ADVANTEST®
株式会社アドバンテスト

取扱説明書

TR45102

バブル・メモリ・ドライバ

MANUAL NUMBER 45102 0B 708

関連マニュアル一覧表

番号	名称	備考
TR4511	シンセサイズド・シグナル・ソース取扱説明書	
TR4512	シンセサイズド・シグナル・ソース取扱説明書	
TR45102	外部キーボード取扱説明書	

目次

1. はじめに

2. パネル面およびバブル・カセットの説明

3. 操作方法

4. ＴＲ４５１０２制御に使用するＴＲ４５１１／１２ ＢＡＳＩＣ ステートメントの文法と解説

5. GPIB制御コマンドの書式と機能

6. 性能諸元

目次

APPENDIX

1. はじめに

1.1　製品概要

TR45102 バブル・メモリ・ドライバは TR4511/12シンセサイズド・シグナル・ソース、オプション07、コントローラ機能の外部記憶装置です。記憶媒体として富士通㈱製のバブル・カセット（FBM-C128GA）を用いますので、耐環境性にすぐれた記憶装置となっています。バブル・メモリはフロッピー・ディスクや磁気テープと違い、モータ、ソレノイド、磁気ヘッド等の機構部品がなく、耐摩耗性に優れ、１つのメモリを半永久的に使用することができます。また、正面パネルに装備されているKEYBOARD端子にTR45103 外部キーボード（JIS 配列のフル・キーボード）を接続すればプログラム作成が容易になり、作業性が向上します。

本器背面パネルにはTTL I/O（TTLレベル入力ポート×８、オープン・コレクタ出力ポート×８）が用意され、同軸リレーの制御や TTLレベル信号の入力等がGPIBシステムで実現されます。（TTL I/O はGPIBによる外部制御が可能です。）

TR45102 にはバブル・カセットのドライブが２ドライブ装備され、プログラム・ファイル、データ・ファイルを作成記憶することができます。バブル・カセット１個は１Mbitの記憶容量を持ち、最大64ファイルまで記憶することが可能です。

データ・ファイルにはランダム・アクセス・ファイルとシリアル・アクセス・ファイルがあり、用途に応じて使い分けることができます。

TR45102 のすべての機能はGPIBにより制御されますので、単に TR4511/12のアクセサリにとどまらず、あらゆる分野への応用が期待できます。

1.2　外観チェックおよび付属品の確認

本器を受領されましたら、まず製品の外観を点検し、輸送中のきず、破損がないかを確認して下さい。

次に、以下の表によって標準付属品をチェックし、数量および規格を確認して下さい。万一きず、破損、付属品の不足などがありましたら、最寄りの営業所または弊社CEセンタ（横浜営業所）へ連絡して下さい。所在地および電話番号は巻末に記載してあります。

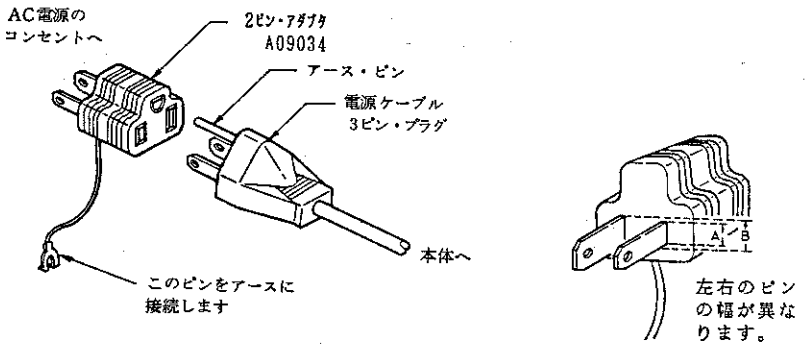
表 1-1　付属品一覧表

品　　名	型　　名	数量	備　　考
電源ケーブル	MP-43A (DCB-DD0717A-1)	1	
ヒューズ	 MDX-1.25A (DFT-AG1R25A-1) MDL-0.6A (DFT-AHR6A-1)	2	出荷時に添付されるヒューズは発注時の仕様により規格が異なります。 AC100V仕様の場合 AC200V仕様の場合
バブル・カセット	A09503 (ESM-000271-1)	1	富士通 FBM-C128GA
GPIBケーブル	408JE-1P5 (DCB-SS1076×01-1)	1	50cm (TR4511 とTR45102 の接続用)
取扱説明書	J45102（本書）	1	

1.3　電源ケーブルとヒューズ

1.3.1　電源ケーブルのプラグとアダプタ

(1)　電源ケーブルのプラグは3ピンになっており、中央の丸い形のピンがアースになっています。
プラグにアダプタを使用してコンセントに接続するときは、アダプタから出ているアース線〔図 1-1(a)〕、または本体背面パネルにあるアース端子のどちらかを、必ず外部のアースと接続して大地に接地して下さい。
付属のアダプタA09034は、電気用品取締法に準拠しています。
このA09034は、〔図 1-1(b)〕に示すように、アダプタの2本の電極の幅A、Bが異なりますので、コンセントに差込むときは、プラグとコンセントの方向を確認して接続して下さい。A09034が使用するコンセントに接続できない場合は、別売品のアダプタKPR-13をお求め下さい。



(a)　(b)

図 1-1　電源ケーブルのプラグとアダプタ

1.3.2　電源ヒューズと電源電圧設定カード

(1)　電源ヒューズを交換する場合は、背面パネルのAC LINE コネクタから、電源ケーブルを外します。
次に、AC LINE コネクタの右側のヒューズ・ボックスのプラスチック・カバーを左にスライドさせます。中にヒューズと、FUSE PULL と書かれたレバーが納められています。レバーを手前に引出すと、ヒューズが取外せます(〔図 1-2〕参照)。
使用する電源の電圧に合わせて、下記の規格のヒューズと交換して下さい。

AC100V±10%, AC120V±10% …… MDX-1.25A (スロー・ブロー・タイプ)
AC220V±10%, AC240V+4%-10% …… MDL-0.6A (スロー・ブロー・タイプ)

(2)　本器は出荷時に、ユーザの指定した電源電圧に設定されています。設定された電源電圧は、ヒューズの下にあるカードに表示されています。設定された電源電圧と異なる電圧で本器を動作させたい場合は、このカードの向きを変更します。以下にその手順を示します。

①　FUSE PULL レバーを手前に引出します。
②　ヒューズを取出します。
③　電圧を表示したカードを先の細いラジオ・ペンチなどで引出します。
④　カードの向き、表裏を変えて、使用する電源電圧が上面の左側に来るようにして、カードを元の位置に差込みます。
⑤　FUSE PULL レバーを元の位置に戻します。

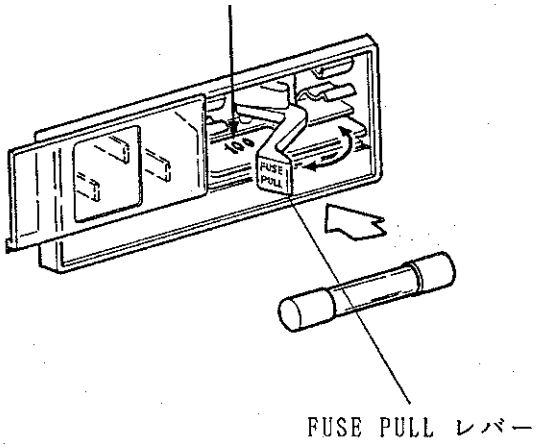


図 1-2　電源ヒューズの交換方法と電源電圧設定カードの設定変更

1.4　使用前の準備および TR4511/12との接続方法

本器は TR4511/12とGPIBを接続して使用します。付属のGPIBケーブル、408JE-1P5 で接続します。

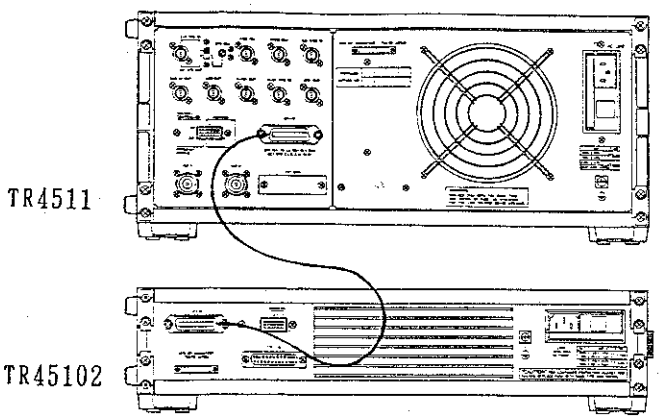


図 1-3a TR4511と45102 の接続

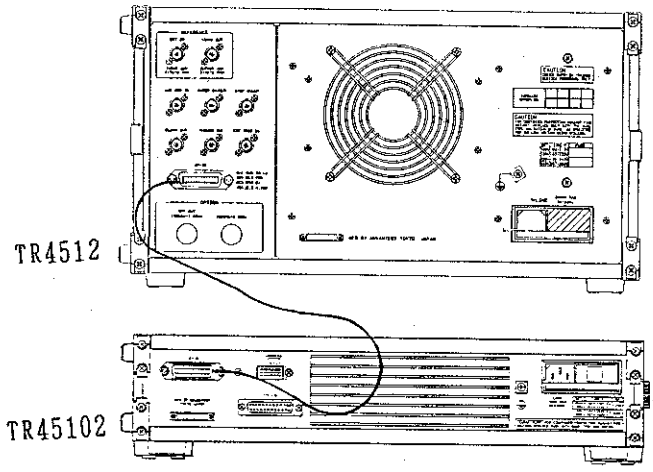


図 1-3b TR4512と45102 の接続

電源投入時の不確定動作によるメモリ内のデータ破壊を防ぐためにバブル・カセットをドライブに挿入した状態での電源ON/OFFは避けて下さい。（本器は万全を期して、電源投入時、接断時、あるいは停電等による電源断時には内部回路を停止する様な設計が施されていますが、万が一の事故を防ぐためにも必ず上記の旨を守って下さい。）
もし上記の理由により記憶データが破壊されても当社では責任を負いかねます。
また不可抗力（停電、火災等）によりバブル・カセットの異常についてもその損害に関する保証はいたしかねますのであらかじめ御了承願います。

1.5　TR45103 キーボードの接続

①　電源投入前にTR45103キーボードと本器を接続します。

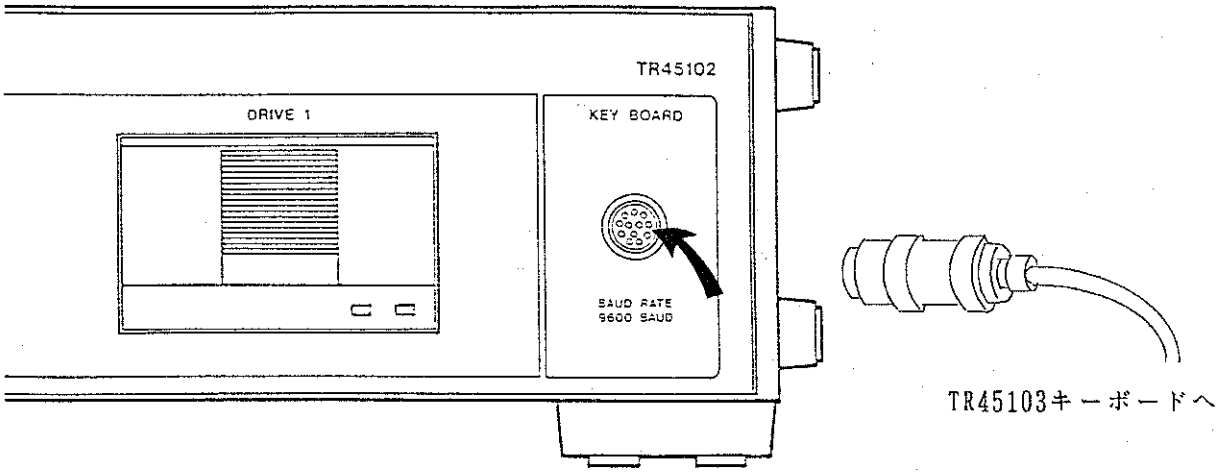


図 1-4　TR45103 キーボードの接続

②　TR45102 の電源を投入し、 TR4511/12をプログラム・モードにします。(TR4511/12 の PROG / EXT を押して下さい。)

TR4511/12 の管面には下図のような表示が出力されます。

```
*** GPIB CONTROLLER V2.0 ***
```

(もし既にプログラムが記憶されている場合は、上記の表示に続いてプログラム・リストが表示されます。)

③　この状態で下のキー操作を行なうと、外部キーによる入力が可能になります。

SHIFT　LABAL　PROG / EXT

なお外部キーボード動作中にコネクタを外すと誤動作することがあります。コネクタを着脱する際には電源を切って下さい。

2. パネル面およびバブル・カセットについて

2.1 正面パネル

① 電源スイッチ
　このスイッチを押し込むと電源がONになります。電源電圧は標準でAC100 V50/60Hzに設定されていますがオプションにて、AC120 V (±10%), AC 220 V(±10%), AC240 V (+4%, -10%) に設定することが可能です。標準電圧以外で本器を使用するときは、商用電源の電圧と本器の設定電圧を確認した上でコンセントに電源ケーブルを差し込んで下さい。
　もし、電圧設定に誤りがありますと、重大な故障につながりますので充分に注意して下さい。

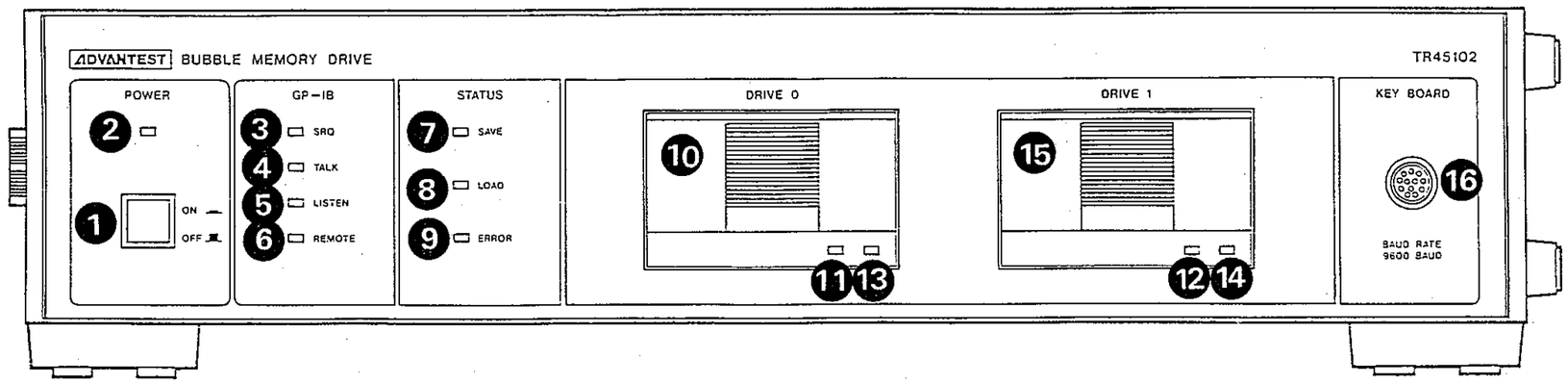


図 2-1　TR45102 正面パネル

② 電源パイロット・ランプ
　電源ON時に点燈します。

③ GPIB SRQランプ
　本器がSRQを発生し、シリアル・ポールを要求しているときに点燈します。

④ GPIB TALK ランプ
　本器がトーカ・アクティブ・ステートに設定されている間このランプが点燈します。（トーカ・アクティブ・ステートとは本器がGPIB上にデータ出力可能な状態に設定されている状態です。）

⑤ GPIB LISTEN ランプ
　本器がリスナ・アクティブ・ステートに設定されている間、このランプが点燈します。（リスナ・アクティブ・ステートとは本器がGPIB上のデータを受信可能な状態に設定されている状態です。）

⑥ GPIB RBMOTE ランプ
　本器がGPIBより制御されるとこのランプが点燈します。ただし、トーカ・アクティブ・ステートでは必ずしもREMOTEランプが点燈するとは限りません。REMOTEランプはGPIBの単線信号RENが真（Lowレベル）のとき、コントローラから本器のリスナ・アドレスを受け取った時に行なわれます。

⑦ STATUS SAVE ランプ
　バブル・カセットに対して書込みを行なった時に点燈します。特にSAVEステートメント実行時に点燈するだけでなく、次のステートメントによっても点燈します。
　　CREATE, COPY, INITIALIZE, OUTPUT, PROTECT, SAVE, DELETE

⑧ STATUS LOAD ランプ
　バブル・カセットに対して読出しを行なった時に点燈します。LOADステートメント実行時の他に、以下のステートメントを実行しても点燈することがあります。
　　COPY, CREATE, DELETE, DIR, ENTER, INITIALIZE, LOAD, OPEN, OUTPUT
　　PROTECT, SAVE

⑨ STATUS ERRORランプ
　本器に対して誤った操作を行なったり、動作中に異常を検出した場合に点燈します。

⑩ ドライブ0
　バブル・カセットを駆動するホルダです。フタを開けて中にバブル・カセットを挿入します。バブル・カセット挿入時は最後まで確実に押込んで、必ずフタを閉じた状態で使用して下さい。（フタが開いたままですと、バブル・カセットに対する読み書き動作が行なえません。）
　またバブル・カセットを取出すときは、ホルダ内のイジェクト・ボタンを押して下さい。

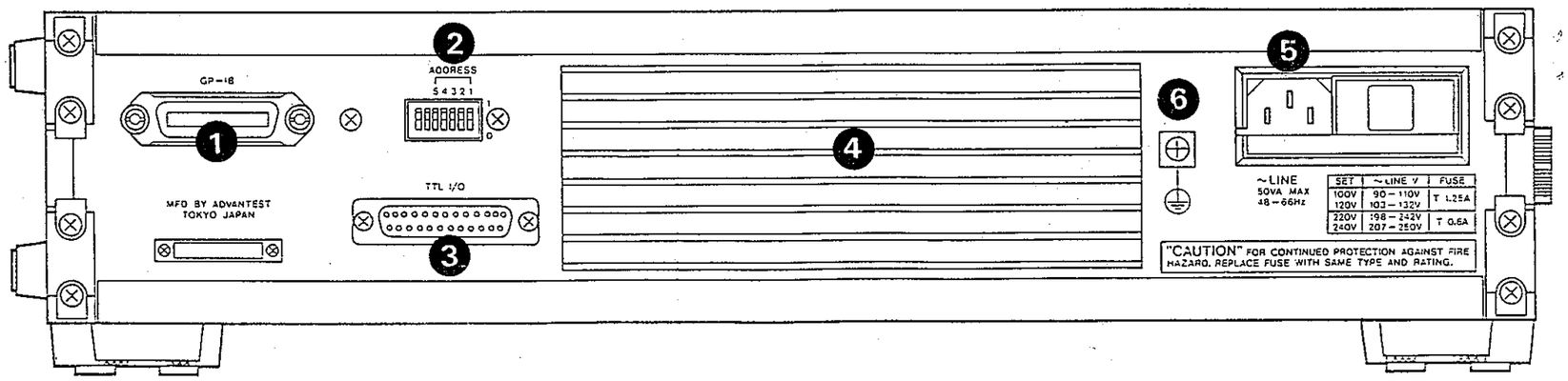
⑪⑫ライト・プロテクト表示ランプ
　ハード的にライト・プロテクトされたバブル・カセットを挿入すると点燈します（　緑色発光）。PROTECT ステートメントによるライト・プロテクトでは点燈しません。

⑬⑭BUSY表示ランプ
　バブル・カセットがアクセスされている間点燈します。

⑮ ドライブ1
　バブル・カセットを駆動するホルダで⑩と同様に扱って下さい。ドライブ0とドライブ1は、外部記憶装置用のステートメント（SAVE, LOAD, CREATE …… 等）のドライブ番号（0または1）により選択することができます。

⑯ KEYBOARD接続端子
　TR45103 キーボードを接続するコネクタです。キーボードとのデータ通信はシリアルで行ないます。KEYBOARD接続端子の仕様は〔6.1 性能諸元〕を参照して下さい。
なお、TR45103の電源（+5V）は本端子から供給します。

2.2 背面パネル



① GPIBコネクタ
　GPIB接続コネクタです。IEEE488-1978に準拠しています。この端子の機能は以下のとおりです。
　　SH1, AH1, T6, L4, SR1, RL1, PP0, DC0, DT0, C0, E1

② ADDRESS スイッチ
　1～5のスイッチによりGPIBのアドレスを設定します。1～5以外のスイッチは常に0に設定しておいて下さい。このスイッチを1に設定した場合、通常の動作と異なった動作をする場合があります。

③ TTL I/O
　TTL レベルの入出力ポートで、入力×8、出力(ｵｰﾌﾟﾝ･ｺﾚｸﾀ) ×8が用意されています。TTL I/O の制御はGPIBより行ないます。

④ 放熱フィン
　放熱用のフィンです。動作中は放熱効率をよくするために周囲に物を近づけない様にして下さい。

⑤ AC LINE コネクタ
　電源ケーブル用のコネクタで、ヒューズ・ホルダ兼用になっています。電源電圧は標準ではAC100V (50/60Hz) になっていますが、オプションによってAC120V, AC220V, AC240V (いずれも50/60Hz) が選択できます。ヒューズは電源ケーブルを抜かないと取外しができないようになっています。また電源ヒューズは電源電圧によって規格が異なりますので注意して下さい。(〔1.3 節〕参照)

⑥ 接地用端子
　アース用の端子です。電源ケーブルは3ピン構造で中央の丸いピンがアースとなりますが、2ピン用のアダプタを使用する場合はアダプタの接地用ケーブル、またはこの接地用端子を大地接地して下さい。

2.3　バブル・カセットについて

(1)　カセット・ケース

バブル・カセットは出荷時に帯電防止処理がされたプラスチックのケースに入っています。ケースは下図のように開けます。

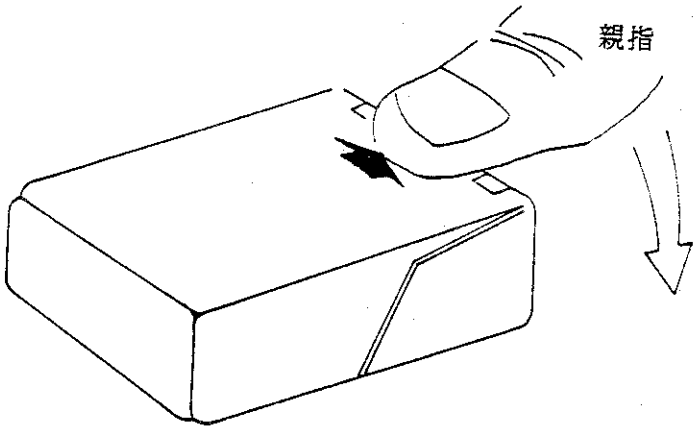


図 2-3　バブル・カセットの開け方

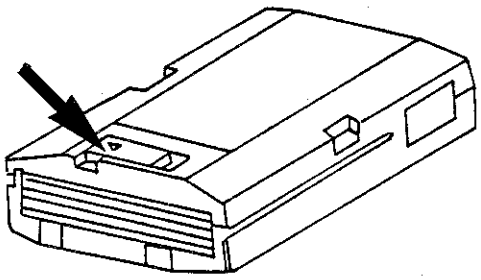
バブル・カセットの保管

カセットを保管する際には、出荷時のプラスチック・ケースに入れて下さい。

(2)　出荷時の記憶情報

・不良ループ情報は書き込んであります。
・データ・エリア（サブ・ページも含む）はすべてクリア（“0”）されています。

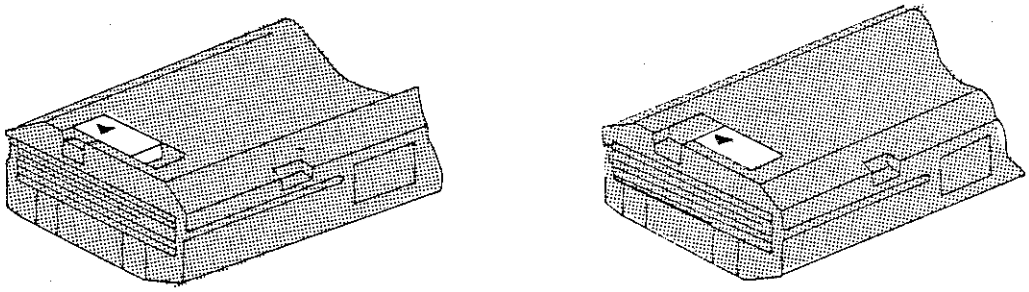
(3)　ライト・プロテクト



バブル・カセットには誤った書き込みによる記憶情報の破壊を防止するため、スライド・タイプのライト・プロテクト・キーがついています。

図 2-4　ライト・プロテクト・キー

注）　本節の図 2-3，4，5，7 のバブル・カセットの図は富士通バブル・メモリ取扱説明書(JBO-0140-F2) より転載させて頂きました。



ライト・プロテクトしていない状態　　ライト・プロテクトした状態

図 2-5　ライト・プロテクト

ライト・プロテクトされたバブル・カセットをホルダ・ユニットに挿入すると、ホルダ・ユニット前面の緑色のLEDが点燈します。

(4)　ホルダ・ユニットの挿入方法

① 指の先でホルダ・ユニットの扉を開けます。
② ライト・プロテクト・キーを下にしてカセットをホルダ・ユニット内に完全に挿入します。
③ 扉を閉じて下さい。扉が閉じない場合にはカセットを強めに押し込んで下さい。
④ 扉を閉じることでカセットは動作可能状態になります。

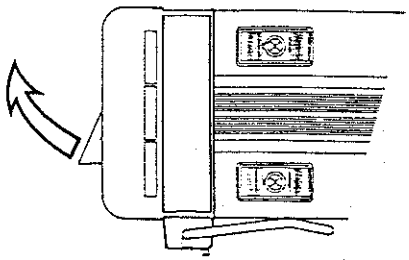


図 2-6　ホルダ・ユニットの挿入

注　意

アクセス中に扉を開くと、カセットの動作はPF信号を入力したのと同じ様に緊急停止します。

(5)　ホルダ・ユニットの取り出し方法

① 扉をあけます。
② イジェクトボタンを押しますとカセットが途中まで出て来ますので、カセットを指ではさんでホルダユニットから取り出して下さい。
③ 扉を閉じて下さい。

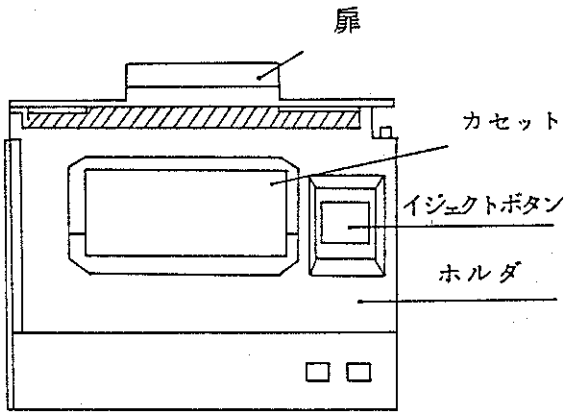


図 2-7　扉を開けた状態の正面図

3. 操作方法

3.1 バブル・カセットのイニシャライズ

初めてバブル・カセットを使用する場合は、まずバブル・カセットのメモリ内容を初期化しなくてはなりません。初期化されていないバブル・カセットには、TR45102 で扱うディレクトリ情報が書込まれていないため、SAVE、LOAD等の操作を行なっても正常に動作しません。ディレクトリとはバブル・カセットの内容を示す目録のようなもので、現在の使用状況（記憶されているファイルの数、種類等）を記録したものです。
バブル・カセットを初期化する場合はINITIALIZEステートメントを用います。

```
INITIALIZE␣n : Volumename↴
```

↴ 記号 ； TR4511/12 の ENTERキー, □ ENTER またはTR45103 の RETURNキー RETURN を押すことを意味しています。

␣ 記号 ； TR4511/12 のCRT 上の表示やプログラムの表記において1 文字分のスペースを意味しています。

n:はドライブの番号を示すもので、0 番のドライブであれば0:、1 番のドライブであれば1:とします。ただし0 番のドライブを使用する場合は省略可能です。
Volumenameはバブル・カセット個々に与えられる名称で、DIR ステートメントを実行したときに表示されるだけです。その他の動作には何ら影響を与えるものではありませんので、必要と思われない場合は省略することができます。

注）ステートメントの実行およびプログラミングに関してはTR4511取扱説明書、またはTR4512プログラミング・マニュアルを参照して下さい。

例）
```
INITIALIZE
INITIALIZE␣ABC
INITIALIZE␣1:VOL_01
```

なお、Volumenameは10文字以内とし、頭文字は必ず英大文字(A, B, C ～ Z) を用いて下さい。詳細は〔3.6 節ファイル名について〕を参照して下さい。

3.2　プログラムの記憶（SAVE）

TR4511/12 オプション07コントローラ機能にて作成したプログラムをバブル・カセットに記憶させることができます。プログラムを記憶させる場合はSAVEステートメントを用います。

```
SAVE␣n:Filename↵
```

n:はドライブの番号を示すもので、バブル・カセットが0番のドライブに入っている場合は0:、1番のドライブに入っている場合は1:とします。但し0番のドライブを使用する場合は特に指定する必要はなく、省略することが可能です。Filenameはプログラムを記憶するファイルの名称で、10文字以内の文字列を用います。（但し1文字目は英大文字を用います。）

例）PROG1 というファイル名を用いて0番のドライブにあるバブル・カセットにプログラムを記憶させる。

```
SAVE␣0:PROG1
```

または

```
SAVE␣PROG1
```

既に、SAVEしてあるプログラム・ファイル名を用いて再度SAVEを実行しますと次のメッセージが表示され処理を中断します。

```
Already exists same file. RESAVE ?
```

このメッセージは、同じ名前のファイルが既に記憶されているが、そのファイルを消して新たにSAVEしなおしてもよいかどうかを尋ねている訳です。古いファイルを消してファイルを更新してもよい場合は、この直後にTR4511/12 のパネルキーから

[MHz] Y　[ENTER]　と入力します。

TR45103から [Y] ↵ とキー入力することでも同じ結果になります。

古いファイルを消去したくない場合は、単に [ENTER] のみを押せばSAVEを中止します。

なおライト・プロテクトされたバブル・カセットあるいはプログラム・ファイルに対してSAVEを行なってもSAVEされずにエラーが発生します。(Write protected)
そのような場合には、ライト・プロテクトを解除したのち再度SAVEを行なって下さい。プロテクトに関しましては〔3.14　ファイルのプロテクト〕を参照して下さい。

3.3　プログラムの読出し(LOAD)

前節で述べましたSAVEステートメントによって記憶したプログラムを読出す場合、LOADのステートメントを用います。

```
LOAD␣n:filename↵
```

n:およびfilenameに関しましてはSAVEステートメントの場合と全く同様に扱って下さい。

例）0番のドライブ内のバブル・カセットに記憶されている、PROG1 という名のプログラム・ファイルを読出す。

```
LOAD␣0:PROG1
```

または

```
LOAD␣PROG1
```

LOADステートメントを実行しますと、TR4511/12 にプログラムが読込まれます。（それまでTR4511/12 にプログラムされていたテキストは消えてしまいますので注意して下さい。）
LOADステートメントはプログラム中で使用することもできます。
今、以下に示す2つのプログラムをそれぞれPROG1, PROG2というファイル名で記憶してあるとします。

プログラム 1

```
10  SCLEAR
20  CURSOR(10,5)
30  DISP␣"EXECUTE␣PROGRAM1"
40  WAIT␣2000
50  LOAD␣PROG2,10
60  END
```

上記プログラムをSAVE PROG1↵でバブル・カセットに記憶する。

プログラム 2

```
10  CURSOR(10, 5)
20  DISP␣"EXECUTE␣PROGRAM2"
30  WAIT␣2000
40  LOAD␣PROG1, 20
50  END
```

上記プログラムをSAVE PROG2↵でバブル・カセットに記憶する。

プログラム1 をLOADし、RUN します。すると、行番号50のところでPROG2 というプログラム・ファイルを読み込み、10番の行から実行します（このときプログラム1 はTR4511からは消えてしまいます）。すなわちプログラム2 が実行される訳です。さらにプログラム2 の40番の行でPROG1 というプログラム・ファイルを読み込み20番の行から実行します（このときプログラム2 はTR4511/12 からは消えてしまいます）。

例からもお判りになると思いますが、プログラム中で

```
LOAD␣n:filename, m
```

としますと、n:のドライブにあるfilenameという名のプログラム・ファイルを読み込み、m 番の行から実行を開始します。

なおLOADステートメントを実行した時に、該当する名前のファイルがディレクトリ中に見つからない場合は"File not found"メッセージを表示して、処理を終了させます。

また、既存のファイルでも、そのファイルがリード・プロテクトされていると、"Read protected"のメッセージを表示して処理を終了させます。いずれの場合もプログラムのLOADは行なわれずに終了します。

3.4　ディレクトリの表示

バブル・カセットのディレクトリに登録されたファイル(SAVE, CREATE ステートメントにより作成されたもの）の一覧表をTR4511/12 の画面上に表示させることができます。ディレクトリ一覧表を表示させるにはDIR ステートメントを実行します。

```
DIR␣n:filename↵
```

n:はバブル・カセットのドライブ番号で、0番のドライブにあるバブル・カセットのディレクトリを表示させるときは、0:、1番のドライブの場合は1:とします。（但し0:は省略できます。）filenameは特定のファイルに関するディレクトリを表示させたいときに用い、すべてのファイル・ディレクトリを表示する場合には必要ありません。

```
DIR
```

を実行しますと、0番のドライブにある、バブル・カセットのディレクトリをTR4511/12管面に表示します。表示例を以下に示します。

```
DIRECTORY   OF  VOL01
0125/2020
FILE NAME    BLKS  LENGTH  TYPE  PRO
----------   -----  -----  ----  ---
PROG1        00001  00512  PROG
PROG2        00001  01020  PROG
Data         00256  00010  RAND  W
Advan        01000  00002  RAND
Comp         00001  01024  SER1
```

ディレクトリ表示の詳細につきましては〔4.3-(6)　DIR 〕を参照して下さい。

ディレクトリ表示が画面一杯になりますと、画面は自動的にスクロールされ順次、続くディレクトリ（ファイル名）が表示されていきます。従って記憶されているファイルが多すぎて一画面に表示しきれないような場合ですと、最初に表示されたファイル名はスクロールによって消されてしまい、結局、全てのディレクトリを見ることができなくなってしまいます。このような場合はfilenameによる指定を行なうと便利です。

filenameはSAVEやLOADで用いたものとほぼ同等の扱い方をしますがDIR ステートメントでは省略指定が可能です。

例えば上記の例に示されるようなディレクトリがあった場合、

```
DIR␣PROG1
```

としますと、PROG1 と名のファイルのみ表示されます。

```
DIRECTORY  OF  VOL01
0125/2020
FILE NAME   BLKS  LENGTH  TYPE  PRO
----------  -----  -----  ----  ---
PROG1       00001  00512  PROG
```

次に

DIR␣P*

としますと

```
DIRECTORY  OF  VOL01
0125/2020
FILE NAME   BLKS  LENGTH  TYPE  PRO
----------  -----  -----  ----  ---
PROG1       00001  00512  PROG
PROG2       00001  01020  PROG
```

となり、頭文字P で始まる名前のファイルのみディレクトリ表示されます。ファイル名の省略は以下のような方法も可能です。

PROG* ……… PROGで始まるファイル名すべて

このようにDIR ステートメントでファイル名を指定すると、ディレクトリの検索が楽になります。

3.5 ファイルのコピー(COPY)

既に記憶されているファイルを別のファイルして複製して、内容を少し変更して使用したい場合、あるいは他人のバブル・カセットにあるファイルを自分のバブル・カセットに複製したい場合がよくあります。このような場合にはCOPYステートメントを使用します。

```
COPY␣n:filename 1␣TO␣n:filename 2↵
```

n:はソースとなるファイルが存在するドライブ番号で、filename 1はソース・ファイルの名前です。m:はディスティーネーションとなるファイルのドライブ番号を示し、filename 2はディスティネーション・ファイルの名前です。n:とm:およびfilename 1とfilename 2は同一のものでも、異ったものでもかまいません。但しn とm が同一の場合はfilename 1とfilename 2は異なるものでなければなりません。n とm は省略時にはいずれも0として扱われます。filename 1とfilename 2が同一の場合はfilename 2を省略することができます。

例）0番のドライブにあるFileA という名のファイルを、同じく0番のドライブにFileBという名でコピーする。

```
COPY␣0:FileA␣TO␣0:FileB
```

または

```
COPY␣FileA␣TO␣FileB
```

例）0番のドライブにあるPROG1 という名のプログラム・ファイルを1番のドライブに同一のファイル名で記憶させる。

```
COPY␣0:PROG1␣TO␣1:PROG1
```

または

```
COPY␣PROG1␣TO␣1:
```

COPYステートメントでもDIR ステートメント同様アスタリスク(*) によってファイル名を省略することができます。

```
COPY␣0:A*␣TO␣1:
```

とした場合、0番のドライブにあるファイルで、頭文字がAで始まる全てのファイルを1番のドライブに既存ファイルと同一名称でコピーします。

例）1番のドライブにPROG1, PROG2, PROG3 という名の3つのファイルがある。3つとも0番のドライブにコピーする。

```
COPY␣1:PROG*␣TO␣0:
```

例）０番のドライブにある全てのファイルを１番のドライブにコピーする。

```
COPY␣0:*␣TO␣1:
```

または

```
COPY␣*␣TO␣1:
```

なおCOPYステートメントを実行する場合、コピー先のバブル・カセットにディスティネーション・ファイルと同名のファイルが存在する場合、エラー(Already exists same file.)になり、処理を中断します。
COPYステートメントによるファイルのコピーは後節で述べるデータ・ファイルに対しても同様に行なうことができます。

3.6　ファイル名について

ファイル名は10文字以内の文字列で、以下に示す文字が使用できます。但し頭文字が必ず英大文字(A～Z)である必要があります。(INITIALIZE ステートメントでのボリューム名についても同じです。)

ファイル名に使える文字記号

```
␣!#$%&()+-,/0123456789=@
ABCDEFGHIJKLMNOPQRSTUVWXY
Z〔〕＼＿ abcdefghijklmn
opqrstuvwxyz
```

3.7　ファイルの保護について

バブル・カセットに記憶したファイルを他人に使用されたくない場合に、特別な保護コードを指定しない限りファイルをアクセスできなくさせる方法があります。
保護コードはSAVEステートメント、COPYステートメント、およびCREATEステートメント（後述）でファイル作成時に付与します。保護コード(Security Code) はファイル名に続けて以下に示す形で指定します。

```
filename＜保護コード（２文字の文字列）＞
```

例としてSAVEステートメント場合を示します。

```
SAVE␣n:filename＜保護コード＞↵
```

保護コードは２文字以内の文字列で、ファイル名に使用できる文字（〔3.6節〕参照）の内から選びます。
SAVE, COPY, CREATEステートメントで保護コードを指定しますと、バブル・カセットのディレクトリにファイル名と同時に登録され、以後いかなる場合においても、保護コードを指定しない限りそのファイルをアクセスすることはできません。保護コードはDIR ステートメントによるディレクトリ表示では表示されませんので秘密性を保持することができる反面、個人的にメモをとる等して明確にしておかないと、二度とそのファイルはアクセスできなくなります。SAVEステートメントで保護コードを指定しますと、LOADステートメントでも保護コードを指定しなくてはなりません。

```
LOAD␣n:filename＜保護コード＞
```

CREATEステートメントで保護コードを指定しますと、OPENステートメント（後述）でも保護コードの指定が必要になります。

```
OPEN n:filename＜保護コード＞␣TO␣#m
```

COPYステートメントではソース・ファイルとディスティネーション・ファイルの両方に保護コードが付きますので少し複雑になります。場合に分けて以下に説明します。

・ソース・ファイルに保護コードが指定されていない場合
　ディスティネーション・ファイルに保護コードを指定することができます。

```
COPY␣n:filename 1␣TO␣m:filename 2＜保護コード＞
```

・ソース・ファイルに保護コードが指定されている場合で、ディスティネーション・ファイル名を指定した場合。
　ソース・ファイルに指定されている保護コードを付け、ディスティネーション・ファイルにも必要に応じて保護コードを指定します。但しソース・ファイルとディスティネーション・ファイルの保護コードは同一である必要はありません。

```
COPY␣n:filename 1＜保護コード＞␣TO␣m:filename 2
COPY␣n:filename 1＜保護コード＞␣TO␣m:filename 2＜保護コード＞
```

・ソース・ファイルに保護コードが指定されていて、ディスティネーション・ファイル名をアスタリスク(*)によって省略した場合。
　ディスティネーション・ファイルにはソース・ファイルと同一の保護コードが付加されます。

　　COPY␣n:filename<保護コード>␣TO␣m:*

・ソース・ファイルをアスタリスク(*)で省略してコピーしようとした場合

　　COPY␣n:*␣TO␣m:

　ソース・ファイルで保護コードが指定されているものはコピーされません。保護コード指定のないファイルのみコピーされます。

なお既にファイルの記憶されているバブル・カセットをINITIALIZEステートメントで初期化しますと、ファイルの保護コードの有無にかかわらずバブル・カセットのディレクトリはクリアされてしまいます。重要なファイルを記憶させてあるバブル・カセットはハード的にライト・プロテクトすることをお勧めいたします。(〔2.3.3 項〕および〔3.14節〕を参照して下さい。)

3.8　ファイルの抹消(DELETE)

バブル・カセットに記憶させたファイルが不要のものとなり消去してしまいたい場合、ファイル名を指定することで特定のファイルのみを抹消することができます。INITIALIZEステートメントではバブル・カセット内のディレクトリ全てがクリアされてしまいますが、DELETEステートメントを用いますと、ファイル１つ１つを選んで抹消することができます。

```
DELETE␣n:filename↵
```

n 番のドライブにあるfilenameという名のファイルを抹消します。もし抹消したいファイルに保護コード（〔3.7節〕参照）が指定してある場合はDELETEステートメントでも保護コードを指定しなくてはなりません。

```
DELETE␣n:filename<保護コード>↵
```

またライト・プロテクトされたバブル・カセットあるいはファイルはプロテクトを解除しないとDELETEステートメントは受付けられずエラー終了します。(Write protected)

例）１番のドライブにあるFileA という名のファイルを抹消する。

```
DELETE␣1:FileA
```

例）０番のドライブにあるFileB という名のファイルを抹消する。
但しこのファイルには%1という保護コードが指定されている。

```
DELETE␣0:FileB<%1>
```

または

```
DELETE␣FileB<%1>
```

複数のファイルを一度に抹消したい時は、アスタリスク(*) によるファイル名の省略という手段があります。（ただしこの方法は抹消したいファイルの頭文字が等しいもののみ有効です。）

例）０番のドライブにPROG1, PROG2という２つのファイルがあり、この両方を抹消する。

```
DELETE␣0:PROG*
```

または

```
DELETE␣PROG*
```

上の例の場合、

```
DELETE␣P*
```

としてもPROG1 およびPROG2 は抹消されますが、他に頭文字が"P" で始まるファイル名が存在しますと、そちらも同時に抹消されてしまいます。DELETEステートメントでファイル名の省略指定を行なうときは、他のファイルに影響をおよぼさないよう注意して下さい。

3.9　データ・ファイル

これまでの説明は主にプログラム・ファイルに関する操作方法について述べましたが、TR45102 ではプログラム・ファイルの他にデータ・ファイルを扱うことができます。データ・ファイルとはプログラム中で演算した結果や文字列データ等を記憶させるファイルのことで、プログラムを実行させながら、データを参照したり、新しく書込んだりすることができます。

データ・ファイルは１つのファイル中に複数のデータを記憶させることが可能で、数値データと文字列データを１つのファイルの中に混在させることもできます。さらにデータ・ファイルにはランダム・アクセス・ファイルとシリアル・アクセス・ファイルの２種類があり、扱うデータをいかにアクセスするかでいずれのタイプのデータ・ファイルを使用するかを選択します。

以下にランダム・アクセス・ファイルとシリアル・アクセス・ファイルの特長を説明します。

3.9.1　ランダム・アクセス・ファイル

ランダム・アクセス・ファイルはファイル中の任意のデータを読み取ったり書込んだりすることができます。しかしファイルの構造上データ長は固定となり決ったフォーマットのデータしか扱うことができません。

データ長とは扱うデータのバイト数（文字数）を意味し、TR4511/12 の数値データは８バイトと内部的に決っていますが文字列の場合は特に決っていません。例えば１文字の文字列データを扱う場合データ長は１バイト、15文字であれば15バイトというように、プログラムの目的によってデータ長は異なります。

TR4511/12 ではバイナリ・データは文字列と同等の扱い方をします。

文字列データのように可変長のデータを扱う場合は、データの中で、最もデータ長の大きいものを基準として扱かわなければなりません。

ただし最小データ長と最大データ長の差が大きくなればなる程、ランダム・アクセス・ファイルではメモリ空間的に無駄が多くなります。（このような場合はシリアル・アクセス・ファイルの方が有利になります。）

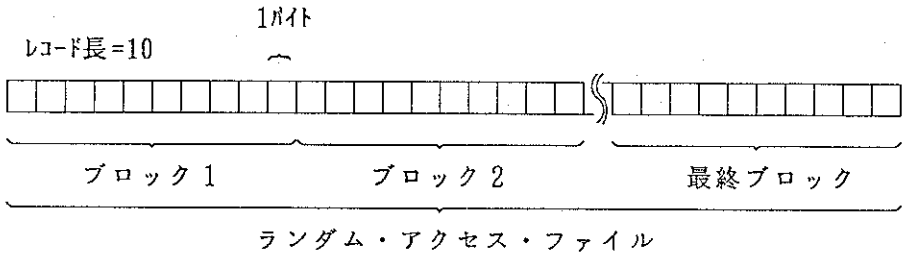
ランダム・アクセス・ファイルでは１データを１ブロックという単位に置換えて、ファイル全体をブロックに分割する構造を持ちます。１ブロックの長さはレコード長という単位で表わされデータ長と次の関係を持ちます。

レコード長＝データ長＋２（バイト）

従って数値データのレコード長は10バイト、文字列データのレコード長は最大文字数＋２ということになります。

データの種類	データ長	レコード長
数値データ	８バイト	10バイト
文字列データ	任意	最大文字数＋２

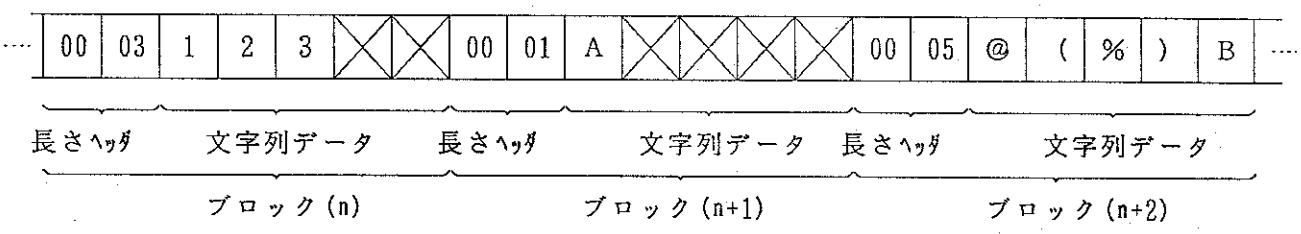
ランダム・アクセス・ファイルの構造を以下に図で示します。



ランダム・アクセス・ファイルは図からも明らかなように、ファイル全体をブロック単位で管理しているため、ブロックを指定することで任意のデータを即時アクセスすることができます。

もう少し詳しく理解するため、データを記憶させた場合の各ブロックの状態を図示しましょう。

レコード長を（データ長=5）とし、“123”、“A”、“@(%)B”という３つの文字列を記憶させたとします。



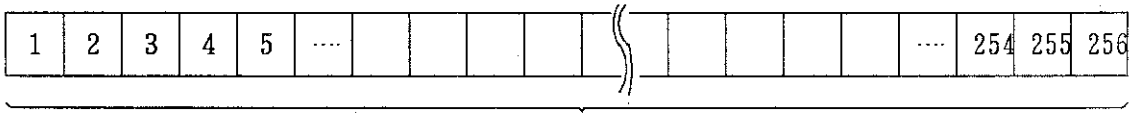
図で長さヘッダというものがありますが、これはそのブロックに記憶されているデータのデータ長を示しています。また各ブロック内に ⊠ の部分がありますが、ここに位置するデータは意味を持たないという意味です。（実際には何らかのでたらめなデータがあるのですが、TR4511/12 としては無効データとして処理しません。）また例の場合ですと、３データのうち無効データが６バイトあり、ほぼ１ブロック分のレコード長に相当する空間が無駄になっています。このようにランダム・アクセス・ファイルでは、任意ブロックのデータを直接アクセスできる反面、扱い方によっては非常に無駄の多いファイルとなってしまいます。ただし数値データは８バイトでフォーマットが固定されていますので、レコード長＝10としておけば無駄は発生しません。

3.9.2　シリアル・アクセス・ファイル

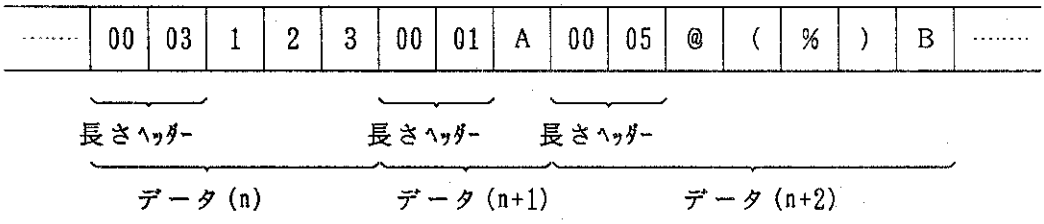
シリアル・アクセス・ファイルではファイル内をブロックで分割するという概念を持たず、ファイルの先頭から終りに向って順次書込みあるいは読出しすることしかできません。しかしファイル内をブロックで区切られていないため、空間的な無駄がなく可変長データを扱う場合に都合よくなっています。

シリアル・アクセス・ファイルはファイル全体を１つのレコード長で定義し、個々のデータに対するレコード長は書込むデータにより任意に決まります。１つ１つのデータに必要な長さはランダム・アクセス・ファイルと同様に数値データならば10バイト、文字列データならば文字数＋２となります。（ただしレコード長としては、データ長（＋２）の総和にさらに＋２しなくてはなりません。

レコード長＝256



ランダム・アクセス・ファイルの場合と同じように“123”、“A”、“@(%)B”の３つの文字列データをシリアル・アクセス・ファイルに記録した場合の例を下に示します。



図で示されるようにシリアル・アクセス・ファイルでは各データが隙間なくつめ込むことができ、ブロック単位でデータ長が制限されることもありません。

ただしシリアル・アクセス・ファイルにおいてもファイル全体の長さは有限ですので、ある程度データのフォーマットを考えておかないと、データがファイルから溢れてしまったり、ファイルの長さが大きすぎたりといった事が生じますので注意して下さい。

3.10　データ・ファイルの作成

データ・ファイルはデータを記録する前にあらかじめ作成しておかなければなりません。データ・ファイルの作成にはCREATEステートメントを用います。CREATEステートメントにはランダム・アクセス・ファイル作成用とシリアル・アクセス・ファイル作成用の2種類があり、それぞれ以下に示します。

```
ランダム・アクセス・ファイルの作成
    CREATE␣RAND␣n:filename, blocks, recode size

シリアル・アクセス・ファイルの作成
    CREATE␣SERI␣n:filename, recode size
```

blocksはファイル中のブロック総数を意味し、recode size はランダム・アクセス・ファイルの場合各ブロックのレコード長、シリアル・アクセス・ファイルの場合ではファイル全体のレコード長を意味します。
またデータ・ファイルに保護コードが必要な場合はSAVEステートメントと同様にfilenameに続けて保護コード2文字を指定します。

```
CREATE␣RAND␣n:filename<保護コード>, blocks, recode size
CREATE␣SERI␣n:filename<保護コード>, recode size
```

なおblocksおよびrecode size は省略指定することができます。省略時に設定される値を下の表に示します。

表 3-1　blocksおよびrecode size のデフォールト値

パラメータ	省略時の値	
	CREATE RAND	CREATE SERI
blocks	256	——
recode size	10	256

例）834 個の数値データを記録させるランダム・アクセス・ファイルを作成する。ドライブ番号は 1で、ファイル名はDataとする。

```
CREATE␣RAND␣1:Data, 834, 10
```

例）512 個の数値データを記録されるシリアル・アクセス・ファイルを作成する。ドライブ番号は 0で、ファイル名はA_DAT とする。

```
CREATE␣SERI␣A__DAT, 512*10
```

または

```
CREATE␣SERI␣0:A_DAT, 5120
```

上の例のようにblocks, recode size の値は数値演算式で表現することも可能です。

例）324 個の文字列データを記録するランダム・アクセス・ファイルを作成。ただしデータのうち最もデータ長の大きいものは15バイト。
0 番のドライブでSIRINGというファイル名を用いる。

```
CREATE␣RAND␣STRING, 324, 15+2
```

または

```
CREATE␣RAND␣0:STRING, 324, 17
```

3.11　データ・ファイルのアクセス方法(OPEN, CLOSE)

データ・ファイルはプログラム・ファイルのようにSAVE, LOADステートメントでファイルを直接アクセスすることはできません。データ・ファイルは必ずバッファを介してデータのやりとりを行ないます。バッファを用いる理由はファイル・アクセス時の時間的節約と複数のファイルを同時にアクセスした場合の混乱を防ぐためです。

バッファは全部で10あり、同時に10種類のデータ・ファイルをアクセスすることができます。バッファの名前は#0～#9で表わされ、これに必要なデータ・ファイルを割付けることでファイルのアクセスが可能になります。バッファとデータ・ファイルの割付けはOPENステートメントで行ないます。

```
OPEN␣#n␣TO␣m:filename
```

n はバッファの番号で０～９のうちのいずれかを用います。m:はドライバ番号です。もしデータ・ファイルに保護コードの指定がある場合はfilenameに続けて保護コードを指定します。

```
OPEN␣#n␣TO␣m:filename＜保護コード＞
```

例）Dataという名のデータ・ファイルがある。これに#2のバッファを割付ける。ドライブ番号は０とする。

```
OPEN␣#2␣TO␣0:Data
```

または

```
OPEN␣#2␣TO␣Data
```

OPENステートメントでバッファとデータ・ファイルを割付けると、その後はバッファに対してデータの入出力を行なうことで、データ・ファイルをアクセスしたことになります。（未割付のバッファに対してのデータの入出力は一切行なえません。）データの入出力はENTER ステートメント、OUTPUTステートメントによって行ないます。（後述）

バッファはバブル・カセットの１ページ（Appendix参照）に相当し、その長さは64バイトです。バッファに書込まれたデータは、バッファが一杯になった時、あるいはブロック指定があった場合、またはCLOSE ステートメントが実行された時にのみバブル・カセットに転送されます。従ってOUTPUTステートメントでバッファにデータを書込んでも、バブル・カセットには記録されていない状態があります。このような時に不用意にプログラムを終了させてしまうと、バブル・カセットにデータを書込んだつもりでも、実際には何も処理されていないようなことになります。（エラー発生によるプログラム終了は特に注意して下さい。）

このような事態を防ぐためには、プログラム終了前にバッファをクローズする必要があります。バッファをクローズしますと、バッファ内に残っているデータ全てを対応するファイルに転送し、バッファとファイルの割付けを解除します。もちろんデータを書込む毎にバッファをクローズしても構いませんが、一単クローズされたバッファは再度OPENステートメントによりファイルの割付けを行なわないと、アクセスできなくなりますのでプログラムが繁雑になってしまいます。特に必要がない限り、バッファのクローズはプログラム終了の直前でよいと思います。（ただし、長時間に渡り稼動させるようなプログラムで

は停電等のアクシデントを考慮し、適当な箇所でファイルのオープン、クローズを繰返すようにした方がよいでしょう。）

また既にOPENステートメントで割付けられているファイルとバッファを、再度OPENステートメントで割付けを行なっても、同様にバッファ内の残りデータはファイルへ転送されます。

バッファをクローズするにはCLOSE ステートメントを使用します。

```
CLOSE␣#n
```

n はバッファの番号です。複数のバッファを一度にクローズさせたい時は次のようにします。

```
CLOSE␣#ℓ, #m, #n, ……, #u
```

例）#3のバッファをクローズします。

```
CLOSE␣#3
```

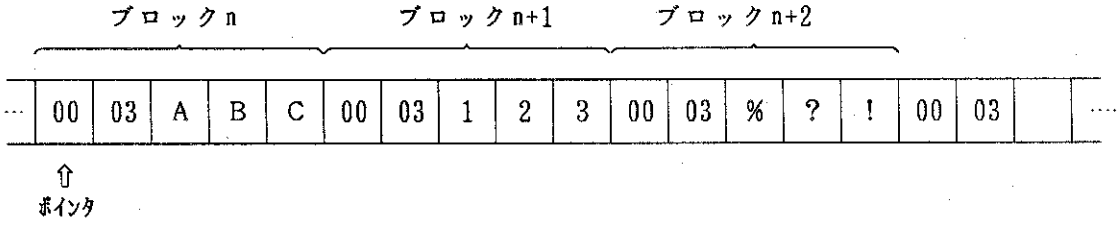
例）#0, #1, #2の３つのバッファを同時にクローズさせる。

```
CLOSE␣#0, #1, #2
```

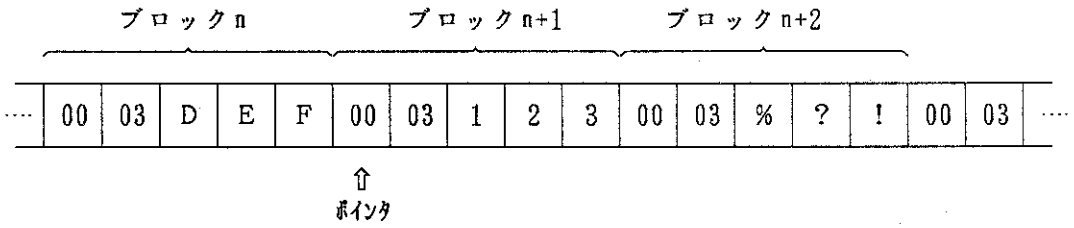
以上ファイルとバッファの割付け、および割付解除について説明しましたが、ファイルのアクセスに関する説明を以下にします。

１つのファイルの中には複数のデータが存在するので、ファイルをアクセスする場合、現在ファイルのどのあたりをアクセスしているのかを明確にしておかなければなりません。ファイル内のどの部分をアクセスしているかを示すものとしてポインタと呼ばれるものがあります。ポインタはTR45102 内部で管理されているもので、直接ユーザがポインタの示す値を参照することはできませんが、プログラムの進め方によりポインタの示す位置を知ることができます。

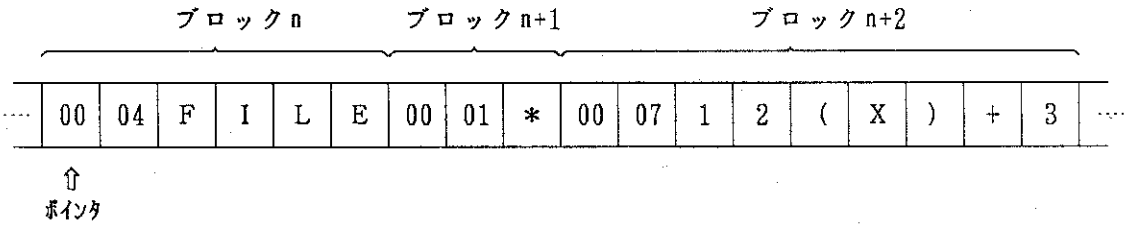
あるデータ・ファイルをアクセスした場合、特にことわりのない限りポインタはアクセスする毎に１つ先へ進みます。（１つ先に進むとは、ファイルの先頭から終端に向って１ステップ進むことを意味します。）いま仮に下図に示すようなランダム・アクセス・ファイルがあるとします。



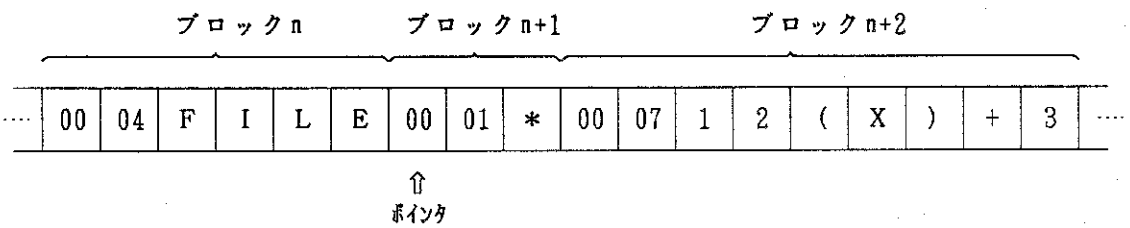
図ではポインタはブロックn の先頭を示しています。この状態でファイルをアクセスしますと、ブロックn に対してデータのアクセスが行なわれます。（実際にはバッファを介していますので、直接ファイルに対してアクセスすることはありませんが見かけ上の動作として理解して下さい。）例えば、図の例でデータを読出しますと、“ABC”という文字列が得られ、逆に“DEF”という文字列を書込みますと、ブロックn に既存データと入れ替え“DEF”が置かれます。そしてブロックn に対するアクセスが行なわれますとポインタは自動的にブロックn+1 の先頭に移ります。例として先の図の状態で“DEF”という文字列データを書込んだ場合を示します。



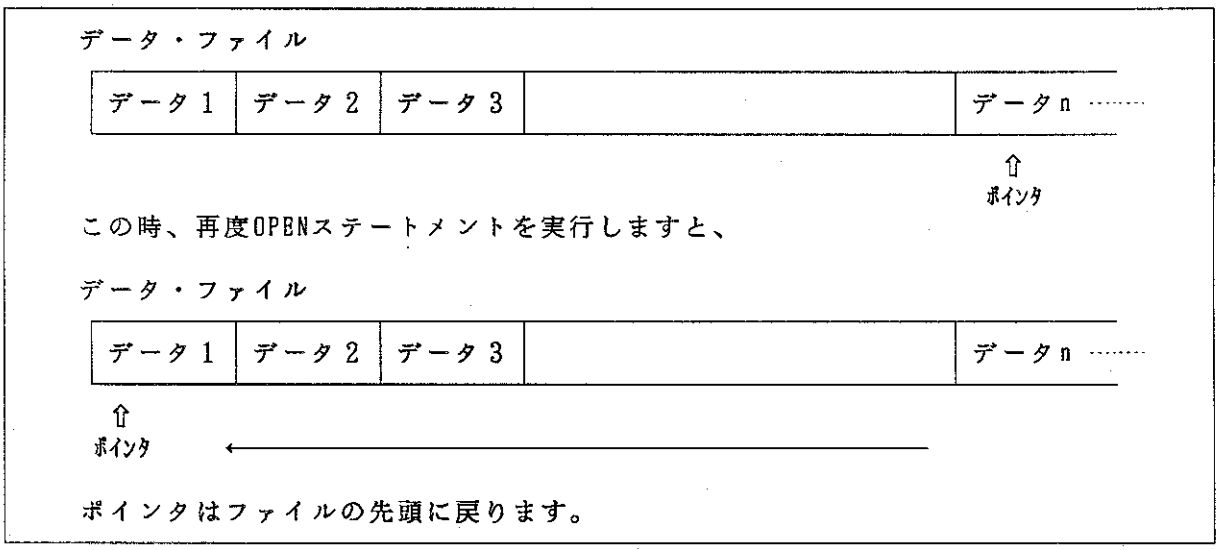
データの読出しを行なった場合にはブロックn のデータはそのままで、ポインタの位置のみがブロックn+1 の先頭に移動します。同様にファイルのアクセスを繰返しますと、ポインタは順次ブロックn+1 、ブロックn+2 、ブロックn+3 と先へ進みます。
このことはシリアル・アクセス・ファイルでも全く同じで、下図に示すシリアル・アクセス・ファイルがあった場合、



このファイルをアクセスしますと、次の図に示す状態となります。

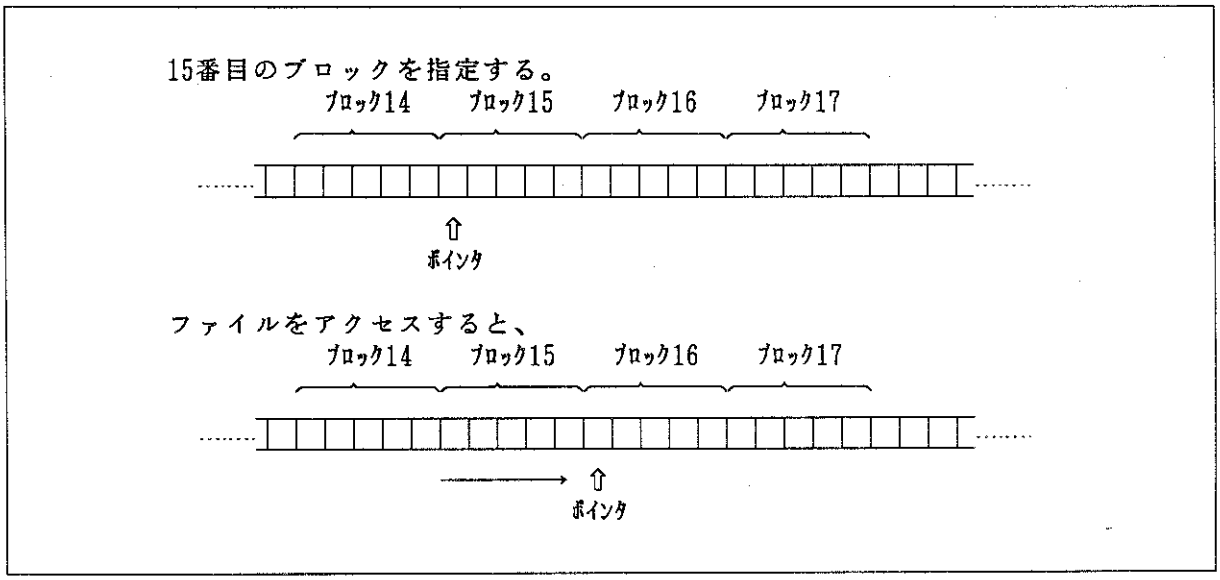


ポインタはOPENステートメントにてファイルとバッファの割付けを行なった直後にファイルの先頭にセットされ、それ以後はアクセスする毎に１ブロック（１データ）づつ順次先へ進みます。ファイルをアクセスした後再びポインタをファイルの先頭へ戻したい時は、OPENステートメントにより、再度ファイルとバッファの割付けを行ないます。但し再割付け時のファイルとバッファは最初にOPENステートメントを実行したときのものと必ず同じでなくてはなりません。（既に割付け済のファイルおよびバッファは他のバッファまたはファイルに割付けることができません。）

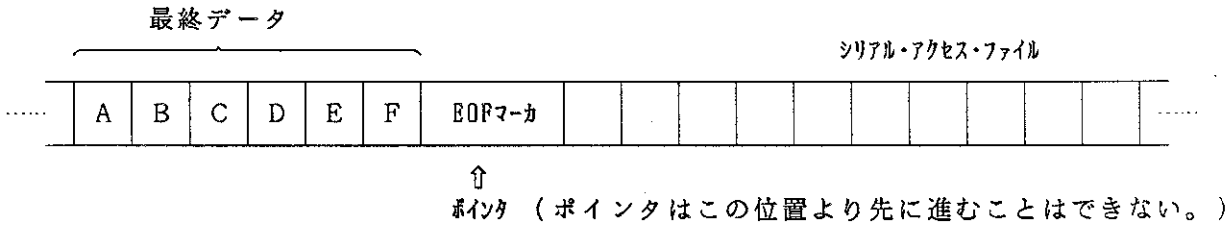


ランダム・アクセス・ファイルでは先の説明に加えて、ブロックを直接指定することができます。いままでの説明ではファイルの途中にあるデータを参照する場合には、ポインタをファイルの先頭から１つづつ目的の場所まで進めていかなくてはなりませんでしたが、ランダム・アクセス・ファイルの場合に限っては、いきなり途中のブロックを指定しアクセスすることができます。（詳細は後述）

ブロックを指定してファイルをアクセスしますと、ポインタは指定されたブロックの先頭へ移動します。但しこの場合もファイルのアクセス終了時には続くブロックの先頭へポインタが移動します。

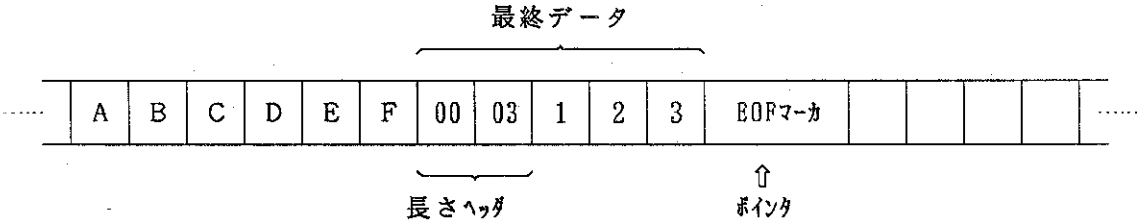


シリアル・アクセス・ファイルにはEOF(End of File) マーカがファイル中に存在します。EOF マーカはデータ・ファイルの論理的な最後部を示し、これより先にはデータが存在しないことを意味します。従って前説のポインタはEOF マーカを超えて設定することはできません。



（ポインタはこの位置より先に進むことはできない。）

EOF マーカの位置にポインタがある時、データの書込みは行なえますが、読出しはできません。EOF マーカの位置でデータの読出しを行ないますとエラーになります。(End of file) EOF マーカの位置でデータを書込みますと、新しく書込んだデータが最終データとなり、EOF マーカはそのデータの後に移動します。上図の状態で文字列データ“123”を書込んだ場合を下に示します。



EOF ポインタはデータ・ファイル作成時(CREATE␣SERIステートメント実行時）にはファイルの先頭に位置し、データを書込む毎に更新されます。

このようにシリアル・アクセス・ファイルでは常にEOF マーカが存在するため、ファイルのレコード長としてはデータ記憶に必要な長さより2バイト以上の余裕が必要です。(EOFマーカは2バイトで表現されます。）

なお、ランダム・アクセス・ファイルにはEOF マーカは存在しませんので、ファイル全体を自由にアクセスすることができます。そのかわりデータ未書込みのファイルをアクセスした場合に、デタラメなデータを読出すことがあります。

3.12 データ・ファイルへの書込み(OUTPUT)

データ・ファイルへデータを書込む場合は、OUTPUTステートメントを用います。OUTPUTステートメントは通常GPIBへのデータ出力として用いられますが、ディスティネーションとしてバッファ名を記述しますとファイルに対してデータを書込みます。但しバッファおよびファイルはOPENステートメントによって割付けられたもの以外は使用できません。

```
OUTPUT␣#n:Data
```

複数のデータを同時に書込みたい場合は、データとデータをカンマ(,) で区切って続けることができます。

```
OUTPUT␣#n:Data1, Data2, Data3, ···, DataN
```

この場合はData1 から順にData2, Data3とファイルに書込まれます。
ランダム・アクセス・ファイルへの書込み時には、バッファ名に続けてブロックを指定することができます。

```
OUTPUT␣#n, block : Data
```

block でブロックの番号を指定しますとポインタがそこに移動し、指定されたブロックに対してDataを書込みます。

例）#0のバッファにランダム・アクセス・ファイルが割付けられている。このファイルの17ブロック目に数値変数Avの内容を書込む。

```
OUTPUT␣#0, 17:Av
```

例）Nameという名のシリアル・アクセス・ファイルがある。このファイルを適当なバッファに割付けた後、"Advantest" という文字列データを書込むプログラムを作る。

```
10␣OPEN␣#1␣TO␣Name
20␣OUTPUT␣#1:"Advantest"
30␣CLOSE␣#1
40␣END
```

例）Dataという名前のランダム・アクセス・ファイルを作成し、1～512 の数値データを記録する。

```
10␣CREATE␣RAND␣Data, 512, 10
20␣OPEN␣#0␣TO␣Data
30␣FOR␣I=1␣TO␣512
40␣OUTPUT␣#0:I
50␣NEXT␣I
60␣CLOSE␣#0
70␣END
```

（40番の行は、OUTPUT␣#0, I:I としても同じ結果を得る。）

例）#2のバッファにランダム・アクセス・ファイルが割付けられている。このファイルの100 番目のブロックから"ABC", "123", "!?#" の3つの文字列データを書込む。

```
OUTPUT␣#2, 100:"ABC", "123", "!?#"
```

3.13　データ・ファイルの読出し（ENTER）

データ・ファイルからデータ読出す場合はENTER ステートメントを用います。ENTER ステートメントは通常GPIBからデータ入力に用いられますが、ソースとしてバッファ名を記述しまと、データ・ファイルからデータを読出します。ENTER ステートメントでもOUTPUT ステートメント同様、OPENステートメントによるバッファとファイルの割付けが行なわれていないとデータの読出しはできません。

```
ENTER␣#n：数値変数または文字列変数
```

ENTER ステートメントのディスティネーションは必ず変数でなければなりません。読出すデータによって数値変数で受けるか、文字列変数で受けるかを決めます。
ランダム・アクセス・ファイルからデータを読出す場合は、ブロックを指定することができます。

```
ENTER␣#n, block:数値変数または文字列変数
```

例）FileA というシリアル・アクセス・ファイルを作成し、0～1023の数値を記録させた後、ファイルの最初からデータを読み出す。

```
 5␣DIM␣A(1023)
10␣CREATE␣SERI␣0:FileA, 1024+2
20␣OPEN␣#0␣TO␣0:FileA
30␣FOR␣I=0␣TO␣1023
40␣OUTPUT␣#0:I
50␣NEXT␣I
60␣SCLEAR
70␣OPEN␣#0
80␣FOR␣I=0␣TO␣1023
90␣ENTER␣#0:A(I)
100␣DISP␣A(I);
110␣NEXT␣I
120␣CLOSE␣#0
130␣END
```

10番の行でレコード長を1024+2としているのはEOF マーカ（2バイト）分余分にファイルを大きく確保するためです。
70番の行で再度OPENステートメントを実行しているのは、ファイルのポインタをファイル先頭に戻すためです。

3.14 ファイルのプロテクト(PROTECT)

3.7節でファイルの保護コードについて説明しましたが、ファイルを保護するもう1つの方法を本節で述べます。
3.7節で説明した保護コードは作成したファイルを他者からアクセスできないようにするためのもので、言わばパスワード的な扱い方をしますが、これから説明するプロテクトは誤操作によるデータの破壊、ファイルの抹消という悲劇的な事故を防ぐためのものです。
プロテクトには以下に示す特長があります。

- ファイルの書込みを禁止する。
- ファイルの読出しを禁止する。
- ファイル名をディレクトリ表示時にマスクする。
- プロテクト状況はディレクトリ表示で明確にされる。
- プログラム・ファイル、データ・ファイルの区別なくプロテクトを行なうことができる。

ファイルのプロテクトはPROTECT ステートメントで行ないます。

```
PROTECT␣n:filename, protect ↴
```

protect はプロテクト・コードの意味で次に示す5種類のものがあります。

i) W …… 書込み禁止。（ライト・プロテクト）
ii) RW…… 読出し、書込み禁止（リード・プロテクト&ライト・プロテクト）
iii) S …… ファイル名のマスク（シークレット・ファイル）
iv) WS…… 書込み禁止、ファイル名のマスク
v) RWS …… 読出し、書込みの禁止、ファイル名のマスク

プロテクト・コードWでプロテクトされたファイルは、SAVE, DELETE, OUTPUTステートメントを受け付けません。(Write protectedエラーを発生します。）プロテクト・コードRWでプロテクトされたファイルはファイルをアクセスする全てのステートメント(SAVE, LOAD, DELETE, OUTPUT, ENTER)を受け付けません。(Read protected またはWrite protected エラーを発生します。）
プロテクト・コードS でプロテクトされたファイルは、DIR ステートメントでディレクトリ表示をした時にファイル名を表示しなくなります。（ただし、ブロック数、レコード長、ファイル・タイプ、プロテクト・コードは表示されます。）プロテクト・コードWSはプロテクト・コードW とプロテクト・コードS の組合せで、ファイル名のマスクとライト・プロテクトを同時に行ないます。プロテクト・コードRWS はプロテクト・コードRSとプロテクト・コードS の組合せで、リード・プロテクト、ライト・プロテクト、ファイル名のマスクを同時に行ないます。
なお既にプロテクトされているファイルのプロテクト解除を行なうにはPROTECT ステートメントをプロテクト・コード無指定で実行します。

```
PROTECT␣n:filename, ↴
```

例）AAA_DAT という名前のデータ・ファイルをライト・プロテクトする。

```
PROTECT␣AAA_DAT, W
```

例）PROGという名前のプログラム・ファイルのプロテクトを解除する。

```
PROTECT␣PROG,
```

ライト・プロテクトには本節で説明したPROTECT ステートメントによるソフト的なものと、バブル・カセット自体をプロテクトしてしまうハード的なものがあります。（〔2.3節〕参照）

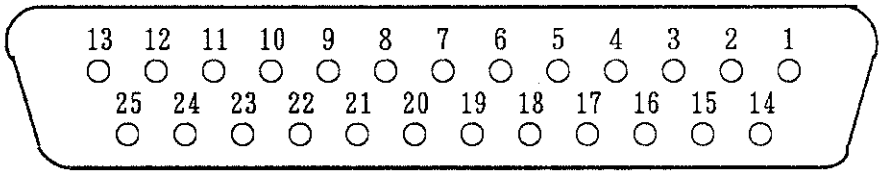
ハード的にライト・プロテクトされたバブル・カセットをドライブに挿入しますと、ドライブ右下の緑色ランプが点燈します。

なお、ハード的にライト・プロテクトされたバブル・カセットはPROTECT ステートメントではプロテクト解除することはできません。バブル・カセットのプロテクト・キーによってプロテクト解除を行なって下さい。

3.15 TTL I/O の使い方

本機の背面パネル左側に25ピンのD-sub コネクタでTTL I/O という端子があります。これは TTLレベルの入出力I/O で同軸リレー等外部機々の制御や TTLレベル信号の監視など多目的に応用できます。コネクタ各ピンの信号を下図に示します。

表 3-1 TTL I/O 信号表



TTL I/O コネクタ・ピン番号

ピン番号	信号名称	ピン番号	信号名称	ピン番号	信号名称
1	} GND	11	OUT7	21	IN6
2		12		22	IN7
3		13	} GND	23	
4	OUT0	14		24	} VCC/GND
5	OUT1	15	IN0	25	
6	OUT2	16	IN1		
7	OUT3	17	IN2		
8	OUT4	18	IN3		
9	OUT5	19	IN4		
10	OUT6	20	IN5		

GND = 0V, VCC = +5V

OUT0～OUT7は出力ポートで、オープン・コレクタ出力となっており、IN0～IN7は入力ポートでTTL レベル入力です。信号仕様の詳細は第6章を参照して下さい。
TTL I/O の操作はGPIBから行ないます。以下に操作方法について説明します。

3.15.1 TTL I/O 出力ポートの設定

出力ポートの設定はTLコマンドを用いて行ないます。
TLコマンドに続けて出力データを送ると、出力ポートには出力データに対応した信号が出力されます。出力データと出力ポートの設定の関係を下表に示します。

表 3-2 出力データと出力ポートの設定の関係

出力ポート	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
出力データ	128	64	32	16	8	4	2	1

表に示される出力データを送ることで、その出力データに対応する出力ポート(OUT0 ～OUT7) がHighレベルに設定されます。

例えばOUT4をHighレベルに設定したい場合は16を出力データとして送ればよい訳です。また2つ以上の出力ポートを同時に設定する場合は、それぞれ対応する出力データの総和を出力データとして送ります。OUT5とOUT2を同時にHighレベルに設定する場合は、32+4=36 を出力データとして送ります。

出力ポートは出力データが送られる度に設定を更新しますので、以前に設定された状態はクリアされてしまいます。例えばいま仮にOUT6がHighレベルに設定されていたとします、この状態をそのままにしてOUT0をHighレベルに設定しようとした場合、単に出力データに1を送るだけでは、OUT6はリセット(LOWレベルに設定) されてしまいます。OUT6の状態を保持するためには、64+1=65 を出力データとして送らなくてはなりません。

例) 出力ポートのOUT7, OUT4, OUT3をHighレベルに設定する。
(ただし、TR45102 のGPIBアドレスは1に設定されているものとする。)

```
OUTPUT␣1:"TL";128+16+8
```

または

```
OUTPUT␣1:"TL152"
```

注意：出力ポートはオープン・コレクタですのでそのままでは電圧出力は行ないません。外部回路でプルアップして下さい。プルアップ抵抗は数 kΩ～数 10kΩの抵抗を用いて、必ず+5V にプルアップして下さい。

3.15.2 TTL I/O 入力ポートからの入力

TTL I/O の15～21ピンに入力された信号は、OTコマンドによりGPIBへ出力することができます。入力ポートと入力データの関係を下表に示します。

表 3-3 入力ポートと入力データの関係

入力ポート	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
入力データ	128	64	32	16	8	4	2	1

いま仮にIN5 のみにHighレベルの信号が入力されていたとします。このとき、TTL I/O を読出しますと、入力データとして32が得られます。入力データはそれぞれの入力ポート(IN7～IN0)に対応する値の総和で与えられます。例えばIN5, IN3, IN0 にHighレベル信号が入力されている場合にTTL I/O を読出しますと、32+8+1=41 が得られます。

例）入力ポートを読出し、各ポートに対する信号入力状況を表示する。

```
10␣SCLEAR
20␣OUTPUT␣1:"OT"
30␣ENTER␣1:Io
40␣FOR␣I=0␣TO␣7
50␣DISP␣"IN";I;
60␣IF␣BIT(Io, I)=0␣THEN␣DISP␣"LOW"
70␣IF␣BIT(Io, I)=1␣THEN␣DISP␣"HIGH"
80␣NEXT␣I
90␣END
```

注　　意

TTL I/O を御使用になる時は必ずTTL レベルの信号で扱って下さい。規定外の信号を入力した場合は故障の原因になりますので絶対に避けて下さい。

3.16 キーボード(TR45103) の扱い方

キーボードの接続に関しては〔1.5 節〕に説明してありますので、ここではキーボードの扱い方を説明いたします。また本節の説明とともにTR45103 取扱説明書も合せて御覧下さい。

まず本器(TR45102) にキーボード(TR45103) を接続して使う場合は、ファンクション・キー銘板の“03 ”をお使い下さい。ファンクション・キー銘板には上下 2 行に分けてファンクションが印刷されていますが、通常は下行のファンクションが動作しています。そしてSHIFT キーを押しながら、ファンクション・キーを押しますと、上行のファンクション動作になります。ファンクション・キーの機能を以下に示します。

F1～F10 : TR4511/12 使用時には特に機能を持ちません。

PROG : TR4511/12 プログラム・モードにします。

EXIT : TR4511/12 でプログラム・モードから抜出します。

LIST : リストの表示をします。

PLIST : GPIBに接続されたプリンタにリストを出力します。

RECALL : TR4511/12 では機能しません。

INS␣IN : 1 行挿入。TR4511/12 の SHIFT + LABEL (INS) キーと同じ動作をします。

DEL␣IN : TR4511/12 では機能しません。

INS␣CHR : 1 文字挿入。TR4511/12 の (INS) キーと同じ意味です。

DEL␣CHR : 1 文字削除。TR4511/12 の (DEL) キーと同じ意味です。

CLR␣LN : TR4511/12 は機能しません。

CLR␣SCR : 画面クリア。TR4511/12 の (CLR) キーと同じ意味です。

STEP : TR4511/12 は機能しません。

DEL␣EOL : TR4511/12 は機能しません。

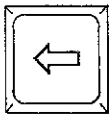
PAUSE : プログラムの一時停止

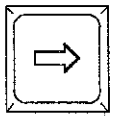
CONT : 一時停止中のプログラムを再び実行させます。

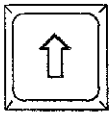
RUN : プログラムを始めから実行します。

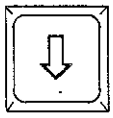
STOP : プログラムの実行を中断させます。

次にカーソル・キーの説明をします。

 　……　カーソルを左へ移動させます。

 　……　カーソルを右へ移動させます。

 　……　カーソルを上へ移動させます。スクロール指定された場合はリストを上方向へスクロールさせます。

 　……　カーソルを下へ移動させます。スクロール指定された場合はリストを下方向へスクロールさせます。

スクロール指定は 

キーにより行ないます。 

キーはトグル設定になっており、通常の状態でこのキーを押しますと、スクロール・キーとして機能し、再度カーソル・キーの機能に戻ります。

なおカーソル・キーはダブル・アクションになっており、軽く押した場合は通常のキー入力動作を行ない、強く押込みますと、高速でリピートします。

キーは以下のように他のキーと組み合せることで特殊な動作をします。これらの機能を使用する場合は必ず 

キーを押しながら他のキーを押します。（先に 

キーを押して下さい。）

…… キーボード内のベルを鳴らします。

…… キー・クリック音をONします。

…… キー・クリック音をOFFします。

…… オート・リピートをONします。

…… オート・リピートをOFFします。

キー・クリック音は、各キーを押した時に発生する“コッコッ”という音です。電源投入時にはキー・クリック音はON状態になっています。
オート・リピートは各キーを押し続けた時に自動的に繰返しキー入力を行なうもので、電源投入時はON状態になっています。

なおTR45102 を介してTR4511/12 をキーボード操作する場合、 

キーおよびカナ文字は扱うことができません。 

キーは常にOFF の状態にしておいて下さい。

3.17　エラー・メッセージ

エラー・メッセージ	意　　味
Bubble full.	バブル・カセットの容量を超えた。
File not found.	指定された名前のファイルが見つからない。
Write protected.	バブル・カセットがライト・プロテクトされている。 ファイルがライト・プロテクトされている。
Read protected.	ファイルがリード・プロテクトされている。
Security code violation.	保護コードが指定されているファイルを保護コードなしにアクセスしようとした。
Already exists same file.	SAVE, CREATEステートメントで既存のファイルと同じファイル名を使用した。
End of file.	ランダム・アクセス・ファイルでファイルの容量を超えてアクセスしようとした。 シリアル・アクセス・ファイルでEOF マーカを超えてアクセスしようとした。
Directory overflow.	1つのバブル・カセットに64以上のファイルを記録しようとした。
No initialize cassette	バブル・カセットがイニシャライズされていない。
File not open	ファイル割付けのされていない、バッファに対してデータの読出し、書込みを行なった。
Bubble cassette error	バブル・カセットにトラブルが発生した。
Already open file.	すでに割付け済みのファイルを別のバッファに割付けようとした。
Already open buff.	すでに割付け済みのバッファを別のファイルに割付けようとした。
Bubble cassette off	ドライブにバブル・カセットが挿入されていない状態でアクセスしようとした。
Ejected bubble cassette	バブル・カセットが抜かれた。

4. 本器の制御に使用するＢＡＳＩＣステートメントの文法

4.1 概要

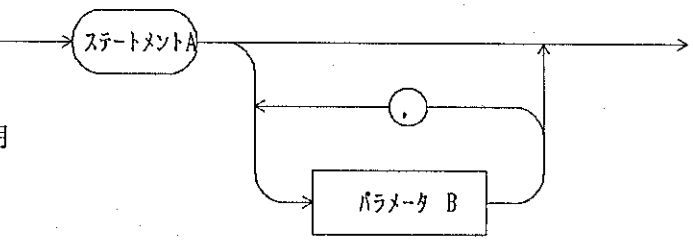
この章では、本器に対するBASIC 制御ステートメントのそれぞれについて 概　説 構　文 解　説 例 の順で説明します。 構　文 の説明においては直観的に理解できるように、図式説明と記述式説明を並記してあります。

・図式説明
構文を各要素に分解し、直線で結んで表わしてあります。
文は必ず矢印の方向に進みます。途中で２つ以上に分岐している場合は、それらのうちいずれかに進みます。また、矢印の方向がループを構成している場合は、何回でもそのループを通ることができます。

・記述式説明
記述式表現には、次に示す記号が用いられています。
〔　〕　：　この記号で囲まれた部分は省略することができます。
｛　｝　：　この記号で囲まれた部分は０回以上繰り返し用いることができます。
｜　　：　“または”の意味です。

＜例＞

構　文



図式説明

記述式説明　ステートメントA ｛パラメータ B｝

解　説　この例ではパラメータB をカンマで区切って必要数繰り返してステートメントA を実行するように指定できることが示されています。

なお本章で解説するステートメントはTR45102 の制御に追加されたもののみです。他のステートメントに関してはTR4511取扱説明書、またはTR4512プログラミング・マニュアルを参照して下さい。

4.2　ステートメント一覧

1.	CLOSE	ファイル、バッファのクローズ
2.	COPY	ファイルの複成
3.	CREATE RAND	ランダム・アクセス・ファイルの作成
4.	CREATE SERI	シリアル・アクセス・ファイルの作成
5.	DELETE	ファイルの抹消
6.	DIR	ディレクトリの表示
7.	ENTER	データ・ファイルからのデータ入力（読出し）
8.	INITIALIZE	バブル・カセットの初期化
9.	LOAD	プログラムのロード
10.	OPEN	ファイル・バッファの割付け
11.	OUTPUT	データ・ファイルへのデータ出力（書込み）
12.	PROTECT	ファイルのプロテクト
13.	SAVE	プログラムのセーブ

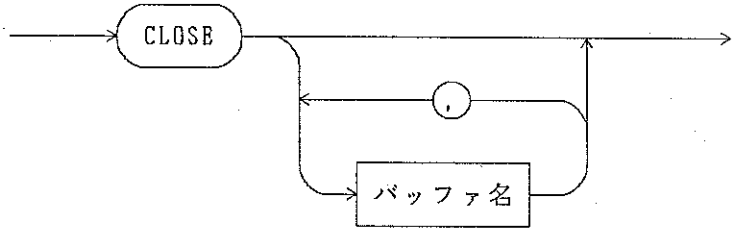
4.3　ステートメントの解説

1. CLOSE

概　要

OPENステートメントによって割り付けられたファイル・バッファをクローズします。（ファイル・バッファの割付けを解除する）

構　文



CLOSE 〔バッファ名〕

バッファ名：#0～#9

解　説

・バッファ名を省略して実行した場合は、現在割付けられている全てのバッファがクローズされます。
・バッファ内に未転送のデータが残されていた場合、対応するファイルにデータを転送した後、バッファをクローズします。
・既に割付けられているバッファをクローズせずに、新たなファイルへ割付けることはできません。
・CLOSE によって割付けが解除されたバッファは、再度OPENステートメントによって割付け作業を行うまで使用することはできません。

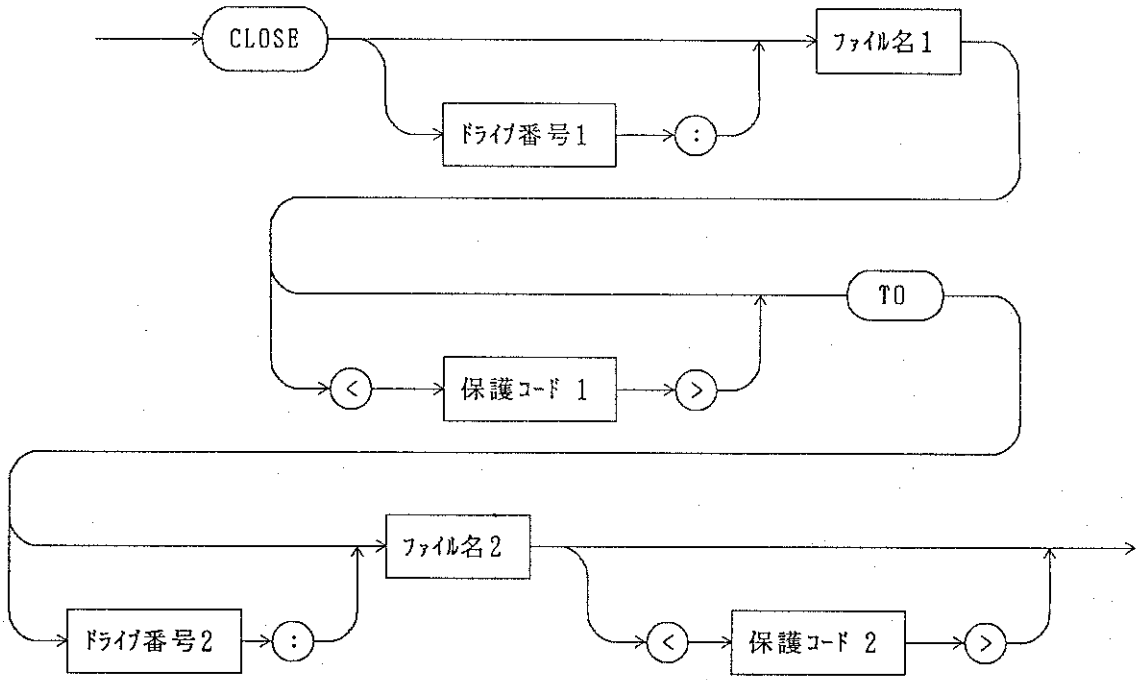
例

```
10  CLOSE  #2
20  CLOSE  #1, #2, #3
30  CLOSE
```

2. ＣＯＰＹ

概　要　バブル・カセットに記憶されているファイルを別のファイルへ複成する。

構　文



COPY〔ドライブ番号1:〕ファイル名1〔<保護コード1>〕TO
〔ドライブ番号2:〕ファイル名2〔<保護コード2>〕

ドライブ番号1: ソースとなるファイルが存在するドライブの番号。0または1。省略時は0として扱われる。
ファイル名1 : ソースとなるファイル名。10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
保護コード : ソース・ファイルの保護コード。
ドライブ番号2: ディスティネーションとなるファイルのドライブ番号。0または1。省略時は0として扱われる。
ファイル名2 : ディスティネーションとなるファイル名。10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
保護コード2 : ディスティネーション・ファイルの保護コード。

解 説

・ドライブ番号1 のバブル・カセットに既に記憶されているファイル (ファイル名1)をドライブ番号2 のバブル・カセットにファイル名2 というファイルとして複成します。
・ドライブ番号1 とドライブ番号2 は同一のものでもかまいません。但しドライブ番号1 とドライブ番号2 が同一の場合、ファイル名1 とファイル名2 は異なった名称にしなくてはなりません。
・ドライブ番号1 とドライブ番号2 が異なる場合は、ファイル名1 とファイル名2 は同一のものでもかまいません。
・COPYステートメントで保護コードを使用する場合は〔3.7 ファイルの保護について〕を参照して下さい。場合によって保護コードの指定の方法が異なります。

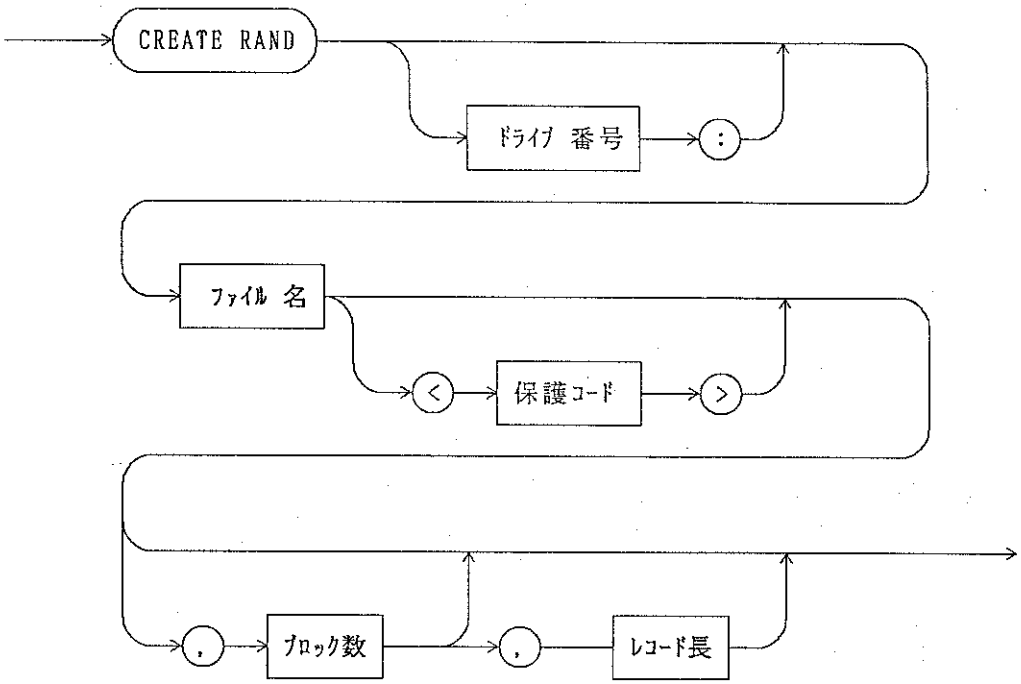
例

```
COPY  0:File A  TO  1:File A
COPY  0:File A  TO  1:Prog 1
COPY  Filename  TO  Advantest
COPY  Data<SC>  TO  1:Data
COPY  Test<SC>  TO  Demo<AB>
```

3. CREATE RAND

概 要 ランダム・アクセス・ファイルをバブル・カセット内に作成します。

構 文



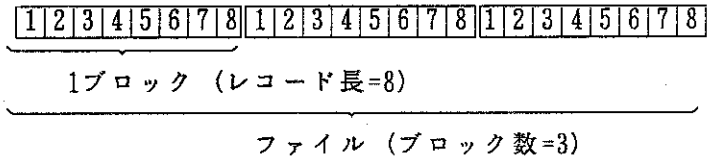
CREATE RAND〔ドライブ番号:〕ファイル名〔<保護コード>〕
〔,ブロック数〔,レコード長〕〕

ドライブ番号 : 0または1,ドライブの番号。省略時は0として扱う。
ファイル名 : 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
保護コード : 2文字以内の文字列。
ブロック数 : 1～65535 の正整数。
レコード長 : 1～65535 の正整数。

解 説

・ファイル名により命名されたランダム・アクセス・ファイルをドライブ番号で指定されたドライブ内のバブル・カセット上に作成します。
・保護コードを指定した場合は、以後ファイル名に保護コードを付加しないと同名のファイルを扱うことはできない。保護コードはディレクトリ表示で読み出すことはできませんので、作成者が個人的に憶えておかなければなりません。特に重要でないファイル、秘密性の必要のないものは保護コードを指定しない方がよいでしょう。

・ブロック数はレコード長で指定された、ある大きさのブロックをいくつ必要かを示すものです。
・レコード長は1つのブロック内で扱えるデータのバイト数を示します。
ブロック数とレコード長の関係を下図に示します。



・ランダム・アクセス、ファイルはブロック単位での読書きは自由に行えますが、ブロック単位（レコード長）以下でのバイト・アクセスは不可能です。従って扱うデータの種類に適した型でレコード長を決定しなくてはなりません。
TR4511コントローラ機能では、実数を扱う場合レコード長として10バイト必要で、文字列の場合だと1文字につき1バイト必要となります。
・CREATE RAND ステートメントでブロック数およびバイト長を省略した場合、自動的にブロック数を256,レコード長を10として処理します。

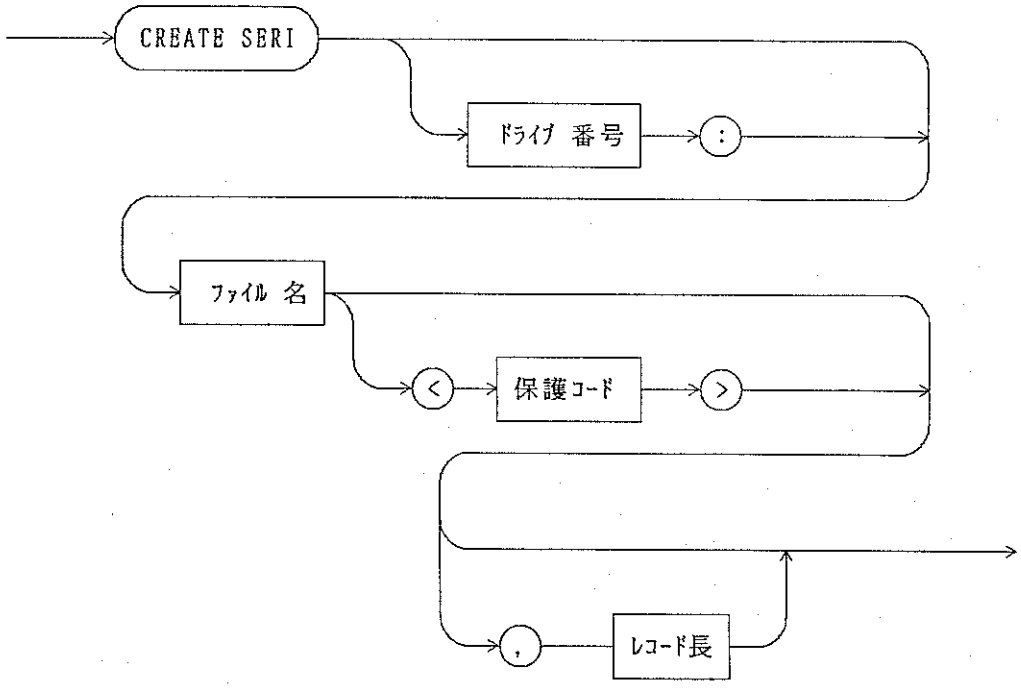
例

```
10  CREATE  RAND  0:Advantest, 100, 16
20  CREATE  RAND  RFile<AB>,200, 8
30  CREATE  RAND  ABC
40  CREATE  RAND  1:Data, 16
```

4. CREATE SERI

概 要 シリアル・アクセス・ファイルをバブル・カセット内に作成します。

構 文



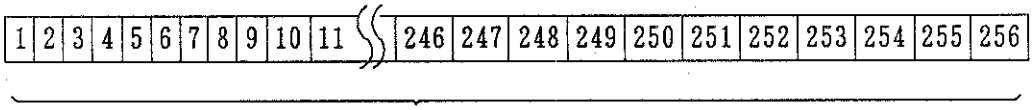
CREATE SERI〔ドライブ番号:〕ファイル名〔<保護コード>〕〔,レコード長〕

ドライブ番号 : 0または1,省略時は0として処理される。
ファイル名 : 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
レコード長 : 1～65535 の正整数。

解 説

・ファイル名により命名されたシリアル・アクセス・ファイルをドライブ番号で指定されたドライブ内のバブル・カセットに作成します。
・保護コードを指定してファイルを作成した場合は、以後ファイル名に保護コードを付加しないと同名のファイルを扱うことはできません。保護コードはディレクトリ表示で読み出すことができませんので、作成者が個人的に憶えておかなければなりません。
・シリアル・アクセス・ファイルはランダム・アクセス・ファイルと違ってブロック単位で読書きするという概念はありません。従ってファイル全体を 1つのブロックとしてレコード長でファイルの大きさを定義しま

す。ただしシリアル・アクセス・ファイルにおいても 1つのファイル内に複数のデータを蓄えることは可能です。



ファイル（レコード長=256）

・CREATE SERI ステートメントでレコード長を省略した場合、自動的にレコード長を256として処理します。

例

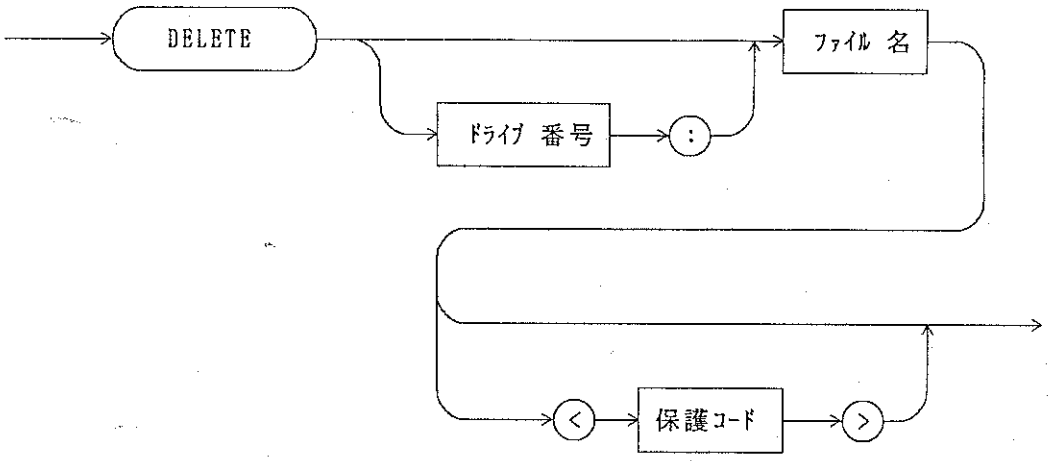
```
10  CREATE  SERI  1:SData, 512
20  CREATE  SERI  Data 01
30  CREATE  SERI  String<SC>, 1024
```

5. DELETE

概 要

既に記録されているプログラム・ファイルあるいはデータ・ファイルをメディアのディレクトリ上から抹消する。

構 文



DELETE 〔ドライブ番号:〕ファイル名〔<保護コード>〕

ドライブ番号 : 0または1 (ドライブを示す)
ファイル名 : 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
保護コード : 2文字以内の文字列。

解 説

・ファイル名によって指定されたファイルを抹消します。
・保護コードが指定されているファイルではDELETE時にも保護コードを付加しないと、ファイルの抹消はできません。
・またライトプロテクトされている場合もDELETEを行うことはできません。
・DELETEステートメントによってファイルを抹消した場合、抹消されたファイルが専有していたメモリ領域は解放され、フリー領域として再使用することができます。(メモリ領域の解放および再使用に関しては、TR45102 内部で自動的に行いますのでユーザは何ら特別な措置を施す必要はありません。)

例

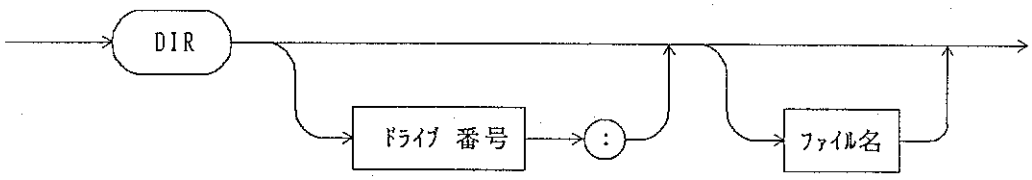
```
10  DELETE  0:Advantest
20  DELETE  Data
30  DELETE  String<SC>
```

6. DIR

概 要

バブル・カセット内に記憶されているファイルのファイル名一覧表(Directory)を表示。

構 文



DIR 〔ドライブ番号:〕〔ファイル名〕

ドライブ番号 : 0または1、省略時は0として扱う。
ファイル名 : 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。

解 説

・バブル・カセット内に既に記憶されているファイルのファイル名、およびファイルの種類(プログラム・ファイル、ランダム・アクセス・ファイル、シリアル・アクセス・ファイル)、ブロック数、レコード長等がTR4511/12 管面に表示される。
・ファイル名を省略した場合は、カセットに記憶されているファイル全てが表示され、ファイル名を指定した場合はそのファイルのみを表示します。

```
DIRRECTORY  OFF  Advantest①
0125/2020②
FILE NAME③ BLKS④ LENGTH⑤ TYPE⑥ PRO⑦
---------- ----- -------- ------ ---
Autostart   00001  01041     PROG  W
DATA_1      00256  00010     RAND
```

ディレクトリ表示例

・①の部分はINITIALIZEステートメントによって指定したバブル・カセットの名前です。
・②の部分はバブル・カセットのメモリ使用状況を意味します。
バブル・カセットは初期設定後2020ページの空領域が用意されます。
(1ページ=64バイト) その内から使用した領域分を****/2020の *の部分で表示します。
例では 125ページ分が現在使用中であることを表しています。

・③の部分はファイル名です。ここでは単にファイル名が表示されるだけで保護コードは表示されません。
また、PROTECT ステートメントによりファイル名をマスク指定した場合はファイル名は表示されません。
・④の部分はブロック数を表示します。プログラム・ファイルとシリアル・アクセス・ファイルでは強制的に0001と表示され、ランダム・アクセス・ファイルではCREATE RAND ステートメントで指定したブロック数が表示されます。
・⑤の部分はレコード長を表示します。データ・ファイルではCREATE RAND, CREATE SERI ステートメントで指定したレコード長が表示され、プログラム・ファイルではファイルに必要としたレコード長が表示されます。
・⑦ファイルの種類が表示されます。

PROG …… プログラム・ファイル
RAND …… ランダム・アクセス・ファイル
SERI …… シリアル・アクセス・ファイル

・⑧の部分は、PROTECT ステートメントによってプロテクト設定されている場合に、その内容を表示します。

W …… ライト・プロテクト
R …… リード・プロテクト
S …… ファイル名のマスク。(シークレット)

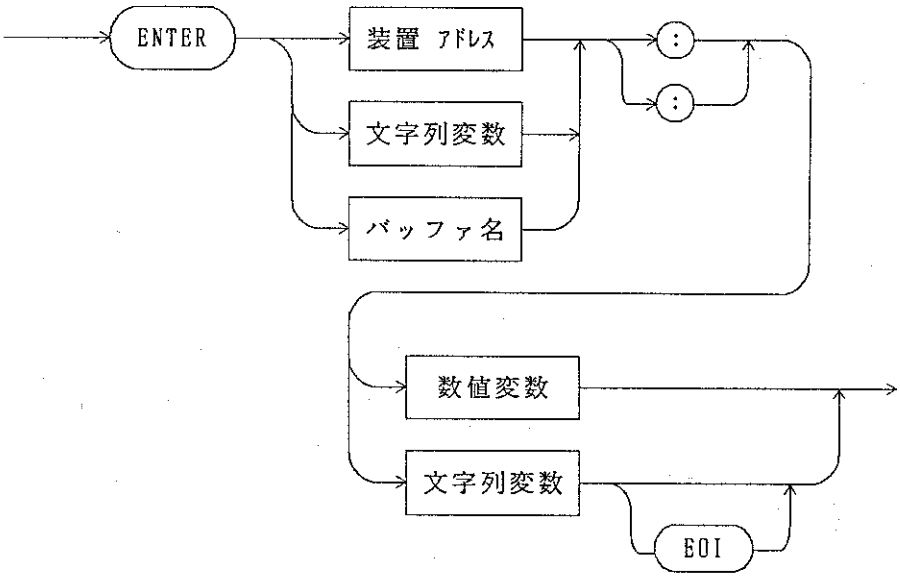
例

```
DIR
DIR   1:Name
DIR   A*
```

7. ENTER

概　要　GPIBからのデータ入力、ファイルからのデータ入力を行います。

構　文



ENTER装置アドレス｜文字列変数｜バッファ名 ：｜；数値変数｜文字列変数〔EOI〕
数値アドレス：GPIBに接続されている装置の
GPIBアドレス、0～30の正整数。
バッファ名　：#0～#9の10通りのうちいづれか。

解　説

・GPIBに関する取扱はTR4511取扱説明書、またはTR4512プログラミング・マニュアルを御覧下さい。
・ソースにバッファ名を指定した場合、バブル・カセットに記憶されているデータ・ファイル（ランダム・アクセス・ファイルまたはシリアル・アクセス・ファイル）よりデータを読み込みます。
・バッファ名はOPENステートメントによってファイルに割付け済みのもののみ有効で未割付のものは使用できません。
・ランダム。アクセス・ファイルから数値変数へ実数を読み込む場合、ファイルのレコード長が10に設定されていないとエラーになります。（ファイルにデータを記憶させる時に、実数として処理しておかないと、数値変数での読み出しは行えません。）
・ファイルから文字列変数へ任意データを読み出す場合、データ長より大きく文字列変数のディメンジョンを設定する必要があります。もし文字列変数のディメンジョンよりデータ長が大きいと処理できません。（データ長は、ランダム・アクセス・ファイルの場合にはレコード長で決まり、シリアル・アクセス・ファイルの場合では、記憶時のデータで決まります。）

例

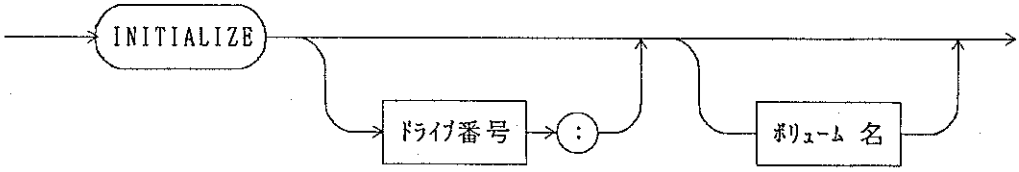
```
10  ENTER  #0 : A
20  ENTER  #1 : B$
```

8. INITIALIZE

概 要

バブル・カセットの初期化を行います。新たに使用するバブル・カセットは最初に必ずこの操作を行って下さい。

構 文



INITIALIZE〔ドライブ番号 :〕〔ボリューム名〕

ドライブ番号: 0または1。省略時の0として扱われる。
ボリューム名: 10文字以内の文字列、但し1 文字目は英大文字のみ使用可能。

解 説

・工場出荷時のバブル・カセットには、TR45102 で扱うファイル管理に関する情報は何も書込まれていません。
従って新品のバブル・カセットを使用する際には、TR45102 で操作できるように初期化しなくてはなりません。
・INITIALIZEステートメントを実行することにより、バブル・カセット内にファイル管理用の情報が書き込まれ、強制的に初期化します。
・INITIALIZEステートメントで指定するボリューム名はバブル・カセットに名称を付与するもので、カセットのボリューム管理に利用すると便利です。ボリューム名は DIRステートメント実行時に表示されるだけで、それ以外には特に意味を持ちません。
・既にプログラム等を記憶させてあるバブル・カセットをINITIALIZEすると、内部に記憶されていたデータは初期化されてしまうため再生できなくなります。
・バブル・カセットをハード的にライト・プロテクトしている場合は、INITIALIZEステートメントは実行できません。

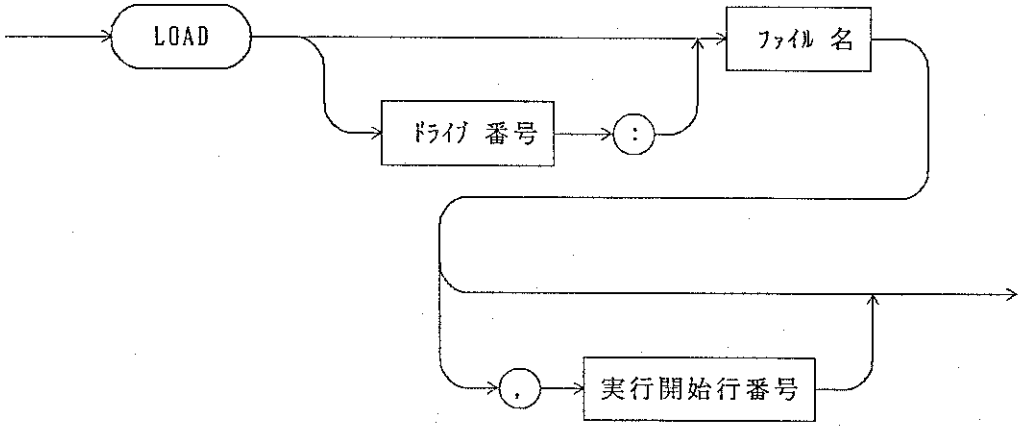
例

```
INITIALIZE   0 : Advantest
INITIALIZE   VOL_01
INITIALIZE
```

9. LOAD

概 要 バブル・カセットに記憶されているプログラムを読込む。

構 文



LOAD〔ドライブ番号 :〕ファイル名〔 ,実行開始行番号〕

ドライブ番号 : 0または1,省略時は0として扱う。
ファイル名 : 10文字以内の文字列。但し1 文字目は英大文字のみ使用可能。
実行開始行番号: 1～32767 の正整数。

解 説

・LOADを実行しますと、SAVEステートメントで記憶されたプログラム・ファイルを読み込み、TR4511/12 のプログラム・テキストとして設定されます。LOADステートメントでプログラムを読み込みますと、それ以前のTR4511/12 内に記憶されていたプログラムは抹消され、バブルから新しいプログラムを読み込みます。
・LOADステートメントに実行開始行番号を指定しますと、プログラムを読み込んだ直後に、指定された行番号から実行開始します。例えば

```
LOAD  ABC, 10
```

を実行した場合、ABC というファイル名のプログラムを読込みそのプログラムの10番の行からプログラムを実行します。
(実行開始行番号を指定しまい場合はプログラムの実行はしません)
実行開始行番号を指定した場合で、読み込んだプログラム中に指定された行番号が無い時は、指定した行番号より後で最も近い行から実行開始します。

例

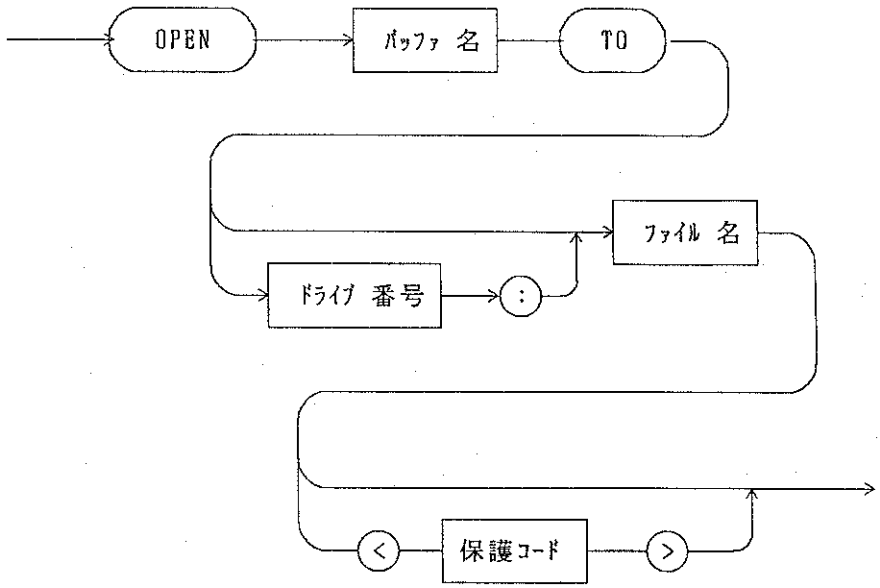
```
LOAD  0 : Advantest
10  LOAD  TEST, 20
20  LOAD  1:Program, 10
```

10. OPEN

概 要

バッファ(#0 ～#9) をランダム・アクセス・ファイルまたはシリアル・アクセス・ファイルへ割付けます。

構 文



OPEN バッファ名 TO 〔ドライブ番号 :〕ファイル名〔<保護コード>〕

バッファ名 : #0～#9の10通りのうちいづれか。
ドライブ番号: 0または1。省略時は0として扱う。
ファイル名 : 10文字以内の文字列。但し 1文字目は英大文字のみ使用可能。
保護コード : 2文字の文字列。

解 説

・ランダム・アクセス・ファイルおよびシリアル・アクセス・ファイルは直接データの入出力を行うことはできません。必ずバッファ(#0～#9)を介してデータのやりとりを行います。従ってデータの読出し、書込みはバッファをソースまたはディスティネーションとして入出力命令(ENTER, OUTPUT) を実行します。
・CREATE RAND または CREATE SERIステートメントでファイル名に保護コードを付加しなくてはなりません。
・既にOPENステートメントで割付けがなされているバッファおよびファイルをさらに別のファイルまたはバッファに割付けることはできません。バッファとファイルは一対一にしか対応させることはできません。
・バッファをプログラム・ファイルに割付けることはできません。バッファの割付け可能なファイルはランダム・アクセス・ファイルとシリアル・アクセス・ファイルのみです。

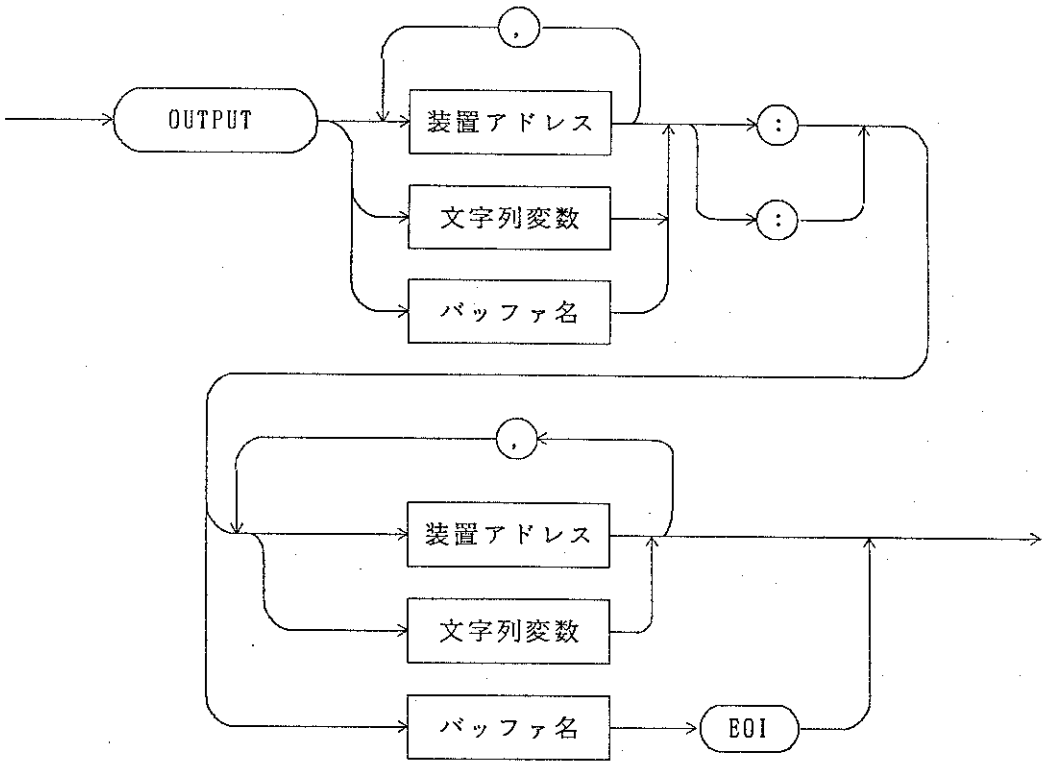
例

```
10  OPEN  #1  TO  0:Data<SC>
20  OPEN  #0  TO  Wave
30  OPEN  #3  TO  1:Volt
```

11. OUTPUT

概 要 GPIBまたは外部記憶装置(TR45102)へデータを出力します。

構 文



OUTPUT 数値アドレス { ,装置アドレス} | 文字列変数 | バッファ名 :| ; <A> | <B>

<A>::= 数値表現式 | 文字列表現式 { ,数値表現式 | 文字列表現式}
<B>::= 文字列変数 EOI

解 説

- GPIBに関する取扱はTR4511取扱説明書、またはTR4512プログラミング・マニュアルを参照して下さい。
- ディスティネーションとしてバッファ名を指定した場合、バブル・カセットに既に作成されているデータ・ファイルへデータを書き込みます。
- ファイル・バッファはOPENステートメントで割付け済みのもののみ使用することが可能です。
- ランダム・アクセス・ファイルへ数値データを書込む場合、ファイルのレコード長は10に設定されていなければなりません。
  (数値データはファイル上では10バイトで表現されます。またENTER ステートメントで数値データを読出す時、レコード長が10以外に設定されているとエラーになります。)

・文字列をファイルに書込む場合、ファイルのレコード長がデータ長より大きく確保されていなくてはなりません。（レコード長≧データ長+2）
・シリアル・アクセス・ファイルに数値データを書込む場合も、ランダム・アクセス・ファイルと同様 1データあたり10バイトの領域を必要とします。
従って、1つのシリアル・アクセス・ファイルに複数の数値データを記憶させる場合は、データの数×10を最低限としてレコード長の指定を行って下さい。（〔CREATEステートメント〕参照）

例

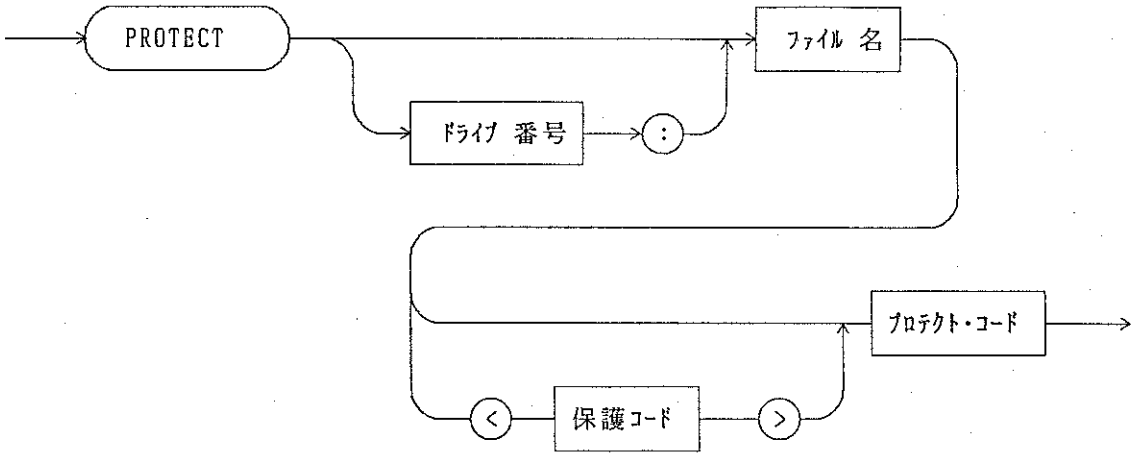
```
10  OUTPUT  #1 : 100
20  OUTPUT  #3 ; (A*2)/C, 25
30  OUTPUT  #5 ;  "ABCD"
40  OUTPUT  #1 ; A$
50  OUTPUT  #6 ; As, Nm$
```

12. PROTECT

概 要

既に記録されているプログラム・ファイルあるいはデータ・ファイルをメディアのディレクトリ上から抹消する。

構 文



DELETE 〔ドライブ番号:〕ファイル名〔<保護コード>〕, プロテクトコード

ドライブ番号 : 0または1。省略時は0として扱われる。
ファイル名 : 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
ﾌﾟﾛﾃｸﾄ･ｺｰﾄﾞ : W(ライト・プロテクト),R(リード・プロテクト) S(シークレット・ファイル) の組み合せ。

解 説

・既に記憶されているファイルに対して、書込み、読出し等の保護を行います。
・ライト・プロテクトが行われた場合、そのファイルに対する書込みを禁止します。 (SAVE, OUTPUTステートメントを受けつけない。)
・リード・プロテクトが行われた場合、そのファイルに対する読出しを禁止します。 (ENTER, LOAD ステートメントを受けつけない)
・シークレット・ファイルは DIRステートメント実行時、ディレクトリ表示としてのファイル名をマスクします。 (表示出力しない。)
但しレコード長、ブロック数、プロテクトの種類に関する情報は表示されます。
・PROTECT ステートメントはプログラム・ファイル、ランダム・ファイル、シリアル・アクセス・ファイルのいづれでも実行可能です。
・ライト・プロテクト、リード・プロテクト、シークレット・ファイルは

それぞれ組み合せで用います。

W ..... ライト・プロテクト
RW ..... リードプロテクト & ライト・プロテクト
A ..... シークレット・ファイル
WS ..... ライト・プロテクト & シークレット・ファイル
RWS ..... リード・プロテクト & ライト・プロテクト & シークレット・ファイル

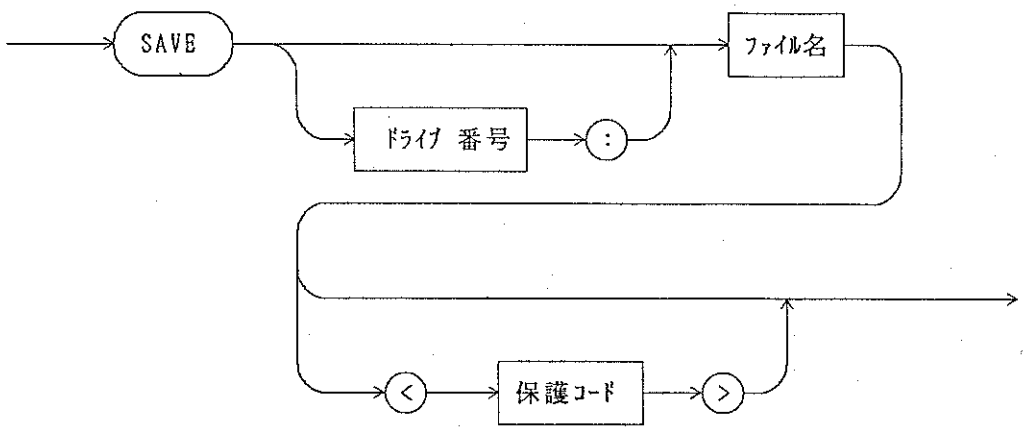
例

```
10  PROTECT  1:Advantest<SC>, W
20  PROTECT  RData,
30  PROTECT  Test, S
40  PROTECT  Program, RW
50  PROTECT  Name, WS
60  PROTECT  Comp, RWS
```

13. SAVE

概 要 プログラムをセーブします。

構 文



SAVE 〔ドライブ番号:〕ファイル名〔<保護コード>〕

ドライブ番号 ： 0または1、省略時は0として扱う。
ファイル名 ： 10文字以内の文字列。但し1文字目は英大文字のみ使用可能。
保護コード ： 2文字以内の文字列

解 説

・Basic プログラムをバブル・カセットに記憶させます。
・既に記憶されているファイルに対して再度SAVEを行いますと、以下のメッセージが表示されSAVEを中断します。

```
Already exists same file.  RE SAVE ?
```

古いファイルを消して新しいプログラムをセーブしたい場合はこの後

TR4511/12 から [MHz] Y [□ ENTER] とキー入力して下さい。

古いファイルを消去したくない場合は単に [□ ENTER] キーのみを押せばSAVEを中止します。

例

```
SAVE  0: Program1<SC>
SAVE  PROG01
```

5. GPIB制御コマンドの書式と機能

5.1 GPIBコマンドの書式と機能

本器をTR4511/12 BASIC を用いずに直接GPIB制御する場合について説明します。
以下に各コマンドについて書式、機能、受け付け条件、プログラム例および注意事項を説明します。書式における［ ］内のものは省略可であることを意味します。［ ］自身はコマンド中にいれないで下さい。

5.1.1 INコマンド：イニシャライズ

バブル・メモリのディレクトリをすべてクリアし、イニシャライズします。バブル・メモリは新しく使用する前に初期化しませんとアクセスできません（NO INITIALIZE BUBBLEエラー）。
なお、本コマンドではデータ・エリアのクリアは行なわれません。

書式	IN [,n] [,Volumename] n ：ドライブ番号（0、1） Volumename ：バブル・カセットに名前を付ける（10文字以内）
機能	IN ドライブ0のバブル・メモリを無名でイニシャライズ。
	IN, n ドライブnのバブル・メモリを無名でイニシャライズ。
	IN, n, Volumename ドライブnのバブル・メモリに、指定されたVolumenameを付けてイニシャライズします。Volumenameを指定しますと、ディレクトリの出力時に最初にVolumenameが出力されます。Volumenameの有無は動作上なんら影響を与えません。

プログラム例

```
OUTPUT 701 ; "IN, 0, Advantest"
OUTPUT 701 ; "IN "
```

5.1.2 CRコマンド :ランダム・アクセス・データ・ファイルの作成

ランダム・アクセス・ファイルとはファイル中の各ブロック・データを任意にアクセスできるファイルです。

書式	CR, n, file name [<SC>], block, record size n : ドライブ番号 file name : ファイル・ネーム (10文字以内) <SC> : 保護コード (2文字) blocks : ブロック数 (1～65535) record size : 1ブロックのバイト数 (1～65535)
機能	ドライブnに指定されたファイル名のデータ・ファイルをランダム・アクセス・ファイルで作成します。

コマンド受付条件

(1) blocks×record size=ファイル容量 (Byte) が65535を超えないこと。
(2) 既に記憶されているファイル名と同一のファイル名を使用することはできない。(PRESENT SAME FILEエラー)
(3) 現在あるファイルのトータルが64ファイル未満であること。(TOTAL FILE OVERエラー)
(4) ファイル作成に要するページ数 (データ・エリア＋ヘッダー・エリア) が確保可能なこと。

注意

CRコマンドによって作成されたランダム・アクセス・ファイルの中味は他のファイル (シリアル・アクセス・ファイル、プログラム・ファイル) とは異なり、データ・クリアされません。したがって書込みをしなくても最初から blocks × record size で指定された分のデータが存在し、このデータは何が書かれているか不明となります。

プログラム例

```
OUTPUT 701 ; "CR, 0, DATA, 10, 256"
OUTPUT 701 ; "CR, 0, DATA2 <AB>, 100, 8"
```

5.1.3 CSコマンド :シリアル・アクセス・データ・ファイルの作成

シリアル・アクセス・ファイルとはファイル内のはじめのデータから順次アクセスするファイルでランダム・アクセスすることはできません。可変長のデータを扱う場合に適しています。

書式	CS, n, file name[<SC>], record size n : ドライブ番号 file name : ファイル・ネーム (10文字以内) <SC> : 保護コード (2文字) blocks : ブロック数 (1～65535) record size : 1ブロックのバイト数 (1～65535)
機能	ドライブnに、シリアル・アクセス・データ・ファイルを作成します。

コマンド受付条件
(1) 同一のファイル名が存在しないこと。
(2) 現在あるファイルのトータルが64ファイル未満であること。
(3) ファイル作成に要するページ数 (データ・エリア+ヘッダ・エリア) が確保可能なこと。

プログラム例
```
OUTPUT 701 ; "CS, 0, SDATA, 4096"
OUTPUT 701 ; "CS, 1, Name <X1>, 256"
```

5.1.4 PRコマンド ファイルのプロテクト

CR, CS, SAコマンドによって作られたファイルをプロテクトします。

書式	PR, n, file name[<SC>], Protect code n : ドライブ番号 file name : ファイル名 <SC> : 保護コード protect code: ␣ : プロテクト解除 W : 書込み禁止 RW : 読み書き禁止 S : ㊙ファイル WS : ㊙ファイルの書込み禁止 RWS: ㊙ファイル読み書き禁止 ㊙ファイルはDirectory出力時にファイル名を隠すものです。
機能	ドライブnのファイルにプロテクト・コードを付与します。

コマンド受付条件
(1) CR, CS, SAコマンドによってファイルが作られていること。

プログラム例
```
OUTPUT 701 ; "PR, 0, ADVANTEST, RW"
OUTPUT 701 ; "PR, 1, BUBBLE<AA>, S"
```

5.1.5 OPコマンド :ファイルのオープン

ランダム・アクセス・ファイルおよびシリアル・アクセス・ファイルではバッファを通してファイルとデータのやりとりを行ないます。直接ファイルとのデータのやりとりはできません。バッファは#0～#9の10通りあり、OPコマンドではこれらのバッファを各ファイルに一対一に対応させます。
このコマンドを実行するとファイルの読み書きのためのポインタがファイルの先頭にセットされるためファイルの先頭から読み書き可能となります。

書式	OP, buff, n, file name [<SC>] buff : バッファ番号 (0 ～ 9) n : ドライブ番号 file name : ファイル名 <SC> : 保護コード
機能	指定のバッファをアサインし、指定されたファイルをオープンします。

注意
すでにOPENされているバッファを再オープン(最初と同一内容のコマンドを送る)するとポインタがファイルの先頭にセットされます。もしバッファに未書込みのデータがあるときはファイルに書き込まれます。

コマンド受付条件
(1) CR, CSコマンドによってファイルが作られていること。
(2) プログラム・ファイルでないこと。
(3) 指定したバッファに他のファイルがアサインされてないこと。
(4) 同一ドライブ番号の同一ファイルをあるバッファに既にアサインしてあるとき他のバッファへアサインできない。

プログラム例
```
OUTPUT 701 ; "OP, 3, 0, DATA"
```

5.1.6 #nコマンド :バッファの指定

データ記録、再生時のバッファを指定します。ランダム・アクセス・ファイルおよびシリアル・アクセス・ファイルのときOPコマンドによってファイルをバッファにアサインした後、どのバッファ (0～9) とデータのやりとりを行なうかを指定する#nを実行することでデータ・モードになり、その後データのやりとりが可能となります。

書式	#n [, block] n : バッファ番号 (0 ～ 9) block : ランダム・アクセス・ファイルのブロック番号指定
機能	block はランダム・アクセス・ファイルのときデータを読み書きしたいブロック番号を指定する場合に用います。block を省略すると以前指定したブロックに引き続いたブロックになります(OP コマンド後は 1 にセットされる)。

コマンド受付条件
(1) 指定するバッファがOPコマンドでOPENされていること。
(2) ランダム・アクセス・ファイルのときのblock がCRコマンドで指定されているブロック番号を超えないこと。
(3) シリアル・アクセス・ファイルではblock 指定できない。

プログラム例
```
OUTPUT 701 ; "#1"
OUTPUT 701 ; "#2, 15"
```

5.1.7 CLコマンド：ファイルのクローズ

書式	CL [, buff] buff : バッファ番号 (0～9)
機能	OPコマンドによってアサインされていたバッファを閉じるシリアル・ファイル (重ね書きでないとき) の書込みのときはバッファに残データがあれば書込み・ディレクトリには書込み済データ容量を書込みます。 buffを省略したときはOPENされている全てのバッファがCLOSE されます

プログラム例
```
OUTPUT 701 ; "CL"
OUTPUT 701 ; "CL, 3"
```

5.1.8 SAコマンド：プログラムのセーブ

書式	SA, n, file name [<SC>] n : ドライブ番号 file name : ファイル名 <SC> : 保護コード
機能	プログラム・ファイルを作成後、データ・モードになり、その後送られて来るデータがファイルに書込まれます。ファイルの容量は最大65535バイトです。

コマンド受付条件
(1) <SC>を除いたfile name と同じファイルがないこと。
(2) 現在あるファイルのトータルが64ファイル未満のこと。
(3) バブル・メモリの残ページが2以上あること。
1ファイル作成に必要最低なページ数は、
データ用——1ページ
ヘッダー用—1ページ
} 合計2ページ必要です。2ページ以上ないときはBUBBLE FULLエラーとなります。

プログラム例
```
OUTPUT 701 ; "SA, 0, Program"
SEN 7 ; DATA Text$ END (Text$ :プログラム・テキストのストリングとする。)
```

5.1.9 LOコマンド：プログラムのロード

書式	LO, n, file name [<SC>] n : ドライブ番号 file name : ファイル名 <SC> : 保護コード
機能	プログラム・ファイルを読み出します。LOコマンド実行直後にTR45102 が Talker Active State になるとプログラム・ファイルを出力します。データの最後ののバイトにEOI を付加し出力します。

コマンド受付条件
(1) SAコマンドによって作られたプログラム・ファイルがあること。
(2) リード・プロテクトでないこと。
(3) ファイルにデータが書かれていること。

プログラム例
```
OUTPUT 701 ; "LO, 0, Prog"
ENTER 701 USING "-K"; Text$
```

5.1.10 DEコマンド：ファイルの抹消

書式	DE, n, file name [<SC>] n : ドライブ番号 file name : ファイル名
機能	DE, n, file name n,file name で指定されたファイルのみ抹消。
	DE, n, * ドライブn のバブル・メモリ内の全ファイルを抹消します。ただし、保護コードのあるものは抹消しません。また抹消はディレクトリの最初のものから一つずつ行ないますがその途中にライト・プロテクトのものがあったならその段階でエラーとなりコマンドは終了します。
	DE, n, AB* ファイル名の頭がABで始まるファイルを抹消する以外は上記と同じ。

コマンド受付条件
(1) CR, CS, SAコマンドによってファイルが作られていること。
(2) ライト・プロテクトされてないこと。

プログラム例
```
OUTPUT 701 ; "DE, 0, DATA"
OUTPUT 701 ; "DE, 1, Prog<AA>"
```

5.1.11　ＤＩコマンド

ASCII フォーマットでディレクトリを出力。DIコマンド実行直後にTR45102 が、Talker Active State になるとディレクトリを出力します。

書式	DI, n [, file name] n :　ドライブ番号　クトリのみを出力します。 file name :　ファイル名
機能	DI, n , file name n, file nameで指定されたファイルのディレクトリのみを出力します。
	DI, n 又はDI, n, * 全ファイルのディレクトリを出力します。保護コードのあるファイルのディレクトリも出力します。
	DI, n, AB* ファイル名の頭がＡＢで始まるファイルのディレクトリを全部出力します。保護コードのあるファイルのディレクトリも出力します。

出力データのフォーマット

(a)　DDDDDDDDDD, UUUU/2024 ㊌㊊
(b)　NNNNNNNNNN␣BBBBB␣LLLLL␣TTTT␣PPP␣㊌㊊
　　NNN ――――――――――――――――――PPP␣㊌㊊
　　⋮
　　NNN ――――――――――――――――――PPP␣
　　㊌㊊+EOI

DDDDDDDDDD= device name
　Inコマンドで指定したものが出力されます。もし10文字に満たないときはその分うしろに␣をつめます。また、INコマンドで指定しなかったときは10文字すべて␣を出力します。
UUUU= 使用済ページ数
　バブル・メモリの使用済のトータル・ページ数。４桁に満たないときは頭に０をつめます。
NNNNNNNNNN= file name
　PRコマンドで S、WS、WRS を指定した場合は10文字すべて␣を出力します。
BBBBB= blocks ［単位バイト］
　ブロック数で５桁に満たないときは頭に０をつめます。シリアル・アクセス・ファイルとプログラム・ファイルのときは１。
LLLLL= length (blocks ×record size)［単位バイト］
　ファイル全体の長さ（容量）を示します。
TTTTT= file type (RAND, SERI, PROG)
PPP= protect code (W ␣␣, WR␣, S␣␣, WS␣, WRS)
　PRコマンドで指定したものを出力します。なお、ファイルが一切作られていないときは(a)のみを出力します。

プログラム例

```
OUTPUT 701 ; "DI, 0, *"
ENTER 701 USING "-K" ; Dir$
PRINT Dir$
PRINT Dir$[23,]
```

5.1.12 COコマンド

<table>
<tr><td>書式</td><td colspan="2">CO, n, file name 1[<SC1>], m, file name 2[<SC2>]

n : ソース・ファイルのドライブ番号
file name 1 : ソース・ファイル名
<SC1> : ソース・ファイルの保護コード
m : ディスティネーション・ファイルのドライブ番号
file name 2 : ディスティネーション・ファイル名
<SC2> : ディスティネーション・ファイルのドライブ番号</td></tr>
<tr><td rowspan="4">機能</td><td>(a)</td><td>CO, n, file name 1[<SC1>], m, file name 2[<SC2>]

ドライブn のfile name 1 のファイルをドライブm のfile name 2 のファイルへコピーします。
保護コードはディスティネーションのファイルにはコピーされないのでディスティネーション・ファイルも保護するならば<SC2> を指定する必要があります。</td></tr>
<tr><td>(b)</td><td>CO, n, file name 1[<SC1>], m, [, *]

ドライブのfile name 1 のファイルをドライブm に同一ファイル名でコピーします。保護コードもディスティネーションのファイルと一緒にコピーされます。</td></tr>
<tr><td>(c)</td><td>CO, n, *, m[, *]

ドライブn のファイル全てをドライブm にコピーします。ただし、保護コードのあるものはコピーされません。</td></tr>
<tr><td>(d)</td><td>CO, n, AB*, m [, *]

ドライブn のファイル名の頭がABで始まるファイル全部をドライブm にコピーします。ただし、保護コードのあるものはコピーされません。</td></tr>
</table>

コマンド受付条件

(1) (a)においてfile name 1=file name 2 のときはn≠m であること。
(2) (b)、(c)、(d)においてはn≠m であること。
(3) 該当するソース・ファイルがあること。
(4) ソース・ファイルと同じファイル名（保護コードを除くファイル名）のファイルがディスティネーション側にないこと。

コピーはディレクトリの最初のものから一つづつ行なうがその途中に同じファイ

ル名のものがあったならその段階でエラーとなりコマンドは終了する。
(5) コピーする前のファイルのトータルが64ファイル未満であること。
連続していくつかのファイルをコピーする場合64ファイルを超えるときはその段階でエラーとなりコマンドは終了。
(6) ディスティネーション・ファイル作成に要するページ数（データ・エリア＋ヘッダー・エリア）が確保可能なこと。もし連続していくつかのファイルをコピーする場合コピーする前に確保可能をチェックし、不能ならばその段階でエラーとなりコマンドは終了。

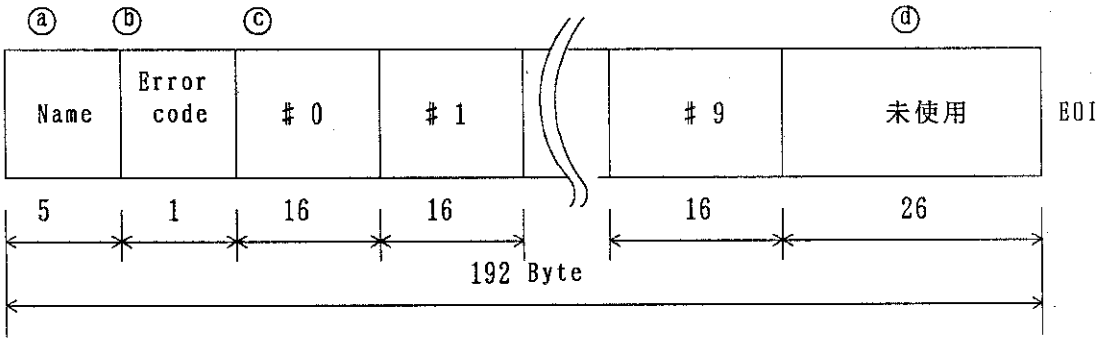
プログラム例

```
OUTPUT 701 ; "CO, 0, Prog, 1, BACKUP"
OUTPUT 701 ; "CO, 0, DATA, 1, *"
OUTPUT 701 ; "CO, 0, TEST, 0, TEST_2"
OUTPUT 701 ; "CO, 0, *, 1, *"
```

5.1.13 NOコマンド：ステータス情報の出力

書式	NO
機能	NOコマンド実行後Talker Active State に入るとTR45102 のステータス情報を出力します。また電源投入直後とIFC 受信によってもNOをコマンド実行します。

ステータス情報の出力フォーマット

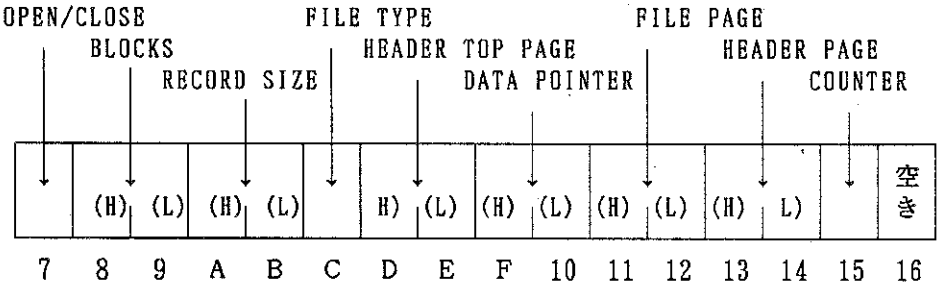


ⓐ 装置名：ASCIIで45102を出力。
ⓑ エラー・コード：各コマンド等実行中に発生するエラー。エラー・コードの一覧を〔表 3-1〕に示します。
ⓒ バッファ・ステータス

エラー・コード (16進数)	意味
00	NO ERROR
01	BUBBLE FULL
02	FILE NOT FOUND
03	HARD WRITE PROTECTED (WPRTE)
04	未使用
05	BUBBLE READ ERROR (UCE)
06	BUBBLE CASSETTE OFF (CAOF, NOMEM)
07	EJECTED BUBBLE CASSETTE (EJECT)
08	SECURITY CODE VIOLATION 保護コードが指定されているfileいるfileを保護コードなしにアクセスしようとした。
09	未使用
0A	IMPROPER FILE TYPE
0B	BUBBLE NO HEADER (NHDR)
0C	BUBBLE MANY DEFECT LOOPS (MDL)
0D	その他のBUBBLE ERROR
0E	未使用
0F	未使用
10	SYNTAX ERROR
11	NO INITIALIZE BUBBLE バブル・メモリがイニシャライズされていない。
12	NO MY LISTENER ADDRESS コマンド・モードのときリスナ・アドレスなしでキャラクタ受信のとき。
13	FILE OPEN OPEN中に受付不可能なコマンドを受信。
14	GPIB COMMAND BUFFER OVER 255文字を超えるコマンドを受信した。
15	PRESENT SAME FILE CR, CS, SAのときすでに同じ名のファイルがあるとき。
16	TOTAL FILE OVER トータル・ファイル数が64を超えるとき。
17	FILE FULL CR, CSで指定されたﾊﾞｲﾄ 数以上のﾃﾞｰﾀ の受信またはﾌﾟﾛｸﾞﾗﾑ･ﾌｧｲﾙで65535 を超える受信。
18	BLOCK ASSIGN OVER ランダム・ファイルで#nで指定のブロック番号がCRコマンドで指定したものより大。
19	FILE NOT OPEN #nコマンドで指定したバッファにファイルがアサインされてない。
1A	FILE ALREADY OPEN すでにOPENされているファイルを別なバッファにアサインしようとした。
1B	READ PROTECTED PRコマンドによるリード・プロテクトされたファイル。
1C	SOFT WRITE PROTECTED PRコマンドによるライト・プロテクトされたファイル。
1D	NOT PRESENT DATA LOコマンド実行時ﾌﾟﾛｸﾞﾗﾑ･ﾌｧｲﾙの中味が空、または#n後のトーカで出力データないとき。
1E	COMMAND NOT FOUND コマンドがみつからない。
1F	BUBBLE POWER OFF BO モードのときバブル・アクセスしようとした。
21	BUFFER ALREAY OPEN すでにOPENされているバッファに別なファイルをアサインしようとした。

ⓒ バッファ・ステータス
ステータス情報の7バイト目から166 バイト目までに#0～#9の各バッファのステータスを出力します。
CLOSE しているバッファは16バイト全て0を出力します。

#0 BUFFER STATUS



OPEN/CLOSE	:	OPEN=FF_H CLOSE=00_H
BLOCKS	:	CRコマンドで指定したblocksで1～$FFFF_H$、シリアル・アクセス・ファイルのときは1。
RECORD SIZE	:	CR, CSコマンドで指定したrecord size で1～$FFFF_H$
FILE TYPE (ASCII)	:	ランダム・アクセス・ファイル=R シリアル・アクセス・ファイル=S
HEADER TOP PAGE	:	ヘッダに使用しているページで一番最初のもの。
DATA POINTER	:	ランダム・アクセス・ファイルのときは現在のブロック番号、シリアル・アクセス・ファイルのときは現在のポインタがファイルの頭から何バイト目かを示します。 初期値=1 最終値=＋1された値
FILE PAGE	:	現在のバッファにあるデータを書き込むとき（逆に読み出すとき）のページを示します
HEADER PAGE	:	現在使用しているヘッダーのページ。
COUNTER	:	バッファ内におけるポインタで1～40_H
空き	:	00_H

以上、FILE TYPE を除き全てバイナリです。

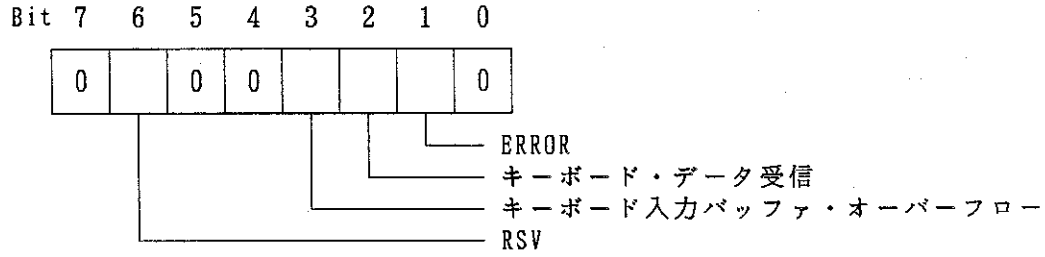
プログラム例

```
OUTPUT 701 ; "NO"
ENTER 701 USING "-K" ; Status$
```

5.1.14 S0/S1コマンド：サービス・リクエストの許可/禁止

書式	Sn n=0,1	
機能	S0 S1	SRQ 発信可 SRQ 発信不可

ステータスバイト



Bit 1=ERROR
コマンド等実行中に発生したエラーで1にセットされる。これはNOコマンドのエラー・コードに対応している。
Bit 2= キーボードデータ受信
キーボードのデータを受信すると1にセットされ（K1モードのとき）このデータが格納されているバッファが空になると0にクリアされます。
Bit 3= キーボード入力バッファ・オーバーフロー
キーボードのデータ格納バッファ（255 バイト）を超えるデータを受信すると1にセットされる。このビットが1のときは受信データは全て無視される。1にセットされているとき1バイト以上のデータを読み出すと0にクリアされ再びデータが受信可になります。
Bit 6=RSV
S0モードのときBIT1, 2, 3のORにてセット/リセット。
S1モードのときは0のまま。
SRQ はS0モード後に発生したBit1～3のいずれかが1にセットされると発信し、その後シリアル・ポーリングを実行するとクリアされる。
また、BIT1～3すべてがクリアされるとクリアされる。
POWER ONによりS1モードに設定

プログラム例

```
OUTPUT 701 ; "S0"
OUTPUT 701 ; "S1"
```

5.1.15 B0／B1コマンド：バブル・カセットのスタンバイ制御

書式	Bn n=0,1	
機能	B0	バブル・カセット・ホルダーのPOWER をOFF にする。このとき，ランダム・ファイルまたはシリアル・ファイルがOPENであったときは、すべてCLOSE する。ただし、このCLOSE はCLコマンドの実行ではないので今までにOPENして書込んでいたデータを有効とするときは先にCLコマンドを実行して下さい。このモードになるとCPUはパワーセーブ状態になります。
	B1	バブル・カセット・ホルダーのPOWERをONにします。

コマンド受付条件
(1) GPIBがREMOTEになっていること。
(2) POWER ONのときはB1モードに設定

プログラム例
```
OUTPUT 701 ; "B0"
OUTPUT 701 ; "B1"
```

5.1.16 K0／K1コマンド：キーボードの許可/禁止

書式	Kn n=0,1	
機能	K0	キーボードのデータ受信。
	K1	キーボードのデータ受信、受信データをバッファに格納。

POWER ONのときはK0モードに設定

プログラム例
```
OUTPUT 701 ; "S0"
OUTPUT 701 ; "S1"
```

5.1.17 OKコマンド：キー入力データの出力

書式	OK
機能	OK コマンド実行後Talker Active State に入るとK1コマンドにて受信したキーボードのデータをバッファから取り出してGPIBに出力します。 バッファのデータをすべて出力したら最終バイトにEOI を付加し、出力します。これでステータスバイトBIT2=0にクリアされます。 これ以降に再びバッファにデータが受信され，それを出力させるときは再びOKコマンドを実行します。キーボードのデータは受信しません。

プログラム例
```
OUTPUT 701 ; "OK"
ENTER 701 USING "-K" ; Key$
```

5.1.18 TLコマンド：TTL I/O 出力ポートの設定

書式	TLn n： 出力データ（0～255までの10進数）
機能	TTL レベル出力ポート（8ビット）にデータを出力します。出力の論理は正論理にて出力。

プログラム例
```
OUTPUT 701 ; "TL255"
OUTPUT 701 ; "TL0"
```

5.1.19 OTコマンド：TTL I/O 入力ポートの読み出し

書式	OT
機能	OTコマンド受信によりTTL レベル入力ポート（8ビット）からデータを入力し、ASCII に変換し、その後Talker Active State に入るとそのデータをGPIBに出力します。

出力フォーマット
×××CR LF
└── 000～255（10進数）3桁で出力します。もし3桁に満たないときは頭に0をつめて3桁にします。

プログラム例
```
OUTPUT 701 ;" OT"
ENTER 701 ; " Ttl"
```

5.1.20 TSコマンド：バブル・カセットの診断

書式	TS
機能	BLC とRWC をバブル・メモリに対し実行します。この直後に、Talker Active Stateに入ると診断結果をGPIBに出力します。 BLC(Boot Loop Check) : バブル・カセット駆動に必要な情報に異常がないかチェックします。 RWC(Read Write Check): 読み書きチェック。既に記憶されているデータを破壊することはありません。

バブル・メモリが2ドライブ分ある場合はBLC コマンドを先に2ドライブ行ないその後RWC コマンドを行ないます。

BLC ドライブ1→ドライブ2
↳ RWC ドライブ1→ドライブ2

ドライブ1がエラーのときはドライブ2のチェックは行なわずドライブ1に関するエラーをセットして、BLC であったなら次のRWC の実行に移ります。

出力フォーマット

BLCxxyRWCxxy(R)(F)
xx : BSR (バブルが出力するエラー・ステータス・レジスタ) そのもののエラー・コードを10進数で出力
00～15 (2桁にて出力)
y : エラーを発生したドライブ番号(0または1)

無エラーのときはxxy=000です。

5.2 プログラミング

5.2.1 各コマンドに共通するコマンド受付条件

イ．バブル・カセット・ホルダーがPOWER ONであることが必要なコマンド。
IN, CR, CS, PR, OP, #n, CL, SA, LO, DE, DI, CO, TS
ロ．OPEN中にも受付可能なコマンド
OP, #n, CL, NO, BO/B1, KO/K1, OK,, TL, OT, TS, SO/S1,

5.2.2 一行中に並べることができるコマンド

SO/S1, KO/K1, BO/B1, OK, TL, OT
これらのものはコマンドとコマンド間を","で区切ることによって一行に並べての送出ができます。

プログラム例
SO, KO, TL123, BO ㊋㊊

OK, OT ㊋㊊ …… Talker のときOTによるデータを出力
OT, OK ㊋㊊ …… Talker のときOKによるデータを出力
↑
Talker時最後に設定したコマンドによるデータを出力。

5.2.2 コマンドの送り方

デリミタは { LF / CR/LF / EOI / CR/LF/EOI } のいずれも使用できます。

コマンドを送るときはそれに先立ってまずLISTENER ADDRESSを送りその後にコマンドを送って下さい。LISTENER ADDRESSがないとNO MY LISTENER ADDRESSエラーとなります。

① UNT, UNL, LISTEN 0
LISTENER ADDRESS

② SA, 0, BUBBLE (デリミタ)

5.2.4 データの書込み方

(1) プログラム・ファイル
SAコマンドを送って、その後につづけてデータを送り、データの最終バイトにEOIを付加することでファイルの書込みは終了します。

(a) UNT, UNL, LISTEN 0

(b) SA, 0, BUBBLE (デリミタ)

(c) データ…… (EOI)

データ・モード

(c)においてデータ書込み中にバブル・メモリがオーバーフローしたとき、またはファイルがオーバ・フロー（65535 バイトを超えるとき）のときはEOI を受信しなくてもファイルの書込みは終了し、オーバ・フロー前のデータは有効になります。

(2) ランダム・アクセス・ファイルおよびシリアル・アクセス・ファイル
OPコマンドによってファイルをバッファにアサインした後、そのバッファを選んでデータのやりとりを行ないます。
バッファ選択は#nコマンドで行ない、このコマンドの実行によりデータ・モードになり、これに続けてデータを送るデータの最終バイトにはEOI を付加し、EOI の受信によって再びコマンド・モードになり、他のバッファ選択やCLコマンド等の受付が可能になります。

① UNT, UNL, LISTEN 0

② #n [, block] (デリミタ)

③ データ…… (EOI)

データ・モード

データはバッファ（64バイト）が一杯になると自動的にファイルに書かれます。
また、ランダム・ファイルとシリアル・ファイル重ね書きのときはEOI の受信によってもファイルへ書込まれます。
シリアル・ファイルではCLコマンドによってバッファにある未書込みデータがファイルに書かれ、さらにディレクトリには書込んだデータの容量を書込みます。（ランダムとシリアルの重ね書きのときは書込んだデータの容量はすでにセット済なのでCLコマンドでは書込まない。）

① UNT, UNL, LISTEN 0

② CL, n (デリミタ)

プログラム例　　#0, #1=ランダム・ファイル、#2= シリアル・ファイル	
UNT, UNL, LISTEN 0 #1, 10　デリミタ データ(1ブロック分)　EOI	#1の10ブロック目へ書込み
UNT, UNL, LISTEN 0 #1　デリミタ データ(1ブロック分)　EOI	ブロックNO. 省略のため次の#1の11ブロック目へ書込み
UNT, UNL, LISTEN 0 #0,5　デリミタ データ(1ブロック分)　EOI	#0の5ブロック目へ書込み
UNT, UNL, LISTEN 0 #1,20　デリミタ データ(2ブロック分)　EOI	#1の20, 21ブロックへ書込み
UNT, UNL, LISTEN 0 #2　デリミタ データ……　EOI	#2へ書込み
UNT, UNL, LISTEN 0 #2　デリミタ データ……　EOI	#2へ先に書込んだ以降に続けて書込まれる。

ランダム・ファイルのときrecord size （1ブロックのバイト長）で指定したバイト長のものを送るが、もし短いバイト長のときは不足分のデータは前のものが残ります。

なおEOI はランダム・ファイルの場合出も最終ブロックの最終バイトに付加します。

1ブロック目	2ブロック目	3ブロック目	EOI

5.2.5 データの読み出し方

(1) プログラム・ファイル
LOコマンドを送り、これに続けてトーカ・アドレスを送ることでデータが出力されます。

① UNT, UNL, LISTEN 0
② LO, 0, BUBBLE (デリミタ)
③ UNT, UNL, TALK 0
④ [データ…… EOI] ← データが出力されて来る。

(2) ランダム・アクセス・ファイル及びシリアル・アクセス・ファイル

データの書込みと同じく#nコマンドでバッファを選択する。これに続けてトーカ・アドレスを送ることでデータが出力されます。

① UNT, UNL, LISTEN 0
② #n [, block] (デリミタ)
③ UNT, UNL, TALK 0
④ [データ…… EOI] ← データが出力されて来る。

出力されるデータはランダムのときには各ブロック単位でそのブロックの最終バイトにEOIが付加されます。

[1ブロック目] EOI [2ブロック目] EOI [3ブロック目] EOI……

シリアルのときは最終バイトにのみEOIが付加されます。

[データ ……] EOI

5.2.6 シリアル・ファイル重ね書き

書込み済のファイルをOPENするとファイルのTOP から読み書き可能になる（ポインタがファイルのTOP にセットされる。）シリアル・ファイルでこのように一度書込み済のデータに重ねてデータを書くことをシリアル・ファイル重ね書きと称します。

5.2.7 ランダム、シリアル・ファイルの書込み・読み出しポインタ

データ書込み中にLISTEN ADDRESS、TALK ADDRESS、IFC を受信するとコマンド・モードに戻り、書込みは終了します。
さらにポインタはシリアルのときはそのままですが、ランダムのときはブロックの途中でこれらを受信するとブロックのTOP バントにポインタが戻ります。
また、ランダム、シリアル重ね書きのときはEOI を受信してないためバッファに残っているデータがバブルに書かれてないので、読み出しモードにすると、今まで受信していたバッファ内のデータは失われます（ランダムまたはシリアル重ね書きのときはバッファが一杯になったときか、EOI の受信でしかバブルへは書きこまれません）。
データ読み出し中にLISTEN ADDRESS、UNTALK、IFC を受信するとコマンドモードに戻り読み出しは終了する。
さらに、ポインタはシリアルのときは相手が最後に受信したデータの次を示します。引き続いて読み出してもデータが抜けるはありません
ランダムのときはブロックの途中にて、これらを受信するとブロックのTOP バイトにポインタが戻ります。

5.2.8 シリアル・ファイルの書込み、読み出しモードの切換

データ書込みモードから読み出しモードに切換えるときはCLコマンドを実行して一度ファイルをCLOSE するかまたは、OPコマンドを実行してバッファの残データをバブルに書込んでから切換える必要があります。逆に、UNTALKなどで読み出しを中止し、書込みモードへの切換を行うときは考慮の必要はありません。

5.2.9 バブル・エラーの発生

ファイルのOPEN中にバブル・メモリにエラーが発生するとOPEN中に書込んだデータを失うことがあります。また、エラー発生により今までOPENしていたバッファはすべてCLOSE します。

5.2.10 エラーのクリア

一度発生したエラーはNOコマンド後、またはIFC 受信後のTalker Active State によるステータス出力後にクリアされます。
また、以下のコマンドの受信によってもクリアされます。ただし、受信したコマンドによるエラーが発生したときはそのエラーがセットされます。

IN, CR, CS, PR, OP, $n, CL, SA, LO, DE, DI, CO

5.2.11 コマンドのキャンセル

SA, #nコマンドによるデータ受信（データ・モード）やLO, #n, DI, NO, OK, OT, TSコマンド後のTalker Active State によるデータの送信は次のものを受信するとキャンセルされます。

イ. LISTEN ADDRESS
ロ. IFC
ハ. LISTEN中（SA, #nのデータモード中）のTALKER ADDRESS
ニ. TALKER中におけるUNTLAK

5.2.12 ファイル・ネーム、デバイス・ネームについて

ファイル・ネーム、デバイス・ネームとしては下記のものが使用でき、最大10文字まで。ただし、ネームの最初の一文字は英大文字であることが必要で“␣”コードは文字と文字の間のもののみ可能です。

```
␣ ! # $ % & ( ) + - . /
0 1 2 3 4 5 6 7 8 9 = @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ]＼
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

5.2.13 保護コード (Security code)について

CR、CS、SAコマンドによってファイルを作成する時に保護コードを指定すると、それ以後ファイルの読み書きを行なうときに保護コードを指定しないとファイルを扱うことはできません。
保護コードはディレクトリ出力時にも出力されません。保護コードに使用できる文字はファイル・ネームと同じもので2文字を< > で囲んで下さい。

5.2.14 GPIBのアドレスの変更

アドレスはPOWER ON時とIFC のとき読み込み設定します。

5.2.15 セルフ・チェック

本器は電源が投入されますと、まずフロント・パネルの全LED を点灯させ、その後ROM のチェック・サムを行ないます。
もしこの際にエラーがあったときは次の状態になり動作不能になります。

点滅するLED	SRQ, TALK, LISTEN, REMOTE
消灯するLED	SAVE, LOAD, ERROR

次にROM のチェック・サムが正常ならばRAM のREAD/WRITEチェックを行ないます。

もし、この際にエラーがあったときは次の状態にあり動作不能になります。

点滅するLED	SAVE, LOAD ERROR
消灯するLED	SRQ, TALK, LISTEN, REMOTE

ROM 、RAM のチェックが正常ならば“POWER”を除くLED はすべて点灯となります。

6. 性能諸元

6.1　TR45102 仕様

ドライブ数　：　2

インタフェース　：　GPIB (IEEE488-1978)
TTL I/O
キーボード(TR45103) 接続用端子
KEYBOARD接続端子仕様
シリアル調歩同期式全二重
スタート／ストップ・ビット：　1ビット
キャラクタ長　：　8ビット
パリティ・チェック　：　無し
ボーレート　：　9600BPS
入出力レベル　：　TTLレベル
（ただし、出力はオープン・コレクタ）
TR45103の電源 (+5V) は本端子より供給

容量　：　64バイト/ページ
2048ページ/ドライブ
131.072Kバイト/ドライブ

アクセス時間　：　シーク時間 ：2.0～39.4ms
データの読み書き時間 ：12.5Kバイト/秒
平均データアクセス時間 ：19.0ms(読出し時)
14.6ms(書込み時)
(ただしGPIB動作による遅延時間を含まず。)

仕様温度環境範囲：　温度 0℃～50℃
相対湿度　85%以下

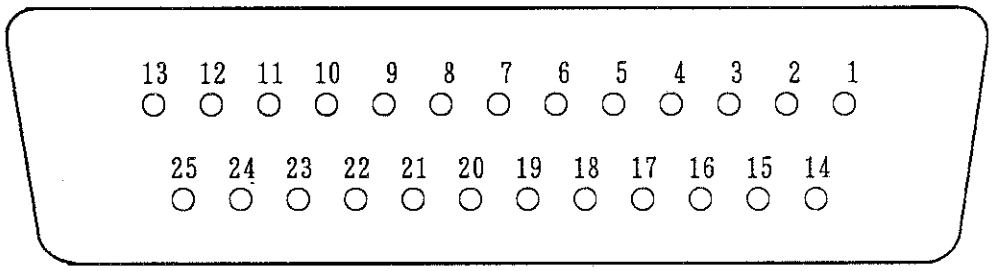
電源　：　AC100V±10%　(仕様によって120V, 220V, 240Vに変更可能)
50Hz/60Hz

消費電力　：　約50VA以下

外形寸法　：　約424(幅) × 88(高) × 550(奥行)mm

重量　：　約12kg以下

6.2 TTL I/O 仕様

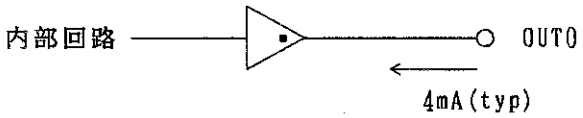


コネクタ規格:DBSP-JB25S(日本航空電子)

ピン番号	信号名称	ピン番号	信号名称	ピン番号	信号名称
1	GND	11	OUT7	21	IN6
2		12		22	IN7
3		13	GND	23	
4	OUT0	14		24	VCC/GND
5	OUT1	15	IN0	25	
6	OUT2	16	IN1		
7	OUT3	17	IN2		
8	OUT4	18	IN3		
9	OUT5	19	IN4		
10	OUT6	20	IN5		

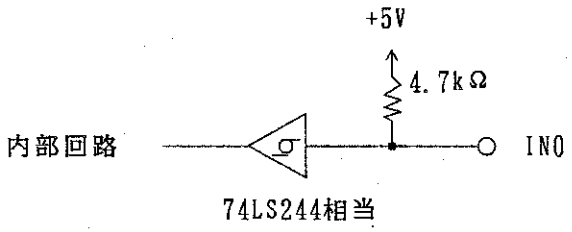
GND=0 V, VCC=+5 V

・出力ポート (OUT0～OUT7)
出力ポートはオープン・コレクタ出力で、1ポート分の等価回路を下図に示します。



出力ポートにプル・アップ抵抗を接続する場合は、数 kΩ～数10 kΩで+5 V電源にプル・アップして下さい。

・入力ポート (IN0～IN7)
入力ポートの内部等価回路を下に示します。



74LS244相当

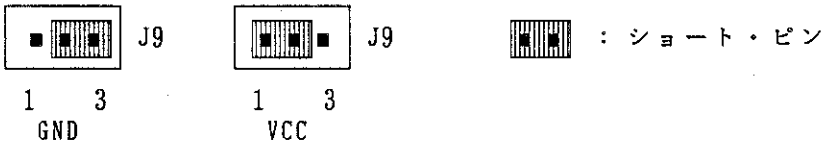
入力定格

パラメータ	MIN	TYP	MAX	単位
V_{IH} Highレベル入力電圧	2		5	V
V_{IL} Low レベル入力電圧			0.8	
入力ヒステリシス電圧		0.4		
I_{IH} Highレベル入力電流*			20	μA
I_{IL} Low レベル入力電流*			-0.2	mA

＊ 入力プルアップ抵抗は含まない。

注 意

TTL I/O の23～25ピンは VCCとGND の2 通りの設定ができますが、通常はGND になっています。このピンを VCCにするにはTR45102 内部のロジック基板(BLM-013300)のJ9のショート・ピン設定を変更しなくてはなりません。参考までにJ9の設定を下図に示します。



6.3　GPIB仕様

6.3.1　GPIBの概要

GPIBは、測定器と、コントローラおよび周辺機器などを、簡単なケーブル（バス・ライン）で接続できるインタフェース・システムです。

GPIBは、従来のインタフェース方法にくらべて拡張性に優れ、使いやすく、また他社製品とも電気的、機械的、機能的に互換性がありますので、1本のバス・ケーブルによって簡単なシステムから高い機能をもった自動計測システムまで構成できます。GPIBシステムにおいては、まずバス・ラインに接続されている個々の構成機器の各々の“アドレス”を設定しておかなければなりません。これらの各機器は、コントローラ、トーカ(TALKER;話し手)、リスナ(LISTENER;聞き手)の3種の役目のうち、1つまたは2つ以上の役目を持つことができます。

システムの動作中は、ただ1つのトーカだけがデータをバス・ラインに送出することができ、複数のリスナがそのデータを受け取ることができます。

コントローラは、トーカとリスナのアドレスを指定して、トーカからリスナにデータを転送したり、またコントローラ自身（この場合はトーカ）からリスナに測定条件などを設定したりします。

各機器間のデータ転送には、ビット・パラレル・バイト・シリアル形式の8本のデータ・ラインが使用され、非同期で両方向へ伝送が行なわれます。非同期システムのため、高速の機器と低速の機器を自由に混在させて接続することができます。

機器間で送受されるデータ（メッセージ）には、測定データや測定条件（プログラム）。

6.3.2　GPIB仕様

準　拠　規　格：IEEE規格488-1978

使　用　コ　ー　ド：ASCIIコード、ただしパック・フォーマット時はバイナリ・コード

論　理　レ　ベ　ル：論理0 “High”状態　+2.4V以上
論理1 “Low” 状態　+0.4V以上

信　号　線　の　終　端：16本のバス・ラインは、下記のようにターミネイトされています。

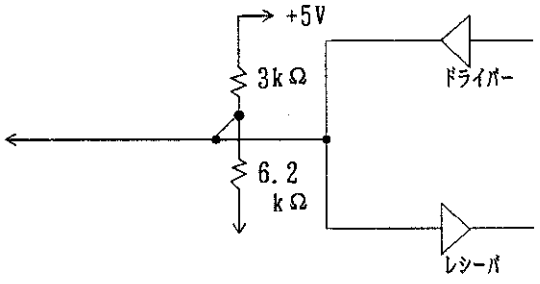


図 6-1　信号線の終端

ドライバ仕様: オープン・コレクタ形式
"Low" 状態出力電圧 ; +0.4V以下, 48mA
"High"状態出力電圧 ; +2.4V以下, -5.2mA

レシーバ仕様: + 16V 以下で"Low" 状態
+20.V 以下で"High"状態

バス・ケーブルの長さ: 全バス・ケーブルの長さは、(バスに接続される機器数)
×2m以下で、しかも20m を越えてはならない。

アドレス指定: 背面パネルのアドレス選択スイッチによって、31種類のトーク・アドレス/リスン・アドレスを任意に設定できる。
アドレス選択スイッチ切換え後は電源の再投入を行うか、IPC信号を送って下さい。

コネクタ: 24ピンGP-IB コネクタ
57-20240-D35A(アンフェノール社製品相当品)

6.3.4 構成機器との接続について

GP-IB システムは、複数の機器によって構成しますので、とくに以下の点に注意して、システム全体の準備を行なって下さい。

(1) TR45102 、コントローラ、周辺機器などの取扱説明書にしたがって、接続する前に各機器の状態 (準備) および動作を確認して下さい。

(2) 測定器との接続ケーブル、およびコントローラなどと接続するバス・ケーブルは、必要以上に長くしないように注意して下さい。全バス・ケーブルの長さは、 (バスに接続される機器数) ×2m以下で、20m を越えないようにして下さい。
なお、当社では標準バス・ケーブルとして以下のケーブルを用意しています。

表 6-1 標準バス・ケーブル (別売)

長 さ	名 称
0.5m	408JE-1P5
1m	408JE-101
2m	408JE-102
4m	408JE-105

(3) バス・ケーブルのコネクタは、ピギバック形で、1 個のコネクタに雌雄両方のコネクタがついており、積み重ねて使用できます。
バス・ケーブルを接続する場合は、3 個以上のコネクタを重ねて使用しないで下さい。
また、コネクタ止めねじで確実に固定して下さい。

(4) 各構成機器の電源条件、接地状態、また必要な場合は設定条件などを確認してから、各構成機器の電源を投入して下さい。
バスに接続されているすべての機器の電源は、必ず「ON」に設定して下さい。もし、電源を「ON」に設定していない機器がありますと、システム全体の動作は保証されません。

表 6-2 アドレス・コード表

ASCII コード キャラクタ		ADDRESS スイッチ					5 ビット 10進コード
LISTEN	TALK	A5	A4	A3	A2	A1	
SP	@	0	0	0	0	0	0
!	A	0	0	0	0	1	1
"	B	0	0	0	1	0	2
#	C	0	0	0	1	1	3
$	D	0	0	1	0	0	4
%	E	0	0	1	0	1	5
&	F	0	0	1	1	0	6
'	G	0	0	1	1	1	7
(	H	0	1	0	0	0	8
)	I	0	1	0	0	1	9
*	J	0	1	0	1	0	10
+	K	0	1	0	1	1	11
,	L	0	1	1	0	0	12
-	M	0	1	1	0	1	13
.	N	0	1	1	1	0	14
/	O	0	1	1	1	1	15
0	P	1	0	0	0	0	16
1	Q	1	0	0	0	1	17
2	R	1	0	0	1	0	18
3	S	1	0	0	1	1	19
4	T	1	0	1	0	0	20
5	U	1	0	1	0	1	21
6	V	1	0	1	1	0	22
7	W	1	0	1	1	1	23
8	X	1	1	0	0	0	24
9	Y	1	1	0	0	1	25
:	Z	1	1	0	1	0	26
;	[	1	1	0	1	1	27
<	＼	1	1	1	0	0	28
=	]	1	1	1	0	1	29
>	～	1	1	1	1	0	30

A. Appendix

A.1　磁気バルブの動作原理

ここでは富士通㈱編技術資料BP-0000-02,「富士通バブル・メモリ」より転載させて頂き、磁気バルブの動作原理を説明します。

A.1.1　磁気バブルとは

磁気バブルとは“磁気のあわ”の意味であり、磁性膜に発生する円筒状の磁区の事をこのように呼びます。

普通、強磁性体単結晶薄膜として、ガトリニウム・ガリウム・ガーネット(GGG) 基板の上に、磁性ガーネット結晶を成長させたものを使います。

この薄膜は、膜面に垂直な方向に、磁区が形成されやすいような性質（一軸磁気異方性）を持ち、磁化状態として、膜面に垂直に上向き（膜上面が Nで膜下面がS)か下向き（膜上面が Sで膜下面がN)の二つの状態しかとる事ができません。

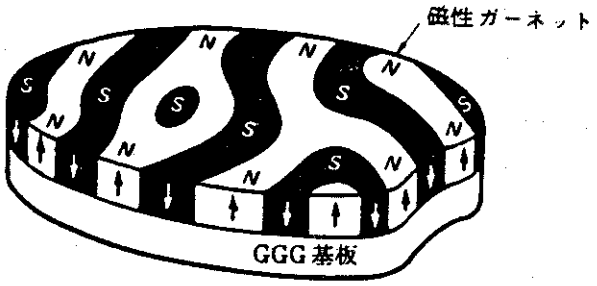
外部磁界がない場合は、薄膜内の磁気エネルギーが最小になるように上向きの磁区と下向きの磁区の面積が等しくなるような形で磁区が存在します（図 A-1a)。この状態に上向きの外部磁界（バイアス磁界）を加えてゆくと、上向きの磁区が増加し、下向きの磁区が減少し、ついには、下向きの磁区が円筒状になります（図 A-1b)。

この円筒状磁区の膜面にあらわれた円形を磁気バブルと呼び、磁区バブルメモリーは、この状態を利用しています。この状態を偏光顕微鏡で見るとこの円形はファラデ効果によって見えます。実際の磁気バブルメモリーでは、この磁気バブルを発生させ安定に保持する適当な強さのバイアス磁界を永久磁石によって与えます。

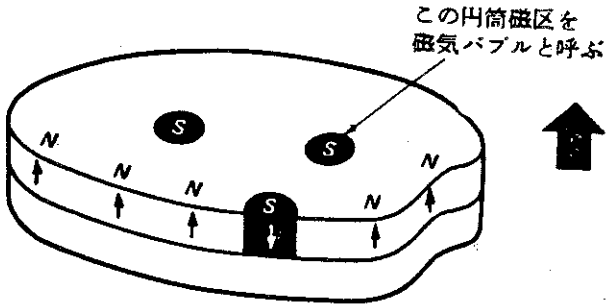
現在用いられている磁気バブルの直径は 2㎛(2㎛バブルという）で10㎜角チップ内に約100万個（つまり1 M ビット・クラス）もいれる事ができます。

さらに上向きのバイアス磁界を加えていくと、円筒状磁区は消滅して上向きの磁区ばかりとなります（図 A-1c)。

(a)　バイアス磁界がない時上向きの磁区と下向きの磁区が同じ割合で存在する。



(b)　バイアス磁界を増加していくと磁気バブルがあらわれる。



(c)　さらにバイアス磁界を増加させると磁気バブルは消滅する。

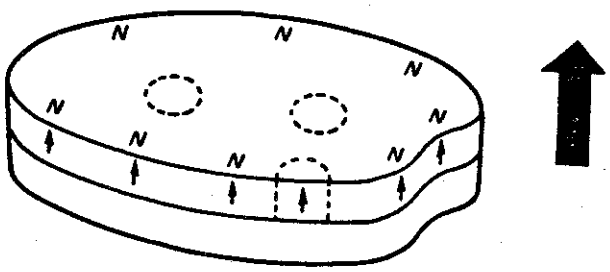


図 A-1　磁気バブルのバイアス存在性

A.1.2　磁気バブルの転送

磁気バブルをメモリとして使用するためには、チップ上に存在するバブルにアドレスを与える必要があります。この方法として、チップ上にバブルの伝播路上にバブルが固定される最小単位を作り、これを　1ビットとし、この最小単位にバブルが有る無に対応して情報の“1”、“0”を決定します。

具体的方法としては〔図 A-2a)に示すように、バブルチップ上にパーマロイ(N_i-F_e合金）で作られた半円形パターン（ハーフディスク）を形成し、これを並べることにより記憶ループを作ります。すなわち、ハーフディスク 1ヶが 1ビットであり、これを並べてループを作る事によりシフトレジスタメモリができます。

次に、シフトレジスタメモリのために、情報を伝播る必要があり、この方法としてパターン面上に水平な方向に回転磁界を与え、ハーフディスクを回転磁界と同一方向に磁化することにより、バブルを吸引し、ハーフディスクが並べられた伝播路にそって移動させます。この回転磁界は、チップを包んでいる互いに直交する鉄のソレノイド・コイルによって作られます（XYコイル）。このコイルに90°位相の異なる三角波電流を流すことにより、チップ面上に正方形の回転磁界が得られます(図 A-2b)。〔図 A-2c)に回転磁界によるバブル移動原理を示します。

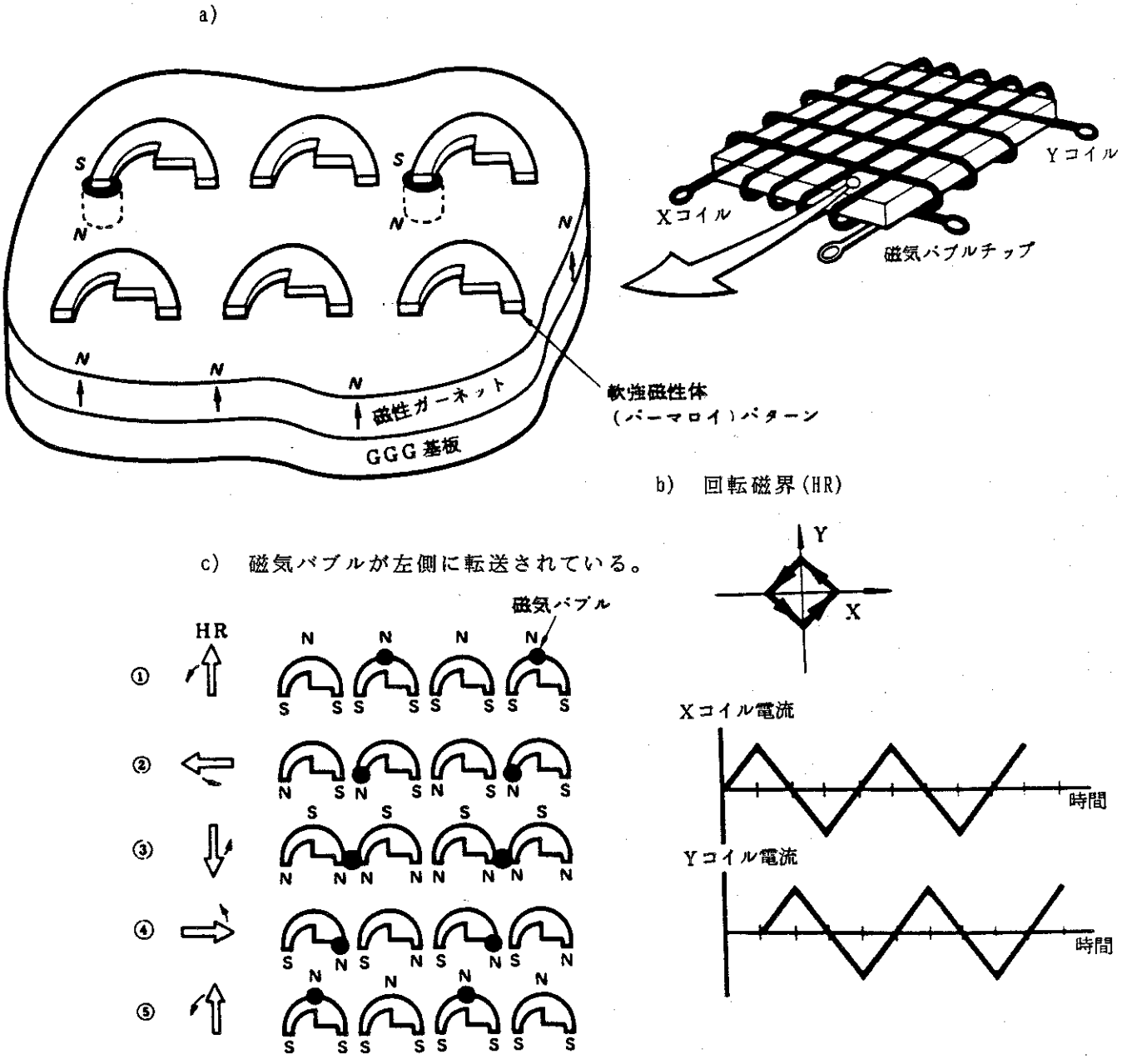


図 A-2　磁気バブルの転送

A.1.3　磁気バブルの制御

磁気バブルをメモリとして使用するには、磁気バブルを用いたシフトレジスタを構成すれば良いが、さらに書き込み、読み出し等の機能が必要となります。

磁気バブルメモリの書き込み動作

磁気バブルメモリでは、バブルの有無を“1”、“0”に対応させます。従って情報書き込みは、バブル 1ヶを発生させる（“1”を書く）、バブルを発生させない（“0”を書く）機能が必要となり、図 A-3a に示すように、バブルの伝播路にヘア・ピン型のコンダクタ・パターンを配置し、このパターンにパルス電流を流し、極部磁界を下げる（ヘア・ピンの中）事によりバブルを発生させることができます。

また、“0”を書く場合は電流を流さなければ良い。

磁気バブルメモリの読み出し動作

磁気バブルの検出には、バブルの有無を電気信号に変換しなければなりません。伝播路に使用しているパーマロイは、磁気抵抗効果（磁界により抵抗値が変化する現象）を持っているため、伝播路とのパーマロイの抵抗を見ればバブルの検出可能です。

しかし、ハーフ・ディスク 1ヶにバブル 1ヶが近ずいた時の値として抵抗値が低すぎるたり、実際には〔図 A-3b 〕に示すように伝播路をパラレルとし、このパラレルの伝播上でバブルを細長く引き延し、伝播路全体に広がるように（ストレッチャー・パターン）とて、検出に適度な抵抗値及び出力電圧が得られるよう、検出器を用います。

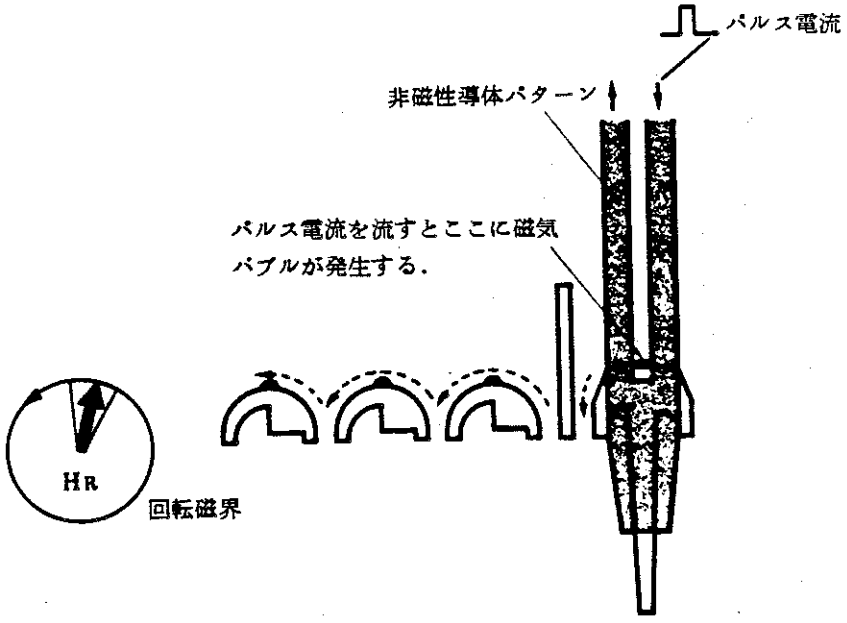


図 A-3a　磁気バブルの書き込み

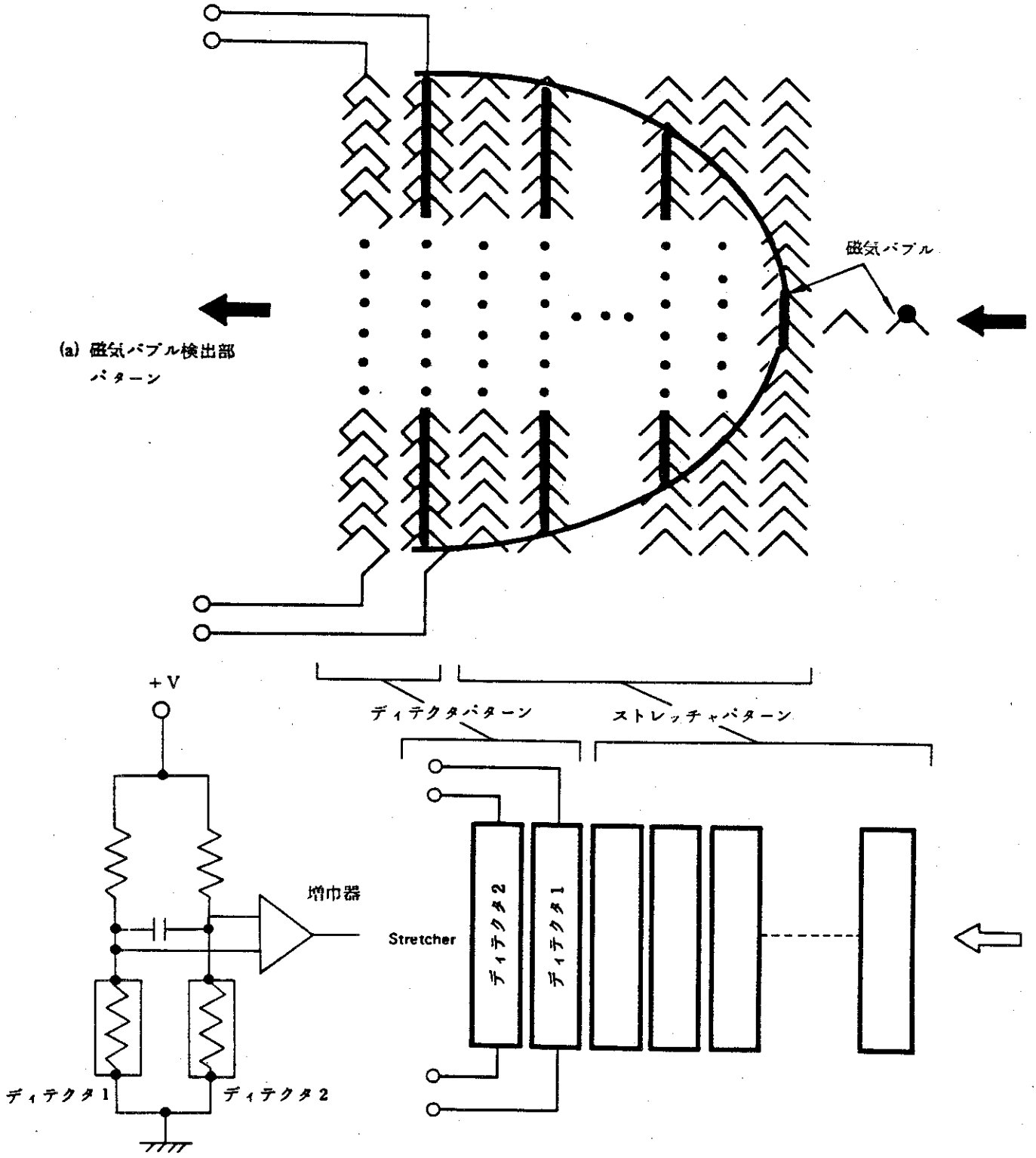


図 A-3b　磁気バブルの読み出し

磁気バブルの非破壊読み出し

磁気バブルの読み出しには、記憶されているバブル情報を記憶ループから検出器迄引き出さなければなりません。このためには、記憶ループ内のバブルをコピーして検出迄に導く方式を採用しています。〔図 A-3c 〕にバブルのコピー回路（レプリケート・ゲート）を示します。

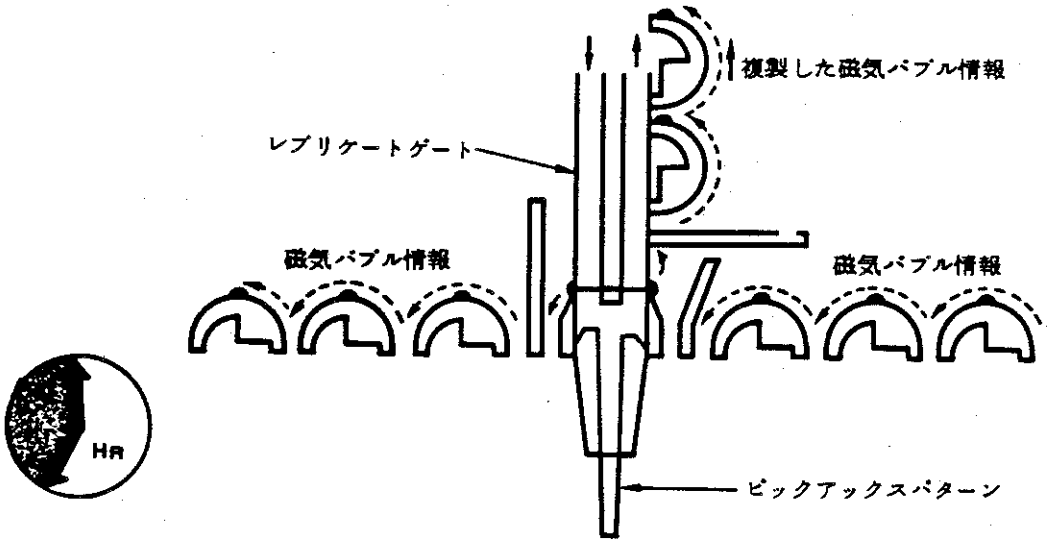


図 A-3c　レプリケート・ゲート

〔図 A-3c 〕において、右側から近づいて来た磁気バブルがピックアップの上部で引き延ばされ、回転磁界が斜線で示された向きにある時、矢印の方向のパルス電流を流すと、磁気バブルは 2等分され、一方は上の方に転送され、もう一方は左へ転送されます。このような働きをする部分もレプリケータと呼ぶ。これより記憶ループ内にあるデータを非破壊で読み出すことができる。

A.1.4　バブルのスワップ

マイナー・ループ内へのバブルの書き込みを行なうゲートであり、マイナー・ループへバブルを書き込むと同時にマイナー・ループ内のバブルをメジャー・ループに出す役目をします〔図 A-4〕において、まず、マイナー・ループ内のバブルがスワップゲート上のハーフ・ディスクに伝播してきた時、約位相210°にて電流による反発磁界により、メジャー・ループ側伝播路へと飛び移る。また、メジャー・ループのバブルは、位相が約90°～ 180°方向の間にメジャー・ループからマイナー・ループに飛び移り、バブルの交換が行なわれます。

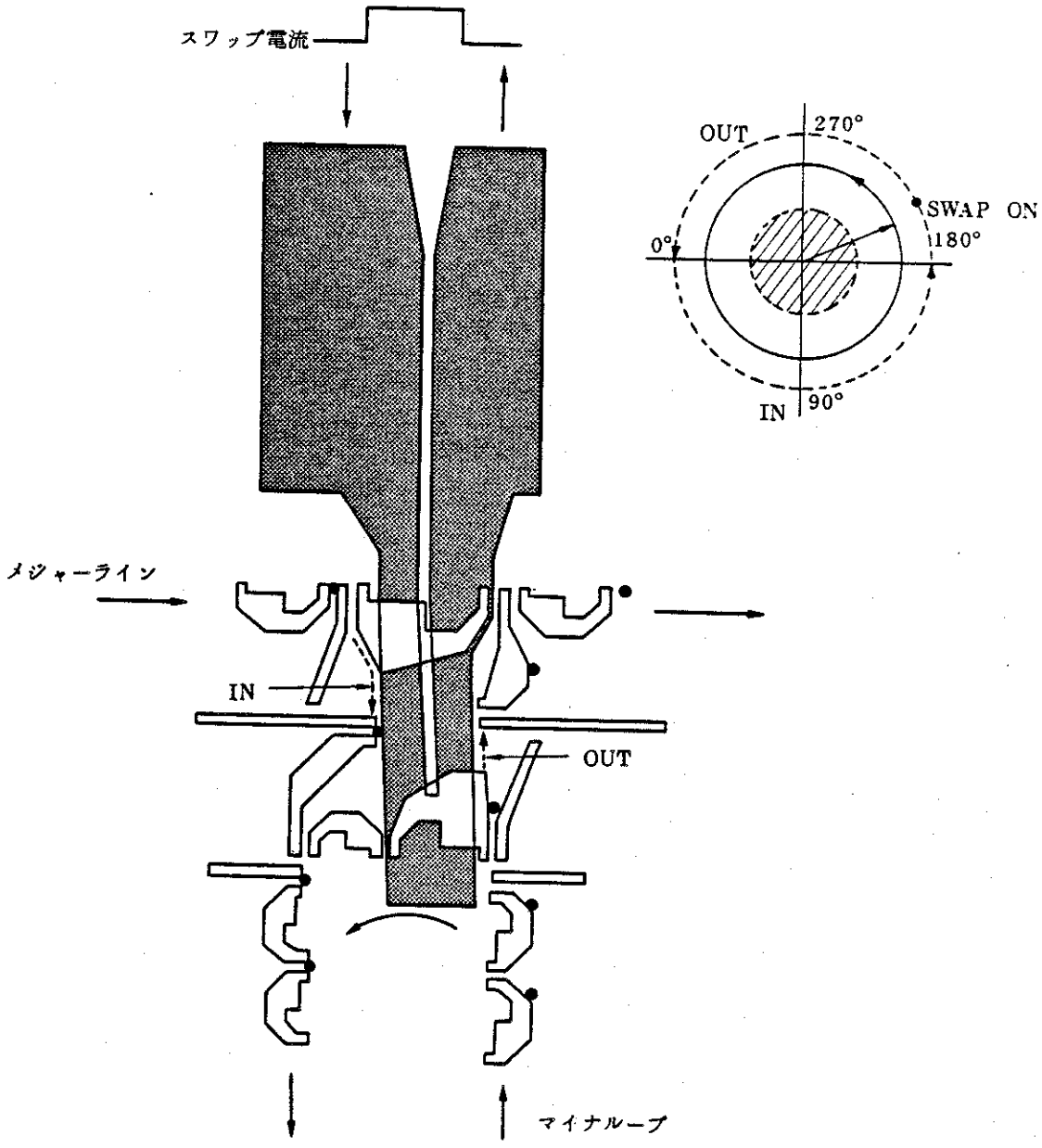


図 A-4　スワップ・ゲート

A.1.5　1Mビット・メモリ・デバイス

デバイスの構造

バブルメモリデバイスは図に示すように、

a)　ガーネットを材料とした磁気バブルチップ
b)　磁気バブルチップ内のバブルを保持するために、チップに垂直方向に磁界（バイアス磁界HB）を印加するバイアスマグネット
c)　チップ内のバブルを転送するため、回転磁界を発生するXYコイル
d)　チップ内のバブルの転送を制御したり、出力信号を外部に引き出す端子等から構成されています。

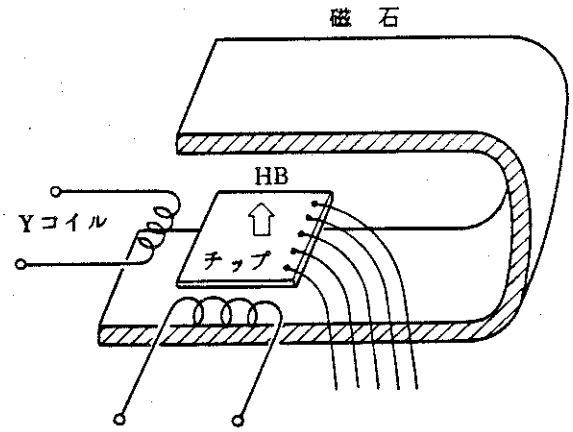


図 A-5a バブル・メモリ・デバイス構造

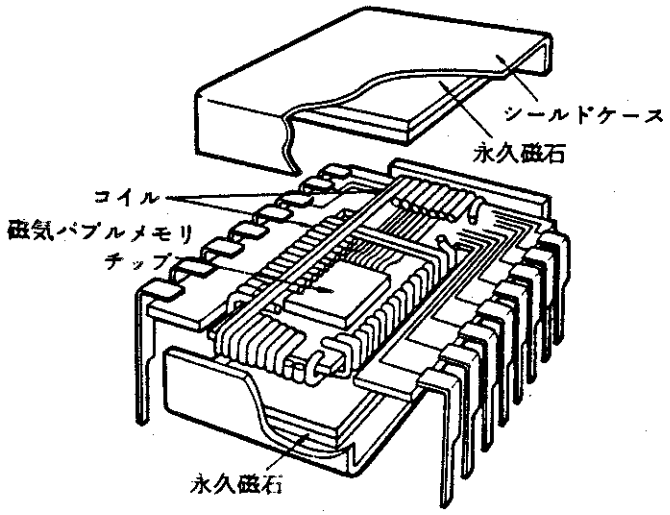


図 A-5b　バブル・メモリ・デバイス構成図

A.1.6　バブル・メモリ・チップの構成

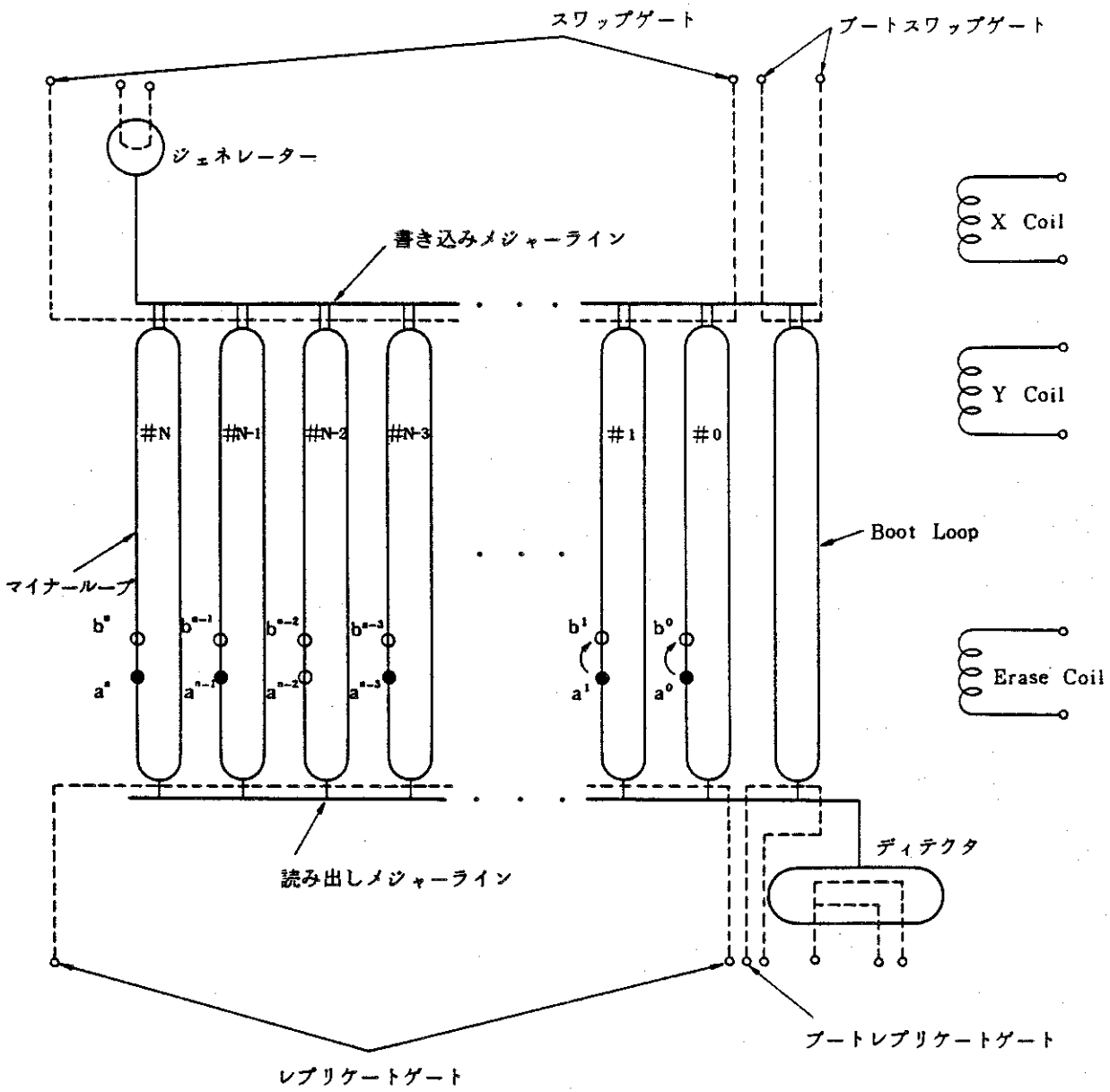


図 A-6　バブル・メモリ・チップ構成図

磁気バブルメモリチップの等価回路を上図に示す、チップは、 2つのマスクレベルにて構成され、まずパーマロイ（N_iとF_eの合金)より作られる。バブルを転送、記憶するパターンとして

a)　情報記憶用マイナー・ループであり、ループ数584本、また、1ループ中には2053ビットの情報が記憶可能。
b)　読み出し又は書き込み時に使用するメジャー・ライン
c)　バブルを電気的信号に変換するため、パーマロイの磁気抵抗効果を利用した検出器があり、次にアルミ・パターンにより作られ、バブル転送を制御するゲートとしてa)バブルの書き込みを行なうジェネレータ・ゲート
b)　書き込みメジャー・ラインからマイナー・ループにデータを入れる。（マイナー・ループ内のデータと入れ換える）スワップゲート
c)　マイナー・ループ内のデータをコピーして、読み出しメジャー・ラインにデータを取り出すレプリ・ゲートにより構成される。
　またチップ内には製造と欠陥が生じ、マイナーループは全部使用できないため、使用可能なマイナーループのデータを記憶する専用のプート・ループがある。

チップ内のデータの取り扱い

　バブル・メモリ内のデータは、図b に示すようにマイナーループ内の横一列（図中a^0～a^n ）のデータを 1つのページとして取り扱います。従って1Mでは、512ビット/ページ×2048ページをデータとして扱います。尚、回転磁界を 1回転させると、各ページのデータは同時にとなりのページに移り（例　a^0からb^0へ）ページ単位にて転送されます。

A.2　ファイル管理について

A.2.1　バブル・カセット内のメモリ構成

バブル・カセット内部では64バイト(512ビット) を 1ページとして 1区切の単位で扱っています。TR45102 ではこの 1ページを基本単位としてディレクトリを構成しファイル管理を行っています。
バブル・カセット内のメモリ構成を下図に示します。

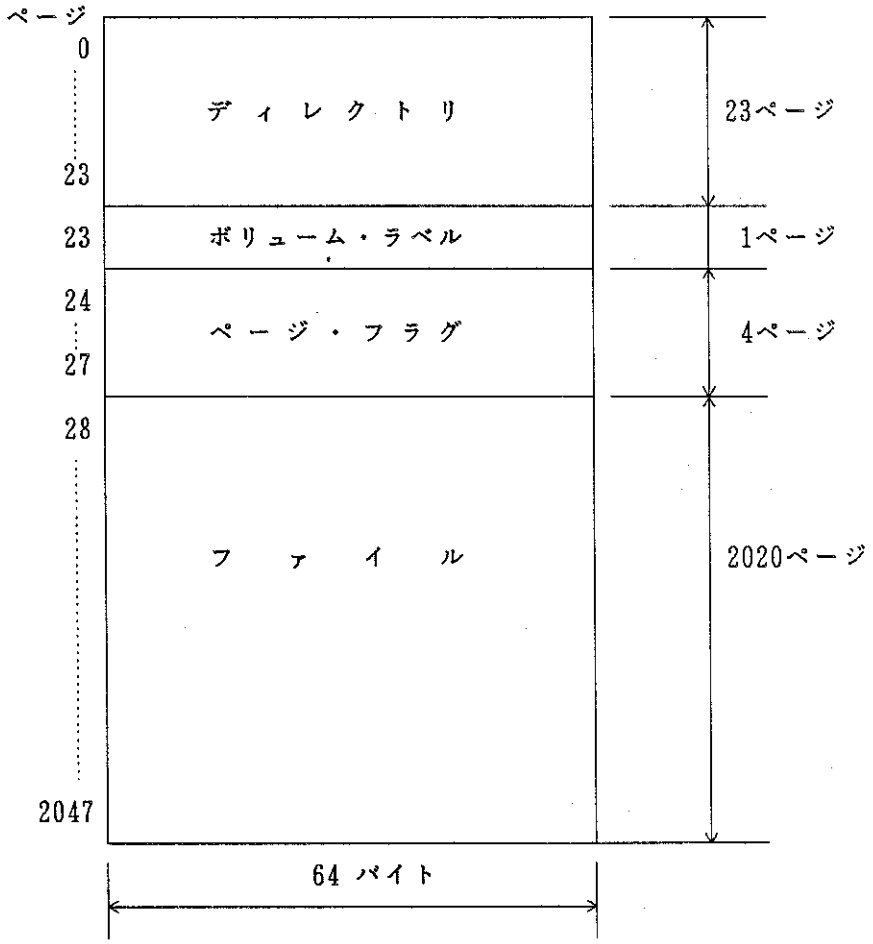
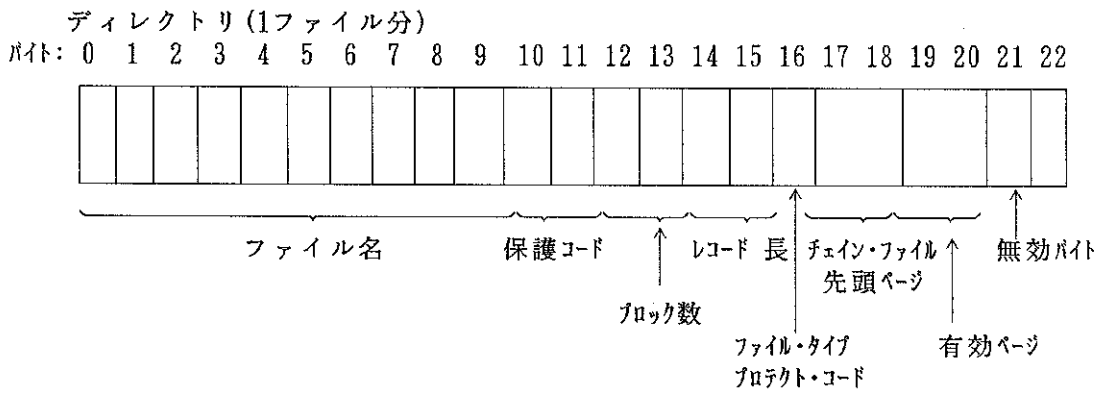


図　A-7 バブル・カセット内のメモリ構成

ディレクトリには記憶されているファイルの名称、保護コード、ブロック数、レコード長、ファイル型式、プロテクト・コード等が書込まれます。バブルカセットをアクセスする時は、常にディレクトリを参照して各ファイルを検索します。ディレトクリは 1ファイルにつき22バイトで構成され、0 ～22ページの中で最大64ファイルまで管理することができます。 1ファイル分のディレクトリを次の図に示します。



ファイル名　　：ファイル作成時にファイル個々に与えられる名称で10文字以下の文字列として扱われます。

保護コード　　：ファイル作成時に、ファイル名とともに指定されたものです。

ブロック数　　：ランダム・アクセス・ファイル作成時に指定されたもの、あるいはプログラム・ファイル・、シリアル・アクセス・ファイルでは 1が記憶されます。

レコード　　：ランダム・アクセス・ファイル作成時に指定されたブロックのレコード長あるいは、プログラム・ファイル、シリアル・ファイルではファイル全体の長（ファイル長）が記憶されます。

ファイル・タイプ　：プログラム・ファイル、ランダム・アクセス・ファイル、シリアル・アクセス・ファイルのいずれかの識別コードが記録されます。

プロテクト・コード：PROTECT ステートメントで指定された、プロテクト・コードが記録されます。

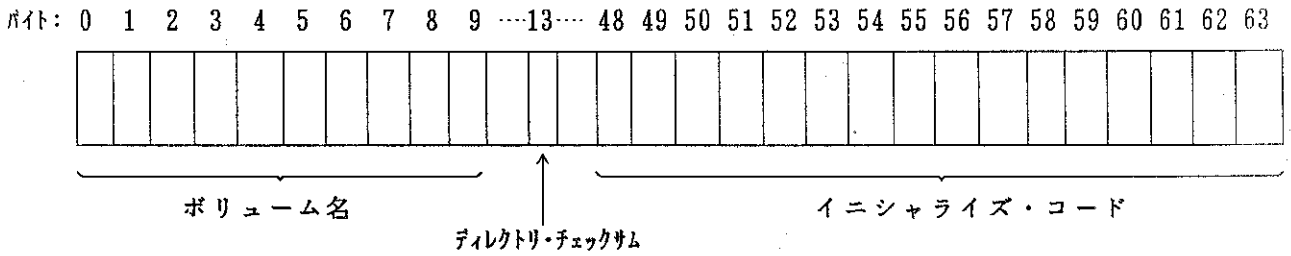
チェイン・ファイル先頭ページ：
チェイン・ファイル（後述）の先頭ページを示します。

有効ページ　　：ファイル作成に必要としたページの総数が記録されます。

無効バイト　　：ファイルの最終ページでファイルに使用されなかったバイト数を示します。(0～63)

ボリューム・ラベルにはINITIALIZEステートメントで与えられたボリューム名とバブル・カセットがイニシャライズされていることを示すコード（16バイト）が記録されます。

ボリューム・ラベル（23ページ）



ボリューム名：バブル・カセットのボリューム名で10文字以内の文字列として記録されます。

ディレクトリ・チェック・サム：0 ～23ページ全体のチェックサム・コードが記録されます。

イニシャライズ・コード：バブル・カセットがイニシャライズされているか否かを示す識別コードが記録されます。

24～27ページには、28～2047の各ページがファイルとして使用中であるか未使用であるかを示すフラグが記録されます。28～2047ページの各ビットは28～2047の各ページに一対一に対応し、ページが使用中であれば“1”未使用であれば“0”が対応するビットに記録されます。例えば下図の場合、28,29,35,37 ページは使用中で、30,31,32,33,34,36～41ページは未使用であることを示します。

バイト：　　(24ページ)

1 1 0 0 0 0 0 1	0 1 0 0 0 0 0 0	·····

ビット：0 1 2 3 4 5 6 7　0 1 2 3 4 5 6 7

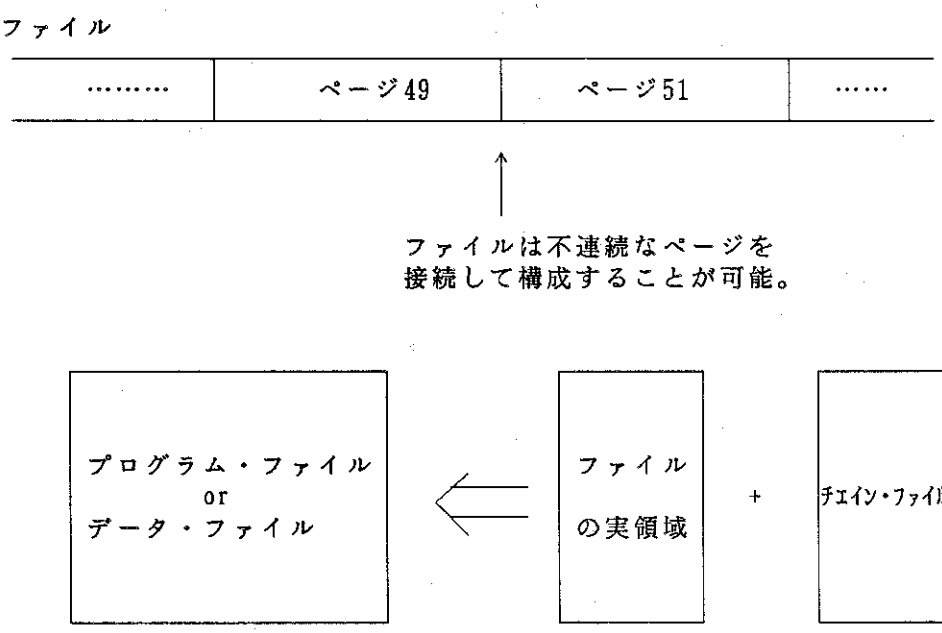
またDELETEステートメントでファイルを抹消した場合にも、抹消されたファイルに対応する各ページは未使用ファイルとして扱われ、24～27ページ内の対応するビットが“0”に書換えられます。

28～2047ページにはファイルが構成され、プログラムやデータが記録されます。

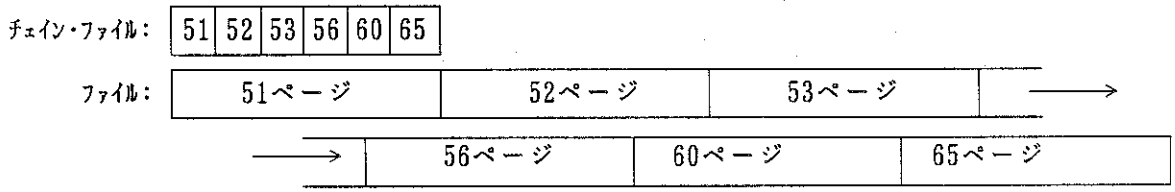
A.2.2　ファイルの管理

TR45102 では、バブル・カセット内のメモリ空間を効率よく活用するため、ファイル作成用のページは 1ページ単位で独立に扱い、必要なページを各々接続して 1つのファイルを構成します。ページを接続する際には必ずしも連続したページである必要はなく、いかなる（不連続）であっても未使用ページであれば接続してしまいます。例えば今50ページが使用中でその前後のページ(49, 51)が未使用であったとして、新しくファイルを作りますと49ページと51ページを接続して、見かけ上連続した 1つのファイルとして構成します。

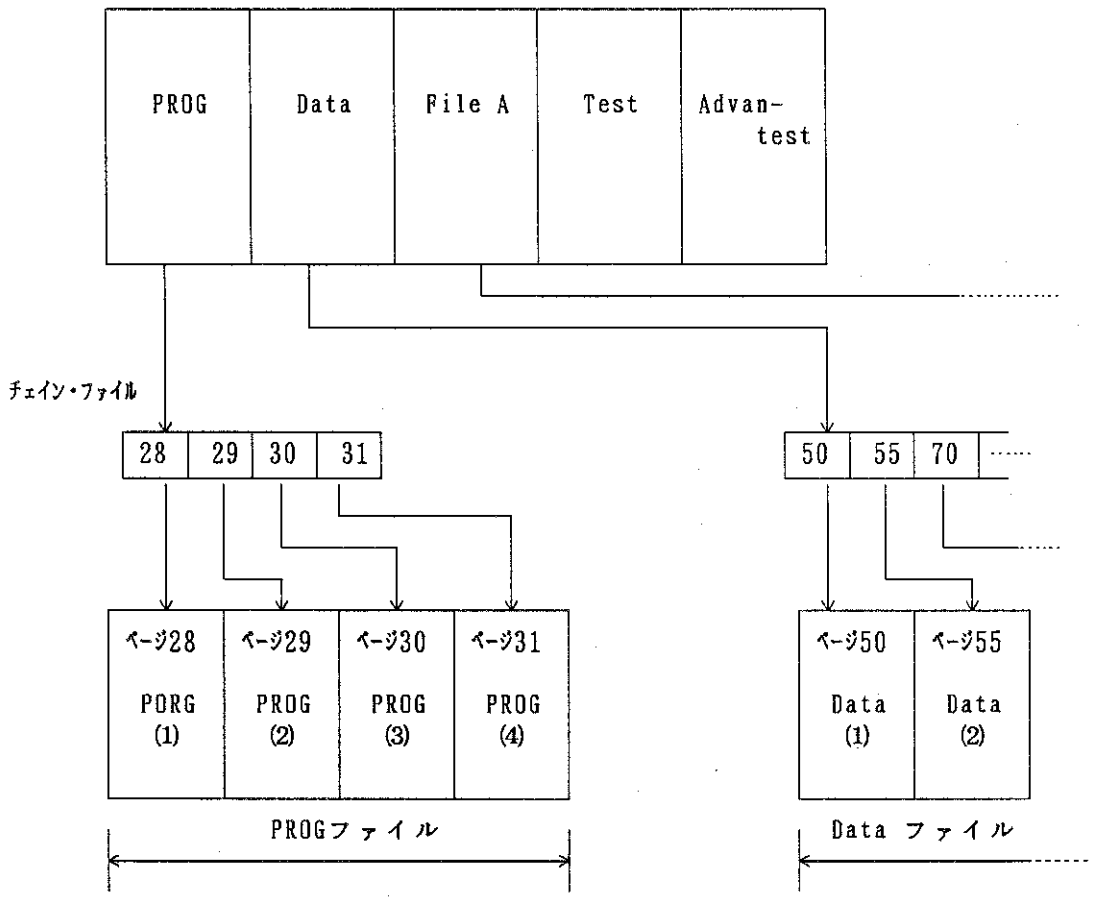
不連続なページを接続するためには、ページの接続情報をどこかで憶えておく必要があります。このためTR45102 ではファイル作成時にページの接続情報を記憶するチェイン・ファイルを同時に作成します。チェイン・ファイルはプログラム・ファイルあるいはデータ・ファイル 1つに対し必ず 1つ存在し、見かけ上はチェイン・ファイルを含めて 1つのファイルとして認識されます。（チェイン・ファイル単独で存在したり、チェイン・ファイルを独立にアクセスすることはできません。）



チェイン・ファイルの中には使用したページ番号を使用した順（ファイルの先頭から終橋に向って）に記憶されます。



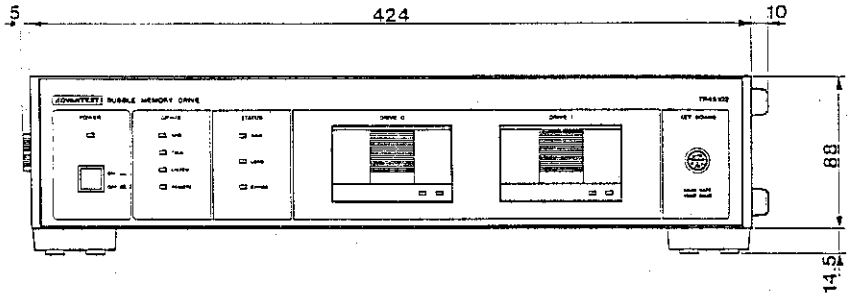
このようにTR45102 ではチェイン・ファイルを用いているため、ファイル作成とファイル抹消を繰返しても、バブル・カセット内に無駄領域が作られることはありません。（一般的にフロッピー・ディスク等のファイル管理方法ですと、連続したセクタでファイルを構成しますので、既存のファイルを抹消した場合にはファイル作成時にアクセスできない無駄な領域が発生します。）



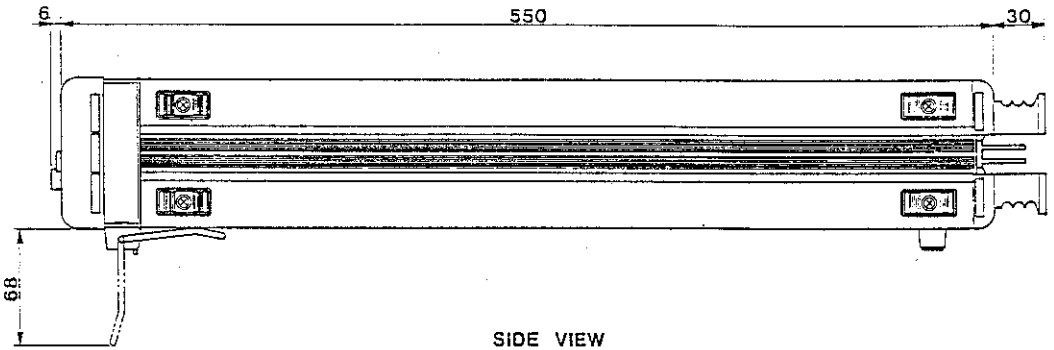
ファイルの構成図

図一覧

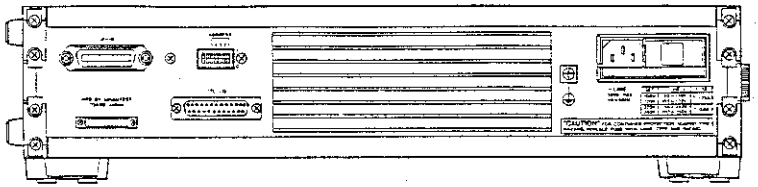
表一覧



FRONT VIEW



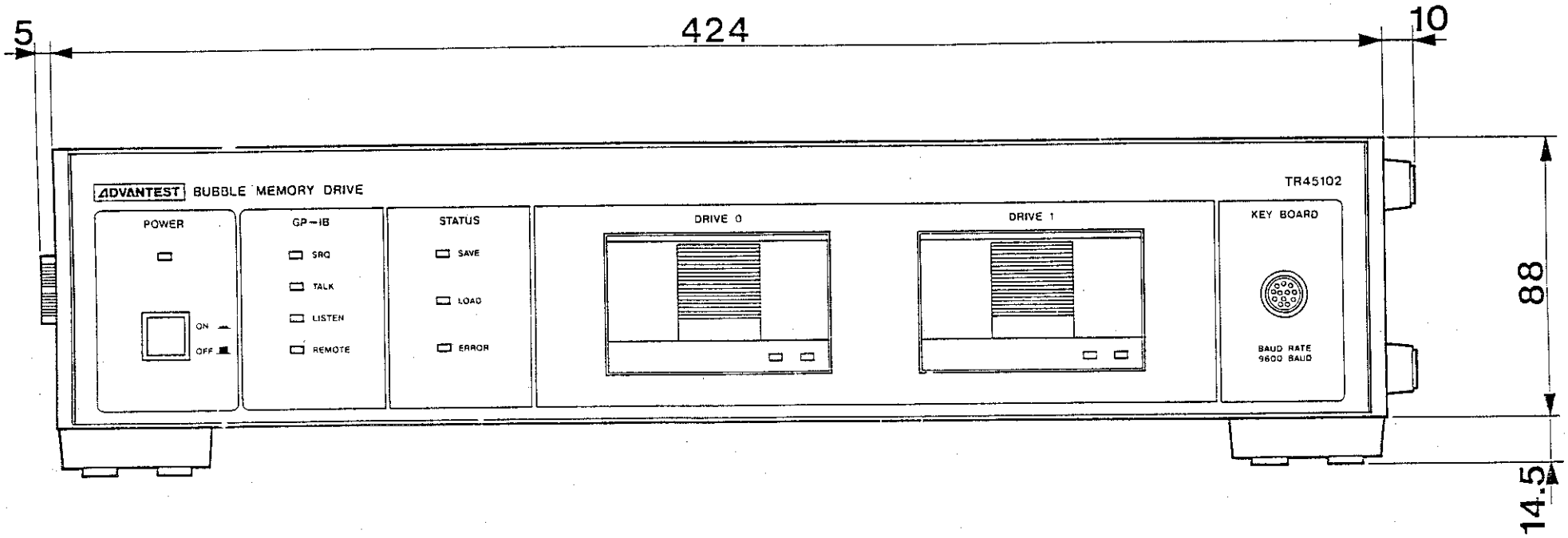
SIDE VIEW



REAR VIEW

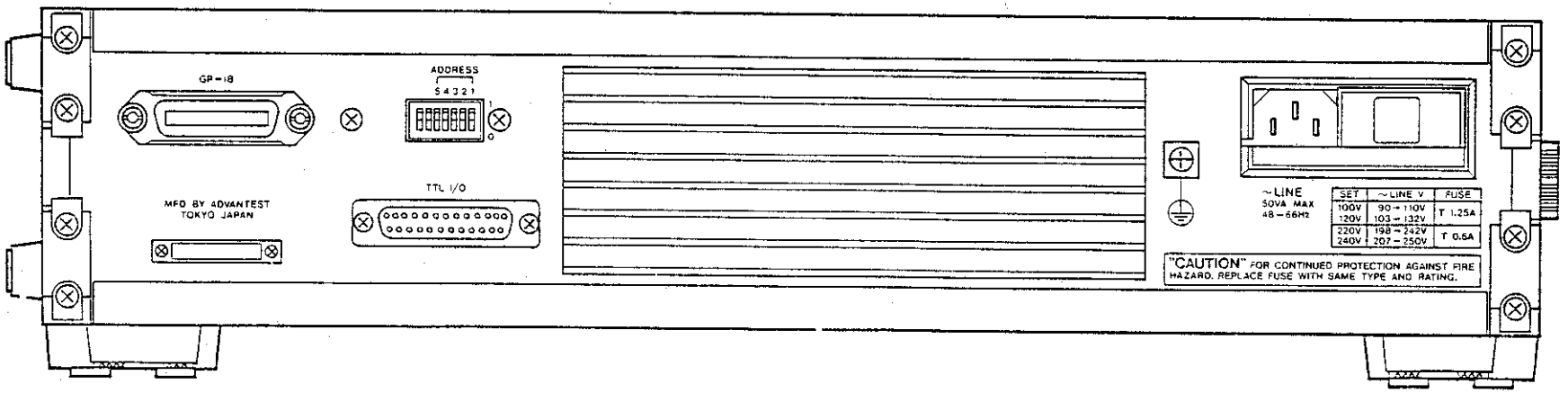
TR45102
EXTERNAL VIEW

4S102EXT1-506-A



FRONT VIEW

45102EXT2-606 - A



REAR VIEW

45102EXT3-606-A

本製品に含まれるソフトウェアのご使用について

本製品に含まれるソフトウェア（以下本ソフトウェア）のご使用について以下のことにご注意下さい。
ここでいうソフトウェアには、本製品に含まれる又は共に使用されるコンピュータ・プログラム、将来弊社よりお客様に提供されることのある追加、変更、修正プログラムおよびアップデート版のコンピュータ・プログラム、ならびに本製品に関する取扱説明書等の付随資料を含みます。

使用許諾

本ソフトウェアの著作権を含む一切の権利は弊社に帰属いたします。
弊社は、本ソフトウェアを本製品上または本製品とともに使用する限りにおいて、お客様に使用を許諾するものといたします。

禁止事項

お客様は、本ソフトウェアのご使用に際し以下の事項は行わないで下さい。

- 本製品使用目的以外で使用すること
- 許可なく複製、修正、改変を行うこと
- リバース・エンジニアリング、逆コンパイル、逆アセンブルなどを行うこと

免　　責

お客様が、本製品を通常の用法以外の用法で使用したことにより本製品に不具合が発生した場合、およびお客様と第三者との間で著作権等に関する紛争が発生した場合、弊社は一切の責任を負いかねますのでご了承下さい。

保証について

製品の保証期間は、お客様と別段の取り決めがある場合または当社が特に指定した場合を除き、製品の納入日（システム機器については検収日）から1年間といたします。保証期間中に、当社の責めに帰する製造上の欠陥により製品が故障した場合、無償で修理いたします。ただし、下記に該当する場合は、保証期間中であっても保証の対象から除外させていただきます。

- 当社が認めていない改造または修理を行った場合
- 支給品等当社指定品以外の部品を使用した場合
- 取扱説明書に記載する使用条件を超えて製品を使用した場合（定められた許容範囲を超える物理的ストレスまたは電流電圧がかかった場合など）
- 通常想定される使用環境以外で製品を使用した場合（腐食性の強いガス、塵埃の多い環境等による電気回路の腐食、部品の劣化が早められた場合など）
- 取扱説明書または各種製品マニュアルの指示事項に従わずに使用された場合
- 不注意または不当な取扱により不具合が生じた場合
- お客様のご指示に起因する場合
- 消耗品や消耗材料に基づく場合
- 火災、天変地異等の不可抗力による場合
- 日本国外に持出された場合
- 製品を使用できなかったことによる損失および逸失利益

当社の製品の保証は、本取扱説明書に記載する内容に限られるものとします。

保守に関するお問い合わせについて

長期間にわたる信頼性の保証、国家標準とのトレーサビリティを実現するためにアドバンテストでは、工場から出荷された製品の保守に対し、カスタマ・エンジニアを配置しています。
カスタマ・エンジニアは、故障などの不慮の事故は元より、製品の長期間にわたる性能の保証活動にフィールド・エンジニアとしても活動しています。
万一、動作不良などの故障が発生した場合には、当社のMS（計測器）コールセンタにご連絡下さい。

製品修理サービス

- **製品修理期間**
  製品の修理サービス期間は、製品の納入後10年間とさせていただきます。
- **製品修理活動**
  当社の製品に故障が発生した場合、当社に送っていただく引取り修理、または当社技術員が現地に出張しての出張修理にて対応いたします。

製品校正サービス

- **校正サービス**
  ご使用中の製品に対し、品質および信頼性の維持を図ることを目的に行うもので、校正後の製品には校正ラベルを貼付けし、品質を保証いたします。
- **校正サービス活動**
  校正サービス活動は、株式会社アドバンテスト カスタマサポートに送っていただく引取り校正、または当社技術員が現地に出張しての出張校正にて対応いたします。

予防保守のおすすめ

製品にはエレクトロニクス部品およびメカニカル部品の一部に寿命を考慮すべき部品を使用しているため、定期的な交換を必要とします。適正な交換期間を過ぎて使用し発生した障害に対しては、修理および性能の保証ができない場合があります。
アドバンテストでは、このようなトラブルを未然に防ぐため、予防保守が有効な手段と考え、予防保守作業を実施する体制を整えています。
各種の予防保守を定期的に実施することで、製品の安定稼働を図り、不意の費用発生を防ぐため、年間保守契約による予防保守の実施をお薦めいたします。
なお、年間保守契約は、製品、使用状況および使用環境により内容が変わりますので、最寄りの弊社営業支店にお問い合わせ下さい。

ADVANTEST

http://www.advantest.co.jp

株式会社アドバンテスト

本社事務所
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 03-3214-7500（代）

第4アカウント販売部（東日本）
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 0120-988-971
FAX: 0120-988-973

第4アカウント販売部（西日本）
〒564-0062 吹田市垂水町3-34-1
TEL: 0120-638-557
FAX: 0120-638-568

★計測器に関するお問い合わせ先
（製品の仕様，取扱い，修理・校正等計測器関連全般）
MS（計測器）コールセンタ
TEL 0120-919-570
FAX 0120-057-508
E-mail : icc@acs.advantest.co.jp