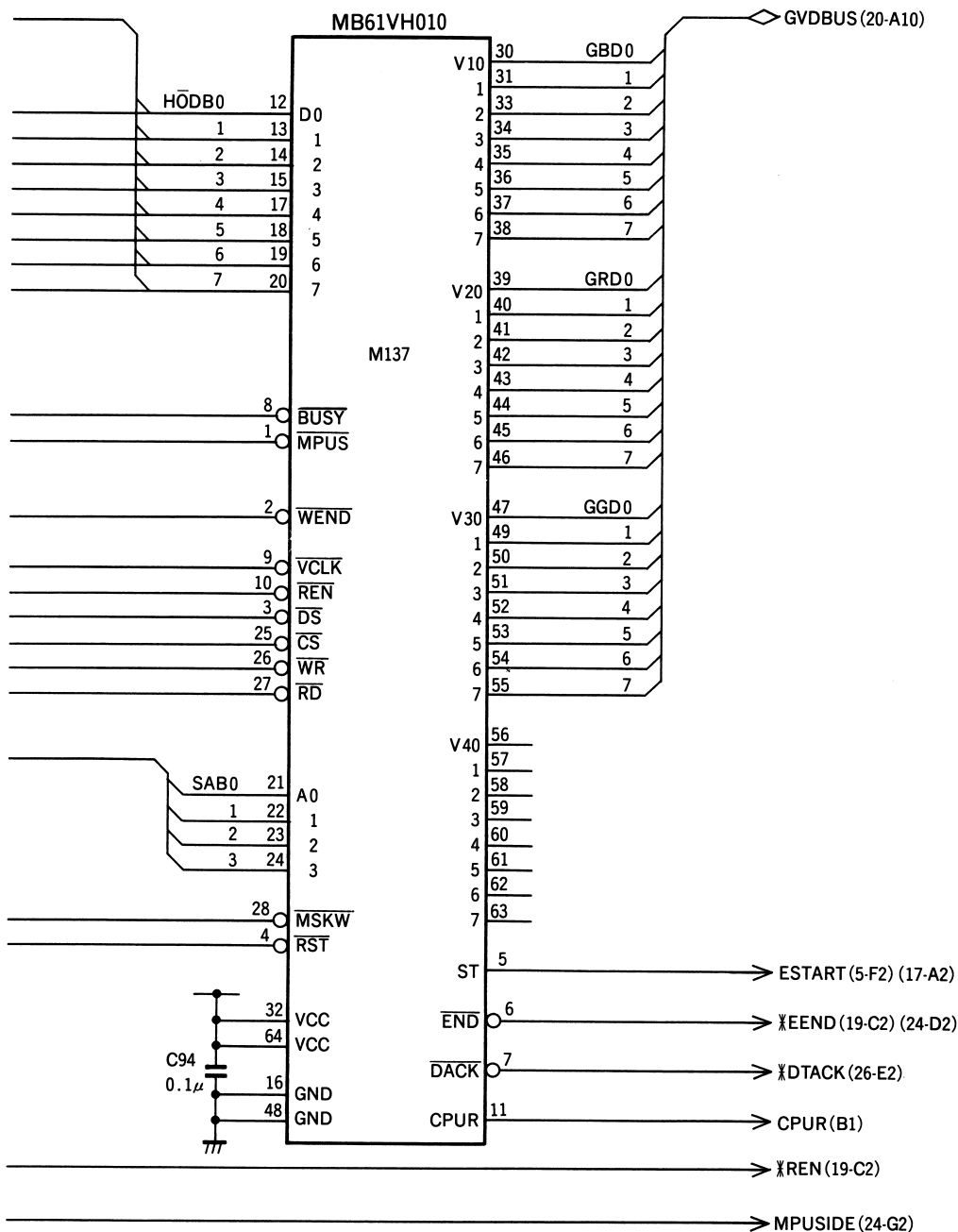
30. VRAMバッファ



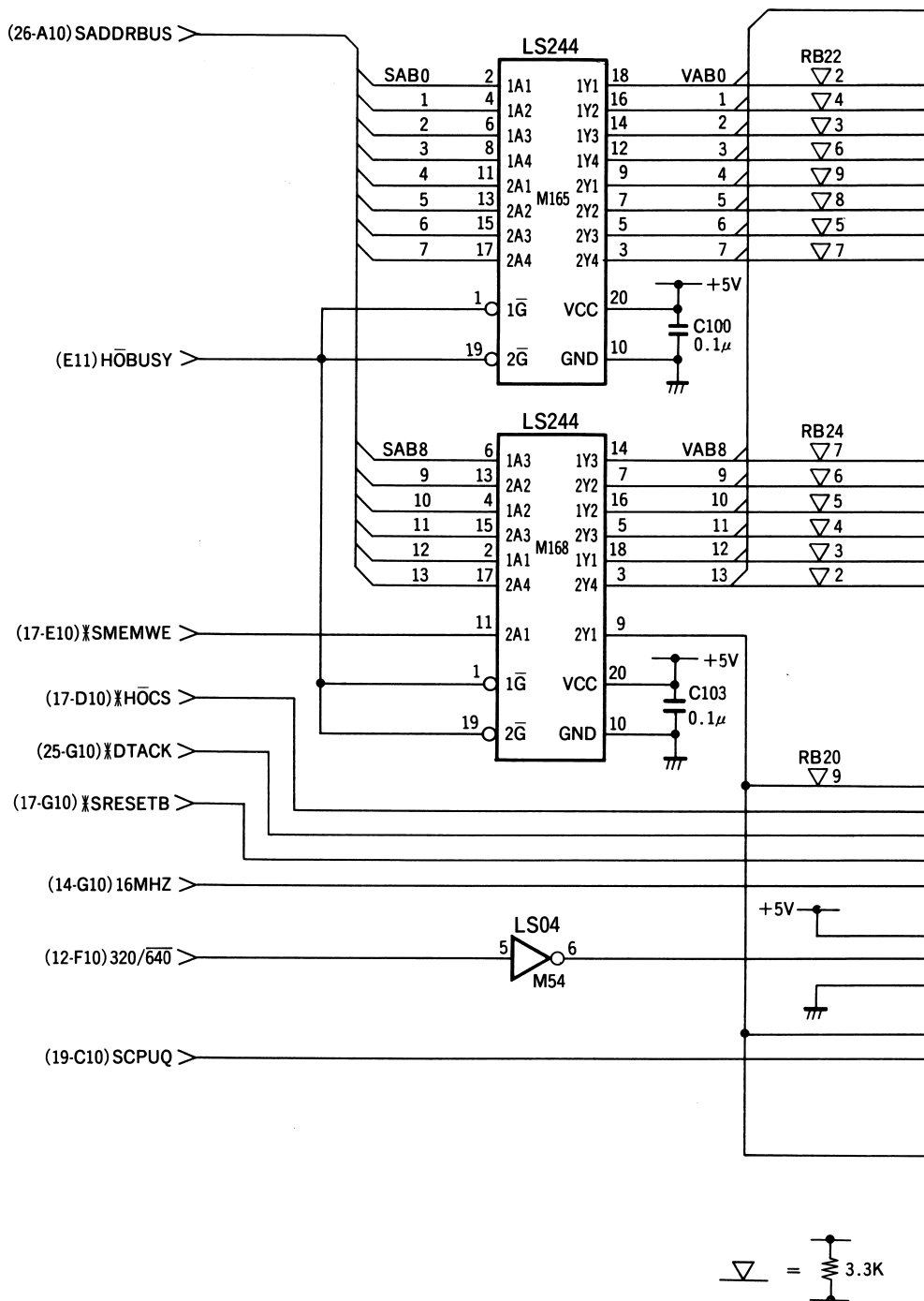


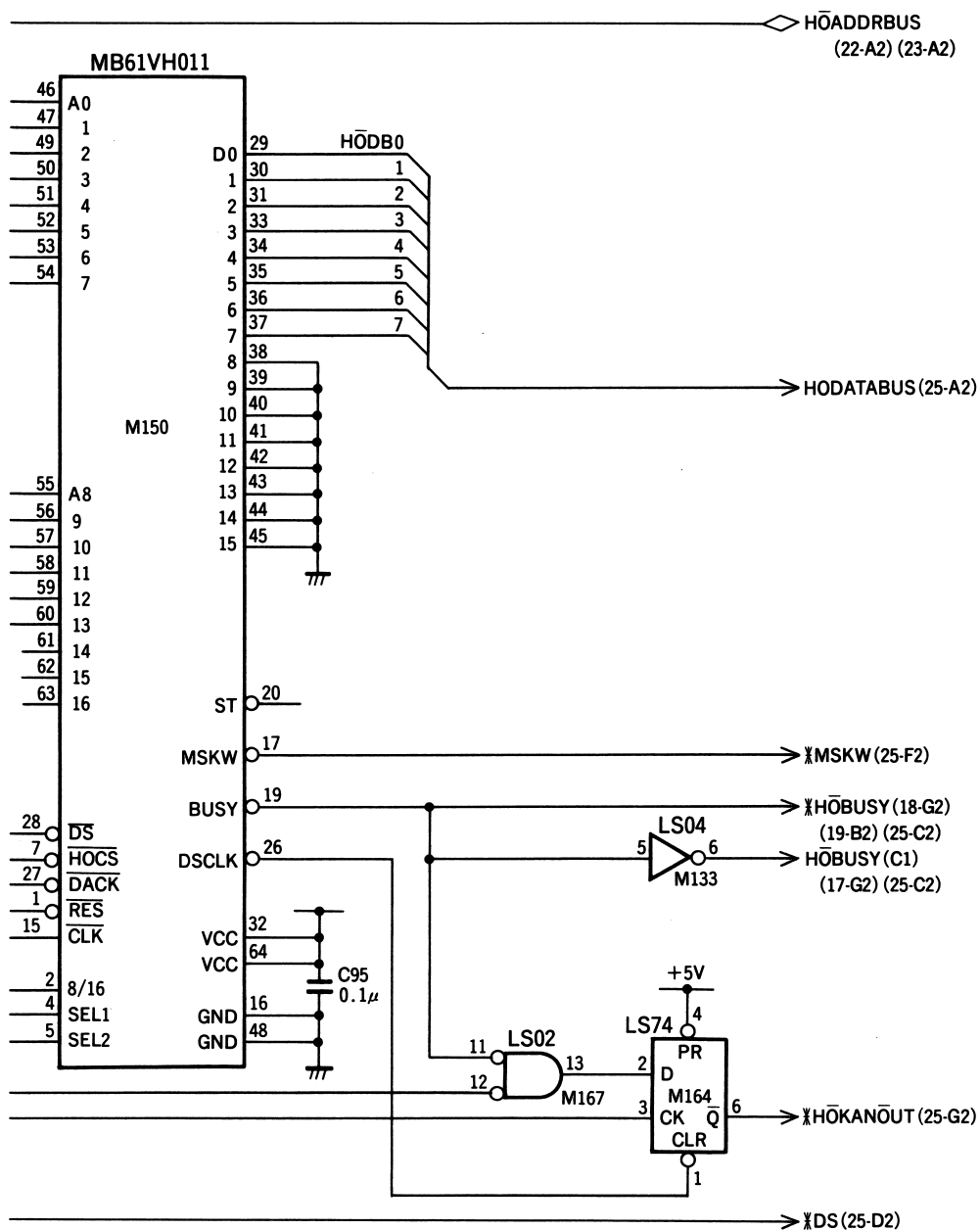
31. 論理演算



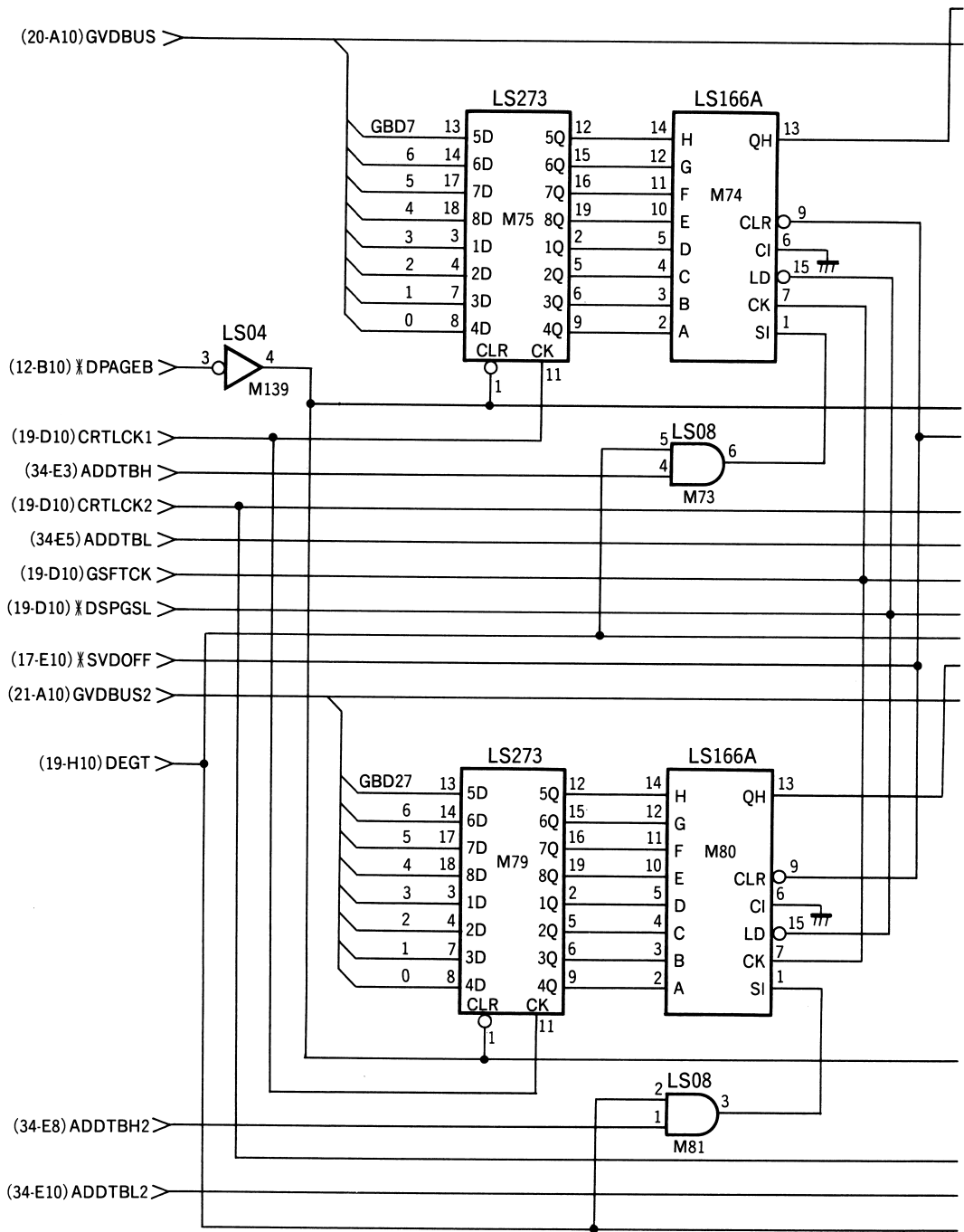


32. 直線補間

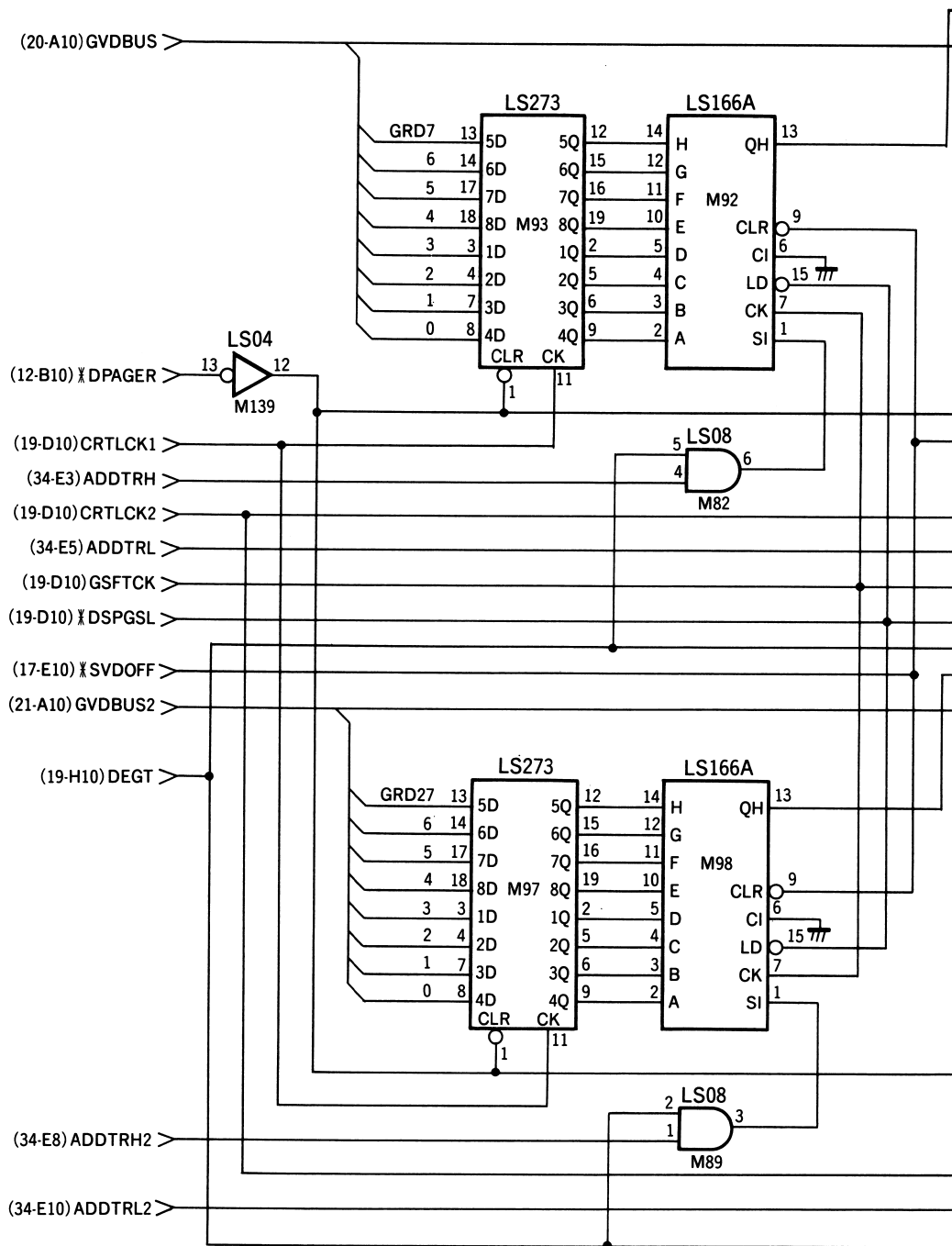


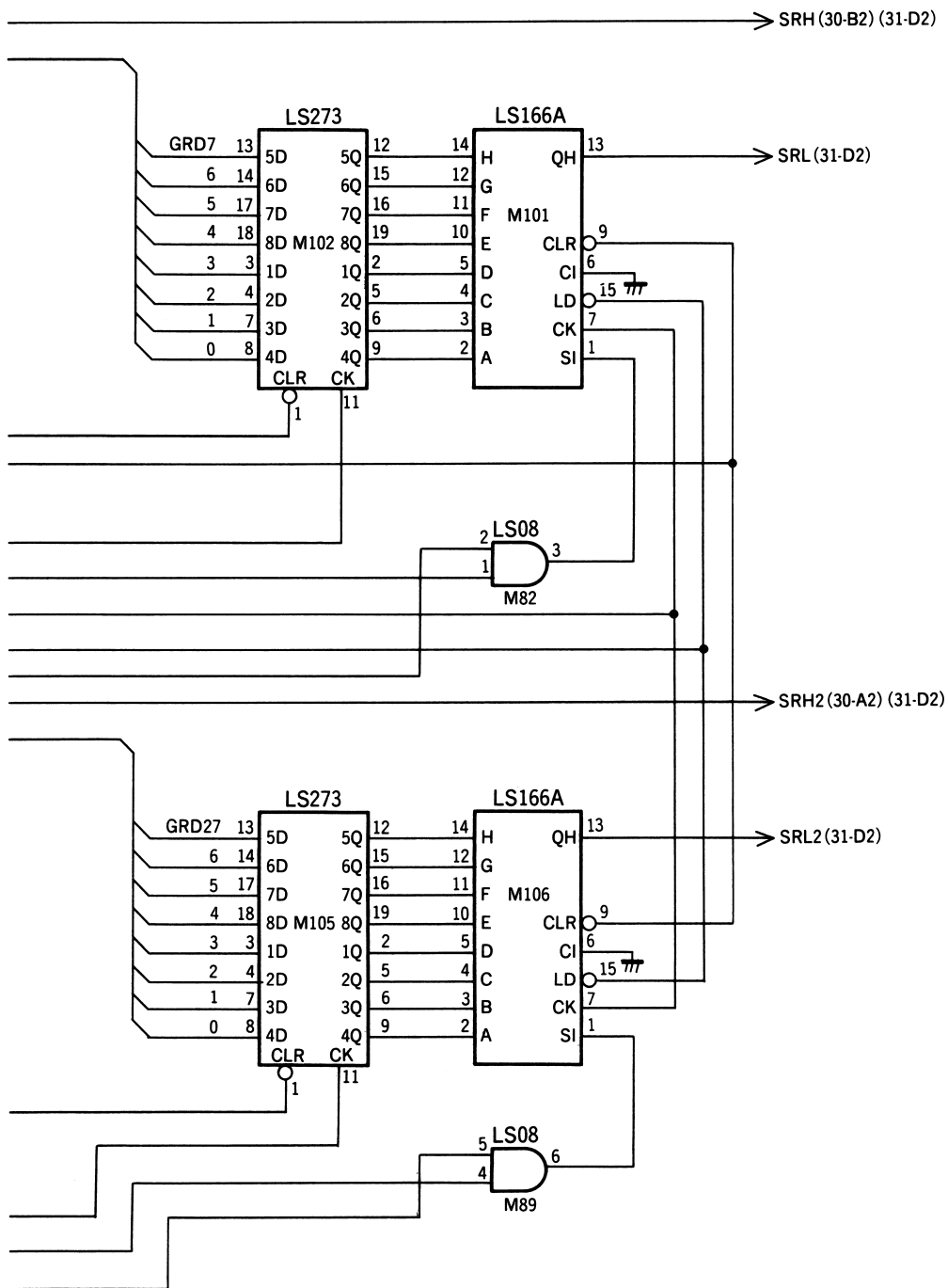


33. シフトレジスタ(青)

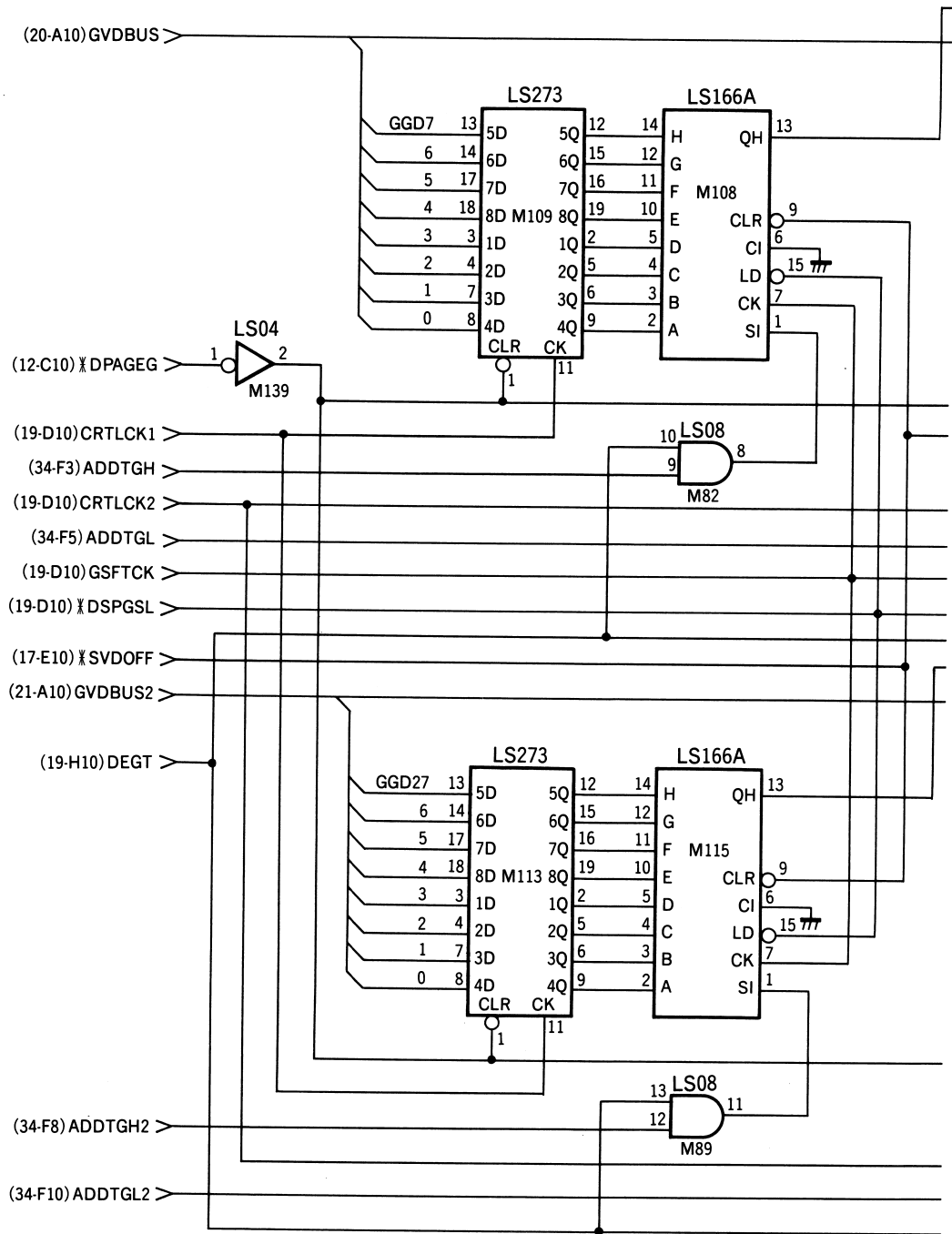


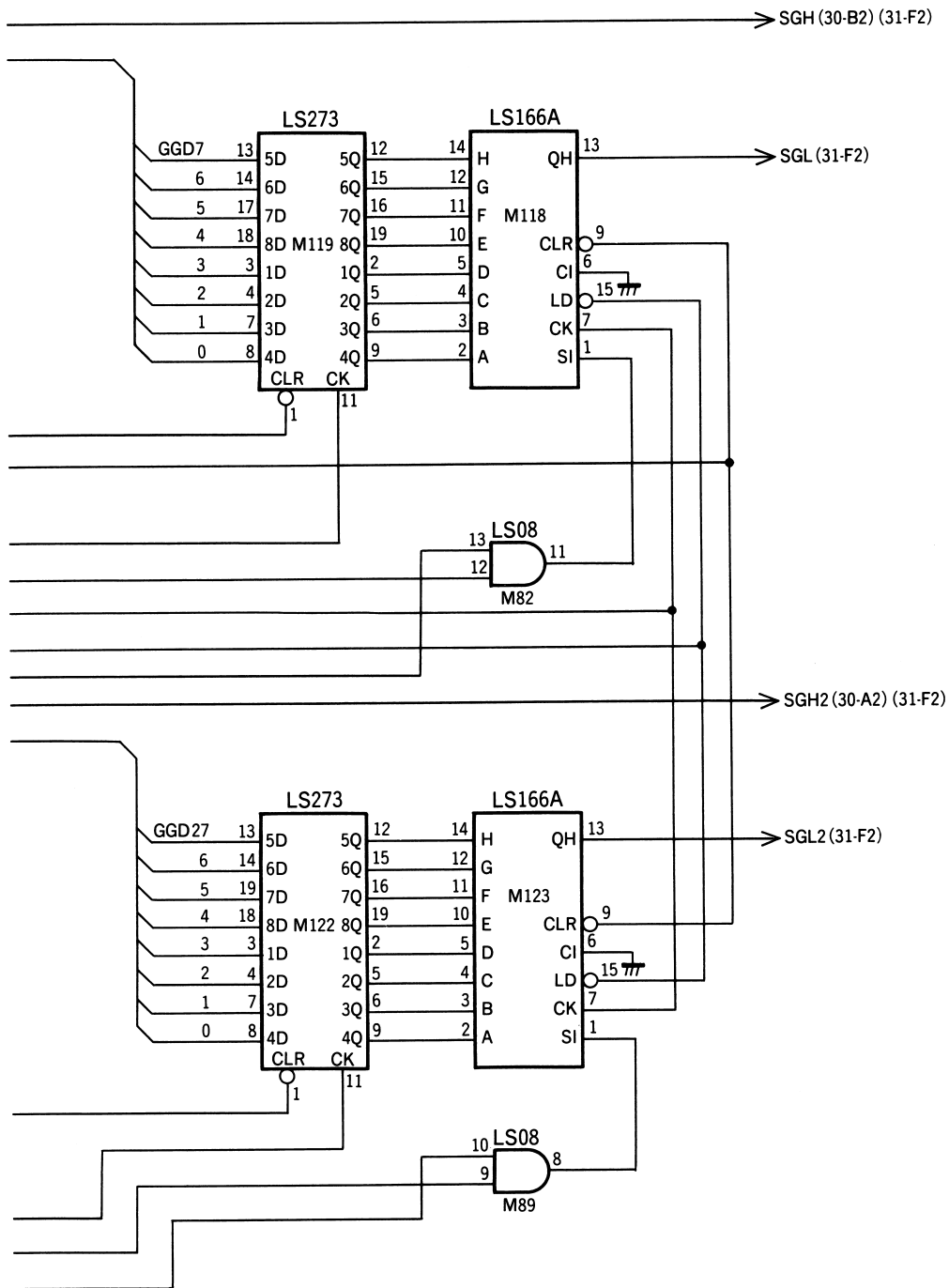
34. シフトレジスタ(赤)



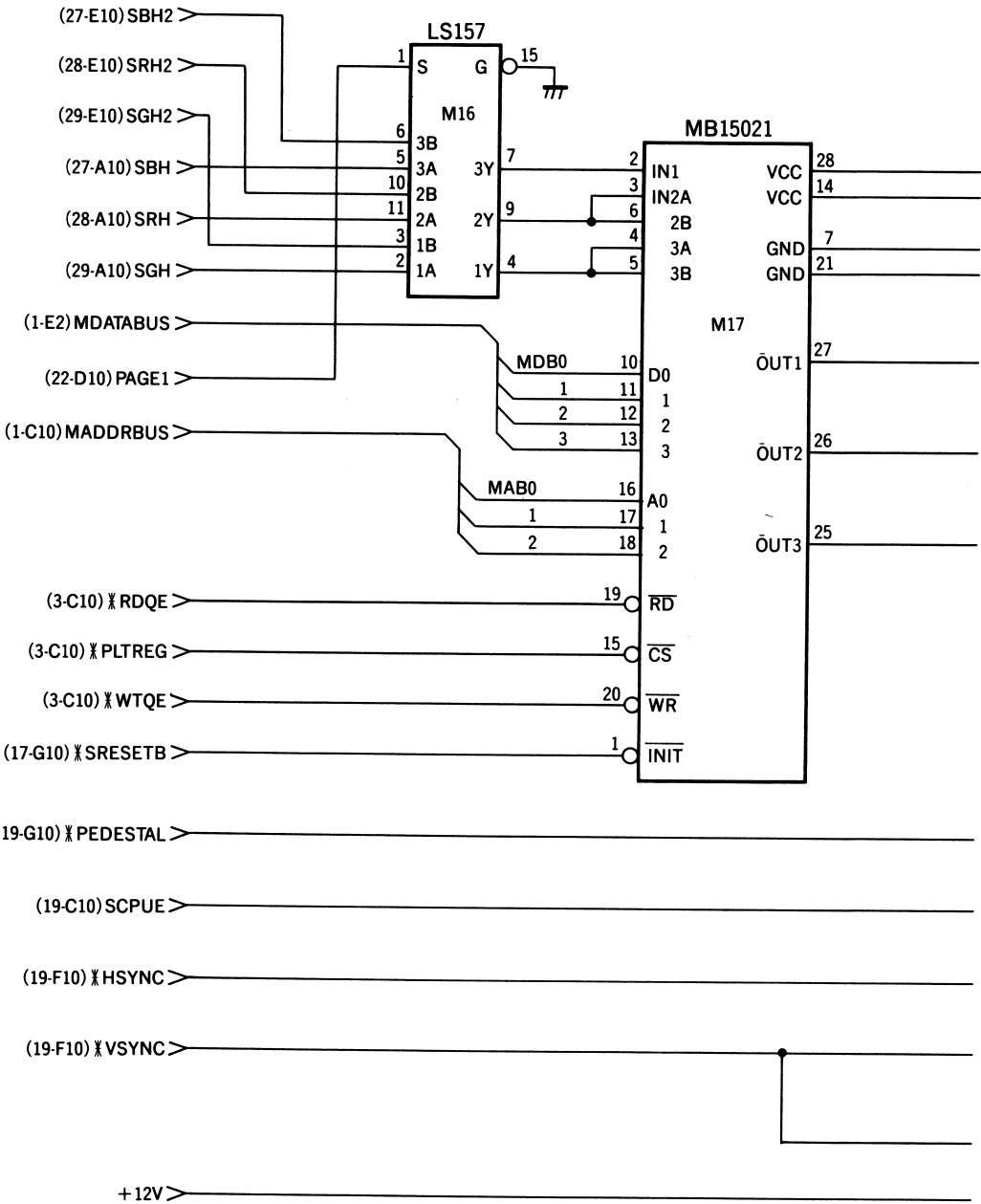


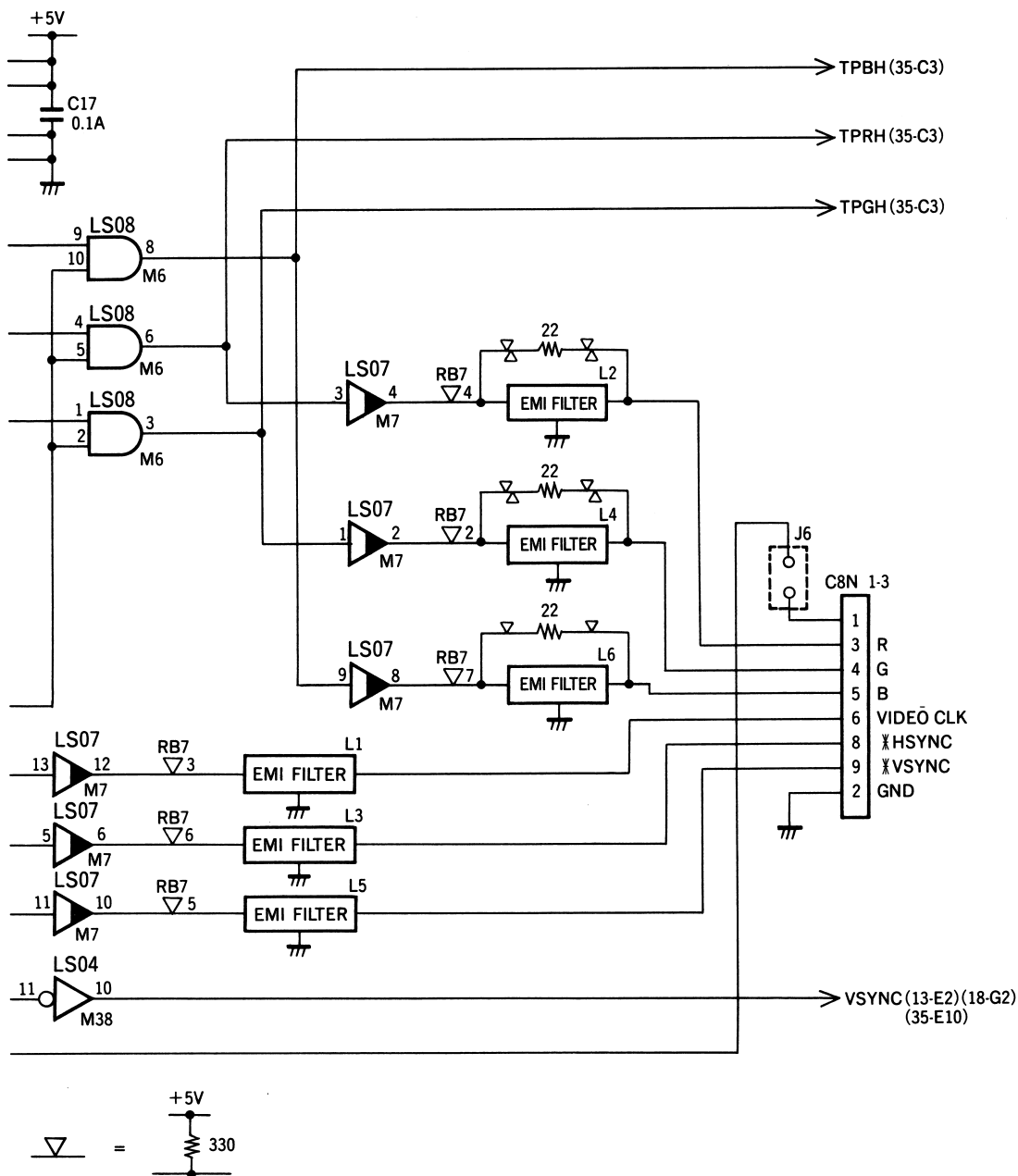
35. シフトレジスタ(緑)



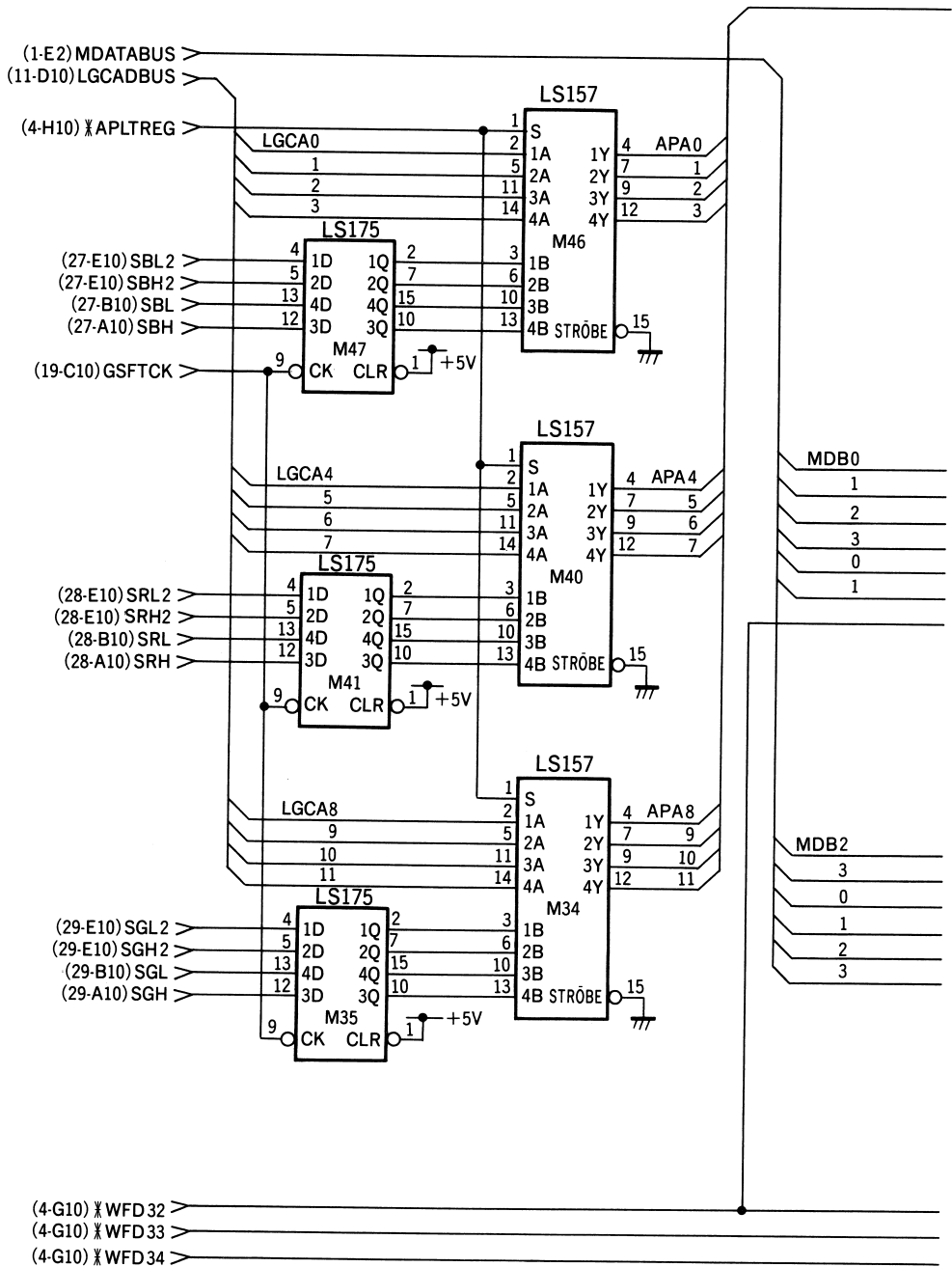


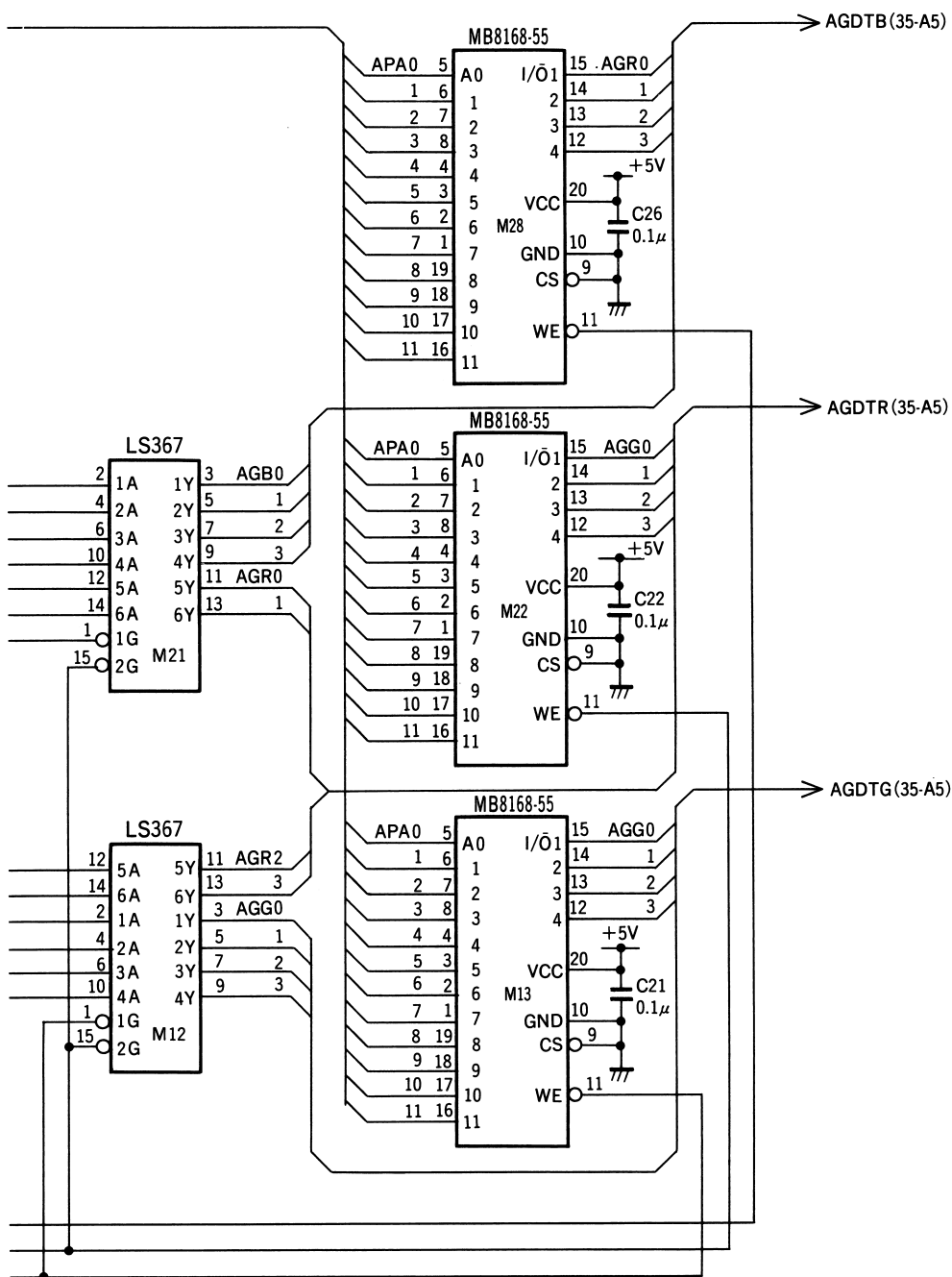
36. TTLバレット



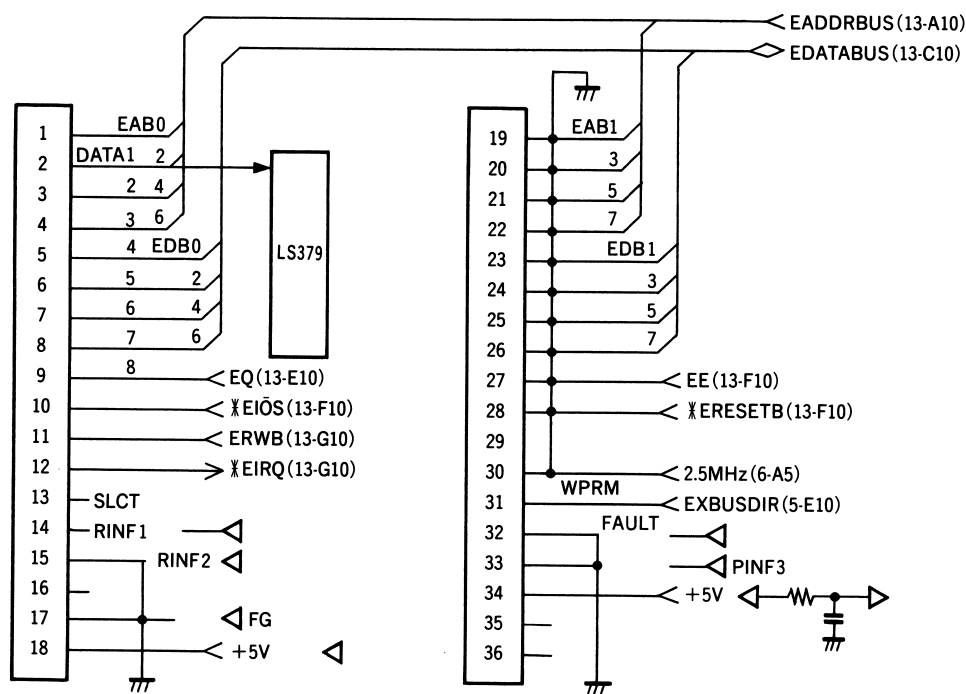


37. アナログバレット

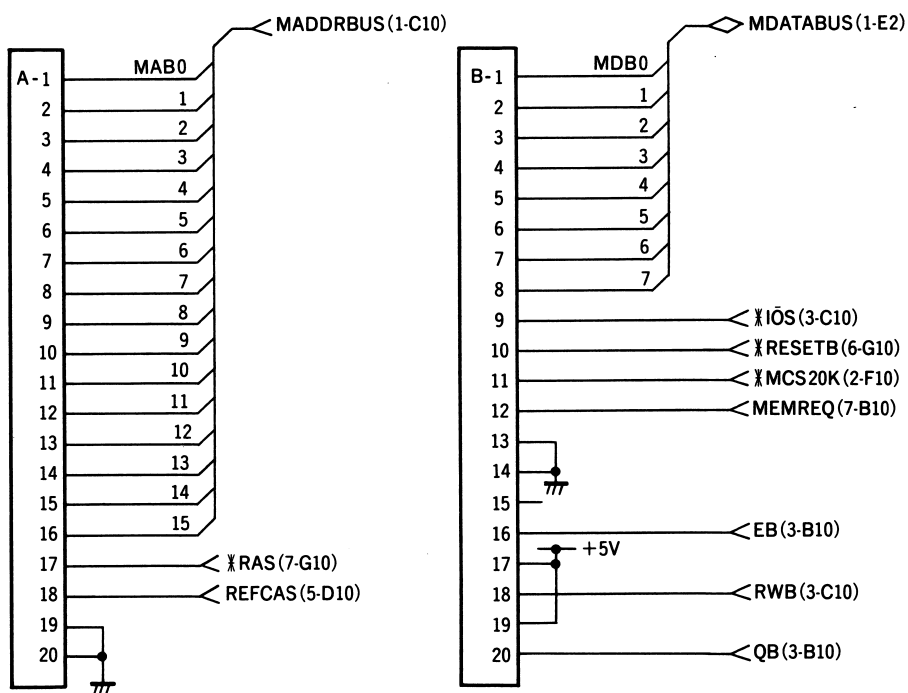




38. I/O, RAMコネクタ

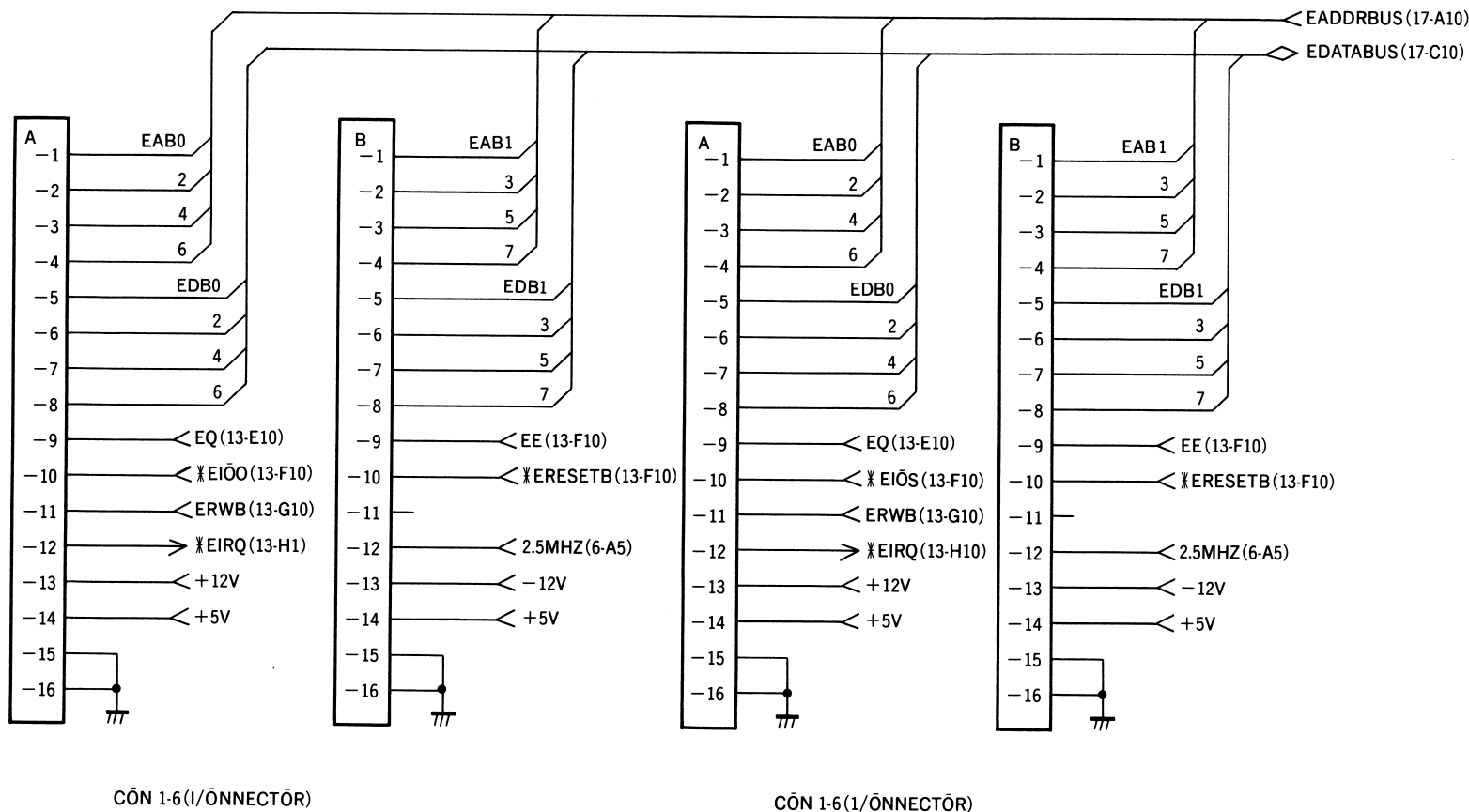


CÔN 1-2 (I/O CÔNECTÔR)

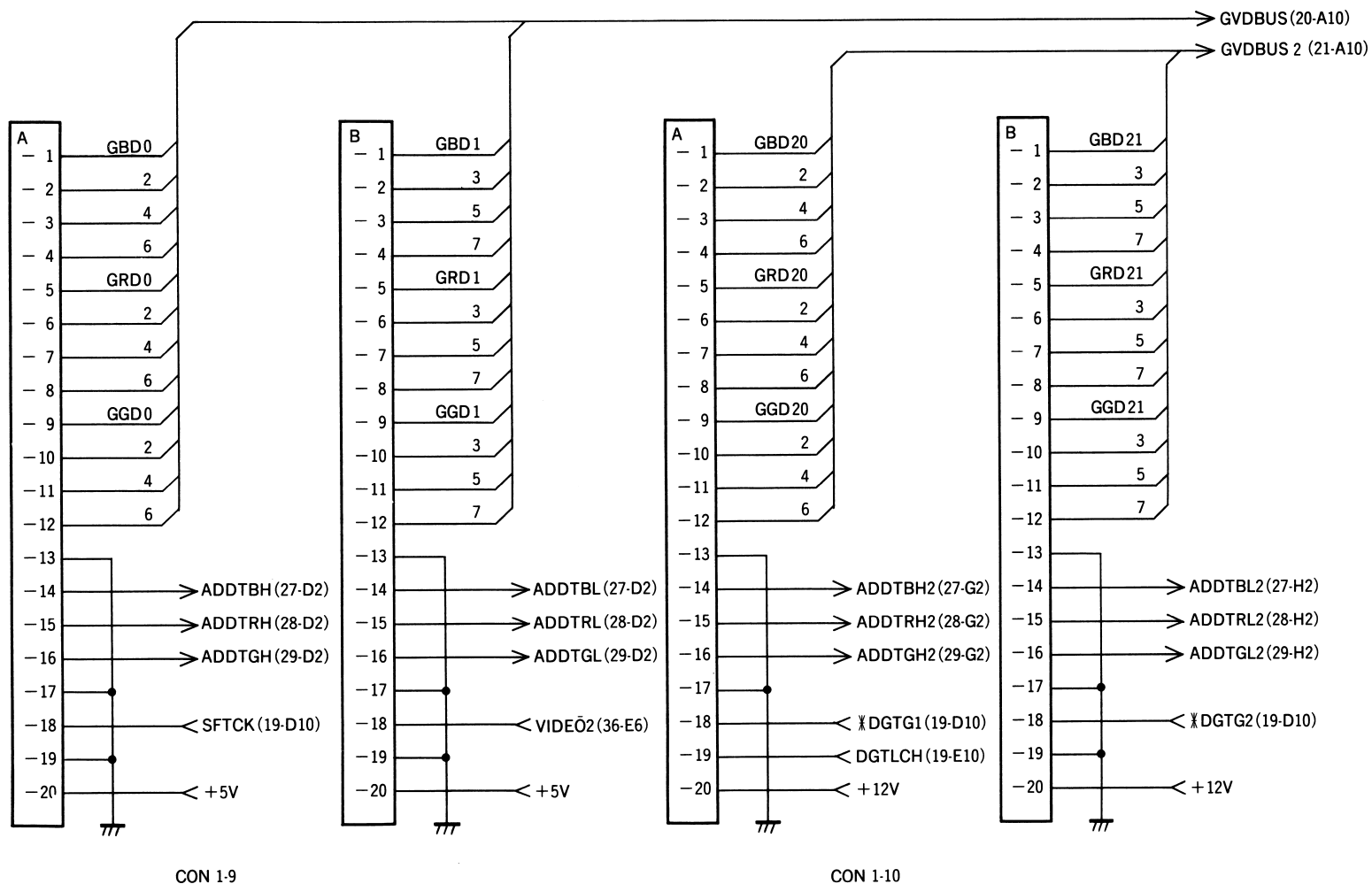


CÔN 1-8 (RAM CARD CÔNNECTÔR)

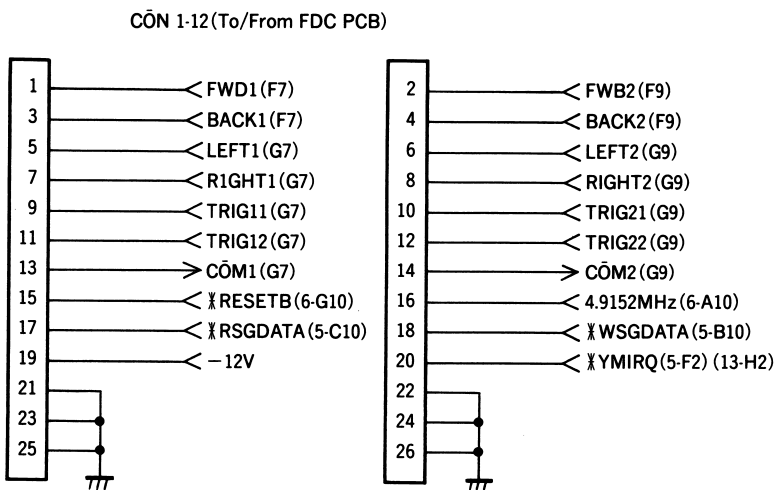
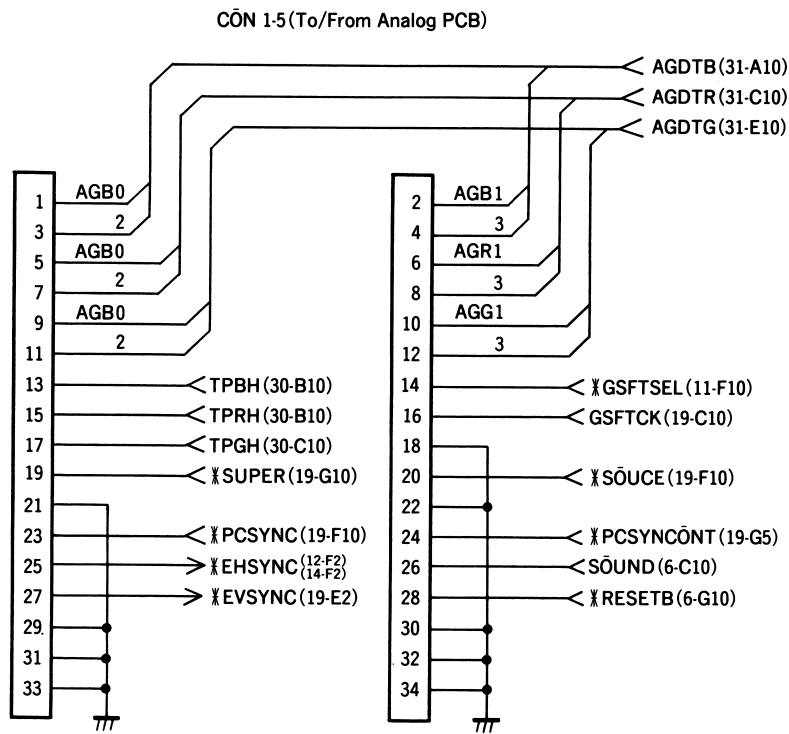
39. I/Oコネクタ



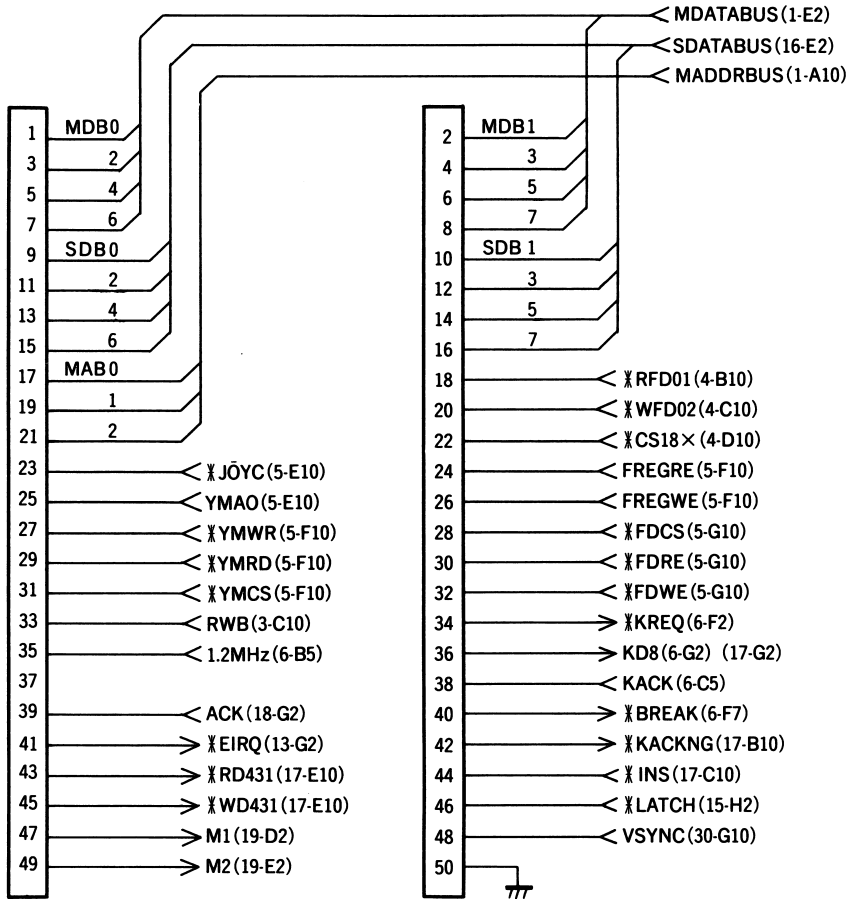
40. デジタイズコネクタ



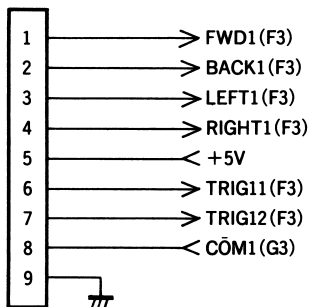
41. コネクタ



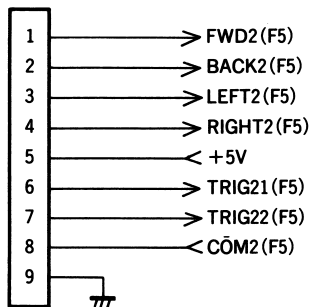
CÔN 1-13 (To/From FDC PCB)



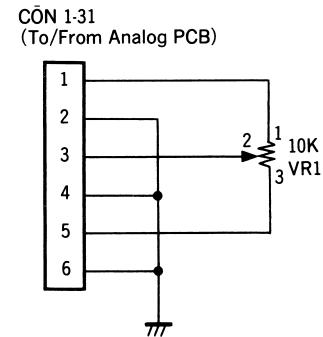
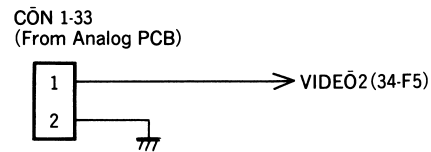
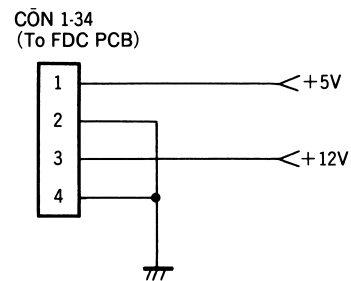
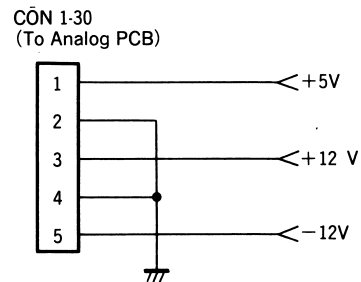
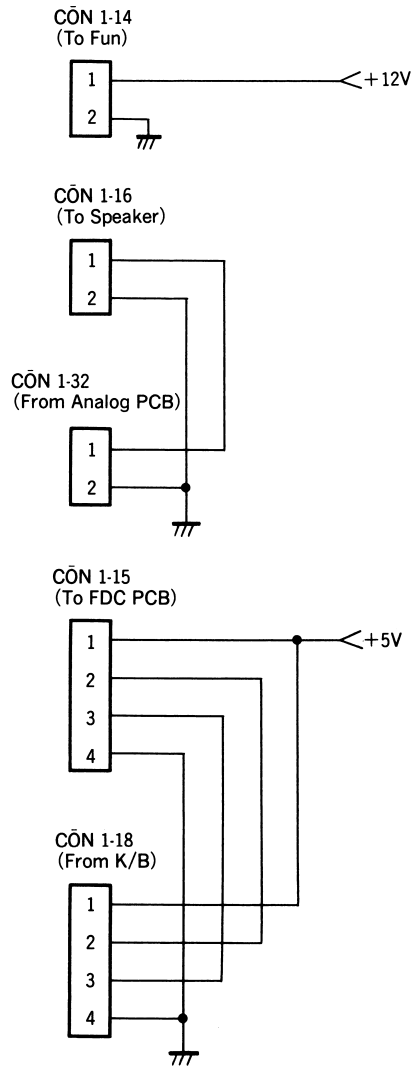
CÔN 1-23 (Joystick-1)



CÔN 1-22 (Joystick-2)

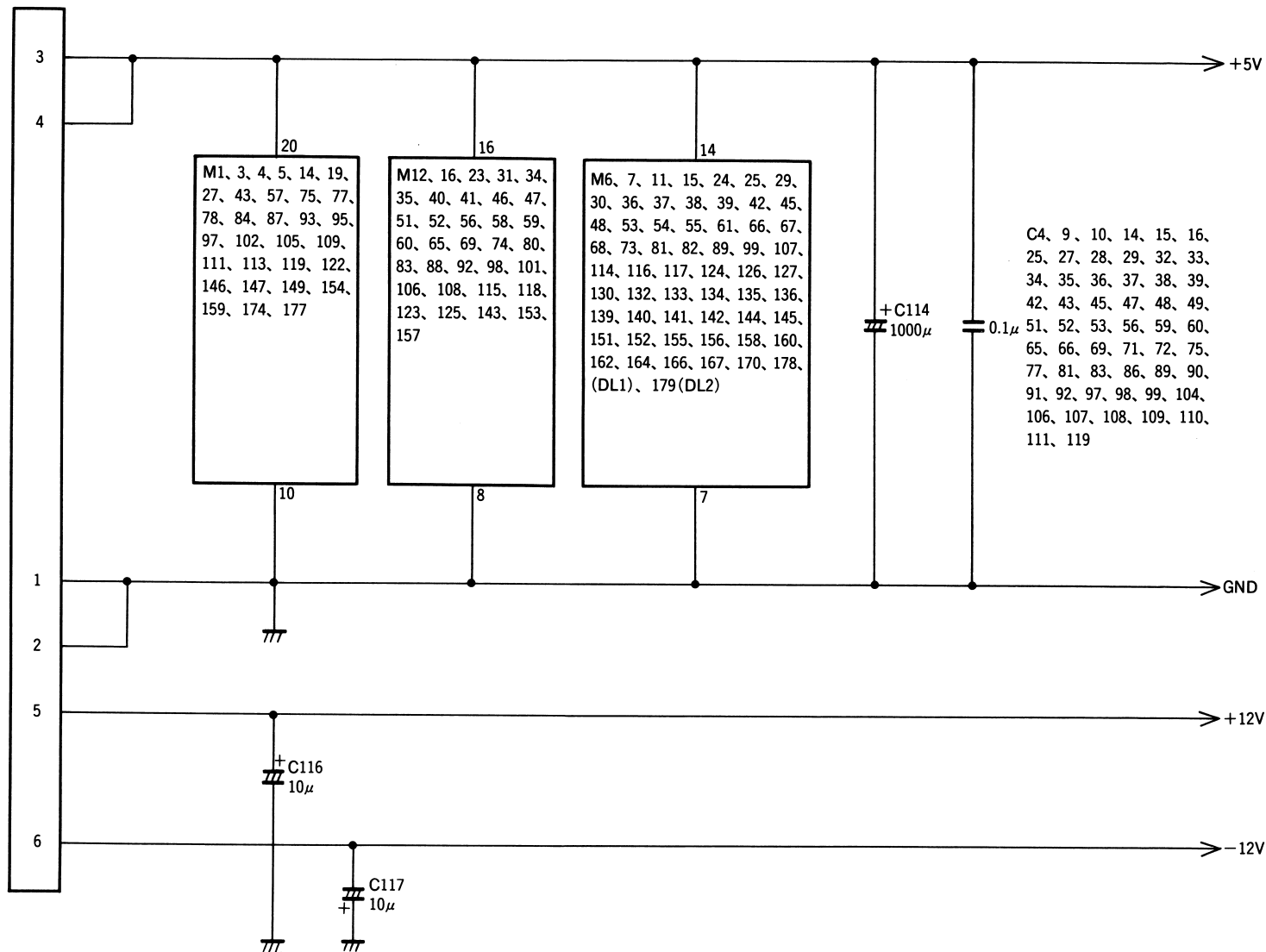


42. コネクタ

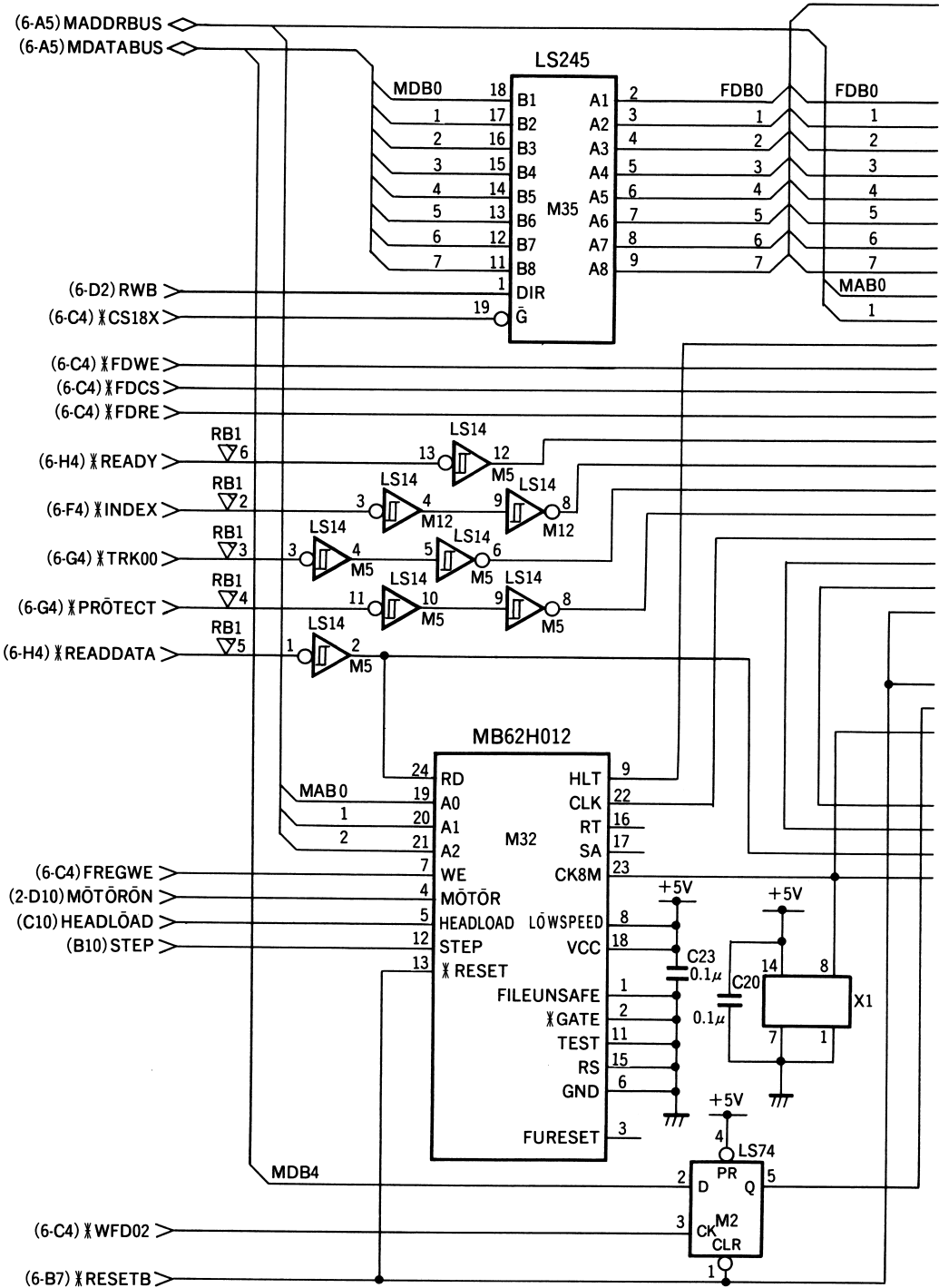


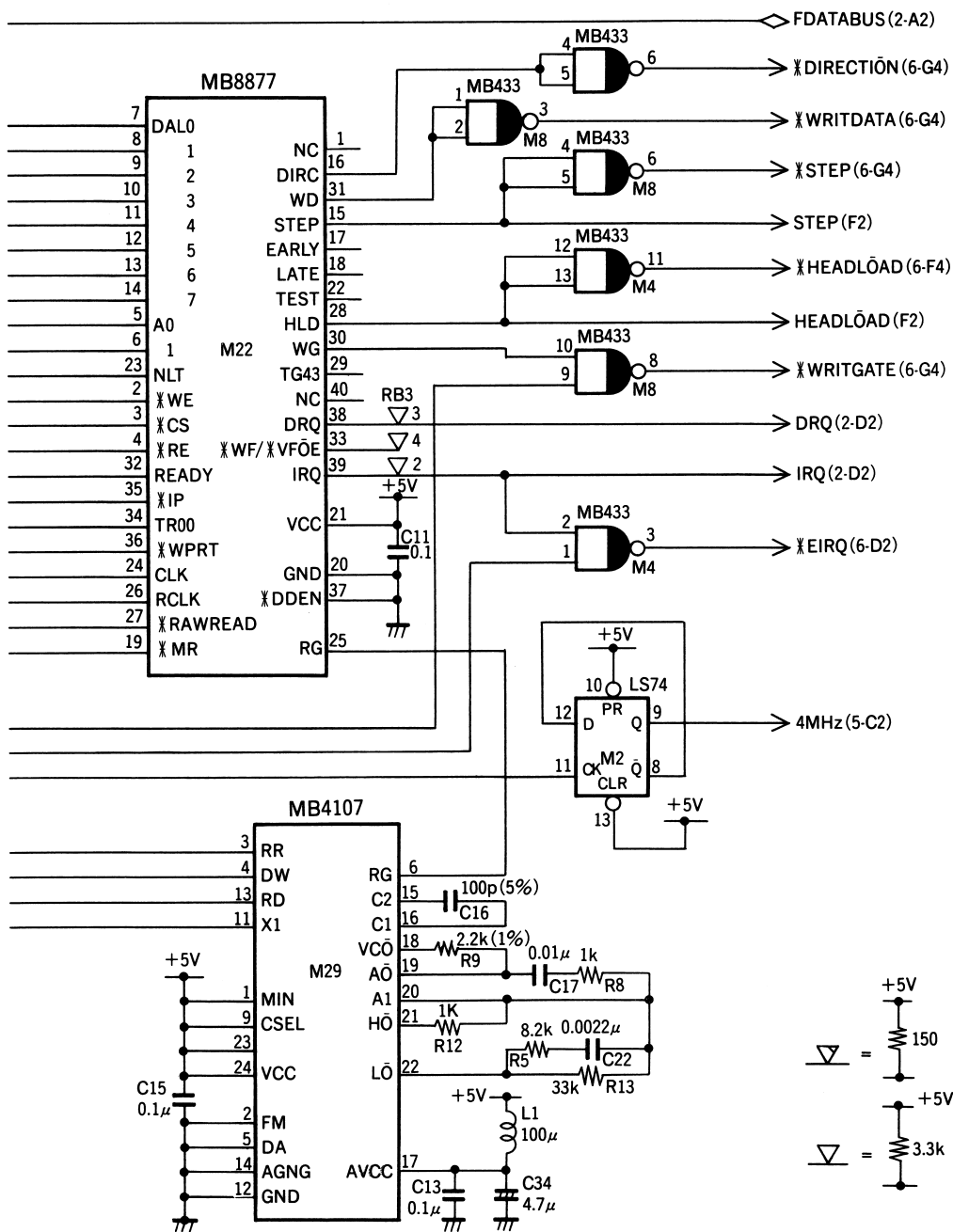
43. P/Sコネクタ

CÔN 1-17
(From P/S)

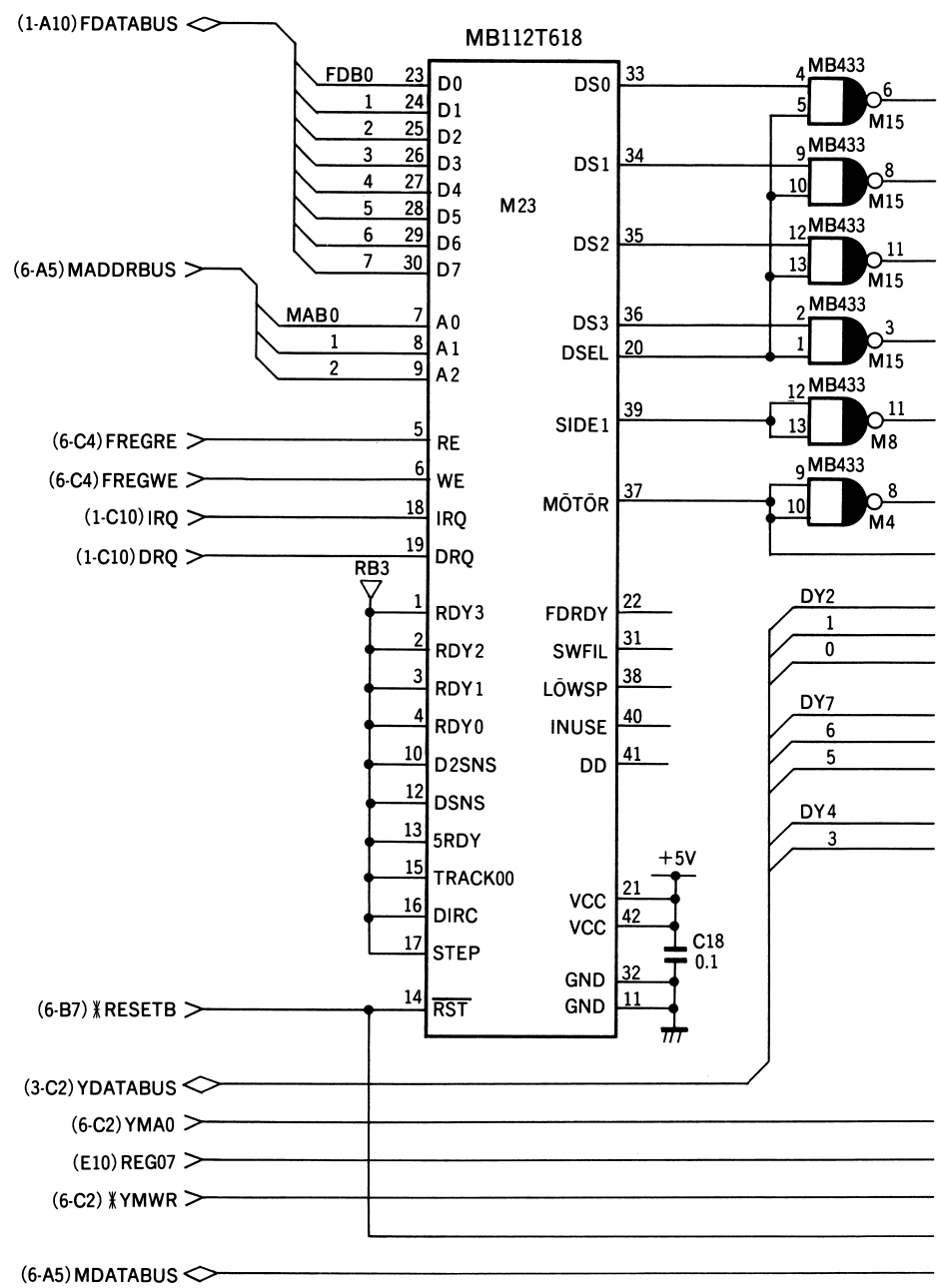


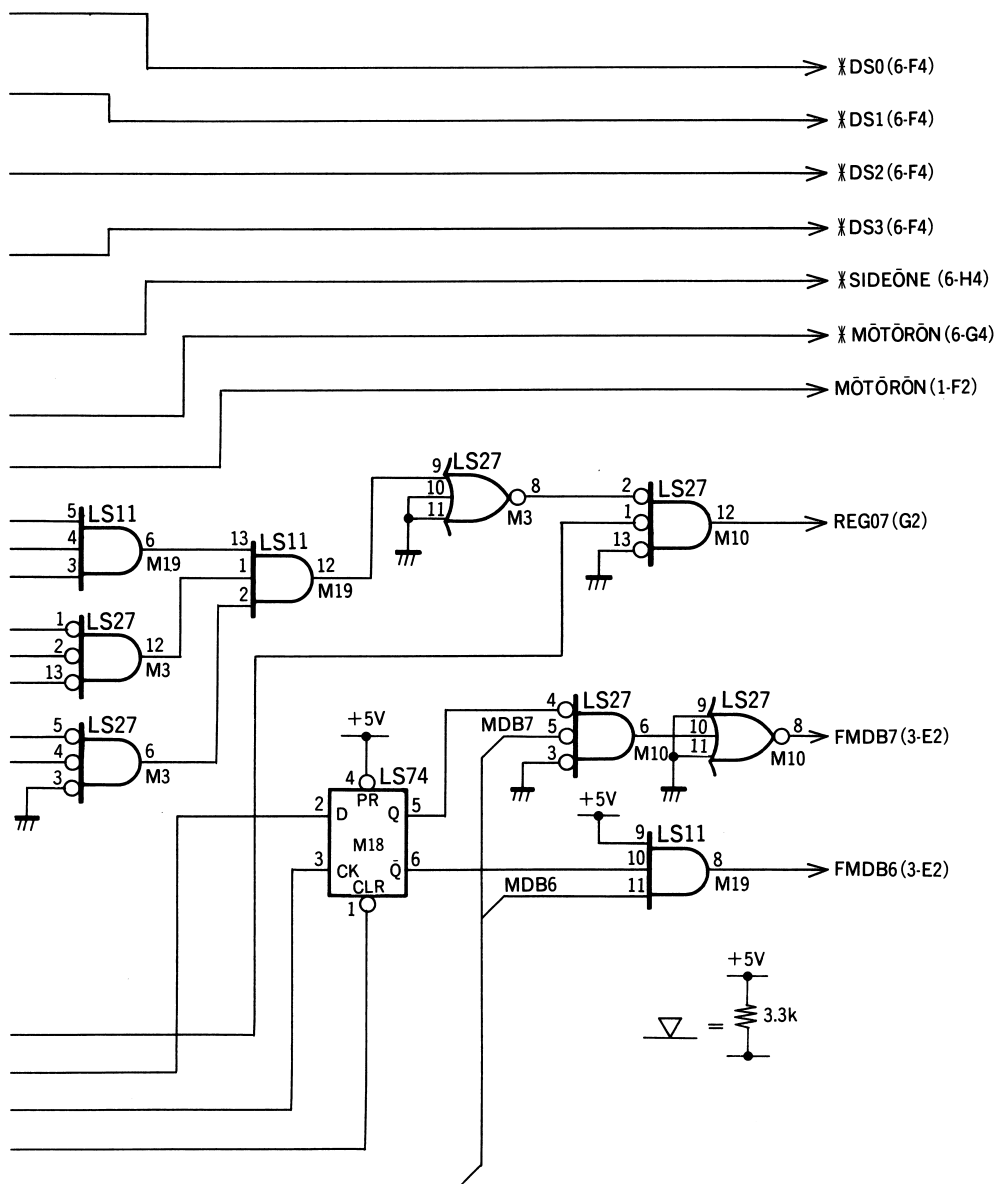
44. FDC VFO etc.



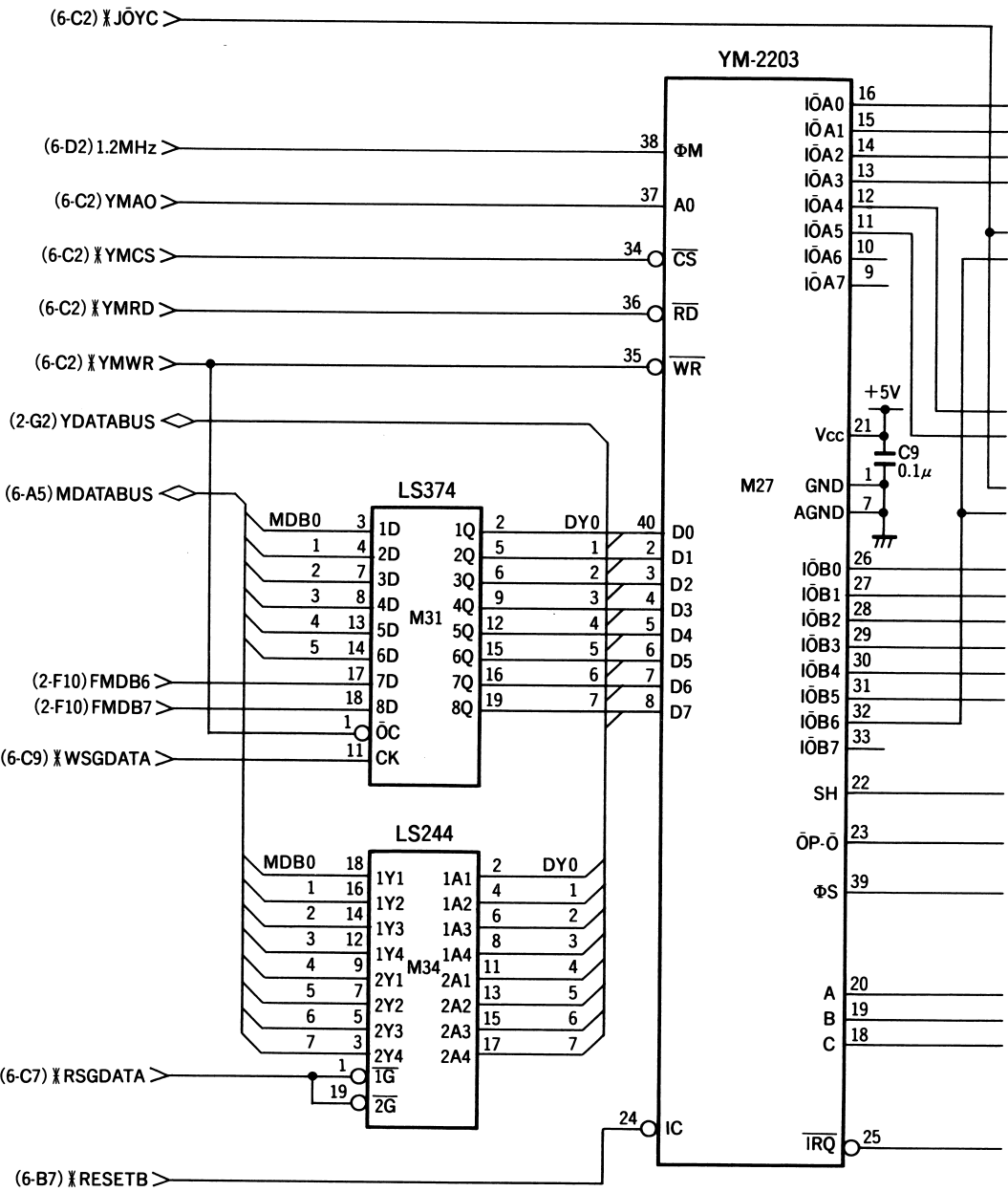


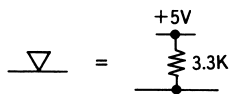
45. FDコントロール

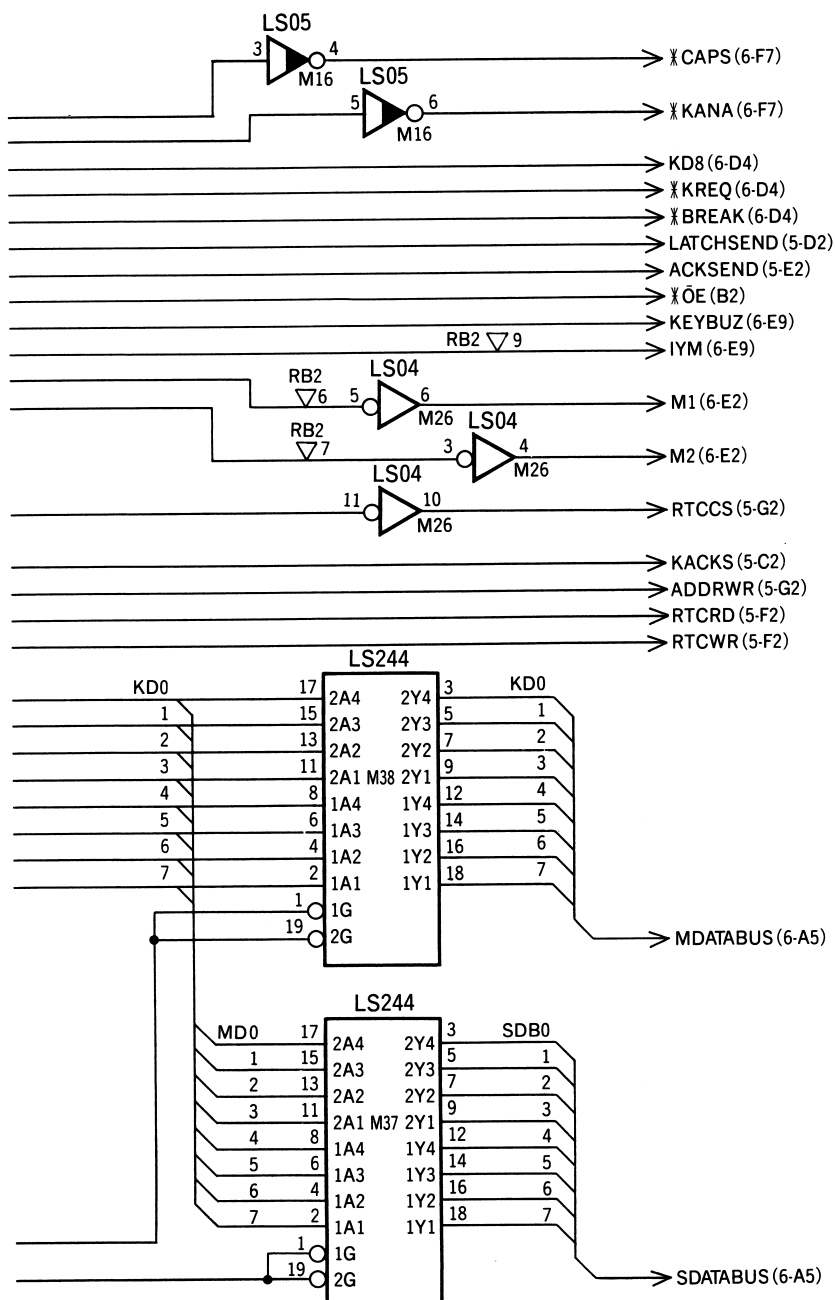




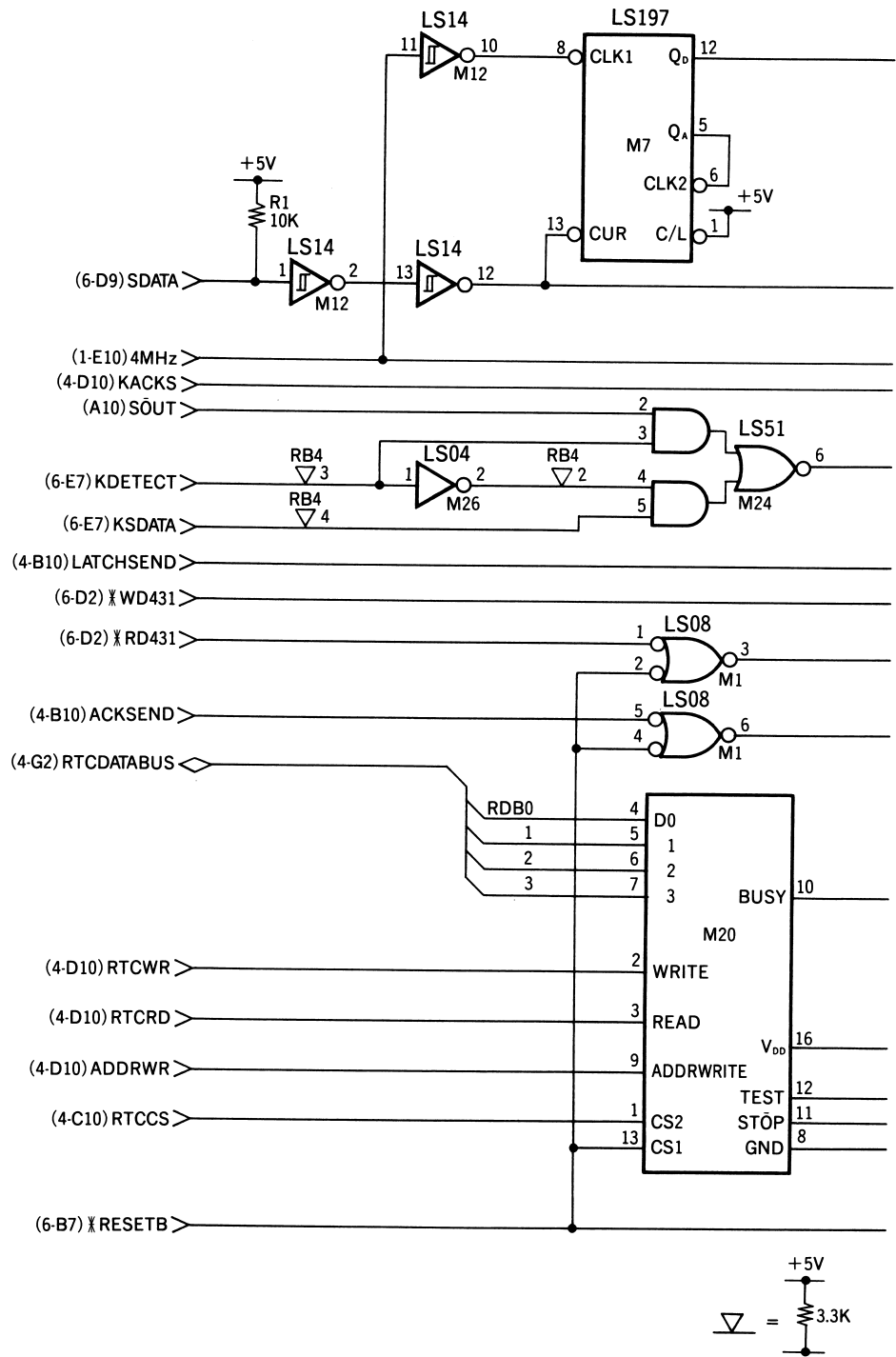
46. FM音源

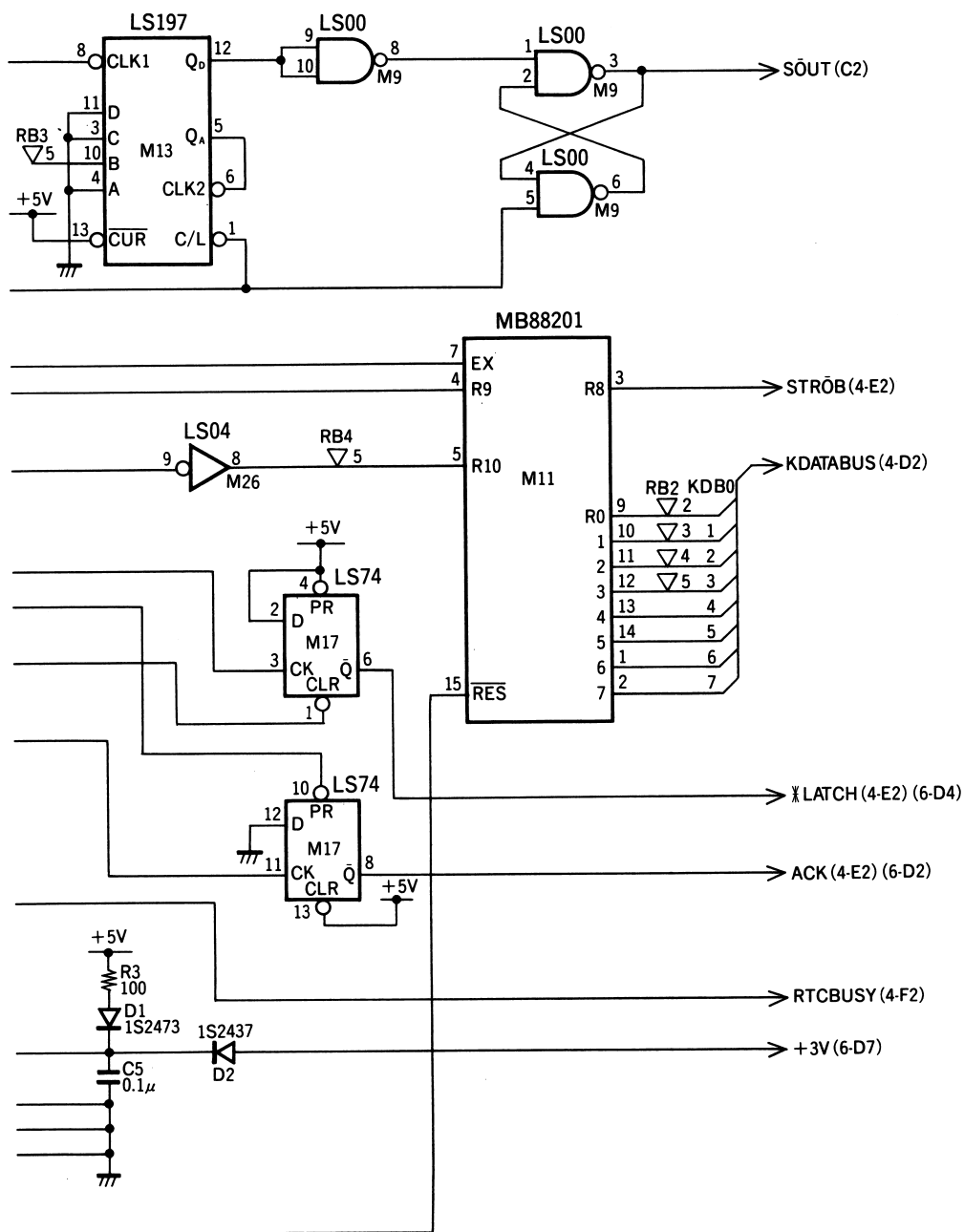




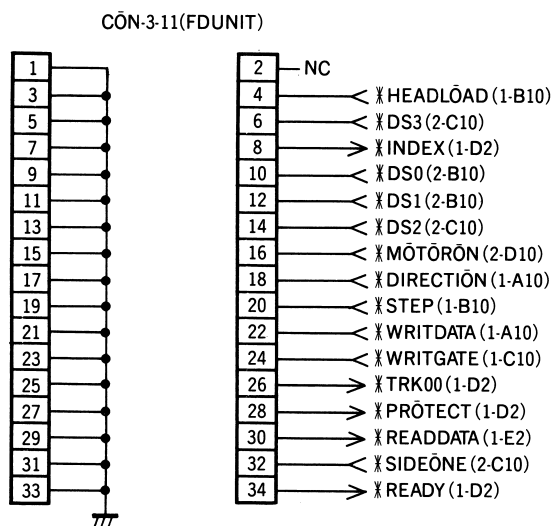
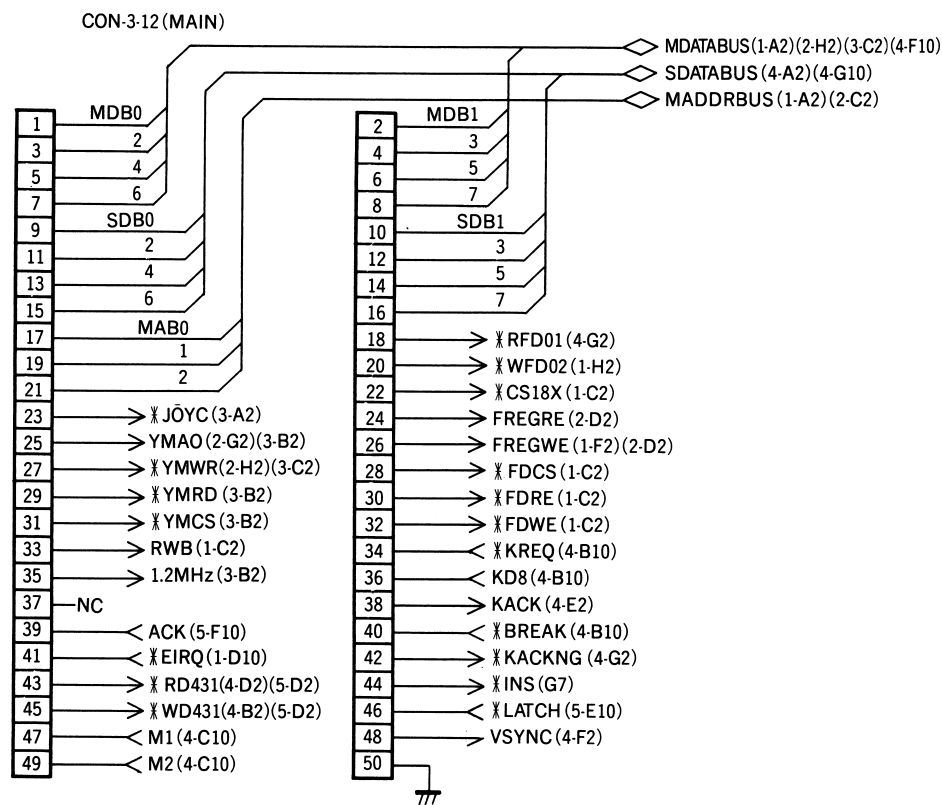


48. S→Pコネクタ,RTC

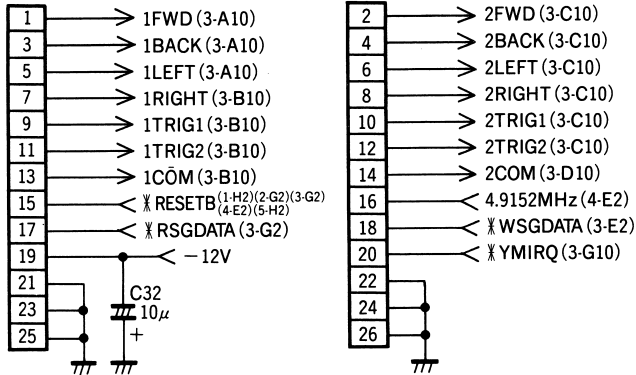




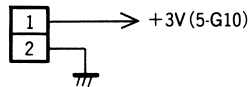
49. コネクタ



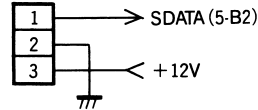
CÔN-3-13(JÖYSTICK)



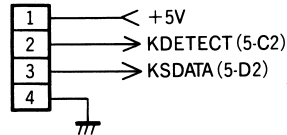
CÔN-3-27(BATTERY)



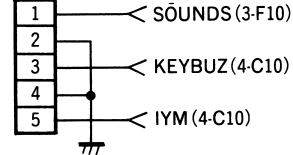
CÔN-3-29(AMP)



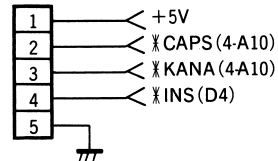
CÔN-3-15(K/B)



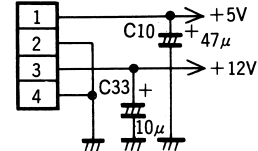
CÔN-3-26(ANALÖG)



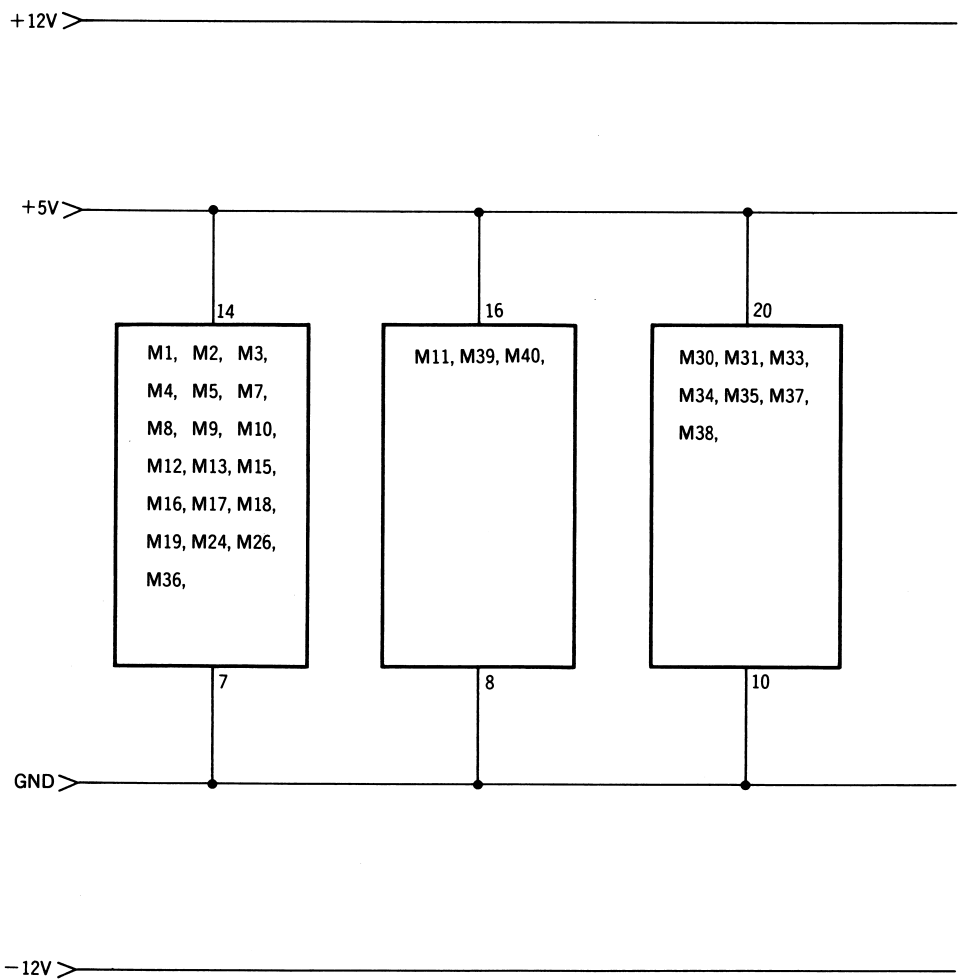
CÔN-3-28(LED)

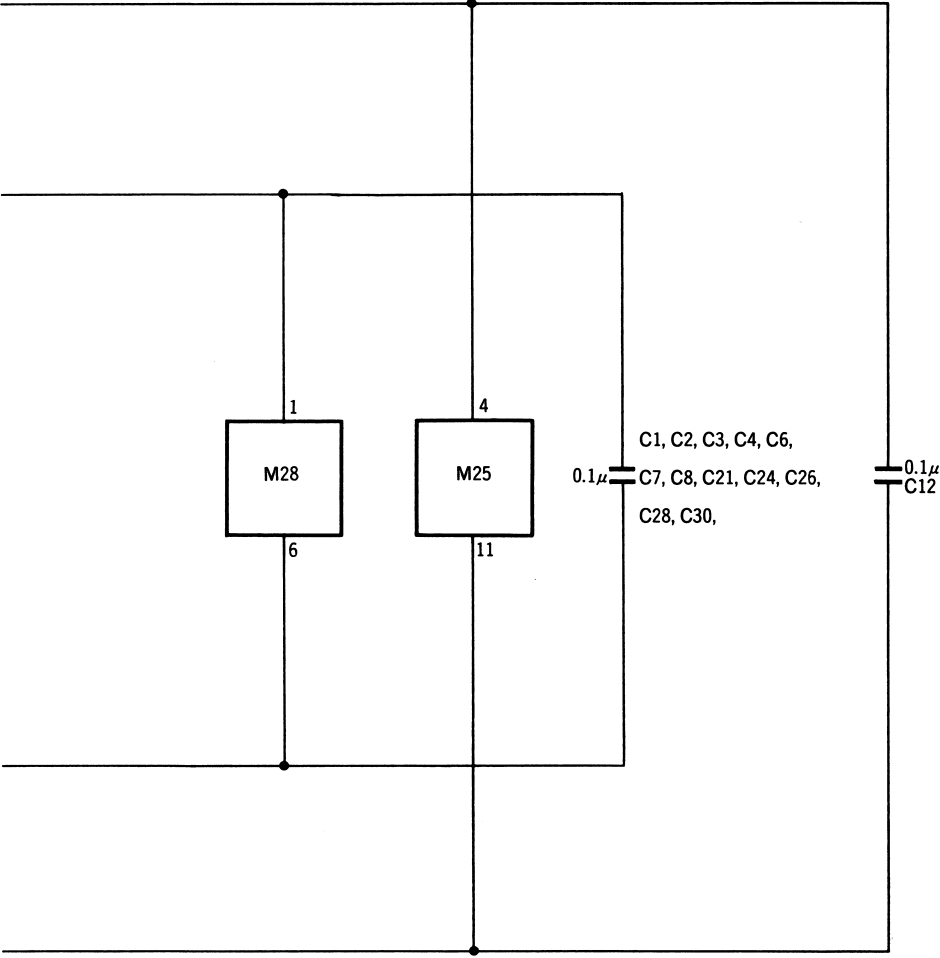


CÔN-3-34(P/S)

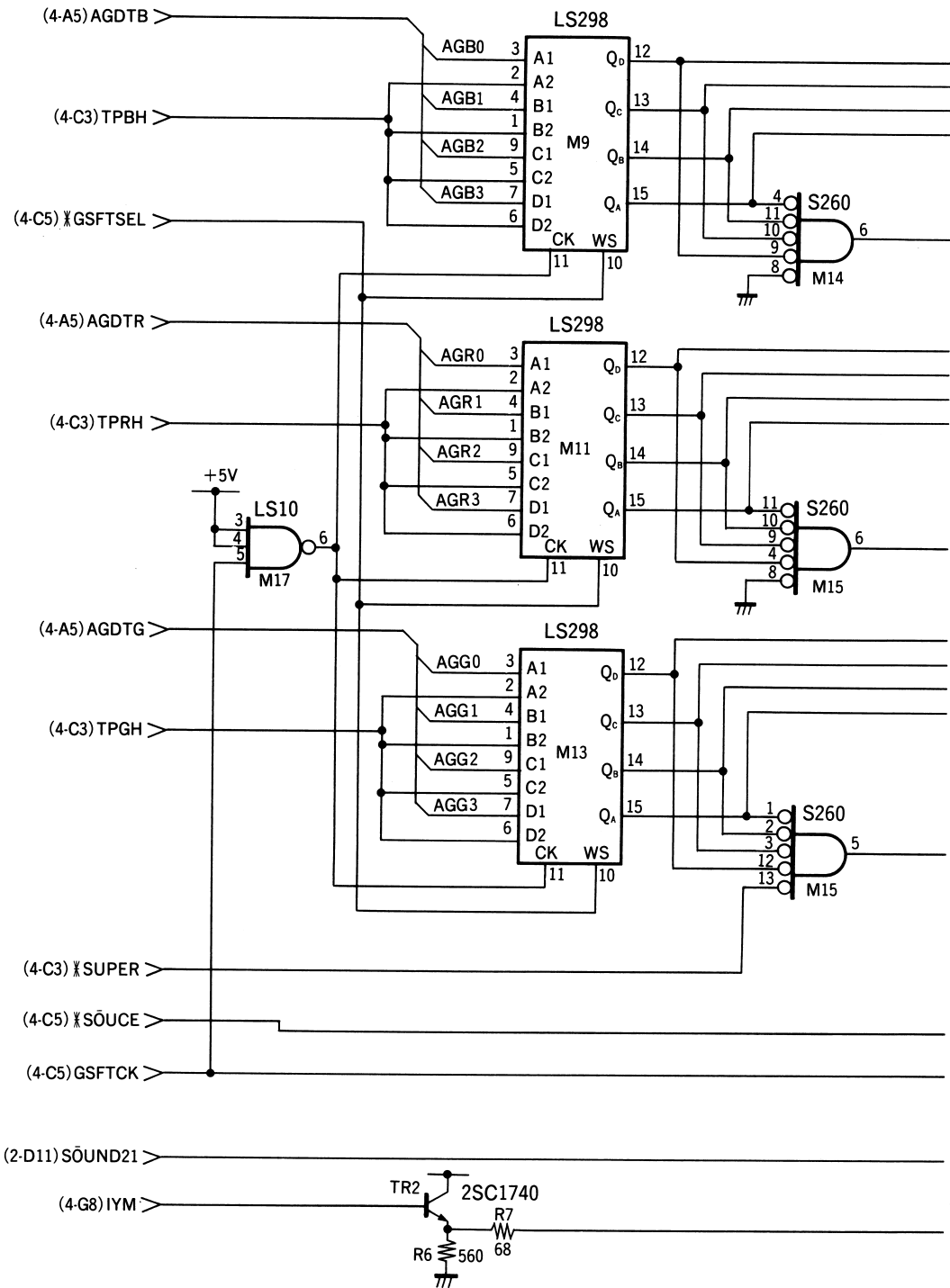


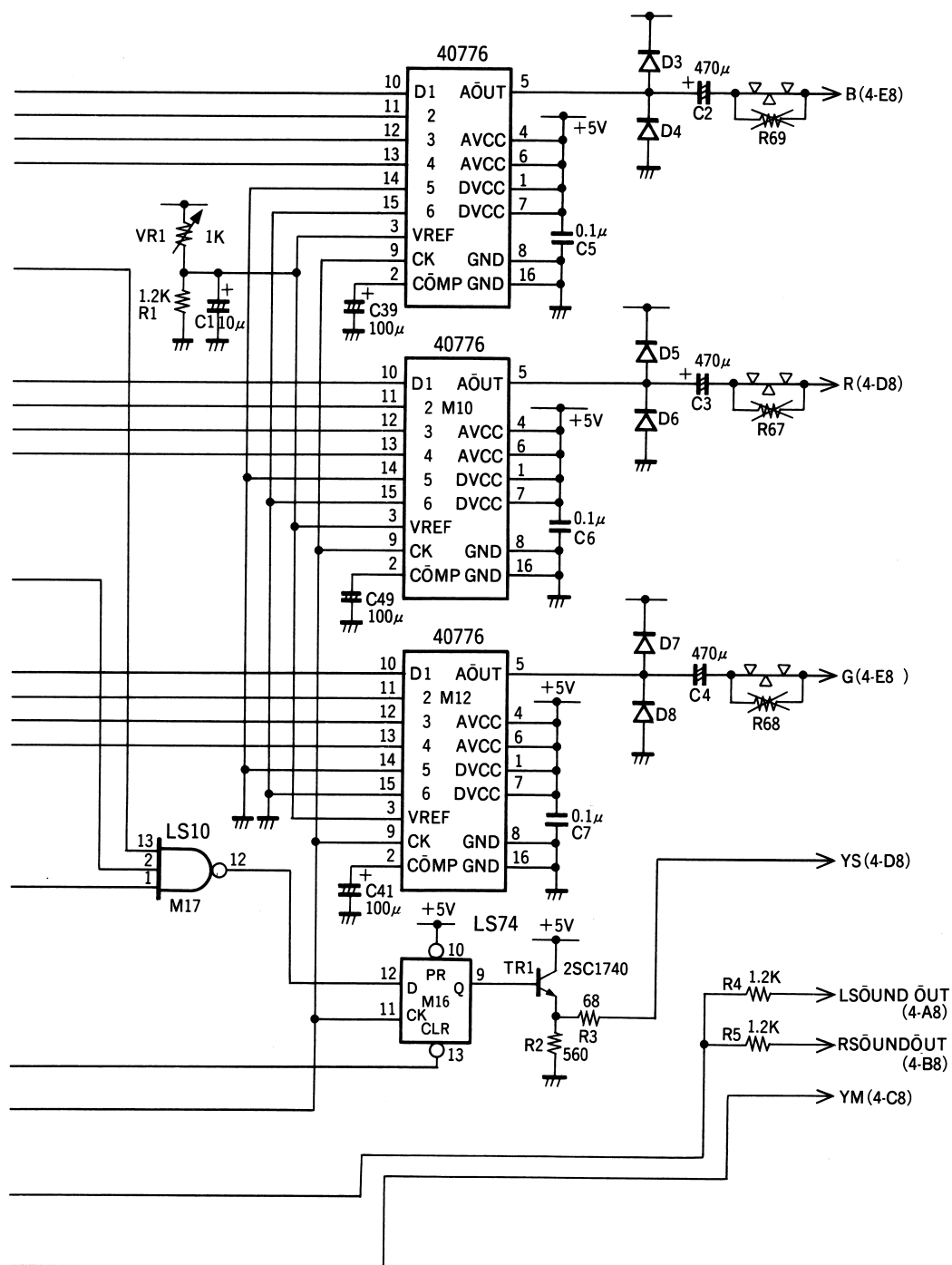
50. P/S of IC



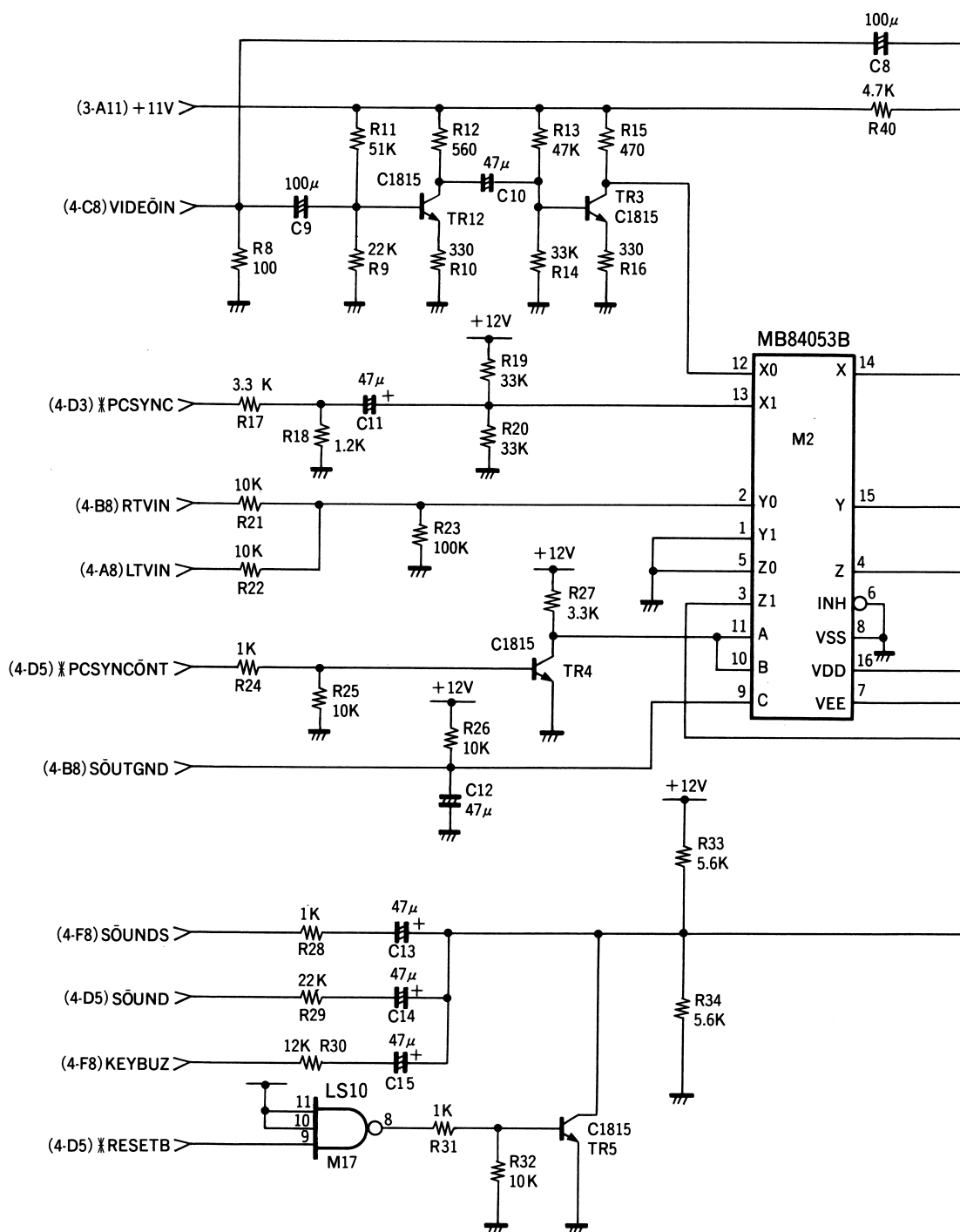


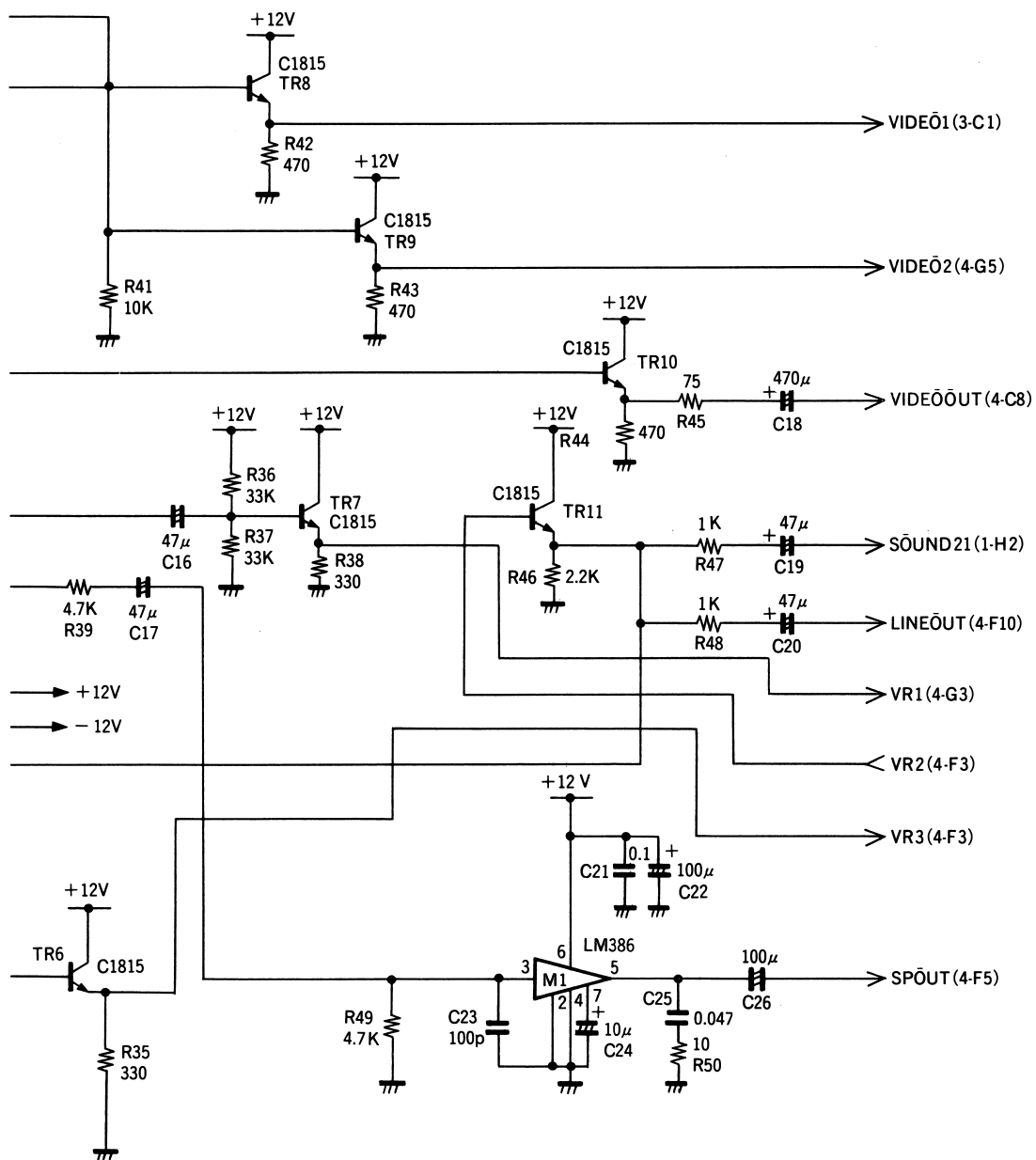
51. D/Aコンバータ



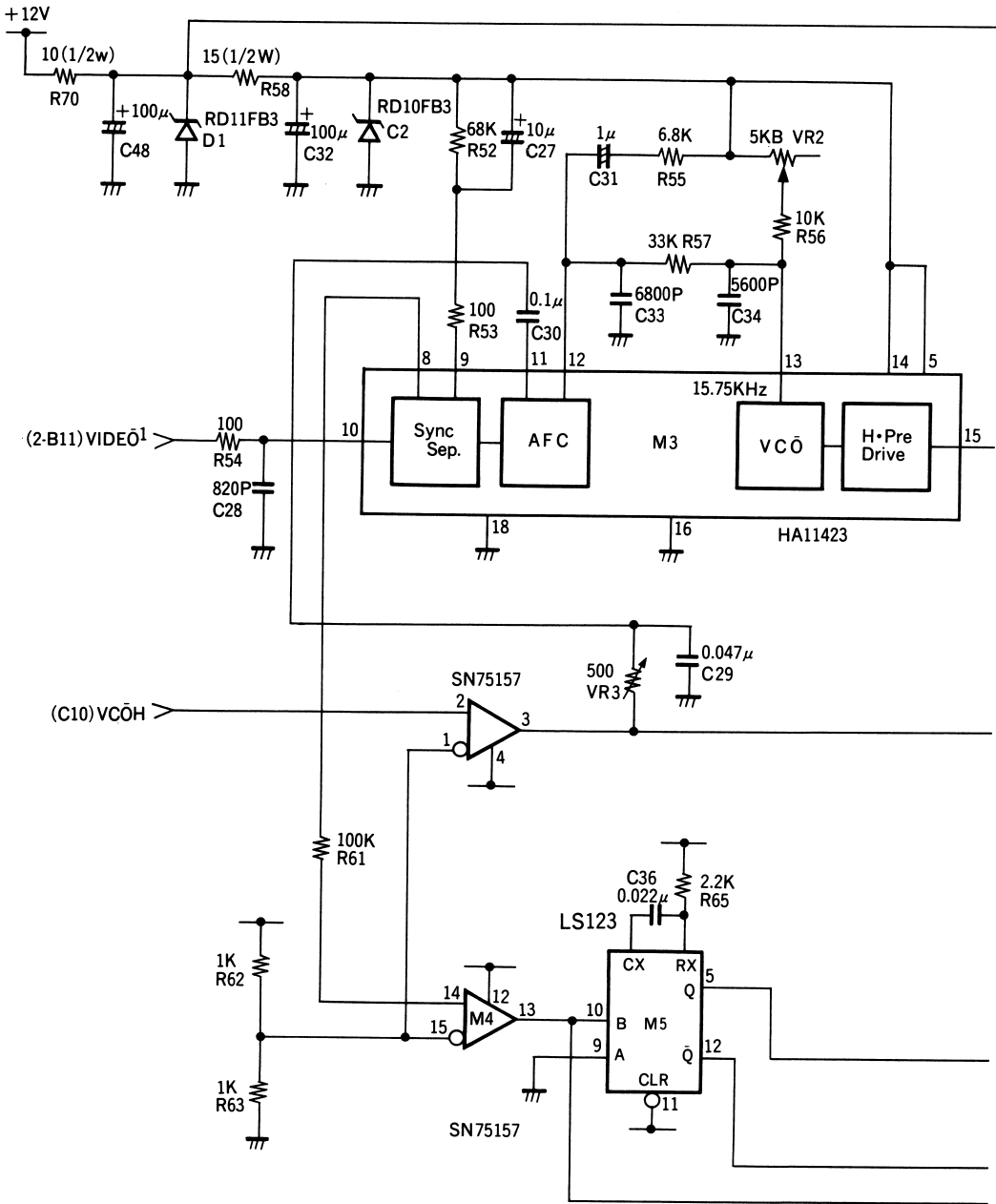


52. アナログスイッチ etc.

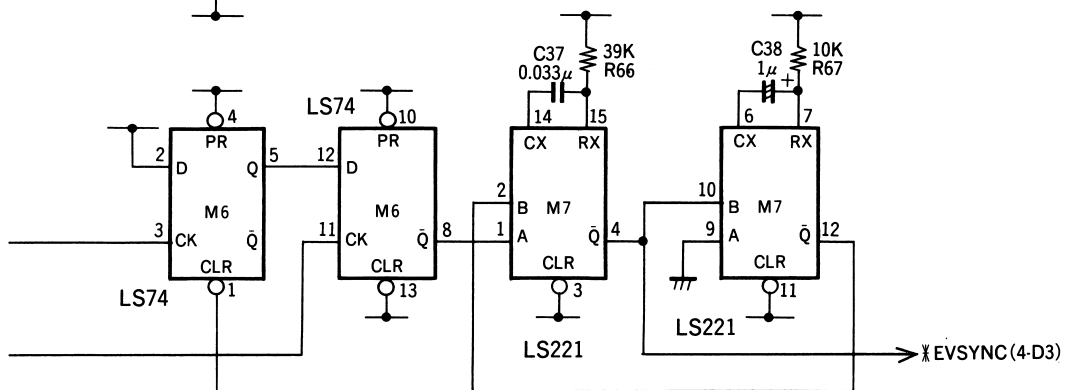
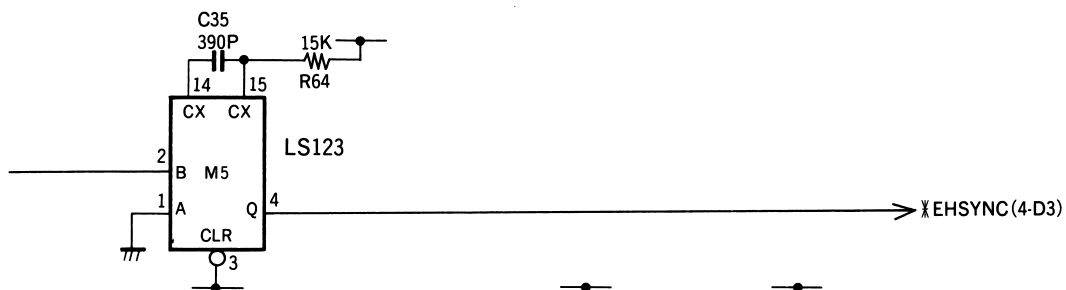
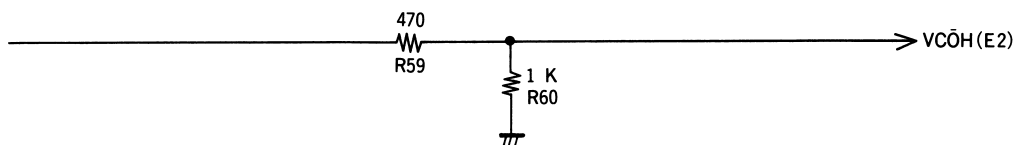




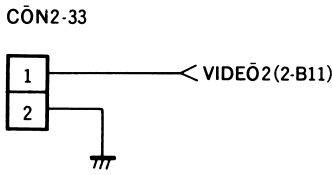
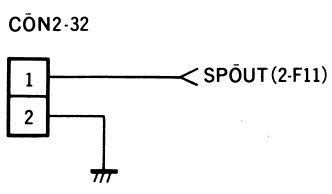
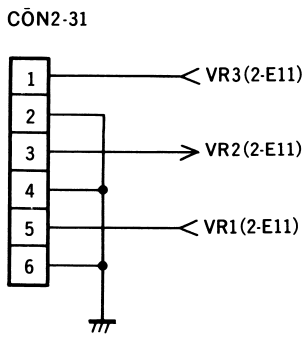
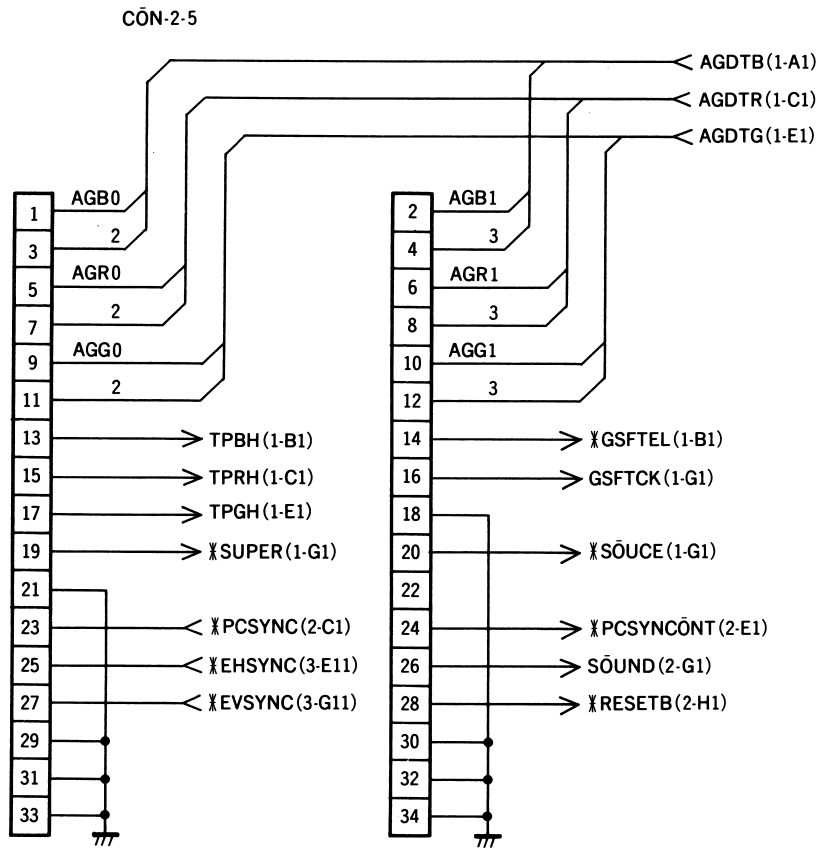
53. SYNCセパレータ



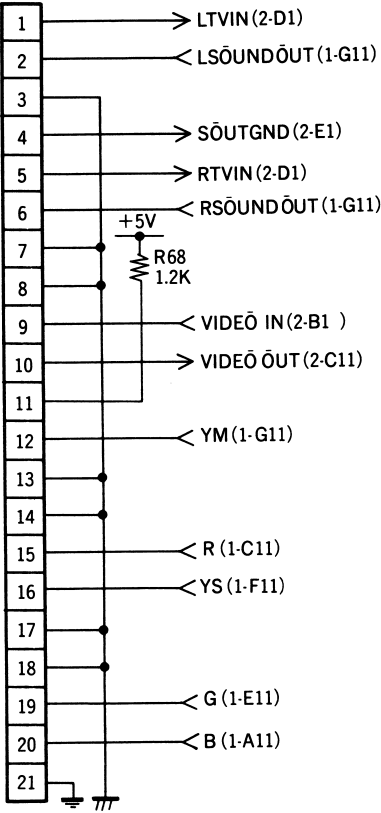
→ +11V(2-A1)



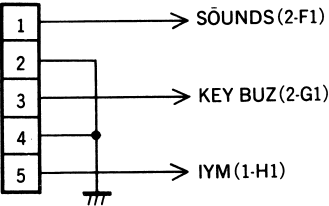
54. コネクタ



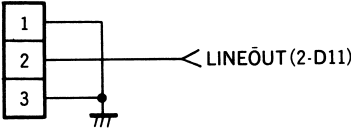
CÔN-2-24



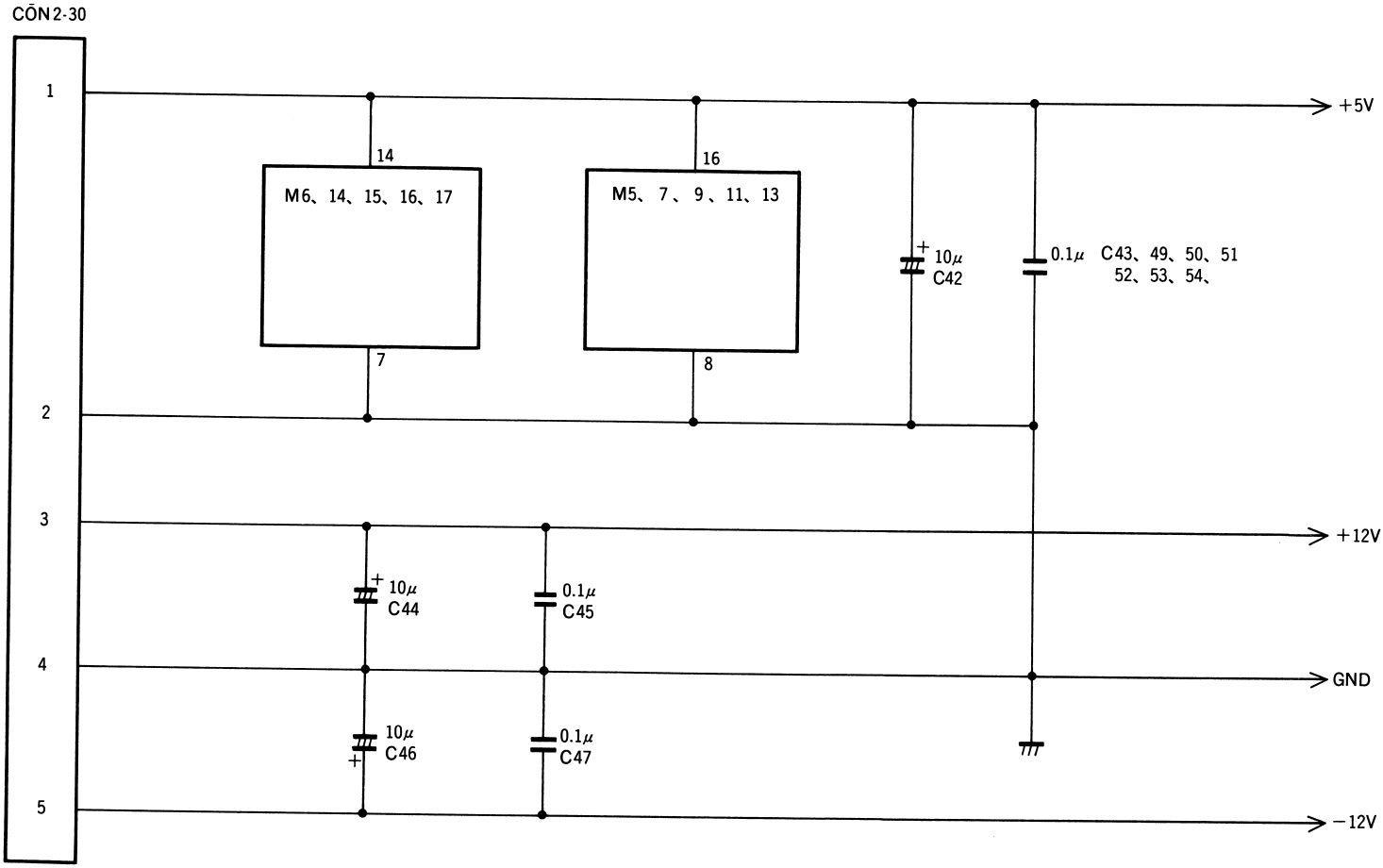
CÔN2-26



CÔN2-25



55. P/Sコネクタ



付 録

A. メインシステムI/Oアドレスマップ

アドレス	内 容	R/W	ビット 構 成							
			7	6	5	4	3	2	1	0
FD00	キーデータ	R	D8							
	オーディオカセット ラインプリンタ	W	プリンタ SLOTIN 0 : セレ クト 1 : 非セ レク ト	プリンタ STRB データを 送るため のストロ ープ (負 パルス)					オーディ オカセッ ト リモート 0 : オン 1 : オフ	オーディ オカセッ ト MIC カセット ライトデ ータ
FD01	キーデータ	R	D7	D6	D5	D4	D3	D2	D1	D0
	ラインプリンタ	W	D7	D6	D5	D4	D3	D2	D1	D0
FD02	オーディオカセット ラインプリンタ	R	オーディオ カセット SPK カセット リードデ ータ		プリンタ DET 2	プリンタ DET 1	プリンタ PE 0 : 正常 1 : 用紙 なし	プリンタ ACKNG IRQ 信号 (負論理)	プリンタ ERROR 0 : エラー 1 : 正常	プリンタ BUSY 0 : レディ 1 : ビジー
	割り込み (IRQ) の マスク	W	0 : マスク 1 : 解除 RS-232C SYNDET RXRDY TXRDY MFD TIMER プリンタ KEY							
FD03	割り込み (IRQ) の フラグ	R					拡張 0 : あり	TIMER 0 : あり	プリンタ 0 : あり	KEY 0 : あり
	ブザー	W	連続ブザー 0 : オフ 1 : オン	単一ブザー 0 : オフ 1 : オン						スピーカ 0 : オフ 1 : オン
FD04	サブインタフェース	R							BREAKキー 0 : オン 1 : オフ	ATENT 0 : あり 1 : なし
FD05	サブステータス 拡張ステータス	R	BUSY 0 : レディ 1 : ビジー							EXTDET 0 : あり
	サブインタフェース	W	HALT 1 : サブ ホルイト ネーブル	CANCEL 1 : サブ IRQリクエ スト						

アドレス	内 容	R/W	ビット 構 成																													
			7	6	5	4	3	2	1	0																						
FD06	RS-232C インタフェース (オプション)	R	シリアル受信データ																													
		D7	D6	D5	D4	D3	D2	D1	D0																							
FD07		W	シリアル送信データ																													
		D7	D6	D5	D4	D3	D2	D1	D0																							
同期モード		R	ステータス レジスタ																													
		DSR 0: "1" 1: "0"	SYNDET 1: SYNC キャラクタ検出	FE 1: フレームエラー	OE 1: オーバーラン エラー	PE 1: パリティエラー	TXE 1: トランスミット エンブティ	RXRDY 1: レシーバレディ	TXRDY 1: トランスミット データバッファ エンブティ																							
		W	モードコマンドデータ(リセット後有効)																													
		SCS SYNC キ ャラクタ 0: ダブル	ESD SYNC 検 出 0: 内部	EP パリティ 0: 奇	PE パリティ 0: ディ セーブル 1: イネ ーブル	L2 キャラクタ長 5ビット 6ビット 7ビット 8ビット	L1 0 0 1 0 1 1	0	0																							
非同期モード			S2 ストップビット	SI	同上	同上	同上	B2 ボーレート分周	BI																							
			<table><tr><td></td><td>S2</td><td>SI</td></tr><tr><td>1ビット</td><td>0</td><td>1</td></tr><tr><td>1.5ビット</td><td>1</td><td>0</td></tr><tr><td>2ビット</td><td>1</td><td>1</td></tr></table>		S2	SI	1ビット	0	1	1.5ビット	1	0	2ビット	1	1				<table><tr><td></td><td>BI</td><td>B2</td></tr><tr><td>1 / 1</td><td>0</td><td>1</td></tr><tr><td>1 / 16</td><td>1</td><td>0</td></tr><tr><td>1 / 64</td><td>1</td><td>1</td></tr></table>		BI	B2	1 / 1	0	1	1 / 16	1	0	1 / 64	1	1	
				S2	SI																											
			1ビット	0	1																											
1.5ビット	1	0																														
2ビット	1	1																														
	BI	B2																														
1 / 1	0	1																														
1 / 16	1	0																														
1 / 64	1	1																														
コマンドデータ																																
EH ハントモ ード	IR 内部リセ ット	RTS リクエスト ツウセン ド	ER エラーリ セット	SBRK ブレーク 送出	RXE 受信イネ ーブル	DTR ターミナ ルレディ	TXEN 送信イネ ーブル																									
1: SYNC キャラク タサーチ	1: リセ ット	1: RTS "0"	1: エラ ーフラグ クリア	1: TXD "L"	0: ディ セーブル 1: イネ ーブル	1: DTR "0"	0: ディ セーブル 1: イネ ーブル																									

FD0B	ブート選択スイッチ ステータス		* 1 *							MODE 0: BAS ICモード 1: DOS モード		
FD0C	リザーブ											
FD0D	PSG 音源 (FM 音源共用)	W	*	BDIR	BCI						BDIR	BCI
				0	0	ハイインピーダンス						
				0	1	データリード					*	*
				1	0	データライト						
FD0E		R	PSG データ									
			D7	D6	D5	D4	D3	D2	D1	D0		
			PSG データ(ライトデータ: レジスタアドレス)									
			D7	D6	D5	D4	D3	D2	D1	D0		

A. メインシステムI/Oアドレスマップ

ビット 構 成																																																
アドレス	内 容	R/W	7	6	5	4	3	2	1	0																																						
FD0F	バンクレジスタ	R	ROM モード																																													
		W	RAM モード																																													
FD10	イニシエータROM ディセーブルレジスタ	W							イニシエータROM 0：有効 1：無効																																							
		rst							"0"																																							
FD11	リザーブ																																															
FD12	サブモードステータス レジスタ	R		320/640 切り換え 1：320 0：640	"1"				DISPTMG ステータス 1：表示 期間 0：ブラ ンク	VSYNC ステータス 1：周期 期間																																						
		W		同 上	"0"																																											
		rst	"0"	"0"																																												
FD13	サブバンクレジスタ	R	"1"																																													
		W	"0"						SB2	SB1																																						
		* <table><tr><td>SE2</td><td>SB1</td><td>サブモニタROM</td></tr><tr><td>0</td><td>0</td><td>旧サブモニタ</td></tr><tr><td>0</td><td>1</td><td>新サブモニタ 3 (640×200モード用)</td></tr><tr><td>1</td><td>0</td><td>新サブモニタ 3 (320×200モード用)</td></tr><tr><td>1</td><td>1</td><td>未定義</td></tr></table>						SE2	SB1	サブモニタROM	0	0	旧サブモニタ	0	1	新サブモニタ 3 (640×200モード用)	1	0	新サブモニタ 3 (320×200モード用)	1	1	未定義	*	*																								
		SE2	SB1	サブモニタROM																																												
		0	0	旧サブモニタ																																												
0	1	新サブモニタ 3 (640×200モード用)																																														
1	0	新サブモニタ 3 (320×200モード用)																																														
1	1	未定義																																														
rst							0	0																																								
FD14	リザーブ																																															
FD15	FM音源	W	コントロールレジスタ "0"				<table><tr><td colspan="4">ビット</td></tr><tr><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>ハイインピーダンス</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>データリード</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>データライト</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>レジスタアドレスラッチ</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ステータスレジスタリード</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>ジョイスティックポートリード</td></tr></table>				ビット				3	2	1	0	0	0	0	0	ハイインピーダンス	0	0	0	1	データリード	0	0	1	0	データライト	0	0	1	1	レジスタアドレスラッチ	0	1	0	0	ステータスレジスタリード	1	0	0	1	ジョイスティックポートリード
							ビット																																									
							3	2	1	0																																						
							0	0	0	0	ハイインピーダンス																																					
0	0	0	1	データリード																																												
0	0	1	0	データライト																																												
0	0	1	1	レジスタアドレスラッチ																																												
0	1	0	0	ステータスレジスタリード																																												
1	0	0	1	ジョイスティックポートリード																																												
R	禁 止																																															
rst						0	0	0	0																																							
FD16		W	ライトデータ, レジスタアドレス																																													
			D7	D6	D5	D4	D3	D2	D1	D0																																						
		R	リードデータ, ステータス																																													
D7	D6	D5	D4	D3	D2	D1	D0																																									

アドレス	内 容	R/W	ビット 構 成								
			7	6	5	4	3	2	1	0	
FDI7	割り込みレジスタ	R					OPN 0 : 割り込み	マウス 0 : 割り込み			
		W						マウス 1 : 許可			
FDI8	320KBフロッピー	R	ステータスレジスタ (type 1)								
			ノットレディ 1 : ノットレディ	Wプロテクト 1 : プロテクト	ヘッドロード 1 : オン	シークエラー 1 : エラー	CRCエラー 1 : エラー	トラック0 1 : 00	インデックス 1 : オン	ビジー 1 : ビジー	
			ステータスレジスタ (type 2, 3)								
					レコードタイプ 0 : データマーク 1 : デリREETDアドレスマーク	レコードノットファウンド 1 : セット		ロストデータ 1 : ロスト	データリクエスト 1 : セット		
		W	コマンドレジスタ								
			TYPE 1	Restore				0000 1V10			
				Seek				0000 1V10			
				Step				001U 1V10			
				Step in				010U 1V10			
				Step out				011U 1V10			
TYPE 2	Read				100m xE00						
	Write(データマーク)				101m xE01						
	Write(デリREETD アドレスマーク)				101m xE00						
TYPE 3	Read Address				1100 0100						
	Read Track				1110 010x						
	Write Track				1111 0100						
TYPE 4	Force Int				1101 i3i2i1i0						
V : Verify, U : Up Date, m : multiple Record E : 30 msec dela, i : interrupt											
FDI9	R/W		トラック レジスタ								
FDIA	R/W		セクタ レジスタ								
FDIB	R/W	データ レジスタ									
FDIC	R	ヘッド レジスタ									
	W	ヘッド レジスタ							0 : サイド0 1 : サイド1		
FDID	R	ドライブ レジスタ									
	W	ドライブ レジスタ モータ 0 : オフ 1 : オン						ドライブ No.			
FDIF	R	DRQ 1 : ON	IRQ 1 : ON								

A. メインシステムI/Oアドレスマップ

アドレス		内 容		R/W	ビット 構 成									
					7	6	5	4	3	2	1	0		
FD20	漢字ROM		W	漢字アドレス(H)										
FD21			W	漢字アドレス(L)										
FD22			R	漢字データ(L)										
FD23			R	漢字データ(R)										
FD30	アナログ パレット番号		W	“ 0 ”					LC11	LC10	LC9	LC8		
			rst											
FD31	ログ		W	LC7	LC6	LC5	LC4	LC3	LC2	LC1	LC0			
			rst											
FD32	Blueレベル レジスタ		W	“ 0 ”					0 : Low ↔ 15 : High BL3 BL2 BL1 BL0					
			rst											
FD33	Redレベル レジスタ		W	“ 0 ”					0 : Low ↔ 15 : High RL3 RL2 RL1 RL0					
			rst											
FD34	Greenレベル レジスタ		W	“ 0 ”					0 : Low ↔ 15 : High GL3 GL2 GL1 GL0					
			rst											
FD35	音声合成 (オプション)		R	BUSY 0 : レディ 1 : ビジー	音声合成 0 : 有り 1 : 無し						ERR 0 : Not 1 : エラー			
			W	データレジスタ D7 D6 D5 D4 D3 D2 D1 D0										
FD37	マルチページ		W	ディスプレイ G R B 1 : ディ セーブル 0 : イネ ーブル		1 : ディ セーブル 0 : イネ ーブル		1 : ディ セーブル 0 : イネ ーブル		G R B 1 : ディ セーブル 0 : イネ ーブル		1 : ディ セーブル 0 : イネ ーブル		
FD38	パレットレジスタ		R								G	R	B	
			W								G	R	B	
FD39			R								G	R	B	
			W								G	R	B	
FD3A			R								G	R	B	
			W								G	R	B	
FD3B			R								G	R	B	
			W								G	R	B	
FD3C			R								G	R	B	
			W								G	R	B	
FD3D			R								G	R	B	
			W								G	R	B	
FD3E			R								G	R	B	
			W								G	R	B	
FD3F			R								G	R	B	
			W								G	R	B	

付 録

アドレス	内 容	R/W	ビット 構 成							
			7	6	5	4	3	2	1	0
FD80	メモリ・マネージメント・レジスタ (MMR)	R/W	CPUアドレス\$0000~0FFFの割り当てメモリアドレス							
FD81		R/W	CPUアドレス\$1000~1FFFの割り当てメモリアドレス							
FD82		R/W	CPUアドレス\$2000~2FFFの割り当てメモリアドレス							
FD83		R/W	CPUアドレス\$3000~3FFFの割り当てメモリアドレス							
FD84		R/W	CPUアドレス\$4000~4FFFの割り当てメモリアドレス							
FD85		R/W	CPUアドレス\$5000~5FFFの割り当てメモリアドレス							
FD86		R/W	CPUアドレス\$6000~6FFFの割り当てメモリアドレス							
FD87		R/W	CPUアドレス\$7000~7FFFの割り当てメモリアドレス							
FD88		R/W	CPUアドレス\$8000~8FFFの割り当てメモリアドレス							
FD89		R/W	CPUアドレス\$9000~9FFFの割り当てメモリアドレス							
FD8A		R/W	CPUアドレス\$A000~AFFFの割り当てメモリアドレス							
FD8B		R/W	CPUアドレス\$B000~BFFFの割り当てメモリアドレス							
FD8C		R/W	CPUアドレス\$C000~CFFFの割り当てメモリアドレス							
FD8D		R/W	CPUアドレス\$D000~DFFFの割り当てメモリアドレス							
FD8E		R/W	CPUアドレス\$E000~EFFFの割り当てメモリアドレス							
FD8F		R/W	CPUアドレス\$F000~FBFFの割り当てメモリアドレス							
FD90	MMR セグメントレジスタ	W							SI	S0
		rst							0	0
FD92	ウィンドウ・オフセット・レジスタ	W	ウィンドウ・オフセット・アドレス(\$00000~0FFFF固定)							
		rst	0	0	0	0	0	0	0	0
FD93	モード・セレクト・レジスタ	R/W	MMR 0 : 無効 1 : 有効	ウィンドウ 0 : 無効 1 : 有効						ブート RAM 0 : RO 1 : R/W
		rst	0	0						0

A. メインシステムI/Oアドレスマップ

		ビット構成								
アドレス	内 容	R/W	7	6	5	4	3	2	1	0
FDE0	マウス インタフェース (MB8873) (オプション)	W	コントロールレジスタ2のビット0 = 0 のとき コントロールレジスタ3への書き込み							
			コントロールレジスタ2のビット0 = 1 のとき コントロールレジスタ1への書き込み							
FDE1		R	ステータスレジスタ							
		複合割り 込みフラ グ	NO USE 0 0 0 0				タイマ3 割り込み	タイマ2 割り込み	タイマ1 割り込み	
		W	コントロールレジスタ2							
		出力許可 0 : 禁止 1 : 許可	割り込み 許可 0 : 禁止 1 : 許可	動作モード				カウント 0 : 16 1 : 8ビ ット	クロック 0 : C 1 : E	レジスタ 0 : レジ スタ3 1 : レジ スタ1
FDE2 E3		R/W	タイマ1 カウンタ (16ビット or 8ビット)							
FDE4 E5		R/W	タイマ2 カウンタ (16ビット or 8ビット)							
FDE6 E7		R/W	タイマ3 カウンタ (16ビット or 8ビット)							
FDE8		R	ステータスレジスタ							
		未使用	SW3 (拡張) 0 : OFF 1 : ON	SW2 0 : OFF 1 : ON	SW1 0 : OFF 1 : ON	Q3	Q2	Q1	Q0	
		W	コントロールレジスタ							
	未使用 (通常は“0”を書き込む)							HC 0 : ホー ルドなし 1 : ホー ルド	カウント (E) 0 : カウ ント 1 : 停止	

アドレス	内 容	R/W	ビ ッ ト 構 成							
			7	6	5	4	3	2	1	0
FDE9	MIDI インタフェース (オプション)	W	あ き リセット時：割り込み禁止(：0)						PIT 割込 I：許可	USART 割込 I：許可
FDEA		R	受信データ							
		D7	D6	D5	D4	D3	D2	D1	D0	
		W	送信データ							
		D7	D6	D5	D4	D3	D2	D1	D0	
FDEB	USART (i8251A)	R	ステータスレジスタ							
		DSR 端子 0：H I：L	SYNDET I：検出	FE I：検出	OE I：検出	PE I：検出	TXE I：あき	RXRDY I：データ OK	TXRDY I：あき	
		W	モードコマンドレジスタ(リセット後有効：同期モード)							
		SCS syncキャラ 0：ダブル I：シングル	SYNDET I：検出	EPパリティ 0：Even I：Odd	PEパリティ 0：ディセーブル I：イネーブル	キャラクタ長 L2 L1		0	0	
		モードコマンドレジスタ(リセット後有効：非同期モード)								
		ストップビット S2 S1	EPパリティ 0：Even I：Odd	PEパリティ 0：ディセーブル I：イネーブル	キャラクタ長 L2 L1		ポーレートクロック B2 B1			
コマンドレジスタ										
		EH sync I：サーチ	IR 内部 I：リセット	RTS I： $\overline{RTS}$ =0	ER エラー I：リセット	SBRK I：TXD=0	RXE 0：ディセーブル I：イネーブル	DTR I：DTR=0	TXE 0：ディセーブル I：イネーブル	
FDEC	PIT (i8253)	R/W	Counter 1							
			D7	D6	D5	D4	D3	D2	D1	D0
FDED			Counter 2							
			D7	D6	D5	D4	D3	D2	D1	D0
FDEE		Counter 3								
		D7	D6	D5	D4	D3	D2	D1	D0	
FDEF		W	Control Word							
		SC1	SC0	RL1	RL0	M2	M1	M0	BCD	

＜OPNレジスタアドレス＞

アドレス	内 容	7	6	5	4	3	2	1	0
7	イネーブルレジスタ	Bポート 入出力方 向 0：入力 1：出力	Aポート 入出力方 向 0：入力 1：出力	NOISE 0：出力する 1：出力しない チャンネル			TONE 0：出力する 1：出力しない チャンネル		
				C	B	A	C	B	A
E	ポートA データレジスタ(入力)	"1" (未使用)	"1" (未使用)	ジョイスティックステータス 0：Pressed 1：Not					
				トリガ2	トリガ1	右	左	後	前
F	ポートB データレジスタ(出力)	"0" (未使用)	ジョイス ティック 切り換え 0：J1 1：J2	J2 COM 出力	J1 COM 出力	J2 トリガ2 相当 出力	J2 トリガ1 相当 出力	J1 トリガ2 相当 出力	J1 トリガ1 相当 出力

B. サブシステムI/Oアドレスマップ

アドレス	内 容	R/W	ビット 構 成							
			7	6	5	4	3	2	1	0
D400	キーデータ	R	D8							
D401	キーデータ	R	D7	D6	D5	D4	D3	D2	D1	D0
D402	キャンセルIRQの アクノリッジ	R	負パルス							
D403	ブザー	R	負パルス							
D404	メインCPUへの アテンションIRQ	R	負パルス							
D408	CRTのON-OFF	R	ON							
		W	OFF							
		rst	OFF							
D409	VRAMアクセスフラグ (SET=アクセス) アクノリッジ	R	SET							
		W	RESET							
		rst	RESET							
D40A	BUSYフラグ	R	READY							
		W	BUSY							
		rst	BUSY							
D40D	インサートモード LEDのON-OFF	R	ON							
		W	OFF							
		rst	OFF							
D40E	スクリーンRAMの オフセットアドレス(High)	W			OA13	OA12	OA11	OA10	OA9	OA9
		rst			0	0	0	0	0	0
D40F	スクリーンRAMの オフセットアドレス(Low)	W	OA7	OA6	OA5	OA4	OA3	OA2	OA1	OA0
		rst	0	0	0	0	0	0	0	0

アドレス	内 容	R/W	ビット 構 成						
			7	6	5	4	3	2	1
D410	論理演算	R/W	コマンドレジスタ						

B. サブシステムI/Oアドレスマップ

アドレス	内 容	R/W	ビット 構 成							
			7	6	5	4	3	2	1	0
D420	直線補間	W	アドレスオフセットレジスタ(H)							
D421		W	アドレスオフセットレジスタ(L)							
D422		W	ラインスタイルパターンレジスタ							
D423		W	ラインスタイルパターンレジスタ							
D424		W	X座標始点レジスタ(H)							
D425		W	X座標始点レジスタ(L)							
D426		W	Y座標始点レジスタ(H)							
D427		W	Y座標始点レジスタ(L)							
D428		W	X座標終点レジスタ(H)							
D429		W	X座標終点レジスタ(L)							
D42A		W	Y座標終点レジスタ(H)							
D42B		W	Y座標終点レジスタ(L)							
D430	補間 BUSY/NMI/ バンクレジスタ	R	ディスプレイ 0: ブラ ンク 1: 表示			直接補間 0: BUSY 1: READY		VSYN ステータス 1: 同期 期間		RESET 0: POWER ON 1: サブ 切り換え
		W	NMI マス クレジスタ 1: マスク 0: 解除	ディスプレ イページ 0: Page 0 1: Page 1	アクティ ブページ 0: Page 0 1: Page 1			オフセッ トレジスタ OA0~ OA4 イネーブル レジスタ 0: ディ セーブル 1: イネ ーブル	CG サブシステム バンクレジスタ	
		rst	"0"	"0"	"0"	"0"		"0"	"0"	"0"
D431	エンコーダデータ レジスタ	R/W	M7	M6	M5	M4	M3	M2	M1	M0
		rst	"0"							
D432	エンコーダ ステータス	R	LATCH 0: 受信 可能 1: 受信 不能	"1"						ACK 0: 送信 未完 1: 送信 完了

C. F-BASIC V3.0 メモリマップ

ア ド レ ス	内 容
6E00～71D5	DISK BASIC イニシャライズ処理
71D6～7313	ワークエリア
7314～732F	DISK コマンド予約語テーブル
7330～733B	DISK コマンド ジャンプテーブル
733C～735C	DISK 関数 予約語テーブル
735D～736E	DISK 関数 ジャンプテーブル
736F～7385	DISK コマンド 分岐ルーチン
7386～73A0	DISK 関数 分岐ルーチン
73A1～7564	DISKINI 文エントリ
7422～	EAT イニシャライズ処理
74CE～	セクタ READ/WRITE 処理
7565～75A3	DSKO\$文エントリ
75A4～75C8	DSKI\$文エントリ
75C9～7712	DISK ファイル KILL 処理
75FB～	EAT 書き込み
762C～	ディレクトリサーチ
7713～78BD	DISK ファイルオープン処理
78BE～7923	DISK ファイルクローズ処理
7924～79F8	DISK ファイル I バイト出力処理
79F9～7B56	DISK ファイル I バイト入力処理
7B57～7C14	FILES 処理
7C15～7C3C	DSKF 文エントリ
7C3D～7C3F	CVI 文エントリ
7C40～7C42	CVS 文エントリ
7C43～7C54	CVD 文エントリ
7C55～7C57	MKI\$文エントリ
7C58～7C5A	MKS\$文エントリ
7C5B～7C71	MKD\$文エントリ
7C72～7CE5	LOC 文エントリ
7CE6～7D4D	NAME 文エントリ
7D4E～7DA6	FIELD 文エントリ
7DA7～7E00	RSET 文エントリ
7DA8～	LSET 文エントリ
7E01～7E03	PUT 文エントリ
7E04～7EAF	GET 文エントリ
7EB0～7FFF	DISK エラー処理
8000～802F	初期設定処理, データ
8030～804D	拡張コマンド予約語テーブル
804E～8059	拡張コマンドジャンプテーブル

アドレス	内 容
805A～8070	拡張コマンド 分岐ルーチン
8071～8074	拡張関数 予約語テーブル
8075～8076	拡張関数 ジャンプテーブル
8077～808D	拡張関数 分岐ルーチン
808E～8447	CHAIN 文エントリ
8448～848A	ERASE 文エントリ
848B～8596	BASIC コールドスタート処理
8597～859E	ブロック転送ルーチン
859F～85A6	3 バイトデータ転送ルーチン
85A7～8815	BASIC イニシャライズルーチン, データ
8816～886B	標準関数 ジャンプテーブル
886C～888F	2 項演算優先順位テーブル, ジャンプテーブル
8890～8A2B	標準コマンド及び2項演算 予約語テーブル
8A2C～8AD9	標準関数 予約語テーブル
8ADA～8B71	標準コマンド ジャンプテーブル
8B72～8D22	中間言語・予約語テーブルの対応データ
8D23～8D56	予約語 I 文字目の見出しテーブル
8D57～8D60	メッセージ(Ready)文字列
8D61～8D68	メッセージ(Break)文字列
8D69～8D92	スタック上のネスティング情報検索
8D93～8DA9	ブロック転送ルーチン
8DAA～8DD0	メモリフルチェック
8DD1～8E87	エラー処理エントリ
8E88～8F31	テキスト エディタ エントリ
8F32～8FBB	NEW 文エントリ
8FBC～8FCF	RESTORE 文エントリ
8FD0～8FD6	END 文エントリ
8FD7～8FD8	Break 処理
8FD9～9001	STOP 文エントリ
9002～9011	CONT 文エントリ
9012～902C	RUN 文エントリ
902D～9038	GO 文エントリ
9039～9054	GOSUB 文エントリ
9055～9070	GOTO 文エントリ
9071～909F	RETURN 文エントリ
90A0～90A2	DATA 文エントリ
90A3～90F0	REM, ,ELSE 文エントリ
90F1～9133	IF 文エントリ
9134～9139	ON(式), ON ERROR の分岐
913A～9194	ON(式) GOTO/GOSUB 文エントリ

ア ド レ ス	内 容
9195～91EC	LET 文エントリ
91ED～924D	単項式の評価
924E～9287	NOT 文エントリ
9288～92A4	多項式の評価
928C～	右カッコの処理
928F～	左カッコの処理
92A5～92BD	減算(－)エントリ
92BE～94C4	数値式の評価
939E～	べき乗(Δ)
93B0～	論理演算
93BB～	¥, MOD
93CD～	除算(/)
946B～	>, =, <
94A8～	AND
94AD～	OR
94B2～	XOR
94B7～	EQV
94BC～	IMP
94C5～94CD	DIM 文エントリ
94CE～950E	変数名テーブル登録
950F～972B	変数サーチルーチン
9547～	単純変数テーブル検索
9570～	単純変数テーブル登録
95C7～	変数型識別子の作成
95E6～	配列変数テーブル検索
966E～	配列変数テーブル登録
972C～975I	FRE 関数エントリ
9752～980C	STR\$関数エントリ
980D～98AD	カベージコレクション処理
98AE～9929	文字列代入処理
922A～9932	LEN 関数エントリ
9933～9946	CHR\$関数エントリ
9947～995I	ASC 関数エントリ
9952～996E	LEFT\$関数エントリ
996F～9978	RIGHT\$関数エントリ
9979～99C9	MID\$関数エントリ
99CA～9A25	VAL 関数エントリ
9A26～9A2F	PEEK 関数エントリ
9A30～9A39	POKE 文エントリ
9A3A～9A4F	行番号定数の評価

アドレス	内 容
9A50～9A7E	TAB(n)関数エントリ
9A7F～9AB2	LPRINT 文エントリ
9AB3～9BB4	PRINT 文エントリ
9B47～	改行処理
9B68～	PRINT 文中のカンマ(,)の処理
9B7E～	PRINT 文中のセミコロン(;)の処理
9BB5～9BDA	SPC(n)関数エントリ
9BDB～9C2A	文字列出力
9C2B～9CD3	PRINT@文エントリ
9CD4～9D58	INSTR 関数エントリ
9D59～9D7A	STRING\$関数エントリ
9D7B～9DC0	SPACE\$関数エントリ
9DC1～9E2B	MID\$関数エントリ
9E2C～9E7D	HEX\$関数エントリ
9E2D～9E7D	OCT\$関数エントリ
9E7E～9E93	&定数エントリ
9E94～9EA4	&O 定数エントリ
9EA5～9ECB	H 定数エントリ
9ECC～9F72	LIST 出力処理
9F73～9F75	DEFSTR 文エントリ
9F76～9F78	DEFINT 文エントリ
9F79～9F7B	DEFSNG 文エントリ
9F7C～9FBB	DEFDBL 文エントリ
9FBC～9FE6	DELETE 文エントリ
9FE7～A019	AUTO 文エントリ
A01A～A01E	TRON 文エントリ
A01B～	TROFF 文エントリ
A01F～A202	PRINT USING 文(!)の処理
A058～	PRINT USING 文(&)の処理
A07F～	PRINT USING 文エントリ
A203～A43E	FOR 文エントリ
A2C7～	NEXT 文エントリ
A36D～	WEND 文のサーチ
A36E～	NEXT 文のサーチ
A43F～A445	PRINT USING のパッチ
A446～A70E	エラーメッセージ テーブル
A70F～A738	ON ERROR GOTO 文エントリ
A739～A743	ERROR 文エントリ
A744～A783	RESUME 文エントリ
A784～A7D8	SWAP 文エントリ

アドレス	内 容
A7D9～A7DE	DEF 文エントリ
A7DF～A829	DEF FN 文エントリ
A82A～A951	FN 関数エントリ
A952～A968	VARPTR 関数エントリ
A969～A99F	DEF USR 文エントリ
A9A0～A9C9	USR 関数エントリ
A9CA～ABF3	RENUM 文エントリ
ABF4～AD53	MON 文エントリ
AC2F～	G コマンド処理
AC95～	M コマンド処理
ACC4～	D コマンド処理
ACFB～	R コマンド処理
AD54～ADDB	WHILE 文エントリ
AD81～	WEND 文エントリ
ADDC～AED1	HARDC 文エントリ
ADF0～	HARDC0 処理
AE69～	HARDC1 処理
AE8C～	HARDC2 処理
AED2～B08C	ANPORT 文エントリ
AF11～	実数減算ルーチン
AF1A～	実数加算ルーチン
AF97～	実数の正規化処理
B00A～	実数の丸め処理
B068～	実数の切り捨て処理
B08D～B0A9	LOG 演算用データ
B0AA～B3BD	LOG 関数エントリ
B0EE～	実数乗算ルーチン
B1DF～	指数部演算ルーチン
B234～	実数除算ルーチン
B3BE～B3D4	SGN 関数エントリ
B3D5～B460	ABS 関数エントリ
B461～B471	FIX 関数エントリ
B472～B608	INT 関数エントリ
B609～BAED	「IN(行番号)」の出力
BAEE～BAF6	SOR 関数エントリ
BAF7～BB35	べき乗(Δ)エントリ
BB36～BB5A	EXP 演算用データ
BB5B～BB93	EXP 関数エントリ
BB94～BBE4	多項式演算処理
BBE5～BBFD	RANDOMIZE 文エントリ

アドレス	内 容
BBFE～BC4B	RND 関数エントリ
BC4C～BC73	RND 演算用データ
BC74～BCCC	CINT 関数エントリ
BC7C～	単精度データの整数化
BC85～	倍精度データの整数化
BCCI～	型変換ルーチン
BCCD～BCDF	CDBL 関数エントリ
BCD3～	整数データの倍精度化
BCD5～	単精度データの倍精度化
BCE0～BD28	CSNG 関数エントリ
BCEE～	倍精度データの単精度化
BCFF～	整数データの単精度化
BD29～BD3E	整数/実数のチェック処理
BD3F～BD4C	減算(－)エントリ
BD4D～BD72	加算(＋)エントリ
BD73～BE0B	乗算(×)エントリ
BE0C～BE32	整数除算(÷)エントリ
BE33～BE49	MOD エントリ
BE4A～BE5F	数値比較(<, =, >)エントリ
BE60～BE65	COS 関数エントリ
BE66～BEB0	SIN 関数エントリ
BEB1～BED3	TAN 関数エントリ
BED4～BEF4	三角関数演算用データ
BEF5～BF47	ATN 関数エントリ
BF48～BFDC	LINE INPUT 文エントリ
BFDD～BFF2	INPUT 文エントリ
BFF3～C168	READ 文エントリ
C169～C287	テキスト逆翻訳ルーチン
C288～C5E1	テキスト翻訳ルーチン
C490～	小文字→大文字変換
C528～	行番号定数の評価
C5E2～C63C	標準関数の分岐処理
C63D～C6EC	BASIC メインループ
C67A～	ダイレクトモード エントリ
C680～	各コマンドへの分岐処理
C6DB～	暗黙 END 処理
C6ED～C72F	省略形キーワードテーブル
C730～C75F	リンクポインタ付け替え処理
C760～C76E	テキスト読み込みルーチン
C76F～C77A	EXEC 文エントリ

ア ド レ ス	内 容
C77B～C7C9	CLEAR 文エントリ
C7CA～C7F0	COLOR 文エントリ
C7F1～C813	COLOR=(パレット, カラー)文エントリ
C814～C87A	CONSOLE 文エントリ
C87B～C8FF	WIDTH 文エントリ
C8BC～	コンソール イニシャライズ処理
C8F0～C911	LOCATE 文エントリ
C912～C922	UNLIST 文エントリ
C923～C929	CSRLIN 文エントリ
C92A～C946	POS 関数エントリ
C947～C952	LPOS 関数エントリ
C953～C9D7	FIRQ 割込処理
C9D8～C9E7	行番号評価, サーチルーチン
C9E8～C9F9	カセット処理ジャンプテーブル
C9FA～CA01	カセット EOF 処理
CA02～CA05	カセット LOF 処理
CA06～CA8C	カセットオープン処理
CA18～	*O*モードのオープン処理
CA73～	*I*モードのオープン処理
CA8D～CAB5	カセット I ブロック入力処理
CAB6～CAE3	カセットクローズ処理
CAE4～CAE5	LOAD ? エントリ
CAE6～CBB4	SKIPF エントリ
CB02～	カセット I バイト入力(バッファから)
CB1E～	カセット I バイト出力(バッファへ)
CB36～	カセット I ブロック出力処理
CB57～	ヘッダブロック読み込み処理
CBB5～CD00	FILES 文エントリ
CD01～CE03	LIST 文エントリ
CD3E～	SAVE 文エントリ
CD58～	プロテクト暗号化処理
CD88～	LLIST 文エントリ
CDA6～	プロテクト復号化処理
CDC6～	プロテクトチェック
CE04～CE72	SAVEM 文エントリ
CE73～CEA7	CLOSE 文エントリ
CEA8～CF21	OPEN 文エントリ
CEDC～	*I*モードのオープン処理
CEDF～	*O*モードのオープン処理
CF22～D02F	RUN 文エントリ

アドレス	内 容
CF29～	MERGE 文エントリ
CF31～	LOAD 文エントリ
D027～	1 バイト入力処理
D030～D119	LOADM エントリ
D06B～	2 バイト入力処理
D072～	1 バイト入力ルーチン
D08E～	1 バイト出力ルーチン
D11A～D11C	EOF 関数エントリ
D11D～D15B	LOF 関数エントリ
D15C～D197	INPUT\$関数エントリ
D198～DIAC	COM クローズ処理
DIAD～D25B	COM オープン処理
D25C～D2B4	COM 1 バイト出力処理
D2B5～D2FB	COM 1 バイト入力処理
D2FC～D3A4	IRQ 割込処理
D311～	KEY IRQ 処理
D315～	TIMER IRQ 処理
D319～	拡張 IRQ 処理
D3A5～D3B3	COM EOF 処理
D3B4～D3C1	COM LOF 処理
D3C2～D4BA	BASIC プログラム割込制御処理
D407～	ON COM 文エントリ
D444～	COM エントリ
D448～	COMMON エントリ
D464～	COM OFF 処理
D475～	COM ON 処理
D484～	COM STOP 処理
D498～	割込みイニシャライズ
D4BB～D50D	TERM モードの PF キー割込処理
D50E～D517	TERM モードの PF キー割込ジャンプテーブル
D518～D52C	TERM モードのオプション転送処理
D52D～D607	TERM 文エントリ
D5D6～	入力文字の出力
D5E3～	LOF チェック
D5EC～	エスケープシーケンス処理
D608～D60C	TERM オペランド既定値データ
D60D～D614	TERM ジャンプテーブル
D615～D616	プリンタクローズ処理
D617～D677	プリンタ 1 バイト出力処理
D64A～	改行処理

アドレス	内 容
D659～	プリンタのレディチェック
D678～D6D7	Abort 処理
D6D8～D757	カセット I ブロック入力処理
D730～	カセット I バイト入力(カセットテープより)
D758～D78B	MOTOR 文エントリ
D76F～	MOTOR ON 処理
D772～	MOTOR OFF 処理
D78C～D793	カセット ギャップ出力処理
D794～D7F6	カセット I ブロック出力処理
D7E3～	カセット I バイト出力(カセットテープへ)
D7F7～D8AB	一行入力処理(スクリーンエディット時)
D8AC～D99D	EDIT 文エントリ
D90F～	CRT I ブロック出力処理
D92F～	フィールド設定処理
D93D～	CRT イレーズ処理
D944～	行番号出力
D94A～	AUTO 編集処理
D99E～D9A3	CRT オープン処理
D9A4～	CRT クローズ処理
D9A5～D9D8	プリンタ オープン処理
D9D9～DA7F	CRT I バイト出力処理
D9DE～	カーソル X, Y 座標の読み取り
D9F4～	サブシステム関係 BIOS の RCB 設定
DA6D～	カーソル X, Y 座標設定
DA80～DAA5	CLS 文エントリ
DAA6～DADE	ファイルからの I 行入力処理
DADF～DB0C	カーソル消去/表示処理
DB0D～DB1E	キーボード処理ジャンプテーブル
DB1F～DB24	キーボード オープン処理
DB25～	キーボード クローズ処理
DB26～DB27	BEEP ON の RCB データ
DB28～DB29	BEEP OFF の RCB データ
DB2A～DB53	BEEP 文エントリ
DB38～	BEEP 処理(一定時間)
DB3F～	BEEPI 処理(BEEP ON)
DB44～	BEEP0 処理(BEEP OFF)
DB49～	BELL (CHR\$(7)) 出力
DB54～DB93	キーボード I 文字入力処理
DB59～	Break チェック ルーチン
DB6D～	INKEY ルーチン

アドレス	内 容
DB84～	INKEY\$文エントリ
DB94～DB97	KEY 文エントリ
DB98～DC24	KEY LIST 文エントリ
DBD5～	コンソール設定ルーチン
DBFB～	PF キー読み込み/表示処理
DC15～DC70	KEY(n)ON/OFF/STOP 処理
DC35～	KEY(n)ON 文エントリ
DC43～	KEY(n)OFF 文エントリ
DC51～	KEY(n)STOP 文エントリ
DC56～	PF キー割込み設定ルーチン
DC71～DCA5	PF キー定義ルーチン
DCA6～DCA9	ON 文エントリ
DCAA～DCCE	ON KEY 文エントリ
DCCF～DCDA	ON INTERVAL 文エントリ
DCDB～DCEE	ON TIME 文エントリ
DCEF～DCF9	ON PEN 文エントリ
DCFA～DDA7	CONNECT 文エントリ
DDA8～DE32	SYMBOL 文エントリ
DE33～DF53	GCURSOR 文エントリ
DEE2～	ファンクション チェック
DEF9～	グラフィック座標設定処理
DF30～	グラフィック座標評価
DF54～DF77	PAINT 境界色設定
DF78～DFA5	POINT 関数エントリ
DFA6～DFAB	PSET 文エントリ
DFAC～E013	PRESET 文エントリ
E014～E035	キャラクタ座標のチェック
E036～E063	X, Y 座標の評価
E064～E119	LINE 文エントリ
E11A～E15B	TIME 関数エントリ
E15C～E181	TIME\$ 関数エントリ
E182～E18F	DATE\$ 関数エントリ
E190～E1E3	DATE 関数エントリ
E1E4～E1F7	TIME 関数エントリ
E1F8～E232	TIME\$ 関数エントリ
E233～E23E	TIME ON 文エントリ
E23F～E248	TIME OFF 文エントリ
E249～E24E	TIME STOP 文エントリ
E24F～E290	TIME(割込み)文エントリ
E272～	タイマ割込み時刻設定

ア ド レ ス	内 容
E29I～E2CE	DATE\$文エントリ
E2CF～E2E8	設定日付,時刻のエラーチェック
E2DA～	うるう年チェック
E2E9～E34C	日付,時刻の設定処理
E34D～E3B9	TIMER 読み込み
E3BA～E40A	INTERVAL 文エントリ
E40B～E43C	INTERVAL ON 文エントリ
E43D～E447	INTERVAL OFF 文エントリ
E448～E44D	INTERVAL STOP 文エントリ
E44E～E452	0 時割込みイネーブル
E453～E4BA	0 時割込み設定
E4BB～E4C7	カラーチェック設定
E4C8～E51F	PAINT 文エントリ
E520～E543	CIRCLE 用三角関数データ
E544～E551	CIRCLE X, Y 座標評価
E552～E7EE	CIRCLE 文エントリ
E6A3～	CIRCLE 演算ルーチン
E7A3～	CIRCLE 開始/終了位置設定
E7EF～E809	GET 文エントリ
E80A～EAF5	PUT 文エントリ
EAF6～EB4A	SCREEN 関数エントリ
EB4B～EB8D	SCREEN 文エントリ
EB8E～EB90	BUBINI 文エントリ
EB91～EB93	BUBW 文エントリ
EB94～EB99	BUBR 文エントリ
EB9A～EBB9	KILL 文エントリ
EBBA～EBF5	SOUND 文エントリ
EBCF～	PSG レジスタ設定処理
EBE5～	PSG 初期化処理
EBF6～EC04	PLAY コマンド設定
EC05～EC13	PLAY コマンド初期化処理
EC14～EC1A	TIMER 割込みイネーブル
EC1B～EC26	TIMER 割込みマスク
EC27～EC28	BEEP OFF RCB データ
EC29～EC71	PLAY 文初期化処理
EC72～F143	PLAY 文エントリ
ED35～	MML コマンド解析ルーチン
ED72～	PLAY 文割込み処理
EE9A～	音の長さ計算
EEDA～	MML コマンド* T° 処理

アドレス	内 容
EEE5～	MML コマンド`L`処理
EEF0～	MML コマンド`O`処理
EEFC～	MML コマンド`V`処理
EEFE～	MML コマンド`S`処理
EF12～	オシレータ書き込み
EF19～	オシレータ読み込み
EF31～	MML コマンド`P``R`処理
EF44～	MML コマンド`A`～`G`処理
EF5C～	MML コマンド`N`処理
EFA1～	MML コマンド`M`処理
EFC9～	音程の設定
F144～F159	PLAY コマンド相対アドレスデータ
F15A～F175	音程データ
F176～F17C	BIOS のバージョン
F17D	BIOS エントリ
FBA5～FBAE	BASIC のバージョン

D. F-BASIC V3.3 メモリマップ

アドレス	内 容
9146～9161	DISK コマンド予約語テーブル
9162～916D	DISK コマンドジャンプテーブル
916E～918E	DISK 関数予約語テーブル
918F～91A0	DISK 関数ジャンプテーブル
91A1～91B7	DISK コマンド分岐ルーチン
91B8～91D2	DISK 関数分岐ルーチン
91ED～922B	常駐部ジャンプテーブル
922C～95F8	非常駐部フックテーブル
922C～	BASIC コールドスタート処理フック
923E～	CHAIN 文フック
9250～	ERASE 文フック
9256～	カセット I ブロック入力処理フック
925C～	カセット I ブロック出力処理フック
9262～	MOTOR OFF 処理フック
9268～	MOTOR 文フック
926E～	MOTOR ON 処理フック
929E～	ROLL 文フック
92A4～	DEF KANJI 文フック
92B0～	KANJI 文フック
92B6～	SCREEN 文フック
92D4～	LINE 文フック
92DA～	PAINT 文フック
92E0～	POINT 関数フック
92E6～	PRESET 文フック
92EC～	PSET 文フック
92F8～	CIRCLE 文フック
9309～	GET 文フック
930F～	PUT 文フック
9315～	CONNECT 文フック
931B～	SYMBOL 文フック
9321～	GCURSOL 文フック
936F～	HARDC 文フック
9380～	HARD0 文フック
9386～	倍精度 EXP 関数フック
9397～	倍精度 LOG 関数フック
939D～	倍精度 SQR 関数フック
93A3～	倍精度 SIN 関数フック
93A9～	倍精度 COS 関数フック
93AF～	倍精度 TAN 関数フック
93B5～	倍精度 ATAN 関数フック

アドレス	内 容
93C7～	MON 文フック
93DF～	DISK ファイルオープン処理フック
93E5～	DISK エラー処理フック
93EB～	DISK ファイル I バイト出力処理フック
93F1～	DISK ファイルクローズ処理フック
93F7～	DISK ファイル I バイト入力処理フック
9403～	FILES 文フック
941B～	GET 文フック
9421～	PUT 文フック
9432～	STICK 文フック
9438～	STRIG 文フック
943E～	JOY STICK イニシャライズ処理フック
9444～	KILL 文フック
944A～	DSKINI 文フック
9450～	DSKO\$ 文フック
9456～	NAME 文フック
945C～	FIELD 文フック
9462～	LSET 文フック
9468～	RSET 文フック
946E～	DSKF 文フック
9474～	CVI 文フック
947A～	CVS 文フック
9480～	CVD 文フック
9486～	MKI\$ 文フック
948C～	MKS\$ 文フック
9492～	MKD\$ 文フック
9498～	LOC 文フック
949E～	DSKI\$ 文フック
94A5～	セクタ READ/WRITE 処理フック
94B3～	FAT 更新フック
94EB～	INPUT 文フック
94F1～	LINE INPUT 文フック
94F7～	READ 文フック
94FD～	RENUM 文フック
9509～	SEARCH 文フック
950F～	CALL 文フック
9515～	TALK 文フック
951B～	リンクポインタ付け替え処理フック
952D～	行番号サーチルーチンフック
9539～	PALETTE 文フック

ア ド レ ス	内 容
954A～	SIMPOSE 文フック
9550～	SIMPOSE 関数フック
9556～	WIDTH 文フック
9562～	COLOR 文フック
9568～	SINPUT 文フック
9574～	CLOCK 文フック
957A～	DATE\$関数フック
9580～	DATE 関数フック
9586～	TIME 関数フック
958C～	TIME\$関数フック
9598～	INTERVAL 文フック
959E～	TIMER 読み込み処理フック
95A4～	VOICE 文フック
95B7～	PLAY 文フック
95BD～	BEEP 文フック
95DB～	PLAY 関数フック
95E7～	SOUND 文フック
95ED～	BGM 文フック
95F3～	OUTM 文フック
962B～9658	初期設定処理データ
9659～9692	拡張コマンド予約語テーブル
9693～96AE	拡張コマンドジャンプテーブル
96AF～96BE	拡張コマンド分岐ルーチン
96BF～972B	拡張関数予約語テーブル
972C～9739	拡張関数ジャンプテーブル
973A～9779	拡張関数分岐ルーチン
977A～9781	ブロック転送ルーチン
9845～989A	標準関数ジャンプテーブル
989B～98BE	2 項演算優先順位テーブル, ジャンプテーブル
98BF～9A5A	標準コマンド及び 2 項演算予約語テーブル
9A5B～9B08	標準関数予約語テーブル
9B09～9BA0	標準コマンドジャンプテーブル
9BA1～9D51	中間言語・予約語テーブルの対応データ
9D52～9D85	予約語 1 文字目の見出しテーブル
9DB0～9DB1	ブロック転送ルーチン
9DC7～9E12	メモリフルチェック
9E13～9F14	エラー処理エントリ
9F15～9FE0	テキストエディタエントリ
9FE1～A073	NEW 文エントリ
A074～A088	RESTORE 文エントリ

アドレス	内 容
A089～A08F	END 文エントリ
A090～A093	Break 処理
A094～A0CC	STOP 文エントリ
A0CD～A0D0	STOP ON 文エントリ
A0D1～A0E7	STOP OFF 文エントリ
A0E8～A0FA	CONT 文エントリ
A0FB～A112	RUN 文エントリ
A113～A11E	GO 文エントリ
A11F～A13D	GOSUB 文エントリ
A13E～A16E	GOTO 文エントリ
A16F～A1A4	RETURN 文エントリ
A1A5～A1A7	DATA 文エントリ
A1A8～A1F5	REM, , ELSE 文エントリ
A1F6～A239	IF 文エントリ
A23A～A23F	ON(式), ON ERROR の分岐
A240～A29C	ON(式)GOTO/GOSUB 文エントリ
A29D～A2F3	LET 文エントリ
A2F4～A353	単項式の評価
A354～A394	NOT 文エントリ
A395～A3B1	多項式の評価
A399～	右カッコの処理
A39C～	左カッコの処理
A3B2～A3CA	減算(－)エントリ
A3CB～A617	数値式の評価
A4B3～	べき乗($\wedge$)
A501～	論理演算
A50C～	¥, MOD
A51E～	除算(/)
A5BE～	>, =, <
A5FB～	AND
A600～	OR
A605～	XOR
A60A～	EQV
A60F～	IMP
A618～A620	DIM 文エントリ
A621～A661	変数名テーブル登録
A662～A8AC	変数サーチルーチン
A69A～	単純変数テーブル検索
A6BF～	単純変数テーブル登録
A74A～	

アドレス	内 容
A757～	配列変数テーブル検索
A7DF～	配列変数テーブル登録
A83D～	配列変数テーブルアドレス計算
A8AD～A920	FRE 関数エントリ
A921～A9EA	STR\$関数エントリ
A9EB～AA8A	ガベージコレクション処理
AA8B～AB05	文字列代入処理
AB06～AB0E	LEN 関数エントリ
AB0F～AB22	CHR\$ 関数エントリ
AB23～AB2D	ASC 関数エントリ
AB2E～AB4A	LEFT\$ 関数エントリ
AB4B～AB54	RIGHT\$ 関数エントリ
AB55～ABA5	MID\$ 関数エントリ
ABA6～AC1E	VAL 関数エントリ
AC1F～AC28	PEEK 関数エントリ
AC29～AC32	POKE 関数エントリ
AC33～AC48	行番号定数の評価
AC49～AC78	TAB(n)関数エントリ
AC79～ACB6	LPRINT 文エントリ
ACB7～AE30	PRINT 文エントリ
ACBD～	WRITE 文エントリ
ADB4～	改行処理
ADDA～	PRINT 文中のカンマ(,)の処理
ADEF～	PRINT 文中のセミコロン(;)の処理
AE31～AE68	SPC(n)関数エントリ
AE69～AEBC	文字列出力
AEBD～AF92	PRINT@文エントリ
AF93～B03D	INSTR 関数エントリ
B03E～B05F	STRING\$ 関数エントリ
B060～B07C	SPACE\$ 関数エントリ
B07D～B0E7	MID\$ 関数エントリ
B0E8～B0E8	HEX\$ 関数エントリ
B0E9～B139	OCT\$ 関数エントリ
B13A～B14F	& 定数エントリ
B150～B160	&O 定数エントリ
B161～B187	&H 定数エントリ
B188～B238	LIST 出力処理
B239～B23B	DEFSTR 文エントリ
B23C～B23E	DEFINT 文エントリ
B23F～B241	DEFSNG 文エントリ

アドレス	内 容
B242～B281	DEFDBL 文エントリ
B282～B2A1	DELETE 文エントリ
B2A2～B2F0	AUTO 文エントリ
B2F1～B2F5	TRON 文エントリ
B2F2～	TROFF 文エントリ
B2F6～B4D8	PRINT USING 文(/)の処理
B32F～	PRINT USING 文(&)の処理
B356～	PRINT USING 文エントリ
B4D9～B757	FOR 文エントリ
B5AE～	NEXT 文エントリ
B659～	WEND 文のサーチ
B65A～	NEXT 文のサーチ
B758～B7DD	テキストポインタのアドレス変換ルーチン
B7DE～B807	ON ERROR GOTO 文エントリ
B808～B812	ERROR 文エントリ
B813～B856	RESUME 文エントリ
B857～B8AC	SWAP 文エントリ
B8AD～B8B8	DEF 文エントリ
B8B9～B90C	DEF FN 文エントリ
B90D～BA82	FN 関数エントリ
BA83～BA99	VARPTR 関数エントリ
BA9A～BAD0	DEF USR 文エントリ
BAD1～BAFE	USR 関数エントリ
BAFF～BBA4	WHILE 文エントリ
BB39～	WEND 文エントリ
BBA5～DD33	ANPORT 文エントリ
BBB4～	実数減算ルーチン
BBBD～	実数加算ルーチン
BC3A～	実数の正規化処理
BCAD～	実数の丸め処理
BD0B～	実数の切り捨て処理
BD34～BD50	LOG 演算用データ
BD51～C079	LOG 関数エントリ
BDAD～	実数乗算ルーチン
BE9E～	指数部演算ルーチン
BEF0～	実数除算ルーチン
C07A～C08E	SGN 関数エントリ
C08F～C11A	ABS 関数エントリ
C11B～C12B	FIX 関数エントリ
C12C～C2DC	INT 関数エントリ

アドレス	内 容
C2DD～C7BF	「IN(行番号)」の出力
C7C0～C8FC	SQR 関数エントリ
C8FD～C921	EXP 演算用データ
C922～C997	EXP 関数エントリ
C998～C9EA	多項式演算処理
C9EB～CA03	RANDOMIZE 文エントリ
CA04～CA51	RND 関数エントリ
CA52～CA79	RND 演算用データ
CA7A～CAD7	CINT 関数エントリ
CA82～	単精度データの整数化
CA8B～	倍精度データの整数化
CACC～	型変換ルーチン
CAD8～CAEA	CDBL 関数エントリ
CADE～	整数データの倍精度化
CAE0～	単精度データの倍精度化
CAEB～CB33	CSNG 関数エントリ
CAF9～	倍精度データの単精度化
CB0A～	整数データの単精度化
CB34～CB49	整数/実数のチェック処理
CB4A～CB57	減算(－)エントリ
CB58～CB7D	加算(＋)エントリ
CB7E～CC16	乗算(×)エントリ
CC17～CC3D	整数除算(÷)エントリ
CC3E～CC54	MOD エントリ
CC55～CC6A	数値比較(<, =, >)エントリ
CC6B～CC9D	COS 関数エントリ
CC9E～CD0B	SIN 関数エントリ
CD0C～CE2E	TAN 関数エントリ
CE2F～CE9E	三角関数演算用データ
CE9F～CEFC	ATN 関数エントリ
CEFD～D01F	テキスト逆翻訳ルーチン
D020～D36F	テキスト翻訳ルーチン
D221～	小文字→大文字変換
D2B9～	行番号定数の評価
D370～D3D2	標準関数の分岐処理
D3D3～D4C5	BASIM メインループ
D41F～	ダイレクトモード エントリ
D425～	各コマンドへの分岐処理
D4B0～	暗黙 END 処理
D4C6～D526	省略形キーワードテーブル

アドレス	内 容
D527～D535	テキスト読み込みルーチン
D536～D541	EXEC 文エントリ
D542～D5A7	CLEAR 文エントリ
D5A8～D663	CONSOLE 文エントリ
D628～	コンソールイニシャライズ処理
D65E～	
D664～D6D6	LOCATE 文エントリ
D6D7～D6E9	UNLIST 文エントリ
D6EA～D6F0	CSRLIN 文エントリ
D6F1～D70D	POS 関数エントリ
D70E～D722	LPOS 関数エントリ
D723～D732	行番号評価, サーチルーチン
D733～D744	カセット処理ジャンプテーブル
D745～D74B	カセット EOF 処理
D74C～D74E	カセット LOF 処理
D74F～D7C0	カセットオープン処理
D761～	*O*モードのオープン処理
D7A7～	*I*モードのオープン処理
D7C1～D7E3	カセット I ブロック入力処理
D7E4～D80D	カセットクローズ処理
D80E～D80F	LOAD ? 文エントリ
D810～D8D8	SKIPF 文エントリ
D829～	カセット I バイト入力(バッファから)
D843～	カセット I バイト出力(バッファへ)
D859～	カセット I ブロック出力処理
D875～	ヘッダブロック読み込み処理
D8D9～DA1C	FILE 文エントリ
DA1D～DB9A	LIST 文エントリ
DA5A～	SAVE 文エントリ
DA74～	プロテクト暗号化処理
DAB9～	LLIST 文エントリ
DADD～	プロテクト復号化処理
DB04～	プロテクトチェック
DB9B～DC4E	SAVEM 文エントリ
DC4F～DC83	CLOSE 文エントリ
DC84～DCF5	OPEN 文エントリ
DCB7～	*I*モードのオープン処理
DCBA～	*O*モードのオープン処理
DCF6～DE6F	RUN 文エントリ
DCFD～	MERGE 文エントリ

ア ド レ ス	内 容
DD05～	LOAD 文エントリ
DE61～	1 バイト入力処理
DE70～DFA8	LOADM 文エントリ
DE9D～	2 バイト入力処理
DF09～	1 バイト入カルーチン
DF23～	1 バイト出カルーチン
DFA9～DFAB	EOF 関数エントリ
DFAC～DFEF	LOF 関数エントリ
DFF0～E02B	INPUT\$関数エントリ
E02C～E040	COM クローズ処理
E041～E0F1	COM オープン処理
E0F2～E154	COM1 バイト出力処理
E158～E1A0	COM1 バイト入力処理
E1AA～E1B8	COM EOF 処理
E1B9～E1C2	COM LOF 処理
E1C3～E2A9	BASIC プログラム割込制御処理
E210～	ON COM 文エントリ
E241～	COM 文エントリ
E245～	COMMON 文エントリ
E261～	COM OFF 処理
E272～	COM ON 処理
E281～	COM STOP 処理
E287～	割込みイニシャライズ
E2AA～E307	TERM モードの PF キー割込処理
E308～E315	TERM モードの PF キー割込ジャンプテーブル
E316～E323	TERM モードのオプション転送処理
E324～E47D	TERM 文エントリ
E47E～E482	TERM オペランド既定値データ
E483～E489	プリンタクローズ処理
E48A～E4B0	改行処理
E4B1～E4C7	プリンタのレディチェック
E4C8～E592	Abort 処理
E593～E657	一行入力処理(スクリーンエディット時)
E658～E736	EDIT 文エントリ
E69E～	CRT 1 ブロック出力処理
E6BE～	フィールド設定処理
E6CB～	CRT イレーズ処理
E6D2～	行番号出力
E6DC～	AUTO 編集処理
E737～E754	プリンタオープン処理

アドレス	内 容
E755～E75A	CRT オープン処理
E75B～E75B	CRT クローズ処理
E75C～E805	CRT 1バイト出力処理
E764～	カーソル X, Y 座標の読み取り
E77A～	サブシステム関係 BIOS の RCB 設定
E7F3～	カーソル X, Y 座標設定
E806～E82E	CLS 文エントリ
E82F～E867	ファイルからの 1 行入力処理
E868～E879	キーボード処理ジャンプテーブル
E87A～E87F	キーボードオープン処理
E880～E88D	キーボードクローズ処理
E88E～E8A4	キーバッファカウンタ読み込み
E8A5～E8D4	キーボード 1 文字入力処理
E8AA～	Break チェックルーチン
E8B9～	INKEY ルーチン
E8C5～	INKEY\$ 文エントリ
E8D5～E8DA	KEY 文エントリ
E8DB～E98C	KEY LIST 文エントリ
E94E～	コンソール設定ルーチン
E973～	PF キー読み込み/表示処理
E98D～E9E8	KEY(n) ON/OFF/STOP 処理
E9AD～	KEY(n) ON 文エントリ
E9BB～	KEY(n) OFF 文エントリ
E9C7～	KEY(n) STOP 文エントリ
E9CE～	PF キー割込み設定ルーチン
E9E9～EA1D	PF キー定義ルーチン
EA1E～EA21	ON 文エントリ
EA22～EA46	ON KEY 文エントリ
EA47～EA52	ON INTERVAL 文エントリ
EA53～EA66	ON TIME 文エントリ
EA67～EA75	ON PEN 文エントリ
EA76～EA7B	BUBINI 文エントリ
EA7C～EA97	KILL 文エントリ
EA98～EC05	テキスト削除
EBB1～	テキストポインタアドレス変換ルーチン
EC06～EC0B	FM 音源音色初期設定処理
EC0C～EC73	0 時割込み処理ルーチン
EC74～EC8E	うるう年チェック
EC8F～	ブレーク割込み処理
ECAB～	アテンション割込み処理

アドレス	内 容
ED09～	拡張 IRQ 処理
EDDD～	MIDI に I 文字出力
F03E～	BIOS RCB インターフェース エントリ
F041～	BIOS レジスタインターフェース エントリ
F044～	BIOS のバージョン
FB9D～	BASIC のバージョン

E. OPNBIOSメモリマップ

アドレス	内 容
\$EDF5～\$EE0D	TIMER-A, TIMER-B 割り込み処理
\$EE0E～\$EE27	メインルーチン
\$EE28～\$EE5B	ジャンプテーブル
\$EE5C～\$EE67	ステータスレジスタ読み込み
\$EE68～\$EE8C	SSG レジスタ 1 バイトデータ書き込み
\$EE8D～\$EEA3	SSG レジスタ 2 バイトデータ書き込み
\$EEA4～\$EEC3	SSG レジスタ 1 バイトデータ読み込み
\$EEC4～\$EED2	SSG レジスタクリア
\$EED3～\$EEE2	SSG 周波数データ書き込み
\$EEE3～\$EEF6	SSG 音量データ書き込み
\$EEF7～\$EF39	FM 音源レジスタ 1 バイトデータ書き込み
\$EF3A～\$EF52	I チャンネルの KEY ON/OFF 設定
\$EF53～\$EF61	全チャンネルのスロット KEY OFF
\$EF62～\$EF9C	音色データ書き込み
\$EF9D～\$EFB7	音階データ書き込み
\$EFB8～\$EFE7	トータルレベル書き込み
\$EFE8～\$F020	音色データ転送
\$F021～\$F03E	キャリア判定

F. BIOS(V3.3)メモリマップ

アドレス	内 容
\$F03E～\$F040	RCB インターフェース エントリ
\$F041～\$F043	レジスタインターフェース エントリ
\$F044～\$F04A	BIOS のバージョン, リリース年月日
\$F04B～\$F085	割り込み処理
\$F086～\$F0AD	RCB インターフェース メインルーチン
\$F0AE～\$F0C6	モードセレクトレジスタ(\$FD93)セットサブルーチン
\$F0C7～\$F112	RCB インターフェース エントリアドレステーブル
\$F113～\$F14A	レジスタインターフェース メインルーチン
\$F14B～\$F15C	レジスタインターフェース エントリアドレステーブル
\$F15D～\$F17C	ACHROT 処理ルーチン
\$F17D～\$F1A7	AKEYIN 処理ルーチン
\$F1A8～\$F1AE	APRTOT 処理ルーチン
\$F1AF～\$F1BB	PRTCHK 処理ルーチン
\$F1BC～\$F1C8	MOTOR 処理ルーチン
\$F1C9～\$F2B5	CTBWRT 処理ルーチン
\$F2B6～\$F393	CTBRED 処理ルーチン
\$F394～	BEEPON 処理ルーチン
\$F397～\$F39C	BEEPOFF 処理ルーチン
\$F39D～\$F3E8	LPOUT 処理ルーチン
\$F3E9～\$F3FE	LPCHK 処理ルーチン
\$F3FF～\$F42E	KANJIR 処理ルーチン
\$F42F～\$F480	漢字 ROM アドレス計算サブルーチン
\$F481～\$F4E6	SUBIN, SUBOUT 処理ルーチン
\$F4E7～\$F5A8	INPUT 処理ルーチン
\$F5A9～\$F5C5	INPUTC 処理ルーチン
\$F5C6～\$F622	OUTPUT 処理ルーチン
\$F623～\$F695	KEYIN 処理ルーチン
\$F696～	DWRITE 処理ルーチン
\$F699～\$F79B	DREAD 処理ルーチン
\$F79C～\$F841	RESTORE, SEEK5 処理ルーチン
\$F842～\$F919	SETMOD 処理ルーチン
\$F91A～\$F9A6	BIINIT 処理ルーチン

G. サブモニターROM1メモリマップ

アドレス	内 容
\$D800～\$D835	ジャンプテーブル
\$D836～\$D9C8	直線のクリッピング処理サブルーチン
\$D9C9～\$DA04	$(4 \text{ バイト}) = (2 \text{ バイト}) \times (2 \text{ バイト})$ 乗算サブルーチン
\$DA05～\$DA34	$(2 \text{ バイト}) = (4 \text{ バイト}) \div (2 \text{ バイト})$ 除算サブルーチン
\$DA35～\$DA86	BOX FULL のクリッピング処理サブルーチン
\$DA87～\$DAA7	4 ビットエリアコード算出サブルーチン
\$DAA8～\$DAC0	ラインスタイルの ROTATE 処理サブルーチン
\$DAC1～\$DC6E	GRAPHIC CURSOR コマンド処理
\$DC6F～\$DD2C	SET VIEWPORT COORDINATE コマンド処理
\$DD2D～\$DD41	ビューポート座標 初期設定サブルーチン
\$DD42～\$DDDB	TILE BOX コマンド処理
\$DDDC～\$DE1F	INKEY コマンド処理
\$DE20～\$DE5F	DEFINE STRING OF PF コマンド処理
\$DE60～\$DE7B	GET STRING OF PF コマンド処理
\$DE7C～\$DE9A	INTERRUPT CONTROL コマンド処理
\$DE9B～\$DEC2	SET TIMER コマンド処理
\$DEC3～\$DEE0	READ TIMER コマンド処理
\$DEE1～\$DFA3	SPECIAL コマンド処理

H. サブモニターROM2メモリマップ

アドレス	内 容
\$D800～\$D820	ジャンプテーブル
\$D821～\$D8B2	CHANGE COLOR コマンド処理
\$D8B3～\$D9D3	GET BLOCK1 コマンド処理
\$D9D4～\$DA29	GET BLOCK2 コマンド処理
\$DA2A～\$DB58	PUT BLOCK1 コマンド処理
\$DB59～\$DC00	PUT BLOCK2 コマンド処理
\$DC01～\$DDB3	GRAPHIC CURSOR コマンド処理
\$DDB4～\$DE73	SET RTC コマンド処理
\$DE74～\$DE96	READ RTC コマンド処理
\$DE97～\$DEA9	DIGITIZE コマンド処理
\$DEAA～\$DF11	TELEVISION CONTROL コマンド処理
\$DF12～\$DF7A	KEYBOARD CONTROL コマンド処理
\$DF7B～\$DF88	キーボードエンコードへデータ送信サブルーチン
\$DF89～\$DF93	キーボードエンコードからデータ受信サブルーチン
\$DF94～\$DF9D	KEYBOARD CONTROL コマンド パラメータチェック
\$DF9E～\$DFA6	RTC の年データ作成サブルーチン

I. サブシステム変数一覧

項目名	内 容	タイプA タイプB	タイプC	大きさ (バイト)
ISYSSP	スタックの初期設定値	\$D000	—	2
..ABRT	アボートフラグ(\$FF: IRQ 発生 \$00: 通常)	\$D002	\$D000	1
..CONT	共有 RAM 継続フラグ保存用ワーク	\$D003	\$D001	1
.CURSR	カーソル表示フラグ	\$D004	\$D01C	1
.CURTM	カーソルリバース表示用カウンタ	\$D005	\$D01D	1
.CURIT	カーソルインターバルカウンタ	\$D006	—	1
.CURST	カーソル状態フラグ(\$FF: オン \$00: オフ)	\$D007	\$D01E	1
.OFST	VARAM オフセット値(アクティブページ用)	\$D008	\$D01F	2
.CSCTL	コンソール制御フラグ	\$D00A	\$D021	1
PUTCFG	PUT 継続フラグ	\$D00B	\$D022	1
CSLCDS	単色表示モードフラグ	\$D00C	\$D023	1
.TABSET	TAB 位置設定テーブル	\$D00D	\$D024	10
..ACL	現在のアトリビュートコード	\$D017	\$D02E	1
..CLEX	キャラクタ表示時のベースイメージ	\$D018	\$D030	1
..CWD	キャラクタの横ドット数	\$D019	\$D031	1
..CHT	キャラクタのアンダーマージン数	\$D01A	\$D032	1
..INS	インサートモードフラグ(\$FF: ON \$00: OFF)	\$D01B	\$D033	1
.EKEY	GET コマンドの終了キーコード	\$D01C	\$D034	1
..LNCD	1 行の文字数	\$D01D	\$D035	2
..DNL	画面の行数(20 or 25)	\$D01F	\$D037	1
..FS	PF キー表示フラグ(\$FF: 表示 \$00: 表示しない)	\$D020	\$D038	1
..VCU	1 キャラクタの VRAM 上のアドレス差	\$D021	\$D039	2
..VLU	1 行の VRAM 上のアドレス差	\$D023	\$D03B	2
..DMAP	画面制御情報(各画面の開始行, 終了行)	\$D025	\$D03D	8
.BPY	バッファポインタの Y 座標	\$D02D	\$D045	1
.BPX	バッファポインタの X 座標	\$D02E	\$D046	1
.BPAD	キャラクタバッファ上のバッファポインタアドレス	\$D02F	\$D047	2
.BAAD	アトリビュートバッファ上のバッファポインタアドレス	\$D031	\$D049	2
.BVAD	VRAM 上のバッファポインタアドレス	\$D033	\$D04B	2
..INIT	イニシャルモードフラグ(\$00: 初期化)	\$D035	—	1
.MDRG	カーソルのあるウィンドウ画面情報	\$D036	\$D04D	3
GTTOPC	キャラクタバッファ上のカーソルのあるウィンドウ先頭アドレス	\$D039	\$D050	2
GTBTMC	キャラクタバッファ上のカーソルのあるウィンドウ終了アドレス	\$D03B	\$D052	2

項目名	内 容	タイプA タイプB	タイプC	大きさ (バイト)
GTTOPA	アトリビュートバッファ上のカーソルのあるウィンドウ先頭アドレス	\$D03D	\$D054	2
GTBTMA	アトリビュートバッファ上のカーソルのあるウィンドウ終了アドレス	\$D03F	\$D056	2
.CY	カーソル Y 座標	\$D04I	\$D058	1
.CX	カーソル X 座標	\$D042	\$D059	1
.CBAD	キャラクタバッファ上のカーソルアドレス	\$D043	\$D05A	2
.CAAD	アトリビュートバッファ上のカーソルアドレス	\$D045	\$D05C	2
.CVAD	VRAM 上のカーソルアドレス	\$D047	\$D05E	2
.CVTOP	消去エリアの開始行, 終了行	\$D049	\$D07D	2
EXFONT	キャラクタフォント拡張データ保存用ワーク	\$D04B	\$D083	18
.FNFTFG	キャラクタフォント選択(\$00:カタカナ \$FF:ひらがな)	\$D05D	—	1
.BAKVO	VRAM オフセット値(インアクティブ ページ用)	\$D05E	—	2
SBKCTL	バンクレジスタ保存用ワーク	\$D060	—	1
HKFLG	キャラクタ表示時の表示色, 背景色	\$D06I	\$D095	2
HKBLUE	キャラクタ表示時のフックアドレス(青)	\$D063	\$D097	3
HKRED	キャラクタ表示時のフックアドレス(赤)	\$D066	\$D09A	3
HKGREE	キャラクタ表示時のフックアドレス(緑)	\$D069	\$D09D	3
C..FNT	キャラクタフォント ロードバッファ	\$D06C	—	10
..BCL	グラフィック背景色	\$D076	\$D02F	3
VIEWX1	ビューポート左端(X座標)	\$D079	—	2
VIEWY1	ビューポート上端(Y座標)	\$D07B	—	2
VIEWX2	ビューポート右端(X座標)	\$D07D	—	2
VIEWY2	ビューポート下端(Y座標)	\$D07F	—	2
.LXI	LINE の開始 X 座標	\$D08I	\$D07I	2
.LYI	LINE の開始 Y 座標	\$D083	\$D073	2
.LX2	LINE の終了 X 座標	\$D085	\$D075	2
.LY2	LINE の終了 Y 座標	\$D087	\$D077	2
.LSTYL	ラインスタイル	\$D089	—	2
.GCL	グラフィック カラーコード	\$D08B	\$D060	1
.BOXX1	BOX の開始 X 座標	\$D08C	\$D069	2
.BOXY1	BOX の開始 Y 座標	\$D08E	\$D06B	2
.BOXX2	BOX の終了 X 座標	\$D090	\$D06D	2
.BOXY2	BOX の終了 Y 座標	\$D092	\$D06F	2
PASTTP	ペイント用ワークエリア開始アドレス	\$D094	—	2
KLOCK	キー入力禁止フラグ(\$FF:禁止 \$00:許可)	\$D096	\$D002	1
KBFFLG	キーバッファフラグ(\$FF:バッファモード \$00:ノンバッファモード)	\$D097	\$D003	1
KCOUNT	キー入力数	\$D098	\$D004	1
KHEAD	キー入力バッファ読み取り用ポインタ	\$D099	\$D005	2

項目名	内 容	タイプA タイプB	タイプC	大きさ (バイト)
KTAIL	キー入力バッファ書き込み用ポインタ	\$D09B	\$D007	2
.PFIRQ	PF キー割り込み番号	\$D09D	\$D009	1
.PFINT	PF キー割り込み制御フラグ	\$D09E	—	2
INDLP	PF キー表示ルーチン ループカウンタ	\$D0A0	\$D06A	1
INDPF	PF キー表示文字数	\$D0A1	\$D06C	1
INDBSV	PF キー表示時の背景色	\$D0A2	\$D06D	3
.TMAC	タイマーアクセスフラグ	\$D0A5	\$D00A	1
.\$TC	タイマー制御レジスタ	\$D0A6	\$D00B	1
.\$T11	割り込み時刻レジスタ	\$D0AB	\$D010	4
.\$T2	タイマーカウンタ	\$D0AF	\$D014	4
.\$T2D	タイマーカウンタ再設定値レジスタ	\$D0B3	\$D018	4

J. エラーメッセージ一覧

(1) F-BASIC 関係

エラー コード	エラーメッセージ	内 容
01	NEXT without FOR	NEXT に対する FOR 文がない。
02	Syntax error	コマンドまたは、文の書き方に誤りがある。
03	RETURN without GOSUB	GOSUB によって呼び出されていないのに RETURN 文に出会った。
04	Out of data	READ 文によって読み込むべきデータがない。
05	Illegal function call	関数やステートメントの呼び方に誤りがある。
06	Overflow	整数値または実数値が、許される範囲を越えている。 または代入される数値が大きすぎる。
07	Out of memory	メモリが足りなくなった。
08	Undefined line number	指定された行番号が定義されていない。
09	Subscript out of range	配列の添字が指定された範囲内でない。
10	Duplicate definition	同じ名前の配列または、ユーザ関数を 2 度宣言した。
11	Division by zero	除算の分母が 0 である。
12	Illegal direct	直接モードで使えないステートメントを用いた。
13	Type mismatch	変数または定数の型が合わない。
14	Out of string space	文字領域が足りなくなった。
15	String too long	文字定数または文字変数が 256 文字を越えた。
16	String Formula Too complex	文字式が複雑すぎる。
17	Can't continue	CONT コマンドによるプログラムの続行ができない。
18	Undefined user function	定義されていない関数を参照している。
19	No RESUME	エラー処理ルーチンに RESUME がない。
20	RESUME without error	エラーが起きていないのに RESUME 文を実行しようとした。
21	Unprintable error.	エラーメッセージの定義されていないエラーを出そうとした。
22	Missing operand	必要なオペランドが抜けている。
23	FOR without NEXT	FOR～NEXT の対応が正しくない。
24	WHILE without WEND	WHILE 文に対応する WEND 文がない。
25	WEND without WHILE	WEND 文に対応する WHILE 文がない。
28	Not supported	用意されていない命令を使おうとした。
50	Bad file number	オープンされていないファイル番号を使用した。
51	Bad file mode	ファイルをオープンしたモードと指定したモードが違っている。
52	File already open	ファイルを二重にオープンしようとした。
53	Device I/O error	使用したデバイスに入出力エラーが発生した。
54	Input past end	ファイルのすべてのデータを読んだ後に、INPUT を実行した。
55	Bad file descriptor	ファイルディスクリプタの記述に誤りがある。
56	Direct statement in file	アスキー形式のプログラムファイル中に、直接ステートメントがあった。

エラーコード	エラーメッセージ	内 容
57	File not open	ファイルがオープンされていない。
58	Bad data in file	ファイル上のデータの形式が正しくない。
59	Device in use	使用中のデバイスに対して、再度オープンしようとした。
60	Device unavailable	I/O デバイスが入出力可能な状態にない。
61	Buffer overflow	入出力バッファがオーバーフローした。
62	Protected program	保護されているプログラムに、書き込み修正を行なおうとした。
63	File not found	指定されたファイルが見つからない。
64	File, already exists	指定されたファイル名はすでに存在している。
65	Directory full	ディレクトリ領域がいっぱいである。
66	Too many open disk files	確保されているファイルの個数を越えてオープンしようとした。
67	Disk full	ディスクがいっぱいである。
68	Field overflow	フィールドの長さが 256 バイトを越えている。
69	String not fielded	FIELD 文で宣言された文字変数以外の変数に LSET, RSET を用いて代入しようとした。
70	Bad record number	指定されたレコード番号は存在しない。
71	Bad file structure	ファイルの構成に誤りがある。
72	Drive not ready	指定されたドライブは Ready 状態にない。
73	Disk write protected	Disk が書き込み保護されている。

(2) BIOS 関係

a) BIOS システムエラー

エラー番号	エ ラ ー 内 容
1	RCB エラー
2	Device Unavailable エラー
3	ブレーク

b) フロッピーディスク関係エラー

エラー番号	エ ラ ー 内 容
10	ドライブノットレディ
11	ディスクライトプロテクテッド
12	ハードエラー(シークエラー, ロストデータ, レコードノットファウンド)
13	CRC エラー
14	DD マーク検出(DD マーク=Deleted Data mark)
15	タイムオーバーエラー

C) プリンタ、カセット関係エラー

エラー番号	エ ラ ー 内 容
50	ペーパーエンブティ
51	プリンタノットレディ
52	オーディオカセットリードエラー

d) サブシステムエラー

エラー番号	エ ラ ー 内 容
60	INIT コマンドパラメータエラー
61	コンソール座標エラー
62	複数バイトのオーダシーケンスにおいて必要なデータがない。
63	グラフィック座標エラー
64	使用できないファンクションコードまたは、未定義ファンクションコードを使用した。
65	座標数が規定数より多い、または少ない。
66	文字数より多い、または少ない。
67	色指定数が規定数より多い、または少ない。
68	ファンクションキー番号エラー
69	パラメータエラー
70	コマンドエラー

(2) サブシステム関係

エラーコード(16進)	内 容
3C	INIT コマンドのパラメータに誤りがあります。
3D	コンソール座標値に誤りがあります。
3E	オーダシーケンスに誤りがあります。
3F	グラフィック座標値に誤りがあります。
40	ファンクションコードに誤りがあります。
41	座標数に誤りがあります。
42	文字数に誤りがあります。
43	色数に誤りがあります。
44	PF キー番号に誤りがあります。
45	コマンド、パラメータに誤りがあります。
46	コマンドコードに誤りがあります。

K. キャラクタコード表

キャラクタコード表(かなモード)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	(SP)	0	@	P	'	p	—	⊥		—	タ	ミ	二	×
1	SH	DI	!	1	A	Q	a	q	—	⊥	。	ア	チ	ム	ト	円
2	SX	D2	"	2	B	R	b	r	—	⊥	「	イ	ツ	メ	士	年
3	EX	D3	#	3	C	S	c	s	—	⊥	」	ウ	テ	モ	コ	月
4	ET	D4	\$	4	D	T	d	t	—	—	、	エ	ト	ヤ	▲	日
5	EQ	NK	%	5	E	U	e	u	—	—	・	オ	ナ	ユ	▲	時
6	AK	SL	&	6	F	V	f	v	—		ヲ	カ	ニ	ヨ	▼	分
7	BL	EB	'	7	G	W	g	w	—		ァ	キ	ヌ	ラ	▼	秒
8	BS	CN	(	8	H	X	h	x		ㇿ	イ	ク	ネ	リ	♠	千
9	HT	EM	)	9	I	Y	i	y		ㇿ	ウ	ケ	ノ	ル	♥	市
A	LF	SB	*	:	J	Z	j	z		ㇿ	エ	コ	ハ	レ	♦	区
B	HM	EC	+	;	K	[	k	]		ㇿ	オ	サ	ヒ	ロ	♣	町
C	CL	→	,	<	L	¥	l			ㇿ	ヤ	シ	フ	ワ	●	村
D	CR	←	-	=	M	]	m			ㇿ	ユ	ス	ヘ	ン	○	人
E	SO	↑	.	>	N	^	n	—		ㇿ	ヨ	セ	ホ	ゝ	/	■
F	SI	↓	/	?	O	—	o	DL	+	ㇿ	ツ	ソ	マ	°	\	

キャラクタコード表(ひらがなモード)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	(SP)	0	@	P	'	p	—	⊥		—	た	み	二	×
1	SH	DI	!	1	A	Q	a	q	—	⊥	。	あ	ち	む	ト	円
2	SX	D2	"	2	B	R	b	r	—	⊥	「	い	つ	め	士	年
3	EX	D3	#	3	C	S	c	s	—	⊥	」	う	て	も	コ	月
4	ET	D4	\$	4	D	T	d	t	—	—	、	え	と	や	▲	日
5	EQ	NK	%	5	E	U	e	u	—	—	・	お	な	ゆ	▲	時
6	AK	SL	&	6	F	V	f	v	—		を	か	に	よ	▼	分
7	BL	EB	'	7	G	W	g	w	—		あ	き	ぬ	ら	▼	秒
8	BS	CN	(	8	H	X	h	x		ㇿ	い	く	ね	り	♠	千
9	HT	EM	)	9	I	Y	i	y		ㇿ	う	け	の	る	♥	市
A	LF	SB	*	:	J	Z	j	z		ㇿ	え	こ	は	れ	♦	区
B	HM	EC	+	;	K	[	k	]		ㇿ	お	さ	ひ	ろ	♣	町
C	CL	→	,	<	L	¥	l			ㇿ	や	し	ふ	わ	●	村
D	CR	←	-	=	M	]	m			ㇿ	ゆ	す	へ	ん	○	人
E	SO	↑	.	>	N	^	n	—		ㇿ	よ	せ	ほ	ゝ	/	■
F	SI	↓	/	?	O	—	o	DL	+	ㇿ	っ	そ	ま	°	\	

L. キー配列とキーコード

(1) FM-7互換モード(9ビットコード)

① 英数モード

BREAK	101	102	103	104	105	106	AVTV	AVTV	AVTV	AVTV	05	0C	12	19	7F
							107	108	109	10A	05	0C	12	1E	7F
											11	0B	02	1A	06
											11	0B	1D	1F	1C

1B	21	22	23	24	25	26	27	28	29		3D	7E	7C	08	
1B	31	32	33	34	35	36	37	38	39	30	2D	5E	5C	08	
09	51	57	45	52	54	59	55	49	4F	50	60	7B			
09	71	77	65	72	74	79	75	69	6F	70	40	5B		0D	
	41	53	44	46	47	48	4A	4B	4C	2B	2A	7D		0D	
CTRL	61	73	64	66	67	68	6A	6B	6C	3B	3A	5D			
	5A	58	43	56	42	4E	4D	3C	3E	3F	5F				
SHIFT	7A	78	63	76	62	6E	6D	2C	2E	2F	22		SHIFT		
	CAP	GRAPH	20	20		20							かな		
			20	20		20									

2A	2F	2B	2D												
2A	2F	2B	2D												
37	38	39	3D												
37	38	39	3D												
34	35	36	2C												
34	35	36	2C												
31	32	33													
31	32	33												0D	
	30	2E	0D												
	30	2E													

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード
 空白については未定義のためコードを出力しません。
 SHIFT時の[PF7]～[PF10]はAVテレビに使用

② CAPモード

BREAK	101	102	103	104	105	106	AVTV	AVTV	AVTV	AVTV	05	0C	12	19	7F
							107	108	109	10A	05	0C	12	1E	7F
											11	0B	02	1A	06
											11	0B	1D	1F	1C

1B	21	22	23	24	25	26	27	28	29		3D	7E	7C	08	
1B	31	32	33	34	35	36	37	38	39	30	2D	5E	5C	08	
09	71	77	65	72	74	79	75	69	6F	70	60	7B			
09	51	57	45	52	54	59	55	49	4F	50	40	5B		0D	
	61	73	64	66	67	68	6A	6B	6C	2B	2A	7D		0D	
CTRL	41	53	44	46	47	48	4A	4B	4C	3B	3A	5D			
	7A	78	63	76	62	6E	6D	3C	3E	3F	5F				
SHIFT	5A	58	43	56	42	4E	4D	2C	2E	2F	22		SHIFT		
	CAP	GRAPH	20	20		20							かな		
			20	20		20									

2A	2F	2B	2D												
2A	2F	2B	2D												
37	38	39	3D												
37	38	39	3D												
34	35	36	2C												
34	35	36	2C												
31	32	33													
31	32	33												0D	
	30	2E	0D												
	30	2E													

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード
 空白については未定義のためコードを出力しません。
 SHIFT時の[PF7]～[PF10]はAVテレビに使用

③ かなモード

BREAK						106	AVTV 107	AVTV 108	AVTV 109	AVTV 10A	05	0C
	101	102	103	104	105						05	0C
											11	0B
											11	0B

12	19	7F
12	1E	7F
02	1A	06
1D	1F	1C

1B			A7	A9	AA	AB	AC	AD	AE	A6				08
1B	C7	CC	B1	B3	B4	B5	D4	D5	D6	DC	CD	CE	B0	08
09			A8									A2		
09	C0	C3	B2	BD	B6	DD	C5	C6	D7	BE	DE	DF		0D
CTRL	C1	C4	BC	CA	B7	B8	CF	C9	D8	DA	B9	D1	A3	0D
SHIFT		AF	BB	BF	CB	BA	D0	D3	A4	A1	A5		DB	SHIFT
		C2							C8	D9	D2			
	CAP	GRAPH	20	20	20	20	20	20	20	20	20	かな		

2A	2F	2B	2D
2A	2F	2B	2D
37	38	39	3D
37	38	39	3D
34	35	36	2C
34	35	36	2C
31	32	33	
31	32	33	0D
30	2E	2E	0D
30	2E	2E	0D

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード (CAPのON/OFFには無関係)
 空白については未定義のためコードを出力しません。
 SHIFT時の[PF7]～[PF10]はAVテレビに使用

④ グラフモード

BREAK						106	AVTV 107	AVTV 108	AVTV 109	AVTV 10A	05	0C
	101	102	103	104	105						05	0C
											11	0B
											11	0B

12	19	7F
12	1E	7F
02	1A	06
1D	1F	1C

1B	F9	FA	FB	FC	F2	F3	F4	F5	F6	F7	8C	8B	F1	08
1B	F9	FA	FB	FC	F2	F3	F4	F5	F6	F7	8C	8B	F1	08
09		FD	F8	E4	E5	9C	9D	F0	E8	E9	8D	8A	ED	
09		FD	F8	E4	E5	9C	9D	F0	E8	E9	8D	8A	ED	0D
CTRL	95	96	E6	E7	9E	9F	EA	EB	8E	89	94	EC		0D
	95	96	E6	E7	9E	9F	EA	EB	8E	89	94	EC		
SHIFT		80	81	82	83	84	85	86	87	88	97	E0		SHIFT
		80	81	82	83	84	85	86	87	88	97	E0		
	CAP	GRAPH	20	20	20	20	20	20	20	20	20	かな		

98	91	99	EE
98	91	99	EE
E1	E2	E3	EF
E1	E2	E3	EF
93	8F	92	
93	8F	92	
9A	90	9B	
9A	90	9B	0D
			0D

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード (CAPのON/OFFは無関係)
 空白については未定義のためコードを出力しません。
 SHIFT時の[PF7]～[PF10]はAVテレビに使用

⑤ コントロールモード

[illegible]

(注)

CTRL + **SHIFT** + **0** はオートリブートOFF
CTRL + **SHIFT** + **1** はオートリブートON
 空白については未定義のためコード出力しません。
 SHIFT時の **PF 7** ~ **PF10** はAVテレビに使用

(2) FM-16 β 準拠モード(9ビットコード)

① 英数モード

① 英数七一下

BREAK

101	102	103	104	105
-----	-----	-----	-----	-----

106	AVTV 107	AVTV 108	AVTV 109	AVTV 10A
-----	-------------	-------------	-------------	-------------

132	131
112	111
133	135
113	115

130	136	134
110	116	114
138	137	139
118	117	119

1B	21	22	23	24	25	26	27	28	29		3D	7E	7C	08
1B	31	32	33	34	35	36	37	38	39	30	2D	5E	5C	08
09	51	57	45	52	54	59	55	49	4F	50	60	7B	0D 0D	
09	71	77	65	72	74	79	75	69	6F	70	40	5B		
CTRL		41	53	44	46	47	48	4A	4B	4C	2B	2A	7D	
		61	73	64	66	67	68	6A	6B	6C	3B	3A	5D	
SHIFT			5A	58	43	56	42	4E	4D	3C	3E	3F	5F	SHIFT
			7A	78	63	76	62	6E	6D	2C	2E	2F	22	
CAP	GRAPH	12E 10E	12D 10D	20 20			かな							

2A	2F	2B	2D	
2A	2F	2B	2D	
37	38	39	3D	
37	38	39	3D	
34	35	36	2C	
34	35	36	2C	
31	32	33	0D 0D	
31	32	33		
30	2E			
30	2E			

(注) 単独押下時：下段コード、**SHIFT**押下時：上段コード
空白については未定義のためコードを出力しません。
SHIFT時の**PF7**～**PF10**はAVテレビに使用

② CAPモード

② CAPモード

BREAK	101	102	103	104	105	106	AVTV	AVTV	AVTV	AVTV	132 112	131 111	130 110	136 116	134 114
											133 113	135 115	138 118	137 117	139 119

1B	21	22	23	24	25	26	27	28	29		3D	7E	7C	08	
1B	31	32	33	34	35	36	37	38	39	30	2D	5E	5C	08	
09	71	77	65	72	74	79	75	69	6F	70	60	7B	0D 0D		
09	51	57	45	52	54	59	55	49	4F	50	40	5B			
CTRL	61	73	64	66	67	68	6A	6B	6C	2B	2A	7D			
	41	53	44	46	47	48	4A	4B	4C	3B	3A	5D			
SHIFT		7A	78	63	76	62	6E	6D	3C	3E	3F	5F	SHIFT		
		5A	58	43	56	42	4E	4D	2C	2E	2F	22			
CAP	GRAPH	12E 10E	12D 10D	20 20				かな							

2A	2F	2B	2D
2A	2F	2B	2D
37	38	39	3D
37	38	39	3D
34	35	36	2C
34	35	36	2C
31	32	33	0D 0D
31	32	33	
30	2E		
30	2E		

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード
空白については未定義のためコードを出力しません。
SHIFT時の[PF7]～[PF10]はAVテレビに使用

③ かなモード

③ かなモード

BREAK	101	102	103	104	105	106	AVTV	AVTV	AVTV	AVTV	10A	132 112	131 111	130 110	136 116	134 114
												133 113	135 115	138 118	137 117	139 119

1B 1B	C7	CC	A7 B1	A9 B3	AA B4	AB B5	AC D4	AD D5	AE D6	A6 DC	CD	CE	B0	08 08
09 09	C0	C3	A8 B2	BD	B6	DD	C5	C6	D7	BE	DE	A2 DF	0D 0D	
CTRL	C1	C4	BC	CA	B7	B8	CF	C9	D8	DA	B9	A3 D1		
SHIFT		AF C2	BB	BF	CB	BA	D0	D3	A4 C8	A1 D9	A5 D2	DB	SHIFT	

CAP	GRAPH	20 20	20 20	20 20		かな	
-----	-------	----------	----------	----------	--	----	--

2A 2A	2F 2F	2B 2B	2D 2D
37 37	38 38	39 39	3D 3D
34 34	35 35	36 36	2C 2C
31 31	32 32	33 33	0D 0D
30 30	2E 2E		

(注) 単独押下時：下段コード，[SHIFT]押下時：上段コード (CAPのON/OFFには無関係)
空白については未定義のためコードを出力しません。
SHIFT時の[PF7]～[PF10]はAVテレビに使用

(3) スキャンコード(MAKE/BREAK CODE)

DC 5C		DD 5D		DE 5E		DF 5F		E0 60		E1 61		E2 62						E3 63		E4 64		E5 65		E6 66		C9 49		CA 4A		C8 48		CD 4D		CB 4B		CF 4F		D0 50		D1 51	
81 01		82 02		83 03		84 04		85 05		86 06		87 07		88 08		89 09		8A 0A		8B 0B		8C 0C		8D 0D		8E 0E		8F 0F		B6 36		B7 37		B8 38		B9 39					
90 10		91 11		92 12		93 13		94 14		95 15		96 16		97 17		98 18		99 19		9A 1A		9B 1B		9C 1C		9D 1D		BA 3A		BB 3B		BC 3C		BD 3D							
D2 52		9E 1E		9F 1F		A0 20		A1 21		A2 22		A3 23		A4 24		A5 25		A6 26		A7 27		A8 28		A9 29				BE 3E		BF 3F		C0 40		C1 41							
D3 53				AA 2A		AB 2B		AC 2C		AD 2D		AE 2E		AF 2F		B0 30		B1 31		B2 32		B3 33		B4 34		D4 54		DA 5A		C2 42		C3 43		C4 44		C5 45					
D5 55		D6 56		D7 57		D8 58		B5 35								DA 5A																									
D5 55		D6 56																																							

(注) 上段はBREAK CODE
下段はMAKE CODE
SHIFT時の [PF7] ~ [PF10] はAVテレビに使用

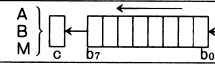
(4) キー配列

					(PC) (SI1) (SI2) (TV)					EL CLS					
BREAK	PF 1	PF 2	PF 3	PF 4	PF 5	PF 6	PF 7	PF 8	PF 9	PF 10	DUP	HOME	INS	↑	DEL
ESC	！ 1ぬ	” 2ふ	#あ 3あ	\$う 4う	%え 5え	&お 6お	・や 7や	(ゆ 8ゆ	)よ 9よ	を 0わ	= 一ほ	- へ	 ¥一	←	
TAB	Qた	Wて	Eい	Rす	Tか	Yん	Uな	Iに	Oら	Pせ	、 @*	)^r [°	↷		
CTRL	Aち	Sと	Dし	Fは	Gき	Hく	Jま	Kの	Lり	+ ；れ	* ：け	{ }む			
SHIFT		っ Zっ	xさ	Cそ	Vひ	Bこ	Nみ	Mも	く、 。ね	。、 る	?・ ／め	- ”ろ	SHIFT		
CAP		GRAPH									かな				

*	/	+	-
7	8	9	=
4	5	6	.
1	2	3	↷
0		.	

(注) 単独押下時：下段の文字、[SHIFT]押下時：上段の文字
SHIFT時の [PF7] ~ [PF10] はAVテレビに使用
[PF7]画面をパソコンモードにします。
[PF8]画面をスーパーインポーズ(高輝度)にします。
[PF9]画面をスーパーインポーズ(ハーフトーン)にします。
[PF10]画面をテレビモードにします。

M. CPU6809命令表

Instruction	Forms	Addressing Modes												Description	5	3	2	1	0				
		Immediate			Direct			Indexed			Extended				Inherent			H	N	Z	V	C	
		Op	~	#	Op	~	#	Op	~	#	Op	~	#		Op	~	#						
ABX														3A	3	I	B+X→X (Unsigned)	●	●	●	●	●	
ADC	ADCA ADCB	89 C9	2 2	2 2	99 D9	4 4	2 2	A9 E9	4+ 4+	2+ 2+	B9 F9	5 5	3 3				A+M+C→A B+M+C→B	↑ ↑	↑ ↑	↑ ↑	↑ ↑	↑ ↑	
ADD	ADDA ADDB ADDD	8B CB C3	2 2 4	2 2 3	9B DB D3	4 4 6	2 2 2	AB EB E3	4+ 4+ 6+	2+ 2+ 2+	BB FB F3	5 5 7	3 3 3				A+M→A B+M→B D+M:M+I→D	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
AND	ANDA ANDB ANDCC	84 C4 IC	2 2 3	2 2 2	94 D4 2	4 4 2	2 2 2	A4 E4	4+ 4+	2+ 2+	B4 F4	5 5	3 3				A^M→A B^M→B CC^IMM→C	● ● ●	↑ ↑ ↑	↑ ↑ ↑	0 0 7	● ● ●	
ASL	ASLA ASLB ASL				08	6	2	68	6+	2+	78	7	3	48 58	2 2	I		8 8 8	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑
ASR	ASRA ASRB ASR				07	6	2	67	6+	2+	77	7	3	47 57	2 2	I		8 8 8	↑ ↑ ↑	↑ ↑ ↑	● ● ●	↑ ↑ ↑	↑ ↑ ↑
BIT	BITA BITB	85 C5	2 2	2 2	95 D5	4 4	2 2	A5 E5	4+ 4+	2+ 2+	B5 F5	5 5	3 3				Bit Test A (M^A) Bit Test B (M^B)	● ●	↑ ↑	↑ ↑	0 0	● ●	
CLR	CLRA CLRB CLR				0F	6	2	6F	6+	2+	7F	7	3	4F 5F	2 2	I	0→A 0→B 0→M	● ● ●	0 0 0	I I I	0 0 0	0 0 0	
CMP	CMPA CMPB CMPD	81 C1 10	2 2 5	2 2 4	91 D1 10	4 4 7	2 2 3	A1 E1 10	4+ 4+ 7+	2+ 2+ 3+	B1 F1 10	5 5 8	3 3 4				Compare M from A Compare M from B Compare M:M+I from D	8 8 ●	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
	CMPS	11 8C	5 5	4 4	11 9C	7 7	3 3	11 AC	7+	3+	11 BC	8 8	4 4				Compare M:M+I from S	●	↑	↑	↑	↑	
	CMPU	11 83	5 5	4 4	11 93	7 7	3 3	11 A3	7+	3+	11 B3	8 8	4 4				Compare M:M+I from U	●	↑	↑	↑	↑	
	CMPX CMPY	8C 10 10 8C	4 5 5 4	3 4 4 3	9C 10 10 9C	6 7 7 3	2 3 3 2	AC 10 10 AC	6+ 7+ 3+ 7+	2+ 3+ 3+ 3+	BC 10 10 BC	7 8 8 4	3 4 4 3				Compare M:M+I from X Compare M:M+I from Y	● ● ●	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
COM	COMA COMB COM				03	6	2	63	6+	2+	73	7	3	43 53	2 2	I	A→A B→B M→M	● ● ●	↑ ↑ ↑	↑ ↑ ↑	0 0 0	I I I	
CWAI		3C	≥20	2													CC^IMM→CC Wait for Interrupt					7	
DAA														19	2	I	Decimal Adjust A	●	↑	↑	0	↑	
DEC	DECA DECB DEC				0A	6	2	6A	6+	2+	7A	7	3	4A 5A	2 2	I	A-1→A B-1→B M-1→M	● ● ●	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	● ● ●	
EOR	EORA EORB	88 C8	2 2	2 2	98 D8	4 4	2 2	A8 E8	4+ 4+	2+ 2+	B8 F8	5 5	3 3				A^M→A B^M→B	● ●	↑ ↑	↑ ↑	0 0	● ●	
EXG	RI, R2	1E	8	2													RI→R2 ²	●	●	●	●	●	
INC	INCA INCB INC				0C	6	2	6C	6+	2+	7C	7	3	4C 5C	2 2	I	A+1→A B+1→B M+1→M	● ● ●	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	● ● ●	
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA ³ →PC	●	●	●	●	●	
JSR					4D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	●	●	●	●	●	
LD	LDA LDB LDD LDS	86 C6 CC 10	2 2 3 4	2 2 3 4	96 D6 DC 10	4 4 5 6	2 2 3 3	A6 E6 EC 10	4+ 4+ 5+ 6+	2+ 2+ 3+ 3+	B6 F6 FC 10	5 5 6 7	3 3 3 4				M→A M→B M:M+I→D M:M+I→S	● ● ● ●	↑ ↑ ↑ ↑	↑ ↑ ↑ ↑	0 0 0 0	● ● ● ●	
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M+I→U	●	↑	↑	0	●	
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M+I→X	●	↑	↑	0	●	
	LDY	10 8E	4 4	4 4	10 9E	6 6	3 3	10 AE	6+ 7+	3+ 3+	10 BE	7 7	4 4				M:M+I→Y	● ●	↑ ↑	↑ ↑	0 0	● ●	
LEA	LEAS LEAU LEAX LEAY							32 33 30 31	4+ 4+ 4+ 4+	2+ 2+ 2+ 2+							EA ³ →S EA ³ →U EA ³ →X EA ³ →Y	● ● ● ●	● ● ↑ ↑	● ● ● ●	● ● ● ●	● ● ● ●	

M : メモリのこと

M+I : メモリでアドレスがIだけ大きいポイントのこと

IMM : メモリ上で命令コード(OP)の次に並ぶ16進値を指す

+ : 加算を表わす

- : 減算を表わす

→ : 数値の転送方向を表わす

^ : 論理積を表わす

v : 論理和を表わす

x : xの否定を表わす

v : 排他的論理和を表わす

R1 : 転送元になるレジスタを表わす

EXG命令では、同時に転送先にもなる

R2 : 転送先になるレジスタを表わす

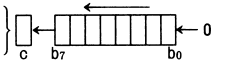
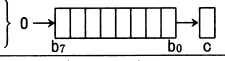
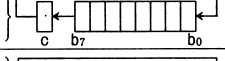
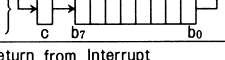
EXG命令では、同時に転送元にもなる
フラグの変化についての記号

● : 変化しない

↑ : 実行結果により変化する

0 : 実行した結果、リセットされる

I : 実行した結果、セットされる

Instruction	Forms	Addressing Modes												Description		5	3	2	1	0			
		Immediate			Direct			Indexed			Extended					Inherent		H	N	Z	V	C	
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#							
LSL	LSLA LSLB LSL				08	6	2	68	6+	2+	78	7	3	48 58	2 2	I I		8 8 8	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
LSR	LSRA LSRB LSR				04	6	2	64	6+	2+	74			44 54	2 2	I I		● ● ●	0 0 0	↑ ↑ ↑	● ● ●	↑ ↑ ↑	
MUL														3D	11	I	A×B→D (Unsigned)		●	●	↑	●	9
NEG	NEGA NEGB NEG				00	6	2	60	6+	2+	70	7	3	40 50	2 2	I I	$\bar{A}+1\rightarrow A$ $\bar{B}+1\rightarrow B$ $\bar{M}+1\rightarrow M$	8 8 8	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
NOP														12	2	I	No Operation		●	●	●	●	●
OR	ORA ORB ORCC	8A CA 1A	2 2 3	2 2 2	9A DA	4 4	2 2	AA EA	4+ 4+	2+ 2+	BA FA	5 5	3 3				A∨M→A B∨M→B CC∨IMM→CC	●	↑ ↑	↑ ↑	0 ↑	● 7	
PSH	PSHS PSHU	34 36	5+ ⁴ 5+ ⁴	2 2													Push Registers on S Stack Push Registers on U Stack		●	●	●	●	●
PUL	PULS PULU	35 37	5+ ⁴ 5+ ⁴	2 2													Pull Registers from S Stack Pull Registers from U Stack		●	●	●	●	●
ROL	ROLA ROLB ROL				09	6	2	69	6+	2+	79	7	3	49 59	2 2	I I		●	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
ROR	RORA RORB ROR				06	6	2	66	6+	2+	76	7	3	46 56	2 2	I I		●	↑ ↑ ↑	↑ ↑ ↑	●	↑ ↑ ↑	
RTI														3B	6 ₁₅	I	Return from Interrupt						7
RTS														39	5	I	Return from Subroutine		●	●	●	●	●
SBC	SBCA SBCB	82 C2	2 2	2 2	92 D2	4 4	2 2	A2 E2	4+ 4+	2+ 2+	B2 F2	5 5	3 3				A-M-C→A B-M-C→B	8 8	↑ ↑	↑ ↑	↑ ↑	↑ ↑	
SEX														1D	2	I	Sign Extend B into A		●	↑	↑	0	●
ST	STA STB STD STS				97 D7 DD 10	4 4 5 6	2 2 2 3	A7 E7 ED 10	4+ 4+ 5+ 6+	2+ 2+ 2+ 3+	B7 F7 FD 10	5 5 6 7	3 3 3 4				A→M B→M D→M:M+1 S→M:M+1	●	↑ ↑ ↑ ↑	↑ ↑ ↑ ↑	0 0 0 0	● ● ● ●	
	STU STX STY				DF 9F 10 9F	5 5 6 6	2 2 3 3	EF AF 10 AF	5+ 5+ 6+ 6+	2+ 2+ 3+ 3+	FF BF 10 BF	6 6 7 7	3 3 4 4				U→M:M+1 X→M:M+1 Y→M:M+1	●	↑ ↑ ↑	↑ ↑ ↑	0 0 0	● ● ●	
SUB	SUBA SUBB SUBD	80 C0 83	2 2 4	2 2 3	90 D0 93	4 4 6	2 2 2	A0 E0 A3	4+ 4+ 6+	2+ 2+ 2+	B0 F0 B3	5 5 7	3 3 3				A-M→A B-M→B D-M:M+1→D	8 8 8	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	↑ ↑ ↑	
SWI	SWI ⁶ SWI2 ⁶ SWI3 ⁶													3F 10 3F 11 3F	19 20 20 20 20	I 2 I I I	Software Interrupt 1 Software Interrupt 2 Software Interrupt 3	● ● ●	● ● ●	● ● ●	● ● ●	● ● ●	
SYNC														13	≥4	I	Synchronize to Interrupt		●	●	●	●	●
TFR	RI, R2	IF	6	2													RI→R2 ²		●	●	●	●	●
TST	TSTA TSTB TST				0D	6	2	6D	6+	2+	7D	7	3	4D 5D	2 2	I I	Test A Test B Test M	● ● ●	↑ ↑ ↑	↑ ↑ ↑	0 0 0	● ● ●	

- インデックス・アドレッシングの～(サイクル数)、および#の値は、ベースになる値のみを示してあり、その形態により必要数だけ加算されます。したがって、～および#の値は、“n+”の形で示しています。
- R1, R2は次の2群の中で、同一ビットのレジスタに限りペアで指定できます。
 <8ビット・レジスタ> A, B, CC, DP (ダイレクト・ページ・レジスタ)
 <16ビット・レジスタ> X, Y, U, S, D, PC
- “EA”とあるのは実効アドレスのことです。

- PSH, PUL 命令は、5 サイクルのほかにプッシュまたはプルするバイト数1ごとに1サイクル加算されます。
- (6) ブランチしないときは5サイクル、するときは6サイクル必要です。
- SWI 命令は、I, F フラグをセットします。SWI2, SWI3 命令は、同フラグを操作しません。
- CCRの値はAND, ORの実行結果として決まります。
- Hフラグの値は定義されていません。
- 近似値算定のため AccB 側の D₇ の値が入ります。

Instruction	Forms	Addressing Mode		#	Description	5	3	2	1	0
		Op	Relative							
BCC	BCC LBCC	24 10 24	3 5(6)	2 4	Branch C = 0 Long Branch C = 0	●	●	●	●	●
BCS	BCS LBBS	25 10 25	3 5(6)	2 4	Branch C = 1 Long Branch C = 1	●	●	●	●	●
BEQ	BEQ LBEQ	27 10 27	3 5(6)	2 4	Branch Z = 0 Long Branch Z = 0	●	●	●	●	●
BGE	BGE LBGE	2C 10 2C	3 5(6)	2 4	Branch ≥ Zero Long Branch ≥ Zero	●	●	●	●	●
BGT	BGT LBGT	2E 10 2E	3 5(6)	2 4	Branch > Zero Long Branch > Zero	●	●	●	●	●
BHI	BHI LBHI	22 10 22	3 5(6)	2 4	Branch Higher Long Branch Higher	●	●	●	●	●
BHS	BHS LBHS	24 10 24	3 5(6)	2 4	Branch Higher or Same Long Branch Higher or Same	●	●	●	●	●
BLE	BLE LBLE	2F 10 2F	3 5(6)	2 4	Branch ≤ Zero Long Branch ≤ Zero	●	●	●	●	●
BLO	BLO LBLO	25 10 25	3 5(6)	2 4	Branch Lower Long Branch Lower	●	●	●	●	●
BLS	BLS LBLS	23 10 23	3 5(6)	2 4	Branch Lower or Same Long Branch Lower or Same	●	●	●	●	●
BLT	BLT LBLT	2D 10 2D	3 5(6)	2 4	Branch < Zero Long Branch < Zero	●	●	●	●	●
BMI	BMI LBMI	2B 10 2B	3 5(6)	2 4	Branch Minus Long Branch Minus	●	●	●	●	●
BNE	BNE LBNE	26 10 26	3 5(6)	2 4	Branch Z ≠ 0 Long Branch Z ≠ 0	●	●	●	●	●
BPL	BPL LBPL	2A 10 2A	3 5(6)	2 4	Branch Plus Long Branch Plus	●	●	●	●	●
BRA	BRA LBRA	20 16 5	3 3	2 3	Branch Always Long Branch Always	●	●	●	●	●
BRN	BRN LBRN	21 10 21	3 5	2 4	Branch Never Long Branch Never	●	●	●	●	●
BSR	BSR HECK	8D 17 9	7 3	2 3	Branch to Subroutine Long Branch to Subroutine	●	●	●	●	●
BVC	BVC LBVC	28 10 28	3 5(6)	2 4	Branch V = 0 Long Branch V = 0	●	●	●	●	●
BVS	BVS LBVS	29 10 29	3 5(6)	2 4	Branch V = 1 Long Branch V = 1	●	●	●	●	●

PSH, PUL 命令の

ポスト・バイトとアドレス操作

PSH, PUL 命令のポスト・バイトは次の図のとおりです。ベースとなるスタック・ポインタによって一部異なる部分があるので注意が必要です。SP をベースにするときはUS がスタックの対象となり、US をベースにするときはSP がスタックに出し入れされます。

b7	b6	b5	b4	b3	b2	b1	b0
PC	S/U	Y	X	DP	B	A	CC

PC = Program Counter

S/U = Hardware/User Stack Pointer

Y = Y Index Register

X = U Index Register

DP = Direct Page Register

B = B Accumulator

A = A Accumulator

CC = Condition Code Register

TFR, EXG 命令のポスト・バイト

TFR, EXG 命令で使用されるポスト・バイトは次の図のとおりです。EXG 命令では転送元・先の区別なく、どちらを先に書いても同じです。

b7	b6	b5	b4	b3	b2	b1	b0
SOURCE (R1)				DESTINATION (R2)			

Code Register

0000 D (A:B)

0001 X Index

0010 Y Index

0011 U Stack Pointer

0100 S Stack Pointer

0101 Program Counter

1000 A Accumulator

1001 B Accumulator

1010 Condition Code

1011 Direct Page

Code Register

N. 音色データ一覧

FM音源カード付属データ				FM-77AVイニシエーターROM							
No.	音色名	No.	音色名	No.	音色名	No.	音色名	No.	音色名	No.	音色名
1	ブラス1	21	ビブラフォン	1	トランペット	21	E, オルガン2	41	シンバル	61	バード
2	ブラス2	22	シロフォン	2	ホルン	22	ハーブシコード	42	ティンパニー	62	ドッグ
3	トランペット	23	コト	3	チューバ	23	クラビネット	43	カウベル	63	テレフォン
4	ストリングス1	24	シタール	4	ブラス1	24	ギター	44	ベル1	64	アラーム
5	ストリングス2	25	クラビネット	5	ブラス2	25	E, ベース1	45	ベル2	65	ワイングラス
6	E, ピアノ1	26	ハーブシコード	6	ベル+ブラス	26	E, ベース2	46	スチールドラム1	66	クラッシュ
7	E, ピアノ2	27	ベル	7	ピッコロ	27	シンセベース	47	スチールドラム2	67	ハートビート
8	E, ピアノ3	28	ハーブ	8	フルート	28	シロフォン	48	パーカッション1	68	フットステップ
9	ギター	29	ベル+ブラス	9	クラリネット	29	グロッケン	49	パーカッション2	69	ユーフォー(UFO)
10	E, ベース1	30	ハーモニカ	10	オーボエ	30	ビブラフォン	50	トレイン1	70	レーザーガン
11	E, ベース2	31	スチールドラム	11	ファゴット	31	ハーブ	51	トレイン2	71	エキスプロージョン1
12	E, オルガン1	32	ティンパニー	12	ストリングス1	32	リコーダ	52	カー	72	エキスプロージョン2
13	E, オルガン2	33	トレイン	13	ストリングス2	33	ハーモニカ	53	モーターサイクル	73	サウンドエフェクト1
14	パイプオルガン1	34	アンビュランス	14	ピアノ	34	チター	54	グランプリ	74	サウンドエフェクト2
15	パイプオルガン2	35	トゥイート	15	E, ピアノ1	35	コト	55	パトロールカー	75	サウンドエフェクト3
16	フルート	36	レインドロップ	16	E, ピアノ2	36	スネアドラム1	56	アンビュランス	76	サウンドエフェクト4
17	ピッコロ	37	ホルン	17	E, ピアノ3	37	スネアドラム2	57	ヘリコプター	77	サインウェイブ
18	オーボエ	38	スネアドラム	18	パイプオルガン1	38	バスドラム	58	シップ		
19	クラリネット	39	カウベル	19	パイプオルガン2	39	オープンハイハット	59	ウェイブ		
20	グロッケン	40	パーカッション	20	E, オルガン1	40	クローズハイハット	60	レカンドロップ		

●FM77AVイニシエータROM音色データ

1	2	3	4
0000 02 02 02 02	0022 42 02 12 34	0044 41 01 10 30	0066 44 02 14 33
0004 19 0F 0F 0F	0026 20 09 09 22	0048 1C 06 06 1A	006A 19 08 06 1E
0008 BF 92 92 92	002A 0D 0E 4E 4B	004C 4E 4F 52 4F	006E 53 4F 56 50
000C 04 03 03 03	002E 1E 1E 1E 1E	0050 1E 1E 1E 1E	0072 1E 1E 1E 1E
0010 00 00 00 00	0032 0C 08 01 0A	0054 0F 09 09 08	0076 08 09 09 05
0014 FA FA FA FA	0036 09 09 0A 09	0058 09 09 0A 09	007A 08 09 0A 09
0018 3D 04 00 2F	003A 3D 00 00 1F	005C 3D 00 00 1F	007E 3D 04 00 4F
001C 00 0A 20 04	003E 00 00 19 00	0060 00 00 19 00	0082 00 0A 18 03
0020 02 00 00	0042 02 00 00	0064 02 00 00	0086 02 00 00
5	6	7	8
0088 41 01 11 31	00AA 44 7A 24 28	00CC 08 08 08 08	00EE 04 04 04 04
008C 1A 1B 0C 0C	00AE 17 19 06 08	00D0 23 1E 0C 0C	00F2 23 23 09 09
0090 11 14 54 54	00B2 52 5F 56 5F	00D4 59 12 53 52	00F6 54 12 52 52
0094 1E 1E 1E 1E	00B6 0C 09 08 0B	00D8 1F 04 04 04	00FA 19 04 04 04
0098 08 05 01 05	00BA 05 05 07 05	00DC 0A 00 00 00	00FE 19 00 00 00
009C 09 09 0A 09	00BE 2A 54 4A 44	00E0 0C 00 08 0B	0102 0F 0D 08 0B
00A0 3D 04 00 5F	00C2 3C 04 00 1F	00E4 34 06 00 5F	0106 1C 06 00 5F
00A4 00 0C 18 03	00C6 00 08 1B 19	00E8 09 07 1D 0B	010A 09 03 1D 0B
00A8 02 00 00	00CA 02 00 00	00EC 02 00 00	010E 02 00 00
9	10	11	12
0110 08 08 04 04	0132 02 02 06 04	0154 00 00 01 01	0176 12 02 1A 12
0114 1F 1E 0C 0C	0136 21 1C 09 09	0158 1B 14 09 09	017A 23 23 1C 06
0118 52 14 52 52	013A 57 57 52 52	015C 51 53 52 54	017E 5E 5C 5E 4D
011C 19 04 04 04	013E 05 04 04 04	0160 05 04 04 04	0182 0B 0D 0A 04
0120 08 00 00 00	0142 00 00 00 00	0164 00 00 00 00	0186 00 00 00 00
0124 08 00 08 0B	0146 08 0D 08 0B	0168 08 0D 08 0B	018A 1A 2A 5A 0A
0128 3C 04 00 3F	014A 34 02 00 1F	016C 34 02 00 1F	018E 3A 04 00 3F
012C 00 06 1B 03	014E 06 00 19 09	0170 01 00 16 09	0192 00 0A 20 00
0130 02 00 00	0152 02 00 00	0174 02 00 00	0196 02 00 00
13	14	15	16
0198 14 04 1C 14	01BA 31 23 07 31	01DC 01 0A 02 01	01FE 3F 01 01 01
019C 1B 33 1B 03	01BE 1E 2B 2F 00	01E0 1E 32 05 00	0202 26 2A 0A 03
01A0 5C 5C 53 4D	01C2 9D 1C 1C DF	01E4 9C DC 9C DC	0206 9D 9C 9F 5E
01A4 07 07 09 04	01C6 07 04 08 04	01E8 07 03 14 08	020A 08 06 08 05
01A8 00 00 00 00	01CA 04 19 1B 17	01EC 00 03 05 05	020E 1F 1F 1F 1F
01AC 13 13 32 09	01CE F0 F0 F0 F4	01F0 55 45 27 A7	0212 F8 F6 F8 F8
01B0 3A 04 00 1F	01D2 32 00 00 1F	01F4 04 00 00 1F	0216 04 00 00 1F
01B4 00 20 20 00	01D6 00 00 20 00	01F8 00 00 1A 00	021A 00 00 1A 00
01B8 02 00 00	01DA 02 00 00	01FC 02 00 00	021E 02 00 00
17	18	19	20
0220 31 23 0C 01	0242 78 65 78 01	0264 78 60 75 10	0286 75 73 58 01
0224 23 26 32 03	0246 14 00 0D 03	0268 1A 1F 17 03	028A 28 10 10 06
0228 9F 9F DF DF	024A 94 94 94 94	026C 94 92 90 92	028E 1F 1E 1C 1E
022C 04 04 04 04	024E 02 02 02 02	0270 02 02 02 02	0292 12 0F 04 0F
0230 1F 1F 1F 1F	0252 00 00 00 00	0274 00 00 00 00	0296 00 00 00 00
0234 F3 F3 F3 F3	0256 06 08 08 08	0278 07 05 07 06	029A 2F 0F 0F 0F
0238 3A 00 00 1F	025A 06 00 00 1F	027C 2C 00 00 1F	029E 1D 04 00 3F
023C 00 00 20 00	025E 00 00 1B 00	0280 00 00 1E 00	02A2 00 08 1E 00
0240 02 00 00	0262 02 00 00	0284 02 00 00	02A6 02 01 01

21	22	23	24
02A8 66 73 31 28	02CA 18 16 05 34	02EC 0C 01 05 01	030E 36 26 3A 34
02AC 13 0E 09 09	02CE 20 1E 26 0C	02F0 21 21 21 09	0312 2D 28 28 03
02B0 1F 1F 9F 1F	02D2 1F 1F DF 5F	02F4 9F 1F 1F 5F	0316 9F 9F 5E 9F
02B4 1E 14 14 14	02D6 0C 0C 02 1D	02F8 0C 04 05 0A	031A 06 1E 1E 1E
02B8 1F 00 00 00	02DA 04 04 04 0A	02FC 04 04 04 02	031E 06 08 12 09
02BC FF 0F 0F 0F	02DE 15 05 F4 08	0300 F7 17 07 AC	0322 04 04 05 05
02C0 2F 04 00 3F	02E2 3A 00 00 1F	0304 38 00 00 1F	0326 3A 04 00 2F
02C4 00 0A 20 00	02E6 00 00 1C 00	0308 00 00 1B 00	032A 00 05 14 00
02C8 02 01	02EA 02 00	030C 02 00	032E 02 00
25	26	27	28
0330 33 30 30 30	0352 30 71 10 20	0374 30 30 30 30	0396 1F 3A 24 02
0334 1F 1D 28 06	0356 1F 03 03 03	0378 24 1B 1A 00	039A 2A 20 23 00
0338 0F 9F DF 9F	035A DC 9F 99 98	037C DF 9F DF 9F	039E 5B 5E 5E 5E
033C 07 09 06 06	035E 0A 0A 08 07	0380 07 14 06 06	03A2 16 12 18 1E
0340 05 06 05 06	0362 00 00 00 00	0384 05 0A 05 06	03A6 0A 09 0B 10
0344 29 19 19 F9	0366 45 55 55 55	0388 28 18 18 38	03AA E8 E5 F9 07
0348 21 00 00 1F	036A 30 04 00 1F	038C 30 00 00 1F	03AE 2B 00 00 1F
034C 00 00 1A 00	036E 00 08 1B 19	0390 00 00 1A 00	03B2 00 00 18 00
0350 02 00	0372 02 00	0394 02 00	03B6 02 00
29	30	31	32
0388 00 00 04 04	03DA 3A 0A 52 02	03FC 04 04 02 02	041E 48 08 14 54
038C 19 23 15 06	03DE 28 32 06 06	0400 23 28 23 00	0422 14 1E 0C 0C
03C0 9F 9F 5F 9F	03E2 SE 5E 57 58	0404 9D 1F 5D DC	0426 1B 54 14 12
03C4 1F 1F 1E 1E	03E6 1F 1F 1F 1F	0408 1C 16 16 15	042A 14 01 02 07
03C8 15 0A 0A 0B	03EA 0A 05 0E 0D	040C 0D 0F 0A 06	042E 1F 02 05 01
03CC 05 05 06 06	03EE 03 04 05 05	0410 04 06 05 03	0432 88 58 FC 5A
03D0 0C 00 00 1F	03F2 24 06 00 1F	0414 39 04 00 3F	0436 2C 04 00 1F
03D4 00 00 12 00	03F6 00 05 17 00	0418 00 09 19 00	043A 00 0F 1A 06
03D8 02 00	03FA 02 00	041C 02 00	043E 02 00
33	34	35	36
0440 0C 0C 08 04	0462 37 21 0F 42	0484 06 06 04 04	04A6 15 30 13 70
0444 28 23 28 06	0466 21 22 2D 0C	0488 1E 22 1E 00	04AA 00 28 00 00
0448 0C 0F 0F 0E	046A 1C 58 19 9F	048C 5F 5F 5F 5F	04AE 5F 5E 9F 5F
044C 05 02 08 08	046E 04 11 03 04	0490 1F 1F 1F 1F	04B2 1F 1E 1F 1E
0450 00 00 00 00	0472 19 04 0F 04	0494 10 0C 0C 00	04B6 05 08 0F 11
0454 99 0A 09 1D	0476 24 02 02 F3	0498 08 06 06 06	04BA 02 08 09 09
0458 01 02 00 2F	047A 02 04 00 2F	049C 33 04 00 2F	04BE 3C 04 00 FF
045C 08 00 16 00	047E 00 05 15 00	04A0 00 0A 14 07	04C2 00 B4 14 00
0460 02 00	0482 02 00	04A4 02 00	04C6 02 00
37	38	39	40
04C8 73 02 04 01	04EA 00 01 00 00	050C 7F 0F 72 05	052E 7F 0F 72 07
04CC 00 0A 00 05	04EE 28 0F 00 00	0510 07 00 18 0F	0532 00 00 0F 0F
04D0 1F 1F 5F 1F	04F2 1D 1D 1D 1D	0514 1F 1F 5F 1F	0536 1F 1F 5F 1F
04D4 00 19 19 13	04F6 19 1B 19 19	0518 00 19 19 19	053A 1C 19 19 19
04D8 00 13 10 13	04FA 00 00 15 15	051C 00 00 1C 14	053E 00 00 11 13
04DC 06 F8 2A FC	04FE 36 F8 1C 1C	0520 00 08 38 5C	0542 06 28 3A 5C
04E0 3C 00 00 2F	0502 2C 00 00 2F	0524 34 00 00 1F	0546 2C 00 00 4F
04E4 00 00 28 00	0506 00 00 28 00	0528 00 00 28 00	054A 00 00 28 00
04E8 02 00	050A 02 00	052C 02 00	054E 02 00

41	42	43	44
0550 7F 0A 7D 0F	0572 16 32 10 70	0594 37 23 24 14	0586 45 75 24 24
0554 07 08 00 00	0576 22 1C 2D 00	0598 08 03 03 03	058A 16 19 0D 0D
0558 1F 1F 5F 1F	057A 5E 5E 5E 5F	059C 9C 5C 5C 5C	058E 5F 5F 5F 5F
055C 00 19 19 19	057E 1F 1E 1F 1E	05A0 18 14 14 14	05C2 09 09 08 08
0560 00 00 0C 0D	0582 0C 0C 0C 08	05A4 11 11 11 11	05C6 05 05 07 05
0564 00 02 36 58	0586 06 04 04 06	05AB 5A 4A 4A 4A	05CA 54 54 44 44
0568 34 02 00 1F	058A 38 00 00 1F	05AC 3D 04 00 2F	05CE 04 04 00 1F
056C 96 00 14 00	058E 00 00 50 00	0580 00 08 1B 08	05D2 00 08 1B 19
0570 02 00	0592 02 00	0584 02 00	05D6 02 00
45	46	47	48
05D8 0E 07 17 02	05FA 16 1D 18 14	061C 48 09 74 08	063E 44 04 74 04
05DC 0A 14 23 0C	05FE 1B 1E 06 06	0620 1E 28 06 06	0642 3C 32 1E 00
05E0 1F 1F 1F 1F	0602 5F 5F 5F 5F	0624 5F 56 59 94	0646 5F 5F 5F 9F
05E4 1A 0A 0A 0A	0606 08 06 0F 0F	0628 0F 0A 0F 0F	064A 0F 0A 0F 0F
05E8 00 00 00 00	060A 00 00 00 00	062C 1F 02 0E 0E	064E 0F 0F 0F 0F
05EC FF F2 F2 F5	060E 85 A5 F7 F7	0630 F7 F7 F6 F6	0652 0A 0A 06 06
05F0 38 00 00 1F	0612 34 00 00 1F	0634 34 04 00 1F	0656 02 06 00 FA
05F4 00 00 1B 00	0616 00 00 1B 00	0638 00 14 1A 0A	065A 0A 96 64 0A
05F8 02 00	061A 02 00	063C 02 00	065E 02 01
49	50	51	52
0660 44 04 74 0D	0682 05 34 01 70	06A4 0F 0F 0F 0F	06C6 70 10 56 40
0664 3C 32 1E 00	0686 00 17 17 03	06AB 00 0C 0C 0C	06CA 14 10 1A 06
0668 5F 5F 5F 9F	068A 18 4E 19 96	06AC 1F 0A 0A 0A	06CE 54 9F 5F 9F
066C 0F 0A 0F 0F	068E 1F 1E 1F 1E	06B0 1F 1F 1F 1F	06D2 04 03 0A 04
0670 0F 0F 0F 0F	0692 00 0A 04 05	06B4 00 00 00 00	06D6 00 00 00 00
0674 0A 0A 06 06	0696 01 0A 0A 06	06B8 02 08 08 08	06DA 00 0A 0A 0A
0678 02 06 00 FA	069A 3C 00 00 3F	06BC 3D 02 00 FF	06DE 1D 06 00 F2
067C 0A 96 64 0A	069E 00 00 15 00	06C0 FF 00 1A 00	06E2 0A 14 3C 00
0680 02 00	06A2 02 01	06C4 01 01	06E6 01 00
53	54	55	56
06E8 70 10 56 40	070A 70 70 00 08	072C 74 0C 04 56	074E 04 03 04 04
06EC 14 10 1A 06	070E 1E 06 0F 06	0730 23 1E 09 18	0752 0D 24 15 06
06F0 54 9F 5F 9F	0712 4A 9D 50 89	0734 14 1E 14 0F	0756 18 98 98 98
06F4 04 03 0A 04	0716 1F 1F 1F 06	0738 00 00 00 00	075A 1C 1E 1E 1E
06F8 00 00 00 00	071A 00 00 00 00	073C 00 00 00 00	075E 00 00 00 00
06FC 0A 0A 0A 0A	071E 02 02 02 F3	0740 F9 F9 F9 F9	0762 0A 0A 0A 0A
0700 35 06 00 F2	0722 18 04 00 FF	0744 3C 04 00 FF	0766 3F 04 00 FF
0704 0A 14 3C 00	0726 00 28 03 00	0748 00 14 0F 00	076A 00 14 0A 00
0708 01 00	072A 02 00	074C 00 01	076E 01 01
57	58	59	60
0770 00 10 00 02	0792 00 01 00 00	07B4 05 34 01 70	07D6 04 04 04 04
0774 09 00 46 06	0796 1F 1E 09 09	07B8 00 7F 03 03	07DA 03 03 03 03
0778 9E 1E 5F 5E	079A 14 1E 0F 19	07BC 18 4E 08 80	07DE 59 54 58 57
077C 14 14 0E 19	079E 00 00 00 00	07C0 1F 1E 0F 1F	07E2 12 12 12 12
0780 00 00 00 00	07A2 00 00 00 00	07C4 00 0A 0C 1F	07E6 10 12 10 11
0784 01 01 01 07	07A6 F9 F9 F9 F9	07C8 01 0A 06 FF	07EA 48 48 48 48
0788 38 06 00 9F	07AA 3C 00 00 1F	07CC 3C 00 00 2F	07EE 07 04 00 FF
078C 14 41 32 00	07AE 00 00 05 00	07D0 00 00 0C 00	07F2 00 87 19 00
0790 01 01	07B2 02 01	07D4 02 00	07F6 00 00

07F8 0C 0C 0C 0C	081A 06 08 04 04	083C 0A 03 03 06	085E 0C 08 0F 0E
07FC 2D 28 00 00	081E 28 00 00 00	0840 32 28 37 0A	0862 18 10 1A 10
0800 59 12 53 54	0822 52 50 50 50	0844 5F 5F 9F 94	0866 1F 0F 0F 0F
0804 1F 04 1E 14	0826 05 13 13 13	0848 05 00 00 00	086A 00 00 00 00
0808 0A 00 12 12	082A 00 00 00 00	084C 06 06 06 06	086E 00 00 00 00
080C 0C 0D 08 08	082E FC FC FC FC	0850 65 65 19 69	0872 F5 F7 F7 F7
0810 34 06 00 FF	0832 3D 04 00 FF	0854 00 04 00 5F	0876 06 04 00 FF
0814 09 64 46 00	0836 00 F0 32 00	0858 00 C8 3C 00	087A 00 FF 4B 00
0818 00 00	083A 00 00	085C 01 01	087E 01 01
0880 4D 78 18 28	08A2 0F 0F 00 00	08C4 00 70 00 00	08E6 08 08 04 0C
0884 32 06 06 06	08A6 0A 08 00 00	08C8 09 00 00 00	08EA 19 19 00 00
0888 59 5B 59 5D	08AA 1F 1F 1F 1F	08CC 5C 09 1F 4B	08EE 52 14 57 53
088C 09 09 08 0B	08AE 05 05 0F 0F	08D0 14 1F 11 1F	08F2 1F 1F 1F 1F
0890 05 05 07 05	08B2 00 00 00 00	08D4 00 00 00 03	08F6 18 12 18 18
0894 54 54 44 44	08B6 F5 F5 F7 F7	08D8 FC FF FC FF	08FA 0A 0D 08 0B
0898 05 04 01 1F	08BA 34 06 00 FA	08DC 04 02 00 9F	08FE 3C 00 00 1F
089C 00 08 1B 0A	08BE FF FF 5A 00	08E0 19 00 0D 00	0902 00 00 19 00
08A0 02 01	08C2 03 01	08E4 01 00	0906 02 00
0908 7C 14 50 46	092A 04 08 0F 08	094C 70 70 00 00	096E 70 10 56 40
090C 14 13 1D 09	092E 1E 0F 09 0E	0950 00 14 03 0C	0972 0D 0A 0C 0C
0910 54 94 54 94	0932 18 14 14 12	0954 4A 9E 5C 9F	0976 54 9F 5F 9F
0914 1F 1F 1F 1F	0936 14 01 02 07	0958 04 03 04 04	097A 04 03 0A 04
0918 00 00 00 00	093A 01 01 01 01	095C 1F 1F 1F 1F	097E 1F 1F 1F 1F
091C 09 09 09 09	093E 08 08 0C 0A	0960 F5 F5 F5 F6	0982 F5 F5 F7 F7
0920 35 04 00 5F	0942 2C 04 00 5F	0964 3A 04 00 FF	0986 3C 06 00 F2
0924 00 14 0A 00	0946 00 14 50 00	0968 00 64 5A 00	098A 0A 14 96 00
0928 00 01	094A 00 01	096C 03 00	098E 03 00
0990 34 32 07 01	09B2 0F 0F 24 1A	09D4 0A 0C 05 06	09F6 0C 08 01 0D
0994 3C 25 0F 00	09B6 14 28 0F 0F	09D8 17 00 0C 0C	09FA 0A 00 00 00
0998 1D 1D 1D 1F	09BA 9F 1F 5F 5F	09DC 1F 1F 0A 0A	09FE 9F 1F 5F 5F
099C 06 00 00 00	09BE 00 00 00 00	09E0 1F 1F 1F 1F	0A02 13 12 14 14
09A0 00 06 06 07	09C2 00 00 00 00	09E4 00 00 00 00	0A06 02 06 0C 0C
09A4 F4 24 04 F8	09C6 A3 D3 F5 F5	09E8 02 08 08 08	0A0A 53 39 29 29
09A8 31 04 00 AF	09CA 0C 04 00 FF	09EC 3C 06 00 FF	0A0E 3C 04 00 7F
09AC 00 FF 78 00	09CE 00 C8 32 00	09F0 1E FF 14 00	0A12 00 2D 19 04
09B0 01 01	09D2 01 00	09F4 01 00	0A16 02 00
0A18 02 02 02 02			
0A1C 7F 7F 7F 00			
0A20 1F 1F 1F 1F			
0A24 00 00 00 00			
0A28 00 00 00 00			
0A2C 0F 0F 0F 0F			
0A30 00 00 00 00			
0A34 00 00 00 00			
0A38 00 00			

●FM音源カード付属音色データ

1	2	3	4
0000 02 02 26 12 0004 19 2A 20 00 0008 8D 15 4F 52 000C 06 07 08 04 0010 02 00 00 00 0014 18 28 18 28 0018 3A 04 00 10 001C 00 0A 09 00 0020 02 00	0022 01 01 2A 11 0026 1A 36 1E 00 002A 8E 4F 59 52 002E 0A 08 00 03 0032 00 00 03 00 0036 15 25 FF 28 003A 3A 00 00 00 003E 00 00 00 00 0042 00 00	0044 01 01 07 01 0048 17 26 28 00 004C 8D 8E 8D 53 0050 0E 0E 0E 03 0054 00 00 00 00 0058 13 13 FA 0A 005C 3A 04 00 10 0060 00 0A 20 00 0064 02 00	0066 11 01 15 11 006A 1D 30 0F 00 006E 59 5C 59 4E 0072 0A 00 08 04 0076 00 00 00 00 007A 15 26 58 06 007E 3A 04 00 10 0082 00 0A 20 00 0086 02 00
5	6	7	8
0088 12 02 1A 12 008C 1B 33 1B 00 0090 5C 5C 53 4D 0094 07 07 09 04 0098 00 00 00 00 009C 13 13 32 07 00A0 3A 04 00 10 00A4 00 20 20 00 00A8 02 00	00AA 11 0A 21 11 00AE 17 39 0A 00 00B2 9A DA 98 D8 00B6 0F 07 0C 0C 00BA 00 03 05 05 00BE 26 46 28 28 00C2 14 00 00 00 00C6 00 00 00 00 00CA 02 00	00CC 3F 01 01 01 00D0 2B 2A 14 00 00D4 9F 9E D8 5E 00D8 0F 06 07 06 00DC 08 08 0A 00 00E0 88 F6 8A F7 00E4 1C 02 00 06 00E8 30 00 1E 00 00EC 02 00	00EE 31 24 0C 01 00F2 21 31 47 00 00F6 9C 9C D8 DA 00FA 04 04 09 03 00FE 03 01 03 00 0102 17 06 02 45 0106 3A 00 00 00 010A 00 00 00 00 010E 02 00
9	10	11	12
0110 32 33 33 31 0114 29 28 23 00 0118 0F 0F 0F 9F 011C 07 04 04 0A 0120 04 04 04 03 0124 F7 FB FB 08 0128 39 06 00 12 012C 02 02 0A 00 0130 02 00	0132 36 30 35 31 0136 1C 16 3A 00 013A 0F 9F 0F 9F 013E 07 09 06 06 0142 07 06 06 08 0146 29 19 19 F9 014A 20 00 00 00 014E 00 00 00 00 0152 02 00	0154 30 30 30 30 0158 17 17 3F 00 015C 9E DC D8 DC 0160 0E 04 0A 05 0164 08 08 08 08 0168 B6 B6 B6 B6 016C 30 00 00 00 0170 00 00 00 00 0174 02 00	0176 35 72 58 31 017A 27 1B 11 00 017E 0F 1F 1F 1F 0182 12 0F 04 0F 0186 00 00 00 00 018A 2F 0F 0F 0F 018E 1D 04 00 10 0192 00 10 1A 00 0196 02 00
13	14	15	16
0198 66 73 31 22 019C 11 0A 0A 00 01A0 1C 1F 9F 1F 01A4 12 0F 0F 0F 01A8 00 00 00 00 01AC FF 0F 0F 0F 01B0 1F 04 00 10 01B4 00 0A 20 00 01B8 02 00	01BA 78 60 74 10 01BE 1F 1F 1B 00 01C2 94 92 90 92 01C6 02 02 02 02 01CA 00 00 00 00 01CE 07 05 07 06 01D2 04 00 00 00 01D6 00 46 1E 00 01DA 02 00	01DC 14 0C 12 06 01E0 26 26 19 00 01E4 92 5E 12 59 01E8 0A 06 04 05 01EC 02 02 00 00 01F0 47 19 08 09 01F4 1C 00 00 00 01F8 00 00 00 00 01FC 02 00	01FE 76 33 06 03 0202 28 26 33 00 0206 0F D0 54 8F 020A 09 08 07 04 020E 03 00 00 00 0212 E5 25 F5 08 0216 3B 06 00 21 021A 01 04 1A 00 021E 02 00
17	18	19	20
0220 04 04 04 04 0224 23 0A 0C 00 0228 1F 14 14 14 022C 0A 08 06 08 0230 0F 00 00 00 0234 59 19 F9 19 0238 3E 04 00 10 023C 00 04 20 00 0240 02 00	0242 31 32 39 34 0246 1F 32 28 00 024A 09 56 DC 54 024E 08 00 0C 06 0252 00 0C 00 00 0256 13 18 58 08 025A 3A 04 00 10 025E 00 09 20 00 0262 02 00	0264 01 04 02 01 0268 27 33 24 00 026C 5F 5F 13 53 0270 00 0F 08 0F 0274 00 00 00 00 0278 00 09 48 09 027C 3B 04 00 10 0280 00 08 1A 00 0284 02 00	0286 3B 07 01 71 028A 21 17 15 00 028E 0F 5F DE DE 0292 0E 10 09 07 0296 00 05 07 04 029A FF A0 16 17 029E 0C 04 00 20 02A2 00 1E 20 00 02A6 02 00

21	22	23	24
02A8 39 05 56 01	02CA 35 33 21 31	02EC 33 13 43 13	030E 37 21 39 31
02AC 1E 41 19 00	02CE 19 1B 0C 00	02FO 1E 1E 1E 00	0312 1C 1D 45 00
02B0 5F 9E D8 9E	02D2 59 59 59 59	02F4 DA DC DD DF	0316 9F 9F 9F 9F
02B4 10 0C 07 05	02D6 17 14 0E 0E	02F8 08 04 05 0A	031A 06 0C 06 0C
02B8 00 0B 0A 0A	02DA 0A 0B 0B 0C	02FC 05 02 04 03	031E 06 06 06 06
02BC BA F6 85 F5	02DE E8 E8 F8 F8	0300 27 36 14 15	0322 01 01 01 05
02C0 24 04 00 32	02E2 3C 06 00 0C	0304 38 00 00 00	0326 02 04 00 10
02C4 00 00 1A 00	02E6 04 0C 14 00	0308 00 00 00 00	032A 00 01 20 00
02C8 02 00	02EA 02 00	030C 02 00	032E 02 00
25	26	27	28
0330 3C 30 39 31	0352 0C 01 1F 53	0374 33 35 21 33	0396 02 02 01 01
0334 21 07 22 00	0356 20 1E 39 00	0378 16 19 0F 00	039A 1C 30 2A 00
0338 DF 1F 1F DF	035A 1F 1F DF 9F	037C 1D CA 1D 5E	039E 5F 1F 5F 9F
033C 04 04 05 01	035E 0C 0C 02 05	0380 04 04 04 04	03A2 0C 06 00 0C
0340 04 04 04 02	0362 04 04 04 07	0384 01 04 03 03	03A6 00 05 06 07
0344 F7 17 07 AC	0366 1A 06 F6 27	0388 20 30 03 03	03AA F4 15 15 14
0348 3B 00 00 00	036A 3A 00 00 00	038C 1C 00 00 00	03AE 39 00 00 00
034C 00 00 00 00	036E 00 00 00 00	0390 00 00 00 00	03B2 00 00 00 00
0350 02 00	0372 02 00	0394 02 00	03B6 02 00
29	30	31	32
03B8 31 3B 31 31	03DA 3A 11 0A 02	03FC 73 38 03 33	041E 75 75 77 05
03BC 1A 24 0A 00	03DE 28 1D 33 00	0400 23 1C 0A 00	0422 16 25 2A 00
03C0 10 1F 12 1F	03E2 14 10 14 0E	0404 56 54 99 99	0426 1D 1D 1D 1F
03C4 06 0F 03 0F	03E6 05 02 08 08	0408 05 07 0C 0C	042A 06 00 00 00
03C8 01 06 06 04	03EA 00 00 00 00	040C 06 06 06 06	042E 00 06 06 07
03CC 03 F2 0C F2	03EE 99 09 09 19	0410 64 64 15 65	0432 F4 24 04 05
03D0 3C 04 00 40	03F2 38 04 00 10	0414 04 00 00 00	0436 39 04 00 10
03D4 00 07 20 00	03F6 00 14 20 00	0418 00 00 00 00	043A 00 0A 09 00
03D8 02 00	03FA 02 00	041C 02 00	043E 02 00
33	34	35	36
0440 35 34 39 30	0462 02 02 02 01	0484 38 32 32 32	04A6 3F 37 21 31
0444 0A 29 15 00	0466 30 1B 1D 00	0488 31 53 25 00	04AA 29 30 16 00
0448 4D 0C CD 14	046A 04 04 04 00	04BC DF D4 15	04AE D2 DE DE DE
044C 02 01 02 01	046E 00 00 00 00	0490 0A 0B 06 0B	04B2 0C 0E 0F 0A
0450 02 01 02 01	0472 02 02 02 02	0494 0F 0F 00 09	04B6 0A 0B 0B 01
0454 08 56 08 18	0476 05 05 05 05	0498 59 19 F9 19	04BA F7 E7 F7 17
0458 3C 00 00 00	047A 3E 04 00 80	049C 3E 06 00 92	04BE 3C 06 00 72
045C 00 00 00 00	047E 00 40 0A 00	04A0 03 3C 0C 00	04C2 04 40 0B 00
0460 02 00	04B2 01 00	04A4 02 00	04C6 03 00
37	38	39	40
04C8 01 01 01 01	04EA 0F 00 00 00	050C 5F 07 5B 02	052E 00 70 0E 11
04CC 1F 1F 00 00	04EE 0E 13 11 00	0510 00 20 23 00	0532 0D 1D 14 00
04D0 10 10 10 10	04F2 1F 1F 1F 1F	0514 1F 1F 1F 1F	0536 1F 1F 1F 5F
04D4 01 01 02 02	04F6 00 18 0F 13	0518 15 15 13 13	053A 01 16 0D 14
04D8 00 00 00 00	04FA 00 00 11 10	051C 13 00 0C 10	053E 00 0F 07 14
04DC 18 18 18 18	04FE 08 88 2C 2C	0520 26 36 26 29	0542 25 68 FA F8
04E0 34 00 00 00	0502 3C 00 00 00	0524 3B 00 00 00	0546 34 00 00 00
04E4 00 00 00 00	0506 00 00 00 00	0528 00 00 00 00	054A 00 00 00 00
04E8 02 00	050A 02 00	052C 02 00	054E 02 00

0. プログラムの入力にあたって

まずマシン語プログラムの入力方法の説明に先立って、BASICプログラムの入力方法についての注意点をあげます。

- ① BASIC プログラムには、F-BASIC V3.0 のみで動作するもの、V3.3 のみで動作するもの、あるいは V3.0/V3.3 の両方で動作するものがあります。これらの区別は、プログラムの先頭のコメントの中で明記してあります。
- ② BASIC プログラムには、マシン語サブルーチンを必要とするものがあります。これもプログラム先頭のコメントに明記してありますので、必要なマシン語サブルーチンをフロッピーディスクにセーブしておく必要があります。(例えば 'LIST1-1 ガヒツヨウデス.' というコメントがあれば、リスト 1-1 を入力して 'L1-1M' というファイルにセーブしておく必要があります。
- ③ プログラムのリストは、編集上の都合で 1 行 70 文字で印字されています。実際にプログラムを入力される場合には、その点に注意して 1 行 80 文字又は 1 行 40 文字に直して入力してください。

```

10 '*****
11 '*      JOYSTIC TEST      *
12 '*      ( LIST 5-10 )    V3.0/V3.3 ..... V3, 3/V3, 3のどちらでも
13 '*      LIST 5-9   カ"   ヒツヨウ   テ"入 ..... 動作することを示します
14 '*****                  マシン語サブルーチンが必要
                               なことを示しています
20 WIDTH 80,25:LOADM "LS-9M" ..... マシン語サブルーチンをロードしています
30 EXEC&H5000
40 A$="..... ウェ..... ミキ"ウェ.. ミキ"..... ミキ"シタ.. シタ..... ヒタ"リシ
   タ. ヒタ"リ... ヒタ"リウェ. "
50 EXEC&H5003
60 I=PEEK(&H5006)
70 LOCATE 36,12
80 PRINT MID$(A$,I*8+1,8)
90 LOCATE 36,14
100 IF PEEK(&H5007) THEN PRINT"TRIG(1) ";
110 IF PEEK(&H5008) THEN PRINT"TRIG(2) ";
120 PRINT"      "
130 GOTO 50

```

マシン語プログラムのリストには、チェックサムつきメモリダンプとアブソリュートアセンブラによるアセンブルリストの2つの形式で示してあります。チェックサムつきメモリダンプをもとにマシン語プログラムを入力するには、次の方法で行ないます。

- ① モニタの M コマンドにてメモリダンプの内容を入力します。その時にはチェックサムを入力する必要はありません。
- ② 入力したマシン語プログラムをフロッピーディスクにセーブします。セーブの方法は、メモリダンプの次に明記してあります。
- ③ 後述のチェックサムプログラムを起動して、ファイル名、先頭アドレス(16進4桁)、終了アドレス(16進4桁)を入力します。チェックサムつきメモリダンプが表示されますから、本文中のメモリダンプと比較してみてください。その時には、チェックサムを比較することによって入力ミスを簡単に発見することができます。
- ④ 入力ミスがあった場合には、モニタの M コマンドにて誤りの部分を入力し直して、②の項目から作業をやり直します。
- ⑤ マシン語プログラムには、F-BASIC V3.0 のみで動作するもの、V3.3 のみで動作するもの、あるいは V3.0/V3.3 のどちらでも動作するものがあります。本文中の記述を十分に確認してプログラムを実行してください。

マシン語データ部分

ADR :	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	: [cs]
5000 :	8E	50	07	6E	9F	FB	FA	0E	00	50	0D	00	12	46	4D	2D	: 24
5010 :	54	65	63	68	6B	6E	6F	77	20	37	37	41	56	0D	0A	00	: 7F
5020 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5030 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5040 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5050 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5060 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5070 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5080 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
5090 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50A0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50B0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50C0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50D0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50E0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
50F0 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	: 00
[cs] :	E2	B5	6A	D6	0A	69	69	85	20	87	44	41	68	53	57	2D	: A3

縦チェックサム

SAVEM "L9-1M", &H5000, &H501E, &H5000セーブの方法

```

1000 '*****
1010 '*      MEMORY DUMP      *
1020 '*      ( CHKSUM ) V3.0/V3.3  *
1030 '*****
1032 CLEAR:&H4000
1040 DIM CSUM(17):CLS:WIDTH 80,25
1050 INPUT "FILE NAME      = ";FILN$
1070 INPUT "START ADDRESS = ";STADR$:STX=VAL("&H"+STADR$)
1080 INPUT "END   ADDRESS = ";EADR$:ENX=VAL("&H"+EADR$)
1090 IF STX>&H6AFF THEN FOR I=STX-&H2000 TO ENX-&H2000+256:POKE I,0:NEXT:LOADM FI
LN$:
1100 FOR STADR=STX TO ENX STEP 256
1110   COLOR 6:PRINT "   ADR : +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +
C +D +E +F : [cs]"
1120   FOR I=0 TO 17:CSUM(I)=0:NEXT
1130   FOR ADR=STADR TO STADR+240 STEP 16
1140     COLOR 5:PRINT "   ";RIGHT$("0000"+HEX$(ADR),4);" : ";
1150     CSUM(16)=0:COLOR 7
1160     FOR I=0 TO 15
1162       IF ADR>&H6AFF THEN DD=PEEK(ADR-&H2000+I) ELSE DD=PEEK
(ADR+I)
1170       PRINT RIGHT$("00"+HEX$(DD),2);" ";
1180       CSUM(16)=(CSUM(16)+DD) MOD 256
1190       CSUM(I)=(CSUM(I)+DD) MOD 256
1200     NEXT
1210     COLOR 5:PRINT " : ";RIGHT$("00"+HEX$(CSUM(16)),2)
1220     CSUM(17)=(CSUM(17)+CSUM(16)) MOD 256
1230   NEXT
1240   PRINT "   ";STRING$(61,"-")
1250   PRINT "   [cs] : ";:FOR I=0 TO 15:PRINT RIGHT$("00"+HEX$(CSUM(
I)),2);"   ";:NEXT:PRINT " : ";RIGHT$("00"+HEX$(CSUM(17)),2)
1260   PRINT
1261   PRINT:COLOR 2:PRINT SPC(20);"<<<  Hit Any Key !! >>>":A$=INP
UT$(1):PRINT:PRINT
1270 NEXT
1280 COLOR 7:END

```

マシン語プログラムのアセンブリリストは、富士通のアブソリュートアセンブラによって作成されています。

アブソリュートアセンブラによりソースプログラムを入力される方は、プログラム先頭部分の“OPT NOGEN”の指定をとってアセンブルし直してください。

モニタの M コマンドにてマシン語プログラムを入力する場合には、次の点に注意して入力してください。

- ① アセンブリリストのアドレスを確認しながら、マシン語プログラムのオブジェクトコード部分のみを入力します。相対ブランチ命令には、ブランチ先のアドレスも表示されていますので、その部分は入力しないように注意する必要があります。
- ② 複数のデータが、`,` (カンマ) で区切って続して定義されている場合には最初のデータのみがオブジェクトコード部分に表示されていて、他のデータは表示されません。また RMB 擬似命令では、確保するバイト数がオブジェクトコード部分に表示されます。ですからデータ定義擬似命令に関しては、ソースリストを参照して正しい値を設定し直さなければなりません。
- ③ 入力し終わったら、モニタの D コマンドにて十分チェックしてフロッピーディスクにセーブします。

PAGE 001 (860626.103841)

V3.0/V3.3のどちらでも
動作することを示します

```

01000
01020
01030 アドレス
01040
01050
01060 5000 オブジェクトコード
01080 FBFA
01090
01100 5000 8E 5007
01110 5003 6E 9F FBFA
01120
01130 5007 0E
01140 5009 500D
01150 5008 0012
01170 5000 46
01180 5010 0D
01190 501F
01200
01210 5000
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END ADDR=501E
PROGRAM ENTRY ADDR=5000
  
```

```

*****
*          LPOUT サンプリングプログラム          *
*          ( LIST 9-1 )          V3.0/V3.3          *
*****
OPT      NOGEN ..... ソースリストの入力時
ORG      $5000 ..... には入力不要
BIOS     EQU      $FBFA
*
ENTRY   LDX      #RCB
        JMP      [BIOS]
*
RCB     FCB      14.0
        FDB      DATA_S
        FDB      DATA_E-DATA_S
DATA_S  FCC      'FM-Techknow 77AV'
        FCB      $0D,$0A
DATA_E  EQU      *
*
END     ENTRY
  
```

索引

<A>

ACHROT73, 82, 91, 92, 94
ADM-3Aオーダー298, 302, 304
APRTOT73, 215, 216, 217
ATTENTION信号89

BEEPOF72, 81
BEEPON72, 81
BIINIT72
BOOT ROM34, 178, 214
BREAKキー割り込み150
BUSY信号89

<C>

CANCEL信号89
CAP LED107, 119, 121, 122, 123
CCR151
CTBRED72, 160
CTBWRT72, 160

<D>

DEFINE STRING OF PF92, 96
DIGITIZE96, 339, 341
DREAD72, 176, 177
DWRITE72, 176, 177
DWTSSG73, 262

<F>

FASTクロック296
FAT169, 172, 361, 364
FATセーブレート364
FDC168, 178, 179, 180, 185
FDCレジスタ178
FIRQ(割り込み)88, 113, 127, 149, 154, 155
FIRQ割り込みフラグ150
FLEX353
FM16 β 準拠モード120
FM音源レジスタ247, 252
FNOSET73, 262
FRQSET73, 262
F-BASICオーダー298, 302
Fナンバー248

<H>

HALT信号89, 97
HDCOPY72, 215

<I>

ID(セクタ)34, 169, 170, 171
IKEMOTO107
INPUT72, 91, 93
INPUTC72, 91, 93
INS LED19, 85, 107, 119, 121
INTERRUPT CONTROL92, 96, 128
IPL34, 169, 170, 175
IRQ(割り込み)
88, 149, 151, 154, 178, 180, 184, 244, 248, 256
IRQ割り込みフラグ150
I/O領域22

<J>

JIS 9 bitモード120

<K>

KANJIR72, 265, 268
KEYBOARD CONTROL72, 96, 121, 122, 124
KEYIN72, 91, 93, 117
KEYON73, 262
KOFFAL73, 262

<L>

LINE92, 95, 96
LINE292, 95, 96
LPCHK72, 215, 216
LPOUT72, 215, 216, 217

<M>

MML243, 256, 260
MMR12, 52, 77, 320, 324, 334
MOTOR72, 160
MSR334

<N>

NCU307
NM割り込み88, 149, 150, 154, 155

<O>

OPNBIOS 71, 72, 77, 79, 262
 OUTPUT 72, 91, 93

<P>

PFキー定義テーブル 86, 87, 127
 PFキー割り込み 89, 90, 127, 147, 152
 PRTCHK 73, 215, 216
 PSGレジスタ 230

<R>

RAM空間 11, 13, 52, 336, 338
 RAMモード 14
 RCBインターフェース 71, 78
 READY RDQ 90
 READ RTC 92, 96, 140
 READ TIMER 92, 96, 140
 REDSSG 73, 262
 REDSTR 73, 262
 RESETベクトル 34
 RESTORE 72, 176, 177
 ROMモード 14
 RS-232C割り込み 147
 RTC 119, 140, 143
 RxRDY割り込み 151

<S>

SCREEN 72, 215, 216
 SEEK5 72, 176, 177
 SELCAR 73, 262
 SELECT DISPLAY MODE 28, 30, 92, 95
 SET RTC 92, 96, 143
 SET TIMER 92, 96, 143
 SLOWクロック 296
 SPECIALコマンド 102, 107
 SSGCLR 73, 262
 SUBIN 72, 91, 92
 SUBOUT 72, 91, 92, 98
 SWI2割り込み 88, 149, 150, 154, 155
 SWI3割り込み 88, 149, 150, 154, 155
 SWI割り込み 88, 149, 150, 154, 155

<T>

TELEVISION CONTROL 96, 339
 TESTコマンド 96, 98
 TRSPRM 73, 83, 262
 TTLSET 73, 262
 TWR 11, 334, 338

<V>

VOLSET 73, 262
 VSYNC 319, 330

<W>

WRTFM 73, 262
 WRTPRM 73, 262
 WRTSSG 73

<X>

Xパラメータ 296, 298
 XON 296, 298
 XOFF 296, 298

<Y>

YAMAUCHI 98, 107

<0>

0時割り込み 90

<4>

4096色(モード) 11, 85, 221, 309, 313, 343

<あ>

アクティブVRAMコード 30, 31
 アクティブページ 29, 85, 309
 アテンション割り込み 150, 152
 アトリビュートバッファ 86, 87, 102, 104, 105
 アナログパレット 313, 318, 343
 アルゴリズム 246, 256

<い>

イニシエータROM 11, 23, 83, 178, 254
 インターバルタイマー割り込み 90, 134

<う>

ウィンドウ領域 13, 16, 17, 22, 338
 裏RAM 13, 21, 36, 58, 178, 196

<え>

エラーステータス 74
 エンベロープ 230

<お>

オシレータ 230

オフセットアドレスレジスタ.....99, 330, 331
 オペレータ.....245
 音階(データ).....73, 251
 音響カブラ.....305
 音色(データ).....23, 73, 83, 244, 254, 257, 260
 音程.....244, 248, 257
 音量(データ).....73, 233, 234
 オートリピート.....121, 122

<か>

外字(フォント).....283, 286
 拡張RAM空間.....11, 20
 拡張コマンドテーブル.....86, 109
かな LED.....107, 119, 121, 122, 124
 ガベージコレクション.....41, 42, 50
 漢字ROM.....71, 265, 271
 漢字SYMBOL.....287
 漢字アドレス.....265, 271
 漢字パターン.....265, 276

<き>

キャラクタROM.....25, 26, 28
 キャラクタコードバッファ.....85, 86, 102
 共有メモリ(RAM).....22, 25, 26, 33, 86, 87, 89, 90, 97
 キーオン.....73, 262
 キーオフ.....73, 262
 キー入力バッファ.....85, 86, 107, 113, 114, 115, 127
 キーの先行入力.....115
 キーボードエンコーダ.....107, 119
 キーボード割り込み.....151

<く>

クラスタ(番号).....168, 171, 172

<こ>

コネクション.....246
 コンソールRAM(バッファ).....19, 25, 26, 86, 87, 218

<さ>

サイド(番号).....167, 177
 サブCPU空間.....11, 19, 325
 サブバンクレジスタ.....88
 サブモニタROM.....25, 26, 86, 87, 126

<し>

シリンダ(番号).....168

<す>

スキャンコードモード.....114, 120, 123
 スキュー.....175, 199
 スタートビット.....295
 ステッピングレート.....181, 199, 214
 ストップビット.....295, 297
 スtringディスクリプタ.....55
 スロット.....246
 スーパーインポーズ.....119, 339, 340

<せ>

セクタ(番号).....167, 168, 188
 セクタアドレス.....168
 セクタシーケンス.....175, 199
 全二重通信.....295, 298, 305

<た>

タイマー割り込み.....89, 90, 127, 151
 ダイレクトアクセス.....19, 104, 324
 単純変数領域.....39, 40, 46
 ターミナル(モード).....291, 296, 302

<ち>

チェックサム.....158
 中間言語(コード).....12, 44, 55, 243
 直線補間.....94, 326

<つ>

通信制御バッファ.....296

<て>

ディスプレイVRAMコード.....31
 ディスプレイタイミング.....319
 ディスプレイページ.....29, 30, 85, 309, 319
 ディレクトリ.....169, 171, 357
 テキストウィンドウ.....11, 18, 21, 52
 テキスト領域.....12, 16, 39, 40
 デジタイズ.....53, 119, 339, 340
 データ伝送速度.....294, 296

<と>

トラック(番号).....167, 177, 198

<の>

ノイズ(ジェネレータ).....230, 231, 232, 242

<は>

配列変数領域.....39, 40, 48, 52

パリティ(チェック) 294, 297
 パレットレジスタ 313
 バンクレジスタ 85, 311
 半二重通信 295, 298, 306
 ハードウェアスクロール 99, 107, 329
 ハードウェア描画機能 326
 ハードコピー 53, 72, 215

<ひ>

ビジー制御 296, 298
 非常駐モジュール 16, 52
 標準窮間 14, 21, 52

<へ>

ヘッド(番号) 168
 ページ切り替え 29, 309, 306

<ほ>

ホスト(コンピュータ) 302
 ボーレート 294, 296, 300

<ま>

マウスセット 134, 135
 マルチページ 30, 33

<み>

ミキサー 230
 未定義命令 55, 56, 60
 未表示VRAM 331

<も>

文字領域 40, 41, 50
 モジュレータ 246
 モデム 291, 293, 305, 307

<め>

メッセージテーブル 19, 21
 メモリモード 33

<よ>

予約時刻割り込み 147

<り>

リンクポインタ 42, 43, 59

<れ>

レジスタインターフェース
 71, 73, 74, 76, 78, 82, 215, 217

<ろ>

論理演算回路 324, 326, 327

<わ>

割り込み許可 147
 割り込み禁止 147
 割り込みフックテーブル 86
 割り込みベクトルテーブル 86, 87, 154
 割り込み保留 147
 割り込みマスク 149, 151

■メディア・サービスのお知らせ

本書では、次のような方々のためにメディア・サービスを行なっています。
どしどし御利用ください。

- 忙しくてプログラムを打ち込む時間をもったいない人
- より理解を深め、テクニック向上を目指すためにソースプログラムの改造にチャレンジしたい人
- たくさんプログラムの詰まったメディアを持っていることに幸せを感じる人

●対応機種

FM-7/NEW7/77/AV

●メディア

5インチ2D/2枚組 3.5インチ2D/2枚組

●価 格

相方とも3,900円

●申し込み方法

巻末にとじこんである「郵便振替用紙」の空欄に、

FMメディアサービス (メディアタイプ)

と明記し、代金3,900円を添えてお近くの郵便局にお申し込みください。

参考文献

- | | |
|----------------------------------|---------------|
| ● F-BASIC V3.0/3.3/3.5 文法書 | 富士通 |
| ● FM77AV ハードウェア解説書 | 富士通 |
| ● FM77AV ディスプレイサブシステム解説書 | 富士通 |
| ● FM77AV BIOS 解説書 | 富士通 |
| ● ビデオデジタイズカード取扱説明書 | 富士通 |
| ● RS-232C IF カード取扱説明書 | 富士通 |
| ● マウスセット取扱説明書 | 富士通 |
| ● MB8876A/77A ユーザーズマニュアル | 富士通 |
| ● PC-Techknow 9800 | システムソフト |
| ● PC-Techknow 8800mk II | システムソフト |
| ● FM-7 F-BASIC 解析マニュアルフェーズ I 基礎編 | 秀和システムトレーディング |
| フェーズ II 探究編 | 秀和システムトレーディング |
| フェーズ III サブシステム編 | 秀和システムトレーディング |
| ● OH / FM | 日本ソフトバンク |
| ● パソコン通信 Q&A | アスキー |

ご注意

- (1) 本書は著作権上の保護を受けています。本書の一部あるいは全部について（ソフトウェア及びプログラムを含む）、株式会社ビー・エヌ・エヌから文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。
- (2) 本書についての電話によるお問合わせには一切応じられません。質問等がございましたら往復はがき又は切手・返信用封筒を同封の上、弊社までお送り下さるようお願いいたします。

FM-7シリーズテクニカルノウハウ

FM-Techknow

FM-7/NEW7/77 & FM77AV 定価3,900円

著 者 阪井末幸 高橋暢生 梅野香織

昭和61年9月10日初版発行

昭和62年12月4日第3刷発行

発行人 樺島正博

発行所 株式会社ビー・エヌ・エヌ

千代田区麹町4-5 紀尾井町レジデンス5F(〒102)

電話 03-238-1321(営業) 03-238-1323(編集)

印刷所 瞬報社写真印刷株式会社

COPYRIGHT © 1986 BY Sueyuki Sakai

Printed in Japan

ISBN4-89369-006-10 C-3055

FM-7シリーズ テクニカルノウハウ

FM-Techknow

FM-7/NEW 7/77 & FM77 AV



株式会社ビー・エヌ・エヌ

定価3,900円